FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Mobile Applikationen in der Industrieautomation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsingenieurwesen Informatik

eingereicht von

Jörg Rohringer, BSc
Matrikelnummer 0325451

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner
Mitwirkung:

Wien, 7. Mai 2017

Jörg Rohringer	Wolfgang Kastner

Erklärung zur Verfassung der Arbeit

Jörg Rohringer, BSc
Beethovengasse 12, 3130 Herzogenburg

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 7. Mai 2017

Jörg Rohringer

Kurzfassung

In der Industrieautomation werden Engineering-Tools eingesetzt, um Anlagenteile zu planen und in Betrieb zu nehmen. Zusätzlich können sie auch für Wartungs und Instandhaltungsaufgaben verwendet werden und bieten vereinzelt auch Diagnosemöglichkeiten. Diese Tools werden meist auf dem Betriebssystem Windows verwendet, das sich auf ortsgebundenen Computern oder Laptops befindet. Zusätzlich werden Daten, die zwischen den Tools und den Geräten in der Anlage ausgetauscht werden, kabelgebunden übertragen. Dies kann ein Problem darstellen, wenn Fehler behoben werden müssen und die Rechner weit entfernt von den fehlerhaften Komponenten sind oder keine Möglichkeit der Verwendung von Laptops vor Ort besteht. Smartphones und Tablets weisen diesen Nachteil der Ortsgebundenheit von Desktops und teilweise auch Laptops nicht auf. Sie verfügen ebenso über graphische Benutzeroberflächen und sind mittlerweile mit leistungsstarken Komponenten ausgestattet, um auch komplexe Programme ausführen zu können. Außerdem bieten die Geräte verschiedenste Möglichkeiten für die kabellose Übertragung von Daten an.

Ziel dieser Arbeit ist es, die Möglichkeiten und Vorteile von mobilen Applikationen in der Industrieautomation zu untersuchen. Dabei sollen im ersten Schritt Anwendungsfälle in der Industrieautomation erarbeitet werden. Anhand dieser Anwendungsfälle sollen dann Anforderungen für die verwendeten Geräte und Kommunikationsprotokolle erarbeitet werden.

Weiters werden Übertragungsprotokolle beschrieben, die von Smartphones und Tablets aktuell verwendet werden. Diese werden auf ihre Eignung für die definierten Anwendungsfälle und die daraus abgeleiteten Anforderungen analysiert.

Auch sollen einerseits „Best Practices“ für die Softwareentwicklung behandelt und andererseits die aktuell meist verwendeten Betriebssysteme von mobilen Geräten besprochen werden.

Im Folgenden wird eine prototypische Applikation beschrieben, mit der verschiedene Aufgaben eines Engineering Tools erledigt werden können.

Zuletzt wird eine wirtschaftliche Analyse von mobilen Applikationen vorgestellt, die auch die Ergebnisse einer Befragung von Personen aus verschiedenen Bereichen der Automation enthält.

Abstract

In industrial automation, engineering tools are used to plan and install components for automation facilities. These tools can also be used for maintainance and offer diagnosis functionality to locate defects. Most of them are Windows-based programs that are used on desktop-computers and laptops. For exchange of data between automation devices and these computers, wired connections are used. In case of errors, this can be a problem, if the distance between the faulty device and the maintenance-computer is large, or there is no way to use a laptop at the location of the device.

Mobile devices like smartphones or tablets have advantages in these cases. They work mainly wireless over diverse communication protocols and are usually smaller. Applications on mobile devices can also be used with rich graphics and use performant components to be able to execute complex programs on them.

In the context of this thesis, the changes and advantages from mobile applications for the use in industrial automation shall be examined. At the beginning, use cases shall be defined for the possible functions that can be executed by applications on mobile devices. From these use cases, requirements are derived.

In the following chapter, several wireless communication protocols are explained, that are currently used in mobile devices, and analyzed, if they are compatible with the requirements of the given use cases.

The next chapter gives an overview of „best practices“ related to software engineering methods. In addition, prominent operating systems of mobile devices are discussed.

Furthermore, a prototype application, that was assembled to show typical functions that are offered by engineering tools, is shown.

The last chapter analyzes the economic aspect of mobile applications, especially in industrial automation setups. Finally, findings of interviews conducted with a number of people working in the field of automation are presented that illustrate trends of mobile applications in the light of Industrie 4.0.

Inhaltsverzeichnis

KAPITEL 1

Einleitung

In der letzten Zeit werden immer wieder neue Schlagworte eingeführt, um die Weiterentwicklung und Verwendung von Technologien in der Industrie zu benennen. Die wohl bekanntesten sind hier „Digitalisierung“, „Digitale Transformation“ und „Industrie 4.0“, oder auch „Smart Factory“ und „Smart Devices“. Diese Worte beschreiben vielfach die verstärkte Nutzung von IT-Technologien, bzw. die Vernetzung von verschiedensten Geräten (Internet-of-Things), etwa über die „Cloud“[1], um die Produktivität und Effizienz zu erhöhen, oder den Komfort der Menschen zu steigern. Oft wird auch von „Big Data“ gesprochen. Hier werden Daten gesammelt, mit deren Hilfe gewisse Voraussagen getroffen oder Wahrscheinlichkeiten für das Eintreten bestimmter Ereignisse berechnet werden können. Diese Entwicklung ist besonders für die Industrieautomation interessant, weil sie es erleichtert, die Steuerung der Vorgänge für die gesamte Wertschöpfungskette weiter zu verbessern und zu automatisieren. Dies führt beispielsweise zu einer besseren Planbarkeit für Wartungen und Produktion und ermöglicht es, Kosten zu reduzieren und Ausfällen vorzubeugen. In [MO16] wurde erhoben, wie die verschiedensten „unscharfen Termini“, also Begriffe die hier verwendet werden, in der Industrie allgemein und für den Bereich des Engineerings im Speziellen verstanden werden, und welche Handlungsempfehlungen sich daraus anleiten lassen. Durchgeführte Befragungen ergaben, dass ein hoher Prozentsatz (84%) der Teilnehmer bereits das mögliche Potenzial von „Industrie 4.0“ für ihr Unternehmen analysiert hat. Weiters werden in 67% von befragten Unternehmen bereits Projekte mit „Industrie 4.0“ Kontext durchgeführt. Auch zum Thema Engineering wurden Erhebungen durchgeführt. Dabei teilten 70% die Meinung, dass Engineering „essentiell“ für Industrie 4.0 sei.

Typische Automationssysteme in der Fertigungs- und Verfahrenstechnik sind schichtenweise aufgebaut und folgen der Automationspyramide. Auf der Feldebene werden Daten gesammelt und die Schnittstelle zum technischen Prozess über Ein- und Ausgangssignale

[1] *Cloud* bezeichnet ein Netzwerk von verteilten Rechnern über die Services bezogen werden können.

hergestellt. Daten werden an die übergeordnete Automationsebene weitergereicht, wo gesteuert und geregelt wird. In der Leitebene kommen SCADA-Systeme (**S**upervisory **C**ontrol and **D**ata **A**cquisition) zum Einsatz, mit deren Hilfe die Überwachung und Bedienung des technischen Prozesses vorgenommen wird. Hauptaufgaben der Betriebsleitebene sind u.a. die Produktionsfeinplanung, Produktionsdatenerfassung und das Qualitätsmanagement. Die Schnittstelle zu wirtschaftlichen Prozessen wird letztlich durch die Unternehmensebene hergestellt. Der Datenaustausch erfolgt einerseits innerhalb der Ebenen (horizontale Kommunikation) aber auch ebenenübergreifend (vertikale Kommunikation). Grob gesagt, kann zwischen Prozessdaten und Engineeringdaten unterschieden werden. Letztere erlauben Wartungs- und Parametriervorgänge vorzunehmen. Diese Informationsverarbeitung übernehmen in vielen Fällen so genannte Engineering Tools.

Engineering Tools, meist Windows-basiert, verfügen über graphische Benutzerschnittstellen. Mit ihrer Hilfe können Anlagenteile und ihre Komponenten geplant und in Betrieb genommen werden. Vereinzelt bieten sie auch Diagnosemöglichkeiten. Diese Tools sind in der Regel auf ortsgebundenen Computern oder Laptops installiert, die kabelgebunden Daten von bzw. an die benötigten Stellen senden bzw. empfangen. Hierbei müssen die Daten über die Ebenen weitergereicht werden. Ein Trend in Richtung Industrial Ethernet ist zu verzeichnen, der der starken Heterogenität von Übertragungstechniken und -protokollen entgegen wirken soll. Die Notwendigkeit einer Kabelverbindung stellt allerdings einen Nachteil für die schnelle Behebung von Problemen auf der Feld- oder Automationsebene dar. Für viele Wartungsaufgaben an Sensoren und Aktuatoren muss unter Umständen für die Durchführung eine entfernte Workstation aufgesucht werden, was entweder zeitliche oder personelle Kosten verursachen kann. Der Aufbau von Automatisierungssystemen in Fertigungs- und Verfahrenstechnik wird in [SKD11] beschrieben.

Mobile Applikationen, die beispielsweise auf Smartphones/Tablets installiert sind, weisen in der Frage der Ortsgebundenheit diesen Nachteil nicht auf. Sie besitzen, genau wie die Workstations, auf denen das Engineering Tool ausgeführt wird, graphische Oberflächen, können aber direkt am Standort der Wartung verwendet werden. Parametrierungen einzelner Komponenten können direkt vor Ort durchgeführt, notwendige Firmware-Updates aktualisiert und deren Funktionsfähigkeit überprüft werden, ohne immer zwischen den Komponenten und der Engineering-Workstation pendeln zu müssen. Auch eine Überprüfung von Ist- und Sollwerten, um etwaige Fehler schnell zu erkennen, kann angedacht werden. Diagnose und Monitoring sind weitere wichtige Punkte. Gleichzeitig wäre es für Mitarbeiterinnen und Mitarbeiter möglich, mithilfe schematischer Pläne der Anlage auf dem Smartphone schneller fehlerhafte Komponenten zu finden, zu warten, oder auszutauschen. Für wenig genutzte Verfahrensweisen im Umgang mit Komponenten der Anlagen könnten auch Kurzanleitungen mit Checklisten in der mobilen Applikation anzeigt werden. Weiters kann man mittels Augmented Reality Informationen schneller und gezielter erhalten [NAS04]. Gleichzeitig können aufgetretene Probleme erkannt werden, auch wenn der Wartungscomputer nicht besetzt ist, bzw. können Wartungsnachrichten (ggf. auch Alarme) direkt an das Smartphone weitergereicht werden.

Natürlich bringt der Einsatz von mobilen Applikationen gewisse Herausforderungen mit

sich. Einerseits besitzen Smartphones im Vergleich zu herkömmlichen PCs begrenzte Ressourcen, was Displaygröße, Speicher und Prozessorgeschwindigkeit betrifft. Andererseits sind auch die Datenübertragungsmöglichkeiten (meist WLAN und Bluetooth) begrenzt, was eine Erweiterung auf andere Funknetz-Standards betrifft. Bezüglich der Displaygröße als vorher genannten Nachteil wäre es möglich, diese Applikationen auch auf Tablets zu verwenden, die in der letzten Zeit immer populärer werden. Gleichzeitig müssen bei einer funk-basierten Anbindung an Automationssysteme auch Kommunikationsfehler berücksichtigt werden, die bei kabelgebundenen Anbindungen seltener zu finden sind (z.B. Ausfall). Auch auf Sicherheitsaspekte muss verstärkt geachtet werden, da ein Kompromittieren einer kabellosen Übertragung einfacher geschehen kann.

Um diese Ziele zu erreichen, müssen, wie im vorigen Abschnitt angedeutet, aber auch Fehlermodelle entwickelt werden, die eine reibungslose funkbasierte Kommunikation ermöglichen. Dabei müssen sowohl Fehler behandelt werden, die aufgrund von Störungen der Datenverbindung auftreten können, als auch Fehler, welche von Dritten mutwillig herbeigeführt werden. Dabei geht es auch um Möglichkeiten der Verschlüsselung der Datenverbindung, um Eindringlinge davon abzuhalten, Daten zu entwenden oder ungewünschte Daten einzuschleusen (sichere Datenkänale).

Es gibt im Bereich der Wartung von Industrieanlagen schon vergleichbare Arbeiten, wie [ZPU11], [LHLW08] und [AS10]. Hier werden mobile Geräte beschrieben, mit deren Hilfe Wartungsarbeiten effizienter durchgeführt werden können. Diese Arbeiten beziehen sich eher auf Applikationen, die nur Informationen über Wartungsprozeduren und gewisse Komponenteninformationen liefern. Kommunikation zwischen den mobilen Geräten und den Aktuatoren bzw. Sensoren findet allerdings nicht statt.

In [MD06] und [TASF11] werden Handhabungs- und Bediengeräte beschrieben, über die Aktuatoren gesteuert werden können. Diese befinden sich jedoch in physischem Kontakt mit dem zu steuernden Gerät, sei es direkt am Gerät oder weiter entfernt, zum Beispiel über Ethernet- oder andere Feldbussysteme.

Im ersten Schritt sollen in dieser Arbeit „Use Cases“ erstellt werden. Diese sollen mittels einer Modellierungssprache (UML) dargestellt werden, um einen besseren Überblick zu erhalten. Durch eine anschließende Analyse sollen Anforderungen für Softwaresysteme und Hardwareanforderungen beispielsweise für Übertragungsmechanismen (z.B. WLAN oder Bluetooth) abgeleitet werden.

Anschließend sollen in mobilen Geräten verwendete Kommunikationsprotokolle beschrieben werden. Die beschriebenen Protokolle umfassen WLAN, Bluetooth, NFC und ZigBee. Bei ZigBee existiert im Moment nur ein Gerät auf dem Markt ([TAZ]), das diesen Standard unterstützt, jedoch bietet dieses Protokoll einige Eigenschaften, die für die Verwendung in der Automation interessant sind. Die Protokolle werden in einer nachfolgenden Analyse auf ihre Tauglichkeit für die zu erfüllenden Aufgaben analysiert.

Das folgende Kapitel umfasst grundlegende Informationen zur Softwareentwicklung, wobei hier auch die momentan verwendeten Entwicklungsprozesse beleuchtet werden. Zusätzlich werden die am häufigsten verwendeten Betriebssysteme behandelt. Außerdem muss für die

folgende Entwicklung einer mobilen Applikation ein geeigneter Prozess für die Erstellung solcher Applikationen gefunden werden ([MP10]).

Schließlich soll in dieser Arbeit als „Proof-of-Concept“ eine prototypische mobile Applikation für Android-basierte Smartphones entwickelt werden, die eine ausgewählte Gruppe der zuvor genannten möglichen Aufgaben übernehmen kann. Mithilfe dieser Applikation soll es möglich sein, ein vorhandenes Netzwerk von Automationsgeräten zu scannen, Statusinformationen abzurufen und bei einzelnen Geräten die Firmware zu aktualisieren.

Weiters wurde ein Befragung von Personen aus dem Bereich der Automation durchgeführt, ob und für welche Bereiche der Industrieautomation bereits mobile Anwendungen eingesetzt werden. Aufgrund dieser soll die wirtschaftliche Relevanz für mobile Applikationen in diesem Bereich dargestellt werden.

Ein Ausblick auf vergleichbare Arbeiten (z.B. Web-basierte Engineering-Tools) und mögliche künftige Entwicklungen soll abschließend gegeben werden.

KAPITEL 2

Use Cases und Anwendungsbereiche von mobilen Applikationen in der Industrieautomation

In diesem Kapitel sollen Anwendungsbeispiele (Use Cases) beschrieben werden. Diese sollen Anwendungsmöglichkeiten von Smartphone-basierten Applikation aufzeigen. Gleichzeitig sollen mit diesen Use Cases auch Voraussetzungen solcher Apps extrahiert werden, die für eine erfolgreiche Verwendung in der Industrieautomation notwendig sind:

1. Es wird ein Scan des Netzwerkes mit anschließendem Firmwareupdate eines Geräts über einen Steuerrechner zentral (über Kabel angeschlossen) ausgeführt. Währenddessen tritt ein Stromausfall ein oder die Verbindung bricht zusammen. Daraufhin muss ein Techniker vor Ort das Gerät reparieren, wofür er auf seinem Smartphone eine Applikation installiert hat, mit der er eine neue Firmware auf das Gerät spielen kann.

2. In einer Industrieanlage sind die verwendeten Geräte als Insel ausgeführt (d.h., sie sind nicht zentral über Kabel mit einem Steuerrechner verbunden). Jede Wartung (z.B. Firmwareupdate) muss direkt am Gerät durchgeführt werden. Erschwerend kommt hinzu, dass in der Halle Laptops aus Sicherheitsgründen nicht verwendet werden dürfen, oder einfach kein Platz zur Verfügung steht, an dem man größere Geräte abstellen könnte. Eine mobile Applikation auf einem Smartphone kann hier die Arbeit vereinfachen.

3. Es tritt ein Fehler bei einem Gerät auf, das sich in einem Schaltschrank mit anderen gleichartigen oder ähnlich aussehenden Geräten befindet. Um einen Bedienfehler des Personals zu minimieren, etwa einen Zahlensturz bei der Beschriftung, wodurch ein anderes, funktionierendes, Gerät serviciert werden würde, ist der Techniker mit einem Smartphone ausgestattet. Eine installierte Applikation auf dem Smartphone kann das fehlerhafte Gerät eindeutig durch eine blinkende LED identifizieren, unter der Annahme, dass diese Funktion nicht auch durch den Fehler beeinträchtigt ist. Mittels umgekehrten Ausschlussverfahrens wäre es auch möglich, falls eine Identifikation des Gerätes nicht mehr möglich ist, über die Identifikation der funktionierenden Geräte das schadhafte Gerät zu ermitteln.

4. Um die Sicherheit in einem Werk zu erhöhen, sind die Zugangstüren mit Zutrittskontrollsystemen ausgestattet, bei denen RFID-Technologie zum Einsatz kommt. Da viele Personen berechtigt sind, sich innerhalb der Anlage zu bewegen, müssen allen Berechtigten Zutrittskarten oder RFID-Tags zur Verfügung gestellt werden, um ihre Arbeit erledigen zu können. Eine Applikation auf einem NFC-fähigen Smartphone kann stattdessen diese Aufgabe erfüllen.

5. Bei standardmäßig durchgeführten Rundgängen in einer automatisierten Anlage werden die Fehlerspeicher der eingesetzten Geräte ausgelesen. Eine Smartphone-basierte Applikation kann die benötigten Informationen schnell und unkompliziert darstellen.

6. Eine Neuinstallation/Wartung von Geräten soll in einer Anlage durchgeführt werden. Dazu müssen eine Reihe von Standardprozeduren abgearbeitet werden, um Sicherheitsnormen zu erfüllen, sowie eine ordnungsgemäße Funktionsweise der Anlage nach der Geräteinstallation/Wartung zu gewährleisten. Um auch unerfahrenen Mitarbeiterinnen und Mitarbeitern eine Installation zu ermöglichen, können Checklisten und Prozeduranleitungen in elektronischer Form auf dem Smartphone hilfreich sein.

7. Im Zuge von Einführungen oder Schulungen werden neue Mitarbeiterinnen und und Mitarbeiter durch die Anlage geführt. Es werden hier Sensoren verwendet, die mit Hilfe drahtloser Kommunikation Daten übertragen. Diese Daten können zur Prozessdarstellung und Kontrolle verwendet werden. Um den neuen Mitarbeiterinnen und Mitarbeitern gewisse Vorgänge in der automatisierten Anlage zu verdeutlichen, kann der Schulungsleiter mit einem Smartphone oder Tablet und einer Visualisierungsapplikation auf diese Sensordaten zugreifen.

8. Während des Betriebs einer Anlage werden die Soll- und Ist-Werte der einzelnen Prozesse überwacht. Dies geschieht über passende Visualisierungen. Während der Rundgänge des Betriebspersonals können die visualisierten Prozesse einfach über Smartphone- oder Tablet-Applikationen ausgelesen und angepasst werden. Auf dem Leitstand muss kein zusätzliches Personal zur Überwachung eingesetzt werden. Dies ist bei größeren Anlagen über eine Client/Server-Architektur, bei kleineren Anlagen über eine direkte Kommunikation mit den betreffenden Geräten möglich.

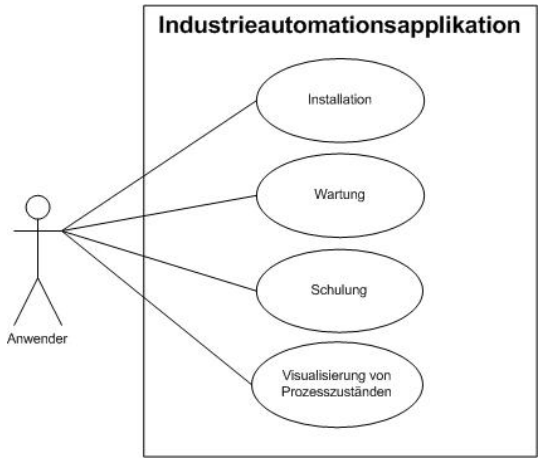

Abbildung 2.1: Use Cases

2.1 Einteilung der Use Cases

Um diese Use Cases anschaulicher zu präsentieren und zu kategorisieren, wurden sie mit Hilfe von UML (**U**nified **M**odelling **L**anguage) modelliert. Abbildung 2.1 zeigt eine grobe Einteilung der vorgestellten Anwendungsfälle in 4 Hauptanwendungsfälle, die in den Abbildungen 2.2, 2.3, 2.4 und 2.5 genauer dargestellt werden.

- Wartung/Diagnose: Der Begriff Wartung umfasst dabei Hilfsmittel, sowohl für den Fehlerfall von Geräten als auch um Geräte standardmäßig kontrollieren zu können, falls dies erforderlich ist.
- Installation/Engineering: Die Installation soll alle Methoden beinhalten, die für ein fehlerloses Funktionieren eines neu zu integrierenden Gerätes notwendig sind.
- Schulung: Die Schulung soll helfen, neuen, unerfahrenen Mitarbeiterinnen und Mitarbeitern mit anschaulichen Methoden ein gutes Verständnis für die Vorgänge innerhalb einer Anlage zu geben. Dadurch können die durchzuführenden Arbeiten schnell und effizient begreiflich gemacht werden.
- Visualisierung: Bei der Visualisierung von Prozesszuständen geht es nicht nur um eine reine Kontrolle der Prozesse innerhalb einer Anlage über den Leitstand. Vielmehr besteht auch eine Möglichkeit der Änderung oder Anpassung von Prozesszuständen außerhalb des Leitstandes.

Diese Gliederung kann nur als abstrakte Darstellung aller möglichen Anwendungsfälle betrachtet werden. Natürlich gibt es zwischen den einzelnen Punkten auch Überschneidungen. Das heißt, Wartung, Installation, Schulung und Prozessvisualisierung verwenden teilweise die gleichen Unteranwendungsfälle.

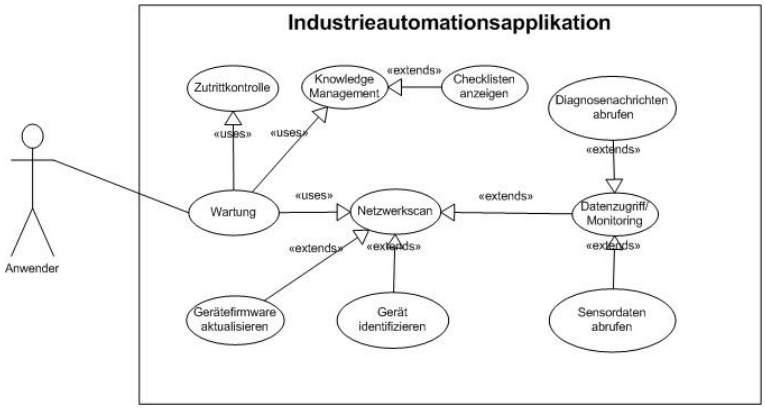

Abbildung 2.2: Use Case Wartung

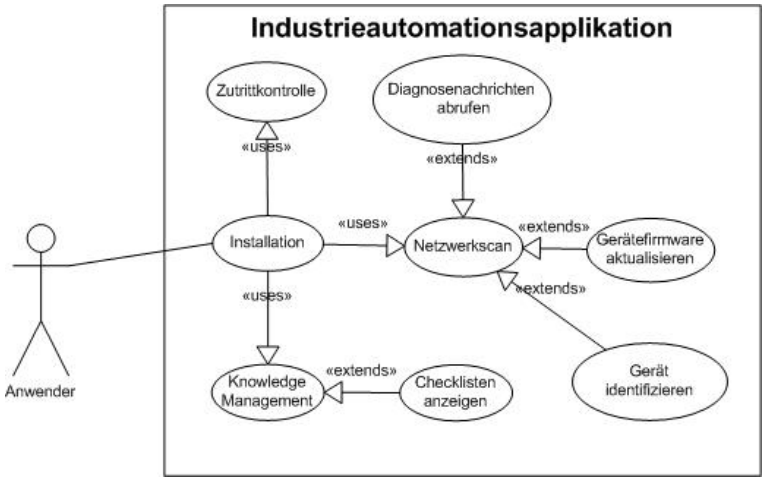

Abbildung 2.3: Use Case Installation

2.2 Unteranwendungsfälle

2.2.1 Zutrittskontrolle

Zutrittskontrollen in Anlagen erfüllen mehrere Aufgaben für ein Unternehmen. Einerseits steckt in der Technologie oft ein beträchtliches Know-How, das vor Anlagen-fremden Personen geschützt werden muss. Andererseits besteht bei automatisierten Anlagen ein erhöhtes Gefahrenpotential für, mit den Sicherheitsprozeduren nicht vertrauten Personen. Da moderne Zutrittskontrollsysteme oft mit RFID-Technik arbeiten, ergibt sich die Möglichkeit diese Anwendung in ein NFC-(**N**ear **F**ield **C**ommunication)-fähiges Smartphone zu integrieren.

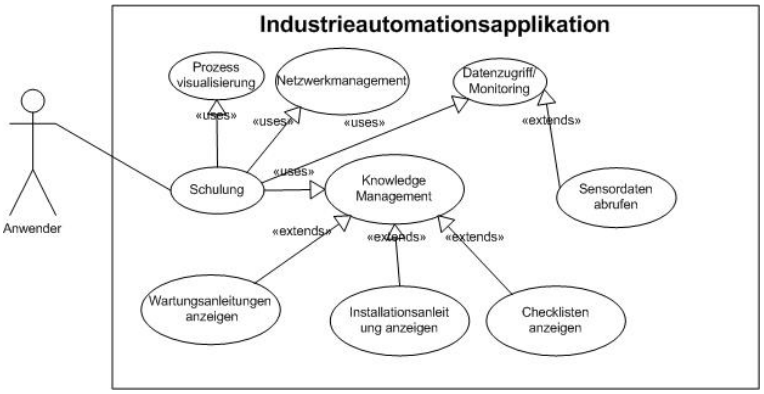

Abbildung 2.4: Use Case Schulung

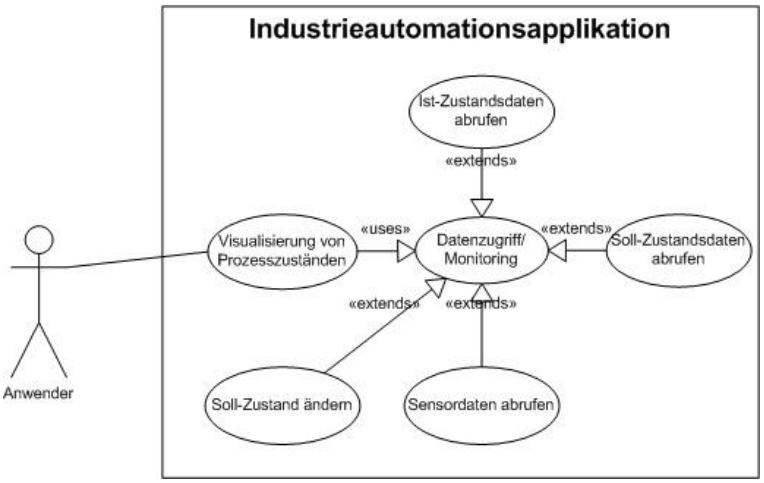

Abbildung 2.5: Use Case Prozessvisualisierung

2.2.2 Netzwerkmanagement

Der Netzwerkscan dient als Grundlage für viele andere Aktionen und kann daher als Generalisierung derselben verstanden werden. Bevor eine Kommunikation mit einem Gerät durchgeführt werden kann, muss zuerst ermittelt werden, ob es überhaupt im Netzwerk vorhanden und damit erreichbar ist. Dazu ist es nötig, dass das Smartphone mit dem Netzwerk kommunizieren also das Protokoll verwenden kann.

Gerät identifizieren

Ein Gerät identifizieren zu können, ist oftmals hilfreich, wenn sich in einem Schaltschrank mehrere Geräte des gleichen Typs befinden. Neben einer physischen Beschriftung kann eine blinkende LED oder ein Summer der Wartungstechnikerin oder dem Wartungstechniker helfen, etwa bei einem bevorstehenden Tausch, Fehler zu vermeiden.

Gerätefirmware aktualisieren

Die Aktualisierung der Firmware eines Gerätes kann mehrere Gründe haben. Zum Einen kann ein im Betrieb erkannter Fehler durch eine neue Version der Firmware behoben werden. Auf der andere Seite können sich die Aufgaben eines Gerätes über die Zeit verändern. Dies ist zum Beispiel der Fall, wenn dem Gerät neue Funktionen hinzugefügt werden, oder sich die zu verarbeitenden Daten ändern.

2.2.3 Datenzugriff/Monitoring

Im Zuge der Überprüfung kann es nötig sein, Daten von einem Gerät auszulesen. Dieser Anwendungsfall ist eine Generalisierung, wobei die Art der Daten nicht genauer spezifiziert wird. Solche Spezialisierungen werden in den nächsten Absätzen besprochen.

Diagnosenachrichten abrufen

Oft besitzen Geräte einen Fehlerspeicher bzw. einen Speicher der gewisse Ereignisse (z.B. eine Spannungsunterversorgung) dokumentiert. Diese Nachrichten können bei etwaigen Fehlerfällen abgerufen werden, um eine Diagnose schnell und effizient durchführen zu können.

Sensordaten abrufen

Um bei einer vorangegangenen Wartung eine Kontrolle der Funktionsfähigkeit durchführen zu können, kann es nötig sein, Sensordaten einer Plausibilitätsüberprüfung zu unterziehen. Dazu ist es notwendig, die Daten auf das Smartphone zu übertragen, und diese in geeigneter, visualisierter Form anzeigen zu können.

Ist-Zustandsdaten der Anlage abrufen

Zur Überwachung von Geräten ist neben den Sensordaten, die von den Geräten verarbeitet werden, auch der innere Zustand der Anlage, der natürlich auch eine Folge der Sensordaten sein kann, notwendig, um ein ordnungsgemäßes Funktionieren der Anlage zu gewährleisten.

Soll-Zustandsdaten abrufen

Bei einer graphischen Anzeige von Daten ist es für das Überwachungspersonal oft hilfreich, neben den Ist-Werten auch die Sollwerte in Relation darzustellen. Damit können Zustandsabweichungen schnell und eindeutig erkannt werden und auch das Ausmaß.

Soll-Zustand ändern

Eine Änderung des Soll-Zustandes bzw. der Soll-Parameter kann im Betrieb notwendig werden, wenn die Qualität der auszuführenden Aktion oder des zu produzierenden

Gutes nicht den Vorgaben entspricht. Diese Anpassungen können entweder im Leitstand mit Zeitverlust passieren, oder ohne Zeitverlust, direkt an der Anlage mit Smartphones oder Tablets, wenn die Ungenauigkeit vor Ort entdeckt wurde.

2.2.4 Knowledgemanagement

Gerade bei Schulung und Ausbildung neuer Mitarbeiterinnen und Mitarbeiter kann es helfen, Anleitungen auf Smartphones oder Tablets abrufen zu können. Diese Anleitungen sollen sozusagen Lernhilfen darstellen. Auch bei diesem Anwendungsfall handelt es sich um eine Generalisierung, die im Weiteren spezialisiert werden soll.

Checklisten anzeigen

Checklisten oder Prozeduranleitungen haben den Zweck Mitarbeiterinnen und Mitarbeiter bei der Erledigung von Aufgaben zu unterstützen. Dies ist dann interessant, wenn die zu erfüllende Aufgabe komplex ist, d.h., zum Beispiel viele Schritte nötig sind, um die Aufgabe zu erfüllen. Diese Listen anzuzeigen oder auch mit ihnen zu interagieren (z.B. ein Häkchen zu setzen oder einen Zeitpunkt einzugeben) stellt für graphische Benutzeroberflächen von Smartphones keine schwere Aufgabe dar. Als positiver Nebeneffekt wird möglicherweise auch der Papierverbrauch verringert. Um eine langwierige Suche der Anleitungen zu verhindern, können RFID-Tags oder Barcodes am Gerät die Suche unterstützen.

Wartungsanleitungen anzeigen

Sollte es im Zuge der Ausbildung zu Wartungsarbeiten innerhalb der Anlage kommen, können die Informationen für die Wartung schnell und unkompliziert auf einem Smartphone oder Tablet angezeigt werden.

Installationsanleitung anzeigen

Da auch die Installation von neuen Geräten in der Ausbildung enthalten sein könnte, sollten neben den Wartungsanleitungen auch Installationsanleitungen angeboten werden können.

2.3 Anforderungsanalyse

Aus den bisher definierten Use Cases soll nun eine Analyse der Anforderungen erstellt werden. Dies ist nötig, um in weiteren Schritten benötigte Technologien und Herangehensweisen bereitzustellen, welche eine bestmögliche mobile Anwendung für die beschriebenen Anwendungsfälle zum Ziel haben soll.

2.3.1 Protokolle

Wenn Applikationen für Smartphones erstellt werden sollen, so müssen einige Einschränkungen in Kauf genommen werden. Die Hardware eines Smartphones oder Tablets lässt sich im Regelfall, außer im Bezug auf den Speicher, nicht erweitern. Soll also nicht eigens für eine sehr spezielle Verwendung ein Smartphone hergestellt werden, das direkt über das zu verwendende Protokoll kommunizieren soll, so muss auf ein bereits auf dem Markt befindliches oder einen Adapter (so ein solcher existiert) zurückgegriffen werden. Daher müssen für die direkte Kommunikation mit Automationskomponenten Übertragungsprotokolle verwendet werden, die auf Smartphones verfügbar sind.

Um den Anforderungen der Industrieautomation zu genügen, muss ein Protokoll gewisse Voraussetzungen erfüllen. Diese Anforderungen unterscheiden sich teilweise stark, je nach Verwendungszweck. Dabei muss nicht notwendigerweise ein Protokoll alle Anforderungen erfüllen, es können auch mehrere Protokolle parallel genutzt werden. Anhand der zuvor beschriebenen Use Cases lassen sich die Anforderungen im Folgenden zusammenfassen:

Mehrwegkommunikation

Soll die Struktur eines Netzwerks erfasst werden (Scan), so ist es notwendig, viele Geräte gleichzeitig ansprechen zu können. Das heißt, es muss möglich sein entweder Multicast- oder Broadcastnachrichten versenden zu können. Bei Multicastnachrichten werden dabei mehrere Geräte angesprochen, die sich zuvor in einer Gruppe (Multicastgruppe) angemeldet haben, um die Nachrichten empfangen zu können. Bei Broadcastnachrichten wird keine Rücksicht auf Gruppenzugehörigkeiten genommen, alle Geräte können diese Nachrichten in der Regel empfangen.

Einwegkommunikation

Im Gegensatz zur Mehrwegkommunikation ist die Einwegkommunikation notwendig, wenn Daten eines spezifischen Gerätes angefordert werden sollen, wie zum Beispiel Diagnosenachrichten. Daher muss es auch die Möglichkeit von Unicastverbindungen geben. Es wäre wenig zweckmäßig, wenn das für ein Gerät gedachte Kommando von allen Geräten verarbeitet werden müsste, allein schon aus Performancegründen, beispielsweise bei Kommunikationsteilnehmern mit geringer Rechenkapazität.

Datenübertragungsrate

Bei einigen Anwendungsfällen ist eine hohe Übertragungsgeschwindigkeit notwendig. Sollen etwa Schulungsunterlagen oder Wartungsanleitungen bereitgestellt werden oder große Datenmengen, wie Videos von Kamerasystemen, übertragen werden, ist eine langsame Verbindung hinderlich. Für die Übertragung von Steuerinformationen oder Zustandsinformationen bei Konfigurationsvorgängen und dem Engineering hingegen ist die Datenübertragungsrate eher nicht so relevant.

Fehlertoleranz

Es müssen Maßnahmen bereitgestellt werden, die dazu geeignet sind, Fehler beseitigen bzw. ignorieren zu können. Wird beispielsweise die Firmware eines Gerätes aktualisiert, darf ein fehlerhaftes oder verloren gegangenes Paket nicht dazu führen, dass die Aktualisierung abbricht, oder, schlimmer noch, das Gerät nicht mehr verwendbar ist. Weiters kann auch die Fähigkeit zur *Selbstheilung* notwendig sein. Dies ist insbesondere dann sinnvoll, wenn die Kommunikation zwischen zwei Geräten über einen direkten Kanal nicht möglich ist - zum Beispiel bei größeren Entfernungen - und Daten über mehrere Stationen weitergeleitet werden müssen.

Erweiterbarkeit

Bei Austausch eines fehlerhaften Gerätes oder dem Hinzufügen von neuen Geräten zu einem Netzwerk ist die Möglichkeit einer automatischen Einbindung sinnvoll, um den Konfigurationsaufwand zu minimieren. Dazu gehört, unter Anderem, das Erkennen, welche Services von den anderen Geräten im Netzwerk angeboten werden (*Service-Discovery*).

Sicherheit

Die Sicherheit spielt in der Industrieautomation eine immer größere Rolle. Fehler von Geräten oder innerhalb des Kommunikationsnetzes, seien es unabsichtlich oder absichtlich herbeigeführte, können einen großen Schaden, wie Personenschäden oder monetäre Schäden, anrichten. Daher sind neben der Schulung von Mitarbeiterinnen und Mitarbeitern auch die Sicherheitsmerkmale der verwendeten Protokolle von großer Bedeutung. Gerade durch die Verwendung von drahtlosen Kommunikationspfaden muss auf die Sicherung großer Wert gelegt werden.

Auf Basis der Use Cases können folgende Anforderungen für die Sicherheit der eingesetzten Protokolle definiert werden:

- Vertraulichkeit (Confidentiality): Daten dürfen nur von dem Teilnehmer gelesen werden können, für den sie bestimmt sind.
- Integrität (Integrity): Daten dürfen nicht verändert werden können, ohne dass diese Veränderung erkannt werden kann.
- Authentifikation (Authentication): Soll ein Service genutzt werden, so muss immer eindeutig nachvollziehbar sein, wer dieses Service benutzen will. Wenn die Identifikation erfolgreich war, soll der Zugriff auf das gewünschte Service erlaubt, andernfalls untersagt werden.
- Schlüsselaustausch (Key management): Es sollen Methoden zur Verfügung stehen, die eine sichere Schlüsselübergabe oder Verteilung gewährleisten.
- Datenaktualität (Data Freshness): Ein wiederholtes Senden von Daten muss, wenn nötig, erkannt werden, um Replay-Attacken zu verhindern.

Anhand dieser Anforderungen,

- Mehrwegkommunikation,
- Einwegkommunikation,
- Datensparsamkeit,
- Datenübertragungsrate,
- Fehlertoleranz,
- und Sicherheit,

sollen Protokolle im Verlauf dieser Arbeit auf ihre Eignung für bestimme Use Cases analysiert werden.

2.3.2 Softwarearchitektur

Mobile Applikationen sind in erster Linie als Ergänzung zu bestehenden Engineering-Werkzeugen zu verstehen. Die verwendeten Kommunikationsprotokolle und Tools sind bei der Einführung der mobilen Applikationen oft schon vorhanden, und die Protokollstacks implementiert. Im Hinblick auf eine Weiterentwicklung der Protokolle und Smartphone- bzw. PC-basierten Anwendungen sollte bei der Implementierung auf eine gemeinsame Codebasis Wert gelegt werden. Damit soll verhindert werden, dass bei einer Änderung unnötige Ressourcen für die parallele Softwareentwicklung oder Änderung verwendet werden müssen. Dies bedarf einer intelligenten Softwarearchitektur, welche die Eigenheiten der einzelnen Plattformen berücksichtigt (siehe Abbildung 2.6).

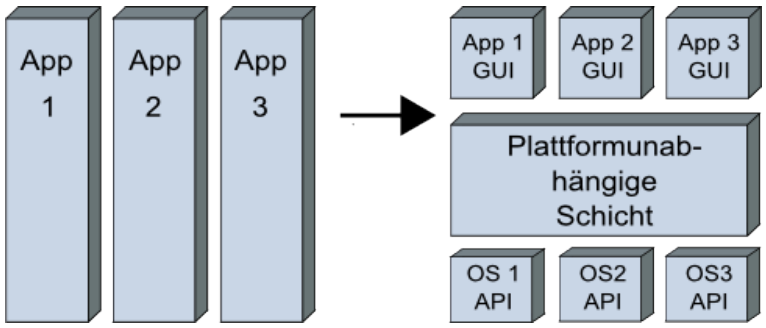

Abbildung 2.6: Wartungsfreundliche Softwarearchitektur

Die wichtigste Anforderung an die Softwarearchitektur, die alle Use Cases umfasst, ist die **Aufteilung in Komponenten**. Sie ist entscheidend für eine spätere Wiederverwendbarkeit und gute Wartbarkeit der Softwareteile. Außerdem sollten einzelne Komponenten über Interfaces abstrahiert werden, um von den restlichen Softwareteilen abgeschirmt zu sein. Damit kann ein einfacher Daten-/ und Komponentenaustausch erreicht werden.

Eine mögliche Aufteilung könnte folgendermaßen aussehen:

- Benutzeroberflächen: Es sollen nach Möglichkeit kleinteilige Ansichten erzeugt werden, die bestimmte Aufgaben erfüllen, und dann zu aufwendigeren zusammengefügt werden. Beispiele dafür sind eine Ansicht der gescannten Geräte eines Netzwerkes und die Veränderung von Parametern einzelner Geräte. Diese Aufteilung ist vorteilhaft, da ein mobiles Gerät meist keinen so großen Bildschirm besitzt, um alle Informationen auf einmal anzeigen zu können. Dies erleichtert außerdem die Wartbarkeit. Die Ansichten sollten außerdem so wenig Logik wie möglich enthalten, da sie selten wiederverwendet werden können, z.B. auf unterschiedlichen Plattformen.

- Zentrale Managementinstanzen: Die Kommunikation, auch innerhalb der Applikation, sollte über Kommandos erfolgen, bei denen über die darunter liegenden Funktionen wenig Wissen vorausgesetzt werden muss. So kann ein Scan durchgeführt werden, bei dem der Entwickler wenig bis kein Wissen über das dafür vorgesehene Protokoll braucht. Dasselbe gilt für die Übertragung von Daten zu und von dem Gerät wie Firmwaredownloads und Diagnoseanfragen. Dadurch können auch Erweiterungen, wie z.B. neue Kommandos, mit geringerem Aufwand hinzugefügt werden.

- Protokollimplementierungen: Bei Verwendung mehrerer Protokolle sollte jeder Protokollstack separat implementiert sein. Dadurch kann bei einer Änderung ein Austausch problemlos durchgeführt werden. Außerdem sollte der Zugriff auf die Implementierungen so generisch wie möglich erfolgen, beispielsweise über Kommandos, bei denen abgefragt werden kann, ob sie verfügbar sind.

2.3.3 Softwareentwicklung

Durch immer kürzere Produktlebenszyklen im Bereich der Endanwenderprodukte nimmt der Leistungs- und Funktionsumfang bei Smartphones schnell zu. Dieser Umstand erfordert daher auch bei der Entwicklung von Apps ein gut durchdachtes ALM (**A**pplication **L**ifecycle **M**anagement). Dadurch ist es auch notwendig, ein geeignetes Vorgehensmodell für die Applikationsentwicklung zu nutzen, wenn es bereits vorhanden ist, oder zu etablieren. Zusätzlich ist es auch nach der ersten Entwicklung wichtig, Prozeduren für Updates und Erweiterungen zu definieren. Es wäre zum Beispiel wenig sinnvoll, wenn für ein Update immer die gesamte Applikation ersetzt werden müsste, zumindest bei Desktopanwendungen. Wenn möglich sollten Updateprozeduren im mobilen Bereich ebenfalls nicht die gesamte Applikation ersetzen, da die Datenmenge im Mobilfunknetz meist begrenzt ist, und die Übertragungsrate niedriger sein kann.

2.3.4 Anleitungen/Checklisten

Für Schulungs- und Wartungszwecke sollten Anleitungen in einer Form zu Verfügung stehen, die auf mobilen Geräten ohne großen Aufwand verwendet werden können, zum

Beispiel im PDF- oder HTML-Format. Besteht die Notwendigkeit gewisse Daten bei der Wartung notieren zu müssen, können beispielsweise Eingabemasken mit dahinterliegender Datenbank Verwendung finden.

KAPITEL 3

Übertragungsprotokolle

In diesem Abschnitt werden Kommunikationsprotokolle beschrieben, die auf mobilen Geräten verfügbar sind. Dabei soll ein Überblick verschafft werden über die Eigenschaften, die für einen Einsatz für die zuvor beschriebenen Use Cases relevant erscheinen. Anschließend soll die Eignung der Protokolle für die einzelnen Use Cases analysiert werden.

3.1 WLAN (IEEE 802.11)

Die Informationen für dieses Unterkapitel wurden unter anderem von [PROT2012, S.279-354] und [IEEE2] extrahiert. Der IEEE 802.11-Standard (siehe [IEEE2]), besser bekannt unter dem Namen WLAN (**W**ireless **L**ocal **A**rea **N** etwork), wurde 1999 eingeführt. Dabei handelt es sich um einen Standard für Funknetzwerke, welcher den MAC- (**M**edium **A**ccess **C**ontrol) und den PHY- (Physical) Layer spezifiziert. WLAN operiert im 2,4 und 5 GHz-Bereich, wobei es sich um das lizenzfreie ISM (**I**ndustrial, **S**cientific and **M**edical Band) Frequenzband handelt. Der Umstand der Lizenzfreiheit ist wohl auch einer der Hauptgründe, weshalb dieses Protokoll im Consumerbereich sehr weit verbreitet ist. Die Datenrate, mit der über WLAN übertragen werden kann, ist aber ebenso ein Grund. Durch die vielen Erweiterungen, die im Laufe der Zeit in den Standard Einzug gehalten haben, wurde die theoretisch maximale Datenrate von ursprünglich 2Mbit/s auf bis zu 6,93Gbit/s in der 802.11ac Erweiterung erhöht. Dies schaffte optimale Verwendungsmöglichkeiten für z.B. Videotelefonie und andere sehr bandbreitenintensive Anwendungen. In der klassischen Industrieautomation sind solche hohen Datenraten noch eher zweitrangig. Sensordaten und Steuerungsbefehle benötigen diese Bandbreite nur selten. Wichtiger ist hier eher die zuverlässige und zeitgerechte Übertragung und, falls es sich um batteriebetriebene Geräte handelt, eine energieeffiziente Ressourcennutzung. Bezüglich Energieeffizienz ist WLAN, zumindest im Consumerbereich, weniger effizient als andere Funktechnologien, wenn großteils kleine Datenmengen übertragen werden.

Das beste Beispiel dafür sind Smartphones, die bei eingeschaltenem WLAN nach kurzer Zeit bereits wieder aufgeladen werden müssen. In der „Industrie 4.0“ [HOL13] könnte die schnelle Datenübertragung gepaart mit effizienten Energiesparmodi, wie beispielsweise in [TLLL09] beschrieben, eher nötig werden. Die maximale Anzahl von Geräten, die gleichzeitig mit einem WLAN verbunden sein können, liegt bei 2007 Geräten.

Die weitere Aufteilung dieses Unterkapitels gliedert sich in folgende Abschnitte:

- Kommunikationsmodi, und
- Sicherheitsmerkmale

3.1.1 WLAN Konfigurationen

Für die Kommunikation über WLAN stehen einige Konfigurationsmöglichkeiten zur Verfügung, die interessantesten Modi sind

- BSS,
- Ad-Hoc Modus,
- ESS,
- Wireless Bridging,
- und Mesh BSS.

Infrastrukturmodus

Der Infrastrukturmodus im 802.11 Standard, auch BSS (**B**asic **S**ervice **S**et) genannt, bezeichnet im Wesentlichen die drahtlose Kommunikation mit Hilfe einer Sterntopologie. Eine beispielhafte Darstellung ist in Abbildung 3.1 zu sehen. Dabei befindet sich in der Mitte der AP (**A**ccess **P**oint), der drahtgebunden mit einem LAN verbunden ist. Die WLAN-Geräte kommunizieren über den AP mit dem LAN sowie anderen im WLAN befindlichen Geräten. Dies ist vorteilhaft, weil ein Gerät, das mit anderen Geräten innerhalb des WLANs Datenpakete austauschen will, außer der Adresse des anderen Gerätes und gewissen Informationen den AP betreffend, keine Informationen benötigt. Zu diesen Informationen gehören beispielsweise die SSID (**S**ervice **S**et **ID**entification) oder das Passwort, falls eine Verschlüsselung stattfindet. Die SSID bezeichnet den Namen des Netzwerks. Der AP übernimmt den Großteil der Arbeit, weil Pakete immer zum AP gesendet, und von dort weiterverteilt werden. Außerdem hat diese Art der Kommunikation den Vorteil, dass Geräte, die aufgrund der gesetzlich beschränkten Sendeleistung von 99mW in Europa über weitere Entfernungen mit anderen drahtlos kommunizierenden Geräten verbunden werden können, da nur die halbe Strecke bis zum AP überwunden werden muss.

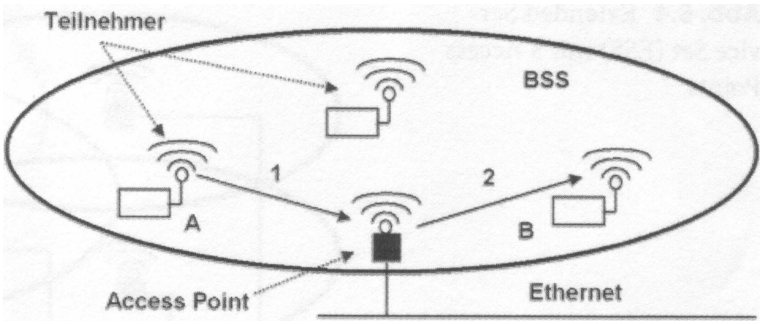

Abbildung 3.1: BSS Schema [PROT2012, S.301]

Ad-Hoc Modus

Der Ad-Hoc Modus, auch IBSS (**I**ndependent BSS) genannt, bezeichnet ein Kommunikationsverfahren, bei dem die Geräte direkt, d.h., ohne Umweg über einen AP, untereinander Daten austauschen können. Dabei muss jeder Kommunikationsteilnehmer über die notwendigen Informationen, wie u.a. die SSID, verfügen. Im Gegensatz zum Infrastrukturmodus müssen jedoch mehr Parameter bei jedem Gerät bekannt sein, was die Konfiguration komplizierter macht. Dieser Umstand dürfte auch der Grund sein, dass dieser Modus in der Praxis seltener Verwendung findet. Ein Vorteil ist jedoch, dass der AP als Single Point of Failure nicht zum Tragen kommt, da alle Teilnehmer gleichberechtigt sind, und jeder, wie in einem drahtgebundenen Netzwerk, mit jedem kommunizieren kann. Durch das Fehlen eines APs verringert sich jedoch die Reichweite, über die die Teilnehmer miteinander kommunizieren können.

ESS

Das ESS (**E**xtended **S**ervice **S**et) ermöglicht eine Erweiterung der Reichweite eines BSS, indem mehrere APs eingesetzt werden, die sich in einem drahtgebundenen Netzwerk befinden (siehe Abb. 3.2). In einem ESS überlappen sich dabei die Funkbereiche der einzelnen APs. Diese APs tauschen die Informationen eines Teilnehmers untereinander aus, wenn dieser aus dem Versorgungsgebiet eines APs in das eines Anderen wechselt, d.h., ab diesem Zeitpunkt kommuniziert nur noch der AP mit dem Endgerät, der die Informationen erhalten hat, der ursprüngliche AP ignoriert das Endgerät. Dabei ist es wichtig, dass alle APs dieselbe SSID verwenden müssen. Ebenso erwähnenswert ist, dass der ursprüngliche IEEE 802.11 Standard die Art, wie die Informationen der einzelnen Teilnehmer ausgetauscht werden soll, nicht genauer spezifiziert. Dies wurde zwar 2003 mit dem Standard 802.11f nachgeholt, jedoch ist dieser Standard nicht verpflichtend für die Hersteller. Daher ist es ratsam, nur APs eines einzigen Hersteller zu verwenden, da oft proprietäre Protokolle verwendet werden, die zwischen verschiedenen Herstellern nicht kompatibel sein müssen.

Wireless Bridging

Die zweite Möglichkeit den Bereich eines Funknetzwerkes zu vergrößern stellt das Wireless Bridging dar. Der Unterschied zum ESS besteht darin, dass die kabelgebundenen Verbindungen der APs zum Netzwerk durch Funkstrecken ersetzt werden. Dies hat den Vorteil, dass für das WLAN, außer einer Stromversorgung der APs, keine weitere Verkabelung notwendig ist.

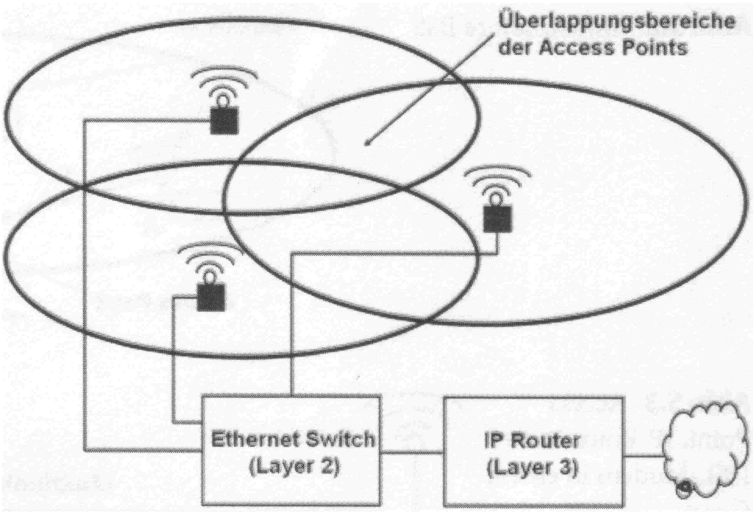

Abbildung 3.2: ESS Schema [PROT2012, S.302]

Mesh BSS

In einer Ergänzung des 802.11 Standards, nämlich der Ergänzung 802.11s, die bereits in [IEEE2] eingearbeitet wurde, wird eine vermaschte Topologie für WLAN beschrieben. Diese Topologie ist insbesondere für größere Ausbreitungsgebiete interessant, da einzelne Teilnehmer Pakete weiterleiten können (Multi-Hop), um die Kommunikation von weit auseinanderliegenden Geräten zu gewährleisten. Es werden keine APs benötigt. Die Teilnehmer des Mesh BSS kommunizieren ausschließlich mit den Mitgliedern des vermaschten Netzwerkes, etwaige andere WLAN-Netze werden ignoriert. Für die Kommunikation mit Geräten außerhalb des Mesh BSS werden sogenannte *Mesh Gates* eingesetzt. Nur über diese Gates ist es Teilnehmern des Mesh BSS möglich, mit anderen Geräten eines BSS zu kommunizieren. In Abb. 3.3 soll die Kommunikation zwischen den verschiedenen Konfigurationen schematisch verdeutlicht werden. Dabei steht in der Abbildung DS für Distribution System und STA für Station, wobei nicht definiert ist, um welche Art von Gerät (statisch oder mobil) es sich handelt. Um mit anderen WLAN-Zellen zu kommunizieren, wird das zuvor genannte Mesh Gate verwendet. Die Kommunikation mit einem drahtgebundenen Netzwerk funktioniert über sogenannte *Portale*, wobei diese einzelnen Komponenten auch in einem Gerät vereint sein können.

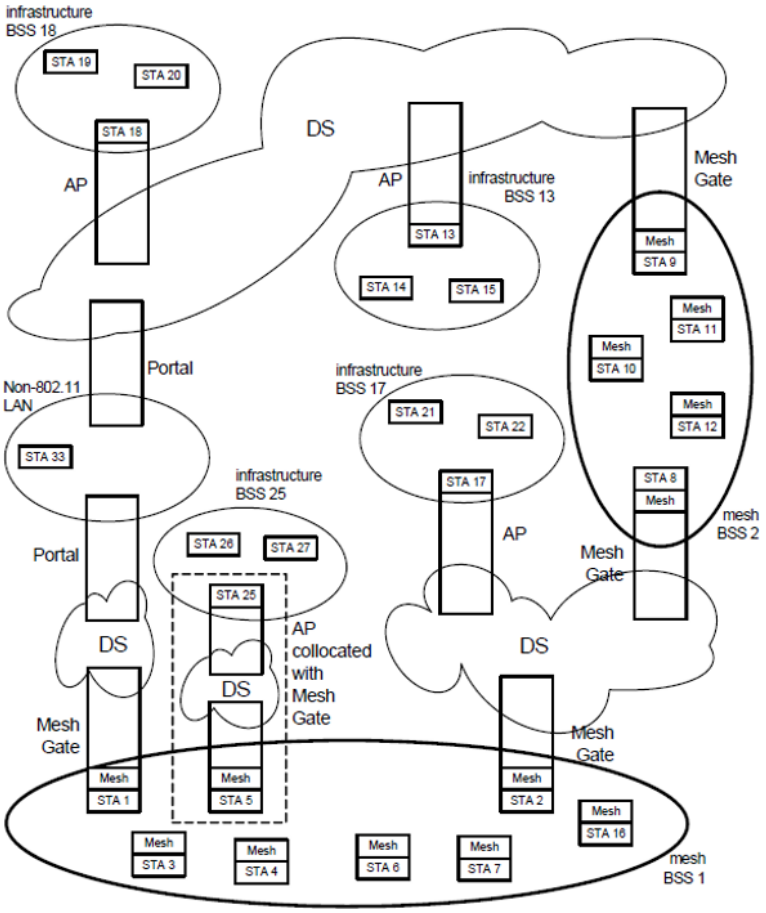

Abbildung 3.3: Mesh BSS Beispielkonfiguration [IEEE2, S.63]

Mit Hilfe dieser Konfigurationsmodi eines WLAN-Netzes können viele Anwendungsgebiete in der Industrieautomation abgedeckt werden, zumindest im Hinblick auf die Art der verschiedenen Topologien. Damit ist eine Vernetzung nicht nur innerhalb der Anlage, d.h., der einzelnen Komponenten möglich, sondern auch eine Verbindung hin zu anderen Bereichen eines Unternehmens, wie z.B. Logistik und Vertrieb, einfach realisierbar.

Drahtlose Kommunikation zwischen den verschiedensten Bereichen eines Unternehmens ist eine einfache und günstige Alternative zu kabelgebundenen Netzen. Durch die ungehinderte Ausbreitung der Funkwellen ergeben sich jedoch sicherheitsrelevante Fragen. Aus diesem Grund beschäftigt sich der nächste Abschnitt mit der Sicherheit bei WLAN-Netzen.

3.1.2 Sicherheitsmerkmale

Der 802.11 Standard bietet sowohl Authentifizierungsmaßnahmen als auch Möglichkeiten der Verschlüsselung von Datenströmen. Authentifizierungsverfahren dienen dazu, um

erkennen zu können, dass empfangene Daten eines Teilnehmers tatsächlich von diesem Teilnehmer stammen und nicht mit einer falschen Identität gesendet wurden. Hierbei handelt es sich im 802.11 Standard einerseits um die Open System Authentication und die Shared Key Authentication bei Verwendung der WEP (**W**ired **E**quivalent **P**rivacy) Verschlüsselung, andererseits um die PSK (**P**re **S**hared **K**ey) und die Enterprise Mode Authentication, bei Verwendung von WPA (**W**ireless **P**rotected **A**ccess) oder WPA2.

Die Open System Authentication ist im Grunde keine stichhaltige Authentifizierungsmethode, da sie lediglich definiert, dass der Open System Authentifizierungsalgorithmus verwendet wird. Da dieser aber bereits auf den Geräten vorhanden ist, und neben der Verwendung dieses Algorithmus keine weiteren Informationen nötig sind, kann nicht sichergestellt werden, dass sich wirklich der richtige Teilnehmer angemeldet hat. Diese Authentifizierungsvariante ist die im Consumerbereich am weitesten verbreitete, falls mit der WEP Verschlüsselung gearbeitet wird.

Bei der Shared Key Authentication besitzen z.B. der AP und der Teilnehmer, der sich anzumelden versucht, denselben Schlüssel. Die Authentifizierungspakete werden zwischen den Teilnehmern mittels WEP verschlüsselt, was diese Art der Authentifizierung etwas sicherer macht als die zuvor genannte. Der Austausch des Schlüssels im Vorfeld der Kommunikation ist im 802.11-Standard nicht geregelt.

Die WEP Verschlüsselung ist heute teilweise noch im privaten Bereich anzutreffen, wenn überhaupt eine Verschlüsselung aktiviert ist. Dies ist problematisch, da schon seit einiger Zeit bekannt ist, dass das verwendete Verschlüsselungsverfahren unsicher ist. Einerseits gibt es nur ein Passwort, das allen kommunizierenden Teilnehmern bekannt sein muss, wodurch die Geheimhaltung schwierig ist. Andererseits wird der Datenstrom immer mit dem selben Passwort verschlüsselt, wodurch das Passwort errechnet werden kann, wenn eine genügend hohe Anzahl an Paketen abgehört wurde. Dies ist auch ein Grund, warum WEP im aktuellen 802.11-Standard, als *„deprecated“*, also als veraltet, bezeichnet wird.

Die PSK-Methode ist der Shared Key Authentication ähnlich, da auch hier ein Schlüssel sowohl beim AP als auch beim Teilnehmer verwendet wird. Im Gegensatz zur Shared Key Authentication werden hierbei die zur Authentifizierung gesendeten *Challenge*- und *Responsespakete* nicht mittels WEP verschlüsselt, sondern bei WPA mit dem TKIP (**T**emporary **K**ey **I**ntegrity **P**rotocol) - hierbei handelt es sich um eine veränderte Variante von WEP - und mit AES (**A**dvanced **E**ncryption **S**tandard) bei Verwendung von WPA2. Interessant ist außerdem die Tatsache, dass bei WPA und WPA2 für die Verschlüsselung der Datenpakete Sessionkeys verwendet werden, und nicht die ganze Kommunikation mit einem Schlüssel, wie bei WEP, gesichert wird.

Die letzte, aber für Firmen wohl interessanteste, Authentifizierungsmethode ist die Enter-

prise Mode Authentication. Bei dieser Methode ist es möglich, einzelne Teilnehmer gezielt zu authentifizieren. Der Unterschied zu den bisherigen Methoden ist die Speicherung der Authentifizierungsinformationen der Teilnehmer auf einem zentralen Server, und nicht, wie sonst üblich, auf dem Accesspoint. Dies geschieht nicht wie bei PSK mit einem Schlüssel für alle Teilnehmer, sondern mittels EAP (**E**xtensible **A**uthentication **P**rotocol). Dabei werden Zertifikate verwendet, die einen öffentlichen Schlüssel enthalten. Über diesen Schlüssel können die Teilnehmer eindeutig authentifiziert werden. Zusätzlich werden mit diesen Schlüsseln die Sessionkeys generiert. Diese Vorgangsweise ist dann sinnvoll, wenn, wie beispielsweise in einem größeren Unternehmen, mehrere Funknetzwerke oder einfach mehrere APs zur Vergrößerung der Reichweite betrieben werden, und verschiedene MitarbeiterInnen unterschiedlichen Zugang zum Netzwerk benötigen. Im Vergleich mit den zuvor genannten Protokollen ist dieses das flexibelste und sicherste Verfahren. Für die Speicherung der Zertifikate werden üblicherweise Unix Server mit RADIUS (**R**emote **A**uthentication **D**ial **I**n **U**ser **S**ervice) bzw. Diameter als Nachfolger oder auf einem Microsoft Server ein Microsoft Authentication Service verwendet.

Aus Gründen der Vollständigkeit, und auch weil mobile Applikationen oft auf Geräten ausgeführt werden, die eine SIM-Karte besitzen, sei hier noch erwähnt, dass noch das EAP-SIM Protokoll existiert. Dieses Protokoll hat den Vorteil, dass ein Nutzer, der sich in ein drahtloses Netzwerk einwählen möchte, keinen Code oder sonstige Eingaben zu tätigen hat, da sich das Gerät mit Informationen, die sich auf der SIM-Karte z.B. des Mobiltelefons befinden, authentifizieren kann.

Die WPA Verschlüsselung stellt zwar eine Verbesserung des WEP Protokolls dar, jedoch gibt es auch bei diesem Protokoll die Möglichkeit der Entschlüsselung einzelner Pakete, wenn auch nicht des gesamten Datenverkehrs. Die Schwachstellen wurden in [TEBM09] beschrieben. Die Entschlüsselung basiert dabei auf der Annahme, dass TKIP verwendet wird, wobei das Intervall, in dem die Sessionkeys erneuert werden, lang (ca. 3600 Sekunden) ist, 802.11e QoS (**Q**uality **o**f **S**ervice) Features unterstützt werden, und das IP-Adressenintervall zu einem Großteil bekannt ist. Dann ist es möglich, ARP-Pakete zu entschlüsseln und den Datenverkehr umzuleiten oder eine gewisse Anzahl an eigenen Paketen zu senden.

Der Vorteil von WPA war es ursprünglich, älteren Geräten die Möglichkeit einer besseren Verschlüsselung zu bieten. Der später entwickelte WPA2 Verschlüsselungsmechanismus erfordert nämlich leistungsfähigere Komponenten. Außerdem ist WPA2 grundsätzlich nicht abwärtskompatibel zu WPA. Einige APs bieten jedoch die Möglichkeit der gemeinsamen Verwendung. WPA2 bietet zusätzlich zu TKIP auch die Möglichkeit der Verschlüsselung mit AES, die momentan als sicherste Verschlüsselungsmethode gilt. Im aktuellen 802.11-Standard ist auch TKIP als *„deprecated“* gekennzeichnet.

Eine weitere Schwachstelle moderner APs ist das Feature WPS (**W**i-Fi **P**rotected **S**etup).

Dabei handelt es sich um eine Möglichkeit, neue Geräte durch eine Taste auf dem AP schnell und unkompliziert anzumelden. Weiters ist es auch möglich, das neue Gerät über die Eingabe eines 8-stelligen Pincodes anzumelden. Aufgrund der schlechten Implementierung ist es möglich, wie in [VIE11] erläutert, den Pincode herauszufinden, und damit in weiterer Folge auch das WPA/WPA2 Passwort. Um diese Sicherheitslücke zu schließen, empfiehlt es sich WPS, wenn möglich, zu deaktivieren.

Da für die Industrieautomation Sicherheitsaspekte einen nicht unwesentlichen Punkt darstellen, kann also zusammengefasst werden:

- Es sollten aktuelle Komponenten für die Netzwerkarchitektur verwendet werden.
- Eine Verschlüsselung mit WPA2 und AES sollte sichergestellt werden.
- Bei größeren Anlagen mit Mitarbeiterinnen und Mitarbeitern unterschiedlicher „Sicherheitsstufen“ empfiehlt es sich, für die Authentifizierung die Enterprise Mode Authentication mit eigenen Authentifizierungsservern zu verwenden.

Weiters bietet WLAN, wie zuvor erwähnt, QoS Funktionalitäten an, die in der Erweiterung 802.11e eingeführt wurden. Diese bietet die Möglichkeit, Datenpakete zu priorisieren. D.h., es können Daten, die zeitkritisch sind, bevorzugt behandelt werden, wie beispielsweise Sensordaten, die zu Regelungszwecken benötigt werden. Für weitere tiefer gehende Informationen zum 802.11 Standard sei auf [PROT2012, S.297ff] verwiesen.

3.2 Bluetooth

Die Informationen für dieses Unterkapitel wurden unter anderem aus [PROT2012, S.355-401], [BLUEHP] und [BLUESTD] zusammengetragen. Bei Bluetooth handelt es sich um einen Funkstandard der SIG (**S**pecial **I**nterest **G**roup), der 1994 von Ericsson entwickelt wurde. Bluetooth arbeitet im 2,4 GHz Band. Aktuell ist die Spezifikation von Bluetooth in der Version 4.2 verfügbar, die im Dezember 2014 veröffentlicht wurde. Die folgenden Erklärungen beziehen sich auf die genannte letzte veröffentlichte Version.

Es stehen 3 verschiedene Möglichkeiten für die Datenübertragung zur Verfügung, nämlich der BR/EDR-Modus (**B**asic **R**ate /**E**nhanced **D**ata **R**ate), der LE-Modus (**L**ow **E**nergy), auch Bluetooth Smart oder Bluetooth Smart Ready genannt, und der AMP-Modus (**A**lternate **M**AC/**P**HY). Der BR/EDR-Modus arbeitet dabei auf maximal 79 verschiedenen Frequenzen mit einer Bandbreite von 1 MHz. Die Datenübertragungsrate der Basic Rate beträgt 1 Megabit/s. Bei Verwendung der Enhanced Data Rate kann eine Datenübertragungsrate von maximal 3 Megabit/s erreicht werden. Der LE-Modus bietet eine Bandbreite von 2 MHz bei einer Datenübertragungsrate von 1 Megabit/s bei maximal 40 Frequenzen, und darüber hinaus im Gegensatz zum BR-EDR-Modus aufgrund anderer Verbindungstypen noch zusätzliche Möglichkeiten zur Einsparung von Energie.

Die Datenübertragung im AMP-Modus basiert auf dem 802.11-Standard und bietet eine theoretische Datenübertragungsrate von 24 Megabit/s [BLUEAMP]. Die reale Datenrate, mit der ein Gerät übertragen kann, richtet sich nach der Anzahl der kommunizierenden Geräte bzw. nach der Anzahl der sendenden Geräte.

Die Entfernungen, die dabei überwunden werden können, richten sich nach der Leistungsklasse der Bluetooth-Module. Derer gibt es drei, nämlich

- Power Class 1: Die Sendeleistung beträgt hier 100mW, mit einer Reichweite von maximal ca. 100 Metern. Hauptanwendungsgebiet, laut [BLUEHP], sind hier industrielle Anwendungsfälle.
- Power Class 2: Die Sendeleistung beträgt hier 2,5mW, mit einer Reichweite von maximal ca. 10 Metern. Diese Klasse ist, laut [BLUEHP], die am meisten eingesetzte Klasse, bspw. in mobilen Geräten.
- Power Class 3: Die Sendeleistung beträgt hier 1mW, mit einer Reichweite von maximal ca. 1 Meter.

Das Wissen um die Leistungsklassen ist auch insofern wichtig, als Bluetooth-Module häufig in batteriebetriebenen Geräten zum Einsatz kommen, bei denen die Reichweite teilweise weniger entscheidend ist als eine lange Laufzeit. Daher werden bei Bluetooth einige Energiesparmodi verwendet, welche die Laufzeit zusätzlich erhöhen (LE-Modus). Dies ist besonders für Sensoren interessant, die, batteriebetrieben, über Monate oder Jahre Daten liefern sollen.

Ein Bluetooth-Modul besteht aus mehreren Komponenten, dem Host, dem HCI (**H**ost **C**ontroller **I**nterface) und einem oder mehreren Controllern. In Abbildung 3.4 werden verschiedene Konfigurationen gezeigt.

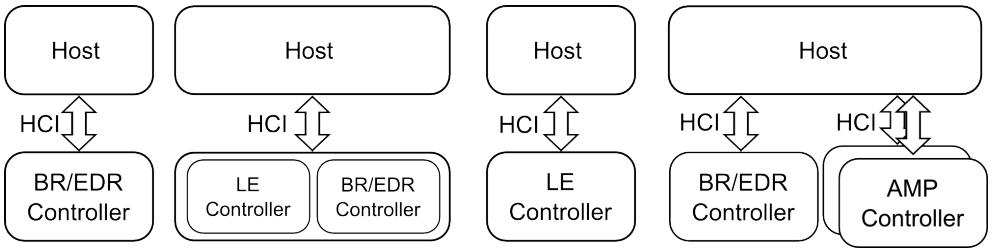

Abbildung 3.4: Mögliche Bluetooth-Konfigurationen

Der Host ist über das HCI mit den Controllern verbunden, wobei Teile des HCI sowohl im Host als auch im Controller implementiert sein können. Der Bluetooth-Standard unterscheidet zwischen primären und sekundären Controllern. Als primärer Controller

wird dabei der BR/EDR- und der LE-Controller bezeichnet, d.h., einer der beiden oder eine Kombination daraus müssen im Controller vorhanden sein. Nur diese beiden Typen verfügen über die Möglichkeit, eine Verbindung mit anderen Geräten herzustellen. Der AMP-Controller wird als sekundärer Controller bezeichnet. Nachdem die Verbindung über einen primären Controller hergestellt wurde, kann die Kommunikation an einen der sekundären Controller, falls mehrere zur Verfügung stehen, übergeben werden.

Die weitere Aufteilung dieses Unterkapitels gliedert sich in folgende Abschnitte:

- Datenübertragungsmechanismen,
- Sicherheitsmerkmale.

3.2.1 Datenübertragungsmechanismen

Obwohl Bluetooth denselben Frequenzbereich wie WLAN nutzt, ist die Wahrscheinlichkeit einer Störung, bei gleichzeitigem Vorhandensein beider Netze, gering. Dies ist zwar möglich, aber nur bei starker Auslastung beider Netze. Bluetooth verwendet zur Störungsvermeidung unter anderem zwei Mechanismen, das Frequency Hopping- und das Adaptive Frequency Hopping Verfahren. Beim Frequency Hopping überträgt der Master Daten auf einem Kanal, die Antwort des Slaves geschieht auf einem anderen Kanal. In Abbildung 3.5 ist dies schematisch dargestellt. Die Abfolge und Kanäle, die verwendet werden sollen, werden vom Master vorgegeben. Dies geschieht aufgrund der Geräteadresse und der Echtzeituhr des Masters. Mit der Version 1.2 wurde das sogenannte AFH (**A**daptive **F**requency **H**opping) Verfahren eingeführt, mit dem Störungen besser vermieden werden können. Dabei fließt in die Auswahl der verwendeten Frequenzen zusätzlich noch eine sogenannte *Channel Map* ein. Dadurch können bestimmte Frequenzen ausgewählt werden, die übersprungen werden, weil der Master Informationen über bestimmte Frequenzen besitzt. Ein Beispiel sind statische Funkübertragungen, die kein Frequency Hopping verwenden, daher permanent dieselben Frequenzen verwenden, und dadurch die Datenübertragung stören. Es gibt jedoch ein Minimum an Frequenzen, die verwendet werden müssen, nämlich 20. Aufgrund dieser Verfahren ist es möglich, mehrere Pico-Netze - so werden Netzwerke, bestehend aus mindestens 2 Teilnehmern, genannt, die über Bluetooth kommunizieren - gleichzeitig an einem Ort zu betreiben, ohne gegenseitige Störung. Mit Bluetooth lassen sich Punkt-zu-Punkt Verbindungen, wie z.B. von Smartphone zu Kopfhörer, oder Pico-Netze realisieren.

Bei diesem Kommunikationsverfahren gibt es nur eine Konfigurationsmöglichkeit, was die Netzwerktopologie eines Pico-Netzes betrifft, nämlich die Sterntopologie. Bluetooth verwendet das Master-Slave-Konzept, wobei ein Master theoretisch mit sehr vielen Slaves kommunizieren könnte. Praktisch ist - wie in der Erläuterung über den BR/EDR-Modus beschrieben - die Anzahl möglicher Slaves durch das Protokoll begrenzt. Master kann dabei jedes der kommunizierenden Geräte sein. Dabei kommt es immer darauf an, wer

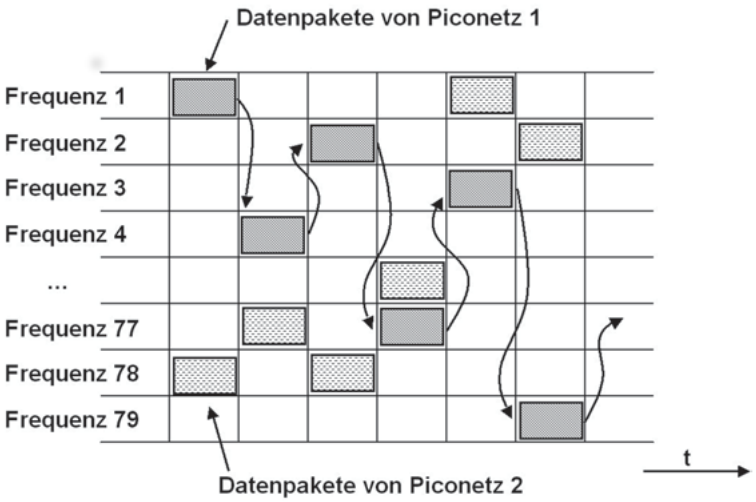

Abbildung 3.5: Frequency Hopping [PROT2012, S.360]

die Verbindung beginnt. Auch bei bereits bestehender Rollenverteilung besteht die Möglichkeit, die Rollen zu tauschen (Master-Slave Role Switch). Um die mögliche Anzahl der kommunizierenden Geräte zu erhöhen, können mehrere Pico-Netze zu sogenannten Scatter-Netzen zusammengeschlossen werden. Zur Verdeutlichung der unterschiedlichen Modi sind diese in Abbildung 3.6 dargestellt. Dabei besteht zwischen die Knoten **D** und **J** in einer 1-zu-1-Verbindung. Die Knoten **F**, **E**, **G** und **H** bilden ein Pico-Netz mit mehreren Slaves, wobei **F** hier die Rolle des Masters übernimmt. Unter Ausschluss der Knoten **L** und **K** bilden alle anderen Knoten ein Scatter-Netz.

Ein Gerät kann Mitglied mehrerer Pico-Netze sein, wobei ein Gerät immer nur einmal in der Rolle des Masters in einem Pico-Netz sein kann. Diese Einschränkung ist notwendig, da das zuvor genannte Frequency Hopping durch die Adresse und die Uhr des Masters berechnet wird. Eine Kommunikation von 2 Geräten in unterschiedlichen Pico-Netzen wird von Bluetooth nicht unterstützt, da keine Routingalgorithmen für die Datenübertragung zwischen zwei Scatter-Netzen zur Verfügung stehen. Diese Funktionalität muss von Protokollen in übergeordneten Schichten übernommen werden. Eine weitere Einschränkung ist, dass ein Slave nur mit einem Master kommunizieren kann. Die direkte Datenübertragung zwischen zwei Slaves ist nicht vorgesehen.

Bluetooth bietet mehrere Möglichkeiten für die Kommunikation an. Diese sind im BR/EDR-Modus und dem LE-Modus unterschiedlich.

BR/EDR-Modus

Das Suchen von Geräten geschieht über den *Inquiry Scan*. Dabei findet unter anderem ein Informationsaustausch über den Namen und angebotene Services ohne einen Verbindungsaufbau statt. Für den Aufbau einer Verbindung von zwei Geräten wird das *Paging* eingesetzt. Voraussetzung dafür ist, dass beide Geräte gewisse Informationen übereinander besitzen. Diese Informationen können entweder über den zuvor genannten *Inquiry Scan*, oder über den im Unterkapitel Sicherheitsmerkmale beschriebenen *Out-of-Band* Mechanismus gewonnen werden.

Besteht eine Verbindung zwischen einem Master und einem Slave, so kann sich der Slave entweder im *Active-* oder *Parked-* Modus befinden. Dabei ist anzumerken, dass maximal 7 Slaves in einem Pico-Netz aktiv sein können. Dabei hat jeder Slave eine eigene 3-Bit Adresse (LT_ADDR), die Adresse 0 ist für Broadcasts reserviert, der Master hat keine Adresse dieses Typs. Die Datenübertragung geschieht in fixen Zeitslots.

Für die Datenübertragung stehen sowohl Unicast-Verbindungen - dabei kann ein Master mit genau einem Slave kommunizieren - als auch Broadcast-Verbindungen - dabei werden alle Slaves angesprochen - zur Verfügung. Unicast-Verbindungen können nur für aktive Slaves verwendet werden. Bei ACL (**A**synchronous **C**onnection-oriented **L**ogical transport) werden Daten ausgetauscht, die keine zeitkritischen Anforderungen haben. Dabei werden die Daten solange gesendet, bis das empfangende Gerät die Daten bestätigt, oder die Daten ihre Gültigkeit verlieren. Diese Kommunikation wird immer vom Master gesteuert. Bei der synchronen Übertragung werden SCO (**S**ynchronous **C**onnection **O**riented) Pakete in dafür reservierten Zeitslots übertragen, wobei ein Slave nicht auf eine Erlaubnis des Masters warten muss, weil in diesen vorgegebenen Intervallen nur dieser Slave senden darf. Die Länge und Anzahl der Slots ist dabei so gewählt, dass eine Bandbreite von 64 Kilobit/s zur Verfügung steht. Bei Verwendung eines Headset mit einem Smartphone wird dieser Verbindungstyp beispielsweise für die Sprachübertragung verwendet. Ein wiederholtes Senden von Paketen ist nicht vorgesehen. Außerdem müssen die Daten eine konstante Datenrate aufweisen. Um höhere Bandbreiten zur Verfügung stellen zu können, wird der eSCO (**e**nhanced **S**ynchronous **C**onnection **O**riented) Pakettyp verwendet. Hier wurde die Bandbreite auf 288 Kilobit/s erhöht. Neben einigen Erweiterungen im Vergleich zu SCO-Übertragungen bietet die eSCO-Übertragung die Möglichkeit, bei Datenverlust Pakete wiederholt zu senden.

Bei Broadcast-Übertragungen stehen 3 verschiedene Typen zur Auswahl. Der ASB (**A**ctive **S**lave **B**roadcast) spricht ausschließlich aktive Slaves an. Der PSB (**P**arked **S**lave **B**roadcast) wird für die Kommunikation mit geparkten Slaves verwendet. Hiermit können auch geparkte Slaves wieder aktiviert werden, wobei entweder der Master einen Slave aktivieren kann, oder der Slave eine Anfrage darüber an den Master senden kann. Der CSB (**C**onnectionless **S**lave **B**roadcast) ist ein Typ, bei dem der Master mit allen Slaves,

die für diesen Broadcast-Typ registriert sind, Informationen übermitteln kann. Ein richtiger Modus zur Übertragung von Multicastnachrichten steht bei Bluetooth nicht zur Verfügung.

In Abbildung 3.6 sind alle möglichen Konfigurationen eines BR/EDR-Pico-Netzes dargestellt.

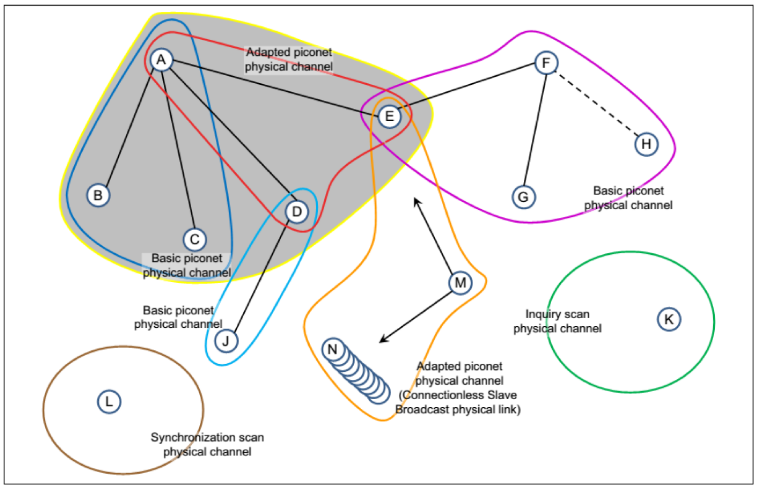

Abbildung 3.6: Pico-Netz Konfigurationen [BLUESTD, S.228]

Der *Adapted piconet physical channel* und der *Basic piconet channel* werden zur Kommunikation zwischen Geräten verwendet, die bereits eine Verbindung aufgebaut haben. Der größte Unterschied der beiden Kanäle ist im ersten Fall die Verwendung des *Adapted frequency hopping*, und im zweiten Fall die Verwendung des *Frequency hopping*. Der *Synchronization scan physical channel* wird von Geräten, die CSBs empfangen wollen, verwendet, um eine Uhrensynchronisation mit dem sendenden Gerät durchführen zu können.

LE-Modus

Anders als der BR/EDR-Modus arbeitet der LE-Modus nicht mit vom Master vorgegebenen Zeitslots sondern mit Events. Es gibt zwei Arten von Events, den *Advertising Event* und den *Connection Event*. Der *Advertising Event* bietet die Möglichkeit, Daten zu übertragen ohne die Notwendigkeit eines Verbindungsaufbaus. Dabei fungiert ein Gerät als *Advertiser*, es bietet so anderen Geräten die Möglichkeit, Anfragen zu senden und Daten auszutauschen. Ein Gerät, dass auf dieses Paket reagiert, wird als *Scanner* bezeichnet, wenn es keine Verbindung aufbauen möchte. Dabei sendet der *Advertiser* ein spezielles *Advertising Packet* (*ADV_SCAN_IND*) auf einem Kanal. Daraufhin antwortet der *Scanner* auf dem selben Kanal mit einer Anfrage (*SCAN_REQ*). Die Antwort auf diese Anfrage (*SCAN_RSP*) wird wiederum auf dem selben Kanal übertragen. Darauf

folgende *Advertising Packets* in dem selben *Advertising Event* werden auf einem anderen Kanal gesendet. Abbildung 3.7 zeigt den Ablauf eines *Advertising Events* mit der Anfrage eines *Scanners*.

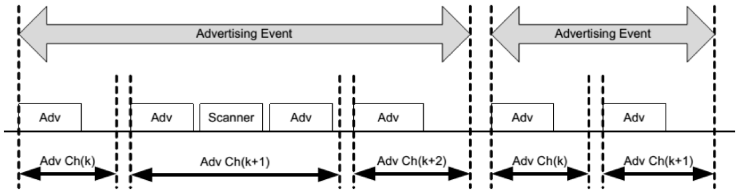

Abbildung 3.7: Advertising Events [BLUESTD, S.172]

Weiters wird diese Übertragungsart auch dazu verwendet, um, einerseits Broadcastnachrichten an beliebig viele Geräte zu senden, und andererseits eine bidirektionale Verbindung zwischen 2 Geräten zu initiieren. Dazu wird ein spezielles Paket im *Advertising Event* gesendet, auf das Geräte reagieren, die eine Verbindung aufbauen wollen. Diese werden auch als *Initiators* bezeichnet. Dabei wird, bei einer erfolgreichen Verbindung, das Gerät, das die Verbindung mit Advertising Paketen begonnen hat, zum Slave, der Initiator erhält die Rolle des Masters. Danach werden in *Connection Events* die Daten übertragen. Interessant in diesem Zusammenhang ist, dass von den zuvor genannten 40 verwendbaren Frequenzen im AFH nur 37 für *Connection Events* verwendet werden dürfen. Die anderen 3 Frequenzen sind ausschließlich für *Advertising Events* reserviert. Abbildung 3.8 zeigt den Verbindungsaufbau und Ablauf. Im Gegensatz zum BR/EDR-Modus, bei dem maximal 7 Slaves aktiv mit dem Master kommunizieren können, kann ein Master im LE-Modus mit einer beliebigen Anzahl - die maximale Anzahl hängt von den zur Verfügung stehenden Ressourcen des Masters ab - von Slaves eine bidirektionale Datenverbindung aufbauen.

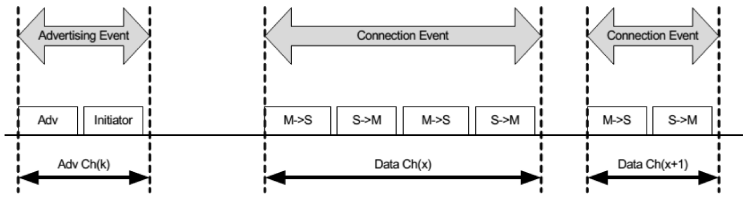

Abbildung 3.8: Connection Events [BLUESTD, S.172]

AMP-Modus

Aufgrund der Vollständigkeit soll hier auch der AMP-Modus kurz erwähnt werden. Hierbei wird die initiale Verbindung mittels Bluetooth aufgebaut. Danach kann, etwa zur Übertragung großer Datenmengen, der AMP-Modus aktiviert werden. Die Datenübertragung erfolgt dann ähnlich wie bei WLAN.

Im folgenden Abschnitt sollen die Sicherheitsmechanismen besprochen werden, die bei Bluetooth genutzt werden.

3.2.2 Sicherheitsmerkmale

Bluetooth bietet verschiedene, je nach Verbindungsart und Version des verwendeten Standards, Authentifizierungsmechanismen und Verschlüsselungsmethoden. Diese sind optional verwendbar, um Herstellern die Möglichkeit zu geben, zu entscheiden, wann welche Methode verwendet werden soll, da nicht alle Anwendungen sicherheitskritisch sind, und die Sicherungsmaßnahmen auch Zeit und Rechenleistung benötigen. Meist ist es nötig, die Geräte vor einer Datenverbindung bekannt zu machen. Dieser Vorgang wird als *Pairing* bezeichnet. Darunter versteht man die erstmalige Verbindungsaufnahme zweier Geräte, bei der ein Schlüssel zwischen zwei Geräten ausgetauscht wird. Für zukünftige Verbindungen kann auf die vorhandenen Schlüssel zurückgegriffen werden, wenn diese auf den Geräten gespeichert sind. Das Speichern der geheimen Schlüssel wird auch als *Bonding* bezeichnet. Die nachfolgend beschriebenen Sicherheitsvorkehrungen sollen dazu dienen,

- Geräte sicher zu authentifizieren,
- Vertraulichkeit der Daten, d.h., eine gute Verschlüsselung zu bieten,
- sowie die Integrität der Daten zu gewährleisten.

Der Bluetooth-Standard definiert auch noch 2 Ziele, die durch die Verwendung der Sicherheitsmechanismen erreicht werden sollen, nämlich:

- Schutz gegen passives Abhören der Kommunikation, und
- Verhinderung von MITM (**M**an **I**n **T**he **M**iddle) Attacken.

Bei der MITM-Attacke handelt es sich um einen Angriff, bei dem ein Eindringling versucht, die Kommunikation über sich umzuleiten. Dabei soll verhindert werden, dass 2 Geräte die eine Verbindung aufbauen wollen, direkt miteinander kommunizieren. Stattdessen versucht der Angreifer die Geräteverbindung zu kompromittieren, damit die beiden Geräte über sein Gerät kommunizieren. Damit hätte er die Möglichkeit, die Kommunikation zu belauschen und gegebenenfalls auch zu verändern.

Bis zur Version 2.0 wurde der SAFER+ (**S**ecure **A**nd **F**ast **E**ncryption **R**outine) Mechanismus der ETH Zürich für die Authentifizierungs- und Schlüsselgenerierung verwendet, und, in modifizierter Form, auch für die Verschlüsselung. Es wurden jedoch Schwachstellen gefunden, um die Authentifizierung zu kompromittieren. In [ShWo05] wurde gezeigt, dass sowohl während des Pairings als auch nach erfolgreichem Pairing, durch die Veranlassung

einer erneuten Übertragung der für das Pairing notwendigen Pakete, sehr einfach der Link Key berechnet werden kann. Der *Link Key* wird bei Bluetooth für die Authentifizierung verwendet und als Parameter für einen *Ciphering Key*. Die Datenverschlüsselung selbst ist von den Schwachstellen allerdings nicht unmittelbar betroffen.

Als Reaktion auf die Schwächen wurde ab der Version 2.1 das Secure Simple Pairing für BR/EDR Verbindungen eingeführt, welches als sicherer gilt. Die Authentifizierungs- und Verschlüsselungsverfahren der früheren Versionen wird aus Kompatibilitätsgründen zwar weiterhin unterstützt, jedoch nicht beschrieben. Beim Secure Simple Pairing wird der ECDH (**E**lliptic **C**urve **D**iffie-**H**ellman) Algorithmus, bei dem es sich um ein Schlüsselaustauschverfahren handelt, der mit öffentlichen und geheimen (Public/Private) Schlüsseln arbeitet, eingesetzt, und kein PIN mehr ausgetauscht. Dieses Verfahren bietet mehr Sicherheit, da bei den Public/Private Schlüsselpaaren nicht der eine Schlüssel aus dem anderen berechnet werden kann, und die Methode nur in eine Richtung funktioniert. Die Algorithmen für die Verschlüsselung und die Authentifizierung wurden nicht geändert.

Bei *Secure Connections*, die in der Version 4.1 eingeführt wurden, wurde als Verschlüsselungsalgorithmus das AES-CCM-Verfahren eingeführt. Für die Authentifizierung wird hier ein HMAC (Keyed-**H**ash **M**essage **A**uthentication-**C**ode), basierend auf der Hashfunktion SHA-256, verwendet. Bei der Schlüsselgenerierung wird eine andere elliptische Kurve eingesetzt. Der entstehende Wert wird im Standard mit HMAC-SHA-256 bezeichnet.

Pairing

Im ersten Schritt des Pairing-Prozesses übertragen die Geräte ihren öffentlichen Schlüssel, außer die Anbahnung des Pairing-Vorgangs geschieht über das später beschriebene *Out-of-Band* Verfahren. Intern wird nach dem Austausch mit dem eigenen privaten Schlüssel und dem öffentlichen Schlüssel des zweiten Gerätes ein Schlüssel mit dem zuvor genannten ECDH-Algorithmus erzeugt.

Der nächste Schritt ist der erste Teil der Authentifizierung. Dieser Schritt gestaltet sich unterschiedlich, je nachdem, welches Verfahren verwendet wird. Es stehen 4 Verfahren zur Verfügung, nämlich

- Numeric Comparison,
- Just Works,
- OOB (**O**ut **o**f **B**and) und
- Passkey Entry.

Numeric Comparison

Die Numeric Comparison ist dazu geeignet, eine MITM vorzubeugen (es besteht die Wahrscheinlichkeit eines erfolgreichen Angriffs von 1:1000000). Voraussetzung für dieses Verfahren ist, dass beide Geräte die Möglichkeit besitzen, sowohl Daten anzuzeigen, als auch Eingaben durch den Benutzer zuzulassen. Abbildung 3.9 zeigt die einzelnen Schritte, die für eine erfolgreiche Authentifizierung durchgeführt werden müssen.

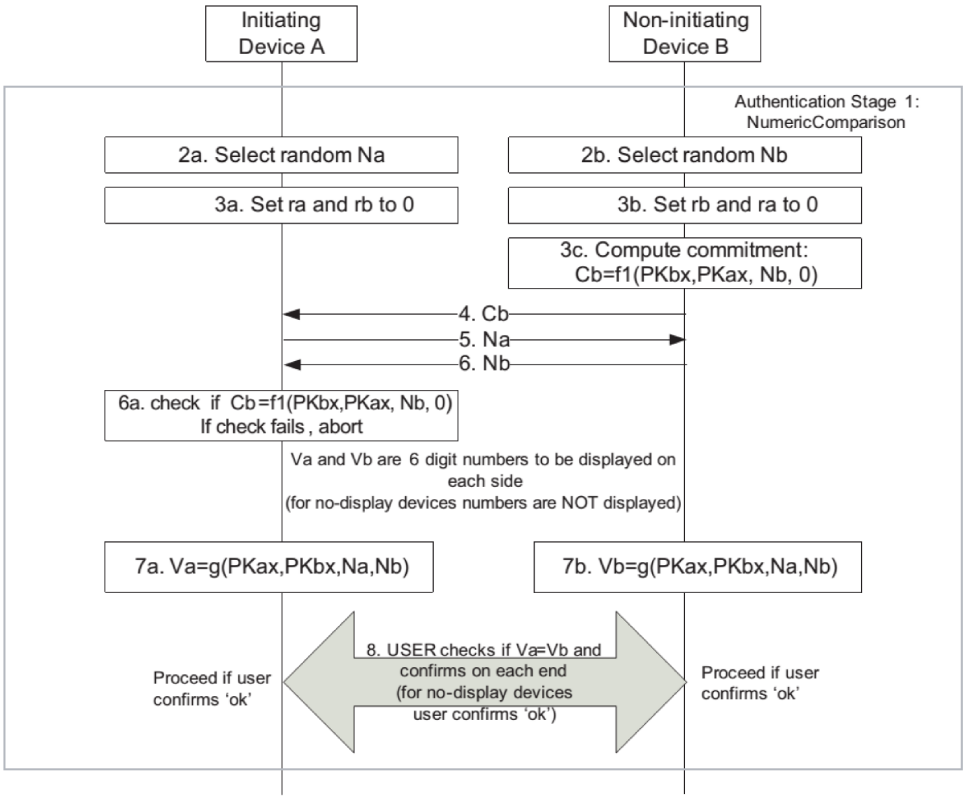

Abbildung 3.9: Numeric Comparison Ablaufdiagramm [BLUESTD, S.1652]

Es werden bei beiden Geräten Zufallszahlen generiert, und gemeinsam mit den öffentlichen Schlüsseln der beiden Geräte wird ein Wert generiert. Die Funktion f_1, die für diese Berechnung herangezogen wird, generiert einen HMAC-SHA-256. Das Gerät, das ursprünglich kontaktiert wurde, übermittelt diesen Wert. Ebenso werden die generierten Zufallszahlen ausgetauscht. Es wird nun geprüft, ob der Wert auf beiden Geräten identisch ist. Bei Ungleichheit wird der Pairingvorgang abgebrochen. Die zuvor generierten Zufallszahlen werden bei jedem Pairingversuch neu generiert. Dadurch wird verhindert, dass ein potentieller Angreifer eine Replay-Attacke - dabei wird eine abgefangene Nachricht erneut gesendet, um eine andere Identität vorzutäuschen - durchführt.

Ebenso wird bei allen hier beschriebenen Authentifizierungsmaßnahmen nach einer gescheiterten Authentifizierung eine Wartezeit eingeführt, die bei folgenden fehlgeschlagenen Versuchen exponentiell gesteigert werden soll. Damit sollen Brute-Force-Attacken - Attacken, bei denen alle möglichen Passwörter für die Entschlüsselung der Daten herangezogen werden - erschwert werden.

Aus den übertragenen Werten - den öffentlichen Schlüsseln und den beiden Zufallszahlen - wird eine 6-stellige Zahl berechnet und auf den Displays der beteiligten beiden Geräte angezeigt. Auch die hier verwendete Funktion g basiert auf dem Hash-Algorithmus SHA-256. Die Berechnung muss auf beiden Geräten dasselbe Ergebnis liefern. Nach der Bestätigung dieser Zahlen auf beiden Geräten ist dieser Authentifizierungsschritt abgeschlossen. Dabei sei noch anzumerken, dass diese angezeigte Zahl ein Artefakt der Sicherheitsalgorithmen ist. Das heißt, kennt der Angreifer nur diese Zahl alleine, ist es dennoch unmöglich, die Identität des jeweils anderen Gerätes anzunehmen, bzw. Nachrichten zu dechiffrieren.

Just Works

Das Just Works Prozedere wird dann eingesetzt, wenn zumindest einer der beiden Kommunikationsteilnehmer weder ein Display besitzt noch eine Eingabemöglichkeit. Dann kann lediglich bestätigt werden, dass die Verbindung akzeptiert wird. Ein Schutz gegen eine MITM Attacke kann nicht gewährleistet werden. Daher sollte darauf geachtet werden, dass während der Pairingprozedur kein eventueller Angreifer diese belauschen kann. Der Ablauf des Informationsaustausches ist derselbe wie bei der Numeric Comparison Prozedur, mit dem Unterschied, dass der Benutzer keine Zahlen aus zuvor genanntem Grund vergleichen kann.

Passkey Entry

Das Passkey Entry Verfahren kann dann eingesetzt werden, wenn zumindest ein Gerät eine Anzeigemöglichkeit für eine 6-stellige Zahl besitzt, und das andere Gerät eine Eingabemöglichkeit bietet. Dies ist beispielsweise bei einem Tablet und einer zu verbindenden Tastatur der Fall. Das Verfahren der Authentifizierung wird in Abbildung 3.10 dargestellt.

Der Benutzer gibt einen „Passkey“ in beide Geräte bzw. den auf dem Display eines Gerätes angezeigten „Passkey“ bei dem anderen Gerät ein. Danach wird für jedes Bit des Passkeys immer die selbe Prozedur durchgeführt. Beide Geräte erzeugen eine Zufallszahl, und berechnen mit der Zufallszahl, den beiden öffentlichen Schlüsseln, und dem jeweiligen Bit des Passkeys einen HMAC-SHA-256. Dieser wird ausgetauscht, ebenso wie die Zufallszahl. Mit den übertragenen Daten verfizieren die Geräte die einzelnen Bits des Passkeys des jeweils anderen Gerätes. Schlägt die Überprüfung eines Bits fehl, wird der ganze Vorgang abgebrochen.

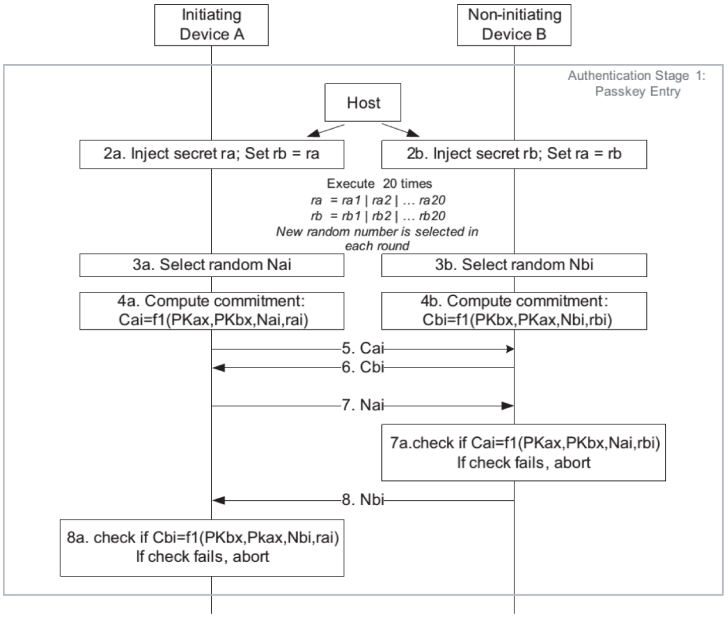

Abbildung 3.10: Passkey Entry Ablaufdiagramm [BLUESTD, S.1656]

Out of Band

Das OOB-Verfahren bietet eine Möglichkeit, sowohl Authentifizierungsinformationen zu übertragen, als auch das Auffinden anderer Geräte ohne die Nutzung einer Bluetooth Funkübertragung. Als Beispiel wird der NFC-Standard genannt. Durch Nutzung dieses Verfahrens kann die Wahrscheinlichkeit einer erfolgreichen MITM-Attacke stark verringert werden. Abbildung 3.11 zeigt die Schritte für eine erfolgreiche Authentifizierung.

Es wird bei beiden Geräten eine Zufallszahl generiert, und ein HMAC erzeugt. Die darauf folgende Übermittlung der Bluetooth-Adressen, des berechneten Werts und der Zufallszahl geschieht aber über einen anderen Übertragungsweg (z.B. NFC). Für die Übermittlung der Daten gibt es 2 unterschiedliche Vorgehensweisen:

1. Es ist eine bidirektionale Übertragung möglich, d.h., beide Geräte können Daten senden und empfangen. Dann findet eine Überprüfung der erhaltenen Werte statt, und die generierten Werte für ra und rb (siehe Abb. 3.11) können weiterverwendet werden.

2. Es ist nur eine unidirektionale Übertragung möglich, wenn beispielsweise eines der Geräte einen passiven NFC-Tag für die Übermittlung der Information bereitstellt, der nur die Möglichkeit zu senden hat. Dann muss das empfangende Gerät seinen Wert von r auf 0 setzen. Dieses Vorgehen stellt sicher, dass die beiden Geräte die selben Werte nach Abschluss der OOB-Prozedur gespeichert haben.

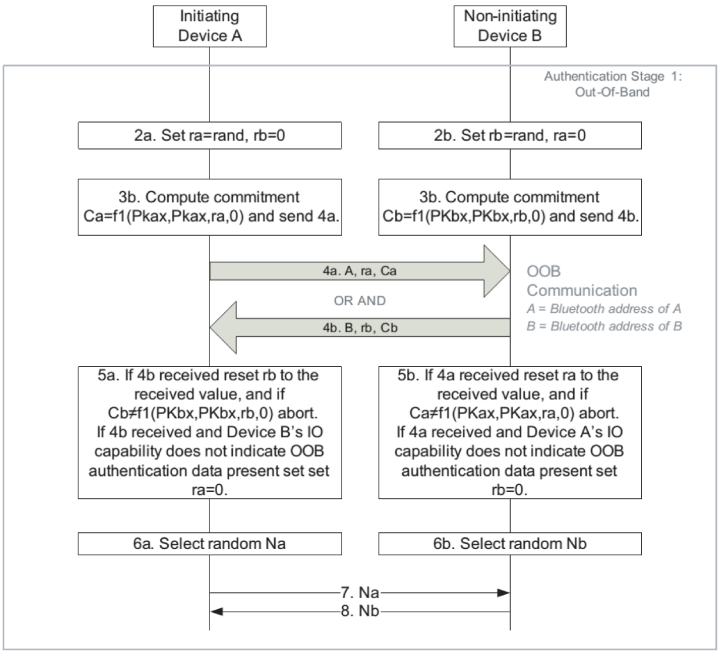

Abbildung 3.11: Out of Band Ablaufdiagramm [BLUESTD, S.1654]

Nun wird auf beiden Seiten eine weitere Zufallszahl generiert und übertragen, wobei dieses Mal wieder die Bluetooth Funkübertragung verwendet wird.

Da die OOB-Prozedur auch das Pairing auslösen kann, und in diesem Fall die öffentlichen Schlüssel nicht zu Beginn übertragen wurden, müssen die öffentlichen Schlüssel nach der OOB-Übertragung, aber vor der Übertragung der Zufallszahlen in Schritt 5, übermittelt werden.

Die nächste Phase ist identisch für alle zuvor beschriebenen Prozeduren. Dabei wird wieder ein HMAC-SHA-256 erzeugt mit den übertragenen Zufallszahlen, den Adressen der beiden Geräte, dem nach der Übertragung der öffentlichen Schlüsseln erzeugten Diffie-Hellman Key, sowie ausgetauschten Informationen über die Fähigkeiten der beiden Geräte bezüglich

- Authentifizierung,
- OOB-Parameter, und
- LMP-Parameter bei BR/EDR, oder LL-Parameter bei BLE.

Danach werden die Werte wieder überprüft. Bei Ungleichheit wird die Pairing-Prozedur abgebrochen.

Damit ist die Authentifizierung abgeschlossen, und es können die Link-Keys (HMAC-SHA-256), die für eine Authentifizierung bei zukünftigen Datenübertragungen benötigt werden, sowie Schlüssel für sichere Datenübertragungen berechnet werden.

Bei BLE ist in der aktuellen Version der Authentifizierungsvorgang identisch mit dem zuvor beschriebenen. In früheren Versionen unterscheidet sich der Vorgang in der Art, dass

- der Authentifizierungsschritt *Numeric Comparison* nicht existiert, und
- die Authentifizierungsschritte *Just Works* und *Passkey Entry* keinen Schutz gegen passives Abhören aufweisen.

Bei Verwendung von AMP wird ein Link-Key erzeugt, wenn auch der BR/EDR Link-Key erzeugt wird.

Bei einer späteren Authentifizierung wird ein Challenge-Response Schema angewandt, das bei früheren Versionen nur eines der beiden Geräte authorisiert, nämlich das empfangende. Dazu sendet ein Gerät einen Wert, gebildet mit der Stromchiffre SAFER-SK128. Die Parameter sind dabei eine Zufallszahl, die dem zweiten Gerät übermittelt wird, die Bluetooth-Adresse des zweiten Gerätes, sowie der Link-Key. Das zweite Gerät berechnet den Wert und sendet ihn zurück. Stimmen die beiden Werte überein, so ist ein Gerät authentifiziert. In der aktuellen Version besteht die Möglichkeit, beide Geräte zu authentifizieren. Dabei werden von beiden Geräten ein HMAC-SHA-256 mit den Adressen der beiden Geräte, dem Link-Key, sowie der Zeichenkette „btdk“ als Eingabewerte erzeugt. Dann übertragen beide Geräte eine Zufallszahl. Mit den beiden Zufallszahlen und dem zuvor generierten Wert als Eingabeparameter wird wiederum ein HMAC-SHA-256 erzeugt, durch welchen sich beide Geräte authentifizieren können. Abbildung 3.12 zeigt diesen Vorgang.

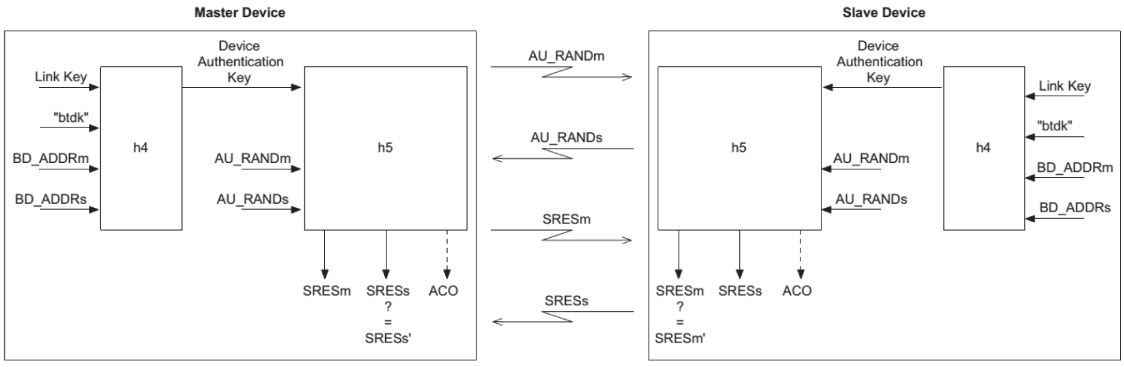

Abbildung 3.12: Authentifizierung Ablaufdiagramm [BLUESTD, S.1640]

Für die verschlüsselte Übertragung von Daten werden bei Bluetooth, abhängig von der Version, zwei Verschlüsselungsalgorithmen verwendet. Die SAFER+ E0 Stromchiffre verwendet dabei einen maximal 128 Bit langen Schlüssel (die Länge des Schlüssels ist abhängig von etwaigen Exportbeschränkungen), die Geräteadresse, eine allgemein bekannte Zufallszahl und die untersten 26 Bit der Echtzeituhr des Masters als Eingabeparameter für einen Schlüsselgenerator, dessen Ergebnis dann mit der Nachricht per XOR-Operation verknüpft wird. Ab der Version 4.0 wird für die Verschlüsselung das AES-CCM-Verfahren angewandt. BLE bietet ausschließlich eine Verschlüsselung nach dem AES-CCM-Verfahren an. Außerdem ist anzumerken, dass Bluetooth, wie WLAN in der WPA-Verschlüsselung, „Session Keys“ verwendet, da für jede neue Aktivierung der Verschlüsselung neue Ciphering Keys verwendet werden sollen.

Wird Bluetooth für die Kommunikation innerhalb einer Automationsanlage eingesetzt, können folgende Punkte für die sichere Verwendung zusammengefasst werden:

- Es sollten nur Komponenten verwendet werden, die die aktuellste Version von Bluetooth mit den erwähnten Sicherheitsmechanismen unterstützen.
- Bei der Datenübertragung sollte nach Möglichkeit eine Verschlüsselung aktiviert werden.
- Da viele Geräte in automatisierten Systemen, wie z.B. Busknoten über kein Display oder eine Eingabemöglichkeit wie eine Tastatur verfügen, sollte bei sicherheitskritischen Datenübertragungen kein *Pairing* über das *Just Works* Prozedere stattfinden. Stattdessen sollte hier eine Übertragung der Authentifizierungsinformationen über das *OOB*-Verfahren erfolgen. Als *OOB*-Verfahren eignen sich beispielsweise NFC oder die Nutzung eines QR-Codes, der sich auf dem Gerät befindet.

Durch die angebotenen Übertragungsverfahren, eignet sich Bluetooth prinzipiell für kleinere Automationsanlagen. Dies ist insbesondere darauf zurückzuführen, dass die möglichen aktiven Teilnehmer begrenzt sind. Was die räumliche Ausdehnung einer Anlage angeht, so kann, bei Verwendung von Geräten der Leistungsklasse 1 durchaus ein effizienter Betrieb auch auf größere Entfernungen gewährleistet werden. Bei zeitkritischen Datenübertragungen kann, wenn der Verlust einzelner Pakete kein Problem darstellt, das *SCO* Verfahren eingesetzt werden. Ist ein Datenverlust nicht tolerierbar, sollte das *eSCO* Verfahren verwendet werden. Müssen große Datenmengen schnell übertragen werden, so kann auf den AMP-Modus zurückgegriffen werden.

Für genauere Informationen zu Bluetooth sei auf [PROT2012, S.355ff] und [BLUESTD] verwiesen.

3.3 NFC

Die Informationen zu den im folgenden beschriebenen Eigenschaften von NFC wurden unter anderem aus [JLMR10] und den Standards [NFCIP-2], [NFCIP-3], [NFCIP-4] und [NFCSEC06] zusammengetragen. NFC (**N**ear **F**ield **C**ommunication) wurde 2002 von NXP Semiconductors und Sony entwickelt, und durch die ECMA (**E**uropean **C**omputer **M**anufacturers **A**ssociation) international standardisiert. Mittlerweile wird NFC vom NFC Forum, bei dem es sich um einen Zusammenschluss verschiedener Firmen handelt, mit dem Ziel der Verbreitung der NFC Technologie weiterentwickelt und überwacht. Bei NFC handelt es sich um eine drahtlose - im Zusammenhang mit NFC wird auch oft von kontaktlos gesprochen - Kommunikationsmethode, deren Reichweite meist bei wenigen Zentimetern liegt. Da NFC auch immer öfters in Smartphones vorhanden ist, bietet es eine gute Ergänzung zu den bisher verwendbaren Trägerkommunikationsprotokollen, um in der Industrie 4.0 das Aufgabengebiet mobiler Applikationen zu erweitern. Es basiert u.A. auf den beiden „RFID“ (**R**adio **F**requency **ID**entification) Standards ISO/IEC 14443 Typ A (MIFARE) der Firma NXP Semiconductors, und JIS X 6319-4 (FeliCa) der Firma Sony. Die Übertragungsgeschwindigkeiten betragen 106 kbit/s bei MIFARE, und 212 kbit/s oder 424 kbit/s bei FeliCa. Wie WLAN und Bluetooth arbeitet NFC im lizenzfreien ISM Band (**I**ndustrial, **S**cientific and **M**edical Band) auf einer Frequenz von 13,56 MHz. Zu den bekanntesten Anwendungsbereichen dieser Protokolle (MIFARE und FeliCa) zählen Zutrittskontroll- oder Zeiterfassungssysteme, bei denen Karten oder spezielle Schlüsselanhänger nur zu den Lesegeräten gehalten werden müssen, um eine Tür zu öffnen, oder Eintritts- bzw. Austrittszeiten zu erfassen. Dabei ist das Lesegerät der aktive Teil, die Karte eine passive Komponente. Durch die Einführung von NFC gilt diese Trennung zwischen aktiven und passiven Komponenten nicht mehr.

Grundsätzlich funktioniert diese Technologie folgendermaßen: Es wird ein Hochfrequenzfeld vom Initiator der Verbindung generiert, das von der Gegenstelle empfangen wird. Die Informationen werden mittels ASK (**A**mplitude **S**hift **K**eying) auf das Trägersignal aufmoduliert. Nun gibt es zwei Kommunikationsmodi, wobei bei ersterem das Trägersignal des Senders mit Hilfe des Lastmodulationsverfahrens als Energiequelle zumindest für die Sendeeinrichtung des Empfängers genutzt wird, um die gewünschten Daten zu übertragen. Dies führt auf der Senderseite zu einem höheren Energieverbrauch. Dieser Modus wird als passiver Modus bezeichnet. Beim zweiten Modus, auch aktiver Modus genannt, erzeugt jeder Kommunikationsteilnehmer sein eigenes Hochfrequenzfeld zur Datenübertragung. Dabei verfügen beide Teilnehmer selbst über eine Energiequelle, um die Daten zu übertragen. Aufgrund der Funktionsweise könnte die Annahme getroffen werden, dass bei NFC immer nur 2 Geräte miteinander kommunizieren können. Tatsächlich können im passiven Modus aber bis zu 16 passive mit einem aktiven Gerät kommunizieren. Dies wird über die SDD (**S**ingle **D**evice **D**etection) realisiert. Dabei gibt der Initiator der Verbindung eine gewisse Anzahl (maximal 16) an Timeslots vor, in denen potentielle Empfänger antworten können. Kommt es zu Kollisionen, wird die Prozedur solange durchgeführt, bis keine Targets mehr im selben Timeslot antworten.

Zusätzlich zu den beiden zuvor genannten Protokollen, aus denen der NFC Standard entstand, wurden mittlerweile noch andere Protokolle für die Datenübertragung hinzugefügt. Das Zusammenspiel verschiedener Protokolle, bzw. die Protokolle selbst, wurde in zwei Standards definiert:

- NFCIP-1 und
- NFCIP-2.

3.3.1 NFCIP-1 ([NFCIP-1])

Der NFCIP-1 Standard definiert ein Protokoll auf Basis der Protokolle MIFARE, FeliCa und des NFCIP-1 Transport Protokolls.

Die Architektur dieses Protokolls veranschaulicht Abb. 3.13. NFC bietet 3 Kommunikationsmöglichkeiten, und zwar:

- Peer-to-Peer Modus: Bei diesem Modus können 2 NFC-Geräte Daten austauschen, wobei das Gerät, das die Kommunikation beginnt, entscheidet, ob der aktive oder der passive Modus verwendet werden soll. Dazu wird das LLCP (**L**ogical **L**ink **C**ontrol **P**rotocol) verwendet, das die Daten dann an höhere Schichten weiterleitet.
- Reader/Writer Modus: Hier übernimmt das NFC-Gerät die Rolle eines Lese/Schreibgerätes für Smartcards und andere passive Komponenten. Es generiert das HF-Feld, um das Target mit Energie für die Datenübertragung zu versorgen. Zu Beginn der Übertragung ist noch offen, welches Protokoll für die spätere Datenübertragung zum Einsatz kommt. Dabei überträgt das passive Element Informationen, ob bspw. der Peer-to-Peer Modus oder ein anderer Standard verwendbar ist. Dadurch ist eine Rückwärtskompatibilität mit anderen, bereits bestehenden, RFID-Infrastrukturen möglich. Außerdem implementieren einige Hersteller von NFC-Geräten zusätzliche RFID-Standards, die im normalen Standard nicht enthalten sind. Die Vorgangsweise bei diesen Systemen ist im Standard NFCIP-2 definiert, um ein breites Anwendungsfeld zu ermöglichen.
- Card Emulation Modus: Bei diesem Modus verhält sich das NFC-Gerät wie eine Smartcard. Es sendet die Daten mit der Energie, die es vom Lesegerät erhält. Dabei werden aber nicht so viele Protokolle abgedeckt wie im Reader/Writer Modus. Dieser Modus bietet aber den großen Vorteil, dass bei sehr stromsparenden NFC-Geräten die Stromversorgung komplett über das Lesegerät gesichert werden kann. Welche Protokolle dabei unterstützt werden, hängt in erster Linie von den verbauten NFC-Komponenten ab.

Durch das im Peer-to-Peer Modus verwendete LLCP werden auch 2 Übertragungsmodi zur Verfügung gestellt die, ähnlich wie UDP und TCP in Verbindung mit dem IP-Protokoll, die Möglichkeit bieten, verbindungslose und verbindungsorientierte Datenübertragungen bereitzustellen. Dabei hat die verbindungslose Variante den Vorteil, dass sie weniger komplex ist, was die Kontrolle des Datenflusses betrifft, und weniger Kommunikations-Overhead benötigt. Die fehlende Kontrolle des Datenflusses hat aber auch zur Folge, dass eine korrekte Übertragung nicht garantiert werden kann, ebenso wenig wie die richtige Reihenfolge der gesendeten Pakete.

Die verbindungsorientierte Methode benötigt im Gegensatz dazu eine geregelte Datenflusskontrolle, die einen Verbindungsaufbau, die Datenübertragung sowie einen Verbindungsabbau beinhaltet. Dadurch ist gewährleistet, dass Daten korrekt gesendet und empfangen werden können, weil es hier möglich ist, die Reihenfolge durch eine Paketnummer einzuhalten, sowie verloren gegangene Pakete gezielt zu wiederholen.

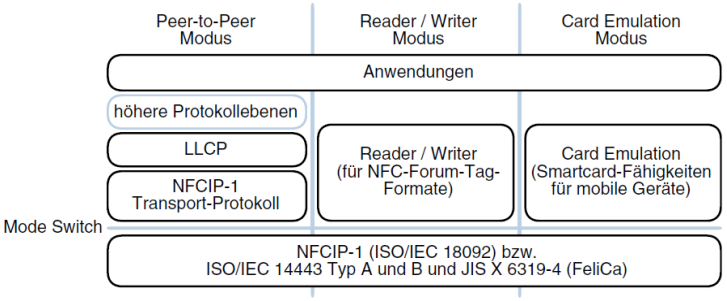

Abbildung 3.13: Aufbau des NFCIP-1 Standards [JLMR10, S.90]

3.3.2 NFCIP-2 ([NFCIP-2])

Der NFCIP-2 Standard definiert das Verfahren, wann welches Protokoll zum Einsatz kommen kann, und wie die Auswahlprozedur für das geeignete Protokoll durchzuführen ist. Bei den auszuwählenden Protokollen stehen 4 Auswahlmöglichkeiten zur Verfügung, nämlich NFC, PCD, VCD und PICC.
NFC behandelt das zuvor beschriebene Protokoll, wobei hier nur der Peer-to-Peer Modus gemeint ist. Von PCD (**P**roximity **C**oupling **D**evice) wird dann gesprochen, wenn die Reichweite der Kommunikationsteilnehmer nicht über 10cm liegt. Diesem Typ entspricht auch der Reader/Writer Modus. Der dritte Protokolltyp, VCD (**V**incinity **C**oupling **D**evice), beschreibt ein RFID Protokoll, bei dem die Entfernung der Geräte unter 1m betragen kann. Der Begriff PICC (**P**roximity **I**ntegrated **C**ircuit **C**ard) steht in diesem Zusammenhang für den Card Emulation Modus, bei dem sich das NFC-Gerät wie eine Smartcard verhält.

Das Schema, nach dem die Auswahl erfolgt, ist in Abb. 3.14 zu sehen. Dabei wird,

falls nicht schon zu Beginn der Kommunikation der Card Emulation Modus ausgewählt wurde, geprüft, ob ein externes HF-Feld empfangen werden kann (Collision Avoidance). Falls dem so ist, wird in den NFC Modus gewechselt, nachdem ein Startbefehl erhalten wurde, um die Anwesenheit und die möglichen Kommunikationsmodi bekanntzugeben. Falls entweder der PCD oder VCD Modus ausgewählt wurde, detektiert das Gerät erneut, ob für die betreffenden Modi ein HF-Feld existiert. Existiert mittlerweile ein Feld, wird die ganze Prozedur erneut durchlaufen. Sonst wird nach einer gewissen Wartezeit vom NFC- Gerät als Initiator der Verbindung ein HF-Signal ausgesendet, um etwaige Kommunikationsteilnehmer zu aktivieren. Die Zeit, in der der Initiator der Verbindung nach einem aktivem HF-Feld suchen muss, bevor er sein HF-Feld aufbaut, wird dabei nach der Formel

$$T_{Wait} = T_{IDT} + n * T_{RFW}$$

berechnet. Der antwortende Teilnehmer muss dann innerhalb eines Zeitintervalls von

$$T_{Wait} = T_{ADT} + n * T_{RFW}$$

antworten.

Zur Erklärung der ersten Formel ist es wichtig, die zweite Formel zu kennen: T_{ADT} ist die aktive Wartezeit nach dem Abschalten des HF-Feldes des Initiators. Sie muss zwischen ca. 56,6μs$(768/fc)$ und 188,7μs$(2559/fc)$ lang sein. fc entspricht dabei der Sendefrequenz von NFC (13,56 MHz).

n ist eine Zufallszahl zwischen 0 und 3, welche die Wahrscheinlichkeit, dass mehrere Geräte zur selben Zeit antworten, minimieren soll. T_{RFW} entspricht ca. 37,8μs$(512/fc)$, und ergibt mit n ein Zeitfenster, in dem die Slaves antworten können.

Nun zur ersten Formel: T_{IDT} ist die initiale Wartezeit, die der Initiator warten muss, bis er sein HF-Feld aktiviert. Sie entspricht einem Wert von 302μs $(4096/fc)$. Dieser Wert ist dabei so gewählt, dass er der Zeitspanne entspricht, in der ein Slave antworten kann, nämlich: maximale Wartezeit T_{ADT} =188,7μs + maximales Zufallsintervall $3 * T_{RFW}$=113,3μs. Dieser Wert entspricht gerundet genau dem Wert von 302μs, und damit der initialen Wartezeit T_{IDT}. n und T_{RFW} ergeben, wie zuvor, das Zufallsintervall, in dem der Initiator seine Übertragung beginnen kann.

3.3.3 Sicherheitsmerkmale

Wie bei den zuvor beschriebenen Protokollen, ist auch bei NFC der Schutz vor einem ungewollten Abhören der Datenverbindung oder einer Kompromittierung der Datenpakete ein wichtiges Thema.

Bei NFC ist die Wahrscheinlichkeit eines erfolgreichen Abhörens, wegen der nötigen Nähe der Kommunikationspartner zueinander zwar geringer, aber laut [NFCSEC06] kann bis

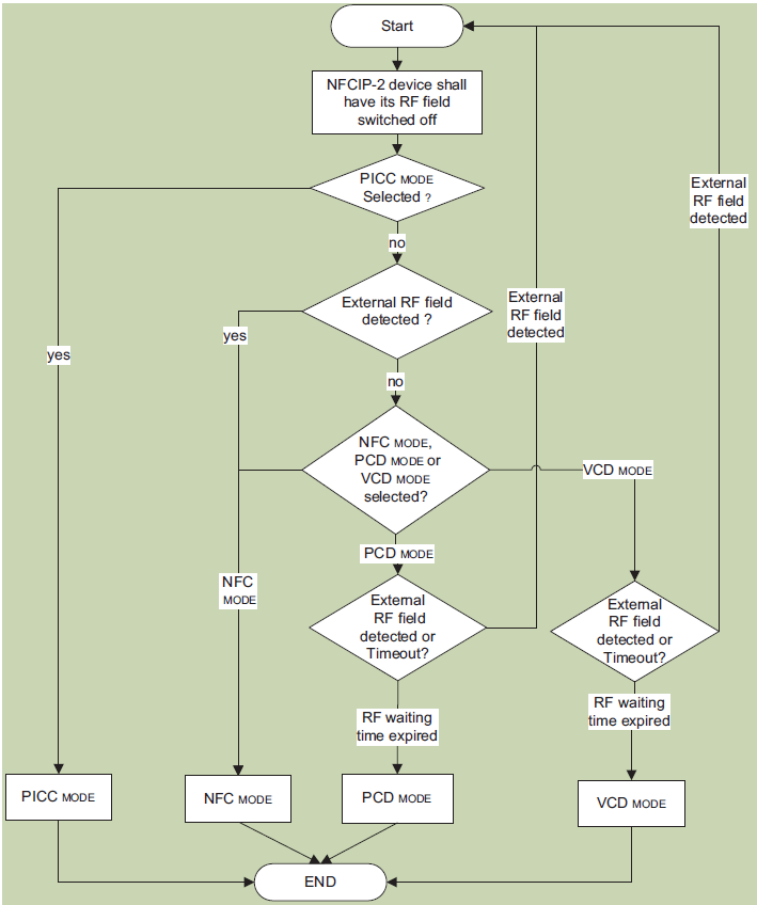

Abbildung 3.14: Aufbau des NFCIP-1 Standards [NFCIP-2, S.3]

zu einer Entfernung bis zirka 10m davon ausgegangen werden, dass Daten eines aktiven (d.h., das HF-Feld wird von dem Gerät betrieben, das gerade sendet) Senders empfangen werden können. Überträgt ein passives Gerät mit dem HF-Feld des Initiators die Daten, kann bis zu zirka 1m die Kommunikation abgehört werden. Diese Angaben sollen eine grobe Vorstellung über die möglichen Distanzen geben, da die maximale Reichweite sehr stark von den verwendeten Komponenten und den Umgebungsbedingungen abhängig ist.

NFC bietet gegen diese und andere Attacken mit dem NFC-SEC (siehe [NFCIP-3], [NFCIP-4], [NFCIP-5], [NFCIP-6] und [NFCIP-7]) sowohl kryptographische Verfahren für die Verschlüsselung der Datenpakete als auch Mechanismen gegen eine Datenmanipulation. Dazu wird die sichere Schlüsselübergabe, im NFC-SEC als SSE (**S**hared **SE**cret Service) bezeichnet, durchgeführt. Die Übergabe kann auf 2 Arten durchgeführt werden mit einer symmetrischen Verschlüsselungsprozedur, und mit einer asymmetrischen. Bei der symmetrischen Prozedur muss ein Authentifizierungsschlüssel, PSAK (**P**re-**S**hared **A**uthentication **K**ey), vor Beginn beiden Teilnehmern bekannt sein. Mit diesem Schlüssel

wird die Prozedur verschlüsselt. Bei der asymmetrischen Prozedur sind beide Teilnehmer im Besitz eines Zertifikats, eines eigenen privaten und öffentlichen Schlüssels, die mit dem ECDSA (**E**lliptic **C**urve **D**igital **S**ignature **A**lgorithm) erzeugt wurden, sowie eines Zertifikats eines TTPs (**T**usted **T**hird **P**arty). Der Ablauf einer Variante ist zum besseren Verständnis in Abbildung 3.15 dargestellt.

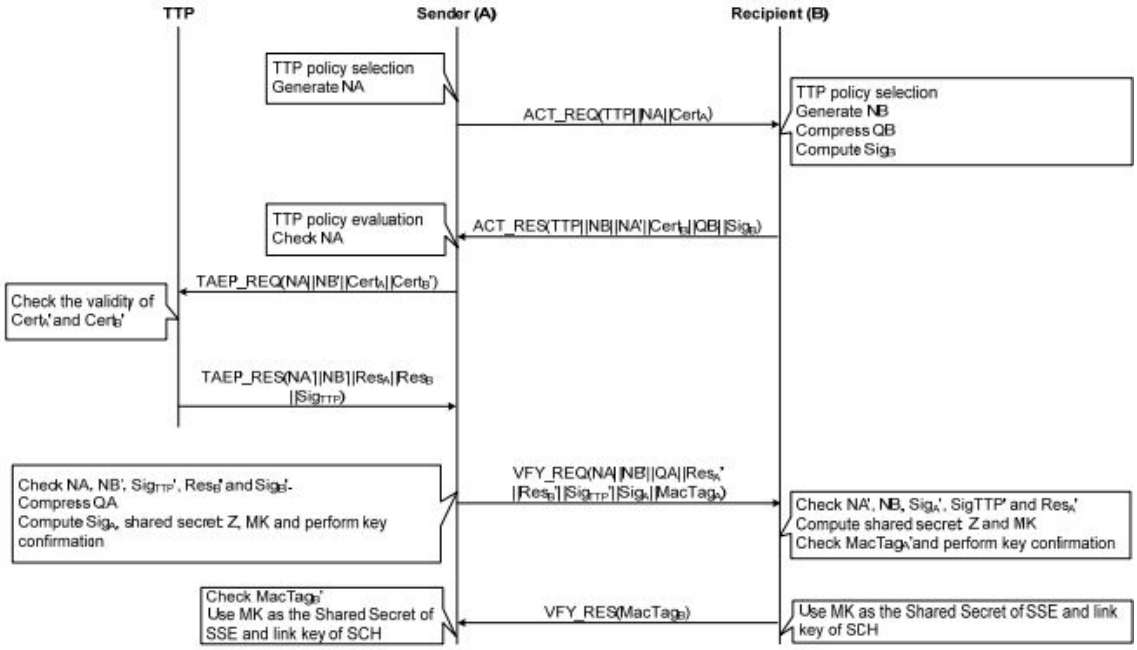

Abbildung 3.15: Asymmetrische Schlüsselgenerierung [NFCIP-5, S.13]

Die eigenen Zertifikate ($Cert_A$, $Cert_B$), Noncen (NA, NB) - diese werden zur Gewährleistung der Datenaktualität verwendet - sowie der öffentliche Schlüssel (QB) und die Signatur (Sig_B) des Empfängers werden ausgetauscht. Danach werden 2 Fälle unterschieden, wobei hier der Wert von TTP entscheidend ist. Im ersten Fall werden die übertragenen Zertifikate von einer Authentifizierungsstelle auf ihre Gültigkeit (Res_A, Res_B) überprüft. Dann kann ein Schlüssel erzeugt werden. Danach werden zur Verifizierung unter anderem Werte ($MacTag_A$, $MacTag_B$) mit dem AES-CMAC-Algorithmus erzeugt, übertragen und überprüft. Nach dieser Prozedur verfügen beide Teilnehmer über einen gemeinsamen Schlüssel. Im zweiten Fall findet die Überprüfung durch das TTP nicht statt, wodurch in der Verfikationsphase die Signatur des TTP und die Ergebnisse der Gültigkeitsüberprüfungen nicht in der Nachricht enthalten sind. Die Verschlüsselung des nachfolgenden Datenverkehrs, im Standard als SCH (**S**ecure **CH**annel Service) bezeichnet, wird mit AES durchgeführt, wobei eine Schlüssellänge von 128-Bit verwendet wird. Gegen eine Datenmanipulation während einer verschlüsselten Kommunikation werden MACs (**M**essage **A**uthentication **C**odes) eingesetzt.

Bei der Nutzung von NFC im Bereich der Industrieautomation stehen aufgrund der

geringen Reichweite weniger Anwendungsmöglichkeiten als bei den anderen beschriebenen Protokollen zur Verfügung. Die interessantesten Anwendungsfälle sind hier:

- Zutrittskontrollsysteme: Dies ist dann sinnvoll, wenn die Anlage entweder in einem geschlossenen Gebäude untergebracht ist, oder zumindest in einem umzäunten Gelände.
- OOB: Als OOB-Verfahren eignet sich NFC aufgrund der sicheren Datenübertragungsverfahren. Außerdem ist das Abhören der gesendeten Daten durch die geringe Reichweite nur schwer möglich.
- Für die Datenübertragung geringer Datenmengen, wie z.B. Diagnosenachrichten ist NFC auch interessant, wenn die auszulesenden Geräte batteriebetrieben sind, da hier der *Reader/Writer*-Modus zur Anwendung gelangen kann, und dadurch die Lebensdauer des Gerätes nicht beeinträchtigt wird.
- Handelt es sich um besonders sicherheitskritische Daten, die von einem Gerät ausgelesen werden müssen, so sollte die Authentifizierungsprozedur über Zertifikate einer *TTPs* zusätzlich gesichert werden. Nachdem die *TTP* in größeren Anlagen wahrscheinlich eine zentrale Instanz darstellt, müssen die nötigen Informationen über andere Protokolle mit einer größeren Reichweite beschafft werden. Dies kann im Fall eines Gerätes mit kabelgebundener Netzwerkverbindung über diese geschehen, bei einem Smartphone könnte hier beispielsweise auf WLAN zurückgegriffen werden.

3.4 ZigBee

Die Informationen für dieses Unterkapitel wurden unter anderem aus [WPAN03] und [ZIGBEESTD] zusammengetragen. Der ZigBee-Standard wurde 2004 von der ZigBee Alliance entwickelt. Die aktuelle Version wurde 2012 veröffentlicht. Sie basiert auf dem IEEE Standard 802.15.4-2003, der die untersten beiden Schichten (PHY und MAC) beschreibt. ZigBee wurde mit einem starken Fokus auf geringen Energieverbrauch entwickelt, um einen Betrieb im Feld zu ermöglichen, der keine externe Stromversorgung benötigt. Außerdem soll ein robuster Datentransfer, sowie ein kostengünstiger Aufbau ermöglicht werden. Im Vergleich zu WLAN und Bluetooth arbeitet ZigBee mit einer weit geringeren Datenübertragungsrate in 3 Frequenzbändern,

- von 868-868,6 MHz mit einer Übertragungsrate von 20 Kilobit/s,
- von 902-928 MHz mit 40 Kilobit/s und
- von 2400-2483,5MHz mit 250 Kilobit/s.

In diesem Kapitel soll das ZigBee-Protokoll sowie Teile des zugrundeliegenden 802.15.4 Standards kurz beschrieben werden. Abbildung 3.16 zeigt den Protokollaufbau.

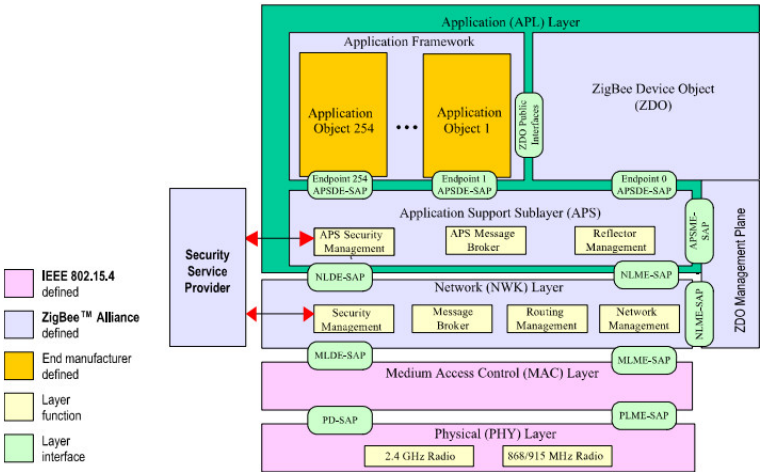

Abbildung 3.16: Schichtenstruktur von ZigBee [ZIGBEESTD, S.2]

Die beiden untersten Schichten (PHY und MAC) werden durch den 802.15.4 Standard definiert. ZigBee definiert die Netzwerk-(NWK)- und Applikations-(APL)-Schicht.

3.4.1 802.15.4

Der 802.15.4 Standard kennt zwei Arten von Geräten, nämlich RFDs (**R**educed **F**unction **D**evices) und FFDs (**F**ull **F**unction **D**evices). Ein FFD muss dabei den gesamten Protokollstack implementieren, ein RFD einige wenige Funktionen. Das führt unter anderem dazu, dass ein RFD nur mit einem FFD kommunizieren kann, nicht jedoch mit einem anderen RFD. Ein FFD kann die Rolle des Masters im Netzwerk - der Standard bezeichnet dieses Gerät als PAN-Koordinator (**P**ersonal **A**rea **N**etwork) - übernehmen. Es ist verantwortlich für das Einbinden neuer Geräte, die Synchronisation, und darüber hinaus für andere Managementaufgaben. Ein Netzwerk kann mehrere PAN-Koordinatoren beinhalten, es gibt aber immer einen ausgezeichneten PAN-Koordinator, der den Kopf des Netzwerks darstellt. Ein RFD hingegen ist immer ein Endpunkt eines Netzwerks.

Topologisch betrachtet bietet der Standard 2 Möglichkeiten der Vernetzung, die Sterntopologie und die Peer-to-Peer-Topologie. Die Peer-to-Peer-Topologie ermöglicht eine beliebige Verbindung zwischen FFDs und RFDs. Als Beispiel zur Vernetzung beschreibt der Standard ein Cluster Tree Network (siehe Abb. 3.17). Dabei bildet ein FFD als PAN-Koordinator den ersten Cluster - der Standard bezeichnet diesen Koordinator als CLH (**CL**uster **H**ead) mit der ID 0 - andere Geräte können sich verbinden. Außerdem besteht die Möglichkeit andere Geräte als CLHs auszuzeichnen. RFDs können lediglich als Blattknoten fungieren. Ebenfalls möglich sind vermaschte Netze, wobei der Standard hier keine Aussage darüber trifft, wie ein Routing, also die Kommunikation von zwei Geräten, die nicht direkt verbunden sind, durchgeführt wird.

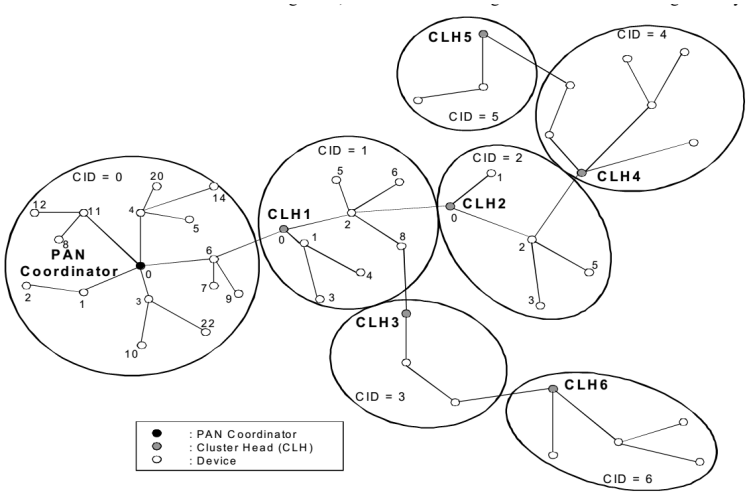

Abbildung 3.17: Aufbau eines Cluster Tree Networks [WPAN03, S.15]

Die physikalische Datenübertragung kann grundsätzlich auf zwei Arten erfolgen, einerseits mit einem *slotted CSMA-CA*-Mechanismus (**C**arrier **S**ense **M**ultiple **A**ccess - **C**ollision **A**voidance) bei der Verwendung von *Beacons* und der *Superframe-Structure*, andererseits mit dem *CSMA-CA*-Mechanismus. Dabei wird die gewünschte Frequenz abgehört, ob ein anderer Teilnehmer gerade sendet. Ist keine Datenübertragung im Gange, so kann das Gerät nach einer bestimmten Wartezeit selbst zu senden beginnen. Ein *Beacon* ist eine Nachricht, die Informationen zur Synchronisation, sowie zur Beschaffenheit des Superframes enthält. Der Hauptunterschied zwischen den beiden Mechanismen besteht darin, dass der slotted CSMA-CA-Mechanismus nur an den Slotgrenzen aktiv wird, um die Frequenz auf ein Vorhandensein eines anderen Senders zu überprüfen. Die Voraussetzung für den slotted CSMA-CA-Mechanismus ist die *Superframe-Structure*, die in Abb. 3.18 zu sehen ist. Der Superframe kann in zwei Teile aufgeteilt werden, einen aktiven und einen inaktiven. Hierbei soll der Koordinator nur im aktiven Teil kommunizieren. Den Rest der Zeit bis zum nächsten Beacon soll er nicht mit dem Netzwerk interagieren (dies dient auch zum Sparen von Energie). Das aktive Intervall zwischen zwei vom PAN-Koordinator gesendeten Beacons wird in 16 gleichlange Slots aufgeteilt. Es kann wiederum in zwei Intervalle aufgeteilt werden, die CAP (**C**ontention **A**ctive **P**eriod) und die CFP (**C**ontention **F**ree **P**eriod). Während der CAP können alle verbundenen Geräte Daten senden und empfangen, wobei der slotted CSMA-CA-Mechanismus angewandt werden muss. In der CFP kann der Koordinator bis zu sieben Slots als GTSs (**G**uaranteed **T**ime **S**lots) vergeben. Ein GTS kann dabei auch mehrere Slots der zuvor erwähnten 7 umfassen. Diese Zeitslots sind dann für einzelne Geräte oder Anwendungen reserviert (der slotted CSMA-CA-Mechanismus findet hier keine Anwendung). Sie dienen ausschließlich der Kommunikation zwischen dem PAN-Koordinator und einem Gerät. Diese Slots sind nötig, falls Anwendungen spezielle Anforderungen bzgl. der notwendigen Bandbreite

stellen, oder gewisse Latenzzeiten nicht überschritten werden dürfen.

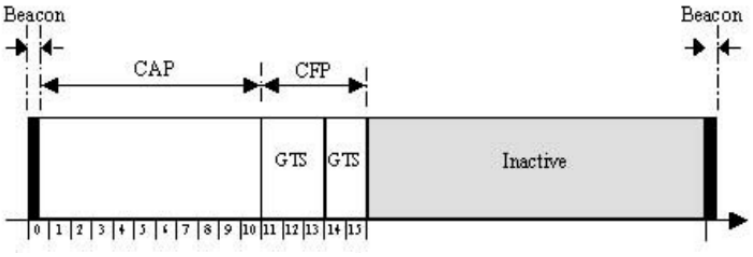

Abbildung 3.18: Aufbau einer Superframe Structure [WPAN03, S.141]

Weiters bietet der Standard verschiedene Kommandos und Sicherheitsmerkmale, welche hier nicht gesondert behandelt werden. Die Sicherheitsmerkmale im Besonderen werden in Kombination mit ZigBee nicht verwendet, da hierfür eine eigene Spezifikation im ZigBee-Standard existiert.

3.4.2 ZigBee

Der ZigBee-Standard definiert die Schichten über der MAC-Schicht. Dabei werden die Netzwerk- und die Applikationsschicht definiert, sowie die Schnittstellen zwischen den genannten. Er kennt 3 Arten von Geräten:

- ZigBee Coordinator: Ein PAN-Koordinator, der das Netzwerk ursprünglich gestartet hat.
- ZigBee Router: Ein FFD, das nicht der ZigBee Coordinator ist, aber innerhalb seines Wirkungskreises Aufgaben eines PAN-Coordinators übernehmen kann, wie z.B. das Weiterleiten von Nachrichten.
- ZigBee End Device: ein RFD oder FFD, das keine Coordinatorfunktionen übernimmt.

Die wichtigsten Funktionen der NWK-Schicht sind das Verteilen (Routen) von Paketen im Netzwerk, Netzwerkfunktionen wie das Starten, Beitreten, Verlassen eines Netzwerks, das Auffinden anderer Geräte, und die richtige Weitergabe von Paketen an die höheren Schichten. Weiters muss die NWK-Schicht auch die Authentizität sowie die Vertraulichkeit von Paketen gewährleisten.

Die Applikationsschicht beinhaltet den APS (**AP**plication **S**upport Sublayer), die ZDOs (**Z**igBee **D**evice **O**bjects), sowie von Geräteherstellern erstellte Applikationsobjekte. Die APS-Schicht ist verantwortlich für das Verteilen von Paketen an zwei oder mehr Applikationsobjekte - hier können auch die ZDOs als Applikationsobjekte gesehen werden - innerhalb des Netzwerks. Insgesamt kann ein ZigBee-Gerät bis zu 254 Applikationsobjekte verwalten. Diese werden über *Endpoints* adressiert. Die Adresse 0 ist für die ZDOs

reserviert, und 255 wird für Broadcast Pakete verwendet, die an alle Applikationsobjekte weitergeleitet werden sollen.

Die ZDOs stellen grundlegende Funktionen zur Verfügung, wie die Initialisierung und Steuerung der darunterliegenden Schichten. Dazu zählen der APS, die NWK-Schicht, sowie der SSP (**S**ecurity **S**ervice **P**rovider), der für die Sicherung der Pakete verantwortlich ist. Den anderen Applikationen bieten die ZDOs Funktionen zum Auffinden von Geräten im Netzwerk und deren angebotenen Services an. Weiters stellen die ZDOs den Applikationsobjekten Informationen über das Netzwerk sowie optionale Services zur Verfügung. Zu diesen Services zählen unter Anderem die Sicherung von Paketen, das Binden von Applikationsobjekten an andere Applikationsobjekte (auch auf entfernten Geräten), sowie die Erstellung von Gruppen innerhalb der Applikationsobjekte eines Gerätes. Ob ein Service optional ist, hängt auch damit zusammen, ob ein Gerät ein ZigBee Coordinator, Router, oder End Device ist. Beispielsweise muss ein ZigBee Coordinator die Möglichkeit bieten, dass andere Geräte sich im Netzwerk anmelden können, ein End-Device benötigt keine derartige Funktion.

Topologisch betrachtet besitzt ZigBee aufgrund der Verwendung des 802.15.4 Standards die Möglichkeiten, der Stern-, sowie der Peer-to-Peer-Topologie. Auf diese aufbauend können vermaschte Netzwerke aufgebaut werden. Weiters können Geräte Datenübertragungen über Unicast-, Multicast- und Broadcastverbindungen realisieren.

3.4.3 Sicherheitsmerkmale

Der ZigBee-Standard definiert eine Sicherheitsarchitektur, die für die Bedürfnisse des Protokolls optimal sein soll. Dabei werden gewissen Annahmen getroffen, die einerseits für eine ordnungsgemäße Sicherheit vorausgesetzt werden, wie z.B. die standardkonforme Implementation des Protokolls, und andererseits die Grenzen der Sicherheit beschreiben, wie z.B. das Fehlen von manipulationssicherer Hardware. Diese Grenzen sind jedoch nicht auf den Standard zurückzuführen, sondern eher auf die Anforderung nach günstigen und wenig energieintensiven Komponenten. Natürlich kann auf Sicherheitsvorkehrungen über Einstellungen verzichtet werden, wobei dies jedoch in den meisten Fällen nicht ratsam ist.

Weiters werden verschiedene Prinzipien vorgestellt, welche die Sicherheit erhöhen sollen. Grundsätzlich ist jede Komponente bzw. Schicht selbst dafür verantwortlich, ihre Daten zu schützen, wenn dies notwendig ist. Beschrieben werden Sicherheitsaspekte der Netzwerk-, der APS- und der MAC-Schicht, wobei die Sicherungsmechanismen der MAC-Schicht im 802.15.4-Standard beschrieben ist. Zusätzlich zur Verschlüsselung wird auch ein MIC (**M**essage **I**ntegrity **C**ode) am Ende des Pakets hinzugefügt, der eine nachträgliche Veränderung der Nachricht erkennbar machen soll, und damit die Authentizität der Nachricht gewährleistet.

Die Kommunikation zwischen Geräten soll immer verschlüsselt stattfinden, außer die Kommunikation erfolgt mit einem neu hinzugefügten Gerät. Darüber hinaus soll die Kommunikation Ende-zu-Ende verschlüsselt werden (*link keys*), um zu verhindern, dass andere Geräte im selben Netzwerk den Datenverkehr entschlüsseln können. Dieselben Schlüssel können in verschiedenen Schichten wiederverwendet werden - im Standard als *Open Trust Model* bezeichnet -, was zu einer verminderten Ressourcenanforderung im Bereich des Speichers führt. Nachdem ZigBee die Möglichkeit bietet, mehrere Sicherheitsstufen zu verwenden, und um die Interoperabilität zu erleichtern, sollen Geräte, die unterschiedliche Sicherheitsbedürfnisse haben, Netzwerke bilden, in denen einheitlich dieselbe Sicherheitsstufe verwendet wird.

Die Verschlüsselung wird, wenn aktiviert, grundsätzlich mit dem AES-CCM-Verfahren mit 128-Bit Schlüsseln durchgeführt, das außerdem gewährleistet, dass die Daten während der Übertragung nicht verändert wurden (Integrität), und der Absender jener ist, der die Daten gesendet hat (Authentizität). Um Replay-Attacken zu erschweren, enthalten alle gesendeten Pakete außerdem fortlaufende Nummer, die die Datenaktualität sicherstellen sollen. Bei Empfang eines Pakets mit einer ungültigen Nummer, soll das Paket verworfen werden. Um die Wahrscheinlichkeit eines erfolgreichen Angriffs auf die Datenübertragungen zu senken, werden für unterschiedliche Datenübertragungen unterschiedliche Schlüssel benutzt:

- Network-Keys: Netzwerkschlüssel werden für Übertragungen genutzt, die alle Geräte betrifft (Broadcast)
- Master-Keys: *Master-Keys* werden zur Erzeugung von *Link-Keys* verwendet.
- Link-Keys: Um die Kommunikation zwischen genau 2 Geräten zu sichern, werden *Link-Keys* verwendet. Weiters kann es sich bei einem *Link-Key* eines *Trust Centers* um einen *Global Link-Key* oder einen *Unique Link-Key* handeln. Dieser Typ beeinflusst die Behandlung von empfangenen bzw. zu sendenden Kommandos hinsichtlich der Notwendigkeit für eine verschlüsselte Übertragung.

Um Schlüssel zu verteilen, stehen für Komponenten eines Netzwerks verschiedene Möglichkeiten bzw. Kommandos zur Verfügung. Eine wichtige Rolle in diesem Zusammenhang kommt dem sogenannten *Trust Center* zu. Es wird unter Anderem für die Verteilung und Speicherung von Schlüsseln innerhalb des Netzwerks benötigt. In einem Netzwerk existiert immer genau ein *Trust Center*, und es muss allen Geräten im Netzwerk bekannt sein. Die Anforderungen an das *Trust Center* sind auch unterschiedlich, je nachdem, welches Sicherheitsniveau im Netzwerk gewählt wird. Dafür stehen bei ZigBee zwei zur Auswahl, der *Standard Security Mode* - auch *Residential Mode* genannt, und der *High Security Mode* - auch *Commercial Mode* genannt. Im *Standard Security Mode* kann das *Trust Center* optional eine Liste aller Schlüssel, die im Netzwerk verwendet werden, speichern, im *High Security Mode* ist das Speichern der Liste verpflichtend. Einen Standard Network-Key

muss das *Trust Center* jedoch in jedem Fall speichern. Muss ein Schlüssel über das Netzwerk übertragen werden, stehen dafür verschiedene Services zu Verfügung.

Das Key-Establishment-Service bietet die Möglichkeit, für zwei Geräte einen Link-Key für die Kommunikation auszutauschen. Dabei müssen anfangs Informationen, wie z.B. ein Master-Key ausgetauscht werden - im Standard auch *Trust Information* genannt -, wobei diese entweder mit ZigBee-Paketen übertragen werden können (in-band), oder über andere Wege (out-of-band), z.B. händisch vor der Installation. Dann wird über das SKKE-(**S**ymmetric-**K**ey **K**ey **E**stablishment) Protokoll der gemeinsame Schlüssel bestimmt. Dabei senden die zwei Geräte spezielle Nachrichten, aufgrund derer am Ende nicht nur der gemeinsame Schlüssel feststeht, sondern zusätzlich eine Authentifizierung der Geräte auch untereinander. Das SKKE-Protokoll weist jedoch, unter gewissen Annahmen, Schwächen auf, wie in [EY10] beschrieben. Dabei besteht das Problem, dass den Protokollnachrichten keine Informationen hinzugefügt werden, die sicherstellen, dass die Pakete aktuell sind, und damit wenig Sicherheit gegen eine Replay-Attacke bieten. Jedoch funktioniert ein erfolgreicher Angriff nur dann, wenn der *Master-Key* eines Gerätepaares nie verändert wird. Ebenfalls wird kritisiert, dass zur Überprüfung der Aktualität der Daten fortlaufende Nummern verwendet werden und keine Nonces oder Zeitstempel. Dadurch besteht die Möglichkeit, dass ein Angreifer über fingierte Nachrichten erzwingen kann, dass Nachrichten verworfen werden, indem er die Zähler in einem Gerät manipuliert, oder zum Überlauf bringt.

Das Transport-Key-Service wird verwendet, um alle zuvor genannten Schlüsseltypen von einem *Trust Center* zu einem anderen Gerät zu senden. Dabei kann die Übertragung verschlüsselt oder unverschlüsselt geschehen. Bei einer unverschlüsselten Übertragung wird darauf hingewiesen, dass die Daten auch über Kommunikationswege (OOB) außerhalb des ZigBee-Protokolls übertragen werden können, um höhere Sicherheit zu erreichen.

Über das Request-Key-Service kann von einem Gerät ein bestehender Master-, Link- oder Network-Key verschlüsselt angefordert werden.

Der Switch-Key-Service wird zwar nicht direkt zu Schlüsselübertragungen verwendet, ist jedoch für die Schlüsselkonfiguration interessant. Mit diesem Service werden Geräten in einem Netzwerk dazu veranlasst, einen anderen Network-Key zu verwenden. Vor diesem Kommando muss über das Transport-Key-Service jedoch ein neuer Network-Key übertragen worden sein, für den Fall, dass Empfänger diesen Schlüssel noch nicht besitzen.

Bevor ein neu hinzugefügtes Gerät in einem Netzwerk Daten in das Netzwerk übertragen kann, muss es authentifiziert werden. Bei einem nicht authentifizierten Gerät darf der Router, an den das Gerät angeschlossen ist, keine gesendeten Pakete des Gerätes in das Netzwerk übertragen. Die Vorgehensweise bei der Authentifizierung hängt

einerseits vom verwendeten Sicherheitsmodus - *Standard Security Mode* oder *High Security Mode* - ab, andererseits davon, ob dem Gerät der Network-Key bekannt ist oder nicht.

Im *Standard Security Mode* wird dem Gerät mit dem Transport-Key-Service ein Network-Key gesendet. War dem Gerät der Schlüssel im Vorfeld bekannt, so wird in der Nachricht der 128-Bit Wert für das Passwort auf Null gesetzt. Sonst wird der Schlüssel als Klartext mit dem Kommando übertragen.

Im *High Security Mode* hängt die Vorgehensweise ebenfalls davon ab, wie das neu hinzugefügte Gerät konfiguriert ist. Ist dem Gerät weder ein Trust-Center-Link-Key oder ein Trust-Center-Master-Key, noch ein Network-Key bekannt, so wird mit dem Transport-Key-Service ein Trust-Center-Link-Key oder ein Trust-Center-Master-Key unverschlüsselt gesendet. Wurde ein Master-Key gesendet, so wird dem Key-Establish-Service und dem darauffolgenden SKKE-Protokoll ein Link-Key generiert. Daraufhin wird ein mit dem Link-Key verschlüsselter Network-Key übertragen. Im letzten Schritt wird mit einem Challenge-Respone-Verfahren die Authentifizierung abgeschlossen. Dieser Schritt wird nur benötigt, wenn das Gerät zu Beginn über keinen Schlüssel verfügt hat. Ist ein Trust-Center-Link-Key vorhanden, so kann der Network-Key sofort gesichert übertragen werden. Bei Vorhandensein eines Network-Keys wird ebenso wie im *Standard Security Mode* vorgegangen.

Nach diesen Schritten gilt ein Gerät als authentifiziert.

Ein weiterer Kritikpunkt an der Sicherheit von ZigBee ist in [TZ15] beschrieben. ZigBee gibt vor, dass einem Gerät, dem kein spezieller *Link-Key* vorinstalliert wurde, ein allgemein bekannter Standard Link-Key zur Verfügung stehen muss, nämlich

„5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39“.

Dies dient, insbesondere bei Verwendung von Geräten unterschiedlicher Hersteller, einer einfacheren Kompatibilität unter den Geräten, stellt jedoch auch ein Sicherheitsrisiko dar.

Für die Nutzung in der Industrieautomation ist ZigBee in vielfacher Weise geeignet. Um eine sichere Verwendung von ZigBee gewährleisten zu können, sollten folgende Punkte beachtet werden:

- Es sollte ein Sicherheitsmodus verwendet werden, bei dem Verschlüsselung aktiviert ist (*High Security Mode*).

- Es sollten, wo möglich, Schlüssel schon vor Installation von Geräten in einem Netzwerk installiert werden.
- Schlüssel sollten regelmäßig erneuert werden.

Durch die verschiedenen Sicherheitsmerkmale und die energieeffizienten Übertragungsmöglichkeit, eignet sich ZigBee nicht nur für den Einsatz in Industriehallen, sondern kann auch abseits von geschlossenen Anlagen verwendet werden, um Überwachungs- und Steuerungsaufgaben zu übernehmen. Gerade durch die Möglichkeit der vermaschten Topologie bieten sich hier auch Aufgaben in weit verzweigten Netzwerken an, wobei auch bei Ausfall einzelner Geräte durch die Selbstheilungseigenschaften dieser Topologie ein weiterer Betrieb möglich ist. Einziger Nachteil dieses Protokolls ist die geringe Datenübertragungsrate, wodurch beispielsweise Videoübertragungen bei der optischen Qualitätssicherung oder Firmwareaktualisierungen problematisch sein können. Auch bei der Übertragung zeitkritischer Prozessdaten kann ZigBee verwendet werden, wenn die *Superframe-Structure* in Verbindung mit *GTSs* genutzt wird.

3.5 Anforderungsanalyse der Protokolle

Aufgrund der Spezifikationen der beschriebenen Protokolle, sollen in diesem Abschnitt deren Verwendungsbandbreite anhand der im Kapitel 2 beschriebenen Anforderungen für die Aufgaben in der Industrieautomation analysiert werden. Hierbei sollen im ersten Schritt, gegliedert nach den Unteranwendungsfällen - die Hauptanwendungsfälle verwenden teilweise die gleichen Unteranwendungsfälle -, die einzelnen Protokolle betrachtet werden. Bezüglich der Sicherheitsanforderungen bieten alle Protokolle bei richtiger Konfiguration einen ähnlich guten Schutz, daher spielen sie bei der Bewertung nur eine untergeordnete Rolle.

3.5.1 Zutrittskontrolle

Bei Zutrittskontrollsystemen soll berechtigten Mitarbeiterinnen und Mitarbeitern einfach und so komfortabel wie möglich Zugang zu gesicherten Bereichen eines Unternehmens gewährt werden. Wichtig hierbei ist eine geringe Reichweite einer Datenübertragung, damit Befugte, die sich in der Nähe des Authorisierungsgerätes aufhalten, nicht versehentlich Zugänge für Unbefugte öffnen. Sinnvoll ist hier eine maximale Entfernung von wenigen Zentimetern. Aufgrund der geringen Reichweite von NFC, ist diese Technologie hier zu bevorzugen. Nachdem Zutrittsberechtigungen meist zentral und in nicht öffentlich zugänglichen Bereichen verwaltet werden, können Schlüssel und damit einhergehende Berechtigungsinformationen aus Sicht des Protokolls sicher vergeben werden. Eine Kompromittierung der Schlüsselerzeugungsprozedur ist damit nahezu ausgeschlossen. Weiters werden keine hohen Datenübertragungsraten zur Übertragung von Berechtigungsschlüsseln - allzu viele Informationen sind hier nicht sinnvoll - benötigt, weshalb die niedrige maximale Datenübertragungsleistung kein Problem darstellt. Als weitere Anforderung ist die Notwendigkeit einer Einwegkommunikation zu sehen, da im Normalfall für genau

eine Person festgestellt werden muss, ob eine Berechtigung besteht. Es existieren auf dem Markt auch Zutrittskontrollsysteme, die mit Bluetooth arbeiten, jedoch ist die maximal mögliche Distanz zwischen Sende und Empfangsanlage mit mehreren Metern zu groß. Für den Privatgebrauch wie in Einfamilienhäusern beispielsweise, bei dem sich keine unbefugten größeren Menschengruppen in der Nähe der Türe aufhalten, ist dies möglicherweise akzeptabel. In Industrieanlagen besteht dadurch aber ein erhöhtes Sicherheitsrisiko.

3.5.2 Netzwerkmanagement

Für die Erfassung der Geräte eines Netzwerks (Netzwerkscan) ist es nötig, dass auch räumlich größere Gebiete abgedeckt werden können. Weiters kann es wichtig sein, auch bei Ausfall einzelner Knotenpunkte dennoch alle fehlerfrei arbeitenden Geräte erreichen zu können. Dazu bedarf es entweder einer großen Reichweite des Übertragungsprotokolls oder der Verfügbarkeit von Mechanismen, die ein Weiterleiten von Paketen über andere Geräte ermöglicht (Routing). Weiters soll die Möglichkeit zur Übermittlung von Broadcast-Nachrichten gegeben sein, um die für einen Scan benötigte Zeit zu verkürzen. Für diese Aufgaben kommen alle beschriebenen Protokolle außer NFC - aufgrund der Reichweite - in Frage. Die Verwendung von Bluetooth bietet sich eher bei kleineren Netzwerken an, da hier keine Möglichkeit der Weiterleitung von Paketen über mehrere Geräte vorgesehen ist. Zusätzlich ist die Menge der Teilnehmer stärker begrenzt. Bei Anwendungsgebieten, wie z.B. Industriehallen, in denen eine großflächige Stromversorgung möglich ist, bietet sich WLAN an. Es lässt sich beliebig erweitern und bietet eine hohe Datenübertragungsrate. Zudem bietet es die Möglichkeit, vermaschte Netzwerke aufzubauen, wodurch auch bei Ausfall einzelner Geräte Daten über andere Wege übertragen werden können. Bei Betrieb im ESS-Modus ist dies nicht gewährleistet, da ein Access Point einen *Single Point of Failure* dargestellt, und bei einem Ausfall ganze Netzwerkteile nicht mehr erreichbar sind. Außerdem ist die Anbindung an bestehende kabelgebundene Netzwerke (wie z.B. Ethernet) sehr einfach möglich. Für große Netzwerke, bei denen kein durchgehender Zugang zu einer Stromversorgung - die Geräte werden von Akkumulatoren oder Batterien betrieben - besteht, zeichnet sich ZigBee durch seinen geringen Stromverbrauch und die Weiterleitungsmöglichkeiten von Paketen aus. Eine Verschlüsselung von Scanvorgängen ist nicht unbedingt notwendig, wenn im Zuge dessen keine sicherheitsrelevanten Daten übertragen werden.

Für die Identifikation eines Gerätes kommen sinnvoller Weise ebenfalls alle Protokolle außer NFC in Frage, unter der Annahme, dass die Person Sichtkontakt zu den Geräten hat, und mehrere Geräte vorhanden sind, deren gleichartiges Aussehen eine visuelle Identifikation erschwert. Um zu verhindern, dass ein manipuliertes Gerät vorgibt, ein anderes Gerät zu sein, kann eine Authentizitätsprüfung sinnvoll sein.

Für die Aktualisierung der Gerätefirmware ist es notwendig, ein höheres Maß an Sicherheit zu gewährleisten. Es muss verhindert werden, dass Angreifer die Firmware bei der

Übertragung verändern (MITM-Attacke). Zusätzlich stellt die Firmware auch intellektuelles Eigentum der Herstellerfirma dar, und kann, wenn die Übertragung abgefangen wird, auch zu monetärem Schaden führen. Daher sollte nur das dafür vorgesehene Gerät die Daten verwenden können. Außerdem kann die Größe der Firmware eines komplexen Gerätes stark variieren, wodurch eine gewisse Bandbreite des Protokolls sinnvoll ist. Sollen mehrere Geräte „gleichzeitig“ aktualisiert werden, ist, neben den Bandbreitenanforderungen, eine Mehrwegkommunikation (Multicast) vorteilhaft, um den Vorgang zu beschleunigen und zu vereinfachen. Eine eindeutige Bevorzugung eines Protokolls kann hier nicht getroffen werden, da je nach Anforderung unterschiedliche Protokolle verwendet werden können. Einzig NFC scheidet hier aufgrund der Reichweite aus. Soll ein paralleler Firmwaredownload möglich sein, so ist auch die Verwendung von Bluetooth schwierig, da es im eigentlichen Sinn keine Multicastkommunikation zur Verfügung stellt. Bei dauernder Stromversorgung ist eindeutig WLAN zu bevorzugen, auch aufgrund der Datenübertragungsrate. Bei batteriebetriebener Betriebsart und größeren Netzwerken ist ZigBee die sinnvollere Wahl. Bei geringer Teilnehmeranzahl ist jedoch Bluetooth ZigBee gegenüber zu bevorzugen, da hier die Übertragungsrate höher ist.

3.5.3 Datenzugriff/Monitoring

Die Übertragung von Daten über den Diagnosezustand, sowie Sensor- und Zustandsdaten, egal, ob von dem Gerät oder auf das Gerät, sollte immer verschlüsselt stattfinden, da es sich bei diesen Daten um Betriebsgeheimnisse handeln kann. Außerdem muss bei einer Übertragung von Daten auf Geräte darauf geachtet werden, dass eine Authentizität sowie Datenaktualität gegeben ist. Hierfür kommen alle genannten Protokolle in Frage, wenn eine gute Erreichbarkeit - z.B. in einem Schaltschrank - der Geräte, mit denen ein Datenaustausch stattfinden soll, gegeben ist. Ist dies nicht der Fall, ist die Eignung zumindest von NFC fraglich. Soll eine Übertragung der Daten in die Leitstelle einer Anlage mit vielen Geräten erfolgen, gilt die beschränkte Eignung ebenfalls für Bluetooth. Sind die übertragenen Datenmengen regelmäßig größer, so ist eine Einsatzmöglichkeit von WLAN, andernfalls von ZigBee gegeben. Umgekehrt ist die Verwendung von ZigBee im Vergleich zu WLAN vorzuziehen, wenn Energieeffizienz eine höhere Priorität hat.

3.5.4 Knowledgemanagement

Die Anwendungsfälle mit Bezug auf Knowledgemanagement beschreiben zwar grundsätzlich die Anzeige von Dokumenten, die Übertragung derselben muss jedoch auch durchgeführt werden. Die Übertragung von Anleitungen und Checklisten wird voraussichtlich nicht von Geräten selbst erfolgen, sondern im Vorfeld übertragen. Dies ist auch deshalb sinnvoll, weil die Wartung und Ablage von Dokumenten zentral, auf Servern, Versionierungsprobleme minimiert. Wären Dokumente auch auf Geräten verfügbar, so kann die Verteilung problematisch sein. Hierbei bietet sich WLAN an, das in Büroräumen oft ohnehin bereits verfügbar ist.

Nach dieser Analyse kann festgestellt werden, dass die Auswahl eines Protokolls für die Verwendung in den genannten Anwendungsfällen nicht immer eindeutig getroffen werden kann. Entscheidend sind Faktoren wie die Anzahl der Geräte, die flächenmäßige Ausdehnung des Verwendungsgebiets, die zu übertragenden Datenmengen, sowie der Energieverbrauch. Abbildung 3.19 zeigt beispielhaft den graphischen Vergleich der Protokolle im Hinblick auf die räumliche Ausdehnung eines Netzwerks und die mögliche Anzahl der Geräte.

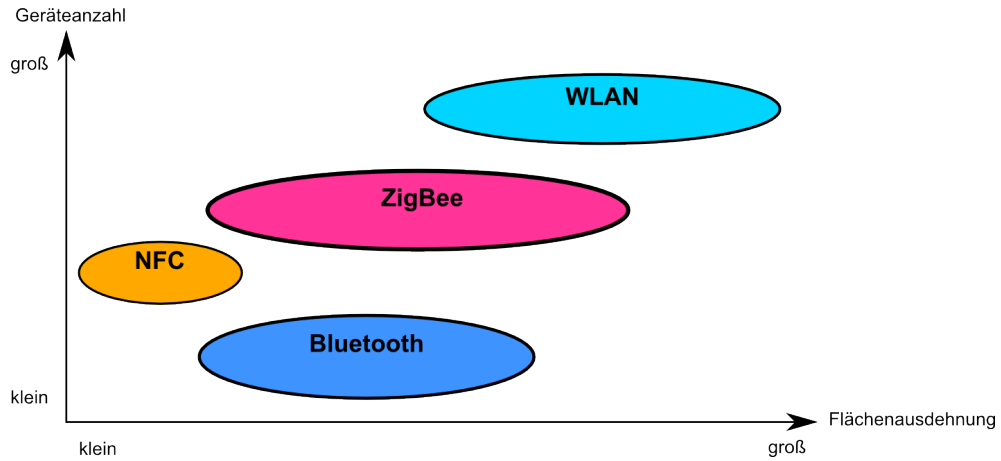

Abbildung 3.19: Vergleich der Protokolle in Bezug auf Anwendungsgebietsgröße und Geräteanzahl

Weiters soll auch ein Überblick der in Unterkapitel 2.3.1 von den Protokollen geforderten Merkmale kurz zusammengefasst werden.

Alle genannten Protokolle beherrschen Unicast-Verbindungen (Einwegkommunikation). Ebenso bieten alle Protokolle Broadcast-Verbindungen an, wobei nur WLAN und ZigBee „richtige“ Multicast-Verbindungen ermöglichen. Die Datenübertragungsraten der Protokolle sind hingegen sehr unterschiedlich. So bietet WLAN theoretisch bis zu 6,93Gbit/s, Bluetooth 3Mbit/s, NFC 424 Kbit/s, und ZigBee 250 Kbit/s. Um Fehler bei der Übertragung zu erkennen, verwenden alle Protokoll CRCs, wobei es Unterschiede bei der Länge der Prüfsummen gibt. Bei WLAN werden 32 Bit verwendet, die anderen 3 Protokolle nutzen 16 Bit, wobei Bluetooth in der Konfiguration BLE 24 Bit verwendet. Weiters wenden alle Protokolle Mechanismen zu Paketkollisionserkennung und -vermeidung an. Um die Datenaktualität bei Datenübertragungen zu gewährleisten, werden bei allen Protokollen Sequenznummern in den Paketen eingesetzt, wobei diese bei NFC nur bei gesicherten Verbindungen verwendet werden. Eine Möglichkeit zur Selbstheilung bieten nur WLAN - bei Verwendung von 802.11s - und ZigBee an, da nur diese vermaschte Topologien ermöglichen, um im Fehlerfall Pakete über andere Wege umzuleiten. Die Erweiterbarkeit im Sinne von Service-Discovery werden nur von Bluetooth und ZigBee angeboten. Um die Authentizität und Integrität von Daten zu gewährleisten, verwen-

den alle Protokolle MACs, und zwar CBC-MACs (**C**ipher **B**lock **C**haining - **M**essage **A**uthentication **C**odes), wobei WLAN bei Broadcast- und Multicast-Nachrichten einen CMAC (**C**ipher-based **M**essage **A**uthentication **C**ode) mit AES-Verschlüsselung verwendet. Für die Verschlüsselung von Übertragungen verwenden alle Protokolle AES. Für die Übertragung der dafür nötigen Schlüssel bieten alle Protokolle geeignete Verfahren an.

KAPITEL 4

Entwicklungsansätze für mobile Applikationen

In diesem Abschnitt soll die Entwicklung von mobilen Applikationen diskutiert werden. Um die Konzepte und Ideen dahinter zu verdeutlichen, wurde dieses Kapitel in folgende Sektionen unterteilt:

- Softwareentwicklung mobiler Applikationen
- Betriebssysteme

4.1 Softwareentwicklung mobiler Applikationen

Bei der Entwicklung von Softwareapplikationen ist, wie in vielen anderen Bereichen, in den letzten Jahren ein Trend zu einer immer schnelleren Einführung neuer Produkte, und, bezogen auf die Elektronik- und Softwarebranche, neuer Geräte und Software zu beobachten. Die Zeit von der Entwicklung bis zur Auslieferung (time-to-market) wird also immer kürzer. Dennoch müssen die Qualitätsmerkmale der Produkte in annähernd gleicher oder sogar höherer Güte erhalten werden. Dies erfordert Prozesse, die geeignet sind, diesen Qualitätserhalt bei gleichzeitiger kurzer Auslieferungsdauer zu gewährleisten. In [AIW10] werden einige Probleme diskutiert, die bei der professionellen Applikationsentwicklung für mobile Geräte im Vergleich zu „normalen“ Desktopprogrammen entstehen. Dazu wurde unter anderem eine Befragung von Entwicklerinnen und Entwicklern mobiler Applikationen durchgeführt, um herauszufinden, ob, und wenn ja, nach welchen Softwareentwicklungsprozessen oder Paradigmen hier vorgegangen wird. Bei dieser Befragung stellte sich heraus, dass

- die Entwicklerteams, die meist aus nicht mehr als 2 Entwicklern bestanden, eher kleine Projekte („averaging several thousand lines of code“) entwickelten.

- eine scharfe Trennung zwischen Webapplikationen und nativen Applikationen vorherrscht. Webapplikationen arbeiten nach dem Client/Server-Prinzip, die Clientanwendung wird eher klein gehalten, der Serverprozess auf dem entfernten Server übernimmt einen Großteil der arbeitsintensiven Arbeitsschritte. Native Applikationen sind genau auf das Gerät zugeschnittene Anwendungen, die alle hardwarespezifischen Funktionen ausnützen können, wie beispielsweise GPS oder Bluetooth.
- die Entwickler sich an „Best Practice“ Methoden bei der Entwicklung gehalten haben, aber selten formale Softwareentwicklungsprozesse verwendeten.
- der Fortschritt bei der Entwicklung der Applikationen nur sehr eingeschränkt dokumentiert wurde, und bezüglich des Codes selbst, wenige Qualitätsmerkmale erfasst wurden.

Aufgrund dieser Studie, der Arbeit in der diese Studie beschrieben wurde, und der persönlichen Erfahrung des Autors mit dieser Arbeit mit Apps für Smartphones sind viele native Applikationen Einzelanwendungen, die bei täglichen Aufgaben hilfreich sein können. Diese Applikationen sind in ihrer Funktionalität aber meist eingeschränkter als Desktopanwendungen, auch aufgrund engerer Handlungsspielräume innerhalb des verwendeten Betriebssystems oder der systemimmanenten geringeren Leistung der Hardware, und sind daher in ihrer Entwicklung teilweise nicht so aufwändig. Webapplikationen werden für Smartphones oft als Erweiterung von bestehenden Webseiten angeboten. Diese Applikationstypen, native und Webapplikationen, bieten oft sehr spezielle Problemlösungswerkzeuge für Einzelpersonen an. Dies kann bei der Entwicklung von größeren Projekten, auch aufgrund einer größeren Leistungsvielfalt der Applikationen zu Problemen führen. Der Grund dieser rasanten Entwicklung von Smartphone-Applikationen ist wahrscheinlich in einem der folgenden Punkte zu suchen:

- Durch die Programmierumgebungen, die für die Entwicklung mobiler Applikationen für Smartphones und Tablets verfügbar sind, ist es sehr einfach möglich, schnell Erfolge in der Erstellung eigener Anwendungen zu erzielen. Es sind zwar Grundkenntnisse in der Programmiersprache für das jeweilige System notwendig, aber aufgrund der unüberblickbaren Vielzahl an Einstiegshilfen und Tutorials im Internet, ist es möglich, innerhalb kürzester Zeit funktionale Programme mit einer ansehnlichen Benutzeroberfläche zu erstellen, und beispielsweise durch die von den Betriebssystemherstellern zur Verfügung gestellten „Appstores“ anderen Nutzern verfügbar zu machen. Dies muss nicht notwendigerweise ein Problem sein, denn es fördert das Verständnis über die grundlegenden Vorgangsweisen bei der Softwareentwicklung für Smartphones.
- Smartphones liegen momentan im Trend. Viele Firmen versuchen daher diesem Trend zu folgen, und Apps für ihre Kunden zur Verfügung zu stellen. Diese oft in Eile erstellten Applikationen funktionieren anfangs nicht richtig, was der sehr kurzen

„Time-To-Market“ geschuldet sein dürfte, und nicht unbedingt der mangelnden Erfahrung der Entwickler. Nimmt man als Beispiel das Betriebssystem Android auf Smartphones, so gibt es eine Vielzahl an Geräten von verschiedenen Herstellern, die unterschiedliche Hardware einsetzen. Für die eingesetzten Komponenten - z.B. WLAN, Bluetooth, oder GPS - müssen Treiber für das Betriebssystem erstellt oder modifiziert werden. Weiters wird von den Herstellern und Telekommunikationsanbietern oft Software auf dem Gerät vorinstalliert. Dies kann dazu führen, dass Applikationen auf dem einen Smartphonetyp problemlos funktionieren, auf dem anderen nicht. Erschwerend kommt der Umstand hinzu, dass es, wieder am Beispiel von Android, selbst durch unterschiedliche Betriebssystemversionen zu Problemen kommen kann. Auch ist ein Update auf eine neuere Version oft nicht möglich, weil die Hersteller oder Netzbetreiber die Versorgung von Geräte mit Updates des Betriebssystems nach kurzer Zeit einstellen, weil die Pflege und Wartung aufwendig und kostenintensiv ist, oder die Kundin oder der Kunde zu einem Umstieg auf ein neueres Modell mit aktueller Software gebracht werden soll.

- Funktioniert eine Applikation nicht ordnungsgemäß, so können etwa Problemberichte an den Entwickler gesendet werden. Dies führt, nachdem die gemeldeten Fehler behoben wurden, meist zu einem Update. Bei der Vielzahl an installierten Applikationen auf dem Smartphone kann es vorkommen, dass oft 10 oder mehr Updates auf einmal installiert werden müssen. Diese Updateflut könnte ein Indiz dafür sein, dass nur ungenügende Tests der Applikation vor der Veröffentlichung durchgeführt wurden. Damit wird der Nutzer eines Smartphones unfreiwillig zum Tester von, unter Umständen unter Zeitdruck schnell hergestellten, Applikationen.

- Es werden Hybrid-Web-Applikationen, welche als Erweiterung eines Internetangebotes dienen, von vielen Unternehmen angeboten. Diese werden wie normale Applikationen auf dem Gerät installiert, und zeigen Webinhalte an, die direkt aus dem Internet bezogen werden. Die Antwort auf die Frage, warum diese Inhalte nicht gleich über einen Browser aufgerufen werden, könnte einerseits daran liegen, dass Unternehmen, wie in Punkt 2 bereits angesprochen, eine eigene App als modern, und auf der Höhe der Zeit erachten. Andererseits ist es natürlich einfacher und schneller eine App zu starten, als über den Browser zu der gewünschten Seite zu navigieren. Eine App hat zudem den Vorteil, dass die Benutzeroberfläche meist ähnlich gestaltet ist, wie man es von nativen Apps des jeweiligen Systems gewöhnt ist, und Navigation und Bedienung intuitiver möglich ist. Außerdem kann die Anzeige für kleinere Bildschirmdimensionen, verglichen mit Computermonitoren, optimiert werden. Nachteilig wirkt sich jedoch der Umstand aus, dass die Flüssigkeit der Arbeit mit dieser Art von Applikationen sehr stark von der Qualität der Internetverbindung und von der Menge und Art der zu übertragenden Daten abhängt.

- Bei der Vielzahl der am Markt vertretenen Geräte und Betriebssysteme, welche alle in unterschiedlicher Weise programmiert werden müssen, benötigt man, wenn nicht

spezielle Frameworks für die Entwicklung von plattformunabhängigen Applikationen verwendet werden wollen oder können, verschiedenste Spezialisten, um zumindest die gängigsten Plattformen abdecken zu können. Dies führt natürlich auch zu einem erhöhten Kostenfaktor, weshalb die Applikationen oft aus Kostengründen oder mangels erfahrener Entwickler extern zugekauft werden. Wegen dieser Vorgangsweise können aber mögliche Synergieeffekte verloren gehen.

Um den geschilderten Problemen entgegenwirken zu können, sollen nachfolgend einige Methoden und Vorgehensweisen besprochen werden, die für die Entwicklung von Softwareprojekten vorteilhaft sind.

Generell sollte bei der Entwicklung von Software eine Methodik verwendet werden, die eine erfolgreiche Durchführung eines professionellen Softwareprojektes ermöglicht. Hierbei gibt es grundsätzlich zwei Ansätze, die verfolgt werden können, traditionelle und agile.

In [MAA05] werden diese Ansätze, bzw. ausgewählte Vertreter davon, verglichen, sowie Vor- und Nachteile beschrieben, die hier überblicksmäßig dargestellt werden sollten. Die traditionellen Ansätze zeichnen sich dadurch aus, dass Phasen eines Projektes aus einer festen Abfolge bestehen, wie zum Beispiel Anforderungsdefinition, Planung, Implementierung, Tests und Auslieferung. Es werden zu Beginn alle Anforderungen übergeben, wobei davon ausgegangen wird, dass diese nicht mehr geändert werden. Außerdem wird genau geplant, wie die Software die Anforderungen erfüllen soll. Dies führt typischerweise zu einer großen Menge an Dokumentation. Der große erforderliche Umfang an Dokumentation und Vorausplanung zu einem relativ frühen Projektzeitpunkt wird umso schwieriger, je komplexer ein Projekt wird. Softwareentwickler werden eher als austauschbare Ressourcen wahrgenommen, denn als Individuen. Es wird hier auch versucht, eher die Personen an den Prozess anzupassen, was auch der Arbeitsmotivation wenig zuträglich ist (siehe [ACJH02]). Gleichzeitig besteht hier jedoch die Möglichkeit, das Projekt in seiner Gesamtheit besser überblicken zu können.

Im Gegensatz dazu soll bei den agilen Ansätzen der hohen Wahrscheinlichkeit, dass sich Anforderungen oft sehr kurzfristig ändern können, Rechnung getragen werden. Außerdem wird die Planung eines Projekts nicht zu Beginn in der Detailtiefe vorgenommen, wie das bei traditionellen Ansätzen der Fall ist. Vielmehr geht es hier darum, dass ein kontinuierlicher Informationsfluss zwischen Kunden, dem Entwicklerteam und anderen Beteiligten stattfindet. Anstatt einer rigiden Planung am Anfang, bei der es aufgrund von Missverständnissen oder falschen Annahmen zu scheiternden Projekten oder zu einem Zeitverzug führen kann, werden die Anforderungen immer wieder nachjustiert und überarbeitet. Ein Ziel dieses Ansatzes ist es auch, dem Kunden so früh wie möglich verwendbare Software zu liefern, auch wenn diese noch nicht alle geforderten Anforderungen erfüllt. Auch die Menge der produzierten Dokumentation ist im Vergleich zu traditionellen Ansätzen niedriger. Dies ist mit der engeren Kooperation und Kommunikation der beteiligten Personen, sowie mit den dafür verwendeten Mitteln (z.B. Whiteboards und Flipcharts) zu erklären. Zusätzlich wird dem Team mehr Gestaltungsfreiheit in der Organisation gegeben,

was einerseits die Motivation erhöht, andererseits die Fähigkeiten und Kompetenzen eines jeden Teammitglieds verbessert. Dadurch kann eine bessere Einhaltung von Zeit und Kostenvorgaben erreicht, sowie die Qualität der Software erhöht werden. Weiters wird die durchschnittliche Teamgröße in agilen Projekten mit 9 Personen beschrieben, wobei auch Projekte mit bis zu 250 Personen genannt werden. Laut [CR16] kann ein Problem bei größeren Entwicklerteams darin bestehen, dass die Kommunikation nach innen (innerhalb der Teams) sowie nach außen (mit den Kunden) zu viel Zeit in Anspruch nimmt. Die Grundlage für agile Softwareentwicklung bietet das „Manifesto for Agile Software Development“ ([AGM]). Es beschreibt unter anderem 4 wichtige Prinzipien der agilen Softwareentwicklung,

- *Individuals and interactions over processes and tools*,
- *Working software over comprehensive documentation*,
- *Customer collaboration over contract negotiation*,
- *Responding to change over following a plan.*

Jedoch muss eine agile Vorgehensweise nicht immer optimal für den Erfolg eines Softwareprojekts sein. Bei größeren Projekten, an denen 50 Entwickler oder mehr an verschiedenen Orten arbeiten, der Kontakt zu Kunden und anderen Beteiligten nicht einfach möglich ist, oder sicherheitskritische Aufgaben erledigt werden müssen, kann es sinnvoll sein, keine rein agilen Prozesse zu verwenden ([LWAC03]). Zusätzlich ist die Kommunikation bei großen Gruppen schwieriger. Außerdem hängt es auch von den Beteiligten Personen ab, in wieweit neben der Verantwortung für die Erstellung der Software auch Entscheidungskompetenzen an Entwickler abgegeben werden können.

Es kann hier zusammengefasst werden, dass es nicht den einzigen richtigen Weg für die Entwicklung von Software beliebiger Art gibt. Vielmehr muss, abhängig von den Anforderungen und der Organisation abgewogen werden, wann agile oder traditionelle Methoden verwendet werden sollen, bzw. ob es nicht auch einen Mittelweg gibt. Dennoch wird in der jüngeren Literatur sichtbar, dass agile Vorgehensweisen einen immer größeren Stellenwert bei der Softwareentwicklung erhalten. Denn bei den meisten Projekten muss auch der Tatsache Rechnung getragen werden, dass die Notwendigkeit von Änderungen während der Entwicklung besteht. Außerdem werden mobile Applikationen, speziell im Industriebereich, im Moment meist als Zusatz zu den bestehenden Desktopanwendungen angeboten, weshalb davon auszugehen ist, dass keine großen Entwicklerteams daran arbeiten. Daher kann durchaus ein agiler Entwicklungsansatz als probates Mittel gesehen werden. Die am häufigsten in der Literatur genannten agilen Methoden sind dabei XP (e**X**treme **P**rogramming) und Scrum ([CR16], [RA15]), weshalb diese beiden hier kurz umrissen werden sollen.

XP wurde Ende der Neunziger-Jahre entwickelt und dann erweitert, und beschreibt Werte und Praktiken, die eine effiziente und qualitativ hochwertige Entwicklung von Software ermöglichen sollen. Die Werte von XP sind Kommunikation, Einfachheit, Feedback, Mut und Respekt. Um diese Werte verständlicher zu machen, und eine schnelle Umsetzung zu ermöglichen, wurden 24 Methoden beschrieben. Jedoch werden nicht alle Methoden dogmatisch verwendet, sondern je nach Notwendigkeit eingesetzt. Die Praktiken werden in 2 Arten unterteilt, nämlich Primäre („*Primary practices*“) und Begleitpraktiken („*Corollary practices*“). In Abbildung 4.1 sind die Methoden skizziert.

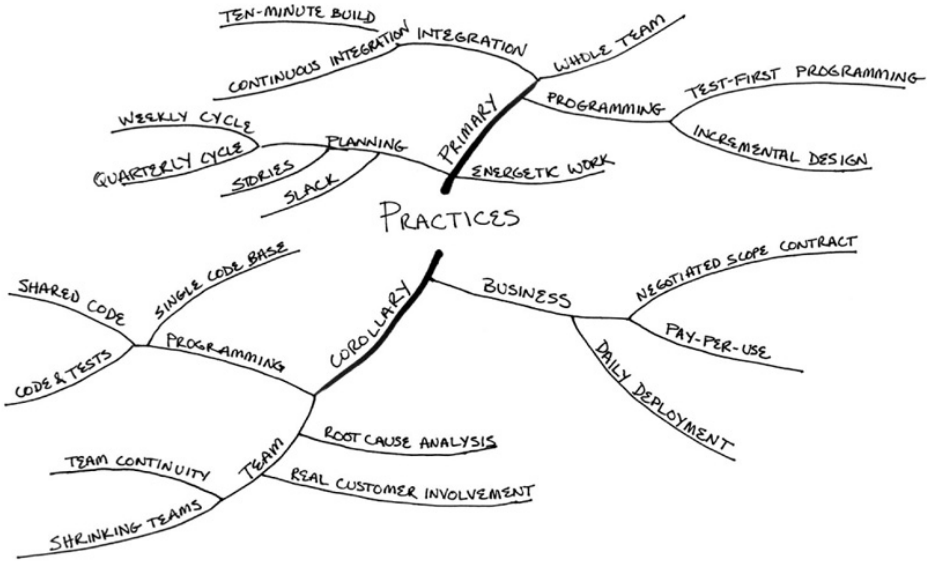

Abbildung 4.1: Methoden von XP [BAG05, S.13]

Es sollen hier nicht alle skizzierten Methoden beschrieben, sondern nur ein paar interessante Methoden umrissen werden, nämlich

- kurze Releasezyklen: Es sollte wöchentliche interne Releases geben. In diesen sollen, in Absprache mit dem Kunden, bestimmte Aufgaben erledigt werden. Diese sollen getestet sein (Unit-Tests). Außerdem sollen quartalsweise Releases erstellt werden, um den Projektfortschritt besser bestimmen und damit dokumentieren zu können.
- Continuos Integration/Ten-Minute Build: Kontinuierliche Kompilierungen des produzierten Codes machen Fehler schnell sichtbar. Außerdem soll die Kompilierung automatisiert, z.B. mit Hilfe von Buildservern, erstellt werden, um die Effizienz zu steigern.
- Incremental Design: Es ist nicht nötig, die komplette Systemarchitektur vor der Implementation zu erstellen. Vielmehr soll ein passendes Design für die aktuell zu erledigenden Aufgaben erstellt werden, und die Designentscheidungen immer wieder überdacht und angepasst werden.

- Stories: Es sollen kurze prägnante „Geschichten“ erstellt werden, welche die Funktionen beschreiben, die implementiert werden sollen.
- Pair Programming: Hierbei arbeiten immer 2 Entwickler an einem Rechner, wobei einer implementiert und der andere unterstützt. Diese Rollen sollen regelmäßig getauscht werden. Dadurch passieren weniger Fehler und der produzierte Code ist qualitativ hochwertiger. Es ist jedoch zulässig, dass auch einzeln entwickelt wird, beispielsweise um Prototypen zu erstellen.
- persönlicher Kontakt: Es ist wichtig und sinnvoll, wenn die Entwickler untereinander oft in Kontakt stehen, um produktiver arbeiten zu können, da hier ein fachlicher Austausch stattfindet, der die Fähigkeiten verbessert.

Einen genaueren Überblick bietet [BAG05].

Scrum definiert im Gegensatz dazu eher einen Prozess, ohne auf bestimmte Methoden genauer einzugehen. In [SF13] wird der Prozess genauer beschrieben. Scrum definiert 3 Rollen, den *Product Owner*, das *Entwicklungsteam* und den *Scrum Master*. Der Product Owner bestimmt dabei, welche Aufgaben die Software zu erfüllen hat. Dies geschieht über das *Backlog*, das die Anforderungen in einer bestimmten Reihenfolge enthält. Es ist über die Entwicklungszeit variabel änderbar. Der Scrum Master ist für die korrekte Durchführung des Prozesses verantwortlich. Zu dessen Aufgaben gehört unter anderem, dass das Team funktioniert, der Product Owner das richtige Verständnis über die Natur des agilen Prozesses hat, sowie die Implementierung in der Organisation funktioniert. Das Entwicklungsteam ist dabei so zusammengestellt, dass eine weitgehende Selbstorganisation stattfindet, und es sollte eine Größe von 3 bis 9 Mitgliedern umfassen. Zeitlich gesehen gliedert sich der Prozess in Sprints, das sind Intervalle von bis zu einem Monat, an dessen Ende eine neue Version der Applikation bereitstehen soll. Dieser Ablauf soll in Abb. 4.2 verdeutlicht werden.

Am Anfang steht dabei eine Planungsphase, in der die zu erfüllenden Aufgaben aus dem Product Backlog in das *Sprint Backlog* übergeführt werden. Dieses wird während des Sprints nicht verändert. Während des Sprints wird jeden Tag ein *Daily Scrum* mit einer Länge von 15 Minuten durchgeführt, in dem besprochen wird, wie der aktuelle Fortschritt ist, und welche Aufgaben als nächstes zu erledigen sind. Bei Sprintabschluss findet ein Review und eine Retrospektive statt. Im Review werden unter anderem die erfüllten Aufgaben vorgestellt, von dem Product Owner abgenommen und das Product Backlog aktualisiert. Außerdem wird vom Entwicklungsteam besprochen, welche positiven bzw. negativen Ereignisse während des Sprint stattgefunden haben, und wie diese gelöst wurden. Im Anschluss findet die Retrospektive statt, in der das Entwicklungsteam über mögliche Verbesserungen für den nächsten Sprint, sowie das Klima innerhalb des Teams und das Zurechtkommen mit dem Prozess und den verwendeten Hilfsmitteln berichtet.

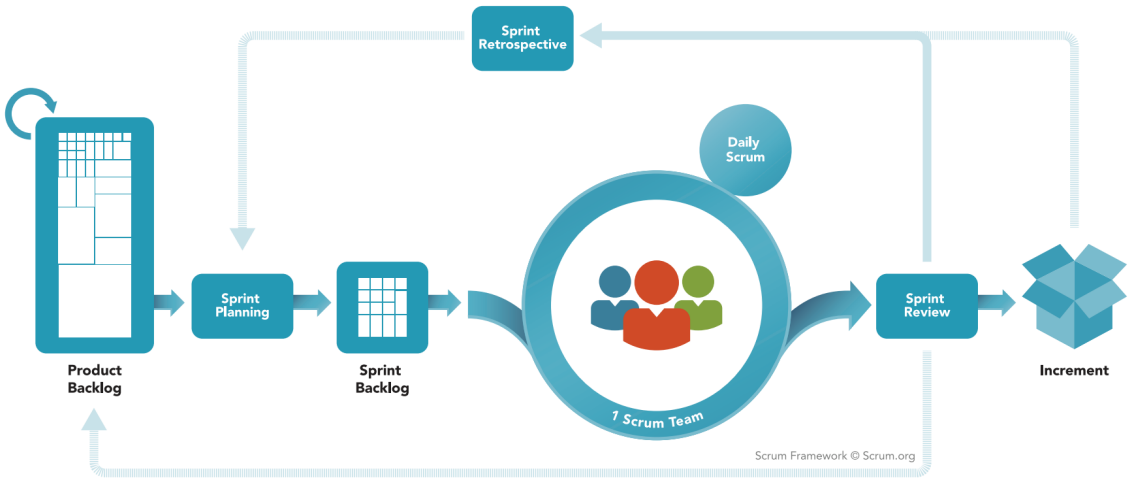

Abbildung 4.2: Zeitlicher Ablauf des Scrumprozesses [WIS16]

Ein Vergleich der beiden vorgestellten Methoden ist schwierig, da die Herangehensweise an ein Softwareprojekt unterschiedlich sind. Gemein ist den Ansätzen, dass ein iterativer Prozess zur Entwicklung verwendet wird, wobei XP bezüglich Änderungen flexibler gestaltet ist, da die Releasezyklen kürzer sind, und auch innerhalb einer Iteration Änderungen erlaubt sind (siehe Abb. 4.3). Außerdem können Methoden aus XP durchaus auch bei Scrum eingesetzt werden. In [RA15] wird dazu beschrieben, dass in der Praxis Mischungen verschiedener agiler Methoden Verwendung finden. Für die Einführung von agilen Methoden ist Scrum besser geeignet, da es einfacher als Prozess im traditionellen Sinn gesehen werden kann, und dadurch den traditionellen Ansätzen näher kommt.

Nach der Entscheidung, welche Methoden bei der Entwicklung von einer mobilen Applikation generell eingesetzt werden sollen, gilt es auch die technische Basis für die Applikation zu beleuchten. Nachdem momentan 3 Betriebssysteme verschiedener Hersteller am gebräuchlichsten sind (siehe auch Kapitel 4.2), muss entschieden werden, welche Art von Applikation entwickelt werden soll, und für welche Betriebssysteme diese angeboten werden sollen. Ist nur ein Betriebssystem im Fokus, so kann durchaus eine native Applikation mit den standardmäßig zu Verfügung gestellten Werkzeugen des Herstellers entwickelt werden. Soll die Applikation jedoch auf mehreren Betriebssystemen ausgeführt werden können, so stehen hierfür andere Möglichkeiten zur Verfügung. In der Literatur ist die Anzahl unterschiedlich, was die verschiedenen Ausprägungen betrifft ([XX13], [EAY16], und [OT12]), daher wird für die nachfolgende Aufzählung eine Unterscheidung nach dem Grad der Verschränkung mit dem Betriebssystem gewählt.

- Native Applikationen: Hierbei muss getrennt für jedes Betriebssystem eine eigenständige Applikation entwickelt werden. Nachdem die aktuell am weitesten verbreiteten

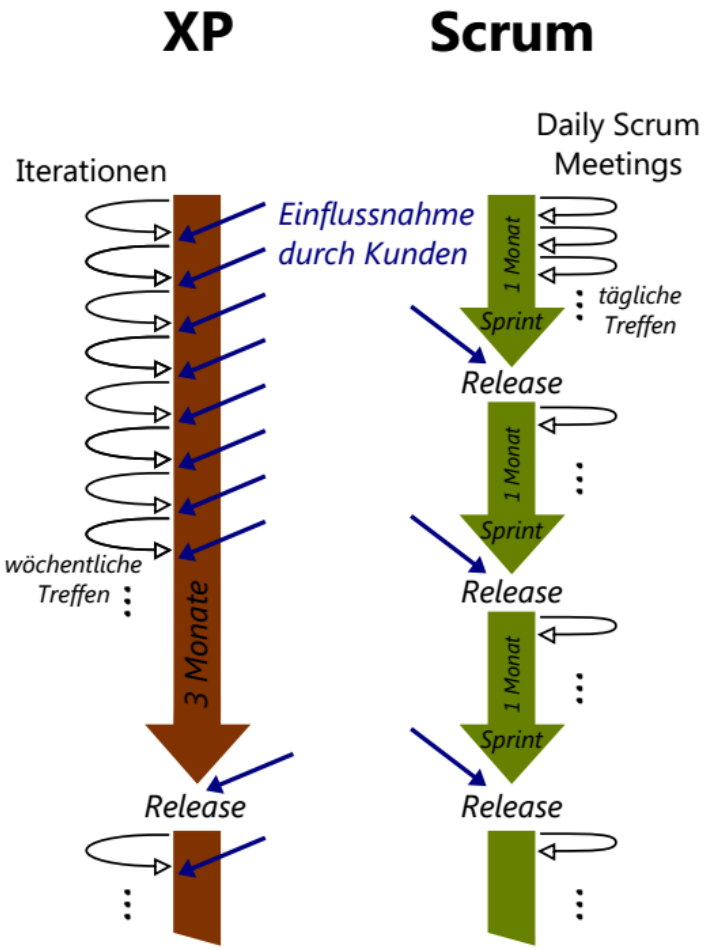

Abbildung 4.3: Vergleich der Zyklen in XP und Scrum [BF12, S.13]

Betriebssysteme alle unterschiedliche Architekturen aufweisen, und jedes davon für die Programmierung einer Applikation auf unterschiedliche Sprachen setzt, ist die Entscheidung für diese Art der Entwicklung nur selten sinnvoll. Jedoch ist hier die optimale Verwendbarkeit aller Funktionen gegeben, was bei den anderen Möglichkeiten teilweise nur sehr eingeschränkt funktioniert.

- Web-Applikationen: Hierbei handelt es sich um Applikationen, die direkt über den Webbrowser dargestellt werden, also im Grunde Webseiten, bei der eine stärkere Interaktion mit dem Benutzer stattfinden kann. Notwendig ist hier immer eine Internetverbindung. Ein Zugriff auf Funktionen des Betriebssystems (GPS, Kamera) ist nicht oder nur sehr eingeschränkt möglich. Zusätzlich ist die Ausführungsgeschwindigkeit von der Internetverbindung abhängig, und reicht nicht an die einer nativen Applikation heran. Vorteilhaft ist jedoch, dass die Applikation auf jeder

Plattform ausgeführt werden kann.

- Hybrid-Web-Applikationen: Hier werden Webinhalte in einem Container auf dem mobilen Gerät angezeigt. Im Vergleich zur reinen Webapplikation besteht eine bessere Verwendbarkeit der gerätespezifischen Funktionen. Zusätzlich wird das Userinterface immer gleich angezeigt, und nicht an das Betriebssystem angepasst, auf dem es verwendet wird. Ein bekannter Vertreter von Frameworks dieser Kategorie ist PhoneGap ([PG17]).
- Hybrid-Applikation: Diese Applikationen sind „native“ Applikationen, bei denen derselbe Code in großen Teilen wiederverwendet werden kann. Dafür wird aus dem produzierten Code ein Kompilat erstellt, das, ausgeführt von einem Interpreter, in der nativen Umgebung ausgeführt wird. Dies bietet den Vorteil, dass die Ausführungsgeschwindigkeit sehr nahe an der wirklich nativer Applikationen liegt. Außerdem können alle Funktionen des Betriebssystems verwendet werden, sofern sie von dem Interpreter unterstützt werden. Ein bekannter Vertreter dieser Art von Frameworks ist Xamarin ([XA17]).

Welche Implementierung der oben genannten angewendet werden soll ist abhängig davon, welche Anforderungen bestehen, bzw. welche Ressourcen zur Verfügung stehen. Besteht der Zweck hauptsächlich darin, dass Daten mit Servern ausgetauscht werden sollen, so können Web-Applikationen oder Hybrid-Web-Applikationen sinnvoll sein. Ist eine starke Nutzung der Hardwarekomponenten im Vordergrund, so stehen die beiden genannten Hybrid-Arten oder eine native Applikation - wenn die Entwickler über die notwendigen Fertigkeiten verfügen - im Vordergrund. Soll das Design der Applikation nahe an den Herstellervorgaben der Betriebssysteme angelehnt sein, so sollte einer Hybrid-Applikation oder einer nativen Applikation der Vorzug gegeben werden. Ein weiterer Grund für die Verwendung von Hybrid-Applikationen kann auch die Wiederverwendbarkeit von Code aus anderen Programmen sein. Einer der wesentlichsten Gründe, keine native Applikation zu erstellen, ist aber sicher die Zeitersparnis, die durch die anderen Ansätze erreicht werden kann.

4.2 Betriebssysteme

Es gibt eine Vielzahl von, mehr oder weniger, gebräuchlichen Betriebssystemen für Smartphones. Um eine Idee von diesen mobilen Betriebssystemen zu bekommen, werden hier die 3 gebräuchlichsten kurz beschrieben. Außerdem sollen einige grundlegende Informationen geliefert werden, die für den Beginn der Entwicklung einer Applikation in den vorgestellten Betriebssystemen von Vorteil sein können. Die Auswahl der beschriebenen Betriebssysteme wurde anhand einer Studie der IDC (**I**nternational **D**ata **C**orporation, [IDC2016]) vorgenommen, welche eine Analyse der Verkaufszahlen der einzelnen Smartphonebetriebssysteme vom vierten Quartal 2015 bis zum dritten Quartals 2016 erstellt hat (siehe Abbildung 4.4). Anhand dieser Studie sind dies die Betriebssysteme Android, iOS und Windows Phone.

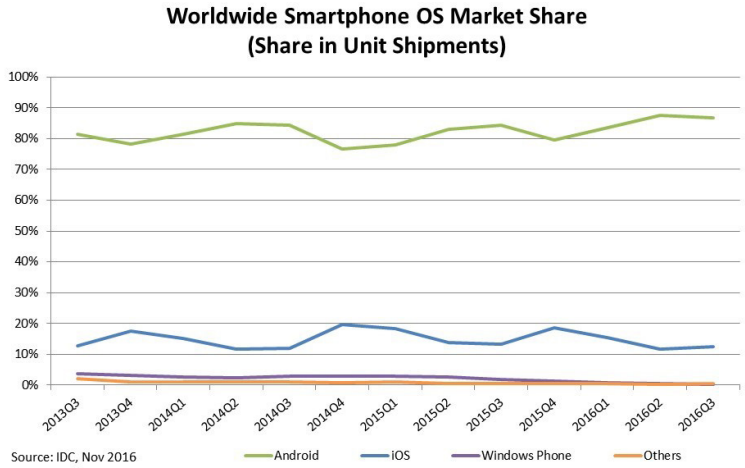

Abbildung 4.4: Entwicklung der Smartphone-Betriebssystem-Absatzzahlen [IDC2016]

4.2.1 Android

Android ist ein, auf dem Linux Kernel basierendes, Betriebssystem, welches ursprünglich vom gleichnamigen Unternehmen für mobile Plattformen entwickelt wurde, und mittlerweile von Google und der Open Handset Alliance weiterentwickelt wird. Damit setzt dieses Betriebssystem als einziges der hier beschriebenen auf einem monolithischen Kernel auf. Das bedeutet, dass im Kernel „alle grundlegenden Systemdienste integriert“ ([HWS2003]) sind. Mit einem Anteil von 86,8% im dritten Quartals 2016 ([IDC2016]) der weltweit verkauften Smartphonebetriebssysteme, ist es das am meisten verbreitete Betriebssystem für Smartphones. Android wird auf den verschiedensten Plattformen, von Smartphones über Digital Media Receiver, bis hin zu Kameras eingesetzt. Da es sich bei diesem Betriebssystem um eine Open-Source-Plattform handelt, verwenden viele Hersteller zum Teil auch eigens adaptierte Android Kernels. Zum Zeitpunkt dieser Arbeit ist es in der Version 7.0 unter dem Namen „Nougat“ verfügbar. In Abbildung 4.5 ist der Architekturaufbau von Android zu sehen.

Der Kern von Android wurde in C und C++ geschrieben. Die Architektur gliedert sich in mehrere Schichten. In [KS15] werden diese beschrieben. Die unterste Schicht, der *Kernel Layer* beinhaltet den *Linux Kernel*, der in regelmäßigen Abständen aktualisiert wird. Weiters beinhaltet der Kernel alle notwendigen Treiber, um die Kommunikation mit der Hardware zu gewährleisten.

Der HAL (**H**ardware **A**bstraction **L**ayer) kapselt die Interaktionsmöglichkeiten mit der Hardware mit Hilfe von Interfaces. Hierüber können beispielsweise die Kamera, Bluetooth, Audiofunktionalitäten oder Sensoren abgefragt oder bedient werden.

Der *Native Libraries Layer* beinhaltet Module, die zur Datenverarbeitung benötigt werden. Dabei ist beispielsweise die *Webkit*-Library dafür zuständig, HTML Daten anzuzeigen.

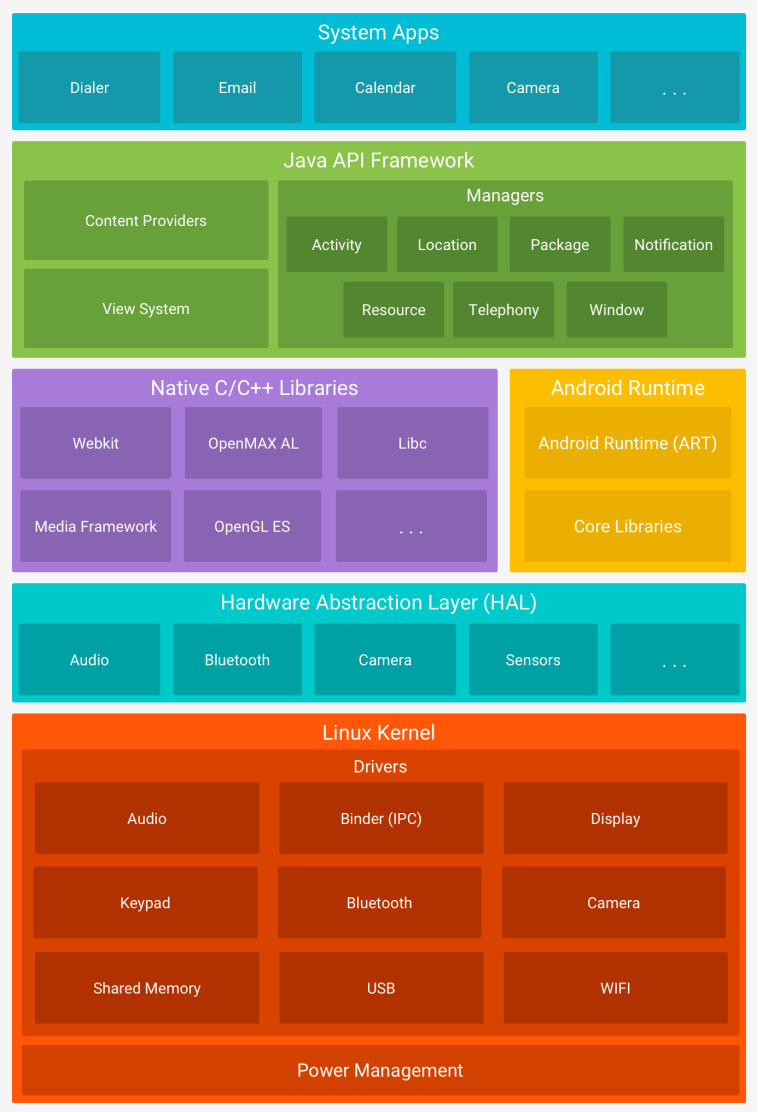

Abbildung 4.5: Architektur von Android ([AND16])

Das *Media Framework* beinhaltet Kodierungsalgorithmen zum Abspielen und Aufnehmen von Mediendatenformaten. Die *OpenGL ES*-Bibliothek ist für das Anzeigen von 2D und 3D Inhalten zuständig. Die *OpenMAX AL*-Bibliothek ([OA16]) ist ebenfalls eine Multimediabibliothek zur Verwendung von Audio-, Video-, und Kamerafunktionalität. Die Libc-Bibliothek ist die Standardbibliothek für die Programmiersprache C.

Der *Android Runtime*-Layer enthält die eigentliche Laufzeitumgebung für Applikationen, die ART (**A**ndroid **R**un**T**ime), sowie Java Bibliotheken.

Der *Java API Framework*-Layer beinhaltet Interfaces, die von Applikationen genutzt

werden können, um mit den darunterliegenden Schichten kommunizieren zu können. Dazu zählen verschiedene Manager, wie der

- *Activity Manager*, der für den Lebenszyklus einer Applikation verantwortlich ist,
- der *Location Manager*, der Zugriff auf GPS und die Mobilfunkinfrastruktur zum Zweck der Standortbestimmung bietet,
- der *Telephony Manager*, der die Telefoniefunktionen abbildet,
- der *Resource Manager*, über den Applikationen auf verschiedene Ressourcen zugreifen können,
- der *Notification Manager*, um den Benutzer z.B. mit einer blinkenden LED zu benachrichtigen,
- der *Package Manager*, der Informationen über die gerade installierten Applikationen liefert und
- der *Window Manager* für die Kommunikation mit dem Display

Weiters beinhaltet dieser Layer auch den *Content Provider*, der die Datenübertragung zwischen zwei Applikationen ermöglicht, und das *View System*, das zur Implementierung der Benutzeroberfläche benötigt wird.

Schlussendlich befinden sich in der obersten Ebene verschiedene Systemapplikationen, um die grundlegenden Funktionen für den Benutzer bedienbar zu machen, wie unter anderem der *Dialer*, um Telefongespräche zu führen, ein Email- und Kalender-Programm, oder auch eine Applikation, um Fotos aufzunehmen.

Als Programmiersprache für Applikationen wird Java verwendet, in Verbindung mit XML für die Erstellung der Benutzeroberfläche. Daneben gibt es für die Erstellung der Benutzeroberfläche auch graphische Editoren. Zur Ausführung des „Java“ Codes wurde bis zur Version 5.0 die DVM (**D**alvik **V**irtual **M**achine) verwendet. Bei der DVM handelt es sich um eine abgewandelte Form der JVM (**J**ava **V**irtual **M**achine). In [CUN09] werden die Unterschiede zwischen JVM und DVM erläutert. Durch eine stärkere Anpassung an die ARM-Architektur, die in mobilen Geräten eingesetzt wird, können die Programme mit der DVM performanter ausgeführt werden. Ein Nachteil dieses Designs ist aber, dass das entstehende Kompilat - der Dalvik Bytecode - nicht mehr plattformunabhängig ist. Es ist auch zu betonen, dass trotz der selben Programmiersprache bzw. des selben Quellcodedateiaufbaus, nicht jedes Java Programm auf Android lauffähig ist [AND2012, S. 69-71], weil nicht alle Bibliotheken, die für die JVM verfügbar sind, für die DVM kompiliert wurden.

Mit der Andriod-Version 5.0 wurde die DVM durch die ART ersetzt, die unter anderem Verbesserungen, wie eine verbesserte GC (**G**arbage **C**ollection) - diese sorgt dafür, dass

nicht länger benötigte Objekte aus dem Speicher entfernt werden - und die AOT (**A**head **O**f **T**ime) Kompilation beinhaltet. Die AOT-Kompilation beschreibt hier die Möglichkeit, aus dem Bytecode direkt einen nativen Maschinencode für das Gerät zu erzeugen. Dies ist ein Vorteil, weil weniger Leistung aufgewendet muss, um den Bytecode zur Laufzeit des Programms zu interpretieren, und damit die Ausführungsgeschwindigkeit zu der Laufzeit und im Besonderen während des Applikationsstarts verbessert wurde. Es wird hier nach wie vor ein JIT Kompilat erzeugt, jedoch wird während der Leerlaufzeit des Geräts aufgrund von gesammelten Informationen über das Verwendungsprofil des Benutzers für die Applikation eine AOT-Kompilation durchgeführt. Auf diese optimierten Programmteile wird während der Laufzeit von dem JIT Übersetzer zugegriffen. Außerdem wird dieser Prozess kontinuierlich weitergeführt, um eine immer bessere Optimierung zu erreichen. In [YB15] wird außerdem beschrieben, dass durch den Wechsel zu der neuen Laufzeitumgebung eine Verbesserung in Bezug auf die Batterielaufzeit erreicht werden konnte. Dies hat damit zu tun, dass die DVM keine JVM (**J**ava **V**irtual **M**achine) ist, sondern Teile der Java Klassenbibliotheken portiert wurden, um unter der ART lauffähig zu sein. Zur Erstellung eines Programmes wird aber dennoch eine Version des JDK (**J**ava **D**evelopement **K**it) benötigt.

Es besteht jedoch auch die Möglichkeit, Anwendungsteile nativ in C zu entwickeln. Dies ist dann interessant, wenn sehr performante Anwendungen, wie zum Beispiel Spiele entwickelt werden sollen.

Zur Entwicklung von Apps wird entweder, wenn Java-basierte Anwendungen realisiert werden sollen, das „Android SDK“ benötigt, oder, wenn native Programmbibliotheken erstellt werden sollen, das „Android NDK“ (**N**ative **D**evelopement **K**it). Die Erstellung einer App kann auf vielen Plattformen erfolgen, wie zum Beispiel

- Windows,
- Linux,
- und auch MacOS.

Bei der Fehlersuche zur Laufzeit einer Applikation stellt die ADB (**A**ndroid **D**ebug **B**ridge), welche in der Android SDK enthalten ist, eine große Hilfe dar. Bei diesem Programm handelt es sich um ein Konsolenprogramm, welches die Log-Informationen des über ein USB-Kabel angeschlossenen Smartphones in Echtzeit anzeigt.

Soll eine App entwickelt werden, dann kann das kostenlos geschehen. Bei einer Veröffentlichung im Google-eigenen App-Store muss ein Account errichtet werden, wobei momentan Kosten von einmalig $25 anfallen. Für die Verbreitung über den App-Store müssen einige Vorgaben, die den Quellcode oder die Programmsymbole umfassen, eingehalten werden. Daneben können Android-Apps aber auch ohne Store verteilt werden. Wird eine kostenpflichtige App im Store vertrieben, verbleiben 30% des Preises als Provision bei Google, 70% erhält der Verkäufer ([AND17]). Weitergehende Informationen können von [GOO13] bezogen werden.

4.2.2 iOS

iOS (**i**Phone **O**perating **S**ystem) ist ein Betriebssystem für mobile Geräte der Firma Apple. Mit einem Anteil von 12,5% im dritten Quartals 2016 ([IDC2016]) ist es das Betriebssystem, welches am zweithäufigsten auf gekauften mobilen Endgeräten betrieben wurde. Verwendet wird iOS in allen Versionen des iPhones, der iPad Tablets und des Digital Media Receivers Apple TV. Aktuell steht iOS in der Version 10.2 zu Verfügung.

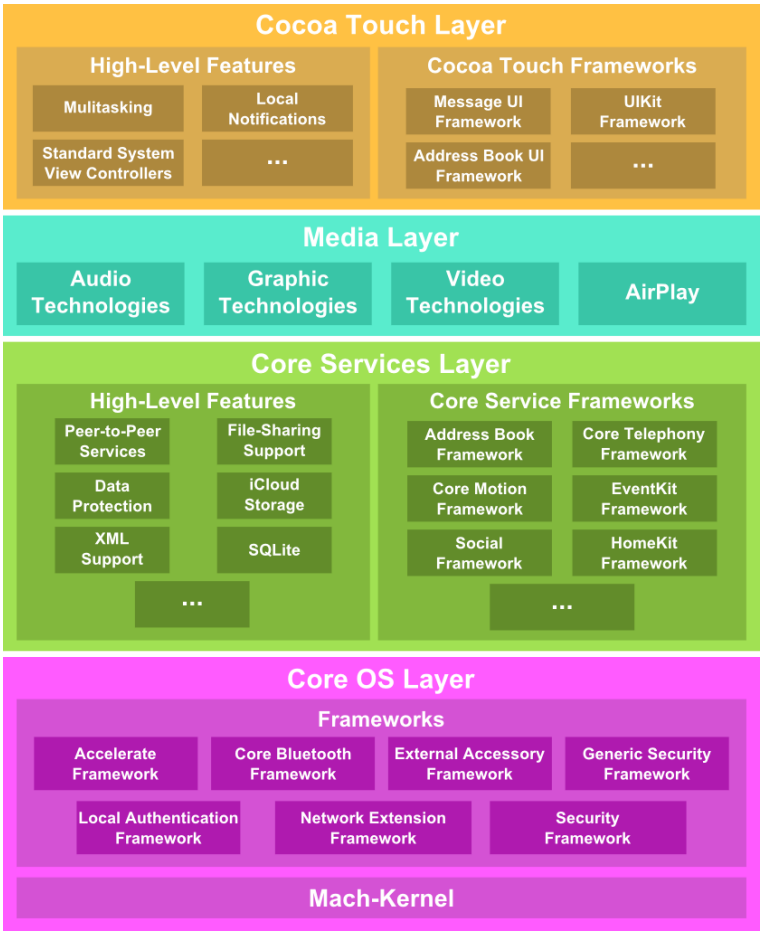

Abbildung 4.6: Aufbau von iOS

Im Kern basiert iOS auf dem Betriebssystem Darwin, einem Betriebssystem, das auf dem *Mach-Kernel*, sowie Teilen des Unix-Systems BSD (**B**erkeley **S**oftware **D**istribution) aufbaut (siehe Abbildung 4.6). iOS setzt damit, im Gegensatz zu Android auf einem Hybridkernel auf. Zusätzlich zum Mach-Kernel befinden sich in der untersten Schicht auch noch verschiedene Frameworks zur Kommunikation mit den einzelnen Hardwarebestandteilen des Gerätes. Zu diesen Frameworks gehören

- das *Accelerate Framework*: Es bietet Möglichkeiten für digitale Signalverarbeitung,

Bildverarbeitung und lineare Algebra.

- das *Core Bluetooth Framework*: Hiermit kann die Bluetooth-Architektur verwendet werden.
- das *External Accessory Framework*: Dieses dient dazu, um mit externer Hardware - entweder kabellos über Bluetooth oder kabelgebunden über den 30-poligen Konnektor - kommunizieren zu können.
- das *Generic Security Framework*: Hier werden unter anderem standardisierte Schnittstellen angeboten, die Funktionen zur sicheren Kommunikation für Applikationen bereitstellen.
- das *Local Authentication Framework*: Es bietet die Verwendung der *Touch Id* an. Hierbei handelt es sich um eine lokale Benutzerauthentifizierung über einen Fingerabdruck-Sensor.
- das *Network Extension Framework*: Mit diesem Framework können VPN-Verbindungen verwaltet werden.
- das *Security Framework*: Es bietet Möglichkeiten für die Sicherung der Daten einer Applikation, sowie die sichere gemeinsame Verwendung von Daten mit mehreren Applikationen auf dem Gerät.

Der *Core Services Layer* unterteilt sich in die *High-Level Features* und die *Core Services Frameworks*. Die *High-Level Features* ermöglichen unter Anderem die Verwendung verschiedener Datenformate (*SQLite* und *XML Support*) und Datensicherheitsfunktionen (*Data Protection*), sowie von dem Austausch von Daten mit *iTunes* (*File-Sharing Support*), der *iCloud* (*iCloud Storage*) und anderen Bluetooth-Geräten (*Peer-to-Peer Services*). Die *Core Services Frameworks* beinhalten eine Vielzahl von Frameworks, die zum Beispiel Zugriff auf das Adressbuch (*Address Book Framework*), den Kalender (*EventKit Framework*) oder die Telefonie-Funktionalität (*Core Telephony Framework*) gestatten. Weiters können auch verschiedenste andere Daten, wie *Social Media Konten* (*Social Framework*) verwaltet, oder Sensordaten (*Core Motion Framework*) und Daten von Geräten zur Heimautomation über das von Apple entwickelte Protokoll *Homekit* (*HomeKit Framework*) ausgetauscht werden.

Der *Media Layer* bietet Funktionen und Frameworks zur Verarbeitung von Video- (*Video Technologies*), Audio- (*Audio Technologies*) und Grafikdateien (*Graphic Technologies*). Außerdem besteht die Möglichkeit, Audio- und Videodaten über *AirPlay* zu kompatiblen Geräten, wie zum Beispiel *Apple TV*, zu übertragen.

Der *Cocoa Touch* Layer unterteilt sich in die *High-Level Features* und die *Cocoa Touch Frameworks*. Die *High-Level Features* bieten Entwicklern Basisinfrastruktur an wie z.B.

- *Standard System View Controllers*: Diese stellen vorgefertigte Benutzeroberflächenelemente zu Verfügung, um Aufgaben wie das Erstellen einer SMS oder E-Mail zu vereinfachen, oder ein Video aufzunehmen.
- *Local Notifications*: Mit Local Notifications kann eine Applikationen den Benutzer über bestimmte Vorgänge in der Applikation informieren, selbst wenn sich diese gerade nicht im Vordergrund befindet.
- *Multitasking*: Hierbei können Applikationen, die sich nicht im Vordergrund befinden, - etwa weil der *Home Button* des Smartphones gedrückt wurde - dennoch im Arbeitsspeicher gehalten werden, und, wenn nötig auch Aufgaben im Hintergrund ausführen, wie z.B. das Herunterladen von Dateien aus dem Netzwerk.

Die Frameworks werden für die graphische Darstellung von Applikationen benötigt. Abbildung 4.6 enthält dabei nur einen kleinen Teil der vielen verfügbaren Frameworks. Hiermit ist es einfacher über verschiedenste Applikationen hinweg ein einheitliches Aussehen zu gewährleisten. Die beispielhaft genannten Frameworks sind:

- *Message UI Framework*: Hiermit wird die graphische Darstellung und auch die Logik für das Erstellen und Senden von E-Mails und SMS-Nachrichten bereitgestellt.
- *UIKit Framework*: Dieses Framework beinhaltet die essentiellen Grundlagen für die Erstellung einer Applikation in iOS. Weiters wird auch der Zugriff auf diverse Sensorinformationen, wie beispielsweise den Batteriestatus ermöglicht.
- *Address Book UI Framework*: Über dieses Framework kann ein einfacher Zugriff auf das Adressbuch durchgeführt werden. Hierbei können Kontakte erstellt, bearbeitet und angezeigt werden.

Im Gegensatz zu Android, welches sehr offen gestaltet ist, was den Vertrieb der Apps und die Verwendung von Android auf anderen Plattformen angeht, ist iOS eher geschlossen. Es ist einerseits nur für Geräte der Firma Apple verfügbar, andererseits können Apps von Privatpersonen legal nur aus dem App-Store bezogen werden. Eine weitere Einschränkung gegenüber Android besteht darin, dass native Apps für iOS nur auf Computern mit MacOS entwickelt werden können, da die Entwicklungsumgebung *XCode* unter keinem anderen Betriebssystem verwendet werden kann. Ein großer Vorteil dieses geschlossenen Systems besteht darin, dass eine entwickelte App meist ohne Probleme auf anderen Apple-Geräten verwendet werden kann. Im Gegensatz dazu kann es bei der Verwendung von Android, welches auf einer Vielzahl von Smartphones verschiedener Hersteller lauffähig sein soll, zu Problemen kommen. Das hat damit zu tun, dass die Smartphonehersteller eigene Features (wie zum Beispiel verschiedene Treiber) in Android integrieren.

Bei der Erstellung von Apps wird die Entwicklungsumgebung *XCode* mit der Apple-eigenen Programmiersprache Objective-C verwendet. Für die Erstellung der Benutzeroberfläche existiert neben der Möglichkeit der Implementierung im Code, noch ein graphischer Editor namens *Interface Builder*.

Hilfreich bei der Fehlersuche innerhalb einer Applikation zur Laufzeit ist auch die Möglichkeit, in XCode Log-Informationen des angeschlossenen Gerätes anzeigen zu lassen. Diese Informationen sind unter dem Punkt „Window->Organizer“ zu finden.

Für die Entwicklung einer App ist es notwendig, einen kostenlosen Entwickler-Account zu registrieren. Damit hat man die Möglichkeit, die App unter MacOS auf einem Simulator für iPhone und iPad zu testen. Soll die App auf einem physischen Gerät ausgeführt werden, muss ein kostenpflichtiger Account angelegt werden. Auf diesem muss das zu verwendende Gerät zuerst registriert werden. Der Account kostet jährlich, je nach Typ des Accounts, momentan $99 oder $299. Für universitäre Einrichtungen gibt es die Möglichkeit eines kostenlosen Accounts. Allen Programmen, die veröffentlicht werden sollen, ist gemein, dass eine Reihe von strikten (strikteren, als bei der Entwicklung von Android Programmen) Vorgaben, wie etwa die Größe der Icons, oder gewisse Vorgaben den Quellcode betreffend, eingehalten werden müssen. Wird eine kostenpflichtige App im App-Store angeboten, so erhält man 70% des Kaufpreises, 30% behält Apple als Provision ein ([APP13]). Neben dem App-Store gibt es noch die Möglichkeit für Unternehmen einen eigenen, unternehmensinternen, Appstore für sogenannte In-House-Apps zu betreiben ([APP13]). Weitere Informationen zu der Architektur von iOS können von [APP17] bezogen werden.

4.2.3 Windows 10

Windows 10 ist das jüngste der 3 beschriebenen Betriebssysteme. Vor der Einführung von Windows 10 existierte ein eigenes Betriebssystem für Mobilgeräte, nämlich Windows Phone. Es wurde 2010 fertiggestellt, und basierte auf dem Windows-CE Kernel. In der aktuellen Version 10 basiert Windows 10 auf dem Windows-NT Kernel, bei welchem es sich ebenfalls, wie bei iOS, um ein Hybridkernel handelt. Mit einem Anteil von 0,3% im dritten Quartal 2016 ([IDC2016]) bei verkauften Smartphones mit diesem Betriebssystem, liegt es auf dem dritten Platz.

Programme können für Windows 10 in C#, C++, JavaScript und Visual Basic entwickelt werden. Für die Erstellung der Benutzeroberfläche stehen drei unterschiedliche Methoden zu Verfügung.

- XAML (e**X**tensible **A**pplication **M**arkup **L**anguage) - bei Verwendung von C#, Visual Basic oder C++ - ist eine von Microsoft entwickelte, XML-basierende, Sprache, die eine bessere Trennung der Benutzeroberfläche von der dahinterliegenden Programmlogik ermöglicht.
- HTML wird für Programme verwendet, die mit Javascript erstellt werden sollen.
- DirectX kann bei Verwendung von C++ gewählt werden. Dabei wird die Benutzeroberfläche ebenfalls in C++ programmiert.

Für die Programmentwicklung wird von Microsoft die kostenlose Entwicklungsumgebung „Visual Studio Community“ angeboten. Im Moment ist „Visual Studio Community“,

wie XCode, das ausschließlich mit MacOS verwendet werden kann, nur für Windows verfügbar.

Mit der Einführung von Windows 10 besteht die Möglichkeit, auf verschiedensten Plattformen, sei es Destop-PCs, Smartphones, Spielekonsolen oder IoT-Geräten (**I**nternet **o**f **T**hings), Windows zu verwenden, und Programme zu entwickeln, die auf all den genannten Geräten ausgeführt werden können. Abb. 4.7 zeigt diese Entwicklung.

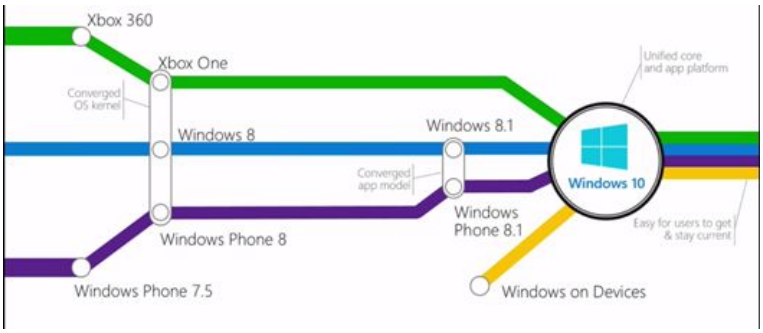

Abbildung 4.7: Überblick von Windows 10 ([WIN10])

Ebenso wie bei Android kann eine Applikation für Windows-basierte Smartphones kostenlos erstellt werden. Soll die Applikation über den *Windows Store* angeboten werden, so ist ein kostenpflichtiger Entwicklerzugang erforderlich. Die Kosten für dieses Konto betragen $19 für Einzellizenzen und $99 für Firmen. Für Schüler und Studenten gibt es noch die Möglichkeit, einen Entwickleraccount kostenlos zu erhalten. In Verbindung mit diesem Account muss auch ein Dreamspark Account erstellt werden, bei dem belegt werden muss, dass man wirklich ein Schüler/Student ist. Wird eine App über den Store kostenpflichtig angeboten, so erhält die Entwicklerin oder der Entwickler, ebenso wie bei Google und Apple 70% des Applikationspreises. Genauere Informationen über die Entwicklung für Applikationen für Windows können über [WIN13] bezogen werden.

KAPITEL 5

Prototypische Implementierung

In diesem Kapitel wird die Implementierung einer Applikation für Aufgaben in der Industrieautomation vorgestellt. In dieser werden die in den vorigen Kapiteln beschriebenen Erkenntnisse und Ideen größtenteils umgesetzt. Größtenteils deshalb, weil für die Implementierung der Applikation ein bereits vorhandenes Engineeringtool, FFT (Festo Field Device Tool), die Grundlage bildet. Dieses Tool wird für Wartungs- und Diagnosezwecke verwendet. Es ist mithilfe des .NET-Frameworks in C# erstellt worden, und basiert wiederum auf eigens von der Firma Festo entwickelten Frameworks, die Erleichterungen bei der Gestaltung von Benutzeroberflächen unter Windows bringen. Die Benutzeroberfläche wurde mit Windows Forms implementiert. Das Tool kommuniziert mit Geräten der Firma Festo über verschiedene proprietäre, Ethernet-basierende und über die serielle Schnittstelle kommunizierende Protokolle, und bietet unter anderem folgende Funktionen:

- Scanfunktion für Ethernet-basierende Produkte
- Automatische Updatefunktion für Anwendung und Firmwaredateien
- Integrierte Datenablage für Firmwaredateien
- Einfach- und Mehrfach-Firmware-Download für ausgewählte Geräte (inkl. Report und Kompatibilitätsprüfung für Hardware und Software)
- Backup- und Wiederherstellungsfunktion
- Einstellung der IP-Adresse für Geräte

Im folgenden Abschnitt soll der Aufbau und die Funktionsweise der, der Applikation zugrundeliegenden, Schichten beschrieben werden.

Layerstrukturen und Funktionen

Die Applikation ist in mehreren Schichten aufgebaut:

- Festo.Scan,
- Festo.Firmware,
- Festo.EditorSDK,
- Festo.Common,
- und Festo.System.

Dieser Aufbau ist zur besseren Illustration in Abb. 5.1 zu sehen.

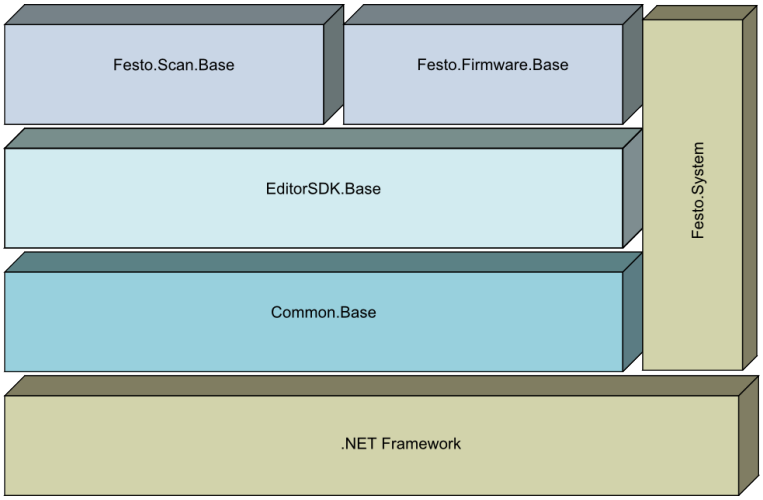

Abbildung 5.1: Schichtenaufbau der Anwendung

Festo.System bietet verschiedene Konvertierungs- und Validierungsfunktionen (z.B. die Validierung einer IP-Adresse) sowie Zugriff auf Downloadfunktionen aus dem Internet an.

Festo.Common und *Festo.EditorSDK* stellen grundlegende Funktionen und Objekte bereit. Es wird unter Anderem ein Variablensystem zur Verfügung gestellt, das in der Windows-Applikation einige Vereinfachungen (z.B. Binding[1]) für die Erstellung und Verwaltung der Benutzeroberfläche bietet, sowie eine Möglichkeit, Übersetzungen für die Benutzeroberfläche wartbar zu gestalten.

Festo.Firmware bildet die Funktionalität ab, Firmware-Dateien für die einzelnen Geräte laden und auf ihre Kompatibilität hin überprüfen zu können. Außerdem stellt sie die Protokollimplementierungen für den Firmware-Download zur Verfügung.

[1] Ändert sich der Wert einer Variable, so wird der Wert auch automatisch im Benutzeroberflächenelement aktualisiert.

Festo.Scan, die zentrale Komponente, stellt die Protokollimplementierungen bereit. Damit können Informationen über Geräte abgerufen werden, wie z.B. Scan-, Diagnose- oder Versionsinformationen. Weiters werden auch Sicherungs- und Wiederherstellungsfunktionen sowie eine Identifikationsfunktion[2] abgebildet. Sogenannte *Scanhandler* bieten über Interfaces abstrahiert die Möglichkeit an, Kommandos[3] für einzelne Geräte auszuführen. Die *Scanhandler* können über den *Scanhandlercontainer* angesprochen werden, wobei dieser eine weitere Abstraktionsebene darstellt. So kann ein Kommando ausgeführt werden, das eine Funktion für ein Gerät ausführt, ohne nähere Kenntnis des Geräteprotokolls. Dieser Zusammenhang der Klassen ist in Abbildung 5.2 in vereinfachter Weise dargestellt.

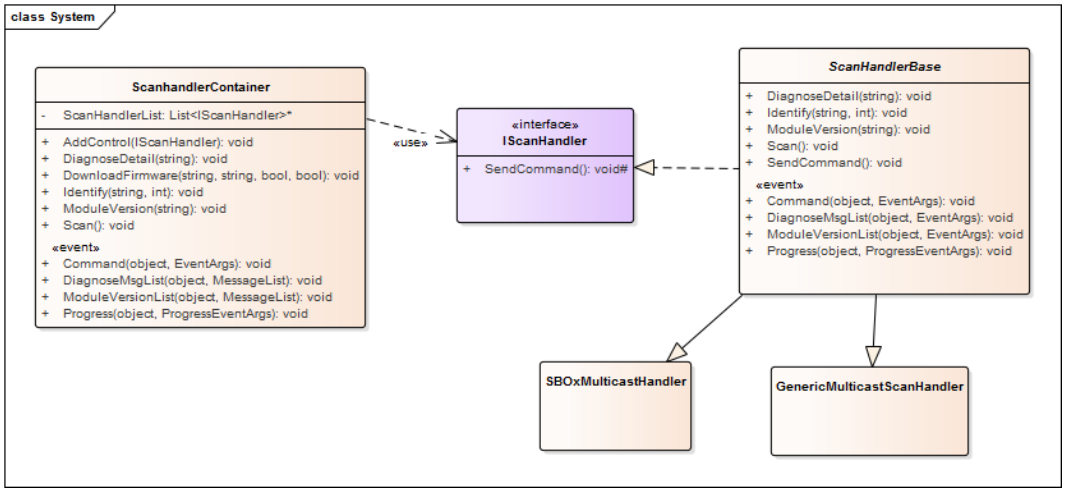

Abbildung 5.2: Vereinfachtes Klassendiagramm des Systems

Implementierung

Bei der Implementierung der mobilen Applikation handelt es sich um einen Prototyp, und stellt somit nicht alle Funktionen des „großen Bruders“ zur Verfügung. Zusätzlich waren gewisse Vorgaben für die Entwicklung der Applikation gegeben:

Die Wichtigste dabei war eine gemeinsame Codebasis für die Windows-Applikation und die Smartphone-Applikation zu schaffen. Aufgrund der umfangreiche Codebasis wäre sonst ein erheblicher Mehraufwand für die Wartung und Weiterentwicklung zu erwarten. Deshalb fiel die erste Wahl des Betriebssystems für die Applikation auf Windows Phone 8. Vor der ersten Analyse der Codebasis wurde fälschlicherweise davon ausgegangen, dass Windows Phone einen ähnlichen Aufbau wie Windows für Desktop-Computer bereitstellt. Jedoch ist die verwendete Architektur, was die verwendbaren Klassen, Namespaces und Bibliotheken angeht, sehr unterschiedlich. Da Windows Phone 8 Anwendungen auf

[2]Das Gerät wird in einen Zustand versetzt, dass zum Beispiel eine LED blinkt, um es in einer Anlage einfach „orten“ zu können

[3]Ein Kommando entspricht einer Funktion.

WinRT[4] basieren, können nur Teile des .NET Frameworks verwendet werden. Außerdem befinden sich die Funktionen und Methoden, wenn vorhanden, oft in anders benannten Namespaces oder Bibliotheken. Die nötige Anpassung der Codebasis wäre nur sehr aufwändig möglich gewesen. Deshalb wurde dieser Ansatz verworfen.

Es wurde nach einem anderen Framework gesucht, das es ermöglicht, die in C#-bestehende Codebasis weiterverwenden zu können. Die Wahl fiel auf Mono, respektive Xamarin. Es bietet die Möglichkeit, den verwendeten Code in großen Teilen beizubehalten, abgesehen von der Benutzeroberfläche. Mit diesem Framework können in C#-geschriebene Programme sowohl für Android, iOS und Windows Phone, als auch für Linux erstellt werden.

Für die Implementierung der Applikation unter Android wurde als Testgerät ein Samsung Galaxy S3 verwendet, mit der Androidversion 4.1.2. Einige interessante Gerätespezifikationen sind

- 1,4 GHz Quad-Core Prozessor,
- WLAN 802.11 a/b/g/n, 2,4 GHz, 5 GHz,
- NFC,
- Bluetooth in der Version 4.0,
- und ein Micro-SD-Steckplatz.

Als Entwicklungsumgebung wurde Microsoft Visual Studio 2010 verwendet, gemeinsam mit einem Plugin von Xamarin. Es wurde eine Solution[5] erzeugt, und im ersten Schritt alle Projekte, welche für eine Ausführung benötigt wurden, importiert. Danach wurden zusätzlich Android-Projekte für alle Windowsprojekte erstellt, und die Quellcodedateien aus den ursprünglichen Projekten verlinkt. Durch diese Verlinkung wurde gewährleistet, dass eine Aktualität der Dateien für alle Plattformen besteht. Die plattformspezifischen Projekte waren notwendig, weil normale .NET-Programmbibliotheken in Bibliotheken von Xamarin für Android nicht referenziert werden können. Die Idee und Verwendung dieser großen Solution hängt auch damit zusammen, dass eine bessere Übersicht über alle Programmkomponenten erhalten wurde. Zusätzlich war eine Fehleranalyse in Visual Studio einfacher möglich.

Für eine erste Analyse der Kompatibilität der bestehenden Programmteile wurde der MoMA (**M**ono **M**igration **A**nalyzer) [MOMA13] verwendet. Dieses Programm untersucht Programmbibliotheken und auch Programme dahingehend, ob sie unter Mono lauffähig sind. Dabei wird geprüft, ob

[4] WinRT ist ein Betriebssystem für ARM-Architekturen.
[5] Eine Solution kann andere Projekte in sich aufnehmen

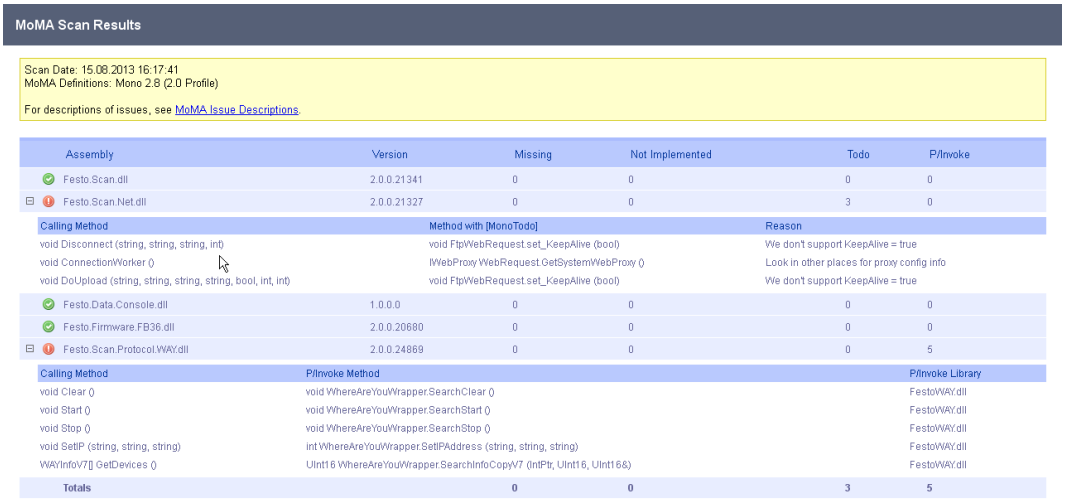

Abbildung 5.3: Ergebnis eines Scans mit MoMA

- die verwendeten Klassen, Funktionen und Methoden der .NET-Frameworks unter Mono implementiert wurden,
- oder plattformspezifische Methodenaufrufe der Betriebssysteme aufgerufen wurden.

Die Analyse ergab erste Hinweise auf problematische Programmteile, die, zur Laufzeit des Programmes, Fehler verursachen können (exemplarisch ist ein derartiges Analyseergebnis in Abb. 5.3 zu sehen). Dabei soll aber nicht der Eindruck vermittelt werden, die Analyse mit MoMA reiche aus, um alle Probleme aufzudecken, die zu beheben sind, um ein auf Windows-basiertes bestehendes Programm plattformunabhängig zu gestalten. Es sind außerdem, von Betriebssystem zu Betriebssystem, verschiedene Eigenheiten zu beachten, um eine wirkliche Plattformunabhängigkeit zu gewährleisten. Im ersten Schritt der Implementierung wurde versucht das FFT unter Ubuntu (Version 12.04) in der Kommandozeile auszuführen, um die Verwendung, vorerst ohne graphische Benutzeroberfläche, zu testen. Die Verwendung der Bibliotheken *System.Windows.Forms.dll* und *System.Drawing.dll* wurde unterbunden, da diese unter Linux zwar teilweise verwendbar sind, bei Android aber nicht. Die *System.Windows.Forms.dll* ist eine Bibliothek, die Objekte und Steuerelemente zur Erstellung der Benutzeroberfläche anbietet. Die *System.Drawing*-Bibliothek wird für die Verwendung von Grafiken benötigt.

Im nächsten Schritt wurde Code entfernt, der für die Steuerung der Benutzeroberfläche verwendet wurde. Dieser befand sich in der Geschäftslogik. Stattdessen sollte dieser Code direkt in der Applikations-Schicht verwendet werden, da er oft nicht wiederverwendbar ist.

Zusätzlich mussten einige verwendete Referenzen auf Bibliotheken, inklusive des verwendenden Codes, entfernt werden, die nur nativ - beispielsweise als C++-Code - und ohne

Sourcecode zur Verfügung standen. Der Grund hierfür war, dass ein natives Kompilat direkt in Maschinencode vorliegt, und daher nicht mit dem Xamarin-Framework kompatibel ist. Der C#-Code wird bei der Kompilation in eine IL (**I**ntermediate **L**anguage) übersetzt. Danach wird das Kompilat, ähnlich wie bei Java, über einen Interpreter zur Laufzeit in Maschinen-Code übergeführt. Aus diesem Grund war es auch nicht möglich, einige Protokolle für die mobile Applikation zu verwenden, wenn diese auf nativen Bibliotheken basierten.

Die nächste Änderung war notwendig, weil gewisse Methodenaufrufe, wie zum Beispiel für die Verwendung eines Multicast-Ports, für Windows und Android unterschiedlich ablaufen müssen. Hier wurde eine Abstraktionsebene eingeführt, um bei Verwendung auf unterschiedlichen Betriebssystemen, die jeweiligen Implementierungen ausführen zu können.

Es wurden einige Kommandos der Desktop-Anwendung implementiert, und zwar der Scan, der Abruf von Diagnoseinformationen, der Firmware-Download und der Abruf von Versionsinformationen[6] sowie der Identifikationsmechanismus. Weiters wurde die Implementierung des Sicherungs- und Wiederherstellungsmechanismus geprüft. Dieses Vorhaben musste jedoch verworfen werden. Auf einem Android-Smartphone, das nicht „gerootet“[7] ist, können die nötigen Ports für eine FTP-Verbindung (Port 20 und 21) nicht verwendet werden. Weiters ist es unwahrscheinlich, dass dienstlich genutzte Smartphones „gerootet“ sind, weil dies ein zusätzliches Sicherheitsrisiko darstellt.

Es soll nun ein kurzer Überblick über die Verwendung der Geschäftslogik in der Applikation gegeben werden. Für die Initialisierung muss der in Listing 5.1 gezeigte Code ausgeführt werden.

Listing 5.1: Initialisierung der Scan-Schicht

```
var scanContainer = new ScanHandlerContainer(true);
scanContainer.AddControl(new GenericMulticast.ScanHandler {ConsoleMode =
    true});
scanContainer.AddControl(new SBOxMulticast.ScanHandler());
scanContainer.Command += finishDelegate;
scanContainer.EnableDiagnose = false;
scanContainer.EnableConsoleOutput = true;
scanContainer.EnableDebugConsoleOutput = false;
scanContainer.Scan();
```

Die Klasse *ScanHandlerContainer* wird verwendet, um auf die einzelnen Funktionen zugreifen zu können. Sie verwaltet die *ScanHandler*, die für die Steuerung der Übertragungsprotokolle verantwortlich sind. Über den Parameter wird gesteuert, ob das Variablensystem und die Übersetzungsfunktionalität im *ScanHandlerContainer* initialisiert werden sollen. Es ist auch möglich, diese Initialisierung selbst durchzuführen. Die zu verwendenden *ScanHandler* werden über die Methode *AddControl* hinzugefügt. Verfügbar

[6] Die Versionen beziehen sich auf die Firmwarekomponenten des Geräts

[7] *Rooten* bezeichnet den Vorgang, bei dem der Benutzer Administratorrechte auf dem Betriebssystem erhält.

waren in der ursprünglichen Implementierung noch andere, jedoch konnten diese aufgrund der zuvor erwähnten Probleme mit nativen Bibliotheken nicht verwendet werden. Dem *Command*-Eventhandler wird im nächsten Schritt der Delegate *finishDelegate* zugewiesen. Dieser wird aufgerufen, wenn ein Kommando beendet wurde. Hier werden zusätzliche Informationen geliefert, die Auskunft darüber geben, ob das Kommando erfolgreich ausgeführt werden konnte, oder, wenn nicht, warum die Ausführung nicht erfolgreich war. Dies ist beispielsweise der Fall, wenn eine Identifikation bei einem Gerät durchgeführt werden soll, dieses Gerät aber inzwischen abgeschaltet wurde. Hier würde eine Zeitüberschreitung auftreten. Die Eigenschaft *EnableDiagnose* würde dazu führen, dass laufend Diagnoseinformationen über die Geräte gesammelt werden. Aufgrund der Implementierung der Protokolle wurde diese Funktion nicht verwendet, da hier Instabilitäten bedingt durch die drahtlose Datenübertragung zu wahrscheinlich waren. Die Eigenschaft *EnableConsoleOutput* wird bei der Desktop-Variante dazu verwendet, um eine Steuerung über die Eingabeaufforderung zu ermöglichen. Aufgrund gewisser Implementierungsdetails war es notwendig, diesen Modus auch in der mobilen Applikation zu verwenden. Ebenfalls deaktiviert wurde die *EnableDebugConsoleOutput*-Eigenschaft, da diese zusätzlich Informationen über die Vorgänge in den tieferen Schichten der Protokollimplementierungen liefern würde. Diese wurden jedoch nicht benötigt. Über die Methode *Scan* wird der Suchvorgang für die gewählten Protokolle gestartet.

In Listing 5.2 ist der Methodenaufruf für das Starten der Identifikationsfunktion eines Gerätes zu sehen. Hier wird einerseits die IP- oder MAC-Adresse als Zeichenfolge des betreffenden Gerätes angegeben, andererseits der Modus für die Identifikation. 0 bedeutet hier ein Abschalten der Identifikation, 1 führt eine „Standard-Identifikation“ durch, und 2 aktiviert die Identifikationsfunktion, bis sie wieder deaktiviert wird. Bei der Standard-Identifikation ist im Gerät definiert, wie lange diese dauert.

Listing 5.2: Identifikation eines Gerätes

```
scanContainer.IdentifyDevice(ipAddress, 0);
```

Listing 5.3 zeigt den Aufruf der Download-Funktion für eine Firmware-Datei auf ein bestimmtes Gerät. Es wird anfangs ein Delegate (*ChangeProgress*) für die Fortschrittsinformation an den Eventhandler *FirmwareProgress* übergeben. Hiermit werden während des Downloads neben dem prozentualen Fortschritt auch alphanumerische Information in Form von Strings für einen besseren Überblick übertragen. Im eigentlichen Methodenaufruf für den Firmware-Download wird als erster Parameter wieder die IP- oder MAC-Adresse des Gerätes angegeben. Der zweite Parameter beschreibt den Pfad, an dem sich die Firmware-Datei befindet. Der dritte Parameter gibt an, ob eine Überprüfung stattfinden soll, um nicht versehentlich eine falsche Datei auf das Gerät zu übertragen. Der letzte Parameter unterdrückt die Ausgabe von Meldungen auf der Konsole. Diese Ausgabe wird nicht benötigt, da durch den Delegate diese Information effizienter angezeigt werden können.

Listing 5.3: Firmware-Download

```
scanContainer.FirmwareProgress += ChangeProgress;
```

```
scanContainer.DownloadFirmware(ipAddress, firmwareFile, true, true);
```

Die letzten beiden Funktionen (Listing 5.4), der Abruf der Diagnosenachrichten sowie der Versionen der Firmwarekomponenten, werden hier gemeinsam beschrieben, weil das Vorgehen in beiden Fällen gleich ist. Es wird einerseits wieder dem *Command*-Eventhandler ein Delegate *finished* übergeben, um über etwaige Fehler informiert zu werden. Andererseits wird dem Eventhandler *DiagnoseMsgList* der *PublishList*-Delegate übergeben. Bei Abfrage der Versionsinformationen wird hier der Eventhandler *ModuleVersionList* verwendet. Mit diesen werden, bei erfolgreicher Übertragung die Nachrichten an die Benutzeroberfläche übergeben. Für den Aufruf von *DiagnoseDetail* oder *ModuleVersion* muss dann eine valide IP- oder MAC-Adresse übergeben werden.

Listing 5.4: Diagnose und Versionsabruf

```
scanContainer.Command += finished;
scanContainer.DiagnoseMsgList += PublishList;
scanContainer.DiagnoseDetail(ipAddress);

scanContainer.ModuleVersionList += PublishList;
scanContainer.ModuleVersion(ipAddress);
```

Der Aufruf der Web-Visualisierung eines Gerätes ist hier nicht gesondert ausgeführt, da bei Unterstützung durch das Gerät nur ein Web-Browser geöffnet, und zu der Adresse des betreffenden Gerätes navigiert wird.

In den folgenden Abbildungen 5.4, 5.5 und 5.6 sind die implementierten Ansichten der mobilen Applikation ersichtlich.

Abbildung 5.4 zeigt von links nach rechts

- den Scan, der zwei Geräte erkannt hat, die über das Multicast-Protokoll gefunden wurden,
- einen Scan, der zusätzlich zu den Geräten auch noch die möglichen Kommandos anzeigt, die ausgeführt werden können, und
- einen Firmware-Download-Vorgang.

Abbildung 5.5 zeigt von links nach rechts

- eine Ansicht zur Aktivierung des Identifikations-Kommandos, bei dem entweder die Identifikation eingeschaltet werden kann und dann manuell auch wieder ausgeschaltet werden muss, oder eine Standardidentifikation, bei der das Gerät eine vorgegebene Zeit die Identifikationsfunktion aktiviert und diese auch selbstständig wieder deaktiviert,
- eine Webseite eines Gerätes, die über das Homepage-Kommando aufgerufen wurde, und

Abbildung 5.4: Screenshots der Anwendung

- ein Ergebnis eines Abrufs von Diagnosenachrichten. Bei diesem Kommando besteht in der Applikation die Möglichkeit einen erneuten Abruf zu starten, um eine Veränderung des Status des Gerätes erkennen zu können.

Abbildung 5.6 zeigt von links nach rechts

- eine Ansicht der Versionsinformationen von verschiedenen Bestandteilen der Firmware des Gerätes. Auch diese Ansicht bietet die Möglichkeit die Versionsinformationen zu aktualisieren, um geänderte Firmwarebestandteile erkennen zu können.
- Ein Einstellungsmenü, das es erlaubt, statt eines richtigen Firmware-Downloads nur eine Demonstration anzuzeigen. Der Grund dafür war die gegebene Instabilität der Protokollimplementierung für den Firmwaredownload, der bei Verwendung über ein WLAN vorlag.

Abbildung 5.5: Screenshots der Anwendung

Abbildung 5.6: Screenshots der Anwendung

KAPITEL 6

Marktwirtschaftliche Analyse

In diesem Kapitel soll eine Analyse über die Marktchancen erstellt werden, die sich durch die Entwicklung und Bereitstellung von mobilen Applikationen allgemein, sowie im Speziellen in der Industrieautomation, ergeben. Hierzu wurde eine Umfrage durchgeführt, anhand derer Aussagen über wirtschaftliche Relevanz und generell eine Notwendigkeit für derartige Applikationen abgeleitet werden sollen.

Die Entwicklung mobiler Applikationen haben für Unternehmen mehrere Gründe. In [MA11] wurde eine Befragung zu diesem Thema durchgeführt. Es wurde untersucht, welchen Nutzen die Entwicklung einer App für den Konsumenten und für den Entwickler hat und welche Art von Applikation (Web-Applikationen oder über einen App-Store beziehbare Apps) als sinnvoll angesehen werden. Zusätzlich wurde unter Anderem erhoben, welche Herausforderungen die Befragten für die Entwicklung der benannten Applikationen sehen. Aus Entwicklersicht waren die wichtigsten Gründe für eine mobile Anwendung ein breiteres Produktportfolio (73%), eine Verbesserung der Kundenbindung (57%), ein besserer Kundenservice (57%), Akquisition von Neukunden (52%) und Kosteneinsparungen (25%). Aus Nutzersicht waren die wichtigsten Gründe für eine mobile Anwendung ein besserer Kundenservice (73%), eine Verbesserung der Kundenbindung (66%), eine Möglichkeit zur Prozessoptimierung (62%), Akquisition von Neukunden durch das Unternehmen, das die Applikation herstellt (41%), und Kosteneinsparungen (40%). Bei der Einschätzung über die zukünftige Nutzung von mobilen Anwendungen bezogen auf Web-Applikationen und über einen App-Store beziehbare B2B-Anwendungen (**B**usiness-**2**-**B**usiness), also Applikationen die primär für die Beziehungen zwischen Unternehmen genutzt werden, waren 61% der Meinung, dass Web-Applikationen stärker genutzt werden, und 31% waren von einer stärkeren Nutzung von „normalen“ Apps überzeugt. Bei den Anwendungen für den B2C-Markt (**B**usiness-**2**-**C**ustomer) waren hingegen 72% der Befragten von einer stärkeren Nutzung von Apps überzeugt, nur 22% von einer Dominanz von Web-Applikationen. Bei den Herausforderungen wurde die Sicherung des Datenschutzes, und, damit verbunden das Vertrauen der Kunden als größtes Hindernis angesehen (79%),

für 76% waren die unterschiedlichen Betriebssysteme und Hardwarekonfigurationen der mobilen Geräte die größte Herausforderung. Dahinter lagen mit 60% die Qualitätssicherung der Applikationen, mit 52% die verschiedenen Browser und damit verbundene Kompatibilitätsprobleme, sowie mit 38% die Möglichkeit der Vermarktung von mobilen Anwendungen.

Diese Befragung trifft in gleicher Weise auch für Unternehmen im Bereich der Industrieautomation zu. Jedoch sind hier auch andere Aspekte zu beleuchten. Daher wurde im Rahmen dieser Arbeit eine Befragung in Zusammenarbeit mit einem Unternehmen im Bereich der Industrieautomation durchgeführt. Die Befragung, verbunden mit einem Gewinnspiel um eine höhere Beantwortungsrate zu erhalten, wurde über einen Newsletter an über 1.000 Empfänger versandt. Es konnten 76 Antworten ausgewertet werden. Aufgrund der geringen Teilnehmeranzahl kann diese Umfrage nicht als repräsentativ gelten. Dennoch können aufgrund des Ergebnisses gewisse Tendenzen hinsichtlich der Sinnhaftigkeit von mobilen Applikationen in der Industrieautomation abgelesen werden.

Die Befragung enthielt 15 Fragen, wobei sich 11 auf mobile Applikationen bezogen, und 4 auf personenbezogene Daten. Es handelte sich bei den Befragten um 74 Männer und 2 Frauen (**Frage 12**). Wegen dieser Inhomogenität der Geschlechter in der Umfrage, wird bei den Fragen nicht zwischen Frauen und Männern unterschieden.

Es sollen der Reihe nach die Fragen analysiert werden, wobei bei einigen Fragen zuvor gemachte Annahmen überprüft werden.

Die ersten 4 Fragen bezogen sich generell auf die Nutzung von Smartphones und Tablets im betrieblichen Umfeld, sowie die verwendeten Betriebssysteme der Geräte. Die anderen Fragen, in denen keine personenbezogenen Daten erhoben wurden, waren darauf gerichtet, herauszufinden, inwieweit mobile Applikationen für verschiedene Bereiche der Industrieautomation als notwendig erachtet werden. Zusätzlich wurden die monetären Möglichkeiten für die Entwicklung erfragt.

Nach den Antworten bei **Frage 1** und **Frage 2** benutzen 75% der Befragten ein Smartphone wobei 40,8% dieses auch dienstlich nutzen. Bei den Tablets ist die Nutzung generell geringer. Hier verwenden lediglich 48,7% ein Tablet, wobei die dienstliche Nutzung mit 9,2% wesentlich geringer ausfällt. Diese Werte sind nachvollziehbar, da Smartphones auch für die Telefonie genutzt werden. Tablets sind momentan noch kein vollwertiger Ersatz für Laptops oder Desktopcomputer. Dies ist wahrscheinlich der Grund, dass die Verbreitung in diesem Bereich geringer ist.

Für dienstlich verwendete Geräte gibt es oft Vorgaben vonseiten des Unternehmens (**Frage 3**). So gaben 46% (Smartphones) bzw. 73,3% (Tablets) der Befragten an, dass es in ihrer Firma Vorgaben für die Geräte hinsichtlich des Herstellers oder des verwendeten Betriebssystems gibt. Wenn Vorgaben existieren, so sind diese bei 14,5% (Smartphones) bzw. 11,7% (Tablets) nur auf den Hersteller bezogen, bei 19,7% (Smartphones) bzw. 15% (Tablets) wird zusätzlich das Modell der Geräte vorgegeben. Diese Werte sind insofern interessant, als bei den Smartphones weniger als die Hälfte der Firmen keine Vorgaben machen. Eine mögliche Erklärung könnte hier darin liegen, dass der Fokus bei den

Dienstgeräten auf die Telefonie bezogen ist, und andere dienst- bzw. sicherheitsrelevante Arbeiten wie der E-Mail-Abruf auf dem Laptop geschieht. Bemerkenswert ist, dass auch bei Tablets Vorgaben in einigen Firmen existieren, was bedeutet, dass Tablets im operativen Bereich verwendet werden.

Frage 4 bezog sich auf die Betriebssysteme der verwendeten Geräte. In Abbildung 6.1 ist diese Verteilung ersichtlich.

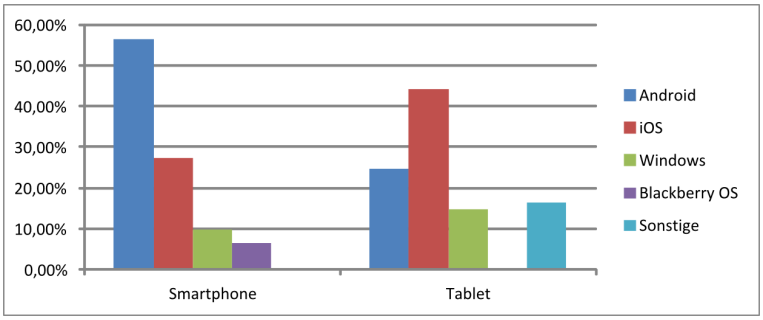

Abbildung 6.1: Verwendete Betriebssysteme bei Smartphones und Tablets

Die Verteilung bei den Smartphones verhält sich ähnlich wie in der Analyse der Verkäufe in [IDC2016], die in Kapitel 4.2 verwendet wurde, wenngleich die Verteilung zugunsten von Android hier weniger stark ausgeprägt ist. Ein gänzlich anderes Bild zeigt die Nutzung der verschiedenen Betriebssysteme bei den Tablets. Hier ist tendenziell eine stärkere Nutzung von Geräten der Firma Apple zu sehen. Interessant bei den Smartphones ist auch, dass in der Stichprobe iOS keine stärkere Verwendung im betrieblichen Umfeld findet. Die Erwartung von einer stärkeren Verwendung entsteht unter Anderem dadurch, dass Geräte, die iOS verwenden, bezüglich der Häufigkeit von Updates und längerer zeitlicher Verfügbarkeit von Updates auch bei „älteren“ Geräten, für einen Einsatz innerhalb eines Unternehmens besser geeignet scheinen.

Bei **Frage 6** wurde erhoben, von wo bereits verwendete Applikationen bezogen wurden. Es waren hier Mehrfachantworten möglich, für den Fall, dass mehrere Applikationen verwendet werden. Zusätzlich bestand noch die Möglichkeit, die Frage offen zu beantworten. Wenn schon Applikationen verwendet wurden, so wurden diese in 7,23% der Fälle selbst hergestellt, in 90,36% von anderen Firmen bezogen. Außerdem wurden 38,55% der Applikationen direkt vom Hersteller der verwendeten Komponenten geliefert. Bei einer offenen Fragebeantwortung war interessant, dass hier eher kleinere Hilfsprogramme, z.B. zur Umrechnung von Einheiten, oder der Berechnung von Kräften, als sinnvoll gesehen werden.

Auf die **Frage 7**, von wo die Applikationen bezogen werden sollten, wenn noch keine verwendet werden, antworteten 34,07%, dass kein Interesse an derartigen Programmen besteht, lediglich 4,55% wünschten eine Applikation direkt vom Komponentenhersteller. 38,64% der Befragten würden eine Fremdentwicklung der Software - diese wird nicht vom

Komponentenhersteller geliefert - bevorzugen, und 22,71% sind an einer Eigenentwicklung interessiert.

Bei **Frage 8** wurde erhoben, in wieweit die Komponenten eines bestimmten Herstellers bevorzugt verwendeten werden würden, wenn dieser auch eine gute Unterstützung der Komponenten mithilfe von mobilen Applikationen bietet. Hier antworteten 63,16% mit „Ja“. Jedoch wären nur 35,53% der Befragten bereit, auch einen Aufpreis für diese Applikationen zu bezahlen (**Frage 11**).

Im Folgenden wird die Frage nach der Nutzung von mobilen Applikationen für spezielle Aufgabengebiete behandelt, nämlich

- Schulungen im Sinne von Unterlagenbereitstellung,
- Prozessvisualisierung,
- Wartung und Diagnose wie z.B. Checklisten oder Anleitungen,
- Installation von Geräten in der Anlage, und
- Konfiguration von Geräten in der Anlage.

In Abbildung 6.2 wird dargestellt, inwieweit mobile Applikationen schon heute in den verschiedenen Bereichen Anwendung finden (**Frage 5**). Es waren hier Mehrfachantworten möglich. Interessant ist hierbei, dass diese schon jetzt in Unternehmen von 22,67% (Installation) bis 38,67% (Prozessvisualisierung) der Befragten genutzt werden. Bei dieser Frage war es unter dem Punkt „Sonstige“ möglich, eine offene Antwort zu geben. Es wurden 2 offene Antworten gegeben, von denen eine hier relevant erscheint. Es wurde darin der Unmut kundgetan, dass es schon mit den verfügbaren Desktop-Applikationen Probleme bezüglich der Kompatibilität verschiedener Versionen von Software über die Laufzeit von Industriemaschinen - mit einer Betriebszeit von 20 Jahren - gibt. Im Vergleich zu der Verwendungsdauer von mobilen Geräten von wenigen Jahren und der Aktualisierungshäufigkeit der Betriebssysteme und Applikationen kann dies ein Problem darstellen. Problematisch ist hier unter Anderem ein möglicher Verlust der Rückwärtskompatibilität - eine Verwendung von Applikationsdaten einer älteren Version in einer Aktuellen ist dann nicht mehr gegeben.

In **Frage 10** wurde erhoben, ob die Teilnehmer der Befragung mobile Applikationen als Ergänzung zu bestehenden Desktop-Programmen gerne nutzen würden. Diese Frage zielte darauf ab, dass momentan meist mit Programmen auf Laptops oder Desktopcomputern gearbeitet wird. Diese Bedienung ist die gewohnte, und es sollte hierbei nicht versucht werden, diese Programme mit allen Mitteln auf mobile Plattformen zu übertragen. Vielmehr sollten Möglichkeiten gefunden werden, unterstützend mit mobilen Applikationen anzusetzen. Für Schulungs und Ausbildungszwecke würden hier 64,47% der Befragten Anwendungen gerne nutzen, bei der Prozessvisualisierung sind es sogar 80,26%, bei der Wartung und Diagnose 77,63%, bei der Installation immerhin 61,84% und bei der Konfiguration von Komponenten 75%.

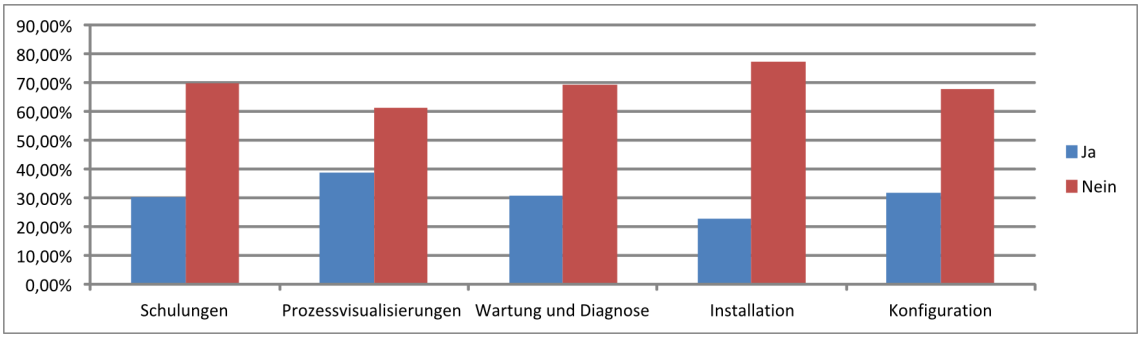

Abbildung 6.2: Verbreitung von Apps in Verwendungsbereichen

Analyse der Wichtigkeit der Verwendungsbereiche (Frage 9)

Um die zukünftige Verbreitung von Applikationen analysieren zu können, wurden die Teilnehmer darüber befragt, wie sie die Wichtigkeit mobiler Applikationen für die genannten Verwendungsbereiche einschätzen würden. Die folgenden Abbildungen zeigen die einzelnen Einsatzgebiete abhängig von Alter und Ausbildungsstand. Die Erwartung war hier, dass generell mit zunehmendem Alter die Wichtigkeit für die jeweiligen Bereiche weniger hoch erachtet würde. Diese Annahme begründet sich damit, dass die Verwendung und Verfügbarkeit von mobilen Geräten in den letzten Jahren stark zugenommen hat, jüngere Menschen schon damit aufwachsen, und dadurch eher geneigt sind, diese Applikationen zu verwenden. Außerdem wurde angenommen, dass Menschen mit höherem Bildungsstand die Wichtigkeit von mobilen Applikationen höher einschätzen, was mit dem Wissen um mögliche Synergieeffekte und Effizienzsteigerungen verbunden ist.

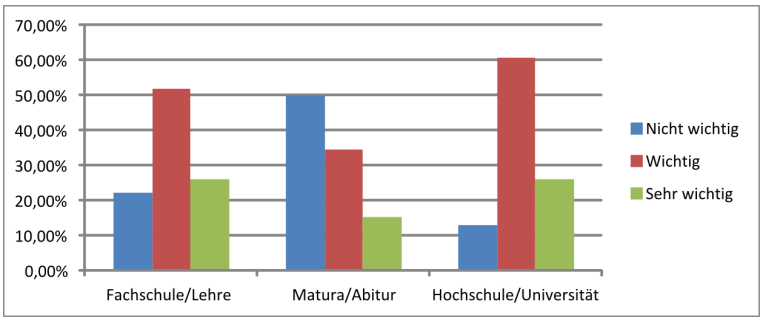

Abbildung 6.3: Apps für Schulung/Ausbildung nach Bildungsstand

Bei Applikationen für Schulungs- und Ausbildungszwecke zeigte sich, dass die Wichtigkeit für die Befragten mit Matura/Abitur am geringsten (50%) ist. Akademiker befanden die Wichtigkeit höher (86,96%) als Personen mit Lehr- oder Fachschulabschluss (77,78%). Der Vergleich nach dem Alter ergab die höchste Akzeptanz bei Personen im Alter von 20-35 (78,26%), gefolgt von den 51-65-Jährigen (76,92%). Die Altersgruppe von 36-50 hielt Applikationen für diesen Bereich am wenigsten (59,26%) für notwendig.

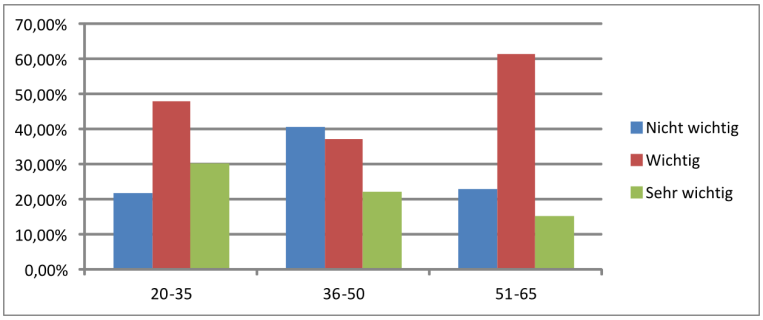

Abbildung 6.4: Apps für Schulung/Ausbildung nach Alter

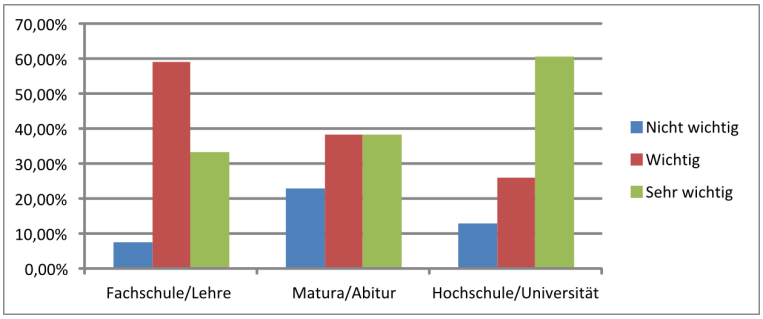

Abbildung 6.5: Apps für Prozessvisualisierung nach Bildungsstand

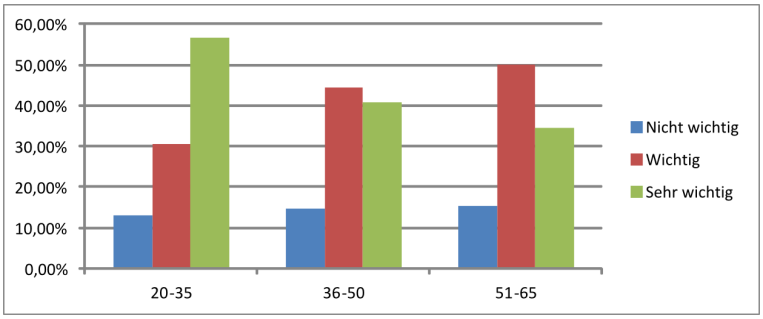

Abbildung 6.6: Apps für Prozessvisualisierung nach Alter

Bei der Prozessvisualisierung wird die Wichtigkeit von Applikationen über alle Personengruppen am höchsten (85,53%) eingeschätzt. In diesem Bereich war die Anzahl der Personen mit Lehr-/Fachschulabschluss jene Personengruppe (92,59%), die Applikationen in diesem Bereich für „wichtig“ oder „sehr wichtig“ hielten. Die Gruppe der Absolventen und Absolventinnen einer Hochschule oder Universität schätzten die Wichtigkeit mit 86,96% höher ein als Personen mit Matura/Abitur (76,92%). Bei den Altersgruppen liegt bei der Prozessvisualisierung die Gruppe der 20-35-Jährigen mit 86,96% vor den 36-50-Jährigen mit 85,19%. Die 51-65-Jährigen erachten die Notwendigkeit mit 84,62% am geringsten. Die hohe Wichtigkeit in diesem Bereich liegt möglicherweise daran, dass

in einer Industrieanlage der Überblick über die Vorgänge wichtig ist, um bei Problemen schnell eingreifen zu können. Ist die Visualisierung auf mobilen Geräten verfügbar so kann ein/e Mitarbeiter/in auch außerhalb des Leitstandes, wenn dieser beispielsweise nicht dauernd besetzt ist, schnell ein Problem erkennen, und aufgrund der Daten auch den Ort des Problems schnell finden.

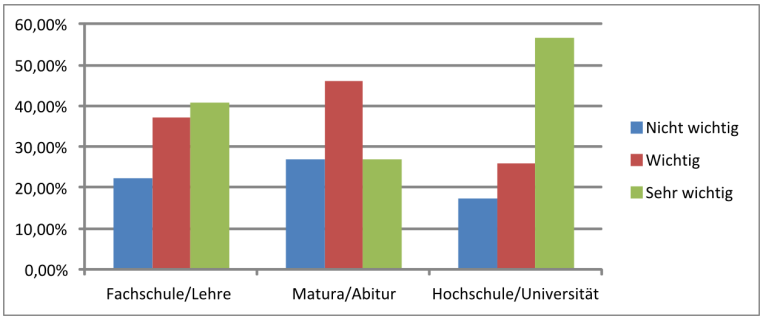

Abbildung 6.7: Apps für Wartung nach Bildungsstand

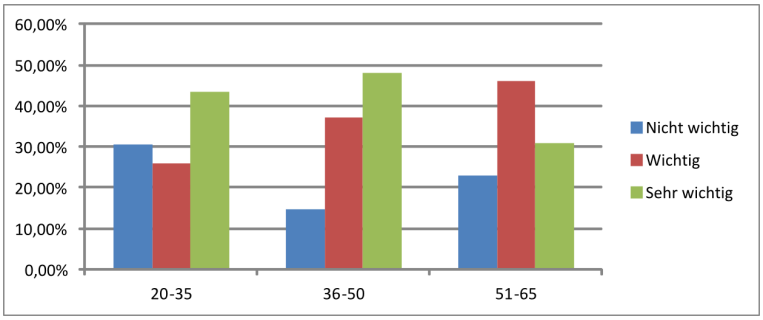

Abbildung 6.8: Apps für Wartung nach Alter

Bei der Wartung wurde die Wichtigkeit von Personen mit Universitäts- oder Hochschulabschluss mit 82,61% wieder am höchsten eingeschätzt. Die Absolventen einer Fachschule oder Lehre bewerteten die Wichtigkeit mit 77,78% höher als die Teilnehmer mit Matura/Abitur (73,08%). Bei den Altersgruppen hielten 85,19% der 36-50-Jährigen das Anwendungsgebiet der Wartung für wichtig, die 51-65-Jährigen waren zu 76,92% dieser Meinung. Mit 76,92% bei den 20-35-Jährigen war die Zustimmung hier am geringsten.

Die Verwendung von mobilen Applikationen im Bereich der Installation war die am stärksten abgelehnte, wobei aber auch hier mehr als die Hälfte der Befragten die zukünftige Verwendung als „wichtig“ oder „sehr wichtig“ ansehen. Hier waren die Personen mit Hochschul- oder Universitätsabschluss mit 73,91% noch am stärksten von der Wichtigkeit überzeugt. Von den Absolventen und Absolventinnen einer Fachschule oder Lehre empfanden 66,67% diesen Bereich als „Wichtig“ oder „Sehr Wichtig“. Bei der Personengruppe mit Matura/Abitur war die Einschätzung der Wichtigkeit mit 57,69% am geringsten. Bei den Altersgruppen waren die 36-50-Jährigen mit 70,37% am stärksten von der Wichtigkeit überzeugt, die 20-35-Jährigen belegten hier Platz zwei mit 65,22% gefolgt von den

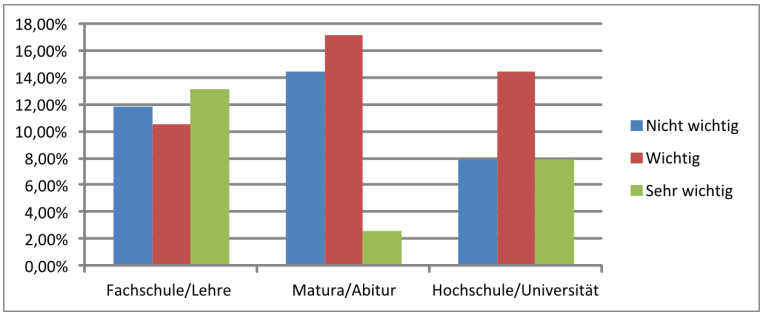

Abbildung 6.9: Apps für Installation nach Bildungsstand

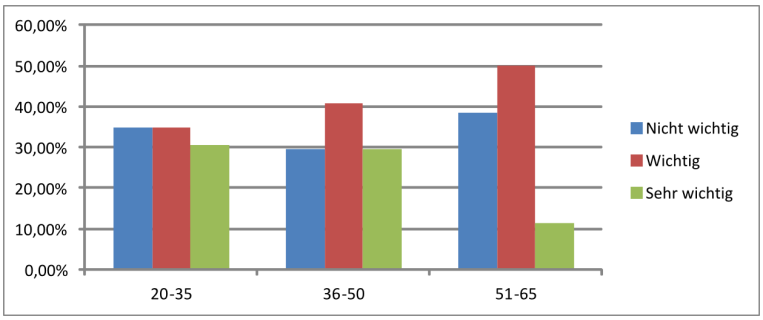

Abbildung 6.10: Apps für Installation nach Alter

51-65-Jährigen mit 61,54%. Ein möglicher Grund für diese Einschätzungen könnte der Umstand sein, dass bei der Inbetriebnahme einer Anlage noch kurzfristige Änderungen bei der Programmierung und Konfiguration der Komponenten vonnöten sind. Hier ist es sinnvoll einen Laptop zur Verfügung zu haben, da eine Bedienung mit Maus und Tastatur effizienter ist. Gleichzeitig kann in einer schmutzigen Umgebung die Bedienung von berührungsempfindlichen Bildschirmen dazu führen, dass die Informationen darauf nicht mehr gut erkennbar sind.

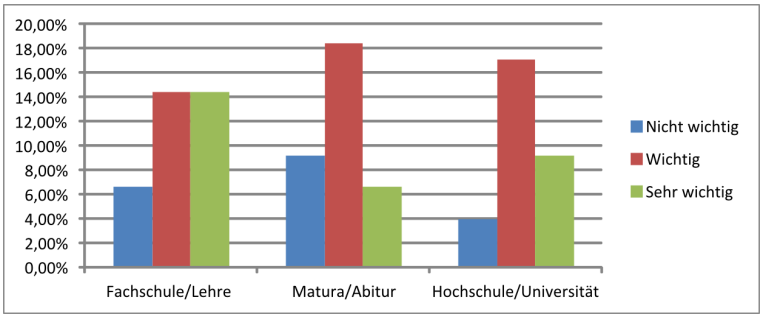

Abbildung 6.11: Apps für Konfiguration nach Bildungsstand

Bei der Konfiguration war die Zustimmung die zweithöchste hinter der Prozessvisualisierung. Auch hier war die Einschätzung, dass mobile Applikationen in diesem Bereich

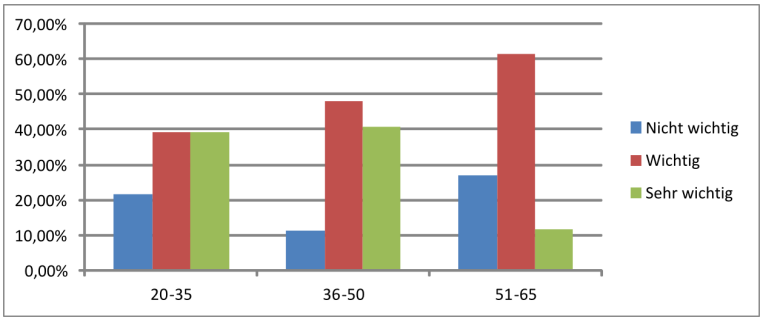

Abbildung 6.12: Apps für Konfiguration nach Alter

zukünftig eine größere Bedeutung haben werden, bei den Befragten mit Hochschul- oder Universitätsabschluss am größten mit 86,96%. Die Gruppe der Absolventen und Absolventinnen einer Lehr- oder Fachschulabschluss war zu 81,48% davon überzeugt, die Teilnehmer mit Matura oder Abitur zu 73,08%. Bei der Analyse nach dem Alter war die Gruppe von 36-50 zu 88,89% von der Zukunft von mobilen Applikationen in diesem Bereich überzeugt, in der Gruppe 20-35 Jahre waren es noch 78,26%. Bei den 51-65-Jährigen schätzen noch 73,08% der Befragten die Verbreitung in diesem Bereich als „Wichtig“ oder „Sehr Wichtig“ ein.

Personenbezogene Fragen

Die Verteilung der Befragten nach Alter und Ausbildung war in der Altersgruppe 20-35 relativ homogen (**Frage 13**, **Frage 14**). Bei den 36-50-Jährigen dominierten Hochschulabsolventen (43,48%) vor Personen mit Lehr- oder Fachschulabschluss (37,04%) und jenen mit Matura/Abitur (26,92%). In der Gruppe der 51-65-Jährigen hatten 42,31% Matura/Abitur. 33,33% dieser Gruppe besaßen einen Lehr- oder Fachschulabschluss, und mit 26,09% waren die Akademiker in dieser Gruppe am schwächsten repräsentiert. In Abbildung 6.13 ist die Verteilung graphisch dargestellt.

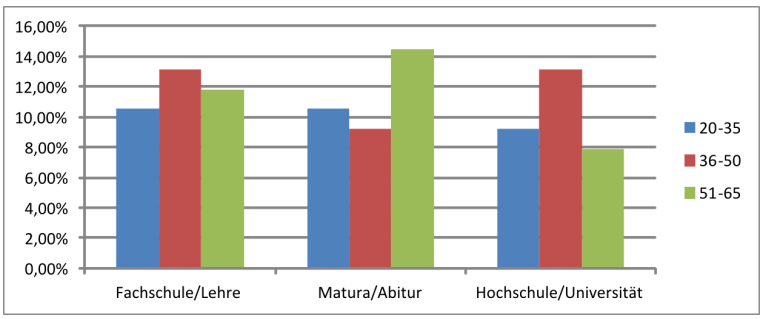

Abbildung 6.13: Verteilung nach Alter und Ausbildung

In **Frage 15** wurde erhoben, in welche Gebiete die Befragten hauptsächlich tätig sind. Zur Auswahl standen *Fertigungstechnik*, *Verfahrenstechnik* und *Prozessautomation*, sowie

die Möglichkeit der freien Beantwortung. Mehrfachnennungen waren möglich. So arbeiten 60,53% der Befragten in der Fertigungstechnik, 25% in der Verfahrenstechnik, und 48,68% in der Prozessautomation. Weiters war jeweils eine Person im Bereich des technischen Einkaufs, der Konstruktion von Sondermaschinen sowie bei Instandhaltungs- und Wartungsabteilungen und bei der Montageautomation tätig. Eine Person war zusätzlich der Geschäftsführer eines Unternehmens. Außerdem waren 2 Personen im Bereich von Schulungen tätig.

Zusammenfassung der Befragung

Zusammenfassend zeigt diese Befragung, dass mobile Applikationen zukünftig eine stärkere Rolle in den verschiedensten Gebieten der Industrieautomation spielen werden. Die Annahmen bezüglich der Einschätzung der verschiedenen Personengruppen trafen nur teilweise zu. Bezogen auf den Ausbildungsstand der Befragten waren die Befragten mit Matura/Abitur generell am wenigsten von der zukünftigen Wichtigkeit von mobilen Applikationen für die abgefragten Bereiche überzeugt. Meist hatten die Akademiker die positivste Sichtweise, wobei in der Prozessvisualisierung diejenigen mit Lehr- oder Fachschulabschluss am meisten von der Wichtigkeit überzeugt waren. Über die Gründe für der Einschätzung der Gruppe mit Matura/Abitur kann hier nur spekuliert werden. Möglicherweise haben die befragten Maturantinnen und Maturanten ihren Arbeitsplatz überwiegend in einem Büro, und befinden sich im Vergleich zu den Absolventinnen und Absolventen einer Lehre oder Fachschule seltener in Industrieanlagen oder haben weniger mit den genannten Verwendungsbereichen zu tun.

Die Annahmen zur Einschätzung der Verwendungsbereiche nach dem Alter konnten ebenfalls nicht verifiziert werden. Nur in der Gruppe der 51-65-Jährigen war, bis auf den Verwendungsbereich „Wartung“, eine durchwegs geringere Akzeptanz im Vergleich mit den beiden anderen Altersgruppen zu erkennen. Dennoch war in allen Altersgruppen zu erkennen, dass die Akzeptanz in allen Verwendungsgruppen mit über 50% gegeben war. Dies lässt darauf schließen, dass einerseits generell eine hohe Technikaffinität bei den Befragten besteht, andererseits eine große Notwendigkeit für das Erlernen des Umgangs mit neuen Technologien und deren Vorteile erkannt wurden.

Aufgrund dieser Befragung sollten mobile Applikationen entwickelt werden, die auch als Ergänzung zu bestehenden Computerprogrammen verwendet werden können, um einer möglichen Angst vor dem Verlust der gewohnten Umgebung entgegenzuwirken. Dabei ist es wichtig bestehende Applikationen konstant zu warten und bei Weiterentwicklungen eine mögliche Rückwärtskompatibilität zu erhalten. Durch den Mehrwert, den diese Applikationen bieten, kann ein besserer Service für Kunden zu Verfügung gestellt werden, wodurch eine stärkere Kundenbindung erreicht werden kann. Damit steigen auch die Chancen, den Absatz von Komponenten zu erhöhen.

KAPITEL 7

Zusammenfassung

Die zunehmende Vernetzung in allen Bereichen von Wertschöpfungsketten ist ein unbestreitbares Faktum. Die Industrieautomation bietet hier starke Potenziale, was eine Zusammenarbeit innerhalb der Schichten der Automationspyramide betrifft, und ebenso schichtübergreifend. Von der Feldebene über die Automationsebene, von der Betriebsleitebene bis zur Unternehmensebene können weitere Synergieeffekte genutzt werden, um eine intelligente Nutzung von Ressourcen zu ermöglichen. Dadurch können auch Produktionsprozesse, wie „Just in Sequence“ oder „Just in Time“, bestmöglich unterstützt werden. Jedoch darf nicht außer Acht gelassen werden, dass diese komplexen Technologien von Menschen bedient werden müssen. Daher bedarf es auch Applikationen, die es den damit arbeitenden Personen erleichtern, die Prozesse zu überwachen, Anlagen zu konfigurieren, in Betrieb zu nehmen und zu warten. Auch die Möglichkeit der Zusammenarbeit mehrerer Personen zur koordinierten gemeinsamen Erledigung von Aufgaben in einer Anlage, beispielsweise bei der Inbetriebnahme, bietet Vorteile. [PMDB07] beschreibt hier eine Konfigurationsmanagementanwendung, die es erlaubt, dass mehrere Mitglieder eines Teams gleichzeitig verschiedene Geräte einer Automationsanlage konfigurieren und parametrieren können.

In dieser Arbeit wurden mobile Applikationen in der Industrieautomation behandelt. Um die Verwendungsmöglichkeiten abbilden zu können, wurden Anwendungsfälle beschrieben, und daraus generische Szenarien erstellt, die mittels UML in Use Cases überführt wurden. Die Use Cases wurden in Haupt- und Unteranwendungsfälle gegliedert, wobei die Hauptanwendungsfälle Installation, Wartung, Schulung, und Visualisierung betrachtet wurden. In der anschließenden Analyse der Use Cases wurden Anforderungen für Datenübertragungsprotokolle, Softwarearchitektur und Softwareentwicklung abgeleitet, die in den folgenden Kapiteln zur Bewertung herangezogen wurden.

Das nächste Kapitel befasste sich mit Übertragungsprotokollen, die bei mobilen Geräten Verwendung finden. Es wurden WLAN, Bluetooth, NFC und ZigBee behandelt. Dabei wurde bei den Übertragungsverfahren neben der allgemeinen Beschreibung ein großer

Fokus auf die verwendeten Sicherheitsmechanismen gelegt. Gerade bei der Verwendung drahtloser Kommunikation wurde hier eine kritische Betrachtung als notwendig erachtet, da eine Datenaufzeichnung und Interaktion von unberechtigten Personen oder Geräten mit Netzwerken einfach geschehen kann. Eine anschließende Analyse der Protokolle auf Basis der aus den Use Cases synthetisierten Anforderungen ergab, dass eine Verwendung aller Protokolle mit gewissen Einschränkungen für einzelne Use Cases möglich ist. Diese Einschränkungen waren im Wesentlichen darauf zurückzuführen, dass Anforderungen in Bezug auf die Menge der verwendbaren Geräte, die notwendige Topologie und Reichweite, sowie die nötige Datenübertragungsrate gegeben sind.

Zukünftig wäre es denkbar, weitere Übertragungsverfahren ([LIL09], [AW03]) zu untersuchen, die momentan nicht in Standard-Smartphones Verwendung finden. Die Kommunikation kann entweder über Adapter geschehen, die über vorhandene drahtlose Mechanismen angesprochen werden ([ZLLL16], [QM15]), oder aber über Adapter, die über physische Verbindungen wie USB angesprochen werden. Weiters können über entsprechende Schnittstellen auch kabelgebundene industrielle Kommunikationsverfahren Verwendung finden, wobei hier auch die mögliche Echtzeitfähigkeit betrachtet werden muss. Auch eine tiefergehende Integration von mobilen Geräten in Automationsanlagen kann angedacht werden ([NLM13], [DA13]).

Im folgenden Kapitel wurden mögliche Entwicklungsansätze zur Erstellung von mobilen Applikationen beleuchtet. Es wurden mögliche Fallstricke erörtert, durch die Projekte scheitern können, etwa aufgrund mangelnder Qualitätsanforderungen. Um diesen und anderen Problemen entgegenzuwirken, wurden 2 allgemeine Softwareentwicklungsstrategien vorgestellt. Hierbei handelte es sich um agile und traditionelle Entwicklungsansätze. Es wurden beide Ansätze besprochen und deren Vor- und Nachteile behandelt. Es konnte gezeigt werden, dass beide Ansätze bzw. eine Kombination daraus genutzt werden können, um Projekte erfolgreich durchzuführen. Aufgrund der Anforderungen an mobile Applikationen, die meist eher als Zusatz zu bestehenden Desktop-Applikationen zur Verfügung stehen, und einem generellen Trend hin zu agilen Prozessen, bei denen auf Änderungswünsche schnell reagiert werden kann, ist jedoch ein agiler Ansatz zu bevorzugen. Aufgrund dieser Erkenntnisse wurden exemplarisch zwei Vertreter dieser Strategie besprochen, nämlich XP und Scrum. Anschließend wurden die möglichen Typen von mobilen Applikationen und deren Eigenschaften hinsichtlich Wartbarkeit, Wiederverwendbarkeit und Plattformunabhängigkeit beschrieben. Hierbei handelte es sich um „Native Applikationen“, „Web-Applikationen“, „Hybrid-Web-Applikationen“ und „Hybrid-Applikationen“. Im Weiteren wurde der Aufbau und wichtige Komponenten der gängigsten Betriebssysteme Android, iOS und Windows 10 beschrieben. Im Fall von Windows 10 konnten aufgrund der wenigen öffentlich zugänglichen Informationen nur ein sehr grober Überblick gegeben werden. Außerdem wurde ein Überblick über die für eine Entwicklung von Applikationen für diese Betriebssysteme notwendigen Entwicklungsumgebungen sowie generelle Informationen zur Veröffentlichung von Applikationen gegeben.

Eine weiterführende Analyse der Verwendung von Vorgehensmodellen für Software-

Projekte wäre sinnvoll. Insbesondere die Verwendung von bestimmten Patterns, die eine übersichtliche und wartbare Softwarearchitektur unterstützen, sollte untersucht und die Verbreitung bewertet und verbessert werden. Dies ist gerade im mobile Bereich wichtig, weil hier verschiedene Plattformen zur Verfügung stehen, die eine unterschiedliche Architektur aufweisen, und dadurch Funktionen darauf abgestimmt werden müssen. Ein Beispiel hierfür ist die Ablage von Dateien im Speicher eines Gerätes. Dies führt zu der Notwendigkeit, dass auch ein gutes Verständnis von Plattformunabhängigkeit gegeben sein muss. Ein anderer Ansatz wäre die Web-basierte Bereitstellung von Daten ([JK12]), wobei diese zentral zur Verfügung gestellt werden müssten, um dann von Mobilgeräten verwendet werden zu können.

Das danach folgende Kapitel befasste sich mit der Erstellung einer prototypischen Implementierung einer Applikation für Android, das einige Aufgaben im Bereich der Industrieautomation ermöglichen soll. Zu den Funktionen zählten ein Scan eines bestehenden Netzwerks, ein Firmwaredownload sowie ein Abruf von Diagnose- und Versionsinformationen einzelner Geräte. Es sollte zudem gezeigt werden, wie eine Wiederverwendbarkeit bestehender Softwarekomponenten bewerkstelligt werden kann. Als Framework wurde Xamarin eingesetzt, das erlaubt, Applikationen in C# zu erstellen und auf mobilen Geräten auszuführen. Bei der Implementierung der Anwendung wurde gezeigt, dass eine Wiederverwendung einer bestehenden Desktop-Anwendung auch für den mobilen Bereich möglich ist, wobei hier gewisse Einschränkungen zu berücksichtigen sind. So konnten beispielsweise nicht alle Suchprotokolle verwendet werden, weil die nativen Bibliotheken mit Xamarin nicht kompatibel waren. Aufgrund der Einschränkungen der Applikation wurde erkannt, dass mobile Applikationen für Smartphones momentan als Erweiterung bestehender Desktop-Anwendungen den Nutzerinnen und Nutzern schnell Informationen präsentieren können, und einfache Wartungsarbeiten ermöglichen. Für eine breitflächigere Anwendung in der Industrieautomation sind im Moment jedoch Desktop-Applikationen zu bevorzugen.

Für eine zukünftige Nutzung von mobilen Applikationen, die eine gemeinsame Codebasis mit Desktop-Anwendungen nutzen sollen, muss auch schon in der Entwurfsphase der Desktop-Applikation auf diese Nutzung Rücksicht genommen werden. Der Aufwand für eine Überarbeitung wäre sonst zu groß. Als Beispiel soll hier der Aufruf der Kommandos im FFT genannt werden. Hier wird der Zugriff nicht über **einen** definierten Einstiegspunkt genutzt. Vielmehr werden über die Kommandozeile Methoden des *ScanHandlerContainers* verwendet. Bei Verwendung der Applikation über die graphische Benutzeroberfläche werden teilweise direkt die *Scanhandler* verwendet. Hierbei handelte es sich um eine „gewachsene“ Codebasis, bei der, je nach Anforderungen, neue Funktionen hinzugefügt wurden. Zusätzlich werden beispielsweise plattformspezifische Bibliotheken für die Benutzeroberfläche, wie die Anzeige einer Messagebox, in allen Schichten verwendet. Hier sollten Information an die oberste Schicht (Applikationsschicht[1]) weitergereicht werden, um diese dann anzuzeigen. Es wäre auch möglich, hier Objekte einzuführen, die, über Interfaces abstrahiert, die genannten Messageboxen anzeigen können. Die Implementierung

[1] In der Applikationsschicht wird die Benutzeroberfläche implementiert.

kann dann für verschiedene Plattformen erfolgen, und dementsprechend eingebunden werden. Weiters sollten die Protokolle, die aufgrund von nativen Bibliotheken nicht verwendet werden konnten, in C# überführt werden. Dadurch könnte die Unterstützung des Produktportfolios vervollständigt werden. Die Verwendung der Sicherungs- und Wiederherstellungsfunktion kann nur bei Verfügbarkeit einer neuen Firmware der Geräte gewährleistet werden. Hier müsste es möglich sein, die Ports für die Datenübertragung über FTP ändern zu können.

In der nachfolgenden wirtschaftlichen Analyse wurde untersucht, inwiefern mobile Applikationen in der Industrie Verwendung finden, bzw. wie allgemein die Notwendigkeit von mobilen Applikationen gesehen wird. Dazu wurde unter anderem eine Befragung von Personen durchgeführt, die im Automations-Kontext arbeiten. Es zeigte sich, dass die Akzeptanz für diese Art von Applikationen gegeben ist, unabhängig von Alter und Ausbildungsstand. Auch wurde festgestellt, dass mobile Applikationen eher als Ergänzung zu bestehenden Desktop-Anwendungen gesehen werden, denn als Substitute. Es konnte auch abgeleitet werden, dass eine Monetarisierung momentan schwierig ist.

Mobile Applikationen werden zukünftig auch in der Industrieautomation eine immer stärkere Rolle spielen. Die momentan verfügbaren Applikationen stellen meist Informationen bereit, die für den Erwerb von Komponenten gedacht sind, oder Hilfestellungen für Auslegungen von Komponenten bieten. Ein Eingriff oder Datenabruf von Geräten ist nur sehr eingeschränkt möglich. Das Ziel sollte es daher sein, Entwicklungen in diesem Bereich zu forcieren, um Kundinnen und Kunden bestmöglich zu unterstützen, auch bei Aufgabe abseits der zuvor genannten Arbeiten.

Literaturverzeichnis

[NAS04] Navab, N. *„Developing Killer Apps for Industrial Augmented Reality.“ Computer Graphics and Applications, IEEE, vol. 24, no. 3, pp. 16 - 20*, 2004

[TAZ] Clark, Sarah *„TazTag unveils first Android phone with NFC and ZigBee„, https://www.nfcworld.com/2012/02/26/313800/taztag-unveils-first-android-phone-with-nfc-and-zigbee/*, zuletzt abgerufen 26.02.2017

[MP10] Mahmoud, Qusay H. ; Popowicz, P. *„A Mobile Application Development Approach to Teaching Introductory Programming.“ Frontiers in Education Conference (FIE), IEEE, pps. T4F-1 - T4F-6,* Oct. 2010

[SKD11] Sauter, T.; Soucek, S.; Kastner, W.; Dietrich, D.; *„The Evolution of Factory and Building Automation“, Industrial Electronics Magazine, IEEE , vol.5, no.3, pp.35- 48*, Sept. 2011

[YS11] Yaling Luo; Shiqiang Zhang, *„Current Situation and Trends of Industrial Control Configuration Software.“ IEEE International Conference on Automation and Logistics (ICAL), pps. 309 - 313* Aug 2011

[FES12] Festo, Bediengeräte, *„FED Datenblatt“ , www.festo.com*, 2011/12

[RM10] Rojas, C.; Morell, P.; *„Guidelines for Industrial Ethernet infrastructure implementation: A control engineer's guide“, 2010 IEEE-IAS/PCA 52nd Cement Industry Technical Conference, IEEE, pp.1-18*, März 2010

[MD06] Malek-Zadeh, A. ; Dietsch, H. *„Web-Based Information Technology for Remote Handling of Electrical Drives and Control Devices: An Integrated Approach.“International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies. ICN/ICONS/MCL 2006.* Apr. 2006

[TASF11] Takagi, M. ; Arata, J. ; Sano, A. ; Fujimoto, H. *„A new encounter type haptic device with an actively driven pen-tablet LCD panel.“ IEEE*

International Conference on Robotics and Biomimetics (ROBIO), pps. 2453 - 2458, Dec. 2011

[ZPU11] Ziegler, J.; Pfeffer, J.; Urbas, L. *„A Mobile System for Industrial Maintenance Support based on Embodied Interaction.“ Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction, TEI '11, ACM, pps. 181 - 188* Jan. 2011

[LHLW08] Lawo M.; Herzog O.; Lukowicz, P.; Witt, H. *„Using Wearable Computing Solutions in Real-World Applications.“ CHI '08 Extended Abstracts on Human Factors in Computing Systems, ACM, pps. 3687 - 3692* Apr. 2008

[AS10] Aleksy M.; Stieger B.*„Supporting Service Processes with Semantic Mobile Applications.“ Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia, ACM, pps. 167 - 172* Nov. 2010

[MO16] Abramivici, Michael; Herzog, Otthein; *„Engineering im Umfeld von Industrie 4.0“, http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/* zuletzt abgerufen 15.03.2017

[IDC2016] ICD, Analyse der Verkaufzahlen von Smartphonebetriebssystemen 4. Quartal 2015 bis 3. Quartal 2016, http://www.idc.com/promo/smartphone-market-share/os, abgerufen am 29.12.2016.

[HWS2003] Helmuth, Christian; Westfeld, Andreas; Sobirey, Michael (2003): μSINA – Eine mikrokernbasierte Systemarchitektur für sichere Systemkomponenten, 8. Deutscher IT-Sicherheitskongress des BSI 2003, Bonn Bad-Godesberg

[APP13] Apple, Entwicklerseite der Firma Apple, https://developer.apple.com/programs/, aberufen am 29.07.2013.

[APP17] Apple, *Dokumentation zu iOS, https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/i* abgerufen am 09.02.2017

[GOO13] Google, Entwicklerseite für Android, https://developer.android.com/index.html, abgerufen am 29.7.2013

[WIN13] Microsoft, Entwicklerseite für Windows Apps, https://developer.microsoft.com/en-us/windows/apps/getstarted, abgerufen am 07.01.2017

[AND16] Google; *Entwicklerseite für Android, https://developer.android.com/guide/platform/index.html*, abgerufen am 29.12.2016

[AND17] Google; *Developer Console Help, https://support.google.com/googleplay/android-developer/answer/112622?hl=en*, abgerufen am 07.01.2017

[KS15] Google; *„Android Architecture and Related Security Risks“, Asian Journal of Technology & Management Research Vol. 05 – Issue: 02*, Jun-Dec 2015

[OA16] Khronos Group; *„OpenMAX AL“, https://www.khronos.org/openmax/al/*, abgerufen am 29.12.2016

[YB15] Yadav, Radhakishan; Bhadoria, Robin Singh;*„Performance Analysis for Android Runtime Environment“, 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT)*, April 2015

[S14] *„Response to Article: Architectural patterns for Mobile Application Development“, https://blog.inf.ed.ac.uk/sapm/2014/02/26/response-to-article-architectural-patterns-for-mobile-application-development-by-s1014475/*, abgerufen am 30.12.2016

[WIN10] *„Architecture of Windows 10“, https://social.technet.microsoft.com/wiki/contents/articles/310 of-windows-10.aspx*, abgerufen am 07.01.2017

[CUN09] Carlo U. Nicola*„Einblick in die Dalvik Virtual Machine.“ IMVS Fokus Report, pps. 5 - 12,* Jan. 09

[AND2012] Mike Bach; *„Mobile Anwendungen mit Android“ Addison-Wesley Deutschland*, 2012

[PROT2012] Martin Sauter; *„Grundkurs Mobile Kommunikationssysteme“, 5., überarb. und erw. Aufl.* Springer Fachmedien Wiesbaden, Deutschland 2013

[TEBM09] Tews, Erik; Beck, Martin; *„Practical attacks against WEP and WPA“ Proceedings of the second ACM conference on Wireless network security, pp. 79 - 86,* März 2009

[IEEE2] IEEE Computer Society, *„IEEE Standard 802.11“,* IEEE, März 2012

[VIE11] Viehböck, Stefan; *„Brute forcing Wi-Fi Protected Setup“, http://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf, Vers. 3, 26.12.2011,* zuletzt abgerufen am 12.10.2013

[ShWo05] Shaked, Yaniv; Wool, Avishai; *„Cracking the Bluetooth PIN“, Proceedings of the 3rd int. conf. on Mobile systems, applications, and services, pp. 39 - 50*, Juni 2005

[BLUEHP] SIG; *„Bluetooth Basics“, http://www.bluetooth.com/Pages/Basics.aspx,* zuletzt abgerufen am 13.10.2013

[BLUEAMP] SIG; *„Pressemitteilung zu Bluetooth HS(AMP)“, http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx?ItemID=16,* zuletzt abgerufen am 15.08.2015

[BLUESTD] SIG; *„Bluetooth Core V4.2“, http://www.bluetooth.com/Pages/Basics-.aspx,* Dezember 2014

[TLLL09] Kuo-Chang Ting, Hung-Chang Lee, Hsiu-Hui Lee, Feipei Lai, *„An Idle Listening-aware Energy Efficient Scheme for the DCF of 802.11n“, IEEE transactions on Consumer Electronics , Vol. 55, No. 2, pps. 447 - 454,* Mai 2009

[HOL13] Philipp Hold, *„Industrie 4.0 - Das vernetzte Wertschöpfungssystem der Zukunft“, http://www.fraunhofer.at/de/presse/pressearchiv/ka_industrie_4.html,* zuletzt abgerufen am 03.10.2013

[ZIGBEESTD] ZigBee Standards Organization; *„ZigBee Specification“*, September 2012

[WPAN03] IEEE; *„IEEE Std 802.15.4“* Oktober 2003

[EY10] Yüksel, Ender; *„Analysing ZigBee Key Establishment Protocols“, https://arxiv.org/pdf/1205.6678.pdf*, abgerufen am 03. August 2016

[TZ15] Zillner, Tobias; *„ZigBee Exploited“, Black Hat USA 2015*, August 2015

[DHTS13] Dementyev, Artem; Hodges, Steve; Taylor, Stuart; Smith, Joshua; *„Power Consumption Analysis of Bluetooth Low Energy, ZigBee and ANT Sensor Nodes in a Cyclic Sleep Scenario“, Wireless Symposium (IWS), 2013 IEEE International*, April 2013

[JLMR10] Langer, Josef; Roland, Michael; *„Anwendungen und Technik von Near Field Communication (NFC)“*, Springer-Verlag 2010

[NFCIP-1] ECMA International; *„ECMA-340 Standard, NFCIP-1“, 3rd Edition,* Juni 2013

[NFCIP-2] ECMA International; *„ECMA-352 Standard, NFCIP-2“, 3rd Edition,* Juni 2013

[NFCIP-3] ECMA International; *„ECMA-385 Standard, NFC-SEC: NFCIP-1 Security Services and Protocol“, 3rd Edition* Juni 2013

[NFCIP-4] ECMA International; *„ECMA-386 Standard, NFC-SEC-01: NFC-SEC Cryptography Standard using ECDH and AES “, 2nd Edition* Juni 2010

[NFCIP-5] ECMA International; *„ECMA-410 Standard, NFC-SEC-03: NFC-SEC Entity Authentication and Key Agreement using Asymmetric Cryptography “, 2nd Edition* Juni 2015

[NFCIP-6] ECMA International; *„ECMA-411 Standard, NFC-SEC-04: NFC-SEC Entity Authentication and Key Agreement using Symmetric Cryptography “, 2nd Edition* Juni 2015

[NFCIP-7] ECMA International; *„ECMA-409 Standard, NFC-SEC-02: NFC-SEC Cryptography Standard using ECDH-256 and AES-GCM“, 2nd Edition* Juni 2015

[NFCSEC06] Haselsteiner, Ernst; Breitfuß, Klemens; *„Security in Near Field Communication (NFC), Strenths and Weaknesses“, RFIDSec,* 2006

[NFCSEC13] Lee, Yun-Seok; Kim, Eun; Jung, Min-Soo *„A NFC based Authentication method for defense of the Man in the Middle Attack“, 3rd Int. Conference on Computer Science and Information Technologie (ICCSET'2013,* Jan 2013

[AIW10] Anthony I. Wasserman, *„Software Engineering Issues for Mobile Application Development.“ FoSER '10 Proceedings of the FSE/SDP workshop on Future of software engineering research, ACM, pp. 397 - 400*, 2010

[MAA05] M.A.Awad; *„A Comparison between Agile and Traditional Software Developement Methodologies“, http://www.unf.edu/ broggio/-cen6940/ComparisonAgileTraditional.pdf*, 2005, zuletzt abgerufen am 15.10.2016

[ACJH02] Cockburn, Alister; Highsmith, Jim; *„Agile software development, the people factor“, IEEE Computer*, Nov. 2001

[AGM] *„Manifesto for Agile Software Developement“, http://agilemanifesto.org/*, 2001, zuletzt abgerufen am 23.10.2016

[LWAC03] Williams, Laurie; Cockburn, Alister; *„Agile Software Development: It's about Feedback and Change“, IEEE Computer*, Juni 2003

[CR16] Choudhary, Bharat; Rakesh, Shanu K; *„An Approach using Agile Method for Software Development“, 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, Feb. 2016

[RA15] Rauf, Abdul; AlGhafees, Mohammed; *„Gap Analysis between State of Practice and State of Art Practices in Agile Software Development“, 2015 Agile Conference* Aug. 2015

[BF12] Franz, Benedikt; *„eXtreme Programming“, http://dme.rwth-aachen.de/en/system/files/file_upload/course/12/proseminar-methodenundwerkzeuge/camerareadyextremeprogramming.pdf* Juni 2012

[BAG05] Beck, Kent; Andres, Cynthia; *„Extreme Programming Explained: Embrace Change“*, Addison-Wesley 2005

[SF13] Schwaber, Ken; Sutherland, Jeff *„Der Scrum Guide“, http://www.scrumguides.org/docs/scrumguide/v1/ScrumGuide-DE.pdf*, Juli 2013, abgerufen am 28. Dezember 2016

[WIS16] *„What is Scrum?“, https://www.scrum.org/Resources/WhatisScrum*, abgerufen am 28. Dezember 2016

[XX13] Xanthopoulos, Spyros; Xinogalos, Stelios; *„A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications“*, BCI ’13 Proceedings of the 6th Balkan Conference in Informatics, Sept. 2013

[OT12] Ohrt, Julian; Turau, Volker; *„Cross-Platform Development Tools for Smartphone Applications.“ Computer, IEEE, vol. 45 no.9, pps. 72 - 79,* Sept. 2012

[EAY16] El-Kassas, Wafaa S.; Abdullah, Bassem A.; Yousef, Ahmed H.; Wahba, Ayman M.; *„Enhanced Code Conversion Approach for the Integrated Cross-Platform Mobile Development (ICPMD)“, IEEE Transactions on Software Engineering, vol. 42, no. 11, pp. 1036 - 1053*

[PG17] Adobe PhoneGap; *http://phonegap.com/*, zuletzt abgerufen am 07.01.2017

[XA17] Xamarin; *https://www.xamarin.com/*, zuletzt abgerufen am 07.01.2017

[MA11] bitkom; *„Apps - Mobile Anwendungen in der ITK Branche“, https://www.bitkom.org/Bitkom/Publikationen/Apps-Mobile-Anwendungen-in-der-ITK-Branche.html*, abgerufen am 4. Februar 2017

[MOMA13] Mono; *„Mono Migration Analyzer“ http://www.mono-project.com/MoMA*, abgerufen am 15.08.2013

[AW03] Willig, Andreas; *„An Architecture for Wireless Extension of PROFIBUS“, Industrial Electronics Society, 2003. IECON '03*, Nov. 2003

[ZLLL16] Zhou, Yuan; Li, Xiaokun; Liu, Mingshan; Lin, Fengxue; *„Research and development of android client for PROFIBUS-DP based on Wi-Fi“, 2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, April 2016

[LIL09] Lily L.; *„60GHz: Opportunity for Gigabit WPAN and WLAN Convergence.“ ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 56 - 61*, Jan. 2009

[PMDB07] Pantoni, R. P.; Mossin, E. A.; Donaires, O. S.; Brandao D. *„Configuration Management for Fieldbus Automation Systems.“ IEEE International Symposium on Industrial Electronics, ISIE, pps. 1844 - 1848,* Jun. 2007

[NLM13] Nicolae, Maximilian; Lucaci, Laurentio; Moise, Ilona; *„Embedding Android Devices in automation systems“, International Symposium for Design and Technology in Electronic Packaging (SIITME)*, Okt. 2013

[DA13] Drumea, Andrei; *„Control of Industrial Systems Using Android-Based Devices„, International Spring Seminar on Electronics Technology (ISSE)*, Mai 2013

[QM15] Qiao, Bo; Ma, Kaixue; *„An Enhancement of the ZigBee Wireless Sensor Network Using Bluetooth for Industrial Field Measurement„, IEEE MTT-S International Microwave Workshop Series on Advanced Materials and Processes for RF and THz Applications (IMWS-AMP)*, Juli 2015

[JK12] Koelsch, James R.; *„Web-based SCADA Gathers More Fans“, https://www.automationworld.com/scada/web-based-scada-gathers-more-fans*, zuletzt abgerufen am 06.04.2017