Technische Universität München
Lehrstuhl für Medientechnik

Dynamische Generierung von sicheren Benutzerschnittstellen in verteilten Systemen

Dipl.-Ing. Univ. Michael Eichhorn

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. Dr.-Ing. habil. Erwin Biebl
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Eckehard Steinbach
	2.	Univ.-Prof. Dr. sc. techn. Andreas Herkersdorf

Die Dissertation wurde am 19.04.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 21.10.2011 angenommen.

Dynamische Generierung von sicheren Benutzerschnittstellen in verteilten Systemen

Dipl.-Ing. Univ. Michael Eichhorn

21. Januar 2012

Für Milena, Emily und Laurence.

Danksagung

Der deutsche Sänger Xavier Naidoo hat in einem seiner Lieder die Phrase „Dieser Weg wird kein leichter sein. Dieser Weg wird steinig und schwer.“ („Dieser Weg“, 2005) gesungen. Obwohl dieses Lied vielleicht in einem anderen Zusammenhang spielte, spiegeln sich manche Phasen meiner Arbeit darin. Ohne die Unterstützung zahlreicher Personen wäre es mir vermutlich nicht gelungen, diese Arbeit während meiner Tätigkeit am Lehrstuhl für Medientechnik an der Technischen Universität München erfolgreich abzuschließen. Ich möchte die Möglichkeit nutzen und diesen hiermit danken.

Als allererstes möchte ich meinen tiefsten Dank an Prof. Dr.-Ing. Eckehard Steinbach aussprechen, der es mir ermöglichte an seinem Lehrstuhl zu forschen. Seine motivierende, unterstützende und klardenkende Art machte die Arbeit zu einer Freude und inspirierte mich immer wieder von neuem.
Desweitern möchte ich mich bei Hans-Ulrich Michel von der BMW Forschung und Technik GmbH und Prof. Dr. sc. techn. Andreas Herkersdorf bedanken. Diese beiden leiteten ein Industrieprojekt bei dem ich teilnehmen durfte. Die Arbeit in diesem Projekt war hilfreich für meine Forschung und ein lehrreiches Erlebnis. Für die erfolgreiche Zusammenarbeit mit den Kollegen des Lehrstuhls für Datenverarbeitung, des Fachgebiets Höchstfrequenztechnik, des Lehrstuhls für integrierte Systeme, des Lehrstuhls für Kommunikationsnetze und des Lehrstuhls für Realzeitsysteme bin ich ebenfalls sehr dankbar.
Prof. Dr. sc. techn. Andreas Herkersdorf möchte ich auch für die Übernahme des Zweitgutachtens und Prof. Dr.-Ing. Dr.-Ing. habil. Erwin Biebl für den Vorsitz danken.
Hervorheben möchte ich die Zusammenarbeit mit Martin Pfannenstein während der letzten Phase meiner Arbeit. Durch diese ergaben sich äußerst fruchtbare Diskussionen, die mich bei der erfolgreichen Beendigung meiner Dissertation sehr unterstützt haben. Ihm und meinem Bürokollegen Quirin Hofstätter, dem ich viele unterhaltsame Stunden im Büro verdanke, bin ich auch für die vermutlich zeitintensiven Korrekturarbeiten sehr dankbar.
Des Weiteren richte ich meinen Dank an Dr. Martin Maier, Bernd Müller-Rathgeber, Robert Nagel, Florian Schweiger, Georg Schroth und alle weiteren Kollegen und ehemaligen Studenten am Lehrstuhl für Medientechnik und Lehrstuhl für Kommunikationsnetze für ihr stets offenes Ohr und ihre Unterstützung.
Meine aufrichtige Dankbarkeit richtet sich auch an meine Freunde und Familie. So haben zahlreiche Gespräche mit meinem Studienkollegen und Freund Christian Roman mir immer wieder die Kraft gegeben weiterzumachen. Dafür bedanke ich mich herzlich bei ihm.
Schlussendlich bin ich zutiefst meiner Frau Milena für ihre Unterstützung und Geduld, jedoch vorallem für unsere beiden wundervollen Kinder Emily und Laurence, dankbar. Sie akzeptierte, dass ich viel Zeit mit der Forschung verbracht habe und oftmals lange Abende im Büro blieb. Ohne ihren Rückhalt und ihre Liebe hätte ich diese Arbeit nicht erfolgreich beenden können.

München, Februar 2011 *Michael Eichhorn*

Kurzfassung

Diese Arbeit beschäftigt sich mit Konzepten und Lösungen für eine zukünftige Interaktion zwischen Benutzern und Geräten, sowie zwischen Geräten in einem verteilten System.

Die Interaktion zwischen Menschen und Geräten nimmt im Alltag stetig zu, wodurch sich neue Anforderungen an die Mensch-Maschine-Schnittstelle (MMS) ergeben. Insbesondere die kooperative Nutzung eines Systems führt bei bisherigen Ansätzen zu einer signifikanten Komplexitätssteigerung. Diese wirkt sich sowohl im Systemdesign als auch in der Systementwicklung negativ aus. Viele Systeme mit einer MMS sind bereits so komplex geworden, dass sie verteilt organisiert werden. Das heißt, es interagieren viele einzelne Teilkomponenten über ein (gemeinsames) Netz miteinander. Durch immer höhere Funktionsumfänge wird auch dieser, für den Nutzer nicht sichtbare, Organisationsaufwand unüberschaubarer.

Dieser gesteigerten Komplexität soll durch diese Arbeit begegnet werden indem die MMS durch ein generisches Konzept vereinheitlicht

und damit vereinfacht wird. Durch die Diskussion von verschiedenen Einsatzszenarien des Konzepts zeigen sich die universellen Einsatzmöglichkeiten dieser Vereinfachung. Diese setzt sich aus einer Lösung für die dynamische Generierung der Benutzerschnittstelle und einer selbstorganisierenden und dienstebasierten Verwaltung der Funktionalitäten, die im Netz zur Verfügung stehen, zusammen. Daher wird in dieser Arbeit ein Konzept für eine zukünftige, einheitliche Strukturierung der Komponenten eines verteilten MMS-Systems vorgeschlagen. Das System sammelt über vereinheitlichte Beschreibungen die verfügbaren Funktionalitäten, bereitet diese auf und bietet sie dem Benutzer strukturiert in Form einer grafischen Repräsentation an. Die aufgezeigte Lösung verwendet als Grundlage weitverbreitete Techniken und Standards wie z.B. das Internet Protocol (IP). Dadurch bezieht dieser Ansatz auch Geräte ein, die zum Entwicklungszeitpunkt noch nicht verfügbar bzw. bekannt sind, sowie persönliche Geräte von Benutzern, mit denen diese dann mit dem MMS-System interagieren können.

Gerade durch den Einsatz von noch unbekannten bzw. sich dynamisch integrierenden Geräten ergibt sich ein Risiko durch beabsichtigte oder unbeabsichtigte Angriffe auf Informationen oder Ressourcen der Architektur. Aus diesem Grund ist die Informationssicherheit innerhalb des vorgestellten MMS-Systems ein weiterer untersuchter Aspekt. Den Sicherheitsrisiken wird in dieser Arbeit durch geeignete Mechanismen begegnet, ohne eine Verringerung des Komforts oder der Flexibilität des Systems in Kauf zu nehmen müssen.

Abstract

This dissertation proposes concepts and solutions for a future interaction between users and devices as well as among devices in a distributed system.

The everyday interaction between humans and devices is constantly increasing, which leads to novel requirements for the design of Human-Machine-Interfaces (HMI). Especially the cooperative use of one system introduces a significant raise of complexity when regarding state-of-the-art systems. This affects the system design as well as the development negatively. Many HMI-enabled systems have become so complex, that they are organized in a distributed manner which means that many components interact together (over a common network) to compose the overall system. As the number of features offered by these systems also increases, the effort to manage all of these components and features, which usually is invisible for the user, is also significantly enhanced.

This growth of complexity is addressed within this dissertation by introducing a generic concept which unifies and eases the HMI system. Different scenarios to which the proposed system can be applied are discussed. The presented scenarios show that the idea is universally applicable. The proposed system consists of an approach to dynamically generate the graphical user interface as well as of a self-organized, service-based management of functionalities provided within the infrastructure.

Hence, a concept for a future unified structuring of the HMI's components is presented. The system gathers the available functionalities which are uniformly described, preprocesses them and offers them to the user by means of a graphical representation. The proposed system is based on well-proven technologies and standards as, e.g., the Internet Protocol (IP). By that, the system design also incorporates devices which are not known during the development process. Furthermore, the user's personal devices can be used to interact with the HMI system.

By taking into account also future devices that are not known up to now, respectively designing an open system, concerns about the security of the system have to be addressed. The security of resources and information can be affected by malicious attacks as well as by misconfiguration of system components. Therefore, this dissertation also addresses security issues by not impairing the usability and comfort at the user side.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen und Akronyme

AAA	Authentifizierung, Autorisierung und Abrechnung
DPWS	Device Profile for Web Services
XML	Extensible Markup Language
GUI	Grafische Benutzeroberfläche (engl. Graphical User Interface)
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
MMS	Mensch-Maschine-Schnittstelle
SOA	Serviceorientierte Architektur
ITU-T	Telecommunication Standardization Sector
TLS/SSL	Transport Layer Security/Secure Sockets Layer
WS	Web Service
WWW	World Wide Web
CA	Zertifizierungsstelle (engl. Certificate Authority)

1. Einführung

Der Alltag in Industrieländern wird heute zunehmend durch die **Interaktion** mit **Maschinen** bestimmt, sei es um den **Nutzer** bei der täglichen Arbeit zu unterstützen oder ihm wichtige **Informationen**, meist maschinell aufbereitet, zukommen zu lassen.

Der Funktionsumfang von Geräten und Maschinen wird dabei immer **umfangreicher** und **komplexer**. Durch die zunehmende Funktionsvielfalt elektronischer Geräte steigen die Anforderungen an die Bedienschnittstelle bzw. die **Mensch-Maschine-Schnittstelle (MMS)**.

Ziel ist es dabei, die allgegenwärtige und immer weiter **ansteigende Informationsflut** zu strukturieren. Da hochgradig vernetzte Geräte selten autonom arbeiten, werden sie auch mit immer komplexeren Schnittstellen ausgestattet. Diese Schnittstellen können nun zum einen für die Kommunikation mit anderen Geräten spezifiziert sein, zum anderen aber auch für den Menschen. Diese Mensch-Maschine-Interaktionen können sehr vielschichtig oder auch sehr einfach realisiert sein. Sie umfassen Eingabe- wie auch Ausgabegeräte, wobei Eingabegeräte auch gleichzeitig Ausgabegeräte sein können und umgekehrt.

Eine Schwierigkeit ist es dabei, dem Menschen **möglichst einfach** auch komplexe Schnittstellen zur Verfügung zu stellen, um den gegebenen Funktionsumfang von Geräten abbilden zu können. Gerade die einheitliche Darstellung unterschiedlichster Funktionen in einem Gesamtsystem stellt dabei eine besondere Herausforderung in puncto **Effizienz und Flexibilität** dar. Dabei kann es sich beispielsweise um ein Fahrzeug, Informations- und Entertainmentsysteme in öffentlichen Verkehrsmitteln wie Flugzeugen oder Zügen oder auch um Bedienschnittstellen zur Heimautomatisierung handeln.

Je größer das Gesamtsystem, desto stärker muss auf die **Skalierbarkeit** des MMS-Systems geachtet werden. Dies gilt besonders, wenn viele Benutzer (gleichzeitig) mit einem vernetzten System wie beispielsweise dem Informationssystem eines Hotels oder eines Krankenhauses interagieren möchten. Ebenso kritisch sind die Kosten, die zum einen durch die (einmaligen) Entwicklungskosten eines solchen Systems entstehen oder beispielsweise durch die **Integration von neuen Diensten** in ein bereits bestehendes System, als auch die laufenden Kosten, wie der Wartung des Systems.

Ein weiteres Ziel dabei ist es, ein System für eine solche Interaktion möglichst allgemein, einfach und flexibel zu gestalten, damit eine **nahtlose Integration** von weiteren Funktionen gewährleistet bleibt.

Das in dieser Arbeit erstellte und untersuchte Konzept hat sich zur Aufgabe gemacht, der wachsenden Funktionsvielfalt und dem damit verbundenen gesteigerten grafischen Darstellungsumfang durch **dynamische Integration von Diensten, effizienter Datenübertragung und flexibleren Darstellungsformen** zu begegnen.

1.1. Motivation

Elektronische Gegenstände im Alltag werden immer stärker vernetzt. Durch das Fehlen von verbreiteten Standards steht diese Vernetzung jedoch noch am Anfang und somit ist auch die Interaktion mit dem Nutzer noch recht eingeschränkt. Es existieren erste Ansätze, aber diese werden noch lange nicht allgegenwärtig genutzt. Sie funktionieren bestenfalls nur innerhalb von Geräteklassen und auf Geräteebene.

Ein Beispiel hierfür wäre die drahtlose Steuerung eines Musikabspielers in einer Wohnung von einem Mobiltelefon aus. Dazu ist ein spezielles Client-Programm auf dem Mobiltelefon ebenso notwendig wie das Gegenstück dazu auf einem Server, der die eigentliche Musikansteuerung übernimmt. Somit kann nun beispielsweise die Lautstärke gesetzt werden, eine Wiedergabeliste erstellt oder die Wiedergabe angehalten und wieder fortgesetzt werden. Soll nun das Netz durch ein weiteres Gerät erweitert werden, muss dieses erneut für die Steuerung der Musik eingerichtet bzw. konfiguriert werden. Anschließend könnte es dann über das Client-Programm mit dem Server kommunizieren und ebenfalls die Musikanlage steuern. Das Client-Programm ist dabei speziell für das jeweilige Endgerät (in diesem Fall das Mobiltelefon) entwickelt und muss für jede weitere Hardware angepasst werden.
Um diesen Vorgang zu erleichtern und zu vereinfachen, wurden in den letzten Jahren viele verschiedene Ansätze vorgeschlagen. Die bisherigen Vorschläge beruhen immer auf dem Prinzip einer stark für das entsprechende Gerät angepassten Oberfläche mit zum Teil sogar angepasster Hardware für die Ein- und Ausgabe. Als Beispiel für diese Ansätze könnte man einen Fernseher nennen. Dieser besitzt eine Fernbedienung mit der er sich steuern lässt. Diese ist auf das jeweilige TV-Gerät abgestimmt und lässt sich in der Regel nicht mit weiterer Unterhaltungselektronik, wie dem DVD-Player, kombinieren.

Mit dem in dieser Arbeit vorgestellten Ansatz wird die Plattformabhängigkeit auf Endgeräten aufgelöst. Es ist nicht mehr relevant, um welches Endgerät es sich handelt, da Standardsoftware zur Darstellung verwendet wird, deren problemlose Installation auf heutigen Geräten vorausgesetzt werden kann oder bereits auf den Geräten vorhanden ist.

Der Fokus dieser Arbeit liegt auf der Darstellung durch grafische Benutzerschnittstellen. Beispielsweise wäre dies die Integration des zuvor erwähnten Mobiltelefons ohne Installation von Zusatzsoftware. Trotzdem würde sich ein beliebiger Musikabspieler über eine grafische Schnittstelle ansteuern lassen. Durch den vorgestellten Ansatz verschwimmen die Grenzen zwischen den Funktionsanbietern und -konsumenten, sodass nicht mehr entscheidend ist, wer welche Rolle übernimmt und sogar beliebige Funktionsketten durch eine Aneinanderreihung einzelner Funktionalitäten gebildet werden können. Durch Selbstkonfiguration und eine intelligente Infrastruktur wird die Flexibilität erreicht, die eine unkomplizierte Integration von Funktionsanbietern wie auch -konsumenten ermöglicht. Zur Realisierung dieses Ansatzes werden Standards verwendet, die eine lose Kopplung der einzelnen Komponenten im Netz ermöglichen. Somit wird eine MMS vorgeschlagen, die den zukünftigen Ansprüchen von Nutzern gerecht wird.

1.2. Überblick über die Dissertation

Diese Arbeit deckt mehrere Aspekte aus den Forschungsbereichen „Nutzer-Interaktion“ und „Geräte-Interaktion“ ab. Abbildung 1.1 zeigt das Zusammenspiel der beiden Bereiche und welche Teilbereiche in den folgenden Kapiteln diskutiert werden[1].

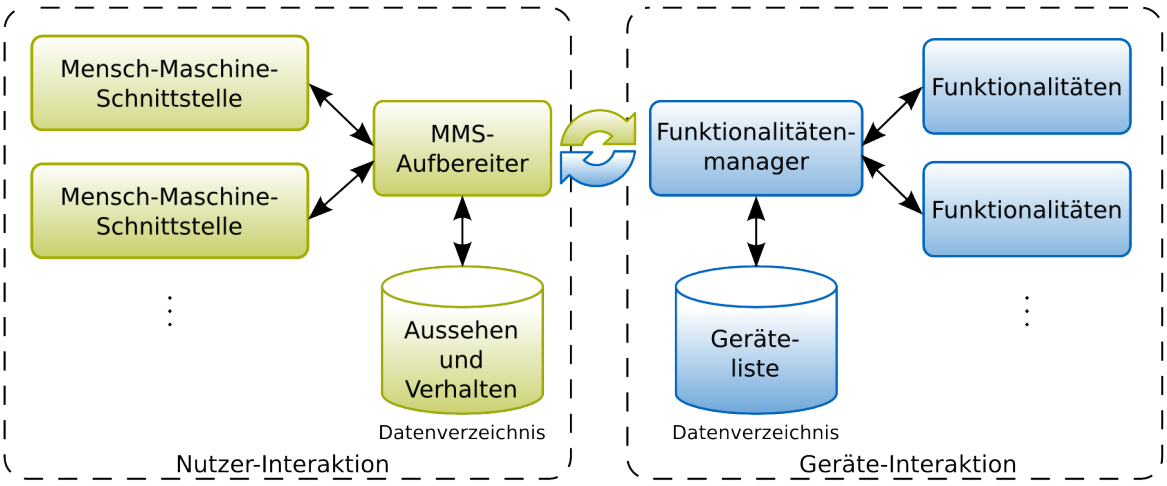

Abbildung 1.1.: *Thematische Übersicht der Arbeit.*

Der Funktionsblock „Nutzer-Interaktion“ ist für das Aufbereiten, Strukturieren und grafische Anbieten der verfügbaren Funktionalitäten verantwortlich. Auf der linken Seite von Abbildung 1.1 sind mehrere *MMS* angeordnet. Dabei könnte es sich beispielsweise um Displays für die grafische Ausgabe handeln. Diese stehen mit einer zentralen Einheit, dem *MMS-Aufbereiter*, in ständigem Kontakt. Von dieser bekommen die MMS Inhalte, die dem Nutzer zugänglich gemacht werden. Diese Inhalte werden in der Schnittstelle mit Hilfe des *Datenverzeichnisses* für das *Aussehen und Verhalten* generiert und auch verwaltet. Wie in Abbildung 1.1 ersichtlich ist, gibt es eine Überlappung an der ein Informationsaustausch zum Funktionsblock „Geräte-Interaktion“ erfolgt. Auf der rechten Seite können beliebig viele *Funktionalitäten* beherbergt sein. Bei diesen handelt es sich z.B. um den DVD-Player im Fahrzeug oder den GPS-Empfänger im Mobiltelefon. Diese werden vom *Funktionalitätenmanager* verwaltet. Dabei werden die verfügbaren *Funktionalitäten* aktiv oder passiv gesammelt, vorverarbeitet und in einem *Datenverzeichnis (Geräteliste)* für die weitere Verwendung abgelegt.

Mensch-Maschine-Schnittstelle

Wie in Abbildung 1.2 zu sehen, ist eine MMS eine Teilmenge eines Mensch-Maschine-Systems, wobei der Bereich „Aufgaben“ durch eine serviceorientierte Architektur [Erl05] abgedeckt und erweitert wird.
MMS existieren schon so lange wie es Maschinen gibt. Mit der Einführung von Maschinen,

[1] Diese Abbildung soll nur einen groben Überblick über die Architektur geben, weitere Details werden in den jeweiligen Kapiteln vorgestellt.

die das Ausführen von Aufgaben für den Menschen erleichtern sollten, wurden Eingabe- und Ausgabekanäle für diese zur Verfügung gestellt. Durch die Kanäle wird es dem Bediener ermöglicht eine Maschine zu steuern, Zustände dieser zu beobachten und eventuell in den Prozess einzugreifen.

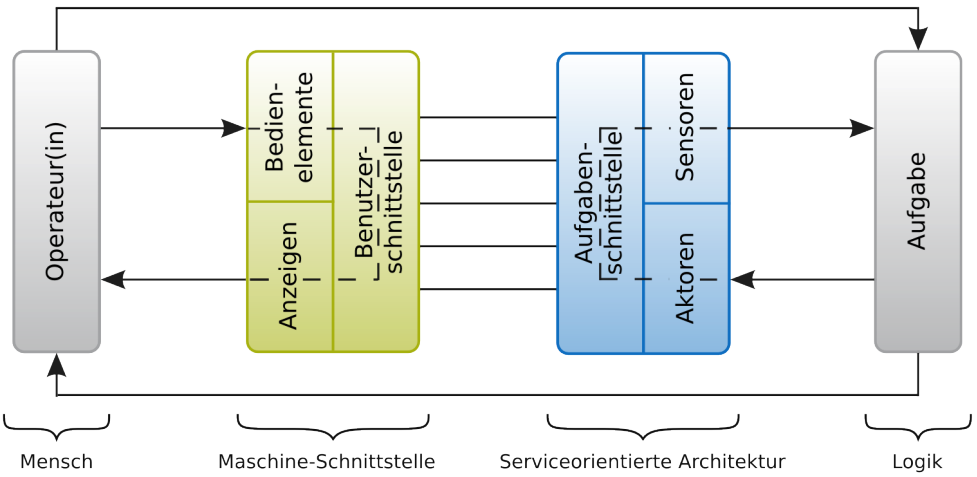

Abbildung 1.2.: *Vereinfachte Sicht eines Mensch-Maschine-Systems (nach [She87]).*

Diese Ein- und Ausgabekanäle können in Hardware, z.B. Steuerknüppel oder Signallampen, oder auch in Software mit Hilfe eines Displays visualisiert, realisiert werden. Die wohl am weitesten verbreiteten visuellen Benutzerschnittstellen sind die folgenden:

- **Grafische Benutzeroberflächen:** Das bekannteste Beispiel dafür ist der Computer. Sehr viele Betriebssysteme bieten eine grafische Oberfläche. Hier werden Systemfunktionalitäten durch Metaphern aus der Erfahrungswelt der Benutzer dargestellt. Beispielsweise symbolisiert der „Papierkorb“ das Löschen von Dateien.

- **Webbasierte Oberfläche:** Eine spezielle Form der grafischen Benutzerschnittstellen wird tagtäglich von Menschen verwendet. Es handelt sich dabei um den üblichen Webbrowser auf einem Computer. Der Webbrowser nimmt Eingaben entgegen und schickt diese über ein Netz an einen Webserver. Dort werden die Eingaben verarbeitet und das Ergebnis wird als Antwort an den Webbrowser übermittelt. Dieser bereitet die Antwort für den Nutzer grafisch auf und stellt sie dar. Hier stellt der verteilte Ansatz des Informationslieferanten und des Informationsdarstellers einen klaren Unterschied zu den allgemeinen grafischen Schnittstellen dar.

Die nachfolgenden Ein- und Ausgabekanäle werden zusätzlich zu den grafischen Benutzeroberflächen im Bereich der computergestützten Benutzerschnittstellen gerne und oft genutzt:

- **Kommandozeilenbasierte Benutzerschnittstelle:** Eine der ältesten Arten von computergestützten Benutzerschnittstellen stellt die Kommandozeile dar. Der Benutzer muss über eine Tastatur die Befehle an die Kommandozeile übergeben, dort werden diese interpretiert und ausgeführt. Eventuelle Rückgaben können dem Benutzer ebenfalls über die Kommandozeile angezeigt werden.

- **Zeichenorientierte Benutzerschnittstellen:** Die zeichenorientierte Benutzerschnittstelle ist eine Weiterentwicklung der zuvor genannten. Hierbei wird der Bildschirm flächig und nicht zeilenorientiert verwendet. Ein sehr bekanntes Beispiel ist hierfür der linuxeigene Dateimanager „Midnight Commander“ (siehe [Ron10]). Bei dieser Schnittstelle gab es Ansätze mit Hilfe von Sonderzeichen Rahmen für Fenster oder ähnliche Elemente wie Menüs, Knöpfe etc. nachzubilden. Entgegen der vorigen Benutzerschnittstelle kann diese zusätzlich auch mit dem Eingabegerät „Computermaus“ bedient werden.

- **Sprachbasierte Benutzerschnittstellen:** Diese Art ist zwar noch nicht sehr verbreitet, aber die sprachbasierte Schnittstelle spielt gerade im Fahrzeug eine wichtige Rolle. Dort ist beispielsweise die mündliche Eingabe einer zu wählenden Telefonnummer genauso alltäglich geworden wie die Ausgabe eines sprachsynthetisierten Hilfemenüs.

Es gibt noch zahlreiche weitere Benutzerschnittstellen, die oft auf die jeweilige Anwendung angepasst sind. Dazu zählen z.B.:

- **Anfassbare Benutzerschnittstellen:** Bei dieser Schnittstellenart werden die Ein- und Ausgabekanäle um die taktile Wahrnehmung erweitert. Ein Beispiel für solche Eingabekanäle sind Touchscreens. Diese haben sich schon zum Teil im Umgang mit Benutzerschnittstellen etabliert. Dabei handelt es sich um eine berührungsempfindliche Schicht, die auf einer grafischen Ausgabe angebracht und zu dieser kalibriert wird. So können dargestellte Menüelemente wie reale Taster betätigt werden. Ebenso gehören Joysticks oder Lenkräder für Computerspiele zu dieser Form der Schnittstellen. Bei dieser Art wird meist nur der taktile Eingabekanal genutzt. Hingegen gibt es weitere Ansätze mit speziellen Eingabecontrollern, wie beispielsweise in [SSV05] verwendet, die zusätzlich auch den Ausgabekanal nutzen und dem Nutzer eine taktile Rückmeldung geben.

- **Wahrnehmungsgesteuerte Benutzerschnittstellen:** Diese Benutzerschnittstellen sind Gegenstand der aktuellen Forschung. Beispielsweise durch elektronische Gestenerkennung soll die Arbeit mit dem Computer erleichtert werden. Dabei werden passive Interaktionen wie Bewegungen mit den Händen, die eine inhaltsbezogene Reaktion darstellen, als zusätzlicher Eingabekanal verwendet [ZLF+09]. Diese müssen dazu detektiert und interpretiert werden. Ebenso wird an den Möglichkeiten eines entsprechenden Ausgabekanals geforscht.

Der weitere Teil der Arbeit ist folgendermaßen gegliedert: Kapitel 2 beschreibt den Stand der Technik der dieser Arbeit zugrunde liegt. In Kapitel 3 werden von den betrachteten Anwendungsfällen alle relevanten Anforderungen abgeleitet, die eine zukünftige MMS abdecken soll. Diese werden kategorisiert und ergeben vier Grundprinzipien auf denen die vorgestellte Architektur beruht. Im darauffolgenden Kapitel 4 wird das entwickelte Konzept vorgestellt, das die zuvor genannten Grundprinzipien erfüllt und das Gesamtsystem beschrieben. Das Kapitel 5 geht auf den Aspekt der Informationssicherheit in der Architektur ein. Dieser Punkt muss die Grundprinzipien in dem hier thematisierten Umfeld stets umschließen. Die Realisierung des in Kapitels 4 vorgestellten Konzepts wird in Kapitel 6 beschrieben. Die Arbeit endet mit Kapitel 7, das die Erkenntnisse und die Ergebnisse der gesamten Arbeit zusammenfasst und einen Ausblick auf mögliche Erweiterungen gibt.

1.3. Beiträge der Dissertation

Diese Arbeit leistet Beiträge im Bereich von serviceorientierten Architekturen, der Darstellung von MMS Inhalten und der sicheren Verknüpfung dieser beiden Themengebiete. Die Hauptbeiträge werden im Folgenden zusammengefasst:

1.3.1. Architekturvorschlag für eine zukünftige MMS

In dieser Arbeit wird eine Architektur vorgestellt und untersucht, die möglichst alle zukünftigen Anforderungen, die in dynamisch erweiterbaren MMS-Systemen auftreten, abdecken soll. Die genannte Dynamik geht dabei von Funktionalitäten aus, die zur Laufzeit des Systems noch nicht vorhanden sind, hinzukommen bzw. wieder entfernt werden. Basierend auf der Anforderungsanalyse in Kapitel 3 wurden die in Abbildung 1.3 gezeigten vier Grundprinzipien erarbeitet, die als Basis für die Konzeptionierung dienen.

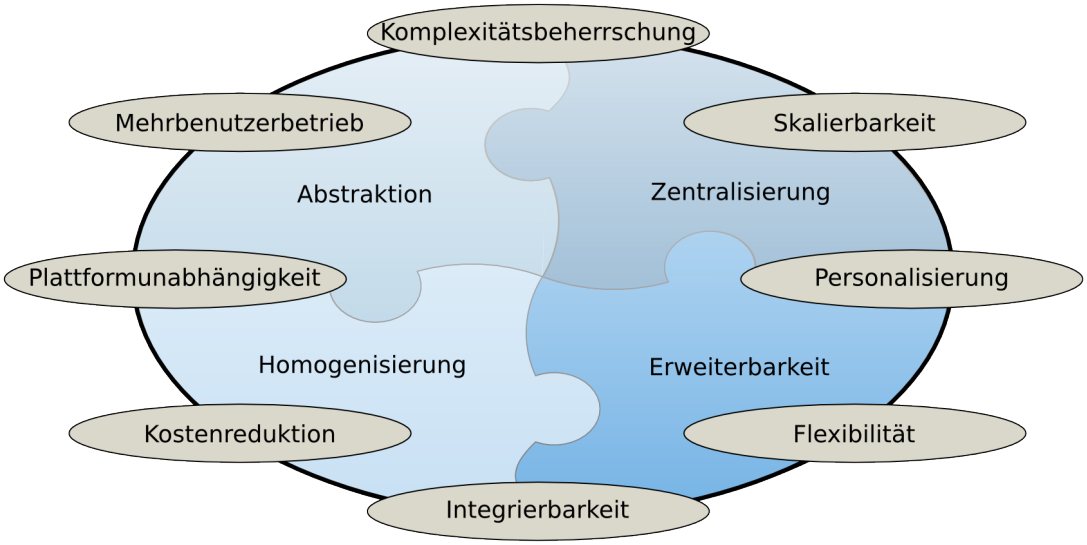

Abbildung 1.3.: *Prinzipien und die dadurch verbesserten Bereiche einer MMS.*

Die Prinzipien Abstraktion, Zentralisierung, Erweiterbarkeit und Homogenisierung ermöglichen zahlreiche Verbesserungen in den Bereichen Komplexitätsbeherrschung, Skalierbarkeit, Personalisierung, Flexibilität, Integrierbarkeit, Kostenreduktion, Plattformunabhängigkeit und Mehrbenutzerbetrieb.

Flexible Einbindung von Funktionalitäten

Durch den vorgestellten Ansatz der allgemeinen Beschreibung jeder Funktionalität wird es ermöglicht, dass diese in die vorgeschlagene Architektur eingebunden werden kann. Dabei ist es unerheblich, ob diese beim Generierungsprozess vorhanden ist oder nachträglich hinzu kommt. Diese kann anschließend von anderen Komponenten gesucht, gefunden und

genutzt werden. Ebenso wird durch die verwendeten Mechanismen ermöglicht, dass sich einzelne Komponenten sehr einfach austauschen lassen.

Erweiterbarkeit durch mobile Geräte

Mit wenig Aufwand ist es durch den vorgeschlagenen Ansatz möglich, beliebige mobile Geräte mit ihren Funktionalitäten in eine MMS-Architektur einzubinden. Dabei können diese Geräte genauso eigene Funktionalitäten mitbringen, die dann anderen Benutzern oder Geräten der Architektur zugänglich gemacht werden. Beispielsweise kann auf diese Weise ein Gerät als Ausgabekanal der MMS dienen und damit auch die anderen Funktionalitäten nutzen. Eine Besonderheit, die es in diesem Zusammenhang zu beachten gibt, ist, dass sich mobile Geräte in der Regel häufig und schnell an der Infrastruktur an- und abmelden. Diese Dynamik wird von der Architektur ebenfalls unterstützt.

1.3.2. Langfristige Erweiterbarkeit

Die Erneuerung oder Ergänzung von Funktionalitäten und Geräten bringt Herausforderungen mit sich. Diesen stellt sich der Ansatz dieser Arbeit in MMS-Systemen durch die Verwendung von standardisierten Schnittstellen. Gerade im Bereich der mobilen Geräte kann ein sehr kurzer Lebenszyklus beobachtet werden. Wenn ein neues Gerät oder eine Funktionalität in das System integriert werden soll, muss das System damit zurecht kommen. Für diesen Zweck eignen sich serviceorientierte Architekturen [DJMZ05] sehr gut und unterstützen dies. So wird eine Nachhaltigkeit gewährleistet.

1.3.3. Dynamische Generierung von Mensch-Maschine-Schnittstellen

Eine Untermenge der MMS sind die grafischen Benutzerschnittstellen (engl. Graphical User Interface, GUI). Da diese beliebig komplex gestaltet werden können, wird diesen eine besondere Aufmerksamkeit zu teil. Ein besonders wichtiger Punkt ist die Aggregation von vielen einzelnen Funktionalitäten eines verteilten Systems in eine Benutzeroberfläche. Das soll möglichst autonom geschehen und deckt auch neu hinzukommende Funktionalitäten ab. Das Aussehen der Ausgabe muss nicht beim Generierungsprozess festgelegt sein. Durch die vorgestellten Techniken wird eine Trennung zwischen Aussehen und Inhalten erreicht, die eine dynamische Änderung des Aussehens ohne Bearbeitung der Inhalte ermöglicht. Dies lässt einen schnellen und einfachen Wechsel des Aussehens zu, der sogar den oft vielseitigen Präferenzen der Nutzer gerecht werden kann. Neben dem Aussehen muss auch der Inhalt der Ausgabe nicht statisch sein, sondern kann auch gegebenenfalls durch hinzukommende Funktionalitäten erweitert werden.

1.3.4. Sicherheit

Durch die Öffnung des Systems ist es möglich dieses durch Hardwarekomponenten von Nutzern oder sogar durch Funktionalitäten von Anbietern zu erweitern. Diese Freigabe fordert ein gewisses Maß an Informationssicherheit um den Betrieb aufrecht zu erhalten und andere Nutzer zu schützen. Ein weiterer wichtiger Aspekt ist es sensible Daten zu schützen. Beispielsweise wünschen sich Fahrzeughersteller, dass gewisse interne Daten des Fahrzeugs nicht an die Öffentlichkeit gelangen. Hingegen darf auch der Datenschutz für Nutzerdaten dabei nicht vernachlässigt werden. Die in Betracht kommenden und nötigen Schutzziele werden in dieser Arbeit näher betrachtet und geeignete Sicherheitsmechanismen werden eingeführt.

1.3.5. Prototypische Implementierung

Durch eine prototypische Umsetzung wird die Machbarkeit des vorgeschlagenen Konzepts gezeigt. Der Prototyp verwendet hauptsächlich Hardware- wie Softwarekomponenten die der automotiven Welt nachempfunden sind. Dies dient als eine Beispielimplementierung für das allgemeine MMS-System, das in dieser Arbeit beschrieben wird. Die Anwendung des Systems ist dabei sehr offen, so kann es beispielsweise sowohl als Kundenterminal zum Probehören von Audiomaterial im Supermarkt als auch als Infotainmentsystem in einem Flugzeug oder Hotel seinen Einsatzzweck erfüllen. Bei der Umsetzung wird großer Wert auf offene Standards gelegt, um deren große Verbreitung und geringe Kosten als Vorteil zu nutzen.

Untersuchungen hinsichtlich der Praktikabilität

Der Prototyp wird sowohl quantitativ hinsichtlich Messgrößen wie Performanz, Netzlasten und Skalierbarkeit als auch qualitativ mit Hilfe von erarbeiteten Anforderungen betrachtet, um daraus Rückschlüsse auf die Praktikabilität in realen Einsatzszenarien schließen zu können. Bei dieser Evaluation liegt ein Fokus einerseits auf der Performanz des Systems selbst (Inter-Service-Kommunikation) und andererseits auf der Praktikabilität für den Nutzer. Ebenso gehört zu den Untersuchungen die Betrachtung der Sicherheitserweiterungen hinsichtlich Performanz und Robustheit gegenüber Bedrohungen der Informationssicherheit mit Hilfe von erprobten Richtlinien auf diesem Gebiet.

2. Stand der Technik

In dem interdisziplinären Forschungsprojekt *IT_Motive2020* (siehe [DEH+11]) in der Kooperation Car@TUM zwischen der Technischen Universität München, der BMW Group und der BMW Forschung und Technik GmbH wurde eine neue IT-Architektur für das Fahrzeug der Zukunft entwickelt. Im Rahmen des Industrieprojektes konnten sehr viele praktische Erfahrungen im automotiven Bereich gesammelt werden. Diese waren hilfreich bei der Durchführung dieser Arbeit und haben Fragen im Bereich der Mensch-Maschine-Schnittstellen (MMS) aufgeworfen. Dies und eine weitgehende Analyse des Stands der Technik bilden die Grundlage für diese Arbeit. In diesem Kapitel wird eine Übersicht über Techniken im Bereich von Benutzerschnittstellen, serviceorientierten Architekturen und Sicherheit in Kommunikationsnetzen gegeben.

2.1. Grafische Benutzerschnittstellen

Dieser Abschnitt stellt relevante Techniken aus dem Bereich der softwarebasierten grafischen Benutzerschnittstellen (engl. Graphical User Interface, GUI) vor. Abschließend werden diese im letzten Teil dieses Kapitels durch ausgewählte Kriterien evaluiert und miteinander verglichen.

2.1.1. Aktuelle Techniken

Im Rahmen dieser Arbeit werden drei große relevante Bereiche genauer betrachtet. Es werden diejenigen ausgewählt, in denen sich in letzter Zeit bezüglich grafischen Benutzerschnittstellen die größten Neuerungen ergaben. Dabei handelt es sich um den Heimcomputerbereich, das automotive Umfeld und die Automatisierungstechnik.

Heimcomputerbereich: Schon seit langem gibt es auf den Betriebssystemen von Heimcomputern grafische Benutzeroberflächen, wie z.B. den Window Manager Gnome [War04]. Eine der wohl bekanntesten im Kontext von intuitiver Bedienung ist die des Betriebssystems Apple „Mac OS“ [Bow01]. Eine Abwandlung dieser findet sich auf dem Apple iPhone wieder und zeigt auch mit einem Touchscreen ein sehr bedienfreundliches Verhalten. Ein weiterer Vertreter im mobilen Gerätebereich ist das System Google Android [Goo10b]. Dieses ist linuxbasiert und auch, wie das iPhone, an ein Marktplatzsystem gekoppelt, durch welches weitere Applikationen nachgeladen werden können. Diese Applikationen sind genauso wie Anwendungen von herkömmlichen Heimcomputeroberflächen meist von Grafikbibliotheken abhängig. Ein Vorteil ist, dass dadurch sehr schnell eine passende GUI zu einer

Anwendung erstellt werden kann. Wenn die Anwendungen immer mit einem ähnlichen Aussehen oder nach einem bestimmten Verhalten aufgebaut sind, fördert dies die Bedienerfreundlichkeit. Die erstellten Anwendungen sind aber sehr plattformabhängig und müssen auf andere Laufzeitumgebungen portiert werden. Das WebOS [Pal10] von Palm ist ähnlich wie die zuvor genannten Betriebssysteme aufgebaut, es ist linuxbasierend und versucht das Internet stärker in das System zu integrieren. Allerdings gelingt das lediglich bei der Integration von beliebten Webseiten wie z.B. dem sozialen Netz Facebook. Diese genannten Beispiele gehören zu den Arten der programmierten GUI (siehe Anhang A.1). Leider fehlt bei der Verwendung dieser für den hier vorgestellten Ansatz die Möglichkeit der verteilten GUI-Generierung. Deswegen wird der pure Einsatz dieser Techniken nicht weiter betrachtet.
Durch Softwareerweiterungen ist es möglich, die Oberflächen eines entfernten Rechners auf einem lokalen Rechner anzuzeigen. Gleichzeitig werden Tastatur- und Mausbewegungen des lokalen Rechners an den entfernten Rechner gesendet. Dadurch wird es ermöglicht auf einem entfernten Rechner zu arbeiten als säße man direkt davor. Beispiele sind die Programme „TightVNC“ [Tig10], hinter dem sich das „Remote Framebuffer Protocol“ (RFP [RW98]) verbirgt oder „rdesktop“ [rde10], das das „Remote Desktop Protocol“ (RDP [Mic10a]) realisiert. Diese Art der Protokolle verpackt den Bildschirminhalt und senden diesen zu dem Zielrechner. Dabei kann das Verpacken bedeuten, dass diese Protokolle, wie bei RFP, auf dem bitmaporientierten Grafikspeicher (engl. Framebuffer) arbeiten, diesen eventuell kodieren und verschicken oder teilweise grafische Beschreibungen des Bildschirminhalts verwenden. Beispielsweise löst XProtocol [SG86] dies auch so, um die Beschreibung an den Zielrechner zu leiten. Ein wesentlicher Nachteil dieses Ansatzes ist das hohe Datenaufkommen und die fehlende Trennung zwischen dem Design und den GUI-Inhalten selbst.
Neben Benutzerschnittstellen, die durch Erweiterungen den gesamten Bildschirminhalt weiterleiten, gibt es auch grafische Schnittstellen, die mit speziellen Interpretern arbeiten. Ähnlich wie zuvor gibt es eine grafische Beschreibung, diese beschreibt aber nicht den Inhalt des Bildschirms, sondern kann abgesetzte unabhängige Oberflächen beschreiben. Ein sehr gutes Beispiel dafür sind Protokolle im Bereich des Internets. Es existieren eine ganze Fülle von solchen, z.B. Hypertext Markup Language (HTML) [MN05], Extensible Hypertext Markup Language (XHTML) [W3C10c], XML User Interface Language (XUL) [Moz10], Wireless Markup Language (WML) [Ope10], um nur ein paar zu nennen. Gerade die ersten zwei sind sehr weit verbreitet und mit einem Standardwebbrowser nutzbar. Dabei wird die formatierte Beschreibung von einem Webserver an den Browser geschickt. Dort wird aus der Beschreibung der gewünschte Inhalt generiert und dargestellt. Unter der Voraussetzung, dass ein Browser nahezu auf jedem Gerät vorhanden ist oder installiert werden kann, gewinnt diese Technik an Bedeutung. Durch die Erweiterung von HTML mit JavaScript und XML ergibt sich das Konzept „Asynchronous JavaScript and XML“ (AJAX) [G$^+$05] mit dem die asynchrone Datenübertragung zwischen dem Browser und dem Server ermöglicht wird. Dieses lässt zu, Hypertext Transfer Protocol (HTTP) [IET10] -Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, d.h. die Seite zu verändern, ohne sie komplett neu zu laden. Dadurch lassen sich Anwendungen realisieren, die im Webbrowser ein desktopähnliches Verhalten emulieren. Diese Techniken bieten aber keine Organisation für GUI-Funktionalitäten sondern lediglich die Möglichkeit der Darstellung bzw. der Beschreibung von GUIs.
Neben diesen gibt es auch weniger verbreitete oder proprietäre Ansätze wie Extensible Application Markup Language (XAML) [Mic10b] oder Magic Extensible Markup Language (MXML) [Ado10]. Diese zählen zu den rein XML-basierten Protokollen. Oft werden solche

Protokolle erweitert um die Darstellung von GUIs zu ermöglichen, meist jedoch auf Kosten der Einfachheit oder Verbreitung.
Es existieren Ansätze bei denen dynamisch aus Oberflächen, die mit Programmiersprachen erstellt werden, Schnittstellen zu Webprotokollen, z.B. HTML, zur Verfügung gestellt werden. Auf diese Weise kann auf ein Programm über den Webbrowser zugegriffen werden. Eine Realisierung davon ist zum Beispiel JavaServer Pages [Sun10a]. Es handelt sich hierbei um eine Technik, nicht um ein Protokoll, für die Übertragung von GUIs. Die Syntax der JavaServer Pages orientiert sich an XML und kann mit Java Code kombiniert werden. Durch einen eigens dafür gedachten Compiler werden JavaServer Pages in Java-Quellcode umgesetzt. Dieser wird mit eventuell zusätzlichem Java-Quellcode von einem Java-Compiler in Bytecode (Servlet[1]) übersetzt. Dieses Servlet kann nun von einem mit einer Java-Ausführungseinheit erweiterten Webserver ausgeführt werden. So wird eine HTTP-Anfrage eines Nutzers an den Server und weiter an dieses Servlet geleitet. Dort wird sie interpretiert und bearbeitet. Anschließend werden die Ergebnisse in einer HTML-Seite an den anfragenden Webbrowser zurückgeschickt. Die Kombination aus der Programmierung mit einer Standardprogrammiersprache und Webtechnologien hat Vorteile bezüglich dem strukturierten Arbeiten mit Quellcode und der einfachen Darstellung von grafischen Elementen. Leider sind die resultierenden HTML Seiten statisch, d.h. es gibt nur die Möglichkeit sie komplett zu laden. Dynamische Effekte, wie solche, die durch AJAX ermöglicht werden, fallen damit weg. Die Erweiterung des Webservers durch Java ist des Weiteren fraglich, weil dadurch eine erhöhte Rechenleistung und Speicherbedarf erforderlich ist. Der wirkliche Vorteil von Java, die Plattformunabhängigkeit, wird zusätzlich durch die Integration in einen Webserver eingeschränkt, weil somit zuerst die Voraussetzung erfüllt sein muss, dass dies durch den Webserver unterstützt wird.
Eine weitere Technik die nach diesem Prinzip vorgeht ist XML11 [XML10]. Auch hier wird die programmierte Oberfläche zu Webprotokollen konvertiert. Diese Webprotokolle beschränken sich jedoch nicht auf HTML, sondern es werden zusätzlich JavaScript verwendet und sogar AJAX-Techniken unterstützt. Somit können durchaus dynamische Effekte im Gegensatz zu zuvor genannten angeboten werden. Die Ausgangssprache zur Programmierung ist wieder Java und damit sind aber auch Anpassungen und Erweiterungen des Webservers nötig.

Automotives Umfeld: Durch den wachsenden Funktionsumfang im Fahrzeug, können auch dort neue Techniken bezüglich der grafischen Benutzeroberflächen beobachtet werden. Die GUI im Fahrzeug wird gegenwärtig mit Standardprogrammiersprachen entwickelt und generiert. In Abbildung 2.1 ist ein Beispiel für die grafische Ausgabe einer solchen zu sehen. Oftmals wird eine Standardprogrammiersprache, wie zum Beispiel C++ verwendet und damit die Oberfläche samt dazugehöriger Anwendung entwickelt. Dabei werden immer öfter Standardkomponenten für die darunterliegende Hardware verwendet, die Anpassungen bezüglich des Fahrzeugumfeldes, wie z.B. stoßsichere Festplatten oder fahrzeugspezifische Schnittstellen wie z.B. CAN [ISO07] bieten.

Neben diesen lässt sich auch im Fahrzeug eine Entwicklung wie im Heimcomputerbereich beobachten. Diese Entwicklung äußert sich aktuell in stark angepassten Betriebssystemen

[1] Im Kontext von Webservern wird oft von einem Servlet gesprochen.

Abbildung 2.1.: *GUI-Ausgabe eines aktuellen Serienfahrzeugs.*

[Hei10], beispielsweise AutoLinQ [Con10] und Moblin [Lin10b]. Bei dem erstgenannten handelt es sich um ein System das auf dem Google Android Betriebssystem basiert. Dadurch kann es durch weitere Applikationen erweitert werden und so an die Vorlieben des Fahrers angepasst werden. Auf diese Weise können gegebenenfalls Funktionalitäten nachgerüstet werden, mit denen ein Fahrzeug nicht ab Werk ausgestattet wurde. Bei dem zweiten handelt es sich um ein angepasstes Linuxderivat, dessen Entwicklung mit Maemo [Nok10a], ebenfalls ein auf mobile Telefone angepasstes Linuxsystem, zu MeeGo [Lin10a] zusammengelegt wurde.
Neben den automobiltauglichen Betriebssystemen existieren in diesem Bereich auch spezielle Protokolle für den Austausch von Informationen. Ein weitgehend unbekanntes Protokoll, Vehicle User Interface Markup Language (vUIML) [JS04], orientiert sich an den zuvor erwähnten Markup Languages. Es wird von der Automotive Multimedia Interface Collaboration (AMI-C) verwendet, um grafische Benutzerschnittstellen zu beschreiben. Die Dokumentation und der Umfang ist genauso wie die Verbreitung nicht sehr umfangreich und wird deswegen nicht weiter betrachtet.
All diese Ansätze bringen keine Neuerungen im Fahrzeug, es wird lediglich die mobile Gerätewelt mit dem Fahrzeug verknüpft und das leider eher in einer physikalisch-mechanischen Art. Das heißt, dass es nahezu keinen Zugang mit einem mobilen Gerät zu den eigentlichen Fahrzeugdaten gibt. Vereinzelt werden Betriebssysteme aus der Gerätewelt auf Hardwarekomponenten im Fahrzeug portiert, um dessen Vorteile zu nutzen. Dabei sind natürlich die gewohnte Bedienung oder auch die weite Verbreitung als Vorteile zu nennen. Meist haben diese GUI-Komponenten aber nur auf abgeschottete Bereiche des Fahrzeugs Zugriff und leiden deswegen an mangelndem Funktionsumfang. Auch die Unterstützung bei der Integration von Geräten wird leider vernachlässigt.

Automatisierungstechnik: Die Automatisierungstechnik geht einen ähnlichen Weg, z.B. mit dem Konzept „Überwachung, Steuerung, Datenerfassung“ (ÜSE) (engl. Supervisory Control and Data Acquisition (SCADA)) um technische Prozesse zu überwachen und zu steuern. Bei dieser Realisierung werden die Steuerungssysteme und deren Parameter auf einem Arbeitsplatz visualisiert. Dies geschieht meist mit speziell dafür angepassten Oberflächen. Definierte Standards existieren hierfür nicht. Hingegen liefert WEBHMI [Ico10] einen sehr interessanter Ansatz. Da SCADA-Systeme vermehrt auf der Basis von TCP-basierten Internettechniken agieren, liegt es nahe auch in diesem Bereich internetbasierte Lösungen für die Visualisierung zu suchen. Die genannte Lösung setzt bei der Aufbereitung der Steue-

rungsdaten auf Webtechnologien. Leider muss die grafische Oberfläche, wie auch bei WEBMI [Atv10] zum Überwachen und Steuern an jede Industrieanlage stark angepasst werden.

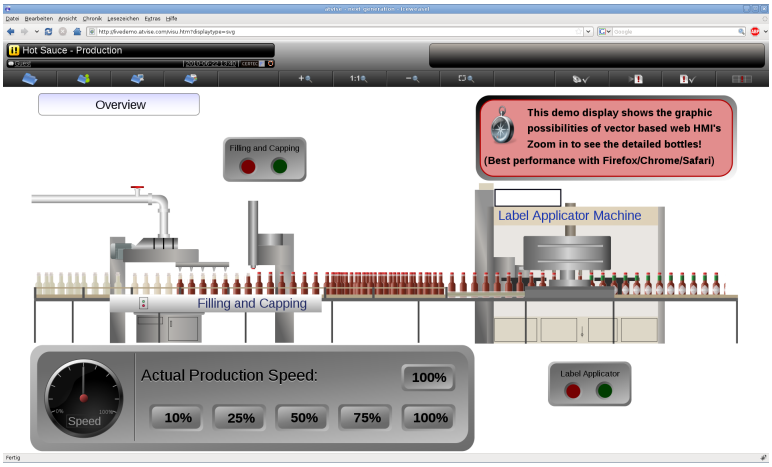

Abbildung 2.2.: *Überwachungs- und Steuerungsoberfläche der Firma Atvise (browserbasiert).*

Doch gerade bei WEBMI zeigen sich deutlich die Vorteile des Ansatzes der Verwendung von Webtechnologien, wie z.B. HTML, für grafische Benutzerschnittstellen [Swe00]. Ein Vorteil ist die Verwendung eines Browsers für jeden beliebigen HTML-basierenden Inhalt (siehe Abbildung 2.2). Des Weiteren wird von den Skalierungsmöglichkeiten der Darstellungsfläche durch den Browser profitiert. Bei diesen Lösungen handelt es sich um einen anwendungsfallspezifischen Ansatz, der keinerlei Möglichkeit der dynamischen Integration von neuen Funktionalitäten in die GUI bietet.

2.1.2. Evaluierung

Um einen Vergleich der aufgezählten Techniken im Bereich von grafischen Schnittstellen vornehmen zu können, müssen Kriterien festgelegt werden mit welchen die in Frage kommenden Techniken bewertet werden. Die Kriterien sollen dabei Punkte abdecken die allgemein bei der Softwareentwicklung interessant sind oder auf dem Gebiet von GUIs eine entscheidende Rolle spielen. Die verwendeten Kriterien werden in Anhang A.2 erläutert. In Tabelle 2.2 werden als mögliche Kandidaten HTML, XML, JSP, XProtocol und XML11 aufgezählt. HTML dient als Platzhalter für den Verbund aus mehreren webtechnologiebasierten Protokollen, da diese meist nur zusammen genutzt werden oder gerade durch ihr Zusammenwirken ihre Vorteile vereinen. Für XML gilt auch, dass dies der Überbegriff für die zuvor genannten XML-basierten Protokolle darstellt. Der Kandidat XProtocol repräsentiert die Protokolle die die grafischen Oberflächen von Anwendungen verpacken und über das Netz verschicken. Da die Unterschiede zwischen den einzelnen Protokollen so gering sind, wird nicht weiter auf einzelne Techniken eingegangen.
Die aufgeführten Protokolle werden durch die in Tabelle 2.1 festgelegte Skala bewertet.

Diese Skala soll die jeweilige Unterstützung oder nötige Anpassungen für jedes Kriterium in der nachfolgenden Tabelle wiedergeben.

Zeichen	Bedeutung
++	sehr gut unterstützt
+	unterstützt mit Anpassungen
−	realisierbar, aber mit hohem Aufwand
−−	nicht möglich
0	neutral oder keine Informationen verfügbar

Tabelle 2.1.: *Skala für die Bewertung.*

	HTML	XML	JSP	XProtocol	XML11
Dokumentation	++	++	+	+	−−
Verbreitung	++	++	−	++	−−
Verfügbarkeit	++	++	++	++	++
Plattformunabhängigkeit	++	++	+	+	+
Komplexität	+	+	+	++	+
Ressourcen	+	+	+	++	+
Prototypenentwicklung	++	-	+	−	+
Look-and-Feel	++	+	+	0	+
Änderungen am Look-and-Feel	++	+	+	0	+
Kopplung	++	+	−−	−−	−−
Interaktion	+	+	−	++	+
Reaktionszeit	+	+	+	++	+
Erweiterbarkeit	+	+	+	++	+

Tabelle 2.2.: *Evaluierung der grafischen Benutzerschnittstellen.*

Wie in Tabelle 2.2 zu sehen ist, erfüllt HTML die meisten relevanten Kriterien und eignet sich am besten für den hier vorgestellten Ansatz hinsichtlich der grafischen Benutzerschnittstellen. Es ist ein weit verbreiteter Standard und äußerst gut dokumentiert.
Jedoch existiert nahezu kein Ansatz für die dynamische Nachrüstung von Funktionalitäten. Die Verwendung von Webtechnologien lässt jedoch eine einfache Verbindung mit serviceorientierten Architekturen zu. Diese können diesen Nachteil kompensieren.

2.2. Serviceorientierte Architekturen

Dieser Abschnitt geht auf serviceorientierte Architekturen ein. Dabei werden die untersuchten Techniken wie zuvor in thematische Bereiche gegliedert. Im letzten Teil wird eine Evaluierung dieser durchgeführt und die Techniken dadurch gegenübergestellt.

2.2.1. Aktuelle Techniken

Auch für serviceorientierte Architekturen (SOA) werden technische Bereiche ausgewählt, die in letzter Zeit interessante Neuerungen erfahren haben. Dabei handelt es sich um die

drei Bereiche Heimcomputer/-entertainment, Automotive und Industrie.

Heimcomputer/-entertainment: Im Bereich der Heimcomputer und des Heimentertainments sind zahlreiche Innovationen bezüglich serviceorientierten Architekturen zu beobachten. Das begründet sich durch die mit SOA einhergehenden Vorteile, die gerade bei verteilten und modular aufgebauten Unterhaltungssystemen eine große Akzeptanz erfahren. Die Technik Home Audio/Video Interoperability (HAVi) wurde für den Heimbereich entwickelt und dient der Vernetzung von Multimediakomponenten. Leider wird die Anwendung dieser Technik auf IEEE1394 Netze beschränkt [Tei01]. Allerdings werden Plug and Play Funktionalitäten wie auch die Unterstützung von Quality-of-Service angeboten. Weitere Arbeiten haben gezeigt, dass sich HAVi mit weiteren Techniken wie Web Services (WS) und Open Service Gateway Initiative (OSGi) kombinieren lässt [BGM07]. So ergeben sich neue Möglichkeiten bei der Migration von Techniken. Ein Nachteil ist die sehr geringe Verbreitung von HAVi auf dem Markt.
Weiter verbreitet ist Java Intelligent Network Infrastructure (Jini). Jini wurde von Sun Microsystems für die spontane Vernetzung von Services und Ressourcen entwickelt. Für diese nutzt es die Java Technologie [Apa10c]. Services und Geräte werden von einem zentralen Systemservice, dem Lookup Service, registriert und verwaltet. Bei dieser Technik wird der Code, der für die Verwendung nötig ist, zum Anfragenden transportiert. Dieser Code kann dynamisch von den Clients heruntergeladen werden, wenn sie den Service nutzen wollen. Jeder Zugriff auf einen Service muss über den Lookup Service stattfinden. Ebenfalls ein Nachteil dieses Ansatzes ist die Voraussetzung von Java, wie in Kapitel 2.1.1 für die grafische Benutzerschnittstelle erläutert.
Sehr ähnlich dazu ist die Technik Java Remote Method Invocation (RMI) [Sun10b]. Im Vergleich zu Jini wird auf den Transport des Codes verzichtet und nur die Verweise mit den jeweiligen Methoden an den Anfragenden weitergegeben. Nach diesen Methoden kann in einem Verzeichnis gesucht werden und bei Bedarf mit den Verbindunginformationen die Methode in die eigene Anwendung eingebunden und darüber genutzt werden. Diese Technik ist dem traditionellen Schema der SOA [Erl05] schon sehr ähnlich. Jedoch ist man immer noch an eine Programmiersprache gebunden, die zudem als nicht sehr ressourcenfreundlich gilt.
Eine weit verbreitete Technik ist Universal Plug and Play (UPnP) [otUF10]. So wird diese Technik in der internationalen Vereinigung Digital Living Network Alliance (DLNA) [Dig10] als Grundlage für die Vernetzung verwendet. Dieses Konsortium hat es sich zur Hauptaufgabe gemacht, technische Leitlinien für vernetzte Geräte zu entwickeln und zu aktualisieren. DLNA bietet zahlreiche Erweiterungen in Kombination mit UPnP an, aber leider keine die die Nachteile von UPnP grundlegend beheben [Hem06]. Bei UPnP handelt es sich um eine einfach zu nutzende SOA für kleine Netze [SKLS05]. Durch die aus SOA bekannten Abläufe Bekanntmachung, Auffindung und Nutzung werden die Interaktion von Services und die netzbasierte Kommunikation zwischen Geräten unterstützt. Durch UPnP werden die Protokolle und die Schnittstelle spezifiziert. Nach dieser Spezifikation ergeben sich während der Teilnahme eines Gerätes an einer UPnP-Architektur sechs Phasen: Adressierung, Auffindung, Beschreibung, Kontrolle, Ereignis und Darstellung. Ein großer Vorteil von UPnP ist, dass die Programmiersprache und das Übertragungsmedium nicht festgelegt ist. Leider werden nur kleine Netze unterstützt und diese dürfen zusätzlich nur IPv4 basiert sein. Ein weiteres Problem ergibt sich mit der exponentiellen Zunahme der Broadcast Nachrichten bei

ansteigender Anzahl an Services/Geräten.
Web Services (WS) [NL04, M$^+$] ist eine der bekanntesten Techniken und besitzt die größte Verbreitung. Die WS-Architektur bietet eine Reihe von modularen Protokollblöcken, die auf verschiedene Weise in Form von Profilen[2] kombiniert werden können. Auf diese Weise wird es ermöglicht, dass verschiedene Anforderungen bei der Interaktion von heterogenen Softwarekomponenten erfüllt werden können. Dazu gehören beispielsweise Interoperabilität, Selbstbeschreibung und Sicherheit [BHM$^+$04]. Durch die Profile wird für eine Anwendung festgelegt, welche Teilmenge an Protokollen verwendet wird. Ebenso wird beschrieben, welche Änderungen der Profile nötig sind und auf welche Weise diese angewendet werden, damit die Kompatibilität zu anderen Services gewährleistet bleibt. Im Gegensatz zu UPnP sind WS für Netze jeglicher Größe ausgelegt. Sie bieten zusätzlich eine Reihe von Spezifikationen für die Service Auffindung, Servicebeschreibung, Sicherheit und Richtlinien an. Das Besondere an WS ist, dass die Implementierung der Logik des Service von der Schnittstelle entkoppelt ist. Sogar in einem derartigen Maß, dass ein Austausch dieser während der Ausführung möglich wäre. Neben diesen positiven Aspekten werden auch weit verbreitete Standards wie XML und SOAP für den Nachrichtenaustausch verwendet. Jedoch werden weder Plug and Play unterstützt, noch eine geeignete Lösung für die Integration von Geräten angeboten. Zudem fehlt die Möglichkeit der grafischen Aufbereitung, lediglich in [HY07, HYP$^+$08, CLV$^+$03] werden erste Ansätze präsentiert um Servicefunktionalitäten automatisiert darzustellen. Diese sind jedoch wenig flexibel und eingeschränkt was die grafischen Möglichkeiten anbelangt.
Eine sehr interessante Art von SOAs stellt das Device Profile for Web Services (DPWS) dar. DPWS wurde im August 2004 zum ersten Mal vorgestellt und im Mai 2005 und Februar 2006 von einem Konsortium (geleitet von der Microsoft Corporation) nochmals geprüft [CCK$^+$05]. Es handelt sich dabei um ein Profil, welches Komponenten der WS-Protokolle vereint um die dynamische Auffindung und die Fähigkeit zur Ereignisabarbeitung von WS zur Verfügung zu stellen. Das Profil legt verschiedene WS-Spezifikationen wie zum Beispiel WS-Addressing, WS-Discovery, WS-MetadataExchange und WS-Eventing für Geräte fest. Zeitweise wurde DPWS als Nachfolger von UPnP bezeichnet. Eine der Bezeichnungen war sogar in einer der ersten Versionen „A Proposal for UPnP 2.0 Device Architecture“. Jedoch ist DPWS nicht kompatibel zu UPnP und repräsentiert damit eine eigene Art von SOA Realisierung. DPWS ist in den Microsoft Betriebssystemen Windows Vista und Windows CE implementiert.

Automotives Umfeld: Auf diesem Gebiet existieren zwei Varianten der Neuerungen. Zum einen werden Zugänge zur vorhandenen IT-Architektur angeboten und zum anderen die gesamte Architektur erneuert. Bei der ersten Neuerung öffnen viele der Fahrzeughersteller teilweise ihre Fahrzeugsysteme, um Consumer Electronic (CE) Geräte zu integrieren. Ein weitverbreitetes Beispiel dafür ist ein Adapter für Apples iPod als eine optionale Erweiterung des Infotainmentsystems in vielen neuen Fahrzeugen. Ist der iPod mit dem Fahrzeug verbunden, wird beispielsweise der Name des derzeitigen Musiktitels auf dem Fahrzeugdisplay angezeigt. Diese Erweiterung ermöglicht dem Fahrer auch, den iPod über das Fahrzeug-Entertainment-System zu steuern. Interessante Aspekte sind hierbei die Erweiterung der Fahrzeugbusse um eine physikalische Schnittstelle wie z.B. Universal Serial Bus

[2] Eine Kombination von Protokollblöcken wird als Profil bezeichnet.

(USB) und die Einbindung in ein vorhandenes Ansteuerungssystem. Letzteres basiert bei dem genannten Beispiel auf einem Standard den das Gerät anbietet und dieser wird über proprietäre Anpassungen in das Fahrzeugnetz und in die GUI eingebunden und genutzt.
Momentan gibt es neuere Ansätze, die versuchen dieses Schema mehr und mehr aufzubrechen, meist jedoch ebenfalls durch proprietäre Techniken. Ein Beispiel hierfür wäre das Betriebssystem „QNX Neutrino RTOS“ [QNX10] von QNX. Dabei handelt es sich um ein Betriebssystem, das zum Beispiel für den Einsatz im automotiven Umfeld geeignet ist. Verschiedenste Dienste und Funktionalitäten können darauf entwickelt werden. Dieses wird zum Teil schon in Fahrzeugen verwendet [Hay09]. Nun wird versucht über ergänzende Techniken diesem Betriebssystem Schnittstellen hinzuzufügen, welche möglichst allen neuen Anforderungen begegnet. Dabei handelt es sich jedoch oft um proprietäre Entwicklungen und diese sind zudem auf Schnittstellen innerhalb des Fahrzeugs beschränkt. Bis zu der Ausdehnung dieses Ansatzes auf die Integration von Geräten ist es noch ein weiter Schritt.
Eine ähnliche Technik stellt Open Service Gateway Initiative (OSGi) [HC04] dar. Diese Technik ist nicht nur im automotiven Bereich vertreten, sie wird ebenso in mobilen Geräten oder in der Gebäudeautomatisierung angewendet. Die Spezifikation von OSGi definiert eine serviceorientierte Plattform, die als Basis für Serviceanbieter, Serviceentwickler und Verkäufer von Softwareanwendungen helfen soll, Services zu verteilen, entwickeln und verwalten. Sie basiert auf Java und bringt damit die typische Plattformunabhängigkeit mit sich. Dadurch verteilte Services werden als Bundle bezeichnet und werden in das Framework integriert. Man könnte einen OSGi Service als eine einfache Java-Schnittstelle bezeichnen, aber die Bedeutung der Services ist nicht klar definiert. Der Hauptnachteil bei dieser Technik ist ebenfalls der hohe Ressourcenverbrauch durch die alleinige Verwendung von Java.
Ein sehr vielversprechender Ansatz für eine serviceorientierte Architektur ist Common Object Request Broker Architecture (CORBA) [V$^+$97]. Hierbei wird ein objektorientierter Ansatz spezifiziert. In CORBA wird durch die genutzte Interface Definition Language (IDL) [Obj10] eine formale Spezifikation der zu entwickelnden Schnittstelle erstellt. Diese ist dafür verantwortlich, bestimmte Anwendungen für Zugriffe, seien sie lokal oder entfernt, zur Verfügung zu stellen. Die Spezifikation wird anschließend mit einem IDL-Compiler in Quellcode umgesetzt und diese mit der Logik zusammen zu einem ausführbaren Code übersetzt. Diese Technik ist unabhängig von Plattformen, Hardware und Programmiersprachen. Bedauerlicherweise hat sie nie eine große Verbreitung erfahren und die äußerst abstrakte Definition der Schnittstellen ist nicht besonders intuitiv und erschwert die Arbeit damit. Eine Alternative zu CORBA stellt Internet Communications Engine (ICE) [HS10] dar. Die Entwicklung von ICE wurde durch CORBA geprägt und ähnelt diesem Ansatz deswegen. Im Vergleich ist ICE schmalgewichtiger und weniger komplex. Allerdings ist es leider auch nicht sehr verbreitet und bekannt.
Im Automobilbereich können derzeit weitere verschiedene neue Ansätze gefunden werden, die in Richtung von flexibleren Infotainment Systemen gehen. Ein Beispiel dafür ist auch Continentals zuvor genanntes AutoLinQ [Con10]-System. Dieses kann durch weitere Applikationen erweitert werden und so an die Vorlieben des Fahrers angepasst werden. Leider werden hier weder die Integration von Services unterstützt, noch bietet AutoLinQ Methoden für die netzweite Nutzung von Services oder den installierten Applikationen an.

Industrielösungen: In der Heimautomatisierung gibt es eine weitere breite Palette an servicebasierten Lösungen, jedoch unterscheidet sich ein Teil dieser nur wenig von den bereits

genannten und muss deswegen nicht näher betrachtet werden. Es gibt jedoch eine große Anzahl von Techniken, die aus Umfeldern mit höheren Performanzansprüchen stammen, beispielsweise Thrift [Apa10a] oder Protocol Buffers [Goo10c]. Das erstgenannte wurde von Facebook eingeführt und hat den Sinn, genauso wie das von Google entwickelte Protocol Buffers, möglichst schnell viele Daten von einem Knoten zum anderen zu befördern. Die Übertragung von großen Datenmengen in kurzer Zeit, möglichst in Echtzeit, wird oftmals als großer Vorteil dieser Techniken angepriesen. Dies ist auch bei REST [PZL08] oder Etch [Apa10b] der Fall. So leistungsmächtig und einfach diese Protokolle sein mögen, so sehr fallen ihre Nachteile ins Gewicht. Große Einschränkungen ergeben sich im Bereich der Dynamik. Ein Haupttreiber der Performanz ist die Serialisierung der Nachrichten und damit der Verzicht auf eine XML Struktur. Somit entfällt das vergleichsweise zeitintensive Analysieren dieser, aber bedauerlicherweise gehen dadurch Aspekte wie die Selbstbeschreibung, variable Datentypen und die Integration von unbekannten Geräten verloren. Des Weiteren bietet keine dieser genannten Techniken komfortable Verwaltungsmethoden an. Es wird lediglich das Protokoll definiert, sodass Mechanismen zur Auffindung, Bearbeitung und Datenmanagement bei Bedarf von Entwicklern händisch, sowohl konzeptionell als auch programmiertechnisch, ergänzt werden müssen.

2.2.2. Evaluierung

Auch für die Evaluierung der SOA werden Kriterien ausgewählt, die in diesem Bereich interessant sind oder bei Entwicklungen für diese entscheidend sind. Eine genaue Erläuterung dieser Kriterien ist in Anhang A.3 zu finden. Durch Gegenüberstellen der Kriterien mit den vorgestellten Techniken soll auch hier ein Favorit gefunden werden, der als Ausgangspunkt für diese Arbeit dient.
In nachfolgender Tabelle werden die in Frage kommenden Techniken aufgeführt. Dabei ist zu erwähnen, dass als Vertreter für die Industrielösungen nur REST bewertet wird. Die Bewertung wird wieder mit der in Tabelle 2.1 aufgeführten Skala durchgeführt.

Wie in Tabelle 2.3 zu sehen ist, ist mit den für SOA relevanten Kriterien die DPWS-Technik als ein vielversprechender Kandidat ausgemacht worden. Gerade die Kombination von WS und UPnP stärkt deren Vorteile und deckt so die Ansprüche ab, die in dieser Arbeit gefordert werden.

2.3. Sicherheit in Kommunikationsnetzen

In diesem Abschnitt wird auf die Sicherheitsaspekte, die für diese Arbeit relevant sind, eingegangen. Dabei wird der Fokus auf Kommunikationsnetze gelegt und eine Übersicht über zahlreiche Techniken auf diesem Gebiet gegeben. Die Sicherheitsaspekte gliedern sich in Schutzziele und Bedrohungen (siehe [IT03]). Nachfolgend wird auf eine Aufteilung in Bereiche wie zuvor verzichtet, da die Sicherheitsmechanismen in allen Bereichen ähnlich sind und meist nur eine Erweiterung für die Absicherung darstellen. Genauso wird von einer Evaluierung abgesehen, da die Techniken bestimmte Zwecke erfüllen und meist durch Neuerun-

	DPWS	WS	REST	CORBA	RMI	UPnP	Jini	HAVi	OSGi
Dokumentation	++	++	+	−	+	+	−	−	−
Verbreitung	+	++	−	−−	−−	++	−	−	+
Verfügbarkeit	++	++	++	−−	−−	+	−−	−	+
Plattformabhängigkeit	++	++	0	0	++	0	−	+	−
Komplexität	++	0	0	−−	−−	+	−−	−	+
Ressourcen	++	0	0	−−	−−	+	−−	−	+
Peer-to-Peer	++	++	−−	++	+	++	−−	−−	−−
Ereignisbearbeitung	++	−	−−	−−	+	++	+	+	+
Geräteunterstützung	++	−	−	+	+	+	+	+	+
Plug and Play	++	−	0	+	+	+	+	+	−
Filterung	0	0	−−	0	+	0	+	+	+
Zustandsspeicherung	++	++	−	++	+	++	+	+	+
Netzverkehr	+	+	−	0	++	−	++	+	++
Sicherheit	++	++	+	0	++	−	+	+	+
Codegenerierung	++	++	0	+	0	0	0	0	0
Flexibilität	++	++	−	++	−	−	+	+	+

Tabelle 2.3.: *Evaluierung der serviceorientierten Architekturen.*

gen ersetzt werden. Somit gibt es weniger konkurrierende Techniken, die verglichen werden müssen, um eine geeignete Wahl zu treffen.

2.3.1. Sicherheit in Webtechnologien

In dem nachfolgenden Kapitel werden die in den jeweiligen Teilbereichen üblichen Sicherheitsmechanismen vorgestellt. Aufgrund der großen Verbreitung von Webbrowsern und Webservern existiert ein für die meisten Anwendungen hinreichender Schutz der Informationssicherheit in dem Bereich der Webtechnologien. Selbst kritische Anwendungen mit personenbezogenen Daten wie Online-Banking oder der neue elektronische Personalausweis [Das10] werden durch die in Webtechnologien genutzten Sicherheitsmechanismen abgedeckt. Dieser erfüllt zwar nicht alle Richtlinien aus Anhang A.6, aber die bisher genutzten Ansätze genügen, um ein gewisses Maß an Sicherheit zu erzielen, das für die meisten Anwendungen ausreichend ist. Nachfolgend werden deshalb die bisher in diesem Feld etablierten Sicherheitserweiterungen erläutert.

Authentifizierung

Es existieren zwei Authentifizierungsmethoden bei Webtechnologien. Zum einen kann ein Nutzername und Passwort für die nutzerbasierte Authentifizierung verwendet werden. Dabei werden diese Nutzerlegitimierungen über den Webbrowser abgefragt und an den Webserver geschickt. Die Realisierung dieser Methode erfolgt zumeist mit serverseitigen Sprachen. Daneben gibt es eine gerätebasierte Art der Authentifizierung. Bei Verbindungsaufbau mit einem Webserver kann der Webbrowser dessen Zertifikate anfordern und so sicher gehen, dass der Webserver auch der ist, der er vorgibt zu sein. Umgekehrt ist dieses Vorgehen eher unüblich. Allerdings gibt es ebenfalls die Möglichkeit, dass der Webbrowser sich über ein

Zertifikat gegenüber dem Webserver authentifiziert. Für diese Art ist eine Erweiterung des Webbrowsers nötig und nicht jeder Webbrowser unterstützt dies. Es handelt sich dabei um Softwareerweiterungen wie auch Hardwareerweiterungen, wie z.B. Smartcards mit Lesegeräten.

Autorisierung

Eine weitverbreitete Art Inhalte vor unautorisiertem Zugriff zu schützen, ist die Verwendung von *.htaccess*. Es handelt sich dabei um eine Konfigurationsdatei, die zusammen mit National Center for Supercomputing Applications (NCSA)-kompatiblen Webservern eingesetzt werden kann. Damit ist es möglich, einen Zugriffsschutz für Dateien oder Verzeichnisse zu gewährleisten. Diese Art von Schutz verwendet einen Nutzernamen und ein Passwort für die Authentifizierung. So einfach diese Art ist, so anfällig kann diese auch gegen Attacken sein. Beispielsweise besteht die Gefahr des Mitschneidens dieser Legitimierung. Deswegen empfiehlt es sich Mechanismen in Verbindung mit HyperText Transfer Protocol Secure (HTTPS) zu nutzen. Dabei werden ebenfalls Nutzername und Passwort abgefragt, und mit dem Nutzerverzeichnis abgeglichen, jedoch erfolgt die Übertragung über einen verschlüsselten Kanal. So empfiehlt es sich, HTTPS und damit Transport Layer Security (TLS)/Secure Sockets Layer (SSL) für die Übertragung zu verwenden. Ein Mitschneiden von Nutzername und Passwort wird dadurch erschwert. Werden diese Legitimierungen beim Webserver empfangen, muss der Entwickler selbst den Zugriff auf die Informationen regeln.

Verschlüsselung

TLS/SSL wird häufig für die Übertragung von sicherheitsrelevanten Informationen zwischen Webbrowser und Webserver verwendet. Auf diese Weise wird damit die Vertraulichkeit sichergestellt.

2.3.2. Sicherheit in Device Profile for Web Services (DPWS)

Da die *Organization for the Advancement of Structured Information Standards (OASIS)* die Verantwortung für Regulierungs- und Standardisierungsaufgaben bezüglich DPWS-Systemen trägt, beschreibt diese auch Richtlinien für Sicherheitsaspekte in diesen [OAS10a]. Neben den in Anhang A.4 erläuterten Anforderungen der OASIS im Bereich der Sicherheit für DPWS, wird anschließend auf die Bündelungen von Techniken, den sogenannten Profilen eingegangen, die OASIS als Lösung hierfür vorsieht.

Sicherheitsprofile

Die Sicherheitsprofile stellen eine Reihe von Empfehlungen für die Implementierungsfreiheiten hinsichtlich Sicherheit in der Transportschicht und vorzugsweise in der Anwendungsschicht dar [MLH+08]. Diese Beschreibung stellt jedoch keine Spezifikation für einen DPWS-

spezifischen Sicherheitsstandard dar. Sie schlägt die Anwendung von mehreren in Netzarchitekturen etablierten Technologien auf dem Bereich der Sicherheit, wie z.B. AES, TLS oder X.509, vor. Diese Beschreibung teilt sich in zwei Profile auf, dem transportschichtbasierten und dem anwendungsschichtbasierten Sicherheitsprofil. Im Nachfolgenden werden diese vorgestellt.

Transportschichtbasiertes Sicherheitsprofil: Dieses Profil wird in den Richtlinien als Voraussetzung für die Abfrage der Beschreibung eines Services und optional für die Verwendung des Services deklariert. Ziel dieses Profils ist es, Sicherheitsregeln in der Transportschicht des Netzprotokollstacks einzuführen, z.B. den Aufbau einer sicheren Ende-zu-Ende Verbindung zwischen Gerätesockets. Der sichere Kanal muss die Abdeckung der Mechanismen Vertraulichkeit, Authentifizierung und Integrität garantieren. Deshalb sind weitverbreitete Technologien wie PKI, TLS/SSL-Sessions, X.509-Zertifikate sowie symmetrische/asymmetrische Kryptografie geeignete Kandidaten um solche Mechanismen in DPWS anzubieten. Des Weiteren soll der sichere Kanal die HTTP-Authentifizierung mit Nutzernamen und Passwort unterstützen, um den Nachrichtenaustausch, der durch Services erzeugt wird, ergänzend zu schützen.

Anwendungsschichtbasiertes Sicherheitsprofil: Das anwendungsschichtbasierte Sicherheitsprofil ist auch als nachrichtenschichtbasiertes Sicherheitsprofil bekannt, da es verwendet werden soll, um die Vertraulichkeit, Authentifizierung und Integrität der SOAP-Nachrichten sicherzustellen. Die Festlegungen gehen nur oberflächlich auf diese Aspekte ein. Es wird dabei vorgeschlagen die Nachrichten, die auf einem UDP-Port übertragen und empfangen werden, zu schützen, wobei ein sicherer Kanal durch TLS bei einer verbindungslosen Kommunikation nicht möglich ist [HLP+09]. Daneben sind die Auffindungs- und Notifikationsfunktionalitäten, die durch die SOAP-Nachrichten realisiert werden, die Hauptadressaten dieses Profils. Durch das OASIS Konsortium wurden in [OAS10b] weitere Richtlinien für dieses Profil festgelegt. Diese sind auch innerhalb des DPWS-Protokolls als WS-Security bekannt. Diese Richtlinien beschreiben Mechanismen, um einen Sicherheitstoken als Teil einer SOAP-Nachricht mitzuschicken.

Diese Mechanismen decken jedoch nicht alle Schutzziele für Web Services ab. Aus diesem Grund ist der Einsatz von weiteren robusten Mechanismen nötig, die auf den Technologien basieren, die bereits für die Transportschicht empfohlen wurden.

2.4. Zusammenfassung

In diesem Kapitel wurde der Stand der Technik zusammengefasst. Gerade viele Neuerungen auf dem Markt des Heimentertainments zeigen die Notwendigkeit einer gründlichen Überarbeitung von Mensch-Maschine-Schnittstelle (MMS)-Systemen. Die vorgestellten Techniken zeigen meist Nachteile bei der Verwendung in verteilten und eingebetteten Systemen. Lediglich Webtechnologien und DPWS bieten die notwendigen Grundlagen für die Erstellung einer zukünftigen MMS. Die Gebiete der grafischen Benutzerschnittstellen, der service-

orientierten Architekturen und Sicherheit in Kommunikationsnetzen werden abgegrenzt und eine Recherche auf diesen durchgeführt. Somit wird der Stand der Technik in den Forschungsgebieten dieser Arbeit ausführlich dargelegt und diskutiert. Um diese Techniken bewerten zu können, wurden für die beiden ersten Teilgebiete Kriterien aufgestellt und die relevanten Techniken durch die je nach Teilgebiet geeigneten Kriterien miteinander verglichen. Es existieren sicherlich weitere Techniken, die nicht erwähnt wurden, aber deren Auflistung und Abwägung würde den Rahmen dieser Arbeit übersteigen und keinen Mehrwert erbringen. Aus diesem Grund wurde eine Auswahl mit den vielversprechendsten Kandidaten getroffen. Durch Evaluierung dieser Kandidaten ergibt sich ein deutliches Bild der Wahl der Techniken für den weiteren Verlauf der Arbeit. In der engeren Auswahl ist nun eine Kombination von Webtechniken für die grafische Benutzerschnittstelle, zum Beispiel eine Kombination aus HTML und JavaScript. Für die serviceorientierte Architektur erscheint DPWS als passendes Gegenstück, welches das Management der Funktionalitäten abdecken kann.
Da sich das dritte Gebiet „Sicherheit in Kommunikationsnetzen“ lediglich auf die Favoriten der Gebiete „Grafische Benutzerschnittstellen“ und „Serviceorientierte Architekturen“ bezieht, wurde hier keine Evaluierung einzelner Techniken vorgenommen.
Als nächsten Schritt gilt es nun die Anforderungen zu definieren, die speziell mit dem Thema der Arbeit verwandt sind und mit diesen eine endgültige Wahl der geeigneten Techniken, mit möglichst geringem Aufwand an nötigen Anpassungen, zu treffen.

3. Anforderungsanalyse

In diesem Kapitel werden verschiedene Anwendungsfälle vorgestellt. Dabei sollen diese möglichst auch denkbare Szenarien in naher Zukunft abdecken. Daraus ergeben sich dann die Anforderungen an eine zukünftige Benutzerschnittstelle. Im Anschluss daran werden die Anforderungen gesammelt und kategorisiert. Diese dienen im weiteren Verlauf der Arbeit als Basis für die vorgestellten Konzepte. Des Weiteren werden exemplarisch Einsatzmöglichkeiten gezeigt um die Vielseitigkeit einer Mensch-Maschine-Schnittstelle (MMS) mit den aufgelisteten Anforderungen zu zeigen.

3.1. Nutzungsszenarien für eine zukünftige MMS

Im Nachfolgenden werden Szenarien beschrieben, die eine zukünftige MMS betreffen können. Damit sind jedoch nicht solche Visionen von Ein- und Ausgabekanälen gemeint, wie beispielsweise in Abbildung 3.1, sondern die gesamte Architektur hinter der MMS. Die Aufzählung der Szenarien soll stufenweise komplexer werden, wobei das erste Szenario bereits den jetzigen Stand der Funktionalitäten im Bereich von MMSs komplett erfüllt. Im Anschluss werden aus diesen die Anforderungen extrahiert und Prinzipien eingeführt, durch welche diese abgedeckt werden können.

Abbildung 3.1.: *Vision zukünftiger Ein- und Ausgabekanäle einer MMS [His10].*

3.1.1. Normalbetrieb

Heutzutage ist es normal, sich in ein Serienfahrzeug zu setzen, die Zündung einzuschalten und durch einen fahrzeugspezifischen Controller als *Eingabekanal* mit dem Fahrzeug zu interagieren (siehe Abbildung 3.2). Die *Repräsentation* der Bedienoberfläche erfolgt hauptsächlich *grafisch* mit Hilfe eines Displays in der Mittelkonsole und steht meist exklusiv als alleiniger *Ausgabekanal* dem Fahrer zur Verfügung. Mit Hilfe eines strukturierten Menüs ist es dabei möglich, sowohl mit *bestimmten Funktionalitäten zu interagieren*, als auch auf *private oder fahrzeuginterne Daten zuzugreifen*. Bei diesen Daten kann es sich um Fahrzeuginformationen wie Reifendruck, Daten von Informationssystemen wie Navigationskarten, Multimediainhalten wie DVD oder auch persönliche Dienste wie E-Mail oder Kalender handeln. Diese *Interaktionsmöglichkeiten* müssen grafisch aufbereitet sein und dem Nutzer *gebündelt* angeboten werden. Ergänzend zu den bereits erläuterten derzeit im Fahrzeug realisierten Eigenschaften, ist es wünschenswert auch zukünftige Anforderungen ohne Komplexitätssteigerung zu unterstützen. So soll zum Beispiel während der Produktion eines georderten Fahrzeugs durch *Selbstkonfiguration* erreicht werden, dass durch das einfache Zusammenstecken einer beliebig erweiterbaren Untermenge von Komponenten bereits eine funktionierende MMS entsteht. Ebenso wichtig ist es, den *Datenaustausch* zwischen Display und der GUI-Aufbereitung so *gering wie möglich* zu halten, um so eine unnötige Belastung des darunterliegenden Kommunikationsnetzes zu verhindern. Aus diesem Grund sollen die Nutzdaten auch komprimiert im Netz verteilt werden.

Abbildung 3.2.: *Cockpit eines aktuellen Serienfahrzeugs (Quelle: [BMW10]).*

3.1.2. Erweiterter Betrieb

Wie in den Abbildungen 3.2 und 3.3 zu sehen ist, können derzeitig bis zu drei Nutzer im Fahrzeug standardmäßig gleichzeitig mit der MMS interagieren. Jedoch handelt es sich bei

diesen Realisierungen um fest verkabelte und statisch eingerichtete Displays und jegliche Erweiterung bedarf immensen Aufwands. Dabei wünscht sich nahezu jeder Passagier eine Interaktion mit dem Fahrzeug. Deswegen ist die Einführung eines echten *Mehrbenutzersystems* von großem Nutzen. So kann der auf der Rückbank sitzende Mitfahrer auch beispielsweise das Multimediaangebot im Fahrzeug nutzen oder Lokalisierungsinformationen abfragen. Die Unterstützung mehrerer Nutzer bedingt auch die Einführung eines *Zugangsmanagements*.

Abbildung 3.3.: *Rückbank eines aktuellen Serienfahrzeugs (Quelle: [BMW10]).*

Sollen bei einer nachträglichen Erweiterung die weiteren Displays einfach nachzurüsten sein, bedarf es einer gewissen Intelligenz bei diesen, sodass sie sich in das System angemessen integrieren können. Dies bedeutet, dass *„Intelligente Displays“* zur Darstellung der GUI als auch zur dynamischen Integration ein Minimalbetriebssystem besitzen und dafür auch genügend Rechenleistung vorhalten müssen. Für die Interaktion mit der GUI kann ein eigener Eingabecontroller oder vielleicht ein selbst ins Fahrzeug mitgebrachter Eingabekanal wie eine Maus oder ein Touchscreen verwendet werden, die *Art des Controllers* ist dabei *nicht entscheidend*. Durch die *lose Kopplung* in einem solchen System wäre es möglich jeden *Eingabekanal* jedem *Ausgabekanal* zuzuordnen. Somit ergibt sich eine äußerst hohe Flexibilität bei der Verteilung und Verknüpfung. Zusätzlich soll dem Nutzer in Zukunft ermöglicht werden diese Zuordnung seiner Ein- und Ausgabekanäle selbst zu konfigurieren. Dies sollte schnell und ohne größere Eingriffe, die den normalen Ablauf stören würden, durchführbar sein. Ebenso wird das Aussehen der grafischen Oberfläche anpassbar sein. Damit wird es ermöglicht, diese teilweise zu personalisieren oder sogar einen fahrzeugübergreifenden Designwechsel vorzunehmen. Somit könnte der Inhalt der MMS immer derselbe sein und lediglich das Aussehen fahrzeugspezifisch wechseln und eine Kostenreduktion durch die Abstraktion des Aussehens von den dahinterliegenden Inhalten wird erreicht. Wird der Designwechsel auf die Nutzer erweitert, so ergibt sich ein Mehrwert für die Kunden durch *Personalisierung*. Die BMW Group vertreibt beispielsweise *verschieden gestaltete MMS* im Mini, BMW und Rolls Royce. Für jede MMS könnte dann dasselbe System verwendet werden

und lediglich die Beschreibung des Aussehens wird ausgewechselt.

3.1.3. Nutzung durch weitere mobile Geräte

Mit dem jetzigen Stand der Technik ist es beispielsweise nur beschränkt möglich zusätzliche Geräte in das Fahrzeugnetz zu integrieren. In Abbildung 3.4 ist die Nutzung eines mobilen Gerätes mit dem Fahrzeug zu sehen, aber diese Nutzung ist auf ein speziell angepasstes Filmangebot begrenzt.

Abbildung 3.4.: *Verwendung eines mobilen Gerätes als Ausgabekanal (Quelle: [BMW10]).*

Dabei werden zur Integration meistens physikalische Schnittstellen wie USB oder Bluetooth zur Verfügung gestellt und lediglich für den Dateitransfer verwendet. Zudem sind diese Schnittstellen jeweils auf bestimmte Geräte, wie zum Beispiel den Apple IPod, angepasst. Nun wollen die Nutzer ihre *mobilen Geräte mit ins Fahrzeug* bringen und als *zusätzlichen Ausgabekanal oder Eingabekanal* für die MMS verwenden. Gerade für Fahrzeuge, die nicht mit der höchsten Ausstattungsstufe versehen wurden, ist die *Integration von fremden Geräten* interessant. Ein besonderes Augenmerk hinsichtlich der Sicherheit muss dabei auf ein *Zugangsmanagement für Geräte* gelegt werden. Es muss gesichert sein, dass nur autorisierte Geräte auf das Fahrzeugnetz zugreifen können. Zusätzlich ist in diesem Fall eventuell eine *Verschlüsselung* nötig, um die Datenübertragung vor nicht berechtigten Nutzern zu schützen. Neben der Sicherheit sind auch die Ressourcen ein Punkt der beachtet werden muss. Das System muss dynamisch auf den zusätzlichen *Ressourcenanspruch von weiteren Ausgabekanälen* reagieren können. Das gleiche gilt für die Freigabe dieser bei der Abmeldung des mobilen Gerätes. Hinsichtlich der Vielfalt der mobilen Geräte werden durch ein adaptives Aufbereiten der GUI-Informationen verschiedene Formate wie z.B. die Auflösung der *verschiedenen Ausgabegeräte unterstützt*. Da durch die Bindung an spezielle mobile Geräte die

Akzeptanz der Nutzer beeinträchtigt würde, muss hinsichtlich der Hardware der verschiedenen mobilen Geräte auf die Unterstützung von *plattformunabhängigen* Lösungen geachtet werden. Somit ist es möglich, nahezu jedes mobile Gerät als zusätzlichen Ausgabekanal oder Eingabekanal zu verwenden.

3.1.4. Erweiterung um Funktionalitäten

Die in Kapitel 3.1.3 genannten mobilen Geräte können neben der Integration als „Intelligentes Display“ auch als Anbieter von Funktionalitäten agieren. In Abbildung 3.5 ist zu sehen, wie bereits in einem Serienfahrzeug beispielsweise die Funktionalität des Dateispeichers eines mobilen Gerätes in das Fahrzeug integriert wird.

Abbildung 3.5.: *Funktionalität eines mobilen Gerätes in einer MMS (Quelle: [BMW10]).*

Leider wird dies im Moment nur stark eingeschränkt angeboten. In Zukunft soll das System eine *dynamische Nachrüstbarkeit* von Funktionalitäten anbieten. Dadurch können dessen Anwendungen als Dienste anderen *Nutzern* im Fahrzeug *zur Verfügung* gestellt werden. Somit könnte ein Nutzer beispielsweise das GPS seines Fahrzeugs funktional nachträglich durch das aktuelle Lokalisierungssystem seines mobilen Gerätes ersetzen. Dazu muss lediglich das Gerät mit dem MMS-System verbunden und die Lokalisierungsanwendung gestartet werden. Diese wird im System über die *allgemeine Beschreibung von Schnittstellen* bekannt gemacht und kann, vom *Zugangsmanagement* autorisiert, durch andere Nutzer oder Services genutzt werden. Voraussetzung hierfür ist ein System das dynamisch auf den *Ressourcenverbrauch der neuen Funktionalitäten* reagieren kann. Durch diesen Erweiterungsmechanismus lassen sich auch leicht Funktionalitäten nachrüsten, die im Fahrzeug verbaut sind. Durch *offene und standardisierte Schnittstellen* wird dies auch Drittanbietern ermöglicht. Möchte der Nutzer beispielsweise zusätzlich zu seinem DVD-Laufwerk ein Bluray-Laufwerk haben,

kann er dieses in der Werkstatt einbauen lassen. Dieses wird wieder mit dem Kommunikationsnetz des Fahrzeugs verbunden und ohne Verkabelungsaufwand in die MMS integriert. Dabei ist die Auflösung der *Abhängigkeit von Funktionalitäten* ein interessanter Punkt. Beim Nachrüsten eines Bluray-Laufwerks stellt sich die Frage wie diese Bluray abgespielt wird. Deswegen müssen sich der Abspieler und der Zuspieler gegenseitig finden und konfigurieren können. Der Abspieler kann ebenso beliebig ausgetauscht werden. Komponenten, wie z.B. der Abspieler, die eine eigene grafische Repräsentation mit sich bringen, können die MMS-Oberfläche durch eigene grafische Gestaltungen erweitern.

3.2. Übersicht der Anforderungen

In Abbildung 1.3 wird die Verzahnung von Grundprinzipien gezeigt und dargestellt welche Herausforderungen damit adressiert werden.
Abstraktion erleichtert die Handhabung des MMS-Systems durch Einfügen von Abstraktionsschichten. Diese Schichten ermöglichen einen modularen Aufbau des Systems und finden sich beispielsweise bei der Trennung eines Programmes zwischen Aussehen und Logik oder beim Mehrbenutzerbetrieb wieder. Durch diese Abstrahierung kann der zunehmenden Komplexität entgegengewirkt werden und gleichzeitig durch die Abschottung einzelner Komponenten eine höhere Robustheit erreicht werden.
Die *Zentralisierung* versucht ehemals verteilte funktionale oder physikalische Komponenten auf eine zu fokussieren. Bei Diensten verringert sich so die nötige Anzahl an Steuergeräten und ermöglicht ein dediziertes Sicherheitsprofil. Des Weiteren wird die Komplexität eingeschränkt, da verteilte Systeme meistens mit einem erhöhten Organisationaufwand einhergehen.
Um ein zukunftssicheres MMS-System anbieten zu können, muss dieses *Erweiterbarkeit* auf der funktionalen Ebene als auch hinsichtlich der Komponenten unterstützen. Dadurch wird die Flexibilität des Systems erhöht und unterstützt die Integration von Komponenten. Ebenso werden kostenintensive Vorgänge für Neuerungen im Vorfeld bedacht und durch geeignete Mechanismen versucht, diese einzudämmen.
Die Vereinheitlichung von Softwarekomponenten erleichtert den Austausch und unterstützt ebenfalls die Integrierbarkeit von Komponenten. Auf diese Weise werden bei einem eventuellen Ausfall die Kosten reduziert und die Integrierbarkeit neuer Geräte unterstützt. Auch für die Verwendung von verschiedenen Architekturen unter der Software ergeben sich einige Vorteile. Diese *Homogenisierung* findet auch bei der Schnittstellenbeschreibung Anwendung. In Tabelle 3.1 werden die Anforderungen nach Szenarien geordnet aufgezählt und gesammelt. Im Mittelpunkt stehen die vier Grundprinzipien (Abstraktion (A), Zentralisierung (Z), Erweiterbarkeit (E), Homogenisierung (H)) als Grundaspekte der zukünftigen MMS-Architektur. Es werden jeweils für die jeweilige Anforderung das Hauptprinzip mit einem „X“ und die Prinzipien die dadurch berührt werden mit einem „(X)“ markiert.

Szenario	Anforderungen	A	Z	E	H
3.1.1	Eingabe- und Ausgabekanal	X			
	Grafische Repräsentation	X			
	Minimalisierte Datenübertragung				X
	Bündelung verschiedener Funktionalitäten		X		
	Interaktionsmöglichkeit für den Nutzer	X			
	Zugriff auf fahrzeuginterne Funktionalitäten/Daten	(X)	(X)		X
	Selbstkonfiguration			(X)	X
3.1.2	Mehrnutzerbetrieb	(X)		X	(X)
	Zugangsmanagement für Nutzer		X	(X)	
	Personalisierung	X			
	Intelligentes Display			(X)	X
	Anpassbares Design	X			
	Verschiedene Arten von Eingabekanälen	(X)		X	
	Lose Zuordnung von Eingabe- und Ausgabekanal	X			
3.1.3	Plattformunabhängigkeit	(X)			X
	Integration von fremden Geräten		(X)	X	
	Anpassungen an den Ressourcenverbrauch neuer Geräte		X		
	Verschiedene Formate der Ausgabekanäle	(X)		X	
	Zugangsmanagement für Geräte		X	(X)	
	Vertraulichkeit		X		
	Hinzufügen eines Ein- oder Ausgabekanals			X	
	Unterstützung verschiedener Ausgabeformate	(X)		X	(X)
3.1.4	Dynamische Nachrüstbarkeit			X	
	Offene/standardisierte Schnittstellen	X			
	Allg. Beschreibung von Schnittstellen	(X)			X
	Anpassungen an den Ressourcenverbrauch der Funktionen	(X)		X	
	Bereitstellung von Funktionen für Nutzer		(X)	(X)	X
	Zugangsmanagement für Funktionen		X		
	Erweiterbarkeit der grafischen Repräsentation			X	
	Auflösung von Abhängigkeiten	X		(X)	

Tabelle 3.1.: *Übersicht der Anforderungen, Abstraktion (A), Zentralisierung (Z), Erweiterbarkeit (E) und Homogenisierung (H) (Hauptprinzipien werden mit „X“ und Nebenprinzipien mit „(X)“ gekennzeichnet).*

3.3. Einsatzmöglichkeiten

Das durch diese Nutzungsszenarien vorgestellte Bild einer zukünftigen MMS lässt sich auch zu anderen Zwecken einsetzen. Die große Stärke dieses Ansatzes ist es an einem Punkt mehrere Funktionalitäten zu bündeln und einem Nutzer gesammelt zugänglich zu machen. Nach Belieben können (eigene) Ein- und Ausgabegeräte dafür verwendet werden und diese können zusätzliche Funktionalitäten zur Verfügung stellen, die ebenfalls in den Verbund aufgenommen werden.
Im Folgenden werden verschiedene Einsatzszenarien, bei welchen sich die Stärken der Visi-

on der vorgeschlagenen MMS-Architektur widerspiegeln exemplarisch aufgeführt.

3.3.1. Fahrzeug

Das Einsatzszenario „Fahrzeug“ wurde bereits durch die beispielhafte Anlehnung bei den Nutzungsszenarien in Kapitel 3.1 eingehend betrachtet. Es wird lediglich der Vollständigkeit halber hier nochmals aufgezählt.

3.3.2. Flugzeug

Bei heutigen Flügen ist es üblich, dass im Vorfeld die Anzahl und Aufteilung der Sitze in der Kabine an die Menge und Kategorien der verkauften Tickets angepasst wird. Hinsichtlich des Infotainmentsystems wäre dies mit dem vorgeschlagenen Ansatz problemlos machbar. Für jeden Sitz muss lediglich jeweils ein „Intelligentes Display“ verbaut und mit dem Kommunikationsnetz verbunden werden. Nach der automatischen Einbindung in die MMS-Architektur hat dieses vollständigen Zugang zu allen angebotenen Funktionalitäten. Nun kann an jedem Platz beispielsweise ein Film des Entertainmentangebots der Fluggesellschaft gestartet werden oder die aktuellen Fluginformationen wie beispielsweise die derzeitige Flughöhe abgerufen werden. Möchte der Fluggast seine eigenen Filme anschauen, kann er sein mobiles Gerät ebenfalls einbinden. Durch ein Zugangsmanagement kann dieser die jeweiligen Filme anschließend auswählen und, falls er das wünscht, dies auch dem Sitznachbarn gewähren. Nach der Landung wird das mobile Gerät wieder entfernt und das System ist wieder in seinem ursprünglichen Zustand. Falls nötig können für den anschließenden Flug auch die Sitze wieder neu angeordnet werden.

3.3.3. Hotel

In der Hotelbranche ist es nicht unüblich, Hotelzimmer mit verschiedenen Ausstattungen anzubieten. So kann es eine preiswerte Variante ohne Bluray-Player geben oder eine mit. Bucht ein Gast nun die Variante ohne Bluray-Player, kann er diese bei Bedarf durch seine eigene Geräte erweitern. Es ist ein Zugang zu dem zentralen MMS-System vorhanden mit welchem er sich durch sein mobiles Gerät verbindet. Auf diese Weise kann er nun mit der grafischen Oberfläche des hoteleigenen MMS-Systems interagieren und beispielsweise die Medieninhalte seines Gerätes anschauen.

3.3.4. Krankenhaus

Krankenzimmer in modernen Krankenhäusern sind meist mit Fernseher ausgestattet. Da man den täglichen Umgang mit seinem mobilen Gerät gewohnt ist, kann sich ein Patient mit der MMS-Architektur des Krankenhauses verbinden. Im Anschluss kann er die Funktionalität seines Gerätes als Eingabekanal in der Architektur integrieren. Auf diese Weise kann er dieses weiterhin als Eingabegerät nutzen, um damit die vorhandenen Funktionalitäten, wie

z.B. die Lageänderung des Bettes zu beeinflussen. Auch Gäste könnten ihre Geräte einbinden und den Patienten durch die Übernahme von Kontroll- oder Steuertätigkeiten entlasten oder unterstützen.

3.3.5. Musik/Video Terminal

In den Elektrohandelsketten kann sich in der Audio-/Videoabteilung etablieren, dass sich ein Kunde mit seinem mobilen Gerät in das MMS-System eines Marktes vor Ort integriert und dieses damit nutzt. Das hat den Vorteil, dass man sich Hörproben beispielsweise mit dem eigenen Gerät anhören könnte, ohne dabei an die Geräte der CD/DVD Terminals gebunden sein zu müssen. Dies ist natürlich nicht nur darauf beschränkt, sondern kann mit entsprechender Ausstattung auch mit Videotrailern genutzt werden. Ebenfalls wäre es möglich, über das GUI eine Bezahlmöglichkeit anzubieten über die sogleich ein eventueller Kauf der Ton- oder Filmträger abgewickelt werden kann.

3.3.6. Ticketautomat

Der äußerst interessante Aspekt des Mobile Payments wurde bereits kurz angeschnitten, er lässt sich aber noch auf weitere Szenarien, bei denen ein Bezahlvorgang eine Rolle spielt, ausdehnen. Beispielsweise kann ein Fahrer sein Fahrzeug auf einem Parkplatz abstellen und einfach sein mobiles Gerät mit dem MMS-System des Parkplatzes verbinden. Auch hier hat er anschließend wieder Zugriff auf die grafische Oberfläche und über diese kann der Fahrer sein Parkticket kaufen und den Bezahlvorgang abwickeln. Ähnlich ist es vorstellbar, dass ein Zugreisender den Bahnhof betritt und über sein verbundenes mobiles Gerät neben weiteren Angeboten, wie z.B. Videostreaming, auch sein Zugticket über dieselbe grafische Oberfläche löst. Dies könnte sogar Einsparungen bei den vorhandenen Ticketautomaten bedeuten.
Neben der Möglichkeit zu bezahlen, sollte auch erwähnt werden, dass es für diesen Ansatz auch möglich ist ein Marktplatzsystem, wie es beim Apple iPhone oder Android bereits etabliert ist, einzuführen. Es wäre sogar noch ausbaufähig, denn es wäre ein System vorstellbar, bei dem die Anwendungen nicht einmal mehr heruntergeladen werden müssen, sondern einfach nur noch als Service eingebunden und somit genutzt werden können.

3.3.7. Museum

Bei einem Besuch im Museum ergeben sich ebenfalls mögliche Neuerungen. Jeder Besucher kann anstelle eines Audioführers sein mobiles Gerät benutzen und sogar je nach Ausstattung seines Gerätes sich zusätzlich durch Videomaterial informieren lassen. Neben diesen Basisfunktionalitäten wie z.B. erweiterte Informationen zu Ausstellungsstücken, kann über diesen Ansatz auch die Interaktion mit Exponaten bewerkstelligt werden. Über eine GUI bekommt jedes Exponat eine grafische Schnittstelle, über welches dieses zu bedienen ist. Ein Nebeneffekt wäre die Reduzierung der immensen Instandhaltungskosten der Eingabegeräte an den Exponaten die mit Demonstrationsfunktionalitäten ausgestattet sind.

3.3.8. Spieleterminal

Bei der abendlichen Zerstreuung in einer mit diesem System ausgestatteten Spielhalle kann der Spieler mit Hilfe seines mitgebrachten mobilen Gerätes anstelle eines Spielautomaten dem Spielvergnügen nachgehen. Dabei stellt der eigene gewohnte „Spielecontroller“ einen entscheidenden Vorteil beim Spielen dar und erhöht den Spielspass. Nach der Verbindung mit der Spielhalleninfrastruktur lassen sich über die GUI verschiedene Spiele auswählen und nach einem eventuellen Bezahlvorgang nutzen. Diese Spiele können als reine Softwarerealisierungen vorliegen wie z.B. der Spieleklassiker Pac-Man oder sogar durch Hardwarekomponenten ergänzt werden, wie z.B. ferngesteuerte Spielzeugautos. Auch die neuen Gebiete Cloud Gaming und Gaming on Demand [Hym10] profitieren von diesem System. Hier wird lediglich die Eingabe von dem mobilen Gerät und die Ausgabe auf das mobile Gerät transportiert und die Logik und Berechnung der Spieleaktionen wird auf entfernten Servern durchgeführt. Somit könnte das MMS-System hierfür ein sehr flexibles Frontend darstellen.

3.4. Zusammenfassung

Durch die genauere Betrachtung eines Nutzungsszenarios wurden die Anforderungen an eine zukünftige MMS herausgestellt. Dabei wurde versucht, sich auf dem Architekturniveau zu bewegen und nicht weiter auf mögliche Endgeräte, wie zum Beispiel Neuroschnittstellen, einzugehen. Wie zu sehen ist, sind im aktuellen Stand der Technik im Fahrzeug die ersten Ansätze hinsichtlich Öffnung des Kommunikationsnetzes vorhanden. Jedoch sind diese meistens nur spezielle Erweiterungen an der ursprünglichen, sehr statischen MMS des Fahrzeugs. Es ergeben sich jedoch zahlreiche weitere Anforderungen, die die aktuelle Realisierung nicht abdecken kann. Diese Anforderungen wurden gesammelt und näher betrachtet. Dadurch wurden vier Grundprinzipien bestimmt, durch welche es jeweils möglich ist, den Anforderungen zu begegnen und diese abzudecken. Diese Grundprinzipien sollen bei der weiteren Entwicklung des Konzepts als Orientierungshilfe dienen, damit allen Herausforderungen begegnet werden kann. Im Anschluss daran wurden mögliche Einsatzszenarios beschrieben um die Vielseitigkeit der zukünftigen Erweiterungen des Ansatzes zu zeigen. Im nachfolgenden Kapitel wird nun das Konzept vorgestellt, das alle Anforderungen durch die genannten Prinzipien abdeckt und sich für die genannten Einsatzzwecke eignet.

4. Konzept und Systemarchitektur

Mit Hilfe der zuvor ausgearbeiteten Anforderungen wurden Konzepte erstellt, welche diese abdecken. In diesem Kapitel werden diese Konzepte für eine zukünftige Benutzerschnittstelle in Verbindung mit serviceorientierten Architekturen vorgestellt. Danach wird die Systemarchitektur nach Teilbereichen gegliedert detailliert vorgestellt und zusammengefasst.

4.1. Konzeptübersicht

Abbildung 4.1 gibt einen Überblick des in dieser Arbeit vorgestellten Gesamtkonzepts [EPS10] mit einer möglichen Verteilung auf Hardwarekomponenten. Das Kommunikationsnetz wird dabei nicht explizit dargestellt. Es wird eine Verbindungstopologie zwischen den Komponenten vorausgesetzt. Lediglich wichtige Kommunikationsbeziehungen werden mit Pfeilen dargestellt.

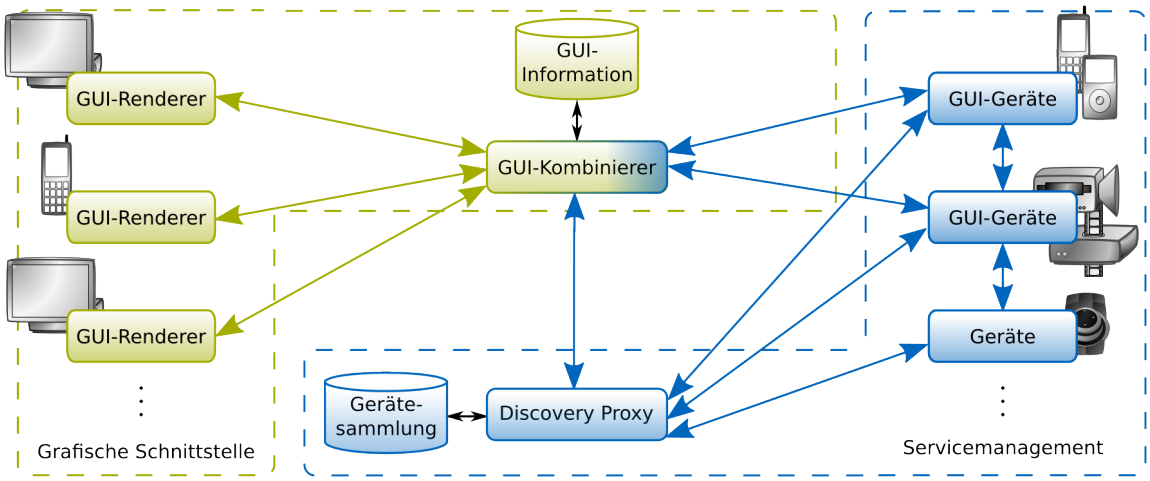

Abbildung 4.1.: *Konzept des MMS-Systems mit exemplarischen Hardwarekomponenten.*

Wie zu sehen ist, wurden die Funktionsblöcke im Vergleich zu Abbildung 1.1 umbenannt. Der linke Block steht nun für die grafische Benutzerschnittstelle (GUI) und der rechte für das Servicemanagement. Das bedeutet z.B., dass ein Eingabecontroller, der zuvor im linken Block seinen Platz gefunden hätte, nun im rechten angeordnet wird, weil dieser seine Funktionalität auch anderen als Dienst anbieten kann. Das MMS-System wird in vier Hauptkomponenten aufgeteilt: GUI-Renderer, GUI-Kombinierer, Discovery Proxy und GUI-Geräte oder Geräte. Im Anschluss werden diese Elemente jeweils nach Themengebiet getrennt und im Detail beschrieben.

4.1.1. Vorteile

Das vorgestellte Konzept bringt wesentliche Vorteile beim Systemdesign von MMS-Architekturen. Dies begründet sich durch die Kombination aus grafischen Benutzerschnittstellen und serviceorientierten Architekturen, mit welcher sich erstmalig dynamische, standardisierte und erweiterbare MMS-Architekturen realisieren lassen. Das Konzept vereinheitlicht alle Funktionalitäten sowie Ein- und Ausgabegeräte. Diese sind zudem selbstbeschreibend. Durch den vorgeschlagenen Ansatz werden die Funktionalitäten sowie Ein- und Ausgabegeräte ohne weitere Anpassungen im gesamten System nutzbar, was das Systemdesign wesentlich vereinfacht. Die zentrale Komplexitätslast herkömmlicher Systeme wird in dem vorgeschlagenen Konzept auf die Einzelkomponenten verteilt. Somit werden diese zu „intelligenten“ Einheiten, welche sämtliches Wissen zur Ansteuerung und Nutzung selbst mitbringen. Diese Informationen müssen somit nicht mehr zentral vorgehalten werden. Nur durch diese Vorgehensweise kann eine einfache Skalierbarkeit als auch Erweiterbarkeit gewährleistet werden. Entgegen der Dezentralisierung der Komplexität wird ein zentrales Management eingeführt, mit dem sich die Abläufe in der Architektur verwalten lassen. Über diese Zentralisierung der Systemverwaltung lassen sich z.B. Mehrbenutzerbetrieb, Geräte-/Funktionsverwaltung und Sicherheit realisieren. Erst damit wird ein so offenes und verteiltes MMS-System verwaltbar und kann seine Vorteile im praxisnahen Einsatz ausspielen.

4.2. Systembeschreibung der grafischen Benutzerschnittstelle

Im Nachfolgenden werden die erstellten konzeptionellen Arbeiten vorgestellt. Dabei wird zunächst auf den Bereich der grafischen Benutzerschnittstellen eingegangen. Anschließend werden die serviceorientierten Architekturen betrachtet. Die Kombination dieser beiden Teilgebiete steht dabei stets im Vordergrund.

4.2.1. Konzeptionierung der GUI

Aufgrund der Anforderungsanalyse in Kapitel 3.2 haben sich die zuvor genannten Techniken als vielversprechende Kandidaten für die Konzeptionierung einer grafischen Schnittstelle hervorgetan. Verschiedene prototypische Realisierungen und Analysen dieser Techniken haben einen Beitrag zur Konzeptionierung geleistet. Das Konzept der GUI wird im Nachfolgenden erläutert.
In Abbildung 4.2 wird der Teilbereich der grafischen Schnittstelle genauer betrachtet.

In dieser Übersicht werden die Hauptkomponenten und die Schnittstellen zueinander gezeigt. Auf diese Weise können die Kommunikationsbeziehungen der grafischen Schnittstelle gesehen werden. Soweit es nicht wichtig für die Erläuterung der GUI-Komponenten ist, wird dabei der Bereich des Servicemanagements außer Acht gelassen.
Der GUI-Renderer setzt sich aus einem Inhaltsgenerator und einer Schnittstelle, der Web-API, zusammen. Der Inhaltsgenerator wird für die Darstellung des grafischen Inhalts genutzt. Um diesen zu erhalten wird beim GUI-Kombinierer durch die Web-API der Inhalt

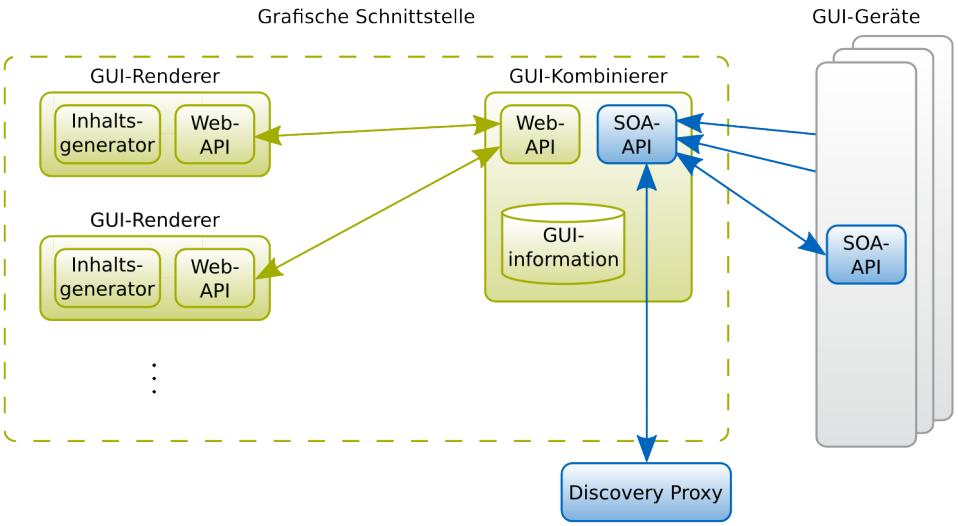

Abbildung 4.2.: *GUI-Komponenten.*

angefragt. Aus diesem Grund besitzt auch der GUI-Kombinierer eine Web-API. Des Weiteren muss er Kenntnis über die verfügbaren Funktionalitäten im Netz erlangen, damit der GUI-Kombinierer den Inhalt aufbereiten kann. Um dieses Wissen zu erfragen, kann er die SOA-API nutzen. Über diese ruft er die verfügbaren Services von dem Discovery Proxy ab. Nachdem er den Inhalt funktional aufbereitet hat, muss er noch das Aussehen ergänzen. Dieses kann er aus einem Datenverzeichnis (GUI-Information) beziehen und ebenfalls dem GUI-Renderer über die Web-API zukommen lassen. Wird aus der GUI heraus ein Service angesteuert, wird dies dem GUI-Kombinierer über die Web-API mitgeteilt. Dieser steuert wiederum den jeweiligen Service über die SOA-API an. Der GUI-Kombinierer dient somit der Vermittlung zwischen dem Servicemanagement und der grafischen Schnittstelle.

GUI-Renderer

Der GUI-Renderer ist die Komponente die für die Darstellung der GUI verantwortlich ist. Das bedeutet, dieser sammelt sämtliche Informationen, bereitet sie auf und generiert eine grafische Ausgabe mit der interagiert werden kann. Von dieser Komponente kann es beliebig viele Instanzen geben. Es bietet sich dabei an, dass es pro Nutzer eine Instanz gibt.
Der Aufbau dieser Komponente wird in Abbildung 4.3 gezeigt. Sie setzt sich aus verschiedenen Elementen für die Interaktion mit Funktionalitäten und für die Darstellung von Inhalten zusammen. Um mit dem GUI-Kombinierer in Verbindung zu treten, wird die Schnittstelle Web-API vorgesehen.

Die **Kontrolllogik** ist verantwortlich für die Koordination von lokalen Nutzereingaben und deren Übergabe an den Inhaltsgenerator. Außerdem leitet sie über die Web-API Nutzereingaben von anderen Komponenten aus dem verteilten System an den Inhaltsgenerator oder lokale Nutzereingaben in das verteilte System. Der **Inhaltsgenerator** kombiniert diese Nutzereingaben zusammen mit den von dem Menügenerator und Anwendungsgenerator abgefragten und gesammelten GUI-Inhalten. Diese Sammlung wird grafisch aufbereitet

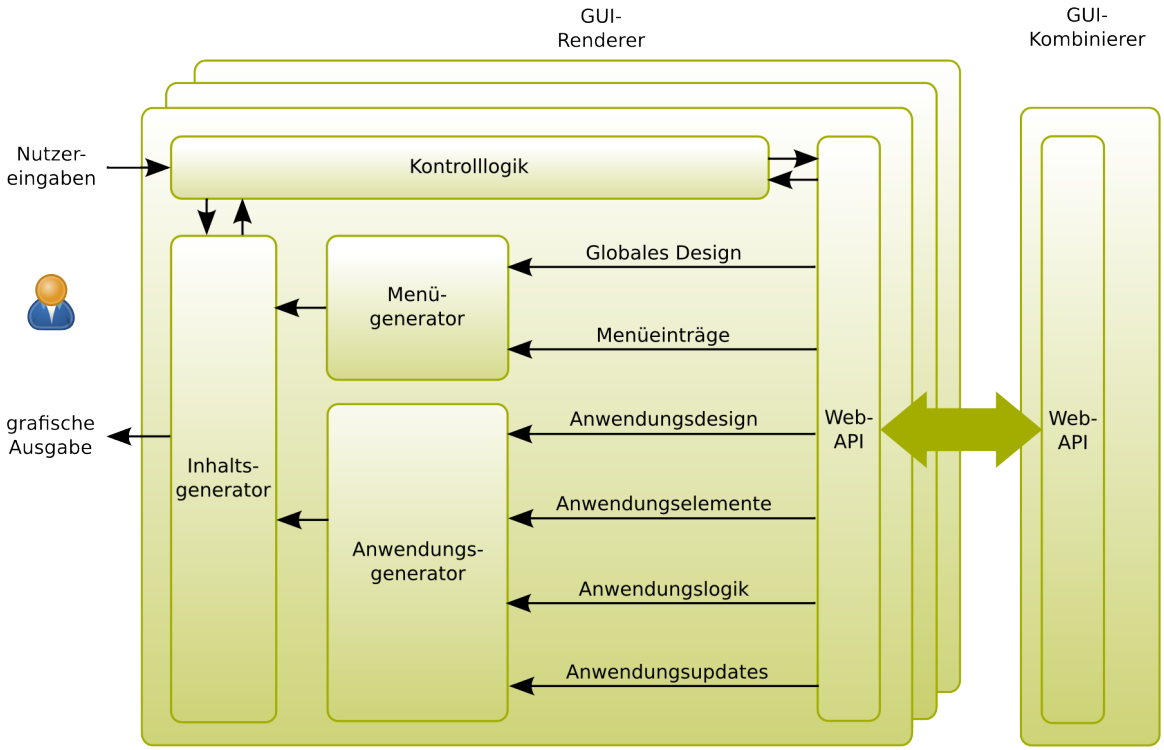

Abbildung 4.3.: *GUI-Renderer.*

und dem Nutzer ausgegeben. Bei der ersten Kontaktaufnahme mit dem GUI-Kombinierer besitzt dieses Element die Zusatzaufgabe über eine vereinfachte Web-API alle Elemente des GUI-Renderers nachzuladen. Erst nach dem Vervollständigen des GUI-Renderers dient die grafische Ausgabe des Inhaltsgenerators als eine funktionstüchtige GUI für das MMS-System. Dazu wird durch den **Menügenerator** das globale Design für die GUI von dem GUI-Kombinierer abgefragt. Dieses wird zusammen mit der Menüstruktur (ebenfalls von dem GUI-Kombinierer) dem Inhaltsgenerator zur Verfügung gestellt. Das **globale Design** stammt von dem GUI-Kombinierer und beinhaltet das Aussehen und Layout der gesamten grafischen Ausgabe. Somit wird das Design von der Logik getrennt und beides kann voneinander unabhängig verarbeitet werden. Der Inhaltsgenerator frägt neben dem Menügenerator den **Anwendungsgenerator** ab. Dieser ist dafür verantwortlich die Informationen über Anwendungsdesign, Anwendungselemente, Anwendungslogik und Anwendungsupdate zu verschmelzen um daraus die grafische Repräsentation einer Anwendung wie z.B. der Videowiedergabe zu generieren. Dabei wird eine Anwendung vom Service zur Verfügung gestellt, die sich hinter dem Aufruf eines Menüeintrags verbirgt. Um die GUI zu vervollständigen, werden diese Informationen ebenfalls an den Inhaltsgenerator weitergegeben. Die Kontrolllogik sucht ebenfalls nach verfügbaren Funktionalitäten im Netz und bereitet diese als **Menüeinträge** auf. So können z.B. die Videowiedergabe mit einem Untermenü und das Radio ebenso mit einem Untermenü in einem Hauptmenü zusammen gefasst werden. Diese Einträge werden durch die Web-API von dem GUI-Kombinierer bezogen und dem Menügenerator zur Verfügung gestellt. Ähnlich wie das globale Design enthält das **Anwendungsdesign** eine Beschreibung eines Aussehens. Nur in diesem Fall wird diese über den GUI-Kombinierer von der jeweiligen Anwendung abgefragt. Das Aussehen der Anwen-

dung wird über die Web-API dem Anwendungsgenerator zur Verfügung gestellt. Dieses beinhaltet Informationen wie beispielsweise die Hintergrundfarbe der Anwendung. Dabei kann durch das Anwendungsdesign das globale Design im Bereich der Anwendung überschrieben werden. Neben dem Design werden auf dieselbe Weise die Anwendungselemente, aus der sich die Anwendung zusammensetzt, an den Anwendungsgenerator geliefert. Dabei handelt es sich um Elemente wie z.B. einen Startknopf. Dynamische Abläufe und Verknüpfungen mit Serviceaufrufen werden durch die **Anwendungslogik** dem Anwendungsgenerator zur Verfügung gestellt. Falls sich durch einen Serviceaufruf Rückgabewerte ergeben, werden diese durch das **Anwendungsupdate** verarbeitet. Dabei leitet der GUI-Kombinierer diese über die Web-API an den Anwendungsgenerator im GUI-Renderer, wenn Daten von der Anwendung auf dem GUI-Gerät an einen Anfragenden losgeschickt werden. Die Daten werden anschließend an der dafür vorgesehenen Stelle in der Anwendung eingesetzt und dargestellt. Die Schnittstelle, die dabei verwendet wird, wird als **Web-API** bezeichnet, weil Konzepte aus den Webtechniken verwendet werden um Daten und Informationen zwischen dem GUI-Renderer und GUI-Kombinierer zu übertragen.

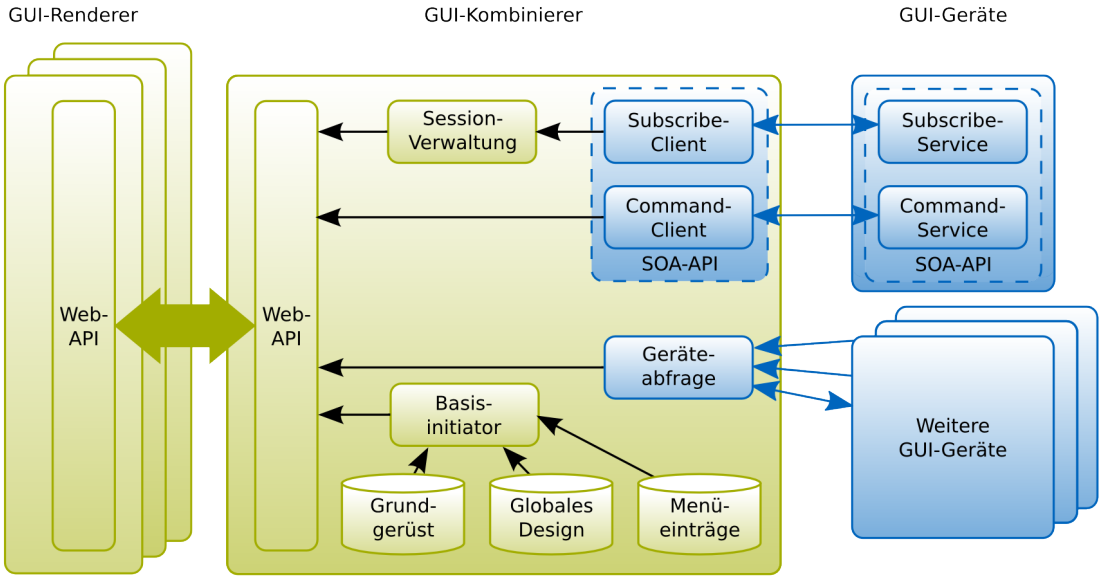

Abbildung 4.4.: *GUI-Kombinierer.*

Wie in Abbildung 4.4 zu sehen ist, vereint der GUI-Kombinierer Schnittstellen zu GUI-Renderern über eine Web-API und zu den Geräten über Servicekomponenten.

Für den GUI-Kombinierer bildet die **Web-API** die Schnittstelle zu dem GUI-Renderer. Über diese kommunizieren alle Elemente des GUI-Kombinierers, die Informationen an den GUI-Renderer weitergeben wollen. Die **Sessionverwaltung** administriert die abonnierten Kanäle aller anfragender GUIs. Falls ein Abonnement über die Web-API von einem GUI-Renderer aus angefordert wird, wird dies an den Subscribe-Client geleitet. Durch den **Subscribe--Client** ruft die Sessionverwaltung Subscribe-Kanäle ab. Dadurch abonniert der Client den jeweiligen Subscribe-Service und damit die eingehenden Benachrichtigungen. Nutzdaten, die in Benachrichtigungen enthalten sind, werden an die Sessionverwaltung weitergegeben. Ähnlich zu dem Subscribe-Client leitet der **Command-Client** Kommandos bzw. Serviceaufrufe an die betreffenden Services. Der Command-Client empfängt die Aufrufe über die Web-

API von dem GUI-Renderer und gibt diese an den Command-Service weiter. Falls eine Antwort von einem Service zurückkommt, wird diese an den aufrufenden GUI-Renderer weitergegeben. Zusammen mit dem Subscribe-Client bilden diese zwei Komponenten die SOA-API des GUI-Kombinierers. Um die Informationen über die Menüstruktur zu sammeln, ruft die **Geräteabfrage** im Speziellen Geräte mit der GUI-Erweiterung ab. Diese melden sich bei der Geräteabfrage mit ihren Kontaktinformationen und ihren Namen und aus diesen Informationen wird eine Menüstruktur erzeugt. Die Struktur wird anschließend über die Web-API an den GUI-Renderer gemeldet. Das letzte Element ist der **Basisinitiator**. Dieser gibt auf Anfrage alle Informationen über den funktionalen Aufbau des GUI-Renderers und einer Basis für die GUI über die Web-API an den GUI-Renderer. Die Information setzt sich aus dem Grundgerüst, dem globalen Design und eventuell aus statischen, d.h. vordefinierten, Menüeinträgen zusammen (siehe Menüeinträge). Das **Grundgerüst** umfasst dabei alle logischen und funktionalen Elemente des GUI-Renderers. Mit anderen Worten, der GUI-Renderer ist zu Beginn ohne jede Funktion. Lediglich wenige Grundfunktionen wie eine minimalisierte Web-API und ein Inhaltsgenerator sind vorhanden. Erst auf Anfrage werden alle Elemente durch den Basisinitiator gesammelt und an den GUI-Renderer übergeben. Dazu enthält das Grundgerüst auch den logischen Basisinhalt für die GUI, welcher nach Bedarf aufgefüllt wird. Zusätzlich zu dem **Grundgerüst** wird eine grafische Beschreibung der Basis für die GUI geliefert, das sogenannte **globale Design**. Somit bekommt das Grundgerüst ein einheitliches Aussehen. Falls ergänzend statische **Menüeinträge** in der GUI gewünscht werden, die nicht von Funktionalitäten im Netz stammen, kann der GUI-Kombinierer diese mit diesem Element zur Verfügung stellen. Auf diese Weise kann beispielsweise ein Konfigurationsmenü für die GUI angeboten werden. Dabei handelt es sich um ein Menü ohne Servicefunktionalitäten, welches aber durchaus noch dynamisch durch weitere Funktionen erweitert werden kann.

GUI-Erweiterungen in den Geräten

Die Lieferanten für die Funktionalitäten sind Geräte. Wenn ein Gerät einen grafischen Beitrag für die GUI liefern möchte, muss es spezielle Erweiterungen unterstützen. Diese Erweiterungen werden in Abbildung 4.5 dargestellt. Die Verbindung zu dem GUI-Kombinierer stellt in diesem Fall die SOA-API dar. Die GUI-Geräte können weitere Services anbieten oder über Clients weitere Services einbinden.

Ein obligatorischer Service ist der **Subscribe-Service**. Bei diesem kann sich ein Subscribe-Client abonnieren. Der Client empfängt ab diesem Moment alle Benachrichtigungen dieses Services. Dies kann genutzt werden, falls in der Anwendungsintelligenz des GUI-Gerätes ein Ereignis auftritt, über das die eingetragenen Interessenten informiert werden sollen. Dies bietet sich besonders bei schnellen Aktualisierungen an, bei denen ein Abfragen der Informationen ineffizient wäre. Ebenso ist der **Command-Service** ein Gegenstück zu dem Command-Client im GUI-Kombinierer. Die Befehle, die von dem Client zu dem Service geschickt werden, leitet der Service zur Bearbeitung an die Anwendungsintelligenz weiter. Die **Anwendungsintelligenz** bildet die eigentliche Intelligenz des Gerätes. Hier findet die Berechnung von Werten oder das Abarbeitung von Aufgaben statt, welche die eigentlichen Funktionalitäten eines Gerätes darstellen. Mit der Anwendungsintelligenz kann die **Abonnenmentlogik** genutzt werden. Diese Logik ist für die Kodierung der Abonnementen

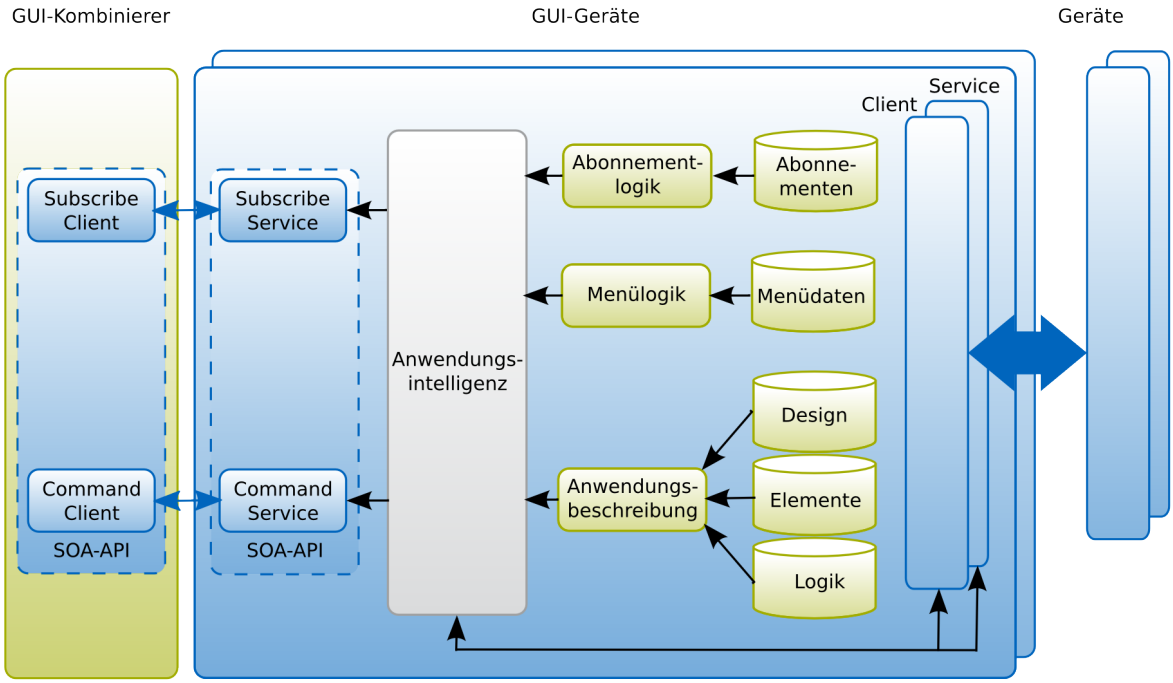

Abbildung 4.5.: *GUI-Erweiterungen für die Geräte.*

verantwortlich. Über den Subscribe-Service können sich beliebig viele GUI-Renderer abonnieren. Um diese auseinanderzuhalten und getrennt bearbeiten zu können, werden diese in dem Abonnementendatenverzeichnis abgelegt und mit Attributen, die beispielsweise die angeforderten Datenströme identifizieren, versehen. Die Abonnementlogik legt ihre Informationen in ein **Abonnementenverzeichnis** ab. Dieses Datenverzeichnis ist für die Aufnahme aller abonnierten GUI-Renderer vorgesehen. Zu jedem Eintrag wird über zusätzliche Attribute festgehalten, welche Benachrichtigungen an diesen gesendet werden sollen. Für zustandsbezogene Änderungen bezüglich des Menüs ist die **Menülogik** vorgesehen. Diese wird ebenfalls mit der Anwendungsintelligenz verwendet. So kann es z.B. sein, dass ein Untermenüpunkt des Gerätes nicht angezeigt wird, weil die Funktionalität nicht verfügbar ist. Dies kann durch die Anwendungsintelligenz abgefragt und weitergeleitet werden. Die Menülogik besitzt das **Menüdatenverzeichnis**. In diesem Datenverzeichnis werden alle Informationen, wie z.B. die einzelnen Menüeinträge oder der hierarchische Aufbau, für das Menü des Gerätes vorgehalten. Bei Bedarf wird die Menüinformation an die Menülogik weitergegeben. Als letztes Element wird die **Anwendungsbeschreibung** von der Anwendungsintelligenz genutzt. Diese Komponente stellt alle Merkmale der Anwendung zur Verfügung. Dabei handelt es sich um das Design, Elemente und Logik der Anwendung. Diese Informationen werden gebündelt und über den Command-Client bei Bedarf dem GUI-Kombinierer übergeben. Die Anwendungsbeschreibung greift dabei auf die Verzeichnisse Design, Elemente und Logik zu. Das **Design** beschreibt das Aussehen der Anwendung, wie z.B. Schriftfarben oder die Position von Elementen. Das Aussehen wird an die Anwendungsbeschreibung geleitet, wenn diese sie anfordert. Durch die **Elemente** werden die Struktur und jedes Element der Anwendung beschrieben. Ein Anwendungselement kann z.B. eine Schaltfläche sein. Wenn die Anwendungsbeschreibung die Elemente weitergeben soll, ruft

sie diese hier ab. Zusätzlich werden mit Hilfe der **Logik** jede Dynamik und alle Abläufe in der Anwendung geregelt. Dazu gehören Serviceaufrufe und deren Verknüpfungen mit Elementen oder Animationen von Elementen. Auch diese Informationen werden bei Bedarf an die Anwendungsbeschreibung weitergegeben. Schlussendlich kann jedes Gerät auf weitere Services über Clients zugreifen oder selbst weitere Services anbieten. Auch GUI-Geräte können auf andere Services zugreifen und um die Funktionen dieser erweitert werden.

4.2.2. Exemplarische Generierung der GUI durch Webtechnologien

Initialisierung des GUI-Renderers

Ein großer Vorteil des vorgeschlagenen Konzepts ist es, dass die gesamte Logik sowie Elemente des GUI-Renderers dynamisch im laufenden Betrieb erweitert werden können. Diese werden zuerst von dem GUI-Kombinierer angefordert. In Abbildung 4.6 wird dieser Vorgang gezeigt.

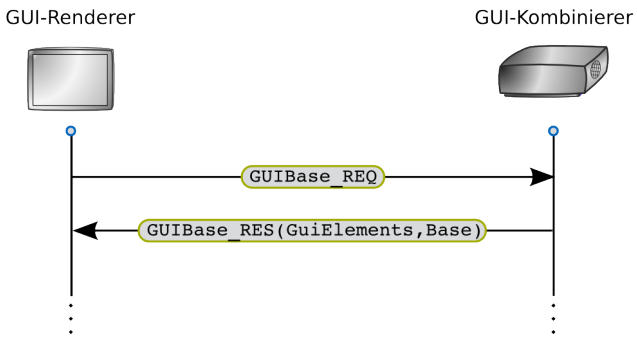

Abbildung 4.6.: *Initialisierung der GUI-Renderers.*

Zuerst fordert der GUI-Renderer über minimalisierte Mechanismen mit der Anfrage `GUIBase_REQ` die funktionalen Elemente des GUI-Renderers und eine strukturelle Basis für die GUI an. Durch die Nachricht `GUIBase_RES` wird dem anfragenden GUI-Renderer die gewünschte Information übergeben. Daraufhin ist der GUI-Renderer initialisiert und kann mit einem Grundsystem die GUI vervollständigen. Durch dieses Vorgehen wird es ermöglicht, dass jeder verwendete GUI-Renderer lediglich eine minimale Untermenge an Grundfunktionen unterstützen muss. Ein weiterer Vorteil ist, dass Aktualisierungen zentral an dem GUI-Kombinierer vorgenommen werden können, und nicht an jedem GUI-Renderer. Diese werden bei der Initialisierung automatisch propagiert.

Aufbau der GUI

Nach der Initialisierung wird der grafische Inhalt dynamisch aus den verfügbaren Funktionalitäten generiert. Dazu ruft der Inhaltsgenerator den weiteren Inhalt durch den die Basis-GUI erweitert wird (siehe Abbildung 4.7) ab.

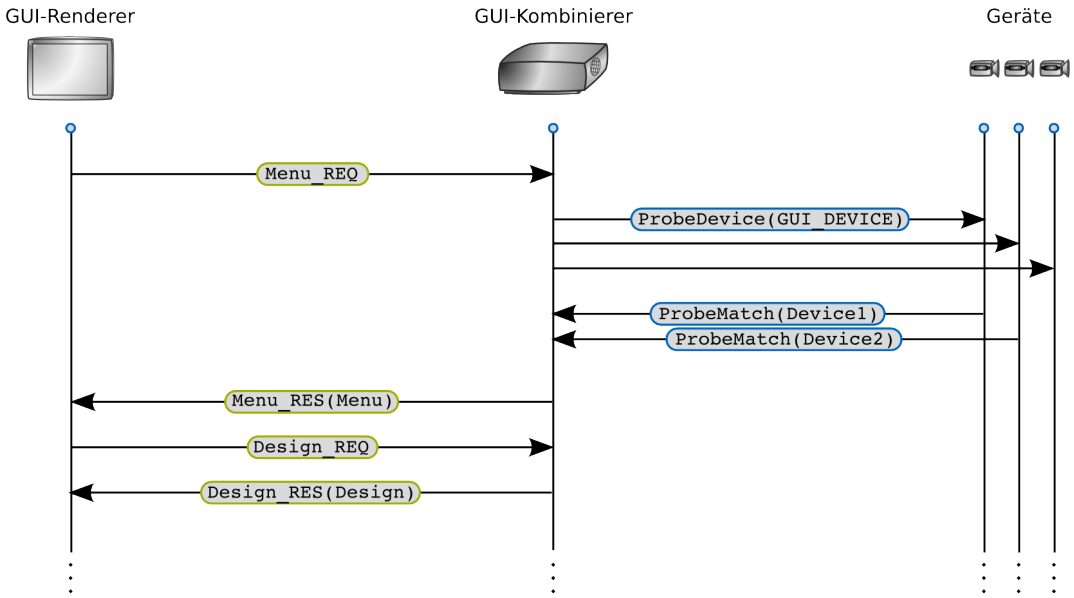

Abbildung 4.7.: *Aufbau des GUI-Grundgerüsts.*

Die Anfrage `Menu_REQ` wird von dem GUI-Renderer über die Web-API an den GUI-Kombinierer gestellt. Mit dieser Anfrage wird die Menüstruktur angefragt, hier könnte auch das statische Menü vom GUI-Kombinierer geladen werden. Beim GUI-Kombinierer wird eine Geräteabfrage im Netz ausgelöst. Es wird dabei nach dem Gerätetypen *GUI_Device* gesucht. Alle Geräte mit diesem Typen antworten mit der Nachricht `ProbeMatch` und mit Informationen über sich, wie dem Namen des Gerätes. Aus diesen Informationen generiert der GUI-Kombinierer eine Menüstruktur und sendet diese mit der Nachricht `Menu_RES` über die Web-API an den GUI-Renderer. Diese Menüstruktur wird in die Basis-GUI eingearbeitet. Das noch fehlende Aussehen wird über die Nachricht `Design_REQ` beim GUI-Kombinierer angefordert. Dieser lässt dem GUI-Renderer das Design zu kommen. Nach der Kombination der Struktur und dem Aussehen kann der GUI-Renderer dem Nutzer eine vollständige GUI anbieten.
Wenn nun ein Nutzer in dieser GUI einen Menüpunkt auswählt, interagiert er dadurch mit einem GUI-Gerät. In Abbildung 4.8 ist dieser Vorgang zu sehen.

Durch die Auswahl eines Menüpunkts wird die Anfrage `LoadDevice` an den GUI-Kombinierer geschickt. Diese Anfrage enthält den betreffenden Menüpunkt bzw. eine Referenz auf ein Gerät und eine Angabe über die momentane Hierarchietiefe in dem Menü. Beim GUI-Kombinierer wird diese Anfrage als `LoadInformation` an das betreffende Gerät mit der Hierarchietiefe weitergeleitet. Nach dem Empfang der Nachricht beim Gerät ergeben sich zwei Möglichkeiten. Entweder die angeforderte Hierarchie besitzt ein Untermenü oder es befindet sich dahinter eine Anwendungsoberfläche. Falls ein Untermenü angefragt wird, werden die betreffenden Informationen über das Untermenü dem GUI-Kombinierer zugesendet. Dieser ordnet die Anfrage dem richtigen GUI-Renderer zu und leitet diesem die Informationen des Untermenüs weiter. Der GUI-Renderer aktualisiert daraufhin seine Inhalte mit diesem Untermenü und stellt sie dem Nutzer dar. Falls in einer Hierarchietiefe eine Anwendung zur Verfügung gestellt wird, werden die betreffenden Informationen der

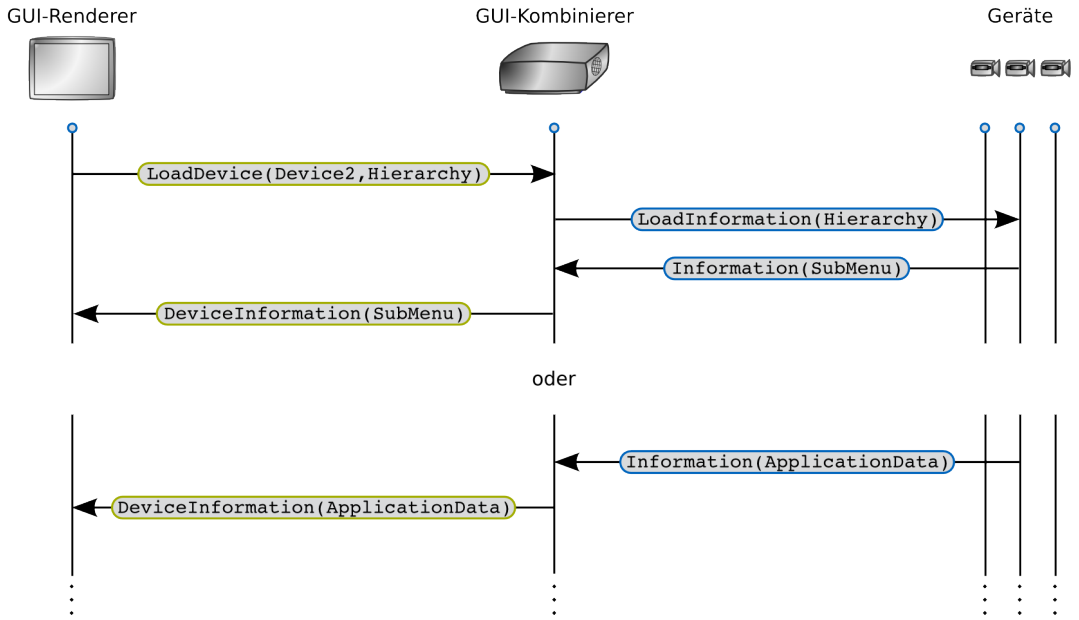

Abbildung 4.8.: *Erweiterung der GUI um Anwendungsinformationen.*

Anwendung an den GUI-Kombinierer weitergegeben und dieser ordnet diese Informationen wiederum dem anfragenden GUI-Renderer zu. Nun kann der GUI-Renderer in seiner GUI dem Nutzer die Oberfläche einer Anwendung anzeigen.
In dieser Darstellung kann ein Nutzer das Abonnieren oder Aufrufen eines Services ausführen. Dies wird in Abbildung 4.9 gezeigt.

Über die Nachricht `Subscribe_REQ` wird zusammen mit den Parametern `Device_ID` und `ServiceName` eine Abonnierung eines speziellen Services bei dem GUI-Kombinierer angefordert. Dafür wird ein Kanal zwischen GUI-Renderer und GUI-Kombinierer aufgebaut. Mit diesem ist es möglich, unaufgefordert Informationen an den GUI-Renderer zu schicken. Die ID für diesen Kanal wird dem GUI-Renderer mitgeteilt. Des Weiteren abonniert der GUI-Kombinierer den gewünschten Service des GUI-Gerätes. Wenn ein Ereignis bei diesem Service auftritt, wird eine Benachrichtigung an den GUI-Kombinierer geschickt. Diese schickt der GUI-Kombinierer über den geöffneten Kanal weiter an den GUI-Renderer, der sie wiederum dem Nutzer grafisch aufbereitet anzeigen kann.
Bei einem Serviceaufruf wird ähnlich vorgegangen. Der Aufruf `ServiceCall` wird vom GUI-Renderer an den GUI-Kombinierer geschickt und von diesem an den betreffenden Service weitergeleitet. Dieser verarbeitet den Serviceaufruf und schickt eventuell eine Antwort zurück.

4.2.3. Nutzdatendarstellung

Die Nutzdatendarstellung betrifft bei der grafischen Schnittstelle visuelles Material wie z.B. Bilder oder Videos. Die Besonderheit bei dieser Darstellung ist, dass diese auch kombiniert mit der GUI ausgegeben werden können. In Abbildung 4.10 ist dies exemplarisch mit einem

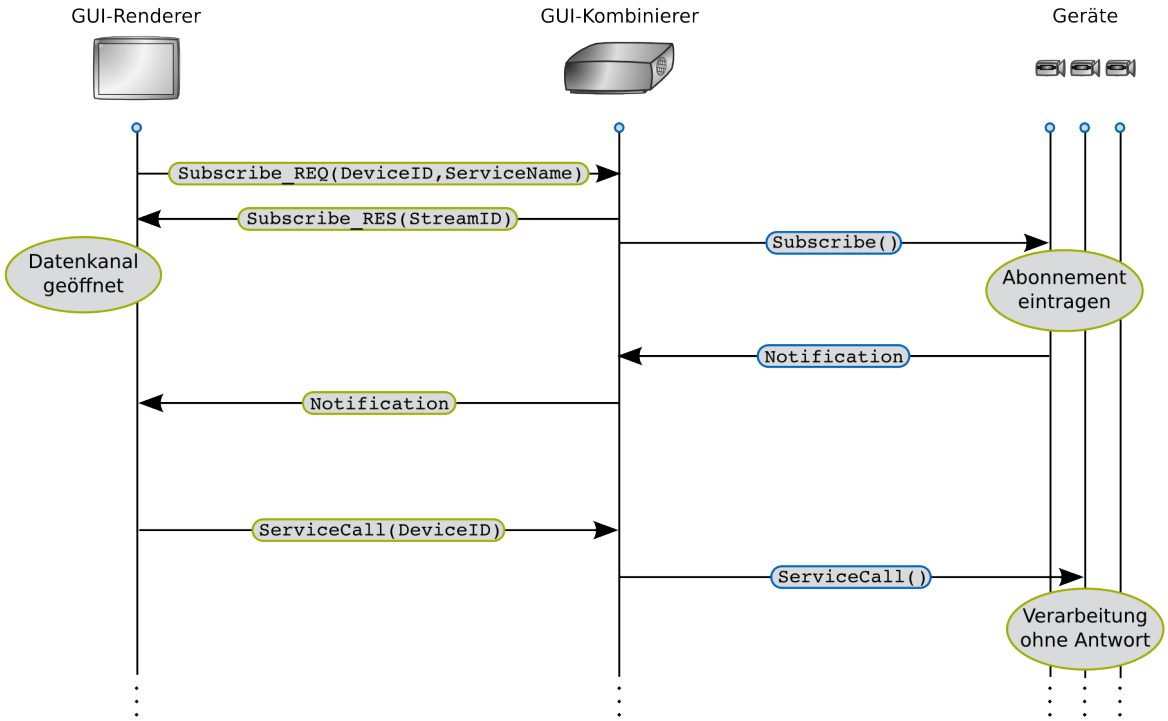

Abbildung 4.9.: *Nutzung eines Gerätes aus der GUI heraus.*

Videoplayer dargestellt.

Bei dem Videoplayer handelt es sich um ein GUI-Gerät. Dieses Gerät bringt damit Anwendungsdesign, Anwendungselemente und Anwendungslogik mit sich. Die Verwendung des Gerätes führt zu einer Ausgabe von bewegten Bildern. Diese können nun auf einem beliebigen Display in Form eines Videos wiedergegeben werden. Dabei würden aber die Anwendungsmerkmale weiterhin in der GUI dargestellt werden. Dies würde dem oberen Szenario (logische Aufteilung) in Abbildung 4.10 entsprechen.
Jedoch ist es gewünscht, dass eine solche Wiedergabe in der GUI zusammen mit den Anwendungsmerkmalen dargestellt wird. So muss sich der Videoplayer auf derselben physikalischen Repräsentation wie der GUI-Renderer befinden (siehe Abbildung 4.10, unten). In der Praxis bietet es sich an, eine Art universellen Abspieler für grafische Nutzdaten in jedem GUI-Renderer vorzuhalten, der diese Aufgabe übernimmt. Durch ihn werden beliebig viele Schichten von grafischen Ausgaben im GUI-Renderer arrangiert. Die Positionierung wird dabei aus den GUI-Inhalten über bestimmte Marker ausgelesen. Des Weiteren ermöglicht die Auslegung dieses Abspielers als GUI-Gerät, dass dieser dieselben Funktionalitäten erlangt wie andere Geräte des Device Profile for Web Services (DPWS) auch, z.B. die Ansteuerung aus der GUI heraus. So könnte beispielsweise neben der Platzierung des Videos durch den HTML-Inhalt, das Video auch über Serviceaufrufe platziert werden.

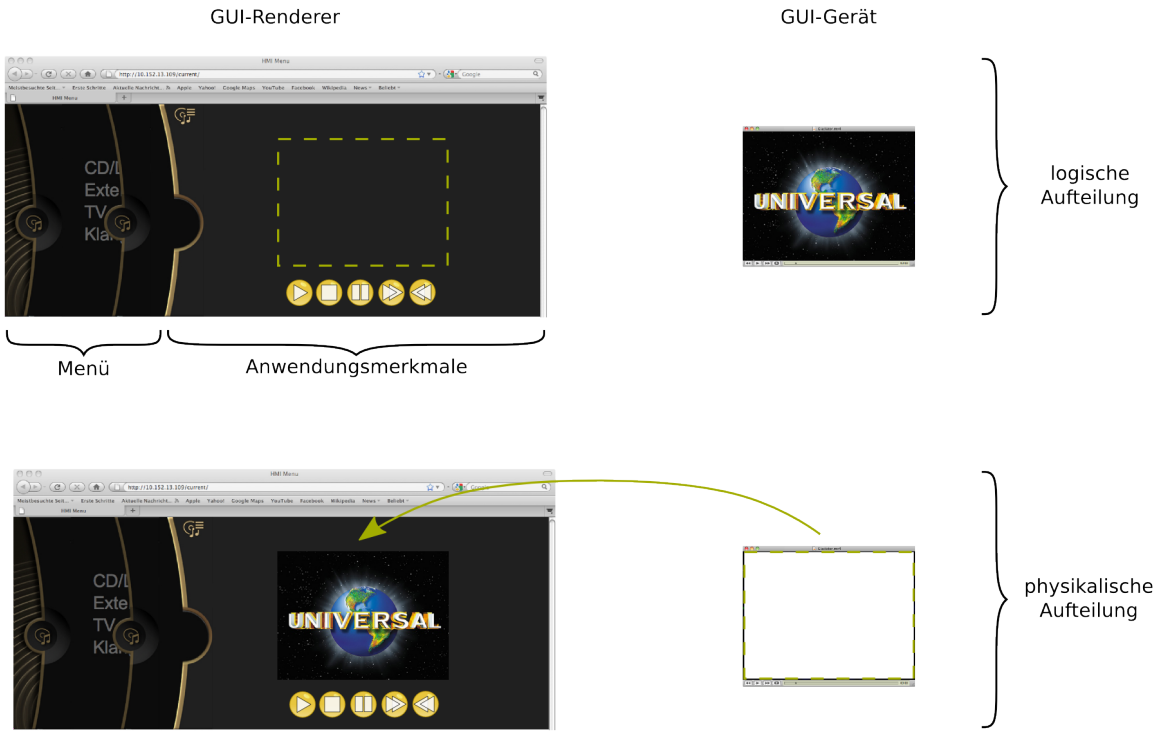

Abbildung 4.10.: *Gegenüberstellung der logischen und physikalischen Nutzdatenausgabe.*

4.3. Systembeschreibung der serviceorientierten Architektur

In serviceorientierten Ansätzen wird die Systemumgebung durch Komponenten modelliert. In den meisten einfachen Szenarien wird jede Komponente über die Anwesenheit anderer in dem System in Kenntnis gesetzt und ist berechtigt diese zu nutzen. Diese Merkmale wurden teilweise beibehalten und erweitert um die zuvor aufgestellten Anforderungen aus Kapitel 3.2 abzudecken. Die nachfolgenden Ausführungen setzen Grundlagen in Web Service Konzepten sowie in DPWS voraus ([Boh09]).

4.3.1. Konzeptionierung des Servicemanagement

Im Nachfolgenden werden die Komponenten des Servicemanagements genauer betrachtet. In Abbildung 4.11 wird eine Übersicht der Komponenten gegeben und in welchen Kommunikationsbeziehungen diese zueinander stehen. Die wichtigsten Komponenten sind dabei die Geräte und der Discovery Proxy (DP), der als Vermittler zwischen den Geräten und dem GUI-Kombinierer agiert.

Bis auf wenige Änderungen, die jeweils erläutert werden, wird versucht den Standard DPWS nicht zu ändern, damit die Kompatibilität weiterhin gegeben bleibt [EPMS10]. Aus diesem Grund wird versucht, die meisten Konzepte als Services zu realisieren die von jedem Entwickler leicht adaptiert werden können. So werden keine Spezialanpassungen nötig und

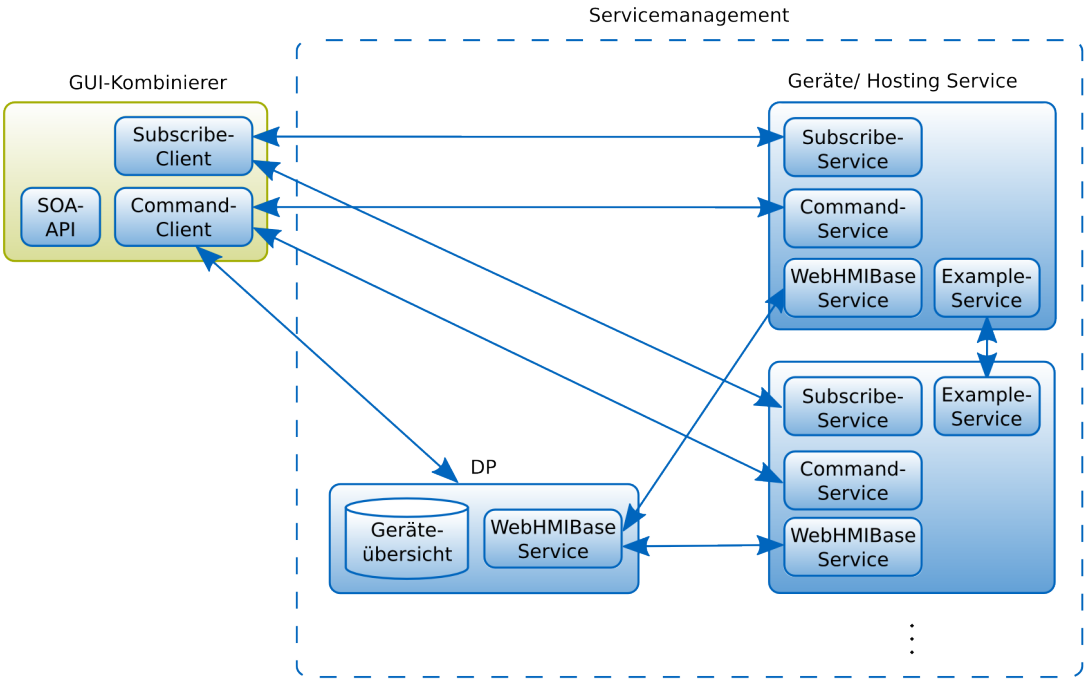

Abbildung 4.11.: *SOA-Komponenten und deren Kommunikationsbeziehungen.*

die bereits entwickelte Software kann weiterhin verwendet werden.

Gerät

In dem DPWS-Standard ist ein Gerät eine abgeschlossene, ausführbare Einheit. Genau genommen ist ein Gerät ein vollständig selbstständig lauffähiges Programm, das unabhängig von weiteren Komponenten ist. Für die MMS-Architektur ist es nötig, dass diese Einheit mindestens einen speziellen Client und einen speziellen Service besitzt. Sie kann aber um beliebig viele Clients oder Services erweitert werden, um Informationen von anderen zu beziehen oder anderen zugänglich zu machen. Diese speziellen Komponenten werden nachfolgend genauer erläutert. Sie werden durch die Anpassungen für die MMS-Architektur für die Verbindungsaufnahme mit dem DP benötigt. Letzterer kann aber auch weitere Clients besitzen und beliebig viele Services anbieten. Für alle Clients wird dieselbe Endpunktreferenz[1] verwendet, da die Zuordnung intern geschieht. Geräten werden Typen zugeordnet, über die die Geräte gesucht und gefunden werden können, z.B. „Videoplayer“. Diese Typen werden für die Serviceverwaltung wie in Kapitel 4.3.2 vorgestellt, verwendet. In der Regel besitzt ein Gerät zusätzlich eine Anwendungsintelligenz. Wie bereits in Kapitel 4.2.1 beschrieben, handelt es sich dabei um die eigentliche Funktionalität des Gerätes. Hier können Auswertungen, Berechnungen oder Abarbeitungen von Befehlen durchgeführt werden. Des Weiteren handelt es sich hierbei um die Zusammenführung aller Clients oder Services in einem Gerät.

[1] WS-spezifische Information zur Adressierung von angebotenen Diensten.

GUI-Geräte: Eine weitere Anpassung ist die Einführung einer speziellen Unterart von Geräten. Dabei handelt es sich um Geräte des Typs *GUI_Device*. Diese wurden ebenfalls bereits in Kapitel 4.2.1 erläutert. Diese DPWS-Geräte werden mit GUI-Elementen erweitert, um somit mit der GUI interagieren zu können.

Service

Die Services sind weiterhin DPWS-konform. Für die vorgeschlagene MMS-Architektur werden lediglich bestimmte Services definiert, die obligatorisch oder optional für die Verwendung in dem vorgeschlagenen System sind. Diese werden im Folgenden vorgestellt.

WEBHMIBase-Service: Dieser Service ist obligatorisch für jedes Gerät, welches an der MMS-Architektur teilnehmen will. Durch diesen Service kann das Gerät seinen Status dem DP mitteilen. Ohne diesen Service wird das Gerät von dem DP ignoriert und es wird von keinem weiteren Gerät der MMS-Architektur gefunden.

Session-Service: Bei diesem Service handelt es sich um einen optionalen Service. Die damit angestrebten Funktionalitäten werden detailliert in Kapitel 4.3.3 erläutert.

Subscribe-Service: Dieses Service ist verbindlich für GUI-Geräte. Über diesen Service können Benachrichtigungskanäle abonniert und die Benachrichtigungen empfangen werden. Ebenso werden darüber GUI-Informationen ausgetauscht (siehe Kapitel 4.2.1).

Command-Service: Auch dieser Service ist für GUI-Geräte obligatorisch. Alle Befehle des GUI-Kombinierers werden über diesen Client an die GUI-Geräte gesendet. Dort werden diese interpretiert und gegebenenfalls eine Antwort zurückgeschickt. Auch dieser Service wird für den Austausch von GUI-Informationen verwendet (siehe Kapitel 4.2.1).

Client

Die Clients werden ebenfalls nicht durch die Änderungen für die MMS-Architektur verändert und entsprechen dem DPWS-Standard. Mit ihnen lassen sich andere MMS-Services nutzen.

WEBHMIBase-Client: Mit diesem Client ist es möglich den Verfügbarkeitsstatus des Gerätes über den WEBHMIBase-Service abzufragen.

Session-Client: Der Session-Client ist das Gegenstück zu dem Session-Service und ebenfalls optional.

Subscribe-Client: Mit dem Subscribe-Client kann sich der GUI-Kombinierer bei den GUI-Geräten abonnieren und deren Benachrichtigungen empfangen.

Command-Client: Für das vorgestellte Konzept ist es wichtig, dass es eine Ansteuerungsmöglichkeit aus den grafischen Benutzerschnittstellen heraus gibt. Für diese Zwecke wurde der Command-Client eingeführt. Dieser Client ist für die Integration in die grafische Benutzerschnittstelle gedacht.

Discovery Proxy (DP)

In der WS-Discovery Spezifikation [BKK+10] wird die optionale Nutzung eines sogenannten Discovery Proxys (DP) spezifiziert. Mit diesem wird es ermöglicht auf Multicast SOAP-over-UDP Nachrichten zu verzichten (siehe Abbildung 4.12). Anstelle dieser können Unicast SOAP-over-HTTP-Nachrichten verwendet werden.

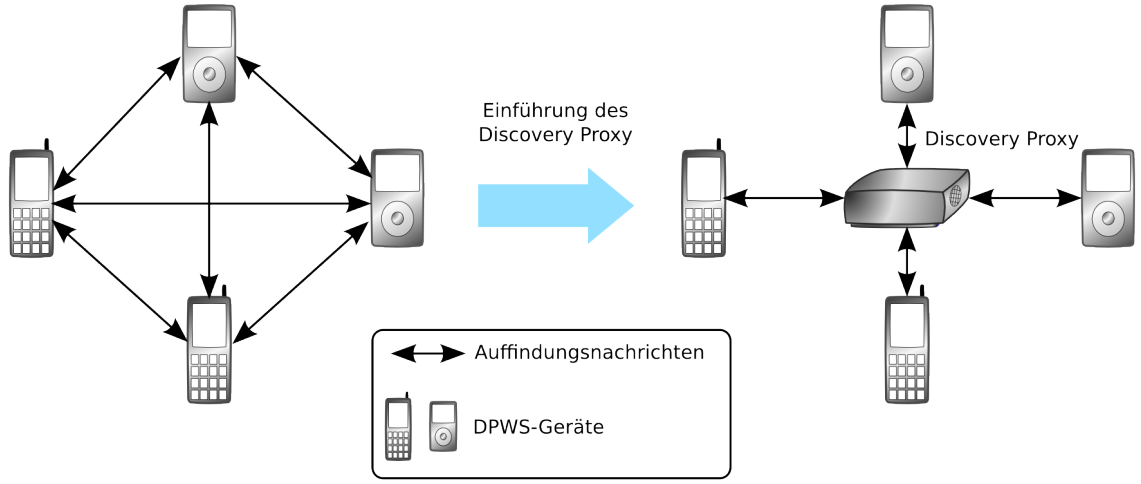

Abbildung 4.12.: *Zentralisierung der Auffindungsnachrichten durch Erweiterung mit dem DP.*

Dies stellt einen Vorteil in sehr großen Netzen dar, um den Nachrichtenverkehr zu verringern. Zusätzlich ermöglicht der DP die Auffindung von Geräten, die sich nicht in demselben Subnetz befinden. Das ergibt sich aus den Einschränkungen des Multicast-Protokolls bei Netzübergängen. Der Discovery Proxy dient dabei als Gateway zwischen diesen. Für die vorgeschlagene Architektur ist die Verwendung eines Discovery Proxys nicht mehr optional, sondern eine Voraussetzung, da durch ihn wichtige infrastrukturelle Erweiterungen realisiert werden. Durch Erweiterungen dieser Arbeit bietet er einen ständigen Überblick an aktiven Geräten für anfragende Clients an. Wie der Name schon andeutet, ist es die Hauptaufgabe des Discovery Proxys die Verteilung der Geräteinformationen zu übernehmen. Er sucht nicht aktiv nach neuen Geräten, sondern jedes neu hinzukommende Gerät meldet sich bei dem DP an. Die neuen Geräte werden in ein zusätzlich eingeführtes Datenverzeichnis im DP eingetragen. In diesem Verzeichnis werden die Geräte markiert, die zur Verwendung bereit sind. Das bedeutet, dass die von der Architektur und vom Entwickler geforderten Abhängigkeiten von anderen Services aufgelöst sein müssen. Nur um die Verfügbarkeit der

Geräte zu testen wird ein Polling Mechanismus angeboten. Mit diesem wird jedes beim DP registrierte Gerät befragt, ob es noch verfügbar ist. In der vorgeschlagenen Architektur arbeitet er jedoch als zentrales Register, in das sich jedes Gerät einträgt.

4.3.2. Exemplarische Serviceverwaltung

In diesem Kapitel wird auf das Zusammenspiel der verwendeten Komponenten des Servicemanagements eingegangen. Dabei werden die Abläufe beschrieben, die hinsichtlich der Verwaltung von Geräten mit Funktionalitäten im System auftreten können.

Serviceregistrierung

Wie zuvor erwähnt, ist die Auflösung der Abhängigkeiten ein Kernpunkt beim Entwurf der Services. Dies durch eine zentrale Komponente zu lösen wäre möglich, aber äußerst unflexibel, weil jede Komponente im Voraus bekannt sein müsste. Dadurch müsste man die Kernkomponente jedes Mal aktualisieren oder eine Neukonfiguration aller Serviceanbieter durchführen, wenn sich die Konfiguration des Systems geändert hat. Meistens werden Geräte mehr als einen Service anbieten die die gleichen Abhängigkeiten besitzen. Deswegen ist es sinnvoll dem Gerät selbst die Abhängigkeitsauflösung zu überlassen. Dadurch können neu teilnehmende Clients nach einem Gerät suchen. Die Abhängigkeitsauflösung beruht auf der Suche nach passenden Gerätetypen. So kann jedes neue Gerät nach den Gerätetypen suchen, die es benötigt. Solange diese nicht gefunden werden, kann die Funktionalität des neuen Gerätes nicht angeboten werden und das Gerät bleibt inaktiv. Deswegen muss im Vorfeld dieser Typ vergeben werden und zusammen mit weiteren Informationen dem Entwickler, der diese Funktionalität integrieren will angeboten werden. In Tabelle 4.1 sind diese Informationen mit Beispielen aufgeführt.

Gerätenamen	Gerätebeschreibung	Typ	Namespace
HP Scanjet G40	Farbscanner für ...	Scanner	http://www.hp.xy/Scanjet/G40
Sony CD GT9UI	CD-Autoradio ...	Autoradio	http://www.sony.xy/CD/GT9UI
JVC KD G351	UKW Radio mit ...	Autoradio	http://www.jvc.xy/KD_G351

Tabelle 4.1.: *Beispieldatensätze für DPWS-Geräte.*

Diese Informationen müssen für jeden Entwickler zugänglich gemacht werden. Dabei handelt es sich zum einen um den Namen des Gerätes, dessen Kurzbeschreibung, den Typ und den Namespace. Der Name und die Kurzbeschreibung können beliebig gewählt werden. Der Typ muss sinnvoll vergeben werden. Mit ihm können die Geräte kategorisiert werden. Zusätzlich gibt es den Namespace, der jedes Gerät eindeutig identifiziert. Dieser Namespace muss neben der Identifizierung auch die Funktion eines Weblinks darstellen. Unter diesem Link sind detaillierte Informationen über das Gerät, dessen Services und Verwendungshinweise hinterlegt. Auf diese Weise kann der Entwickler eines neuen Gerätes auf Informationen zurückgreifen um eventuell weitere Services in sein Gerät einzubinden.

Dynamische Integration eines Gerätes

Für die Integration von Geräten in die MMS werden einige Erweiterungen vorgenommen. Diese sind nötig, um den Anforderungen einer zukünftigen MMS zu begegnen. Die dynamische Integration eines Gerätes wird in Abbildung 4.13 gezeigt.

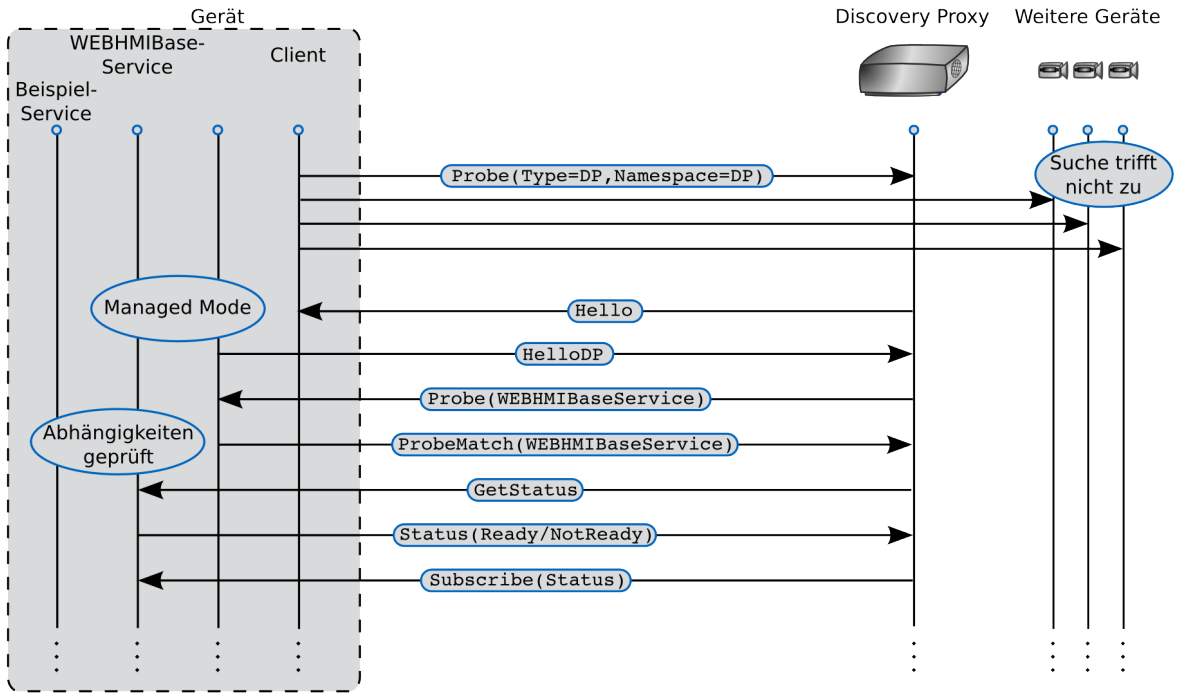

Abbildung 4.13.: *Dynamische Integration eines Gerätes.*

Es werden immer nur Geräte in der DPWS-Architektur bekannt gemacht und nicht einzelne Services. Deswegen ist eine der ersten Aktionen, die ein neu zu integrierendes Gerät durchführt die Suche nach dem Discovery Proxy (DP). Dazu schickt der integrierte Client eine `Probe()`-Nachricht mit den Suchparametern `Type` und `Namespace` an alle vorhandenen Geräte. Da die Suche nach einem DP gestartet wurde, antwortet lediglich der DP mit einer `Hello`-Nachricht dem anfragenden Client. Der Client schaltet daraufhin das Gerät in den *Managed Mode*. Aufgrund dieser Änderung werden Anfragen nach Geräten von dem Client nur noch an den DP gerichtet und nicht mehr per Multicast an alle Teilnehmer im Netz. Das ist nötig damit inaktive Geräte nicht gefunden werden können. Durch deren Auffindung könnten sie verwendet werden und dies wird in einem inaktiven Zustand zu Problemen führen.
Danach meldet sich das Gerät DPWS-konform mit einem `HelloDP` beim DP an. Der DP fragt mit einer `Probe()`-Nachricht nach, ob das Gerät den WEBHMIBase-Service besitzt. Falls es diesen Service besitzt, schickt das Gerät ein `ProbeMatch()` zurück. Nach dem Empfang dieser Nachricht nimmt der DP das neue Gerät in seine Liste von Geräten auf und setzt dieses auf den Status *inaktiv*. Daraufhin sendet der DP den Serviceaufruf `GetStatus` des WEBHMIBase-Service. Als Antwort bekommt er den Status zurück, der entweder *aktiv* oder *inaktiv* sein kann. Dieser Status wird durch den DP in der Geräteliste aktualisiert. Der Status kann von jedem Gerät selbstständig gesetzt werden. Durch ihn wird es ermöglicht

Abhängigkeiten aufzulösen. Mit anderen Worten, wenn ein Gerät auf die Funktionalitäten eines anderen Gerätes angewiesen ist um die eigene Funktionalität zu erfüllen, kann dies über den Status signalisiert werden. Beispielsweise benötigt ein Videoplayer um ein Video wiederzugeben einen Zuspieler. Solange es keine Quelle mit einem Video gibt, wird somit der Videoplayer nicht in der GUI angeboten. Das bedeutet, dass alle Geräte ohne diesen WEBHMIBase-Service vom DP ignoriert werden.
Damit der DP sofort auf Änderungen des Status eines Gerätes reagieren kann, abonniert er den Statuskanal um eine sofortige Benachrichtigung über einen Statuswechsel zu bekommen. Im Anschluss ist das Gerät vollständig integriert und steht allen anderen Geräten mit seinen Funktionalitäten zur Verfügung (vorausgesetzt das Gerät hat einen aktiven Status).

Nutzung eines Services

Nach der erfolgreichen Integration eines Gerätes stehen dessen Funktionalitäten als Service allen anderen Teilnehmern zur Verfügung. In Abbildung 4.14 ist die Nutzung eines Services durch einen Client dargestellt.

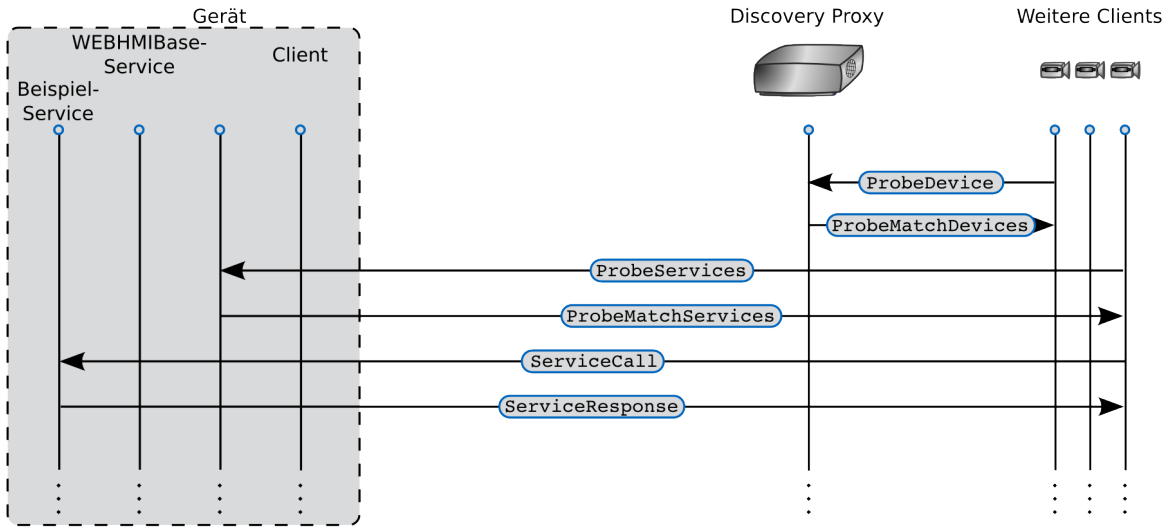

Abbildung 4.14.: *Nutzung eines Services.*

Zunächst fragt der Client bei dem DP nach dem gesuchten Gerät (`ProbeDevice`) mit den Parametern `Type` und `Namespace`. Der DP antwortet, vorausgesetzt das betreffende Gerät hat sich als *aktiv* gemeldet, mit den nötigen Informationen um das Gerät zu finden. Der Client ruft daraufhin bei dem Gerät den gewünschten Service ab. Dieses antwortet mit den angefragten Informationen über den Service. Nun ist es möglich, dass der Client einen Serviceaufruf tätigt und gegebenenfalls eine Antwort darauf zurückbekommt.
Services und Clients können beliebig kombiniert werden. In Abbildung 4.15 ist beispielhaft eine Signalkette mit DPWS-Komponenten dargestellt.

Gerät 1 besitzt einen Client A der den Service A aufrufen möchte. Dieser ist auf Gerät 2 zu finden (1). Damit das Gerät 2 dem Client A den Service A anbieten kann, muss dieser aber mit seinem Client B zuerst den Service B auf Gerät 3 kontaktieren (2). Wurde die Signalkette

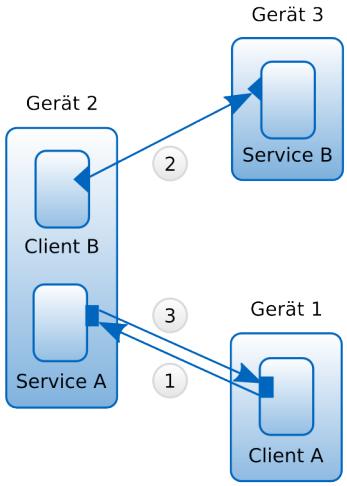

Abbildung 4.15.: *Beispielhaftes Aufbauen einer Signalkette durch DPWS-Komponenten.*

abgearbeitet, kann Client A schlussendlich die Informationen von Service A weiterverarbeiten (3).

Abmeldung eines Gerätes

Bei der Abmeldung handelt es sich um den Vorgang der durchlaufen wird, bevor ein Gerät nicht mehr in der Infrastruktur integriert ist. Dabei muss zwischen einem kontrollierten und einem unkontrollierten Abmelden unterschieden werden. Bei ersterem wird dieser Vorgang aktiv vom Gerät, z.B. beim Abschalten durch den Nutzer ausgelöst (siehe Abbildung 4.16). Wohingegen bei der zweiten Möglichkeit äußere Einflüsse der Auslöser sind, beispielsweise ein Kabelbruch. Bei diesem Szenario muss über regelmäßige Anfragen durch den DP die Bereitschaft von jedem Gerät festgestellt werden. Auf diese Weise kann das Verschwinden eines Gerätes bemerkt werden und entsprechend reagiert werden.

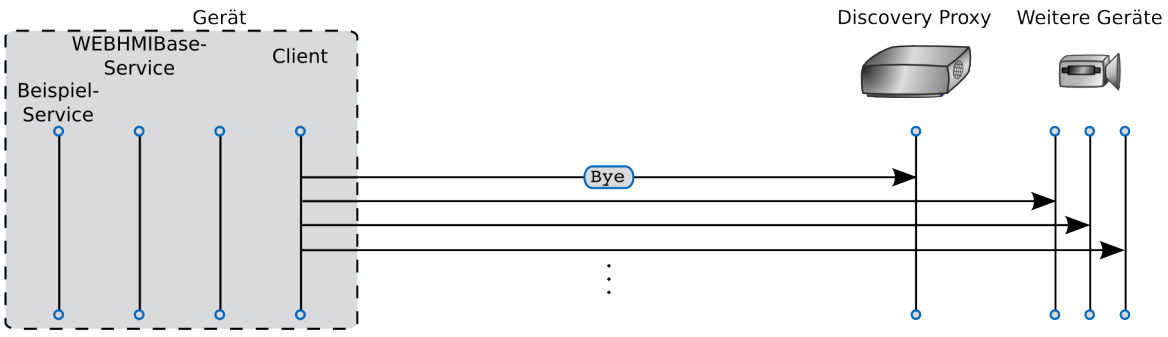

Abbildung 4.16.: *Abmeldung eines DPWS-Gerätes in der MMS-Architektur.*

Abbildung 4.16 zeigt wie durch eine `Bye()`-Nachricht eines Gerätes an den DP und an alle weiteren Geräte die Abmeldung dieses Gerätes bekannt gegeben wird.

4.3.3. Exemplarische Nutzdatensteuerung

In diesem Abschnitt wird auf die Steuerung und Verwendung von Nutzdaten in der serviceorientierten Architektur eingegangen. Dazu werden die verschiedenen Arten der Nutzdaten genauer betrachtet. Es wird die zuvor beschriebene (funktionierende) Infrastruktur vorausgesetzt. Auf die Darstellung des Discovery Proxys wird dabei verzichtet, da er nicht relevant ist und gegenüber dem Client und Service transparent ist. Ebenso wird auf die detaillierte Darstellung von einzelnen Infrastrukturkomponenten verzichtet.

Klassifizierung der Nutzdaten durch Szenarien

Prinzipiell gibt es zwei Möglichkeiten Nutzdaten in dem vorgeschlagenen System zu übertragen. Zum einen können diese innerhalb (In-Band) der DPWS-Kommunikation übertragen werden [LR09] oder außerhalb (Out-Band). Dabei werden eigene Protokolle oder Erweiterungen für die Übertragung verwendet. Beide Ansätze haben ihre Vor- und Nachteile. Ein wichtiger Punkt ist dabei die *Strukturierung des Informationsaustausches*. Dazu zählt ob und wie viel Eigentwicklung in dem Ablauf des Nachrichtenaustausches für die Nutzdaten aufgebracht werden muss. Im Gegenzug reduziert die Verwendung von vorgegebenen Strukturierungen die Menge an Datenverkehr. Der *Overhead durch das eventuell verwendete interne Protokoll*, bezogen auf Protokolle, die für den jeweiligen Anwendungszweck optimiert wurden, kann daher vernachlässigt werden. Was bei der Auswahl der Nutzdatenübertragung nicht vernachlässigt werden darf, ist die *Handhabung des Austausches*. So wird durch DPWS alles zur Verfügung gestellt, was nötig ist, um Nutzdaten zu übertragen. Dies zieht jedoch auch Einschränkungen nach sich. Der Entwickler möchte vielleicht sein eigenes Protokoll verwenden und das ist innerhalb von DPWS nicht ohne weiteres möglich. In Tabelle 4.2 werden die Vor- und Nachteile zusammengefasst und bewertet. Für die Bewertung wird wieder die Skala aus Tabelle 2.1 herangezogen.

Aspekt	In-Band	Out-Band
Strukturierung des Informationsaustausch	++	−
Overhead durch benutztes Protokoll	−	+
Handhabung des Austausches	++	−
Verwendung eines eigenen Protokolls	−−	++

Tabelle 4.2.: *Übersicht der Vor- und Nachteile der Nutzdatenübertragungsarten.*

Wie aus der Tabelle 4.2 hervorgeht, halten sich die Vor- und Nachteile der verschiedenen Arten der Nutzdatenübertragung in der Waage. Deswegen werden in Anhang A.5 drei repräsentative Szenarien vorgestellt, die hinsichtlich ihrem Nutzdatenaufkommen unterschiedlich sind. Durch diese wird eine Klassifizierung eingeführt, wann welche Art der Übertragung am sinnvollsten ist.

Aus diesen Szenarien geht hervor, dass die Wahl ob Nutzdaten In-Band oder Out-Band übertragen werden, sehr von der Anwendung abhängt (siehe Abbildung 4.17). Bedarf es zusätzlicher Funktionen, die die DPWS-internen Protokolle nicht bieten oder sind größere und vermehrt auftretende Datenmengen zu erwarten, bietet es sich an, die Nutzdaten über

ein Out-Band Protokoll zu realisieren. In den anderen Fällen kann von der komfortablen, integrierten In-Band Übertragung durchaus profitiert werden. So liegt diese Entscheidung in der Hand des Entwicklers.

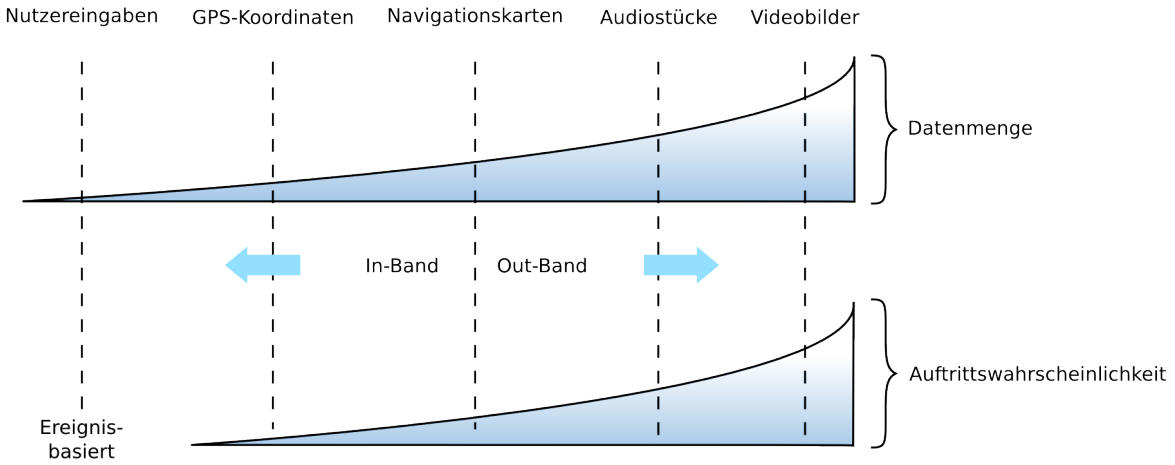

Abbildung 4.17.: *Übersicht der Vor- und Nachteile der Nutzdatenübertragungsarten.*

Neben den Fragen nach der jeweiligen Art der Nutzdatenübertragung ergeben sich aus diesen Szenarien auch Punkte wie die Vorgehensweise bei mehreren Geräten oder der Behandlung von mehreren Clients an einem Service und umgekehrt. Dabei ist es interessant wie die Clients unterschieden werden und wie die Ansteuerung der Services durch die Clients geregelt wird. Diese Fragen werden im Anschluss diskutiert und geklärt.

Einführung eine Sessionmanagements

Um die verschiedenen Konstellationen der Nutzdatenverteilung (siehe Abbildung A.5) abdecken zu können, werden Gruppen eingeführt. Es gibt neben der einfachen Konstellation die Möglichkeit einen Service mit weiteren Clients synchron zu nutzen (Abbildung A.5, links unten). In diesem Fall greifen die Clients auf dieselbe Instanz eines Services zu. Jede Interaktion eines Clients mit dem Service wird von den anderen Clients bemerkt. Diese Kommunikationsbeziehung ist ohne große Anpassungen der Ansteuerung möglich.
Im Beispiel des Videostreamings ist es aber oft nicht gewünscht, dass ein anderer die gerade stattfindende Videowiedergabe beeinflusst. Aus diesem Grund wird die in Abbildung A.5, rechts oben zu sehende parallele Nutzung eines Services, realisiert.
Diese beiden verschiedenen Arten der Servicenutzung lassen sich auch beliebig kombinieren. Eine Möglichkeit wird in Abbildung A.5 rechts unten gezeigt.
Jede grauhinterlegte Fläche repräsentiert dabei eine Gruppe, die den gleichen Videostream empfängt. Durch die ergänzende getrennte Bearbeitung und Verwaltung der Benutzer wird diese Zusammenfassung unterschiedlicher Benutzer auch als Session bezeichnet. Somit existiert für jede Session ein abgeschotteter eigener Kontext. Eine Session kann zu einem beliebigen Zeitpunkt etabliert werden und danach wieder beendet werden. Ebenso ist es möglich einer bereits bestehenden Session beizutreten. Das Sessionmanagement wird vollständig über Services geregelt [EPBS11]. Auf diese Weise kann jeder Entwickler selbst entscheiden,

ob er dies integrieren möchte. Auch die Verknüpfung mit weiteren Services fällt durch diese Realisierung aufgrund der serviceorientierten Natur leicht.

Neue Session etablieren und beitreten: Um Sessions nutzen zu können, müssen diese zunächst erstellt werden. Dadurch wird ermöglicht, dass ein Client von einem Service unabhängig anderer behandelt wird und dieser selbstständig agieren kann. Allerdings können auch weitere Benutzer an dieser Session teilnehmen. Dieser Vorgang wird in Abbildung 4.18 dargestellt.

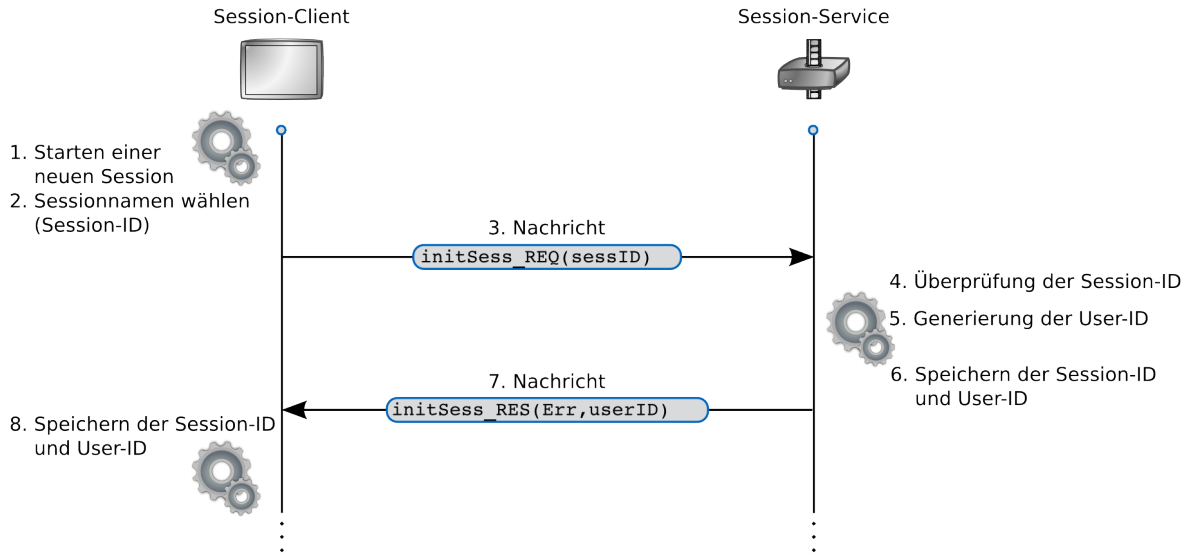

Abbildung 4.18.: *Erstellen einer Session.*

Wenn ein Benutzer eine neue Session starten möchte, verwendet er dazu den Session-Client, integriert in einem beliebigen Gerät. Er wählt dazu einen Namen für die zu erstellende Session aus. Dieser Name dient später dazu die verschiedenen Sessions zu unterscheiden. Der vergebene Name, auch Session-ID (sessID) genannt, kann frei gewählt werden. Durch die Anfrage `initSess_REQ(sessID)` stellt der Client den Wunsch eine neue Session mit der jeweiligen Session-ID zu etablieren an den Session-Service, der im jeweiligen Zielgerät integriert ist. Durch den DPWS-basierten Serviceaufruf werden weitere Informationen mit dieser Nachricht mitgeliefert. So ist es bei der Etablierung der Session wichtig zu wissen, wer die Session erstellen möchte. Dadurch ist es möglich, die verschiedenen Teilnehmer zu unterscheiden und die jeweiligen Serviceaufrufe korrekt zuzuordnen. Ein Unterscheidungsmerkmal ist z.B. die IP-Adresse und der Port des Gerätes des Clients. Diese Informationen können aus der DPWS-Nachrichtenstruktur ausgelesen werden.
Nach dem Empfang dieser Anfrage mit Session-ID beim Session-Service überprüft dieser, ob die ID noch verfügbar ist. Mit Hilfe dieser Überprüfung wird sichergestellt, dass jede Session nur einmal vergeben wird. Damit können Sessions klar getrennt werden und die Teilnehmer eindeutig einer Session zugeordnet werden.
Sofern der Name der Session noch nicht belegt ist, wird dieser reserviert und in eine Listenstruktur beim Service eingetragen. Die Session gilt damit als erstellt und der Name steht

anderen nicht mehr zur Verfügung. Der Session-Service generiert im Anschluss daran nach einem beliebigen zufälligen ID-Erstellungsprinzip eine User-ID für den Sender der Anfrage. Diese ID steht repräsentativ für den Benutzer und seine Zusatzinformationen wie der IP-Adresse und Port. Diese User-ID wird nun der Session zugeordnet und gespeichert.
Der Session-Service besitzt nun als Ergebnis eine Liste mit den derzeitigen laufenden Sessions und deren Teilnehmern. Zusätzlich ist es möglich, über die User-ID die Eigenschaften der jeweiligen Benutzer abzufragen. Das hat den Vorteil, dass bei zukünftigen Anfragen lediglich die User-ID (userID) des Benutzers mit angefügt werden muss. Mit dieser kann der Benutzer identifiziert werden und die zugehörige Session bedient werden.
Dazu wird im nächsten Schritt mit der Nachricht `initSess_RES(Err,userID)` der Session-Client über die Erstellung der gewünschten Session und der User-ID in Kenntnis gesetzt. Ergänzend wird in dieser Nachricht ein Feld für einen Fehlercode (Err) vorgehalten. Konnte die Session z.B. nicht erstellt werden, weil der gewünschte Name schon vergeben ist, kann dies in diesem Feld mit dem entsprechenden Fehlercode signalisiert werden. Daraufhin kann der Client mit einem erneuten Versuch des Sessionaufbaus mit einem anderen Namen reagieren.
Abschließend speichert der Session-Client ebenfalls die Session-ID mit der User-ID und die Session ist somit bei allen Teilnehmern korrekt erstellt.

Vorhandener Session beitreten: Um vorhandenen Sessions beitreten zu können, müssen zunächst Informationen über diese eingeholt werden. Alle Informationen über laufende Sessions werden jeweils serviceseitig gespeichert. Der Benutzer hat somit keine Daten darüber, welche Sessions bei welchen Services mit welchen Session-IDs bereits erstellt wurden. Um darüber Kenntnis zu erlangen, muss eine Möglichkeit zur Verfügung gestellt werden, um bei dem jeweiligen Service dessen laufende Sessions zu erfragen. Die Realisierung dieser Möglichkeit wird in Abbildung 4.19 gezeigt.

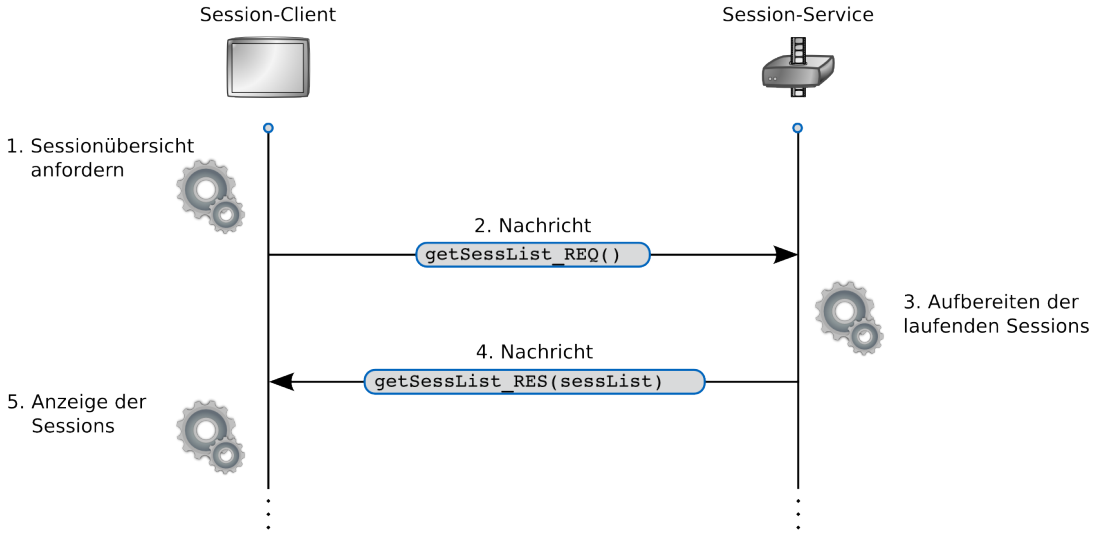

Abbildung 4.19.: *Abfragen vorhandener Sessions.*

Als erstes kann der Benutzer über den Session-Client eine Übersicht bei einem bestimmten Session-Service anfordern. Der Client verschickt dazu die Nachricht `getSessList_REQ()`

an den Service. Durch den Empfang erstellt der Service eine Übersicht (sessList) mit allen, ihm bekannten Sessions. Diese Übersicht wird einer Antwort `getSessList_RES (sessList)` beigefügt und an den Client geschickt. Der Client entnimmt dieser Nachricht die Übersicht, bereitet sie auf und bietet sie dem Benutzer an.
Diese Übersicht kann nun verwendet werden, um eine vorhandene Session auszuwählen und dieser beizutreten. Sie kann jedoch auch einfach zur Statusabfrage von laufenden Sessions benutzt werden. Sind keine laufenden Sessions vorhanden, wird eine leere Übersicht zurückgegeben und der Client kann daraufhin eine neue Session starten.
Nach der erfolgreichen Abfrage der Übersicht aller verfügbaren Sessions eines Services, besitzt nun das Gerät des Session-Clients eine Liste von diesen. Aus dieser kann der Benutzer sich für eine Session entscheiden und eine auswählen um dieser beizutreten. Die Teilnahme an diesen ausgewählten Sessions wird in Abbildung 4.20 dargestellt.

Zunächst signalisiert der Benutzer durch den Session-Client den Teilnahmewunsch an einer Session. Dazu gibt er den Namen der gewünschten Session ein. Mit dieser Information wird die Anfrage `partSess_REQ(sessID)` generiert und an den jeweiligen Session-Service gesendet. Der empfangende Session-Service überprüft zuerst, ob ihm die Session-ID bekannt ist und somit ob die Session bereits existiert. Das Ergebnis führt im Folgenden zu unterschiedlichen Vorgehensweisen.

- **Session-ID existiert nicht:**
  Bei der Überprüfung der laufenden Sessions im Session-Service kann die angefragte Session-ID nicht gefunden werden. Der Service folgert daraus, dass die Session nicht existiert und der Client deswegen nicht an dieser teilnehmen kann. Der Service teilt dies dem Client über die Nachricht `partSess_RES(Err)` mit. In dieser Nachricht ist wiederum ein Fehlercode enthalten, welcher im Session-Client einer Bedeutung zugeordnet wird. Diese Bedeutung kann dem Benutzer daraufhin angezeigt werden. Zu diesem Zeitpunkt ist der Benutzer über das Misslingen der Teilnahme an der Session informiert und kann bei Bedarf den Vorgang mit einer anderen Session-ID wiederholen.

- **Session-ID existiert:**
  Ist die Session-ID bekannt so ist die entsprechende Session erstellt worden und der Benutzer kann an dieser teilnehmen. In einem weiteren Schritt wird nun überprüft, ob der Service aktiv bereits genutzt wird, wie z.B. ein Videostreaming.

  - **Keine aktive Servicenutzung:**
    Wird keine aktive Servicenutzung festgestellt, so wird der Benutzer der Session hinzugefügt. Dazu wird wie bei der Erstellung einer Session eine User-ID erzeugt und der Session zugeordnet. Damit ist der Benutzer nun ein neuer Teilnehmer der Session. Dies wird dem Client mit der Nachricht `partSess_RES(Err,userId)` mitgeteilt. Dieser entnimmt die enthaltene User-ID dieser Nachricht nach deren Empfang. Abschließend speichert er diese mit der Session-ID und damit ist die Teilnahme an der gewünschten Session erfolgreich abgeschlossen.

  - **Aktive Servicenutzung:**
    Falls der Service gerade aktiv genutzt wird, müssen dem Session-Client alle nötigen Informationen für diese Nutzung mitgeteilt werden, z.B. welcher Codec

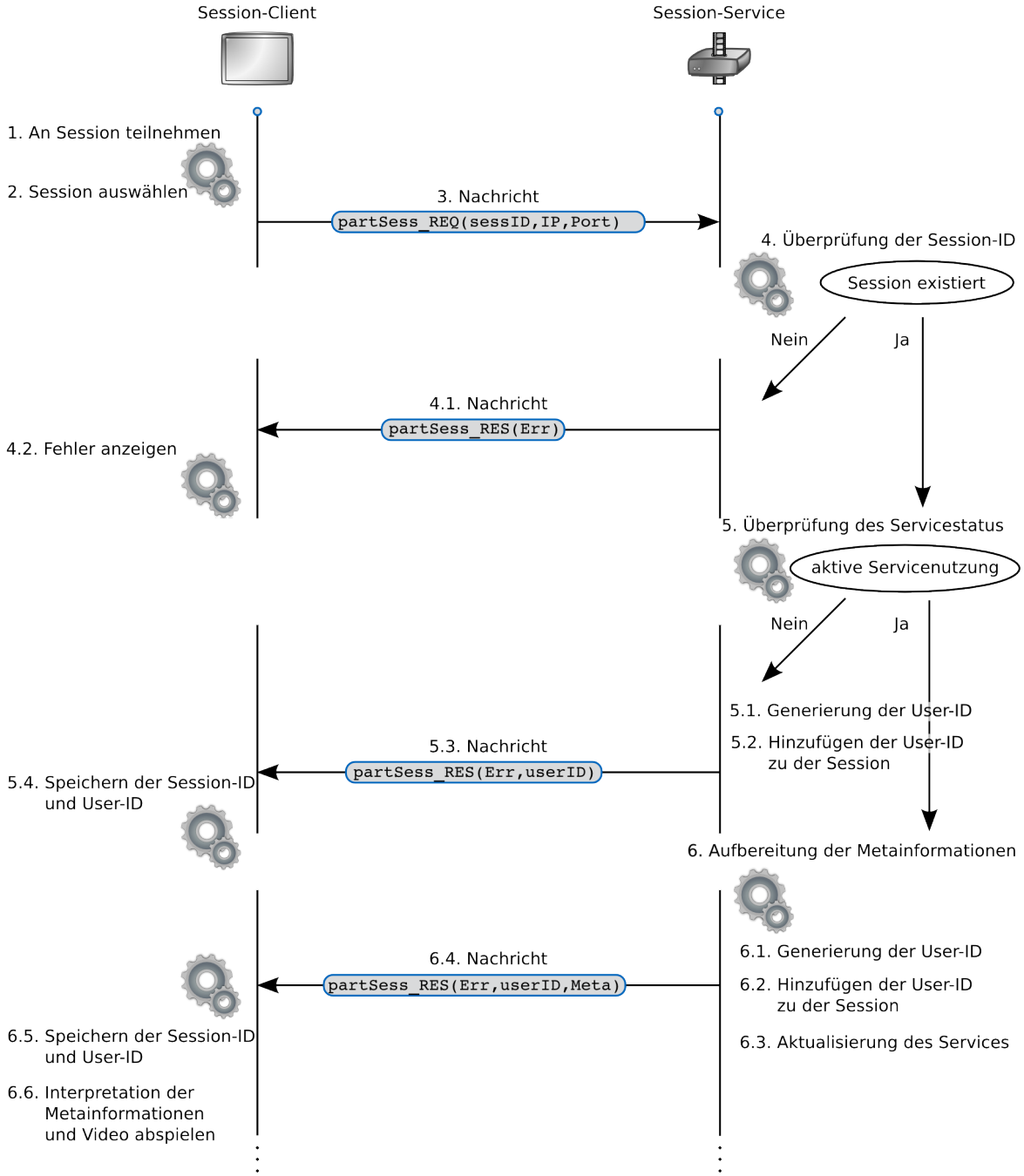

Abbildung 4.20.: *Teilnahme an einer Session.*

oder Auflösung bei einem Videostreaming für die Dekodierung und Darstellung des Videos nötig sind. Ähnlich wie bei keiner aktiven Nutzung wird zuerst eine User-ID generiert und diese der Session zugeteilt. Allerdings werden nun ergänzend Metainformationen (Meta), die für die Nutzung des Services nötig sind, ausgelesen. Des Weiteren wird die Liste der aktiven Clients im Service aktualisiert. Darauffolgend wird die Nachricht `partSess_RES(Err,userID,Meta)`

zusammengestellt. Dabei weist der Fehlercode auf eine bereits aktive Servicenutzung hin. Bei dem Empfang dieser Nachricht speichert der Session-Client die User- und Session-ID. Abschließend wird aufgrund des Fehlercodes die aktive Servicenutzung eingerichtet, z.B. der Empfang und die Anzeige des Videostreams. Die Parameter hierfür werden den Metadaten entnommen.

Session verlassen: Eine letzte wichtige Funktion für das Sessionmanagement wurde noch nicht behandelt. Bisher kann eine Session eingerichtet werden, an bestehenden teilgenommen werden und der Service im Anschluss genutzt werden. Jedoch will der Benutzer nach der Nutzung die Session eventuell verlassen. Dies kann aus verschiedenen Gründen passieren. Entweder will der Benutzer die Nutzung des Gerätes beenden oder aber er will an einer anderen Session teilnehmen. Der Ablauf des Verlassens einer Session ist in Abbildung 4.21 zu sehen.

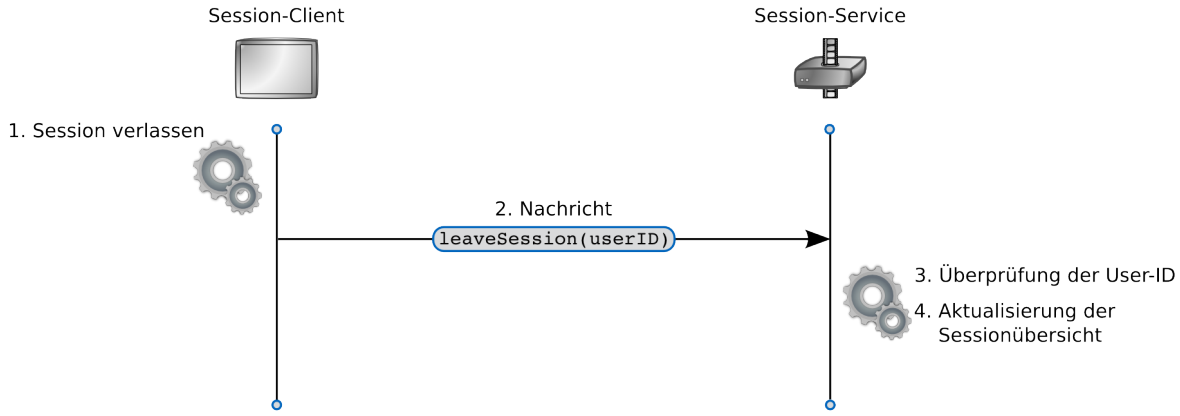

Abbildung 4.21.: *Verlassen einer Session.*

Um das Verlassen einzuleiten besteht für den Nutzer die Möglichkeit dies durch den Session-Client mit der Nachricht `leaveSess(userId)` zu signalisieren. Dazu wird die User-ID des Clients in die Nachricht integriert und an den Session-Service gesendet. Bei Erhalt der Nachricht überprüft der Session-Service anhand der User-ID welcher Session der Client angehört. Der Service löscht den Benutzer aus der Liste der betreffenden Session und damit wird dieser keine Benachrichtigungen oder weitere Metainformationen für diese Session erhalten. Der Benutzer kann jederzeit wieder an dieser Session teilnehmen, falls er das beabsichtigt.
Es kann vorkommen, dass nach dem Verlassen einer Session keine Teilnehmer mehr in der Session vorhanden sind. In diesem Fall wird die Session nicht mehr benötigt und die Session-ID kann gelöscht werden. Danach steht diese ID wieder zur Verfügung und kann bei einer neuen Erstellung einer Session wieder erteilt werden.

Verwendung: In Abbildung 4.22 ist beispielhaft ein Szenario für ein Videostreaming als Nachrichtenablaufdiagramm dargestellt. In diesem Szenario greifen zwei Clients (Videoplayer) auf einen Service (Videostreamer) zu. Der Ausgangspunkt ist eine fertig eingerichtete Session. Das bedeutet der erste Client hat bereits eine Session aufgebaut und der zweite Client hat diese ausgewählt und daran teilgenommen.

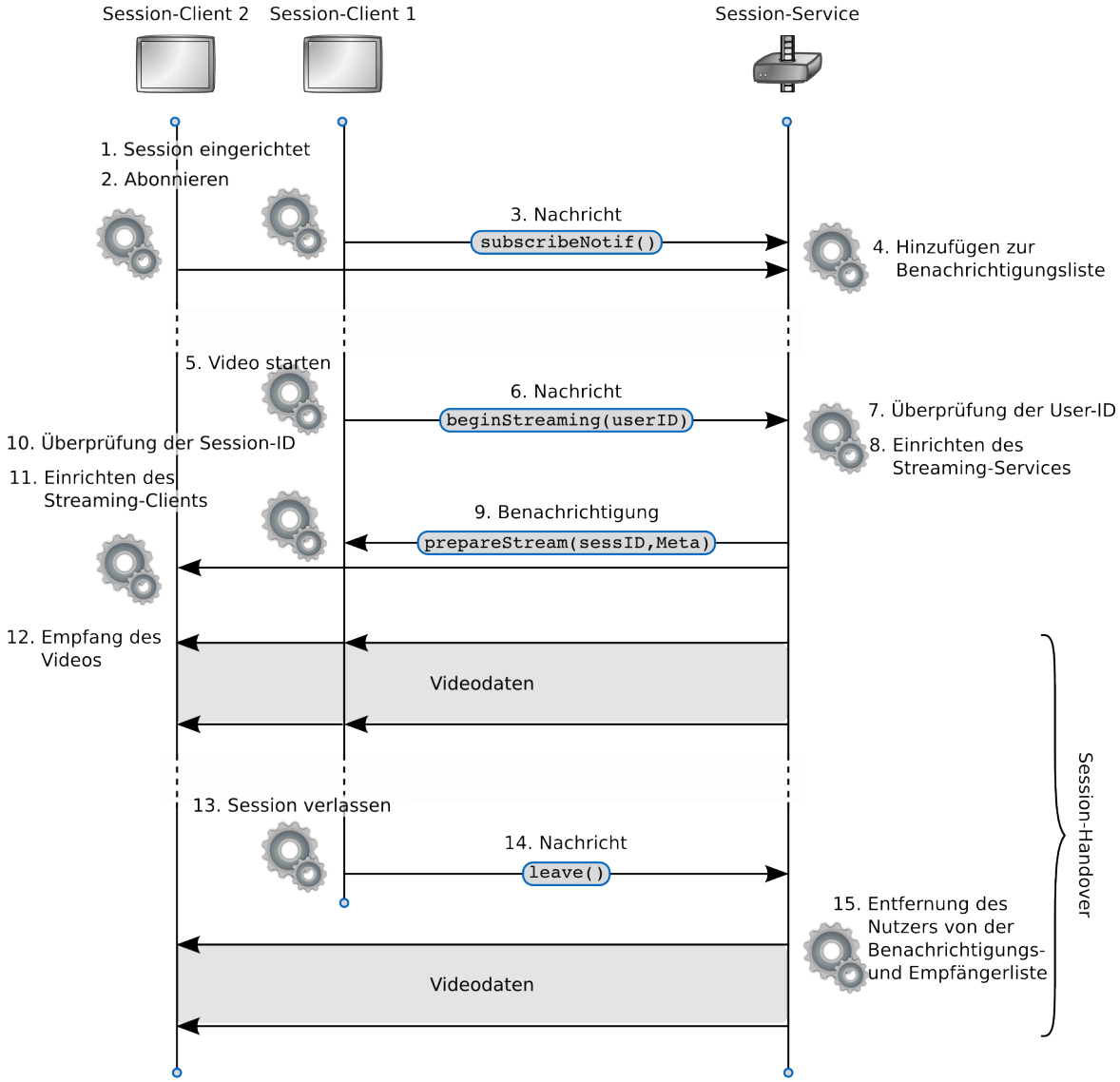

Abbildung 4.22.: *Verwendung des Sessionmanagements bei einem Videostreaming.*

Im Anschluss daran abonnieren beide Clients über eine Nachricht den Benachrichtigungskanal des Session-Service. Der Service fügt diese in seinem Sessionmanagement zu der Benachrichtigungsliste dieser Session hinzu. Ab jetzt werden die Clients über Ereignisse in dieser Session informiert.
Wenn sich nun Client 1 dazu entschließt einen Service aufzurufen, z.B. das Videostreaming zu starten, schickt er dazu eine Nachricht an den Session-Service. Der Service überprüft die Zugehörigkeit des Clients anhand der User-ID und informiert alle Teilnehmer der Session über dieses Ereignis mit Hilfe des Benachrichtigungskanals. Zusätzlich versorgt er diese mit Hintergrundinformationen, die in diesem Fall wichtig sind für den Empfang und die Darstellung des Videos (z.B. den verwendeten Codec). Danach wird der Serviceaufruf durch den Streaming-Service abgearbeitet und startet in diesem Beispiel das Streaming des Videos. Nun können beide Clients den Videostream empfangen und verarbeiten.

Nach einiger Zeit entscheidet sich Client 1 dafür, die Session zu verlassen. Er teilt dies dem Session-Service über eine `leave()` Nachricht mit. Der betreffende Client wird daraufhin aus der Benachrichtigungsliste der Session beim Session-Service entfernt und weder mit Videodaten versorgt noch über Ereignisse dieser Session informiert.
Es kommt zu einer Sessionübergabe. Mit anderen Worten, der ursprüngliche Initiator der Session (Client 1) hat die Session verlassen und diese wurde durch einen anderen Teilnehmer (Client 2) aufrecht erhalten. Während des gesamten Vorgangs waren alle Teilnehmer der Session gleichberechtigt und konnten auf den Service zugreifen.

4.4. Zusammenfassung

Das vorgeschlagene Konzept bietet durch die Kombination eines verteilten Ansatzes für die Generierung von grafischen Benutzerschnittstellen und DPWS für die serviceorientierte Einbindung von Funktionalitäten Vorteile verglichen mit dem aktuellen Stand der Technik. Durch die Orientierung an den Anforderungen aus Kapitel 3 ergibt sich ein abdeckendes Konzept für ein zukünftiges MMS-System, das in einem verteilten System dynamisch Funktionalitäten integrieren kann.
In diesem Kapitel wurde die Konzeptionierung der grafischen Benutzerschnittstelle und des Servicemanagements des MMS-Systems vorgestellt. Des Weiteren wird die Verschmelzung dieser beiden Bereiche gezeigt. Das dadurch vorgeschlagene MMS-System lässt die dynamische Integration von Funktionalitäten von mitgebrachten oder fest installierten Diensten zu. Diese Erweiterungsmöglichkeit begründet sich durch den serviceorientierten Ansatz und fördert des Weiteren die schnelle Entwicklung von solchen Diensten. Die MMS ist durch das zentrale Management mehrbenutzerfähig ausgelegt. Durch ein optionales Sessionmanagement ergibt sich die Möglichkeit in dem Mehrbenutzerbetrieb jeden Benutzer einzeln zu behandeln und auf verschiedene Services abzubilden. So werden Ein- und Ausgabegeräte aufgetrennt und beliebig koppelbar und die funktionsspezifische Entwicklung von Geräten wird vermieden (ganzheitlicher Ansatz). Die durchgeführte Ausarbeitung der GUI der MMS bietet Vorteile wie die minimalisierte Datenübertragung zu den Displays aufgrund der Beschreibung durch Grafikbefehle und dadurch gleichzeitig eine flexible Designmöglichkeit bis hin zur Personalisierung. Des Weiteren wird ein Nachlademechanismus für alle nötigen Inhalte der GUI vorgesehen. Durch die Verringerung der initial benötigten Komponenten können auch auf Standardausgabegeräten mit einem Webbrowser die Funktionalitäten der MMS genutzt werden. Zusätzlich müssen nicht alle verwendeten Geräte aktualisiert werden. Wenn Änderungen an den logischen Elementen des GUI-Renderers oder des GUI-Inhalts vorgenommen werden, werden diese bei der nächsten Initiierung nachgezogen.
Erst durch die Verwendung von Webtechniken werden die eingeführten Realisierungen ermöglicht. Aus diesem Grund liegt es nahe, den GUI-Renderer mit einem Webbrowser zu realisieren und den GUI-Kombinierer mit einem Webserver. Beide erfüllen die konzeptionellen Anforderungen und ergänzen das MMS-System durch weitere Vorteile.
Ein Aspekt der Anforderungen wird bisher noch nicht in dem Konzept berücksichtigt. Es handelt sich dabei um die Informationssicherheit. Da dieser Aspekt einen breiten Bereich abdeckt wird dieser gesondert im folgenden Kapitel betrachtet.

5. Sicherheitsaspekte

In diesem Kapitel wird im Detail auf die für diese Arbeit relevanten Sicherheitsaspekte eingegangen. Ein System, das so offen und verteilt gestaltet ist, muss sorgfältig gegen bösartige Angriffe abgesichert werden. Das Absichern eines IT-Systems ist dabei kein trivialer Vorgang. Zuerst werden dabei mit Hilfe von erprobten Richtlinien die Bedrohungen analysiert und darauf aufbauend die Schutzziele definiert. Meist werden hierfür in den Bereichen der kabelgebundenen oder mobilen Netze sogenannte AAA-Richtlinien (siehe Anhang A.6) verwendet. AAA steht dabei für Authentifizierung, Autorisierung und Abrechnung. Die Gewichtung der AAA-Richtlinien wird in der Regel abhängig von den zuvor analysierten Bedrohungen für jeden Anwendungsfall speziell angepasst und durch weitere spezifische Anforderungen erweitert, um das gewünschte Maß an Sicherheit zu erzielen.
Diese Richtlinien dienen der Erstellung von AAA-konformen Systemen. Damit lässt sich schlussendlich eine Aussage über das Sicherheitsniveau des Gesamtsystems treffen. Die in Kapitel 2.3 bereits erwähnten Sicherheitselemente der verwendeten Techniken und die zuvor genannten Richtlinien werden miteinander kombiniert und als Sicherheitserweiterungen auf die MMS-Architektur angewendet. So soll eine maximale Sicherheit bei vertretbarem Aufwand und Anwendbarkeit für den Nutzer erreicht werden. Die dafür nötigen Sicherheitsaspekte werden erläutert und deren Umsetzung präsentiert.

5.1. Sicherheitsanforderungen in der MMS-Architektur

Durch das vorgestellte Konzept ist es möglich, die MMS-Architektur durch eigene Funktionalitäten zu erweitern. Dies kann auf Seiten der Services oder auf Seiten der grafischen Benutzerschnittstelle geschehen. Der offene und verteilte Ansatz dieser Architektur macht es nötig, Schutzziele [ITU11] zu erfüllen. Zusätzliche Services haben grundsätzlich Zugriff auf alle angebotenen Dienste der Architektur. Gewisse Einsatzszenarien machen es aber nötig, diesen Zugriff zu limitieren. Daher muss eine dedizierte *Zugangskontrolle* geschaffen werden. Die Zugangsregeln sollten zentral an einem Punkt der Architektur gespeichert werden. Aus diesem Grund ist auch eine *Authentifizierung* unumgänglich, somit können die Regeln auch auf die gesicherten Identitäten angewendet werden. Eine bestätigte Identität ermöglicht die *Nichtabstreitbarkeit*, wenn man Vorgänge in der Architektur protokollieren und für spätere Zwecke auswerten möchte. Ein weiterer großer Punkt ist die *Vertraulichkeit*. Alle übertragenen Daten sollten nur berechtigten Services und Nutzern zugänglich sein. Daher empfiehlt es sich Verschlüsselungsmechanismen einzusetzen. Es wäre ohne die *Datenintegrität* immer noch möglich, Daten zu verändern, ohne dass es der Empfänger bemerkt. Neben diesen Aspekten muss in einem System, das mit personenbezogenen Daten arbeitet, die *Privatsphäre* geschützt werden.

5.2. Integration in die grafische Schnittstelle

Da in dieser Arbeit großen Wert auf Standards gelegt wird, ist das Konzept für die grafische Schnittstelle auch an Standards orientiert. Durch die Verwendung von standardisierten Webtechnologien bietet es sich an, auch Sicherheitsmechanismen aus diesem Bereich zu nutzen (siehe Kapitel 2.3.1). Bei der Integration ergeben sich unterschiedliche Szenarien. Da der GUI-Renderer mit einem Webbrowser gleichzusetzen ist, kann zum einen ein Szenario erdacht werden, in dem jeder herkömmliche Webbrowser auf einem beliebigen Betriebssystem als grafische Schnittstelle genutzt werden kann. Bei diesem ergibt sich aber die Problematik, dass keinerlei Erweiterung des Servicemanagements möglich ist. So muss beim Standardwebbrowser auf HTML-Techniken zurückgegriffen werden. Beispielsweise ist bei jeder neuen Verbindung eine nutzerbasierte Authentifizierung nötig (siehe Kapitel 2.3.1). Ansonsten könnte jeder, der das Gerät anderweitig nutzt, auch auf die gesamte grafische Schnittstelle des MMS-Systems zugreifen. Dies kann durch einen eigenen, selbstentwickelten Renderer im zweiten Szenario gelöst werden. Über dies hinweg bietet dieser weitere Funktionen und Vorteile, wie z.B. das Branding durch den Anbieter.
Ansonsten könnte der Standardbrowser die Sicherheitsanforderungen über sogenannte Plugins lösen. Eine solche Realisierung wird in [IR09] vorgeschlagen. Diese hat aber den Nachteil, dass dieses Plugin für verschiedene Webbrowser angeboten werden muss.

5.2.1. Standardrenderer

Viele der heutigen Webbrowser bieten keine weitreichenden Realisierungen der eigenen Authentifizierung durch Zertifikate gegenüber dem Webserver. Deswegen muss dies durch die Nutzung von Nutzername und Passwort gelöst werden. So werden lediglich die Sicherheitsmechanismen der Authentifizierung und der Zugangskontrolle unterstützt. In Abbildung 5.1 wird die Kommunikation zwischen einem Standardrenderer und einem GUI-Kombinierer gezeigt. Dabei wurden die Unterkomponenten in diesen zwei Kommunikationspartnern auf das Nötigste reduziert.

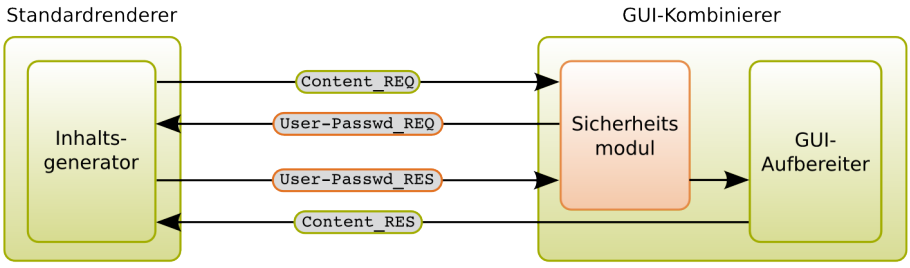

Abbildung 5.1.: *Sichere Integration eines Standardrenderers (Webbrowser).*

Nach der Inhaltsanfrage an den GUI-Kombinierer oder Webserver werden dem Standardrenderer bzw. Standardwebbrowser Inhalte mit einem Eingabeformular zurückgesendet. Mit Hilfe dieses Eingabeformulars kann der Nutzer seinen Nutzernamen und Passwort angeben. Diese Informationen werden in einer Antwort an den Webserver geschickt. Die Übertragung sollte über HTTPS bzw. TLS/SSL gesichert werden. Der Webserver kann nun

den Nutzer mit den in einer Datenbank hinterlegten Daten verifizieren. Ist der Nutzer autorisiert Inhalte zu bekommen, werden diese entsprechend an ihn zurückgeschickt. Dieser stellt den Inhalt mit Hilfe der Renderengine dar. In diesem Fall wird lediglich der Nutzer authentifiziert. Da aber eventuell ein unsicheres Gerät genutzt werden kann, bietet es sich an, die Inhalte nach Vertrauensstellung zu filtern. Beispielsweise möchte vielleicht ein Fahrzeughersteller nicht, dass Designelemente seiner GUI nach außen dringen und würde diese Elemente nicht an solche Teilnehmer weiterleiten.

5.2.2. Eigener Renderer

Die vorgeschlagene Lösung des GUI-Renderers ist nicht auf den Standardbrowser angewiesen. Entscheidend ist nur die standardisierte Generierung der grafischen Inhalte mit Hilfe von webtechnologiebasierten Grafikbefehlen. Aus diesem Grund bietet es sich an lediglich die Renderengine eines Standardbrowsers zu verwenden und diese mit einem gesicherten Client des Servicemanagements zu koppeln. Im Folgenden wird die sichere Einbindung eines Clients durch DPWS vorausgesetzt, die dazugehörige Realisierung wird im nachfolgenden Kapitel 5.3 vorgestellt. Wie in Abbildung 5.2 zu sehen ist, werden die zwei kommunizierenden Partner durch zwei Sicherheitskomponenten erweitert.

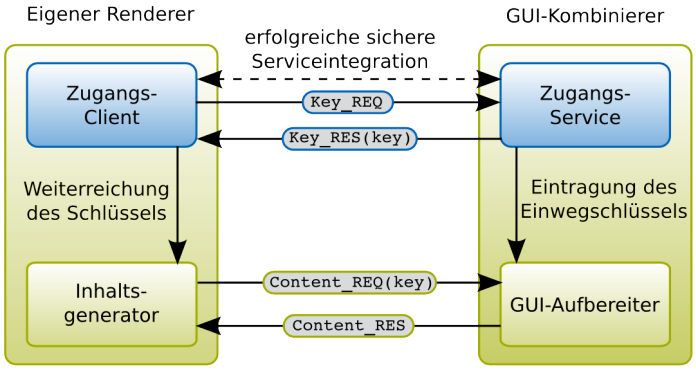

Abbildung 5.2.: *Sichere Integration eines eigenen Renderers.*

Durch die Verwendung eines Zugangs-Clients ist es möglich, ein Gerät mit einer Renderengine zu authentifizieren. Nach diesem Vorgang muss dieser einen Schlüssel anfordern. Mit Hilfe des Schlüssels bekommt der GUI-Renderer Zugang zu dem Inhalt der MMS. Dazu existiert in dem GUI-Kombinierer ein korrespondierender Zugangs-Service. Dieser nimmt die Anfrage entgegen und generiert einen Einwegschlüssel. Der Service gibt diesen an den GUI-Aufbereiter weiter. Dieser trägt den Schlüssel in eine Liste mit allen zugangsberechtigten Teilnehmern ein. Des Weiteren schickt der Zugangs-Service diesen Schlüssel an den Zugangs-Client. Nach dem Eingang des Schlüssels reicht der Client diesen an die Renderengine weiter. Die Engine verwendet im Anschluss den Schlüssel in der Anfrage nach dem Inhalt an den GUI-Aufbereiter. Beim Empfang dieser Anfrage überprüft der GUI-Aufbereiter, ob der Schlüssel in seiner Zugangsliste vorhanden ist. Falls er vorhanden ist, streicht er diesen aus der Liste und schickt dem GUI-Renderer den angeforderten Inhalt der MMS. Nun kann der GUI-Renderer den Inhalt darstellen und mit dem GUI-Aufbereiter darüber wie in Kapitel 4.2.2 interagieren.

Mit dieser Lösung wird die nutzerbasierte Authentifizierung unnötig, da das Gerät über die Sicherheitserweiterungen authentifiziert wird. Dies bietet eine Verbesserung der Bedienerfreundlichkeit. Falls z.B. dieses System in einem Fahrzeug eingebaut wird, müssen die Zugangsdaten nicht bei jedem Starten des MMS-Systems erneut eingegeben werden. Bei Bedarf kann aber auch diese Authentifizierung als Ergänzung verwendet werden.

5.3. Konzeptionierung des sicheren Servicemanagements

Entsprechend den in Kapitel 2.3.2 erwähnten Sicherheitsprofilen und Anforderungen wird gezeigt, dass die DPWS-Sicherheitsempfehlungen Mechanismen beschreiben um Authentifizierung, Vertraulichkeit und Integrität sicherzustellen. Leider werden Aspekte hinsichtlich der Sicherheit die darüber hinausgehen in [OAS10a] und [OAS10b] nicht genauer betrachtet oder festgelegt. So werden Themen, wie z.B. verschiedene Authentifizierungsmodi, Schlüsselverteilung und Ressourcenmanagement nicht durch die genannten Spezifikationen abgedeckt [CLB08]. Da die vorgeschlagene Architektur nicht nur das Minimum der festgelegten Sicherheitsanforderungen erfüllen soll, werden die bereits spezifizierten Sicherheitsfunktionalitäten erweitert. Auf diese Weise ist es möglich, die Abdeckung der Sicherheitsmechanismen auszudehnen und die Standardisierung der Web Services auszuweiten.
Die AAA-Richtlinien (siehe Anhang A.6) bieten sich aus drei Gründen hervorragend als Orientierungshilfe für die Sicherheitsaspekte in der MMS-Architektur an. Als erstes können von den AAA-Prinzipien Techniken und Mechanismen abgeleitet werden, um die Schutzziele abzudecken und so Zugangskontrolle, Nichtabstreitbarkeit und Privatsphärenschutz zu bieten. Das bedeutet, dass Gegenmaßnahmen gegen die in [IT03] aufgezählten Bedrohungen ergriffen werden. Der zweite Grund ist, dass die AAA-Richtlinien keine speziellen oder eigenen Sicherheitsmechanismen vorschlagen, sondern eine Sammlung von Bedingungen, die erfüllt werden müssen, um ein gewisses Sicherheitsniveau in einem System zu erreichen. Der dadurch folgende Zusammenschluss von mehreren Techniken ermöglicht Flexibilität und dynamische Skalierbarkeit in einer Sicherheitsinfrastruktur. Der letzte Grund ist, dass die AAA-Richtlinien Evaluierungsgegenstand vieler IETF-Standards sind. Das bedeutet, dass durch diese ständigen Kontrollen und etwaigen Rückmeldungen von Evaluierenden die Effizienz der Richtlinien steigt und von den resultierenden Verbesserungen das eigene System profitieren kann.
Nachfolgend wird nun detailliert auf die in der vorgeschlagenen Architektur verwendeten Mechanismen eingegangen und deren Anpassungen erläutert, welche durch die AAA-Richtlinien inspiriert wurden.

5.3.1. Übersicht

Die für die vorgeschlagene MMS-Architektur gewählte Sicherheitslösung wird so gestaltet, dass nicht nur die minimalen Anforderungen, die in [OAS10a] empfohlen werden, erfüllt werden. Es werden auch zahlreiche Erweiterungen unterstützt, die schlussendlich eine Typisierung und Evaluierung (Kapitel 6.4.2) durch die AAA-Richtlinien erlauben.
Somit wird eine Reihe von Techniken angewendet und an die vorgeschlagene Architektur

angepasst um die Bereiche Authentifizierung, Autorisierung und Abrechnung stärker abzudecken. Diese Erweiterungen finden sich in den Sicherheitsmechanismen Vertraulichkeit, Integrität, Authentifizierung, Zugangskontrolle und Nachweisbarkeit. Durch die vorteilhafte und sinnvolle Kombination von Sicherheitstechniken und -methoden ergibt sich eine robuste Absicherung der MMS-Architektur. Mit diesen Maßnahmen wird ein sicherer Informationsaustausch zwischen bereits vorhandenen oder spontan teilnehmenden Kommunikationspartnern erreicht. Schlussendlich wird durch diesen Ansatz eine Auftrennung des DPWS-Netzes eingeführt. In dem Netz können sich Geräte mit verschiedener Vertrauensstellung befinden. So kann es eine öffentliche Domäne geben in der sich spontane Teilnehmer aufhalten dürfen, bis hin zu einer nutzerbasierten Abschottung mit der sich beispielsweise eine Kindersicherung realisieren lassen würde. Die Gruppen wissen durch die DPWS-Mechanismen zwar voneinander, können aber nicht miteinander kommunizieren. Durch die Gruppen werden Geräte, Services und Nutzer abgedeckt und somit kann ein sehr feingranulares Regelwerk erstellt werden.
Ein verbindlicher Punkt der Sicherheitsvorkehrungen geht aus den Sicherheitsprofilen von DPWS in Kapitel 2.3.2 hervor. Es handelt sich dabei um die Verschlüsselung auf der Transportschicht. Deswegen wird der Aufbau eines TLS/SSL-Kanals zu einem der zentralen Punkte um eine sichere Kommunikation zwischen Client und Service zu gewährleisten. Die Einrichtung einer TLS/SSL-Verbindung bedarf der koordinierten Zusammenarbeit von Techniken wie Verschlüsselung, Hashfunktionen für die Integrität, digitaler Signaturen und weiteren Techniken für den Schutz von Daten, wie z.B. einer Public-Key-Infrastructure (PKI). Die Verwendung einer PKI in Verbindung mit einer Zertifizierungsstelle (engl. Certificate Authorization, CA) ist eine weitverbreitete Lösung zur Sicherstellung von Identitäten. Die zuvor erwähnten kryptografischen Anforderungen werden von einer PKI ebenso abgedeckt wie auch das Management der Zertifikate. Dabei kann es sich um die Generierung, Validierung und den Widerruf dieser Zertifikate handeln. Die darauf basierenden Sicherheitsmechanismen werden angewendet, um eine ergänzende Validierung der Daten durchzuführen oder eine Verschlüsselung durch den TLS/SSL-Kanal zu unterstützen.
Die Sicherheitserweiterungen der vorgeschlagenen Architektur erfüllen die Anforderungen von Geräten mit eingeschränkten Ressourcen, wie z.B. mobiler Geräte oder eingebetteter Hardware. Zudem werden Systemaspekte, wie z.B. die Selbstorganisation die durch DPWS angeboten wird, durch die Sicherheitserweiterungen abgedeckt. Aus diesem Grund wird die bisherige MMS-Architektur um die Komponenten AAA-Server und CA erweitert.
Bei dem AAA-Server handelt es sich um eine zentrale Einheit, in der alle Sicherheitsfragen wie zum Beispiel die Validierung von Nutzername/Passwort, Regelung der Zugangsrechte, und parametrisierte Anfragen (nach Zugangsrechten) abgearbeitet werden. Für jede Komponente mit der kommuniziert wird, wird ein unabhängiger TLS/SSL Kanal aufgebaut. Wie in Tabelle 5.1 zu sehen ist, setzt sich der Server aus mehreren Komponenten zusammen.

Die CA ist für die Verifikation, Validierung, Signierung sowie den Widerruf von Zertifikaten und Generierung von Schlüsseln zuständig. Durch die Erweiterung mit diesen Komponenten wird die Kommunikation zwischen Client und Server verändert. Dies spielt eine wichtige Rolle für Anfrage-/Antwortnachrichten zwischen den Teilnehmern. Wie die Komponenten beispielhaft in eine solche Architektur integriert werden können, ist in Abbildung 5.3 zu sehen.

Komponente	Beschreibung
Teilnehmerverzeichnis	Hierarchische Baumstruktur
Datenbank	Eingebettete Datenbank für Verzeichnisprotokolle
Sicherheitsmanager	Modul für die Absicherung der Nachrichten
Verzeichnisserver	Einheit für die Bearbeitung von Verzeichnisabfragen
Zugangskontrollliste	Zugangsregeln in Form von Datenbankeinträgen

Tabelle 5.1.: *Übersicht der funktionalen Komponenten eines AAA-Servers.*

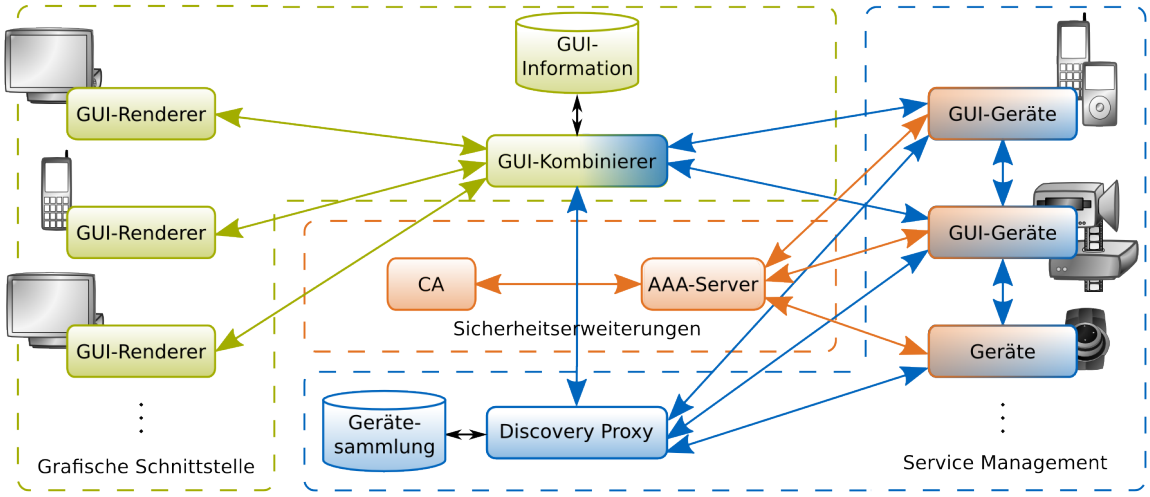

Abbildung 5.3.: *Integration der Sicherheitserweiterungen in das Gesamtsystem.*

Der rote Bereich kennzeichnet die neuen Sicherheitskomponenten. Neben dem physikalischen Integrieren dieser Komponenten ergeben sich Änderungen der Abläufe zwischen diesen und den ursprünglichen Komponenten. Wie die Abläufe durch die Sicherheitserweiterungen für die AAA-Mechanismen im Detail aussehen, wird im Folgenden betrachtet.

5.3.2. Authentifizierung

Die wichtigste Grundlage für den Authentifizierungsvorgang ist die Verteilung von X.509v3-Zertifikaten [ITU10b] an jeden Teilnehmer der MMS-Architektur. So kann sich jeder Client selbst gegenüber anderen authentifizieren und die Möglichkeit der Vertraulichkeits- und Integritätsverifikation bei anderen sicheren Services nutzen. Die Authentifizierung wird über den Austausch, die Validierung und Signierung von Identifikationsnachweisen aus dem X.509v3-basierten Zertifikat durchgeführt. Jedes Gerät in dem sicheren Netz muss ein eigenes und einzigartiges, von einer verantwortlichen CA ausgestelltes Zertifikat, besitzen. Das Zertifikat eines Gerätes beinhaltet einen eindeutigen Namen (engl. Distinguished Name, DN) für das Gerät, den Ort der Zertifizierung, den Ort der Widerrufsliste und den öffentlichen Schlüssel des Gerätes. Es enthält noch weitere Informationen zur Generierung des Zertifikats, aber diese Daten sind für den eigentlichen Authentifizierungsvorgang nicht relevant.
Bevor ein Service verwendet wird, werden zum Zwecke der gegenseitigen Authentifizie-

rung die betreffenden Zertifikate zwischen dem anfragenden Client und dem Service ausgetauscht. So können sich beide Teilnehmer über die Identität des jeweils anderen sicher sein (*gerätebasierte Authentifizierung*). Des Weiteren wird aus dem Zertifikat des Clients der DN extrahiert, um mit diesem über die Zugriffsrechteregelung eine *servicebasierte Authentifizierung* durchzuführen. Neben der bereits genannten Authentifizierung besteht die ergänzende Möglichkeit, durch eine Legitimierung mit Nutzername und Passwort über HTTP eine letzte *nutzerbasierte Authentifizierung* durchzuführen. Zusammengefasst (siehe Tabelle 5.2) werden mit dem Client während der Authentifizierung drei Überprüfungen durchgeführt.

Überprüfung	Parameter
Gerätebasiert	X.509v3-Zertifikat
Nutzerbasiert	Nutzername und Passwort (optional)
Servicebasiert	Servicenamen (DN extrahiert aus dem Zertifikat)

Tabelle 5.2.: *Authentifizierungsmethoden für einen Client.*

Vertraulichkeit ist ein weiteres wichtiges Schutzziel und dieses wird ebenfalls durch die Zertifikate abgesichert. Um Zugriff auf Informationen oder die Integrität dieser sicher zu stellen, z.B. um sogenannte Man-in-the-middle Angriffe abzuwehren, werden Mechanismen zur Ver- und Entschlüsselung eingesetzt. Für die vorgeschlagene Architektur bedeutet das, dass die CA die privaten Schlüssel und die dazugehörigen öffentlichen für jeden Teilnehmer generiert. Der öffentliche Schlüssel wird zusätzlich mit weiteren Informationen ergänzt und als Zertifikat zusammen mit dem privaten Schlüssel als Paar an den jeweiligen Teilnehmer ausgegeben. Dieser speichert diese Informationen lokal und kann somit die Verschlüsselung aller Nachrichten durchführen. Dazu wird mit Hilfe des Zertifikats eine TLS/SSL-Sitzung [DR08] aufgebaut und für die sichere Übertragung verwendet. Ein weiterer Sicherheitsmechanismus in diesem Zusammenhang ist die Sicherstellung der Integrität der Daten. Damit soll verhindert werden, dass Informationen während der Speicherung, Übertragung oder Bereitstellung gekürzt, erweitert oder modifiziert werden. Dies wird in der vorgeschlagenen Architektur ebenfalls durch die TLS/SSL Verbindung abgedeckt. Jeder Sender signiert alle Nachrichten mit seinem persönlichen, einzigartigen, nicht übertragbaren, privaten Schlüssel. Diese Signierung wird nicht für die Nutzdaten durchgeführt, sondern für das Ergebnis einer Hash-Funktion über die Nutzdaten. Der Empfänger kann damit einen Integritätstest durchführen.
Zusammengefasst ist es nun möglich zwischen authentifizierten Teilnehmern über das Netz vertrauliche Daten auszutauschen, die nicht manipuliert werden können. Das bedeutet, dass diese nur von einem bestimmten Empfänger entschlüsselt werden können und dieser verifizieren kann, ob diese von einem bestimmten Sender versendet wurden.

5.3.3. Autorisierung

Die zentrale Komponente hinsichtlich der Autorisierung in der MMS-Architektur ist das Lightweight Directory Access Protocol (LDAP)-basierte [ITU10a] Verzeichnis. In diesem werden die Zugriffsrechte für jedes Gerät mit den dazugehörigen Services und Nutzern konfiguriert. Die besonderen Merkmale dieser Konfiguration sind die Baumstruktur, die Kompa-

tibilität zu Domänennamen und das zugrundeliegende X.500-Zugriffsverzeichnisprotokoll [ITU10a]. In Abbildung 5.4 ist eine Beispielkonfiguration für die vorgeschlagene Architektur zu sehen.

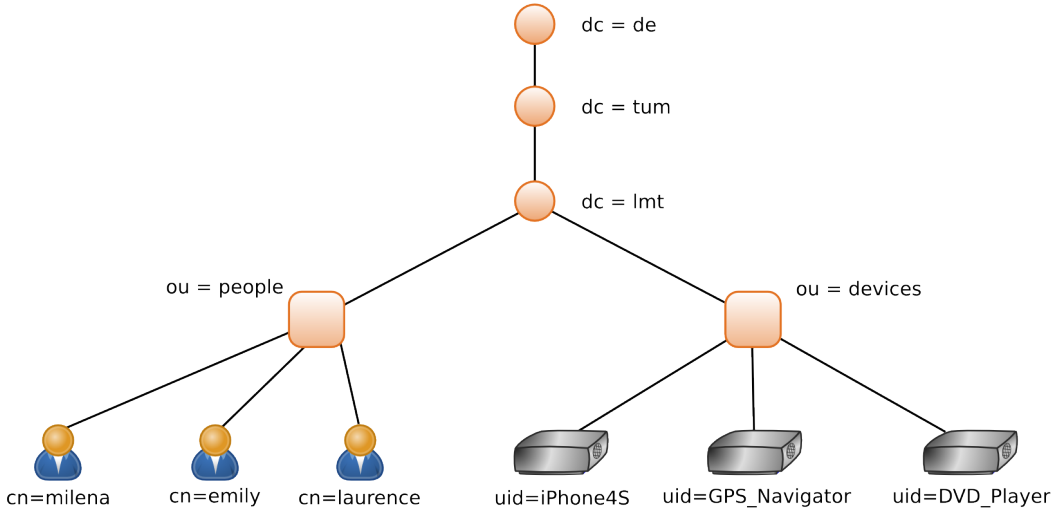

Abbildung 5.4.: *LDAP-Beispielkonfiguration.*

Wie in Abbildung 5.4 zu sehen ist, wird exemplarisch die Domäne *lmt.tum.de* als Ursprung gewählt. Ab dem *lmt* Knoten werden zwei Gruppen definiert, die Nutzer und die Geräte. Die Nutzer stellen die potenziellen Bediener der Geräte im Netz dar. Die Gruppenangehörigen der *people.lmt.tum.de* Domäne sind Bestandteil der nutzerbasierten Authentifizierung (siehe voriges Kapitel). Hierzu wird ihr DN Eintrag, z.B. *milena.people.lmt.tum.de*, als Nutzername und das betreffende Passwort aus deren LDAP-Eintrag verwendet. Die Gruppe der Geräte (*devices.lmt.tum.de*) beinhalten alle im Netz vorhandenen Geräte. Diese werden durch Attribute in dem LDAP-Verzeichnis in verschiedene Sicherheitszonen aufgeteilt. Auf diese Weise kann z.B. ein öffentlicher Bereich für mobile Geräte von sicherheitskritischen internen Funktionen trennen. Abbildung 5.5 zeigt zwei Beispieleinträge eines LDAP-Verzeichnisses.

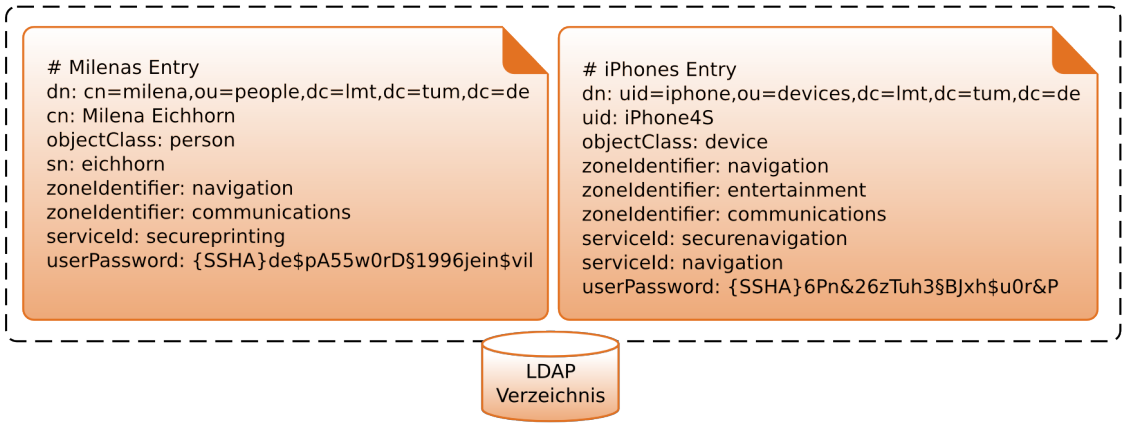

Abbildung 5.5.: *Beispieleinträge der Verzeichnisstruktur des LDAP-Verzeichnisses.*

Der erste Eintrag beinhaltet den DN. Dieser kann sich aus dem Common Name (CN), der

Organisational Unit (OU) und den Domain Components (DC) zusammensetzen. Der DN muss innerhalb des Verzeichnisses eindeutig sein, da er für die gerätebasierte Authentifizierung verwendet wird. Der DN Eintrag des Verzeichnisses wird identisch in das X.509v3-Zertifikat kopiert. Die weiteren Zeilen beschreiben die jeweiligen Attribute mit ihren Werten. Abhängig vom Typ (*objectClass*) des Eintrags sind einige dieser Attribute vordefiniert. Dabei gilt es die beiden Attribute *zoneIdentifier* und *serviceId* hervorzuheben. In der vorgeschlagenen Architektur ist es möglich, durch diese ein Zonenmodell für die Zugriffsrechte einzuführen. So kann mit *zoneIdentifier* festgelegt werden, welchen Zonen der betreffende Eintrag angehört. Und mit dem zweiten Attribut, der *serviceId*, wird festgelegt welche Services auf den betreffenden Eintrag (bzw. Nutzer oder Gerät) zugreifen dürfen. Diese beiden Attribute werden während der Validierung der Zugangsrechte und der servicebasierten Authentifizierung verwendet.

Bezüglich den Richtlinien der Autorisierung spielt die Zugangsliste (engl. Access Control List, ACL) eine wichtige Rolle. Durch die Verwendung des LDAP-Verzeichnisses für die Einträge der Teilnehmer (Nutzer oder Geräte) und den konfigurierten Zugangsrechten (ACL), wird es dem AAA-Server ermöglicht zu entscheiden, ob Nachrichten im Netz, z.B. Serviceaufrufe zwischen Clients und Services, weitergeleitet werden. Dazu werden durch den AAA-Server drei Überprüfungen vorgenommen um die Kommunikation zu erlauben oder zu verbieten. Als erstes wird das Zielgerät der angeforderten Verbindung festgestellt, danach muss geklärt werden, welches die Quelle ist (Nutzer oder Gerät) und abschließend werden die Attribute der Quellnutzer oder -geräte aus dem LDAP-Verzeichnis abgefragt und abhängig von diesen der Zugriff geregelt. Um eine konsistente Zugriffsregelung zu gewährleisten, muss jede ACL diese drei Schritte durchlaufen. In Abbildung 5.6 wird ein Beispieleintrag einer ACL gezeigt.

```
# User's password Access Control List
access to dn.subtree="dc=lmt,dc=tum,dc=de" attrs=userPassword
  by self=wr
  by users=none
  by anonymous auth
  by dn.base="cn=Admin,dc=lmt,dc=tum,dc=de" write
  by * none
                    ⋮
```

Abbildung 5.6.: *Beispieleinträge einer ACL.*

Die erste Zeile eines solchen Eintrags beschreibt welchen Bereich (Attribute) des jeweiligen Teilnehmers diese Zugangsregelung betrifft. Dies entspricht dem Durchlaufen der Schritte eins und drei. Jede Zeile im Anschluss beschreibt jeweils wer zugreifen darf und auf welche Art der Zugriff erfolgt, beispielsweise steht in Abbildung 5.6 *wr* für *write/read*. Diese Regeln entsprechen dem Durchlaufen des zweiten Schrittes.

5.3.4. Abrechnung

Um den AAA-Richtlinien zu entsprechen ist die Abrechnung ein letzter wichtiger Aspekt. Durch diese soll es möglich sein, die Nutzung von sicheren Ressourcen mitzuprotokollieren. Zur Protokollierung wird ein ereignisbasierter Mitschnitt erstellt, der im Systemmanagement, bei der Planung oder in der Abrechnungsanalyse verwendet werden kann. Die für diese Architektur verwendete Abrechnung basiert auf Echtzeit-Zeitstempeln, Beschreibungen dieser und Mitschnitte über die Authentifizierungs- oder Autorisierungsvorgänge. Aus diesem Grund werden sicherheitstechnische Ereignisse wie nutzerbasierte, gerätebasierte, servicebasierte Authentifizierungen, Validierungen von Zertifikaten, Überprüfungen der Zertifikatswiderrufsliste (engl. Certificate Revocation List, CRL), Erstellung von TLS/SSL-Kanälen, Abfragen nach Nutzernamen/Passwörtern, Abfragen von Zugangsrechten und ähnliches als Nachricht einschließlich eines Zeitstempels und einer Nachrichtenbeschreibung im Mitschnitt festgehalten. Abbildung 5.7 zeigt die Syntax eines kurzen Auszugs eines solchen Mitschnitts.

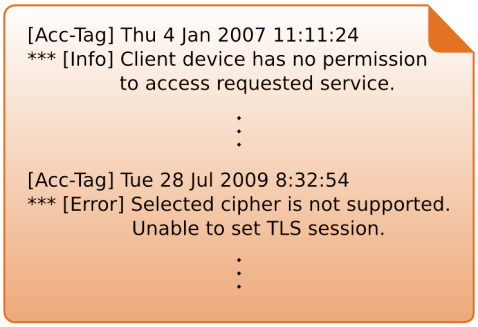

Abbildung 5.7.: *Einträge für beispielhafte Abrechnungsmitschnitte.*

Der Zeitstempel setzt sich aus den Feldern Tag, Monat, Jahr, Stunde, Minute und Sekunde zusammen. Wie zu sehen ist, gibt die Nachrichtenbeschreibung den Typen der Nachricht wieder, z.B. „*Info*“. Im Anschluss folgt eine genaue Erklärung über das aufgetretene Ereignis.
Eine der Hauptanwendungen für die Abrechnungseinträge ist die dadurch entstehende Nichtabstreitbarkeit. Da der Mitschnitt den kompletten Umfang aller Ereignisse mit den jeweiligen Ergebnissen zeigt, kann weder der Client oder der Service noch der AAA-Server die Gültigkeit der ausgeführten Ereignisse anfechten.

Wie die Abläufe innerhalb der Protokolle im Detail für diese AAA-Erweiterungen aussehen, wird nachfolgend an einem allgemeinen Beispiel erläutert.

5.4. Exemplarische Verwendung eines sicheren Services

Im Folgenden wird beispielhaft der Nachrichtenfluss zwischen einem Client, einem Service, dem AAA-Server, der CA und dem Discovery Proxy (DP) gezeigt. Diese Darstellung legt

Wert auf die Sicherheitsanforderungen aus den Bereichen Authentifizierung, Autorisierung und Abrechnung. Dabei ist zu sehen wie die ursprünglichen DPWS-Nachrichten durch die Sicherheitserweiterungen zum Teil ergänzt oder verändert werden.
Die initiale Phase beinhaltet einen Zwei-Wege-Handshake zwischen dem Client und dem DP. Dieser wird von allen Clients durchgeführt um die Endpunktreferenzen des angefragten Services zu erfahren. Nachdem die Adresse des Services dem Client bekannt ist, folgt noch vor jeglichem SOAP-basierten Nachrichtenaustausch die Kommunikation zwischen Client und Service um einen TLS/SSL-Kanal aufzubauen. In Abbildung 5.8 wird dieser erste Teil der Nachrichten gezeigt, der die initiale Kommunikation zwischen Client und DP und dem TLS/SSL-Verbindungsaufbau zwischen Client und Service darstellt. Die Nachrichten der DPWS-Kommunikation werden mit blau dargestellt, wobei die sicherheitsverwandten Nachrichten mit rot markiert werden.

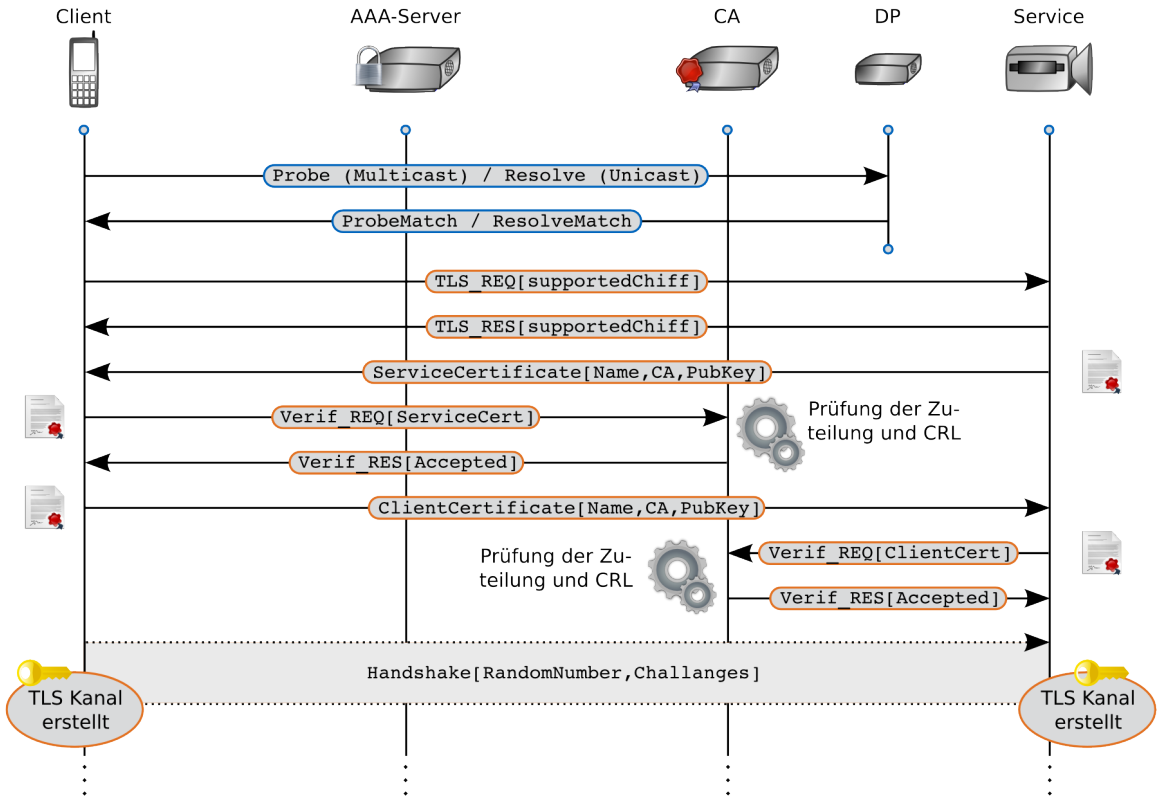

Abbildung 5.8.: *Initiale DPWS-Kommunikation zwischen Client/DP und die gerätebasierte Authentifizierung.*

Die erste Kommunikationsbeziehung besteht aus dem Austausch von Nachrichten des Typs `Probe/Resolve` und `ProbeMatch/ResolveMatch` zwischen Client und DP. Dies ist Bestandteil der üblichen initialen DPWS-Kommunikation. Die Nachrichten dienen zur Auffindung eines bestimmten Services. Der Austausch dieser Nachrichten erfolgt dabei noch ohne Anwendung von Sicherheitsmechanismen. Danach werden die vom Client unterstützten Chiffren für die nachfolgende Kommunikation dem Service mitgeteilt. Der Service kann daraufhin aus diesen eine Chiffre auswählen und sie dem Client mitteilen. Im Anschluss werden die Zertifikate gegenseitig ausgetauscht. Dazu schickt der Service sein eigenes X.509v3-

Zertifikat an den Client. Dieses enthält den Namen des Services, Lokalisierungsinformationen der CA, die Widerrufsliste für Zertifikate (CRL) und zusätzlich den öffentlichen Schlüssel des Services. Wenn das Zertifikat empfangen wurde, lässt der Client dieses durch die CA validieren und prüft ob es eventuell in der CRL eingetragen wurde. Falls beide dieser Überprüfungen erfolgreich sein sollten, schickt der Client im Gegenzug sein eigenes X.509v3-Zertifikat an den Server, um es auf dieselbe Art durch diesen überprüfen zu lassen. Die CA muss eine vertrauenswürdige Instanz darstellen, da beide Teilnehmer von dieser Informationen verifizieren lassen. Die Identität der Geräte ist in den Zertifikaten enthalten und diese können abhängig von dem festgelegten Vertrauensniveau von der CA angenommen oder zurückgewiesen werden können. Aus diesem Grund ist die Überprüfung durch die CA und CRL als gerätebasierte Authentifizierung bekannt. Ab diesem Zeitpunkt haben der Client und der Service die zu verwendende Chiffre festgelegt und ihre öffentlichen Schlüssel ausgetauscht. Mit diesen Informationen lässt sich eine letzte Handshake-Phase einleiten. In dieser wird durch eine Reihe von gegenseitigen Challenge/Response Verfahren geprüft, ob das Gegenüber vertrauenswürdig ist. Diese Verfahren werden durch die Verwendung der öffentlichen/privaten Schlüssel gesichert durchgeführt. Nach der erfolgreichen Durchführung dieses Handshakes ist der TLS/SSL-Kanal zwischen dem Client und dem Service aufgebaut und kann genutzt werden. Alle nachfolgenden Nachrichten, die über einen sicheren Kanal verschickt werden sollen, verwenden diesen. Im Folgenden werden diese in den Diagrammen mit roten gestrichelten Pfeilen dargestellt. Falls unverschlüsselter Verkehr auftritt, wird dieser weiterhin mit schwarzen durchgezogenen Pfeilen dargestellt.

In Abbildung 5.9 wird die nächste Phase des Nachrichtenablaufs dargestellt. Zunächst wird ein zweiter TLS/SSL-Kanal zwischen dem Service und dem AAA-Server aufgebaut. Diese Verbindung dient später der Überprüfung der Zugangsberechtigungen. Durch diesen werden alle Anfragen und Antworten zwischen den Endpunkten geschützt. Diese sichere Verbindung wird durch die gleichen Schritte wie zuvor zwischen dem Client und dem Service, d.h. Festlegung der Chiffre (`TLS_REQ()/TLS_RES()`), Validierung des AAA-Serverzertifikats (`ServerCert()`), Überprüfung des Servicezertifikats (`ServiceCert()`, dieses Mal durch den AAA-Server) und den abschließenden Handshake aufgebaut. Ähnlich wie zuvor wird bei einem positiven Ergebnis all dieser Operationen ein sicherer Kanal etabliert. Dadurch kann die nutzerbasierte Authentifizierung in einer sicheren Umgebung durchgeführt werden. Hierbei handelt es sich um die Verwendung von Nutzername/Passwort als Legitimierung, die von dem Client an den Service geschickt werden um Zugang zu diesem gewährt zu bekommen (`userauth_REQ()`). Nach dem Empfang dieser Informationen überträgt der Service diese über die TLS/SSL-Verbindung in Form einer LDAP-Anfrage an den AAA-Server (`LDAP_QRY()`). Dadurch wird die Existenz eines solchen Nutzers überprüft und die Zugangsrechte von diesem in der ACL erfragt. Nur wenn beide Überprüfungen ein positives Ergebnis zurückliefern schickt der AAA-Server eine *Accepted* Nachricht an den Service. Der Service leitet diese weiter an den Client und akzeptiert somit diesen Nutzer.

Die dritte und letzte Phase eines typischen Nachrichtenablaufs (Abbildung 5.10) zeigt die Nutzung von Services in einer sicheren Umgebung.

Zunächst werden die üblichen DPWS-Nachrichten `GetMetadata` und `GetMetadata Response` ausgetauscht. Der Austausch erfolgt nun über eine sichere Verbindung, da die gerätebasierte und nutzerbasierte Authentifizierung erfolgreich durchgeführt wurde. Aus

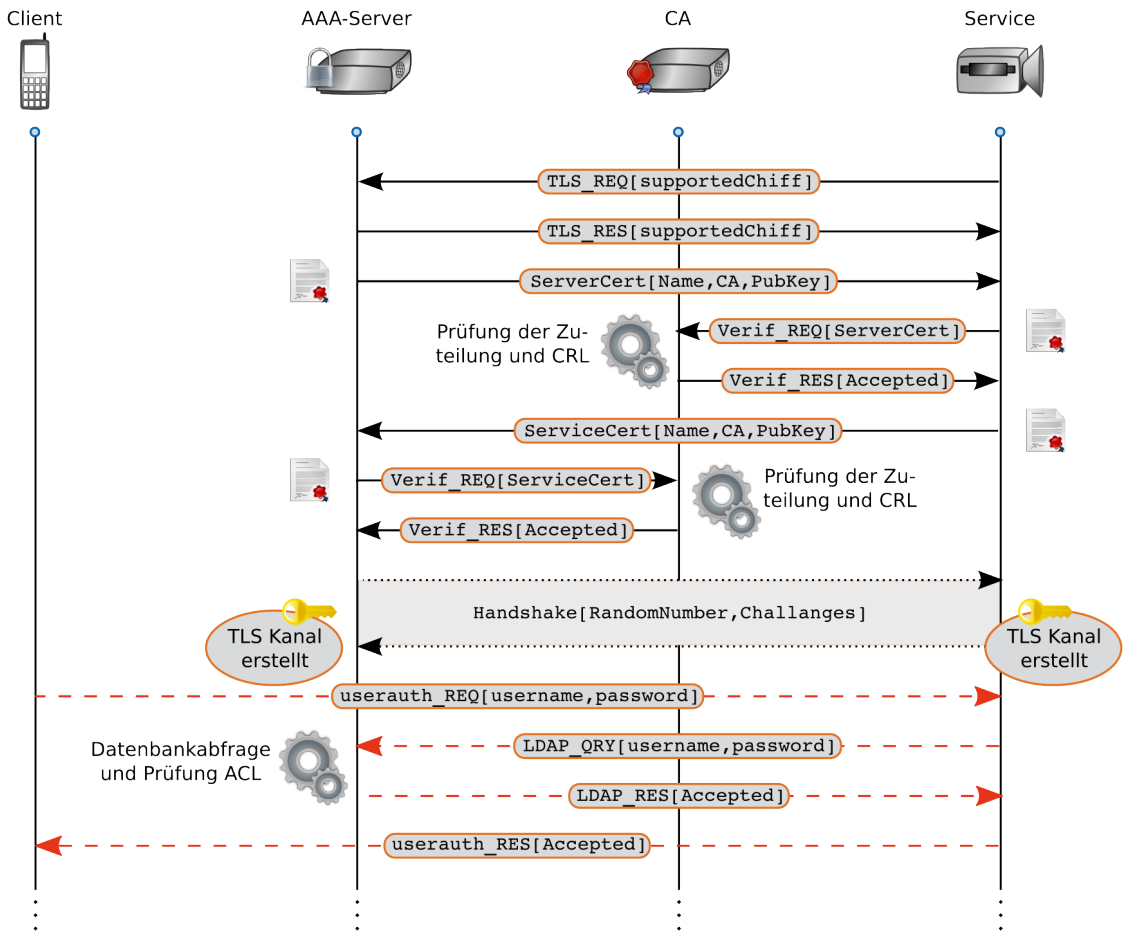

Abbildung 5.9.: *TLS/SSL Aufbau zwischen Service/AAA-Server und nutzerbasierte Authentifizierung durch eine LDAP-Anfrage.*

diesem Grund ist es dem Client gestattet, die Beschreibung des DPWS-Gerätes und dessen verfügbaren Services anzufordern und zu erhalten. Im Anschluss daran kann der Client nun durch SOAP-Nachrichten bestimmte Serviceaufrufe tätigen. Dafür erstellt der Service nach dem Eingang eines Serviceaufrufs eine LDAP-Anfrage und fragt mit dieser bei dem AAA-Server an. So erfährt der Service, ob der anfragende Client überhaupt die Rechte besitzt um auf den Service zuzugreifen. Der Vorgang wird als servicebasierte Authentifizierung bezeichnet. Falls die Authentifizierung negativ verläuft, wird dieses Ergebnis durch den Service dem Client mitgeteilt. Falls sie positiv verläuft, wird der Serviceaufruf in der SOAP-Nachricht verarbeitet oder beantwortet, je nachdem wie der angefragte Service ausgelegt ist. Die servicebasierte Authentifizierung erlaubt für einen bestimmten Client in der MMS-Architektur den Servicezugriff oder die Servicenutzung einzuschränken.
Ergänzend muss erwähnt werden, dass in Abbildung 5.10 der Serviceaufruf für das Einschalten („*ON*“) nicht verschlüsselt ist. Das bedeutet, dass der Service so konfiguriert ist, dass für diesen Aufruf die Kommunikation über einen unsicheren HTTP-Port abgewickelt werden kann. Dies dient als Beispiel für performante Services, deren Daten nicht sicherheitsrelevant sind, aber dafür diese schneller übertragen wollen (siehe Kapitel 6.3.3). Das Gerät kann damit auf die Verschlüsselung verzichten, muss sich aber dennoch sicher integrieren.

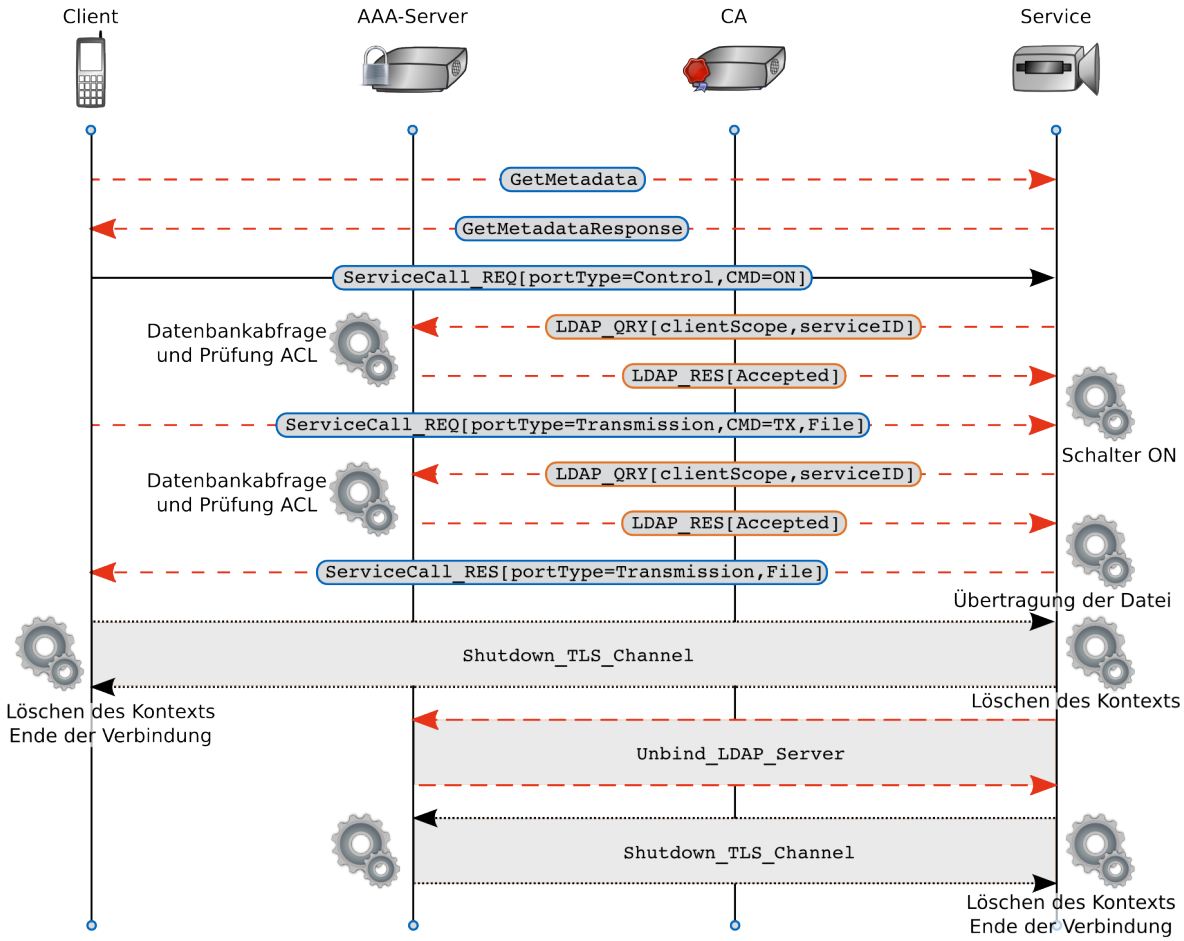

Abbildung 5.10.: *Servicebasierte Authentifizierung und (un-)verschlüsselte Serviceaufrufe.*

Dagegen nutzt der Serviceaufruf „TX“ für die Anfrage nach der Übertragung von Daten und deren Antwort den sicheren Kanal. Dieses unterschiedliche Verhalten hängt von der Implementierung des Services ab. Dieser Vorteil ermöglicht jedem Entwickler selbst zu entscheiden, ob er Performanz oder Sicherheit bevorzugt. Trotz allem werden für beide Fälle jeweils die gerätebasierte, nutzerbasierte und die servicebasierte Authentifizierung durchgeführt.

5.5. Zusammenfassung

In diesem Kapitel werden die Sicherheitsmechanismen präsentiert, die sich an den AAA-Richtlinien orientieren. Die erarbeiteten Konzepte zur Absicherung der vorgeschlagenen Architektur werden, aufgeteilt in Authentifizierung, Autorisierung und Abrechnung, vorgestellt und exemplarische Nachrichtenabläufe gezeigt.
Zusammenfassend kann gesagt werden, dass die Erweiterungen durch eine Gruppe von Techniken und Protokollen Einfluss auf verschiedene Schichten der Kommunikationsprotokolle in Rechnernetzen nehmen. Dementsprechend zeigt Abbildung 5.11 einen Überblick

der Protokolle für die vorgeschlagene Lösung kombiniert mit den wichtigsten Netzteilnehmern und ihren betreffenden Modulen.

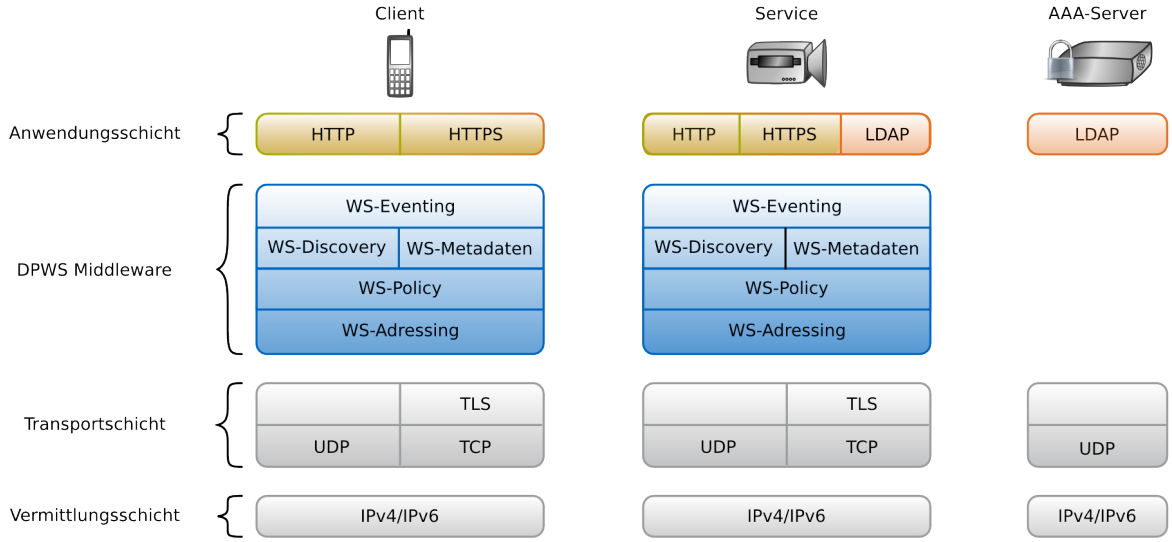

Abbildung 5.11.: *Protokollstack der vorgeschlagenen Architektur mit Sicherheitserweiterungen.*

Wie zu sehen ist, haben die Sicherheitserweiterungen Einfluss auf die Anwendungs- und Transportschicht. Die durch das OASIS Konsortium vorgeschlagenen Empfehlungen beschränken sich auf die Authentifizierung, Vertraulichkeit und Integrität. Diese Einschränkung wird durch die Anwendung der AAA-Richtlinien aufgehoben. Durch diese werden die Sicherheitsmechanismen Privatsphäre, Zugangskontrolle und Nichtabstreitbarkeit in der Transportschicht ergänzt. Die Verwendung von Mechanismen wie TLS/SSL in dieser Schicht hat sich als großer Vorteil hervorgetan um das Sicherheitsniveau der MMS-Architektur zu steigern. Dank der verzeichnisbasierten Zugangsregeln wird es ermöglicht ein Zonenmodell einzuführen. Damit ergibt sich die Möglichkeit ein dediziertes und fein gegliedertes Zugangsmanagement anzubieten. Das Zugangsmanagement bezieht dabei nicht nur teilnehmende Geräte und Services mit ein, sondern auch Nutzer. Ein weiterer Aspekt dieser Lösung ist die Flexibilität. Es ist möglich alle neuen Geräte zu unterstützen wie auch auf die unterschiedlichen Hardwareanforderungen oder Nutzdaten einzugehen. Auf diese Weise können z.B. Nutzdaten wegen der Performanz oder fehlenden Hardwareressourcen unverschlüsselt übertragen werden.
Die vorgeschlagene Sicherheitslösung ist nun konzeptionell erstellt, im nachfolgenden Kapitel wird nun diese im Zusammenspiel mit den anderen Komponenten sowohl qualitativ als auch quantitativ evaluiert. Damit soll eine Aussage über die Qualität der Lösung getroffen werden.

6. Realisierung und Performanzbetrachtungen

Die Konzepte aus Kapitel 4 sind größtenteils an Webtechnologien und serviceorientierte Architekturen angelehnt. Daher bietet es sich an, bereits realisierte Techniken aus diesen Bereichen für die Realisierung des MMS-Systems zu verwenden. Die Anwendung und die Evaluation dieser Techniken werden im folgenden Kapitel erläutert.

6.1. Grafische Benutzerschnittstelle

Die grafische Benutzerschnittstelle setzt sich aus zwei großen Komponenten zusammen. Zum einen besteht sie aus dem GUI-Renderer für den Abruf der GUI, deren Darstellung und der Interaktion dieser mit dem Nutzer. Die zweite Komponente der grafischen Schnittstelle ist der GUI-Kombinierer für das Zusammensetzen der GUI, Interaktion mit den Funktionalitäten und Bereitstellung der GUI an den GUI-Renderer.

6.1.1. GUI-Renderer Aufbau

Es gibt verschiedene Arten den GUI-Renderer zu implementieren. Der entscheidende Punkt ist die Verwendung von Webtechnologien, wie z.B. HTML, da sich dies vom Konzept her als Realisierung anbietet. Üblicherweise wird hierbei auf eine HTML/JavaScript Engine zurückgegriffen. Die gesamte Intelligenz, die für die dynamische Generierung der GUI im Renderer zuständig ist, wird über diese Techniken realisiert. So kann hierfür ein Standardbrowser wie z.B. Mozilla Firefox genauso verwendet werden wie eine eigene Entwicklung die auf einer HTML Engine wie z.B. Webkit [The10e] basiert. Die Engine kann sehr komfortabel durch die C++-Bibliothek Qt [Nok10b] eingebunden werden. Auf diese Weise lassen sich mit wenigen Codezeilen bereits Webseiten darstellen. Damit durchgeführte Eigenentwicklungen besitzen den Vorteil der vollkommenen Freiheit bei der Entwicklung. So ist man nicht auf den Umfang der potentiell ungenutzten Funktionen in einem Standardbrowser angewiesen, durch den diese Webbrowser eventuell auch langsamer in der Verarbeitung werden. Zu dieser Freiheit gehört auch der Aspekt *Individualisierbarkeit* des eigenen Webbrowsers. Ein Fahrzeughersteller kann beispielsweise sein Corporate Design diesem Webbrowser mitgeben. Auch die nötige *Verwendung von Web Services* in dem MMS-System kann leicht realisiert werden, da keine Nachimplementierungen in JavaScript oder als eigenes Plugin nötig sind. Es ist dabei möglich die DPWS-Kontrollmechanismen direkt in den Webbrowser zu integrieren. Die *Abstimmung mit dem GUI-Inhalt* ist ein weiterer Aspekt. So kann ein eigener Webbrowser eventuell Informationen aus dem Inhalt herausnehmen und für die Konfiguration weiterer Komponenten nutzen. Davon kann beispielsweise die *Multimediaunterstützung* profitieren. Ähnlich wie bei HTML5 wird ein Markierungstag in dem HTML-Code gesetzt

an dem das Multimediaelement platziert werden soll. Dabei kann es sich z.B. um die Position eines Videos handeln. Leider ist man bei einem Standardbrowser auf die eingebauten meist standardkonformen Mittel angewiesen. Bei einem eigenen Webbrowser kann die Positionierung des Videos ebenfalls über den HTML-Code geschehen und der Webbrowser kann zusätzlich um die Ansteuerung mit Web Services erweitert werden. Da meist große Gemeinschaften oder Firmen hinter den Standardbrowsern stehen, ist die *Wartung* bei Eigenentwicklungen ein eher aufwändiges Unterfangen. Die *Abdeckung* spricht sicherlich ebenfalls für die Verwendung von Standardbrowsern. Im Vergleich zu einem eigenen Webbrowser sind sie eindeutig weiter verbreitet. Hingegen werden die Standardbrowser meist nur von etablierten Plattformen unterstützt. Hier kann wiederum eine Eigenentwicklung als Lösung verwendet werden. Betrachtet man die *Sicherheit* muss klar gesagt werden, dass man bei dem Standardbrowser auf die integrierten Mechanismen angewiesen ist. Hingegen ist man bei einer Eigenentwicklung wesentlich freier (siehe Kapitel 5.2). Des Weiteren lässt sich ein eigenentwickelter Webbrowser bezüglich der *Performanz* optimieren, da viele Aspekte, wie z.B. das Surfen mit mehreren Karteireitern, nicht unterstützt werden müssen. In Tabelle 6.1 ist eine Zusammenfassung der genannten Aspekte zu sehen. Für die Bewertung wird die Skala aus Tabelle 2.1 verwendet.

Aspekt	Eigenentwicklung	Standardbrowser
Abdeckung	−	++
Performanz (bezogen auf GUI-Darstellung)	++	−
Sicherheit	++	+
Individualisierbarkeit	++	−
Multimediaunterstützung	++	−
Abstimmung mit Inhalt	++	−
Integrierbarkeit von Web Services	++	−
Wartung	+	++

Tabelle 6.1.: *Vergleich zwischen eigenentwickeltem Webbrowser und Standardbrowser.*

Es geht deutlich hervor, dass eine Eigenentwicklung des Webbrowsers für das MMS-System viele Vorteile besitzt. Nichtsdestotrotz werden für das MMS-System beide Varianten vorgeschlagen. Es bietet sich an, den eigenentwickelten Webbrowser auf speziell dafür vorgesehener Hardware, wie z.B. intelligenten Fahrzeugdisplays, zu verwenden und auf mobilen Geräten den meist bereits vorhandenen Standardbrowser. Auf diese Weise kann die korrekte Funktionsweise auf eigenen Komponenten des MMS-Systems gewährleistet werden und dennoch ein eventuell eingeschränkter Umfang an Funktionalitäten auch auf fremden Standardkomponenten angeboten werden. Es kann über eine Einschränkung der Zugriffsrechte für nicht eigenentwickelte Webbrowser nachgedacht werden, falls Bedenken hinsichtlich der Sicherheit aufkommen. Um die erweiterten Zugriffsrechte auch auf mobilen Geräten anbieten zu können, kann der eigene Webbrowser als Erweiterung für die mobilen Geräte angeboten werden.

Das Grundgerüst der GUI stellt eine HTML-Struktur dar. Dieses wird bei der Initiierung mit den konzeptionellen Elementen des GUI-Renderers übertragen. Diese Elemente stellen die logischen und funktionalen Abläufe dar. Die dynamischen Vorgänge in den Abläufen werden mit JavaScript nachgebildet. Das Aussehen dieser Struktur und Logik wird mit Hil-

fe von Cascading Style Sheets (CSS) festgelegt. Aufgrund dieser Trennung zwischen Inhalt und Design wird es möglich, Änderungen an diesen Teilen unabhängig voneinander durchzuführen. Die verwendeten Webtechnologien sind durchaus für die Darstellung der vorgeschlagenen GUI geeignet. Dies wird in den Performanzbetrachtungen evaluiert und auch in Beispielen [Goo10a] gezeigt. Es wird dort diskutiert, welche eindrucksvollen Effekte mit JavaScript und CSS möglich sind. Sollen eventuell auch wesentlich komplexere Effekte, wie z.B. 3D-Animationen, realisiert werden, ist dies ebenfalls mit JavaScript möglich. In [Khr10] wird die JavaScript-basierte Schnittstelle WebGL für OpenGL präsentiert, damit wird es möglich ohne weiteren Aufwand OpenGL in einem Webbrowser zu verwenden.

Die Nutzdatendarstellung wird nur in dem eigenentwickelten Webbrowser realisiert. So ergibt sich der maximale Komfort für die Ansteuerung und Darstellung. Neben der HTML-Engine wird in Qt auch die SOA-API und ein eigenes Real-Time Transport Protocol (RTP) integriert. Somit werden beispielsweise Videodaten außerhalb des DPWS-Protokolls übertragen. Nach dem Empfang dieser werden die Videodaten mit Hilfe der Codec-Sammlung FFmpeg [FFm10] dekodiert und anschließend dargestellt. Dabei werden die Positionierungen und das Aussehen der Darstellung über den HTML-Code der GUI festgelegt. Die Ansteuerung der Darstellung, z.B. das Stoppen der Videowiedergabe, wird über die Kontrollmechanismen mit Serviceaufrufen von DPWS bewerkstelligt.

Ein weiterer wichtiger Punkt neben der anwendungsbasierten Realisierung des GUI-Renderers ist das darunterliegende Betriebssystem. In dem automotiven Bereich ist es beispielsweise unerlässlich, dass die MMS zur Verfügung steht sobald der Fahrer sich in das Fahrzeug setzt und die Zündung einschaltet. In Serienfahrzeugen wird zum Teil bereits detektiert, wann der Fahrer mit seinem Schlüssel das Fahrzeug öffnet und in diesem Moment wird der Hochfahrprozess des Betriebssystems mit den MMS-Funktionen gestartet. Lediglich der Bildschirm des Displays bleibt noch ausgeschaltet bis zum Starten der Zündung. Dies wirkt auf den Fahrer wie ein instantanes Starten des Systems. Bedenkt man dieses Vorgehen bleiben etwa 10-15 Sekunden für das Hochfahren des Systems. Um diesen Ansprüchen gerecht zu werden, wird ein Minimalsystem auf Basis von Microcore [Shi10] als Basis für den Webbrowser realisiert. In Kapitel 6.3.2 werden Messungen zu diesem System vorgestellt. Das Minimalsystem wird dabei auf das Nötigste reduziert. Die Reduzierung beinhaltet den Linux-Kernel mit Basisprogrammen und Mechanismen für die grafische Ausgabe um direkt in den Framebuffer zu schreiben.

6.1.2. GUI-Kombinierer Aufbau

Da der GUI-Kombinierer mehrere Nutzer mit HTML-basierten GUI-Inhalten versorgen soll, werden Webserver als Realisierungsmöglichkeit betrachtet. Einer der meistverbreiteten Webserver ist Apache [The10a]. Dieser bringt einen großen Funktionsumfang mit sich und verbraucht aus diesem Grund sehr viele Ressourcen. Es gibt auch weniger ressourcenintensive Varianten von Webservern wie z.B. lighttpd [The10b]. Hingegen ist die fehlende Möglichkeit der Erweiterungen durch eigene Programmteile ein Nachteil vieler Webserver. So ist es zwar beispielsweise möglich über Common Gateway Interface (CGI) vorkompilierte Programme auszuführen, aber diese werden in einer eigens dafür gestarteten Programmverzweigung ausgeführt und können nicht mit weiteren Prozessen interagieren oder dynamisch auf Er-

eignisse reagieren. Bei der Recherche nach Lösungen dafür hat sich tntnet [The10d] als eine brauchbare Lösung herausgestellt. Dabei handelt es sich um einen in C++ geschriebenen Webserver, der es durch eine eigene Beschreibungssprache ermöglicht, selbstgeschriebenen C++-Code als Webanwendung anzubieten. Somit ist es möglich, die SOA-API in den Webserver zu integrieren und gleichzeitig den HTML-basierten GUI-Inhalt Webbrowsern anzubieten. Neben der Integration von C++-Code bringt dieser Webserver auch einen Geschwindigkeitsvorteil mit sich.

Durch die Verwendung von browserbasierten Techniken ergibt sich die Herausforderung Daten asynchron an den Webbrowser zu schicken. Dieses Vorgehen wäre normalerweise ein großes Sicherheitsproblem. Es wäre z.B. denkbar, einem Nutzer beim Surfen ungewollt andere Webseiten anzeigen zu lassen. Da in der vorgeschlagenen MMS-Architektur aber durchaus die Notwendigkeit besteht, z.B. Benachrichtigungen über eine Controllerbetätigung zu übermitteln, wird hierfür die Technik Comet [CM08] verwendet. Diese Technik ist auch als Ajax Push, Reverse Ajax oder HTTP Streaming bekannt. Dabei wird über eine offen gehaltene HTTP-Anfrage zwischen Webbrowser und Webserver erlaubt, Daten vom Webserver aktiv nach initialer Anmeldung an den Webbrowser zu schicken, ohne dass dieser die Daten im weiteren Verlauf anfordert. In Abbildung 6.1 wird der Unterschied zwischen Ajax und Comet gezeigt.

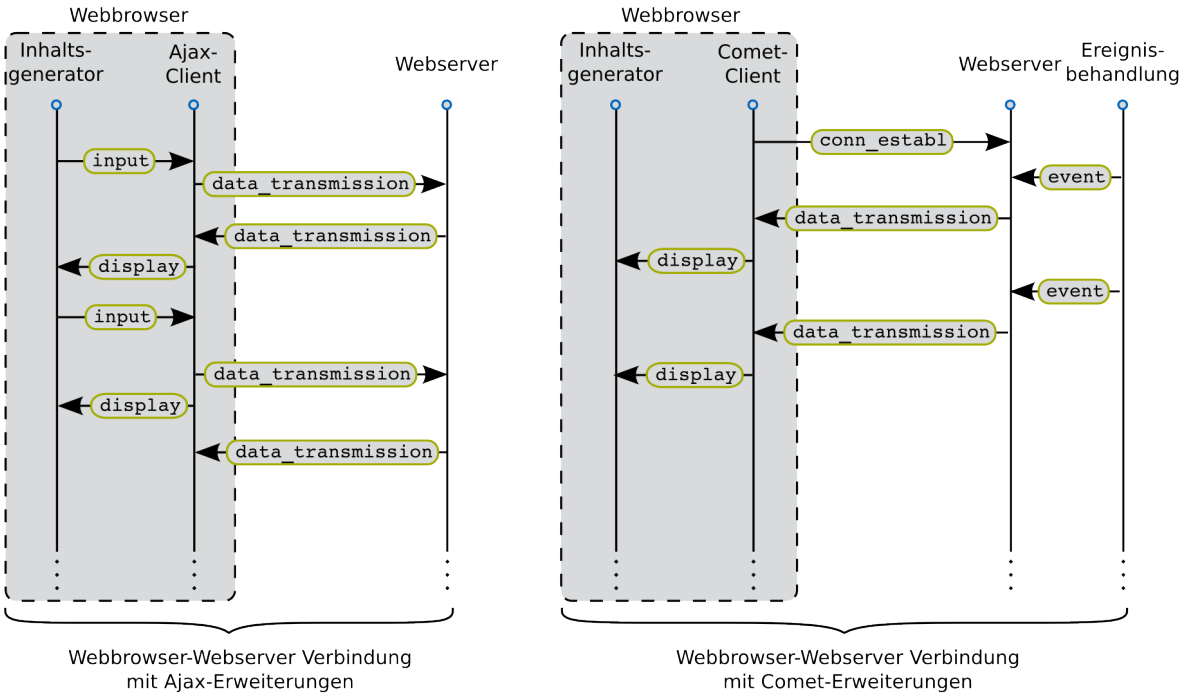

Abbildung 6.1.: *Vergleich Ajax-Technik mit Comet-Technik.*

Links in Abbildung 6.1 ist eine gewöhnliche Verbindung zwischen Webbrowser und Webserver mit Ajax Erweiterungen zu sehen. Der Inhaltsgenerator und Ajax-Client sind im Webbrowser vereint. Aus den Nachrichtenabläufen geht deutlich hervor, dass es sich um eine asynchrone Verbindung handelt. Es werden durch den Ajax-Client Inhalte vorgehalten und bei Bedarf zur Darstellung übergeben. Der Ajax-Client bezieht dazu im Hintergrund die Da-

ten von dem Webserver.
In Abbildung 6.1 wird auf der rechten Seite eine Verbindung mit Comet-Elementen gezeigt. Die Elemente Inhaltsgenerator und Comet-Client sind wiederum im Webbrowser vereint. Des Weiteren gibt es eine Ereignisbehandlung im Webserver oder außerhalb des Webservers, die aber auf diesen zugreift. Durch die Comet-Technik wird es ermöglicht, die Ereignisse, die durch die Ereignisbehandlung auftreten, dem Webserver mitzuteilen. Da der Comet-Client im Voraus eine Verbindung zwischen Webserver und Webbrowser geöffnet hat, können die Ereignisse nun über diese Verbindung ohne Anfragen direkt an den Webbrowser übertragen werden. Auf diese Weise wird die Übertragung effizienter gestaltet und Ereignisse werden zeitnäher übertragen.

6.1.3. Serviceerweiterungen im GUI-Gerät

Ein wichtiger Beitrag zur grafischen Benutzerschnittstelle in einem GUI-Gerät liefert die Information für die grafische Repräsentation. Die durch DPWS übertragenen Daten beinhalten Beschreibungen der Menüstruktur eines GUI-Gerätes. In Abbildung 6.2 wird diese Menüstruktur in gespeicherter Darstellung und in logischer Darstellung gezeigt.

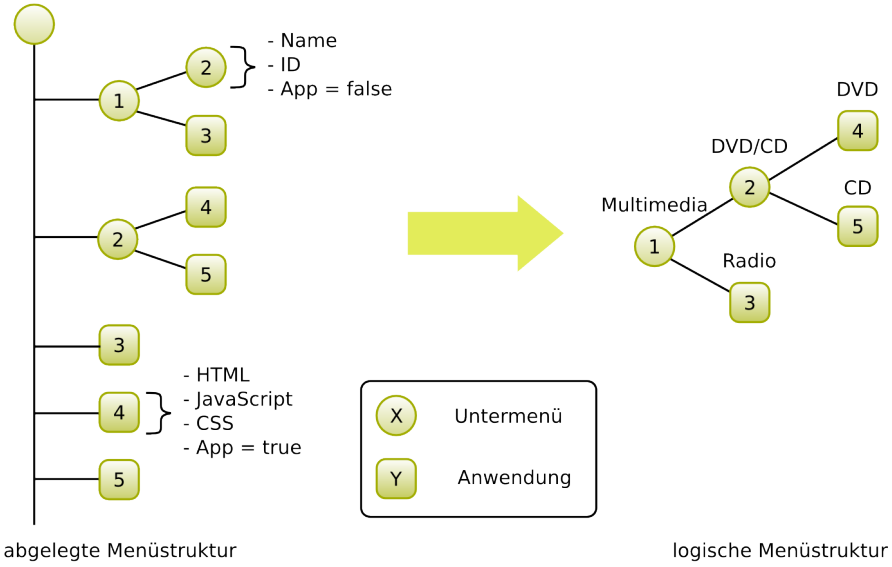

Abbildung 6.2.: *Gegenüberstellung der im Speicher abgelegten Menüstruktur gegen die logische Darstellung.*

Die abgelegte Menüstruktur zeigt das Menü wie es vorgehalten wird. Von der ersten Hierarchieebene aus verzweigen sich nur zwei Arten von Knoten. Dabei handelt es sich entweder um Untermenüknoten oder um Anwendungsknoten. Ein Untermenüknoten besitzt Verknüpfungen zu weiteren Untermenüknoten oder Anwendungsknoten. Auf diese Weise lassen sich alle Menüeinträge beliebig kombinieren. Wird beispielsweise die ID 1 abgefragt, werden die beiden Unterknoten 2 und 3 mit deren Datenfeldern zurückgegeben. Ein Untermenü besitzt die Datenfelder Name, ID und App. Der Name bestimmt dabei den später im Menü dargestellten Anzeigenamen, die ID wird verwendet um damit eventuell dessen

Unterknoten abzufragen und über das App-Feld kann zwischen Untermenü und Anwendung unterschieden werden. Wird aus dem zuvor angeforderten Unterknoten der ID 1 nun die ID 3 ausgewählt, stellt dies eine Verknüpfung zu einer Anwendung dar. In diesem Fall werden nun über das Datenfeld „HTML“ die HTML-Struktur, über „JavaScript“ die Logik und über das Datenfeld „CSS“ das Aussehen der Anwendung übertragen. Um zu erkennen, dass es sich dabei um eine Anwendung handelt, wird das Datenfeld App in diesem Fall auf *true* gesetzt. Mit diesen Mechanismen lässt sich die in Abbildung 6.2 rechts gezeigte logische Menüstruktur realisieren. Zur Verdeutlichung wurden hier exemplarisch Namen von möglichen Menüpunkten ergänzt.

6.2. Servicemanagement

Die Realisierung des Servicemanagements stützt sich auf die Verwendung von DPWS Core [SOA10]. Es handelt sich dabei um eine in der Programmiersprache C geschriebene Implementierung der DPWS-Spezifikation. Dieser in eigene Programme einbettbarer DPWS-Protokollstack wird um eine C++-Anbindung erweitert. Auf diese Weise wird eine problemlose Integration in die bereits auf C++ basierenden Komponenten ermöglicht. Neben dieser Anbindung wird für die schnelle Entwicklung und leichtere Wartbarkeit eine Compiler-Umgebung auf Basis von SCons [The10c] realisiert.
Der DPWS Core wird nahezu unverändert verwendet. Es wird in den DPWS-Geräten eine Beschränkung ergänzt, die verhindert, dass ein DPWS-Gerät auf Suchanfragen über Multicast reagiert, wenn ein Discovery Proxy im DPWS-Netz vorhanden ist (siehe Kapitel 4.3.2). Diese Beschränkung hat keinerlei Auswirkung DPWS-Geräte mit einem nichtmodifizierten DPWS-Protokoll. Diese antworten weiterhin auf Multicast-Suchanfragen, trotz der Anwesenheit eines Discovery Proxys.

6.2.1. Initialisierungsprozess eines Gerätes mit einem Service

Im Nachfolgenden werden die Ergänzungen der Sicherheitserweiterungen in einem Gerät mit einem Service erläutert. Dazu wird dessen Initialisierungsablauf gezeigt und farbig hervorgehoben an welchen Stellen die Erweiterungen zum Einsatz kommen.

1. **Initialisierung des DPWS-Stacks:** Hier werden Klassen, Strukturen und Routinen des DPWS Core initialisiert, um die verschiedenen Ressourcen in der Bitübertragungs-, Sicherungs-, Vermittlungs-, Transport- und Anwendungsschicht zu reservieren.

2. **Parametrisierung des DPWS-Stack:** Setzen der Parameter für den DPWS-Stack.

3. **Erstellen des Serviceobjekts:** Die optionalen nichtsicheren Serviceparameter werden eingerichtet.

4. **Erstellung des sicheren Serviceobjekts:** Die sicheren Serviceparameter werden eingerichtet.

5. **Einrichtung von TLS:** Die Initialisierung von OpenSSL und Erstellung der TLS-Umgebung findet statt. Der private Schlüssel des Gerätes und das Zertifikat werden geladen. Informationen zur Auffindung der CA und CRL werden bestimmt und zusätzlich werden die unterstützten Chiffren festgelegt.

6. **Erstellen und konfigurieren des Gerätes:** Zuerst werden die Geräteeigenschaften gesetzt und danach die Struktur des Gerätes für das Anbieten eines Services vorbereitet.

7. **Erstellen und konfigurieren des Services:** Die Endpunktreferenzen werden gesetzt und der Service konfiguriert.

8. **Erstellen und konfigurieren des sicheren Services:** Auch für die sicheren Services werden die Endpunktreferenzen gesetzt und die Services dementsprechend konfiguriert.

9. **Aktivieren des Gerätes:** Das Gerät ist fertig eingerichtet und bereit im Netz zu agieren.

Im Vergleich zu einem regulären Initialisierungsablauf wird dieser um die Einrichtung des TLS, Erstellung eines sicheren Serviceobjekts und Konfiguration eines sicheren Services ergänzt.

6.2.2. Client Aufbau

In der anschließenden Auflistung wird der Ablauf einer Initialisierung eines Gerätes mit Client gezeigt. Die Sicherheitsergänzungen in diesem Fall werden wiederum farbig hervorgehoben.

1. **Initialisierung des DPWS-Stacks:** Hier werden Klassen, Strukturen und Routinen des DPWS Core initialisiert um die verschiedenen Ressourcen in der Vermittlungs-, Transport- und Anwendungsschicht zu reservieren.

2. **Initialisierung der DPWS-Struktur für Client-Operationen:** Einrichtung von zusätzlichen Programmstrukturen für DPWS-interne Vorgänge, die vom Client abhängig sind.

3. **Suche nach Geräten:** Durch WS-Discovery werden die nötigen Geräte gesucht und die Suchergebnisse lokal abgespeichert.

4. **Einrichtung von TLS:** Es wird die OpenSSL Bibliothek eingerichtet, der TLS-Kontext erstellt und der private Schlüssel sowie das Zertifikat des Gerätes geladen. Die Speicherungsstelle der CA und CRL wird bestimmt und die unterstützten Chiffren werden festgelegt.

5. **Etablierung der DPWS-Sicherheit:** Die Routinen für die nutzerbasierte und servicebasierte Authentifizierung werden in die DPWS-Sicherheitsstruktur und -klassen geladen.

6. **Empfang und Verarbeitung von Metainformationen des Services:** Mit Hilfe von WS-Transfer und WS-MetadataExchange werden die Metainformationen des Services empfangen und verarbeitet.

7. **Abfrage des Serviceendpunkts:** Der Service wird nach den jeweiligen Serviceendpunkten gefiltert.

Verglichen mit der ursprünglichen Initialisierung wird die Einrichtung von TLS und der DPWS-Sicherheitsumgebung ergänzt.

6.2.3. AAA Server und CA Aufbau

Der AAA Server und die CA stellen die Basis für die Sicherheitserweiterungen der MMS-Architektur dar. In Tabelle 6.2 werden die Komponenten des AAA-Servers mit den dazugehörigen Realisierungen gezeigt.

Komponente	Realisierung
Teilnehmerverzeichnis	OpenLDAP
Datenbank	Berkeley Database
Sicherheitsfunktionen	OpenSSL

Tabelle 6.2.: *Übersicht der realisierten Komponenten eines AAA-Servers.*

Für die Realisierung der Struktur des Teilnehmerverzeichnisses wird OpenLDAP verwendet. Die Einträge des Verzeichnisses werden in der Berkley Database gespeichert. Alle weiteren Sicherheitsfunktionen werden von der Bibliothek OpenSSL zur Verfügung gestellt. An den verwendeten Realisierungen werden keine Änderungen für die Anwendung in der MMS-Architektur nötig. Die CA wird ebenfalls über OpenSSL nachgebildet, dabei werden manuell Zertifikate für die Geräte erstellt und verteilt.

6.2.4. Aufbau des Discovery Proxys

In DPWS Core sind bereits Mechanismen für die Erstellung eines Discovery Proxys vorhanden. Diese werden für die Realisierung verwendet. Für das vorgeschlagene MMS-System ist nur wichtig, dass in den DP der WEBHMIBase-Client integriert wird. Dies ist essentiell für das Funktionieren der MMS-Architektur. Diese Ergänzung stellt keine Änderung an DPWS Core dar, sondern wird transparent als Service gelöst.

6.3. Quantitative Evaluierung

In diesem Abschnitt wird eine quantitative Evaluierung der beiden Teilgebiete „Grafische Benutzerschnittstelle“ und „Servicemanagement“ durchgeführt. Dabei werden relevante Aspekte wie Verzögerungen oder Auslastungen von Ressourcen im System genauer betrachtet.

6.3.1. Szenariobeschreibung

Nachfolgend werden die Szenarien, die für die Evaluierung herangezogen werden, erläutert. Die Beschreibung wird in die grafische Schnittstelle und das Servicemanagement aufgeteilt.

Grafische Benutzerschnittstelle

Die Szenarien der grafischen Benutzerschnittstelle setzen sich aus dem GUI-Renderer, GUI-Kombinierer und einem GUI-Gerät zusammen. Die Realisierung entspricht einem Webbrowser, Webserver und einem Beispielservice mit GUI-Erweiterungen. Der Sicherheitsaspekt wird hier nicht mitbetrachtet, weil sich dieser eher bei der Servicenutzung bemerkbar macht. Bei der Evaluierung werden die drei Komponenten mit einem 10/100 Mbit/s Fast Ethernet Switch verbunden. In Tabelle 6.3 sind die Spezifikationen der Hardware der Komponenten dargestellt.

Komponente	Prozessor	RAM (MB)	Betriebssystem
GUI-Renderer	Intel Pentium 4 @ 2,40GHz	512	Debian 5.0.3 (Lenny)
	Intel Core 2 Duo @ 2,53GHz	4000	Mac OS X 10.6.5
GUI-Kombinierer	Intel Core2 @ 2,16GHz	2000	Ubuntu 10.04 (Lucid)
GUI-Gerät	Intel Core2 @ 2,13GHz	1000	Debian 5.0.6 (Lenny)

Tabelle 6.3.: *Hardwarespezifikation der Teilnehmer in den Testszenarien (Grafische Benutzerschnittstelle).*

Des Weiteren werden zwei verschiedene GUI-Inhalte generiert. Das erste Design ist sehr einfach gehalten (Simple) und das andere ist eine Adaption der in BMW-Serienfahrzeugen verbauten Oberfläche (BMW). Da das vorgeschlagene System auch mit einem Standardbrowser nutzbar sein soll, werden zur Evaluierung Mozilla Firefox und der Apple Safari Webbrowser in den jeweiligen Szenarien hinzugezogen. Damit zeitliche Messungen zwischen den Komponenten durchgeführt werden können, wird eine gemeinsame Zeitbasis durch das Precision Time Protocol (PTP) eingeführt.

Simple-GUI: Diese Oberfläche ist sehr einfach gehalten. Es gibt lediglich eine einfache HTML-Struktur, auf CSS für das Aussehen und zusätzliche Animationen mittels JavaScript wird verzichtet. Die Simple-GUI ist zusammen mit dem Grundgerüst, das nötig für die Funktionalitäten des GUI-Renderers ist, die kleinste mögliche Kombination einer GUI des MMS-Systems. In Abbildung 6.3 ist ein Screenshot der Simple-GUI zu sehen.

Wie zu sehen ist, erscheint die GUI einfach und leer. Es sind lediglich Menüeinträge zu sehen und die in HTML-realisierte Ansteuerung für das Navigieren durch das Menü.

BMW-GUI: Neben der zuvor genannten minimalistisch gehaltenen Oberfläche, wird mit dieser GUI eine durchschnittlich komplexe GUI gezeigt. Diese besitzt wiederum eine HTML-

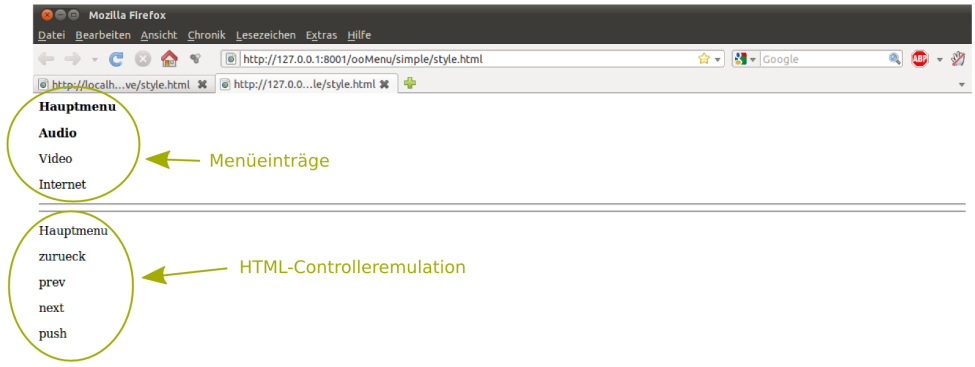

Abbildung 6.3.: *Screenshot der Simple-GUI.*

Struktur. Hierbei wird das Aussehen jedoch mit CSS-Beschreibungen und Hintergründen aufgewertet. Auch Animationseffekte werden über JavaScript ergänzt. Diese GUI soll keine perfekte Kopie der originalen BMW-Oberfläche darstellen, sondern nur als Beispiel für ein komplexes GUI-Design dienen.

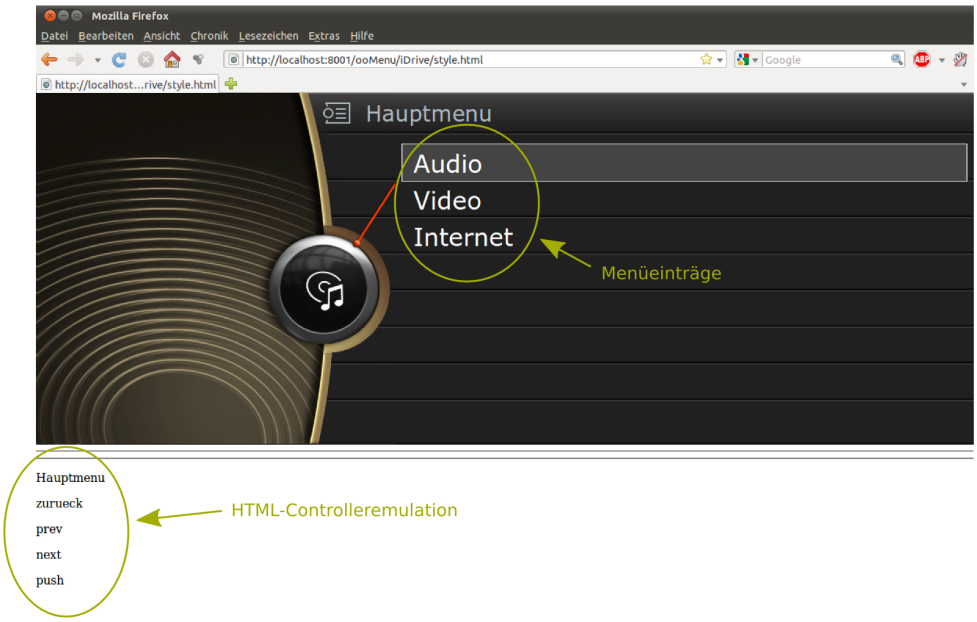

Abbildung 6.4.: *Screenshot der BMW-GUI.*

In Abbildung 6.4 ist ein Screenshot der BMW-GUI abgebildet. Es kann deutlich eine Ähnlichkeit zu der Abbildung 2.1 erkannt werden, die Menüstruktur ist auch dieselbe. Es wird lediglich ein anderes Aussehen und Verhalten über die Inhalte gelegt.

In Tabelle 6.4 wird eine Übersicht über den Speicherverbrauch der Grundelemente und GUI-Teile gegeben. Deutlich zu erkennen ist der geringe Speicherbedarf für das Grundgerüst des GUI-Renderers. Dabei handelt es sich um alle logischen Elemente, die im GUI-Renderer benötigt werden um die Interaktion mit dem Nutzer durchzuführen. Die Gesamtgröße beläuft sich auf 21,2 KB und umfasst größtenteils JavaScript-Code.

	Größe (KB)	Größe mit Grundgerüst (KB)
Grundgerüst	21,2	–
Simple-GUI	4,2	25,4
BMW-GUI	158,1	179,3
Anwendung	45,3	–

Tabelle 6.4.: *Speicherbedarf der vorgestellten GUIs einschließlich Grundgerüst.*

Wie bereits erwähnt bedarf es einer minimalen GUI um eine Funktionstüchtigkeit des MMS-Systems zu gewährleisten. Dafür werden etwa 4,2 KB benötigt. Das bedeutet, dass die Verwendung der minimalen Kombination für eine GUI addiert mit dem Grundgerüst 25,4 KB ausmacht. Der Komplexität der GUI sind nahezu keine Grenzen gesteckt. Die BMW-GUI zeigt in etwa bereits ein ähnliches Aussehen und Verhalten wie die originale BMW-Oberfläche. Der Speicherbedarf wird hier mit 158,1 KB aufgeführt und ergibt zusammen mit dem Grundgerüst 179,3 KB. Der Speicherbedarf der Beispielanwendung verhält sich ähnlich. Auch hier kann die Komplexität beliebig viel Speicherplatz benötigen, aber zu Evaluationszwecken wird die vorgestellte Beispielanwendung einfach gehalten.

Servicemanagement

Die quantitative Evaluierung des Servicemanagements bezieht gleichzeitig auch die Sicherheitsaspekte mit ein. Die Testszenarien bestehen immer aus drei Komponenten, dem Client, dem Service und dem AAA-Server/CA. Diese werden über einen 10/100 Mbit/s Fast Ethernet Switch verbunden. Auf allen drei Teilnehmern werden die für die MMS wichtigen Servicekomponenten installiert. Dabei handelt es sich um DPWS Core und die Erweiterungen durch diese Arbeit, X.509v3-Zertifikate, generiert mit dem RSA Algorithmus und einem 512 Bit langen Schlüssel sowie sicherheitsrelevante Software wie OpenSSL, OpenLDAP und den Sicherheitserweiterungen dieser Arbeit. Die Hardwarespezifikation der Teilnehmer wird in Tabelle 6.5 gezeigt.

Komponente	Prozessor	Speicher	Betriebssystem
Client	Intel Core2 @ 2,13GHz	1 GB	Debian 5.0.6 (Lenny)
Service	Intel Core2 @ 2,16GHz	2 GB	Ubuntu 10.04 (Lucid)
AAA-Server/CA	ARM XScale-IXP42x	32 MB	Debian 5.0.5 (Lenny)

Tabelle 6.5.: *Hardwarespezifikation der Teilnehmer in den Testszenarien (Servicemanagement).*

Die zu evaluierende servicebasierte Anwendung stellt eine Funktionalität dar, die ein Datenpaket mit variabler Größe empfängt, verarbeitet und zurückschickt. In dem Datenpaket können beliebige Daten enthalten sein. Der Ablauf der Testszenarien sieht folgendermaßen aus: Zuerst werden die für die sichere Übertragung nötigen Schritte durchgeführt. Das bedeutet die Authentifizierung und Autorisierung. Dann erstellt der Client ein Datenpaket mit dem gewünschten Inhalt und sendet dieses über den sicheren verschlüsselten Kanal. Nach dem Empfang des Pakets entschlüsselt der Empfänger dieses nur zu Demonstrationszwecken und sendet es zurück an den Client. Die Abläufe der Kommunikation entsprechen etwa denen in Kapitel 5.4.

Für die Performanzanalyse werden neben den fünf verschiedenen Chiffren DES, RC4, AES128, 3DES und AES256 die Größen 1 MB, 10 MB und 100 MB für das Datenpaket verwendet. Um eine genaue Zeitmessung durchführen zu können, werden alle Teilnehmer wiederum durch PTP synchronisiert.

6.3.2. Grafische Benutzerschnittstelle

In diesem Abschnitt werden relevante Aspekte hinsichtlich der grafischen Benutzerschnittstelle erläutert.

Aufstartzeiten des GUI-Renderers mit Basissystem

In Tabelle 6.6 werden verschiedene Kombinationen aus möglichen GUI-Render Realisierungen miteinander verglichen. Als Basis für die Realisierung diente in diesen Messungen das Minimalsystem Microcore. Die in Tabelle 6.6 gezeigten Zeiten beinhalten die Systemstartzeit und die Startzeit für den GUI-Renderer.

Realisierung	Firefox	Qt X11-basiert	QT framebufferbasiert
Aufstartzeit	26 s	18 s	11 s
Speichergröße	21,5 MB	84,6 MB	79,6 MB

Tabelle 6.6.: *Vergleich der aufgesetzten Minimalsysteme.*

Wie zu sehen ist, erreicht der Standardbrowser Mozilla Firefox eine Aufstartzeit von etwa 26 Sekunden. Diese Zeit lässt sich durch die Implementierung eines eigenen Browsers noch weiter reduzieren. Mit einem X11-basierten Qt-System wird bereits eine Aufstartzeit von 18 Sekunden erreicht. Durch den Verzicht auf den X-Server lässt sich die Aufstartzeit auf 11 Sekunden verringern. Dies ist bereits im Rahmen von automotiven Anforderungen bezüglich des Aufstartverhaltens. Zusätzlich kann durch Hardwareanpassungen noch eine weitere Reduzierung erreicht werden.
Die Speichergröße nimmt verglichen zu dem Standardbrowser zu. Dies ist jedoch durch die verwendeten Qt-Bibliotheken bedingt, bei denen die verwendete Speichernutzung noch optimiert werden kann.

Initiales Laden des GUI-Inhaltes

Diese Messungen werden mit einem Webbrowser, einem Webserver und einem GUI-Gerät durchgeführt. Es werden die erwähnten zwei verschiedenen Basis-GUIs „Simple“ und „BMW“ und zusätzlich eine einfach aufgebaute Beispielanwendung verwendet. Die Basis-GUI wird vom Webserver direkt und die Anwendungs-GUI über den Webserver vom GUI-Gerät bezogen. In Abbildung 6.5 wird die Ladezeit zusammen mit der Darstellungszeit gezeigt. Die Ladezeit umschließt alle Vorgänge des Anforderns, Empfangens und Ausführens der Funktionen für den GUI-Inhalt, wie z.B. das Ansteuern eines GUI-Gerätes. Dabei beinhaltet die Darstellungszeit die Ladezeit sowie die Zeit für das Rendern des Inhalts.

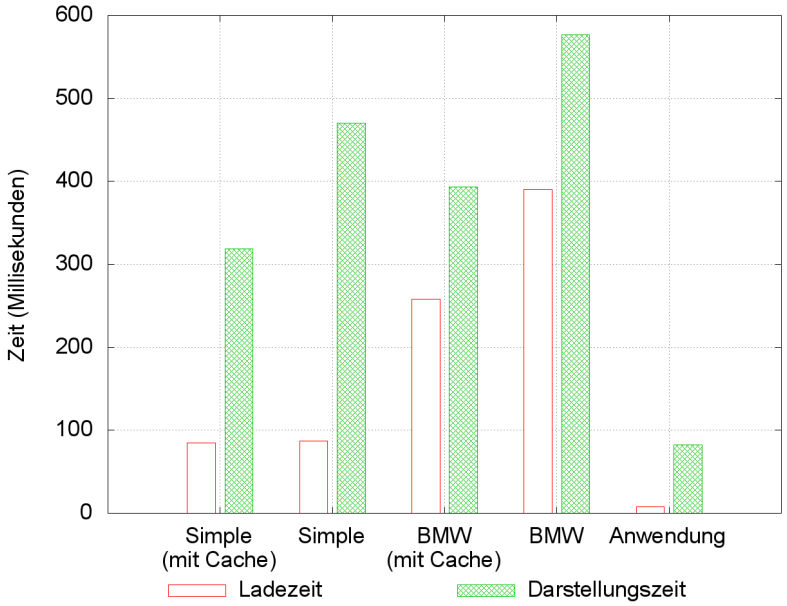

Abbildung 6.5.: *Initiale Dauer für das Laden verschiedener GUI-Inhalte.*

Die Ladezeiten werden für die Basis-GUIs einmal mit dem im Webbrowser eingebauten Cache und einmal ohne diesen betrachtet. Der Cache bewirkt ein Zwischenspeichern von bereits geladenen Informationen und kommt zum Einsatz falls der gleiche Inhalt nochmals geladen werden soll. In diesem Fall werden die Inhalte aus dem Cache anstatt vom Webserver bezogen.
Es geht deutlich hervor, dass die steigende Komplexität der GUI einen Anstieg der Ladezeit bewirkt. So ist die Ladezeit der „BMW“-Oberfläche mit und ohne Cache höher als die der „Simple“-GUI. Die Verwendung des Caches reduziert die Zeit. Die sehr einfach gehaltene „Simple“-Oberfläche erreicht mit Cache eine Darstellungszeit von 85 ms und selbst die bereits ansehnliche „BMW“-Oberfläche benötigt maximal eine Darstellungszeit von etwa 580 ms. Dabei handelt es sich um die Zeiten, die initial nötig sind und wesentlich kürzer sind als die Systemstartzeit. Danach werden die Basiselemente nur noch im Webbrowser entsprechend neu angeordnet oder es werden Anwendungsdaten nachgeladen. Bei einer einfachen Ausführung einer Anwendung bewegt sich die Darstellungszeit bei lediglich 82 ms und da bei jeder Verwendung die Anwendung neu geladen wird, kann keine Betrachtung mit Cache durchgeführt werden.

Aufstartzeit des GUI-Kombinierers

Diese Messungen werden mit einem Webserver durchgeführt. Es handelt sich dabei um den vorgestellten Webserver tntnet mit der integrierten SOA-API. Für die Bereitstellung der SOA-API werden die beiden Clients „Command“ und „Subscription“ (siehe 4.3.1) initiiert. Die Aufstartzeit stellt die Zeit dar, die benötigt wird um den GUI-Kombinierer zu starten ohne eine Gerätesuche durchzuführen. Die Messungen ergeben, dass sich die Zeit, die die beiden Clients benötigen, nahezu identisch ist und sich bei etwa 2,6 Sekunden bewegt. Die

Zeit die für das Starten des Webservers noch zusätzlich benötigt wird, ist vernachlässigbar.

Dauer der GUI-Gerätesuche

Die Dauer die benötigt wird um die GUI-Geräte zu finden wird in diesen Messungen genauer betrachtet. Dafür wird ein Webserver und mehrere GUI-Geräte verwendet. Das GUI-Gerät wird je nach Anzahl mehrmals auf einer Hardware initiiert. In Abbildung 6.6 ist die Suchdauer und die Anzahl der Geräte zu sehen.

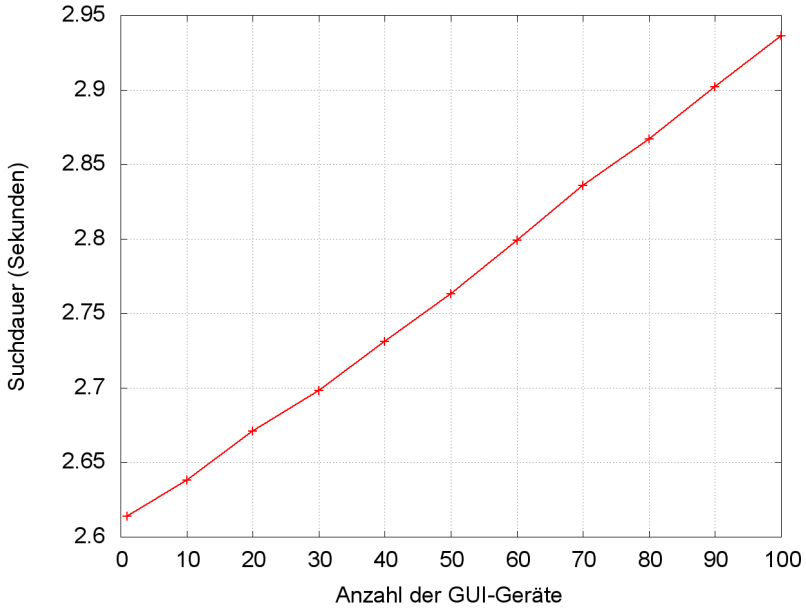

Abbildung 6.6.: *Suchdauer im Browser bis alle Geräte gefunden werden und dem Nutzer bereitstehen.*

Der Verlauf zeigt eine lineare Charakteristik, wobei der Suchvorgang an sich etwa 2,62 Sekunden in Anspruch nimmt und sich pro Gerät verlängert. Ein Anstieg der GUI-Geräte um 10 Stück bewirkt dabei einen Zuwachs von etwa 0,04 Sekunden bei der Suchzeit. Dieser Zusammenhang wird in Gleichung 6.1 dargestellt.

$$t = 2,62s + n * 0,004s \tag{6.1}$$

Die Dauer der GUI-Gerätesuche t steigt linear mit der Anzahl der Geräte n.

Benachrichtigungsverzögerung im GUI-Renderer

Für die Messung der Verzögerung bei Benachrichtigungen wird ein Webbrowser, Webserver und ein GUI-Gerät verwendet. Als Webbrowser werden Mozilla Firefox und Apple Safari verwendet. Diese abonnieren die zu evaluierende Anzahl der Benachrichtigungskanäle. Im Anschluss sendet das GUI-Gerät eine Benachrichtigung aus. Die Benachrichtigung enthält

einen Zeitstempel des Aussendens, am Webbrowser wird daraus dann die Differenz aus dem Aussendezeitpunkt und Verarbeitungszeitpunkt gebildet. Somit wird gezeigt wie lange es dauert, bis Benachrichtigungen empfangen und verarbeitet werden. In Abbildung 6.7 wird die Verzögerungszeit gegen die Anzahl der gleichzeitigen Benachrichtigungen gezeigt.

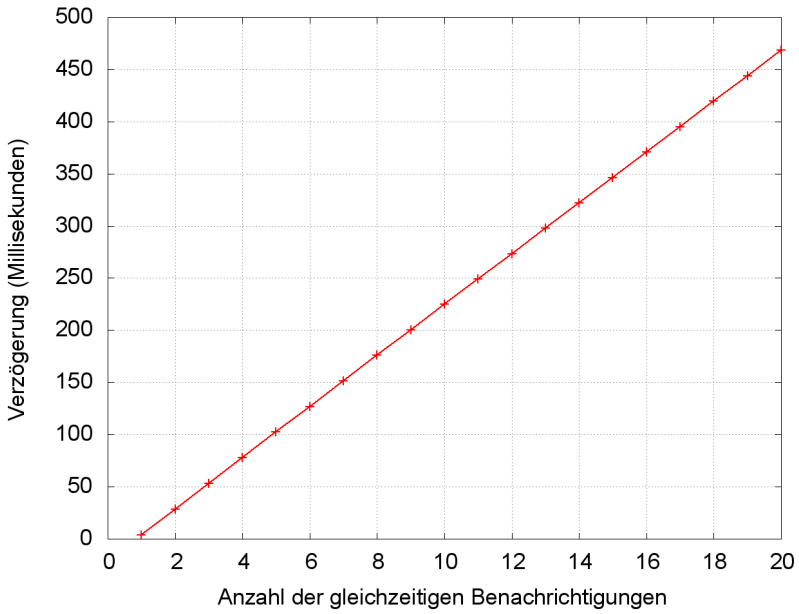

Abbildung 6.7.: *Verzögerung der Benachrichtigungen im Webbrowser Mozilla Firefox.*

Im Mozilla Firefox steigt die Verzögerung linear mit der Anzahl der gleichzeitigen Benachrichtigungen an. Die erste Benachrichtigung benötigt etwa 4 ms für die Verarbeitung. Darauffolgende Benachrichtigungen verbrauchen aber jeweils etwa 25 ms zusätzlich für die Abläufe. Bei korrespondierenden Messungen mit dem Webbrowser Apple Safari stellt sich heraus, dass ein Unterschied zwischen der Abarbeitung der Benachrichtigung in den verschiedenen Webbrowsern besteht. Im Mozilla Firefox hat sich zusätzlich gezeigt, dass jede weitere Benachrichtigung erst nach 25 ms verarbeitet werden kann (einschließlich der Verarbeitungszeit). Dagegen wird der Apple Safari nicht blockiert und die Verzögerung durch die Benachrichtigung ist nahezu konstant.
Neben den Benachrichtigungen können auch Serviceaufrufe durchgeführt werden. Bei diesen ist die Verzögerung ähnlich. Es werden 4 ms von der Auslösung in der GUI bis zum Empfang bei dem jeweiligen Service benötigt. Falls eine Antwort erwartet wird, benötigt diese ebenfalls wiederum etwa 4 ms. Somit werden für einen Umlauf etwa 8 ms nötig. Diese Zeit ist so gering, dass sich grafische Effekte wie die Änderung eines Knopfes in der grafischen Darstellung ohne bemerkbare Beeinträchtigung durchführen lassen. Man könnte daher das Aussehen eines Start-Knopfs instantan so ändern, dass dieser ohne eine für den Nutzer sichtbare Verzögerung zu einem Stopp-Knopf wird.

Prozessorauslastung

Für diese Messungen werden ein Webserver und ein GUI-Gerät verwendet. Um die Prozessorauslastung zu analysieren wird die jeweilige Anzahl der GUI-Geräte gestartet und die Auslastung des Prozessors auf dem Webserver und auf dem GUI-Gerät gemessen. Die einzelnen GUI-Geräte werden auf einem Gerät initialisiert und werden in Abbildung 6.8 und 6.9 als auf einem Gerät „vereinte GUI-Geräte“ dargestellt. Abbildung 6.8 zeigt die Auslastung des Prozessors für den Webserver und das Gerät.

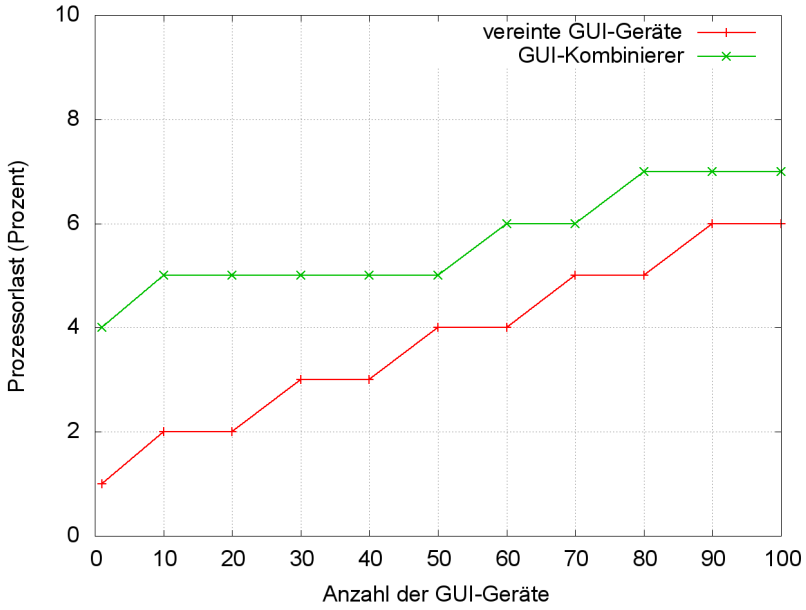

Abbildung 6.8.: *Prozessorauslastung von GUI-Geräten und Webserver (GUI-Kombinierer).*

Für die auf einem Gerät gestarteten GUI-Geräte kann ein linearer Anstieg mit einer Steigung von 0,5% pro 10 GUI-Geräte beobachtet werden. Der Anstieg der im Webserver durch die Zunahme an GUI-Geräten verursacht wird, besitzt auch annähernd einen linearen Anstieg nur ist die Steigung flacher. So machen 10 GUI-Geräte etwa eine Prozessorlast von zusätzlich 0,2% aus. Dieser geringe Anstieg unterstreicht die Vorteile der Verwendung von Webtechnologien für die GUI-Generierung und -Verteilung.
Der Prozessor beim GUI-Renderer ist nur zu Beginn der Initialisierung kurz belastet, wenn die Elemente und das Grundgerüst mit der jeweiligen GUI geladen werden. Im Anschluss daran wird nahezu keine Auslastung mehr gemessen.

Speicherauslastung

Die verwendeten Komponenten für die Messung bestehen aus einem Webserver und einem Gerät. Wiederum wird auf einem Gerät die jeweilige Anzahl an GUI-Geräten gestartet und die Speichernutzung auf dem Gerät und auf dem Webserver mitprotokolliert. Der Speicherverbrauch wird nach der Suche und dem Auffinden der Geräte durch den Webserver gemessen.

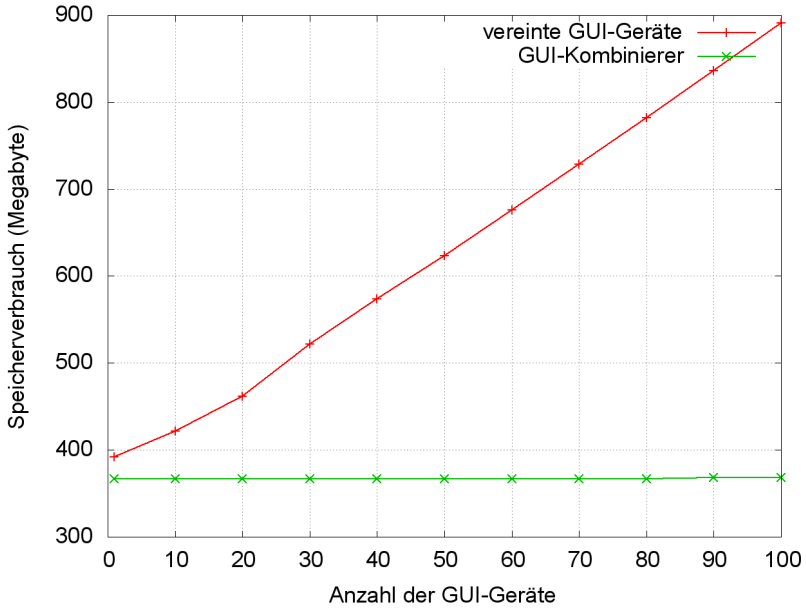

Abbildung 6.9.: *Speichernutzung von GUI-Geräten und Webserver (GUI-Kombinierer).*

Die steigende Zahl der GUI-Geräte resultiert in einem linearen Anstieg bei der Speichernutzung. Die Nutzung beträgt etwa 50 MB pro 10 GUI-Geräte. Der Speicherverbrauch bei zunehmenden Geräten bedeutet dagegen nahezu keinen Zuwachs im Webserver.
Der Speicherverbrauch des GUI-Renderers ist auch nahezu konstant. Er benötigt einmal für sich und für den GUI-Inhalt Speicher, danach werden jedoch alle nicht benötigten Elemente wieder aus dem GUI-Inhalt entfernt. Auf diese Weise wird innerhalb der GUI äußerst ressourcenfreundlich hinsichtlich der Speichernutzung gehandelt.

6.3.3. Servicemanagement

Im Folgenden werden Performanzbetrachtungen für das Servicemanagement durchgeführt. Dabei geht es um Messungen hinsichtlich der Prozessor- und Speicherauslastung, Zeiten des initialen Aufstartverhaltens, der Übertragungsverzögerung und Auslastung des Netzes.

Prozessorauslastung

Diese Messungen wurden mit dem Datenpaket-Service durchgeführt. Dabei werden die verschiedenen Chiffren der Verschlüsselung für drei verschiedene Dateigrößen (1 MB, 10 MB, 100 MB) evaluiert. Die dabei erzeugte Prozessorauslastung für den Service wird in den nachfolgenden Abbildungen gezeigt.

In Abbildung 6.10 ist die CPU Auslastung für die Messung mit einer 1 MB Datei zu sehen. Während dem Zeitraum von der 8. Sekunde bis zur 15. Sekunde sorgt die Übertragung der Nachrichten für signifikante Ausschläge der Auslastung. Auch wenn die Chiffren 3DES und

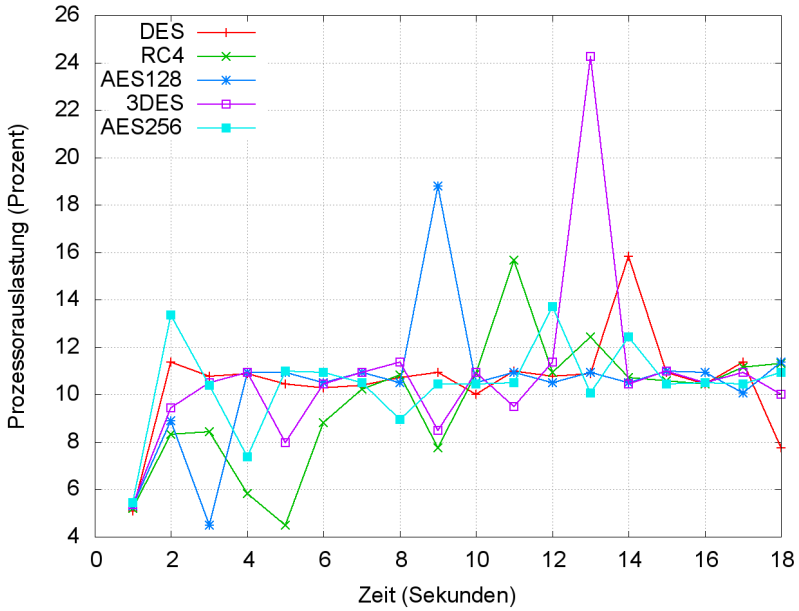

Abbildung 6.10.: *Prozessorauslastung eines Services (1 MB Datei).*

AES128 hier 25% und 20% der Auslastung ausmachen, benötigen die restlichen Chiffren lediglich etwa 15% oder weniger. Dennoch ist das Verhalten von AES256 erwähnenswert, da dieses während der gesamten Messung nie mehr als 15% und während unausgelasteten Zeiten nur 6% der Prozessorauslastung in Anspruch nimmt.

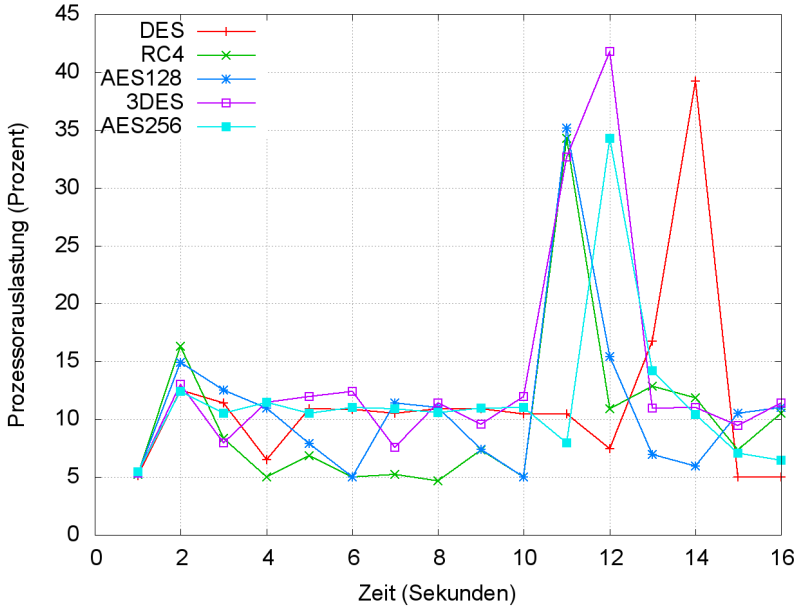

Abbildung 6.11.: *Prozessorauslastung eines Services (10 MB Datei).*

Abbildung 6.11 zeigt das Verhalten der Chiffren für die Messung mit einer 10 MB Datei. Zur 10. Sekunde wird die Übertragung initiiert und dauert etwa 2 Sekunden. Die Prozesso-

rauslastung ist für jeden Algorithmus nahezu gleichförmig, wobei die Werte von DES und 3DES bis über 40% steigen und die Werte von RC4, AES128 und AES256 sich zwischen 32% und 35% bewegen. Für die restliche Zeit ist der Verlauf für alle Chiffren stabil und ähnlich bei Werten von etwa 5% bis 15%.

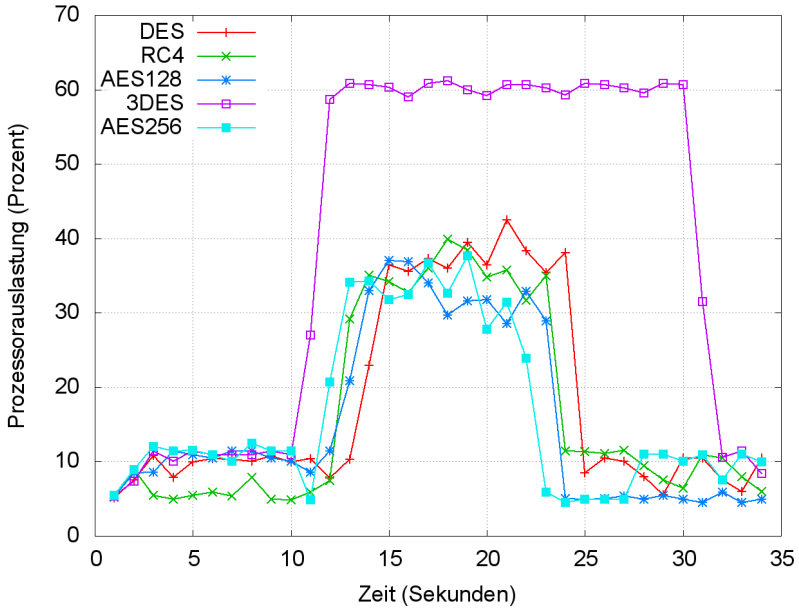

Abbildung 6.12.: *Prozessorauslastung eines Services (100 MB Datei).*

Das letzte Diagramm dieser Messreihen (Abbildung 6.12) zeigt die noch ausstehende Messung mit einer Datei mit 100 MB Größe. Die Übertragung startet zur 10. Sekunde und dauert etwa 15 Sekunden. Die meisten Chiffren belasten dabei den Prozessor nur zu 40%. Alle Verfahren kehren nach der Übertragung in eine unbelastete Phase zurück. Nur 3DES belastet den Prozessor ab dem Start der Übertragung bis zur 32. Sekunde mit etwa 60%. Das bedeutet, dass diese Chiffre eine längere und stärkere Auslastung des Prozessors im Vergleich zu den restlichen Chiffren bewirkt. Die stärkere Belastung des Prozessors durch 3DES ist durchgängig über alle Messungen hinweg sichtbar.

Speicherauslastung

Ähnlich wie in den vorherigen Messungen wird der Service für die Dateiübertragung mit drei verschiedenen Dateigrößen und den unterstützten Chiffren verwendet. Dieses Mal wird aber die Speichernutzung durch den Service betrachtet.
Die Abbildung 6.13 zeigt die Nutzung von Random-Access Memory (RAM), inklusive der Speichernutzung des Betriebssystems, während der Übertragung einer 1 MB Datei.

Alle Werte der Chiffren sind stabil und zeigen keine großen Schwankungen. Die einzig nennenswerte Beobachtung in diesem Diagramm ist der erhebliche Unterschied des Speicherverbrauchs der Chiffre DES im Vergleich zu den anderen. Dieser verbleibt bei 212 MB während der gesamten Laufzeit des Algorithmus, während die anderen nicht mehr als 180

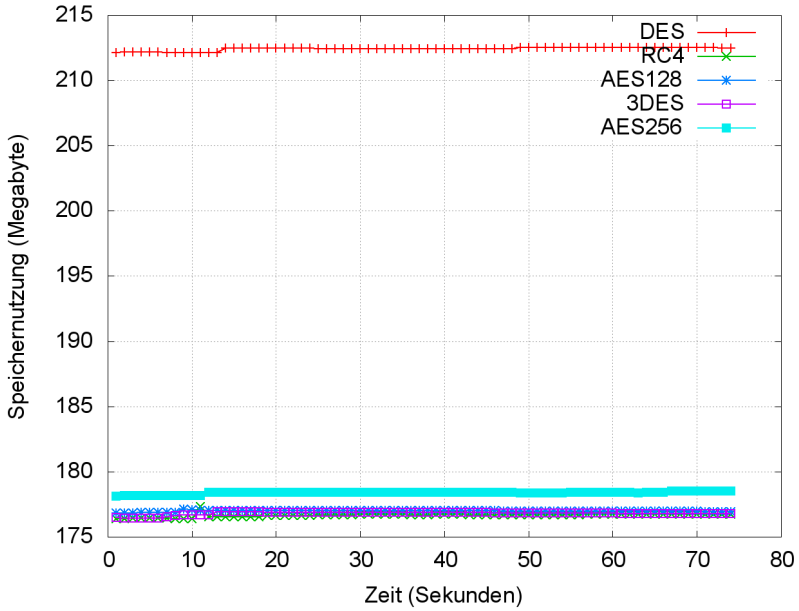

Abbildung 6.13.: *Speichernutzung eines Services (1 MB).*

MB benötigen. Dies ist ebenfalls auf die interne Struktur dieses blockbasierten Chiffres zurückzuführen. Die nächsten Messungen mit den 10 MB Dateien (Abbildung 6.14) unterscheiden sich nicht signifikant von der zuvorigen.

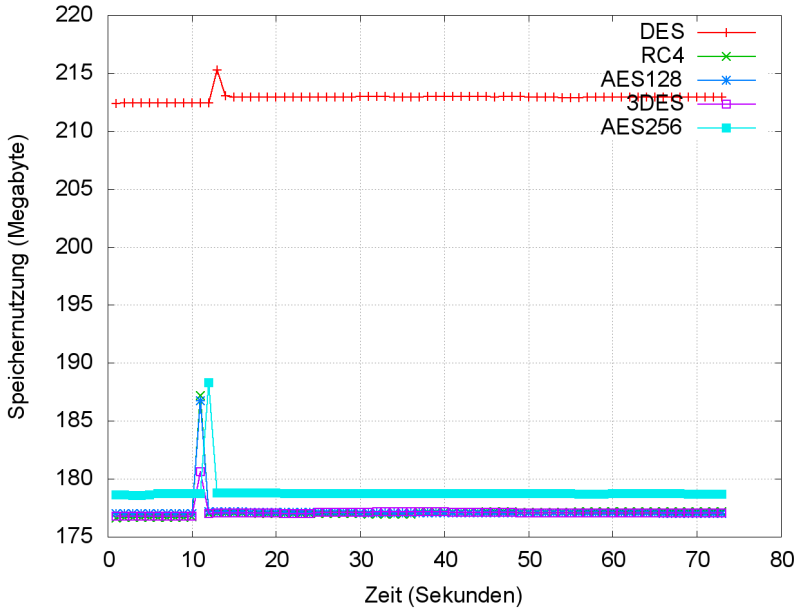

Abbildung 6.14.: *Speichernutzung eines Services (10 MB).*

Wieder bewegen sich die Werte von DES durchgängig bei etwa 212 MB und der Ressourcenverbrauch der anderen Algorithmen bei einer unteren Grenze von etwa 180 MB. Zusätzlich kann bei diesem Messvorgang ein Ausschlag nach etwa 10 Sekunden nachdem die

Übertragung gestartet wurde beobachtet werden. Dieser überschreitet bei allen Algorithmen außer DES aber nicht die 190 MB Marke. Aus beiden Graphen lassen sich eine geringe Auswirkung auf den Speicher durch 1 MB oder 10 MB Dateiübertragungen ableiten.
Dagegen stellt Abbildung 6.15 das Verhalten für eine 100 MB Datei dar. In diesem Fall ist der Ausschlag durch die Übertragung ausgeprägter. Durch die längere Dauer und höhere Werte erreichen alle Chiffren eine Nutzung des Speichers von 280 MB, abgesehen von 3DES mit einer maximalen Nutzung von 310 MB. Aus den Abbildungen wird ersichtlich, dass der Anstieg der Speichernutzung jeweils der zu verschlüsselnden Daten entspricht. Diese Speicherausnutzung ließe sich durch eine ressourcenfreundliche und auf eingebettete Systeme angepasste Implementierung senken.

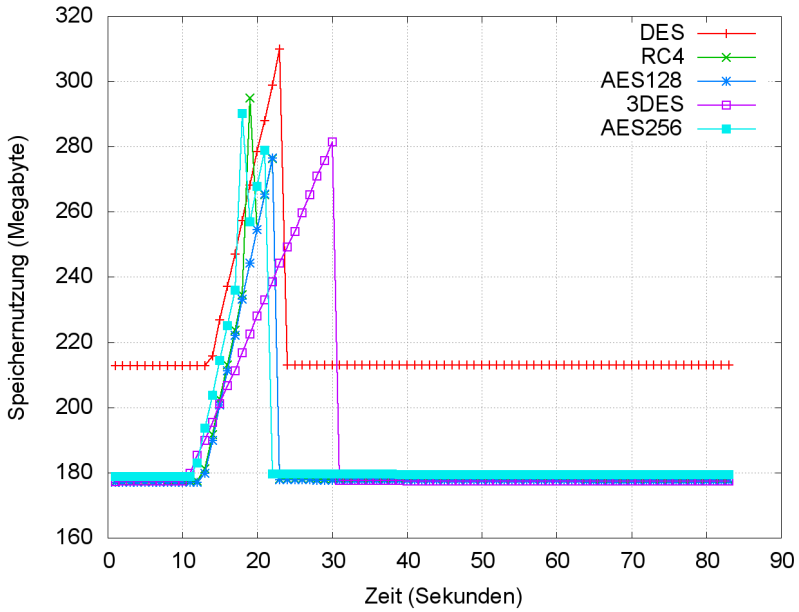

Abbildung 6.15.: *Speichernutzung eines Services (100 MB).*

Zusätzlich muss angemerkt werden, dass 3DES mindestens 8 Sekunden länger benötigt um den benutzten Speicher freizugeben und zu der gewohnten Speichernutzung von 180 MB zurückkehrt. Bei dieser Chiffre ist ebenfalls eine flachere Steigung d.h. ein langsamerer Speicherzugriff verglichen zu den anderen Chiffren bemerkbar.

Initiales Aufstartverhalten

Der folgende Test beabsichtigt die Zeit zu messen, die für die DPWS-Initialisierung und für das Etablieren der Sicherheitserweiterungen für den Service und für den Client benötigt wird.
In Abbildung 6.16 ist zu sehen, dass durch den Service weniger als 1 ms für das Durchführen der DPWS-Initialisierung, eingeschlossen der Gerätekonfiguration, Serviceerstellung und der Aktivierung des Gerätes, benötigt wird. Zusätzlich werden etwa 8 ms durch die Sicherheitsmechanismen, welche die TLS Erstellung, LDAP-Konfiguration und Einrichtung der Sicherheitsparameter beinhaltet, verbraucht.

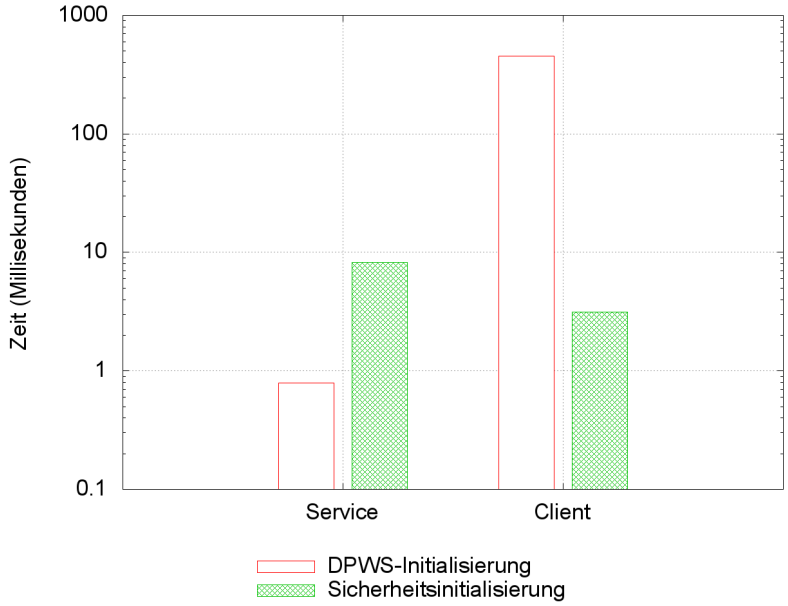

Abbildung 6.16.: *Initiale Aufstartzeit jeweils für Service und Client.*

Zusammengefasst werden nicht mehr als 9 ms für das Aufstarten eines sicheren Services unabhängig von der gewählten Chiffre benötigt. Dagegen vergeht bei dem Client mehr Zeit für die Initialisierung der DPWS-Komponente als für die Sicherheitserweiterungen. Der zuerst genannte Vorgang dauert 450 ms bis die initiale Phase abgeschlossen wurde und lediglich etwa 1 ms für den sicherheitsrelevanten Anteil. Der Grund hierfür ist die notwendige Interaktion zwischen Service und Client um das Aufstarten zu vervollständigen. DPWS beinhaltet zu diesem Zweck die Suche nach dem betreffenden Service, die Auswertung der Ergebnisse dieser Suche und den Austausch diesbezüglicher Metainformationen. Diese Vorgänge und besonders die Suche sind vom Netz und der Verzögerung des Gegenübers abhängig und machen den größten Anteil der Verzögerung aus. Dagegen wird durch die Sicherheitserweiterungen lediglich der TLS Kanal initiiert und die weiteren Sicherheitsmechanismen eingerichtet. Auch hier haben die Chiffren keinerlei Einfluss auf die Initialisierungsphase.

Übertragungsverzögerung

Um einen sicheren Serviceaufruf abzuarbeiten, wird eine 3-Wege Kommunikation benötigt. Zu Beginn authentifizieren sich Client und Service gegenseitig unter Zuhilfenahme der CA. Des Weiteren sendet der Client den Serviceaufruf an den Service und dieser leitet ihn weiter an den AAA-Server. Dort werden die betreffenden Sicherheitsabfragen gemacht und zum Service zurückgeschickt. Dieser beantwortet schließlich den Serviceaufruf des Clients (für weitere Details siehe Kapitel 5.4). Im Anschluss an diesen Vorgang findet der gesicherte Austausch von Daten zwischen Service und Client statt. Die folgenden Messungen wurden wiederum mit dem Datenpaket-Service für verschiedene Dateigrößen (1 MB, 10 MB und 100 MB), mit jeder unterstützten Chiffre für die sichere Übertragung und für die unsiche-

re Übertragung zu Vergleichszwecken vorgenommen. Abbildung 6.17 zeigt die Dauer einer Übertragung im ungesicherten Fall, das bedeutet keine der Chiffren werden hier angewendet.

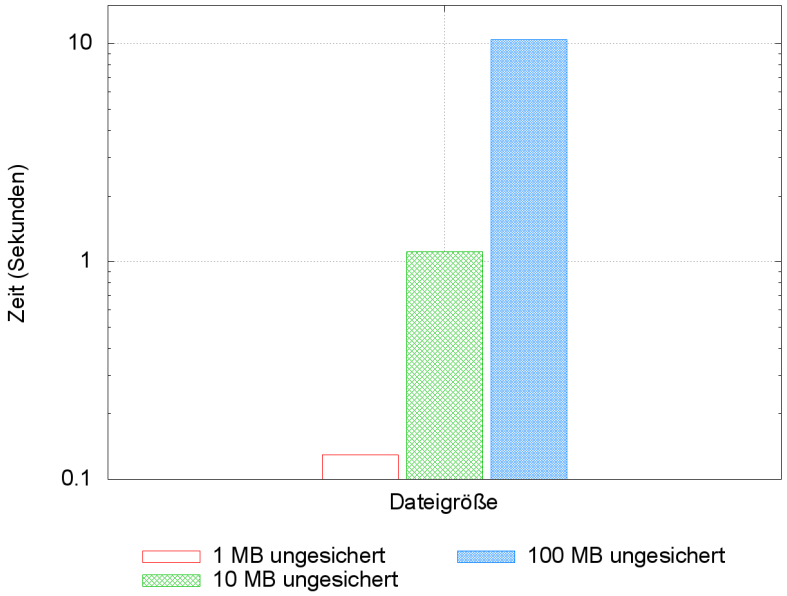

Abbildung 6.17.: *Dauer für die unsichere Übertragung in einem Client.*

Die Übertragung einer 1 MB Datei dauert 0,13 Sekunden, die der 10 MB Datei 1,11 Sekunden und für die 100 MB Datei werden 10,48 Sekunden gemessen. Darauf aufbauend zeigt Abbildung 6.18 die zusätzliche Zeit, die für die Übertragung mit einer der unterstützten Chiffren veranschlagt werden muss. Die zusätzliche Dauer setzt sich zum einen aus der Zeit für die Verschlüsselung der Daten, als auch aus der 3-Wege Kommunikation zum Erstellen eines sicheren Kanals, zusammen.

Werden die Werte der 1 MB Datei betrachtet, zeigen diese, dass AES128 und AES256 die geringsten Verzögerungen, dicht gefolgt von den Chiffren DES und RC4 besitzen. Mit weniger als 0,1 Sekunden zusätzlicher Verzögerung bewegt sich der Wert der Chiffre 3DES nicht allzu weit entfernt von den anderen Algorithmen. Bei einer Dateigröße von 10 MB zeigen die Messungen ein ähnliches Verhalten. Die Chiffren AES128 und AES256 gehen mit etwa 0,05 Sekunden Verzögerung wiederum als die effizientesten Chiffren hervor. Hingegen zeigen die restlichen Algorithmen eine nur wenig schlechtere Performanz. Lediglich 3DES benötigt etwa 0,15 Sekunden länger als die restlichen Chiffren. Schlussendlich zeigt bei der Übertragung der 100 MB Datei AES256 das beste Verhalten unter den AES Algorithmen. Diese benötigt nur etwa 0,1 Sekunden für die zusätzliche Dauer um die sichere Nachrichtenübertragung durchzuführen, während dagegen DES und RC4 bis zu 0,35 Sekunden dafür benötigen. Zusammengefasst kann gesagt werden, dass sich die Gesamtverzögerung unabhängig von der Dateigröße kaum zwischen den verschiedenen unterstützten Chiffren unterscheiden. Obwohl sich mit AES128 und AES256 bezüglich der Verzögerung kleine zeitliche Einsparungen gegenüber den anderen Algorithmen ergeben. Bemerkenswert hierbei ist, dass die stärksten Chiffren (AES128 und AES256) durchweg die geringste Verzögerung mit sich

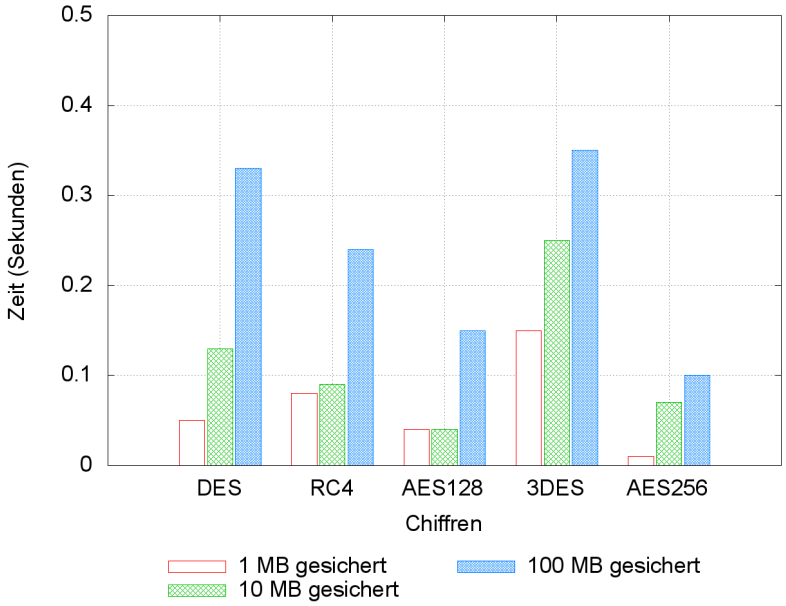

Abbildung 6.18.: *Zusätzliche Dauer für die sichere Übertragung in einem Client.*

bringen. Beispielsweise benötigt die sichere Übertragung einer 1 MB Datei mit AES256 nur 0,14 Sekunden (kombinierte Werte aus Abbildung 6.17 und 6.18).

Netzauslastung

Für die Messung der Netzauslastung wurde ein Vergleich zwischen sicheren und unsicheren Services durchgeführt. Dafür wurde der Service für die Datenpaketübertragung mit den verschiedenen Dateigrößen jeweils einmal mit und einmal ohne Sicherheitserweiterungen genutzt. Das Ziel war dabei, den Umfang an gesendeten und empfangenen Daten auf der Serviceseite zu messen. Dies geschieht durch Mitschneiden des eingehenden und ausgehenden Verkehrs. Dieser wird durch das Aushandeln der TLS-Chiffren, der Konfiguration der sicheren Services und der Bestimmung der Ports für die sichere Verbindung verursacht. Diese Messungen wurden nicht für verschiedene Chiffren durchgeführt, da der Umfang der gesendeten und empfangenen Daten jeweils der gleiche ist. Es werden keine zusätzlichen Informationen durch die Verwendung von symmetrischen Algorithmen an die ursprünglichen Nachrichten angehängt. Deswegen liegt der Fokus in Abbildung 6.19 auf der Netzauslastung für die verschiedenen Dateigrößen bei der Übertragung.

Bei den Messungen mit der 1 MB Datei wurde ermittelt, dass durch die sichere Verbindung 1,25 MB und durch die ungesicherte Verbindung 1,237 MB Verkehr verursacht werden. Diese geringe Differenz beruht auf dem zusätzlichen Verkehr durch den Zertifikatsaustausch und dem verpflichtenden Handshake durch die TLS-Kanal Erstellung. Da sich solche Nachrichten meist nur aus wenigen hundert Bytes zusammensetzen, nimmt die Belastung des Verkehrs durch das Hinzufügen von sicheren Übertragungen nur unwesentlich zu. Für den Fall mit der 10 MB Dateiübertragung durch den Datenpaket-Service werden zum einen 12,333

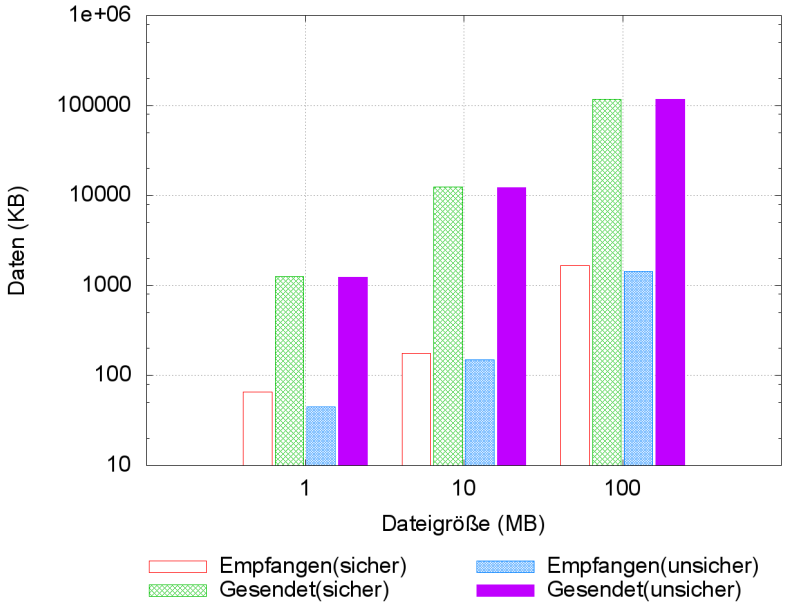

Abbildung 6.19.: *Netzauslastung durch einen Service.*

MB und zum anderen 12,295 MB für den sicheren und für den unsicheren Service empfangen. Der Grund hierfür ist wieder derselbe wie zuvor. Wenn jedoch von der ursprünglichen Größe der Datei (10 MB) ausgegangen wird, erscheinen 2,3 MB Overhead viel. Dieser wird durch die Fragmentierung bei großen Datenpaketen und der hierfür notwendigen zusätzlichen Daten verursacht. Dieser Vorgang wird bei der 100 MB Datei sehr deutlich. Hier werden für die sichere Verbindung 117,436 MB und für die unsichere Verbindung 117,143 MB gemessen. In diesem Fall bedeutet das, dass der Overhead nahezu 7,5 MB für den sicheren Service und 7,1 MB für den unsicheren Service beträgt. In Tabelle 6.7 wird der prozentuale Zuwachs für die unterschiedlichen Dateigrößen bei sicherer und unsicherer Übertragung gezeigt.

Dateigröße	unsicher	sicher
1 MB	23,7 %	25 %
10 MB	22,95 %	23,33 %
100 MB	17,143 %	17,436 %

Tabelle 6.7.: *Relative Betrachtung des Overheads bei der Datenübertragung.*

Wie zu sehen ist, hat die Absicherung der Kommunikation nur einen geringen Einfluss auf die tatsächlich übertragenen Daten. Bei der näheren Betrachtung des Verkehrs ist zu sehen, dass auch ein kleiner Anteil jeder Übertragung für Bestätigungsantworten durch das Transportprotokoll und durch die Nachrichten der Zeitsynchronisierungsprotokolle eingenommen werden.
Zusammengefasst bedeutet dies, dass die Anpassungen keinen signifikanten Einfluss auf die Belastung des Netzes zwischen Client und Service ausübt, da sich der zusätzliche Verkehr sich aus Zertifikats- und kryptografischen Nachrichtenaustausch zusammensetzt, die eher schmalgewichtige Nutzdaten darstellen. Der darauf folgende Verkehr umfasst dieselbe

Menge, ob er nun verschlüsselt für einen sicheren Service oder im Klartext für einen unsicheren Service auftritt. Ähnlich verhält es sich mit dem Verkehr zwischen dem Service und dem AAA-Server. Dieser setzt sich aus LDAP-basierten Anfragen zusammen, deren Aufbau selten mehr als etwa 100 Bytes besitzt und so nahezu bedeutungslos für die Netzressourcen ist.

6.3.4. Auswertung der quantitativen Evaluierung

In diesem Abschnitt werden die Ergebnisse aus den Messungen zusammengefasst und ausgewertet. Bei der Performanzbetrachtung der grafischen Benutzerschnittstelle wird deutlich, dass einige Aspekte von der verwendeten Renderengine bzw. vom Webbrowser abhängen. So kann eine unterschiedliche Abarbeitung der Benachrichtigungen durch die JavaScript-Umgebung beobachtet werden. Je nach Implementierung ergeben sich hier Effekte wie z.B. längere Verzögerungen. Um dem zu entgehen sollte die Eigenentwicklung des GUI-Renderers dementsprechend angepasst werden oder Konfigurationsvorschläge von Parametern für die jeweiligen Standardbrowser veröffentlicht werden. Bei den Parametern für den Webbrowser ergeben sich ebenfalls Unterschiede zwischen den Realisierungen. Es gibt in Standardwebbrowsern eine Beschränkung der persistenten Verbindungen von Webserver zu Webbrowser. Bei Mozilla Firefox ist diese beispielsweise auf sechs Stück eingestellt. Dies kann aber über ein Konfigurationsmenü geändert werden. Da diese persistenten Verbindungen den abonnierten Kanälen entsprechen, muss für das MMS-System der Webbrowser umkonfiguriert werden oder die Anzahl der Benachrichtigungskanäle für eine GUI eingeschränkt werden. Auch der nahezu beliebig komplexe GUI-Inhalt lässt nur schwer eine Aussage über die Prozessorlast am GUI-Renderer zu. Hingegen muss aber dieser Inhalt nur initial geladen werden. Ein Vergleich der dafür benötigten Zeit wird in Tabelle 6.8 gezeigt. Somit ist das einzige Kriterium hinsichtlich der Inbetriebnahme des MMS-System, dass die Aufstartzeit des GUI-Renderers (~11 s) zusammen mit der Ladezeit der GUI (~320 ms) innerhalb der Zeit bleibt, die für den Nutzer als akzeptabel angenommen werden kann (im automotiven Bereich <15 Sekunden).

	Darstellungszeit (mit Cache)	Darstellungszeit (ohne Cache)
Simple	319 ms	470 ms
BMW	393 ms	577 ms
Anwendung	–	82 ms

Tabelle 6.8.: *Darstellungszeiten des GUI-Inhalts für zwei erstellte GUIs und eine Anwendung.*

Neben dem GUI-Renderer ist auch das Aufstartverhalten des GUI-Kombinierers entscheidend. Für diesen sind mindestens 2,6 Sekunden zusammen mit dem Hochfahren seines Betriebssystems mit etwa 8 Sekunden nötig. Zu dieser Zeit kommt noch die von der Anzahl der Geräte abhängige linear ansteigende Suchdauer nach den GUI-Geräten im Netz dazu. Da der Anstieg der Dauer etwa 0,04 Sekunden pro 10 Geräten ausmacht, der Suchvorgang an sich 2,62 Sekunden benötigt und man ein mittelgroßes MMS-System mit etwa 30-50 GUI-Geräten annimmt, beläuft sich diese Zeit auf etwa 2,73 Sekunden. Somit wäre die Gesamtzeit für das Initialisieren des GUI-Kombinierers bei etwa 13 Sekunden. Auch wenn diese Zeiten für den automotiven Bereich zu lang sein sollten, existieren Ruhestandtechniken, mit denen

diese weiter verkürzt werden können.
Hinsichtlich der Prozessorauslastung und dem Speicherverbrauch wird deutlich, dass ein Anstieg der GUI-Geräte im Netz lediglich zu einem geringen Anstieg der Nutzung dieser Ressourcen führt. So wird durch die Grundfunktionalitäten des MMS-Systems der GUI-Kombinierer nicht stark belastet. Sollten die Ressourcen ausgelastet werden, kann das MMS-System und seine Komponenten auf stärkerer Hardware betrieben werden, oder durch aus dem Bereich der Webtechnologien bekannte Lastverteilungsmechanismen wie Load Balancing und Content Distribution Networks gegen Überlast geschützt werden.

Eine Zusammenfassung der Performanzbetrachtungen für das Servicemanagement wird nachfolgend gegeben. Es werden die Prozessorauslastung, die Speichernutzung und Übertragungsverzögerung der unterstützten Chiffren im Zusammenhang mit den verwendeten Dateigrößen von 1 MB bis 100 MB zusammengefasst und verglichen.
Die von den zuvorigen Diagrammen (vergleiche Abbildung 6.10 - 6.15) gesammelten Werte zeigen, dass die Algorithmen DES, AES128 und AES256 das beste Verhalten bezüglich der Prozessorauslastung zeigen. DES schneidet jedoch bei der Betrachtung der Speichernutzung schlecht ab. Diese Chiffre benötigt mehr Speicher als die anderen Algorithmen und das sowohl bei kleineren als auch bei größeren Dateiübertragungen. Dies ist bedingt durch den blockbasierten Ansatz dieses Algorithmus. Im Bereich der eingebetteten Systeme mit begrenzten Ressourcen muss dies als Nachteil angesehen werden. Aus diesen Betrachtungen folgt, dass AES128 bezüglich der Prozessorauslastung und Speichernutzung die beste Performanz erzielt.
Wird die Übertragungsverzögerung betrachtet (Tabelle 6.9), wird schnell klar, dass sich die Chiffren AES128 und AES256 erneut als vorteilhaft gegenüber den anderen hervor heben.

Mechanismus	Dateigröße (MB)		
	1	10	100
DES	0,18 s	1,23 s	10,81 s
RC4	0,21 s	1,20 s	10,72 s
AES128	0,17 s	1,15 s	10,62 s
3DES	0,28 s	1,36 s	10,38 s
AES256	0,14	1,17	10,66
ohne	0,13	1,11	10,48

Tabelle 6.9.: *Vergleich der Übertragungslatenz der verschiedenen Verschlüsselungsalgorithmen.*

Wie zu erwarten war, ist durch alle Chiffren ein leichter Anstieg der Verzögerung im Vergleich zu der unsicheren Messung zu sehen. Natürlich hängt diese verhältnismäßig von den Prozessor- und den Speicherspezifikationen der zur Evaluierung verwendeten Hardware ab.
Die Tabelle 6.10 fasst die Netzauslastung auf der Seite des Services zusammen und zeigt den Unterschied zwischen der sicheren und unsicheren Serviceeinbindung bezüglich der gesendeten Daten.

Es ist zu sehen wie ein Anstieg tatsächlich übertragener Daten gegenüber der Ausgangsdatengröße entsteht. Dabei handelt es sich um zusätzliche Informationen, die für die Fragmentierung der Daten relevant sind. Soll beispielsweise eine 1 MB große Datei versendet werden, müssen dazu 1,237 MB für die unsichere bzw. 1,251 MB für eine gesicherte Übertra-

Dateigröße (MB)	Gesendete Daten (MB) sicher	Gesendete Daten (MB) unsicher
1	1,251	1,237
10	12,333	12,295
100	117,496	117,143

Tabelle 6.10.: *Vergleich der gesendeten Daten mit verschiedenen Dateigrößen.*

gung aufgewendet werden. Tabelle 6.11 zeigt den gleichen Vergleich für die empfangenen Daten.

Dateigröße (MB)	Empfangene Daten (MB) sicher	Empfangene Daten (MB) unsicher
1	0,066	0,045
10	0,176	0,150
100	1,655	1,436

Tabelle 6.11.: *Vergleich der empfangenen Daten (aus Sendersicht) mit verschiedenen Dateigrößen.*

Wie zu sehen ist, sind die Daten, die vom Sender aufgrund seiner Übertragung empfangen werden, sehr gering. Es handelt sich dabei hauptsächlich um Bestätigungen der Transportprotokolle und TLS Übertragungsinformationen. Grundsätzlich gilt: je mehr Daten gesendet werden, desto mehr Bestätigungen können erwartet werden.

Zusammenfassend kann gesagt werden, dass die Netzauslastung zwischen den verschiedenen Chiffren nahezu konstant ist. Des Weiteren ist die Verzögerung beim Aufstarten ebenfalls zwischen diesen konstant. Deswegen können die Kriterien für die Auswahl der geeigneten Chiffren auf die Prozessorauslastung, Speichernutzung und Übertragungsverzögerung beschränkt werden. Bezüglich dieser ist es empfehlenswert, AES128 und AES256 als Chiffren für die Verwendung des sicheren Kanals zu verwenden. Dies gilt besonders, wenn eingebettete Geräte mit begrenzten Hardwareressourcen verwendet werden, da die Wahl von Chiffren wie RC4 oder 3DES die Prozessorleistung durch Services mit großen Datenströmen beeinträchtigen können. Gleichermaßen kann die Bevorzugung des DES Algorithmus zu einem Speicherüberlauf bei Services mit einem hohen Speicherkonsum führen. Die beiden Favoriten AES128 und AES256 werden mitunter als die stärksten Chiffren hinsichtlich der Sicherheit angesehen.

6.4. Qualitative Evaluierung

Die qualitative Betrachtung in diesem Abschnitt bewertet zum einen die in Kapitel 3.2 ausgearbeiteten Anforderungen an ein zukünftiges MMS-System. Zum anderen werden die vertieften Sicherheitsaspekte analysiert. Da sich die AAA-Richtlinien aus Anhang A.6 besonders dafür eignen, werden diese ebenfalls für die Evaluierung herangezogen. Auf diese Weise kann eine Einschätzung der Qualität der vorgeschlagenen Gesamtlösung gegeben werden. Für die Evaluierung wird die in Tabelle 6.12 aufgeführte Skala verwendet.

Bewertung	Beschreibung
++	Vollkommen abgedeckt
+	Teilweise abgedeckt
–	Mit Aufwand abdeckbar
––	Nicht abgedeckt
0	Neutral/Außerhalb der Betrachtung

Tabelle 6.12.: *Bewertungsskala für die qualitative Evaluierung.*

6.4.1. Abdeckung der Anforderungen einer zukünftigen MMS

Nachfolgend wird jede Anforderung nach der Kategorisierung aus Kapitel 3.2 getrennt in Abstraktion, Erweiterbarkeit, Zentralisierung und Homogenisierung bewertet.

Abstraktion

Eingabe- und Ausgabekanal: Das vorgeschlagene MMS-System unterstützt jegliche Art von Ausgabegeräten. Neben den gesondert betrachteten visuellen Ausgabegeräten können z.B. auch Audiosysteme integriert werden. Hierbei muss lediglich der Nutzdatentransport geregelt werden. Die Ansteuerung, Einbindung etc. wird durch das DPWS-Konzept abgedeckt. Des Weiteren kann das DPWS-Konzept auch für Eingabegeräte angewendet werden. Auf diese Weise können ereignisbasierte Eingaben durch Nutzer in dem verteilten System detektiert und an ihren Bestimmungsort geleitet werden. Somit gilt diese Anforderung als vollständig abgedeckt und wird mit ++ bewertet.

Grafische Repräsentation: In diesem MMS-System wird großen Wert auf die grafische Ausgabe gelegt. So werden Webtechnologien verwendet, um damit eine geeignete GUI zu erstellen und dem Nutzer anzubieten. Diese Repräsentation abstrahiert die Darstellung von der Logik. Dabei wird die Darstellung über eine HTML Engine bewerkstelligt. Mit dieser Realisierung ist es möglich funktionstüchtige GUIs anzubieten. Diese Anforderung wird damit als vollkommen abgedeckt angesehen (++).

Interaktionsmöglichkeit für jeden Nutzer: Durch die Verwendung von Standardtechnologien, die bereits für die Interaktion mit dem Nutzer ausgelegt sind, wird diese Anforderung vollkommen abgedeckt (++).

Personalisierung: Hierbei hilft die Abstraktionsschicht von Webtechniken. Durch die Verwendung von Webtechnologien für die grafische Ausgabe, kann diese beliebig angepasst werden. Ähnlich wie im Internet die angebotenen Webseiten an eigene Bedürfnisse angepasst werden können, wird das Aussehen der GUI-Inhalte von Designbeschreibungen festgelegt. Auch für einzelne Nutzer können diese Beschreibungen integriert werden. Das Sys-

tem lässt dies zwar zu, diese Funktion wurde aber nicht realisiert. Deswegen ist die Personalisierung teilweise abgedeckt (+).

Anpassbares Design: Die Grundlage für die Personalisierung wird unterstützt, siehe vorige Anforderung. So ist es möglich ein globales Design anzugeben und durch verschiedene Anwendungsdesigns zu erweitern. Somit werden verschiedene Designs der GUI vollkommen unterstützt (++).

Lose Zuordnung von Ein- und Ausgabekanal: Der serviceorientierte Ansatz unterstützt die lose Zuordnung von Ein- und Ausgabegeräten. So können diese beliebig gekoppelt werden und sogar mehrere zu einem verbunden werden. Diese Kopplung unterliegt keinerlei Einschränkung, deswegen wird diese Anforderung mit ++ bewertet.

Offene/standardisierte Schnittstellen: Diese Anforderung wird vollkommen durch das DPWS-Konzept abgedeckt (++). Die offenen und standardisierten Schnittstellen sind ein Grundkonzept von Web Services und ermöglichen die Flexibilität und Erweiterbarkeit von Web Services.

Auflösung von Abhängigkeiten: Die Einführung des Discovery Proxys ermöglicht die vollkommene Abdeckung (++) der Abhängigkeitsauflösung. Dabei kümmert sich jedes Gerät selbst darum, die von ihm benötigten Funktionalitäten zu finden. Sind diese Abhängigkeiten erfüllt, meldet das Gerät dies dem Discovery Proxy und der schaltet das Gerät für die Auffindung frei.

Übersicht:

Nahezu alle Anforderungen aus dem Bereich der Abstraktion werden in Tabelle 6.13 vollkommen abgedeckt. Bei der Anforderung „Personalisierung“ kommt es zu Einschränkungen. Die Mechanismen für die Erfüllung dieser Anforderung werden von dem MMS-System unterstützt, aber es wird kein Benutzermanagement für die jeweils eigenen Modifikationen des Designs realisiert. Das globale Design kann natürlich nach eigenen Präferenzen angepasst werden, aber es müssten zusätzlich Mechanismen implementiert werden um mehrere Designs parallel zu verwalten.

Erweiterbarkeit

Mehrnutzerbetrieb: Webtechniken und serviceorientierte Architekturen sind geeignet für die Verwendung im Mehrbenutzerbetrieb, sie bieten jedoch keine eigenen Mechanismen für die Organisation von mehreren Benutzern. Aus diesem Grund wird ein Sessionmanagement eingeführt mit dem dies geregelt wird. Die Abdeckung wird mit ++ bewertet.

Bereich	Anforderung	Bewertung
Abstraktion	Eingabe- und Ausgabekanal	++
	Grafische Repräsentation	++
	Interaktionsmöglichkeit für jeden Nutzer	++
	Personalisierung	+
	Anpassbares Design	++
	Lose Zuordnung von Ein- und Ausgabekanal	++
	Offene/standardisierte Schnittstellen	++
	Auflösung von Abhängigkeiten	++

Tabelle 6.13.: *Bewertung der Abstraktionsanforderungen der MMS.*

Verschiedene Arten von Eingabekanälen: Jede Funktionalität kann durch Services realisiert werden. Zu jedem Service gibt es eine Servicebeschreibung. Aufgrund dieser Beschreibung kann mit jeder Art eines Eingabegerätes interagiert werden. Somit werden alle Arten von Eingabegeräten unterstützt (++).

Integration von fremden Geräten: Das gesamte MMS-System, das bedeutet die grafische Schnittstelle und das Servicemanagement, hält Mechanismen bereit, die in Kombination mit den servicebasierten Erweiterungen des Gerätes die Integration von unbekannten Geräten ermöglichen. Durch die uneingeschränkte Integrationsfähigkeit wird diese Anforderung als vollkommen abgedeckt angesehen (++).

Verschiedene Arten der Ausgabekanäle: Die MMS-Architektur unterstützt alle Arten von Ausgabekanälen. Jeder Kanal muss für die Verwendung von Entwicklern beschrieben werden und diese Beschreibung muss den anderen Teilnehmern zur Verfügung gestellt werden. Beispielsweise kann ein Lautsprechergerät angeboten werden, das über einen Befehl Audiodaten ausgeben kann. Diese werden In-Band oder Out-Band an das Gerät übertragen und dort verarbeitet. Die grafische Ausgabe wird detailliert betrachtet. Es ergeben sich hierbei Besonderheiten hinsichtlich der Verschmelzung des Ein- und Ausgabekanals, die gesondert bearbeitet werden. Diese vollkommene Abdeckung wird mit ++ bewertet.

Hinzufügen eines Ein- oder Ausgabekanals: Die standardisierten Schnittstellen und der serviceorientierte Ansatz ermöglichen das einfache und schnelle Hinzufügen von Ein- oder Ausgabegeräten. Ein neu zu integrierender Kanal muss die DPWS-basierten MMS-Systemschnittstellen unterstützen. Somit unterstützt das MMS-System das Hinzufügen von neuen Kanälen vollkommen (++).

Unterstützung verschiedener grafischer Ausgabeformate: Durch die Verwendung von Webbrowsertechniken für die GUI werden auch verschiedene Formate und Auflösungen der Displays unterstützt. Bei realisierten Webseiten im Internet ist oft zu beobachten wie

sich diese an die verschiedenen Displaygrößen anpassen. Das MMS-System stellt alle Funktionalitäten zur Verfügung, um dies zu realisieren, aber ein GUI-Inhalt der sich auf verschiedene Formate anpasst wurde nicht erstellt. Aus diesem Grund wird diese Anforderung mit – bewertet.

Dynamische Nachrüstbarkeit: Neue Geräte für die grafischen Schnittstellen, wie auch zur Bereitstellung von Funktionalitäten können zur Laufzeit des MMS-Systems nachgerüstet werden. Aus diesem Grund wird für diesen Aspekt + + vergeben.

Anpassungen an den Ressourcenverbrauch der Funktionen: Aufgrund des servicebasierten Ansatzes sind die DPWS-Geräte selbst für die Bereitstellung der Ressourcen für ihre Funktionalitäten verantwortlich. Es müssen aber Ressourcen von dem MMS-System vorgehalten werden, um Anfragen an die Funktionalitäten zu stellen und zu verwalten. Die Konzepte der grafischen Schnittstelle und des Servicemanagements unterstützen dies. Hingegen wird dieser Aspekt nicht explizit betrachtet und deswegen mit – bewertet.

Erweiterbarkeit der grafischen Repräsentation: Wird durch das Servicemanagement eine Funktionalität nachgerüstet, die eine Anwendungsoberfläche mit sich bringt, kann diese in die GUI integriert werden. Dies wird durch das erweiterbare Konzept der grafischen Schnittstelle vollkommen unterstützt (+ +).

Übersicht:

Im Bereich der Erweiterbarkeit werden alle Anforderungen außer „Unterstützung verschiedener Ausgabeformate“ und „Anpassungen an den Ressourcenverbrauch der Funktionen“ vollkommen abgedeckt (siehe Tabelle 6.14). Beide genannten Aspekte können mit Aufwand erfüllt werden. Der erste Aspekt mindert die universelle Abdeckung von allen verfügbaren Displayformaten. Deswegen müssen eventuell händisch Anpassungen an dem GUI-Inhalt vorgenommen werden. Außerdem können sich auch suboptimale grafische Darstellungen auf manchen Displays ergeben. Der zweite Aspekt wird mit einer geringen Anzahl an Funktionen von den verwendeten Techniken abgedeckt. Allerdings können sich bei einer sehr hohen Anzahl von Funktionen Probleme ergeben. Deswegen muss je nach Anwendungszweck auf diesen Aspekt geachtet werden.

Zentralisierung

Bündelung verschiedener Funktionalitäten: Durch den Einsatz einer serviceorientierten Architektur für das Servicemanagement lassen sich leicht Funktionalitäten in Form von Services bündeln. Dies ist einer der Grundgedanken von SOAs. Dieser Aspekt gilt als vollkommen abgedeckt (+ +).

Bereich	Anforderung	Bewertung
Erweiterbarkeit	Mehrnutzerbetrieb	+ +
	Verschiedene Arten von Eingabekanälen	+ +
	Integration von fremden Geräten	+ +
	Verschiedene Formate der Ausgabekanäle	+ +
	Hinzufügen eines Ein- oder Ausgabekanals	+ +
	Unterstützung verschiedener Ausgabeformate	–
	Dynamische Nachrüstbarkeit	+ +
	Anpassungen an den Ressourcenverbrauch der Funktionen	–
	Erweiterbarkeit der grafischen Repräsentation	+ +

Tabelle 6.14.: *Bewertung der Erweiterbarkeitsanforderungen der MMS.*

Zugangsmanagement für Nutzer: Der Zugang für Nutzer wird durch Authentifizierungsmethoden geregelt. Die dafür nötigen Informationen sind der Nutzername und Passwort. Auf diese Weise kann der Zugang für Benutzer über Geräte als auch über die grafische Ausgabe geregelt werden. Detaillierte Bewertungen folgen in Kapitel 6.4.2. Somit wird dieser Punkt mit + + bewertet.

Anpassungen an den Ressourcenverbrauch neuer Geräte: Dieser Aspekt wird in der Arbeit nicht betrachtet. Es handelt sich hier um Betrachtungen, die sich mehr auf die Hardware des Gerätes und das Kommunikationsnetz konzentrieren. Deswegen liegt dieser Punkt außerhalb der Betrachtung (0).

Zugangsmanagement für Geräte: Das Management für den Zugang basiert auf Zertifikaten. Jedem Gerät mit einem gültigen Zertifikat wird der Zugang gewährt. Erläuterungen werden in Kapitel 5.3 gegeben. Das MMS-System deckt diesen Punkt vollkommen ab (+ +).

Vertraulichkeit: Dieser Anforderung wird durch die Verwendung von kryptografischen Funktionen begegnet. Dabei handelt es sich um öffentliche und private Schlüssel um die Ver- und Entschlüsselung vorzunehmen. Detaillierte Beschreibungen zu diesem Punkt werden in Kapitel 5.3 gegeben. Vertraulichkeit wird unterstützt und mit + + bewertet.

Zugangsmanagement für Funktionen: Durch die servicebasierte Authentifizierung wird mit Hilfe des Zertifikats und der Zugriffsregeln der Zugang zu Funktionen geregelt. Diese Anforderung wird in Kapitel 5.3 erläutert und als vollkommen abgedeckt betrachtet (+ +).

Übersicht:

Die einzige Anforderung im Bereich der Zentralisierung, die nicht abgedeckt wird, ist „Anpassungen an den Ressourcenverbrauch neuer Geräte“ (siehe Tabelle 6.15). Diese Anforderung bezieht sich auf Schichten des OSI-Modells, die nicht im Fokus dieser Arbeit liegen.

Bereich	Anforderung	Bewertung
Zentralisierung	Bündelung verschiedener Funktionalitäten	++
	Zugangsmanagement für Nutzer	++
	Anpassungen an den Ressourcenverbrauch neuer Geräte	0
	Zugangsmanagement für Geräte	++
	Verschlüsselung	++
	Zugangsmanagement für Funktionen	++

Tabelle 6.15.: *Bewertung der Zentralisierungsanforderungen der MMS.*

Homogenisierung

Minimalisierte Datenübertragung: Ein großer Vorteil der Verwendung von Webtechniken ist die Übertragung von Grafikbefehlen, die nach dem Empfang gerendert werden. Mit anderen Worten: Es werden keine gerenderten Bilder übertragen, sondern Grafikbefehle, die erst am Display interpretiert werden. Auch die Trennung von Aussehen und Logik leistet einen Beitrag dazu, die Datenübertragung zu mindern. Da sich Nutzdaten in der GUI öfter als das Aussehen ändern und dynamische Abläufe am Aussehen an jedem Display lokal durchgeführt werden, ist eine hohe Einsparung möglich und dieser Punkt gilt daher als vollkommen erfüllt (++).

Zugriff auf fahrzeuginterne Funktionalitäten/Daten: Der Zugriff auf fahrzeuginterne Funktionalitäten ist mit der vorgeschlagenen MMS-Architektur möglich. Jedoch ergeben sich Einschränkungen bezüglich der Realzeitfähigkeiten (siehe Kapitel 6.3.4). Diese Einschränkungen können mit Aufwand aufgehoben werden (–).

Selbstkonfiguration: Der serviceorientierte Ansatz ermöglicht von Seiten der Funktionalitäten das selbstständige Suchen von Services und Anbieten eigener Services. Dadurch kann sich jedes Gerät selbst konfigurieren. Die grafische Schnittstelle basiert ebenfalls auf Konzepten mit denen es möglich ist einen minimal ausgestatteten GUI-Renderer (Webbrowser) zu verwenden um die GUI darzustellen. Dabei werden Logik und Aussehen nachgeladen und dynamisch konfiguriert. Die vollständige Abdeckung der Selbstkonfiguration durch das MMS-System wird mit ++ bewertet.

Intelligentes Display: Das Minimalsystem ermöglicht eine Plattform anzubieten, die wenig Hardwareressourcen benötigt aber auch genug Intelligenz für den GUI-Renderer zur grafischen Ausgabe bietet. Gerade für eingebettete Systeme wie im automotiven Bereich ist dieser Zusammenhang wichtig. Die Anforderung wird durch die Entwicklung eines Minimalssystems in Kombination mit der vorgestellten GUI-Lösung prototypisch nachgewiesen und gilt als vollkommen abgedeckt (++).

Plattformunabhängigkeit: Diese Anforderung wird durch die Verwendung der HTML Engine vollkommen abgedeckt. Zum einen wird eine Unabhängigkeit bei der grafischen Schnittstelle durch die Homogenisierung von Protokollen für die Darstellung erreicht. Zum anderen lässt sich auch in dem Servicemanagement eine Unabhängigkeit durch die Definierung des vereinheitlichenden SOA-Konzepts finden. So wird durch die große Verbreitung von Webbrowsern, die für die Realisierung des GUI-Renderers herangezogen werden können, auf vielen Betriebssystemen profitiert. Ebenso ist die Verwendung eines Standards bei der Realisierung der servicebasierten Funktionalitäten von Vorteil. Somit gilt diese Anforderung als vollkommen abgedeckt (+ +).

Allg. Beschreibung von Schnittstellen: Aufgrund der Verwendung eines serviceorientierten Ansatzes, werden allgemeine Schnittstellenbeschreibungen für die Services gefordert. Ohne diese sind Aspekte wie die Nutzung von unbekannten Service nicht möglich. Diese Anforderung wird mit + + bewertet.

Bereitstellung von Funktionen für Nutzer: Jedem Gerät wird die Bereitstellung von Funktionen ermöglicht. Hierfür muss es seine Funktionalitäten in einer Beschreibung festhalten und diese bei Bedarf anfragenden Clients zur Verfügung stellen. Die darauffolgende Nutzung kann durch die Sicherheitserweiterungen reglementiert werden. Damit ist dieser Aspekt vollkommen abgedeckt (+ +).

Übersicht:

Aus Tabelle 6.16 geht hervor, dass bis auf einen alle Aspekte in dem Bereich der Homogenisierung abgedeckt werden. Der Punkt „Zugriff auf fahrzeuginterne Funktionalitäten/Daten“ kann nur mit Aufwand erfüllt werden. Da fahrzeuginterne Funktionalitäten oftmals einen Realzeitanspruch besitzen, muss das MMS-System erweitert werden, damit dieses auch einen Realzeitnachweis erbringen kann. Beispiele für zeitkritische und sicherheitsrelevante GUI-Funktionen sind z.B. die Anzeige von Warnhinweisen wie Fußgängererkennung, witterungsbedingte Straßenwarnungen oder Kollisionswarnungen. Diese Erweiterungen schränken mit großer Wahrscheinlichkeit die dynamische Erweiterbarkeit von Funktionalitäten ein. Deswegen muss über Mechanismen für die Einführung von Quality-of-Service-Diensten nachgedacht werden.

6.4.2. Abdeckung der Anforderungen der AAA-Richtlinien

Allgemeine Anforderungen

Skalierbarkeit: Diese Anforderung liegt außerhalb dieser Betrachtung (0), da die Unterstützung von tausenden Nutzern, Anfragen oder Geräten nicht den primären Aspekt dieser Architektur darstellt. Da diese Architektur vermehrt aus eingebetteten und mobilen Geräten

Bereich	Anforderung	Bewertung
Homogenisierung	Minimalisierte Datenübertragung	++
	Zugriff auf fahrzeuginterne Funktionalitäten/Daten	–
	Selbstkonfiguration	++
	Intelligentes Display	++
	Plattformunabhängigkeit	++
	Allg. Beschreibung von Schnittstellen	++
	Bereitstellung von Funktionen für Nutzer	++

Tabelle 6.16.: *Bewertung der Homogenisierungsanforderungen der MMS.*

aufgebaut ist und diese aufgrund ihrer limitierten Ressourcen nicht allzu viele Anfragen bearbeiten können, wird die Skalierbarkeit bereits durch die Hardware eingeschränkt. Trotzdem ist es theoretisch möglich, die limitierenden Hardwarekomponenten der Architektur, wie z.B. den Webserver oder AAA-Server mit mehr Ressourcen auszustatten und damit die Architektur zu skalieren.

Ausfallsicherheit: Diese Anforderung liegt ebenfalls außerhalb der betrachteten Aspekte der MMS-Architektur. Dies begründet sich damit, dass nicht nur die softwarebasierte Entwicklung von diesem Thema betroffen ist, sondern auch die Hardwaregeräte, die aktiv für die Serviceverwendung zuständig sind. Die zugrundeliegenden Sicherheitskonzepte erfordern eine zentrale Auslegung des AAA-Servers und der CA. Um trotzdem eine Ausfallsicherheit zu gewährleisten, ließen sich jedoch bereits vorhandene Lösungen für Ausfalltechniken (hardware- wie softwarebasiert) anpassen. Da dieser Aufwand möglich wäre, wird dieser Aspekt mit – bewertet.

Gegenseitige Authentifizierung: Wird die vorgeschlagene Architektur betrachtet, fallen zwei Arten der gegenseitigen Authentifizierung auf. Die erste wird zwischen dem Client und dem Service auf Geräteebene durchgeführt. Dabei werden die X.509v3-Zertifikate zwischen den kommunizierenden Teilnehmern ausgetauscht, um die Gültigkeit mit Hilfe der CA zu validieren. Somit können sich Client und Service über die Identität des Gegenübers sicher sein. Die zweite gegenseitige Authentifizierung wird zwischen dem Service und dem AAA-Server angewendet, um die Verifikation der Identität dieser beiden Komponenten in beide Richtungen zu gewährleisten. Dies ist nötig um sensiblen Verkehr wie Passwörter und Verifizierungen bei sicherheitsrelevanten Anfragen zu schützen. Im Hinblick auf diese Aspekte kann diese Anforderung als vollkommen abgedeckt (++) angesehen werden.

Übertragungssicherheit in der Transportschicht: Sicherheitslösungen, die auf TLS/SSL Verbindungen aufbauen, werden auch als Transportschicht-Verschlüsselungen bezeichnet. Es werden hierbei implizit die Sicherheitsmechanismen Authentifizierung, Vertraulichkeit und Integrität für die unteren Netzschichten betrachtet. Die TLS/SSL Verbindung ist socketbasiert und bietet für die Netz- und Transportschicht eine Verschlüsselung der Daten an, wodurch alle Nachrichten, die über diese Schichten gesendet werden, verschlüsselt sind um eine mögliche Umleitung des Verkehrs oder eine Klartext-Interpretation eines Dritten zu

verhindern. Durch die Verwendung dieser Sicherheitslösung wird diese Richtlinie als vollkommen abgedeckt angegeben (+ +).

Vertraulichkeit der Informationen: In Anbetracht der Beschreibungen in Kapitel 2.3 kann geschlussfolgert werden, dass diese Anforderung erfüllt ist. Da die realisierte Transportschicht-Sicherheit auf sicheren Verbindungen beruht, wird die Datenvertraulichkeit ebenfalls unterstützt. Das vorgestellte MMS-System verwendet öffentliche und private RSA-Schlüssel mit drei verschiedenen Längen, 512, 1024 und 2048 Bits (siehe Kapitel 5.3) um den Schlüssel der symmetrischen Verschlüsselung auszutauschen. Über den damit aufgebauten sicheren HTTPS-Kanal können anschließend verschlüsselte SOAP-Nachrichten versendet werden. Beispielsweise erlaubt die verwendete PKI den Schutz der Nachrichten gegenüber sogenannten Man-in-the-middle Angriffen. Des Weiteren werden solche Schlüssel und X.509v3-Zertifikate innerhalb der Geräte (auf dem Speichermedium) durch eine Verschlüsselung mit den AES256-SHA Chiffre geschützt. Dementsprechend gilt diese Richtlinie als vollkommen abgedeckt (+ +).

Integrität der Informationen: Erneut bietet der TLS/SSL Kanal hier eine zufriedenstellende Lösung. Der Signierungsvorgang der Nachrichten wird durch den privaten Schlüssel des Gerätes und mit einem Hash-Wert der Nachricht mit einer festen Länge durchgeführt. Diese Signatur wird auf der Seite des Gegenübers nochmal berechnet und mit dem betreffenden öffentlichen Schlüssel des Senders geprüft. Abhängig von dem Ergebnis dieser Berechnung kann eine Integrität der Nachricht angenommen werden oder, falls Informationen verändert wurden, ein Verwerfen dieser durchgeführt werden. In dem MMS-System dieser Arbeit werden technisch ausgereifte Algorithmen für die Herleitung der Hash-Werte von Nachrichten wie z.B. SHA-1 oder SHA-256 herangezogen. In diesem Zusammenhang kann von einer vollkommenen Abdeckung dieser Anforderung gesprochen werden (+ +).

Zertifikatsaustausch: Diese Anforderung wird durch die vollständige Abdeckung der vorigen Anforderungen der gegenseitigen Authentifizierung und der Übertragungssicherheit in der Transportschicht auch vollständig mitabgedeckt. Das Netz muss es ermöglichen ein X.509v3-Zertifikat eines Clients oder Services zu dem jeweils anderen zu übertragen. Dieser Zertifikatsaustausch ist essentiell für die Erstellung eines TLS/SSL-Kanals als auch für die eindeutige Identifizierung aller Kommunikationspartner. Durch die Verwendung von dafür vorgesehenen OpenSSL-Funktionen wird diese Anforderung als vollkommen abgedeckt markiert (+ +).

Zuverlässige Übertragung: In diesem Zusammenhang bezieht sich der Begriff Zuverlässigkeit auf Eigenschaften der Transportschicht. Das beinhaltet Schutzmechanismen gegen Paketverlust und ähnliches wie die erneute Übertragung eines Pakets, Empfangsbenachrichtigungen, Huckepack Informationen, Fehlerkontrollen, Flusskontrollen und zeitabhängige Antworten. Innerhalb des vorgeschlagenen MMS-System werden die durch das Transmission Control Protocol (TCP) verursachten Bestätigungspakete für SOAP-Nachrichten unterstützt, um einen zuverlässigen Transport zu ermöglichen. Der tiefere Einblick in diese Tech-

nik liegt außerhalb der Betrachtungen dieser Arbeit. Allerdings bieten vorhandene und verwendete Techniken hinsichtlich der zuverlässigen Übertragung bereits Mechanismen, wie Überlaststeuerung oder Staukontrolle und deshalb wird dieser Aspekt als vollkommen abgedeckt bewertet (+ +).

Unterstützung von IPv4: Die eigentliche Adressierung der Netzschicht innerhalb der MMS-Architektur wird an die Realisierung des DPWS übergeben. Dessen WS-Addressing Mechanismen kümmern sich um die Kommunikation zwischen den Services und Endpunkten. Die Kompatibilität mit IPv4 und IPv6 wird vollständig durch die Realisierung unterstützt. Zudem unterstützen die Erweiterung der DPWS-Sicherheit und die Schnittstellen von OpenSSL beide Adressierungsversionen für abgesicherte Services und TLS/SSL. Aufgrund dieser Betrachtung kann diese Richtlinie mit + + bewertet werden.

Unterstützung von IPv6: Die Unterstützung von IPv6 ist analog zu IPv4 und wird im vorigen Absatz bereits betrachtet (+ +).

Unterstützung von Kommunikationsschnittstellen: Die CA und der AAA-Server können als Netzübergangsknoten angesehen werden, die sich eventuell in einer anderen Netzdomäne neben dem MMS-System befinden. Zum einen besitzen diese jedoch nicht die Fähigkeit Anfragen zu vermitteln oder weiterzuleiten. Sie dienen lediglich zur Bearbeitung von Anfragen, nicht jedoch zum Vermitteln zwischen Teilnetzen. Das Konzept sieht einen Betrieb über mehrere Subnetze nicht vor, unterstützt dabei aber prinzipiell transparente Kommunikationsschnittstellen, die in Form von Proxys zwischen Teilnetzen vermitteln können. Durch die Berücksichtigung dieser Einschränkungen wird die Anforderung mit teilweise abgedeckt bewertet (+).

Prüffähigkeit: Die definierte Sicherheitsarchitektur spezifiziert eine Gruppe von Mitschneidemarkern, um Ereignisse während der TLS/SSL-Kanal Erstellung wie Authentifizierungsprozesse, LDAP-Anfragen, Verifizierungen der Zugangsrechte, AAA-Server Sessions und Zertifikatswiderrufe mitzuschneiden. Solche Marker kennzeichnen einen möglichen Erfolg, Misserfolg oder besitzen einen informativen Charakter. Der Hauptnutzen ist die Fehlersuche bei Sicherheitsvorgängen. Durch die Erfüllung dieser Anforderung wird diese als vollkommen abgedeckt erachtet und mit + + bewertet.

Gemeinsames Geheimnis: Keine der von der Architektur durchgeführten Sicherheitsvorgänge benötigen ein gemeinsames Geheimnis. Durch die Nutzung von X.509v3-Zertifikaten innerhalb des MMS-System besitzt jeder Kommunikationspartner ein eigenes Geheimnis. Abweichend davon werden bei der nutzerbasierten Authentifizierung einfache Legitimierungsnachweise, wie z.B. Nutzername und Passwort verwendet. Diese könnten als gemeinsames Geheimnis angesehen werden. Allerdings kann die nutzerbasierte Authentifizierung als optionaler Schritt in der Sicherheitsarchitektur konfiguriert werden. Deswegen kann zusammengefasst gesagt werden, dass die Richtlinie teilweise abgedeckt wird (+).

Unterstützung von servicespezifischen Metainformationen: Diese Anforderung liegt außerhalb der Betrachtung (0) der Sicherheitserweiterungen. Die Definition von Metainformationen oder zusätzlichen Attributen um die Funktionalitäten des AAA-Systems zu erweitern, entspricht plattformabhängigen schichtenübergreifenden Operationen. Zusätzlich muss dies auch von den Services unterstützt werden. Die Aufgaben und Ziele der MMS-Architekturen sind klar an der Integration von Sicherheitsrichtlinien orientiert.

Übersicht:

Im allgemeinen Bereich der AAA-Richtlinien liegen die zwei Aspekte „Skalierbarkeit“ und „Unterstützung von servicespezifischen Metainformationen“ außerhalb der Betrachtung und tragen nicht zu der Evaluierung bei. Die zwei Aspekte „Unterstützung von Kommunikationsschnittstellen“ und „Gemeinsames Geheimnis“ werden zum Teil abgedeckt. Dabei kann der erste Aspekt sehr einfach erweitert werden und bei dem zweiten handelt es sich um einen optionalen Schritt. Lediglich für die Ausfallsicherheit muss Aufwand betrieben werden, um diese zu gewährleisten. Dafür müssen Erweiterungen im Bereich der Hardware und Software vorgenommen werden.

Bereich	Anforderung	Bewertung
Allgemein	Skalierbarkeit	0
	Ausfallsicherheit	–
	Gegenseitige Authentifizierung	++
	Übertragungssicherheit in der Transportschicht	++
	Vertraulichkeit der Informationen	++
	Integrität der Informationen	++
	Zertifikatsaustausch	++
	Zuverlässige Übertragung	++
	Unterstützung von IPv4	++
	Unterstützung von IPv6	++
	Unterstützung von Kommunikationsschnittstellen	+
	Prüffähigkeit	++
	Gemeinsames Geheimnis	+
	Unterstützung von servicespezifischen Metainformationen	0

Tabelle 6.17.: *Bewertung der allgemeinen AAA-Richtlinien.*

Authentifizierung

Unterstützung von Network Access Identifier (NAI): Zwei Arten von NAI können in der sicheren MMS-Architektur unterschieden werden. Die erste wird bei der nutzerbasierten Authentifizierung als Nutzername in Form eines konventionellen Zeichenformats verwendet. Mit diesem kann sich ein bestimmter Nutzer gegenüber dem Sicherheitsmanagement legitimieren und sich von einem Client aus Zugang auf einen Service beschaffen. Die zweite Art von NAI ist wichtig bei der geräte- und servicebasierten Authentifizierung. Bei beiden

Vorgängen werden die Inhalte des CNs aus dem X.509v3-Zertifikat extrahiert und als Identifikationslegitimierung während der Einrichtung eines sicheren Kanals verwendet. Ebenso wird dies für die Anfragen nach Rechten durch den Service an den AAA-Server genutzt. Somit wird diese Anforderung als komplett abgedeckt angesehen (+ +).

Challenge Handshake Authentication Protocol (CHAP)-Unterstützung: CHAP ist sehr stark an Punkt-zu-Punkt Protokollen (PPP) orientiert. Es legt mehr Wert auf die sichere Übertragung der Passwörter als dessen Vorgänger Password Authentication Protocol (PAP). Diese Anforderung liegt nicht in der Betrachtung dieser Arbeit (0), da die serviceorientierte Philosophie von DPWS der leitungsvermittelnden Natur von PPPs entgegen steht. Jedoch werden ergänzend die durch CHAP ermöglichten Schutzmechanismen wie Authentifizierung und Verschlüsselung bereits durch vorher erwähnte Anforderungen abgedeckt.

Unterstützung des Extensible Authentication Protocol (EAP): Gegenwärtig werden die Authentifizierungsinformationen von den Sicherheitserweiterungen innerhalb von IP-Paketen ohne Rücksicht auf die Kapselungsmethode von EAP verwendet. Die Unterstützung würde das Ergänzen von weiteren Authentifizierungssystemen neben den TLS-Kanälen benötigen. Zusätzlich würde dies die Entwicklung von Anfrage/Antwort Mechanismen bedeuten, die mehr als einen Umlauf benötigen. Deswegen müssten Felder für die Kennzeichnung während der Paketerstellung vorgesehen werden um betreffende Authentifizierungsparameter aufzunehmen. Die derzeitige Lösung deckt diesen Aspekt nicht ab und wird dementsprechend mit – – bewertet.

Password Authentication Protocol (PAP)-Unterstützung: Während der nutzerbasierten Authentifizierung kann die Übertragung von Anfragen und Antworten durch die Verwendung von Passwörtern als Klartext durchgeführt werden. Die Klartextübertragung kann verwendet werden, wenn kein sicherer Kanal zwischen dem Client, AAA-Server und Service verfügbar ist. Durch den Einsatz einer TLS/SSL Verbindung können diese Informationen durch kryptografische Funktionen geschützt werden. Die gewöhnliche Abfrage des Nutzernamen/Passworts eines Client kann deswegen auch gesichert mit PAP in dem MMS-System durchgeführt werden. Aus diesen Gründen wird diese Anforderung mit vollkommen abgedeckt bewertet (+ +).

Erneute Authentifizierung bei Bedarf: Bis zur Verwendung eines Services und über den Nutzungszeitraum hinweg werden die nutzerbasierte, gerätebasierte und servicebasierte Authentifizierung lediglich einmal durchgeführt. Dabei werden die wichtigen Authentifizierungsparameter im Speicher der Programmstruktur der Anwendung vorgehalten. Sobald entweder der Client oder der Service die Verbindung trennt, gehen die gesamten Parameter verloren. Im Anschluss daran wird der gesamte Prozess für die Verbindung zu demselben Teilnehmer oder einem anderen wiederholt. Vereinfacht kann gesagt werden, dass eine erneute Authentifizierung aufgrund der einfachen Nachrichtenübertragung bei einem neuen Aufbau der Verbindung unnötig ist aber mit zusätzlichem Aufwand realisiert werden kann (–).

Autorisierung ohne Authentifizierung: Die realisierte indizierte Suche in dem LDAP-Verzeichnis und die Verifizierung mit der Zugriffskontrollliste basieren auf dem Wert des CN-Feldes aus dem X.509v3-Zertifikat des Clients. Dieses Feld wird extrahiert und einmalig beim Service gespeichert, um damit alle nachfolgenden Authentifizierungsvorgänge durchzuführen. Dadurch wird eine erneute Authentifizierung unnötig. Da die Anfragen und Verifikationen lediglich durch ein Identifikationsattribut (CN) anstelle von ergänzenden Passwörtern, gemeinsamen Geheimnissen oder ähnlichen kryptografischen Schlüsseln durchgeführt wird, deckt das MMS-System diese Anforderung vollkommen ab (++).

Übersicht:

Die AAA-Richtlinien im Bereich Authentifizierung sind nur zum Teil abgedeckt. So liegt die Richtlinie „CHAP Unterstützung“ außerhalb der Betrachtung, da der vorgeschlagene Ansatz dem Punkt-zu-Punkt-Vermittlungsgedanken widerspricht. Die Anforderung „Erneute Authentifizierung bei Bedarf“ kann mit Aufwand abgedeckt werden, aber die Abdeckung würde auch zu Verzögerungen bei der Übertragung durch den zusätzlichen Aufwand führen. Die „Unterstützung des EAP“ wird nicht abgedeckt. Eine Unterstützung dieser Richtlinie erfordert die vollständige Überarbeitung der Sicherheitserweiterungen des MMS-Systems und bedeutet keinen Mehrwert.

Bereich	Anforderung	Bewertung
Authentifizierung	Unterstützung von NAI	++
	CHAP Unterstützung	0
	Unterstützung des EAP	−−
	PAP Unterstützung	++
	Erneute Authentifizierung bei Bedarf	−
	Autorisierung ohne Authentifizierung	++

Tabelle 6.18.: *Bewertung der AAA-Richtlinien aus dem Bereich Authentifizierung.*

Autorisierung

Statische und dynamische IP Adressvergabe: Ein DPWS-Netz und dessen Sicherheitserweiterungen arbeiten mit im Vorfeld statisch oder dynamisch konfigurierten IP-Adressen. Das MMS-System trägt dabei nicht die Verantwortung für die Adressenvergabe und ein Service im System nutzt die IP-Adresse des Wirtssystems, egal ob diese nun statisch oder dynamisch zugewiesen wird. Diese Nutzung wird durch die Mechanismen WS-Discovery und WS-Addressing, die DPWS mit sich bringt, ermöglicht. In diesem Zusammenhang kann von einer vollkommen abgedeckten Anforderung ausgegangen werden (++).

Unterstützung von Remote Authentication Dial-In User Service (RADIUS): Die Sicherheitserweiterung des MMS-Systems wurde nicht entwickelt, um neben der RADIUS Technik zu bestehen oder zwischen diesen zu übersetzen. Umgekehrt sind die entwickelten Erweiterungen dafür gedacht die sicherheitsspezifischen Anforderungen für DPWS-Umgebungen,

die nicht auf Altlasten von RADIUS aufbauen, abzudecken. Dementsprechend wird die Entwicklung von Attributen und Protokollprimitiven, um externe Protokolle zu unterstützen, nicht beachtet und ist nur mit Aufwand abdeckbar (–).

Möglichkeit der Abweisung: Wie zuvor erwähnt wurde können die CA und der AAA-Server als eine Art Vermittlungsknoten in dem MMS-System angesehen werden. Reine Netzübergänge werden in der derzeitigen Lösung nicht angeboten. Im Moment werden alle Antworten auf Anfragen zum AAA-Server geleitet, um dort bearbeitet zu werden und schlussendlich an den Client zurückgeschickt zu werden. Die Anpassung von Vermittlungsknoten, sodass diese Zusagen oder Zurückweisungen auf Anfragen versenden können, ist in der Lösung nicht vorgesehen und wird nicht abgedeckt (––).

Ausschließen von Schicht 2 Tunneling: Die Realisierung von Tunnel Protokollen, wie z.B. Layer 2 Tunneling Protocol (L2TP) und Point-to-Point Tunneling Protocol (PPTP) wird auf der Sicherungsschicht der Erweiterungen nicht unterstützt. Auf der anderen Seite gibt es die Möglichkeit der Erstellung von Tunnelsessions in der Vermittlungs- und Transportschicht durch die TLS/SSL Verbindungen. Mechanismen wie Authentifizierung, Integrität und Zugangskontrolle erreichen eine gute Performanz in dem AAA-System durch die Verwendung von IP Headerflags oder Unterscheidung mit logischen Ports anstelle von Frameverarbeitung der Datenpakete oder physikalischen Switchmechanismen. Somit wird diese Anforderung mit ++ bewertet.

Erneute Autorisierung bei Bedarf: Wie in Abschnitt 5.3 erläutert, ist die Regelung der Zugangsrechte der Kern der Autorisierungsmechanismen. Solche Regeln werden bei jeglichem Authentifizierungsvorgang verifiziert, um die Zugangsrechte eines authentifizierten Nutzers oder Geräts zu bestätigen. Nichtsdestotrotz ist lediglich der Service fähig, eine erneute Autorisierung bei Bedarf anzufordern, wohingegen normalerweise nur jedes Mal wenn ein Client einen bestimmten Service abruft die betreffende ACL-Überprüfung durchgeführt wird. Die Absicht hinter der bedarfsabhängigen Autorisierung ist es, mögliche Änderungen der Zugangsrechte in Echtzeit zu beachten, aber die ACL-Einträge für Geräte und Nutzer bleiben solange bestehen bis ein neuer sicherer Kanal angefordert wird. Diese Aspekte führen bei der Bewertung zu einer teilweisen Abdeckung (+).

Unterstützung von Zugriffsregeln und Filtern: Zugangsregeln, Filter und Richtlinien werden vollständig durch die Sicherheitserweiterungen abgedeckt. Diese werden auf Geräte, Nutzer und Services angewendet, um die Netzinteraktion zwischen diesen zu reglementieren. Die Definition der Autorisierungsregeln folgt der ITU X.500 Empfehlungen für hierarchische baumbasierte Verzeichnisse wie beispielsweise LDAP. Durch die Befolgung dieser Regeln können z.B. Filterungsabläufe nach Domänenbereichen vorgenommen werden. Die angewendete Syntax für die Konfiguration der Zugangsregeln stammt von OpenLDAP. Mit dieser können Richtlinien für den Zugang eingerichtet werden, die den Ursprung und das Ziel des Zugreifenden festlegen und ob Attribute (Ressourcen) beschrieben oder gelesen

werden dürfen. Zweifelsfrei werden Zugangsrechte und Filter durch die Sicherheitserweiterungen vollständig abgedeckt und entsprechend mit + + bewertet.

Zustandswiederherstellung: Diese Anforderung bezieht sich auf das Ressourcenmanagement im MMS-System. So soll bei Ausfällen oder Fehlern der zuvorige Zustand des Systems wiederherstellbar sein. Dazu müssen Mechanismen angeboten werden um Ressourcen zuzuteilen. Die Sicherheitserweiterungen des MMS-Systems konzentrieren sich nicht auf die Zuteilung von Verarbeitungsaufgaben oder die intelligente Ressourcenverteilung in den Geräten. Die vorgeschlagene Lösung legt mehr Wert auf die Auflösung der Sicherheitsanforderungen als auf die Optimierung der Performanz oder die Neuorientierung der physikalischen Ressourcen. Dank der Ausführungen kann diese Anforderung als außerhalb der Betrachtung der Architektur angesehen werden (0).

Unaufgeforderter Verbindungsabbruch: Es gibt drei mögliche Szenarien in denen ein unaufgeforderter Verbindungabbruch auftreten kann. Eine Verbindung wird nach einer missglückten gerätebasierten, nutzerbasierten oder servicebasierten Authentifizierung abgebrochen. Sobald einer dieser Abläufe nicht erfolgreich durchgeführt wurde, wird eine Trennung des Client und Service ausgelöst. So werden keine weiteren Nachrichten übermittelt. Durch die gescheiterte Authentifizierung wird vom dem MMS-System angenommen, dass eine Sicherheitsbeeinträchtigung durch nicht autorisierte oder bösartige Nutzer- oder Gerätezugriffe aufgetreten ist. Auf ähnliche Weise führt ein verdächtiges Verhalten in der Kommunikation zwischen Service und AAA-Server zu einem sofortigen und unaufgeforderten Verbindungsabbruch. Zusammenfassend kann diese Richtlinie als vollständig abgedeckt angesehen werden (+ +).

Übersicht:

Auch im Bereich der Autorisierung gibt es Einschränkungen bezüglich der Abdeckung von AAA-Richtlinien. So liegt die Wiederherstellung des Systemzustands außerhalb des Fokus dieser Arbeit. Des Weiteren kann mit Aufwand die Anforderung „Erneute Autorisierung bei Bedarf“ erfüllt werden. Hierfür müssen Mechanismen bereitgestellt werden, mit denen bei Verbindungen eine erneute Autorisierung beliebig durchgeführt werden kann. Die Abdeckung der Richtlinie „Unterstützung von RADIUS“ gestaltet sich aufwändiger. Es müssten Anpassungen der jetzigen Sicherheitserweiterungen vorgenommen werden, um zusätzlich RADIUS zu unterstützen. Und somit würde sich eine zweite Sicherheitslösung in dem MMS-System etablieren, die zusätzlich abgestimmt werden müsste. Die Richtlinie „Möglichkeit der Abweisung“ wird nicht abgedeckt, da keinerlei Vermittlungsknoten in dem MMS-System vorgesehen sind.

Abrechnung

Echtzeitabrechnung: Die Sicherheitsarchitektur führt eine Reihe von Sicherheitsvorgängen durch, die entweder Abrechnungsmitschnitte mit informativem oder fehlerbeschreiben-

Bereich	Anforderung	Bewertung
Autorisierung	Statische und dynamische IP Adressvergabe	++
	Unterstützung von RADIUS	−
	Möglichkeit der Abweisung	−−
	Ausschließen von Schicht 2 Tunneling	++
	Erneute Autorisierung bei Bedarf	+
	Unterstützung von Zugriffsregeln und Filtern	++
	Zustandswiederherstellung	0
	Unaufgeforderter Verbindungsabbruch	++

Tabelle 6.19.: *Bewertung der Autorisierungsrichtlinien.*

dem Charakter in Echtzeit generieren. Der Aufbau wird in Abbildung 5.7 gezeigt und dieser wird erstellt sobald das Ereignis aufgetreten ist. Die Abrechnungsinformationen zeigen eine Beschreibung der durchgeführten Vorgänge und das korrespondierende Ergebnis. Zusätzlich wird dazu ein Zeitstempel für die zusammenhängende Protokollierung festgehalten. Diese Richtlinie ist damit vollkommen abgedeckt (++).

Verpflichtung zur kompakten Kodierung: Die Abrechnungsmitschnitte werden nur lokal jeweils bei den Komponenten selbst vorgehalten. Das bedeutet, dass keine Abrechnungsnachrichten über das Netz des MMS-Systems übertragen werden. Deswegen werden keine Kodiertechniken realisiert, die den Overhead reduzieren. Nichtsdestotrotz verwendet die Sicherheitsarchitektur die Abrechnungsmitschnitte für Anwendungen, die einen Mehrwert bilden wie der Ablaufverfolgung und der Fehlersuche. Aufgrund der genannten Fakten wird die Anforderung als nicht abgedeckt bewertet (−−).

Erweiterbarkeit der Abrechnungsmitschnitte: Die Abrechnungsinformationen in der Sicherheitsarchitektur sind nur für die Fehlersuche und Ablaufsverfolgung ausgelegt. Die Erweiterungen der Abrechnungseigenschaften durch die Kennzeichnung mit speziellen Attributen, Overhead Flags oder anwendungsspezifischen Nutzdaten liegt außerhalb der Betrachtung der vorgeschlagenen Lösung (0).

Gebündelte Abrechnung: Hinsichtlich der in der Anforderung „Verpflichtung zur kompakten Kodierung“ aufgeführten Argumente wird diese ebenfalls von den Sicherheitserweiterungen als nicht abgedeckt angesehen (−−).

Garantierte Zustellung: Das Verteilen der Abrechnungsinformationen von einem Gerät zu einer anderen Stelle des MMS-Systems liegt nicht im Fokus dieser Arbeit. Aus diesem Grund werden in dem Design der Sicherheitserweiterungen keine Mechanismen für die sichere Übertragung betrachtet. In diesem Fall würde die Verantwortung für die Authentifizierung und Autorisierung an untere Netzprotokollschichten, wie z.B. der Vermittlungs- oder der Transportschicht, übergeben werden. Das Protokollieren der Abrechnung für einen

garantierten Transport wird ebenfalls auf diese Schichten übertragen. Aus diesem Grund liegt die Richtlinie außerhalb der Betrachtung (0).

Zeitstempel für die Abrechnung: Das Auftreten von Ereignissen, wie der nutzer-, geräte- oder servicebasierten Authentifizierung genauso wie die Verifizierung der Zugangsrechte und alle verwandten Zwischenschritte für die Einrichtung von sicheren Verbindungen kann aufgezeichnet werden. Eine entsprechende Aufzeichnung erfolgt mit Zeitstempelinformationen, die den Tag, Monat, Jahr und die Zeit der Sicherheitsvorgänge festhalten. In diesem Sinn kann diese Richtlinie als vollkommen abgedeckt angesehen werden (++).

Dynamische Abrechnung: Diese Richtlinie ermöglicht die Unterstützung von mehreren Abrechnungsaufzeichnungen pro Verbindung, um dynamische Authentifizierungen und Autorisierungen mitzuschneiden. Durch die Sicherheitserweiterungen werden zwei Hauptarten von Aufzeichnungen definiert. Die erste stellt Informationen und die zweite Fehler dar. Beide erzeugen Einträge nach jedem Authentifizierungs- oder Autorisierungsvorgang während einer aktiven Verbindung. Abhängig von dem Ergebnis der Aktion gehört der Abrechnungsmitschnitt zu den Informations- oder Fehlerbeschreibungen. Deswegen ist die Richtlinie erfüllt und wird mit ++ bewertet.

Übersicht:

Der Bereich der Abrechnung ist kein Hauptaspekt der vorgeschlagenen Sicherheitserweiterungen. Aus diesem Grund werden die kompakte Kodierung und die gebündelte Abrechnung, um den Netzverkehr zu minimieren und freie Ressourcen des Netzes besser auszunutzen, nicht realisiert. Die Richtlinien „Erweiterbarkeit der Abrechnungsmitschnitte“ und „Garantierte Zustellung“ liegen außerhalb des Fokus dieser Arbeit und werden nicht betrachtet.

Bereich	Anforderung	Bewertung
Abrechnung	Echtzeitabrechnung	++
	Verpflichtung zur kompakten Kodierung	−−
	Erweiterbarkeit der Abrechnungsmitschnitte	0
	Gebündelte Abrechnung	−−
	Garantierte Zustellung	0
	Zeitstempel für die Abrechnung	++
	Dynamische Abrechnung	++

Tabelle 6.20.: *Bewertung der AAA-Richtlinien aus dem Bereich Abrechnung.*

Auswertung der qualitativen Betrachtung

Zusammenfassend kann gesagt werden, dass nahezu alle relevanten Anforderungen an eine zukünftige sichere MMS-Architektur aus der Analyse in Kapitel 3.2 erfüllt werden. Diese

waren die Leitprinzipien für die Konzeptionierung und Realisierung einer dynamischen Generation von Benutzerschnittstellen in verteilten Systemen.
Die ergänzenden Sicherheitserweiterungen decken alle nötigen AAA-Richtlinien in den Bereichen Allgemein, Authentifizierung, Autorisierung und Abrechnung ab. Wenn eine Anforderung nicht abgedeckt wird, so ist diese nicht entscheidend um die Schutzziele in dem vorgeschlagenen MMS-System zu erfüllen oder letztere werden von anderen Anforderungen bereits mitabgedeckt. Aufgrund der qualitativen Betrachtung kann von einer ausreichenden Stärke der Informationssicherheit für die Sicherheitserweiterungen gesprochen werden, durch die die Schutzmechanismen aus Kapitel 2.3 *Zugangskontrolle*, *Authentifizierung*, *Nichtabstreitbarkeit*, *Datenvertraulichkeit*, *Kommunikationssicherheit*, *Datenintegrität*, *Verfügbarkeit* und *Privatsphäre* abgedeckt werden.

6.5. Demonstratoren

Während der Durchführung dieser Arbeit sind zahlreiche Demonstratoren entstanden. Diese dienten neben der Präsentation der Ergebnisse vor Kollegen und Interessierten auch zur quantitativen Evaluierung und Machbarkeitsanalyse in der Praxis. So entstanden neben kleineren Teilaufbauten ein Labordemonstrator für exemplarische Anwendungen der vorgeschlagenen MMS und ein funktionstüchtiger Fahrzeugdemonstrator mit der in [DEH+11] vorgestellten IT-Architektur.

6.5.1. Labordemonstrator

In Abbildung 6.20 ist der Aufbau des Labordemonstrators zu sehen. Der Aufbau zeigt vereinfacht die wichtigsten Komponenten des vorgeschlagenen MMS-Systems.

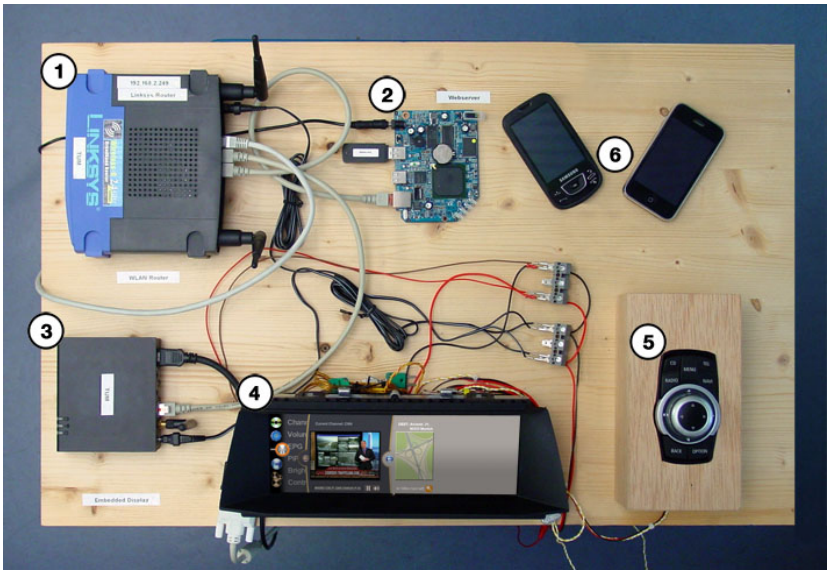

Abbildung 6.20.: *Labordemonstrator mit Ein- und Ausgabekanälen.*

Die zentrale Komponente ist ein drahtloser Router (1), der sowohl den Zugang für mobile Geräte darstellt als auch das ethernetbasierte Kommunikationsnetz. Des Weiteren setzt sich der Demonstrator aus einem eingebetteten Webserver (2) und einem weiteren eingebetteten System mit Webbrowser (3) zusammen. Daran angeschlossen ist ein automotives Display (4). Zusammen mit der vorigen Komponente werden diese beiden auch als intelligentes Display (3+4) bezeichnet. Für die Interaktion wird der Aufbau durch einen automotiven Eingabecontroller (5) und mobile Geräte (6) für die Ein- und Ausgabe erweitert.
Der Laboraufbau zeigt exemplarisch verschiedene Aspekte des MMS-Systems. So wird die GUI mit Webtechnologien durch den Webserver zur Verfügung gestellt und diese kann z.B. von dem intelligenten Display abgerufen werden. Zusätzlich bietet der Eingabecontroller seine Funktionalität als Service in dem Netz an. Das Display kann daher die Ereignisse des Controllers abonnieren. Ab diesem Moment ist es möglich, durch den Controller mit der grafischen Ausgabe zu interagieren. Auch die Erweiterung durch mobile Geräte wird mit den Smartphones gezeigt. Diese rufen ebenfalls die Inhalte des Webservers ab und es kann mit diesen über den eingebauten Touchscreen interagiert werden. Ergänzend können sich die mobilen Geräte ebenfalls bei dem Controller anmelden wodurch es möglich wird, dass angemeldete Geräte gleichzeitig auch mit dem Controller gesteuert werden können. Das bedeutet, dass die Eingabe des Controllers nun von einem bestimmten Anzeigesystem entkoppelt ist und innerhalb der Architektur frei verfügbar ist. Nicht zu sehen ist in der Abbildung, dass ohne großen Aufwand zwischen verschiedenen Designs gewechselt werden kann. So werden zwei Designs aus dem Fahrzeugbereich nachgebildet und der Demonstrator um diese und drei eigene Entwürfe ergänzt.

6.5.2. Fahrzeugdemonstrator

Alle Konzepte aus [DEH+11] wurden in dem erwähnten Forschungsprojekt *IT_Motive2020* in einem Fahrzeugdemonstrator vereint. Dabei handelt es sich um ein Serienfahrzeug der BMW Group. Wie in Abbildung 6.21 zu sehen ist, wurden die Komponenten für die neue IT-Infrastruktur im Kofferraum verbaut.

Der Fahrzeugdemonstrator wurde so ausgelegt, dass dieser für den Transport sein ursprüngliches Fahrzeugnetz nutzen kann und man zu Demonstrationszwecken auf das zusätzlich integrierte vorgeschlagene IT-Netz umschalten kann. Dieses Netz stellt die Basisfunktionalitäten der Konzepte aus dem Projekt *IT_Motive2020* zur Verfügung. Des Weiteren wurden zentrale Komponenten wie der iDrive-Controller und das integrierte Fahrzeugdisplay ebenfalls in dieses Netz miteingebunden um die Interaktion mit dem Menschen zu zeigen. Mit Hilfe von verschiedenen Szenarien wurden die Stärken und Vorteile des Ansatzes aus *IT_Motive2020* gezeigt.

6.6. Zusammenfassung

In diesem Kapitel wird die zuvor erarbeitete und abgesicherte MMS-Architektur evaluiert und die Realisierung vorgestellt. Bei der Realisierung ist es wichtig zu erwähnen, dass für

Abbildung 6.21.: *In einem Fahrzeugdemonstrator integriertes Schattennetz.*

das Konzept Webkomponenten wie Webbrowser und Webserver verwendet werden. Es werden die wichtigsten Aspekte der Realisierung vorgestellt und die dafür verwendeten Techniken genannt. Die Evaluierung wird auf zwei Arten vorgenommen. Die erste berücksichtigt die quantitative Betrachtung und die zweite die qualitative Betrachtung.
Für die quantitative Betrachtung werden im Vorfeld die verwendeten Szenarien eingeführt und anschließend die aussagekräftigsten Ergebnisse vorgestellt. Es kann gezeigt werden, dass die MMS-Architektur im Hinblick auf auftretenden Verzögerungen, Prozessor- und Speicherauslastungen eine sehr gute Performanz zeigt. Das System eignet sich auch für den Einsatz in eingebetteten Systemen, wie z.B. im automotiven Bereich. Zudem wird durch die Evaluierung gezeigt, dass der Nutzer keine störenden Effekte verspürt und von der Komplexität des verteilten Systems unberührt bleibt.
Für die qualitative Betrachtung werden die Anforderungen für eine zukünftige MMS-Architektur und die AAA-Richtlinien für die Sicherheit verwendet, um diese gegen die vorgeschlagene Architektur zu bewerten. Bei dieser Bewertung zeigte die Architektur eine gute Abdeckung aller relevanten Anforderungen. Die nicht oder nur teilweise abgedeckten Anforderungen sind für den Bestimmungszweck des MMS-Systems von geringer Bedeutung oder werden durch andere erfüllte Anforderungen kompensiert.
Schlussendlich werden Demonstratoren vorgestellt, die während der Durchführung dieser Arbeit entstanden sind. Diese dienten dazu die Machbarkeit zu demonstrieren und ebenfalls um die quantitative Evaluierung durchzuführen.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurde ein System für die dynamische Generierung von sicheren Benutzerschnittstellen in verteilten Systemen erarbeitet und evaluiert. Die Ergebnisse wie auch weitere interessante Aspekte, die an diese Arbeit anschließen können, werden im Nachfolgenden zusammengefasst.

7.1. Ergebnisse

Das erarbeitete System orientiert sich an zuvor aufgestellten Anforderungen einer zukünftigen Mensch-Maschine-Schnittstelle. Dadurch ergeben sich wesentliche Vorteile des Konzepts gegenüber dem Stand der Technik. Die **Dezentralisierung der Endkomponenten** des Systems führt zu einer verbesserten Anpassung an die Bedürfnisse der Nutzer. Das System kann beispielsweise **dynamisch durch Komponenten erweitert** werden, die als Ausgabegeräte für die grafische Darstellung agieren. Um die Dezentralisierung realisieren zu können, muss eine **Zentralisierung des Managements** eingeführt werden. Auf diese Weise kann durch das Management ohne viel Aufwand beispielsweise eine **Mehrnutzerfähigkeit** erreicht werden. Ebenfalls wird die **dynamische Integration von Diensten** wesentlich erleichtert. Somit können Funktionalitäten auf Geräten von Nutzern vorhanden sein, die dann in dem MMS-System anderen Nutzern zur Verwendung angeboten werden. Durch die Organisation in dezentrale Komponenten und einem zentralen Management wird eine **Skalierbarkeit** ermöglicht, die lediglich von der zugrundeliegenden Hardware abhängt, jedoch keine Anpassungen des Konzepts erfordern. Die entwickelten Schnittstellen ermöglichen eine **Homogenisierung zwischen den Ein- und Ausgabegeräten**. Mit anderen Worten: Die Grenze zwischen diesen Geräten verwischt, da diese nur noch als Dienste dargestellt werden. Durch die **Flexibilität** des Systems werden grafische Ausgaben basierend auf Funktionalitäten der Geräte unterstützt, wie auch die direkte Kommunikation zwischen Geräten selbst. Des Weiteren wird durch eine **Entkopplung des Aussehens von den Inhalten** eine **universelle Anwendbarkeit** des Systems und dessen Design ermöglicht. So kann beispielsweise im Fahrzeugbereich das gleiche System modellreihenübergreifend verwendet und das Design je nach Bedarf an die aktuelle Baureihe oder sogar an die persönlichen Vorlieben des Fahrers angepasst werden. Diese Abtrennung wird durch webbasierte Protokolle für die grafische Darstellung ermöglicht, die wiederum aus Grafikbefehlen bestehen und deren Interpretation vom Webbrowser übernommen wird. Auf diese Weise wird die **Datenübertragung** zwischen GUI-Renderer (Webbrowser) und GUI-Kombinierer (Webserver) **minimiert**. Die **Modularisierung des Aussehens, der Logik und der Struktur** vermeidet die funktionsspezifische Entwicklung von Geräten (ganzheitlicher Ansatz) und setzt sich auch in der Entwicklung der Softwarekomponenten fort. Somit kann durch die servicebasierten und standardisierten Schnittstellen ein **schnelles Entwickeln** gewährleistet werden.

Außerdem wird die Entwicklung durch die Verwendung des Webbrowsers und damit einer **plattformübergreifenden** und standardisierten Komponente zur Darstellung des GUI-Inhalts beschleunigt. Da Webbrowser heutzutage auf fast allen internettauglichen Geräten anzutreffen sind, werden diese auch für nahezu alle Hardwarearchitekturen und Betriebssysteme angeboten.

Durch die Öffnung des MMS-Systems für bis zum jetzigen Zeitpunkt noch unbekannte Geräte und Funktionalitäten, ist eine Betrachtung der **Informationssicherheit** unerlässlich. Zu diesem Zweck wird das System um ein **Zugangs- und Zugriffsmanagement** erweitert, mit welchem ein **Zonenmodell** realisiert werden kann. Das Rechtemanagement erstreckt sich dabei auf alle in der Architektur vorhandenen Kommunikationskomponenten: Nutzer, Geräte und sogar Services. Durch das Zonenmodell ist es möglich, ein umfassendes Regelwerk für die Kommunikation und Nutzung von Ressourcen zwischen den Kommunikationskomponenten aufzustellen und auf ein beliebiges Szenario anzuwenden. Mit Hilfe von Zertifikaten und einer Public-Key-Infrastruktur wird zum einen eine **eindeutige Identität** sowie die **zugesicherte Autorisierung** der Komponenten im System ermöglicht. Die Autorisierung bzw. die Authentifizierung zur Sicherstellung der Identitäten ist im Vergleich zu der **Verschlüsselung** obligatorisch. Beispielsweise kann sich ein ressourcenlimitiertes Gerät dazu entschließen, seine Daten bei der Übertragung nicht zu verschlüsseln. Es muss sich aber in jedem Fall über die erwähnten Autorisierungsmechanismen sicher in die Infrastruktur integrieren. Um das System noch weiter auch an **leistungsschwache Geräte anzupassen**, kann die verwendete Chiffre zur Sicherung des Kommunikationskanals im Vorfeld aus mehreren Chiffren ausgehandelt werden. Somit kann auf diesen Geräten durch leistungsschwächere Chiffren der Ressourcenverbrauch stufenweise gesenkt werden.

Das gesamte MMS-System einschließlich der Sicherheitserweiterungen wird quantitativ wie auch qualitativ evaluiert. Die **quantitative Betrachtung** zeigt, dass die minimale Ausprägung des MMS-Systems bei den **kritischen Punkten** wie Verzögerung, Speicherplatz und Prozessorauslastung weit unterhalb der dafür **definierten oberen Grenzwerte** agiert. Dies gilt auch unter Betrachtung der eingeführten Sicherheitserweiterungen.
Mit der **qualitativen Betrachtung** werden alle aufgestellten Anforderungen an ein zukünftiges sicheres MMS-System analysiert und bewertet. Die Evaluierung hat gezeigt, dass sämtliche **relevanten Anforderungen** durch das Konzept **abgedeckt werden**. Einige Anforderungen bleiben außerhalb des Fokus dieser Arbeit, wieder andere werden teilweise abgedeckt. Letztere sind jedoch nicht kritisch bzw. werden deren Aspekte teilweise bereits von erfüllten Anforderungen abgedeckt.

7.2. Ausblick auf weitere Arbeiten

Einige der Konzepte dieser Arbeit bauen auf dem derzeitigen Stand der Technik und auf in den jeweiligen Teilbereichen etablierten Standards auf. Die dadurch erarbeiteten und vorteilhaften Erweiterungen können wiederum in diese Standards eingebracht werden.
So ist es denkbar, dass im Bereich der serviceorientierten Architekturen (SOA) der modulare Aufbau des Device Profile for Web Service (DPWS) durch ein **WS*-Protokoll für die grafische Repräsentation**, wie es in dieser Arbeit vorgestellt wird, **erweitert** werden kann.

Damit würde ein Service nicht nur eine WSDL-Beschreibung seiner Funktionalitäten besitzen, sondern könnte auch die vorgeschlagene grafische Beschreibung anbieten. Auf diese Weise kann jedes DWPS-Gerät seine eigene GUI mitbringen und jeder Nutzer des Gerätes kann diese bei sich integrieren.
Im Teilgebiet der grafischen Benutzerschnittstellen können ebenfalls Vorschläge für eine Standarderweiterung bzw. Weiterentwicklung gemacht werden. Durch eine **Erweiterung der HTML5-Spezifikation durch DPWS** wäre es möglich, die serviceorientierte Funktionsweise mit der grafischen Darstellung zu verknüpfen. Somit wird beispielsweise ein im Webbrowser integrierter Videoplayer nicht nur über den HTML-Inhalt konfigurierbar oder steuerbar, sondern auch über ein verteiltes System. Damit ließe sich eine homogene Verbindung zwischen SOA und Webtechnologien schaffen.

Neben den Erweiterungsvorschlägen für Standards kann über eine Änderung der Struktur des Systems nachgedacht werden. Die **Verarbeitung und Speicherung von Daten in sogenannten „Clouds“** nimmt verstärkt zu. Da eine zentrale Komponente der vorgeschlagenen MMS-Architektur ein Webserver ist, kann über die **Auslagerung des Webservers**, eventuell auch einschließlich der Sicherheitskomponenten, in das Internet nachgedacht werden. Auch die Verteilung von Funktionalitäten über das Internet wäre denkbar. Somit kann auf die GUI von überall zugegriffen werden und es ergeben sich zahlreiche neue Möglichkeiten wie die **Fernwartung** durch Administratoren oder eine **Fernunterstützung**. Auch hinsichtlich der Funktionalitäten würden sich Vorteile ergeben wie z.B. die **stetige Aktualität** dieser, da die Funktionalitäten direkt eingebunden werden könnten und nicht als ausführbare Programme lokal verfügbar gemacht werden müssen. Eventuell kann sich somit die GUI in einen **geräteübergreifenden permanenten Begleiter** wandeln, der den Nutzer als eine Art persönlicher Helfer unterstützt.

A. Anhang

A.1. Methoden der GUI-Generierung

Auf dem Gebiet der grafischen Benutzerschnittstellen gibt es mehrere Möglichkeiten, um die Funktionalitäten einer Anwendung dem Nutzer zugänglich zu machen. Diese lassen sich in drei Bereiche aufteilen. Zum einen kann eine GUI programmiert werden, sie kann durch Befehlsauflistungen beschrieben werden oder sie kann durch semantische Beschreibungen abgebildet werden. Bei den zuletzt genannten Arten wird durch Festlegung bestimmter Abläufe der Befehle oder Beschreibungen im Allgemeinen von Protokollen gesprochen.
Im Folgenden werden Vorteile und Einschränkungen dieser drei Schemas diskutiert und die Bedeutung für diese Arbeit erörtert.

Programmierung der GUI: Bei dieser Methode wird die GUI zusammen mit der Anwendung programmiert. Das bedeutet, dass bei der Entwicklung in der Regel dieselbe Programmiersprache verwendet wird und ein einziges ausführbares Programm für eine Zielarchitektur erstellt wird. Somit befindet sich an dem Ausführungsort der GUI auch immer die Anwendung. Folglich ist eine der Trennung der beiden nahezu unmöglich. Ein einfaches Beispiel für eine Anwendung mit einer programmierten GUI ist xcalc [X C10] (siehe Abbildung A.1)
Das Programmieren einer Anwendungsoberfläche bringt viele Vorteile mit sich. Während der Ausführung einer Anwendung muss die GUI nur von der jeweiligen Komponente, z.B. dem X-Window System (siehe [X. 10]) , die auf dem zugrundeliegenden System das Rendern übernimmt, grafisch aufbereitet und anschließend dargestellt werden. Durch das bereits im Vorfeld gestaltete und formatierte GUI wird eine äußerst geringe Reaktionszeit erreicht. Auch die starre Verbindung der bereits programmierten Funktionalitäten mit den GUI-Elementen spart Ausführungszeit. Diese Methode eignet sich sehr gut für Anwendungen die komplexe und hochinteraktive grafische Benutzeroberflächen anbieten. Allerdings ergeben sich auch Nachteile dieser Methode. Eine programmierte GUI hängt immer von der Laufzeitumgebung ab. So müssen bei Bedarf verschiedene Implementierungen in Betracht gezogen werden. Des Weiteren ist die GUI zusammen mit der Anwendung programmiert, das bedeutet, dass im Fall einer Änderung der Anwendung auch die GUI neu programmiert oder zumindest angepasst werden muss. Ein erweitertes Architekturmuster das diese Methode als Grundlage nutzt ist „Modell/Präsentation/Steuerung (engl. Model View Controller, MVC) [LR06]. Dieses wird verwendet, um die Softwareentwicklung in die drei Einheiten Datenmodell, Präsentation und Programmsteuerung zu strukturieren. Dadurch wird es ermöglicht, dass bei Änderungen oder Portierungen der Software lediglich die relevante Einheit abgeändert werden muss und z.B. nicht das Datenmodell. Die anderen Nachteile bleiben weiterhin bestehen.

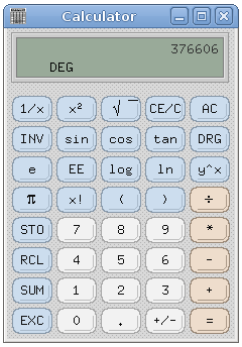

Abbildung A.1.: *Taschenrechner „xcalc“ als einfaches Beispiel für eine programmierte GUI.*

Beschreibung der GUI: Eine GUI kann auch beschrieben werden. Dabei gibt es eine Auflistung von Befehlen, die bestimmte Bedeutungen haben. Zusätzlich zu dieser Beschreibung benötigt es einen Interpreter, der diese in eine GUI umsetzt. In Abbildung A.2 wird anhand einer Beispieldatei gezeigt, wie bei Scalable Vector Graphics (SVG) [W3C10a] Beschreibungen in ein Bild umgesetzt werden. Dabei werden in der Datei als erstes die SVG-konformen Metainformationen angegeben und danach durch die verfügbaren SVG-Befehle grafische Elemente beschrieben. In der Abbildung handelt es sich dabei links um das SVG-Dokument selbst, rechts um die Interpretation in Form eines blauen Quadrats, sowie eines roten Kreises mit jeweils einer schwarzen Umrandung. Diese Beschreibungen lassen sich je nach Aufwand beliebig komplex gestalten, dabei ist es nur wichtig, dass der Interpreter angepasst wird.

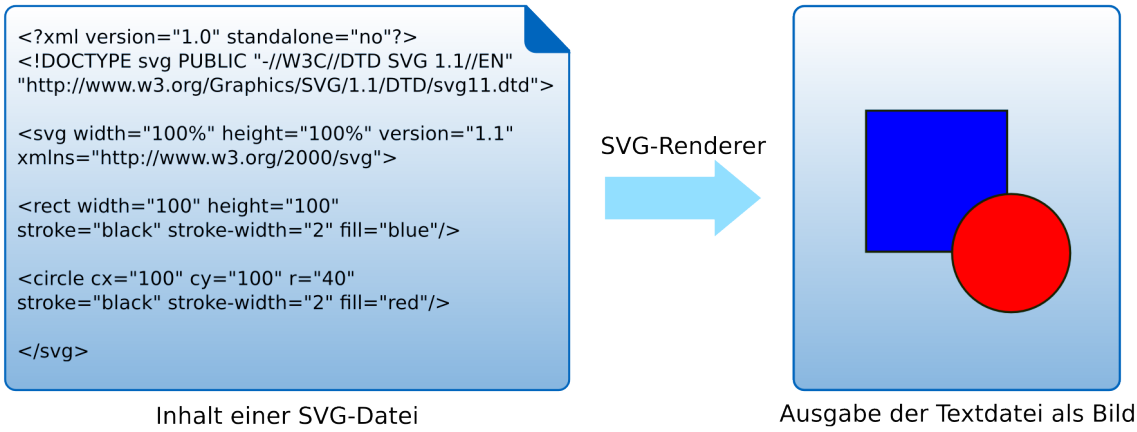

Abbildung A.2.: *Beispiel für die Beschreibung eines Grafikelements (SVG).*

Es ist von Vorteil, wenn man sich bei der Wahl der Beschreibung an einem Standard orientiert. Zu diesem gibt es mit großer Wahrscheinlichkeit einen weitverbreiteten Interpreter und man erspart sich die Implementierung eigener Lösungen. Für verschiedene Laufzeitumgebungen existieren meist auch entsprechende Implementierungen. Es muss deswegen nicht explizit darauf geachtet werden, jede zu unterstützen. Zusätzlich muss man sich nicht um die Pflege oder Wartung dieser kümmern und kann meist von einer Entwicklergemeinde profitieren die sich dieser Problematik widmet. Dieses Vorgehen erspart Zeit und Kosten und der eigentliche Aufwand beschränkt sich auf die Entwicklung des Aussehens der GUI. Hochkomplexe und äußerst interaktive GUIs, wie z.B. Desktop-Oberflächen, lassen sich lei-

der damit nur mit einigem Aufwand realisieren.

Semantische Beschreibung der GUI: Eine grafische Benutzerschnittstelle kann auch semantisch beschrieben werden. Ähnlich wie zuvor gibt es eine Beschreibung der GUI. Hingegen existieren nun eine abstrakte und eine exakte Repräsentation der GUI. Durch die abstrakte Beschreibung wird die Bedeutung einzelner Elemente festgelegt und durch die exakte Beschreibung werden diese Elemente definiert und initialisiert. Die Umsetzung dieser Beschreibungen wird auch von einem Interpreter übernommen, dieser muss aber weitaus komplexer als derjenige zuvor sein. Diese Methode ist ebenfalls plattformunabhängig.
Leider gibt es nicht sehr viele offene Realisierungen. Es existiert ein erster Ansatz der BMW Group. Dieser basiert auf den Standards XML [Har01] und OWL [HKRS07] und wird nur durch firmeneigene Erweiterungen von dem GUI-Framework „Fluid“ unterstützt. Dieses stellt einen „GUI-Generator“ zur Verfügung, welcher das jeweilige Nutzerinterface generiert. Dazu interagiert dieser mit einem „GUI-Widget-Verzeichnis“, in dem jedes Element semantisch beschrieben vorliegt. Hierbei handelt es sich um einen proprietären Ansatz und die Entwicklung ist aufgrund der Komplexität teuer und umständlich. Des Weiteren ist nicht festgelegt, ob weitere Firmen diesen Ansatz nutzen würden.
Die semantische Beschreibung ist für alltägliche und nicht sehr komplexe GUI sehr gut geeignet, z.B. ein MP3-Player, da durch die Ähnlichkeit der GUI der Vorteil der Wiederverwendbarkeit voll ausgenutzt werden kann. Die abstrakte Beschreibung wird immer wieder verwendet und lediglich die exakte angepasst. Ein Problem ist dagegen die Verwendung von unbekannten GUI-Elementen, bei denen deren Behandlung zu Problemen führen können. Ein Hauptnachteil bei dieser Methode ist, dass diese hinsichtlich GUIs noch von keinem Standard unterstützt wird.

A.2. Evaluierungskriterien für grafische Benutzerschnittstellen

Um die vorgestellten Techniken im Bereich der grafischen Benutzerschnittstellen evaluieren zu können, wurden folgende relevante Kriterien erarbeitet:

- **Dokumentation:** Da für die Anwendung einer Technik die Dokumentation essentiell ist, soll deren Grad mit diesem Kriterium bewertet werden. Je ausführlicher die Dokumentation, desto besser ist dies für die Entwickler.

- **Verbreitung:** Auch die Verbreitung spielt eine große Rolle und soll zusammen mit der Bekanntheit hier betrachtet werden. So kann gegebenenfalls von Entwicklergemeinschaften profitiert werden. Von großem Nutzen für die Akzeptanz ist es, wenn die Technik bereits weit verbreitet ist.

- **Verfügbarkeit:** Damit keine hohen Lizenzkosten anfallen, darf dieser Punkt nicht vernachlässigt werden. Von Vorteil sind dabei Lizensierungsmodelle von freier Software mit keinen oder angemessenen Kosten.

- **Plattformunabhängigkeit:** Gerade mit dem Aufkommen vieler neuer mobiler Geräte ist die Portierung von Software ein wichtiger Gesichtspunkt geworden. Eine hohe

Unabhängigkeit ist als positiver Aspekt zu bewerten.

- **Komplexität:** Der Aufwand um eine Software zu entwickeln wird in diesem Punkt bewertet. Punkte wie Einarbeitung in die Technik oder auch Strukturierung der Technik werden hier betrachtet und eine geringe Komplexität hat den größten Nutzeffekt.

- **Ressourcen:** Da beispielsweise im Fahrzeugbereich oft eingebettete Systeme verwendet werden, wird hier festgehalten wie hoch der Ressourcenverbrauch für die Darstellung der GUI am Client ist. Ein geringer Verbrauch ist als positiver Aspekt einzustufen.

- **Prototypenentwicklung:** Bei der Entwicklung neuer Anwendungen, die eine grafische Oberfläche besitzen sollen, ist es von großem Vorteil wenn man sich am Anfang der Realisierung Gedanken über deren Aussehen machen kann. Dadurch ist es möglich mit wenig Aufwand und möglichst schnell einen ersten Prototypen bereitzustellen.

- **Look-and-Feel:** Dieses Kriterium beschreibt wie gut das Verändern des Erscheinungsbildes unterstützt wird. Dabei gilt es als positiv wenn eine sehr gute Unterstützung vorhanden ist.

- **Änderungen am Look-and-Feel:** Zu dem vorherigen Kriterium was die Unterstützung an sich bewertet, evaluiert dieses die Anstrengung bei der Veränderung des Aussehens. Ein geringer Aufwand ist von großem Nutzen.

- **Kopplung:** Ein großer Vorteil für eine grafische Benutzeroberfläche ist es, wenn diese nicht an ein bestimmtes Design gebunden ist und man dieses dynamisch ohne großen Aufwand austauschen kann um die darunter liegende Anwendung zu verändern. Dieser Punkt betrachtet die Trennung von Design und Anwendung, eine lose Kopplung ist somit von Nutzen.

- **Interaktion:** Mit diesem Punkt wird der Grad der Interaktion bewertet, der dem Nutzer durch die jeweilige Technik ermöglicht wird. Je höher der Grad ist, desto höher wird er gewichtet.

- **Reaktionszeit:** Für jede Mensch-Maschine Interaktion ist die Reaktionszeit entscheidend. Ist diese zu lange, wird der Nutzer ungeduldig und vermutet einen Fehler bei der Bearbeitung. In diesem Fall wird er höchstwahrscheinlich das System in Frage stellen und unzufrieden sein.

- **Erweiterbarkeit:** Dieser interessante Aspekt soll die Erweiterbarkeit um neue Grafikelemente widerspiegeln. Falls ein neues Element hinzukommt, soll der Aufwand für die Anpassungen hierfür betrachtet werden. Dabei ist ein geringer Aufwand nützlich.

A.3. Evaluierungskriterien für serviceorientierte Architekturen

Folgende relevante Kriterien für die Evaluierung der serviceorientierten Architekturen wurden bestimmt:

- **Dokumentation:** Eine gute und ausführliche Dokumentation ist auch hier unabdingbar für eine sinnvolle Verwendung der vorgestellten Techniken.

- **Verbreitung:** Um von der Verbreitung und vielleicht bereits bestehenden Lösungen zu profitieren, wird dieser Punkt zur Evaluierung aufgestellt.

- **Verfügbarkeit:** Hier wird wieder hinsichtlich der Kosten für die Lizenzierung und ähnliche Aspekte, wie z.B. Folgekosten, bewertet.

- **Plattformabhängigkeit:** Um möglichst viele Geräte zu unterstützen ist dies ein wichtiger Aspekt. Deswegen ist eine weite Abdeckung von Plattformen und verschiedenen Programmiersprachen wünschenswert. Gerade um die allgemeine Akzeptanz zu fördern, ist es wichtig eine plattformunabhängige Lösung anzubieten.

- **Komplexität:** Um Kosten einzusparen, kann durch eine geringere Komplexität der Aufwand für die Entwicklung gesenkt werden. Ein positiver Nebeneffekt ist dabei das Verringern der möglichen Fehlerquellen durch weniger Komplexität.

- **Ressourcen:** Innerhalb von Bereichen mit eingebetteten Systemen ist es wichtig, einen niedrigen Ressourcenverbrauch zu erreichen. Das bedeutet, dass die jeweilige SOA wenig Anforderungen an die Hardware stellen darf. Ebenso werden Bedingungen wie niedriger Energieverbrauch, niedrige Basiskosten und Ausfallsicherheit an die Hardware gestellt. Deswegen wird die Wahl einer SOA aus dem Bereich der eingebetteten Systeme bevorzugt.

- **Peer-to-Peer:** Mit diesem Punkt soll das Thema Ausfallsicherheit angeschnitten werden. Dies ist von großer Bedeutung z.B. im automotiven Bereich. Deswegen ist von Vorteil wenn sich die Anwendungen direkt mit anderen Anwendungen austauschen können und nicht nur über eine zentrale Instanz. Bewertet wird hier der Grad der Notwendigkeit einer zentralen Einheit.

- **Ereignisbearbeitung:** Das Abonnieren und das Benachrichtigen bei Ereignissen ist ein relevantes und elegantes Konzept für den Nachrichtenaustausch. Dies mindert den Netzverkehr außerordentlich, verglichen zu abfragebasierten Ansätzen. Gerade im Fahrzeug liegen viele ereignisbasierte Anwendungen vor und deswegen ist ein Verzicht undenkbar.

- **Geräteunterstützung:** Geräte können mehrere Services mit sich bringen. Die unterschiedlichen Services auf dem Gerät müssen separat behandelt werden und es dürfen sich keine Einschränkungen der einzelnen Servicefunktionalitäten während der gleichzeitigen Nutzung ergeben.

- **Plug and Play:** Mobile Geräte sollten während der Laufzeit des Systems integrierbar sein. Aus diesem Grund ist eine nahtlose Rekonfiguration der SOA ohne die normalen Abläufe zu stören nötig. Falls möglich, sollte dies ohne die Beteiligung des Nutzers geschehen.

- **Filterung:** Ein wichtiger Aspekt im Zusammenhang mit Plug and Play ist die Filterung. Geräte sollten lediglich andere Geräte finden, die zu ihren Funktionalitäten ver-

wandt sind, die sich im selben Sicherheitsumfeld befinden oder deren Zugriffsrechte dies erlauben.

- **Zustandsspeicherung:** Ein Teil des Nachrichtenaustausches hängt nicht nur von den Teilnehmern ab, sondern auch von dem internen Zustand der Services. Dies muss beachtet werden um unnötiges Nachrichtenaufkommen einzudämmen.

- **Netzverkehr:** Es gibt selten Szenarios in denen es unerheblich ist, wie viel Verkehr an Nachrichten verursacht wird. Im Hinblick auf Energiekosten und der verwendeten Netzinfrastruktur gilt dies beispielsweise auch in Fahrzeugen. Des Weiteren muss sichergestellt werden, dass genügend Kapazitäten für sicherheitsrelevante Services wie z.B. die Airbag Sensoren freigehalten werden.

- **Sicherheit:** Sicherheit ist ein sehr bedeutsames Thema bei offenen Architekturen wie SOA. Dabei kann es sich sowohl um die Abwehr von bösartigen Angriffen handeln, wie auch das Schützen geistigen Eigentums. Deswegen muss eine SOA Mechanismen anbieten, die dies unterstützen.

- **Codegenerierung:** Um den Entwicklern das Arbeiten mit der Technik zu erleichtern ist die automatische Codegenerierung ein interessanter Punkt. Auch die Akzeptanz dieser würde durch die niedrigeren Kosten profitieren.

- **Flexibilität:** Wenn nahezu jede Art von Gerät in eine SOA integriert werden soll, muss diese eine hohe Flexibilität hinsichtlich des Kommunikationsschemas und der Verwaltung dieser Geräte bieten.

A.4. Empfohlene Sicherheitsanforderungen für DPWS

In der OASIS Spezifikation [OAS10a] werden eine Reihe von Anforderungen definiert, die erfüllt werden müssen, damit DPWS-Geräte gewissen Sicherheitsrichtlinien gerecht werden. Nachfolgend werden diese vorgestellt.

A.4.1. Adressierung:

Ein Service darf keine unsicheren HTTP-Adressen für Aufrufe anbieten. Hingegen darf er logische Endpunktreferenzen[1] und HTTPS in Kombination mit xAddrs[2] verwenden. Auf diese Weise wird eine Veröffentlichung von Informationen an andere Netzteilnehmer verhindert.

[1] Diese setzen sich aus einer Adresse (URI) und Eigenschaften dieser Referenz zusammen, um untergeordnete Ressourcen eindeutig zu definieren.

[2] Dabei handelt es sich um eine HTTP oder HTTPS Adressrepräsentation, welche zusätzlich eindeutige IDs, sogenannte Universally Unique Identifier (UUID), enthalten.

A.4.2. Legitimierung:

Jeder Client und Service muss zur Legitimierung ein eigenes, einzigartiges Zertifikat besitzen. Dieses muss dem X.509v3-Standard [ITU10b] entsprechen, um Abläufe der Zertifizierungsstelle wie den Widerruf, Speicherung oder Reaktivierung unterstützen zu können. Dementsprechend muss das Zertifikat den Gerätenamen, die auszustellende Vertrauensstelle (in diesem Fall die Zertifizierungsstelle) und den öffentlichen Schlüssel beinhalten, um das Minimum der Sicherheitsanforderungen abdecken zu können.
Zusätzlich kann ein Service Mechanismen anbieten, um die Identität eines Clients durch die Prüfung unterer Netzschichten, wie zum Beispiel der IP-Adresse, durchzuführen.

A.4.3. Auffindung:

Ein Service darf keine `Probe Match`-Nachricht schicken, wenn die Anfrage von einem Client außerhalb des Netzes des Services stammt und die `Probe` Nachricht über Multicast versendet wurde. Aufgrund von Konfigurationsfehlern bei Filterregeln oder Zugriffsrechten der Vermittlungsknoten könnten sich dadurch Sicherheitsprobleme auf Geräten außerhalb der Domäne des Services ergeben. Des Weiteren muss ein Client eine `Probe Match`-Nachricht verwerfen, wenn diese nach `MATCH_TIMEOUT` Sekunden nachdem die letzte `Probe`-Nachricht gesendet wurde, empfangen wurde. Dies wurde festgelegt, weil veraltete Nachrichten von bösartigen oder nicht-autorisierten Geräten versendet sein könnten.
Dasselbe Vorgehen gilt für `Resolve Match`-Nachrichten. Diese dürfen ebenfalls nicht gesendet werden, wenn sich der anfragende Client außerhalb des Netzes des Services befindet und über Multicast die `Resolve`-Nachricht empfangen wurde. Die `Resolve Match`-Nachricht muss ebenfalls von einem Client verworfen werden, wenn diese mehr als `MATCH_TIMEOUT` Sekunden nach dem Versand der letzten `Resolve`-Nachricht empfangen wurde.

A.4.4. Sichere Auffindung:

Ein Client sollte seine Auffindungsnachrichten durch *WS-Discovery Compact Signatures* signieren um somit dem Service Authentifizierung und Integrität anbieten zu können. Wenn eine empfangene Auffindungsnachricht keine Signatur besitzt oder diese nicht verifizierbar ist, kann diese verworfen werden.
Eine Voraussetzung für die Verwendung von *WS-Discovery Compact Signatures* ist die weitere Erfüllung der nachfolgenden Anforderungen „Authentifizierung“ und „Integrität“.

A.4.5. Datenübertragung:

Beschreibungs-, Kontroll- und Notifikationsnachrichten zwischen dem Client und dem Service müssen über einen sicheren Kanal übertragen werden. `Probe`-Nachrichten müssen von

einem Service über einen HTTPS-Kanal empfangen und beantwortet werden. Auf der anderen Seite kann ein Client eine `Probe`-Nachricht, die über einen nicht sicheren Kanal versendet wurde, ignorieren. Des Weiteren kann der Client einen sicheren Kanal verwenden, um mehrere angefragte Services zu nutzen, wenn diese unter derselben Adresse und Port zu erreichen sind. Alle nachfolgenden Nachrichten, die über HTTP ausgetauscht werden, sollten den existierenden sicheren Kanal nutzen.
Die Voraussetzung für einen sicheren Kanal sind die nachfolgend erläuterten Punkte „Authentifizierung“, „Integrität“ und „Vertraulichkeit“.

A.4.6. TLS/SSL Chiffre Unterstützung:

Ein Service muss mindestens den empfohlenen asymmetrischen, kryptografischen Algorithmus RSA mit einem symmetrischen 128 Bit-Schlüssel, RC4 Chiffre und SHA für die Integritätsprüfung unterstützen. Hingegen kann von einem Service eine Erweiterung um RSA mit einem symmetrischen 128 Bit-Schlüssel, AES-Chiffre und SHA verwendet werden.

A.4.7. Authentifizierung:

Ein Client muss bei allen Geräten, die Multicast-Nachrichten verarbeiten, eine geeignete Legitimierung verwenden. Dies ist nötig um das Vertrauensniveau beim Beitritt von neuen Geräten aufrecht zu erhalten.
Bei der Verwendung eines durch TLS/SSL abgesicherten Kanals muss ein Service für jede Verbindung darüber hinaus die Möglichkeit der Authentifizierung anbieten. Andernfalls kann nicht für die Glaubwürdigkeit der Endpunkte garantiert werden.

A.4.8. Service Authentifizierung:

Ein Client muss beim Aufbau einer TLS/SSL Session eine Authentifizierung mit dem Service durchführen. Diese beinhaltet die Verwendung von X.509v3-Zertifikaten zur Legitimierung und zur Abdeckung der Authentifizierung, Vertraulichkeit und Integrität. Ein Service muss sich hingegen selbst gegenüber einem Client durch ein X.509v3-Zertifikat zum Zeitpunkt des TLS/SSL Handshake authentifizieren, um den Hostnamen verifizieren zu lassen und den jeweiligen öffentlichen Schlüssel mitzuteilen.

A.4.9. Client Authentifizierung:

Ein Service kann die Authentifizierung eines Clients durch die Übertragung eines X.509v3-Zertifikats oder durch die Nachfrage nach einem Nutzernamen und Passwort über einen HTTPS-Kanal anfordern. In dem Fall, dass die Zertifikate nicht verfügbar sind oder die Verifizierung nicht möglich ist, kann der Service die HTTP-Authentifizierung alternativ über einen Nutzernamen und Passwort durchführen. Der Nutzername/Passwort-Handshake für

die Client-Authentifizierung muss über einen sicheren Kanal stattfinden, damit so die Vertraulichkeit und Integrität der Informationen weiterhin gewährleistet bleibt.

A.4.10. Integrität:

Falls ein Service einen TLS/SSL-Kanal oder *WS-Discovery Compact Signatures* nutzt, muss er die Integrität für jede darüber stattfindende Verbindung sicherstellen. Auf diese Weise können Veränderungen, Fälschungen oder Kürzungen der ursprünglichen Nachrichten erkannt werden.
Daneben darf ein Service keine SOAP-Nachricht [Sit10] ohne Mechanismen für den Schutz der Integrität der Informationsblöcke versenden. Deswegen kann er SOAP-Nachrichten ohne einen solchen Schutz abweisen. Die Integrität des darauf folgenden Nutzdatenblocks der SOAP-Nachricht muss hingegen nicht durch den Service gesichert werden.

A.4.11. Vertraulichkeit:

Falls ein Service eine sichere TLS/SSL-Verbindung verwendet, muss diese die Vertraulichkeit unterstützen, um den Datenschutz der Nachrichten zu gewährleisten. Deswegen darf der Service keine SOAP-Nachrichten ohne Verschlüsselung des Nutzdatenteils verschicken und kann gleichermaßen jede SOAP-Nachricht ohne verschlüsselten Nutzdatenanteil zurückweisen.

A.5. Nutzdatenszenarien

Nachfolgend werden die Szenarien beschrieben, die für Klassifizierung der Nutzdaten verwendet wurden. Aus diesen gehen auch Kernaspekte für die Konzeptionierung eines Sessionmanagements hervor.

A.5.1. Global Positioning System (GPS):

In diesem Szenario ist es durch einen GPS-Empfänger möglich, die derzeitigen Koordinaten zu ermitteln. Durch dessen Integration in das vorgeschlagene MMS-System ist dieser in der Lage, die ermittelten Koordinaten anderen Geräten zur Verfügung zu stellen. Voraussetzung hierfür ist es, dass die interessierten Geräte ebenfalls an der Infrastruktur teilnehmen.

In Abbildung A.3 wird eine beispielhafte Vernetzung gezeigt, die sich aus drei Komponenten zusammensetzt. Die Komponente GPS-Empfänger ist mit der Empfangseinheit für das GPS-Signal ausgestattet und besitzt die Logik um die aktuellen Positionskoordinaten zu berechnen. Mit dieser sind nun zwei weitere Komponenten über das Netz verbunden. Das Navigationssystem, wie auch der GPS-Datenlogger, benötigen die Koordinaten. Das Navigationssystem navigiert damit den Nutzer zu einem bestimmten Punkt und bereitet dies mit

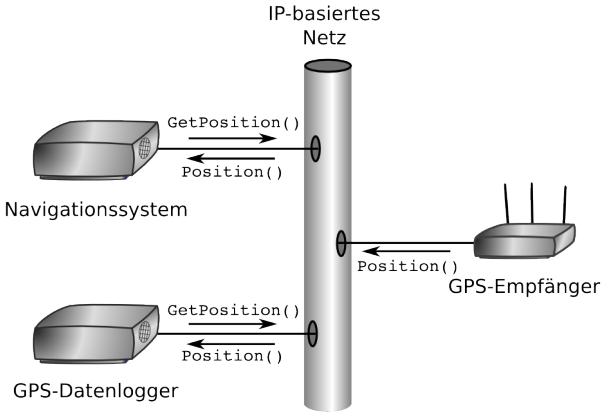

Abbildung A.3.: *GPS Szenario.*

Kartenmaterial auf. Der Datenlogger schreibt diese mit, um den begangenen Weg zu dokumentieren. Für diese Konstellation kann nun der GPS-Empfänger als ein Service betrachtet werden, der die anderen zwei Komponenten als Dienstnutzer (Clients) mit Informationen versorgt. Für diesen Zweck bietet der GPS-Empfänger einen Serviceaufruf, wie zum Beispiel `GetPosition()` an. Je nach Bedarf kann durch diesen Aufruf ein Client beispielsweise als Antwort `Position(lat,long)` mit den Positionsdaten zurückbekommen und diese weiterverarbeiten. Bei diesen Koordinaten handelt es sich um eine kleine Datenmenge mit einer ungewissen Auftrittswahrscheinlichkeit. Das bedeutet, dass je nach Client die Informationen öfters oder nur selten ausgetauscht werden.

A.5.2. Eingabecontroller:

Da ein Eingabecontroller nicht immer direkt mit dem Bestimmungsgeräte seiner Eingabe verbunden sein muss, ergibt sich ein weiteres Szenario. Der Controller kann ebenfalls die Infrastruktur nutzen und seine Informationen an andere Teilnehmer weitergeben.

Als Beispiel wird in Abbildung A.4 ein Verbund aus Eingabegeräten und Bildschirmgeräten gezeigt. Bei dem Eingabegerät handelt es sich beispielsweise (vereinfacht) um einen Controller, mit dem Nutzereingaben wie „oben“, „unten“, „rechts“, „links“ und „Bestätigung“ entgegengenommen werden können. Diese können von weiteren Geräten verarbeitet werden. Das Bildschirmgerät könnte eines dieser Geräte sein. So kann dort ein Menü dargestellt werden und die Informationen des Eingabegerätes können verwendet werden um durch das Menü zu navigieren. In diesem Beispiel stellt der Eingabecontroller den Service dar und die Bildschirmgeräte die Clients. Diese Verteilung der Rollen kann vertauscht werden. Da zum einen aber die Nutzereingaben unbestimmt auftreten und zum anderen sich mehrere Clients für dieselbe Information eines Eingabecontroller interessieren können, wurde diese Verteilung gewählt. Auch wenn die Nutzereingaben schnellstmöglich zu dem Empfänger sollen, kann durch das direkte Übertragen anstelle einer Abfrage Zeit gespart werden. Bei dieser bietet der Controller einen Abonnier-Service an, zum Beispiel `ReceiveUserInput()`, bei dem sich die Interessenten registrieren können und dann beim Auftreten von Nutzereingaben über diese informiert werden. So werden geringe Datenmen-

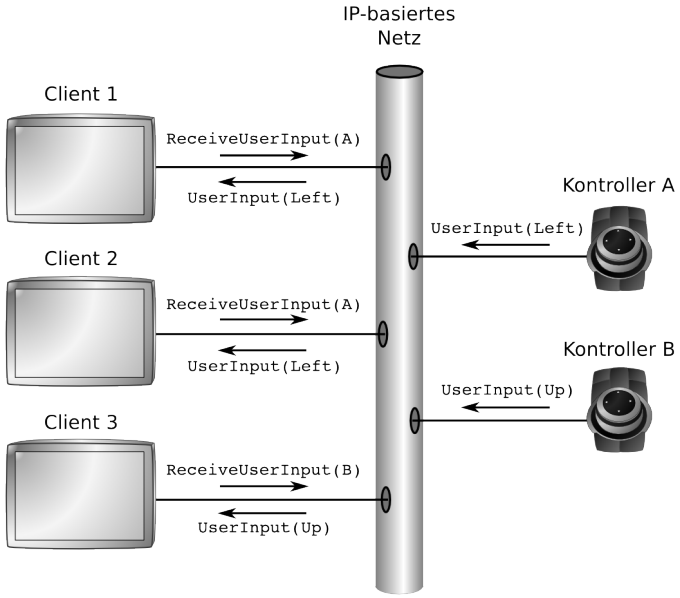

Abbildung A.4.: *Eingabecontroller Szenario.*

gen direkt beim Auftreten, das durch den Nutzer bestimmt wird, beispielsweise über die Notifikation `UserInput(Command)` an die Clients geschickt und dort verarbeitet.

A.5.3. Videowiedergabe:

In diesem Szenario bietet ein Gerät (Videostreamer) Videos für Geräte an, die diese abspielen können (Videoplayer). Nach der Teilnahme beider Typen von Geräten an der vorgeschlagenen Infrastruktur, können die Inhalte des Videos von Sender zum Empfänger übertragen werden.

Abbildung A.5 zeigt verschiedene Konstellationen von Videostreamern und Videoplayern. Zunächst ist nur der einfache Fall relevant, die anderen Fälle werden im weiteren Verlauf des Kapitels näher erläutert. In diesem einfachen Fall dient der Videostreamer als Service und bietet einen Serviceaufruf wie `StartVideo()` an. Dieser Serviceaufruf kann von Videoplayern (Client) genutzt werden um die Aussendung des Videos beim Streamer zu starten. Dabei wird jeweils ein Videobild des Videos, in ein Datenpaket gepackt und an den Empfänger geschickt. Dort wird dieses empfangen, entpackt und dargestellt. Wegen der hohen Datenmenge eines Videobildes und der hohen Auftrittswahrscheinlichkeit wird diese Übertragung außerhalb von DPWS geregelt. Aspekte wie Rückmeldung des Empfängers über die Kanaleigenschaften sprechen für ein eigenes Protokoll. Des Weiteren müssen in diesem Szenario zuvor noch weitere Informationen ausgetauscht werden, um beispielsweise den Decoder richtig zu initialisieren.

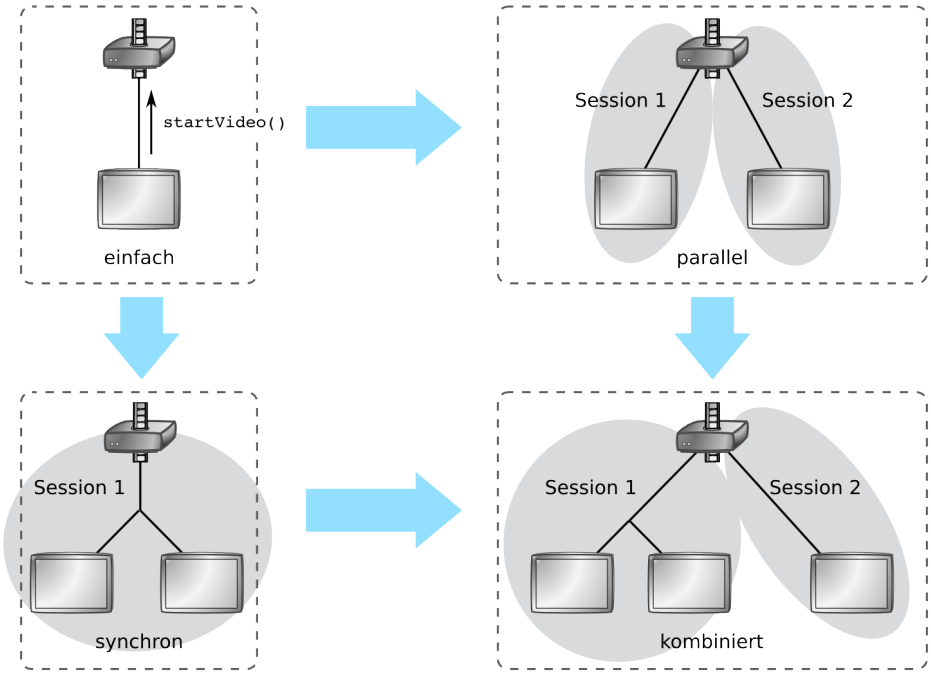

Abbildung A.5.: *Mögliche Konstellationen eines Videostreamingszenarios.*

A.6. Sicherheitsanforderungen in IT-Systemen

Die Internet Engineering Task Force (IETF) definiert durch den Begriff AAA-Richtlinien in Form von Requests for Comments (RFCs) [MJB+01, CHM00] eine Reihe von Anforderungen, um die Stärke der Informationssicherheit bezüglich der Authentifizierung, Autorisierung und Abrechnung in IT-Systemen zu bestimmen. Vereinfacht kann gesagt werden, dass sich die Sicherheit einer dadurch betrachteten Architektur mit der Anzahl der erfüllten Anforderungen erhöht. Einer der größten Vorteile der AAA-Richtlinien ist der modulare Aufbau dieser Anforderungen und die dadurch ermöglichte Adressierung von nur bestimmten Schutzzielen. Somit ist es möglich, Sicherheitsmechanismen die auf einen der drei Sicherheitsbereiche ausgerichtet sind, mit einer stärkeren Gewichtung zu berücksichtigen. Im Nachfolgenden werden die AAA-Anforderungen aufgelistet und im Detail vorgestellt.

A.6.1. Allgemeine Anforderungen

Skalierbarkeit

Mit dieser Anforderung wird die Notwendigkeit adressiert, dass das System auf nahezu beliebig viele Nutzer und dementsprechend zahlreiche gleichzeitige Anfragen reagieren kann. Beispielsweise soll eine AAA-Architektur fähig sein, tausende von Geräten, AAA-Komponenten, und Kommunikationsschnittstellen wie z.B. Router oder Netzzugangspunkte unterstützen zu können. Mit dieser Fähigkeit wird sichergestellt, dass es zu keiner Überlastung des Systems durch eine große Anzahl an Teilnehmern kommen kann.

Ausfallsicherheit

Eine AAA-konforme Architektur muss Mechanismen unterstützen, mit denen beim Auftreten eines Ausfalls oder einer Auslastung einer Systemkomponente auf eine Ersatzkomponente ausgewichen kann. Durch diese Art der Selbstheilung kann der reibungslose Betrieb des Systems im Fehlerfall gewährleistet werden.

Gegenseitige Authentifizierung

Durch diese Anforderung wird die gegenseitige Authentifizierung zwischen einem AAA-Client und einem AAA-Server gefordert. Dies gewährleistet die Möglichkeit der bestätigten gegenseitigen Identifizierung der Teilnehmer und verhindert vorgetäuschte falsche Identitäten.

Übertragungssicherheit in der Transportschicht

Die AAA-Richtlinien fordern die Möglichkeit der Nutzung von Authentifizierung, Vertraulichkeit und Integrität in der Transportschicht. Da dadurch ein sicherer Kanal zwischen zwei Kommunikationspartnern erstellt werden kann, gehört diese Sicherheitsanforderung auch zu der Gruppe der Ende-zu-Ende Sicherheit.

Vertraulichkeit der Informationen

Des Weiteren wird die Vertraulichkeit auf der Datenebene mit den dazugehörigen Attributen verlangt. Dadurch wird festgelegt, dass, unabhängig von der Anzahl der Zwischenknoten, nur der Zielknoten die Daten entschlüsseln kann und so eine unerwünschte Veröffentlichung verhindert wird.

Integrität der Informationen

Diese Anforderung stellt die Authentifizierung und Integrität auf der Datenebene mit den jeweiligen Attributen sicher. Um die Integrität zu prüfen, muss auch die Identität gesichert sein, deshalb ist eine Kombination dieser beiden Mechanismen nötig. Wie zuvor bei der Vertraulichkeit, muss die Authentifizierung dazu über mehrere Zwischenknoten persistent bleiben. Der ursprüngliche Integritätsnachweis kann ebenfalls über Zwischenknoten geführt werden. Eine Veränderung der Daten durch Zwischenknoten beeinträchtigt den Integritätsnachweis und kann somit detektiert werden.

Zertifikatsaustausch

Es muss gesichert sein, dass Zertifikate jeglicher Art zwischen Teilnehmern ausgetauscht werden können. Zu diesem Zweck soll eine zusätzliche Infrastruktur, die den Zertifikatsaustausch regelt, vermieden werden, um die Komplexität niedrig zu halten und die Kompatibilität mit bereits verwendeten Formaten zu garantieren. Somit soll eine effiziente Bearbeitung der Zertifikate ermöglicht werden.

Zuverlässige Übertragung

Die AAA-Vorschläge definieren mit diesem Punkt, dass die Übertragung gegen Paketverluste, genauso wie gegen Ausfälle resistent sein muss. Ebenfalls müssen Kontrollmechanismen für die erneuten Übertragungen der AAA-Anwendungen, wie z.B. Benachrichtigungen von erfolgreichen Zustellungen, Huckepack-Übertragungen von Bestätigungen der AAA-Nachrichten und zeitlich festgelegten AAA-Antworten vorhanden sein.

Unterstützung von IPv4

Die Richtlinien fordern für die Adressierung die Verwendung von IPv4.

Unterstützung von IPv6

Sie fordern ebenfalls die Unterstützung von IPv6-Adressierung.

Unterstützung von Kommunikationsschnittstellen

Hinsichtlich der Kommunikationsschnittstellen legen die Richtlinien fest, dass Vermittlungsknoten und Netzübergänge, die an einer AAA-Infrastruktur teilnehmen, unterstützt werden müssen. Diese Schnittstellen sind nötig, um zwischen AAA-Komponenten aus verschiedenen administrativen Domänen zu vermitteln. Dies soll dabei möglichst transparent auf den Transportwegen geschehen.

Prüffähigkeit

Durch diesen Punkt wird verlangt, dass es möglich sein muss, definitiv bestimmen zu können, welche Operationen mit AAA-Paketen auf dem Weg von einem AAA-Server zu einem AAA-Client durchgeführt werden. Diese Prüffähigkeit ermöglicht es, Vorgänge und Abläufe im Netz zu beobachten und nachzuweisen.

Gemeinsames Geheimnis

Diese Anforderung verbietet die Nutzung eines gemeinsamen Geheimnisses als primären Mechanismus um auf der Datenebene Vertraulichkeit und eine nichtabstreitbare Identität der Kommunikationspartner zu gewährleisten. Durch die ITU-T wird empfohlen, gemeinsame Geheimnisse lediglich als Vervollständigung des Absicherungsablaufs zu verwenden. Besonders der Austausch, das Management und das Sicherheitsniveau bei der Verwendung von gemeinsamen Geheimnissen können nicht als robust angesehen werden.

Unterstützung von servicespezifischen Metainformationen

Dieser Punkt verlangt von dem jeweiligen AAA-Protokoll, dass es durch Dritte erweitert werden kann. So soll es möglich sein, Metainformationen oder Attribute zu definieren, die spezifisch für einen bestimmten Service gedacht sind.

A.6.2. Authentifizierung

Durch die Authentifizierung wird eine Information eines Teilnehmers verifiziert, in den betrachteten Fällen handelt es sich dabei meist um Identifikationsnachweise.

Unterstützung von Network Access Identifier (NAI)

Diese AAA-Richtlinie fordert die Unterstützung von NAI für die Bezeichnung von Nutzern und Geräten. Dazu können die sogenannten URL oder URI verwendet werden.

Challange Handshake Authentication Protocol (CHAP)-Unterstützung

Dieser Punkt beschreibt, dass die Übertragung von CHAP Informationen erlaubt ist. Dieser Mechanismus wird oft von Network Access Servern (NAS) benutzt, um Nutzer des Point-to-Point Protokolls zu authentifizieren. Durch diese Unterstützung wird ein zusätzliches Maß an Vertraulichkeit gewährleistet.

Unterstützung des Extensible Authentication Protocol (EAP)

Diese Anforderung definiert die Möglichkeit der Nutzung des EAP und damit die Übertragung von dessen Nutzdaten. EAP soll die Erweiterbarkeit um Mechanismen garantieren. Da einige EAP-Authentifizierungsmechanismen mehr als einen Nachrichtenumlauf benötigen, muss auch dies unterstützt werden. Des Weiteren sollen die EAP Aushandlungsnachrichten transparent für das AAA-System sein.

Password Authentication Protocol (PAP)-Unterstützung

Durch die Richtlinien wird definiert, dass es möglich sein muss, Klartextgeheimnisse auf eine sichere Art übertragen zu können (z.B. vertrauliche Passwörter für Anwendungen innerhalb des Netzes). Zusätzlich zur PAP-Unterstützung wird ein gewisses Maß an Schutz gegen die Veröffentlichung von verschlüsselten Passwörtern entlang des Übertragungspfades benötigt. Diese Vorkehrungen sind für Anwendungen, die für Authentifizierungsvorgänge auf einfache, textbasierte Passwörter zurückgreifen, nötig.

Erneute Authentifizierung bei Bedarf

Dieser Punkt fordert die Unterstützung einer erneuten Authentifizierung von Nutzern bei Bedarf. Dadurch können Änderungen am Authentifizierungsprozess dynamisch aktualisiert werden. Dieser Vorgang kann von den Nutzern selbst, einem AAA-Client oder einem AAA-Server ausgelöst werden.

Autorisierung ohne Authentifizierung

Das AAA-System darf die Nutzerlegitimierungen nicht für Autorisierungsvorgänge verwenden. Nicht erlaubte Aktionen müssen unterbunden werden, wenn zuvor keine Authentifizierung, sondern nur eine Identifikation oder die Behauptung über eine Identität stattgefunden hat. Zugriffsregeln sollen deswegen auf vordefinierten Nutzerprofilen mit festgelegten Privilegien basieren.

A.6.3. Autorisierung

Bei der Autorisierung werden bestimmten Teilnehmern Rechte gewährt, hier wird dies meistens mit Zugriffsrechten verwendet.

Statische und Dynamische IP Adressvergabe

Durch diesen Punkt wird die Unterstützung von statischer und dynamischer Adresszuweisung sowohl über IPv4 als auch über IPv6 während des Autorisierungsprozesses gefordert.

Unterstützung von Remote Authentication Dial-In User Service (RADIUS)

Ein AAA-konformes System soll fähig sein, mit der weitverbreiteten Technik RADIUS zusammenzuarbeiten. So soll ein Server entweder beide Protokolle (RADIUS und sein eigenes Sicherheitsprotokoll) beherrschen oder zwischen seinem und RADIUS vermitteln.

Möglichkeit der Abweisung

Das AAA-System soll ermöglichen, dass ein Vermittlungsknoten einem Kommunikationsteilnehmer den Zugang verweigern kann, ohne die Zugangsinformationen an den AAA-Server weitergeleitet zu haben. Zusätzlich kann dieser den Zugang ebenfalls verwehren, nachdem er eine positive Quittierung durch einen AAA-Server zurückbekommen hat.

Ausschließen von Schicht 2-Tunneling

Die Richtlinien verbieten das Schicht 2-Tunneling, wie es z.B. bei Virtual Private Networks (VPN) verwendet wird. Auf diese Weise wird zwar eine Ende-zu-Ende Vertraulichkeit realisiert, jedoch ist es möglich, in diesen unteren Schichten unbemerkt Daten zu versenden und zu empfangen. Das bedeutet, dass die Umgehung von Sicherheitsmechanismen und Firewalls auf höheren Schichten ermöglicht wird. Gerade in Firmen kann dies zu Problemen hinsichtlich der Verbreitung von firmeninternen Informationen oder der Einschleusung von Schadsoftware führen.

Erneute Autorisierung bei Bedarf

Hierdurch soll ermöglicht werden, dass der AAA-Client oder der AAA-Server eine erneute Autorisierung anstoßen darf. Des Weiteren soll es möglich sein, aktualisierte Autorisierungsinformationen an ein Gerät zu schicken. Dies ist nötig, weil ein Server einen Nutzer zu Beginn der Kommunikation autorisiert hat Dienste zu nutzen. Während der Nutzung könnten sich diese Rechte jedoch geändert haben. Daher muss erneut geprüft werden, ob der Nutzer noch autorisiert ist, die Dienste zu verwenden.

Unterstützung von Zugriffsregeln und Filtern

Mit dieser Anforderung soll sichergestellt werden, dass es möglich ist den Zugriff auf Geräte und Nutzer und die Autorisierungen für diese festzulegen. Mit diesen Regelungen kann ein granulares und gruppenbasiertes Zugriffsmanagement für alle Teilnehmer eingeführt werden.

Zustandswiederherstellung

Es soll möglich sein, den aktuellen Zustand der Ressourcen zu sichern. Das bedeutet, dass AAA-Server, AAA-Clients oder Vermittlungsknoten Wiederherstellungsmechanismen anbieten müssen, um Datenverlust bei Fehlern, Neustarts von AAA-Komponenten oder Kommunikationsausfällen vorzubeugen.

Unaufgeforderter Verbindungsabbruch

Dieser Punkt bezieht sich darauf, dass ein Server fortlaufende Anfragen nach Autorisierungsrechten auch in einer Session unterbrechen kann. Auf diese Weise wird die ständige Anfrage nach Zugriffsrechten unterbunden und die damit verbundene ungewünschte Auslastung von Ressourcen verhindert (ein Beispiel hierfür wäre eine Denial-of-Service Attacke (DOS)).

A.6.4. Abrechnung

Bei der Abrechnung werden sicherheitsrelevante Informationen auf allen AAA-Geräten mitprotokolliert. Somit wird ein detaillierter Verlauf von allen Vorgängen aufgezeichnet, der es ermöglicht im Fehlerfall eine Analyse durchzuführen und eine Nichtabstreitbarkeit von Aktionen zu garantieren.

Echtzeitabrechnung

Die AAA-Richtlinien fordern, dass Ereignisse auf den verteilten Komponenten synchron mitgeschnitten werden müssen. Durch eine geeignete Zeitsynchronisierung ergeben sich somit Aufzeichnungen mit derselben Zeitbasis für alle Operationen im Netz. Die Genauigkeit der Zeitbasis sollte sich im Bereich von Sekunden bewegen.

Verpflichtung zur kompakten Kodierung

Diese Anforderung beschreibt, dass nicht zu viel Daten-Overhead durch die Abrechnungsmitschnitte entstehen soll. Auf diese Weise sollen die eventuell begrenzten Ressourcen des Netzes und die Rechenleistung von Geräten geschont werden.

Erweiterbarkeit der Abrechnungsmitschnitte

Des Weiteren soll es möglich sein, diese Mitschnitte durch eigene Datenfelder zu erweitern. Gerade bei der dynamischen Erweiterung durch neue Funktionalitäten sollte diese bedacht werden, um so hinsichtlich der Mitschnitte genügend Flexibilität bieten zu können.

Gebündelte Abrechnung

Das AAA-System soll in der Lage sein, mehrere Abrechnungseinträge zu puffern oder zu speichern, um diese gebündelt zu übertragen, wenn wenig Verkehr im Netz vorhanden ist. Dies soll helfen während Auslastungsphasen des Netzes Übertragungskapazitäten zu sparen.

Garantierte Zustellung

Mit dieser Anforderung wird festgelegt, dass es auf Anwendungsebene Bestätigungen für Nachrichten geben soll. Diese werden gesendet wenn der Empfänger eine Nachricht empfangen hat und die Verantwortung für die Verarbeitung der Nachrichteninformationen übernommen hat. Dies soll eine zuverlässige Übertragung für Abrechnungsinformationen ermöglichen, falls dies nicht genug durch die darunterliegenden Protokollschichten gewährleistet wird.

Zeitstempel für die Abrechnung

Die AAA-Richtlinien legen fest, dass die Zeit zusammen mit dem Ereignis protokolliert werden muss. Die Ereignisse beschreiben Vorgänge wie Einloggen, Ausloggen, Authentifizierung, Autorisierung und generelle Sicherheitsoperationen. So kann eine chronologische Abfolge der Ereignisse rekonstruiert werden. Im Falle von Ausfällen kann dies die Fehlersuche unterstützen.

Dynamische Abrechnung

Diese Richtlinie hält fest, dass das AAA-System dynamische Authentifizierungs- und Autorisierungsvorgänge mitschneiden können muss. Das bedeutet, dass es während der Durchführung eines Vorgangs möglich sein muss, mehrere Abrechnungseinträge zu erstellen. So kann nicht nur eine Art von Ereignis mitprotokolliert werden, sondern Informationen verschiedener Ursprünge, wie z.B. nutzerdefinierte, sicherheits- oder anwendungsbasierte Ereignisse.

Bibliografie

Veröffentlichungen des Autors

[DEH+11] DRÖSSLER, SEBASTIAN, MICHAEL EICHHORN, STEFAN HOLZKNECHT, BERND MÜLLER-RATHGEBER, HOLM RAUCHFUSS, MICHAEL ZWICK, ERWIN BIEBL, KLAUS DIEPOLD, JÖRG EBERSPÄCHER, ANDREAS HERKERSDORF, WALTER STECHELE, ECKEHARD STEINBACH, RAYMOND FREYMANN, KARL-ERNST STEINBERG und HANS-ULRICH MICHEL: *An RT-Capable Virtualized Automotive ICT Infrastructure*. In: *Advances in Real-Time Systems, Herausgeber: Samarjit Chakraborty*. Springer Verlag, erscheint 2011.

[EPBS11] EICHHORN, MICHAEL, MARTIN PFANNENSTEIN, RAINER BODENDORFER und ECKEHARD STEINBACH: *A Novel Multimedia Session Management Approach for In-Vehicle Middleware based on DPWS*. In: *3rd International Workshop on Multimodal Interfaces for Automotive Applications (MIAA) in conjunction with IUI 2011, the International Conference on Intelligent User Interfaces*, Palo Alto, Kalifornien, USA, Februar 2011.

[EPMS10] EICHHORN, MICHAEL, MARTIN PFANNENSTEIN, DANIEL MUHRA und ECKEHARD STEINBACH: *A SOA-based Middleware Concept for In-vehicle Service Discovery and Device Integration*. In: *IEEE Intelligent Vehicles Symposium*, San Diego, USA, Juni 2010.

[EPS10] EICHHORN, MICHAEL, MARTIN PFANNENSTEIN und ECKEHARD STEINBACH: *A Flexible In-Vehicle HMI Architecture based on Web Technologies*. In: *2nd International Workshop on Multimodal Interfaces for Automotive Applications (MIAA) in conjunction with IUI 2010, the International Conference on Intelligent User Interfaces.*, Hong Kong, China, Februar 2010.

[ESS08] EICHHORN, MICHAEL, MARTIN SCHMID und ECKEHARD STEINBACH: *A Realtime Streaming Architecture for In-Car Multimedia: Design Guidelines and Prototypical Implementation*. In: *Proceedings of the IEEE International Conference on Vehicular Electronics and Safety (ICVES) 2008*, Columbus, Ohio, USA, September 2008.

[HEP+11] HANKA, OLIVER, MICHAEL EICHHORN, MARTIN PFANNENSTEIN, JÖRG EBERSPÄCHER und ECKEHARD STEINBACH: *A Distributed Public Key Infrastructure based on Threshold Cryptography for the HiiMap Next Generation Internet Architecture*. Future Internet, Vol. 2011, Nr. 3:14–30, 2011.

[MREM08a] MÜLLER-RATHGEBER, BERND, MICHAEL EICHHORN und HANS-ULRICH MICHEL: *A unified Car-IT Communication-Architecture: Design Guidelines and prototypical Implementation*. In: *2008 IEEE Intelligent Vehicles Symposium (IV 2008)*, Eindhoven, The Netherlands, Juni 2008.

[MREM08b] MÜLLER-RATHGEBER, BERND, MICHAEL EICHHORN und HANS-ULRICH MICHEL: *A unified Car-IT Communication-Architecture: Network Switch Design Guidelines*. In: *Proceedings of the IEEE International Conference on Vehicular Electronics and Safety (ICVES) 2008*, Columbus, Ohio, USA, September 2008.

[SSE+10] SCHROTH, GEORG, FLORIAN SCHWEIGER, MICHAEL EICHHORN, ECKEHARD STEINBACH, MICHAEL FAHRMAIR und WOLFGANG KELLERER: *Video Synchronization using Bit Rate Profiles.* In: *IEEE International Conference on Image Processing (ICIP 10)*, Hong Kong, China, September 2010.

[ZETK06] ZÖLS, STEFAN, MICHAEL EICHHORN, ANTHONY TARLANO und WOLFGANG KELLERER: *Content-based Hierarchies in DHT-based Peer-to-Peer-Systems*. In: *Proceedings of 2nd Workshop on Next Generation Service Platforms for Future Mobile Systems (SPMS 2006) in conjunction with the 2006 International Symposium on Applications and the Internet (SAINT 2006)*, Phoenix, Arizona, USA, Januar 2006.

Allgemeine Veröffentlichungen

[BGM07] BOTTARO, A., A. GÉRODOLLE und S. MARIÉ: *Combining OSGi technology and Web Services to realize the plug-n-play dream in the home network*. OSGi Community Event, Munich, Germany, 2007.

[BHM+04] BOOTH, D., H. HAAS, F. MCCABE, E. NEWCOMER, M. CHAMPION, C. FERRIS und D. ORCHARD: *Web services architecture.* W3C Working Group Note, Vol. 11, 2004.

[Boh09] BOHN, H.: *Web Service Composition for Embedded Systems: WS-BPEL Extension for DPWS*, Seiten 19–49. Sierke, 2009.

[Bow01] BOWLER, T.: *Mac OS X.* Personal Computer World, Vol. 24, Nr. 5, 2001.

[CCK+05] CHAN, S., D. CONTI, C. KALER, T. KUEHNEL, A. REGNIER, B. ROE, D. SATHER, J. SCHLIMMER, H. SEKINE, J. THELIN et al.: *Devices profile for web services*. Microsoft Developers Network Library, 2005.

[CHM00] CALHOUN, P., T. HILLER und P. MCCANN: *Criteria for Evaluating AAA protocols for Network Access.* IETF RFC2989, November 2000.

[CLB08] CHOLLET, S., P. LALANDA und A. BOTTARO: *Transparently adding security properties to service orchestration*. 22nd International Conference on Advanced Information Networking and Applications, 2008.

[CLV+03] CONINX, K., K. LUYTEN, C. VANDERVELPEN, J. VAN DEN BERGH und B. CREEMERS: *Dygimes: Dynamically generating interfaces for mobile computing devices and embedded systems*. Lecture notes in computer science, Seiten 256–270, 2003.

[CM08] CRANE, D. und P. MCCARTHY: *Comet and Reverse Ajax: The Next Generation Ajax 2.0*. Springer, 2008.

[DJMZ05] DOSTAL, W., M. JECKLE, I. MELZER und B. ZENGLER: *Service-orientierte Architekturen mit Web Services*. Elsevier, Spektrum, Akad. Verl., 2005.

[DR08] DIERKS, T. und E. RESCORLA: *RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2*. Internet Engineering Task Force, 2008.

[Erl05] ERL, T.: *Service-oriented architecture: concepts, technology, and design*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2005.

[G+05] GARRETT, J.J. et al.: *Ajax: A new approach to web applications*. 2005.

[Har01] HAROLD, E.R.: *XML bible*. Hungry Minds, 2001.

[Hay09] HAYES, J.: *Smart drivers-[IT in-vehicle]*. Engineering & Technology, 4(11):59, 2009.

[HC04] HALL, R.S. und H. CERVANTES: *An OSGi implementation and experience report*. In: *IEEE Consumer Communications and Networking Conference, CCNC 2004*, Seiten 394–399, Las Vegas, Nevada, USA, January 2004.

[Hem06] HEMEL, A.: *Universal Plug and Play: Dead simple or simply deadly?* In: *5th System Administration and Network Engineering Conference, Delft, The Netherlands*, 2006.

[HKRS07] HITZLER, P., M. KRÖTZSCH, S. RUDOLPH und Y. SURE: *Semantic Web: Grundlagen*. Springer-Verlag New York Inc, 2007.

[HLP+09] HERNANDEZ, V., L. LOPEZ, O. PRIETO, J. F. MARTINEZ, A. B. GARCIA und A. DASILVA: *Security Framework for DPWS Compliant Device*. 3rd International Conference on Emerging Security Information, Systems and Technologies, 2009.

[HY07] HE, J. und I.L. YEN: *Adaptive user interface generation for web services*. In: *IEEE International Conference on e-Business Engineering, 2007. ICEBE 2007*, Seiten 536–539, 2007.

[HYP+08] HE, J., I.L. YEN, T. PENG, J. DONG und F. BASTANI: *An Adaptive User Interface Generation Framework for Web Services*. In: *Proceedings of the 2008 IEEE Congress on Services Part II*, Seiten 175–182. IEEE Computer Society Washington, DC, USA, 2008.

[IR09] IACONO, L. und H. RAJASEKARAN: *Secure browser-based access to web services*. In: *IEEE International Conference on Communications (ICC)*. IEEE, 2009.

[ISO07] ISO 11898: *Road vehicles – Controller area network (CAN) – Part 1-5*. ISO, Geneva, Switzerland, 2003-2007.

[IT03] ITU-T: *X.805: Security architecture for systems providing end-to-end communications*. 2003.

[JS04] JALICS, L. und F. SZCZUBLEWSKI: *AMI-C Content-Based Human Machine Interface (HMI)*. SAE SP, Seiten 15–20, 2004.

[LR06] LAHRES, B. und G. RAÝMAN: *Praxisbuch Objektorientierung*. Galileo Computing, 2006.

[LR09] LAM, G. und D. ROSSITER: *A SOAP-Based Streaming Content Delivery Framework for Multimedia Web Services*. In: *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE*, Seiten 1097–1102. IEEE, 2009.

[M+] MELZER, I. et al.: *Service-orientierte Architekturen mit Web Services*. Konzepte-Standards-Praxis, Vol. 2.

[MLH+08] MARTINEZ, J.-F., M. LOPEZ, V. HERNANDEZ, K. JEAN-MARIE, A. B. GARCIA, L. LOPEZ, C. HERRERA und C. J. SANCHEZ-ALARCOS: *A security architectural approach for DPWS-based devices*. CollECTeR Iberoamérica, 2008.

[MN05] MÜNZ, S. und W. NEFZGER: *HTML Handbuch*. Franzis Verlag, 2005.

[NL04] NEWCOMER, E. und G. LOMOW: *Understanding SOA with Web Services (Independent Technology Guides)*. Addison-Wesley Professional, 2004.

[PZL08] PAUTASSO, C., O. ZIMMERMANN und F. LEYMANN: *Restful web services vs."big"web services: making the right architectural decision*. International World Wide Web Conference, Proceeding of the 17th international conference on World Wide Web, 2008.

[RW98] RICHARDSON, T. und K.R. WOOD: *The rfb protocol*. ORL, Cambridge, January, 1998.

[SG86] SCHEIFLER, R.W. und J. GETTYS: *The X window system*. ACM Transactions on Graphics (TOG), Vol. 5, Nr. 2, Seiten 79–109, 1986.

[She87] SHERIDAN, T.B.: *Supervisory control*. Handbook of human factors, Seiten 1243–1268, 1987.

[SKLS05] SONG, H., D. KIM, K. LEE und J. SUNG: *UPnP-based sensor network management architecture*. In: *Proc. International Conference on Mobile Computing and Ubiquitous Networking*, 2005.

[SSV05] SETH, A., H.J. SU und J.M. VANCE: *A desktop networked haptic VR interface for mechanical assembly*. In: *Proceedings of IMECE*, Seiten 1–8, 2005.

[Swe00] SWEENEY, M.: *BUS: a Browser based User interface Service for Web based Applications*. In: *User Interface Conference, 2000. AUIC 2000. First Australasian*, Seiten 103–109, 2000.

[Tei01] TEIRIKANGAS, J.: *HAVi: Home Audio Video Interoperability*. Helsinki: University of Technology, 2001.

[V+97] VINOSKI, S. et al.: *CORBA: Integrating diverse applications within distributed heterogeneous environments*. IEEE Communications Magazine, Vol. 35, Nr. 2, Seiten 46–55, 1997.

[War04] WARKUS, M.: *The official GNOME 2 developer's guide*. No Starch Press Reading, MA, 2004.

[ZLF+09] ZHANG, S., X. LIANG, J. FU, J. CHENG und W. GENG: *A perceptual user interface for motion retrieval*. In: *Machine Learning and Cybernetics, 2009 International Conference on*, Band 4, Seiten 2211–2216. IEEE, 2009.

Zitierte Webseiten

[Ado10] ADOBE: *Einführung in Flex: Erste Schritte.* http://www.adobe.com/de/devnet/flex/quickstart/coding_with_mxml_and_actionscript/, letzter Zugriff Mai 2010.

[Apa10a] APACHE SOFTWARE: *Thrift*. http://incubator.apache.org/thrift/, letzter Zugriff Mai 2010.

[Apa10b] APACHE SOFTWARE FOUNDATION: *Etch*. https://cwiki.apache.org/ETCH/, letzter Zugriff Mai 2010.

[Apa10c] APACHE SOFTWARE FOUNDATION: *Java Intelligent Network Infrastructure (Jini)*. http://incubator.apache.org/river/doc/arch2_0.html, letzter Zugriff Mai 2010.

[Atv10] ATVISE: *WebMI, The HMI & SCADA Revolution.* http://www.atvise.com/, letzter Zugriff Mai 2010.

[BKK+10] BEATTY, JOHN, GOPAL KAKIVAYA, DEVON KEMP, THOMAS KUEHNEL, BRAD LOVERING, BRYAN ROE, CHRISTOPHER ST. JOHN, GUILLAUME SIMONNET JEFFREY SCHLIMMER, DOUG WALTER, JACK WEAST, YEVGENIY YARMOSH und PRASAD YENDLURI: *Web Service Dynamic Discovery (WS-Discovery)*. http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf, letzter Zugriff November 2010.

[BMW10] BMW GROUP: *BMW Group PressClub Global*. https://www.press.bmwgroup.com/pressclub/p/pcgl/startpage.htm, letzter Zugriff Mai 2010.

[Con10] CONTINENTAL: *AutoLinQ*. http://www.conti-online.com/generator/www/de/de/continental/automotive/themes/passenger_cars/interior /connectivity/autolinq/pi_autolinq_de.html, letzter Zugriff Mai 2010.

[Das10] DAS BUNDESMINISTERIUM DES INNERN: *Der neue Personalausweis*. http://www.personalausweisportal.de, letzter Zugriff Dezember 2010.

[Dig10] DIGITAL LIVING NETWORK ALLIANCE: *Digital Living Network Alliance (DLNA)*. http://www.dlna.org, letzter Zugriff November 2010.

[FFm10] FFMPEG PROJECT: *FFmpeg*. http://www.ffmpeg.org, letzter Zugriff Dezember 2010.

[Goo10a] GOOGLE INC.: *Chrome Experiments - Not your mother's JavaScript*. http://www.chromeexperiments.com/, letzter Zugriff Dezember 2010.

[Goo10b] GOOGLE INC.: *Google Android*. http://code.google.com/android/, letzter Zugriff Mai 2010.

[Goo10c] GOOGLE INC.: *Protocol Buffers Developer Guide*. http://code.google.com/apis/protocolbuffers/docs/overview.html, letzter Zugriff Mai 2010.

[Hei10] HEISE: *Continental setzt auf Android im Auto*. http://www.heise.de/open/meldung/Continental-setzt-auf-Android-im-Auto-182420.html, letzter Zugriff Mai 2010.

[His10] HISTORY OF ECONOMICS PLAYGROUND: *Zotero and scholarship for historians*. http://historyofeconomics.wordpress.com/2008/12/09/zotero-and-scholarship-for-historians/, letzter Zugriff September 2010.

[HS10] HENNING, MICHI und MARK SPRUIELL: *Distributed Programming with Ice*. http://www.zeroc.com/doc/Ice-3.4.1-IceTouch/manual/, letzter Zugriff November 2010.

[Hym10] HYMAN, PAUL: *On demand games offer new revenue stream*. http://www.msnbc.msn.com/id/5388739, letzter Zugriff November 2010.

[Ico10] ICONICS: *WebHMI, Web-Based Real-time Automation Software.* http://www.iconics.com/products/webhmi.asp, letzter Zugriff Mai 2010.

[IET10] IETF: *Hypertext Transfer Protocol, HTTP/1.1.* http://tools.ietf.org/html/rfc2616, letzter Zugriff Mai 2010.

[ITU10a] ITU-T RECOMMENDATION: *X.500 - The Directory: Overview of concepts, models and services*. http://www.itu.int/itudoc/itu-t/aap/sg17aap/history/x500/index.html, letzter Zugriff November 2010.

[ITU10b] ITU-T RECOMMENDATION: *X.509 - Public Key Infrastructure (PKI) for single sign-on (SSO) and Privilege Management Infrastructure (PMI)*. http://www.itu.int/itudoc/itu-t/aap/sg17aap/history/x509/index.html, letzter Zugriff November 2010.

[ITU11] ITU: *Draft ITU-T Recommendation X.805*. https://datatracker.ietf.org/documents/LIAISON/file835.pdf, letzter Zugriff Januar 2011.

[Khr10] KHRONOS GROUP: *WebGL - OpenGL ES 2.0 for the Web*. http://www.khronos.org/webgl/, letzter Zugriff Dezember 2010.

[Lin10a] LINUX FOUNDATION: *MeeGo*. http://meego.com/, letzter Zugriff Mai 2010.

[Lin10b] LINUX FOUNDATION: *moblin.org*. http://moblin.org/, letzter Zugriff Mai 2010.

[Mic10a] MICROSOFT CORPORATION: *Understanding the Remote Desktop Protocol (RDP)*. http://support.microsoft.com/?scid=kb%3Ben-us%3B186607&x=13&y=11, letzter Zugriff Mai 2010.

[Mic10b] MICROSOFT CORPORATION: *Übersicht über XAML*. http://msdn.microsoft.com/de-de/library/ms752059.aspx, letzter Zugriff Mai 2010.

[MJB+01] MITTON, D., M. ST. JOHNS, S. BARKLAY, D. NELSON, B. PATIL, M. STEVENS und B. WOLFF: *Authentication, Authorization and Accounting: Protocol Evaluation*. IETF RFC3127, Juni 2001.

[Moz10] MOZILLA: *XUL (XML User Interface Language)*. https://developer.mozilla.org/en/XUL, letzter Zugriff Mai 2010.

[Nok10a] NOKIA: *Maemo*. http://maemo.nokia.com/, letzter Zugriff Mai 2010.

[Nok10b] NOKIA CORPORATION: *Qt - A cross-platform application and UI framework*. http://qt.nokia.com/products/, letzter Zugriff Dezember 2010.

[OAS10a] OASIS STANDARD: *Device Profile for Web Services Version 1.1*. http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf, letzter Zugriff November 2010.

[OAS10b] OASIS STANDARD SPECIFICATION: *Web Service Security: SOAP Message Security 1.1 (WS-Security 2004)*. http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMeassageSecurity, letzter Zugriff November 2010.

[Obj10] OBJECT MANAGEMENT GROUP: *Interface Description Language Syntax and Semantics chapter*. http://www.omg.org/cgi-bin/doc?formal/02-06-07, letzter Zugriff Mai 2010.

[Ope10] OPEN MOBILE ALLIANCE: *WAP Forum*. http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html, letzter Zugriff Mai 2010.

[otUF10] UPNP FORUM, CONTRIBUTING MEMBERS OF THE: *Universal Plug and Play (UPnP) Spezifikation*. http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf, letzter Zugriff Mai 2010.

[Pal10] PALM: *WebOS Developer Center*. http://developer.palm.com/, letzter Zugriff Mai 2010.

[QNX10] QNX SOFTWARE SYSTEMS: *QNX Neutrino RTOS*. http://www.qnx.com /products/neutrino_rtos/, letzter Zugriff Mai 2010.

[rde10] RDESKTOP: *rdesktop: A Remote Desktop Protocol Client*. http://www.rdesktop.org/, letzter Zugriff Mai 2010.

[Ron10] RONNEBURG, FRANK: *Midnight Commander*. http://debiananwenderhandbuch.de/mc.html, letzter Zugriff April 2010.

[Shi10] SHINGLEDECKER, ROBERT: *Tiny-/Microcore - Toolkit for Linux*. http://www.tinycorelinux.com, letzter Zugriff Dezember 2010.

[Sit10] SITRA, NILO: *SOAP Version 1.2 Part 0: Primer*. http://www.w3.org/TR/2003/REC-soap12-part0-20030624, letzter Zugriff November 2010.

[SOA10] SOA4D: *SOA4D Forge: DPWS Core*. https://forge.soa4d.org /projects/d-pwscore/, letzter Zugriff Dezember 2010.

[Sun10a] SUN: *JavaServer Pages Technology.* http://java.sun.com/products/jsp/, letzter Zugriff Mai 2010.

[Sun10b] SUN: *Remote Method Invocation (RMI), Documentation*. http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp, letzter Zugriff Mai 2010.

[The10a] THE APACHE SOFTWARE FOUNDATION: *The Apache HTTP Server Project*. http://httpd.apache.org/, letzter Zugriff Dezember 2010.

[The10b] THE LIGHTTPD DEVELOPER GROUP: *lighttpd fly light*. http://www.lighttpd.net/, letzter Zugriff Dezember 2010.

[The10c] THE SCONS FOUNDATION: *SCons: A software construction tool*. http://www.scons.org/, letzter Zugriff Dezember 2010.

[The10d] THE TNTNET DEVELOPMENT TEAM: *High Performance Web Services in C++*. http://www.tntnet.org/, letzter Zugriff Dezember 2010.

[The10e] THE WEBKIT COMMUNITY: *The WebKit Open Source Project*. http://webkit.org/, letzter Zugriff Dezember 2010.

[Tig10] TIGHTVNC: *TightVNC Software.* http://www.tightvnc.com/, letzter Zugriff Mai 2010.

[W3C10a] W3C: *Scalable Vector Graphics*. http://www.w3.org/Graphics/SVG/, letzter Zugriff Mai 2010.

[W3C10c] W3C: *XHTML 1.0 The Extensible HyperText Markup Language.* http://www.w3.org/TR/xhtml1/, letzter Zugriff Mai 2010.

[X. 10] X. ORG FOUNDATION: *X Window System*. http://www.x.org, letzter Zugriff Mai 2010.

[X C10] X CONSORTIUM: *Scientific Calculator for X*. http://www.thelinuxblog.com/linux-man-pages/1/xcalc, letzter Zugriff Mai 2010.

[XML10] XML11: *XML11*. http://www.xml11.org, letzter Zugriff Mai 2010.