Aus dem Institut für Telematik
der Universität zu Lübeck

Direktor:
Prof. Dr. rer. nat. Stefan Fischer

Technologien für eine interoperable und automatisierte Vernetzung von medizinischen IT-Systemen

Inauguraldissertation
zur
Erlangung der Doktorwürde
der Universität zu Lübeck

Aus der Sektion Informatik / Technik

Vorgelegt von
Herrn Dipl.-Inf. David Gregorczyk
aus Lübeck

Lübeck 2014

Erster Berichterstatter: Prof. Dr. rer. nat. Stefan Fischer

Zweiter Berichterstatter: PD Dr.-Ing. Christian Werner

Tag der mündlichen Prüfung: 14.07.2014
Zum Druck genehmigt. Lübeck, den 15.07.2014

Kurzfassung

Bis heute hat sich die Interoperabilität zwischen medizinischen Geräten im Krankenhaus nur unzureichend durchsetzen können. Das im Unternehmensumfeld etablierte Konzept einer Service-orientierten Architektur (SOA) ist ein Ansatz, Interoperabilitätsprobleme zu lösen. Bei SOA handelt es sich jedoch um ein Paradigma, das lediglich einen systematischen Weg zur Etablierung von Interoperabilität definiert. Für die praktische Umsetzung kommen je nach Anwendungsgebiet unterschiedliche Technologien zum Einsatz.

Es hat sich herauskristallisiert, dass die auf offenen Standards basierende Web-Service-Technologie zur Lösung von Interoperabilitätsproblemen zwischen medizinischen Geräten geeignet ist. Hierfür wird ein standardisiertes Web-Service-Profil, das Devices Profile for Web Services (DPWS), genutzt. Mit Hilfe des DPWS kann eine Plug-and-Play-fähige Netzwerkumgebung aus IT-Systemen beliebiger Art konstruiert werden.

Die Konnektivität zwischen medizinischen Geräten auf Basis von Web-Services ist effektiv und effizient umsetzbar. Konnektivität allein schafft jedoch keinen Mehrwert für den Operationssaal der Zukunft. In dieser Arbeit werden daher Anwendungsprotokolle und Methoden für eine interoperable und automatisierte Vernetzung von medizinischen Geräte- und IT-Systemen vorgestellt. Zunächst erfolgt die Beobachtung und formalisierte Beschreibung von Problemstellungen aus dem Klinikumfeld. Die Problemstellungen liefern Anforderungen für weiterführende Arbeiten.

Um kontinuierliche oder periodisch wiederkehrende Vitalparameter effizient übertragen zu können, wird eine für das DPWS angepasste Multicast-Transport-Bindung entwickelt und prototypisch umgesetzt.

Eine Anforderung vieler medizinischer Geräte ist außerdem die Abfrage von Patientendaten aus einem Krankenhausinformationssystem. Diese Abfrage erfolgt heutzutage üblicherweise über Punkt-zu-Punkt-Verbindungen und muss für jedes Gerät manuell durchgeführt werden. In dieser Arbeit wird ein Protokoll zur robusten und automatisierten Verteilung der Patientendaten vorgestellt.

Abschließend erfolgt die Realisierung und Bewertung eines Video-Live-Streaming-Verfahrens zur Einblendung bildgebender Systeme auf Tablet-PCs und Smartphones. Hiermit soll das Klinik-Personal zukünftig in der Lage sein, Video-Streams über das Netzwerk auf beliebige Endgeräte zu transportieren. Das Live-Streaming kann außerdem als Schulungswerkzeug eingesetzt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Abkürzungsverzeichnis

ACR	American College of Radiology ... 2
ADT	Admission, Discharge and Transfer ... 80
AE	Application Entity ... 81
API	Application Programming Interface ... 25
ARPANET	Advanced Research Projects Agency NET ... 1
BGA	Blutgasanalyse ... 48
CConOps	Clinical Concept of Operations ... 41
CLDC	Connected Limited Device Configuration ... 54
CORBA	Common Object Request Broker Architecture ... 15
CSS	Cascading Style Sheets ... 16
DCPS	Data-Centric-Publish-Subscribe ... 63
DDS	Data Distribution Service for Real-time Systems ... 57
DDS	Data Distribution Service ... 57
DICOM	Digital Imaging and Communications in Medicine ... 2
DIM	Domain Information Model ... 3
DNS	Domain Name System ... 32
DPWS	Devices Profile for Web Services ... 12
DSJ	DirectShow Java-Wrapper ... 114
DTD	Document Type Definition ... 19
EHR	Electronic Health Record ... 79
EXI	Efficient XML Interchange ... 74
FDA	Food and Drug Administration ... 6
FTP	File Transfer Protocol ... 17
GOP	Group of Picture ... 118
GUI	Graphical User Interface ... 55
HL7	Health Level 7 ... 2
HTML	Hypertext Markup Language ... 16
HTTP	Hypertext Transfer Protocol ... 16
HTTPS	HTTP Secure ... 19
ICE Standard	Integrated Clinical Environment Standard ... 35
IDL	Interface Definition Language ... 16
IETF	Internet Engineering Task Force ... 16
IGMP	Internet Group Management Protocol ... 72
IHE	Integrating the Healthcare Enterprise ... 81
IIOP	Internet Inter-Orb Protocol ... 17

IP	Internet Protocol . . . 1
ITI-TF	IT Infrastructure Technical Framework . . . 85
ITU	Internationalen Fernmeldeunion . . . 103
JEDI	Java Event-based Distributed Infrastructure . . . 66
JMEDS	Java Multi Edition Stack . . . 54
JMS	Java Message Service . . . 17
JMS	Java Message Service . . . 17
KIS	Krankenhausinformationssystem . . . 1
LAN	Local Area Network . . . 51
LIS	Laborinformationssystem . . . 44
MD PnP	Medical Device „Plug-and-Play“ Interoperability Program . . . 35
MDD	Medical Device Directive . . . 4
MEP	Message Exchange Pattern . . . 21
MJPEG	Motion JPEG . . . 104
MLD	Multicast Listener Discovery . . . 72
MOWS	Management of Web Services . . . 19
MPBetreibV	Medizinprodukte-Betreiberverordnung . . . 6
MPEG	Moving Picture Experts Group . . . 102
MPG	Medizinproduktegesetz . . . 1
MPSV	Medizinprodukte-Sicherheitsplanverordnung . . . 6
MRT	Magnetresonanztomographie . . . 37
MTU	Maximum Transmission Unit . . . 55
MUWS	Management Using Web Services . . . 19
NACK	Negative Acknowledgement . . . 75
NEMA	National Electrical Manufacturers Association . . . 2
OASIS	Organization for the Advancement of Structured Information Standards . . . 16
OMG	Object Management Group . . . 63
OPMS	OP-Management-System . . . 39
PACS	Picture Archiving and Communication System . . . 38
PDC	Patient Demographics Consumer . . . 87
PDQ	Patient Demographics Query . . . 84
PDS	Patient Demographics Supplier . . . 87
PMA	Pre-market Approval . . . 7
PMN	Pre-market Notification . . . 7
QBP	Query By Parameter . . . 80
QoS	Quality of Service . . . 29
RIM	Reference Information Model . . . 3
RIS	Radiologieinformationssystem . . . 38
RMI	Remote Method Invocation . . . 15
RPC	Remote Procedure Call . . . 15
RTCP	Real-Time Control Protocol . . . 101
RTMP	Real Time Messaging Protocol . . . 102

RTP	Real-Time Transport Protocol	101
RTSP	Real-Time Streaming Protocol	101
SAML	Security Assertion Markup Language	33
SA-WSDL	Semantic Annotations for WSDL and XML Schema	19
SCP	Service Class Provider	81
SCU	Service Class User	81
SGML	Standard Generalized Markup Language	16
SMTP	Simple Mail Transfer Protocol	17
SMTT	Smart Monitor Timing Tool	113
SOA	Service-orientierte Architektur	9
SOA4D	Service-oriented Architectures for Devices	54
SOAP	Simple Object Access Protocol	16
SODA	Service-oriented Device Architecture	37
SOP-UID	Service Object Pair Class Unique Identifier	81
STS	Security Token Service	32
TCP	Transmission Control Protocol	1
TLS	Transport Layer Security	19
UDDI	Universal Description, Discovery and Integration	17
UDP	User Datagram Protocol	1
UKSH	Universitätsklinikum Schleswig-Holstein	35
UPnP	Universal Plug and Play	26
URI	Uniform Resource Identifier	15
URL	Uniform Resource Locator	25
UUID	Universally Unique Identifier	73
VOD	Video-on-Demand	100
W3C	World Wide Web Consortium	16
WAN	Wide Area Network	61
WLAN	Wireless LAN	77
WS4D	Web Services for Devices	54
WS-BPEL	Web Services Business Process Execution Language	19
WSDAPI	Web Services on Devices Application Programming Interface	54
WSDL	Web Service Description Language	16
WS-I	WS Interoperability Organization	26
WWW	World Wide Web	1
XML	Extensible Markup Language	16

Kapitel 1.

Einleitung

Es sind nunmehr über 40 Jahre vergangen, seitdem das amerikanische Verteidigungsministerium mit der Entwicklung des *Advanced Research Projects Agency NET (ARPANET)*, dem Vorgänger des heutigen Internet, begann und infolgedessen das *Internet Protocol (IP)*, *Transmission Control Protocol (TCP)* und *User Datagram Protocol (UDP)* zur Vernetzung von IT-Systemen entwickelt wurden. Inzwischen haben sich die Technologien zu einem festen Bestandteil des täglichen Lebens gewandelt, z. B. bei der Mobilfunkkommunikation, der Hausautomation oder ganz einfach bei der Benutzung des *World Wide Web (WWW)*. Im Hinblick auf aktuelle Forschungsbemühungen [33] ist zu erkennen, dass die Vernetzung von IT-Systemen auch in Zukunft einen wichtigen Stellenwert in der Gesellschaft einnehmen wird.

Neben den typischen Diensten für Privathaushalte existieren zahlreiche Einsatzmöglichkeiten der vernetzten Computersysteme in der Wirtschaft. Beispielsweise erfolgt die Buchhaltung in größeren Unternehmen im Allgemeinen über vernetzte Computersysteme, Industrieanlagen sind mit Hilfe von Fernwartungssoftware steuerbar und durch Online-Shops ist ein komplett neuer Markt erschlossen worden.

Die Möglichkeit des Datenaustausches über IT-Netzwerke macht sich auch das Gesundheitswesen zu Nutze. In Krankenhäusern werden Patientendaten typischerweise elektronisch erfasst und über ein *Krankenhausinformationssystem (KIS)* zu Abrechnungs-, Diagnose- und Therapiezwecke genutzt. Serversysteme speichern diese Daten und verbreiten sie über verschiedene Schnittstellen. Neben der abteilungsübergreifenden Vernetzung entstehen Kommunikationsbedürfnisse mit den Laboren, Operationssälen und den Intensivstationen. Zahlreiche Medizingeräte unterstützen dabei das diensthabende Personal. Unter einem Medizingerät versteht man in diesem Zusammenhang beliebige, elektronisch betriebene Geräte, die für diagnostische oder therapeutische Zwecke im medizinischen Bereich eingesetzt werden. Eine formalisierte, übergeordnete Definition liefert das *Medizinproduktegesetz (MPG)* mit dem Begriff *Medizinprodukt* [10, § 3 Begriffsbestimmungen].

Bisher sind Medizingeräte typischerweise als Insellösungen konzipiert. Eine Informationsabfrage aus dem KIS oder gegenseitige Wechselwirkungen von Geräten unterschiedlicher Hersteller finden nur in unzureichendem Ausmaß statt. Dabei ist die

Vernetzung von Medizingeräten aktueller als je zuvor. Waren vor 50 Jahren noch einige wenige rudimentäre Vorrichtungen während einer Operation im Einsatz, sind es heute hochmoderne Maschinen für die Anästhesie und Blutgasanalyse, elektronische Schneidinstrumente, Operationsmikroskope, 3D-Navigations- und Endoskopiegeräte als auch mobile Röntgengeräte. Die Operateure, Anästhesisten und Assistenten sind zunehmend mit der Bedienung der Geräte und der Beobachtung von Monitoren beschäftigt, so dass es wünschenswert ist, mithilfe eines adäquaten Informationsaustauschs das Krankenhauspersonal bei der Arbeit zu entlasten, um damit in letzter Konsequenz auch die Patientensicherheit zu erhöhen.

1.1. Interoperable medizinische Geräte

Die Idee der interoperablen Medizingerätekommunikation ist keinesfalls neu oder revolutionär. Es gibt viele verschiedene Standards und Produkte, die Interoperabilität propagieren. Hierbei stellt sich zunächst die Frage nach der Bedeutung des Begriffes *Interoperabilität*. Dieser stammt aus dem Lateinischen und setzt sich zusammen aus den Wörtern *inter* für *zwischen* und *opera* für *Arbeit*. Im Duden wird die allgemeine Bedeutung des Wortes *Interoperabilität* als „[Wieder]herstellung einer Einheit [aus Differenziertem]“ und „Vervollständigung“ [25] beschrieben. Auf IT-Systeme bezogen spricht man von Interoperabilität, wenn unabhängige, heterogene Systeme die Fähigkeit besitzen, hinsichtlich physikalischer als auch virtueller Schnittstellen verwertbare Informationen auszutauschen. Lesh et al. [74] führen Interoperabilität als kontinuierliche Größe ein und stellen fest, dass sie auf verschiedenen Komplexitätsebenen existieren kann. Die niedrigste Komplexität stellen hierbei z. B. physikalische Steckverbindungen dar, die höchste Komplexität wird IT-Systemen bei semantischer Interaktion abverlangt.

Drei in der Medizintechnik bekannte Standards zur Herstellung von Interoperabilität sind *Digital Imaging and Communications in Medicine (DICOM)* [22], *Health Level 7 (HL7)* [23] und die ISO/IEEE 11073 [64].

DICOM ist ein vom *American College of Radiology (ACR)* und der *National Electrical Manufacturers Association (NEMA)* entwickelter offener Standard für medizinisches Bilddaten-Management und spezifiziert Bildformate und Protokolle für einen interoperablen Datenaustausch zwischen medizinischen Anwendungen. Eine Einführung in die Thematik bietet die Diplomarbeit von Schöch [120]. Die Spezifikation umfasst neben der Codierung für zweidimensionale Bilddaten auch eindimensionale Signalverläufe, dreidimensionale Bildinformationen und Videos. Darüber hinaus existiert die Möglichkeit, OP-Planungsdaten und ärztliche Befundberichte auszutauschen. Insgesamt ergibt sich so eine Spezifikation von über 4500 Seiten (Stand 2012). Da kein Medizingerät alle Datenformate und Dienste umsetzt, werden stattdessen sogenannte

DICOM-Conformance-Statements definiert, welche die Unterstützung bestimmter Merkmale des Standards ausweisen.

DICOM findet typischerweise bei der Datenakquisition durch bildgebende Systeme und der Bildverarbeitung Anwendung; einrichtungsweite Informationssysteme unterstützen heute üblicherweise HL7. HL7 steht sowohl für eine Gruppe von Standards zum interoperablen Datenaustausch in medizinischen IT-Landschaften als auch der gleichnamigen Organisation, die diese Standards für das Gesundheitswesen entwickelt. Im Gegensatz zu DICOM bietet HL7 jedoch nicht die Möglichkeit, OP-Planungsdaten zu verwalten. *HL7, Inc.* wurde in den USA gegründet mit dem Ziel, eine nationale Industrienorm zu definieren. Inzwischen werden die Normen auch auf internationaler Ebene erarbeitet, in Deutschland beispielsweise vertreten durch *HL7 Deutschland e. V.* [42]. Die Standards sollen die medizinischen Prozessabläufe vereinfachen und Interoperabilität zwischen verschiedenen Herstellern und Betreibern gewährleisten. Die zwei wesentlichen Ausprägungen des HL7-Standards sind die Versionen 2 und 3. Version 2 beschreibt einen pragmatischen Ansatz für den Nachrichtenaustausch, um anstehende Kommunikationsbedürfnisse effizient erfüllen zu können. Das Datenformat beschränkt sich auf einfach strukturiertem Text ähnlich dem CSV-Format [61]. Die ausgetauschten Nachrichten sind somit durch Menschen kaum interpretierbar, wodurch sich die Wartung von HL7-Software-Produkten sehr schwierig gestaltet. Darüber hinaus führt das lose Format häufig zu semantischen Inkonsistenzen. Mit Version 3 des Standards wurde ein umfassendes *Reference Information Model (RIM)* entwickelt, das die Objekte und Prozesse im Krankenhaus präzise abbildet. Dank der Nutzung eines offenen, erweiterbaren Datenformats lassen sich ausgetauschte Nachrichten besser nachvollziehen und integrieren. HL7 Version 3 ist jedoch in seiner Komplexität kaum beherrschbar und die Migration von HL7 Version 2 sehr kostenintensiv. In der Praxis wird es heute noch nicht produktiv eingesetzt.

Die beiden vorhergehenden Standards orientieren sich im Wesentlichen an der Kommunikation zwischen einzelnen Abteilungen im Krankenhaus; direkte Geräte-zu-Geräte-Kommunikation im OP-Saal oder auf der Intensivstation ist nicht vorgesehen. Zu diesem Zweck wurde die ISO/IEEE-11073-Standardfamile entwickelt, welche ein ganzheitliches Konzept für den Datenaustausch zwischen verschiedenen medizinischen Geräten abbildet. Die Standardfamilie ist in sieben Standardserien unterteilt, von denen die drei wichtigsten Serien die ISO/IEEE 11073-1xxxx, -2xxxx und -3xxxx sind. Die 1xxxx-Serie beschreibt die Basis für alle anderen Teile des Standards, d. h. Sprachelemente, Semantiken und das sogenannte *Domain Information Model (DIM)*, ein objektorientiertes Datenmodell zur Modellierung virtueller Geräteabbildungen und Nomenklaturen. Der zweite Teil bildet die drei anwendungsorientierten Ebenen des ISO/OSI-Schichtenmodells ab und beschreibt den logischen Nachrichtenaustausch zwischen Medizingeräten unter Berücksichtigung zukünftiger Entwicklungen. Die Schichten für den Nachrichtentransport und physikalische Verbindungen sind in der ISO/IEEE 11073-3xxxx-Serie beschrieben. Es existieren Spezifikationen sowohl für kabelgebundene als auch kabellose Verbindungen. Die ISO/IEEE 11073 ist eine sehr

komplexe Standardfamilie und wird nur im unzureichendem Ausmaß von Herstellern unterstützt. Darüber hinaus entsprechen die Transportprotokolle nicht mehr dem Stand der Technik. Eine umfassende Einführung von Schmitt et al. kann in [123] nachgelesen werden.

Neben den oben beschriebenen Standards gibt es auch Produktlösungen für die Kommunikation von Medizingeräten untereinander und mit dem KIS. Dazu zählen z. B. komplett eingerichtete OP-Säle wie der EndoALPHA von Olympus [89] oder der OR.1 von Storz [72]. Diese Lösungen sind durch die exakte Abstimmung aller Komponenten zwar interoperabel, allerdings nur so lange keine Produkte anderer Hersteller verwendet werden. Die Kommunikationsprotokolle sind proprietär und lassen deshalb keine herstellerübergreifenden Gerätekombinationen zu.

1.2. Regulatorische Rahmenbedingungen

Wenn Medizinprodukte hergestellt und in Verkehr gebracht werden, sind zahlreiche Richtlinien und Normen einzuhalten. Dies gilt insbesondere auch für die Software, die auf Medizingeräten betrieben wird. In diesem Abschnitt werden die Grundlagen zu den regulatorischen Rahmenbedingungen medizinischer Software sowie die gegenwärtigen Perspektiven für den Einsatz in vernetzten Medizingeräte-Systemen erläutert.

Die Rechtslage in Europa ist im Wesentlichen durch drei Richtlinien gekennzeichnet:

- der Medizinprodukterichtlinie 93/42/EWG, in englisch als *Medical Device Directive (MDD)* bezeichnet,
- der Richtlinie 98/79/EG über In-vitro-Diagnostika und
- der Richtlinie 90/385/EWG über aktive implantierbare medizinische Geräte.

Die 98/79/EG ist die jüngste der drei Richtlinien und behandelt die regulatorischen Bedingungen für Laborgeräte. In der 90/385/EWG sind entsprechend die Regelungen für aktive implantierbare Medizingeräte wie z. B. Herzschrittmacher festgehalten. Die MDD behandelt vorwiegend die Ermöglichung des freien Warenverkehrs von Medizinprodukten in Europa. Darüber hinaus enthält sie die grundlegenden Anforderungen, um ein technisches Mindestniveau für alle auf dem europäischem Markt befindlichen Medizinprodukte zu gewährleisten.

Um die grundlegenden Anforderungen zu erfüllen, müssen sich Hersteller von Medizinprodukten nach den Grundsätzen der integrierten Sicherheit richten. Das bedeutet unter anderem, eine Risikominimierung durchzuführen, indem die vertriebenen Produkte sicher ausgelegt und Schutzmaßnahmen integriert werden. Außerdem müssen die Medizinprodukthersteller die Anwender über ein mögliches Restrisiko informieren.

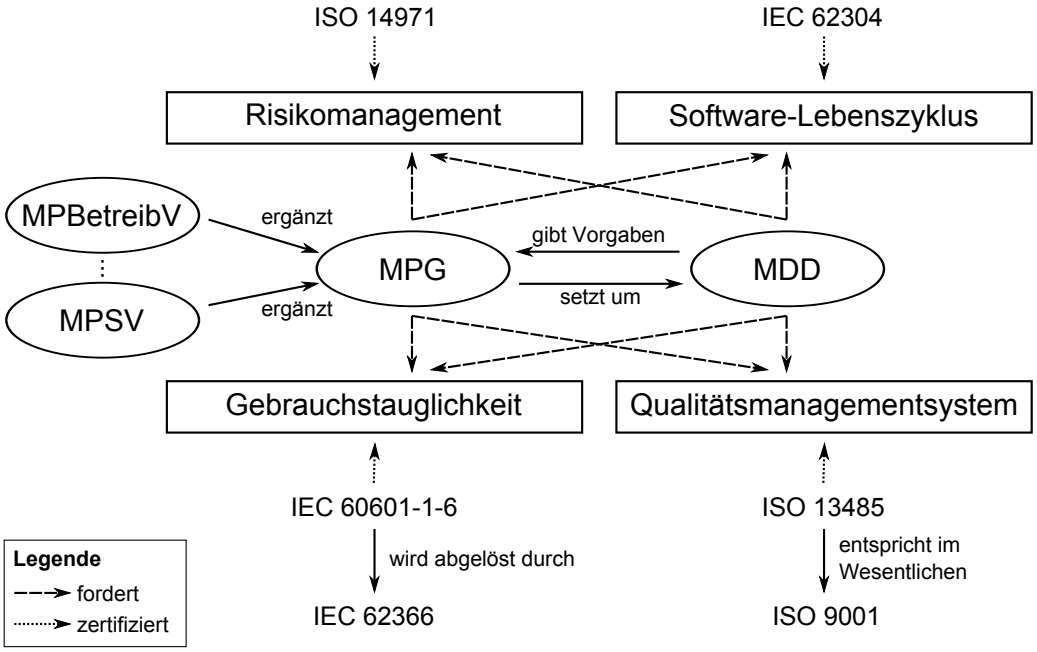


Abbildung 1.1.: Regulatorische Landkarte für medizinische Software. In Anlehnung an [70, S. 10].

Da nicht von allen Medizinprodukten das gleiche Risiko für einen Menschen ausgeht, sind im MDD die Risikoklassen I, IIa, IIb und III festgesetzt. Von Klasse-I-Produkten geht hierbei das geringste und von Klasse-III-Produkten das größte Risiko aus. In Abhängigkeit der Risikoklasse ändert sich der Aufwand für die Aktivitäten im Rahmen eines Konformitätsbewertungsverfahrens. Mit diesem Verfahren bestätigt ein Hersteller, dass sein Produkt die grundlegenden Anforderungen aus der MDD erfüllt. Wie das Verfahren anzuwenden ist, wird in den Anhängen der Richtlinie erläutert. Aufgrund der Komplexität kommt für medizinische Software üblicherweise die Anwendung des Anhang II in Frage. Hierbei hat der Hersteller ein vollständiges Qualitätssicherungssystem zu führen. Dieses System muss durch Audits einer benannten Stelle nach ISO 13485 zertifiziert sein.

In Deutschland wird die MDD durch das *Medizinproduktegesetz (MPG)* in nationales Recht umgesetzt. Abbildung 1.1 skizziert den Zusammenhang zwischen der MDD, dem MPG und deren regulatorischen Anforderungen. Gemäß der Richtlinie bzw. des Gesetzes muss ein Medizinproduktehersteller Qualitätsaspekte berücksichtigen, ein Risikomanagement durchführen, Software-Lebenszyklus-Prozesse anwenden und einen Nachweis über die Gebrauchstauglichkeit erbringen. Für alle vier Aspekte existieren europäische Normen, die bezüglich der MDD harmonisiert sind.

Zusätzlich zur MDD wird das MPG durch acht Verordnungen ergänzt, z. B. durch die *Medizinprodukte-Betreiberverordnung (MPBetreibV)* und die *Medizinprodukte-Sicherheitsplanverordnung (MPSV)*.

Die EN ISO 13485 zertifiziert das Qualitätsmanagement und entspricht einer für Medizinproduktehersteller angepassten Norm der allgemeinen Qualitätsnorm ISO 9001. Mit der ISO 13485 werden Mindestanforderungen an das Design, die Entwicklung, Produktion, Installation sowie Instandhaltung eines Medizinproduktes gesetzt.

Die EN ISO 14971 regelt die Anwendung eines Risikomanagements für Medizinprodukte. Sie legt ein Verfahren fest, um die Risikobeherrschung über den gesamten Lebenszyklus eines Medizinproduktes zu etablieren. Hierbei gilt es das Risiko, welches von einem Medizinprodukt ausgeht, zu minimieren. Etwaige Restrisiken können gegen Abwägung mit dem Nutzen für den Patienten in Kauf genommen werden.

Anhand der EN 62304 kann der Software-Lebenszyklus zertifiziert werden. Die Norm ist speziell für Medizingeräte-Software vorgesehen. In dieser prozessorientierten Norm werden alle Aktivitäten für die Durchführung eines Software-Lebenszyklus beschrieben. Diese umfassen die Planung, Analyse, den Architekturentwurf, das detaillierte Design, die Implementierung, Software-Integration, Prüfung, Software-Freigabe sowie den Wartungsprozess. Die Wahl des Vorgehensmodells kann vom Hersteller bestimmt werden. Demnach ist es dem Hersteller freigestellt, ob er z. B. nach dem V-Modell oder Scrum entwickelt.

Mit der EN 62366 bzw. EN 60601-1-6 wird die Gebrauchstauglichkeit zertifiziert. Die EN 62366 ist dabei die jüngere Norm und hat ihren Ursprung in der EN 60601-1-6. Die beiden Normen legen fest, welche Anforderungen bezüglich der Analyse, Spezifikation, Entwicklung, Verifizierung und Validierung der Gebrauchstauglichkeit an den Hersteller gestellt werden. Der Anwendungsbereich beschränkt sich dabei ausschließlich auf den bestimmungsgemäßen Gebrauch und Benutzerfehler. Sabotage ist nicht Teil der Betrachtung, wobei der vorhersehbare Missbrauch berücksichtigt werden muss. Eine detaillierte Zusammenfassung der Normen und Zusammenhänge mit dem MDD kann in [70] nachgelesen werden.

Ein weiterer integraler Bestandteil bei der Entwicklung medizinischer Software und der erste Schritt für die Erfüllung der rechtlichen Vorgaben ist die Definition der Zweckbestimmung und des bestimmungsgemäßen Gebrauchs. Hiermit wird unter anderem definiert, wie das Produkt die Diagnose, Therapie oder Überwachung von Krankheiten, Verletzungen oder physiologischer und anatomischer Parameter unterstützt. Anhand der Zweckbestimmung lässt sich zum einen schlussfolgern, ob es sich bei einem entwickelten Produkt überhaupt um ein Medizinprodukt handelt. Zum anderen wird anhand der Zweckbestimmung außerdem die Klassifizierung und das Konformitätsbewertungsverfahren bestimmt.

Am amerikanischen Markt wird die Inverkehrbringung von Medizinprodukten durch die *Food and Drug Administration (FDA)* kontrolliert. Ähnlich wie in der MDD sind

Medizingeräte nach Klassen aufgeteilt, wobei Klasse-I-Geräte mit dem geringsten und Klasse-III-Geräte mit dem größten Risiko für den Patienten einhergehen. Die FDA unterscheidet im Wesentlichen zwei Arten der Zulassung. Die *Pre-market Approval (PMA)* ist für alle nicht klassifizierten und die meisten Klasse-III-Produkte vorgesehen. Sie erfordert einen validen wissenschaftlichen Nachweis hinsichtlich der Wirksamkeit als auch der Sicherheit eines Medizingerätes und ist somit sehr aufwändig.

Für alle anders klassifizierten Geräte, die keine PMA erfordern, muss bei der FDA eine 510(k) *Pre-market Notification (PMN)* eingereicht werden. Hierbei wird das neue Medizingerät mit einem bereits durch die FDA zugelassenen Medizingerät verglichen und gegebenenfalls benötigte Nachweise über zusätzliche Anforderungen beigefügt. Nach der Einreichung des 510(k)-PMN-Antrags erhält der Hersteller von der FDA eine Gültigkeitsbescheinigung und kann sein Gerät damit bei der FDA registrieren. Nach der Registrierung darf das Medizinprodukt in den USA vertrieben werden. Für viele Geräte der ersten Risikoklasse ist lediglich die Einhaltung der *Good Manufacturing Practice* und eine Registrierung bei der FDA erforderlich.

Das durch die EN ISO 14971 zertifizierte Risikomanagement ist für Hersteller gedacht, die *ein* Medizinprodukt in Verkehr bringen wollen. Dabei kann das Produkt aus mehreren Einzelkomponenten bestehen, es muss jedoch ein Gesamt-Restrisiko vom Hersteller verantwortet werden. Medizinprodukte in einem IT-Netzwerk zu betreiben, das nicht vom Hersteller kontrolliert wird, ist somit nicht möglich, so dass eine dynamische Vernetzung von Medizingeräten rechtliche Probleme hervorruft. Eine nachträgliche Änderung am System, die möglicherweise sogar mit einer Änderung der Zweckbestimmung einhergeht, würde den erneuten Durchlauf der Konformitätsbewertung erfordern. Um einen Verbund von mehreren Geräten zu betreiben, müsste also ein Gesamtsystem zugelassen werden, in dem jeder Ge- und Missbrauch mit jedem am Netzwerk teilnehmenden Gerät berücksichtigt ist.

Es gibt zwar Systeme, in denen Medizingeräte verschiedener Hersteller in einem IT-Netzwerk zusammen betrieben werden. Hierbei handelt es sich jedoch um feste Komplettsysteme, in denen ein dedizierter Hersteller für die Integration verantwortlich ist. Eine nachträgliche Änderung oder die Einbindung von Geräten anderer Hersteller ist aus rechtlicher Sicht nicht ohne weiteres möglich.

Um dennoch medizinische Geräte verschiedener Hersteller in einem IT-Netzwerk betreiben zu können, das nicht unter der Kontrolle eines Herstellers liegt, ist in 2010 die internationale Norm IEC 80001-1 verabschiedet worden. Diese Norm wird nach dem ersten Inverkehrbringen von Medizinprodukten angewandt, wenn es keinen einzelnen Hersteller gibt, der die Gesamtverantwortung für die Konformität und das Restrisiko übernimmt. Sie richtet sich damit vornehmlich an die Betreiber von Medizinprodukten und beschreibt die Aufgaben und Verantwortlichkeiten für den sicheren Betrieb eines *medizinischen IT-Netzwerkes*. Ein wesentlicher Teil besteht darin, dass die IEC 80001-1 einen Risikomanagement-Prozess für den gesamten Lebenszyklus des IT-Netzwerkes fordert. Die Gerätehersteller müssen den Betreibern hierfür die

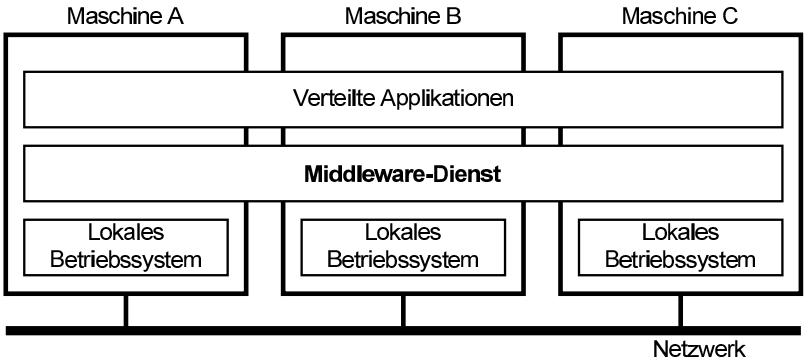


Abbildung 1.2.: Strukturierung verteilter Systeme mit Middleware-Technologien. Die Middleware wird zwischen der Anwendung und dem Betriebssystem angeordnet. Sowohl die Anwendungs- als auch Middleware-Schicht erstreckt sich logisch über alle verteilten Anwendungen. Entnommen aus [129].

notwendigen Restrisiko-Informationen bereitstellen, die über das Risikomanagement der EN ISO 14971 hinausgehen. Die IEC 80001-1 ist eine junge Norm, die bisher kaum in der Praxis angewandt wird. Sie geht mit einem erhöhten Administrationsaufwand einher, womit die Einführung neuer Technologien verlangsamt wird und nicht mehr so flexibel auf Kundenwünsche eingegangen werden kann [2]. Aufgrund dieses Umstandes wird sich erst noch zeigen, ob sie für den Produktiveinsatz geeignet ist und von den Medizinprodukteherstellern und Krankenhausbetreibern angenommen wird.

1.3. Architekturen für verteilte Systeme

Nach Tanenbaum und van Steen ist ein verteiltes System „eine Menge [> 1] voneinander unabhängiger Computer, die dem Benutzer wie ein einzelnes kohärentes System erscheinen“ [129]. In diesem Satz steckt eine ganz entscheidende Aussage: Ein Benutzer hat bei der Bedienung eines verteilten Systems den Eindruck, er hätte es mit nur einem einzigen System zu tun. Dies gilt sowohl für die Endbenutzer als auch den Anwendungsprogrammierer. Ein wesentlicher Aspekt ist demnach, dass Anwendungsprogrammierern die Komplexität des zu Grunde liegenden Netzwerkes weitgehend verborgen bleibt. Technisch löst man dies mit einer wie in Abbildung 1.2 dargestellten Software-Schicht zwischen dem Netzwerk und der Anwendung, die als Middleware bezeichnet wird. Diese Abstraktionsebene bietet dem Anwendungsentwickler eine einheitliche Programmierschnittstelle, die ihm die Implementierung von Protokollen und Datenformaten der unteren Schichten abnimmt.

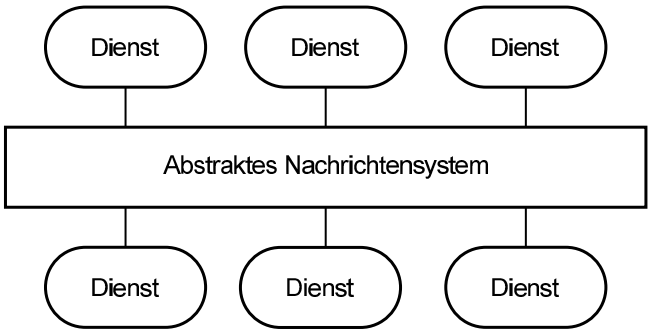


Abbildung 1.3.: Das SOA-Konzept: Dienste sind über ein abstraktes Nachrichtensystem verbunden.

Es gibt zahlreiche Architekturkonzepte für verteilte Systeme [121]. Im Unternehmensumfeld hat sich vor allem die *Service-orientierte Architektur (SOA)* durchgesetzt. Für die Medizingerätevernetzung ist dieses Konzept interessant, da es eine für Unternehmen etablierte Methodik mit zahlreichen verfügbaren und offenen Umsetzungsstandards bereitstellt.

Der Begriff SOA wurde das erste Mal 1996 vom Marktforschungsunternehmen Gartner benutzt [124] und beschreibt ein Integrationskonzept, bei dem *virtuelle* Dienste (engl. Services) in einem verteilten System über wohldefinierte Schnittstellen zur Verfügung gestellt werden [93]. Die Dienste sind an ein abstraktes Nachrichtensystem (z. B. Netzwerk oder Kommunikationsserver) wie in Abbildung 1.3 gebunden, so dass ein Informationsaustausch zwischen den Diensten ermöglicht wird. Das SOA-Konzept stammt aus dem Unternehmensumfeld und wurde entwickelt, um die dort meist gewachsenen, heterogenen IT-Netzwerke zu vereinheitlichen und besser auf Änderungen in den Geschäftsprozessen reagieren zu können. Ziel der SOA ist die Gewährleistung der Interoperabilität über Plattform- und Unternehmensgrenzen hinweg [121].

Im Mittelpunkt stehen deshalb die prozessorientierte Modellierung und die Integration heterogener Systeme. Mit Prozessorientierung ist in diesem Zusammenhang die Hintereinanderausführung von Diensten gemeint. Fachliche Funktionalität wird in atomare Einheiten zerteilt und durch Komposition auf Geschäftsprozesse abgebildet. Abbildung 1.4 illustriert die Komposition anhand eines Beispiels. Ändern sich die Geschäftsprozesse, lassen sich einzelne betroffene Dienste unter der Bedingung, dass sie lose gekoppelt sind, austauschen. Unter der Prämisse, dass das SOA-Konzept konsequent eingehalten wird, ist bei einer Änderung der Prozesse lediglich eine lokale Restrukturierung des Systems erforderlich.

Man unterscheidet zwei verschiedene Ansätze bei der Komposition von Diensten. Der erste Ansatz wird als *Orchestrierung* bezeichnet und beschreibt die Zusammensetzung von mehreren Basisdiensten zu einem komplexeren neuen Dienst. Nach außen hin

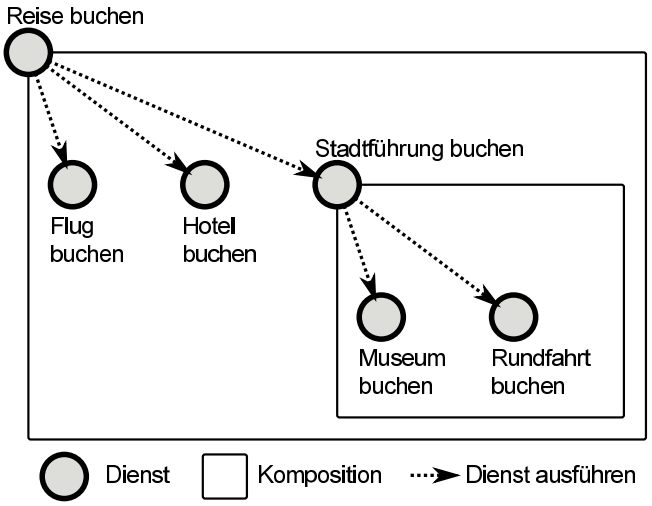


Abbildung 1.4.: Dienstekomposition am Beispiel einer Buchung im Reisebüro. Die Buchung setzt sich aus verschiedenen Prozessen zusammen, wobei Prozesse aus Unterprozessketten bestehen können.

präsentiert sich der orchestrierte Dienst jedoch als ein weiterer gewöhnlicher Basisdienst [121]. Im Kontrast zur Orchestrierung steht die *Choreographie*, bei der Dienste zu einem Geschäftsablauf anhand zulässiger Nachrichtenabfolgen zwischen einem oder mehreren Kommunikationspartnern zusammengesetzt werden [117].

Neben der Orchestrierung und Choreographie spielt auch die Integration heterogener IT-Landschaften eine wichtige Rolle im SOA-Kontext. Große Unternehmen verfügen häufig über Software-Systeme, die zum Teil mehrere Jahrzehnte im Einsatz sind und mit modernen Varianten zu einer ganzheitlichen Lösung kombiniert werden sollen. In einem solchen Fall ist es sinnvoll, die Altsysteme als Dienste in eine SOA zu integrieren, indem Software-Wrapper die proprietären Kommunikationsschnittstellen in standardisierte, wohldefinierte SOA-Schnittstellen übersetzen.

Die Realisierung einer SOA wird durch die drei Rollen *Dienstanbieter*, *Dienstnutzer* und *Dienstverzeichnis* charakterisiert. Zusammen mit den vier Interaktionen *veröffentlichen*, *finden*, *binden* und *ausführen* sind sie im SOA-Rollenmodell in Abbildung 1.5 illustriert. Der Dienstanbieter stellt zunächst seine Funktionalität in Form von Diensten zur Verfügung, indem er seine Dienstbeschreibung in einem Dienstverzeichnis publiziert (1). Dieser Schritt ist notwendig, da Dienste andernfalls nicht auffindbar wären. Interessierte Dienstnutzer können im Dienstverzeichnis suchen und sich die Referenz auf entsprechende Dienste geben lassen (2). Danach kann die Dienstbeschreibung vom Dienstanbieter über die Referenz abgerufen und so eine Bindung zwischen Nutzer und Anbieter hergestellt werden (3). Anschließend ist der Nutzer mithilfe der Dienstbeschreibung in der Lage, den Dienst auszuführen (4).

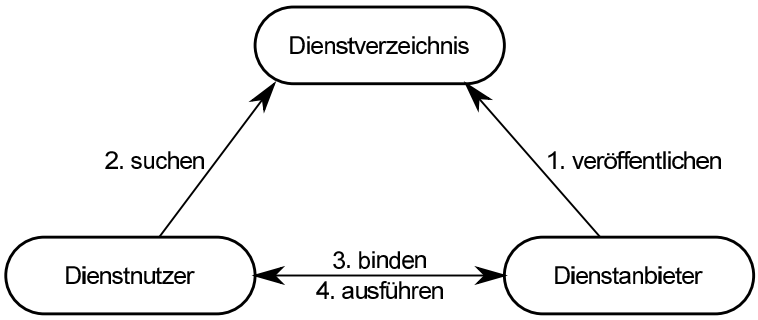


Abbildung 1.5.: Das SOA-Rollenmodell mit den drei Akteuren Dienstanbieter, Dienstnutzer und Dienstverzeichnis.

1.4. Motivation

Im Jahr 2007 untersuchte das Unternehmen Kaiser Permanente die Kostenentwicklung für die Integration medizinischer Geräte [71]. Dem Ergebnis zufolge beträgt der Anteil dieser Integration 40 Prozent der Gesamtkosten eines Gerätes. Das sind jährlich etwa 40 Millionen US-Dollar. Ein Standard für die Gerätekommunikation kann gemäß der Analyse die Kosten um bis zu 30 Prozent senken, was einer Einsparung von jährlich 12 Millionen US-Dollar entspricht. In Deutschland betrugen die Gesamtkosten der Krankenhäuser 2011 83, 4 Milliarden Euro. Die Kosten stiegen 2012 auf 86, 8 Milliarden Euro an. Es ist ein zunehmender Kostendruck für die Betreiber zu verzeichnen, dem durch eine adäquate Vernetzung medizinischer IT-Systeme und Geräte begegnet werden kann [128].

Interoperabilität zwischen medizinischen Geräten kann nicht nur die Kosten für die Krankenhausbetreiber reduzieren. Unterstützende Assistenzsysteme, gegenseitige Überwachung und automatisierte Wechselwirkungen zwischen Geräten erhöhen zudem die Patientensicherheit und entlasten das Krankenhauspersonal bei der Arbeit.

Nach Lesh et al. existiert zwischen medizinischen Geräten praktisch keine Interoperabilität [74]. Diese Aussage aus dem Jahr 2007 ist auch 2012 immer noch aktuell [27, S. 13f]. Es gibt zwar Standards wie HL7, DICOM und die ISO/IEEE 11073. Diese sind jedoch entweder für dedizierte Bereiche im Krankenhaus spezifiziert oder wurden von den Herstellern aufgrund ihrer Komplexität nicht angenommen. In den letzten Jahren hat es deshalb zahlreiche Forschungsbemühungen gegeben mit dem Ziel, einen Kommunikationsstandard zu konstruieren, der die Integration medizinischer Geräte vereinfacht. Ein vielversprechender Ansatz ist ein in [111] entwickeltes, SOA-basiertes Plug-and-Play-Konzept, das die Web-Service-Technologie einsetzt (vgl. Kapitel 2 ab Seite 15). Der Vorteil des vorgestellten Konzeptes ist die Verwendung offener Standards, mit denen Medizinprodukteherstellern der Zugang zur Vernetzung vereinfacht wird.

Wenn Medizingeräte heute miteinander vernetzt werden, handelt es sich üblicherweise um Komplettsysteme oder Insellösungen. Bei Komplettsystemen sind die Betreiber auf einen vordefinierten Gerätepark angewiesen, während Insellösungen teuer eingekauft werden müssen und nur in speziellen Kombinationen nutzbar sind. In beiden Varianten kommen in der Regel proprietäre Protokolle zum Einsatz, so dass sich die Betreiber von einem Zulieferer abhängig machen. Firmen wie Capsule Technologie [1] entwerfen Hardware-Komponenten und Software-Bausteine zur Vernetzung nicht kompatibler, medizinischer Geräte. Geräte mit seriellem Ausgang können mit Adapterkabeln und Modulen angebunden werden. Die speziell konstruierte *DataCaptor Device Interfaces Library* übersetzt proprietäre Protokolle von mehr als 500 unterschiedlichen Medizingeräten. Die konvertierten Protokolle werden jedoch nicht über offene Schnittstellen zur Verfügung gestellt, sondern lediglich in eigenen Applikationen verwertet.

1.5. Zielsetzung und Aufbau

Die Möglichkeit, Konnektivität zwischen beliebigen Medizingeräten herzustellen, hat als alleinstehendes Merkmal keinen Mehr- und Marktwert. Konnektivität ohne Anwendungsbereiche ist sowohl für Hersteller als auch Betreiber nur ein weiterer Kostenfaktor. Um einen Mehrwert zu schaffen und die interoperablen Technologien aus wirtschaftlicher Sicht interessant zu gestalten, ist die Definition von Anwendungsfällen und exakten technischen Prozessen erforderlich.

Zahlreiche Konnektivitätskonzepte wurden entwickelt, jedoch ohne Berücksichtigung von Anwendungsfällen. Somit existieren diverse Plattformen, mit denen Konnektivität zwischen Medizingeräten hergestellt werden kann. Da jedoch keine Anwendungsprozesse definiert sind, kann Interoperabilität nur auf syntaktischer Ebene zugesichert werden. Eine formalisierte Beschreibung der Anwendungsfälle durch z. B. Protokolle ist nicht existent.

Ziel dieser Arbeit ist es, Anwendungsfälle zu erschließen und deren Effektivität in einer Machbarkeitsstudie nachzuweisen. Aus den Prozessen der Anwendungsfälle werden Protokolle abgeleitet, mit denen Medizingeräte zukünftig in der Lage sind, einen Mehrwert für das klinische Personal zu schaffen. Der Aufbau dieser Arbeit lautet wie folgt.

Zunächst werden in Kapitel 2 Web-Service-Grundlagen erläutert. Als Erstes stehen hierbei die drei Basistechnologien, mit denen sich das SOA-Paradigma abbilden lässt, im Fokus. Dem schließen sich weitere Elemente aus dem Web-Service-Technologiestapel an. Maßgebend für die Kommunikation zwischen Geräten ist das nachfolgend erläuterte *Devices Profile for Web Services (DPWS)*, welches im Wesentlichen Plug-and-Play und Publish-Subscribe in lokalen Netzwerken unterstützt. Das Kapitel schließt mit einer Einführung in ein für medizinische Software taugliches DPWS ab.

In Kapitel 3 werden zahlreiche Anwendungsfälle und deren Nutzen für das klinische Personal beschrieben. Die Szenarien basieren hierbei auf den Beobachtungen von OP-Abläufen sowie Befragungen von Chirurgen, Anästhesisten und Assistenten. Die Anwendungsfälle liefern konkrete Probleme, die es zu lösen gilt. Das Kapitel wird mit einer Machbarkeitsstudie abgeschlossen, die ein System für die Vernetzung und die erschlossenen Anwendungsfälle prototypisch präsentiert.

Kapitel 4 führt eine Erweiterung des im DPWS verankerten Standards WS-Eventing ein. Ziel ist es, eine Multicast-Bindung zu definieren, um Ereignisnachrichten effizient im Netzwerk zu verteilen. Ein wichtiges Ziel der Erweiterung ist die Abwärtskompatibilität mit der ursprünglichen WS-Eventing-Spezifikation. Die Multicast-Bindung kann dazu genutzt werden, Web-Service-gestützte Streaming-Formate einzuführen. Solch ein Verfahren eignet sich z. B. für die Übertragung von kontinuierlichen Vitalparametern.

Ein omnipräsenter Anwendungsfall bei der Integration medizinischer Geräte im OP-Saal ist die Bereitstellung von Patienteninformationen. Bisher erfolgt diese Bereitstellung an jedem Gerät manuell. Kapitel 5 beschreibt ein Protokoll für die halbautomatisierte Verarbeitung von Patientendaten. Da Patientendaten vor der Verwertung vom klinischen Personal bestätigt werden müssen, beinhaltet das Protokoll neben Mechanismen für die fehlerrobuste Zustellung auch ein Bestätigungsverfahren. Mit diesem Verfahren ist es möglich, die Patienteninformationen von jedem beliebigen Gerät aus im Netzwerk zu akkreditieren.

Der technische Fortschritt hat die Nutzung mobiler Rechnersysteme in den letzten Jahren revolutioniert. Smartphones und Tablet-PCs finden zunehmend auch im Gesundheitswesen Anwendung. In Kapitel 6 wird untersucht, ob sich Smartphones bzw. Tablet-PCs für das Video-Live-Streaming eignen, um zukünftige Anwendungen im E-Health- oder Schulungsbereich zu ermöglichen. Zunächst führt das Kapitel die aktuellen Technologien ein und beschreibt existierende Protokolle für den Einsatz von Video-Live-Streaming-Formaten. Mit Hilfe eines Hardware-Encoders wird ein Live-Stream zur Verfügung gestellt und die Übertragungslatenz zu verschiedenen mobilen Endgeräten gemessen und bewertet.

Das letzte Kapitel fasst die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf weiterführende Forschungsthemen.

Kapitel 2.

Web-Service-Grundlagen

Web-Services bilden die technische Grundlage für diese Arbeit. Nach der Definition der Wikipedia ist ein Web-Service „eine Software-Anwendung, die mit einem *Uniform Resource Identifier (URI)* eindeutig identifizierbar ist und deren Schnittstelle als XML-Artefakt definiert, beschrieben und gefunden werden kann“ [142]. Tatsächlich gestaltet sich die Definition des Begriffes als wesentlich komplexer, wenn man aktuelle Entwicklungen im Bereich verteilter Anwendungen berücksichtigen möchte. Eine allgemeingültige Erklärung für Web-Services gibt es nicht. Dies liegt zum Einen daran, dass der Begriff als solches nicht geschützt ist und sich bisher keine einheitliche Definition durchsetzen konnte. Zum Anderen existieren zwei Schlüsseltechnologien, die auf den Konzepten Service- und Ressourcen-orientierter Architekturen basieren und beide als Web-Services bezeichnet werden können: SOAP-Web-Services [152] und REST-Web-Services [30]. Thematisch behandelt diese Arbeit SOAP-Web-Services, weswegen im Folgenden mit einen Web-Service stets der SOAP-Web-Service gemeint ist.

Mit der Etablierung des IP in den achtziger Jahren [55] stieg die Anzahl verteilter Software-Systeme signifikant an. Es stellte sich heraus, dass die Handhabung reiner Socket-basierter Applikationen in heterogenen Umgebungen wie dem Internet äußerst ineffizient war. In den Folgejahren wurde die Entwicklung von Middleware-Technologien vorangetrieben, um so die im Allgemeinen auf Binärdaten basierenden Protokolle für verschiedene Betriebssysteme und Programmiersprachen leichter zugänglich zu machen.

Die Firma Xerox entwickelte 1981 als eine der Ersten mit *Courier* ein Protokoll zur Realisierung von *Remote Procedure Calls (RPCs)*. Die grundlegende Idee des Protokolls ist es, dem Anwendungsentwickler die Socket-Programmierung abzunehmen und den Zugriff auf entfernte Prozesse durch einen vorgetäuschten Funktionsaufruf zu ermöglichen. Für den Entwickler macht es bei der Programmierung keinen Unterschied mehr, ob die Ausführung einer Funktion im eigenen oder auf einem entfernten Prozess erfolgt. Diese Idee wurde in den folgenden Jahren von verschiedenen Software-Herstellern aufgegriffen und äußerte sich in offenen Spezifikationen wie Sun-RPC [54], *Common Object Request Broker Architecture (CORBA)* [75] und *Remote Method*

Invocation (RMI) [90]. Zusätzlich wurde das Konzept der *Interface Definition Language (IDL)* entwickelt. In einem IDL-Dokument kann vorab die Schnittstelle eines RPC-Servers spezifiziert und anschließend automatisch in Programm-Code übersetzt werden.

Ab Mitte der neunziger Jahre entstand parallel zu den Arbeiten an den RPC-Systemen die *Extensible Markup Language (XML)* [144]. Ziel war es, eine einfache Auszeichnungssprache zur Strukturierung beliebiger Daten zu etablieren, um der Informationsflut des rasant wachsenden WWW zu begegnen. XML ist in seiner ursprünglichen Form nicht als Datenaustauschformat für Middleware-Technologien vorgesehen gewesen. Dave Winer erkannte 1998 jedoch, dass sich XML wegen der Fähigkeit zur Modellierung beliebiger Datenstrukturen und der Plattformunabhängigkeit auch zur Serialisierung von RPC-Daten eignet. In Zusammenarbeit mit Microsoft entwickelte er XML-RPC, das 1999 konzeptuell verfeinert als *Simple Object Access Protocol (SOAP)* bei der *Internet Engineering Task Force (IETF)* zur Standardisierung eingereicht wurde. Das Fundament für SOAP-Web-Services war gelegt.

Ab 2000 wurde SOAP vom *World Wide Web Consortium (W3C)* spezifiziert und in den Folgejahren mit einer Vielzahl von Erweiterungen verfeinert. Nach und nach löste es ältere Middleware-Ansätze wie CORBA und XML-RPC ab. Heute ist SOAP ein Eigenname und wird im Zusammenhang mit einer Vielzahl an Erweiterungen und der *Web Service Description Language (WSDL)*, einer IDL für SOAP, vornehmlich als Web-Service-Technologie bezeichnet. Eine ausführliche Beschreibung zur zeitlichen Entwicklung von SOAP-Web-Services kann in [141] nachgelesen werden. Im weiteren Verlauf dieser Arbeit wird der Begriff Web-Service gemäß der Definition vom W3C [148] verwendet (übersetzt ins Deutsche): Ein Web-Service ist eine Software-Anwendung, welche die direkte Maschine-zu-Maschine-Interaktion über ein Netzwerk unterstützt. Er hat eine Schnittstelle, die in einem maschinenlesbaren Format vorliegt. Andere Anwendungen interagieren mit dem Web-Service auf Basis des maschinenlesbaren Formats, indem sie SOAP-Nachrichten austauschen. Die SOAP-Nachrichten werden typischerweise mithilfe des *Hypertext Transfer Protocol (HTTP)* und XML sowie weiteren Web-Standards transportiert.

Web-Service-Spezifikationen werden im Wesentlichen von zwei Gremien verabschiedet: dem *World Wide Web Consortium (W3C)* und der *Organization for the Advancement of Structured Information Standards (OASIS)*.

Das W3C wurde im Jahre 1994 gegründet und beschäftigt sich mit der Entwicklung von Web-Technologien. Darunter fallen z. B. die aus dem WWW bekannten Vertreter *Hypertext Markup Language (HTML)* und *Cascading Style Sheets (CSS)* als auch die Web-Service-Technologien XML, SOAP und WSDL. Die Ausarbeitung der Spezifikationen erfolgt hierbei in bestimmten Arbeitsgruppen (engl. Working-Groups).

Die OASIS existiert bereits seit 1993 und war zunächst mit der Weiterentwicklung der *Standard Generalized Markup Language (SGML)*, dem XML-Vorreiter, beschäftigt.

Nach dem Triumph von XML stand jedoch zunehmend dessen Verwertung für die technische Abwicklung von Geschäftsprozessen im Fokus der Arbeit. Eine der bekanntesten Schöpfungen ist *Universal Description, Discovery and Integration (UDDI)* (vgl. Abschnitt 2.1.3), ein Dienst zur Auffindung von Web-Services. Inzwischen erarbeitet die OASIS zahlreiche Web-Service-relevante Standards und ist insbesondere an der Erstellung von Web-Service-Profilen zur Steigerung der Interoperabilität beteiligt.

Beide Gremien erarbeiten offene Standards, die von jeder Privatperson oder kommerziellen Institution frei abrufbar sind und uneingeschränkt genutzt werden dürfen. Die freie Verfügbarkeit der Standards und die etablierten Kommunikationsmechanismen machen die Web-Service-Technologie interessant für die Medizingerätevernetzung. In den folgenden Abschnitten werden grundlegende Web-Service-Spezifikationen erläutert und in den Kontext der Gerätevernetzung gesetzt. Die Ausführungen dienen zur Vermittlung grundsätzlicher Funktionsweisen und besseren Nachvollziehbarkeit in anschließenden Kapiteln.

2.1. Der Web-Service-Technologiestapel

Die Ausgestaltung der Web-Service-Basistechnologien wird in verschiedenen Arbeitsgruppen des W3C durchgeführt. Die drei bekanntesten sind die *W3C Web Service Architecture Working Group* zur Definition des grundlegenden Web-Service-Konzeptes, die *XML Protocol Working Group* zur Definition von XML-Spezifikationen und die *Web Services Description Working Group* zur Definition von SOAP, WSDL usw.

Zentrales Resultat der *W3C Web Service Architecture Working Group* ist der Web-Service-Technologiestapel (engl. *Web Service Technology Stack*). Dieser beschreibt das Zusammenspiel verschiedener Komponenten in einer Web-Service-Middleware. Die grundsätzliche Idee ist es, von speziellen Protokollen und Technologien zu abstrahieren, so dass Web-Services in vielfältiger Weise zusammengesetzt werden können. Die Modularität birgt jedoch auch die Gefahr, dass Web-Services verschiedener Anbieter inkompatibel zueinander sind, sobald sie nicht exakt dieselben Versionen der Web-Service-Standards umsetzen. Abhilfe schafft in einem solchen Fall der nebenläufige Betrieb verschiedener Module, um die Anforderungen mehrerer Kommunikationspartner zu erfüllen.

Abbildung 2.1 illustriert die Komponenten des Web-Service-Technologiestapels. Auf der untersten Ebene ist die Kommunikationsschicht (engl. *Communication Layer*) angesiedelt. Über diese Komponente erfolgt der Nachrichtentransport zwischen zwei Kommunikationspartnern. Prinzipiell kann jedes beliebige Transportprotokoll eingesetzt werden wie z. B. HTTP, *Simple Mail Transfer Protocol (SMTP)* und *File Transfer Protocol (FTP)* als auch Lösungen wie *Internet Inter-Orb Protocol (IIOP)* oder *Java Message Service (JMS)*. Dank der strikten Trennung zwischen Serialisierung und Transport sind Web-Services unabhängig von bestehender IT-Infrastruktur.

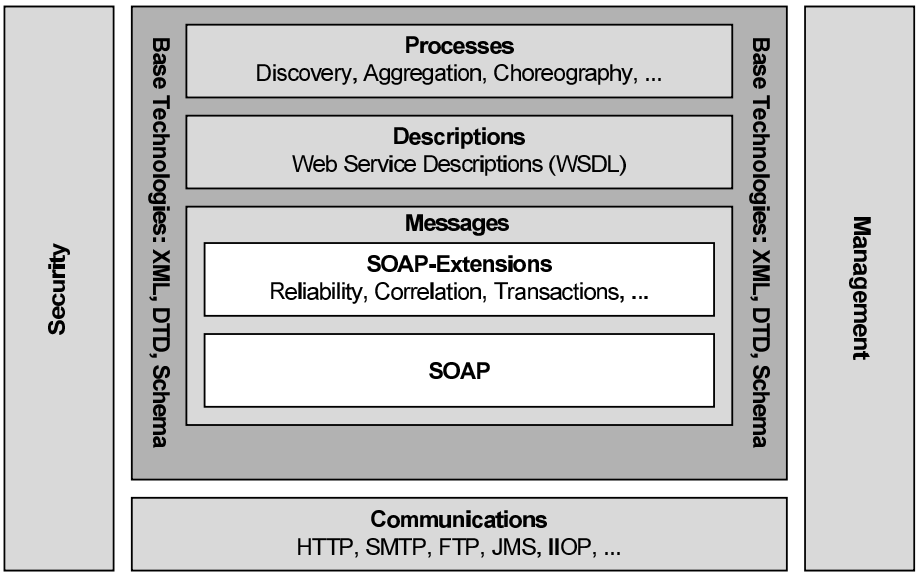


Abbildung 2.1.: Web-Service-Technologiestapel, entnommen aus [148].

Die Anpassung an spezielle Netzwerkumgebungen wird vereinfacht, da lediglich die Kommunikationsschicht angepasst werden muss.

Alle Schichten oberhalb der Kommunikationsschicht verwenden als gemeinsame Basis XML. In der Nachrichtenschicht (engl: *Message Layer*) empfiehlt das W3C die Nutzung von SOAP. Der zentrale Gedanke ist dabei, dass die XML-Serialisierung unabhängig von Programmiersprachen und Betriebssystemen eingesetzt werden kann und SOAP einen generischen Erweiterbarkeitsmechanismus definiert. Die Erweiterungen beginnen typischerweise mit einem *WS* und werden insgesamt als *WS-** bezeichnet. Drei verbreitete Kandidaten sind *WS-Addressing* [149], *WS-ReliableMessaging* [102] und *WS-AtomicTransaction* [98]. Mit WS-Adressing wird ein vom Transportprotokoll unabhängiger Adressierungsmechanismus für Web-Services beschrieben, mit dem es möglich ist, Empfänger, Absender und Nachrichten eindeutig zu kennzeichnen. WS-ReliableMessaging ermöglicht einen zuverlässigen Nachrichtentransport und garantiert beispielsweise die Zustellung einer Nachricht. Die WS-AtomicTransaction-Spezifikation definiert Transaktionssteuerung für Web-Services.

Die dritte Ebene stellt die Beschreibungsschicht dar (engl. *Description Layer*). Die Aufgabe dieser Schicht ist die exakte Dienstdefinition, so dass Dienstaufrufer nachvollziehen können, wie gültige Anfragen inklusive Eingabe- und Ausgabeparameter syntaktisch zu konstruieren sind. Die vom W3C empfohlene Dienstbeschreibungssprache ist WSDL. In einem WSDL-Dokument sind die ausführbaren Operationen, Formate der Eingabe- und Ausgabeparameter sowie Transportbindungen enthalten. Anhand der Transportbindungen bekommt ein Nutzer das Verfahren (z. B. TCP) und die phy-

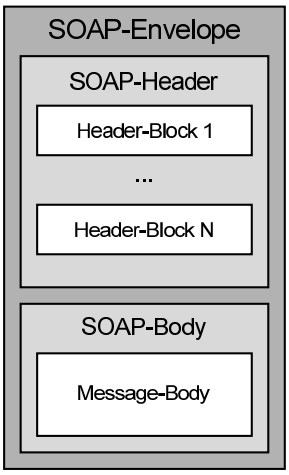
sikalischen Abrufadressen (z. B. IP) eines Dienstes mitgeteilt. Ergänzend zur WSDL gibt es verschiedene Erweiterungen für Dienstbeschreibungen wie *WS-Policy* [157] und *Semantic Annotations for WSDL and XML Schema (SA-WSDL)* [151]. Ersteres erlaubt die Definition von Anforderungen an einen Web-Service, beispielsweise Sicherheitsmerkmale. Zweiteres dient dazu, Web-Service-Beschreibungen mit semantischen Informationen anzureichern. Zusätzliche Attribute im WSDL-Dokument referenzieren semantische Modelle und erlauben die automatische Interpretation und Umformung von Ein- und Ausgabeparametern.

Die oberste Schicht wird als Prozessschicht (engl. *Process Layer*) bezeichnet. Hier werden höher angesiedelte Aufgaben wie das Auffinden oder Komponieren von Web-Services spezifiziert. Die meisten Standards aus der Prozessschicht werden von der OASIS erlassen. Dazu gehören z. B. das oben bereits erwähnte UDDI als auch die Orchestrierungssprache *Web Services Business Process Execution Language (WS-BPEL)* [99].

Die vertikalen Säulen in Abbildung 2.1 charakterisieren Bereiche, die mehrere Ebenen im Web-Service-Technologiestapel erschließen. Zentral gelegen ist XML mit seinen Grammatikbeschreibungssprachen *Document Type Definition (DTD)* [158] und XML-Schema [147], die sich über alle Nachrichten-basierten Ebenen erstrecken. DTD ist hier jedoch nur der Vollständigkeit wegen aufgenommen worden. Es wurde von XML-Schema weitestgehend verdrängt und spielt für die Praxis keine Rolle mehr. Links und rechts im Web-Service-Technologiestapel befinden sich außerdem die Bereiche *Security* und *Management*.

Sicherheit stellt man typischerweise auf Transportebene mit *HTTP Secure (HTTPS)* [57] basierend auf der *Transport Layer Security (TLS)* [62] oder auf Nachrichtenebene mit *WS-Security* [96] her. WS-Security vereint verschiedene XML-basierte Sicherheitstechnologien wie *XML Signature* [159] und *XML Encryption* [146]. Darüber hinaus bauen zahlreiche weitere Standards wie *WS-Trust* [104], *WS-SecureConversation* [103] und *WS-Federation* auf WS-Security auf. WS-Trust spezifiziert dabei die Rolle eines Vermittlers von *Security-Tokens* und WS-SecureConversation ermöglicht die Herstellung eines gemeinsamen sicheren Kommunikationskontextes. Mit WS-Federation können Echtheit, Integrität und Vertraulichkeit zwischen institutionsübergreifenden Kommunikationspartnern hergestellt werden.

Die Säule *Management* des Technologiestapels adressiert die Verwaltung von Web-Services hinsichtlich des zugrundeliegenden IT-Netzes. Dazu gehören verschiedene Fähigkeiten wie die Kontrolle und Überwachung von Web-Services sowie die Aufzeichnung von Nutzungsstatistiken. Wesentlicher Hintergrund ist dabei die Feststellung der Verfügbarkeit und die Erkennung und Behebung von Leistungsengpässen. Die OASIS hat hierzu zwei umfassende Spezifikationen zusammengestellt, bezeichnet als *Management of Web Services (MOWS)* [94] und *Management Using Web Services (MUWS)* [95]. Die Verwaltungsaufgaben werden selbst als Dienst bereitgestellt und sind nicht auf Web-Services beschränkt. Es wäre z. B. auch denkbar, Geräte in einem Netzwerk zu überwachen.




```
<?xml version="1.0" ?>
<env:Envelope
 xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
   <hb:sampleheader xmlns:hb="http://example.org/hdrblock"
    env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
    env:mustUnderstand="true">
     <!-- Header-XML -->
   </hb:sampleheader>
   <!-- weitere Header -->
  </env:Header>
  <env:Body>
   <mb:samplebody xmlns:mb="http://example.org/msgbody">
    <!-- Body-XML -->
   </mb:samplebody>
  </env:Body>
</env:Envelope>
```

Abbildung 2.2.: Links: schematischer Aufbau einer SOAP-Nachricht. Rechts: exemplarische XML-Serialisierung. In Anlehnung an [141].

Bisher wurden die Eckpfeiler des Web-Service-Technologiestapels beschrieben. Im Folgenden wird auf die drei wesentlichen Web-Service-Basistechnologien eingegangen, mit denen das Konzept einer SOA umgesetzt werden kann.

2.1.1. SOAP

SOAP ist das vom W3C empfohlene Protokoll für den plattformunabhängigen Austausch strukturierter Daten in einer Web-Service-Umgebung. Wie bereits erwähnt, ist die Bezeichnung *Simple Object Access Protocol* inzwischen hinfällig und durch SOAP als Eigennamen ersetzt worden – womöglich, weil dieses Protokoll ungleich komplexer ist als verwandte Kandidaten wie XML-RPC und demnach nicht *simple*. Außerdem zielt SOAP nicht ausschließlich darauf ab, den Zugriff auf Objekte wie etwa in einer objektorientierten Programmiersprache zu gewährleisten.

Die letzte als W3C-Empfehlung herausgegebene Version ist die zweite Edition von SOAP 1.2 im April 2007. Die Spezifikation besteht aus vier Teilen, dem *Primer* [152], dem *Messaging Framework* [153], den *Adjuncts* [154] und der *Specification Assertions and Test Collection* [155]. Der erste Teil beschreibt eine nicht-normative Einführung in das SOAP-Protokoll, bestehend aus den Grundlagen zur generellen Verwendung und dem Protokollaufbau sowie der Nutzung von den Transportbindungen HTTP und SMTP. Der zweite Teil beschreibt die formale Definition und die Systematik zur Erweiterung von SOAP-Nachrichten. Im dritten Teil der Spezifikation sind verschiedene Protokollzusätze wie die Codierung von RPC-Daten, Nachrichtenaustauschmuster und Details zu Transportbindungen erklärt. Der letzte Teil besteht aus einer Gruppe

verschiedener Testfälle, mit denen die Konformität von SOAP-Engines zu SOAP 1.2 erhöht werden kann, die jedoch nicht als Garant für die vollständige und korrekte Erfüllung der Spezifikation gilt.

Links in Abbildung 2.2 ist der schematische Aufbau einer SOAP-Nachricht dargestellt. Ähnlich wie bei einem Postbrief besteht jede Nachricht aus einem Umschlag (*SOAP-Envelope*) und dem Inhalt (*SOAP-Body*). Hinzu kommen optionale Parameter, die üblicherweise Verwaltungsinformationen enthalten (*SOAP-Header*) und bezogen auf einen Postbrief etwa mit Empfänger- und Absenderadresse vergleichbar sind. Rechts im Bild ist die XML-serialisierte Form aufgezeigt. Erkennbar sind die obligatorischen Elemente Envelope und Body, das optionale Header-Element sowie die exemplarischen, beliebigen Inhalte *sampleheader* und *samplebody*.

Eine SOAP-Nachricht wird zwischen verschiedenen SOAP-Knoten (engl. *SOAP Nodes*) transportiert. Der Absenderknoten wird als *Ultimate Sender*, der Empfängerknoten als *Ultimate Receiver* und alle Zwischenknoten als *Intermediaries* bezeichnet. Letztere erzeugen dabei im Wesentlichen die Daseinsberechtigung für die Verschlüsselung auf Nachrichtenebene. Es könnte beispielsweise verlangt werden, dass ein Intermediary bestimmte Teile einer Nachricht digital signieren soll oder nicht entschlüsseln darf. Die Verschlüsselung auf Transportebene ist in einem solchen Fall unzweckmäßig. Darüber hinaus sei an dieser Stelle angemerkt, dass einzelne Knoten nicht zwangsweise auf verschiedenen Rechnern liegen, sondern sogar im selben Prozess verarbeitet werden können. Ein Knoten ist demnach nur eine abstrakte Einheit und nicht an physikalische Adressen gebunden.

Der Header in einer SOAP-Nachricht ist für Protokollerweiterungen wie WS-Addressing oder WS-AtomicTransaction vorgesehen. WS-Addressing speichert dort z. B. Nachrichten-, Sender- und Empfänger-Identifikatoren ab. Jeder Block im Header kann hierbei mit drei SOAP-spezifischen Attributen signiert werden: *role*, *mustUnderstand* und *relay*. Das Attribut role gibt an, welche Knoten einen Header-Block verarbeiten müssen. Mit mustUnderstand wird festgelegt, ob ein Knoten einen Header-Block interpretieren muss oder ignorieren darf. Die Angabe von relay schreibt vor, dass ein Header-Block weitergesendet werden soll, falls er nicht verarbeitet wurde. Im Body einer SOAP-Nachricht können beliebige Daten hinterlegt werden. Damit es nicht zu Namenskonflikten kommt, sind Anwendungsentwickler und Standardisierungsgremien dazu angehalten, jede Strukturinformation sowohl im Body als auch im Header mit XML-Namensräumen [161] zu versehen.

In der SOAP-Spezifikation sind außerdem *Message Exchange Patterns (MEPs)* beschrieben. Übersetzt heißt MEP Nachrichtenaustauschmuster und beschreibt die Abfolge eines Nachrichtenaustausches zwischen zwei Kommunikationspartnern. SOAP 1.2 liefert zwei vordefinierte MEPs: *Request-Response* und *Response*. Request-Response beschreibt eine Anfrage in Form einer SOAP-Nachricht gefolgt von einer Antwort in Form einer SOAP-Nachricht. Response beschreibt eine Nicht-SOAP-Anfrage gefolgt

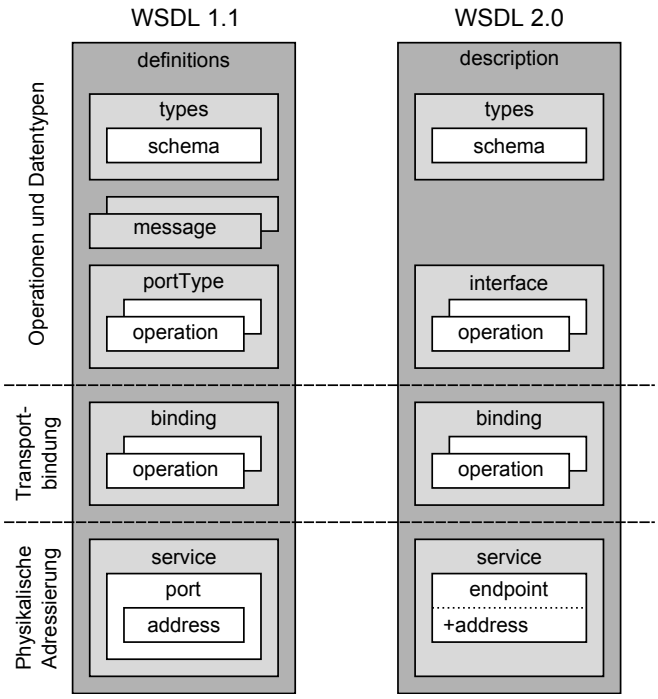


Abbildung 2.3.: Vergleich des XML-Schemas von WSDL 1.1 und WSDL 2.0, entnommen aus [111].

von einer SOAP-Antwort. Im Allgemeinen wird die HTTP-Transportbindung mit den Methoden *POST* und *GET* genutzt, um genau diese beiden MEPs abzubilden.

Der vorangegangene Abschnitt hat Einzelheiten zum SOAP-Protokoll erläutert. Bezogen auf das SOA-Rollenmodell (vgl. Abbildung 1.5 auf Seite 11) ermöglicht es den Nachrichtenaustausch zwischen Dienstanbietern und -konsumenten, also den Vorgang des Dienstaufrufs. Nachfolgend wird näher auf die Schnittstellenbeschreibungssprache WSDL eingegangen.

2.1.2. WSDL

Schnittstellenbeschreibungssprachen bilden ein wichtiges Instrument, um Programmcode für Dienstkonsumenten automatisch zu erzeugen und die Suche über Verzeichnisdienste zu erleichtern. Die grundlegende Aufgabe der *Web Service Description Language (WSDL)* ist es deshalb, eine exakte und maschinenlesbare Beschreibung von Schnittstellen zu ermöglichen. Wesentliche Abgrenzungsmerkmale zu anderen IDLs sind hierbei die Erweiterbarkeitsmechanismen und die Verwendung des XML-Standards.

WSDL liegt derzeit in zwei Versionen vor: WSDL 1.1 [145] und WSDL 2.0 [156]. Zwar ist die Version 1.1 bis heute nicht im Status einer Empfehlung, sie ist jedoch immer noch häufiger im Einsatz als ihre Nachfolgerversion 2.0, welche den Empfehlungsstatus hat. Möglicherweise war die Notwendigkeit für WSDL 2.0 nicht in dem Ausmaß gegeben wie bis zum Zeitpunkt der Veröffentlichung angenommen wurde. Obgleich sich beide Varianten in ihrem Format signifikant unterscheiden, bietet Version 2.0 keine Umgestaltung oder Verbesserung der Funktionalität. Abbildung 2.3 stellt die Formate gegenüber. Beide Versionen enthalten zunächst ein Datenformat (*types*), welches für den Datenaustausch zwischen den beschriebenen Web-Services benötigt wird. WSDL 1.1 sieht an dieser Stelle die häufig genutzte Datenbeschreibungssprache XML-Schema vor, während in der Version 2.0 beliebige Beschreibungssprachen wie zum Beispiel DTD oder RELAX NG [53] eingesetzt werden können. In WSDL 1.1 wird das Datenformat anschließend über separierte *message*-Elemente referenziert. Diese Referenzen wurden vom W3C als überflüssig empfunden und aus der WSDL-2.0-Spezifikation entfernt.

Ein Web-Service setzt sich immer aus mehreren Operationen zusammen. Diese Operationen sind in beiden WSDL-Versionen unter einer abstrakten Schnittstellenbeschreibung zusammengefasst, unterscheiden sich jedoch in der Bezeichnung durch *portType* und *interface*. Jede Operation erhält einen Bezeichner und die Datentypen, die über die Operation ausgetauscht werden. Dabei sind verschiedene Nachrichtenaustauschmuster (engl. MEPs) möglich, mit denen gesteuert werden kann, in welche Richtung die Kommunikation initiiert und durchgeführt wird. Während Version 1.1 vier MEPs festlegt, sind in Version 2.0 zwei Basismuster und zusätzliche drei Muster in einer Erweiterung definiert. Die beiden wichtigsten Kandidaten sind *Request/Response* bzw. *Out/In* sowie *Notification* bzw. *Out-Only*. Die MEPs der WSDL-Spezifikation dürfen nicht mit denen aus der SOAP-Spezifikation verwechselt werden. Es handelt sich hierbei um zwei unterschiedliche Konzepte, die denselben Namen tragen.

Zusätzlich zur Schnittstellenbeschreibung benötigen beide Versionen der WSDL eine Transportbindung und die physikalische Adresse, unter der ein Web-Service abrufbar ist. Die Transportbindung ist im Element *binding* untergebracht. Jede Operation bekommt eine Transportbindung zugeordnet, überlichweise SOAP-over-HTTP. Denkbar sind jedoch beliebige Varianten wie SOAP-over-TCP, SOAP-over-SMTP oder auch JMS. Die physikalische Adresse wird im Element *service* definiert. WSDL 1.1 benutzt hierzu die Elemente *port* und *address*, während WSDL 2.0 *endpoint* verwendet und die Adresse als Attribut vorschreibt.

WSDL-Dokumente bieten in ihrer Grundform ausschließlich maschinenlesbare Informationen über die syntaktische Nutzung eines Services. Semantische Beschreibungen (was bewirkt ein Operationsaufruf und in welcher Weise ist er zu nutzen) sind bis auf ein informelles *documentation*-Element, welches für Anwendungsentwickler vorgesehen ist, nicht dokumentiert. Abhilfe schafft in diesem Zusammenhang die Erweiterung SA-WSDL, welche es erlaubt, mithilfe von Annotationen semantische Modelle zwischen XML-Schema-Definitionen und WSDL-spezifischen Elementen zu definieren. In

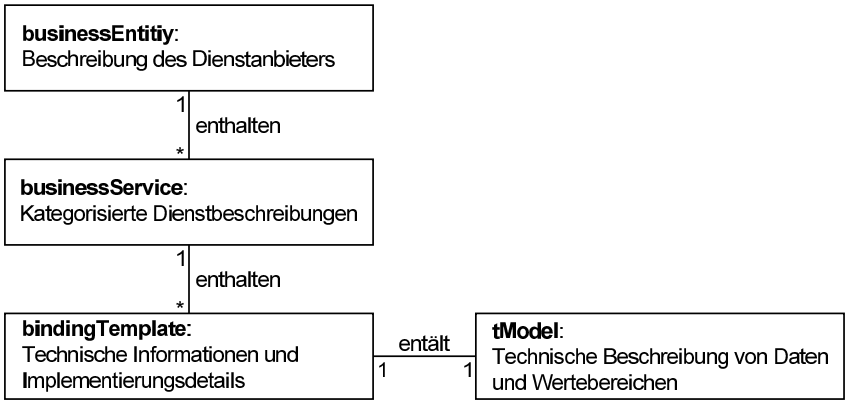


Abbildung 2.4.: Beziehungen zwischen den UDDI-Datentypen, in Anlehnung an [92].

SA-WSDL ist lediglich die Einbindung der Annotationen beschrieben, nicht jedoch die Transformationsregeln, nach denen die semantischen Modelle integriert werden.

2.1.3. UDDI

SOAP und WSDL bilden die wesentlichen Grundpfeiler für die Umsetzung der zwei SOA-Rollen Dienstanbieter und Dienstkonsument (vgl. Abbildung 1.5 auf Seite 11). In diesem Abschnitt wird eine technische Spezifikation für die Rolle des Verzeichnisdienstes beschrieben.

Der bekannteste Verzeichnisdienst für Web-Services wurde im August 2000 eingeführt und trägt den Namen *Universal Description, Discovery and Integration (UDDI)* [92]. Die ursprüngliche Vision war es, dass Firmen ihre Dienstbeschreibungen auf zentralen Dienste-Servern vorhalten und somit Handelspartner auf unkompliziertem Wege ausfindig gemacht werden können. Neben den allgemeinen Firmenkontaktdaten stehen damit die technischen Schnittstellen für Business-to-Business-Anwendungen unmittelbar zur Verfügung. Die Vision von UDDI konnte sich im Unternehmensumfeld nicht durchsetzen. Anfang 2006 gaben die größten UDDI-Verfechter Microsoft, IBM und SAP bekannt, ihre Verzeichnisdienste abzuschalten [118]. Heute existieren nur noch wenige internetweite UDDI-Verzeichnisse. Der Fokus hat sich verschoben in spezialisierte Register, die überwiegend firmenintern eingesetzt werden. Für die Medizingerätekommunikation spielt UDDI eine untergeordnete Rolle, da es auf zentrale Instanzen angewiesen ist und somit einen *Single Point of Failure* darstellt. Im Folgenden wird daher nur ein kurzer technologischer Abriss gegeben.

UDDI 3.02 besteht aus einem technischen Rahmenwerk, einem Datenmodell in Form von XML-Schemata sowie einer Reihe von WSDL-Dokumenten. Das Datenmodell

ist in die Datentypen *businessEntity*, *businessService*, *bindingTemplate* und *tModel* gegliedert. Die Beziehung zwischen den Elementen ist in Abbildung 2.4 skizziert. Die businessEntity ist das Stammelement eines UDDI-Verzeichniseintrags. Es enthält neben informellen Angaben über den Dienstleister, wie z. B. der Postanschrift und Telefonnummern, ein oder mehrere businessService-Einträge. Elemente vom Typ businessService enthalten informelle Angaben über einzelne Dienste, beispielsweise deren Name und natürlichsprachliche Funktionsbeschreibungen. Darüber hinaus können die Dienste in Kategorien eingeordnet werden. Im bindingTemplate wird der technische Zugangspunkt zu einem Dienst definiert. Typischerweise steht an dieser Stelle ein *Uniform Resource Locator (URL)*, unter dem der korrespondierende Web-Service abgerufen werden kann. Weiterhin kann jedes bindingTemplate ein sogenanntes tModel referenzieren. Das tModel enthält Informationen zum Transportprotokoll sowie eine technische Beschreibung der Datentypen, die mit einem Dienst ausgetauscht werden. Verschiedene Anbieter referenzieren dieselben tModel-Instanzen, um die Kompatibilität mit anderen Schnittstellen zu ermöglichen.

Die WSDL-Dokumente, welche als Teil der UDDI-Spezifikation mitgeliefert werden, bieten *Application Programming Interfaces (APIs)* zur Suche (*Inquiry API*) in einem Dienstverzeichnis und zur Verwaltung (*Publication API*) von Dienstanbieter-Einträgen an. Üblicherweise stellen UDDI-Frameworks auch eine Web-Oberfläche für die Verwaltung bereit. Während das Publication API nur Verwendung findet, wenn sich ein Anbieter in einem Verzeichnis registrieren oder etwas an einem Eintrag ändern möchte, wird die Suche über das Inquiry API wesentlich häufiger gebraucht. Die Suche setzt sich dabei aus zwei Schritten zusammen: *find* und *get* – jeweils auf jeden Datentyp anwendbar. Mit find werden grobe Ergebnislisten zusammengetragen; mit get Details zu einzelnen Ergebnissen abgefragt. Neben dem Publication API und Inquiry API existieren noch zwei weitere APIs. Das *Security API* dient zu Authentifikations- und Autorisierungszwecken. Mit dem *Subscription API* können sich Dienstnutzer gemäß Publish/Subscribe über Änderungen in einem UDDI-Verzeichnis informieren lassen.

2.2. Das Devices Profile for Web Services

Wie bereits erwähnt, entstehen durch die Vielfalt verschiedener Module im Web-Service-Technologiestapel häufig Interoperabilitätsprobleme. Dies liegt zum einen daran, dass Web-Service-Spezifikationen teilweise von mehreren Unternehmen entworfen und bei den unabhängigen Gremien zur Standardisierung eingereicht werden. Zum anderen liegen zahlreiche Spezifikationen in verschiedenen Versionen vor und sind trotz der Sorgfalt während des Standardisierungsprozesses oftmals mehrdeutig ausgelegt. Um diese Probleme zu umgehen, helfen sogenannte Web-Service-Profile weiter. Ein Web-Service-Profil beschreibt eine Zusammenfassung einer Gruppe von Web-Service-Standards in vorab definierten Versionen. Ein solches Spezifikationskon-

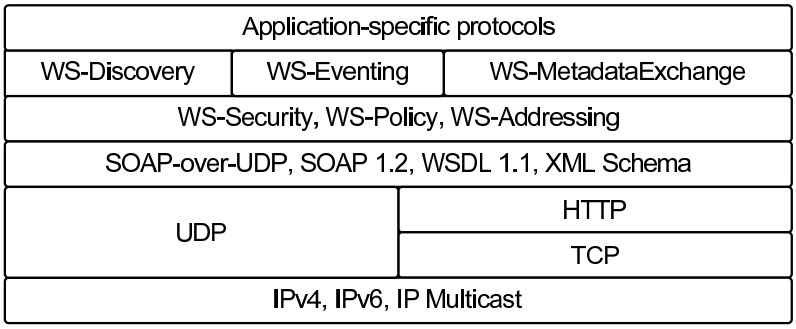


Abbildung 2.5.: In der DPWS-Spezifikation referenzierte Internet- und Web-Service-Standards, in Anlehnung an [170].

glomerat wird mit zusätzlichen Nutzungskonventionen ergänzt, um eindeutige Regeln für die Verwendung festzulegen.

Von 2002 bis 2010 [15] spezifizierte die *WS Interoperability Organization (WS-I)*, ein offener Industrieverband, verschiedene Profile zur Steigerung der Web-Service-Interoperabilität. Die zwei wichtigsten Entwürfe sind das *WS-I Basic Profile* [135, 136, 138, 139] und das *WS-I Basic Security Profile* [137, 140]. Beide Profile liegen in mehreren Versionen vor. Während das WS-I Basic Profile diverse Spezifikationen zusammenbringt, um eine SOA umsetzen zu können, liegt der Schwerpunkt im WS-I Basic Security Profile auf der Komposition von Standards zur Absicherung des Kommunikationskanals auf Nachrichten- und Transportebene. Die WS-I hat inzwischen ihre Arbeiten finalisiert und ist mit der OASIS fusioniert.

Im Mai 2004 publizierte Microsoft erstmals ein weiteres Web-Service-Profil, das auf die Vernetzung ressourcenbeschränkter Geräte zugeschnitten ist: DPWS. Die Spezifikation zielt darauf ab, ähnlich wie *Universal Plug and Play (UPnP)*, Geräte in lokalen Netzwerken automatisch zu verbinden. Anders als UPnP ist DPWS jedoch inhärent kompatibel mit der Web-Service-Technologie und ihren Erweiterungsmechanismen. Im Juli 2008 bei der OASIS zur Standardisierung eingereicht, wurde es in einer nicht ratifizierten Version [122] erstmals im Jahr 2007 im Betriebssystem *Microsoft Windows Vista* integriert. Damit ist es seitdem möglich, Daten zwischen den DPWS-kompatiblen Netzwerkgeräten wie beispielsweise Druckern und Heimrechnern automatisch auszutauschen.

DPWS liegt derzeit in Version 1.1 vor [100]. Eine Version 1.2 ist in Planung, wird jedoch seit 2009 nicht mehr weiterentwickelt [107]. Aus diesem Grund sind im DPWS immer noch veraltete Web-Service-Spezifikationen wie die *Member Submissions* von WS-Eventing [150] und WS-MetadataExchange [160] enthalten, die inzwischen auch als *Recommendation* vorliegen. In Abbildung 2.5 sind alle im DPWS referenzierten Internet- und Web-Service-Spezifikationen dargestellt. Ähnlich wie im ISO/OSI-Schichtenmodell

sind auf den untersten Ebenen transportrelevante Technologien angesiedelt. Während das WS-I Basic Profile ausschließlich den Transport über HTTP erlaubt, sind im Rahmen des DPWS auch UDP-Transportbindungen denkbar.

Oberhalb der Transportprotokolle sind die relevanten Web-Service-Basisstandards angesiedelt. DPWS nutzt insbesondere SOAP 1.2 sowie das veraltete WSDL 1.1. Auf den übrigen Ebenen beinhaltet es eine Reihe von Web-Service-Erweiterungen, von denen WS-Security und WS-Addressing bereits genannt wurden. Die Ebene *Application-specific protocols* ermöglicht es Anwendern, DPWS durch eigene Protokolle zu erweitern.

2.2.1. Referenzierte Web-Service-Standards

Bei *WS-Policy* handelt es sich um eine Spezifikation des W3C [157], die es erlaubt, Web-Service-Richtlinien in maschinenlesbarer Form zu definieren. Mit Hilfe von WS-Policy können beispielsweise Anforderungen sowohl auf Anbieter- als auch Nutzerseite an die Sicherheit, Güte oder Version eines Dienstes gestellt werden. DPWS legt die Richtlinie *http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/Profile* fest, welche die Kompatibilität mit einem DPWS-konformen Dienst definiert. Diese Richtlinie wird typischerweise im binding-Element eines WSDL-Dokuments notiert (vgl. Abbildung 2.3 auf Seite 22).

WS-MetadataExchange [165] wurde ebenfalls vom W3C spezifiziert und ermöglicht die Abfrage von Metadaten eines Web-Service-Endpunktes. DPWS nutzt WS-MetadataExchange für den Transfer von dienstspezifischen Daten wie WSDL-Dokumenten.

Eine weitere Spezifikation des W3C ist *WS-Transfer* [166]. Bei WS-Transfer handelt es sich um einen Standard zum Abruf von XML-basierten Web-Service-Ressourcen. Ähnlich wie in HTTP die Operationen *Get*, *Put*, *Delete* und *Push* werden in WS-Transfer die Operationen *Get*, *Put*, *Delete* und *Create* über SOAP-Anfragen abgebildet. Im Rahmen des DPWS findet WS-Transfer ausschließlich für die Abfrage gerätebezogener Metadaten wie Modellbezeichnungen und -nummern sowie für die Bekanntmachung von Web-Service-Endpunkten Anwendung.

WS-Eventing und *WS-Discovery* spielen im Zusammenhang mit dieser Arbeit eine besondere Rolle und werden deshalb in Kapitel 4 ab Seite 61 und Kapitel 5 ab Seite 79 separat erläutert. Im Kontext vom DPWS wird WS-Discovery eingesetzt, um mit Hilfe von SOAP-over-UDP und IP-Multicast Geräte im Netzwerk zu finden. WS-Eventing ist eine Erweiterung, die es erlaubt, Publish/Subscribe-Funktionalität für Web-Services bereitzustellen.

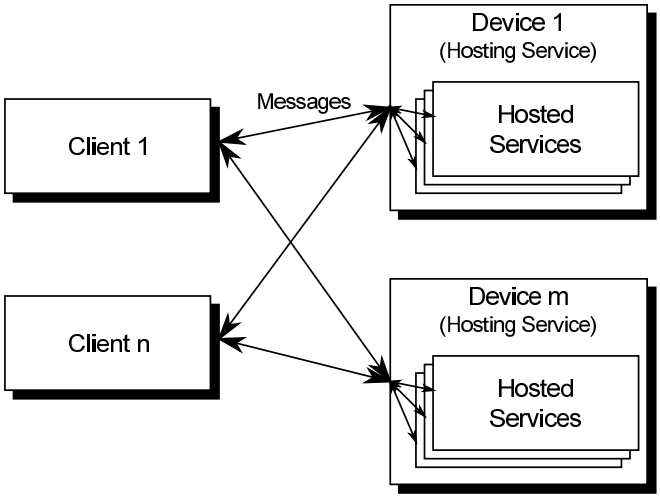


Abbildung 2.6.: Das Client-Server-Modell des DPWS, in Anlehnung an [100, Abschnitt 1.2.2].

2.2.2. Architektur und Konzept

Abbildung 2.6 skizziert das Dienstanbieter- und Dienstnutzermodell des DPWS. In diesem Modell werden *Clients*, *Hosting Services* und *Hosted Services* unterschieden. Der wesentliche Unterschied zu gewöhnlichen Web-Service-Architekturen ist hierbei die Klassifikation eines Gerätes als Hosting Service. Jedes Gerät stellt demnach einen virtuellen Dienst dar, der mehrere Web-Services in Form eines Hosted Service kapselt. Für jeden Hosted Service kann ein WSDL-Dokument abgerufen werden.

Damit ein Client die WSDL-Beschreibungen abrufen kann, muss er zunächst gemäß des zugrundeliegenden SOA-Paradigmas passende Geräte finden. Anstelle eines zentralen Verzeichnisdienstes wie UDDI werden Suchanfragen dezentral mit Hilfe von WS-Discovery an alle Teilnehmer eines Ad-hoc-Netzwerkes gestellt. Nachdem ein passendes Gerät gefunden ist, kann mit WS-MetadataExchange und einem WS-Transfer-Get die Gerätebeschreibung abgerufen werden. Die Metadaten enthalten wie oben beschrieben Herstellername, Modell usw. Darüber hinaus liefert die WS-Transfer-Anfrage eine Liste mit Referenzen aller Hosted Services. Die Metadaten eines Hosted Services lassen sich über eine native WS-MetadataExchange-Anfrage abrufen und enthalten ein WSDL-Dokument mit der Dienstbeschreibung.

Zur Absicherung der Kommunikation wird im DPWS die Nutzung von HTTPS empfohlen. Diese Empfehlung sollte insbesondere berücksichtigt werden, wenn ressourcenschwache Geräte kommunizieren, da die Verschlüsselung auf Nachrichtenebene bereits durch die notwendige Codierung ins Base64-Format zusätzliche Prozessorlast erzeugt.

Eine Ausnahme bildet WS-Discovery, welches hauptsächlich auf SOAP-over-UDP-basiertem Nachrichtenaustausch aufbaut. Für diesen Fall ist das *Compact Signature Format* [106] vorgesehen.

2.3. DPWS für medizinische Geräte

Ein Kommunikationsrahmenwerk, das in sensiblen medizinischen Umgebungen wie dem Operationssaal oder der Intensivstation eingesetzt wird, erfordert neben den Bestandteilen der DPWS-Spezifikation weitere technische Merkmale. In [111] sind hierfür verschiedene Fähigkeiten beschrieben, die ein medizinisches Netzwerk aus Web-Service-Diensten bereitstellen sollte [111]. Dazu gehören unter anderem die Optimierung der Dienstgüte, die Unterstützung von zweikanaliger Datenübertragung als auch die Diensteauffindung außerhalb von Subnetzgrenzen. Zur Absicherung der Kommunikation schlägt er ein Authentifikations- und Autorisierungs-Management vor, welches ohne zentrale Instanzen auskommt.

2.3.1. Fair-Switch-Technologie

Für Punkt-zu-Punkt-verbundene IT-Systeme stellt die Dienstgüte des Netzwerkes, auch als *Quality of Service (QoS)* bezeichnet, im Allgemeinen eine untergeordnete Rolle dar. Den Kommunikationspartnern steht die volle Bandbreite zur Verfügung und die beiden IT-Systeme können alleine aushandeln, wie sie die Verbindung nutzen. In IP-Netzwerken muss die verfügbare Bandbreite üblicherweise auf mehr als zwei Kommunikationspartner verteilt werden. Entweder verzichtet man hier auf eine Dienstgüte im Sinne der Best-Effort-Übertragung oder führt eine pauschale Priorisierung ein. Für dynamisch vernetzte Medizingeräte ist diese vereinfachte Betrachtung nicht ausreichend. Verschiedene Anwendungsfälle zwischen zwei Medizingeräten erfordern eine garantierte Nachrichtenübertragung. Durch Schad-Software befallene Medizingeräte könnten außerdem das gesamte Netzwerk durch Denial-of-Service-Attacken blockieren. Zu diesem Zweck wird in [113] eine Fair-Switch-Technologie eingeführt, anhand derer die Bandbreite eines über ein Switch gesteuertes Netzwerkes gleichmäßig verteilt wird. Damit stehen n an einem Switch angeschlossenen Geräten $\frac{1}{n}$ der Bandbreite garantiert zur Verfügung, sofern das Switch über n^2 Warteschlangen verfügt. Um die Fairness zu maximieren, werden die Warteschlangen gemäß des Round-Robin-Verfahrens abgearbeitet. In einem exemplarischen Ethernet-Netzwerk mit Gigabit-Switch und 50 angeschlossenen Geräten entstünde hiermit eine Bandbreite pro Gerät von 20 MBit/s.

2.3.2. Zweikanaligkeit

Die Patientensicherheit nimmt bei der Vernetzung von Medizingeräten eine zentrale Rolle ein. Gemäß IEC 60601-1 ist ein Medizingerät so zu gestalten, dass ein erster Fehler im System dessen Nutzung während seiner Lebensdauer hinsichtlich der Sicherheit nicht negativ beeinflusst. Ein Medizingerät ist demnach sicher, wenn es eine der folgenden beiden Aspekte erfüllt:

1. Es wird eine Maßnahme ergriffen, die das Risiko abzüglich einer vernachlässigbaren Wahrscheinlichkeit sicher minimiert. Dies ist z. B. der Fall, wenn ein Watchdog das Gerät bei einem Fehler in einen sicheren Zustand versetzt. Unter einem Watchdog versteht man eine Systemkomponente, die das Gerät zur Laufzeit überwacht.
2. Eine nicht sichere Maßnahme wird durch eine zweite Maßnahme ergänzt. Die Wahrscheinlichkeit, dass die zweite Maßnahme ebenso ausfällt, muss vernachlässigbar sein.

Die Erstfehlersicherheit lässt sich bei der Fernsteuerung von Medizingeräten durch das Konzept der Zweikanaligkeit lösen. In [114] wird dazu ein Verfahren vorgestellt, mit dem zwischen zwei Geräten ein zweiter Datenkanal etabliert wird. Dieser Datenkanal kann sowohl in der Zeit als auch in der Repräsentation vom ersten getrennt werden. Da die Trennung in der Zeit jedoch eine zustandsbehaftete Verbindung zwischen den Kommunikationspartnern erfordert, ist die Nutzung der Trennung in der Repräsentation für lose gekoppelte Medizingeräte besser geeignet. Die Trennung in der Repräsentation bedeutet in diesem Zusammenhang, dass eine Information in zwei verschiedenen Repräsentationen vorliegt, und über eine einzige Nachricht versendet werden kann.

Abbildung 2.7 illustriert den Prozess der Datenübertragung für eine zweikanalig ausgelegte Middleware. Auf der dienstkonsumierenden Seite liegt die zu übertragende Information in zwei unabhängigen Datenobjekten vor. Dies können z. B. Java-Objects oder C-Structs sein. Die Middleware serialisiert zunächst die Daten und bildet anschließend eine Prüfsumme. Um zu überprüfen, ob der Serialisierer oder Parser fehlerhaft sind, werden die serialisierten Daten mit dem Parser anschließend in das programmiersprachenineterne Datenformat überführt und eine Kopie angefertigt. Danach kann diese Kopie mit dem zweiten Eingangskanal verglichen werden. Sind der Serialisierer oder Parser fehlerhaft, schlägt der Vergleich fehl. Andernfalls arbeiten die Komponenten korrekt, so dass der Versand der serialisierten Daten an den Dienstanbieter erfolgen kann.

Auf der dienstanbietenden Seite erstellt der Parser zunächst ein Datenobjekt und konstruiert eine Kopie. Diese Kopie wird serialisiert und die Prüfsumme mit der Prüfsumme aus der eingehenden Nachricht (dem zweiten Kanal) verglichen. Sind der Parser, Serialisierer oder Kopiervorgang fehlerhaft gewesen, schlägt der Vergleich der

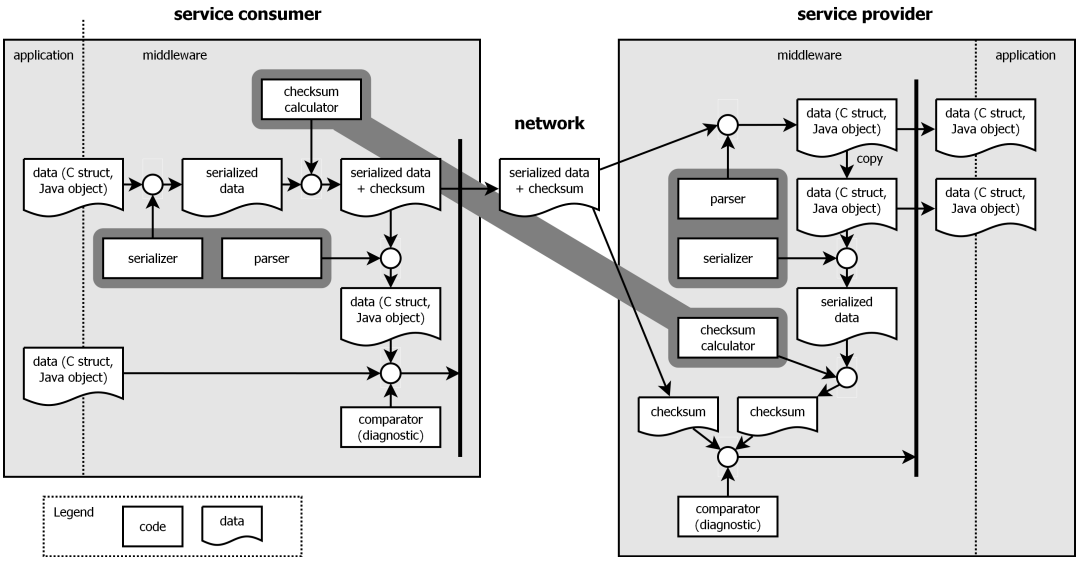


Abbildung 2.7.: Prüfsummenbasierte Zweikanaligkeit zur Herstellung funktionaler Sicherheit, entnommen aus [114].

Prüfsumme fehl. Die Middleware des Dienstanbieters erkennt den Fehler und kann dem Dienstkonsumenten antworten, dass die Übertragung fehlgeschlagen ist. Der Dienstkonsument könnte anschließend einen weiteren Übertragungsversuch initiieren. Ist die übertragene mit der erzeugten Prüfsumme gleich, können beide Objekte an die Applikation durchgereicht werden.

Der vorgestellte Prozess zur Konstruktion eines zweiten Kanals ist prüfsummenbasiert. Durch die gegenseitige Verifizierung des Serialisierers mit dem Parser wird sichergestellt, dass kein Fehler bei der Vorbereitung der Nutzdaten entsteht.

2.3.3. WS-Discovery über Subnetzgrenzen

DPWS bietet für das Auffinden von Diensten das WS-Discovery-Protokoll an. Mit diesem Protokoll ist es möglich, in Subnetzen anhand von Multicast-Nachrichten Geräte bzw. Dienste zu finden und die WSDL-Dokumente als auch Adressen abzurufen. Ein Gerät muss damit nicht vorab seine Netzwerkumgebung kennen. Da die Adressvergabe zur Laufzeit erfolgt, wird der Grad der losen Kopplung des Systems erhöht.

Ein Nachteil des WS-Discovery-Protokolls ist die Einschränkung des Suchraumes auf ein Subnetz. Es existiert keine Möglichkeit, Dienste außerhalb eines Subnetzes zu suchen und in einen Geräteverbund transparent einzuhängen. Abhilfe schafft das in [112] entwickelte Discovery-Protokoll. Hierbei kommt ein hybrides Discovery zum

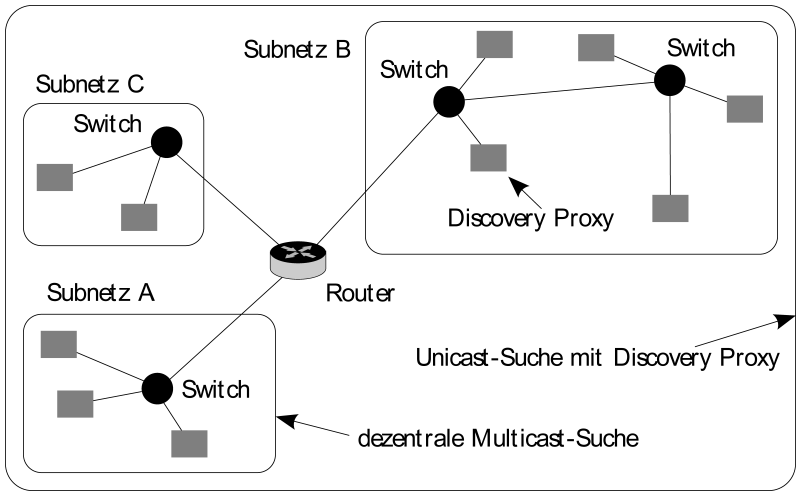


Abbildung 2.8.: Architektur für ein WS-Discovery-Protokoll über Subnetzgrenzen, entnommen aus [112].

Einsatz, bei der die lokale Suche dezentral und die globale Suche zentral abgefertigt wird. Abbildung 2.8 zeigt die Architektur des Systems. Für lokale Suchen innerhalb eines Subnetzes kommt der Ad-hoc-Modus von WS-Discovery zum Einsatz, während für globale Suchen außerhalb eines Subnetzes ein Discovery-Proxy verwendet wird. Damit der Discovery-Proxy von außerhalb des Netzwerkes auffindbar ist, kann das *Domain Name System (DNS)* genutzt werden. Hierbei wird zu einer Domain im DNS-Server ein Service-Eintrag definiert, der den Discovery-Proxy adressiert.

2.3.4. Datenschutz und Rechte-Management

Sicherheit spielt in der Medizin eine übergeordnete Rolle. Neben der Patientensicherheit ist im Rahmen der Medizingerätevernetzung vor allem auch die Datensicherheit und der Schutz vor unbefugtem Zugriff hervorzuheben. Typischerweise kommen hierfür Authentifikations- und Autorisierungsverfahren zum Einsatz, in denen Berechtigungen einzelner Geräte von zentralen Verzeichnisdiensten, auch als *Security Token Services (STSs)* bezeichnet, abgerufen werden können. Hierbei erfragt ein Dienstkonsument bei einem STS ein Token, welches ihm den Zugriff auf einen Dienstanbieter erlaubt. Der STS prüft, ob der Konsument den Anbieter nutzen darf und erteilt ihm anschließend gegebenenfalls die Zugriffsberechtigung. Der Nachteil dieses Verfahrens besteht darin, dass zentrale Verzeichnisdienste einen Flaschenhals in der Kommunikation darstellen. Fiele der STS aus, wären die Geräte im Operationssaal nicht mehr in der Lage, Informationen auszutauschen.

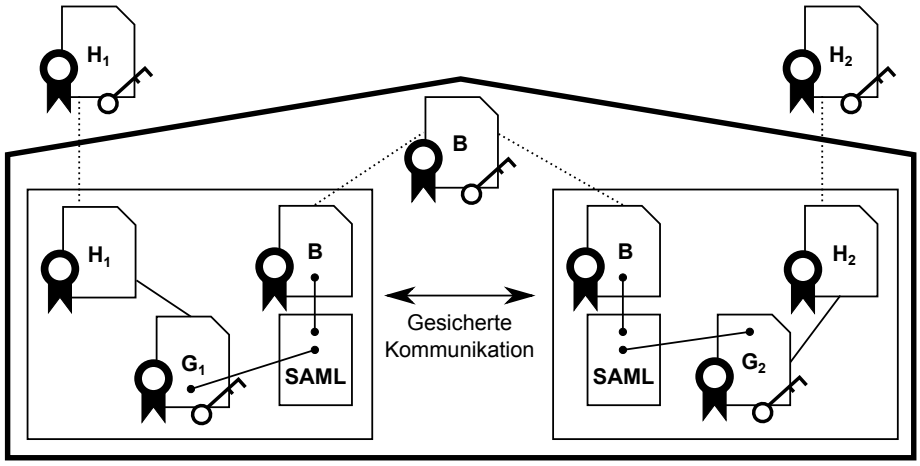


Abbildung 2.9.: Sicherheitsarchitektur. Die mit einem B markierten Zertifikate sind die Wurzelzertifikate des Betreibers. Mit H markierte Zertifikate entsprechen den Wurzelzertifikaten der Hersteller 1 bzw. 2. Die Zertifikate eines Gerätes sind mit G und dem entsprechenden Index bestückt. Ein Schlüssel bedeutet den Besitz des privaten Schlüssels. In den SAML-Dokumenten sind die Berechtigungen von Dienstkonsumenten und -anbietern hinterlegt.

Um diesem Problem zu begegnen, ist in [116] ein Konzept für eine dezentrale Zugriffskontrolle publiziert worden. Die Idee besteht darin, dass alle relevanten Sicherheitsinformationen vorab auf jedem Gerät installiert werden. Somit entfällt die Abfrage eines zentralen STS.

Technisch kann dieses Verfahren beispielsweise mit der *Security Assertion Markup Language (SAML)* [91] und X.509-Zertifikaten [19] gelöst werden. Üblicherweise vergibt eine vertrauenswürdige Instanz zur Laufzeit SAML-Assertions. Diese Assertions enthalten XML-basierte Authentifikations- und Autorisierungsinformationen, die zum Schutz vor Integritätsverletzungen digital signiert sind. Die Assertions entsprechen im Prinzip einer Vollmacht und räumen einem Konsumenten gegenüber einem Dienstanbieter bestimmte Nutzungsrechte ein.

Die zugrundeliegende Architektur ist in Abbildung 2.9 grafisch dargestellt. Um die dezentrale Berechtigung zu ermöglichen, werden Vollmachten nicht zur Laufzeit ausgestellt, sondern vorab einmalig bzw. in wiederkehrenden Intervallen auf den Geräten installiert. Voraussetzung ist, dass der Medizingerätebetreiber seine eigene Public-Key-Infrastruktur verwaltet und über ein eigenes Wurzelzertifikat verfügt. Jede SAML-Asseration wird durch einen Aussteller (in dem Fall Betreiber) digital signiert. Weiterhin bezieht sich die SAML-Assertion auf ein Subjekt (hier ein Gerätezertifikat).

Die SAML-Assertions werden mit dem vom Medizingerätebetreiber ausgestellten Wurzelzertifikat digital signiert. Das Wurzelzertifikat muss, da im Netzwerk kein herkömmlicher STS vorhanden ist, von einem IT-Netzwerk-Manager auf jedem Gerät samt der SAML-Assertions abgespeichert werden. Der IT-Netzwerk-Manager übernimmt damit die Aufgabe des STS. Die Geräte vertrauen dem Wurzelzertifikat des Betreibers und erlauben den Zugriff entfernter Geräte, so dass auf den zentralen STS verzichtet werden kann.

Um die SAML-Assertions und das Wurzelzertifikat zu installieren, muss der Medizingerätehersteller dem IT-Netzwerk-Manager des Betreibers Zugriff gewährleisten. Dieser Zugriff kann anhand eines separaten Zertifikats erfolgen, das der Hersteller dem Betreiber beim Verkauf seines Gerätes mitliefert. Ein solches zusätzliches Zertifikat ist in Abbildung 2.9 nicht enthalten. Medizingeräte eines Herstellers vertrauen sich gegenseitig und benötigen deshalb nicht zwingend eine durch den Betreiber signierte SAML-Assertion.

Kapitel 3.

Anwendungsszenarien am Beispiel eines neurochirurgischen Eingriffs

Um den Nutzen der SOA-basierten Medizingeräteintegration darzustellen, bedarf es einer Analyse der Abläufe im Operationssaal inklusive möglicher Prozessoptimierungen. Dieses Kapitel beschreibt verschiedene Anwendungen vernetzter Medizingeräte am Beispiel eines neurochirurgischen Eingriffs und bildet den Leitfaden für weiterführende Arbeiten. Die Durchführung einer Machbarkeitsstudie skizziert die Anforderungen hinsichtlich gerätetechnischer Voraussetzungen und Middleware-Produkte für die Gerätekonnektivität. Wesentliche Teile dieses Kapitels wurden vorab in [37] publiziert. Die Anwendungsszenarien können dabei helfen, allgemeine Anforderungen an interoperable Medizingeräte zu definieren. Die Machbarkeitsstudie zeigt, dass die anvisierte Funktionalität zielführend umgesetzt werden kann und diskutiert mögliche konzeptuelle Schwächen. Darüber hinaus wird ersichtlich, in welchen Punkten der Geräteintegration weiterer Forschungs- und Entwicklungsbedarf besteht.

In Abschnitt 3.1 werden zunächst verwandte Arbeiten vorgestellt. Abschnitt 3.2 beschreibt verschiedene Anwendungsszenarien anhand eines neurochirurgischen Eingriffs am *Universitätsklinikum Schleswig-Holstein (UKSH)*. Anschließend wird in Abschnitt 3.3 ein prototypisches System umgesetzt, um die erarbeiteten Anwendungsszenarien nachstellen zu können. Abschnitt 3.4 bewertet das prototypische System. Das Kapitel wird mit einem Fazit in Abschnitt 3.5 abgeschlossen.

3.1. Verwandte Arbeiten

Anwendungsszenarien für die Vernetzung von Medizingeräten sind im Rahmen unterschiedlicher Forschungs- und Standardisierungsaktivitäten entwickelt worden. Goldman et al. vom *Medical Device „Plug-and-Play“ Interoperability Program (MD PnP)* [80] definieren verschiedene Aspekte der Interoperabilität als Teil des *Integrated Clinical Environment Standard (ICE Standard)* [18]. Für jeden Aspekt liefern sie jeweils einen Präzedenzfall und unterscheiden im Wesentlichen *Safety Interlocks*, *Process Control*,

Decision Support Systems, *Smart Alarm Systems*, *Physiological Closed Loop Control* sowie *Medical Device Plug-and-Play Interoperability*.

Safety Interlocks beschreiben die gegenseitige Überwachung und Steuerung von Medizingeräten, um die Patientensicherheit maßgeblich zu erhöhen. Angeführt wird dieser Interoperabilitätsaspekt durch den Anwendungsfall einer automatischen Synchronisation zwischen Röntgen- und Beatmungsgerät. Er wurde vorgeschlagen von Arney et al. [7] basierend auf einem Artikel von Ann S. Lofsky [76] über einen Zwischenfall während einer laparoskopischen Gallenblasenentfernung. Ein Röntgengerät soll eingesetzt werden, um eine Aufnahme vom (aufgrund der Atmung bewegten) Brustkorb zu machen. Um den Brustkorb still zu halten und somit die Qualität der Aufnahme zu verbessern, wird kurzzeitig das Beatmungsgerät abgeschaltet. Eine Kette unvorhersehbarer Umstände führt dazu, dass das Beatmungsgerät nicht mehr eingeschaltet wird und der Patient verstirbt. Durch die gegenseitige Steuerung von Röntgen- und Beatmungsgerät hätte dieser tragische Unfall verhindert werden können.

Unter *Process Control* und *Decision Support Systems* ist die Anbindung und Bewertung elektronischer Patientenakten zur Fernüberwachung und Entscheidungsunterstützung zu verstehen. Process Control legt den Fokus hierbei auf die Einbindung von Geräten in verschiedene Krankenhausprozesse. Beispielsweise könnte ein dahingehend integriertes System Infusionspumpen eines Patienten beobachten, das klinische Personal entsprechend physiologischer Messungen anleiten, Laboranfragen generieren und Dosierungen verifizieren. Decision Support Systems sammeln zunächst Daten medizinischer Diagnosegeräte. Anschließend werten Algorithmen diese Daten anhand klinischer Observationen aus und schlagen therapeutische Maßnahmen vor.

Ein gängiges Problem bei chirurgischen Eingriffen ist die Stummschaltung von Alarmen [76]. Gelegentlich wird die therapeutische Funktion eines Gerätes während einer Intervention für einen bestimmten Zeitraum deaktiviert. Beispielsweise übernehmen bei Herzoperationen Herz-Lungen-Maschinen sowohl die Funktion des Herzen als auch der Lunge, so dass zwischenzeitlich auf den Einsatz eines separaten Beatmungsgerätes verzichtet werden kann. Dies erfordert jedoch, dass die Anästhesisten bzw. Chirurgen das Beatmungsgerät reaktivieren, sobald die Herz-Lungen-Maschine nicht mehr läuft. Wird die Reaktivierung vergessen und sind zusätzlich die Alarme stumm geschaltet, kann infolgedessen ein Patient aufgrund fehlender Beatmung gesundheitliche Folgeschäden davontragen. *Smart Alarm Systems* adressieren genau solche Anwendungsfälle. Bezogen auf das vorhergehende Beispiel kann durch die Vernetzung von Anästhesiegerät, Herz-Lungen-Maschine und Monitoring-Einheit die Alarmsteuerung automatisiert werden. Die Alarme des Beatmungsgeräts werden unterdrückt, während die Herz-Lungen-Maschine arbeitet, und anschließend automatisch freigegeben.

Ein weiteres Beispiel aus dem ICE Standard ist das *Physiological Closed Loop Control*. *Closed Loop Control* ist ein Begriff aus der Steuerungstechnik und beschreibt ein selbstjustierendes Regelungssystem. Erfasste Daten werden in separaten Einheiten akkumuliert und fließen ins Quellsystem zur Verwertung zurück. Der ICE Standard

erläutert Physiological Closed Loop Control anhand einer Steuerung von Insulin- und Blutzuckerwerten. Dabei verabreichen zwei Infusionspumpen in Abhängigkeit des gemessenen Blutzuckerspiegels entsprechend Insulin oder Glukose, um die Werte in einem gesundheitlich unbedenklichen Bereich zu halten. Unerwartete Grenzüberschreitungen können über ein Alarmsystem an das klinische Personal weitergeleitet werden.

Der ICE Standard definiert einen weiteren Aspekt, der gleichzeitig Leitsatz der MD PnP ist: *Medical Device Plug-and-Play Interoperability*. Da alle vorhergehenden Anwendungsbeispiele ebenfalls auf Interoperabilität abzielen, liegt der Schwerpunkt dabei auf dem Begriff *Plug-and-Play*. Dessen Kernaussage ist, dass Geräte unabhängig vom Hersteller zur Laufzeit austauschbar sein sollten. Angeführt wird der Aspekt durch ein Beispiel aus der postoperativen Nachsorge. Ein Patient soll nach einem Aortenklappenersatz auf die Intensivstation verlegt werden und wird infolgedessen an ein Beatmungsgerät angeschlossen. Das Gerät als solches ist mit einem klinischen Informationssystem verbunden, welches alle Geräteparameter nachvollziehen kann. Bereits wenige Sekunden nach Aktivierung der mechanischen Beatmung wird ein Defekt am Gerät festgestellt, so dass es durch ein anderes ausgetauscht werden muss. Obwohl auch das neue Beatmungsgerät mit dem klinischen Informationssystem verbunden werden kann, ist eine Datenübernahme aufgrund proprietärer Protokolle und Hardware nicht ohne weitere Arbeitsschritte möglich. Offene und interoperable Kommunikationsstandards können den Austauschprozess wesentlich vereinfachen.

Ein weiteres Szenario, das technisch weniger detailliert beschrieben ist, kommt von Mauro et al. [79]. Sie diskutieren die SOA-basierte Integration von Medizingeräten in die IT-Landschaften des Krankenhauses. Hierbei betrachten sie insbesondere die betriebswirtschaftlichen Vorteile, welche sich aus den optimierten klinischen Prozessen ergeben. Der fiktive Patient Herr Müller soll aufgrund einer ambulanten Diagnose stationär aufgenommen und operiert werden. Der für die Intervention vorgesehene Behandlungspfad erfordert vorab eine *Magnetresonanztomographie (MRT)*. Es wird angenommen, dass sowohl die Geräte für die MRT als auch für die Operation serviceorientiert in die IT-Architektur des Krankenhauses eingebunden sind. Freie Geräte können automatisch gesucht und reserviert werden. Dies löst eine manuelle Buchung von Ressourcen ab, erspart den damit verbundenen Zeitaufwand und ermöglicht eine dynamischere Koordination von Reinigungs- und Wartungsarbeiten. Alle vorab erhobenen Daten wie die MRT-Bilder oder Vitalparameter des Patienten lassen sich über das KIS abfragen und in die medizinische Dokumentation übernehmen. Eine weitere Komponente kann die Daten zudem für klinische Studien verwerten. Zukünftig könnte sich hiermit ein Gesamtsystem ergeben, das Prozesse, Daten und Informationstechnologien nahtlos subsumiert.

Bohn et al. [9] beschreiben in ihrer Arbeit eine *Service-oriented Device Architecture (SODA)*, die den Fokus auf eine zentrale Steuereinheit (OP-Cockpit) für medizinische Geräte und den Datenaustausch zwischen OP-Planung, Intervention und der

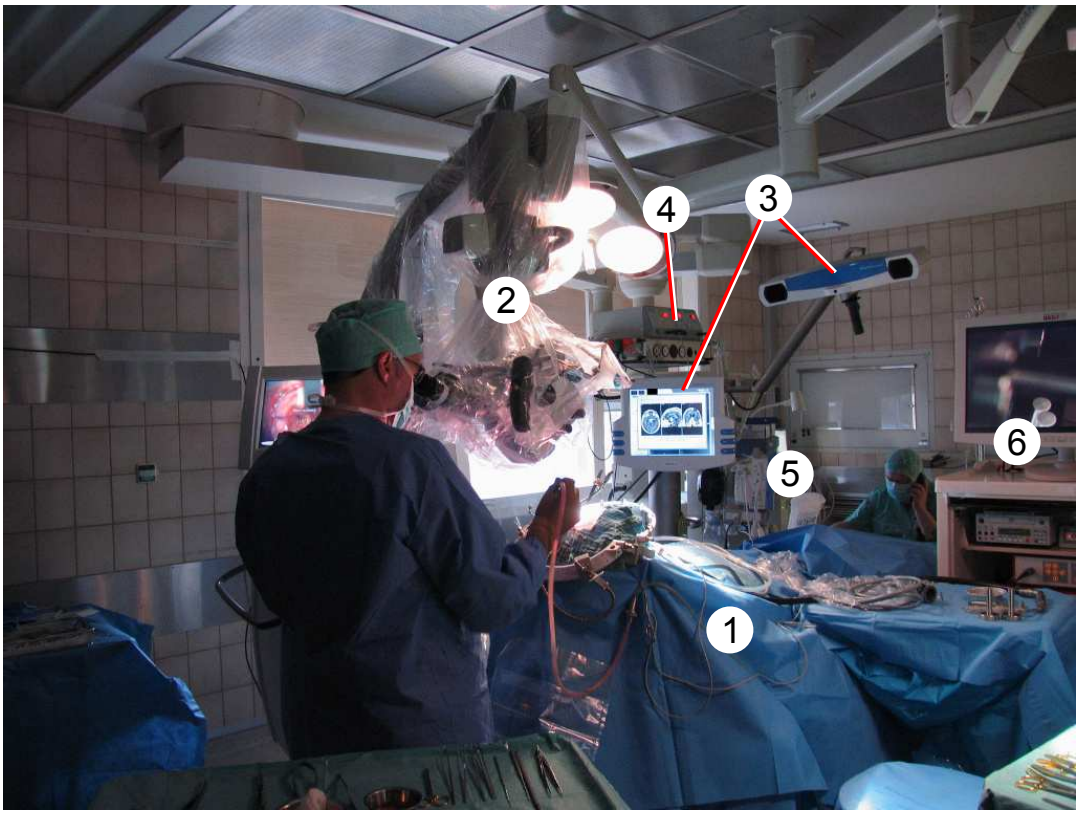


Abbildung 3.1.: Foto eines OP-Saals für Neurochirurgie am Universitätsklinikum Schleswig-Holstein in Lübeck. Zentral gelegen sind der OP-Tisch (1) und das Operationsmikroskop (2). Im Hintergrund ist das Tracking-System (3) mit dem Navigationsmonitor erkennbar. Direkt darüber befindet sich der Ultraschalldissektor (4); dahinter der Anästhesiearbeitsplatz (5). Rechts im Bild ist das Endoskop aufgebaut (6).

Dokumentation legt. Ein Single-Sign-On- und Single-Patient-Lookup-System sorgen für einen uniformen Zugang zum KIS, *Radiologieinformationssystem (RIS)*, *Picture Archiving and Communication System (PACS)*, etc. Die zentrale Steuereinheit ist steril gehalten und liefert den Benutzern ein einheitliches Schnittstellen- und Interaktions-Design. Das Integrationssystem wurde in verschiedenen neurochirurgischen Eingriffen klinisch bewertet und die Akzeptanz der Anwender anhand von Fragebögen festgestellt.

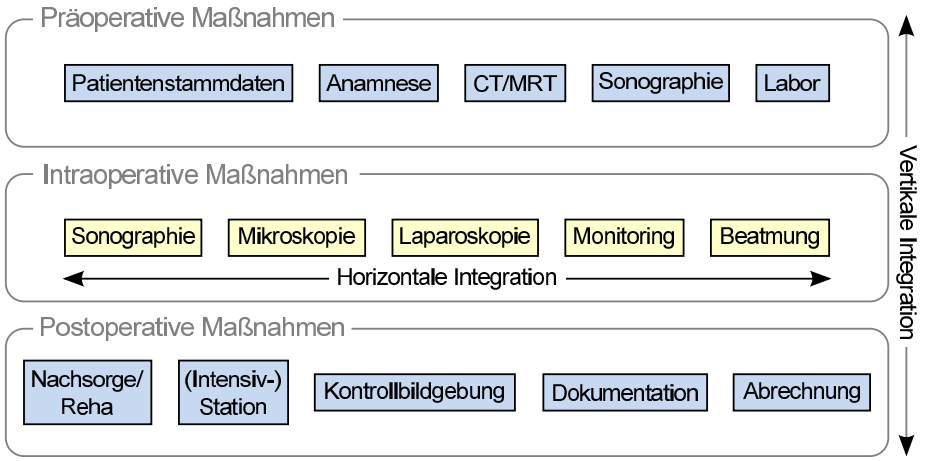


Abbildung 3.2.: Visualisierung vertikaler und horizontaler Integrationsrichtungen.

3.2. Analyse eines neurochirurgischen Eingriffs

Ähnlich wie Bohn et al. wird für die Akquise von Anwendungsfällen ein neurochirurgischer Eingriff (zur Entfernung eines Tumors) herangezogen. Um möglichst realitätsnahe Anforderungen zu erhalten, werden sowohl die Chirurgen, Anästhesisten als auch Assistenten bei der alltäglichen Arbeit beobachtet und die Arbeitsschritte hinsichtlich ihres Optimierungspotentials unter dem Aspekt der IT-Vernetzung evaluiert. Der Eingriff erfolgt im UKSH in Lübeck. Ein Foto des OP-Saals ist in Abbildung 3.1 dargestellt. Am chirurgischen Arbeitsplatz befinden sich ein Operationsmikroskop, ein Navigationssystem, ein Endoskop sowie ein Ultraschalldissektor. Der Anästhesiearbeitsplatz ist mit einem Vitaldaten-Monitor, einem Beatmungsgerät, einem Warmluftgenerator und einer Infusionspumpe ausgestattet. Weiterhin enthält der OP-Saal eine dedizierte Steuerungseinheit für Raumlicht und Klimaanlage, ein mobiles Blutgasanalysegerät sowie jeweils einen PC für die Verbindung zum KIS und zum *OP-Management-System (OPMS)*. Das OPMS ist im Allgemeinen für die OP-Planung vorgesehen und nutzt das KIS, um Patientendaten abzurufen. Es gibt weitere Datenquellen wie beispielsweise das RIS und das PACS.

Zur besseren Bewertung werden zwei Integrationsrichtungen unterschieden. Abbildung 3.2 illustriert diese beiden Richtungen. Unter *Vertikaler Integration* ist die Anbindung von Medizingeräten mit der IT-Infrastruktur des Krankenhauses zu verstehen. Darunter fällt zum Beispiel der Transport von Patientenstammdaten, Röntgenaufnahmen und Laborwerten zwischen einzelnen Abteilungen. Daten, die in präoperativen Maßnahmen erhoben werden, stehen im OP-Saal zur Verfügung. Während des Eingriffs aufgezeichnete Daten können (ggf. auch rückgekoppelt) für postoperative Maßnahmen wie medizinischer Dokumentation, Abrechnung oder Laborauswertung herangezogen

werden. *Horizontale Integration* beschreibt das Zusammenspiel verschiedener Geräte im OP-Saal oder in anderen dedizierten Bereichen wie der Intensivstation und Pathologie. Beispielsweise entspricht die Maschine-zu-Maschine-Interaktion am Anästhesiearbeitsplatz zwischen einem Monitoring-Gerät und einer Infusionspumpe einem horizontalen Integrationsszenario. Maßgebend ist in beiden Fällen ein gemeinsames Kommunikationssystem sowie die interoperable Codierung ausgetauschter Informationen.

Ein wesentlicher Unterschied beider Integrationsrichtungen besteht in den Anforderungen zum Antwortzeitverhalten. In diesem Zusammenhang unterscheidet man weiche, feste, harte und keine Echtzeit [169]. Dabei ist Echtzeit grundsätzlich durch einen Zeitrahmen definiert, in dem ein System den Abschluss einer Berechnung garantieren muss. Der Begriff sollte nicht mit der subjektiven Wahrnehmung einer Systeminteraktion verwechselt werden. Weiche Echtzeit definiert die zeitlichen Grenzen als Richtlinien. Überschreitungen sind in gewissen Rahmen tolerierbar. Bei Videoübertragungen beispielsweise führt die Überschreitung der Antwortzeit oder dem Ausfall von Datenpaketen lediglich zu einem leichten Ruckeln. Bei festen Echtzeitbedingungen wird die durchgeführte Aktion nach Überschreiten der Zeitbedingung wertlos und kann abgebrochen werden. Das ist zum Beispiel beim Abruf von Positionsdaten eines sich bewegenden Objektes der Fall. Harte Echtzeit stellt die höchsten Anforderungen an die Antwortzeit. Hier müssen die Zeitbedingungen garantiert werden, da ansonsten ein irreversibler Schaden entsteht. Dazu zählt beispielsweise das rechtzeitige Anhalten eines autonomen Fahrzeugs. *Keine Echtzeit* drückt lediglich aus, dass ein System asynchron reagiert und ein zeitlicher Rahmen nicht zugesichert werden kann. Das ist z. B. für E-Mail-Anwendungen der Fall.

Für die Mensch-Maschine-Kommunikation ist in der Regel die feste oder weiche Echtzeitbedingung ein ausreichendes Kriterium, da Antwortzeiten in Ausnahmefällen überschritten werden dürfen. In jedem Fall muss das System bei verlorenen oder zu spät eintreffenden Nachrichten in einen stabilen, risikoarmen Zustand versetzt werden. Die vertretbaren Antwortzeiten können dabei durchschnittlich bis zu 150[1] Millisekunden betragen, ohne dass es dem Systembenutzer auffällt [73]. Harte Echtzeit ist mit gängigen IT-Netzwerken (z. B. Ethernet) aufgrund der Best-Effort-Eigenschaft nicht umsetzbar. Für Applikationen mit harten Echtzeitanforderungen sind deshalb nach wie vor hoch spezialisierte (Bus-)Systeme einzusetzen. Für die vertikalen Integrationsszenarien gilt üblicherweise keine Echtzeitbedingung. Zum WWW oder E-Mail-Verkehr vergleichbare Antwortzeiten sind ausreichend.

Das MD PnP hat eine Vorgehensweise für den Entwicklungsprozess vernetzter IT-Systeme und Geräte im Krankenhaus definiert [34]. Abbildung 3.3 zeigt die einzelnen Schritte bis zur abschließenden technischen Umsetzung. Dieses Kapitel beschreibt die

[1] *150 Millisekunden* ist ein Durchschnittswert für die Wahrnehmung akustischer Signale. Die vertretbare Antwortzeit ist immer von mehreren Parametern abhängig wie der Art des Reizes (akustisch, visuell), dem Alter und physiologischer/psychologischer Randbedingungen.

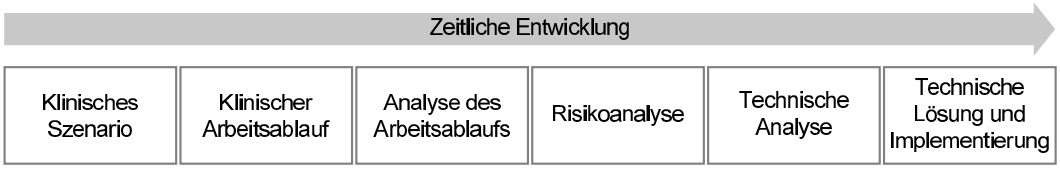


Abbildung 3.3.: Iterativer Arbeitsablauf für die Umsetzung eines klinischen Anwendungsfalls in ein technisches Endprodukt. Entwickelt vom MD PnP, in Anlehnung an Abbildung 1 aus [34].

klinischen Szenarien und Arbeitsabläufe auf einer hohen Abstraktionsebene. Es bildet somit den ersten Schritt auf dem Weg zu einem interoperablen Gesamtsystem. Jeder Anwendungsfall ist in die folgenden zwei Aspekte gegliedert.

1. Klinisches Szenario

Das klinische Szenario ist eine kurze Beschreibung einer Situation oder eines Ereignisses im klinischen Umfeld. Zweck dieser Beschreibung ist die Bereitstellung von klinischer Hintergrundinformation und die Motivation zur Entwicklung einer technischen Lösung. Das Szenario teilt sich wiederum auf in den aktuellen Stand der Technik (*Current State*) und einer möglichen Alternative auf Basis eines interoperablen Systems (*Proposed State*). Dabei wird angenommen, dass alle Medizingeräte bereits nahtlos miteinander verbunden sind und es keine technischen, ökonomischen, rechtlichen oder regulatorischen Hindernisse gibt.

2. Clinical Concept of Operations

Der Begriff *Clinical Concept of Operations (CConOps)* wurde vom MD PnP geprägt und entspricht einer detaillierten Beschreibung der an einem klinischen Szenario beteiligten Geräte und Personen. Dabei ermöglicht CConOps eine Verbesserung der Sicherheit und Effektivität eines Szenarios mit Hilfe einer speziellen Lösung für den Proposed State. CConOps werden anhand folgender Punkte charakterisiert:

Gerätetypen, IT-Systeme und Software Beschreibung der genutzten Geräteausstattung samt Ein- und Ausgaben, Sensorik sowie den Interaktionen mit anderen Geräten oder Systemen. Zudem sollten ggf. betroffene Systeme wie PACS, KIS und RIS genannt werden. Die Beschreibung der Gerätetypen, IT-Systeme und Software kann außerdem Kommunikationsflüsse und -verbindungen skizzieren.

Klinische Prozesse Benennung der Auswirkungen auf Prozesse aus klinischer als auch betriebswirtschaftlicher Sicht.

Involvierte Nutzer Aufzählung des vom Szenario betroffenen Personals. Das könnten zum Beispiel Chirurgen, Anästhesisten, Intensivmediziner und Assistenten sein.

Neue Ausstattung oder Prozesse Angabe neuartiger Geräte und Produkte, welche zum gegenwärtigen Zeitpunkt noch nicht existieren, jedoch eine wesentliche Verbesserung des Szenarios bewirken würden.

Vorteile des Proposed State Details zu den positiven Auswirkungen des Proposed State hinsichtlich Sicherheit, Effizienz oder Teamwork.

Risikoanalyse und mögliche Entschärfungen Die Risikoanalyse beschreibt negative Effekte auf die Patientensicherheit. Dabei gilt, dass es Risiken geben darf, solange sie beherrscht oder entschärft werden können.

Die im Rahmen dieses Kapitels untersuchten Anwendungsfälle sind nach dem oben beschriebenen Schema der MD PnP strukturiert. Es ist nicht ausgeschlossen, dass einige Aspekte redundant oder optionaler Natur sind und deshalb aus der Beschreibung entfallen. Grundsätzlich gilt, dass sich Szenarien nur dann als nützlich erweisen, wenn das klinische Personal entweder genauso viele oder weniger Arbeitsschritte durchführen muss, um ein Ziel zu erreichen. Eine Ausnahme für diese Regel ist jedoch, dass die signifikante Erhöhung der Patientensicherheit auch mehr Arbeitsschritte generieren darf.

Bei einem weiteren OP-Besuch am UKSH und im Rahmen eines Hands-on Chirurgiekurses an der Technischen Universität München [21] wurden die vorliegenden Szenarien in ihrem Nutzen bestätigt. Im Vergleich mit den Funktionen aus der von Bohn et al. [9] beschriebenen Architektur bestätigen sich insbesondere das zentrale OP-Cockpit sowie die Kombination von prä-, intra- und postoperativen Informationen.

3.2.1. Vertikale Integration

Die vertikale Integration spielt bei allen Operationsarten eine wichtige Rolle. Im Allgemeinen ist zu beobachten, dass Geräte bzw. einzelne Geräteverbünde als Insellösungen konzipiert sind und nicht mit dem KIS, OPMS oder sonstigen angrenzenden IT-Systemen kommunizieren können. Die folgenden drei Anwendungsfälle beschreiben den möglichen Nutzen der dynamischen Vernetzung mit angrenzenden IT-Landschaften.

Klinisches Szenario: Übertragung der digitalen Patientenakte

Im OP-Saal befinden sich dedizierte Rechner für den Zugang zum KIS und OPMS. Die OP-Assistenz nutzt die Rechner, um OP-Planungsdaten und Patientendaten auszulesen. Die passende Patienten-ID muss auf beiden Geräten selektiert und (aus rechtlichen Gründen) von einem verantwortlichen, menschlichen Akteur bestätigt

werden. Eine Verteilung der Patientenakte auf einzelne Medizingeräte findet nicht statt. Außerdem ist die manuelle Eingabe der Daten an den Medizingeräten notwendig. Sollte beispielsweise eine Anbindung über DICOM-Arbeitslisten existieren, ist vorab typischerweise eine Bestätigung der ausgelesenen Daten an jedem einzelnen DICOM-kompatiblen Gerät erforderlich. Hiermit kann zwar auf die manuelle Dateneingabe verzichtet werden, die separate Bestätigung erzeugt dennoch zahlreiche Arbeitsschritte.

Eine Arbeitserleichterung entsteht, wenn die Daten automatisch vom KIS an alle Geräte übertragen und nur einmal an einem Gateway-Rechner bestätigt werden müssen. Der Gateway-Rechner ist hierbei eine vereinheitlichte Schnittstelle aus allen Datenquellen, die über angrenzende IT-Systeme an den OP-Saal gekoppelt sind. Für vernetzte und zum Gateway kompatible Geräte entfällt der fehleranfällige Prozess der manuellen Dateneingabe. Das Personal muss nur einen Datensatz bestätigen.

CConOps: Übertragung der digitalen Patientenakte

An der Übertragung der digitalen Patientenakte bzw. weiterer OP-Planungsdaten sind alle Geräte beteiligt, die derartige Information verwerten können. Das sind im Rahmen des untersuchten Eingriffs z. B. das OP-Mikroskop, das Navigationssystem, die Endoskopieeinheit, das Blutgasanalysegerät und der Anästhesiearbeitsplatz. Die Bündelung relevanter Daten erfolgt über den Gateway-Rechner. Für das Personal entsteht eine unmittelbare Arbeitsentlastung, da insgesamt weniger Arbeitsschritte notwendig sind, um die administrativen Aufgaben vor dem eigentlichen Eingriff durchzuführen. Eine spürbare Kostenersparnis für den Betreiber wird erst entstehen, wenn alle Geräte interoperabel sind und ausgetauscht werden können, ohne aufwendige Neukonfigurationen in Auftrag geben zu müssen. Nutzer des Systems sind vorzugsweise die OP-Assistenten und Chirurgen. Ein typisches Risiko in diesem Szenario entsteht dadurch, dass Patientendatenquellen falsch identifiziert werden oder Fehler bei der Übertragung entstehen. Durch den Einsatz digitaler Signaturen und Protokollierungsverfahren ist die Vermittlung falscher Daten vermeidbar bzw. hinsichtlich der Nachweisbarkeit im Schadenfall zurückführbar. Es muss sichergestellt sein, dass aus der Verwertung der transferierten Daten keine unmittelbare Gefahr für den Patienten entsteht, andernfalls sind weitere Mechanismen notwendig, um die korrekte Zustellung eines Datums zu garantieren. Beispielsweise entsteht durch die Übertragung des Gewichts zur Berechnung einer Narkosedosis ein höheres Risiko als durch die Übertragung des Namens zur Anzeige auf einem Display.

Klinisches Szenario: Berichterstattung pathologischer Befunde

Während der Entfernung eines Tumors aus der Gehirnregion wird das extrahierte Gewebe von der OP-Assistenz in die Pathologie gebracht. In einem Schnelltestverfahren

kann die Dignität[2] des Tumors intraoperativ diagnostiziert werden. Gutartige Tumore wachsen lokal begrenzt und dringen nicht in benachbartes Gewebe ein. Die Chancen für den Patienten stehen gut, bei einer vollständigen Entfernung zu genesen. Bösartige Tumore hingegen bilden Metastasen und richten damit massiven Schaden an den Zellen an. Mit dem pathologischen Befund kann der Chirurg entscheiden, ob die weitere Entfernung des Tumors sinnvoll ist, oder ob der Eingriff aufgrund des nicht unerheblichen Risikos für den Patienten abgebrochen wird. Das Ergebnis wird hierbei telefonisch mitgeteilt. Da die OP-Assistenz nicht ausreichend dazu qualifiziert ist, die Information weiterzureichen, wird stattdessen der Telefonhörer an das Ohr des Chirurgen gehalten.

In einem integrierten System hingegen wird der pathologische Befund über ein *Laborinformationssystem (LIS)* bis in den OP-Saal übermittelt und dem Chirurgen in einem für ihn vorgesehenen Sichtbereich eingeblendet. Die OP-Assistenz wird bei der Arbeit entlastet und der unsterile Telefonhörer gefährdet nicht den sterilen Bereich. Darüber hinaus entfällt der fehlerträchtige, telefonische Informationsweg.

CConOps: Berichterstattung pathologischer Befunde

Für die Bereitstellung der pathologischen Befunde ist der Zugriff auf ein LIS erforderlich. Bei einem neurochirurgischen Eingriff ist die Einblendung des Ergebnisses im Sichtfeld des OP-Mikroskops zielführend. Es ist darüber hinaus denkbar, jeden beliebigen Monitor für die Informationsvisualisierung zu nutzen (z. B. das Display des Navigationsgerätes oder des Endoskops). Im Prozess involviert sind nach wie vor der Chirurg, die OP-Assistenz als auch der Pathologe. Die OP-Assistenz erfährt jedoch eine Arbeitsentlastung, da sie lediglich die Probe zum Labor bringen muss und nicht mehr als Mittelsmann für die Übertragung des Befundes zuständig ist. Für den Patienten sinkt das Risiko einer Infektion durch das unsterile Telefon, auch ist die Gefahr geringer, dass durch Signalstörungen im Telefon Fehlinterpretationen stattfinden. Es kann vorkommen, dass der Befundbericht bei der Übertragung verfälscht wird. Digitale Signaturen verhindern die Korrumpierung derartiger Daten. Es entsteht demnach kein weiteres Risiko, welches nicht auch vorher schon gegenwärtig war. Fällt das Netzwerk aus, ist weiterhin der telefonische Weg zur Übermittlung des Befundes möglich.

Klinisches Szenario: Automatisierte Dokumentation

Nach einem Eingriff fällt üblicherweise verschiedenes Dokumentationsmaterial an. In diesem Szenario ist Bildmaterial vom Operationsmikroskop und Navigationsgerät erhoben worden. Denkbar sind jedoch auch Vitalparameter und in Zukunft auch weitere

[2] Die Dignität beschreibt die Eigenschaft eines Tumors, ob er gut- oder bösartig ist.

Daten wie beispielsweise die Leistungsregulation und Aktivierungszeit elektronischer Schneidinstrumente. Die Bilder werden separat auf CD/DVD oder USB-Stick kopiert und später manuell weiterverarbeitet.

Der vollintegrierte OP erfordert keine manuellen Kopiervorgänge. Das erhobene Dokumentationsmaterial wird semantisch annotiert (u. a. mit den Stammdaten aus der elektronischen Patientenakte) und zur Weiterverarbeitung über das IT-Netzwerk an den Büro-PC des Chirurgen geleitet bzw. in einer Krankenhausdatenbank gespeichert.

CConOps: Automatisierte Dokumentation

An der automatisierten Dokumentation sind alle Geräte beteiligt, die Bildaufnahmen, Videosequenzen, Tonaufnahmen oder strukturierte Datensätze generieren sowie mindestens ein Speicherort, in der die Daten abschließend oder temporär gespeichert werden. Die Senke könnte hierbei das zentrale Datenbanksystem des Krankenhauses sein oder ein PC, der für die Nachbearbeitung vorgesehen ist. Da die Abfertigung der Daten automatisch erfolgt, ist kein weiteres Personal in dem Szenario involviert. Lediglich der Nutzer, der anschließend die Dokumentationsdaten auswertet, ist am Prozess beteiligt. Anhand der automatisierten Dokumentation können logistische als auch Sachkosten eingespart werden, da fortan keine physikalischen Datenträger mehr transportiert werden müssen. Damit erübrigt sich außerdem die Lagerung und Inventarisierung von CDs/DVDs oder USB-Sticks. Der Chirurg kann sich nach dem Eingriff darauf verlassen, dass das generierte Dokumentationsmaterial auf seinem Büro-PC verfügbar ist. Für den Patienten entsteht kein weiteres Risiko. Es kann maximal zu einem Datenverlust kommen, der auch bei der Übertragung via physikalischer Datenträger möglich ist.

3.2.2. Horizontale Integration

Weitere Anwendungsfälle sind im Rahmen der horizontalen Integrationsanalyse entstanden. Dabei wurden zahlreiche kleine Arbeitsschritte identifiziert, die durch adäquate Konnektivität automatisiert oder eingespart werden können.

Klinisches Szenario: Automatische Regulation der Körperkerntemperatur

Um die Transpiration des Chirurgen bei der körperlich anspruchsvollen Arbeit möglichst gering zu halten, wird mit Hilfe der Klimaanlage eine kühle Umgebungstemperatur im OP-Saal erzeugt. Die reduzierte Temperatur wiederum kann jedoch aufgrund des narkotisierten Zustandes in einer Unterkühlung des Patienten resultieren. Um

dies zu vermeiden, setzt man ein Wärmeluftgerät ein, das ausschließlich die Körperkerntemperatur des Patienten beeinflusst. Zwischendurch muss das Gerät manuell reguliert werden, um die Körperkerntemperatur in einem unbedenklichen Bereich zu halten.

Im vernetzten OP-Saal erfolgt ein automatischer Abgleich des Wärmeluftgerätes mit dem Temperatursensor der Monitoring-Einheit. Ist die Körperkerntemperatur des Patienten zu hoch, schaltet sich das Wärmeluftgerät ab, bis sich der Zustand wieder normalisiert hat.

CConOps: Automatische Regulation der Körperkerntemperatur

An diesem Szenario sind exakt zwei Geräte beteiligt: ein Wärmeluftgerät und eine Monitoring-Einheit mit einem Sensor für die Körperkerntemperatur. Die Körperkerntemperatur wird weiterhin vom Anästhesisten verifiziert. Im Normalbetrieb reguliert sich das Wärmeluftgerät jedoch in Abhängigkeit des Temperaturwertes der Monitoring-Einheit von selbst. Wärmeluftgeräte verfügen üblicherweise über eine primitive elektronische Steuerung ohne Netzwerkanbindung. Integrierbare Geräte müssten deshalb erst in den Markt gebracht werden. Der klinische Nutzen ist insgesamt gering, so dass dieser Anwendungsfall als Komfortfunktion angesehen werden kann. Eine Verbesserung der Patientensicherheit entsteht jedoch in dem Augenblick, wenn die Körperkerntemperatur einen bedenklichen Wert annimmt und der Alarm der Monitoring-Einheit stummgeschaltet ist. Der Anästhesist könnte diesen Alarm unbeabsichtigter Weise übersehen. Da das Wärmeluftgerät die Überschreitung der Körperkerntemperatur erkennt, kann es entsprechend reagieren. Fehler in der Übertragung führen unter Umständen dazu, dass falsche Temperaturwerte kommuniziert werden. Durch zweikanalige Übertragung respektive digitaler Signaturen kann die Vermittlung korrekter Werte abgesichert werden.

Klinisches Szenario: Gleichzeitige Auslösung von Screenshots an mehreren Geräten

Das Operationsmikroskop nimmt im neurochirurgischen Umfeld eine besondere Rolle ein, da es das Zentrum des Eingriffs darstellt. Endoskope bieten häufig nicht die nötige Dynamik und Brennweite, um effizient in den filigranen Strukturen des Gehirns eingesetzt werden zu können. Der Operateur ist demnach in der Lage, Knöpfe am Operationsmikroskop zu bedienen, wobei andere Geräte oftmals außer Reichweite sind. Während der OP werden zu Dokumentationszwecken Screenshots vom Sichtfeld des Mikroskops und dem Navigationsbildschirm gemacht. Für die Screenshot-Funktion des Mikroskops liegt ein Fußschalter bereit. Das Navigationsgerät verfügt lediglich über eine nicht-sterile, drucksensitive Bildschirmoberfläche und ist für eine fachgerechte Bedienung zu weit vom Operateur entfernt. Für jeden Screenshot muss deswegen die

OP-Assistenz herbeigerufen werden. Auf eine verbale Absprache hin aktivierten beide gemeinsam den jeweiligen Auslöseschalter.

Durch die Vernetzung von Operationsmikroskop und Navigationsgerät kann ein synchronisierter Screenshot bei betätigtem Mikroskop-Fußschalter erfolgen, so dass die OP-Assistenz nicht mehr beauftragt werden muss.

CConOps: Gleichzeitige Auslösung von Screenshots an mehreren Geräten

Wie bereits erwähnt, sind in diesem Szenario das Operationsmikroskop und das Navigationssystem beteiligt. Wie die Bilder anschließend verwertet werden, ist Teil der vertikalen Integration. Der Screenshot wird entweder vom Chirurgen selbst oder vom assistierenden Chirurgen initiiert. Es wäre auch denkbar, die Funktion mit Hilfe eines zentralen Bedienfeldes von der OP-Assistenz ausführen zu lassen. Durch den gemeinsamen Screenshot verringert sich die zeitliche Diskrepanz zwischen den Bildern. Die OP-Assistenz muss bestenfalls die Arbeit nicht mehr unterbrechen und schwer zugängliche Gerätschaften bedienen. Für den Patienten resultiert kein weiteres Risiko. Durch eine Rückmeldung kann signalisiert werden, ob der gemeinsame Screenshot erfolgt ist, so dass im negativen Fall auf die manuelle Bedienung ausgewichen werden kann.

Klinisches Szenario: Fernsteuerung elektronischer Schneidinstrumente

Im Zusammenhang mit dem Operationsmikroskop gibt es einen weiteren Anwendungsfall. Üblicherweise wird das Mikroskop über dem Kopf des Patienten positioniert und befindet sich somit auch in unmittelbarer Nähe zum Chirurgen. Das Mikroskop ist für die Mensch-Maschine-Interaktion mit Fußschaltern und Handstücken ausgestattet. Einige Mikroskope erlauben die Einblendung von Information im Sichtfeld des Mikroskops oder auf einem externen Monitor. Bei einem neurochirurgischen Eingriff zur Entfernung eines Tumors kommt zudem ein elektronisches Schneidinstrument zum Einsatz. Diese Geräte sind häufig außer Reichweite vom Chirurgen platziert. Abbildung 3.1 auf Seite 38 illustriert diesen Sachverhalt. Das elektronische Schneidinstrument befindet sich auf einem anderen Gerät gestapelt, etwa zwei Meter entfernt vom Chirurgen. Für die Regulation der elektronischen Schneidleistung muss ein Dritter beauftragt werden. In diesem Fall stand der Anästhesist auf und erhöhte oder reduzierte ohne Sicht auf die Bedienelemente die Schneidleistung, bis der Chirurg „stopp“ rief.

Im vollintegrierten Operationssaal kann das elektronische Schneidinstrument ferngesteuert werden. Der Chirurg reguliert die Leistung über einen Fußschalter oder ein Handstück. Um die häufige Akkommodation der Augen zu vermeiden, werden die Änderungen im Sichtfeld des Mikroskops eingeblendet. Um Fußschalter einzusparen, kann

die Aktivierung des Schneidinstruments ebenfalls mit Hilfe des Mikroskopfußschalters gesteuert werden.

CConOps: Fernsteuerung elektronischer Schneidinstrumente

In diesem Szenario ist ein elektronisches Schneidinstrument im Fokus der Betrachtung. Das können Hochfrequenz- oder Ultraschalldissektoren sein. Als Gegenstück kommt ein entferntes Gerät wie das Operationsmikroskop oder eine andere zentrale Steuereinheit in Frage. Vom Anwendungsfall ist nur der Chirurg betroffen. Die Notwendigkeit, einen Dritten in die Regulation der Leistung einzubeziehen, entfällt. Viele elektronische Schneidinstrumente verfügen heutzutage noch nicht über physikalische Schnittstellen; vermutlich, weil bisher kein Bedarf zur Vernetzung vorgesehen war und das Schneidinstrument ein sehr sensibles, invasives Gerät darstellt. Die Fernsteuerung kann die Sicherheit für den Patienten erhöhen, weil die Wahrscheinlichkeit für Fehleinstellungen reduziert wird. Sind die Leistungswerte im Sichtfeld des Mikroskops sichtbar eingeblendet, bedarf es außerdem keiner Akkommodation der Augen, was die körperliche Beanspruchung für den Chirurgen positiv beeinflussen kann. Aus Sicht der Risikoanalyse stellt die Fernsteuerung des elektronischen Schneidinstruments eine besondere Herausforderung dar. Die Übermittlung der Geräteleistung und Steuerungsdaten muss garantiert werden können. Fällt das Netzwerk aus, müssen Werte weiterhin von einem Gerätedisplay ablesbar sein. Weitaus schwerwiegender ist die Aktivierung respektive Deaktivierung des Schneidkopfes. Aus technischer Sicht ist zu gewährleisten, dass ein Verbindungsabbruch die Schneidwirkung unmittelbar beendet bzw. die Aktivierung durch verspätet eintreffende Nachrichten zurückgehalten wird. Es ist sicherzustellen, dass es immer ein Reservesystem für die Bedienung gibt. Das Reservesystem ist üblicherweise die herkömmliche Geräteschnittstelle.

Klinisches Szenario: Automatisierte Übernahme von Blutgasanalysewerten

In jeder Vollnarkose-OP ist eine *Blutgasanalyse (BGA)* für die Überwachung und Steuerung von Beatmungsparametern vorgesehen. BGA-Geräte sind typischerweise mobile Einheiten, die zur selben Zeit von verschiedenen OP-Sälen eingesetzt werden können. Während der OP wird in regelmäßigen Abständen eine BGA durchgeführt. Hierzu bringt die OP-Assistenz Blutproben zum BGA-Gerät und überreicht dem Anästhesisten anschließend einen Zettel mit ausgedruckten Analysewerten. Diese wiederum überträgt der Anästhesist auf ein separates Protokollformular.

Sind Anästhesie-Arbeitsplatz und BGA-Gerät interoperabel, erübrigt sich der Ausdruck sowie die mehrfache manuelle Übertragung der Messwerte. Das BGA-Gerät sendet die Daten über das Netzwerk zum Anästhesie-Arbeitsplatz. Dort können die Werte gegengeprüft und vom Anästhesisten bestätigt werden. Die Dokumentation der Werte geschieht im Rahmen des postoperativen Vorgangs in einem automatisierten,

vertikal integrierten Verfahren. Der Analysevorgang dauert typischerweise mehrere Minuten. Ein BGA-Gerät stellt deshalb zusätzlich einen Dienst bereit, der über den Abschluss der Analyse informiert.

CConOps: Automatisierte Übernahme von Blutgasanalysewerten

Zentrum dieses Anwendungsfalls sind das BGA-Gerät und der Anästhesie-Arbeitsplatz. Welches Gerät des Anästhesie-Arbeitsplatzes die BGA-Werte empfängt, ist hierbei irrelevant. Voraussetzung ist, dass die Werte dem Anästhesisten auf digitalem Weg zur Verfügung gestellt werden können. Nutzer in dem Szenario sind der Anästhesist und die OP-Assistenz. Beide werden in ihrer Arbeit entlastet. Zusätzlich wird die fehleranfällige, manuelle Datenübertragung vermieden. Ein unmittelbares Risiko für den Patienten entsteht nicht. Wie in den vorhergehenden Szenarien ist dazu die Übermittlung korrekter Werte essentiell.

Klinisches Szenario: Zentrales OP-Cockpit

Die OP-Assistenz hat während eines Eingriffs zahlreiche Aufgaben zu bearbeiten. Darunter fallen die Bedienung des OPMS, der Transport von Blutproben, die Annahme von Telefonaten sowie die Steuerung einzelner Medizingerätefunktionen und der Raumtechnik. Da die Geräte im gesamten OP verteilt sind und typischerweise Platzprobleme vorherrschen, muss die OP-Assistenz für jede Interaktion durch den gesamten Saal laufen und die Geräte auf Zuruf bedienen.

Im vollintegrierten OP-Saal werden Bedienelemente an einer zentralen Stelle gebündelt. Das *zentrale OP-Cockpit* liefert Auskunft über die Betriebsbereitschaft beteiligter Geräte und kann Vitalparameter visualisieren. Gerätefunktionen lassen sich bei Bedarf über uniforme Mensch-Maschine-Schnittstellen fernsteuern.

CConOps: Zentrales OP-Cockpit

Für diesen Anwendungsfall kommen alle Geräte im OP-Saal in Frage. Die OP-Assistenz profitiert von einem OP-Cockpit am meisten, da alle Einstellungen von einer zentralen Instanz aus beobachtet und gesteuert werden können. Durch die eingesparte Zeit kann sich die OP-Assistenz auf ihre Kernaufgaben konzentrieren. Für den Chirurgen hat das OP-Cockpit den Vorteil, den Status wie z. B. Kalibrierungsinformationen bestimmter Geräte vor dem Eingriff einzusehen, um damit wiederum die Betriebsbereitschaft folgern zu können. Ein OP-Cockpit stellt ein neuartiges Gerät oder eine neuartige Software dar. Solche Cockpits findet man derzeit nur in Komplett-Systemen. Für den dynamisch vernetzten Operationssaal wäre zukünftig die Umsetzung in einer kompakten Form wie Tablet-PCs oder als Software-Erweiterung in bestehenden

Gateway-Rechnern denkbar. Mit dem OP-Cockpit ergeben sich zahlreiche Risiken, wie aus den vorhergehenden Szenarien bereits ersichtlich wurde. Auch hier gilt, dass ein solches System eine Komforteinrichtung darstellt, die bei einem Ausfall durch alternative Eingaben kompensierbar sein muss. Die Kommunikationswege müssen gemäß der Risikoklasse einer Funktion gegen Verfälschung und unterlassener Zustellung abgesichert werden.

3.3. Umsetzung einer SOA-basierten Middleware

Es existieren bereits zahlreiche Modelle, nach denen eine SOA-basierte Herangehensweise für Interoperabilitätsprobleme zwischen medizinischen Geräten erfolgen kann. Verschiedene Arbeiten [111, 45, 31, 78] zu dem Thema haben gezeigt, dass DPWS für die Integration von Medizingerätesystemen prädestiniert ist. Die Arbeiten basieren jedoch im Wesentlichen auf theoretischen Konzepten und virtuellen Geräteumsetzungen. Ziel dieses Abschnitts ist die realitätsnahe Umsetzung einer SOA-Lösung auf Basis von DPWS. Eine Machbarkeitsstudie beschreibt den Stand der Technik und die Anforderungen an existierende Geräte. Sie bestätigt den Nutzen der ausgearbeiteten Konzepte und zeigt, in welchen Bereichen weiterer Forschungsbedarf besteht.

3.3.1. Hardware-Analyse

Zunächst wird untersucht, welche hardware-seitige Technik in den typischen Medizingeräten neurochirurgischer Eingriffe verbaut ist. Drei Geräte werden betrachtet:

- Ein Ultraschalldissektor der Firma Söring GmbH
- Ein neurochirurgisches Operationsmikroskop mit Videodokumentationssystem der Firma MÖLLER-WEDEL GmbH
- Ein Monitoring-Gerät mit angeschlossener Herzfrequenz-Sensorik der Firma Drägerwerk AG & Co. KGaA

Der Ultraschalldissektor verfügt weder über eine virtuelle noch physikalische Schnittstelle. Solche Geräte sind für die anvisierten Vernetzungsszenarien unbrauchbar, da ein nachträglicher Verbau von Computer-Hardware weder praktikabel noch aus regulatorischer und monetärer Sicht durchführbar ist. In diesem Fall ist es notwendig, neue Geräte in den Markt zu bringen, die über entsprechende Hardware-Voraussetzungen verfügen und mit modernen Platinen ausgestattet sind. Derartige Entwicklungen sind bei Söring inzwischen in Planung. Geräte dieser Art verfügen im Allgemeinen über Anschlüsse für verschiedene Handstücke, einen Fußschalter zur Aktivierung des Schneidvorgangs und Bedienelemente zur Regulierung und Anzeige der Schneidleistung.

Das Operationsmikroskop setzt sich ebenfalls aus mehreren Komponenten zusammen. Der Mikroskopkopf besteht aus einem Okular, einer Videokamera und zwei Handstücken. Eine separate Lichtquelle beleuchtet des Mikroskopsichtfeld. Der Mikroskopkopf hängt an einem Stativ, das mit einer digitalen Stativsteuerung, einem Rack für weiteres Equipment und einer RS-232-Schnittstelle ausgestattet ist. Ein Videodokumentationssystem kann für die Aufzeichnung von Bildmaterialien und zur Stativsteuerung verwendet werden. Die Rechenleistungsparameter und physikalischen Schnittstellen (u. a. *Local Area Network (LAN)*) sind vergleichbar mit denen herkömmlicher PCs. Das Stativ tauscht Daten über ein proprietäres Binärprotokoll aus.

Das betrachtete Monitoring-Gerät unterstützt sowohl RS-232 als auch LAN, kommuniziert jedoch über die proprietäre Infinity-over-LAN-Schnittstelle der Firma Dräger. Andere proprietäre Schnittstellen sind vorhanden, um beispielsweise verschiedene Sensorik anzuschließen. Zur Laufzeit können zahlreiche Vitalparameter gemessen und visualisiert sowie verschiedene Alarme konfiguriert werden.

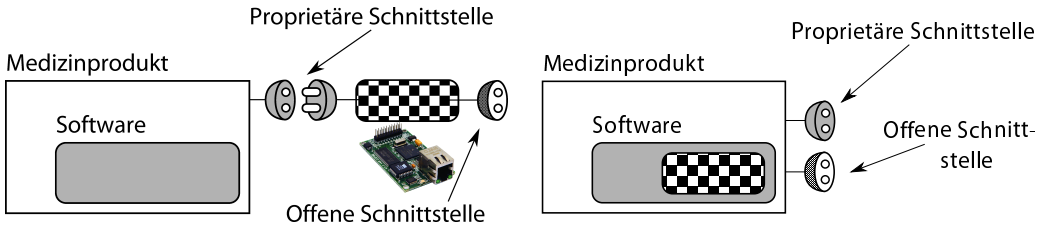
Die oben genannte Geräteaufstellung entspricht keiner vollständigen OP-Saal-Ausrüstung, bildet jedoch eine Untermenge, mit der die wesentlichen Aspekte für beliebige Gerätekonstellationen erfasst werden können. Üblicherweise können folgende Merkmale unterschieden werden:

1. Geräte ohne physikalischer Kommunikationsschnittstelle
2. Geräte mit physikalischer, veralteter Schnittstelle (z. B. RS-232)
3. Geräte mit physikalischer, moderner Schnittstelle (LAN)

In Abhängigkeit der vorliegenden technischen Ausstattung kann die Integration eines Medizinproduktes auf verschiedenen Komplexitätsstufen angeordnet sein. Geräte ohne physikalischer Schnittstelle erfordern zunächst die Bestückung mit neuer Hardware. Die Hardware kann so gewählt werden, dass sie den Anforderungen für den Betrieb der Integrations-Software genügt. Wie oben bereits erwähnt, können existierende Medizingeräte nicht ohne weiteres mit neuer Hardware nachgerüstet werden. Stattdessen müssen sie als *neue* Produktlinie in den Markt gebracht werden und den Zulassungsprozess unter Einbezug der neuen Komponente durchlaufen.

Proprietäre Schnittstellen sind vorzugsweise mit Hilfe des *Legacy Wrapper Pattern* [29] zu integrieren (vgl. Abbildung 3.4(a)). Ein spezialisierter Kleinstrechner wird mit der proprietären oder veralteten physikalischen Schnittstelle verbunden. Der Kleinstrechner transformiert die Daten aus dem Medizingerät in ein integrierbares Format und stellt sie über eine moderne und standardisierte physikalische Schnittstelle als Dienst zur Verfügung.

Im einfachsten Fall verfügt ein Medizinprodukt bereits über ausreichende Ressourcen zur Kommunikationsanbindung. Die Programmlogik für die Kommunikation wird



(a) Wrapper-Pattern (b) Standardumsetzung

Abbildung 3.4.: Vergleich zwischen einer Standardumsetzung und dem Einsatz des Wrapper-Patterns zur Integration von Altgeräten in interoperable Medizingeräteumgebungen. Links wird die Integrations-Software in der vorhandenen Software des Medizinprodukts ergänzt und über eine passende physikalische Schnittstelle veröffentlicht. Das Wrapper-Pattern in der rechten Abbildung macht die physikalische Schnittstelle über eine separate Rechnereinheit zugänglich. Die karierten Flächen symbolisieren die Logik für die Integrationsschnittstelle; die grauen Flächen entsprechend die bereits vorhandene Medizingeräte-Software.

in die bestehende Geräte-Software integriert, wie in Abbildung 3.4(b) dargestellt. Darüber hinaus ist es denkbar, die Programmlogik als nebenläufigen Prozess auf dem Betriebssystem des Medizingerätes auszuführen.

Im Rahmen dieser Umsetzung wird für den Dissektor stellvertretend ein Notebook eingesetzt. Das Gerät ist demnach ausschließlich virtuell vorhanden. Das Operationsmikroskop verfügt über ein Dokumentationssystem, das einen vollständigen Rechner mit Ethernet-Schnittstelle liefert. Die Dokumentationssoftware kann so manipuliert werden, dass sie das DPWS-Protokoll spricht. Das Monitoring-Gerät wird mit dem Legacy Wrapper Pattern ans Netzwerk gekoppelt.

3.3.2. Middleware-Analyse

DPWS bildet ein Profil aus zahlreichen Web-Service-Erweiterungen ab. Zweckmäßige Beschränkungen eliminieren die aus den einzelnen Erweiterungen hervorgehenden Mehrdeutigkeiten zur Steigerung der Interoperabilität. Dennoch verbirgt sich hinter der Spezifikation ein hoher Komplexitätsgrad, der gegen die Neuentwicklung einer konformen Middleware spricht. Für die Umsetzung ist deshalb der Einsatz einer existierenden Middleware sinnvoll. Zu beachten ist hierbei jedoch, dass *Off-the-shelf-software* oder *Software of unknown provenance* einen erheblichen Mehraufwand für das Qualitätismanagement medizinischer Software bedeutet. Tabelle 3.1 zeigt eine Auswahl der gängigsten Web-Service-Middleware-Produkte und ihre DPWS-Eignung.

Name	Sprache	Lizenz	DPWS 1.0	DPWS 1.1
gSOAP	C/C++	GPL + gSOAP		
Axis2/C	C	Apache		
WS4D-gSOAP	C	GPL/LGPL	X	
SOA4D-DPWS-Core	C	GPL	X	X
WSDAPI	C++	komm.	X	
WWSAPI	C/C++	komm.		
csoap	C	GPL		
Qt SOAP	C++	LGPL + komm.		
Metro (JAX-WS RI)	Java SE	GPL 2 + CDDL		
Axis2	Java SE	Apache 2.0		
WS4D-Java-JMEDS	Java ME	EPL	X	X
WS4D-Java-Axis2	Java SE	Apache 2.0	X	
SOA4D-DPWS4J	Java ME	LGPL	X	
Apache CXF	Java SE	Apache 2.0		
ServiceMix-http	Java SE	Apache 2.0		
SAM	Java SE	Apache 2.0		
kSOAP	Java ME	N/A		
mSOAP	Java ME	GPL 2		
.NET WCF	C#	komm.		
.NET Micro Framework	C#	Apache 2.0		
WSDAPI for .NET	C#	komm.		
Levitate	C#	komm.	?	?
NuSOAP	PHP	LGPL		
PHP native	PHP	PHP 3.01		
WSO2 WSF/PHP	PHP	Apache 2.0		
SOAP::Lite	Perl	GPL + Artistic		
SOAP::WSDL	Perl	GPL + Artistic		
XML::Compile::SOAP	Perl	GPL + Artistic		
dXML	ObjC/iOS	BSD		
eSOL	N/A	N/A		
PeerlessNet WS SDK	N/A	N/A		
Life|ware	N/A	N/A		
Eurotech	N/A	N/A		
WSO2 WSF	*several*	Apache 2.0		

Tabelle 3.1.: Liste verschiedener Web-Service-Frameworks, nach Programmiersprachen gruppiert. Die populären Sprachen C/C++ und Java sind in den ersten Zeilen aufgeführt. Ein *X* markiert die jeweilige Unterstützung des DPWS 1.0 oder 1.1. Ein Fragezeichen signalisiert die Unterstützung für das DPWS ohne Kenntnis der Version. Ein leeres Feld bedeutet, dass kein DPWS in der jeweiligen Version unterstützt wird. Einige Produkte sind unter mehreren Lizenzen verfügbar, was durch ein *Plus* gekennzeichnet ist. Die Abkürzung *komm.* steht für eine kommerzielle Lizenz.

Auffällig ist die Häufigkeit C- und Java-basierter Umsetzungen in Vergleich zu den restlichen Programmiersprachen. Die Dominanz von Java und C erklärt sich möglicherweise durch deren allgemeine Popularität [133]. Das Schlusslicht bildet die hauptsächlich von Apple eingesetzte Sprache *Objective-C*, obwohl diese den dritten Platz des Programmiersprachenindex besetzt. Vermutlich wurde die Entwicklung eines DPWS-konformen Frameworks nicht vorangetrieben, da es ursprünglich vom Konkurrenten Microsoft entwickelt worden ist. Da Apple für das Netzwerk-Discovery eine eigene Variante des Zeroconf-Protokolls einsetzt, ist kein Bedarf für die Funktionalität des DPWS vorhanden.

Bei der Betrachtung DPWS-konformer Middleware-Lösungen spielen zwei Projektgruppen eine besondere Rolle. Die erste trägt den Namen *Web Services for Devices (WS4D)* und ist eine Ansammlung verschiedener Werkzeuge in den Programmiersprachen C/C++ und Java zum Einsatz auf ressourcen-beschränkten Geräten.

Die Java-Version unterteilt sich in den *Java Multi Edition Stack (JMEDS)* und WS4D-Axis2. JMEDS wird von der Universität Dortmund und der Firma Materna Information & Communication publiziert. Es basiert auf der Java *Connected Limited Device Configuration (CLDC)* und bringt diverse eigene Bibliotheken z. B. zur Realisierung von HashMaps und HTTP-Servern mit. WS4D-Axis2 liefert Plugins für die Apache-Axis2-SOAP-Engine zur Realisierung von DPWS-Diensten. Dieses WS4D-Werkzeug ist weniger für den Einsatz auf ressourcen-beschränkten Geräten, sondern vielmehr zur Anbindung an Applikations-Server gedacht. WS4D-gSOAP ist die C-basierte Erweiterung des gSOAP-Frameworks [28] zur DPWS-konformen Anbindung von Web-Services und wird im Wesentlichen von der Universität Rostock entwickelt. Die Plugin-Architektur von gSOAP erlaubt die transparente Einbindung von WS-Discovery, WS-Eventing, WS-Addressing usw. Dienste werden anschließend ähnlich erstellt wie über die native gSOAP-API. WS4D-gSOAP ist zwar GPL/LGPL-lizensiert, jedoch werden zur vollen Nutzung des gSOAP-Frameworks Lizenzgebühren erhoben.

Das zweite Projekt läuft unter dem Namen *Service-oriented Architectures for Devices (SOA4D)*. Es bietet neben einem Java-Framework (DPWS4J) ebenfalls eine auf gSOAP-basierende C-Variante (DPWS Core). DPWS4J kann zudem als OSGi-Bundle in eine OSGi-Umgebung eingebettet werden. Das Projekt wird von Schneider Electronics vorangetrieben und zielt ebenso auf die Vernetzung eingebetteter Systeme ab.

Es gibt zahlreiche weitere Middleware-Produkte wie die *Service Abstract Machine*, unter der jedoch seit 2009 keine Entwicklungsaktivitäten mehr zu verzeichnen sind. Proprietäre Lösungen kommen z. B. von *Levitate Technologies*, *Life/ware*, *Eurotech* und *PeerlessNet Web Services*. Die telefonische und elektronisch postalische Kontaktaufnahme zu diesen Firmen scheiterte jedoch, weshalb keine detaillierten Informationen zu den Produkten verfügbar sind. DPWS ist ebenfalls in Microsoft-Produkten wie dem *Web Services on Devices Application Programming Interface (WSDAPI)* verankert.

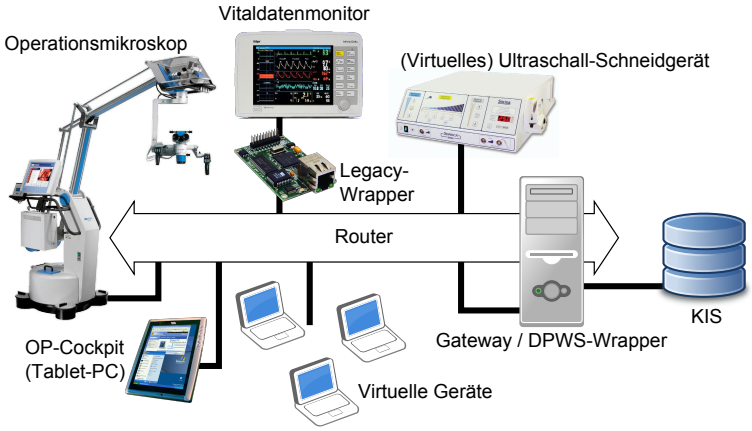
Da der Quelltext jedoch nicht verfügbar ist, können keine Erweiterungen vorgenommen werden. Es eignet sich demnach nicht mehr, sofern Erweiterungen im Kern der Middleware erforderlich sind.

3.3.3. Aufbau und Implementierung

Für die prototypische Umsetzung der Anwendungsszenarien werden der WS4D-gSOAP- und WS4D-JMEDS-Stack eingesetzt. Für die Identifizierung bekommt jedes Gerät einen dedizierten WS-Discovery-*Type*. Durch den Type werden Zusicherungen an die Operationen und Datentypen der WSDL-Schnittstelle gemacht, so dass ein Dienstkonsument entscheiden kann, ob die gewünschte Funktionalität in einem Gerät verfügbar ist. Die reine Nutzung von Types skaliert nicht für beliebige Schnittstellenzusicherungen, sondern sollte als grobe Geräte-Identifikation genutzt werden, beispielsweise so, dass der Type `http://med.devices.org/OperatingMicroscop/HNO` für „Dies ist ein HNO-Operationsmikroskop“ steht. Types, welche z. B. alle einzelnen Messwerte eines Monitoring-Gerätes repräsentieren, skalieren nicht, da sie während des Discovery-Prozesses aufwändig gefiltert werden müssen. Darüber hinaus werden Types als Teil von UDP-Multicast-Nachrichten versandt. Große Transportpakete können hier zur Überschreitung der *Maximum Transmission Unit (MTU)* führen, was folglich einem Paketverlust gleich käme. Für Anwendungsfälle wie der Charakterisierung einzelner Messwerte müssen weitere Schichten oberhalb der DPWS-Middleware geschaffen werden, um eine entsprechende Dynamik abzubilden.

Abbildung 3.5 zeigt den Versuchsaufbau zur Demonstration verschiedener Anwendungsfälle. Alle verfügbaren Geräte sind zur Nutzung von WS-Discovery in ein Subnetzwerk integriert. Da der Ultraschalldissektor nicht über physikalischen Schnittstellen verfügt, kommt stellvertretend ein *Graphical User Interface (GUI)* auf einem Laptop zum Einsatz. Neben dem Operationsmikroskop, dem Patientenmonitor und dem Schneidinstrument ist der Versuchsaufbau mit einem Beamer stellvertretend für eine OP-Leuchte, einem Dummy-KIS-Gateway sowie einer Dummy-Navigationsplattform ausgerüstet. Ein Samsung-Galaxy-Tablet-PC dient als OP-Cockpit.

Alle drei Demonstrator-Medizingeräte sind jeweils mit einer Strategie aus Abbildung 3.4 umgesetzt. Das Monitoring-Gerät ist mit Hilfe des Legacy-Wrapper-Pattern (vgl. Abbildung 3.4a) und das Operationsmikroskop sowie das Schneidinstrument mit dem Verfahren der direkten Software-Manipulation (vgl. Abbildung 3.4b) programmiert. Für das Legacy-Wrapper-Pattern kommt ein handelsüblicher Laptop mit WS4D-JMEDS zum Einsatz. Testweise wurde auch WS4D-gSOAP auf einer Netburner-Platine [86] installiert und getestet. Die Software übersetzt das Dräger-Infinity-Protokoll in ein offenes, auf XML- und ISO/IEEE-11073-basiertes Protokoll, mit denen einzelne Messwerte und Alarme von interessierten Geräten abonniert werden können.



(a) Logischer Geräteverbund

(b) Physikalischer Versuchsaufbau

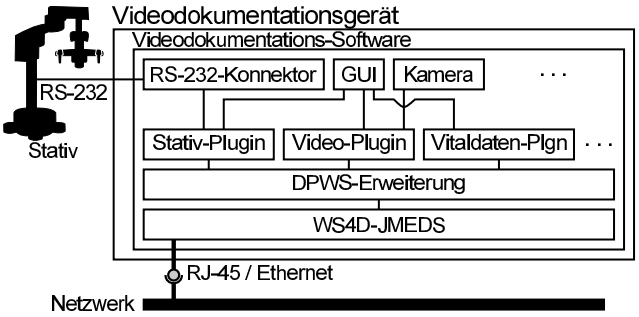
Abbildung 3.5.: Abbildung (a) skizziert den logischen Versuchsaufbau zur Umsetzung der in Abschnitt 3.2 beschriebenen Anwendungsfälle. Abbildung (b) zeigt den physikalischen Aufbau des Systems. Als virtuelle Geräte kommen eine Navigationsplattform, das elektronische Schneidinstrument und eine durch einen Beamer symbolisierte OP-Leuchte zum Einsatz.

Das Operationsmikroskop stellt im Versuchsaufbau neben dem OP-Cockpit eine zentrale Anzeige- und Steuereinheit dar. Somit lassen sich beispielsweise synchronisierte Screenshots durch den Screenshot-Knopf am Operationsmikroskop steuern oder Vitalparameter einblenden. Gleiches gilt für das OP-Cockpit. Weitere Geräte wie der virtuelle Ultraschalldissektor, die Monitoring-Einheit, das OP-Licht sowie die virtuelle Navigationsplattform stellen ausschließlich Dienste zur Verfügung. Abbildung 3.6a illustriert die Software-Architektur für das Operationsmikroskop. Die Web-Service-Schnittstelle wird als autonomer Thread in die Videodokumentation eingebettet und nutzt deren GUI zur Visualisierung kompatibler Geräte im Netzwerk (vgl. Abbildung 3.6b, weiß umrandetes Feld). Weiterhin wird das GUI um Schaltflächen für mögliche Netzwerkinteraktionen ergänzt. Damit lassen sich beispielsweise Alarme und Vitalparameter anzeigen (vgl. Abbildung 3.6c, weiß umrandete Felder). Die Übersetzung der Stativkommunikation in Web-Service-Operationen erfolgt anhand einer in der Videodokumentation verankerten Bibliothek zur Abfrage des seriellen RS-232-Ports. Ein Plugin-Mechanismus erlaubt die dynamische Erweiterung beliebiger Funktionalität zur Inanspruchnahme entfernter Dienste.

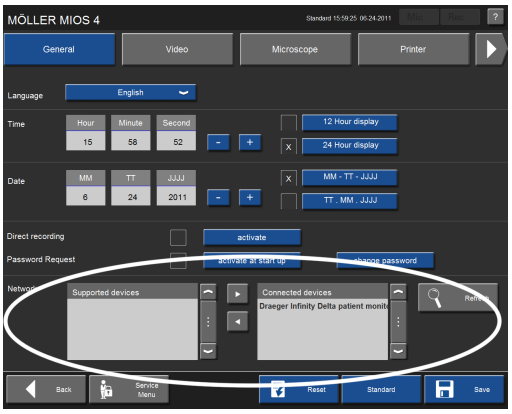
3.4. Evaluation

Der Web-Service-Technologie wird aufgrund der XML-Serialisierung eine schlechte Performance nachgesagt. Dies könnte sich auch bei der Vernetzung von Medizingeräten als Nachteil herausstellen. Zunächst wird deshalb hinterfragt, ob Web-Services tatsächlich langsamer sind als vergleichbare Middleware-Technologien und ob sie sich generell für die Vernetzung von Medizingeräten eignen. In [115] wurde bereits nachgewiesen, dass Web-Services ausreichend effizient sind und sich mit anderen Middleware-Technologien messen können. Hierfür sind zwei wesentliche Aspekte des Nachrichtenaustausches zwischen Medizingeräten untersucht worden, wie sie vom MD PnP [81] unterschieden werden: Datenverteilung und Fernsteuerung. Datenverteilung beschreibt die Verteilung klinischer Daten an mehrere Empfänger. Hierbei können Nachrichten sowohl periodisch als auch episodisch zugestellt werden. Ein Beispiel für periodische Datenverteilung ist die Übertragung von Vitalparametern wie der Herzfrequenz, wohingegen die Patientenstammakte nur episodisch verteilt werden muss. Den zweiten Aspekt des Nachrichtenaustausches bildet die Gerätefernsteuerung. Dabei verbinden sich in der Regel zwei Geräte Punkt-zu-Punkt und steuern sich in uni- oder bilateraler Richtung. Der wesentliche Unterschied besteht darin, dass im ersten Fall lediglich Daten gelesen und im zweiten Fall auch Daten geschrieben werden. Gerätefernsteuerung erfolgt beispielsweise autonom zwischen zwei Maschinen wie im Anwendungsfall von Arney et al. [7] zur Synchronisierung eines Beatmungsgerätes mit einem Röntgengerät; oder initiiert durch einen menschlichen Akteur wie im Anwendungsfall des OP-Cockpits.

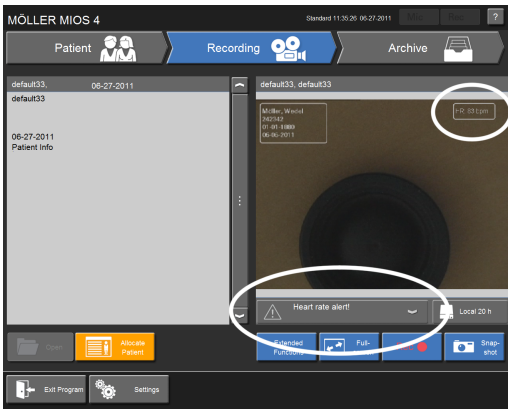
Im Vergleich zum Publish/Subscribe-System *Data Distribution Service (DDS)* [88] (vgl. Abschnitt 4.1 auf Seite 62) schneiden unverschlüsselte Web-Services im Rahmen



(a) Architektur der DPWS-Integration



(b) GUI-Screenshot 1



(c) GUI-Screenshot 2

Abbildung 3.6.: In (a) ist die Architektur für die Geräte-Software des Videodokumentationssystems der Firma Möller-Wedel skizziert. Abbildung (b) und (c) zeigen die Bedienoberfläche des Videodokumentationssystems.

der Fernsteuerung ähnlich gut ab. Ein Vergleich mit verschlüsselten Web-Services wurde nicht durchgeführt, da das eingesetzte DDS-Framework zum Zeitpunkt der Auswertung über keine entsprechende Funktionalität verfügte. Die Datenverteilung mit Web-Services ist zum Teil langsamer als mit DDS. Dies ist vor allem auf die XML-Serialisierung zurückzuführen, welche im Vergleich zur binären Übertragung von Zahlen aufwendiger ist. Web-Services sind jedoch für typische medizinische Anwendungen ausreichend performant. Das bestätigt neben der exemplarischen Messung in [115] die Durchführung der in Abschnitt 3.2 beschriebenen Szenarien anhand des Versuchsaufbaus aus Abbildung 3.5 auf Seite 56. Insbesondere in den durch menschliche Akteure initiierten Interaktionen stellt nicht die Systemantwortzeit, welche in den exemplarischen Messungen nicht mehr als zehn Millisekunden benötigt, sondern die Reaktionszeit der menschlichen Sinnesorgane eine Mindestanforderung an die Effizienz des Systems dar. Der Benutzer hat trotz der Latenz von zehn Millisekunden das Gefühl der Unmittelbarkeit bei der Benutzung der vernetzten Funktionalität.

Die Gestaltung passender Schnittstellenbeschreibungen in Form von WSDL-Dokumenten ist bei speziell zugeschnittenen Lösungen, in denen die Anforderungen aller Hersteller von Medizinprodukten bekannt sind, unproblematisch. Eine erste Herausforderung ist deshalb die Ausgestaltung standardisierter Schnittstellenbeschreibungen, die Kompatibilität über verschiedene Hersteller hinweg garantiert. Technische Probleme entstehen vorwiegend, wenn sich eine Schnittstelle dynamisch zur Laufzeit ändert. Beispielsweise ist es denkbar, dass die Änderung der Sensorik an einem Medizingerät die verfügbare Schnittstelle beeinflusst. Es wird deshalb notwendig sein, für produktive Endsysteme feingranulare Anforderungsanalysen zu skizzieren, um daraus umfassende Schnittstellenbeschreibungen abzuleiten, welche die entsprechende Dynamik abbilden können.

3.5. Ergebnis

Die Analyse zahlreicher Anwendungsfälle sowohl aus vertikaler als auch horizontaler Integrationssicht zeigt, dass der Bedarf für die Interoperabilität zwischen Medizingerätesystemen und angrenzenden IT-Landschaften eben ist und diese sowohl die Patientensicherheit erhöhen als auch Kosten einsparen kann. Bis heute haben sich jedoch keine einheitlichen Integrationskonzepte im medizinischen Bereich durchgesetzt, weswegen verschiedene Standards koexistieren. Eine mögliche Herangehensweise zur Steigerung der Interoperabilität kann dadurch erreicht werden, dass aktuelle Standards aus der Medizintechnik untereinander und mit modernen Kommunikationstechnologien wie dem DPWS zu integrieren. Damit ließe sich langfristig eine offene und von einem breiten Spektrum an Software-Ingenieuren handhabbare Alternative zu den existierenden Lösungen etablieren. Neben der Patientensicherheit kann hiermit auch die Herausbildung von IT-Experten in Unternehmen und die damit einhergehende Abhängigkeit an einzelne Ingenieure verhindert werden.

Kapitel 4.

Multicast-Bindung für WS-Eventing

Es gibt hinsichtlich der Medizingerätevernetzung weitere Anforderungen an eine Integrations-Middleware. Diese umfassen neben anwendungsspezifischen Protokollen auch Protokolle, die in der DPWS-Spezifikation nicht enthalten sind. Dieses Kapitel behandelt einen Ansatz zur Datenübertragung mittels UDP-Multicast zur Realisierung effizienter, stream-basierter Publish-Subscribe-Web-Services. Das Protokoll ist so gestaltet, dass DPWS-Frameworks mit wenig Aufwand nachgerüstet werden können. Die Erweiterung ist mit der DPWS-Spezifikation kompatibel. Die Inhalte aus diesem Kapitel wurden vorab in [36] veröffentlicht.

Web-Services basieren meist auf dem *Client/Server*- bzw. *Request/Response*-Kommunikationsmuster. Ein Dienstnutzer (Client) stellt eine Dienstanfrage bei einem Dienstanbieter (Server) und erhält daraufhin eine Antwort. Die Antwort muss hierbei nicht zwangsweise mit der Anfrage synchronisiert sein, sondern kann je nach eingesetztem Transport-Protokoll asynchron erfolgen. Maßgebend für das Request-Response-Muster ist das bidirektionale Kommunikationsverhältnis zwischen Client und Server. Ein weiteres, sehr verbreitetes *Message Exchange Pattern (MEP)* wird als *Publish-Subscribe* bezeichnet. Hierbei abonnieren mehrere Client-Instanzen einen beliebig gefilterten Nachrichtendienst eines Servers. So lange mindestens ein Abonnement besteht, werden anfallende Daten ereignisgetrieben an die Interessenten verteilt. Es besteht demnach eine 1-zu-n-Kommunikationsbeziehung. Die Client-Instanzen in Publish-Subscribe-Systemen werden auch als Consumer oder Ereignissenken und die Server-Instanzen als Producer, Ereignisquelle oder Publisher bezeichnet.

Die Effizienz von Publish-Subscribe-Systemen ergibt sich aus dem verwendeten Ereignisfilter als auch dem zugrundeliegenden Einsatzbereich. Der Einsatzbereich bestimmt hierbei im Wesentlichen die Skalierbarkeit des Systems [14, S. 3]. Multicast-getriebene Datenverteilung ist beispielsweise in LANs einfacher als in *Wide Area Networks (WANs)*, da das Netzwerk weniger eingeschränkt ist. Je nach Infrastruktur und Anforderung kann ein Publish-Subscribe-System deshalb unterschiedlich effizient umgesetzt sein.

In Abschnitt 3.4 auf Seite 57 werden zwei verschiedene Kommunikationsmuster bei der Medizingerätevernetzung vorgestellt: Fernsteuerung und Datenverteilung. Bei-

de Muster decken sich mit den korrespondierenden MEPs Request/Response und Publish-Subscribe. Die Datenverteilung kann darüber hinaus in zwei weitere Aspekte unterteilt werden: ereignis-basierte und stream-basierte Datenverteilung. Bei der ereignis-basierten Verteilung wird ein abgeschlossener Datensatz an alle Ereignissenken transferiert, sobald er an der Ereignisquelle vorliegt. Dies kann entweder in episodischen (also sporadischen) oder periodischen Zeitabständen geschehen. Ein klassischer Anwendungsfall für die ereignis-getriebene Datenverteilung ist die Übertragung der Patientenstammakte auf Medizingeräte (vgl. auch Kapitel 5 auf Seite 79). Für die stream-basierte Verteilung gilt, dass Datenpakete typischerweise in periodischen Zeitabständen verschickt werden, um eine kontinuierliche Datenzustellung zu gewährleisten. Im Operationssaal fallen solche Daten vorwiegend am Anästhesiearbeitsplatz in Form von kontinuierlich verfügbaren Vitalparametern an. Zum Teil wird auch Videomaterial erzeugt, für dessen Transport allerdings etablierte Standards existieren [58, 59, 60]. Eine Neuentwicklung für Videosignale ist aus Sicht des Autors deshalb nicht erforderlich. Der wesentliche Unterschied zwischen der ereignis- und stream-basierten Datenverteilung liegt in der Anforderung an die Zuverlässigkeit des Transports. Während es beispielsweise in Bezug auf die Medizingerätekommunikation nicht vertretbar ist, dass Teile der Patientenstammakte verfälscht werden oder verloren gehen, können korrumpierte Herzfrequenzkurven durch Interpolation ggf. vollständig oder ausreichend gut rekonstruiert werden. Ein gelegentlicher Sequenzausfall ist somit tolerierbar.

Im Allgemeinen sind Web-Services nicht für die Übertragung von stream-basierten Medien geeignet. Im medizinischen Bereich kann es jedoch vorkommen, dass von Körpersensoren gemessene Signalverläufe in (weicher) Echtzeit ausgetauscht und visualisiert werden müssen (vgl. den OP-Cockpit-Anwendungsfall in Kapitel 3 auf Seite 49). Mit DICOM sind hier bereits Terminologien und Codierungen definiert worden [82]. Ergänzend zum DICOM-Standard wird im Folgenden eine UDP-Multicast-Transport-Bindung für Publish-Subscribe-Web-Services definiert, welche die Signalverläufe in Subnetzwerken effizient an mehrere Empfänger verteilen kann. Eine UDP-Multicast-Transport-Bindung ist für Publish-Subscribe-Web-Services, insbesondere mit WS-Eventing, bisher nicht vorgesehen. Die Transport-Bindung kann in DPWS integriert werden und erlaubt die Übertragung von Streaming-Informationen, ohne auf Transportprotokolle anderer Standards wie z. B. DICOM angewiesen zu sein.

4.1. Verwandte Arbeiten

Für die Konstruktion einer UDP-Multicast-Bindung für Publish-Subscribe-Systeme werden zunächst verwandte Arbeiten und Systeme untersucht. Es gibt zahlreiche Konzepte sowie Implementierungen, die unter freier Lizenz stehen oder kommerziell vermarktet werden. Drei der bekanntesten Vertreter sind Active MQ [3], WebSphere MQ [51] und DDS [88]. Active MQ ist eine von der Apache Foundation veröffentlichte

Open-Source-Implementierung des JMS. Neben JMS, welches bereits als vollwertiges Publish-Subscribe-System zu verstehen ist, unterstützt Active MQ außerdem WS-Notification [97]. Eine Bindung für das in der DPWS-Spezifikation enthaltene WS-Eventing ist nicht existent. WebSphere MQ ist ein Produkt der Firma IBM, das populäre Spezifikationen wie JMS und SOAP unterstützt. Ob eine Bindung für UDP-Multicast vorhanden ist, geht aus den verfügbaren Unterlagen zum Produkt nicht hervor. Als Drittes in der Reihe existiert die von der *Object Management Group (OMG)* standardisierte DDS-Spezifikation. DDS ist eine Middleware zur Kommunikation von Echtzeitapplikationen. Der elementare Bestandteil des DDS ist durch QoS charakterisiert. Die durch die Spezifikation umgesetzte Middleware garantiert dabei keine Echtzeit, sondern meldet lediglich die Verletzung von QoS-Regeln. DDS ist für Netzwerke mit tausenden Knoten konzipiert und entkoppelt die Sender und Empfänger voneinander, um die nötige Skalierbarkeit und Flexibilität zu garantieren. Dies erreicht es mit dem sogenannten *Data-Centric-Publish-Subscribe (DCPS)*. Jede Applikation verbreitet dabei ihre Informationen über einen Publisher in einem globalen Datenraum. Zum Lesen dieses Datenraums bedarf es eines Subscribers. Die Middleware übernimmt hierbei das Verteilen der Daten. Anhand benutzerdefinierter Metainformationen kann DDS an nahezu jedes Transportprotokoll gebunden werden. Eine native Unterstützung für die Web-Service-Technologie ist nicht gegeben, jedoch durch eine Erweiterung herstellbar. Aus der DDS-Spezifikation geht nicht hervor, ob WS-Eventing-basierte Transport-Bindungen mit UDP möglich sind.

Carzaniga stellt in seiner Dissertation [14] grundsätzliche Publish-Subscribe-Konzepte sowie theoretische Ansätze für inhaltsbasierte Filtertechniken vor. Wesentlicher Teil seiner Arbeit ist die Umsetzung des Publish-Subscribe-Systems SIENA mit dem Fokus auf Skalierbarkeit beim Einsatz in WANs. Der Multicast-Verteilung kommt darin nur eine geringe Bedeutung zu. SIENA bedient sich neben den klassischen Dienstprimitiven wie *subscribe* und *unsubscribe* einer neuen Methodik: *advertise*. Durch advertise können im Gegensatz zu einzelnen Benachrichtigungen ganze Benachrichtigungssequenzen beobachtet werden. Bei der Umsetzung von SIENA nutzt Carzaniga verschiedene Strategien für die Verteilung von Benachrichtigungen. Durch die Nutzung von *Ereignis-Servern* können flexible Topologien (hierarchisch, Peer-to-Peer oder hybrid) konstruiert werden, um Nachrichten zwischen Ereignisquellen und -senken auszutauschen. Der letzte Teil seiner Arbeit beschreibt eine Simulationsumgebung zur quantitativen Bewertung des SIENA-Frameworks.

Einen alternativen Weg zur Herstellung von Publish-Subscribe schlagen Skjervold et al. vor [127]. Sie beschreiben ein auf einem Proxy-Mechanismus basierendes System, um Commercial-off-the-shelf-Produkte in einem Publish-Subscribe-Verbund zu integrieren. Hierfür werden gewöhnliche Request-Response-Web-Services mit Hilfe eines Overlay-Netzwerkes umgeformt. Die als DSProxy bezeichnete Middleware läuft auf derselben Maschine wie der Request-Response-Web-Service und führt in periodischen Abständen einen Poll durch, um ankommende Ereignisse zu beobachten. Sobald ein Ereignis detektiert wurde, wird es mit einem proprietären Protokoll an entfernte DSProxy-

Instanzen weitergeleitet. Die DSProxy-Methodik ist ein geeignetes Mittel, um robustes Publish-Subscribe in existierende Web-Service-Infrastrukturen zu integrieren, ohne die Web-Service-Applikationen als solches zu modifizieren. Allerdings sind keine Multicast-Bindungen vorgesehen, da der Schwerpunkt vielmehr bei der Integration der Off-the-shelf-Produkte liegt. Darüber hinaus ist im Allgemeinen vom Einsatz proprietärer Protokolle abzuraten. Die Erweiterung bestehender, offener Protokolle sollte grundsätzlich mit offen zugänglichen Protokollen erfolgen.

4.2. Web-Service-relevante Publish-Subscribe-Systeme

Im Kontext der Web-Service-Technologie koexistieren drei Kandidaten, welche für die Konstruktion von Publish-Subscribe-Systemen vorgesehen sind: *WS-Events* [16], *WS-Notification* [97] und *WS-Eventing* [167]. WS-Events ist die älteste der drei Spezifikationen und wurde 2003 von HP veröffentlicht. Heute spielt WS-Events praktisch keine Rolle mehr in der Web-Service-Welt. WS-Notification ist die anspruchsvollste der drei Spezifikationen und wurde zunächst von IBM und Globus Alliance in 2004 verabschiedet. Die Spezifikation ist derzeit als OASIS-Standard verfügbar und setzt sich aus den drei Teilen *WS-BaseNotificaction*, *WS-BrokeredNotification* und *WS-Topics* zusammen.

WS-BaseNotification stellt hierbei die Basis des Standards dar. Enthalten sind grundlegende Publish-Subscribe-Elemente wie die Herstellung und Auflösung von Abonnements, die Angabe von Filtern über XPath [162], die Verwaltung von Abonnements sowie die Definition der Entitäten *Producer* und *Consumer*. WS-BrokeredNotification ist die erste Erweiterung des WS-BaseNotification-Standards um eine Vermittler-Instanz, welche die Ereignisquelle und -senke voneinander entkoppelt. Dadurch kann die Anzahl von Verbindungen zwischen Quellen und Senken reduziert und die Identität der Ereignisquellen verborgen werden. WS-Topics beschreibt ein Protokoll zur Definition von themenbasierten Filtertechniken, die neben der standardmäßigen Variante zu XPath eingesetzt werden können. Dabei kann eine Ereignissenke Topics abonnieren, die Synonyme und Hierarchien erlauben. Hierarchien zu abonnieren bedeutet in diesem Zusammenhang, dass das Abonnement eines Topics in einer Hierarchie ebenfalls das Abonnement aller als Kindelemente enthaltenen Topics umfasst.

Die im Rahmen dieses Kapitels betrachtete Web-Service-Spezifikation ist WS-Eventing, da sie im DPWS-Standard referenziert wird (vgl. Abschnitt 2.2 auf Seite 25). WS-Eventing ist im Wesentlichen eine kompakte Ausführung von WS-Notification und entspricht im Umfang etwa dem WS-BaseNotification-Standard. WS-Eventing wurde erstmals im Januar 2004 von Microsoft herausgegeben und anschließend zusätzlich von IBM, Sun und CA Technologies vorangetrieben. Seit 2009 wird es im Rahmen der *Web Services Resource Access Working Group* [143] weiterentwickelt und erhielt 2011 den Status einer W3C-Empfehlung [167]. Ein Vergleich zwischen WS-Notification

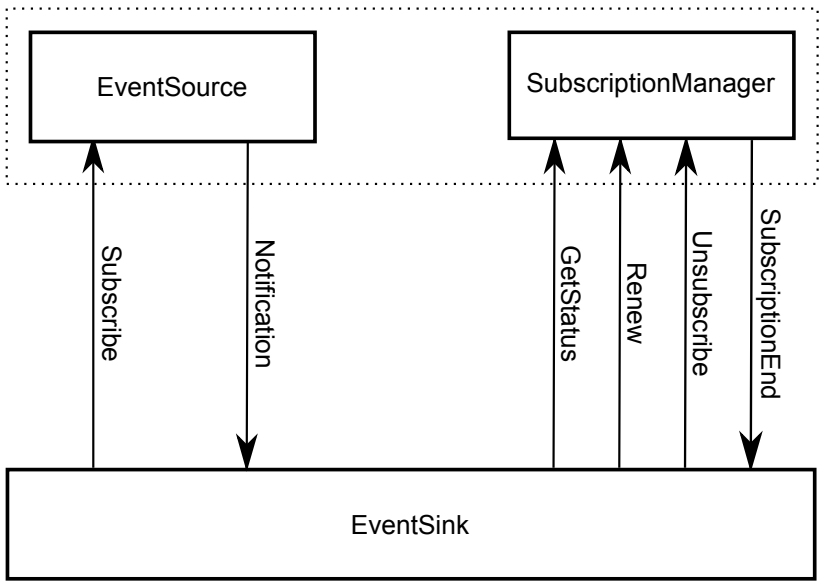


Abbildung 4.1.: Entitäten und Operationen aus der WS-Eventing-Spezifikation.

und WS-Eventing findet sich in [43, S. 14ff]. Seit 2006 gibt es außerdem das Bestreben verschiedener Firmen wie Intel, IBM und Microsoft, WS-Notification und WS-Eventing zu vereinheitlichen [17]. Der daraus resultierende Standard heißt WS-EventNotification. Er subsumiert WS-Eventing und erweitert es mit zahlreichen Funktionen aus WS-Notification. Zum Zeitpunkt der Veröffentlichung dieser Arbeit gab es jedoch keine weiteren Aktivitäten mehr. Es ist deshalb davon auszugehen, dass WS-EventNotification nicht mehr weiterentwickelt wird.

In Abbildung 4.1 sind die Entitäten und Operationen aus der WS-Eventing-Spezifikation abgebildet. Der Lebenszyklus eines Abonnements gliedert sich in drei Teile:

1. Subscribe-Phase: durch den Aufruf der Operation *Subscribe* abonniert eine Ereignissenke (*EventSink*) eine Ereignisquelle (*EventSource*). Anschließend bekommt die Ereignissenke einen *SubscriptionManager* zugewiesen. Diese Verwaltungsinstanz kann von der Senke dazu verwendet werden, das Abonnement abzufragen (*GetStatus*), zu erneuern (*Renew*) und zu beenden (*Unsubscribe*). Erwähnenswert ist in diesem Zusammenhang, dass die Verwaltungsinstanz auf andere Web-Service-Endpunkte ausgelagert werden kann (Abbildung 4.1 kennzeichnet diese lose Kopplung durch gestrichelte Linien). Damit kann die Verwaltung beispielsweise von einem anderen Rechner übernommen werden. Es sind jedoch keine Protokolle in WS-Eventing vorgesehen, die ein solches Verhalten zwischen EventSource und SubscriptionManager abbilden.

2. Subscription-Phase:

   a) Sobald an der Quelle ein Ereignis vorliegt, wird die Senke anhand einer *Notification*-Nachricht informiert. Jede Benachrichtigung enthält im Wesentlichen die Endpunkt-Referenz der Ereignisquelle als auch die gefilterten Nutzdaten.

   b) Für ein Abonnement kann eine Gültigkeitsdauer beim Subscribe-Prozess ausgehandelt werden. Enthält die Subscribe-Nachricht keinen Wert, wählt die Ereignisquelle einen aus und teilt sie der Senke mit. Durch eine *Renew*-Anfrage steht der Ereignissenke anschließend eine Operation zur Verlängerung der Gültigkeitsdauer vor Ablauf eines Abonnements zur Verfügung.

   c) Anhand einer GetStatus-Anfrage kann eine Ereignisquelle den Status eines Abonnements abfragen. Der Status setzt sich standardmäßig aus der Gültigkeitsdauer zusammen, ist jedoch dank eines any-Elements im XML-Schema der GetStatus-Nachrichtendefinition beliebig erweiterbar.

3. Unsubscribe-Phase:

   a) Ist die Ereignissenke nicht weiter an Informationen von der Ereignisquelle interessiert, kann mit Hilfe einer Unsubscribe-Nachricht das Abonnement beendet werden.

   b) Für den Fall, dass ein Abonnement vorzeitig beendet werden muss, hält WS-Eventing zusätzlich die *SubscriptionEnd*-Anfrage vor. Damit ist die Ereignisquelle in der Lage, der Senke die Beendigung mitzuteilen.

An welchen Daten eine Ereignissenke interessiert ist, kann anhand eines Filter-Dialekts im Rahmen der Subscribe-Anfrage vorgegeben werden. In Publish-Subscribe-Systemen unterscheidet man typischerweise zwischen kanal-, themen-, und inhaltsbasierten Abonnements (engl. channel-based, subject-based und content-based) [14, S. 14f]. Die einfachste Variante ist das kanalbasierte Abonnement. Interessierte Parteien schreiben sich dabei auf einem dedizierten Kommunikationskanal ein. Die Ereignisquelle verteilt aufkommende Ereignisse auf die verfügbaren Kanäle. Ein Beispiel für die kanalbasierte Verteilung ist der CORBA-Event-Service [87]. Die themenbasierte Variante erweitert das kanalbasierte Verfahren um einen flexibleren Adressierungsmechanismus. Der Unterschied besteht darin, dass Abonnements auf ein oder mehrere Themen abgebildet werden können. Ein Abonnement definiert demnach eine Menge an Themen, nach denen die Benachrichtigungen gefiltert werden sollen. Vor dem Versand werden die Themen von der Ereignisquelle evaluiert. Ein Beispiel für die themenbasierte Filterung bietet die von Cugola et al. entwickelte *Java Event-based Distributed Infrastructure (JEDI)* [20]. Der ausdrucksstärkste Filtermechanismus ist durch das inhaltsbasierte Abonnement gegeben. Im Gegensatz zu themenbasierten Benachrichtigungen können komplexe Instruktionen auf die gesamte abzufragende Datenstruktur angewendet werden. Ein gewöhnlicher inhaltsbasierter Dialekt für Publish-Subscribe-Web-Services

Präfix	Spezifikation
`wse`	WS-Eventing [150, 167]
`s12`	SOAP 1.2 [152]
`wsa`	WS-Addressing [149]
`wsd`	WS-Discovery [106]
`wsevd`	WS-EventDescriptions [164]
`xsd`	XML-Schema [147]

Tabelle 4.1.: Liste verwendeter Namensräume für die SOAP-over-UDP-Multicast-Erweiterung.

ist *XPath*[1] (XPath-Adressierung), während DPWS beispielsweise die Unterstützung des eigens definierten *Action*-Filters[2] vorschreibt. Dieser Filter erlaubt das Abonnement von Nachrichten, die anhand einer WS-Addressing-Action identifiziert werden. Sie entsprechen damit in etwa dem themenbasierten Verfahren.

4.3. Multicast-Erweiterung

Die im Folgenden beschriebene Multicast-Erweiterung basiert auf den zwei Versionen der in [150, 167] definierten WS-Eventing-Spezifikation. WS-Eventing beschreibt eine pragmatische Lösung des Publish-Subscribe-Paradigmas. Protokolle für eine Multicast-Transport-Bindung sind jedoch nicht vorgesehen. Die OASIS hat in diesem Zusammenhang bereits SOAP-over-UDP standardisiert [101]. Der SOAP-over-UDP-Standard definiert UDP-basierte Web-Services, indem es die verschiedenen MEPs skizziert und ein URI-Gerüst vorschlägt. Bisher fehlt jedoch die Integration mit WS-Eventing zur Umsetzung Multicast-basierter Publish-Subscribe-Web-Services mit DPWS. Hierfür muss WS-Eventing um eine SOAP-over-UDP-Bindung ergänzt werden. Darüber hinaus ist zu spezifizieren, welche Definitionen des WS-Eventing-Standards durch SOAP-over-UDP beeinflusst werden. Für die Erweiterung wird im Folgenden ein XML-Namensraum definiert:

```
xmlns:sume = "http://www.w3.org/2011/03/ws-evt/sume"
```

Die Abkürzung `sume` steht für *SOAP-over-UDP-Multicast-Extension*. Der Namensraum ist bisher nicht standardisiert und wird im Rahmen dieser Arbeit als ein

[1] URI: http://www.w3.org/2011/03/ws-evt/Dialects/XPath10
[2] URI: http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/Action

```
1 [Action]
2   http://www.w3.org/2011/03/ws-evt/Subscribe
3 [Message Body]
4  <wse:Subscribe>
5   <wse:EndTo>
6    <!-- usual EndTo-EPR -->
7   </wse:EndTo>
8   <wse:Delivery>
9    <sume:UDPMulticast/>
10  </wse:Delivery>
11  <wse:Expires>PT10M</wse:Expires>
12  <wse:Filter Dialect="http://docs.oasis-open.org/ws-dd/ns/dpws
       /2009/01/Action">
13   http://www.example.org/EventedAction
14  </wse:Filter>
15 </wse:Subscribe>
```

Quelltext 4.1: Exemplarischer SOAP-Subscribe-Request mit Multicast-Erweiterung. Der Übersicht halber ist die SOAP-Nachricht auf die WS-Addressing-Action und die Nutzdaten beschränkt.

möglicher Kandidat vorgeschlagen, um die Erweiterungselemente für WS-Eventing zu beschreiben. Weitere Namensräume werden durch ein Präfix abgekürzt verwendet. Sie sind in Tabelle 4.1 aufgelistet. Für WS-Eventing kommen in Abhängigkeit der Version (Recommendation oder Member-Submission) zwei verschiedene Namensräume zum Einsatz. Da die Multicast-Erweiterung für beide Varianten gelten soll, wird der Einfachheit halber ausschließlich das Präfix `wse` ohne zusätzlicher Versionsunterscheidung verwendet. Im Folgenden sind alle für WS-Eventing notwendigen Anpassungen beschrieben. Das Protokoll ist so entworfen, dass die Nutzung mit WS-Eventing nicht beeinträchtigt wird. Das bedeutet insbesondere, dass eine konforme Ereignisquelle ohne Multicast-Unterstützung von einer Ereignissenke mit Multicast-Untertstützung zwar angefragt werden kann, jedoch mit einem entsprechenden SOAP-Fault abgefertigt wird. Damit steht der Senke anschließend frei, sich für eine native Subscription einzuschreiben. Die Multicast-Protokollspezifikation ist im Folgenden auf die in Abschnitt 4.2 beschriebenen drei Phasen des Publish-Subscribe-Lebenszyklus bezogen.

4.3.1. Modifikation der ersten Phase

Gemäß der WS-Eventing-Spezifikation ist die erste Phase des Subscribe-Prozesses durch eine Subscribe-Nachricht gefolgt von einer SubscribeResponse-Nachricht charakterisiert. Eine exemplarische Subscribe-Nachricht ist in Quelltext 4.1 abgebildet. Essentielle Bestandteile sind die Elemente `wse:Delivery`, `wse:Format`, `wse:Filter` und `wse:Expires`. Das `wse:Delivery`-Element enthält ein `wse:NotifyTo`-Element,

```
1 <wse:EventSource>
2   <wse:DateTimeSupported />
3   <wse:FilterDialect URI="http://www.w3.org/2011/03/ws-evt/
       Dialects/XPath10"/>
4   <wse:EndToSupported/>
5   <sume:UDPMulticastSupported/>
6 </wse:EventSource>
```

Quelltext 4.2: Exemplarische Multicast-Policy-Assertion.

welches typischerweise eine Endpunkt-Referenz der anfragenden Ereignissenke beinhaltet. Für die Verteilung von Multicast-Nachrichten ist an dieser Stelle keine Endpunkt-Referenz erforderlich, da Benachrichtigungen an eine von der Ereignisquelle determinierte Multicast-Adresse versandt werden. Diese Änderung ist gemäß WS-Eventing dank `xsd:any` zulässig. Anstelle von `wse:NotifyTo` signalisiert ein leeres `sume:UDPMulticast`-Element der Ereignisquelle, dass die Benachrichtigungen an eine Multicast-Adresse zu senden sind. Unterstützt die Quelle kein Multicast, muss sie mit einem *UnusableEPR*-Fault antworten, da keine für WS-Eventing konforme Endpunkt-Referenz gegeben ist. Um dieses Try-and-Error-Prinzip zu vermeiden, kann eine Senke durch das Auslesen einer WS-Eventing-Policy-Assertion bereits vorab erfragen, ob Multicast-Unterstützung vorhanden ist oder nicht (vgl. Quelltext 4.2). Zu diesem Zweck ist der Erweiterungspunkt der *EventSource*-Assertion um ein leeres `sume:UDPMulticastSupported`-Element zu ergänzen.

Ziel der in diesem Kapitel beschriebenen Erweiterung ist vornehmlich die Integration in den DPWS-Protokollstapel. Da DPWS 1.1 jedoch auf einer veralteten WS-Eventing-Member-Submission ohne Unterstützung einer EventSource-Assertion basiert, sollte zu Gunsten der Kompatibilität auf deren Nutzung verzichtet werden.

Ein weiterer Unterschied zwischen der Recommendation [167] und der Member-Submission [150] von WS-Eventing ist der Umgang mit dem sogenannten *Delivery-Mode* (in der Recommendation umdefiniert zum *Delivery-Format*). Für die Member-Submission existiert ein Attribut namens `wse:Mode`, das in das `wse:Delivery`-Element eingefügt wird. Die Recommendation beinhaltet ein separates Element im Subscribe-Request, bezeichnet als `wse:Format`. Der Delivery-Mode gibt der Ereignisquelle vor, in welchem Modus Benachrichtigungen zugestellt werden sollen. Die Submission definiert *Push*[3] als Standard-Wert. Dieser besagt, dass Nachrichten asynchron an die Senke zugestellt werden. Das in der Spezifikation nicht näher erläuterte Gegenstück könnte sich durch ein periodisches *Poll* charakterisieren. Für die Recommendation ist kein Push mehr vorgesehen und es werden ausschließlich das Format *Wrapped*[4] und *Unwrapped*[5]

[3] http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryModes/Push
[4] http://www.w3.org/2011/03/ws-evt/DeliveryFormats/Wrap
[5] http://www.w3.org/2011/03/ws-evt/DeliveryFormats/Unwrap

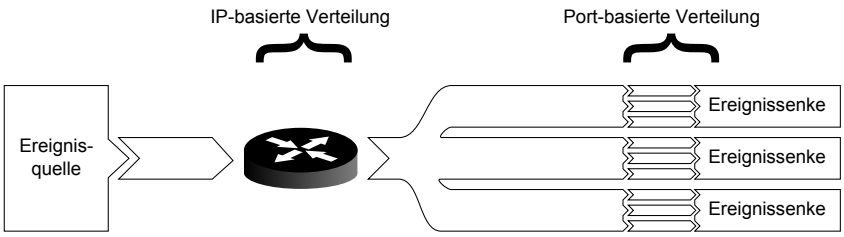


Abbildung 4.2.: Filtertechniken für kanalbasiertes Multicast-Eventing.

eingeführt. Beide Modi sind in einer ähnlichen Form bereits in der Submission genannt. Von Wrapped ist die Rede, wenn die Nutzdaten einer Benachrichtigung in einem gesonderten Element mit dem Namen `wse:Notify` im SOAP-Body hinterlegt sind. Unwrapped hingegen kommt ohne ein umschließendes Element aus.

Für die Multicast-Zustellung bedeutet die Definition verschiedener Delivery-Modes ein deutlicher Mehraufwand, da z. B. Wrapped- und Unwrapped-Benachrichtigungen nicht über die gleiche Adresse versandt werden sollten. Dies könnte dazu führen, dass einer Ereignissenke falsche bzw. nicht interpretierbare Informationen übermittelt werden. Jedem Delivery-Mode sollte deshalb gemäß dieser Spezifikation ein separater Multicast-Kanal gewidmet werden. Beispielsweise ließen sich alle Wrapped-Benachrichtigungen an 239.192.0.101:4001 und alle Unwrapped-Benachrichtigungen an 239.192.0.101:4002 transferieren. Dabei obliegt die Auswahl der Multicast-Endpunkte der Ereignisquelle. Für die Integration mit DPWS spielen Delivery-Modes keine Rolle, da üblicherweise nur der Push-Mode eingesetzt wird.

Eine weitere Fragestellung beim Design eines Multicast-basierten WS-Eventing ist der Umgang mit Filtern. Da die vorliegende Protokollerweiterung primär für DPWS spezifiziert ist, erübrigt sich ein Großteil der Komplexität. Laut der DPWS-Profilbeschreibung muss WS-Eventing lediglich den Action-Filterdialekt[6] unterstützen. Hierbei kann z.B. eine Verteilung der verschiedenen Actions auf dedizierte Multicast-Adressen oder Ports erfolgen, wie in Abbildung 4.2 dargestellt. Bei IP-basierter Verteilung wird die Filterung der Kanäle auf den Routern vorgenommen (weniger Bandbreitennutzung, hoher Multicast-IP-Verbrauch). Bei Port-basierter Verteilung filtert das Betriebssystem der Ereignissenke die Nachrichten (mehr Bandbreitennutzung, geringer Multicast-IP-Verbrauch). Ersteres Verfahren ist kritisch, wenn viele Ereignisquellen im Netzwerk agieren.

Vorschläge für eine effiziente inhaltsbasierte Verteilung von Benachrichtigungen kommen von Banavar et al. [8] und Cao et al. [13]. Diese Verfahren sind jedoch wesentlich komplizierter als das kanal- oder themenbasierte Verfahren. Sie zeigen, dass inhaltsbasierte Filter ebenfalls in Multicast-Umgebungen umgesetzt werden können, hierfür

[6] URI: http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/Action

```
1 [Action]
2   http://www.w3.org/2011/03/ws-evt/SubscribeResponse
3 [Message Body]
4  <wse:SubscribeResponse>
5    <wse:SubscriptionManager>
6      <wsa:Address>
7        http://wse.example.org/SubscriptionManager
8      </wsa:Address>
9    </wse:SubscriptionManager>
10   <wse:Expires>2013-07-01T00:00:00.000-00:00</wse:Expires>
11
12   <sume:NotificationDestination>
13     <wsa:Address>
14       soap.udp://239.192.0.101:4002
15     </wsa:Address>
16   </sume:NotificationDestination>
17 </wse:SubscribeResponse>
```

Quelltext 4.3: Exemplarische SubscribeResponse-Nachricht.

allerdings zusätzliche Infrastruktur außerhalb handelsüblicher Router-Hardware erforderlich ist.

Mit `wse:Expires` sind Ereignissenken in der Lage, ein Ablaufdatum für die Gültigkeit von Abonnements vorzuschlagen. Die Ereignisquelle entscheidet sich jedoch für einen Zeitraum, den sie tatsächlich bewilligt. Im Zusammenhang mit der Multicast-Funktionalität entsteht hier eine weitere Besonderheit. Da Ereignisse an eine Multicast-Adresse versandt werden, können Benachrichtigungen generell von jedem Netzwerkteilnehmer abgehört werden, sofern die Adresse bekannt ist. Eine Ereignisquelle kann auf den Versand von Benachrichtigungen jedoch verzichten, sollte kein Abonnement vorhanden sein. Deshalb gilt: Abonnements müssen weiterhin regulär durchgeführt und von der Ereignisquelle verwaltet werden, jedoch mit der Besonderheit, dass nur noch das zeitlich am längsten gültige Abonnement für die Entscheidung, ob Benachrichtigungen an eine Multicast-Adresse versandt werden, maßgebend ist.

In Quelltext 4.3 ist die Antwort auf die exemplarische Subscribe-Anfrage aus Quelltext 4.1 illustriert. Der gewöhnliche `wse:SubscribeResponse`-Body enthält neben der Angabe des SubscriptionManagers und dem Gültigkeitszeitraum ein weiteres Element mit dem Namen `sume:NotificationDestination`. Die `sume:NotificationDestination` wiederum enthält eine Referenz auf einen UDP-Multicast-Endpunkt. Dieser wird durch folgendes URI-Gerüst charakerisiert [101, Abschnitt 5.1]:

```
soap.udp://<host>:<port>[<rel_path>][?<query>]
```

```
1 <s12:Envelope
2   xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
3   xmlns:wsa="http://www.w3.org/2005/08/addressing"
4   xmlns:wsd="http://docs.oasis-open.org/ws-dd/ns/discovery
       /2009/01">
5   <s12:Header>
6     <wsa:Action>
7       http://www.example.org/EventedAction
8     </wsa:Action>
9     <wsa:MessageID>
10      uuid:82a74510-7fed-41b2-834f-7703825026fc
11    </wsa:MessageID>
12    <wsa:To>soap.udp://239.192.0.101:4002</wsa:To>
13    <wsd:AppSequence InstanceId="306104540" MessageNumber="2"/>
14  </s12:Header>
15  <s12:Body>
16    <!-- Nutzdaten -->
17  </s12:Body>
18 </s12:Envelope>
```

Quelltext 4.4: Exemplarische Benachrichtigung.

Ausschlaggebend für die Identifikation eines UDP-basierten Endpunkts ist der Bezeichner vor dem Doppelpunkt, der auch als Schema bekannt ist: `soap.udp`. Es folgt eine Multicast-IP-Adresse, die von der Ereignisquelle ausgewählt wird. Alle für eine Anfrage relevanten Ereignisse werden an diese Adresse versandt. Demzufolge muss eine Ereignissenke, sobald sie die Antwort von der Quelle erhalten hat, mittels dem *Internet Group Management Protocol (IGMP)* bzw. dem *Multicast Listener Discovery (MLD)* die entsprechende Multicast-Gruppe abonnieren. IGMP verwendet man typischerweise im Zusammenhang mit IPv4, während MLD bei IPv6-Multicast Anwendung findet. Nachdem der SubscriptionManager verfügbar und die Multicast-Adresse abonniert ist, kann die Ereignissenke auf Benachrichtigungen warten. Für SubscriptionEnd-Nachrichten an einzelne Ereignissenken kann das gewöhnliche WS-Eventing-Protokoll verwendet werden.

4.3.2. Modifikation der zweiten Phase

Die zweite Phase beschreibt die Zustellung von Benachrichtigungen. Quelltext 4.4 illustriert eine exemplarische Nachricht. Hauptbestandteile sind die im Header verankerten Elemente `wsa:Action`, `wsa:MessageID` und `wsa:To`. Für gewöhnlich sind auch WS-Addressing-Referenzparameter denkbar. Da Referenzparameter jedoch dynamisch sind, eignen sie sich nicht für die Multicast-Erweiterung. Im `wsa:Action`-Element ist stets ein charakterisierender, fester URI zu hinterlegen. Für DPWS entspricht dieser

```
1 <s12:Envelope
2   xmlns:s12="http://www.w3.org/2003/05/soapenvelope"
3   xmlns:wsa="http://www.w3.org/2005/08/addressing"
4   xmlns:wse="http://www.w3.org/2010/08/ws-evt" >
5   <s12:Header>
6     <wsa:Action>
7       http://www.w3.org/2010/08/ws-evt/SubscriptionEnd
8     </wsa:Action>
9     <wsa:To>soap.udp://239.192.0.101:4002</wsa:To>
10  </s12:Header>
11  <s12:Body>
12    <!-- ggf. Nutzdaten -->
13  </s12:Body>
14 </s12:Envelope>
```

Quelltext 4.5: Exemplarische SubscriptionEnd-Nachricht.

dem während des Subscribe-Prozesses definierten Filter. Das `wsa:Message`-Element enthält die eindeutige Identifikation der Nachricht in Form eines *Universally Unique Identifier (UUID)* [67]. Ein UUID ist ein weltweit eindeutiger Bezeichner. Das `wsd:AppSequence`-Element ist nicht Teil des WS-Eventing-Standards und spielt für wiederholt versendete Benachrichtigungen in Abschnitt 4.3.4 eine Rolle.

Eine Besonderheit im Zusammenhang mit der Multicast-Erweiterung stellt das `wsa:To`-Element dar. Während einer Benachrichtigung typischerweise die Endpunkt-Referenz des Empfängers hinzugefügt wird, ist im Multicast-Fall die Angabe der Multicast-Adresse ausreichend.

4.3.3. Modifikation der dritten Phase

Die dritte Phase ist durch die Unsubscribe- sowie SubscriptionEnd-Nachricht charakterisiert. Eine Ereignissenke beendet ein Abonnement üblicherweise, indem es eine Unsubscribe-Nachricht versendet. Im Rahmen der Multicast-Erweiterung ändert sich an dieser Anfrage nichts. Zusätzlich sollte jedoch nach dem UnsubscribeResponse ebenfalls das Abonnement des zugrundeliegenden Multicast-Kanals via IGMP bzw. MLD beendet werden.

Für den Fall, dass ein SubscriptionEnd an alle Abonnenten verteilt wird, kann ebenfalls die Multicast-Funktionalität eingesetzt werden. Quelltext 4.5 illustriert eine entsprechende Nachricht an eine Multicast-Adresse. Nach dem Empfang einer per Multicast versandten SubscriptionEnd-Nachricht quittieren alle Ereignissenken ihr WS-Eventing- als auch Multicast-Abonnement. Um bei vielen Netzwerkteilnehmern die Kollisionen von Nachrichten zu vermeiden, sollte eine Eregnissenke eine randomisierte Zeit warten, bevor es die IGMP- bzw. MLD-Instruktionen initiiert.

4.3.4. Probleme der Multicast-Kommunikation

Die Verteilung mit UDP-Multicast hat den Vorteil, eine Nachricht effizient an mehrere Teilnehmer versenden zu können. Bedauerlicherweise gehen mit UDP-basierter Multicast-Kommunikation auch Nachteile und Einschränkungen einher. Zunächst wird Multicast-Kommunikation nur in Subnetzwerken umfassend unterstützt. Internetweit konnte sie sich noch nicht durchsetzen. Der Grund dafür ist, dass Internetknotenpunkte Datenbanken mit Multicast-Adresszuweisungen speichern und verwalten müssten. Für die damit verbundenen Kosten gibt es bislang keine adäquaten Bezahlmodelle [125]. Eine technische Alternative zur internetweiten Multicast-Kommunikation wurde 1992 von Savetz et al. [119] entwickelt. Das als *MBONE* bezeichnete System nutzt ein Overlay-Netzwerk, um Multicast-Nachrichten zunächst in Subnetzwerken mit multicast-unterstützender Hardware auszutauschen und bei Bedarf über Unicast-Kanäle in andere Subnetze zu tunneln. MBONE konnte sich nicht durchsetzen und spielt für die Industrie praktisch keine Rolle mehr.

Ein weiterer Nachteil ergibt sich aus der Nutzung des verbindungslosen UDP-Protokolls. Da Pakete in IP-Netzwerken bei Überschreiten der *Maximum Transmission Unit (MTU)* fragmentiert werden, kann es vorkommen, dass eine über UDP versendete Nachricht in mehreren Teilen ohne Erhaltung der Reihenfolge beim Empfänger eintrifft. Da keine Paketnummerierung existiert, sind die Nachrichten nicht mehr rekonstruierbar. Darüber hinaus ist es auch denkbar, dass Fragmente während des Transports verloren gehen und die Nachricht damit nicht vollständig beim Empfänger eintrifft. Neben der Nachrichtenfragmentierung ist im UDP-Protokoll ebenfalls keine Methodik zur Empfangsbestätigung von Nachrichten vorgesehen. Eine Ereignisquelle kann somit nicht nachvollziehen, ob alle Senken die Benachrichtigung erhalten haben.

Um den aufgezählten Problemen zu begegnen, gibt es verschiedene Möglichkeiten. Für die Unterstützung von internetweitem Multicast sind zunächst Finanzierungsmodelle erforderlich. Vorschläge hierzu überschreiten den Fokus dieser Arbeit und sind für die Vernetzung in limitierten IT-Krankenhausnetzen irrelevant. Gegen die Nachrichtenfragmentierung lassen sich die nachfolgenden Lösungsansätze wählen.

Eine offensichtliche Variante besteht darin, das Überschreiten der MTU zu vermeiden. Die Größe der MTU ist jedoch von der Netzwerkstruktur und -Hardware abhängig und somit in jedem Anwendungsfall unterschiedlich. Gemäß DPWS gilt eine Nachrichtengröße von 4096 Bytes als unbedenklich [100, Abschnitt 2.2]. In jedem Fall sollte die Nachricht so klein wie möglich serialisiert werden. Hierzu könnten z.B. Kompressionsverfahren wie FastInfoset [68] oder *Efficient XML Interchange (EXI)* [163] beitragen. Alternativ ist auch der Einsatz von auf UDP basierenden Protokollerweiterungen denkbar [141, S. 143ff]. Mithilfe einer Nachrichtennummerierung von UDP-Paketen ließe sich die Reihenfolge empfangener SOAP-Nachrichten rekonstruieren. Hier wiederum entsteht jedoch der Nachteil, dass derartige UDP-Erweiterungen unzureichend

```
1 [Action]
2   http://www.w3.org/2011/03/ws-evt/sume/
        NotificationRetransmission
3 [Message Body]
4   <sume:NotificationRetransmission
5    InstanceId="123"
6    MessageNumber="456"
7    SubscriptionManager="http://subman-uri.net"/>
```

Quelltext 4.6: Exemplarische RetransmissionRequest-Nachricht.

bekannt und standardisiert sind. Somit ist eine Umsetzung nur unter erheblichem Aufwand realisierbar.

Zur Steigerung der Zuverlässigkeit bietet sich der Versand von negativen Bestätigungen an. Die als *Negative Acknowledgements (NACKs)* bezeichneten Nachrichten sind jedoch nur dann sinnvoll einsetzbar, wenn die Ereignissenken wissen, zu welchen Zeitpunkten Benachrichtigungen eintreffen müssen. In Publish-Subscribe-Systemen mit periodischen Zustellungen können NACKs benutzt werden, um eine erneute Benachrichtigungszustellung anzufordern. Periodische Übertragungen kommen beispielsweise bei kontinuierlichen Signalen zum Einsatz. Im Bezug auf den OP wären dies etwa Herzfrequenzkurven. Um die Funktionalität des NACK umzusetzen, bedarf es der Festlegung einer Deadline für die periodischen Benachrichtigungen. Hierfür wird dem `sume:NotificationDestination`-Endpunkt der Referenzparameter `sume:Period` hinzugefügt. Dieser Parameter beschreibt die maximale Dauer t zwischen dem Versand zweier Benachrichtigungen in Millisekunden.

Erhält eine Senke eine Benachrichtigung n und bleibt der Empfang der Benachrichtigung $n+1$ nach t Millisekunden aus, sendet es einen *RetransmissionRequest* an die Ereignisquelle. Damit die Ereignisquelle weiß, um welche Nachricht es sich handelt, müssen für periodische Benachrichtigungen Sequenznummern mitgeführt werden. Hierfür sieht der WS-Discovery-Standard beispielsweise das `wsd:AppSequence`-Element vor, welches im Header einer SOAP-Nachricht zu finden ist. Ein Beispiel zeigt Quelltext 4.4 auf Seite 72, Zeile 13. Das Attribut `InstanceId` entspricht einem eindeutigen, inkrementierten Wert, wohingegen `MessageNumber` die Sequenznummer einer Nachricht darstellt. Das `wsd:AppSequence`-Element wird im Rahmen dieser Erweiterung für periodische Benachrichtigungen wiederverwendet.

Ein Beispiel eines RetransmissionRequest zeigt Quelltext 4.6. Die Ereignisquelle stellt auf eine solche Anfrage hin die verlorene Benachrichtigung mit Hilfe der SOAP-Response-Nachricht erneut zu. Die Quelle muss deshalb periodische Benachrichtigungen mit Retransmission-Unterstützung für den Zeitraum der Deadline vorhalten können. Neben der `wsd:AppSequence`-Information enthält eine Retransmission-Anfrage den Subscription-Manager, um das Abonnement zuordnen zu können. Kann die an-

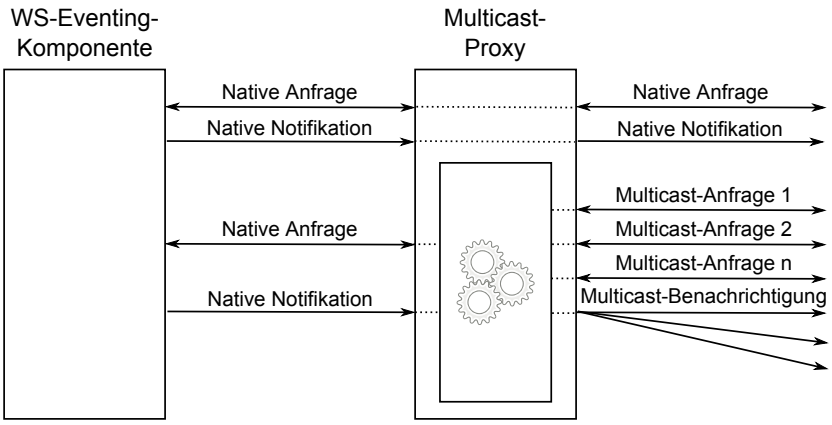


Abbildung 4.3.: Umsetzung eines Multicast-Proxys zur Bereitstellung des SOAP-over-UDP-Multicast.

geforderte Benachrichtigung nicht mehr zugestellt werden, wird ein SOAP-Fault mit Code `s12:Sender` und Subcode `sume:RetransmissionFailed` geliefert.

4.4. Implementierung

Dieser Abschnitt präsentiert die prototypische Implementierung der WS-Eventing-Erweiterung in bestehende Middleware-Systeme. Die im Folgenden als Multicast-Proxy bezeichnete Software-Komponente wird hierbei mithilfe des Proxy-Entwurfsmusters [32, S. 213ff] mit der nativen WS-Eventing-Implementierung (WS-Eventing-Komponente) verbunden. Die Architektur des Systems ist in Abbildung 4.3 dargestellt. Alle Anfragen durchlaufen den Multicast-Proxy, wobei gewöhnliche Anfragen an die WS-Eventing-Komponente durchgereicht und Multicast-Anfragen vom Proxy verarbeitet werden. Alle Multicast-Abonnements werden vom Multicast-Proxy verwaltet und stellen sich für die WS-Eventing-Komponente transparent als ein einzelnes Abonnement dar.

Die Umsetzung basiert auf dem frei verfügbaren WS4D-JMEDS-Framework [170, 77] in der Version 2.0beta7. Zur Verwaltung der Multicast-Abonnements wird eine zusätzliche Multicast-Proxy-Klasse eingeführt, welche die Eventing-Kommunikation verarbeitet und an die native WS-Eventing-Komponente weiterleitet. Der Multicast-Proxy speichert alle Multicast-Abonnements in einer Hash-Map. Beim Versenden einer Multicast-Benachrichtigung aus der nativen WS-Eventing-Komponente an den Multicast-Proxy wird das `wsa:To`-Element angepasst und die Nachricht an die im Proxy hinterlegte Multicast-Adresse weitergeleitet. Die Nachricht wird hierbei nur einmalig in XML umgeformt, womit sich für n nativ serialisierte Nachrichten eine

Effizienzsteigerung um den Faktor n ergibt. Dies betrifft jedoch nicht die Belastung des Netzwerkes. Im *Wireless LAN (WLAN)* erfolgt standardmäßig ein Broadcast, Hub-Geräte erzeugen Duplikate ausnahmslos für jede Ethernet-Verbindung und Switch-Geräte für jeden Multicast-Empfänger. Jedoch muss der Nachrichtensender seine Verbindung zum Verteiler nur einmal in Anspruch nehmen.

Für die NACK-Funktion puffert der Multicast-Proxy anfallende Benachrichtigungen und kümmert sich um den erneuten Versand verloren gegangener Nachrichten. Um die Wahrscheinlichkeit zu erhöhen, dass eine Benachrichtigung für den erneuten Versand zur Verfügung steht, werden sie um die zweifache Länge der NACK-Deadline vorgehalten. Die Berechnungskomplexität für die Verwaltung von `wse:Expires`-Informationen lässt sich senken, indem jeweils nur das zeitlich längste Abonnement herangezogen wird. Scheidet die jeweilige Senke vorher aus den Abonnements aus (z. B. durch Versenden einer Unsubscribe-Nachricht), werden dennoch Benachrichtigungen verschickt. Da der Multicast-Kanal ohnehin von jedem abgehört werden kann, stellt dies keine Erhöhung des Sicherheitsrisikos dar. Es führt jedoch dazu, dass ggf. Benachrichtigungen den Kommunikationskanal belasten, ohne das eine Verwertung der Daten erfolgt. Für unbegrenzt laufende Abonnements wird ein Zähler mitgeführt. Läuft der Zähler ab, ist also kein unbegrenzt laufendes Abonnement mehr vorhanden, wird ein Renew-Request an die native WS-Eventing-Komponente gestellt. Der Renew-Request enthält dabei den Zeitpunkt des zeitlich längsten Abonnements aus dem Multicast-Proxy.

4.5. Evaluation

Im vorangegangenen Abschnitt wurde bereits erläutert, dass der wesentliche Performance-Gewinn durch die reduzierte Serialisierung auf Seite der Ereignisquelle erzielt wird. Der theoretisch zu erzielende Zeitgewinn liegt hiermit für n zu serialisierende Nachrichten bei n. Hierbei ist jedoch zu berücksichtigen, dass native WS-Eventing-Implementierungen den Serialisierungsaufwand ebenfalls verringern können, indem sie eine serialisierte Vorlage im Speicher vorhalten. In dieser Vorlage ist vor dem Versand jeder Benachrichtigung lediglich die Endpunkt-Referenz zu ergänzen.

Da UDP im Gegensatz zu TCP weder über Flusskontrolle noch Paketnummerierungen verfügt, ist damit zu rechnen, dass Pakete während des Transports verloren gehen. Die Multicast-Erweiterung eignet sich aus diesem Grund nur für die Übertragung von kleinen XML-Dokumenten, bei denen der Verlust tolerierbar ist. Für periodisch wiederkehrende Ereignisse kann ein NACK genutzt werden, um verlorene Benachrichtigungen erneut anzufordern. Die in diesem Kapitel entworfene SOAP-over-UDP-Multicast-Erweiterung ist ein zu DPWS kompatibles Transportprotokoll, mit dem streaming-basierte Datenübertragung durchgeführt werden kann. Es stellt daher eine Alternative zu Transportprotokollen, wie sie im DICOM-Standard vorgesehen sind, dar, um eindimensionale Signalverläufe zu übertragen.

4.6. Ergebnis

In diesem Kapitel wurde eine Protokollerweiterung für die WS-Eventing-Spezifikation vorgestellt. Die Erweiterung erlaubt die Bindung von UDP-Multicast zur effizienten Verteilung verlustbehafteter, kleiner XML-Dokumente. Es eignet sich somit zum Beispiel für die Verteilung von periodisch wiederkehrenden Benachrichtigungen oder für die stream-basierte Codierung eindimensionaler Signalkurven. Mit Hilfe von NACK-Anfragen kann die Übertragungszuverlässigkeit für periodisch versandte Nachrichten gesteigert werden.

Kapitel 5.

Robuste und halbautomatisierte Verteilung elektronischer Patientendaten

In Kapitel 3 wurden verschiedene Anwendungsprozesse beschrieben und eine prototypische, SODA-basierte Middleware für die Gerätekonnektivität eruiert. Ein Szenario aus diesem Kapitel ist die im Rahmen der vertikalen Integrationsrichtung genannte Übertragung von elektronischen Patientendaten auf die an einer Intervention beteiligten Medizingeräte. Wie aktuelle Untersuchungen zeigen, ist dieser Anwendungsfall bis heute sehr fehleranfällig [27, S. 11f]. Elektronische Patientendaten werden im Englischen als *Electronic Health Record (EHR)* bezeichnet. Sie stehen im Allgemeinen für eine systematische Ansammlung digital als auch analog vorliegender Patienteninformationen, die für eine einrichtungsübergreifende Verwertung vorgesehen sind. Darunter fallen z. B. die Patientenstammdaten, Bild- und Videomaterial, Medikationen, Vitalparameter sowie Diagnosedaten [130]. Im Kontext dieses Kapitels sind EHRs beliebig gefilterte elektronische Patientendaten, die individuell in Abhängigkeit des jeweiligen Verwertungskontextes gewählt sind.

Einige Geräte wie Operationsmikroskope, Navigationssysteme sowie Blutgasanalysegeräte benötigen für die Erzeugung von personen- bzw. fallbezogenem Dokumentationsmaterial verschiedene Patientendaten. Es gibt im Wesentlichen drei Wege, wie diese Patientendaten und das erfasste Dokumentationsmaterial zusammengebracht werden können. Ein fehleranfälliger und aufwändiger Weg besteht in der Kopplung digitaler und analoger Medien. Dabei werden beispielsweise die von einem Blutgasanalysegerät gemessenen Werte ausgedruckt und anschließend zusammen mit der Patientenkennung in eine analoge Patientenstammakte übertragen. Eine weitere Methodik besteht in der manuellen Datenübernahme an allen betroffenen Medizingeräten durch händisches Abtippen. Bevor die Schnitt-Naht-Zeit, also die Zeit der eigentlichen Operation, beginnt, trägt ein Assistent alle erforderlichen Patientendaten an der Mensch-Maschine-Schnittstelle von jedem Medizingerät ein. Der letzte und eleganteste Weg ergibt sich bei modernen Medizingeräten aus der Abfrage einer externen Datenbank. Üblicherweise werden hierfür anhand einer Punkt-zu-Punkt-Verbindung mit einem KIS bzw. OPMS Patientenlisten abgefragt, ein dedizierter Eintrag akkreditiert und lokal im

jeweiligen Medizingerät vorgehalten. Dieser Prozess ist für alle anderen Geräte, die ebenfalls an Patientenstammdaten interessiert sind, wiederholt durchzuführen.

In diesem Kapitel wird ein Konzept zur halbautomatisierten und robusten EHR-Verteilung vorgestellt. Die halbautomatisierte Verteilung hat den Vorteil, dass Daten nicht mehr separat an einzelnen Geräten eingelesen und akkreditiert werden müssen, sondern eine Synchronisierung mit allen interessierten Parteien durchgeführt wird. Die Akkreditierung ist nur noch an einem einzelnen Gerät erforderlich und nicht wie bisher separat an jeder Mensch-Maschine-Schnittstelle. Aus regulatorischen Gründen kann auf eine einmalige Bestätigung eines EHR-Datensatzes nicht verzichtet werden, weshalb eine vollautomatische Patientendatenverteilung nicht durchführbar ist. Im Folgenden wird der Begriff *automatisiert* im Kontext der Patientendatenverteilung synonym mit dem Begriff *halbautomatisiert* verwendet. Die einmalige Akkreditierung bleibt dennoch ein Teil des Anwendungsprozesses.

Das entwickelte Protokoll ist robust gegenüber Geräteausfällen sowohl auf Seite des Datenlieferanten als auch -konsumenten. Das Protokoll garantiert, dass alle Geräte (außerhalb von Fehlerfällen) zu jedem Zeitpunkt über aktuelle EHRs verfügen. Die Inhalte dieses Kapitels wurden vorab in [38] veröffentlicht.

5.1. Verwandte Technologien

Es gibt derzeit keine vergleichbaren Arbeiten zur automatisierten Patientendatenverteilung, da bisher ausschließlich die Punkt-zu-Punkt-Abfrage umgesetzt wird. Dies hängt zum einen mit der Zulassungsproblematik im medizintechnischen Umfeld und zum anderen mit der technologischen und semantischen Vielfalt der existierenden Protokolle zusammen. Im Folgenden werden die populärsten Ansätze zur EHR-Akquise vorgestellt und bewertet.

HL7 ist ein umfassender Standard für die Abbildung von Krankenhausprozessen. Der zur Zeit im klinischen Umfeld verbreitete Standard HL7 Version 2.x definiert zahlreiche Nachrichten für administrative, logistische, betriebswirtschaftliche und klinische Prozesse. Für alle Aufgaben gibt es Anfragen, die einer bestimmten Nachrichtenklasse zugeordnet werden [50]. Für die Aufnahme, Verlegung und Entlassung von Patienten sind dies z. B. Nachrichten vom Typ *Admission, Discharge and Transfer (ADT)*. Zur Abfrage von Patientendaten gibt es den Typ *Query By Parameter (QBP)*. HL7v2.x erlaubt somit die proaktive Abfrage von Patientenstammdaten. Hierfür muss jedoch vorab die technische Methodik des Nachrichtenaustauschs bekannt sein. Dazu gehört beispielsweise das Wissen, ob der Datenaustausch über Dateien oder Sockets erfolgt. Ein Mechanismus für die automatische Verteilung der Patientendaten ist nicht vorgesehen.

Im *Domain Information Model (DIM)* der ISO/IEEE 11073-10201 gibt es eine Klassenbeschreibung für Patientenstammdaten [65, S. 147ff]. Gemäß der Definition im Standard entspricht der Umfang dieser Klasse gerade der Menge an Informationen, die von medizinischen Geräten benötigt werden. Unter dem Klassen-Identifier *MDC_MOC_PT_DEMOG* sind verschiedene Attribute zusammengefasst, die anhand eines *GET*- oder *EVENT-REPORT*-Services von einer Patientendatenquelle empfangen werden können. Hiermit wäre eine automatische Patientendatenverteilung generell umsetzbar. Das DIM bietet jedoch keinen Prozess, der es erlaubt, die Datensätze mit einer einzelnen Bestätigung auf alle interessierten Geräte zu verteilen. Die Patientendaten müssten somit an jedem Gerät einzeln bestätigt werden, womit sich keine wesentliche Arbeitsersparnis für das Krankenhauspersonal ergäbe. Es bliebe außerdem zu definieren, wie im Fall von Geräteausfällen sowohl auf Seite der Quelle- als auch der Senke zu verfahren ist.

Ein häufig verwendeter Ansatz zur Abfrage von Patienteninformationen sind DICOM-Worklists. Das Worklist-Konzept steht im vierten Teil des DICOM-Standards, der *Service Class Specification*, beschrieben [84, S. 186ff]. Es ist ähnlich wie HL7 in der Applikationsschicht angesiedelt und basiert auf einem Request-Response-Muster, das unabhängig von der zugrundeliegenden Kommunikationsschicht über verschiedene physikalische Wege transportiert werden kann. Zunächst ist der integrale Bestandteil einer Worklist eine Ansammlung mehrerer Aufgaben, denen eine Ausführungsreihenfolge zugrunde liegt. Für die Abarbeitung einer Worklist finden sich immer zwei *Application Entitys (AEs)* zusammen. Die AE, welche die Worklist anfragt, wird als *Service Class User (SCU)* bezeichnet. Die angefragte AE heißt *Service Class Provider (SCP)*. Beide Rollen lassen sich auf das SOA-Paradigma zurückführen. Um eine Worklist zu verarbeiten, kann der SCU die Methode *C-FIND* ausführen, der ein *Service Object Pair Class Unique Identifier (SOP-UID)* übergeben wird. Der SOP-UID wiederum determiniert das abzufragende Worklist-Model, welches beispielsweise Patientendaten beschreibt [83, S. 337ff]. Die zahlreichen Datenmodule können im dritten Teil des DICOM-Standards nachgelesen werden [83]. Obwohl auch DICOM in der Lage ist, Patientendaten abzurufen, existiert kein Protokoll für die automatische Verteilung an mehrere AEs. Da für jeden Datenaustausch zunächst eine Peer-to-Peer-Verbindung zwischen einem SCP und einem SCU aufgebaut werden muss, ist eine standardkonforme Umsetzung nicht durchführbar.

Integrating the Healthcare Enterprise (IHE) [46] ist eine seit 1998 durch Medizinproduktehersteller als auch Anwender vorangetriebene Initiative mit dem Ziel eines einheitlichen Datenaustausches im Gesundheitswesen. Anstatt neue Standards zu definieren, versucht sich die Initiative an der Harmonisierung bestehender Standards wie der ISO/IEEE 11073, HL7, DICOM sowie von W3C-Spezifikationen. Sie beschreibt, wie die Standards zu verwenden sind und veröffentlicht dazu sogenannte Integrationsprofile. Die IHE bietet hierfür ein öffentlich zugängliches Interoperabilitätsforum an und bindet fortlaufend Experten aus beteiligten und betroffenen Organisationen ein, um relevante Abläufe mit validen Anforderungen zu spezifizieren. Die Arbeiten

der IHE sind dabei in verschiedene Domänen unterteilt wie z. B. der Pathologie, Kardiologie und Augenheilkunde. Eine weitere Domäne wird als IT-Infrastructure bezeichnet und beschäftigt sich u. a. mit einer Methodik zum Abgleich unterschiedlicher Patienten-IDs und dem Austausch von Patientenstammdaten. Bei letzterem greift die IHE hauptsächlich auf die Nachrichtentypen aus HL7 Version 2.x [49] sowie der Web-Services/XML-unterstützenden HL7 Version 3 [48] zurück. Da die IHE-Profile hinsichtlich des Patientendatenaustausches auf HL7 basieren, sind auch diese Arbeiten für die automatisierte Datenverteilung nur bedingt einsetzbar. Das grundlegende Konzept eignet sich jedoch für die Beschreibung eines neuen Protokolls, welches in den folgenden Abschnitten vorgestellt wird.

5.2. Annahmen und Voraussetzungen

Die automatisierte EHR-Verteilung ist hinsichtlich des ISO/OSI-Schichtenmodells auf der Anwendungsebene angeordnet. Dadurch ergeben sich Bezüge zu zahlreichen Aufgaben, die von einer Middleware zu erfassen sind. In diesem Abschnitt sind die Annahmen und Voraussetzungen für das zu entwickelnde Anwendungsprotokoll in Bezug auf eine zugrundeliegende Middleware skizziert.

5.2.1. Discovery und Publish-Subscribe

Um den Konfigurationsaufwand für den Verteilungsprozess zu minimieren, wird ein dynamischer Discovery-Dienst benötigt, mit dem Geräte selbständig gefunden und für die Datensynchronisierung vorbereitet werden können. Darüber hinaus ist für die Synchronisierung der EHRs ein Publish-Subscribe-Mechanismus notwendig, um Änderungen an Datensätzen mitteilen zu können. Die in Abschnitt 3.3 vorgestellte DPWS-Middleware unterstützt WS-Discovery für die Geräteauffindung sowie WS-Eventing für die Publish-Subscribe-Funktionalität und bildet somit eine technische Grundlage für die Verteilung der Patientendaten.

5.2.2. Authentifikation und Autorisierung

IT-Sicherheit spielt im Krankenhaus eine essentielle Rolle. Dazu gehören insbesondere die Nachvollziehbarkeit von Interaktionen mit IT-Systemen sowie die Privatsphäre der Patienten. In Abschnitt 2.3.4 auf Seite 32 wurde bereits ein dezentralisiertes Konzept vorgestellt, um die Kommunikation in verteilten Systemen abzusichern. Dieses Konzept kann ebenfalls für das hier vorgestellte Verfahren angewandt werden, jedoch mit der Besonderheit, dass neben der Geräte- auch eine Personenauthentifizierung erforderlich ist. Die Personenauthentifizierung könnte analog durch ein mit einem Root-Zertifikat des Betreibers ausgestellten Zertifikats erfolgen. Somit müsste jede EHR-Bestätigung

als Nachweis mit der digitalen Signatur des Angestellten, der den Datensatz bestätigt hat, versehen werden.

Im Rahmen dieser Arbeit wird angenommen, dass ein adäquates Authentifizierungsverfahren vorliegt. Hierfür kommt z. B. WS-Security in Frage [96, S. 27ff]. Digitale Signaturen werden dabei im Header-Element einer SOAP-Nachricht hinterlegt. Das zu entwickelnde Protokoll kann somit ohne Einschränkungen um eine Authentifizierung erweitert werden.

5.2.3. Kommunikationskontext

Bevor EHRs sicher übertragen werden können, muss ein Kommunikationskontext zwischen den Geräten hergestellt werden. Ein Kommunikationskontext bedeutet in diesem Zusammenhang, dass Geräte durch eine gemeinsame Kennung wissen, in welchem logischen Kontext sie agieren. Dies ist wichtig, um eine versehentliche OP-übergreifende Fernsteuerung zu verhindern oder versehentlich Patientendaten an Geräte im Nachbar-OP zu schicken.

Als Kommunikationskontext im Operationssaal wäre die Zuordnung zu einem Patienten denkbar. Dieser Kontext kann als Patientenkontext bezeichnet werden. Je nach nationalen gesetzlichen Regeln oder den üblichen Vorgehensweisen in einem Krankenhaus werden für die Identifizierung eines Patienten nicht die Patientenkennung verwendet, sondern z. B. auch, wie in HL7, die Fall-ID [126]. Diese bezieht sich immer auf einen dedizierten Krankenhausaufenthalt. Eine Patienten-ID kann demnach mehreren Fall-IDs zugeordnet werden.

Für die Zuordnung eines Kommunikationskontextes eignen sich daher weitere Kriterien wie Raum, Subnetz und Gerätegruppen. Darüber hinaus kann es sinnvoll sein, einen virtuellen Kontext zu definieren, der eine OP-Sitzung repräsentiert. Hiermit ergeben sich die folgenden möglichen Kandidaten für eine Kontextbestimmung

- Patientenkontext
- Fallkontext
- Raumkontext
- Netzwerkkontext
- Gruppenkontext
- Sitzungskontext

Grundsätzlich entspricht der Kommunikationskontext einer eindeutigen Kennung, die zwischen beteiligten Geräten geteilt wird. Da sich der Kontext aus verschiedenen Randbedingungen ableiten lassen kann, sollte er jedoch nicht exklusiv in Form einer Patienten-ID oder Fall-ID vorliegen. Da mehrere Operationssäle an einem Subnetz hängen können und auf Intensivstationen üblicherweise mehrere Patienten in einem Zimmer liegen, ist ebenfalls davon abzuraten, das Subnetz als exklusiven Kommunikationskontext einzusetzen.

Die Akquise des Patientenkontextes ist nicht trivial. Generell kann die Kontextakquise manuell oder automatisch erfolgen. Für die automatische Akquise bedarf es eines intelligenten Systems aus präzisen Sensoren, um Geräte lokalisieren zu können. Bis heute ist keine zufriedenstellende Lösung für eine feingranulare und zuverlässige geographische Lokalisierung von Objekten entwickelt worden. Darüber hinaus ist für die Determinierung eines Kontextes nicht nur der räumliche Zusammenhang maßgebend, sondern auch der Patient. Somit reicht der ausschließliche Verlass auf sensorisch erfasste Daten nicht aus und es müssen weitere Kriterien für eine Kontextbestimmung herangezogen werden.

Da die Beschreibung einer automatischen Kontextakquise den Rahmen dieser Arbeit sprengen würde, nutzt das vorgestellte Protokoll Identifikationsnummern, die manuell vorgegeben werden. Um den damit einhergehenden Arbeitsaufwand für das Krankenhauspersonal zu verringern, wäre es denkbar, diese Nummern initiativ festzulegen. Diese Systematik funktioniert für alle ortsinvarianten Geräte, lässt sich allerdings unzureichend auf mobile Einheiten wie Röntgen- oder Blutgasanalysegeräte übertragen.

5.2.4. Semantik

Eine Vorbedingung für interoperable IT-Systeme ist die Standardisierung von Datenformaten und Semantiken bzw. Terminologien. Nur durch die strikte Regulierung aller für die Kommunikation erforderlichen Parameter können Geräte verschiedener Hersteller zusammenarbeiten. Mit Hilfe des DPWS ist durch SOAP, XML-Schema und WSDL bereits syntaktische Interoperabilität herstellbar. Um die Bedeutung übertragener Informationen sicherzustellen, wäre es denkbar, die Daten mit semantischen Beschreibungen wie *SNOMED CT* [52] auszuzeichnen. Andere Terminologien sind beispielsweise das HL7v3-RIM oder ISO/IEEE-11073-DIM. Die IHE empfiehlt die Benutzung des HL7v3-XML-Formats. Quelltext 5.1 zeigt einen beispielhaften HL7v3-EHR-Datensatz aus der Spezifikation der IHE *Patient Demographics Query (PDQ)*. Die Bedeutung der Daten ergibt sich aus der Definition im Standard und der XML-Schema-Angabe. Die Betrachtung von Semantiken und Terminologien beim Austausch von EHRs ist nicht Teil dieser Arbeit und daher vom Verteilungsprotokoll entkoppelt.

```
<patient classCode="PAT">
 <id root="1.2.840.114350.1.13.99998.8734"
   extension="34827R534"/>
 <statusCode code="active"/>
 <patientPerson>
  <name>
   <given>Jim</given>
   <family>Jones</family>
  </name>
  <telecom value="tel:+1-795-555-4745" use="HP"/>
  <administrativeGenderCode code="M"/>
  <birthTime value="19630713"/>
  <addr>
   <streetAddressLine>
    8734 Blue Ocean Street
   </streetAddressLine>
   <city>Other City</city>
   <state>IL</state>
  </addr>
  <!-- ... -->
  </patientPerson>
 <!-- ... -->
</patient>
```

Quelltext 5.1: Beispielhafter HL7v3-EHR-Datensatz aus der Spezifikation der IHE-PDQ [47].

5.3. Verteilungsprotokoll

Für die EHR-Übertragung zwischen der IT-Infrastruktur des Krankenhauses und den Medizingeräten im OP bzw. auf der Intensivstation existieren Ansätze, die das Krankenhauspersonal nur bedingt bei der Arbeit unterstützen. Im Allgemeinen fehlt ein Mechanismus, der die automatisierte Verteilung der EHRs auf die Geräte ermöglicht. Die IHE-Initiative hat mit dem *IT Infrastructure Technical Framework (ITI-TF)* eine Basisarchitektur definiert, die jedoch ausschließlich an einem Request-Response-Muster angelehnt ist (vgl. Abbildung 5.1). Gleiches gilt für DICOM-Worklists. Für eine synchronisierte Datenhaltung sind jedoch Publish-Subscribe-Systeme erforderlich, da sich ansonsten die Datenverteilungsprozesse unnötig verkomplizieren. Daraus folgt, dass sowohl der IHE- als auch der DICOM-Ansatz aus architektonischer Sicht ungeeignet sind. Die ISO/IEEE 11073 hingegen definiert im Teil 20601 [66, S. 48ff] die Operationen *GET*, *SET*, *ACTION* und *EVENT REPORT*. GET und SET sind Dienste für die generische Abfrage bzw. das Setzen von Werten. Mit ACTION können Aktionen auf einem Gerät ausgeführt werden, womit im Wesentlichen Gerätefernsteuerung unterstützt wird. Die EVENT-REPORT-Operation repräsentiert ein Publish-Subscribe-

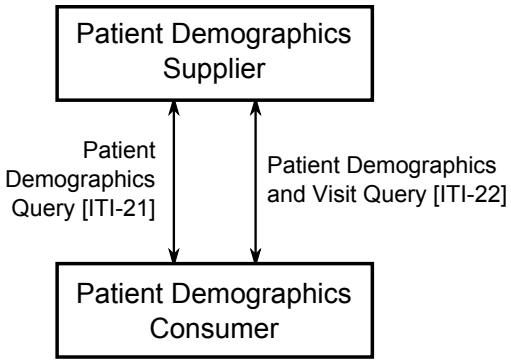


Abbildung 5.1.: In der IHE-PDQ-Architektur des ITI-TF heißen Server und Client *Patient Demographics Supplier* und *Patient Demographics Consumer*. Die beiden Transaktionen ITI-21 und ITI-22 ermöglichen den Zugriff auf Patientenstammdaten im HL7-Format.

Kommunikationsmuster und kann von Systemen genutzt werden, um z. B. Änderungen an Messwerten oder Alarme mitzuteilen. Das Vorgehen der ISO/IEEE 11073, GET und EVENT REPORT zur Datensynchronisierung zwischen medizinischen Systemen einzusetzen, kann als adäquate Methodik für die EHR-Akquise genutzt werden. Anstelle der komplexen und zum Teil veralteten Kommunikationstechnologien aus der ISO/IEEE 11073 basiert der hier beschriebene Ansatz jedoch auf DPWS (es sind generell auch andere technologische Unterbauten vorstellbar). Nachfolgend wird zunächst die Architektur und anschließend das Verfahren für die Verteilung beschrieben.

5.3.1. Architektur

In Abbildung 5.2 ist eine beispielhafte physikalische Infrastruktur eines OP-Saals illustriert. Üblicherweise wird ein Subnetz über die Geräte aufgespannt und von der IT-Infrastruktur des Krankenhauses abgeschottet. Dieser Schritt ist deshalb wichtig, da durch das Verbinden eines medizinischen Produktes mit einem IT-Netzwerk das IT-Netzwerk selbst zum Medizinprodukt wird und die Regularien aus dem MPG einhalten muss. Diese Beschränkung stellt für die Vernetzung von medizinischen Geräten unterschiedlicher Risikoklassen hinsichtlich der Zulassung ein enormes Problem dar. Für die Betrachtungen in diesem Kapitel bleibt dieses Problem vorerst unberücksichtigt, da es in ein anderes Themengebiet fällt. Unabhängig von der heutigen Zulassungsproblematik erfolgt die Abschottung auch aus datensicherheitstechnischen Gründen.

Um Informationen aus dem IT-Netzwerk des Krankenhauses in das LAN des Operationssaals zu transportieren, wird ein Gateway-System eingesetzt. Das Gateway-System

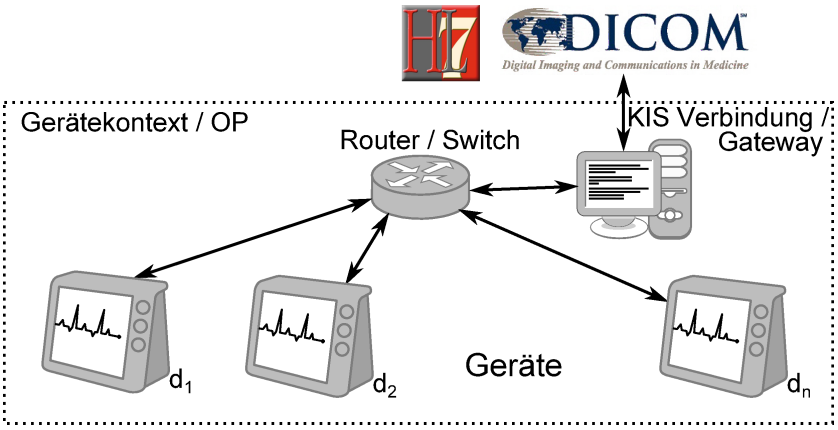


Abbildung 5.2.: Physikalische Infrastruktur eines Operationssaals. Das KIS ist typischerweise mit einem Gateway-Rechner verbunden. Die direkte Verbindung mit dem Router/Switch ist nicht ratsam, da dies bedeuten würde, dass das lokale Gerätenetz Teil des Krankenhausnetzwerkes ist.

könnte beispielsweise ein zentraler OP-Rechner mit zwei Netzwerkkarten sein. Ein gemeinsamer Kommunikationskontext erlaubt allen Geräten im OP-Saal den Bezug zu einem Patienten. In dieser vereinfachten Darstellung würde der Kommunikationskontext durch das Subnetz eindeutig determiniert. Der Router bzw. das Switch verbindet die Geräte über eine logische Bus-Topologie, so dass jeder Netzwerkteilnehmer mit jedem anderen Daten austauschen kann.

Die EHR-liefernde bzw. EHR-konsumierende Seite der Kommunikation werden in Anlehnung an die IHE als *Patient Demographics Supplier (PDS)* bzw. *Patient Demographics Consumer (PDC)* bezeichnet (auch wenn Patientenstammdaten streng genommen nicht das gesamte Spektrum patientenrelevanter Daten abbilden). Sofern ein geeigneter Kommunikationskontext existiert, setzt sich der generelle Prozess für die Verteilung der EHRs aus vier Schritten zusammen.

1. PDC-Geräte suchen im Netzwerk nach einem PDS.
2. Jeder PDC abonniert beim PDS die für sich passenden EHR-Daten.
3. Jeder PDC abonniert außerdem einen *Heartbeat-Notifier*. Der Heartbeat-Notifier ist ein Publish/Subscribe-Dienst, der in periodischen Abständen eine leichtgewichtige Benachrichtigung verschickt. Damit signalisiert der PDS, dass er sich in einem stabilen Zustand befindet.
4. Jeder PDC erfragt beim PDS initial, ob ein geeigneter EHR-Datensatz vorhanden ist.

Präfix	Spezifikation
`wse`	WS-Eventing [150]
`s12`	SOAP 1.2 [152]
`wsa`	WS-Addressing [149]
`wsd`	WS-Discovery [106]
`wst`	WS-Transfer [166]
`wsm`	WS-MetadataExchange [165]

Tabelle 5.1.: Liste verwendeter Namensräume für die Patientendatenverteilung.

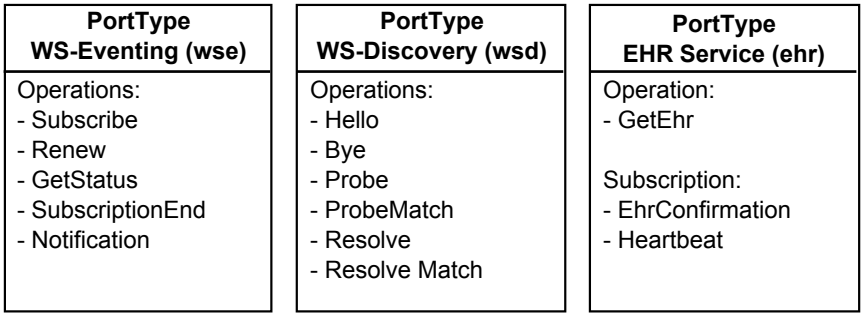


Abbildung 5.3.: Erforderliche Dienstschnittstelle für jeden PDS. Die im Kasten *EHR Service* beschriebenen Operationen kommen in der Anwendungsschicht neu hinzu. Die restlichen Operationen der Port-Types von WS-Eventing und WS-Discovery sind in der zugrundeliegenden DPWS-Middleware vorhanden. In Klammern steht jeweils das Namensraumpräfix des Port-Types.

5.3.2. Detaillierte Protokolldefinition

Das vorgestellte Protokoll bedient sich standardisierter Methoden aus dem Web-Service-Bereich, um die Kompatibilität mit DPWS herzustellen. Alle für das Protokoll relevanten Namensräume werden durch ein Präfix abgekürzt verwendet und sind in Tabelle 5.1 aufgelistet. Der speziell für das Protokoll eingeführte Namensraum ist bisher nicht standardisiert und wird im Rahmen dieser Arbeit als ein möglicher Kandidat vorgeschlagen, um die WSDL-Schnittstelle für die EHR-Verteilung zu beschreiben. Der Namensraum lautet

```
xmlns:ehr = "http://applications.devices.med/2012/03/ehr".
```

Typischerweise verfügt der Gateway-PC über eine Schnittstelle zum KIS bzw. PACS und kann folglich alle denkbaren Patientendaten abfragen. Eine Konvertierungseinheit kann die Übersetzung der Daten in ein DPWS-kompatibles Web-Service-Format übernehmen oder sie in ihrer binären Ursprungsform weiterleiten. Die in diesem Protokoll angewandten Schnittstellen sind in Abbildung 5.3 dargestellt. Die zugehörige WSDL-Beschreibung befindet sich im Anhang A.1 ab Seite 129. Zunächst suchen alle in einem Kontext liegenden PDC-Geräte im Netzwerk nach einem PDS, das die Daten von einem Gateway-PC verwerten kann. Hierfür verschickt jedes Gerät mit Hilfe von WS-Discovery eine `wsd:Probe`-Anfrage per Multicast ins Netzwerk. Der Kontext wird durch das WS-Discovery-spezifische Element `wsd:Scope` charakterisiert. Das `wsd:Scope`-Element enthält eine beliebige Menge an URIs und kann als Filter für den Probe-Vorgang genutzt werden. Ein illustrativer Eintrag wäre beispielsweise ein vorab vereinbarter UUID [67]. Das `wsd:Type`-Element enthält im Gegensatz zum `wsd:Scope` einen URI zur Kennzeichnung eines PDS und ist hier spezifiziert als

`http://applications.devices.med/2012/03/ehr#PDS`.

Die `wsd:Probe`-Anfrage wird von allen Geräten mit passendem `wsd:Scope` und `wsd:Type` durch ein `wsd:ProbeMatch` quittiert. Jeder `wsd:ProbeMatch` enthält (mindestens) eine Endpunkt-Referenz, von der ein PDC die WSDL-Schnittstelle gemäß des Vorgehens aus der DPWS-Spezifikation erfragen kann. Falls es mehrere PDS-Instanzen im Subnetzwerk gibt, kann eine beliebige gewählt werden. PDCs müssen sich darauf verlassen können, dass jeder EHR-Lieferant kohärente Daten anbietet. Sollten zwei oder mehrere PDS-Instanzen disjunkte EHR-Daten zur Verfügung stellen, muss der Akquiseprozess bei allen Instanzen separat durchgeführt werden. Für die Zuordnung eines PDS kann die eindeutige Endpunkt-Referenz des Gerätes genutzt werden. Diese ist gemäß der DPWS-Spezifikation über Reinitialisierungen hinweg eindeutig [100, Abschnitt 2.5].

Nach Abschluss des Suchvorgangs abonniert der PDC in einem ersten Schritt mit Hilfe von WS-Eventing seine gewünschten EHR-Daten. Sowohl das XML-Schema der Nutzendaten als auch ein geeigneter Filterdialekt zur Selektion obliegen einer Standardisierungsmaßnahme, die nicht Teil dieses Protokolls ist. Da DPWS lediglich ein Mindestmaß an Filterdialekten vorgibt, können ohne Beeinträchtigung der Flexibilität weitere Dialekte eingebunden werden. Ein Vorschlag für die Selektion von Patientendaten ist im Anhang A.2 auf Seite 132 beschrieben. In Abbildung 5.4 ist das Sequenzdiagramm für den Basisprozess illustriert. Nach dem Discovery-Prozess erfolgt die Subscribe-Anfrage für die EHR-Daten als auch des Heartbeat-Signals. Die Heartbeat-Benachrichtigungen sind zur Wahrung der Übersichtlichkeit in Abbildung 5.4 nicht dargestellt. Mit der Bestätigung des EHR-Datensatzes durch das klinische Personal (Caregiver) verschickt das PDS eine EhrConfirmation-Benachrichtigung an alle Abonnenten. Sobald ein PDC nicht mehr an Daten interessiert ist, beendet er seine Abonnements.

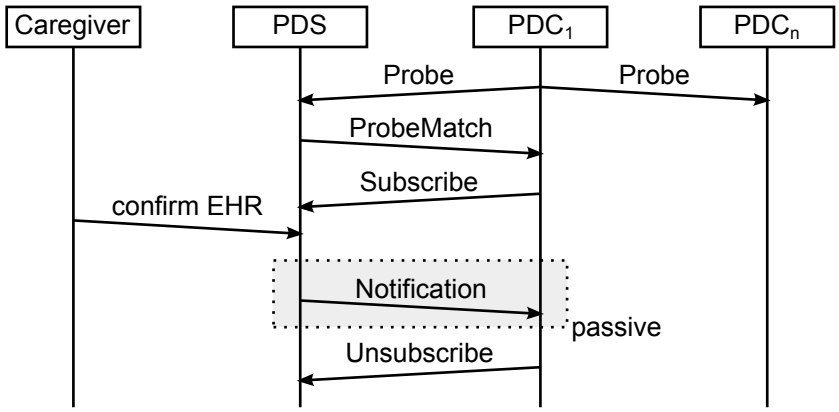


Abbildung 5.4.: Basisprozess zur Verteilung von EHR-Daten. Zunächst finden sich PDS und PDC mit Hilfe von WS-Discovery. Anschließend schreibt sich der PDC beim PDS für die Patientendaten ein. Die Daten erhält der PDC über ein passives Verfahren in Form einer Benachrichtigung.

Sofern zugesichert werden kann, dass ein EHR-Datensatz am PDS erst bestätigt wird, nachdem alle PDCs ihr EHR-Abonnement durchgeführt haben, ist das oben genannte Vorgehen zielführend. In einer Prozesskette mit menschlichen Akteuren kann eine solche Zusicherung im Normalfall nicht garantiert werden. Um dennoch eine automatisierte Datenübernahme zu gewährleisten, ist neben der in Abbildung 5.4 gezeigten passiven Push-Methodik ein aktives Pull seitens des PDCs erforderlich. Abbildung 5.5 zeigt den Prozessablauf unter Verwendung von zwei Kommunikationsphasen. Mit Hilfe der zusätzlichen aktiven Datenabfrage ist das Protokoll in der Lage, den temporären Ausfall von PDC-Instanzen zu kompensieren. Darüber hinaus schließt es den Fall ein, dass die EHR-Bestätigung durch einen menschlichen Akteur bereits vor der Subscription-Phase eines PDCs erfolgen kann.

Für die Detektierung veralteter Daten muss ein PDS jeden Datensatz mit einer eindeutigen Kennung auszeichnen. Hierfür eignen sich zum Beispiel UUIDs, die in einem dedizierten SOAP-Body-Element mit jedem EHR-Datensatz übertragen werden. Wenn ein PDC einen Datensatz abfragt oder eine Benachrichtigung eintrifft, erhält er über das Element `ehr:RecordId` den eindeutigen Bezeichner u_{origin}. Ein beispielhafter SOAP-Body ist in Quelltext 5.2 dargestellt. Ein formales Schema aller XML-Datentypen dieses Protokolls ist im Anhang A.1 ab Seite 129 verzeichnet. Um den eindeutigen Bezeichner bei einem Geräteausfall nicht zu verlieren, persistiert der PDC das Datum als u_{last}. Trifft ein weiterer Datensatz vom PDS ein, signalisiert $u_{origin} \neq u_{last}$, dass sich der EHR geändert hat. Die Daten im PDC müssen aktualisiert werden. Ist $u_{origin} = u_{last}$, kann der eingetroffene Datensatz ignoriert werden.

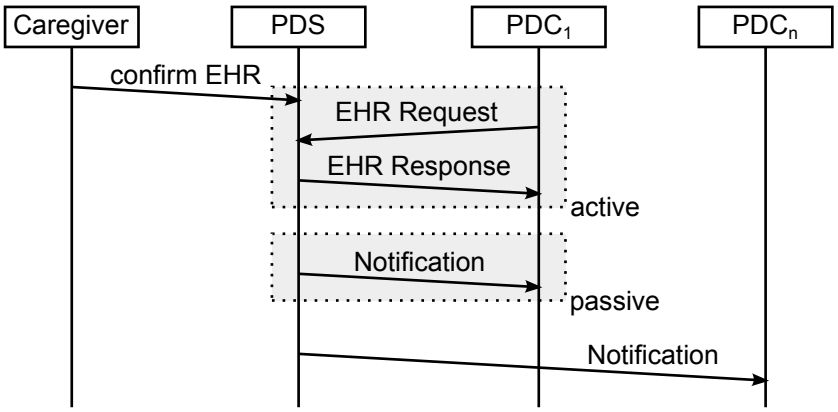


Abbildung 5.5.: Prozesskette für den Zwei-Phasen-EHR-Akquiseprozess. Mit einer aktiven und einer passiven EHR-Abfrage kann die Akkreditierung der Daten am PDS zu einem beliebigen Zeitpunkt erfolgen.

Der letzte Protokollschritt unterstützt die Erkennung von PDS-Ausfällen. Damit jeder PDC in Erfahrung bringen kann, ob der PDS noch aktiv ist und die Patientendaten aktuell sind, kann eine Benachrichtigung abonniert werden, die in periodischen Zeitabschnitten ein Heartbeat-Signal an alle PDCs verschickt. Jede Benachrichtigung liefert sowohl die Deadline für die nächste Benachrichtigung als auch die Endpunkt-Referenz des PDS (vgl. Quelltext 5.3). Es ist denkbar, die Heartbeat-Nachricht auch mit Hilfe des in Kapitel 4 vorgestellten UDP-Verfahrens zu übermitteln. Trifft eine Benachrichtigung nicht rechtzeitig ein, so sollte der betroffene PDC proaktiv überprüfen, ob der PDS noch verfügbar ist. Dazu fragt der PDC den Zustand des Heartbeat-Abonnements mit Hilfe von `wse:GetStatus` ab.

Damit ein PDS beschreiben kann, welche patientenrelevanten Daten er anbietet und in welchem Format diese vorliegen, ist die Nutzung von WS-Policy einsetzbar. Mit WS-Policy definiert ein PDS die Art der Daten, das Datenformat sowie das Code-System der abrufbaren EHRs. Das normative Policy-Modell ist zusammen mit der vorgeschlagenen Filtertechnik im Anhang A.2 auf Seite 132 skizziert. Eine exemplarische Policy-Assertion für ein PDS, der Patientendaten im fiktiven IEEE/ISO-11073-XML-Format liefert, ist in Quelltext 5.4 illustriert. Die Policy-Assertion beschreibt mehrere Kanäle, die verschiedene Arten von Patientendaten zur Verfügung stellen. Darunter fallen beispielsweise Patientenstammdaten, Bildmaterial und Medikationen. Da es wie in Quelltext 5.4 Fälle geben kann, in denen ein Code-System durch mehrere synonyme Bezeichner charakterisiert ist, kann die Policy mehrere Code-Systeme pro Kanal aufnehmen.

```
<ehr:EhrSet>
  <ehr:CodingSystem version="20130114">
    x73_XML
  </ehr:CodingSystem>
  <ehr:RecordId>
    urn:uuid:dd3efc30-5121-11e3-8f96-0800200c9a66
  </ehr:RecordId>
  <!-- EHR-Nutzdaten -->
</ehr:EhrSet>
```

Quelltext 5.2: Beispielhafter SOAP-Body einer GetEhr-Abfrage bzw. einer EhrConfirmation-Benachrichtigung. Die Version und der Identifikationsbezeichner des Codierungssystems sind fiktiv gewählt. Im vorliegenden Quelltext könnte es sich beispielsweise um eine XML-Repräsentation von ISO/IEEE-11073-Termen handeln. Im Feld `ehr:RecordId` befindet sich der eindeutige Identifikationsbezeichner u_{origin} für den EHR-Datensatz.

```
<ehr:Heartbeat>
  <ehr:DeviceEpr>
    urn:uuid:58fb8d10-4def-11e3-800e-f99cec06d47e
  </ehr:DeviceEpr>
  <ehr:Timeout>5000</ehr:Timeout>
</ehr:Heartbeat>
```

Quelltext 5.3: Beispielhafter Heartbeat-Datensatz, bestehend aus dem Timeout bis zum nächsten Heartbeat als auch der Geräte-Endpunkt-Referenz zur Identifikation des Heartbeats.

5.4. Evaluation

Das hauptsächliche Ziel bei der Gestaltung eines neuartigen Protokolls zur robusten und halbautomatischen Verteilung von Patientendaten ist die Senkung des Arbeitsaufwandes für das diensthabende Personal. Mit Hilfe von WS-Discovery und WS-Eventing ist es gelungen, einen Verteilungsprozess zu definieren, der EHR-Lieferanten in einem Subnetzwerk dezentralisiert finden und die Daten automatisch an interessierte Geräte übertragen kann. Die Datenbasis bleibt auch bei einem Systemausfall einzelner EHR-Datensätze erhalten. Es ist lediglich eine einmalige Bestätigung durch einen menschlichen Akteur erforderlich, um die Daten auf alle interessierten Geräte zu bringen.

Um die Effektivität und Effizienz aufzuzeigen, wird im Folgenden eine prototypische Umsetzung des vorgestellten Protokolls in einer exemplarischen Testumgebung evaluiert. Für den Versuch stehen zehn gleichartige HP-xw4200-Workstations zur Verfügung.

```
<ehr:DataSource>
  <ehr:Channel>
    <ehr:Type>http://applications.devices.med/2012/03/ehr/
       PatientDemographics</ehr:Type>
    <ehr:SupportedCodingSystem>
      <ehr:CodingSystemId version="20130114">
        x73_XML
      </ehr:CodingSystemId>
      <ehr:CodingSystemId version="20130114">
        11073-10101-XML
      </ehr:CodingSystemId>
    </ehr:SupportedCodingSystem>
  </ehr:Channel>
</ehr:DataSource>
```

Quelltext 5.4: Exemplarisches WS-Policy-Modell eines PDS.

Jeder PC ist mit einem *Intel Pentium 4 CPU 3.4 GHz 2 Core*, 4 GB RAM und einer Netzwerkkarte der Marke *Broadcom NetXtreme BCM 5751 Gigabit Ethernet PCI Express* ausgestattet. Alle Rechner sind über ein 1-GB-Switch des Typs *Cisco Catalyst 4506* miteinander verbunden. Das Protokoll ist mit Hilfe des DPWS-Frameworks JMEDS [77] 2.0beta7 umgesetzt worden und wird auf einer Java 7 Virtual Machine getestet. Auf einem der Rechner ist das EHR-liefernde PDS-System installiert. Die neun restlichen Einheiten sind mit einem PDC ausgestattet.

Zunächst werden zum Nachweis der Effektivität verschiedene Testfälle durchgeführt. Diese Testfälle zeigen, dass das System robust arbeitet und die Anforderungen hinsichtlich der automatischen Verteilung und einmaligen Bestätigung erfüllt. Die Testfälle werden mithilfe einer prototypischen Software durchgeführt. Die Software nutzt das JMEDS-Framework [134] in der Version 2.0beta7, um die Service-, Disovery- und Publish-Subscribe-Komponenten des DPWS abzubilden.

1. Der PDS wird gestartet und ein EHR akkreditiert. Anschließend werden PDC_1 bis PDC_9 gestartet. Jeder PDC erfragt nach dem Finden des PDS und dem Abonnment des EHRs mittels GetEhr die Patientendaten. Die Synchronisierung ist somit erfolgreich.

2. Der PDS wird gestartet. Nach dem Boot-Vorgang werden PDC_1 bis PDC_9 gestartet. Mit Abschluss des Suchvorgangs, der Einschreibung für das EHR-Abonnement und der GetEhr-Abfrage wird ein Datensatz akkreditiert. In diesem Fall erhalten alle PDCs die Informationen durch eine EhrConfirmation-Benachrichtigung. Die Synchronisierung ist erfolgreich.

3. Der PDS ist nicht verfügbar. PDC_1 bis PDC_4 werden gestartet. Danach erfolgt die Aktivierung des PDS mit anschließender Akkreditierung des EHR-

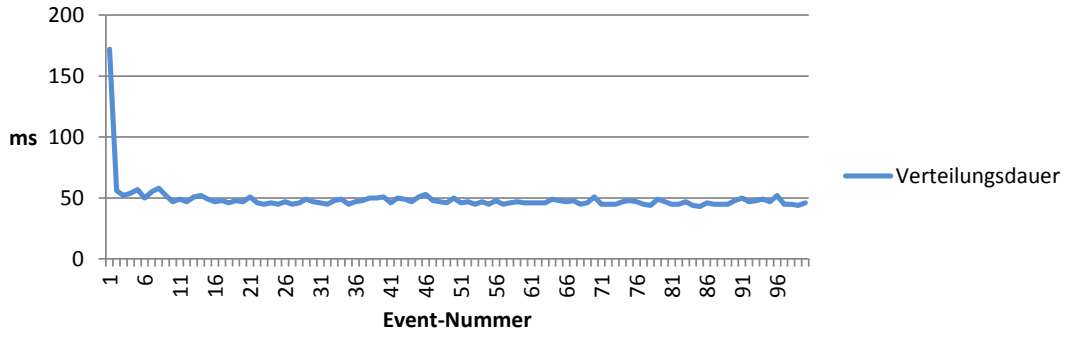
Datensatzes. Zum Schluss werden die restlichen PDCs hochgefahren. PDC_1 bis PDC_4 erhalten den akkreditierten EHR-Datensatz über eine EhrConfirmation-Benachrichtigung und PDC_5 bis PDC_9 über ein GetEhr. Mit Abschluss des Vorgangs sind alle PDCs synchronisiert.

4. Nach der Verteilung der Patientendaten wird ein beliebiger PDC ab- und eingeschaltet, um einen Ausfall im System zu simulieren. Hierbei wird zudem die Abmeldung anhand einer WS-Discovery-Bye-Message unterlassen. Sobald der ausgefallene PDC wieder betriebsbereit ist, sucht er nach dem für seine Daten zuständigen PDS. Dessen Endpunkt-Referenz als auch die EHR-Daten hat der PDC vorab persistiert und gleicht diese mit Hilfe eines GetEhr-Requests und den eindeutigen UUIDs der vorliegenden EHR-Daten ab. Der zuständige PDS bemerkt aufgrund der eindeutigen Endpunkt-Referenz des ausgefallenen PDCs, dass dieser sich erneut anmeldet, und entfernt das veraltete EHR- als auch Heartbeat-Abonnement. Das System befindet sich anschließend in einem stabilen und mit allen PDC-Geräten synchronisierten Zustand.

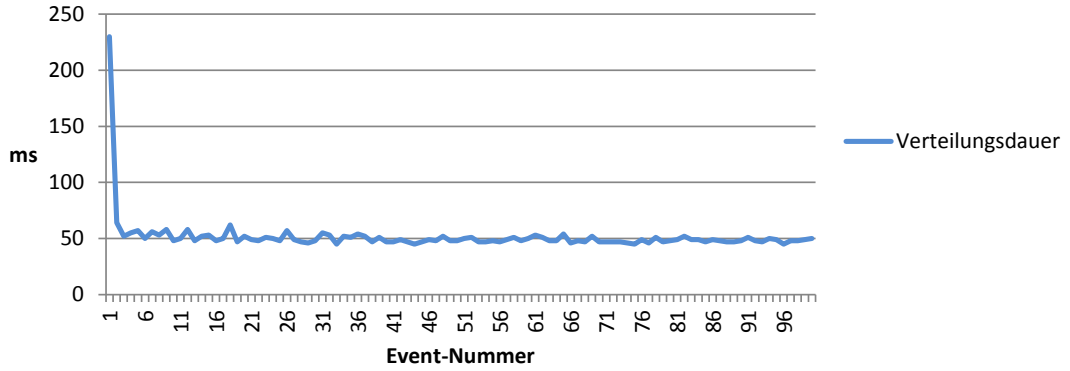
5. Im letzten Fall wird nach der Verteilung der Patientendaten ein Ausfall des PDS simuliert. Da nach Ablauf des Timeouts PDC_1 bis PDC_9 keine Heartbeat-Nachricht erhalten haben, gehen sie davon aus, dass die vorliegenden Patientendaten möglicherweise veraltet sind. Die PDCs versuchen sich mit dem ausgefallenen PDS über eine GetEhr-Anfrage zu verbinden. Schlägt die Anfrage fehl, entfernen sie die WS-Eventing-Abonnements und markieren ihre Daten als veraltet. Mit dem Beitreten des PDS zum Netzwerk über WS-Discovery-Hello wird der Verteilungsprozess erneut initiiert. Anschließend sind die Daten synchronisiert.

Als nächstes wird die Verteilungsdauer für die EHR-Benachrichtigungen mit drei, sechs und neun Abonnenten gemessen. Die Dauer für das Auffinden der Geräte mit WS-Discovery fließt nicht mit in die Messung ein. Abbildung 5.6 illustriert die drei Messungen. Auf der X-Achse sind die Anzahl der versendeten Benachrichtigungen und auf der Y-Achse die korrespondierenden Zeiten abgetragen. Pro Versuchsreihe verschickt das PDS 100 EhrConfirmation-Nachrichten. Die Nachrichten enthalten jeweils etwa zehn Kilobyte Nutzdaten in Form eines exemplarischen IHE-PDQ-Response-Dokuments [47].

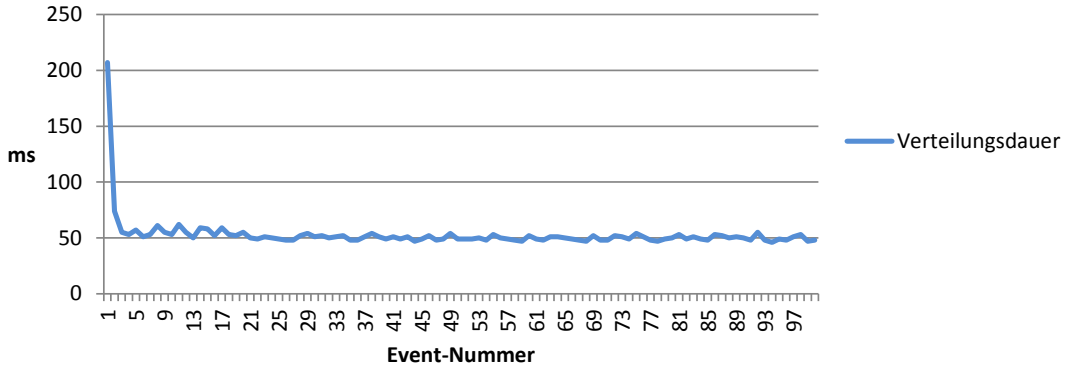
Es ist zu erkennen, dass in allen Versuchsreihen die ersten drei Benachrichtigungen wesentlich mehr Zeit für die Zustellung beanspruchen als die restlichen Benachrichtigungen. Dies ist darauf zurückzuführen, dass Java nach dem Start der Anwendung Variablen setzt, die im Anschluss wiederverwendet werden können. Anschließend pendelt sich die Übertragung auf etwa 50 Millisekunden ein. Für drei, sechs und neun PDCs ergibt sich kein wesentlicher Zeitunterschied für die Zustellung der Benachrichtigungen. Mit neun Geräten hat man für den OP-Bereich bereits einen realistischen Umfang an PDCs erreicht.



(a) Drei PDCs.



(b) Sechs PDCs.



(c) Neun PDCs.

Abbildung 5.6.: Entwicklung der Verteilungsdauer bei Variation der Anzahl an PDCs.

Es ist zu erwarten, dass sich die Gerätedichte im OP in Zukunft erhöhen wird. Als nächstes bleibt zu untersuchen, ob das Protokoll skaliert. Hierfür ist im Folgenden ein Modell für die Anzahl versendeter Nachrichten konstruiert und ausgewertet worden. Für die Ermittlung der Anzahl gilt: jeder verschickte SOAP-Envelope zählt als eine Nachricht. Eine Multicast-Nachricht mit K Multicast-Kanal-Abonnements entspricht dem Versand von K Nachrichten. Untersucht wird das Szenario, wenn sich alle PDCs im Netzwerk befinden und der PDS mit einer WS-Discovery-Hello-Nachricht beitritt. Die EHR-Daten werden lediglich auf einem Kanal, also einer `wse:Subscription`, ausgegeben. Die Anzahl an Nachrichten M für die EHR-Verteilung ergibt sich dann aus folgender Formel:

$$M = 2n + 12x + xp + 2\left\lfloor \frac{t}{r} \right\rfloor x + \left\lfloor \frac{t}{d} \right\rfloor x \tag{5.1}$$

Hierbei definiert $n \in \mathbb{N}$ die Anzahl an DPWS-Netzwerkteilnehmern, $x \in \mathbb{N}$ die Anzahl an PDCs, $p \in \mathbb{N}_+$ die Anzahl der Patientenwechsel (mindestens einer wird vorausgesetzt), $r \in \mathbb{R}_+$ die maximale Dauer zwischen zwei `wse:Renew`-Nachrichten, $d \in \mathbb{R}_+$ die maximale Dauer zwischen zwei Heartbeat-Nachrichten und $t \in \mathbb{R}_+$ den Zeitraum zwischen der Abonnements aller PDCs und dem Austritt des PDS aus dem DPWS-Netzwerkverbund. Es werden $2n$ Nachrichten verschickt, wenn der PDS dem Netzwerk beitritt (`wsd:Hello`) und austritt (`wsd:Bye`). Der Term $12x$ setzt sich aus folgenden Faktoren zusammen: $2x$ Nachrichten für `wst:Get` und `wst:GetResponse`, $2x$ für `wsm:GetMetadata` und `wsm:GetMetadataResponse` sowie $4x$ für das EHR- und Heartbeat-Abonnement ($2x$ `wse:Subscribe`- und $2x$ `wse:Unsubscribe`-Nachrichten).

Im vorliegenden Szenario ist $t < r$, $t < d$, $n = 10$, $x = 9$ und $p = 1$. Damit ergeben sich 137 versendete Nachrichten für die Synchronisierung von 9 PDCs mit einer Netzwerk-Teilnehmergröße von 10. Abbildung 5.7 skizziert die Entwicklung der Nachrichtenanzahl bei Variation der teilnehmenden PDCs. Anhand der Grafik lässt sich ein linearer Zuwachs ablesen. Im Verhältnis zu herkömmlichen Systemen für die Abfrage von Patientendaten entsteht ein massiver Aufwand hinsichtlich der Generierung und Übertragung von SOAP-Nachrichten. Dieser Aufwand ist jedoch in Hinblick auf die zugrundeliegende leistungsstarke Hardware der Rechner und des Netzwerkes irrelevant. Darüber hinaus kann mit der automatisierten Verteilung menschliche Arbeitskraft eingespart und die Patientensicherheit erhöht werden.

5.5. Ergebnis

Eine vollautomatisierte Patientendatenverteilung, also die Verteilung von EHR-Daten ohne die Bestätigung durch einen menschlichen Akteur, ist nach der derzeitigen Gesetzeslage nicht durchführbar.

In diesem Kapitel wurde deshalb ein Protokoll für die robuste und *halbautomatisierte* Verteilung elektronischer Patientendaten entwickelt. Das Protokoll basiert auf DPWS

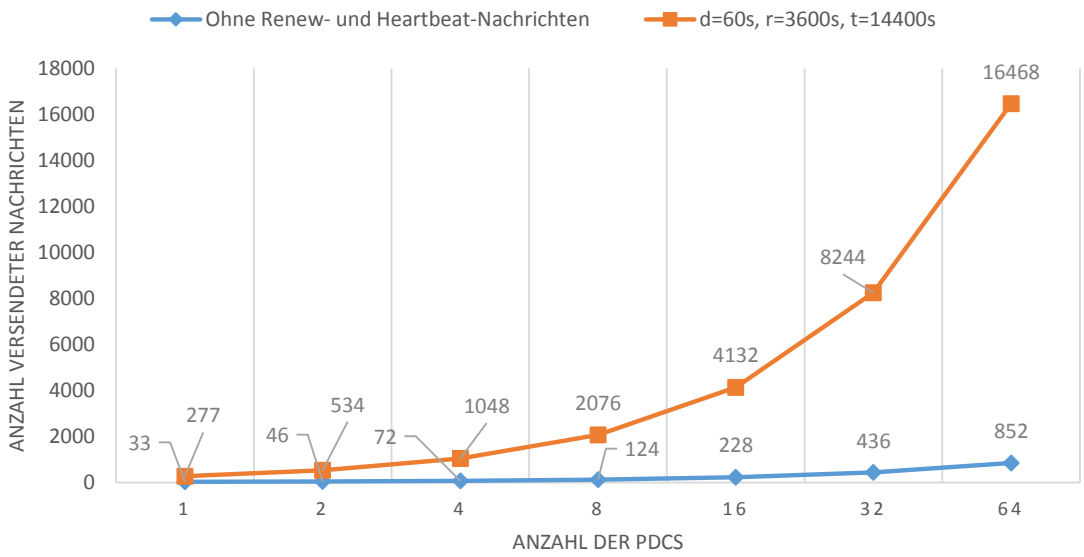


Abbildung 5.7.: Nachrichtenaufkommen für die EHR-Verteilung unter Berücksichtigung verschiedener Einflussfaktoren. Die Variablennamen d, r und t beziehen sich definitionsgemäß auf Formel 5.1.

und den Konzepten der IHE sowie IEEE/ISO 11073. Durch eine aktive und passive Datenabfrage sowie einer Heartbeat-Benachrichtigung sind alle EHR-Konsumenten zu jedem Zeitpunkt mit aktuellen Datensätzen synchronisiert. Im Vergleich zu existierenden Ansätzen müssen menschliche Akteure ausschließlich bei der einmaligen Akkreditierung der Daten in den Verteilungsprozess involviert sein. Somit ergibt sich ein direkter Vorteil hinsichtlich der Reduzierung des Arbeitsaufwandes für das OP-Personal. Da keine Daten mehr manuell übernommen werden müssen, erhöht sich außerdem die Patientensicherheit. Die Evaluation hat ergeben, dass im Vergleich zu verwandten Systemen ein erhöhter Nachrichtenaustausch erfolgt. Da moderne Hardware jedoch ausreichend Ressourcen zur Verfügung stellt, entsteht hierdurch kein Engpass hinsichtlich der Effizienz des Systems.

Offene Fragestellungen gibt es bezüglich der Umsetzung der Authentifikation und Autorisierung zwischen PDS- und PDC-Instanzen sowie der Umsetzung des Kommunikationskontextes. Darüber hinaus sollten zu Optimierungszwecken Filtertechnologien für den Zugriff auf die EHR-Daten entwickelt werden. Ein weiteres Forschungsziel ist neben der Verteilung der Patientendaten auch eine automatisierte, postoperative Dokumentation patientenbezogener Daten, die während einer Operation erhoben werden.

Kapitel 6.

Verzögerungsarmes Video-Live-Streaming auf Tablet-PCs und Smartphones

Zahlreiche chirurgische Eingriffe werden mit Unterstützung von bildgebenden Systemen wie Operationsmikroskopen oder Endoskopiegeräten durchgeführt. Operationsmikroskope erlauben u. a. Operationen an den filigranen Strukturen im Gehirn. Die Endoskopie unterstützt minimalinvasive Verfahren wie beispielsweise die Laparoskopie. In beiden Varianten wird üblicherweise Videomaterial aufgezeichnet und gegebenenfalls auf verschiedenen Monitoren im Operationssaal angezeigt. Die Aufzeichnung des Videomaterials ist für die Dokumentation, für anschließende Auswertungen oder zu Lehrzwecken vorgesehen. Die Live-Spiegelung kann von Zuschauern mitverfolgt werden. Sie ist ein adäquates Mittel für Lehrzwecke, da nicht alle Teilnehmer an einer OP direkte Einsicht in das Operationsumfeld haben.

Ein weiteres Anwendungsgebiet, in dem die Übertragung von Videomaterial einen wichtigen Stellenwert einnimmt, ist die Telemedizin. Bei der Telemedizin handelt es sich um die Verknüpfung von Telekommunikation und Informatik mit dem Gesundheitswesen, insbesondere der Pathologie, Radiologie und Chirurgie. Sie ist dabei als eine Unterklasse der umfassenden Gesundheitstelematik anzusehen [40, S. 6] und soll sowohl die Diagnostik als auch Therapie unter Überbrückung räumlicher Distanzen gewährleisten. Dabei lassen sich im Wesentlichen die Szenarien Observation und Interaktion unterscheiden. Bei der Observation wird der Operationsverlauf von einem Facharzt beobachtet und bewertet. Hierfür sind vor allem eine ausgezeichnete Bildqualität und eine verzögerungsarme, flüssige Wiedergabe erforderlich. Denkbar ist ebenfalls die Observation zu Lehrzwecken. Von Interaktion ist in diesem Zusammenhang die Rede, wenn Medizingeräte von einem Arzt ferngesteuert werden. Neben der verzögerungsarmen Bildübertragung benötigt der Arzt in diesem Fall zudem die unmittelbare Reaktion ferngesteuerter Geräte, da als nachlaufen empfundene Interaktionen für den Arbeitsprozess kontraproduktiv sind.

Mit der Entwicklung und dem Vertrieb von Smartphones bzw. Tablet-PCs wurde jüngst ein neuer Markt erschlossen, der das Potential hat, gewöhnliche Desktop-PCs und Laptops zunehmend aus dem Consumer-Segment zu verdrängen. Ein eindeutiges

Zeichen hierfür bietet die Verdoppelung der Absatzzahlen für die mobilen, berührungssensitiven Geräte zwischen 2009 und 2011 [11, 12]. Gemäß der Definition aus dem Duden zeichnet sich ein Smartphone dadurch aus, dass es im Vergleich zu gewöhnlichen Mobiltelefonen mit einem größeren als auch berührungssensitiven Bildschirm ausgestattet ist und über zusätzliche Sensorik verfügt [26]. Darüber hinaus sind Smartphones in der Lage, Apps[1] herunterzuladen und auszuführen. Tablet-PCs besitzen typischerweise ähnliche Leistungsmerkmale wie Smartphones, sind jedoch etwas größer dimensioniert. Damit ergibt sich in der Regel eine bessere Akkulaufzeit und außerdem ein größerer Anzeigebildschirm.

Auch im Unternehmensumfeld finden Smartphones und Tablet-PCs immer häufiger Anwendung, so dass sich die Frage stellt, inwieweit die Telemedizin von der neuen Technologie profitieren kann. In diesem Kapitel wird untersucht, ob sich die am Consumer-Markt etablierten mobilen Geräte für telemedizinische Anwendungen eignen. Die Live-Übertragung von Videosignalen ist hierbei von besonderem Interesse, da sie neue Möglichkeiten für die telemedizinische Observation hervorbringen. Beispielsweise könnten Bewertungen von externen Chirurgen eingeholt werden, ohne dass diese in ihrem Büro am Schreibtisch sitzen müssen. Es wäre auch denkbar, Tablet-PCs als alternative Monitore im OP-Saal einzusetzen. Die Ergebnisse aus diesem Kapitel wurden vorab in [39] veröffentlicht.

6.1. Video-Streaming- und Codierungsgrundlagen

Im vorhergehenden Abschnitt wurden die zwei wichtigsten Plattformen für mobile Betriebssysteme vorgestellt. Als nächstes werden die verwendeten Fachbegriffe aus dem Bereich des Video-Streaming und der Videocodierung erläutert. Anschließend wird untersucht, welche Technologien für das Live-Streaming in Frage kommen.

6.1.1. Video-Streaming

Video-Streaming kann generell in zwei Bereiche unterteilt werden: *Video-on-Demand (VOD)* und Live-Streaming. Bei VOD-Diensten erfolgt die Wiedergabe auf Basis heruntergeladener Videodateien. Hierbei steht die Bildqualität als auch die flüssige Wiedergabe als Qualitätsmerkmal im Vordergrund. Eine Startverzögerung von mehreren Sekunden ist tolerierbar. Das Live-Streaming hingegen erfordert eine möglichst verzögerungsarme Bildübertragung, wobei zugunsten der zeitlichen Anforderungen möglicherweise auf eine optimale Bildqualität oder flüssige Wiedergabe verzichtet werden muss. Die gängigen Betriebssysteme für Tablet-PCs und Smartphones unterstützen die im Folgenden erläuterten Protokolle.

[1] Apps sind Applikationen für mobile Betriebssysteme wie sie auf Smartphones oder Tablet-PCs zum Einsatz kommen. Der Begriff App ist ebenfalls im Duden festgehalten worden [24].

HTTP Progressive Downloading

Ein typischerweise von VOD-Diensten angewandtes Protokoll ist das HTTP Progressive Downloading [109]. Bei dieser Technik handelt es sich genau genommen nicht um eine Streaming-Technologie, sondern vielmehr um die Fähigkeit von Abspielgeräten, bereits geladene Teile einer durch HTTP abgerufenen Datei während des Downloads abzuspielen. Der Client lädt eine Videodatei von einem Web-Server und fängt bereits vor Abschluss des gesamten Downloads an, die Datei abzuspielen. Einige Web-Server können für das Live-Streaming Videodateien simulieren, die fortlaufend mit neuem Videomaterial befüllt werden. Eine Client-Instanz kann daraufhin ein Video unbekannter Länge laden und es während der Pufferung abspielen. Ein Nachteil ergibt sich daraus, dass beim HTTP Progressive Downloading eine Datei vom Web-Server auf den Client heruntergeladen wird. Dynamische Anpassungen an die verfügbare Bandbreite sind somit nicht möglich. Zudem verursacht das Verfahren für lange Videos entsprechend viel Speicherplatzverbrauch.

HTTP Live Streaming

Ein von Apple im Rahmen von QuickTime X entwickeltes Live-Streaming-Protokoll heißt HTTP Live Streaming und ist derzeit als IETF Internet Draft [63] publiziert. Es ist neben HTTP Progressive Downloading das einzige Streaming-Verfahren, das von Apple iOS unterstützt wird. HTTP Live Streaming basiert auf einem H.264-codierten Video-Stream, der in kleine Teile segmentiert ist. Als Container kommt ein MPEG2-Transport-Stream zum Einsatz. Die einzelnen Segmente werden von einem Web-Server zum Download angeboten und die Reihenfolge der Segmente durch eine M3U-Multimedia-Abspielliste vorgegeben. HTTP Live Streaming ist damit für VOD und Live-Streaming einsetzbar. Beim Live-Streaming wird die M3U-Liste kontinuierlich aktualisiert. Hiermit ist es anders als beim HTTP Progressive Downloading möglich, variable Bandbreiten zu berücksichtigen. Ein Nachteil dieses Streaming-Verfahrens besteht in der minimalen Länge eines Segments aus der M3U-Abspielliste, das gemäß der Richtlinien in der Spezifikation nicht unter zehn Sekunden liegen sollte. Kleinere Werte sind möglich, induzieren jedoch eine erhöhte Server-Last.

RTP/RTCP/RTSP

Das *Real-Time Transport Protocol (RTP)* [59], *Real-Time Control Protocol (RTCP)* [58] und *Real-Time Streaming Protocol (RTSP)* [56] bilden eine Protokollfamilie für die Übertragung audiovisueller Daten über ein IP-basiertes Netzwerk. RTP spielt in dem Konglomerat die Hauptrolle und beschreibt die Architektur sowie das Paketformat. Typischerweise kommt dabei UDP-Unicast oder -Multicast zum Einsatz. Das verwendete Transportprotokoll wird um weitere Metadaten wie Zeitstempel und Sequenznummern

erweitert. Unter Nutzung von echtzeitfähigem IP ist RTP somit ebenfalls echtzeitfähig, erlaubt jedoch keine Ressourcenreservierung oder Aushandlung einer Dienstgüte. Zu diesem Zweck wiederum wurde RTCP spezifiziert. Mit RTCP können QoS-Parameter ausgehandelt und eingehalten werden. Dazu tauschen Client und Server in periodischen Zeitabständen Steuernachrichten aus. Auf Basis der Steuernachrichten kann der Server z. B. mit adaptiver Bandbreitenanpassung auf Leistungsengpässe reagieren. Darüber hinaus erlaubt RTCP die Synchronisierung von Bild- und Tonsequenzen. Mit RTSP können weitere Kontrolloptionen für Video-Streams angewandt werden. Darunter fällt beispielsweise die Pausierung und Fortsetzung der Übertragung oder das Anspringen beliebiger Positionen.

RTMP

Das *Real Time Messaging Protocol (RTMP)* [110] ist ein von Adobe Systems Incorporated zunächst proprietär entwickeltes Protokoll zur Übertragung von Audio- und Videodaten von einem Medien-Server zu einem Flash-Player. Im April 2009 wurde die Protokollspezifikation offengelegt. RTMP nutzt standardmäßig TCP/IP für die Übertragung des Daten-Streams, kann jedoch auch über HTTP getunnelt werden und ist daher mit HTTPS-Bordmitteln in der Lage, verschlüsselte Übertragungen herzustellen. Da RTMP auf TCP aufbaut, skaliert es nicht so gut wie UDP-basierte Verfahren.

TS-over-UDP/RTP

Die *Moving Picture Experts Group (MPEG)* ist eine Gruppe von Experten zur Spezifikation von Audio- und Videokompressionsverfahren sowie zur Definition entsprechender Containerformate. Einer der bekanntesten Ableger ist MPEG-2, ein Standard für die verlustbehaftete Kompression von Audio- und Videodaten. Neben dem Kompressionsverfahren führt der Standard zudem den Transport-Stream MPEG-TS (kurz TS) ein. MPEG-TS erlaubt die Codierung und Synchronisierung von Bild- und Tonsignalen, entspricht also im Wesentlichen einem generischen Containerformat. Der Transport-Stream kann zum Beispiel direkt über UDP oder RTP übertragen werden. Dieses Übertragungsverfahren wird als TS-over-UDP/RTP bezeichnet.

6.1.2. Video-Codierungen

Der vorhergehende Abschnitt hat ausschließlich die von Tablet-PCs und Smartphones unterstützten Nachrichtenübertragungs-Protokolle vorgestellt. Im Folgenden wird erläutert, welche Video-Codierungen für die mobilen Betriebssysteme in Frage kommen und welche Containerformate verwendet werden können.

MPEG-2/TS

In Abschnitt 6.1.1 wurde bereits MPEG-TS als Containerformat von MPEG-2 genannt. Dieser im ersten Teil der MPEG-2-Spezifikation beschriebene Transport-Stream ist im zweiten bzw. dritten Teil der Spezifikation durch eine Video- bzw. Audiocodierung ergänzt.

MPEG-4/mp4

Ähnlich wie die Audio-Videocodierung aus MPEG-2 ist *MPEG-4 AVC* ein von der *Internationalen Fernmeldeunion (ITU)* empfohlener Standard zur effizienten Codierung von audiovisuellen Signalen. Er ist besser bekannt unter den Bezeichnungen MP4 und H.264. MPEG-4 benötigt bei gleichbleibender Qualität im Vergleich zu MPEG-2 eine geringere Datenrate. Dies liegt vor allem daran, dass für das MPEG-4-Kompressionsverfahren Wavelet-Tranformationen berechnet werden. Die benötigte Bandbreite zur Übertragung eines Video-Streams kann damit bei mehr als verdoppeltem Rechenaufwand halbiert werden [108].

VP8/WebM

VP8 ist ein zu H.264 konkurrenzfähiges Videokompressionsformat, das unter einer modifizierten BSD-Lizenz frei verfügbar ist. Die Videocodierung wurde ursprünglich von On2 Technologies entwickelt und später von Google übernommen. Er ist Teil der Video-Codec-Reihe *TrueMotion* und reiht sich in weitere patentfreie Codecs wie VP3 (Theora) und VP6 (Adobe Flash) ein. Das für VP8 übliche Containerformat heißt WebM und wird im Rahmen des von Google gesponserten WebM-Projektes [131] entwickelt. WebM ist im Gegensatz zu anderen Containerformaten ausschließlich für VP8-codiertes Video- und Vorbis-codiertes Audiomaterial vorgesehen. Somit kann jede WebM-kompatible Software das Containerformat auch tatsächlich abspielen. WebM wird inzwischen von allen gängigen Web-Browsern unterstützt. WebM-Decodierung wird derzeit nur von Android ab Version 2.3.3 angeboten.

Motion-JPEG

Gängige Videokompressionsformate wie MPEG-2, H.264 und VP8 nutzen typischerweise Inter-Frame-Codierungen zur Reduzierung des Datenaufkommens. Unter Inter-Frame-Codierung versteht man die Bezugnahme zwischen einzelnen Bildern einer Videosequenz. Anstatt jedes Bild separat zu codieren, können Differenzbilder und Bewegungsvektoren eingesetzt werden, um den Speicherplatzverbrauch des Videos zu senken. Die Verwendung solcher Verfahren benötigt jedoch neben der Decodierung der Bilder weitere Rechenleistung und Vorlaufzeit zum Abspielen. Darüber hinaus können

derartig komprimierte Videos nicht bildgenau geschnitten werden. Zwei mögliche Alternativen sind entweder die Verwendung von unkomprimierten Videosignalen oder *Motion JPEG (MJPEG)*.

Bei MJPEG handelt es sich um eine Intra-Frame-Codierung, d. h. es gibt keinen Bezug zwischen einzelnen Bildern einer Videosequenz. Diese Art der Codierung wird häufig in günstigen Aufnahmegeräten angewandt, da bis auf ein JPEG-Komprimierer keine weitere Software gebraucht wird und der Berechnungsaufwand auf die JPEG-Kompression beschränkt ist. MJPEG wird von vielen Webkit-basierten Browsern wie dem Apple Safari, Google Chrome und Mozilla Firefox nativ unterstützt. Da für das Einbinden lediglich ein HTML-Image-Tag notwendig ist, kann es im Smartphone- und Tablet-PC-Sektor flexibel eingesetzt werden und obliegt keiner Beschränkung wie beispielsweise einer Videoanzeige im Vollbildmodus.

6.1.3. Gütekriterien

Für die Bewertung eines Video-Live-Streams kommen verschiedene Gütekriterien in Frage. Drei übliche Kriterien sind die Auflösung, die Bitrate und die Ende-zu-Ende-Latenz. Die Auflösung wird durch die Anzahl der Bildpunkte in horizontaler und vertikaler Richtung angegeben. Je höher die Auflösung ist, desto schärfer kann das decodierte Bild dargestellt werden. Sie kann jedoch auch zu hoch gewählt sein, wenn die Auflösung der Anzeige beim Endgerät kleiner ausfällt. Dies hätte eine Verschwendung von Ressourcen hinsichtlich der Berechnung und Bandbreite zur Folge. Wird die Auflösung hingegen zu gering gewählt, resultiert dies im decodierten Video in einer sichtbaren Blockbildung mit Informationsverlust.

Die Bitrate eines Video-Streams gibt das Datenaufkommen an, das auf dem Kommunikationskanal für die Übertragung in Anspruch genommen wird. Ziel bei der Übertragung von audiovisueller Informationen ist stets die Reduzierung des Datenaufkommens, da bereits ein unkomprimiertes Video mit einer Auflösung von 320x240 Pixeln bei einer Farbtiefe von 32 Bit und 30 Bildern pro Sekunde den Kommunikationskanal mit etwa 73,7 MBit/s belastet. Im Vergleich liefert IEEE-802.11g-standardkonformes WLAN lediglich eine Bitrate von 54 MBit/s. Um die Auslastung zu regulieren, kann die verfügbare Bandbreite erhöht und die Bitrate des Video-Streams reduziert werden.

Als Ende-zu-Ende-Latenz eines Video-Streams bezeichnet man den zeitlichen Versatz zwischen der Aufzeichnung eines Bildes über die Sensorik eines Aufnahmegerätes bis zur Wiedergabe auf einem entfernten Anzeigebildschirm. Sie setzt sich somit aus verschiedenen Teillatenzen zusammen [41]. Abbildung 6.1 stellt die einzelnen Einflussfaktoren für die Ende-zu-Ende-Latenz auf. Die erste Latenzquelle entsteht bei der Bildaufnahme mit einer Kamera. Hochwertige High-Definition-Videokameras erzeugen bei einer 720p-Auflösung mit 60 Bildern pro Sekunde eine Latenz von weniger als fünf Millisekunden. Bei der Nutzung von *Firewire* oder ähnlichen Ausgabesystemen

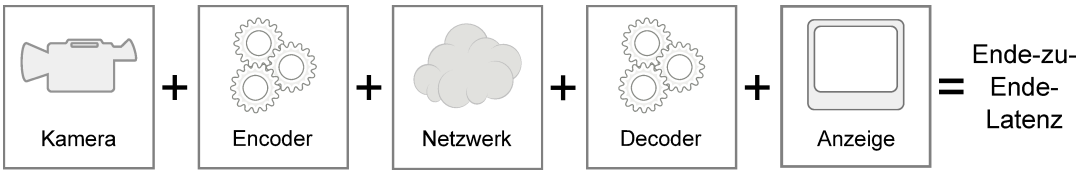


Abbildung 6.1.: Zusammensetzung der Ende-zu-Ende-Latenz bei der Übertragung von Video-Live-Streams durch Addition der aufgeführten Teillatenzen (in Anlehnung an [41]).

mit integrierter Videokompression kann sich die Latenz entsprechend erhöhen. Einige Kameras wenden außerdem Algorithmen zur Bildverbesserung an, die ebenfalls Auswirkungen auf die Latenz haben. Außerdem ist darauf zu achten, dass Kameras aus dem Consumer-Bereich spürbare Latenzen erzeugen können.

Ein wesentlicher Teil der Ende-zu-Ende-Latenz entsteht durch den Video-Encoder. Video-Codierungen wie MPEG-2 und H.264 verwenden aufwendige Berechnungen für die Videokompression. Ein Beispiel für die hohe Berechnungskomplexität ist die Bewegungserkennung, wie sie in MPEG-2 und H.264 Anwendung findet. Bei der Bewegungserkennung werden Objekte aus Referenzbildern detektiert und in nachfolgenden Bildern wiederverwendet. Dazu müssen sogenannte Makroblöcke gefunden und mit Hilfe von Bewegungsvektoren verschoben werden [108]. So entstehen z. B. beim Fernseh-Broadcast Verzögerungen von 300 bis 4000 Millisekunden [41, S. 2], um eine optimierte Bildqualität zu erhalten. Für eine effizientere Encodierung können statt der üblichen Software-Encoder auch Hardware-Encoder eingesetzt werden. Hardware-Encoder verfügen über spezialisierte Codierungs-Chips. Damit wird eine Encodierung in bis zu 50 Millisekunden berechnet.

Ein ebenfalls signifikanter Latenzfaktor ist das Netzwerk, welches für die Übertragung eines Video-Streams verwendet wird. Es erzeugt an verschiedenen Punkten Verzögerungen. Darunter fallen die Netzwerkkarten von Sender und Empfänger, die Anzahl der Zwischenstationen (Router/Switches), die Leitungskapazität und das Leitungsmedium sowie die Distanzüberbrückung. Ein einfach geswitchtes Netzwerk kommt praktisch ohne Jitter und Paketverlust aus. Die Zeit für die Überbrückung der Distanz liegt typischerweise unter weit unter zehn Millisekunden [41, S. 3]. Demgegenüber benötigen WANs wesentlich mehr Zeit für die Übertragung und leiden unter einer erhöhten Varianz in der periodischen Zustellung von Paketen. Um die Auswirkungen der Varianz zu kompensieren, werden Warteschlangen beim Decoder eingesetzt, welche die Latenz weiter erhöhen.

Nach der Übertragung der Daten über ein Netzwerk entsteht die nächste Teillatenz im Decoder. Dieser benötigt im Verhältnis zum Encoder und Netzwerk wenig Zeit, um den Video-Stream zu decodieren. Der Hardware-Decoder der Firma Hai-

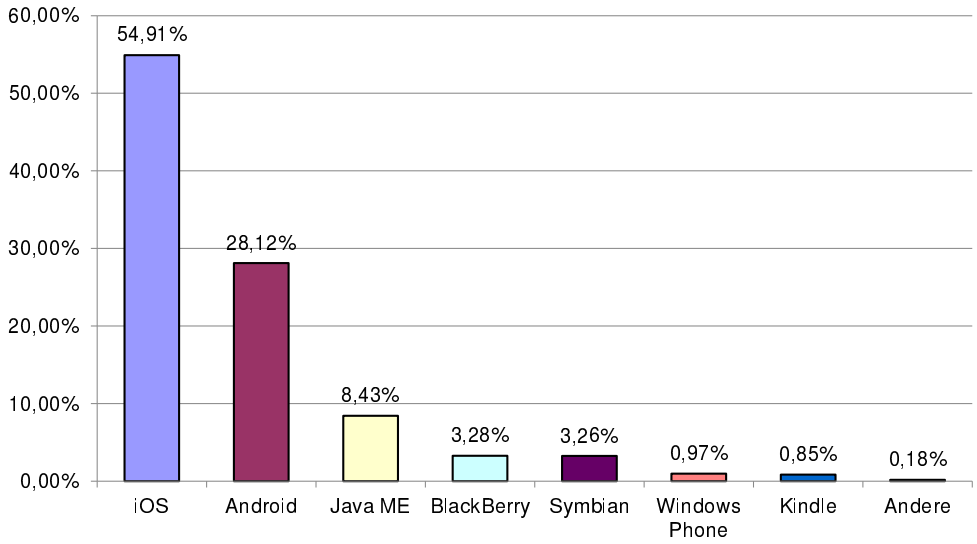


Abbildung 6.2.: Marktanteile mobiler Betriebssysteme im August 2013 (entnommen aus [85]). Die Systeme iOS und Android liegen klar vorne.

Vision Network Video schlägt für ein High-Definition-H.264-Stream mit etwa zehn Millisekunden zu Buche, während Software-Encoder wie der frei verfügbare *VideoLAN-VLC-Player* bei 350 Millisekunden liegen [41, S. 5]. Auf iOS oder Android basierende Medienabspiel-Software bietet üblicherweise die Möglichkeit, Hardware-Unterstützung für die Decodierung zu nutzen, sofern diese auf dem jeweiligen Endgerät vorhanden ist.

Der letzte Einflussfaktor für die Latenz ist der Anzeigebildschirm. Während CRT-Monitore praktisch keine Verzögerung erzeugen, sind es bei TFT- und Plasma-Anzeigen noch Reaktionszeiten von 4 bis zu 120 Millisekunden. Die enormen Latenzen entstehen häufig durch vorgeschaltete Bildoptimierungsverfahren, Deinterlacing und Skalierung.

6.2. Betriebssysteme für mobile Geräte

Trotz der Vielfalt verschiedener Smartphone- und Tablet-PC-Produzenten gibt es nur eine geringe Menge unterschiedlicher Betriebssysteme. Die Betriebssysteme *iOS* von Apple und *Android* von Google nehmen dabei mit Abstand den größten Marktanteil ein [85]. Das Schlusslicht bilden die mobilen Betriebssysteme *Blackberry*, *Symbian*, *Windows Phone* und *Kindle* (vgl. Abbildung 6.2). Alle Systeme werden üblicherweise auf vergleichsweise ressourcenschwacher Hardware betrieben. Die Leistungsparameter sind im Gegensatz zu Kleinstcomputern und PC-Systemen älterer Generationen zwar

hervorragend, die Möglichkeit zur Übertragung und Wiedergabe audiovisueller Signale ist jedoch auf bestimmte Protokolle und Anzeigemodi beschränkt. Im Folgenden werden die beiden populärsten Betriebssysteme für Smartphones und Tablet-PCs eingeführt und nativ verfügbare Streaming-Formate und Videocodierungen erläutert.

6.2.1. Apple iOS

Apple Inc. brachte 2007 erstmals das *iPhone*-Smartphone auf den Markt. Das dazugehörige Betriebssystem trägt den Namen *iPhone OS*. Es wurde 2010 in *Apple iOS* umbenannt, da es zusätzlich auf den hauseigenen Tablet-PCs und tragbaren Medienabspielgeräten installiert ist. Apple iOS basiert auf dem PC-Betriebssystem *Apple Mac OSX* und wird mit dem *Safari Mobile Web-Browser* ausgeliefert. Dieser Web-Browser verwendet die *WebKit*-Engine [5] zur Anzeige von Web-Inhalten. Hinsichtlich der Wiedergabe von Videos setzt Apple dabei ausschließlich auf HTML5-Video mit Unterstützung für das H.264-Baseline-Profile-3.0 und -3.1 [6]. Andere Verfahren wie beispielsweise das RTSP über Apple *Quicktime* oder das RTMP über Adobe *Flash* werden nicht unterstützt. Der Web-Browser bietet jedoch die Möglichkeit, Motion-JPEG unter Verwendung eines HTML-Image-Tags einzusetzen. Apple iOS ermöglicht außerdem den Einsatz der Streaming-Verfahren *HTTP Progressive Downloading* und *HTTP Live Streaming*. Ein wesentlicher Nachteil ergibt sich aus der Einschränkung, dass HTML5-Video lediglich im Vollbildmodus eingeblendet werden kann.

6.2.2. Google Android

Nur ein Jahr nach der Veröffentlichung des Apple iPhone stellt die *Open Handset Alliance* das auf einem Linux-Kernel basierende Betriebssystem Android vor. Der Quellcode wird von Google unter einer Apache-Lizenz offen und frei publiziert. Android verfügt ähnlich wie iOS über einen WebKit-basierenden Web-Browser, der im Gegensatz zur Apple-Variante mit der Google-Chrome-V8-Javascript-Engine ausgestattet ist. Sogenannte *Web-Views* erlauben die Verwendung der HTML- und Javascript-Engine aus jeder Anwendung heraus. Die integrierte Videoabspiel-Software unterstützt RTSP, HTTP Progressive Downloading und das *HTTP Live Streaming draft protocol* [35]. Im Gegensatz zu iOS ist es möglich, die Installation des Abspielwerkzeugs für Adobe Flash durchzuführen. Hiermit können Flash-Videos über RTMP und *HTTP Dynamic Streaming* abgespielt werden. Android unterstützt dabei die nachfolgenden Videoformate:

- H.263 im Container 3GPP (Dateinameerweiterung .3gp) oder MPEG-4 (Dateinameerweiterung .mp4)
- H.264 AVC Baseline Profile im Container 3GPP, MPEG-4 und MPEG-TS (Dateinameerweiterung .ts)

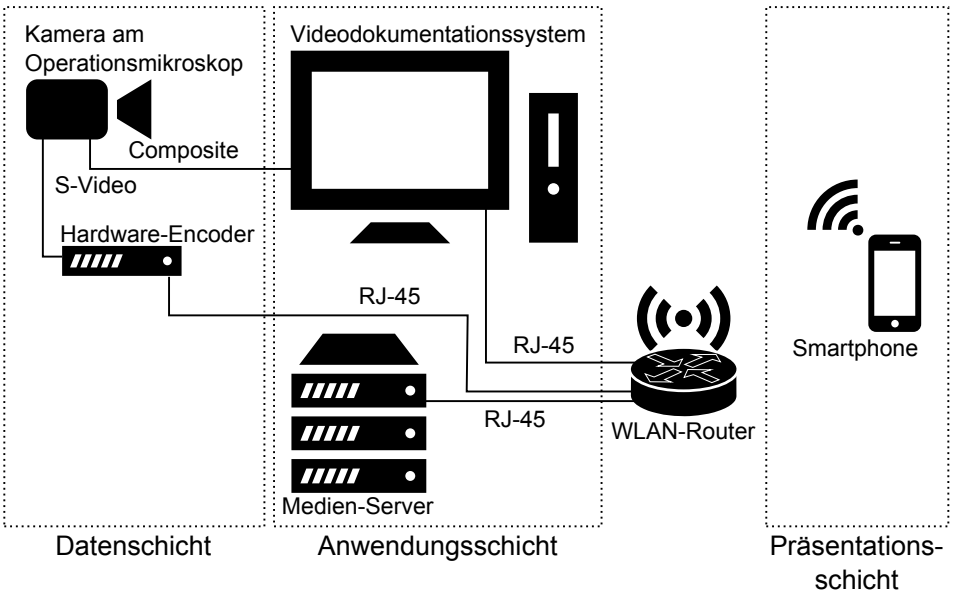


Abbildung 6.3.: Physikalischer Aufbau für den Video-Live-Streaming-Versuch. Der Hardware-Encoder, das Videodokumentationssystem sowie der Medien-Server sind über RJ-45 an einen WLAN-Router angeschlossen. Das Smartphone kommuniziert mit dem WLAN-Router über seine Funkschnittstelle. Das Videodokumentationssystem nutzt eine Framegrabber-Karte, um die Signale der Kamera mittels Composite-Kabel zu digitalisieren.

- MPEG-4 SP im Container 3GPP
- VP8 im Container WebM

6.3. Realisierung

Die Motivation für ein verzögerungsarmes Video-Live-Streaming auf Tablet-PCs und Smartphones ist die Verwendung eben jener Geräte in der Telemedizin. Ein Anwendungsbeispiel für den Einsatz des Video-Live-Streamings ist die Anzeige und Fernsteuerung eines Operationsmikroskops. Die Fernsteuerung wurde bereits in Kapitel 3 umgesetzt. In einem nächsten Schritt wird die Anzeige des Mikroskopsichtfeldes auf einem Tablet-PC bzw. Smartphone durchgeführt und zusammen mit der DPWS-Komponente die bildgestützte Fernsteuerung des Gerätes ermöglicht. Abbildung 6.3 skizziert den physikalischen Aufbau des Systems. Ein Operationsmikroskop besteht typischerweise aus einem Stativ, einem Mikroskopkopf, einer Lichtquelle, einer Kamera

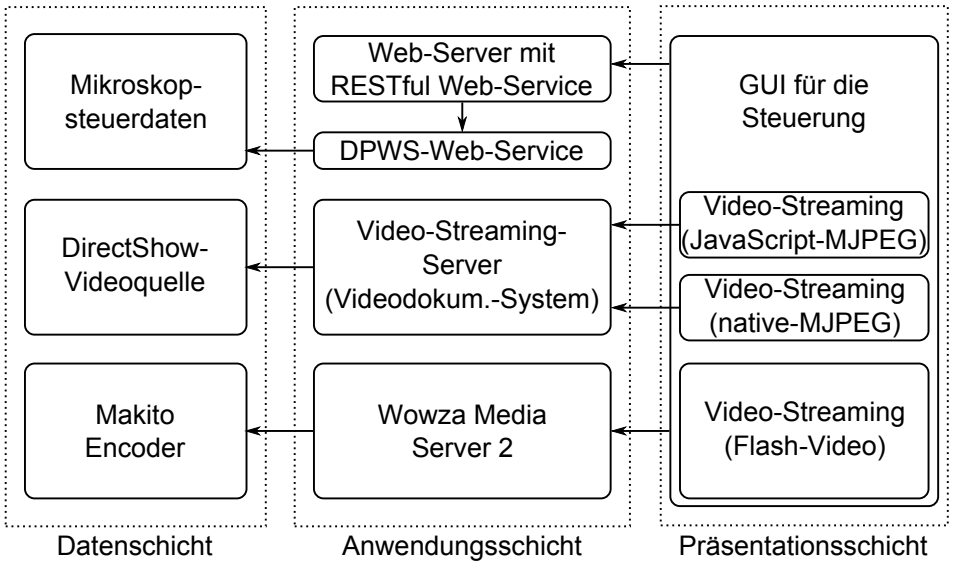


Abbildung 6.4.: Video-Live-Streaming-Architektur.

und einem Videodokumentationssystem. Die Kamera und das Videodokumentationssystem sind in Abbildung 6.3 symbolisch dargestellt. Das Kamerabild wird im Rahmen des vorgestellten Testverfahrens über eine Framegrabber-Karte im Videodokumentationssystem des Mikroskops weiterverarbeitet. Ein S-Video-Kabel schickt das Videosignal zusätzlich an einen HaiVision *Makito Encoder*. Der HaiVision Makito Encoder ist ein industrietauglicher Hardware-Encoder für H.264-komprimiertes Videomaterial. Im Zusammenhang mit einem Makito Hardware-Decoder ist laut Herstellerangaben eine Ende-zu-Ende-Latenz von 70 Millisekunden erreichbar. Der Hardware-Encoder soll die Übertragungslatenz minimieren und in einem Produktivsystem das Videodokumentationssystem bezüglich des Codierungsaufwandes entlasten.

Die physikalische Netzwerkverbindung wird durch einen WLAN-Router hergestellt. Auf einem separaten Server läuft sowohl eine protoypische Web-Applikation zur Visualisierung des Video-Live-Streams als auch ein Wowza Medien-Server [168] zur Übersetzung des Videomaterials vom Makito Encoder. Der Wowza Medien-Server wird benötigt, weil die Containerformate des Makito Encoders nicht mit den unterstützten Containerformaten der Endgeräte kompatibel sind. Damit ergibt sich der in Abbildung 6.4 illustrierte logische Systemaufbau. Links in der Abbildung befindet sich die Datenschicht. Diese setzt sich zum einen aus dem Videosignal und zum anderen aus der Gerätekonfiguration des Operationsmikroskops zusammen. In der Mitte ist die Anwendungsschicht dargestellt. Da Web-Anwendungen nicht DPWS-kompatibel sind, ist eine Übersetzung zu einem RESTful-Web-Service erforderlich

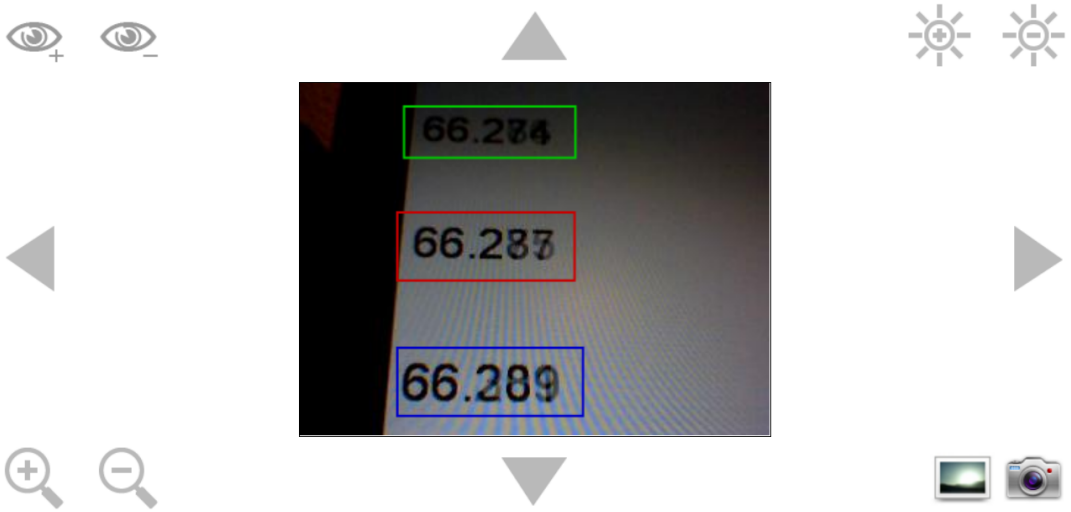


Abbildung 6.5.: Screenshot des Video-Live-Streaming-GUI mit eingeblendetem Testbild.

(in der Abbildung oben zu sehen). Für MJPEG wird das Videodokumentationssystem des Operationsmikroskops genutzt. Der Wowza Medien-Server konvertiert das TS-over-UDP/RTP-verpackte H.264-Videosignal aus dem Makito Encoder zum Flash-kompatiblen RTMP-Containerformat. Rechts im Bild ist die Präsentationsschicht abgebildet. Das GUI wird vom Web-Server zur Verfügung gestellt und umfasst drei verschiedene Anzeigemodi:

1. Video-Streaming über JavaScript-gesteuertes MJPEG
2. Video-Streaming über natives MJPEG
3. Video-Streaming über RTMP mit einem Adobe-Flash-Plugin

Die ersten beiden Varianten sollen bei geringer Auflösung mit Hilfe von Intra-Frame-Codierungen für eine verzögerungsarme Übertragung sorgen. Die Auflösung muss gering gewählt werden, da MJPEG durch die einfache Aneinanderreihung der Bilder sehr viel Bandbreite benötigt. Der Vorteil ist, dass MJPEG im Gegensatz zum RTMP-Flash-Video nicht im Vollbildmodus gestartet werden muss, sondern in einem HTML-Image-Tag zusammen mit anderen Bedienelementen darstellbar ist. Die dritte Variante nutzt die Inter-Frame-Codierung aus dem Makito Encoder, um einen Vergleich mit den Intra-Frame-Codierungen durchführen zu können.

Abbildung 6.5 zeigt das GUI für die mobilen Endgeräte. Mit den Pfeiltasten kann der Mikroskopkopf geschwenkt werden. Die Symbole in den Ecken des Bildschirms ermöglichen die Änderung von Fokus, Zoom und Lichtintensität sowie das Starten

einer Videoaufnahme und das Auslösen von Screenshots. Die genannten Steuerbefehle werden über eine RESTful-Web-Service-Schnittstelle an das Videodokumentationssystem weitergeleitet, dort in DPWS-konforme Aufrufe übersetzt und diese wiederum über eine RS-232-Verbindung an die Steuerplatine im Stativ des Operationsmikroskops übermittelt. In der Mitte befindet sich der Video-Live-Stream, der vorab wahlweise über Intra- bzw. Inter-Frame-Codierungen eingeblendet werden kann. Letztere Methode erlaubt jedoch nur die Darstellung im Vollbildmodus, so dass die simultane Steuerung und Beobachtung des Video-Streams nicht möglich ist. Auf Basis des Versuchsaufbaus können im Folgenden Bewertungen hinsichtlich der Kompatibilität zwischen verschiedenen Smartphones und Tablet-PCs sowie der Übertragungslatenz getroffen werden.

6.4. Evaluation

Das vorliegende System ist in der Lage, Video-Live-Streams von einem Mikroskop abzugreifen und Steuerbefehle zu initiieren. Zunächst wird untersucht, ob das GUI einheitlich dargestellt wird. Da für die Oberflächengestaltung ausschließlich HTML und JavaScript unter Verwendung skalierbarer Elemente eingesetzt wird, ist davon auszugehen, dass zwischen Geräten verschiedener Hersteller keine optischen Unterschiede feststellbar sind. Die Adobe-Flash-Variante kann mit Apple-Geräten nicht eingesetzt werden, da grundsätzlich keine Unterstützung für Flash vorhanden ist. Um zu zeigen, dass das GUI einheitlich aussieht, wird die Applikation auf unterschiedlichen Endgeräten getestet:

- Apple iPhone 3G (iOS)
- Motorola Defy MB 525 (Android)
- LG P990 Optimus Speed (Android)

Abbildung 6.6 zeigt die grafischen Oberflächen auf den obengenannten Geräten bei Nutzung des MJPEG-Streaming. Es sind keine wesentlichen Unterschiede erkennbar. Das Bedienungskonzept ist ebenfalls auf allen Geräten gleichermaßen umgesetzt. Störend wirken hingegen die Adressleisten der umschließenden Browser-Applikationen. Um diesem störenden Effekt entgegenzuwirken, könnte ein Produktivsystem von einer App profitieren, die eingebettete Browser unterstützt. Bekannte Vertreter solcher in Apps eingebetteter Browser sind Apache Cordova [4] und iUI [69].

Im nächsten Teil der Evaluation wird die Latenz des Live-Video-Streams untersucht. Für die Latenzmessungen stehen ebenfalls das iPhone, das Defy sowie das P990 zur Verfügung.

(a) Apple iPhone 3G

(b) Motorola Defy MB 525

(c) LG P990

Abbildung 6.6.: GUI auf verschiedenen Endgeräten.

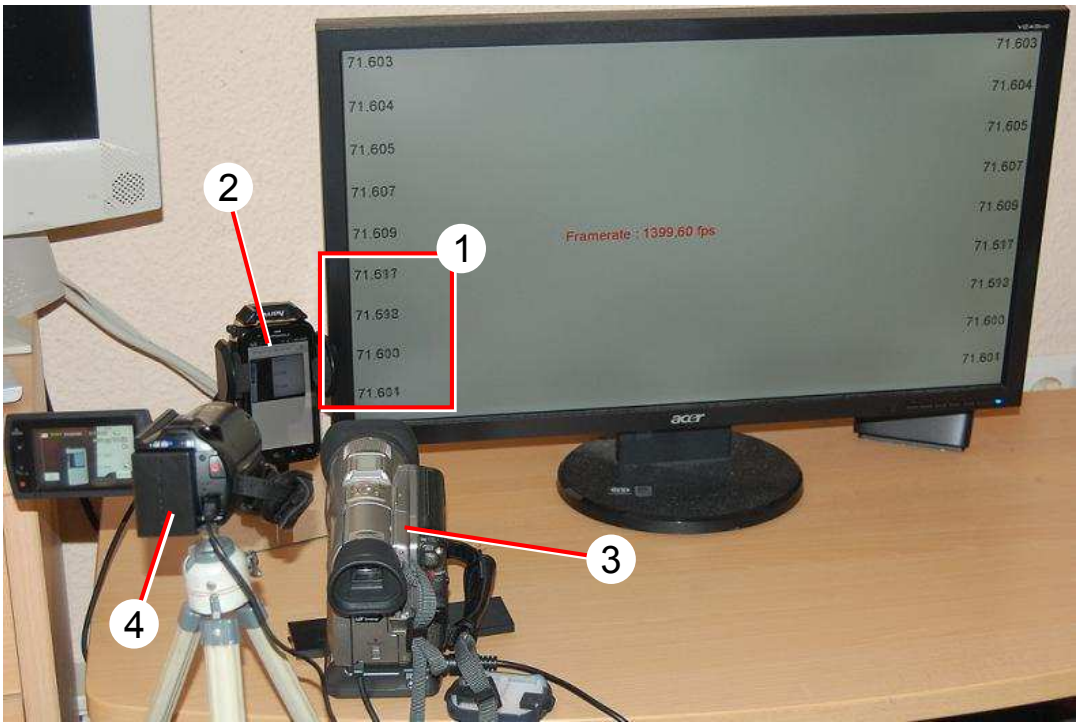


Abbildung 6.7.: Versuchsaufbau für die Latenzmessung, hier am Beispiel des RTMP-Video-Streams mit dem Motorola Defy. (1) SMTT erzeugt ein Videobild mit eingeblendeten Zeitstempeln. (2) Die Kamera der Videoquelle zeichnet den Bildschirm auf. (3) Nach dem Transport über ein Netzwerk wird das Bild auf dem Defy eingeblendet. (4) Eine weitere Kamera zeichnet sowohl den SMTT-Anzeigebildschirm als auch das Bild der Videosenke auf.

6.4.1. Messverfahren und Versuchsaufbau

Zur Messung der Latenz wird ein Verfahren zur Bestimmung des Input-Lags bei TFT-Monitoren erweitert [132]. Das speziell für den Input-Lag konstruierte Messwerkzeug *Smart Monitor Timing Tool (SMTT)* blendet präzise Timer am linken und rechten Rand eines Anzeigebildschirms ein. Zum Berechnen der Zählwerke werden über 1000 Bilder pro Sekunde generiert. Um die Bilder unmittelbar auf dem Anzeigebildschirm auszugeben, ist die vertikale Synchronisierung auf der Grafikkarte deaktiviert. Die Messung der Übertragungslatenz erfolgt über den SMTT-Timer. Abbildung 6.7 illustriert einen exemplarischen Versuchsaufbau. Ein Anzeigebildschirm wird mit eingeblendetem SMTT-Timer von der Kamera der Videoquelle aufgezeichnet. Der SMTT-Anzeigebildschirm ist neben dem Anzeigebildschirm der Videosenke platziert. Eine unabhängige Kamera zeichnet sowohl den SMTT-Anzeigebildschirm als auch

den Anzeigebildschirm der Videosenke auf. Mit einer Videoschnitt-Software kann die aufgezeichnete Videosequenz anschließend bildgenau ausgewertet werden. Als Referenzbild in der aufgezeichneten Videosequenz gilt der Wechsel zum SMTT-Testbild. Die Latenz ergibt sich aus der Subtraktion der Timer-Werte, die zum einen von der Videoquelle und zum anderen von der Videosenke abgelesen werden können:

$$t_{latenz} = t_{quelle} - t_{senke}$$

Um den Einfluss von Messungenauigkeiten zu reduzieren, werden die Werte von zehn aufeinanderfolgenden Versuchsreihen mit dem arithmetischen Mittel zusammengefasst. Für die Ermittlung der Latenz bei Inter- und Intraframe-Codierungen sind zwei Versuchsaufbauten jeweils mit einer DV-Kamera und einer Webcam konstruiert worden, da zum Zeitpunkt der Intraframe-Messaufnahmen kein Operationsmikroskop mit Videograbber-Karte zur Verfügung stand.

Versuchsaufbau für H.264

Für die Messaufnahme des H.264-Video-Streams kommt das in Abschnitt 6.3 konstruierte System zur Anwendung. Das Videosignal gelangt über eine S-Video-Schnittstelle der Kamera zum Makito Encoder. Der Makito Encoder skaliert und encodiert das Signal, und sendet es TS-over-UDP-verpackt an einen Wowza Medien-Server. Dieser Zwischenschritt ist erforderlich, da die getesteten Betriebssysteme iOS und Android keinen der Containerformate vom Makito Encoder interpretieren können und somit das H.264-codierte Video konvertiert werden muss. Der Wowza Medien-Server erzeugt aus dem TS-over-UDP-Stream einen RTMP-Stream. Das Empfangsgerät ruft daraufhin den RTMP-Stream vom Medien-Server ab. Der Makito Encoder und der Medien-Server sind über Twisted-Pair-Kabel mit einem WLAN-Router verbunden. Die Empfangsgeräte nutzen die Funkschnittstelle für den Abruf des Video-Live-Streams.

Versuchsaufbau für MJPEG

Der MJPEG-Versuchsaufbau kommt ohne einen separaten Server aus. Mit Hilfe einer Webcam wird ein Videosignal an das Videodokumentationssystem des Operationsmikroskops via USB transportiert und dort mit der Software *DirectShow Java-Wrapper (DSJ)* [44] in Form eines DirectShow-Filters in einen MJPEG-Stream (native-MJPEG) oder in Einzelbilder (JavaScript-MJPEG) konvertiert. MJPEG kann von den Empfangsgeräten direkt beim Videodokumentationssystem abgerufen werden und geht nicht den Zwischenschritt über den Makito Encoder.

Technische Spezifikationen der Testgeräte

Im Folgenden sind alle Gerätenamen und -eigenschaften aufgelistet, welche für die Versuchsreihe benutzt werden. Die verwendeten Geräte beeinflussen die Präzision der Messungen, da in allen Komponenten Verzögerungen entstehen. Diese Verzögerungen tragen zur Messungenauigkeit der einzelnen Versuche bei. Für die Darstellung des Timers wird ein unabhängiger PC verwendet:

Mainboard
Abit A-N68SV

Prozessor
AMD Athlon 64 X2 5200+

RAM
4GB DDR2

Grafikkarte
NVIDIA GeForce 9400 GT

Monitor
Acer V243HQ TFT, 1980x1080 Pixel bei 60 Hertz

Laut Herstellerangaben des SMTT-Werkzeugs wird die Genauigkeit des Timers praktisch nur durch die Rechenleistung der CPU bestimmt. Das vorliegende System liefert etwa 1400 Bilder pro Sekunde ohne vertikaler Synchronisierung und ist somit für eine millisekundengenaue Bewertung geeignet. Der Monitor gibt bei 60 Hertz alle 16,67 Millisekunden ein Bild aus. Der REST-Web-Service und der Medien-Server sowie das Videodokumentationssystem laufen auf einem dedizierten Laptop der Marke Fujitsu Siemens Amilo La1703:

Prozessor
AMD Turion 64 X2 Mobile TL-50

RAM
1GB DDR2

Für das MJPEG-Verfahren kommt eine Webcam der Marke Philips SPC530NC zum Einsatz. Die unterstützten Auflösungen der Kamera sind 640x480, 352x288, 320x240, 176x144 und 160x120 Pixel mit einer Farbtiefe von 16 Bit und einer Bildwiederholrate von 30 Bildern pro Sekunde. Das Bildmaterial für das H.264-Video-Streaming kommt aus einer Panasonic NV-DX 100 DV-Kamera mit 50 Halbbildern pro Sekunde und einer Auflösung von 720x576 Pixeln. Das Gerät zur Aufzeichnung des Quell- und Zielbildschirms ist eine Samsung HMX-H200-Full-HD-Kamera. Sie zeichnet Videos im 720p-Format mit 50 Vollbildern pro Sekunde auf. Hinsichtlich des IT-Netzwerks wird eine physikalische Sterntopologie mit einem Speedport 503W WLAN-Router

Zielgerät	OS	CPU	RAM	GPU
Apple iPhone 3G	iOS 4.2.1	Samsung 32-Bit RISC ARM11 620 MHz	128 MB	PowerVR MBX Lite 3D GPU
Motorola Defy MB525	Android 2.2.2	800 MHz TI OMAP3630-800 ARM Cortex-A8	512 MB	PowerVR SGX530
LG P990	Android 2.2.2	Nvidia Tegra 250 AP20H 1 GHz Dual Core	512 MB	Nvidia Tegra 250 AP20H

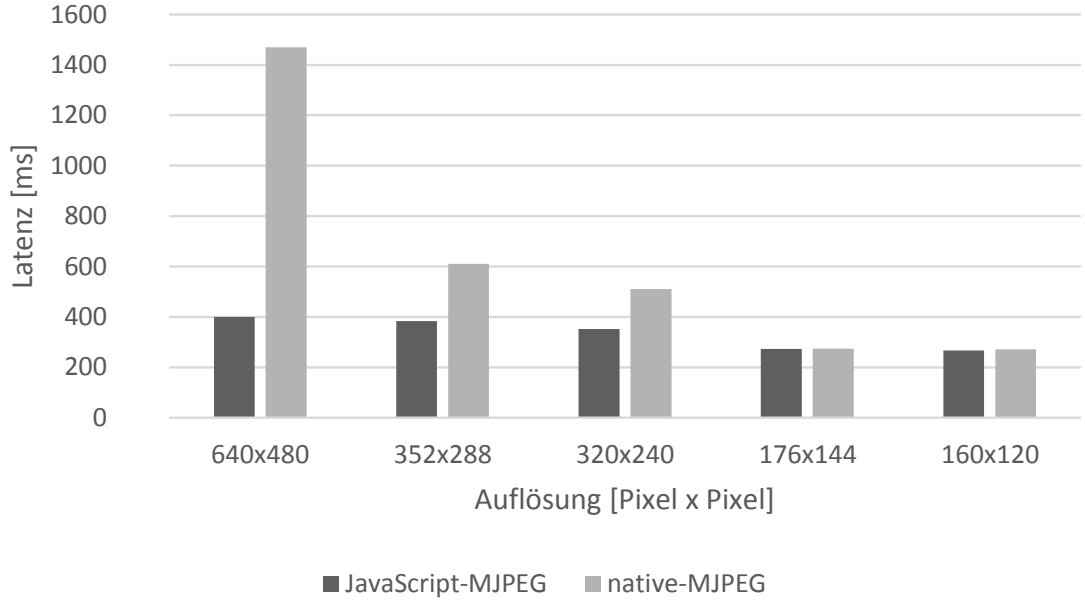
Tabelle 6.1.: Technische Details untersuchter Endgeräte für das Video-Live-Streaming. Das LG P990 bietet eine vergleichbare Rechenleistung zu Nvidia-Tegra-2-basierten Tablet-Computern (z. B. das Samsung Galaxy Tab).

installiert. Der Makito Hardware-Encoder, das Videodokumentationssystem bzw. der Medien-Server sind über 100-BASE-T-Ethernet und die Zielgeräte über ein IEEE-802.11g-kompatibles WLAN mit dem Router verbunden. Technische Einzelheiten der Zielgeräte sind in Tabelle 6.1 aufgezeichnet.

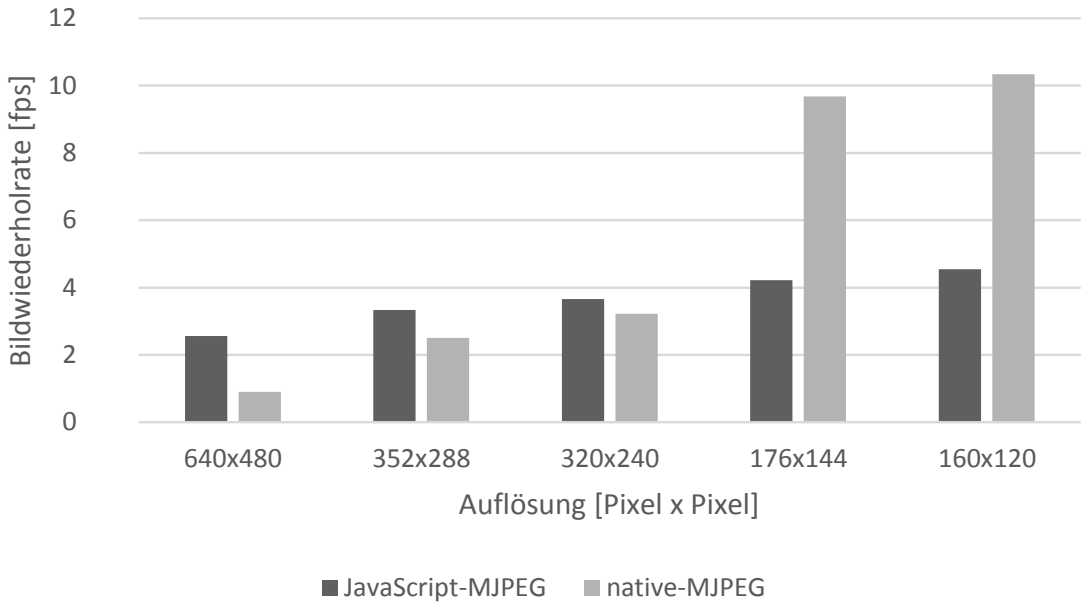
6.4.2. Auswertung

Das Auswertungsverfahren ist für alle Versuche identisch. Mit der Videoschnitt-Software *Avidemux 2.5* wird der Zeitpunkt ermittelt, an dem der Ladebalken des Browsers erlischt. Dies schließt im Fall des H.264-Streamings die Initialisierungsphase des Adobe Flash-Players ein. Anschließend wird der Zeitpunkt des ersten angezeigten Bildes im Video-Stream herausgesucht und die Differenz gebildet. Da der Übertragungsweg zwischen MJPEG und H.264 differiert, ist ein Vergleich der Verfahren nicht aussagekräftig. Stattdessen werden native-MJPEG sowie JavaScript-MJPEG auf dem iPhone 3G gegenübergestellt sowie der Einfluss der Rechenleistung auf das MJPEG-Verfahren untersucht. Die Auswertung des H.264-Streamings erfolgt ausschließlich auf dem Motorola Defy und dem LG P990, da für das iPhone kein Adobe Flash-Player zur Verfügung steht. Gemessen werden jeweils die Ende-zu-Ende-Latenzen und Bildwiederholraten bei verschiedenen Auflösungen.

Abbildung 6.8a zeigt den Einfluss der Skalierung von JavaScript- und native-MJPEG auf die Ende-zu-Ende-Latenz anhand des iPhone 3G. Analog visualisiert Abbildung 6.8b die Bildwiederholrate. Es ist zu erkennen, dass native-MJPEG für niedrige Auflösungen bei vergleichsweise ähnlicher Latenz doppelt so hohe Bildwiederholraten liefert. Dies ist darauf zurückzuführen, dass JavaScript-MJPEG für den Abruf jedes Bildes eine neue HTTP-Anfrage beim Server stellt, wohingegen native-MJPEG lediglich einen einzelnen TCP-Socket öffnet. Mit steigender Auflösung relativiert sich



(a) Messungen der Latenz für verschiedene Auflösungen



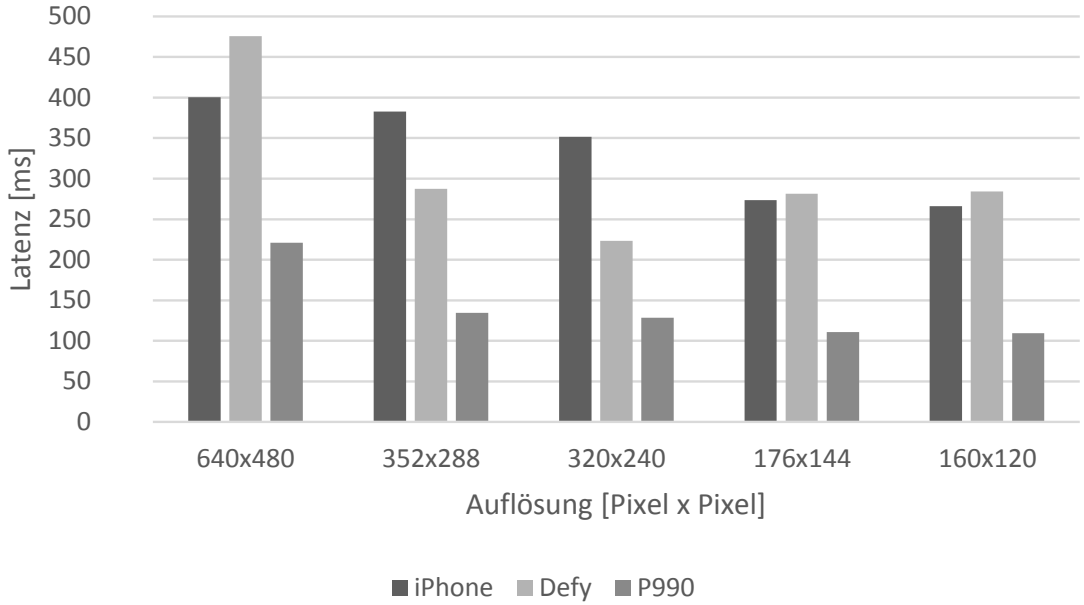
(b) Messungen der Bildwiederholrate für verschiedene Auflösungen

Abbildung 6.8.: Vergleich der Latenz- und Bildwiederholrate zwischen dem JavaScript- und native-MJPEG-Verfahren auf dem iPhone.

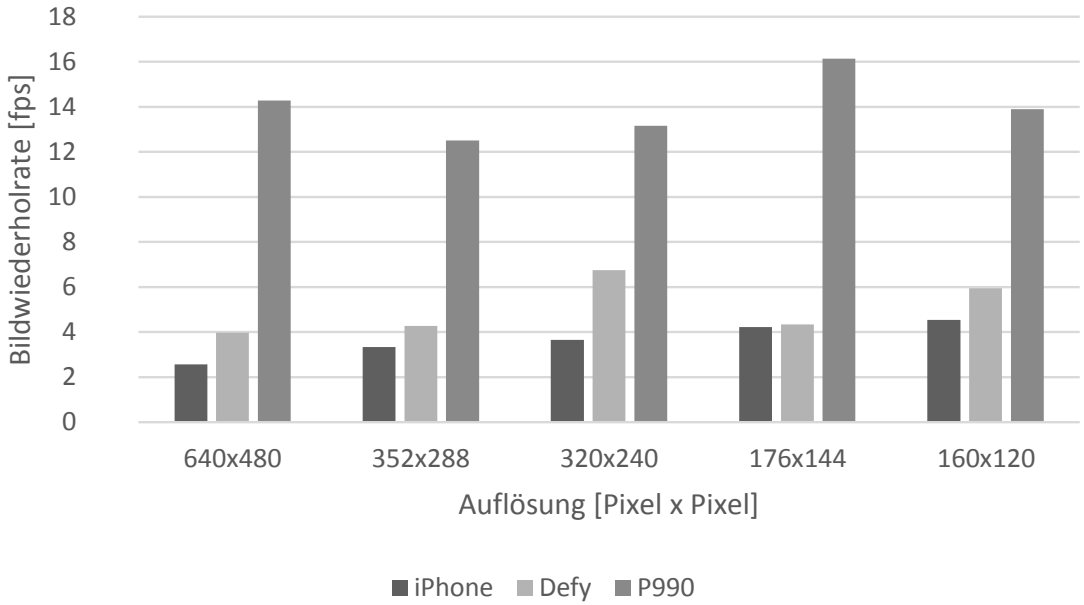
der Einfluss der kontinuierlichen HTTP-Anfragen auf die Bildwiederholrate. Erwartungsgemäß vervielfacht sich beim native-MJPEG-Verfahren die Latenz mit steigender Auflösung. Interessanterweise ist für das JavaScript-Verfahren kein derartiges Verhalten zu verzeichnen. Während sich die Latenz für native-MJPEG bei einem Anstieg der Auflösung von 320x240 auf 640x480 Pixel fast verdreifacht, steigt sie für JavaScript-MJPEG lediglich um einen Faktor von etwa $1,14$. Dabei liefert JavaScript-MJPEG in diesem Auflösungsbereich eine gute Bildwiederholrate im Vergleich zum nativen Ansatz. Eine objektive Begründung für dieses Verhalten lässt sich nicht herleiten. Vermutlich ist ein erhöhter Berechnungsaufwand im MJPEG-Encoder für die unterschiedlichen Faktoren verantwortlich. Insgesamt ist die Performance sowohl hinsichtlich der Bildwiederholrate mit einem Maximum von $10,34$ Bildern pro Sekunde als auch der Latenz mit einem Minimum von 266 Millisekunden für das Video-Live-Streaming subjektiv unzureichend, da keine flüssige und unmittelbare Wiedergabe erfolgt.

Als nächstes wird untersucht, ob durch die Änderung der Geräteleistungs-Parameter eine Verbesserung der Latenz bzw. Bildwiederholrate erzielt werden kann. Hierfür wird im Folgenden repräsentativ das JavaScript-MJPEG-Verfahren betrachtet und der Einfluss der Rechenleistung ermittelt. Zu diesem Zweck sind in Abbildung 6.9 die Latenzen und Bildwiederholraten der untersuchten Smartphones bei verschiedenen Auflösungen visualisiert. Während sich die Latenz für das 400-MHz-starke iPhone 3G und das 800-MHz-starke Motorola Defy nur marginal unterscheidet (wobei das iPhone bei einer hohen Auflösung sogar effizienter arbeitet), ergeben sich durch die Nutzung des LG P990 mit einem 1000-MHz-Dual-Core-Prozessor erhebliche Verbesserungen. Steigt die Latenz für das iPhone bei einer Auflösung von 640x480 Pixel auf 400 Millisekunden, kommt das LG mit etwa der Hälfte der Zeit aus. Zusätzlich ergeben sich auch noch Bildwiederholraten von 14 Bildern pro Sekunde für das LG im Vergleich zur Bildwiederholrate des iPhones mit etwa 2 Bildern pro Sekunde. Aus dem Diagramm lässt sich somit ablesen, dass die Erhöhung der Rechenleistung einen signifikanten Einfluss auf die Latenz und Bildwiederholrate aufweisen kann und die Endgeräte, trotz des geringeren Aufwands für die Decodierung im Vergleich zur Encodierung, einen Flaschenhals bei der Wiedergabe des Video-Live-Streams darstellen.

Zuletzt wird die Latenz und Bildwiederholrate im Zusammenhang mit dem H.264-Video-Live-Streaming bewertet. Hierbei lässt sich neben der Auflösung auch die Größe der *Group of Picture (GOP)* variieren. Unter der GOP versteht man in einer Inter-Frame-Codierung eine Gruppe von Bildern, die aufeinander aufbauen und somit in Abhängigkeit zueinander stehen. Zwischen zwei vollständig codierten Referenzbildern werden zugunsten der Kompression und somit der erforderlichen Bandbreite typischerweise Differenzbilder platziert. Die Anzahl des Abstandes zwischen zwei Referenzbildern wird als GOP-Größe bezeichnet. Die Variation der GOP-Größe erzeugt im Allgemeinen mehr Rechenaufwand bei der En- und Decodierung. Außerdem muss anfangs abgewartet werden, bis das erste Referenzbild geladen ist, womit die Latenz negativ beeinflusst wird. Abbildung 6.10a und 6.10b zeigen den Einfluss der GOP-Größe auf die Latenz bzw. Bildwiederholrate. Für diese Versuchsreihe wird das Motorola



(a) Messungen der Latenz für verschiedene Auflösungen



(b) Messungen der Bildwiederholrate für verschiedene Auflösungen

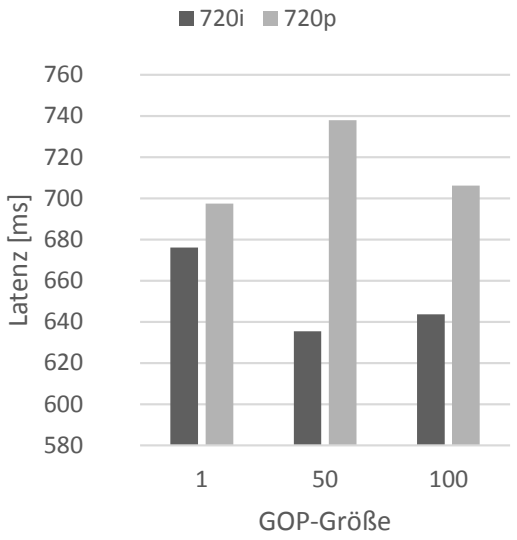
Abbildung 6.9.: Vergleich der Latenz- und Bildwiederholrate des JavaScript-MJPEG-Verfahrens mit verschiedenen Rechenleistungen. Zur Erinnerung: das iPhone verfügt über eine 400-MHz-CPU, das Defy über eine 800-MHz-CPU und das P990 über eine 2x1000-MHz-CPU.

Defy genutzt. Die Versuche werden jeweils mit Videos im 720i- bzw. 720p-Modus durchgeführt. Das *i* steht für *interlaced* und das *p* für *progressive*. Für die Ausgabe im Progressive-Modus muss der Makito Encoder ein Deinterlacing durchführen, da die DV-Kamera ausschließlich Halbbilder liefert. Dadurch verschlechtert sich die Leistung sowohl hinsichtlich der Latenz als auch der Bildwiederholrate entsprechend. Im Rahmen der durchgeführten Versuche mit einer GOP-Größe von 1, 50 und 100 ist kein signifikanter Einfluss auf die Latenz feststellbar. Der Grund dafür liegt im von der Kamera aufgezeichneten Videomaterial, welches aufgrund des übermäßig statischen Inhalts wenig Raum für Optimierung zulässt (vgl. Abbildung 6.7 auf Seite 113). Es ist jedoch die Tendenz erkennbar, dass eine geschickt gewählte GOP-Größe Vorteile hinsichtlich einer Reduzierung der Latenz und Erhöhung der Bildwiederholrate haben kann. Die beliebige Steigerung der GOP-Größe erzeugt allerdings keinen proportionalen Performance-Gewinn. Durch die Anhebung der GOP-Größe auf 50 kann die Bildwiederholrate von $6,4$ auf $12,5$ Bilder pro Sekunde im Progressive-Modus deutlich gesteigert werden. Das Verbesserungspotential ist in der vorliegenden Versuchsreihe bereits bei einem GOP-Größensprung von 50 auf 100 ausgeschöpft.

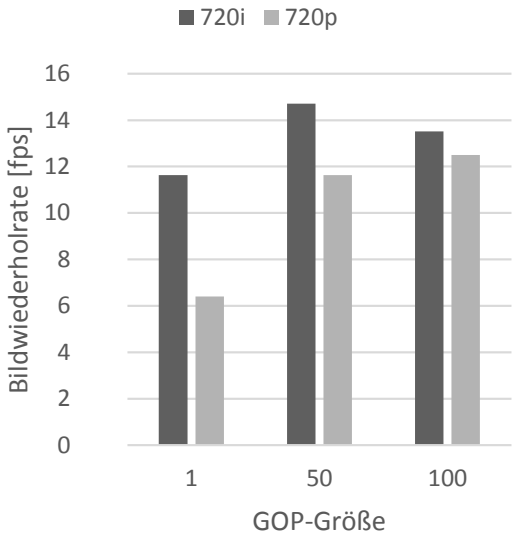
Für die Auswirkung der Rechenleistung der Endgeräte auf die Latenz und Bildwiederholrate bei der Wiedergabe des H.264-Live-Streamings werden abschließend Messungen mit drei Auflösungen im Interlaced-Modus durchgeführt (vgl. Abbildung 6.10c und 6.10d). Das Motorola Defy mit einem 800-MHz-Single-Core liegt hinsichtlich der Latenz bei 720x576 und 352x288 Pixeln knapp hinter dem LG P990 mit einem 1000-MHz-Dual-Core. Bei einer Auflösung von 540x576 Pixeln tariert sich die Latenz sogar aus. Hinsichtlich der Bildwiederholrate gibt es ein ähnliches Ergebnis. Die 720x576-Pixel-Auflösung liefert für beide Geräte etwa die gleiche Rate. Ein deutlicher Unterschied ist für die niedrig gewählte Auflösung von 352x288 Pixeln zu erkennen. Mit 25 Bildern pro Sekunde unterscheidet sich die Bildwiederholrate beim LG P990 nicht nur um mehr als 10 Bilder vom Motorola Defy, sondern ermöglicht auch das ruckelfreie Betrachten des Video-Streams.

6.4.3. Fehlerquellen

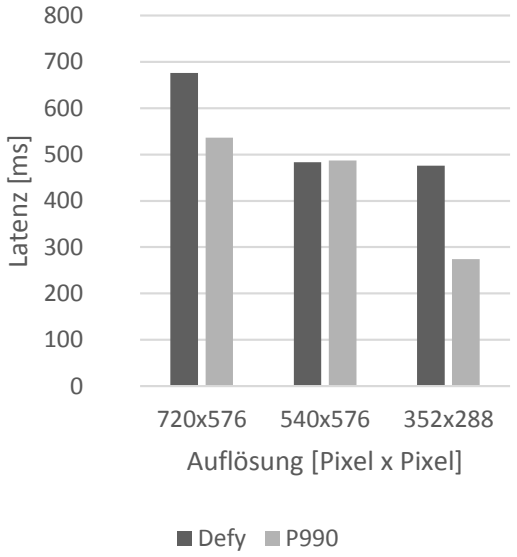
Die in den Versuchsreihen erhobenen Zeitstempel sind aufgrund der Messfehler in den Ein- und Ausgabekomponenten bezüglich einer Betrachtung im Millisekundenbereich ungenau. Die Messfehler entstehen vornehmlich in den Videoquellen und bei der Bildschirmanzeige. Sie ergeben sich aus den zeitlichen Auflösungen der einzelnen Komponenten und können mit jedem Testdurchlauf variieren. Für eine millisekundengenaue Bewertung müsste jede Komponente mindestens 1000 Bilder pro Sekunde liefern. Stattdessen befinden sich die Bildwiederholraten in einem Bereich von 30 bis 60 Bildern pro Sekunde. Tabelle 6.2 zeigt die zeitlichen Auflösungen der am Versuchsaufbau beteiligten Geräte und die Verzögerung zwischen der Verarbeitung zweier Bilder. Im schlechtesten Fall zeichnet jede Kamera kurz vor einem Bildwechsel ein veraltetes Bild



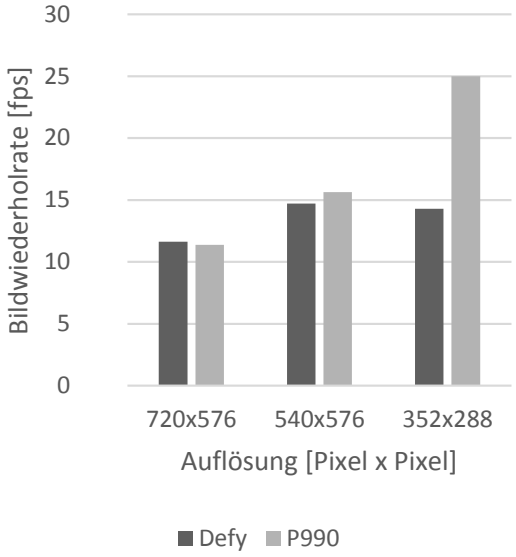
(a) Latenz in Abhängigkeit der GOP-Größe auf dem Motorola Defy.



(b) Bildwiederholrate in Abhängigkeit der GOP-Größe auf dem Motorola Defy.



(c) Latenz in Abhängigkeit der Auflösung beim Motorola Defy und LG P990.



(d) Bildwiederholrate in Abhängigkeit der Auflösung beim Motorola Defy und LG P990.

Abbildung 6.10.: Auswertung des Einflusses der GOP-Größe (vgl. Abbildungen 6.10a und 6.10b) und der Auflösung (vgl. Abbildungen 6.10c und 6.10d) auf die Latenz und Bildwiederholrate bei Nutzung des H.264-Video-Live-Streamings. Die Bezeichnungen 720i und 720p repräsentieren jeweils die Messung in einer Auflösung von 720x576 Pixeln in Halb- bzw. Vollbildern.

Komponente	Auflösung [fps]	Zeitlicher Versatz [ms]
Bildschirm	60	16,67
Webcam	30	33,33
DV-Kamera	50	20,00
Bildschirm	50	20,00

Tabelle 6.2.: Zeitliche Auflösung der Ein- und Ausgabegeräte. Der zeitliche Versatz definiert den Zeitraum, der zwischen der Verarbeitung von zwei Bildern vergeht.

auf. Die Video-Live-Streaming-Kamera erfasst das Bild mit 50 respektive 30 Bildern pro Sekunde und die Anzeige auf dem Endgerät gibt es mit 60 Bildern pro Sekunde wieder. Die Analyse-Kamera zeichnet damit ein Bild auf, das maximal so alt sein kann wie die Summe aus dem zeitlichen Versatz der Video-Live-Streaming-Kamera und dem Anzeigebildschirm des Endgerätes. Genauer gesagt gilt für den durch die zeitliche Auflösung induzierten Messfehler t_{fehler} bei dem zeitlichen Versatz einer Video-Live-Streaming-Kamera t_{kamera} und einem Wiedergabebildschirm $t_{bildschirm}$

$$0 \leq t_{fehler} < t_{kamera} + t_{bildschirm} \,.$$

Für den Versuchsaufbau mit der Webcam ergibt sich hieraus ein Fehlerwert, der unter $33,33ms + 16,67ms = 50ms$, und für den Versuchsaufbau mit der DV-Kamera ein Fehlerwert, der unter $20ms + 16,67ms = 36,67ms$ liegt.

6.5. Ergebnis

Das Video-Live-Streaming auf Smartphones und Tablet-PCs kann zukünftig einen erheblichen Einfluss auf die Telemedizin ausüben. Es stellt sich zunehmend die Frage, ob diese Geräte für ein angemessen präzises Live-Streaming geeignet sind. Beispielsweise gilt es für die Beobachtung chirurgischer Eingriffe, ruckelfreies, verzögerungsarmes und qualitativ hochwertiges Videomaterial zu liefern. Hierzu ist in diesem Kapitel ein System für die Videoübertragung und Gerätefernsteuerung zwischen einem Operationsmikroskop und mobilen Endgeräten getestet worden. Anhand einer Versuchsreihe hat sich gezeigt, welche technischen Möglichkeiten aktuelle mobile Betriebssysteme eröffnen und wo die Grenzen der vergleichsweise ressourcenschwachen Hardware liegen.

Ein Nachteil für Systeme, die neben der puren Anzeige eines Video-Streams auch grafische Bedienelemente erfordern, ist die Beschränkung der Videowiedergabe im Vollbildmodus. Hierbei handelt es sich jedoch ausschließlich um eine künstliche Beschränkung, der durch eine entsprechende Software-Umsetzung entgegengewirkt werden kann.

Für das Video-Live-Streaming sind zwei verschiedene Versuchsaufbauten konstruiert worden. In beiden Aufbauten erfolgten die Messungen der Ende-zu-Ende-Latenz sowie der Bildwiederholrate bei verschiedenen Auflösungen. Der erste Versuchsaufbau nutzt eine Webcam zur Übertragung eines MJPEG-Streams und der zweite Versuchsaufbau eine DV-Kamera und einen Hardware-Encoder zur Übertragung eines H.264-Video-Streams. Für das MJPEG-Verfahren konnten die besten Werte für das Streaming bei der kleinsten Auflösung von 160x120 Pixeln ermittelt werden, mit einer minimalen Latenz von 266 Millisekunden und einer maximalen Bildwiederholrate von $10,34$ Bildern pro Sekunde. Diese Werte sind sowohl aus Sicht einer akzeptablen Latenz- und Bildwiederholrate als auch hinsichtlich der Auflösung für einen produktiven Einsatz unzureichend.

Der zweite Versuchsaufbau zeichnet einen Video-Stream mit einer DV-Kamera auf, leitet das Signal an einen Hardware-Encoder weiter und publiziert es anschließend H.264-codiert über TS-over-UDP/RTP. Der Hardware-Encoder minimiert die Zeit für die Encodierung des Video-Materials. Da jedoch das Container-Format des Hardware-Encoders von den Endgeräten nicht unterstützt wird, ist ein Medien-Server notwendig, der das Signal umverpackt. Daraus resultiert eine weitere Latenzquelle. Zusammen mit dem Medien-Server ergibt sich für den H.264-codierten Video-Stream eine minimale Ende-zu-Ende-Latenz von etwa 274 Millisekunden und eine Bildwiederholrate von 25 Bildern pro Sekunde bei einer Auflösung von jeweils 352x288 Pixeln. Trotz des zwischengeschalteten Medien-Servers entstehen hiermit im Vergleich zur MJPEG-Lösung bessere Messwerte. Bei 25 Bildern pro Sekunde lässt sich der Video-Stream bereits ohne wahrnehmbares Ruckeln betrachten.

Insgesamt ist zu beobachten, dass die Leistung der Endgeräte signifikante Auswirkungen auf die Latenz und Bildwiederholrate hat, so dass sich nicht die Encodierung allein als limitierender Faktor beim Video-Live-Streaming auswirkt. Zukünftig sollten die mobilen Endgeräte für optimierte Live-Streaming-Verfahren mehrere Video-Codierungen und Containerformate unterstützen, so dass aus einem breiten Spektrum an Technologien zurückgegriffen werden kann. Mit der stetig steigenden Rechenleistung ist davon auszugehen, dass Smartphones und Tablet-PCs in den nächsten Jahren in der Lage sein werden, verzögerungsarmes, ruckelfreies und qualitativ angemessenes Live-Video-Streaming zu betreiben.

Kapitel 7.

Zusammenfassung und Ausblick

Medizingerätekonnektivität ohne Anwendungen erzeugt keinen Mehrwert für das klinische Personal. Ziel dieser Arbeit war die Weiterentwicklung der auf offenen Standards basierenden Web-Service-Technologie und im Speziellen DPWS. Hiermit sollen Interoperabilitätsprobleme auf der Anwendungsebene langfristig behoben und ein Mehrwert für das klinische Personal und die Patienten geschaffen werden.

7.1. Zusammenfassung

Zunächst wurden in Kapitel 1 technische und rechtliche Grundlagen erörtert und die vorliegende Arbeit motiviert. Kapitel 2 behandelte anschließend die Grundlagen der Web-Service-Technologie. Als erstes standen hierbei die drei Basistechnologien SOAP, WSDL und UDDI im Fokus. Danach wurden die für die Kommunikation zwischen Geräten maßgeblichen Spezifikationen eingeführt, welche in DPWS zusammenfließen. Zuletzt erfolgte eine Einführung in ein für medizinische Software taugliches DPWS, das auf die Bedingungen in klinischen Einrichtungen zugeschnitten ist.

Um aufzuzeigen, welches Potential in der Vernetzung von Medizingerätelandschaften steckt, sind in Kapitel 3 zahlreiche Anwendungsfälle und deren Nutzen für das klinische Personal beschrieben worden. Die vorliegenden Anwendungsfälle konnten durch die Beobachtung von OP-Abläufen und Befragung des klinischen Personals abgeleitet werden. Abschließend illustrierte eine Machbarkeitsstudie, dass die Umsetzung der Anwendungsfälle auf Basis von DPWS möglich ist.

Kapitel 4 beschrieb eine Erweiterung des in DPWS verankerten Standards WS-Eventing. Anhand einer Multicast-Bindung sind Geräte in der Lage, Ereignisnachrichten effizient in einem Netzwerk zu verteilen. Diese Fähigkeit wird für medizinische Geräte dann interessant, wenn beispielsweise kontinuierliche Signale wie Herzfrequenzkurven zu mehreren Empfängern transportiert werden müssen.

In Kapitel 5 wurde ein Anwendungsfall aus der vertikalen Integration betrachtet. Ziel war die robuste und automatisierte Verteilung von beliebigen Patientendaten an alle in einem Interventionskontext beteiligten medizinischen Geräte. Hierzu wurde

ein Protokoll entwickelt, dass mit Hilfe von Push- und Pull-Nachrichten und einem Heartbeat-System die Daten zwischen einer Quelle und mehreren Senken synchronisiert. Ein besonderes Merkmal des Protokolls ist die Bestätigung von Patientendaten an einem beliebigen Gerät, wodurch die Beschränkung auf ein KIS-Gateway aufgehoben wird.

Ein Anwendungsfall aus der Telemedizin, der auch im Rahmen eines OP-Steuer-Cockpits motiviert wird, beschreibt die Übertragung eines Video-Live-Streams aus einem OP-Saal heraus. Der Live-Stream umfasst dabei beispielsweise das Sichtfeld eines Operationsmikroskops und kann dazu genutzt werden, um entfernte Diagnosen oder Schulungen multimedial zu unterstützen. Der technische Fortschritt hat dabei die Nutzung mobiler Rechnersysteme für die Anzeige von Videomaterial ermöglicht. Im letzten Kapitel wurde untersucht, ob die Übertragung und Anzeige von Video-Live-Streams die nötige Güte erreicht, um im klinischen Umfeld effizient eingesetzt werden zu können. Mit Hilfe eines Hardware-Encoders wurde ein Live-Stream zur Verfügung gestellt und die Übertragungslatenz zu verschiedenen mobilen Endgeräten gemessen und bewertet. Durch die beschränke Rechenleistung der Geräte und die Einschränkung auf bestimmte Videocodierungen sowie Containerformate ist eine effiziente Verwertung heute noch nicht denkbar. Durch den technischen Fortschritt ist es aus Sicht des Autors jedoch nur noch eine Frage der Zeit, bis die nötige Rechenleistung verfügbar sein wird.

7.2. Ausblick

Die technische Umsetzung für Konnektivität ist gegeben. Bis heute ist jedoch keine adäquate Lösung bekannt, die den Nutzen der Vernetzung sowohl für Medizintechnikhersteller als auch Klinikbetreiber gleichermaßen evident darstellt. Die Hersteller können die Verantwortung für dynamisch betriebene Medizinprodukte nicht garantieren. Für die Betreiber bedeutet die Übernahme des Risikomanagement-Prozesses nach ISO/IEC 80001-1 einen erheblichen monetären Mehraufwand. Eine wichtige Aufgabe besteht daher darin, ein Zulassungsmodell zu entwerfen, dass die Verwaltungskosten gering hält und die Patientensicherheit maßgebend erhöht, so dass die Betreiber motiviert werden, interoperable Medizingeräte einzusetzen. Damit würden Betreiber bei der Anschaffung neuer Geräte die Interoperabilität als obligatorische Bedingung aufführen, so dass auch auf Seiten des Herstellers das Interesse für die Umsetzung gesteigert wird.

Diese Arbeit hat erste Entwicklungen für anwendungsorientierte Protokolle vorgestellt. Weiterführende Arbeiten bestehen in der Optimierung der Protokolle und Einreichung in einschlägigen Standardisierungsgremien. Darüber hinaus gibt es zahlreiche Anwendungsszenarien, für die noch kein technischer Wegweiser existiert. Nicht zuletzt sind

dann die Hersteller an der Reihe, ihre medizinische Software aufzurüsten, um die Vernetzung weiter voranzutreiben.

Anhang A.

Datenmodell des EHR-Protokolls

Das Datenmodell aus Kapitel 5 setzt sich aus einer WSDL-Beschreibung samt XML-Schema sowie einer normativen Definition des WS-Policy-Modells zusammen. Dieser Abschnitt verzeichnet beide Modelle und erläutert Aspekte, die in Kapitel 5 nicht behandelt sind.

A.1. WSDL-Dokument

In Quelltext A.1 ist das für die Verteilung der EHR-Daten korrespondierende WSDL-Dokument aufgezeigt. Die Dienstschnittstelle der WS-Eventing-Operationen ist in [150, Appendix III – WSDL] und die Dienstschnittstelle der WS-Discovery-Operationen in [105] beschrieben.

Da die Anzahl der Benachrichtigungs-Operationen für das vorgeschlagene kanalbasierte Filterverfahren von der Anzahl an EHR-Quellen abhängt, ist in der vorliegenden WSDL-Beschreibung nur die umfassende Operation *EhrConfirmationEvent* aufgezeigt (vgl. Quelltext A.1, Zeile 78). Für jede EHR-Quelle muss ein PDS eine analog parametrisierte Benachrichtigung generieren, wenn das kanalbasierte Filterverfahren angewandt werden soll.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <wsdl:definitions targetNamespace="http://applications.devices.
     med/2012/03/ehr/" xmlns:n1="http://schemas.xmlsoap.org/wsdl/
     soap12/" xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
      xmlns:tns="http://applications.devices.med/2012/03/ehr/"
     xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="
     http://www.w3.org/2001/XMLSchema">
3
4  <!-- XML-Schema -->
5  <wsdl:types>
6  <xs:schema
7   attributeFormDefault="unqualified"
8   elementFormDefault="qualified"
```

```
 9   targetNamespace="http://applications.devices.med/2012/03/ehr/"
10   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
11     <xs:complexType name="CodingSystemIdType">
12     <xs:simpleContent>
13      <xs:extension base="xs:string">
14       <xs:attribute name="version" type="xs:string"/>
15      </xs:extension>
16     </xs:simpleContent>
17    </xs:complexType>
18     <xs:complexType name="CodingSystemType">
19     <xs:sequence>
20      <xs:element name="EhrType" type="xs:anyURI"/>
21       <xs:element name="CodingSystemId" type="
            tns:CodingSystemIdType"/>
22     </xs:sequence>
23    </xs:complexType>
24
25    <xs:element name="EhrSet">
26     <xs:complexType>
27      <xs:sequence>
28       <xs:element name="CodingSystem" type="tns:CodingSystemType
            "/>
29       <xs:element name="RecordId" type="xs:anyURI"/>
30       <xs:any minOccurs="0"/>
31      </xs:sequence>
32     </xs:complexType>
33    </xs:element>
34
35    <xs:element name="EhrQuery">
36     <xs:complexType>
37      <xs:sequence>
38       <xs:element name="EhrType" type="xs:anyURI"/>
39       <xs:any minOccurs="0"/>
40      </xs:sequence>
41     </xs:complexType>
42    </xs:element>
43
44    <xs:element name="HeartbeatMetaInformation">
45     <xs:complexType>
46      <xs:sequence>
47       <xs:element name="DeviceEpr" type="xs:anyURI"/>
48       <xs:element name="Timeout" type="xs:integer"/>
49      </xs:sequence>
50     </xs:complexType>
51    </xs:element>
52   </xs:schema>
53  </wsdl:types>
54
```

```
55 <!-- WSDL-1.1-Messages -->
56 <wsdl:message name="GetEhrOperationMessage">
57  <wsdl:part element="tns:EhrQuery" name="parameters"/>
58 </wsdl:message>
59 <wsdl:message name="GetEhrOperationResponseMessage">
60  <wsdl:part element="tns:EhrSet" name="parameters"/>
61 </wsdl:message>
62 <wsdl:message name="EhrConfirmationEventMessage">
63  <wsdl:part element="tns:EhrSet" name="parameters"/>
64 </wsdl:message>
65 <wsdl:message name="HeartbeatEventMessage">
66  <wsdl:part element="tns:HeartbeatMetaInformation" name="
       parameters"/>
67 </wsdl:message>
68
69 <!-- WSDL-1.1-PortType -->
70 <wsdl:portType wse:EventSource="true" name="EhrSupplier"
      xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing">
71  <wsdl:operation name="GetEhrOperation">
72   <wsdl:input message="tns:GetEhrOperationMessage" name="
        GetEhrOperation" wsam:Action="http://applications.devices
        .med/2012/03/ehr/EhrSupplier/GetEhrOperation"/>
73   <wsdl:output message="tns:GetEhrOperationResponseMessage"
        name="GetEhrOperationResponseMessage" wsam:Action="http:
        //applications.devices.med/2012/03/ehr/EhrSupplier/
        GetEhrOperationResponse"/>
74  </wsdl:operation>
75  <wsdl:operation name="EhrConfirmationEvent">
76   <wsdl:output message="tns:EhrConfirmationEventMessage" name="
        EhrConfirmationEvent" wsam:Action="http://applications.
        devices.med/2012/03/ehr/EhrSupplier/EhrConfirmationEvent"
        />
77  </wsdl:operation>
78  <wsdl:operation name="HeartbeatEvent">
79   <wsdl:output message="tns:HeartbeatEventMessage" name="
        HeartbeatEvent" wsam:Action="http://applications.devices.
        med/2012/03/ehr/EhrSupplier/HeartbeatEvent"/>
80  </wsdl:operation>
81 </wsdl:portType>
82
83 <!-- Gueltiges Binding gemaess DPWS-Spezifikation -->
84 <wsdl:binding name="EhrSupplierBinding" type="tns:EhrSupplier">
85  <n1:binding style="document" transport="http://schemas.xmlsoap
       .org/soap/http"/>
86  <wsdl:operation name="GetEhrOperation">
87   <n1:operation soapAction="http://applications.devices.med
        /2012/03/ehr/EhrSupplier/GetEhrOperation"/>
88   <wsdl:input>
```

```
89      <n1:body use="literal"/>
90     </wsdl:input>
91     <wsdl:output>
92      <n1:body use="literal"/>
93     </wsdl:output>
94    </wsdl:operation>
95    <wsdl:operation name="EhrConfirmationEvent">
96     <n1:operation soapAction="http://applications.devices.med
          /2012/03/ehr/EhrSupplier/EhrConfirmationEvent"/>
97     <wsdl:output>
98      <n1:body use="literal"/>
99     </wsdl:output>
100   </wsdl:operation>
101   <wsdl:operation name="HeartbeatEvent">
102    <n1:operation soapAction="http://applications.devices.med
          /2012/03/ehr/EhrSupplier/HeartbeatEvent"/>
103    <wsdl:output>
104     <n1:body use="literal"/>
105    </wsdl:output>
106   </wsdl:operation>
107  </wsdl:binding>
108 </wsdl:definitions>
```

Quelltext A.1: WSDL-Beschreibung einer PDS-Instanz.

A.2. WS-Policy-Modell

Quelltext A.2 zeigt das WS-Policy-Modell für PDS-Instanzen. Das WS-Policy-Modell erlaubt die Abfrage einzelner EHR-Kanäle und stellt somit einen einfachen, kanalbasierten Filtermechanismus dar. Ein PDS bietet einen oder mehrere EHR-Kanäle an (`ehr:Channel`-Element). Jeder Kanal liefert dabei Informationen wie z. B. Patientenstammdaten, Bilder und Medikationen in bestimmten Codierungen. Zur Charakterisierung des Datenzugriffs wird ein EHR-Typ (`ehr:Type`) und ein oder mehrere synonyme Codierungssysteme (`ehr:SupportedCodingSystem`) vorgegeben. Anhand der letzten Angabe kann ein PDC entscheiden, ob er die Daten des PDS interpretieren kann oder nicht. Um die Daten eines Modells zu abonnieren, muss im WSDL-Dokument des EHR-Services für jeden `ehr:Type` eine gleichnamige Benachrichtigung definiert sein. PDCs rufen den Kanal DPWS-konform über den Filter-Dialekt

`http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/Action`

vom PDS ab. Die GetEhr-Operation (vgl. Quelltext A.1) bekommt ebenfalls den `ehr:Type` übermittelt, um einen EHR-Kanal zu referenzieren. Diese Form der Abfrage

von Patientendaten unterliegt der Einschränkung, dass keine zusätzliche Filterung mit WS-Eventing durchgeführt werden kann. Um auf diese Weise weitere Filter zu etablieren, müsste die Subscription-Anfrage erweitert werden. Diese Erweiterung ist generell realisierbar, führt jedoch dazu, dass die Konformität zu DPWS verloren geht. Das vorliegende Modell bietet somit einen Kompromiss zwischen der Kompatibilität mit DPWS und einer bedarfsgetriebenen Filterung.

```
<ehr:DataSource ...>
  <ehr:Channel>
    <ehr:Type>xsd:anyURI</ehr:Type>
    <ehr:SupportedCodingSystem>
      <ehr:CodingSystemId version="xsd:string">
        xsd:string
      </ehr:CodingSystemId> +
    </ehr:SupportedCodingSystem>
    xsd:any*
  </ehr:Channel> +
  xsd:any*
</ehr:DataSource>
```

Quelltext A.2: WS-Policy-Modell einer PDS-Instanz.

Literaturverzeichnis

[1] *Capsule Technologie*. http://www.capsuletech.de/. – zuletzt abgerufen am 19. Juli 2014

[2] Ahlbrandt, Janko ; Dehm, Johannes ; Röhrig, Rainer ; Wrede, Christian ; Imhoff, Michael: *Risikomanagement für medizinische Netzwerke in der Intensiv- und Notfallmedizin – Gemeinsames Positionspapier zur Norm IEC 80001-1*. Version: November 2012. http://www.vde.com/de/verband/pressecenter/pressemappen/documents/medica_2012/pp_risikomanagement.pdf. – zuletzt abgerufen am 19. Juli 2014

[3] Apache Software Foundation: *Apache ActiveMQ*. http://activemq.apache.org/. – zuletzt abgerufen am 19. Juli 2014

[4] Apache Software Foundation: *Apache Cordova*. http://cordova.apache.org. – zuletzt abgerufen am 19. Juli 2014

[5] Apple Inc.: *The WebKit Open Source Project*. https://www.webkit.org/. – zuletzt abgerufen am 19. Juli 2014

[6] Apple Inc.: *Safari HTML5 Audio and Video Guide*. Version: Dezember 2012. http://developer.apple.com/library/safari/documentation/AudioVideo/Conceptual/Using_HTML5_Audio_Video/Using_HTML5_Audio_Video.pdf. – zuletzt abgerufen am 19. Juli 2014

[7] Arney, David ; Goldman, Julian M. ; Whitehead, Susan F. ; Lee, Insup: Synchronizing an X-ray and Anesthesia Machine Ventilator: A Medical Device Interoperability Case Study. In: *Proceedings of the International Conference on Biomedical Electronics and Devices (BioDevices)*, 2009, S. 52–60

[8] Banavar, Guruduth ; Ch, Tushar ; Mukherjee, Bodhi ; Nagarajarao, Jay ; Strom, Robert E. ; Sturman, Daniel C.: An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In: *19th IEEE International Conference on Distributed Computing Systems*, 1999, 262–272. – zuletzt abgerufen am 19. Juli 2014

[9] Bohn, Stefan ; Franke, Stefan ; Neumuth, Thomas: Interoperability of medical devices within the operating room using service-oriented integration. In: *Abstract Book of the 34th Annual Int. Conf. of IEEE Engineering in Medicine and Biology Society (EMBC 2012)* (2012)

[10] Bundesministerium für Justiz und Verbraucherschutz: *Gesetz über Medizinprodukte (Medizinproduktegesetz - MPG).* Version: August 1994. http://www.gesetze-im-internet.de/mpg/BJNR196300994.html. – zuletzt abgerufen am 19. Juli 2014

[11] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.: *Smartphone-Absatz 2011 über der 10-Millionen-Marke.* Version: November 2010. http://www.bitkom.org/de/presse/66442_65897.aspx. – zuletzt abgerufen am 19. Juli 2014

[12] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.: *Tablet-PCs boomen.* Version: Februar 2011. http://www.bitkom.org/de/presse/8477_67058.aspx. – zuletzt abgerufen am 19. Juli 2014

[13] Cao, Fengyun ; Singh, Jaswinder P.: Efficient event routing in content-based publish/subscribe service network. In: *Proceedings of the 23rd INFOCOM*, 2004

[14] Carzaniga, Antonio: *Architectures for an Event Notification Service Scalable to Wide-area Networks*, Politecnico di Milano, PhD thesis, 1998

[15] Cassidy, Ruth: *WS-I Completes Web Services Interoperability Standards Work.* Version: November 2010. http://www.ws-i.org/docs/press/pr_101110.pdf. WS-I Pressemitteilung. – zuletzt abgerufen am 19. Juli 2014

[16] Catania, Nicolas ; Kumar, Pankaj ; Murray, Bryan ; Pourhedari, Homayoun ; Vambenepe, William ; Wurster, Klaus: *Web Services Events (WS-Events) Version 2.0.* Version: 2003. http://xml.coverpages.org/WS-Events20030721.pdf. – zuletzt abgerufen am 19. Juli 2014

[17] Cline, Kevin ; Cohen, Josh ; Davis, Doug ; Ferguson, Donald F. ; Kreger, Heather ; McCollum, Raymond ; Murray, Bryan ; Robinson, Ian ; Schlimmer, Jeffrey ; Shewchuk, John ; Tewari, Vijay ; Vambenepe, William: *Toward Converging Web Service Standards for Resources, Events, and Management.* A Joint White Paper from Hewlett Packard Corporation, IBM Corporation, Intel Corporation and Microsoft Corporation. http://msdn.microsoft.com/en-us/library/aa480724.aspx. Version: März 2006. – zuletzt abgerufen am 19. Juli 2014

[18] Committee F29.21 on Devices in the Integrated Clinical Environment: *ASTM F2761 - 09 Medical Devices and Medical Systems - Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE) - Part 1: General requirements and conceptual model.* ASTM International. http://www.astm.org/Standards/F2761.htm. Version: 2009. – zuletzt abgerufen am 19. Juli 2014

[19] Cooper, David ; Sentesson, Stefan ; Farrell, Stephan ; Boeyen, Sharon ; Housley, Russell ; Polk, Tim: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard).* Version: Mai 2008. `http://tools.ietf.org/html/rfc5280`. – zuletzt abgerufen am 19. Juli 2014

[20] Cugola, G. ; Nitto, E. D. ; Fuggetta, A.: Exploiting an Event-based Infrastructure to Develop Complex Distributed Systems. In: *Proceedings of the 20th International Conference on Software Engineering (ICSE 98)*, 1998

[21] Deutsche Gesellschaft fuer Biomedizinische Technik (DGBMT): *2. Hands-on-Intensivkurs für Ingenieure: Chirurgie in der Praxis*

[22] DICOM: *PS 3.1-2009 : Digital Imaging and Communications in Medicine (DICOM) – Part 1: Introduction and Overview.* Version: 2009. `ftp://medical.nema.org/medical/dicom/2009/09_01pu.pdf`. – zuletzt abgerufen am 19. Juli 2014

[23] Dolin, Robert H. ; Alschuler, Liora ; Boyer, Sandy ; Beebe, Calvin ; Behlen, Fred M. ; Biro, Paul V. ; Shabo, Amnon: HL7 Clinical Document Architecture, Release 2. In: *J Am Med Inform Assoc.* (2006), S. S. 30–39

[24] Duden: *Duden | App | Rechtschreibung, Bedeutung, Definition.* `http://www.duden.de/rechtschreibung/App`. – zuletzt abgerufen am 19. Juli 2014

[25] Duden: *Duden | Integration | Rechtschreibung, Bedeutung, Definition, Synonyme, Herkunft.* `http://www.duden.de/rechtschreibung/Integration`. – zuletzt abgerufen am 19. Juli 2014

[26] Duden: *Duden | Smartphone | Rechtschreibung, Bedeutung, Definition.* `http://www.duden.de/rechtschreibung/Smartphone`. – zuletzt abgerufen am 19. Juli 2014

[27] ECRI Institute: *Health Devices - Top 10 Health Technology Hazards for 2013.* Version: November 2012. `https://www.ecri.org/Documents/Secure/Health_Devices_Top_10_Hazards_2013.pdf`. – zuletzt abgerufen am 19. Juli 2014

[28] Engelen, Robert A. V. ; Gallivan, Kyle A.: The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks. In: *Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002)*, 2002, S. 128–135

[29] Erl, Thomas: *SOA Design Patterns.* 1st. Upper Saddle River, NJ, USA : Prentice Hall PTR, 2009. – ISBN 978–0136135166

[30] Fielding, Roy: *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine, PhD thesis, 2000

[31] Franke, S. ; Bohn, S. ; Burgert, O.: A communication framework for modular surgical assistance systems based on Device Profiles for Web Services (DPWS). In: *International Journal of Computer Assisted Radiology and Surgery* 6 (2011), S. S. 285

[32] Gamma, Erich ; Helm, Richard ; Johnsen, Ralph ; Vlissides, John: *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994. – ISBN 978–0201633610

[33] Gluhak, Alex ; Krco, Srdjan ; Nati, Michele ; Pfisterer, Dennis ; Mitton, Nathalie ; Razafindralambo, Tahiri: A survey on facilities for experimental internet of things research. In: *IEEE Communications Magazine* 49 (2011), S. S. 58–67

[34] Goldman, Julian M.: *Clinical Scenario #1: Patient Controlled Analgesia*. Version: August 2012. `http://www.mdpnp.org/uploads/PCA_Doc_1_of_series_Scenario_1_Medication_Infusion_Clinical_Scenario_Narrative_Description_v5.1.pdf`. – zuletzt abgerufen am 19. Juli 2014

[35] Google: *Android Supported Media Formats*. Version: Juli 2011. `http://developer.android.com/guide/appendix/media-formats.html`. – zuletzt abgerufen am 19. Juli 2014

[36] Gregorczyk, David: WS-Eventing SOAP-over-UDP Multicast Extension. In: *2012 IEEE 19th International Conference on Web Services* 0 (2011), S. S. 660–665. ISBN 978–0–7695–4463–2

[37] Gregorczyk, David ; Busshaus, Timm ; Fischer, Stefan: A Proof of Concept for Medical Device Integration using Web Services. In: *International Multi-Conference on Systems, Signals & Devices*, 2012

[38] Gregorczyk, David ; Busshaus, Timm ; Fischer, Stefan: Robust and Semi-automatic Electronic Health Record Dissemination Using the Devices Profile for Web Services. In: *The Eighth International Conference on Internet and Web Applications and Services* (2013), Juni, S. S. 38–44

[39] Gregorczyk, David ; Gabrecht, Alexander: Poster Abstract: An Approach to Remotely Control Operating Microscopes. In: *Abstractband der 45. Jahrestagung der Deutschen Gesellschaft für Biomedizinische Technik (DGBMT) im VDE*, DE GRUYTER Verlag, September 2011

[40] Haas, Peter: *Gesundheitstelematik: Grundlagen, Anwendungen, Potenziale (German Edition)*. Springer, 2006. – ISBN 978–3540207405

[41] Haivision Network Video: *Understanding End-to-End Latency for Network Video Applications*. Version: 2013. `http://www.techex.co.uk/resources/white-papers/understanding-end-to-end-ip-video-latency/download`. – zuletzt abgerufen am 19. Juli 2014

[42] HL7 Deutschland e. V.: *HL7 Deutschland e. V.* http://www.hl7.de. – zuletzt abgerufen am 19. Juli 2014

[43] Huang, Yi ; Gannon, Dennis: A Flexible and Efficient Approach to Reconcile Different Web Services-based Event Notification Specifications. In: *Specifications, International Conference on Web Services (ICWS)*, 2006, S. 735–742

[44] Humatic: *dsj - DirectShow <> Java wrapper.* http://www.humatic.de/htools/dsj.htm. – zuletzt abgerufen am 19. Juli 2014

[45] Ibach, Bastian ; Benzko, Julia ; Radermacher, Klaus: OR-Integration based on SOA – Automatic detection of new Service Providers using DPWS. In: *International Journal of Computer Assisted Radiology and Surgery* 1 (2010), S. S. 195–196

[46] IHE International, Inc.: *Integrating the Healthcare Enterprise.* http://www.ihe.net. – zuletzt abgerufen am 19. Juli 2014

[47] IHE International, Inc.: *PDQv3-Beispiel.* ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/PDQV3/02_PDQQuery1Response.xml. – zuletzt abgerufen am 19. Juli 2014

[48] IHE International, Inc.: *IHE IT Infrastructure Technical Framework Supplement – Patient Identifier Cross-Reference HL7 V3 (PIXV3) and Patient Demographic Query HL7 V3 (PDQV3).* Version: August 2004. http://www.ihe.net/Technical_Framework/upload/IHE_ITI_Patient_Demo_Query_2004_08-15.pdf. – zuletzt abgerufen am 19. Juli 2014

[49] IHE International, Inc.: *IHE IT Infrastructure Technical Framework Supplement 2004-2005 – 10 Patient Demographics Query (PDQ).* Version: August 2004. http://www.ihe.net/Technical_Framework/upload/IHE_ITI_Patient_Demo_Query_2004_08-15.pdf. – zuletzt abgerufen am 19. Juli 2014

[50] iNTERFACEWARE, Inc.: *HL7 Messages and Descriptions.* Version: 2013 September. http://www.interfaceware.com/hl7-standard/hl7-messages.html. – zuletzt abgerufen am 17.09.2013

[51] International Business Machines Corporation: *WebSphere MQ.* http://www-01.ibm.com/software/integration/wmq/. – zuletzt abgerufen am 19. Juli 2014

[52] International Health Terminology Standards Development Organisation (IHTSDO): *SNOMED CT Starter Guide.* Version: Februar 2014. http://ihtsdo.org/fileadmin/user_upload/doc/download/doc_StarterGuide_Current-en-US_INT_20140222.pdf. – zuletzt abgerufen am 19. Juli 2014

[53] International Standardization Organization (ISO): *ISO/IEC 19757-2:2008: Information technology – Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG*. 2008

[54] Internet Engineering Task Force (IETF): *RFC 1057: RPC: Remote Procedure Call Protocol Specification: Version 2*. Version: Juni 1988. `http://tools.ietf.org/html/rfc1057/`. – zuletzt abgerufen am 19. Juli 2014

[55] Internet Engineering Task Force (IETF): *Internet Growth (1981-1991)*. RFC 1296 (Informational). `http://www.ietf.org/rfc/rfc1296.txt`. Version: Januar 1992. – zuletzt abgerufen am 19. Juli 2014

[56] Internet Engineering Task Force (IETF): *Real Time Streaming Protocol (RTSP)*. Version: April 1998. `http://www.ietf.org/rfc/rfc2326.txt`. – zuletzt abgerufen am 19. Juli 2014

[57] Internet Engineering Task Force (IETF): *HTTP Over TLS. RFC 2818 (Informational)*. Version: Mai 2000. `http://tools.ietf.org/html/rfc2818`. – zuletzt abgerufen am 19. Juli 2014

[58] Internet Engineering Task Force (IETF): *Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP) (Proposed Standard)*. Version: Oktober 2003. `http://tools.ietf.org/html/rfc3605`. – zuletzt abgerufen am 19. Juli 2014

[59] Internet Engineering Task Force (IETF): *RTP: A Transport Protocol for Real-Time Applications (Internet Standard)*. Version: Juli 2003. `http://tools.ietf.org/html/rfc3550`. – zuletzt abgerufen am 19. Juli 2014

[60] Internet Engineering Task Force (IETF): *Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth (Proposed Standard)*. Version: Juli 2003. `http://tools.ietf.org/html/rfc3556`. – zuletzt abgerufen am 19. Juli 2014

[61] Internet Engineering Task Force (IETF): *RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files*. Version: 2005. `http://tools.ietf.org/html/rfc4180`. – zuletzt abgerufen am 19. Juli 2014

[62] Internet Engineering Task Force (IETF): *The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard)*. Version: August 2008. `http://tools.ietf.org/html/rfc5246`. – zuletzt abgerufen am 19. Juli 2014

[63] Internet Engineering Task Force (IETF): *HTTP Live Streaming*. Version: April 2013. `http://tools.ietf.org/html/draft-pantos-http-live-streaming-11`. – zuletzt abgerufen am 19. Juli 2014

[64] ISO/IEEE: *ISO/IEEE 11073: Health informatics — Point-of-care medical device communication*. Juni 2004

[65] ISO/IEEE: *ISO/IEEE 11073: Health informatics — Point-of-care medical device communication — Part 10201: Domain information model*. Juni 2004

[66] ISO/IEEE: *ISO/IEEE 11073: Health informatics — Personal health device communication — Part 20601: Application profile — Optimized Exchange Protocol*. Dezember 2008

[67] ITU INTERNATIONAL TELECOMMUNICATION UNION: *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 object identifier components*. Version: September 2004. `http://www.itu.int/rec/T-REC-X.667-200409-S/en`. – zuletzt abgerufen am 19. Juli 2014

[68] ITU INTERNATIONAL TELECOMMUNICATION UNION: *Information technology – Generic applications of ASN.1: Fast infoset*. Version: Juni 2005. `http://www.itu.int/rec/T-REC-X.891-200505-I`. – zuletzt abgerufen am 19. Juli 2014

[69] iUI: *iUI – web framework for smartphones & high-end devices*. `http://www.iui-js.org/`. – zuletzt abgerufen am 19. Juli 2014

[70] Johner, Christian ; Wittorf, Sven ; Hölzer-Klüpfel, Matthias: *Basiswissen Medizinische Software*. Dpunkt.Verlag GmbH, 2011. – ISBN 978–3898646888

[71] Kaiser Permanente: *Analysis of Implementing Integrated Systems*. MDPnP Booklet February 2007, Februar 2007

[72] KARL STORZ GmbH & Co. KG: *KARL STORZ OR1*. `http://www.karlstorz.com/cps/rde/xchg/SID-CF744F30-788546A4/karlstorz/hs.xsl/522.htm`. – zuletzt abgerufen am 19. Juli 2014

[73] Kosinski, Robert J.: *A Literature Review on Reaction Time*. Version: September 2012. `http://biae.clemson.edu/bpc/bp/Lab/110/reaction.htm#Type%20of%20Stimulus`. – zuletzt abgerufen am 19. Juli 2014

[74] Lesh, Kathy ; Weininger, Sandy ; Goldman, Julian M. ; Wilson, Bob ; Himes, Glenn: Medical Device Interoperability-Assessing the Environment. In: *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSSMDPnP)*. Cambridge, Massachusetts, USA : IEEE Computer Society, Juni 2007, S. 3–12

[75] Linnhoff-Popien, Claudia: *CORBA: Kommunikation und Management*. Springer, 1998. – ISBN 978–3540640134

[76] Lofsky, Ann S.: TURN YOUR ALARMS ON! In: *Newsletter – The Official Journal of the Anesthesia Patient Safety Foundation 19*, 2004, S. 43

[77] Materna GmbH: *Java Multi Edition DPWS Stack (JMEDS)*. http://ws4d.e-technik.uni-rostock.de/jmeds/. – zuletzt abgerufen am 19. Juli 2014

[78] Mauro, Christian ; Leimeister, Jan M. ; Krcmar, Helmut: SODA@Med – Ein Framework zur serviceorientierten Integration medizinischer Geräte in Krankenhausinformationssysteme. In: *10th International Conference on Wirtschaftsinformatik* (2011)

[79] Mauro, Christian ; Sunyaev, Ali ; Leimeister, Jan M. ; Krcmar, Helmut: Service-orientierte Integration medizinischer Geräte – eine State of the Art Analyse. In: *Wirtschaftsinformatik 2009 – Business Services: Konzepte, Technologien und Anwendungen* (2009), S. S. 119–128

[80] MD PnP Program: *MD PnP – Getting Connected for Patient Safety*. http://www.mdpnp.org/. – zuletzt abgerufen am 19. Juli 2014

[81] MDPnP: *MDPnP Booklet February 2007: Use Case Demonstration: X-Ray/Ventilator*. Version: Februar 2007. http://mdpnp.org/uploads/MDPnP_Booklet_February_2007_p23-29.pdf. – zuletzt abgerufen am 19. Juli 2014

[82] National Electrical Manufacturers Association (NEMA): *Digital Imaging and Communications in Medicine (DICOM) – Supplement 30: Waveform Interchange*. Version: November 1999. http://medical.nema.org/Dicom/supps/sup30_lb.pdf. – zuletzt abgerufen am 19. Juli 2014

[83] National Electrical Manufacturers Association (NEMA): *Digital Imaging and Communications in Medicine (DICOM) – Part 3: Information Object Definitions* . Version: 2011. http://medical.nema.org/Dicom/2011/11_03pu.pdf. – zuletzt abgerufen am 19. Juli 2014

[84] National Electrical Manufacturers Association (NEMA): *Digital Imaging and Communications in Medicine (DICOM) – Part 4: Service Class Specifications*. Version: 2011. http://medical.nema.org/Dicom/2011/11_04pu.pdf. – zuletzt abgerufen am 19. Juli 2014

[85] Net.Applications: *Mobile/Tablet Operating System Market Share*. Version: August 2013. http://marketshare.hitslink.com/. – zuletzt abgerufen am 19. Juli 2014

[86] Netburner Inc.: *Netburner SB70LC Datasheet*. http://www.netburner.com/support/documents/sb70-lc/496-1-14/file. – zuletzt abgerufen am 19. Juli 2014

[87] Object Management Group (OMG): *Event Service Specification*. Version: Oktober 2004. http://www.omg.org/spec/EVNT/1.2/PDF/. – zuletzt abgerufen am 19. Juli 2014

[88] Object Management Group (OMG): *Data Distribution Service for Real-time Systems, Version 1.2*. Version: Januar 2007. http://www.omg.org/spec/DDS/1.2/. – zuletzt abgerufen am 19. Juli 2014

[89] Olympus Europa SE & Co. KG: *Olympus - EndoALPHA*. Version: 19.10.2012. http://www.olympus-europa.com/endoalpha. – zuletzt abgerufen am 19. Juli 2014

[90] Oracle, Inc.: *JDK 6 Remote Method Invocation (RMI)-related APIs & Developer Guides*. Version: Dezember 2012. http://docs.oracle.com/javase/6/docs/technotes/guides/rmi/index.html. – zuletzt abgerufen am 19. Juli 2014

[91] Organization for the Advancement of Structured Information Standards (OASIS): *SAML Specifications*. http://saml.xml.org/saml-specifications. – zuletzt abgerufen am 19. Juli 2014

[92] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: UDDI Version 3.0.2*. Version: Oktober 2004. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm. – zuletzt abgerufen am 19. Juli 2014

[93] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: Reference Model for Service Oriented Architecture 1.0*. Version: Oktober 2006. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html. – zuletzt abgerufen am 19. Juli 2014

[94] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1*. Version: August 2006. http://docs.oasis-open.org/wsdm/wsdm-mows-1.1-spec-os-01.pdf. – zuletzt abgerufen am 19. Juli 2014

[95] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: Web Services Distributed Management: Management Using Web Services (MUWS 1.1) Part 1*. Version: August 2006. http://docs.oasis-open.org/wsdm/wsdm-muws1-1.1-spec-os-01.pdf. – zuletzt abgerufen am 19. Juli 2014

[96] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)*. Version: Februar 2006. https://www.oasis-open.org/committees/download.php/16790/

wss-v1.1-spec-os-SOAPMessageSecurity.pdf. – zuletzt abgerufen am 19. Juli 2014

[97] Organization for the Advancement of Structured Information Standards (OASIS): *WS-Notification*. Version: Oktober 2006. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn. – zuletzt abgerufen am 19. Juli 2014

[98] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.1*. Version: April 2007. http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-os/wstx-wsat-1.1-spec-os.html. – zuletzt abgerufen am 19. Juli 2014

[99] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: Web Services Business Process Execution Language Version 2.0*. Version: April 2007. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf. – zuletzt abgerufen am 19. Juli 2014

[100] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: Devices Profile for Web Services Version 1.1*. Version: Juli 2009. http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html. – zuletzt abgerufen am 19. Juli 2014

[101] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: SOAP-over-UDP Version 1.1*. Version: Juli 2009. http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.html. – zuletzt abgerufen am 19. Juli 2014

[102] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2*. Version: Februar 2009. http://docs.oasis-open.org/ws-rx/wsrm/v1.2/wsrm.html. – zuletzt abgerufen am 19. Juli 2014

[103] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: WS-SecureConversation 1.4*. Version: Februar 2009. http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.pdf. – zuletzt abgerufen am 19. Juli 2014

[104] Organization for the Advancement of Structured Information Standards (OASIS): *OASIS Standard: WS-Trust 1.4*. Version: Februar 2009. http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.pdf. – zuletzt abgerufen am 19. Juli 2014

[105] Organization for the Advancement of Structured Information Standards (OASIS): *Web Services Dynamic Discovery (WS-Discovery)*. Version: Juli 2009. http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01. – zuletzt abgerufen am 19. Juli 2014

[106] Organization for the Advancement of Structured Information Standards (OASIS): *Web Services Dynamic Discovery (WS-Discovery) Version 1.1*. Version: 2009. http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html. – zuletzt abgerufen am 19. Juli 2014

[107] Organization for the Advancement of Structured Information Standards (OASIS): *Working Draft: Devices Profile for Web Services Version 1.2*. Version: November 2009. https://www.oasis-open.org/committees/document.php?document_id=35276. Oasis Open. – zuletzt abgerufen am 19. Juli 2014

[108] Ostermann, Jörn ; Bormans, Jan ; List, Peter ; Marpe, Detlev ; Narroschke, Matthias ; Pereira, Fernando ; Stockhammer, Thomas ; Wedi, Thomas: Video coding with H.264/AVC: Tools, Performance, and Complexity. In: *IEEE CIRCUITS AND SYSTEMS MAGAZINE* (2004), Q1

[109] Ozer, Jan: *Streaming Vs. Progressive Download Vs. Adaptive Streaming*. Version: Mai 2011. http://www.onlinevideo.net/2011/05/streaming-vs-progressive-download-vs-adaptive-streaming/. – zuletzt abgerufen am 19. Juli 2014

[110] Parmar, H. ; Thornburgh, M.: *Adobe's Real Time Messaging Protocol*. Version: Dezember 2012. http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/rtmp/pdf/rtmp_specification_1.0.pdf. Adobe Systems Incorporated. – zuletzt abgerufen am 19. Juli 2014

[111] Pöhlsen, Stephan: *Entwicklung einer Service-orientierten Architektur zur vernetzten Kommunikation zwischen medizinischen Geräten, Systemen und Applikationen*, Universität zu Lübeck, Dissertation, 2010

[112] Pöhlsen, Stephan ; Buschmann, Carsten ; Werner, Christian: Integrating a Decentralized Web Service Discovery System into the Internet Infrastructure. In: *Proceedings of the 6th IEEE European Conference on Web Services (ECOWS08)*. Dublin, Ireland, November 2008

[113] Pöhlsen, Stephan ; Franz, Frank ; Kück, Kai ; Meyer, Jörg-Uwe ; Werner, Christian: Fair-Queued Ethernet for Medical Applications. In: *Proceedings of the 7th IEEE International Symposium on Network Computing and Applications (IEEE NCA08)*. Cambridge, MA, USA, Juli 2008

[114] Pöhlsen, Stephan ; Schöch, Winfried ; Schlichting, Stefan: A Protocol for Dual Channel Transmission in Service-Oriented Medical Device Architectures based on Web Services. In: *3rd Joint Workshop on High Confidence Medical Devices, Software, and Systems & Medical Device Plug-and-Play Interoperability*, 2011. – zuletzt abgerufen am 19. Juli 2014

[115] Pöhlsen, Stephan ; Schlichting, Stefan ; Werner, Christian: A Comparison of DDS and Web Services for PoC Inter-Device Connectivity. In: *4th International Conference on Pervasive Computing Technologies for Healthcare*, 2010

[116] Pöhlsen, Stephan ; Schlichting, Stefan ; Werner, Christian: *Praktische Umsetzung einer sicheren Zugriffskontrolle für Web-Services am Beispiel medizinischer Geräte*. PIK – Praxis der Informationsverarbeitung und Kommunikation 33, 2010

[117] Reichert, Manfred ; Scholl, Dietmar: Komposition, Choreograhpie und Orchestrierung von Web Services – Ein Überblick. In: *EMISA Forum* Band 24, Heft 2 (2004), S. S. 21–32

[118] SAP News Desk: *Microsoft, IBM, SAP To Discontinue UDDI Web Services Registry Effort*. Version: Dezember 2005. `http://soa.sys-con.com/node/164624`. SOA World Magazine. – zuletzt abgerufen am 19. Juli 2014

[119] Savetz, Kevin ; Randall, Neil ; Lepage, Yves: *MBONE: Multicasting Tomorrow's Internet*. John Wiley & Sons Inc, 1996. – ISBN 978–1568847238

[120] Schöch, Winfried: *Einsatz von DICOM-SR in der Pathologie*, Universitat zu Lübeck, Diplomarbeit, 2008

[121] Schill, Alexander ; Springer, Thomas: *Verteilte Systeme*. Springer, 2007. – ISBN 978–3540205685

[122] Schlimmer, Jeffrey ; Thelin, Jorgen: *Devices Profile for Web Services*. Version: Februar 2006. `http://specs.xmlsoap.org/ws/2006/02/devprof/devicesprofile.pdf`. Microsoft Corporation. – zuletzt abgerufen am 19. Juli 2014

[123] Schmitt, L. ; Falck, T. ; Wartena, F. ; Simons, D.: Novel ISO/IEEE 11073 Standards for Personal Telehealth Systems Interoperability. In: *Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability*, 2007, S. 146–148

[124] Schulte, W. Roy ; Natis, Yefim V.: *"Service Oriented" Architectures, Part 1*. Gartner, Inc., April 1996. – Research Note SPA-401-068

[125] Sietmann, Richard: Der fünfte Weg – Die Zukunft von IPTV, Internet-TV und die Netzneutralität. In: *Heise Zeitschriften Verlag* 3 (2011), S. S. 72–77

[126] Singh, Jayant: *HL7 Identifiers (Person, Patient, Account, Visit)*. Version: Februar 2013. http://www.j4jayant.com/articles/hl7/15-hl7-identifiers. – zuletzt abgerufen am 19. Juli 2014

[127] Skjervold, Espen ; Hafsøe, Trude ; Johnsen, Frank T. ; Lund, Ketil: Enabling Publish/Subscribe with COTS Web Services across Heterogeneous Networks. In: *IEEE International Conference on Web Services*. Miami, FL, USA, 2010, 660–668. – zuletzt abgerufen am 19. Juli 2014

[128] Statistisches Bundesamt: *Pressemitteilung Nr. 392: Stationäre Krankenhauskosten 2012 auf 4060 Euro je Behandlungsfall gestiegen*. Version: November 2011. https://www.destatis.de/DE/PresseService/Presse/Pressemitteilungen/2013/11/PD13_392_231.html. – zuletzt abgerufen am 19. Juli 2014

[129] Tanenbaum, Andrew ; van Steen, Marten: *Verteilte Systeme: Grundlagen und Paradigmen*. Pearson Studium, 2003. – ISBN 978–3827370570

[130] The MITRE Corporation: *Electronic Health Records Overview*. Version: April 2006. http://www.himss.org/files/HIMSSorg/content/files/Code%20180%20MITRE%20Key%20Components%20of%20an%20EHR.pdf. – zuletzt abgerufen am 19. Juli 2014

[131] The WebM Project: *WebM: an open web media project*. http://www.webmproject.org/. – zuletzt abgerufen am 19. Juli 2014

[132] Thiemann, Thomas: *Eine Untersuchung des bisherigen Testverfahrens der als Input Lag bekannten Reaktionszeit eines TFT-Monitors*. Version: August 2009. http://www.prad.de/new/monitore/specials/inputlag/inputlag.html. – zuletzt abgerufen am 19. Juli 2014

[133] TIOBE Software BV: *TIOBE Programming Community Index for March 2013*. Version: März 2013. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. – zuletzt abgerufen am 19. Juli 2014

[134] University of Rostock: *Web Services for Devices*. http://www.ws4d.org. – zuletzt abgerufen am 19. Juli 2014

[135] Web Services Interoperability Organization (WS-I): *Basic Profile Version 1.0*. Version: April 2004. http://www.ws-i.org/Profiles/BasicProfile-1.0.html. WS-I. – zuletzt abgerufen am 19. Juli 2014

[136] Web Services Interoperability Organization (WS-I): *Basic Profile Version 1.1*. Version: April 2006. http://www.ws-i.org/Profiles/BasicProfile-1.1.html. WS-I. – zuletzt abgerufen am 19. Juli 2014

[137] Web Services Interoperability Organization (WS-I): *Basic Security Profile Version 1.0*. Version: März 2007. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html. WS-I. – zuletzt abgerufen am 19. Juli 2014

[138] Web Services Interoperability Organization (WS-I): *Basic Profile Version 1.2*. Version: November 2010. http://ws-i.org/Profiles/BasicProfile-1.2-2010-11-09.html. WS-I. – zuletzt abgerufen am 19. Juli 2014

[139] Web Services Interoperability Organization (WS-I): *Basic Profile Version 2.0*. Version: November 2010. http://ws-i.org/Profiles/BasicProfile-2.0-2010-11-09.html. WS-I. – zuletzt abgerufen am 19. Juli 2014

[140] Web Services Interoperability Organization (WS-I): *Basic Security Profile Version 1.1*. Version: Januar 2010. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1.html. WS-I. – zuletzt abgerufen am 19. Juli 2014

[141] Werner, Christian: *Optimierte Protokolle für Web Services mit begrenzten Datenraten*, Universität zu Lübeck, Dissertation, 2006

[142] Wikipedia: *Webservice*. http://de.wikipedia.org/w/index.php?title=Webservice&oldid=105444218. – zuletzt abgerufen am 11. Juli 2012 um 12:51 Uhr

[143] World Wide Web Consortium (W3C): *Web Services Resource Access Working Group*. http://www.w3.org/2002/ws/ra/. – zuletzt abgerufen am 19. Juli 2014

[144] World Wide Web Consortium (W3C): *Recommendation: Extensible Markup Language (XML) 1.0*. Version: Februar 1998. http://www.w3.org/TR/1998/REC-xml-19980210. – zuletzt abgerufen am 19. Juli 2014

[145] World Wide Web Consortium (W3C): *Note: Web Services Description Language (WSDL) 1.1*. Version: März 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315. – zuletzt abgerufen am 19. Juli 2014

[146] World Wide Web Consortium (W3C): *Recommendation: XML Encryption Syntax and Processing*. Version: Dezember 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210. – zuletzt abgerufen am 19. Juli 2014

[147] World Wide Web Consortium (W3C): *Recommendation: XML Schema Part 0: Primer Second Edition*. Version: Oktober 2004. http://www.w3.org/TR/2004/REC-xmlschema-0-20041028. – zuletzt abgerufen am 19. Juli 2014

[148] World Wide Web Consortium (W3C): *Working Group Note: Web Services Architecture*. Version: Februar 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211. – zuletzt abgerufen am 19. Juli 2014

[149] World Wide Web Consortium (W3C): *Recommendation: Web Services Addressing 1.0 - Core*. Version: Mai 2006. http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/. – zuletzt abgerufen am 19. Juli 2014

[150] World Wide Web Consortium (W3C): *W3C Member Submission: Web Services Eventing (WS-Eventing)*. Version: Juli 2006. http://www.w3.org/Submission/2006/SUBM-WS-Eventing-20060315/. – zuletzt abgerufen am 19. Juli 2014

[151] World Wide Web Consortium (W3C): *Recommendation: Semantic Annotations for WSDL and XML Schema*. Version: August 2007. http://www.w3.org/TR/2007/REC-sawsdl-20070828/. – zuletzt abgerufen am 19. Juli 2014

[152] World Wide Web Consortium (W3C): *Recommendation: SOAP Version 1.2 Part 0: Primer (Second Edition)*. Version: April 2007. http://www.w3.org/TR/2007/REC-soap12-part0-20070427. – zuletzt abgerufen am 19. Juli 2014

[153] World Wide Web Consortium (W3C): *Recommendation: SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. Version: April 2007. http://www.w3.org/TR/2007/REC-soap12-part1-20070427. – zuletzt abgerufen am 19. Juli 2014

[154] World Wide Web Consortium (W3C): *Recommendation: SOAP Version 1.2 Part 2: Adjuncts (Second Edition)*. Version: April 2007. http://www.w3.org/TR/2007/REC-soap12-part2-20070427. – zuletzt abgerufen am 19. Juli 2014

[155] World Wide Web Consortium (W3C): *Recommendation: SOAP Version 1.2 Specification Assertions and Test Collection (Second Edition)*. Version: April 2007. http://www.w3.org/TR/2007/REC-soap12-testcollection-20070427. – zuletzt abgerufen am 19. Juli 2014

[156] World Wide Web Consortium (W3C): *Recommendation: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. Version: Juni 2007. http://www.w3.org/TR/2007/REC-wsdl20-20070626/. – zuletzt abgerufen am 19. Juli 2014

[157] World Wide Web Consortium (W3C): *Recommendation: Web Services Policy 1.5 - Framework*. Version: September 2007. http://www.w3.org/TR/2007/REC-ws-policy-20070904/. – zuletzt abgerufen am 19. Juli 2014

[158] World Wide Web Consortium (W3C): *Recommendation: Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Version: November 2008. http://www.w3.org/TR/2008/REC-xml-20081126/#dt-doctype. – zuletzt abgerufen am 19. Juli 2014

[159] World Wide Web Consortium (W3C): *Recommendation: XML Signature Syntax and Processing (Second Edition)*. Version: Juni 2008. http://www.w3.org/TR/2008/REC-xmldsig-core-20080610. – zuletzt abgerufen am 19. Juli 2014

[160] World Wide Web Consortium (W3C): *W3C Member Submission: Web Services Metadata Exchange 1.1 (WS-MetadataExchange)*. Version: August 2008. http://www.w3.org/Submission/2008/SUBM-WS-MetadataExchange-20080813/. – zuletzt abgerufen am 19. Juli 2014

[161] World Wide Web Consortium (W3C): *Recommendation: Namespaces in XML 1.0 (Third Edition)*. Version: Dezember 2009. http://www.w3.org/TR/2009/REC-xml-names-20091208. – zuletzt abgerufen am 19. Juli 2014

[162] World Wide Web Consortium (W3C): *XML Path Language (XPath) 2.0 (Second Edition)*. Version: Dezember 2010. http://www.w3.org/TR/2010/REC-xpath20-20101214/. – zuletzt abgerufen am 19. Juli 2014

[163] World Wide Web Consortium (W3C): *Efficient XML Interchange (EXI) Format 1.0*. Version: März 2011. http://www.w3.org/TR/2011/REC-exi-20110310/. – zuletzt abgerufen am 19. Juli 2014

[164] World Wide Web Consortium (W3C): *Recommendation: Web Services Event Descriptions (WS-EventDescriptions)*. Version: Dezember 2011. http://www.w3.org/TR/2011/REC-ws-event-descriptions-20111213/. – zuletzt abgerufen am 19. Juli 2014

[165] World Wide Web Consortium (W3C): *Recommendation: Web Services Metadata Exchange (WS-MetadataExchange)*. Version: Dezember 2011. http://www.w3.org/TR/2011/REC-ws-metadata-exchange-20111213/. – zuletzt abgerufen am 19. Juli 2014

[166] World Wide Web Consortium (W3C): *Recommendation: Web Services Transfer (WS-Transfer)*. Version: Dezember 2011. http://www.w3.org/TR/2011/REC-ws-transfer-20111213/. – zuletzt abgerufen am 19. Juli 2014

[167] World Wide Web Consortium (W3C): *W3C Recommendation: Web Services Eventing (WS-Eventing)*. Version: Dezember 2011. http://www.w3.org/TR/ws-eventing/. – zuletzt abgerufen am 19. Juli 2014

[168] Wowza Media Systems: *Wowza Media Server*. http://www.wowza.com/media-server. – zuletzt abgerufen am 19. Juli 2014

[169] Wörn, Heinz ; Brinkschulte, Uwe: *Echtzeitsysteme*. Springer, 2005. – S. 321–322. – ISBN 978–3540205883

[170] Zeeb, E. ; Moritz, G. ; Timmermann, D. ; Golatowski, F.: WS4D: Toolkits for Networked Embedded Systems Based on the Devices Profile for Web Services. In: *39th International Conference on Parallel Processing Workshops (ICPPW)*, 2010, 1-8. – zuletzt abgerufen am 19. Juli 2014