Universität Rostock

Traditio et Innovatio

Zuverlässige und herstellerübergreifende Medizingeräteinteroperabilität

—

Beiträge zur IEEE 11073 SDC-Normenfamilie

Der Fakultät für Informatik und Elektrotechnik der Universität Rostock
zur Erlangung des akademischen Grades eines

Doktor-Ingenieur (Dr.-Ing.)

vorgelegte Dissertation von
Martin Kasparick
geb. am 10.02.1986 in Wismar

Rostock, 2021

Gutachter:

Prof. Dr.-Ing. Dirk Timmermann
Institut für Angewandte Mikroelektronik und Datentechnik
Universität Rostock

Prof. Dr. rer. nat. Stefan Fischer
Institut für Telematik
Universität zu Lübeck

Dissertation
Fakultät für Informatik und Elektrotechnik

Martin Kasparick
Institut für Angewandte Mikroelektronik und Datentechnik

https://doi.org/10.18453/rosdok_id00003032

Gutachter:

- Prof. Dr.-Ing. Dirk Timmermann
  Universität Rostock
  Fakultät für Informatik und Elektrotechnik
  Institut für Angewandte Mikroelektronik und Datentechnik
  Richard-Wagner-Str. 31, 18119 Rostock-Warnemünde

- Prof. Dr. rer. nat. Stefan Fischer
  Universität zu Lübeck
  Institut für Telematik
  Ratzeburger Allee 160, 23562 Lübeck

Tag der Einreichung: 28.10.2020
Tag der Verteidigung: 11.03.2021

Danksagung

In der Rückschau auf meine bisherige wissenschaftliche und berufliche Laufbahn möchte ich die Gelegenheit ergreifen, mich bei den vielen Personen zu bedanken, die mich begleitet und in meiner Entwicklung unterstützt haben.

Mein großer Dank gilt meinem Doktorvater Prof. Dr.-Ing. Dirk Timmermann und meinem langjährigen Arbeitsgruppenleiter Dr.-Ing. Frank Golatowski am Institut für Angewandte Mikroelektronik und Datentechnik (IMD). Ohne die Betreuung und die Anstellung mit langfristigen Perspektiven und Sicherheiten wäre diese Arbeit nicht möglich gewesen. Insbesondere das in mich und meine Arbeit gesetzte Vertrauen und die Freiheiten, mit denen ich die wissenschaftlichen Fragestellungen bearbeiten und mich in der Normierung engagieren konnte, haben maßgeblich zum Gelingen dieser Dissertation beigetragen. In Forschungsprojekten wurde mir die Möglichkeit gegeben mich nicht nur fachlich, sondern auch persönlich weiterzuentwickeln und zudem eigene inhaltliche Akzente zu setzen. Ebenso möchte ich mich bei Prof. Dr. rer. nat. Stefan Fischer für die externe Begutachtung der Dissertation bedanken. Ein besonderer Dank gilt allen, die die Dissertation ganz oder in Teilen gelesen und kommentiert haben und mit ihrem konstruktiven Feedback zur Verbesserung beigetragen haben.

Des Weiteren möchte ich den vielen wissenschaftlich wie nicht-wissenschaftlich tätigen Kolleginnen und Kollegen am IMD danken. Der Zusammenhalt und die Hilfsbereitschaft sorgten für ein außergewöhnlich gutes Arbeitsumfeld. Die Flurgespräche und fachlichen Diskussionen, Teamarbeiten an Projekten und Veröffentlichungen, gemeinsam betreute studentische Arbeiten, aber auch gemeinschaftliche Mittagspausen am Strand trugen zu einer tollen Atmosphäre bei.

Diese Arbeit ist in verschiedenen Verbundprojekten entstanden. Die Projekte haben nicht nur Medizingeräte vernetzt, sondern auch Menschen. Aus Kollegen sind Freunde geworden. Die vielen Dienstreisen quer durch die Republik mit teils tagesfüllenden Reisezeiten waren so meist mehr eine Freude als eine Belastung. Viele Freunde, insbesondere aus Aachen, Leipzig und Lübeck, aber auch darüber hinaus, haben mich begleitet und fühlen sich zu Recht durch diese Zeilen angesprochen. Gemeinsam haben wir in teils nächtlichen Implementierungssessions Prototypen entwickelt, um die besten Konzepte gerungen, Dokumente vorangetrieben und eine Menge Spaß dabei gehabt. Explizit möchte ich den Kollegen der beteiligten Firmen danken, die stets für den nötigen Praxisbezug sorgten. Insbesondere die Zusammenarbeit mit Lars Mündermann hat für mich neue Perspektiven eröffnet. Aus der Vielzahl von beteiligten Projektpartnern möchte ich besonders Björn Andersen, Max Rockstroh und Stefan Franke (in alphabetischer Reihenfolge) hervorheben, die mich während der gesamten Zeit begleitet haben und zu guten Freunden geworden sind.

Abschließend möchte ich meinen Freunden und meiner Familie für die Unterstützung danken, insbesondere Sarah Sahl und Danielle Gluns, die in dieser Zeit an meiner Seite waren und meinen Eltern, Monika und Heinz Kasparick, die mir durch ihre Unterstützung und ihren Rückhalt den Raum für die bestmögliche Ausbildung gegeben haben.

Lebenslauf

Martin Kasparick
geb. am 10.02.1986 in Wismar, Deutschland

2013 – 2021	Promotionsstudium und wissenschaftlicher Mitarbeiter an der Universität Rostock, Institut für Angewandte Mikroelektronik und Datentechnik Abschluss: Promotion
2006 – 2013	Studium der Informatik an der Universität Rostock Abschluss: Dipl.-Inf.
1996 – 2005	Gymnasium Dorf Mecklenburg Abschluss: Abitur

Zusammenfassung

Medizinische Eingriffe, Behandlungen und Untersuchungen im Krankenhaus werden immer komplexer. Der Grad der Technisierung und Digitalisierung steigt stetig an. Dabei stellen aktuelle Medizingeräte isolierte Insellösungen dar, die keine Informationen und Interaktionsmöglichkeiten nach Außen bereitstellen. Allenfalls ist dies innerhalb des geschlossenen Ökosystems eines Herstellers möglich – eine herstellerübergreifende Vernetzung und Interoperabilität existieren bisher nicht. Immer wieder treten während Operationen Probleme auf, die in einem vernetzten Operationssaal (OP-Saal) nicht existieren würden. Experten gehen von jährlich mehreren Tausend Todesfällen aus, die durch interoperable Medizingeräte vermieden werden könnten. Der mangelnde Informationsaustausch und fehlende Fernsteuerungsmöglichkeiten erschweren zudem medizinische Arbeitsabläufe im Krankenhaus im Allgemeinen. Allein die Flut an Alarmen und Fehlalarmen schädigt Patienten und medizinisches Personal direkt und indirekt.

Viele innovative Lösungen erfordern eine Vernetzung über Herstellergrenzen hinweg, die einen zuverlässigen Austausch von Informationen und Steuerungsbefehlen ermöglicht. Die neue *IEEE 11073 Service-oriented Device Connectivity (SDC)*-Normenfamilie hat das Potential, die grundlegende Technologie zur Lösung dieser Herausforderungen darzustellen. Diese Normenfamilie, an deren Konzeption und Erstellung der Autor mitgewirkt hat, wird in der vorliegenden Arbeit vorgestellt. Der Fokus liegt auf einer detaillierten Einführung in die sogenannten *Kernstandards*, *IEEE 11073-10207*, *-20701* und *-20702*. Eine Performanzevaluation von verschiedenen Open-Source-Implementierungen zeigt die technische Umsetzbarkeit der Vernetzungstechnologie.

Diese Arbeit beleuchtet drei Anwendungsbereiche, die sich an den Bedarfen der klinischen Akteure orientieren. Die zuverlässige Auslösung von Gerätefunktionalitäten, beispielsweise der Rotation einer chirurgischen Fräse, über das Netzwerk kann die Anzahl der benötigten Fußschalter in komplexen Operationen reduzieren. Damit können Probleme wie Fehlauslösungen vermieden werden. Das zweite Konzept widmet sich ebenfalls der vernetzten Steuerung. Es erlaubt es, beliebig viele Fernsteuerungselemente und Fernsteuerungsoperationen verschiedener Geräte miteinander dynamisch zu assoziieren. Fernsteuerungselemente sind in diversen Formen in heutigen OP-Sälen vorhanden, z. B. Schalter an den Handgriffen von Endoskopen. Das vorgestellte Konzept bewerkstelligt die Auslösung von Fernsteuerungsoperationen, z. B. Parameteränderungen, an einem Gerät eines anderen Herstellers. Ein Beispiel ist das Verändern von Parametern wie der Fräsendrehzahl. Dies erlaubt es den steril operierenden Akteuren, Geräteeinstellungen selbstständig zu verändern. Heutige Probleme wie Wartezeiten oder fehlerhaftes Verstellen können reduziert werden. Der dritte Anwendungsfall beschreibt verteilte Alarmierungssysteme. Diese realisieren eine zuverlässige Generierung von Alarmsignalen an dem Ort, an dem es sinnvoll für Ärztinnen und Pflegende ist. Gerade auf einer Intensivstation, auf der mehrere Patienten gleichzeitig zu versorgen sind, ist dies häufig nicht am Bett des alarmauslösenden Patienten, sondern dort, wo sich die zuständige Person gerade aufhält.

Für die genannten Anwendungsbereiche werden die technischen Anforderungen abgeleitet. Auf dieser Basis werden neuartige Lösungen mittels *IEEE 11073 SDC* entwickelt. Der Fokus liegt dabei auf Patientensicherheit, Flexibilität und dem effektiven Einsatz der Fähigkeiten der Geräte-zu-Geräte-Kommunikation der neuen Normenfamilie. Für alle vorgestellten Konzepte werden die Realisierbarkeit anhand prototypischer Implementierungen gezeigt und mögliche technische wie regulatorische Einschränkungen diskutiert.

Die Innovation der Arbeit liegt im Beitrag zu den offenen *IEEE 11073 SDC*-Technologien und deren konsequenter Nutzung in den Anwendungsbereichen. Es wird nachgewiesen, dass die vorgestellten Konzepte die Aufgaben bestehender, proprietärer Systeme ohne Einschränkungen erfüllen können. Damit sind erstmals herstellerübergreifende, flexible Ad-hoc-Medizingeräteensembles möglich. Die medizinischen, technischen und wirtschaftlichen Potentiale dieser Lösungen sind enorm.

Abstract

Medical interventions, treatments, and examinations in hospitals are becoming ever more complex. The degree of technology-supported procedures as well as digitization increase continuously. However, current medical devices represent isolated solutions, which do not transmit information to or interact with external devices. At the most, this is possible within closed ecosystems of a specific company – manufacturer-independent connectivity and interoperability do not exist so far. This consistently leads to problems that would not exist in networked operating theaters. Experts assume that several thousand deaths per year could be avoided with interoperable medical devices. Furthermore, the insufficient exchange of information and lacking possibilities for remote control hamper medical proceedings in hospitals. The huge number of alarms and false alarms in itself harms patients and medical personnel directly and indirectly.

Many innovative solutions require networking between products of different manufacturers, providing a safe exchange of information and remote controls. The new *IEEE 11073 Service-oriented Device Connectivity (SDC)* family of standards has the potential to become the basic technology for meeting these challenges. The author has participated in the conceptualization and development of this family of standards. This work focuses on the so-called *Core Standards*, *IEEE 11073-10207*, *-20701*, and *-20702*. A performance evaluation of different open-source implementations demonstrates the technical feasibility of the networking technology.

In addition, this work addresses three fields of application that are geared to the needs of clinical actors. The safe activation of a device's functionality, such as the rotation of a surgical drill, via the network can decrease the number of foot switches required in complex operations. This can help to reduce related problems such as the erroneous activation of device funktionalities. The second concept of networked controls enables the dynamic association of a random number of remote-control elements and remote-controllable operations. Remote-control elements are currently available in operating theaters in various forms, such as switches at the handles of endoscopes. The concept presented here enables a safe remote configuration of settings of another manufacturer's device. An example is the manipulation of parameters like a drill's rotation speed. This allows the sterile operating personnel to manipulate settings independently. Thus, current problems such as waiting times or faulty settings caused by third actors can be reduced. The third field of application discusses distributed alarm systems. They realize the safe generation of alarm signals in places that are convenient for doctors and nursing staff. This is often the current location of the staff rather than the bed of the patient who triggers the alarm, in particular in intensive care units (ICUs), where several patients need to be supervised at the same time.

Technical requirements are specified for each field of application mentioned above. On this basis, novel solutions by means of *IEEE 11073 SDC* are developed. The focus is on patient safety, flexibility, and effectively utilizing the abilities of device-to-device communication of the new family of standards. The realizability of all concepts is shown by prototypical implementations. Moreover, potential technical and regulatory constraints are discussed.

The innovation of the work at hand consists of contributing to the open *IEEE 11073 SDC* technologies and their consequent utilization for the fields of application. It provides proof that the concepts presented here are able to fulfill all tasks of existent, proprietary systems without any limitations. Thereby, manufacturer-independent and flexible ensembles of medical devices become possible for the first time. The medical, technical, and economic potential of these solutions is enormous.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AAMI	Association for the Advancement of Medical Instrumentation
AF	Assured Forwarding
ANSI	American National Standards Institute
AOT	Ahead-of-Time
API	Application Programming Interface
ASTM	American Society for Testing and Materials
BICEPS	Basic Integrated Clinical Environment Protocol Specification
BMBF	Bundesministerium für Bildung und Forschung
BMWi	Bundesministerium für Wirtschaft und Energie
BSON	Binary JSON
CAN	Controller Area Network
CAO	Controller Access Object
CAP	Controller Access Protocol
CoAP	Constrained Application Protocol
CRD	Controller Resource Definition
CT	Computertomographie
DDS	Data Distribution Service
DevSpec	Device Specialization
DHCP	Dynamic Host Configuration Protocol
DICOM	Digital Imaging and Communications in Medicine
DiffServ	Differentiated Services
DIM	Domain Information Model
DIN	Deutsches Institut für Normung
DoS	Denial-of-Service
DPWS	Devices Profile for Web Services
DSCP	Differentiated Services Code Point
ECRI	Emergency Care Research Institute
EEG	Elektroenzephalogramm
EF	Expedited Forwarding
EKG	Elektrokardiogramm
EKU	Extended Key Usage
EPR	Endpoint Reference
ESG	Entfernter Alarmsignalgenerator
EtherCAT	Ethernet for Control Automation Technology
EXI	Efficient XML Interchange
FDA	Food and Drug Administration
FHIR	Fast Healthcare Interoperability Resources
GBoard	Intel Galileo Board
GC	Garbage Collection
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HF	Hochfrequenz
HIMSS	Healthcare Information and Management Systems Society
HL7	Health Level 7
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
ICCAS	Innovation Center Computer Assisted Surgery
ICE	Integrated Clinical Environment

ICU	Intensive Care Unit
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IHE	Integrating the Healthcare Enterprise
IoT	Internet of Things
IQR	Interquartile Range
ISO	International Organization for Standardization
ITS	Intensivstation
JC	Joint Committee for Medical Device Interoperability
JDK	Java Development Kit
JIT	Just-in-Time
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KI	Künstliche Intelligenz
KMU	kleines oder mittleres Unternehmen
LCIM	Levels of Conceptual Interoperability Model
LOINC	Logical Observation Identifiers Names and Codes
MAUDE	Manufacturer and User Facility Device Experience
MDD	Medical Device Directive
MDIB	Medical Data Information Base
MDPWS	Medical Devices Communication Profile for Web Services
MDR	Medical Device Regulation
MDS	Medical Device System
MD PnP	Medical Device „Plug-and-Play“
mmHg	Millimeter-Quecksilbersäule, Torr
MMI	Mensch-Maschine-Interface
ModSpec	Module Specification
MoVE	Modular Validation Environment for Medical Device Networks
MPG	Medizinproduktegesetz
MRT	Magnetresonanztomographie
NIH	National Institutes of Health
NTP	Network Time Protocol
OASIS	Organization for the Advancement of Structured Information Standards
OP	Operation
OP-Saal	Operationssaal
OR	Operating Room
ORiN	Open Robot/Resource interface for the Network
OSCP	Open Surgical Communication Protocol
Pa	Pascal
PAR	Project Authorization Request
PAS	Primäres Alarmsystem
PHB	Per-Hop Behavior
PHD	Personal Health Device
PKP	Participant Key Purpose
PoC	Point-of-Care
PoCSpec	Modular Specialisations for Point-of-Care Medical Devices
PROFINET	Process Field Network
QoS	Quality of Service
RFC	Request for Comments
RFID	Radio-Frequency Identification

RPi	Raspberry Pi
RTSJ	Real-Time Specification for Java
RTT	Round Trip Time
S_pO_2	Pulsoximetrisch gemessene, quasi-arterielle, Sauerstoffsättigung
SCB	Karl Storz Communication Bus
SCO	Service and Control Object
SCOT	Smart Cyber Operating Theater
SDC	Service-oriented Device Connectivity
SDN	Software-Defined Networking
SNOMED CT	Systematized Nomenclature of Medicine Clinical Terms
SOA	Service-Oriented Architecture
SODA	Service-Oriented Device Architecture
SOMDA	Service-Oriented Medical Device Architecture
SPoF	Single Point of Failure
SRTB	Surgical Real-Time Bus
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TJC	The Joint Commission
TLS	Transport Layer Security
TSN	Time-Sensitive Networking
UDDI	Universal Description, Discovery and Integration
UDI	Unique Device Identification
UDP	User Datagram Protocol
UL	Underwriters Laboratories Inc.
UML	Unified Modeling Language
URI	Uniform Resource Identifier
UUID	Universally Unique Identifier
VMD	Virtual Medical Device
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XML	Extensible Markup Language

Glossar

Alarmmüdigkeit

Alarmmüdigkeit (englisch *Alarm Fatigue*) kann als sensorische Überlast aufgrund einer übermäßigen Anzahl von Alarmen und einer daraus resultierenden Desensibilisierung gegenüber Alarmen beschrieben werden. Dies führt zu einem Verpassen bzw. Ignorieren von Alarmen [A 1, 2]. Insbesondere klinisch irrelevante Alarme und Fehlalarme spielen eine große Rolle bei der Entstehung der Alarmmüdigkeit, da die medizinisch entscheidenden Alarme aufgrund der Fehlalarme nicht oder zu spät beachtet werden [A 3, 4].
Eine einheitliche und standardisierte Definition vom *Alarmmüdigkeit* existiert nicht [A 5].

Alarmsignalgenerierung

Für ein beliebiges Alarmsignal ist die Generierung der Oberbegriff, der beschreibt, dass das Alarmsignal dem Nutzer in der entsprechenden Form mitgeteilt wird. Formen sind etwa das Anzeigen von visuellen Alarmsignalen, das Erzeugen von Alarmtönen für akustische Alarmsignale oder das Erzeugen von wahrnehmbaren Impulsen bei haptischen Alarmsignalen.

Alarmsystem

Im Sinne der Norm *IEC 60601-1-8* [A 6] ist ein *Alarmsystem* ein Teil eines Medizingerätes bzw. Medizingerätesystems, das Alarmbedingungen detektiert und ggf. Alarmsignale generiert. Siehe auch Primäres Alarmsystem (PAS), Verteiltes Alarmsystem und Verteiltes Alarmierungssystem.

Alert

IEEE 11073-20701 definiert *Alert* als Synonym für die Gesamtmenge aus patientenbezogenen physiologischen Alarmen (engl. *Alarms*), technischen Alarmen und Hinweisen für die Benutzenden (*Advisory Signals / Notifications*). Ohne Beschränkung der Allgemeinheit wird in diesem Dokument der Einfachheit halber der deutsche Begriff Alarm genutzt, wenn es für die Betrachtung unerheblich ist, ob es sich im medizinischen Sinne um einen Alarm oder einen Hinweis handelt.

Entfernter Alarmsignalgenerator (ESG)

Der Begriff *entfernter Alarmsignalgenerator* beschreibt einen Teilnehmer in einem *verteilten Alarmierungssystem*, der Alarmsignale generiert und nicht das *primäre Alarmsystem* selbst ist. Die räumliche Distanz ist dabei unerheblich. Entscheidend ist die *externe* Generierung außerhalb des *primären Alarmsystems*. In einem *verteilten Alarmierungssystem* können beliebig viele *entfernte Alarmsignalgeneratoren* existieren. Beispiele sind zentrale Arbeitsstationen oder mobile Endgeräte.
Die Norm *IEC 60601-1-8* [A 6] nutzt Beschreibungen wie „Ausgabe von Alarmsignalen an *entfernt* aufgestellten Geräten“. Daher wird in dieser Arbeit der Begriff *entfernt* genutzt. Als alternative Bezeichnung wäre auch *externer Alarmsignalgenerator* denkbar.

Fernsteuerung

Unter einer Fernsteuerung im Kontext der *IEEE 11073 SDC*-Standardfamilie und dieser Arbeit versteht man das Auslösen einer beliebigen Operation, die den Zustand eines Vernetzungspartners (*Service Provider*) verändert, durch einen anderen Vernetzungspartner (*Service Consumer*). Entscheidend ist dabei nicht die physikalische Entfernung zwischen steuerndem und zu steuernden Geräten, sondern dass die Steuerung zwischen zwei Geräten mittels *IEEE 11073 SDC* erfolgt. In der *IEEE 11073 SDC*-Normenfamilie werden zwei gleichbedeutende englische Begriffe genutzt: *External Control* hat den Begriff *Remote Control*, der in den *Kernstandards* genutzt wird, aufgrund möglicher Missverständnisse bezüglich der „Entfernung“, abgelöst.

Garbage Collection

Bei der *Garbage Collection (GC)* werden im *Heap-Speicher* eines Prozesses nicht mehr referenzierte (und damit nicht mehr benötigte) Objekte gesucht und entsprechend gelöscht. Die Java

GC wird automatisch von der *Java Virtual Machine (JVM)* ausgeführt, sodass den Implementierenden das Speichermanagement erleichtert wird [A 7, 8]. Für nähere Informationen über die verschiedenen Arten bzw. Tiefen der *GC* sei beispielweise auf [A 7] verwiesen.

IEEE 11073 SDC Device Specializations
Device Specializations (*Gerätespezialisierungen*, *DevSpecs*) sind Teil der *IEEE 11073* Normenfamilie. Sie beschreiben die Modellierung der Netzwerkrepräsentation und die zu implementierende Repräsentation des Gerätezustands im Vernetzungskontext. Die medizinisch/therapeutischen Anforderungen an die Geräte gehen zumeist aus anderen Standards hervor, wie etwa der *IEC 60601* Serie. Existieren solche partikulären Normen, sind diese als Voraussetzung zu erfüllen. Der Fokus von *Device Specializations* liegt auf der Netzwerkrepräsentation. Die Serie der *Device Specializations* der IEEE 11073 SDC-Familie trägt aufsteigende Standardnummern beginnend mit der Bezeichnung *IEEE 11073-10721* (Hochfrequenz-Chirurgiegeräte): *IEEE 11073-107xx*, wobei *xx* die entsprechenden Endziffern (≥ 21) darstellen. Einen Spezialfall stellen die *Module Specifications (ModSpecs)* des *IEEE 11073-10720* Standards dar. In dieser Norm werden Teilspezifikationen zusammengefasst, die von mehreren Geräteklassen genutzt werden können.

IEEE 11073 SDC Kernstandards
Die *Kernstandards* (*Core Standards*) der *IEEE 11073 SDC*-Standardfamilie umfassen die Normen *IEEE 11073-10207* (*Domänen-Informations- und Service-Modell*, *IEEE 11073-20701* (*Architektur und Binding*) und *IEEE 11073-20702* (*Medical Devices Profile for Web Services*). Zusammen bilden sie die technische Spezifikation von *IEEE 11073 SDC*. Auf den *Kernstandards* bauen die *Participant Key Purpose (PKP)* Standards und *IEEE 11073 SDC Device Specializations* auf.

Integrated Clinical Environment (ICE)
Ein *Integrated Clinical Environment (ICE)* ist eine klinische Umgebung einer Patientin in einem potentiell hoch kritischen Zustand, in der mehrere, heterogene Medizingeräte verschiedener Hersteller interoperabel zu einem medizinischen Gerätesystem zusammengeführt werden [A 9].

Inverkehrbringen („Zulassen“) von Medizinprodukten
Verordnungen und Gesetze regeln welche Geräte als Medizinprodukte in der Patientenversorgung eingesetzt werden dürfen. Für den europäischen Wirtschaftsraum sind dies etwa die *EU-Verordnung 2017/745 „Medical Device Regulation (MDR)“* [A 10], die die *EU-Richtlinie 93/42/EWG „Medical Device Directive (MDD)“* [A 11] ersetzt[0.1], bzw. die entsprechenden nationalen Gesetze, wie etwas das deutsche *Medizinproduktegesetz (MPG)* [A 12]. Es wird definiert unter welchen Voraussetzungen das *(erstmalige) Inverkehrbringen*, also die „[erstmalige] [...] Abgabe eines Produktes [...] zum Vertrieb, zum Verbrauch oder zur Verwendung auf dem Unionsmarkt [...]“ (Art. 2 Nr. 27. und 28. [A 10]), erfolgen darf.
Grundsätzlich ist hierfür eine erfolgreiche Konformitätsbewertung erforderlich. Die Konformitätsbewertung „bezeichnet das Verfahren, nach dem festgestellt wird, ob die Anforderungen [der *MDR*] an ein Produkt erfüllt worden sind“ (Art. 2 Nr. 40. [A 10]). Dies zieht typischerweise das Einhalten von verschiedenen Normen für Sicherheits- und Leistungsanforderungen des Produkts, Qualitäts- und Risikomanagement etc. nach sich. Das Konformitätsbewertungsverfahren unterscheidet sich in Abhängigkeit von verschiedenen Faktoren, wie etwa der Risikoklasse des Medizinprodukts oder der vom Hersteller gewählten Prozessvariante. Abgesehen von nicht-sterilen Produkten mit der niedrigsten Risikoklasse I ist zwingend eine sogenannte *Benannte Stelle* beteiligt. Eine *Benannte Stelle* „bezeichnet eine Stelle, die Konformitätsbewertungstätigkeiten einschließlich Kalibrierungen, Prüfungen, Zertifizierungen und Kontrollen durchführt und dabei als Drittpartei tätig wird [...], die gemäß [der *MDR*] [durch eine nationale,

[0.1] Nach einer dreijährigen Übergangsfrist was die verpflichtend Anwendung der *MDR* ab Mai 2020 vorgesehen. Coronabedingten wurde dies um ein Jahr auf Mai 2021 verschoben.

zuständige Behörde] benannt wurde“ (Art. 2 Nr. 41. und 42. [A 10]).
Umgangssprachlich kann das Ausstellen der Konformitätserklärung auch als „Zulassung“ und das Konformitätsbewertungsverfahren als „Zulassungsverfahren“ bezeichnet werden. Die Verordnungen und Gesetze nutzen diese Begriffe nicht. Aufgrund der Häufigkeit der Verwendung im Alltag und zur Vereinfachung werden diese Begriffe im Zuge der vorliegenden Arbeit meist dennoch genutzt.

Just-in-Time Compilation

Unter *Just-in-Time (JIT)*-Kompilierung versteht man die Übersetzung von Bytecode, seltener von Quellcode, in Maschinencode zur Programmlaufzeit. Im Gegensatz dazu wird bei der *Ahead-of-Time (AOT)*-Kompilierung das gesamte Programm vor der Ausführung in Maschinencode überführt [A 13, 14].
Für diese Arbeit ist die *JIT*-Kompilierung von Java von Interesse. Java Quellcode wird vor der Ausführung zunächst in *Bytecode* übersetzt. *Bytecode* ist plattformunabhängig und wird daher von der *JVM* zur Laufzeit interpretiert. Das Interpretieren, also das Ausführen des *Bytecodes* in der Laufzeitumgebung der virtuellen Maschine, ist im Allgemeinen langsamer als die Ausführung von Maschinencode auf der tatsächlichen Plattform. Daher übersetzt der *JIT*-Compiler zur Laufzeit den Bytecode (bzw. Teile des Bytecodes) in Maschinencode für die reale Plattform [A 15]. Durch das Kompilieren zur Laufzeit können, in der sogenannten „Aufwärmphase“, gezielt Optimierungen vorgenommen werden, die über das *AOT*-Kompilieren hinausgehen [A 13, 16, 17].

Medizingeräteensemble

Ausgehend von der Definition eines „Clinical Worksapces“ in der Norm *IEEE 11073-20701* [D 1] wird der Begriff Medizingeräteensemble in dieser Arbeit wie folgt genutzt: Ein Medizingeräteensemble ist die Menge aller Medizingeräte, die mit einem Patienten interagieren, diesen überwachen oder behandeln. Dies schließt die Medizingeräte ein, die für diese Zwecke im Zusammenhang mit dem entsprechenden medizinischen Vorgang direkt vorbereitet und eingerichtet sind, aber zum aktuellen Zeitpunkt keine aktive Aufgabe übernehmen. Ebenso sind ggf. temporäre Medizingeräteensemble-Teilnehmer eingeschlossen. Im OP-Saal umfasst dies meist anästhesiologische und chirurgische Geräte. Ein Beispiel hierfür ist das folgende Medizingeräteensemble: Anästhesiegerät, Multiparameter-Patientenmonitor, Spritzenpumpen, endoskopischer Geräteturm (u.a. Kamera, Lichtquelle, Insufflator, Sauger), OP-Tisch. Zusätzlich kann etwa ein HF-Chirurgiegerät hinzukommen, das nicht permanent im Einsatz ist, aber für den konkreten Eingriff vorgesehen ist. Ein mobiles Röntgengerät kann temporär Teil dieses Medizingeräteensembles sein. Nicht Teil dieses Medizingeräteensembles ist beispielsweise ein OP-Mikroskop, welches sich in demselben OP-Saal befindet, aber nicht Teil des Eingriffs ist.
Im Sinne dieser Arbeit wird angestrebt, dass Medizingeräteensembles per *IEEE 11073 SDC* vernetzt werden bzw. sind.

Open Surgical Communication Protocol (OSCP)

Open Surgical Communication Protocol (OSCP) ist bzw. war eine im Zuge des *OR.NET*-Projekts gebräuchliche Bezeichnung für Konzept und Umsetzung der interoperablen Medizingerätevernetzung. Im Wesentlichen sind *OSCP* und die *Kernstandards* der *IEEE 11073 SDC*-Normenfamilie gleichbedeutend. Da *OSCP* international und in der Standardisierung nicht genutzt wird, wird auch in dieser Arbeit weitestgehend auf die Bezeichnung verzichtet. Am Ende des *OR.NET*-Projekts hat sich folgende Formel herausgebildet: *OSCP = IEEE 11073 SDC Kernstandards + OR.NET*-spezifische Erweiterungen (wie etwa das *Digital Imaging and Communications in Medicine (DICOM)*-Konfigurationsmanagement über die neue Technologie).

Point-of-Care (PoC)-Medizingerät

Eine einheitliche Definition von *Point-of-Care (PoC)*-Medizingeräten existiert derzeit nicht.

Im Sinne der *IEEE 11073* grenzen sich *PoC*-Medizingeräte und persönliche Medizingeräte, sogenannte *Personal Health Devices (PHDs)*, wie folgt gegeneinander ab: *PoC*-Medizingeräte werden in der Regel von (ausgebildeten) Leistungserbringern, wie Ärzten, Pflegepersonal etc., patientennah zur Versorgung und/oder Diagnostik eingesetzt. Beispiele sind Patientenmonitore, Beatmungsgeräte, chirurgische Geräte wie Endoskope etc. Die *PoC*-Medizingeräte werden typischerweise, meist nach einer entsprechenden Aufbereitung, für die Behandlung vieler verschiedener Patientinnen genutzt.
PHD-Medizingeräte werden hingegen direkt von einem Menschen für sich selbst genutzt und sind in der Regel auch nur einer Person zugeordnet. Die Nutzung erfolgt oft im häuslichen/privaten Umfeld. Die Kommunikations- und Sicherheitsanforderungen sowie die Komplexität der Geräte unterscheiden sich entsprechend dieser Nutzungsszenarien. Beispiele sind Blutzuckermessgeräte, Fitnesstracker etc.
Wird in der vorliegenden Arbeit der Begriff „Medizingerät“ genutzt, ist ein *PoC*-Medizingerät im Sinne der *IEEE 11073* gemeint, soweit es nicht explizit anders formuliert ist oder aus dem Zusammenhang klar hervorgeht. Es sei angemerkt, dass der Begriff *PoC* im Zusammenhang mit Labordiagnostik häufig anders genutzt wird.

Primäres Alarmsystem (PAS)
In einem *verteilten Alarmsystem* wird das Medizingerät als *primäres Alarmsystem* bezeichnet, das die Alarmbedingung überwacht und die Verantwortung für die Alarmierung trägt. Typische Beispiele sind Vitalparameterminitore oder Spritzenpumpen. Der Begriff ist nicht normativ definiert, wird aber in den erläuternden Ausführungen zu *verteilten Alarmsystemen* im Anhang A.2 der Norm *IEC 60601-1-8* [A 6] genutzt. Auch der Begriff *Quellalarmsystem* kann genutzt werden.

Verteiltes Alarmierungssystem
In dieser Arbeit wird für das eigene Konzept bewusst der Begriff *Alarmierungssystem* genutzt. Dies ist eine sprachliche Abgrenzung gegenüber dem allgemeinen Begriff *Alarmsystem*, im Sinne der Norm *IEC 60601-1-8* [A 6]. Es soll klargestellt werden, dass sich die hier entwickelten Konzepte ausschließlich auf den Bereich der *Alarmsignalgenerierung*, im *verteilten* Fall, bezieht. Aspekte wie die Anforderungen an die Bestimmung von Alarmbedingungen und Zuweisung von Prioritäten, Lautstärken, Farben etc. von Alarmsignalen usw. sind jedoch nicht Teil dieser Arbeit.

Verteiltes Alarmsystem
Im Sinne der Norm *IEC 60601-1-8* [A 6] ist ein *verteiltes Alarmsystem* ein Alarmsystem, das mehr als eine Komponente eines Medizingerätesystems enthält. Die Komponenten eines verteilten Alarmsystems können potentiell weit voneinander entfernt sein. Einzelne Komponenten können sich auch außerhalb der Patientenumgebung befinden. Die Form des Datenaustauschs zwischen den Teilen des verteilten Alarmsystems ist dabei unerheblich. Die Norm unterscheidet zwischen *verteilten Alarmsystemen* mit und ohne Bestätigungen der Übermittlung von Alarmbedingungen. Systeme ohne Bestätigungen, die damit einen geringeren Grad an Sicherheit aufweisen, werden auch als *verteilte Informationssysteme* bezeichnet.

X.509.v3-Zertifikat
Kryptografische Zertifikate dienen der sicheren Identifizierung von Vernetzungsteilnehmern. Im Bereich der Internetprotokolle haben sich die X.509 Zertifikate [A 18] durchgesetzt. Diese kommen auch bei der sicheren Kommunikation auf Basis der *Transport Layer Security (TLS)* zum Einsatz. Aktuell wird die Version 3 (X.509.v3 [A 19]) genutzt. Die Zertifikatsstruktur sieht eine Reihe von Informationen vor: Versionsnummer, (Aussteller-weit eindeutige) Seriennummer, Signaturalgorithmus-ID, Zertifikatsaussteller, Gültigkeitszeitraum, Zertifikatsinhaber (*Subject*), öffentlicher Schlüssel des Zertifikatsinhabers, Erweiterungen. Der gesamte Inhalt wird

vom Aussteller signiert. Zertifikatsinhaber und -aussteller werden durch den sogenannten *X.500 Distinguished Name* [A 20] beschrieben. Dieser enthält Informationen wie den gebräuchlichen Namen (*Common Name*), Organisation, Organisationseinheit, Land, Bundesland und Ort [A 21]. Zudem lassen X.509.v3-Zertifikate Erweiterungen zu. Eine solche Erweiterung ist die *Extended Key Usage (EKU)*, die im RFC 5280 [A 22] beschrieben wird. Mittels der *EKU* können Gebrauchszwecke und Rollen spezifiziert werden, für die der zertifizierte öffentliche Schlüssel genutzt werden kann.

1 Einleitung und Motivation

„Das (medizinische) Internet der Dinge kann (in den USA) 50.000 Menschenleben pro Jahr retten.“ (Stan Schneider [A 23])

Mittels einer „weit verbreiteten Interoperabilität von Medizinprodukten könnten 36 Milliarden US-Dollar an Kosten im US-Gesundheitswesen pro Jahr eingespart werden.“ (West Health Institute [A 24])

„Interoperabilität ist bei Medizinprodukten so gut wie nicht vorhanden.“ (Kathy Lesh et al. [A 25])

Diese Aussagen internationaler Expertinnen und Experten bzw. Institutionen zeigen exemplarisch die enormen Potentiale und Herausforderungen von herstellerübergreifender Medizingeräteinteroperabilität für die Gesundheitsversorgung[1.1]. Sie waren gleichzeitig Motivation und Mahnung für die vorliegende Arbeit.

1.1 Notwendigkeit herstellerübergreifender Medizingerätevernetzung

Medizinische Eingriffe, Diagnosen und Behandlungen werden immer komplexer. Dies gilt gleichermaßen für die beteiligten Medizingeräte. Die technische Unterstützung ist aus heutigen Krankenhäusern nicht mehr wegzudenken und nimmt stetig zu. Entsprechend steigt aber auch die Wahrscheinlichkeit, dass Probleme mit solch komplexen Systemen auftreten. Einer der kritischsten Bereiche im Krankenhaus ist dabei der Operationssaal (OP-Saal) [A 26, 27], der für diese Arbeit den Rahmen der Anwendungsfälle vorgibt. Der Fokus auf diesen Teilbereich ist in der Ausrichtung des *OR.NET*-Projekts begründet, in dessen Umfeld große Teile dieser Arbeit entstanden sind. Nichtdestotrotz sind die Herausforderungen und die entwickelten Konzepte und Lösungen auf die Gesamtheit der professionellen Medizingeräte in allen technisierten Bereichen der Gesundheitsversorgung im Krankenhaus übertragbar, insbesondere auf die Intensivstation (ITS), die Notaufnahme, den Schockraum etc.

1.1.1 Medizinischer Alltag

Ein relevant hoher Anteil der unerwarteten Ereignisse, die teilweise zu ernsthaften Schäden, bis hin zum Tod von Patienten führen können, sind auf gerätebezogene Probleme zurückzuführen [A 28–30]. So berichten etwa Weerakkody et al. [A 29], dass der Median der Probleme mit Equipment/Technologie bei 23,5 % aller Probleme liegt.

Die Firma *Real-Time Innovations (RTI)*, ein Anbieter von grundlegender Vernetzungstechnologie, geht in ihrem Whitepaper davon aus, dass alleine in den USA pro Jahr 50.000 Todesfälle durch eine umfassende Vernetzung von Medizingeräten, dem *(Medical) Internet of Things (IoT)*, vermieden werden können [A 23].

Zur Veranschaulichung diene folgendes reales Beispiel, dass von Lofsky [A 31] dokumentiert wurde. Bei einer 32-jährigen Patientin sollte eine laparoskopische, d. h. minimalinvasive, Entfernung der Gallenblase (Cholezystektomie) unter Vollnarkose vorgenommen werden. Bei einem minimalinvasiven Eingriff wird der Operationsraum nicht durch einen großen Schnitt komplett eröffnet. Die Operation erfolgt durch kleine Schnitte, durch die Instrumente ins Operationsgebiet geführt werden. Für eine Sichtverbindung wird ein endoskopisches Kamerasystem genutzt, während meist ein Insufflator bzw. ein Pumpensystem den Operationsraum entsprechend mit Gasen bzw. Flüssigkeiten erweitern. Je nach Eingriff kommen weitere Instrumente zum Einsatz. Diese reichen von rein mechanischen

[1.1] Es sei angemerkt, dass sich solche Aussagen auf die Situation in Krankenhäusern in Industrieländern beziehen. In den Gesundheitssystemen von Entwicklungs- und Schwellenländern können die Herausforderungen elementarer sein, da teilweise bereits eine grundlegende technische Ausstattung und Ausbildung fehlt.

Abbildung 1.1: Routineoperation: Minimalinvasiver Eingriff im Bauchraum (sogenannte *Laparoskopie*) mit einer Vielzahl an genutzten technischen, eingebetteten Systemen. (Foto: Paul Polach [A 33].)

Instrumenten wie Skalpellen oder Pinzetten bis hin zu technischen Systemen wie Hochfrequenz (HF)-Chirurgiegeräten zum Schneiden von Gewebe und Veröden von Blutungen, Motorsystemen für Fräsen oder Bohrer, Ultraschall-basierten Schneidegeräten, Saug-Spül-Pumpen etc. Typische minimalinvasive Eingriffe sind Laparoskopien in der Bauchhöhle oder Arthroskopien in Gelenken. Aufgrund der Durchführungsart mittels kleiner Schnitte wird die minimalinvasive Chirurgie auch umgangssprachlich als Schlüsselloch-Chirurgie bezeichnet.

Eine minimalinvasive Gallenblasenentfernung ist ein Routineeingriff. Komplikationen sind (eigentlich) nicht zu erwarten. Während des Eingriffs wurde eine Röntgenaufnahme vorgenommen. In der Regel wird für einen solchen Vorgang die Beatmung durch den Anästhesisten gestoppt, um die Bewegungen während der Aufnahme zu minimieren. So können Bewegungsartefakte im Röntgenbild vermieden werden. Dies geschah auch bei der betrachteten Operation. Der Röntgentechniker hatte Probleme, den Röntgenfilm unter dem OP-Tisch zu entfernen, da sich dieser verklemmt hatte. Der Anästhesist half bei der Entfernung der Röntgenfilmkassette, sodass die Operation wiederaufgenommen werden konnte. Nach einiger Zeit stellte der Anästhesist auf dem Elektrokardiogramm (EKG) einen stark verlangsamten Herzschlag (Bradykardie) fest. Ihm wurde klar, dass das Beatmungsgerät nach der Aufnahme des Röntgenbildes nicht wieder gestartet wurde. Die Patientin ist letztendlich verstorben.

Eine interoperable Vernetzung zwischen Röntgen- und Beatmungsgerät hätte die beschriebene Situation grundlegend verändert. Arney et al. [A 32] beschreiben etwa ein Geräteensemble, in dem sich die beiden Geräte synchronisieren, sodass die Beatmung ggf. gar nicht unterbrochen werden muss oder automatisch nach der Aufnahme des Röntgenbildes fortgesetzt wird. Die im Einsatz befindlichen Röntgen- und Beatmungsgeräte stammen typischerweise nicht von demselben Hersteller. Entsprechend kann eine Synchronisierung nur auf der Basis eines herstellerübergreifenden Standards erfolgen. Dieses Beispiel zeigt, dass Interoperabilität eine Schlüsselrolle für die Erhöhung der Patientensicherheit spielen kann.

Abbildung 1.2: Gefährliche Alltagssituation im OP-Saal: Verändern von Parametern am Medizingerät. (Foto entspricht *Figure 8* aus [A 34].)

Die Beispiele für alltägliche Probleme, die auf der Basis einer umfassenden Interoperabilität der Medizingeräte im OP-Saal gelöst werden können, sind in der Regel mit weniger weitreichenden Folgen für die Patienten verbunden, als im oben beschriebenen Fall des abgeschalteten Beatmungsgerätes. Dennoch erschweren sie die Arbeitsabläufe in einem beträchtlichen Maß. Abbildung 1.1 zeigt eine *laparoskopische* Operation. Wie bereits beschrieben hat der Chirurg bei einem solchen Eingriff keine direkte Sicht auf das Operationsgebiet. Die Sichtverbindung wird durch ein endoskopisches Kamerasystem ermöglicht, dessen Bild auf einem Monitor angezeigt wird. Diese und weitere notwendige chirurgische Medizingeräte befinden sich im OP-Saal in Abbildung 1.1 am linken Bildrand, unterhalb des Monitors. Es ist zu erkennen, dass ein Großteil der Geräte verdeckt sind und durch das OP-Team nicht eingesehen werden können. Die eingestellten Geräteparameter sind folglich nicht zu erkennen. Eine Vernetzung zwischen den beteiligten Geräten ermöglicht es, solche Informationen in das Sichtfeld der Chirurgin einzublenden. Beispielsweise können Informationsüberlagerungen im Randbereich des endoskopischen Bildes realisiert werden. Neben Informationen der chirurgischen Geräte wird häufig die Bereitstellung von ausgewählten Vitalparametern für die Chirurgen als sinnvoller Vernetzungsanwendungsfall nachgefragt.

Über die Informationsbereitstellung hinaus kann die Fernsteuerung von Geräten über Herstellergrenzen hinweg Arbeitsabläufe vereinfachen und Stress im OP verringern. Das (hoch) sterile chirurgische Team darf keine Geräte außerhalb des Operationsfelds berühren. Müssen Geräteparameter verändert werden, so muss dies durch andere Personen vorgenommen werden. Abbildung 1.2 zeigt beispielhaft zu welch waghalsigem Vorgehen dies teilweise führt.

Nicht selten kommt es auch zu der Situation, dass zum fraglichen Zeitpunkt niemand verfügbar ist, der diese Aufgabe übernehmen kann. Daher kommt der sogenannte *Springer* zum Einsatz. Dabei

handelt es sich meist um eine OP-Pflegefachkraft, die sich um Aufgaben außerhalb des hoch sterilen Operationsbereichs kümmert. Hierzu gehört etwa das Holen von zusätzlich benötigten Instrumenten oder Objekten, das Verändern von Geräteparametern etc. Der *Springer* ist meist für mehrere OP-Säle gleichzeitig zuständig und muss daher zwischen diesen wechseln (*springen*). So kommt es vor, dass die Operation für mehrere Minuten mit unerwünschten Geräteeinstellungen fortgeführt bzw. unterbrochen werden muss oder Einstellungsänderungen aufgrund von Kommunikationsproblemen anders vorgenommen werden, als dies von den Chirurgen gewünscht wird [A 35]. Ein häufiges Beispiel im OP-Alltag ist die Verstellung des OP-Tisches für eine andere Lagerung der Patientin [C 2]. Ist kein nicht-steriler Akteur verfügbar, verzögert sich die Operation meist [C 3]. Dabei kommt es auch immer wieder zu Abstimmungsproblemen [A 36] und Missverständnissen, wie etwa Prof. Hans Clusmann vom Universitätsklinikum der Rheinisch-Westfälischen Technischen Hochschule (RWTH) Aachen aus der Praxis berichtet: während der Operateur eine „Fuß-tief"-Veränderung wünscht, wird zunächst „Kopf-tief" verfahren [C 4].

Herstellerübergreifende Fernsteuerungsmöglichkeiten können solche Probleme lösen oder mildern. Neben neuen Bedienkonzepten kann insbesondere die Nutzung vorhandener Steuerelemente im sterilen Bereich einen großen Beitrag leisten. So könnten verfügbare Knöpfe an Handgriffen von Endoskopen oder Mikroskopen genutzt werden.

Die Kombination beider Bereiche, Informationsbereitstellung und Fernsteuerung über Herstellergrenzen hinweg, hat das Potential, einer weiteren großen Herausforderung im klinischen Alltag zu begegnen – der enormen Anzahl von Alarmen und Fehlalarmen, die in vielen Bereichen des Krankenhauses von Medizingeräten generiert werden. Daraus resultierende Probleme wie Alarmmüdigkeit und Desensibilisierung können zum Tod von Patientinnen führen, der Lärm behindert den Genesungsprozess der Patientinnen und schadet dem Pflegepersonal [A 1–3, 37–40]. Verteilte, herstellerübergreifende Systeme auf der Basis offener Standards können einen wichtigen Beitrag dazu leisten, dass Alarmsignale dort generiert werden, wo es für Ärztinnen oder Pflegende sinnvoll ist. Im Fall einer ITS ist dies oft nicht das Bett des alarmgenerierenden Patienten, sondern ein anderer Ort, an dem sich zu diesem Zeitpunkt ein zuständiger Akteur aufhält. Das Zusammenführen, Abgleichen, Auswerten etc. von verschiedenen Informationsquellen bis hin zu Methoden der *künstlichen Intelligenz (KI)* haben das Potential, die Anzahl von Fehlalarmen zu reduzieren [A 2, 40]. Für eine tiefere Analyse der Herausforderungen im Bereich von Alarmen sei auf Kapitel 9 verwiesen.

Der interoperable Informationsaustausch zwischen Medizingeräten und den angrenzenden Informationssystemen kann es in Zukunft ermöglichen, die richtigen Informationen zur richtigen Zeit, im richtigen Umfang, am richtigen Ort zur Verfügung zu stellen und so Ärztinnen und Pflegepersonal zu unterstützen. Auf der Basis vernetzter Systeme können neuartige Assistenz- und Entscheidungsunterstützungssysteme entwickelt werden [A 41, 42]. Der Behandlungserfolg könnte durch solche Entwicklungen in Zukunft verbessert werden. Zusätzlich bietet die Interoperabilität von Medizingeräten das Potential, klinische Abläufe effizienter zu gestalten, was etwa eine kürzere Operations- und damit Narkosedauer zur Folge haben kann. Auch hier ist von einem positiven Effekt für den Behandlungserfolg auszugehen, da angenommen werden kann, dass die Wahrscheinlichkeit von Komplikationen mit einer längeren Narkosezeit ansteigt [A 43, 44].

Ärzte und Pflegepersonal müssen viel Zeit für die Dokumentation aufbringen. Chirurgen verbringen täglich deutlich mehr als vier Stunden (4:22 h), Anästhesie- und Intensivmediziner etwas mehr als dreieinhalb Stunden (3:32 h) mit der Dokumentationsarbeit. Im Pflegebereich liegt der Dokumentationsaufwand in den beiden Disziplinen bei etwa drei Stunden (2:53 h bzw. 3:13 h) pro Tag. Dies ist mit hohen Kosten verbunden [A 45]. Der Informationsaustausch über Systemgrenzen hinweg kann einen Beitrag leisten, um diese Zeiten zu verringern, indem etwa das handschriftliche Übertragen von Geräteparametern entfällt. Eine Mehrheit der Ärzte gibt an, sie würden die gewonnene Zeit nutzen, um mehr Zeit für ihre Patienten zu haben und die Versorgungsqualität zu erhöhen [A 45].

1.1.2 Kostenreduktion

Experten gehen davon aus, dass eine herstellerübergreifend interoperable Vernetzung auf der Basis von offenen Standards immense Kosten sparen kann. Das *West Health Institute*, eine US-amerikanische Non-Profit-Organisation, kommt in einer Studien zu dem Ergebnis, dass eine weit verbreitete Medizingeräteinteroperabilität 36 Milliarden US-Dollar an unnötigen Kosten pro Jahr allein im US-Gesundheitssystem einsparen kann. Es werden verschiedene Haupttreiber für die Kostenreduktion angeführt:

- Reduktion von unerwünschten, die (Patienten-)Sicherheit gefährdenden, Ereignissen,
- Vermeidung redundanter Untersuchungen,
- Effizientere Prozesse, etwa durch die Vermeidung manueller, fehleranfälliger und zeitaufwändiger Eingabe von Informationen und
- Reduktion der Verweildauer von Patientinnen durch den umfassenden Zugriff auf relevante klinische Informationen.

Einschränkend ist anzumerken, dass eine interoperable Geräte-zu-Geräte-Kommunikation, wie sie durch die *IEEE 11073 SDC*-Normenfamilie ermöglicht wird, nicht allein alle Bereiche abdecken kann, die in der Studie Erwähnung finden. Insbesondere die Anbindung an die Krankenhausinformationssysteme stellt einen wichtigen Baustein dar, der in der beschriebenen Form nicht Teil von *IEEE 11073 SDC* ist. Der ebenso in der Entwicklung befindliche Standard *HL7 FHIR (Health Level 7 Fast Healthcare Interoperability Resources* hat das Potential, diesen Herausforderungen zu begegnen [A 46], [B 10, 18].

Integrierte OP-Säle sind bereits heute am Markt verfügbar. In solchen OP-Sälen können die Medizingeräte Informationen und ggf. Steuerbefehle austauschen. Die Vernetzung erfolgt auf der Basis proprietärer Protokolle einzelner, meist großer Hersteller. Geräte anderer Hersteller sind typischerweise nicht oder nur in einem sehr geringen Umfang in das System integriert. Damit stellen heutige OP-Integrationslösungen monolithische, proprietäre Insellösungen dar. Dies hat sowohl Auswirkungen auf die Nutzerinnen der Geräte, als auch auf den Markt der Hersteller.

Hat sich ein Krankenhaus für einen Anbieter eines integrierten OP-Saals entschieden, ist es kaum möglich, Geräte in die Vernetzung zu integrieren, die nicht von diesem Hersteller oder einem direkten Kooperationspartner angeboten werden. Es kommt folglich zum sogenannten *Vendor-Lock-in-Effekt*. Somit stehen für die Behandlung neue, innovative Geräte, die ggf. verbesserte Eigenschaften aufweisen, nicht zur Verfügung. Andererseits bedeutet dies für die Klinikbetreiber, dass nicht die Geräte mit dem besten Preis-Leistungs-Verhältnis beschafft werden können, sondern nur die, die kompatibel zur proprietären Vernetzungslösung sind.

Für die Hersteller von Medizingeräten verursacht die Integration immense Kosten. *Kaiser Permanente* gehen in einer Studie davon aus, dass die Geräteintegration derzeit 40 % der Entwicklungskosten ausmacht [A 47]. Die Autoren kommen zu dem Schluss, dass sich bis zu 30 % dieser Kosten durch die Nutzung eines Gerätekommunikationsstandards reduzieren lassen. Dies stellt ein Potential dar, das für Innovationen genutzt werden kann. Der Aufwand reduziert sich sowohl bei den bestehenden Anbietern integrierter OP-Säle, als auch bei den Herstellern, die Fremdgeräte in diese Systeme einbringen. Letzte sind typischerweise kleine und mittlere Unternehmen (KMUs). Diese sind von hoher Bedeutung, denn „über 90 % der deutschen Medizintechnik-Unternehmen sind KMUs; sie bilden das innovative Rückgrat der Branche“ [A 48]. Hier setzt auch die öffentliche Förderung von Projekten wie *OR.NET*, *MoVE*, *Modular Specialisations for Point-of-Care Medical Devices (PoCSpec)* etc. (siehe auch Kapitel 2.3) an: Die Förderung von Innovationen und Kooperationen steht ebenso im Fokus wie die Zusammenführung von Marktteilnehmern, die zusammen eine kritische Masse für die Erstellung und Durchsetzung neuer Normen erreichen. Einzelne KMUs könnten dies nicht leisten.

Der Aufwand und die damit verbundenen Kosten für das Testen und die Konformitätsbewertung („Zulassung“)[1.2] von Medizingeräten sind enorm. Die EU-Verordnung *Medical Device Regulation (MDR)* ist bzw. wird für die *Point-of-Care (PoC)*-Medizingeräte bindend [A 10]. Branchenverbände gehen davon aus, dass die Anwendung der *MDR* durch ihr Ziel, höhere Qualitätsstandards zu erreichen, zu einem weiter steigenden Aufwand führen wird [A 49]. Durch die Vernetzung ergeben sich in diesem Bereich große Herausforderungen. Stand heute müssen konkrete Gerätepaarungen *in Verkehr gebracht* („zugelassen“) werden. Bereits bei neuen Produktversionen eines der beteiligten Geräte kann eine erneute Konformitätsbewertung erforderlich werden. Dies verursacht entsprechend hohe Kosten bei allen beteiligten Firmen. Offene Standards ermöglichen es, den Zulassungsprozess grundlegend zu verändern: weg von der Bewertung konkreter Gerätepaarungen hin zu einer Konformitätsbewertung der einzelnen Geräte gegen die Interoperabilitätsstandards und aufbauende Profile in Verbindung mit entsprechenden Zweckbestimmungen. Zwei oder mehr entsprechend *in Verkehr gebrachte* Geräte können und dürfen folglich für die Patientenbehandlung vernetzt werden, ohne dass diese konkrete Paarung als *System* (im Sinne der *MDR*)[1.3] *in den Verkehr gebracht* werden muss. Internationale Normen erleichtern einen solchen effizienten und dennoch sicheren Prozess [A 50, 51].

1.1.3 Zusammenfassung

Zusammenfassend kann festgestellt werden, dass eine herstellerübergreifende Interoperabilität von Medizingeräten auf der Basis offener Standards immense Potentiale in verschiedenen Bereichen der Gesundheitsversorgung birgt, beispielsweise:

- Verbesserung der Patientensicherheit;
- Verbesserung der Arbeitsabläufe des klinischen Personals:
  - Reduktion der Arbeitslast während der Behandlung,
  - Reduktion des Stresslevels während der Behandlung und
  - Dokumentationsunterstützung sowie
- Verbesserung des Behandlungserfolgs;
- Kostensenkung in der Gesundheitsversorgung durch verbesserte Abläufe und Auflösung des *Vendor-Lock-in-Effekts*;
- Kostenreduktion in der Geräteentwicklung durch einen geringeren Integrationsaufwand und
- Kostenreduktion im Test- und Zulassungsprozess der Medizingeräte.

Insbesondere stellt die Interoperabilität eine Schlüsseleigenschaft für Innovationen und verbesserte Gerätefunktionalitäten dar. Trotz dieses Potentials und der steigenden Anzahl von Geräten in der Gesundheitsversorgung sind die meisten derzeit hergestellten Medizingeräte nicht dafür ausgelegt, interoperabel mit anderen Geräten anderer Hersteller zu agieren [A 32]. Obwohl die Feststellung von Lesh et al. [A 25], dass „Interoperabilität bei Medizinprodukten so gut wie nicht vorhanden“ ist, bereits im Jahr 2007 erfolgte, hat sie kaum an Gültigkeit verloren [A 52].

1.2 Ziel und Aufbau der vorliegenden Arbeit

Die vorliegende Arbeit soll einen Beitrag zur zuverlässigen, herstellerübergreifenden Medizingeräteinteroperabilität auf der Basis der *IEEE 11073 SDC*-Normenfamilie leisten. Dabei besteht die

1.2 Siehe auch Glossareintrag zum Thema *Inverkehrbringen („Zulassen“) von Medizinprodukten*.

1.3 Ein *System* im Sinne der *MDR* „bezeichnet eine Kombination von Produkten, die zusammen verpackt sind oder auch nicht und die dazu bestimmt sind, verbunden oder kombiniert zu werden, um einen spezifischen medizinischen Zweck zu erfüllen“ (Art. 2 Nr. 11. [A 10]).

Arbeit im Kern aus zwei Teilen. Im ersten Teil, den Kapiteln 3 und 4, werden die Kernstandards der *IEEE 11073 SDC*-Normenfamilie und die zugrunde liegenden Konzepte detailliert vorgestellt. Dieser Teil bildet die technische Grundlage für diese Arbeit. Die internationale Abstimmung des letzten Teils der drei *Kernnormen* wurde Ende 2018 beendet. Seit Anfang 2019 sind sie damit komplett veröffentlicht und seit 2020 von der *International Organization for Standardization (ISO)* anerkannt. Dies zeigt die Aktualität und zeitliche Relevanz der Arbeit.

Dieser erste Teil der Arbeit stellt eine Besonderheit im Vergleich zu anderen Dissertationen dar. Einerseits werden die Grundlagen präsentiert, die für das Verständnis der Arbeit notwendig sind. Andererseits sind die entsprechenden Kapitel bereits Teil der eigenen Leistung des Autors, da er einer der Entwickler und Co-Autoren der *IEEE 11073 SDC*-Normenfamilie ist. Eine klare Trennung zwischen Grundlagen und Eigenleistung ist daher nicht möglich. Ebenso ist eine klare Zuordnung zu einzelnen Autoren kaum möglich, da eine Standardisierung ein konsensorientierter Prozess aller Beteiligten ist. Dies trifft in Teilen auch auf die Inhalte des Kapitels 6 zu, das Ergebnisse eines Arbeitspakets des *OR.NET*-Forschungsprojekts (siehe Abschnitt 2.3) beschreibt. Alle anderen Teile dieser Arbeit wurden maßgeblich vom Autor erarbeitet. Hinweise zu Co-Autoren finden sich am Anfang der entsprechenden Kapitel.

Der zweite Teil der Arbeit beginnt in Kapitel 5 mit einer Performanzuntersuchung von zur Verfügung stehenden Implementierungen der *IEEE 11073 SDC*-Normen. Ziel ist es, zu evaluieren, ob die Normen praxistauglich umgesetzt werden können. In Kapitel 6 wird das *Session*-Konzept des *OR.NET*-Projekts erörtert, das die dynamische Zusammenstellung von Geräteensembles für einen bestimmten Patienten bei einer bestimmten medizinischen Behandlung ermöglicht.

In den Kapiteln 7 bis 9 wird an einzelnen technischen Anwendungsbereichen das Potential der Medizingerätevernetzung auf der Basis von *IEEE 11073 SDC* verdeutlicht. Betrachtet werden Anwendungen, für die ein zuverlässiger Austausch von Fernsteuerungskommandos benötigt wird, sowie verteilte Alarmierungssysteme. Die Tiefe der Beschreibungen soll es ermöglichen, weitere Implementierungen der Konzepte zu erstellen.

In Kapitel 7 wird beschrieben, wie eine sichere Auslösung von Gerätefunktionalitäten über das Netzwerk mittels *IEEE 11073 SDC* erfolgen kann. Anwendungsbeispiele sind etwa die Auslösung der Energieabgabe eines chirurgischen Lasers / eines HF-Chirurgiegerätes, die Rotation einer chirurgischen Fräse etc. Solche Funktionalitäten können ein hohes Risiko für Patienten und Ärztinnen/Pflegepersonal darstellen. Somit sind technische Vorkehrungen zu treffen, sogenannte *Risikokontrollmaßnahmen*. Diese müssen, im Sinne von *Safety* und *Security*, einen sicheren Betrieb in einem potentiell unzuverlässigen Netzwerk ermöglichen und das Risiko beherrschbar machen.

In zukünftigen Medizingeräteensembles können viele beteiligte Medizingeräte eine Steuerung/Konfiguration durch Netzwerkteilnehmer zulassen. Hierfür würde eine beliebige Anzahl von Schaltelementen, wie Fußschalter, Schalter an Handgriffen von Endoskopen oder Mikroskopen, Touchscreenbasierte Bedienpanels etc. zur Verfügung stehen. Die Kombinations- und Konfigurationsmöglichkeiten steigen mit jedem zusätzlichen Netzwerkteilnehmer an. Einerseits ermöglicht dies einen erheblichen Mehrwert für die Nutzerinnen der Geräte, andererseits muss die Komplexität technisch beherrschbar bleiben. Konzepte für die dynamische Zuordnung zwischen den Fernsteuerungselementen und den zu steuernden Geräten werden in Kapitel 8 vorgestellt.

Der dritte technische Anwendungsbereich dieser Arbeit (Kapitel 9) sind verteilte Alarmierungssysteme, die es ermöglichen, ein Alarmsignal dort zu generieren, wo es für Pflegepersonal und Ärzte sinnvoll ist und die Patientinnen möglichst wenig stört. Dies ist nicht notwendigerweise an dem Ort, an dem die Alarmbedingung erkannt wird. Fernsteuerungsoperationen werden hier etwa zum Bestätigen oder Pausieren von Alarmen durch entfernte Netzwerkteilnehmer genutzt. Aus technischer Sicht sind zusätzlich die zwischen den alarmanzeigenden und den alarmgenerierenden Vernetzungspartnern

ausgetauschten Kommandos für die vorliegende Arbeit entscheidend. Sie ermöglichen einen zuverlässigen Betrieb, bleiben aber dem Endnutzer verborgen.

Die Innovation liegt bei den betrachteten Anwendungsbereichen nicht darin, die technischen Anwendungsfälle an sich zu ermöglichen, sondern die Realisierung auf der Basis einer offenen und standardisierten Vernetzung. Bereits heute sind geschlossene, proprietäre Systeme verfügbar, die den beschriebenen Funktionsumfang ermöglichen. Die Neuerung besteht in der Nutzung offener Standards. Diese Arbeit zeigt, dass die Nutzung der Selbstbeschreibungsfähigkeiten der Standards die Art der Medizingeräteentwicklung und -nutzung nachhaltig verändern kann. Derzeit müssen vernetzte Medizingerätesysteme in sehr enger Abstimmung der beteiligten Komponenten entwickelt und zugelassen werden. Die in dieser Arbeit vorgestellten Mechanismen basieren ausschließlich auf Informationen, die standardisiert zur Laufzeit ausgetauscht werden. Somit können die Vernetzungspartner in Übereinstimmung mit den entsprechenden Standards entwickelt und zugelassen werden, ohne dass zur Entwicklungszeit bekannt ist, mit welchen Geräten die Vernetzung zur Laufzeit erfolgen wird.

1.3 Abgrenzung der vorliegenden Arbeit

Die herstellerübergreifende Interoperabilität von Medizingeräten ist ein breites Feld mit vielfältigen technischen, regulatorischen und wirtschaftlichen Aspekten. Im Zuge einer Dissertation können lediglich Teilaspekte betrachtet werden. Die vorliegende Arbeit konzentriert sich auf technische Aspekte. Die *IEEE 11073 SDC*-Kernstandards und eine Auswahl von technischen Anwendungsfällen im Bereich der Fernsteuerung von Medizingeräten und dem Alarmmanagement stehen dabei im Fokus. Es werden Lösungsbausteine beschreiben, die für reale Medizingeräte in klinischen Anwendungsfällen genutzt werden können.

Die Nutzung herstellerübergreifender Interoperabilität wird diverse Bereiche des Produktlebenszyklus verändern. So werden neue bzw. noch unerprobte Konzepte und Methoden für die Zulassung benötigt. Diese Aspekte werden in der vorliegenden Arbeit nicht berücksichtigt. Die Risikobetrachtungen für die präsentierten technischen Anwendungsfälle geben zwar Hinweise für den späteren Einsatz, das eigentliche Risikomanagement kann aber nur für konkrete Geräte und konkrete klinische Anwendungsfälle vorgenommen werden.

Neue Funktionalitäten und Anwendungen auf der Basis von vernetzten Medizingeräten erfordern veränderte bzw. neue Konzepte der Mensch-Maschine-Interaktion. Im Fall einer Fernsteuerung von Geräteparametern eines anderen Gerätes müssen eine Reihe von Aspekten sichergestellt werden. Beispielweise muss den Nutzenden klar sein, welche Parameter an welchem Gerät von der Aktion betroffen sind, in welchem Kontext die Parameter stehen, welche weiteren Parameteränderungen resultieren können etc. Das Mensch-Maschine-Interface (MMI) ist damit ein wichtiger Teil der Risikobetrachtung und kritisch für die Patientensicherheit. Die Entwicklung von MMIs stellt einen eigenen Forschungsbereich dar und ist nicht Teil dieser Arbeit.

Die Arbeit beschäftigt sich mit technischen Aspekten der sicheren, im Sinne von zuverlässigen, Medizingeräteinteroperabilität. Im Deutschen ist Wort *Sicherheit* sehr weit gefasst und schließt die englischen Begriffe *Safety* und *Security* ein. *Safety* stellt die Betriebssicherheit dar, d. h. den Schutz der Patientin und der Nutzer der Medizingeräte. Bezogen auf die Vernetzung beschreibt *Safety* die Zuverlässigkeit. Unter *Security* ist der Schutz der Informationen zu verstehen. Dies schließt etwa Aspekte des Patientendatenschutzes, aber auch den Schutz vor veränderten Steuerbefehlen/Informationen oder den Schutz vor nicht autorisierten Vernetzungspartnern ein. Grundsätzlich kann *Safety* nicht ohne *Security* erreicht werden [A 53]. Dies gilt insbesondere in vernetzten Systemen, die auf offenen Standards und Standardnetzwerktechnologien basieren. Dennoch muss sich diese Arbeit auf den Bereich der *Safety* beschränken. *Security* wird nicht behandelt. Die Arbeit fokussiert auf ausgewählte *Safety*-Aspekte in den einzelnen technischen Anwendungsfällen, insbesondere auf solche, die direkt mit der Vernetzung zusammenhängen. Werden in der vorliegenden Arbeit die Begriffe „Sicherheit“, „sicher“

oder ähnliche genutzt, so sind sie im Sinne der *Safety* und Zuverlässigkeit zu verstehen, soweit dies nicht anderweitig beschrieben wird.

1.4 Anmerkungen zu Begrifflichkeiten und Zitierweise

Im Sinne der Genderneutralität wird in der vorliegenden Arbeit beliebig zwischen weiblichen und männlichen Formen gewechselt, sofern kein Bezug zu konkreten Personen besteht. Hiermit seinen gleichermaßen alle biologischen und sozialen Geschlechter und Geschlechtsidentitäten gemeint.

Im Themenbereich der vorliegenden Arbeit existieren eine Vielzahl von feststehenden englischsprachigen Fachbegriffen, die auch im deutschen Sprachraum als bekannt vorausgesetzt werden können. So ist etwa der englische Begriff *Service Provider* gebräuchlicher als die deutsche Bezeichnung *Dienstanbieter*. Einige Begriffe lassen sich kaum ins Deutsche übersetzen: Für *Point-of-Care (PoC)*[1.4] werden weder die wörtliche Übersetzung „Ort der Pflege“, noch kontextspezifische Versuche, wie „patientennah“, dem Begriff vollständig gerecht. Hinzu kommen die englischsprachigen Begriffe der *Institute of Electrical and Electronics Engineers (IEEE)*-Normen und anderer Standards. Deutsche Übersetzungen sind häufig unpräzise, umständlich oder schlicht wenig gebräuchlich. In der vorliegenden Arbeit wird daher bei der ersten Nutzung solcher Fachbegriffe eine deutsche Übersetzung gegeben, im weiteren Verlauf aber typischerweise die englische oder eingedeutschte Form genutzt. Insbesondere für die Termini der *IEEE 11073 SDC*-Normenfamilie erleichtert dies die Assoziation und Verknüpfung mit den Standarddokumenten. Für Definitionen von Begriffen, die im Zuge dieser Arbeit genutzt werden, sei zudem auf das Glossar verwiesen.

Die *IEEE 11073 SDC*-Normenfamilie ist ein Teil der etablierten *IEEE 11073*-Serie von Normen. Die „*klassische*“ *IEEE 11073* wird in Abschnitt 2.5 beschrieben, während sich das Kapitel 4 intensiv mit der neuen *IEEE 11073 SDC*-Normenfamilie beschäftigt. Zum besseren Verständnis der Normenbezeichnungen sei kurz auf Aspekte des Standardisierungsprozesses eingegangen.

Auf der Basis von Verträgen zwischen der *IEEE* und verschiedenen anderen Standardisierungsinstitutionen, beispielsweise der *International Organization for Standardization (ISO)*, der *International Electrotechnical Commission (IEC)* oder dem *Deutsches Institut für Normung (DIN)*, durchlaufen die meisten Standards der *IEEE 11073*-Familie ein spezielles, beschleunigtes und verkürztes Verfahren und werden so von diesen Standardisierungsinstitutionen anerkannt. Der Lesbarkeit wegen wird in dieser Arbeit die Kurzform *IEEE 11073-**, stellvertretend für den vollständigen Titel unter Einbeziehung der anerkennenden Organisationen, genutzt. Ebenso wird auf das Mitführen des Jahres-Suffixes verzichtet, wenn dieser keine inhaltlichen Veränderungen kennzeichnet. Die *IEEE* ändert beispielsweise das Jahres-Suffix mit der Anerkennung durch die *ISO*. So sind etwa *IEEE 11073-10207-2017* und *ISO/IEEE 11073-10207-2019* inhaltlich identisch. Es sei zusätzlich darauf hingewiesen, dass gerade im Anerkennungsprozess verschiedene Bezeichnungen für denselben Standard parallel in offiziellen Quellen und Dokumenten existieren, z. B.: *ISO/IEC/IEEE 11073-20701-2020*[1.5], *DIN EN ISO 11073-20701:2020-07*[1.6], *ISO/IEEE 11073-20701:2020*[1.7].

Während sich ein Standard in der Entwicklung bzw. Abstimmung befindet, wird der Standardnummer ein „P“ zur Kennzeichnung als Standardisierungsprojekt vorangestellt. Da die *IEEE 11073 SDC*-Familie in den letzten Jahren standardisiert wurde, finden sich in der Literatur häufig noch die Bezeichnungen IEEE P11073-10207, -20701 und -20702. Wenn es aus dem Zusammenhang klar oder inhaltlich nicht relevant ist, wird das Präfix in dieser Arbeit weggelassen.

Das englische Wort „Standard“ hat im Deutschen zwei Übersetzungen: Standard und Norm. Die *DIN EN 45020* bzw. *ISO/IEC Guide 2:2004* definiert eine Norm als „ein Dokument, das mit Konsens

1.4 Siehe auch Glossareintrag zum Thema Point-of-Care (PoC)-Medizingerät.

1.5 `https://standards.ieee.org/standard/11073-20701-2020.html` (besucht am: 27.10.2020).

1.6 `https://www.beuth.de/de/norm/din-en-iso-11073-20701/323308635` (besucht am: 27.10.2020).

1.7 `https://www.iso.org/standard/78227.html` (besucht am: 27.10.2020).

erstellt und von einer anerkannten Institution angenommen wurde [...]“ [A 54]. Die Anforderungen an Standards sind deutlich geringer. Sie müssen weder von einer anerkannten Institution angenommen, noch im Konsens gefasst werden. So können sich Industrie- oder De-facto-Standards herausbilden, die eine weite Verbreitung und Bedeutung in der Praxis haben, aber keine Normen sind. Standard kann als Oberbegriff angesehen werden, da eine Norm den Anforderungen an einen Standard genügt. Die *IEEE* ist eine von der *ISO* anerkannte Normungsorganisation [A 55]. Wie bereits beschrieben, erfolgt zudem die Anerkennung durch weitere Organisationen, wie etwa die *ISO*. Damit handelt es sich bei der *IEEE 11073 SDC*-Serie unstrittig um Normen im Sinne der genannten Definition. Im allgemeinen Sprachgebrauch ist die Bezeichnung Standard im wissenschaftlichen und industriellen Umfeld allgegenwärtig. Dies mag auf den Einfluss der englischsprachigen Dokumente und Organisationen zurückzuführen sein. Dieser Realität entsprechend werden in der vorliegenden Arbeit beide Begriffe genutzt.

Die vorliegende Arbeit verfügt über mehrere Literaturverzeichnisse. Mit dem Präfix „A“ gekennzeichnete Referenzen verweisen auf das Literaturverzeichnis mit allen externen Quellen, die ohne Beteiligung des Autors entstanden sind (siehe Anhang A). Eigene Arbeiten des Autors, als Erst- oder Co-Autor, die in wissenschaftlichen Journalen oder auf wissenschaftlichen Konferenzen veröffentlicht wurden, beginnen mit dem Präfix „B“. Handelt es sich um eigene Arbeiten, deren Charakter eher im Transfer zwischen Wissenschaft und Praxis liegt, wie etwa anwenderbezogene medizintechnische Fachzeitschriften bzw. Messen und Konferenzen, so wird das Präfix „C“ genutzt. Die entsprechenden Listen befinden sich in den Anhängen B und C. Das Präfix „D“ und die zugehörige Liste in Anhang D wird für Normen bzw. Normungsprojekte genutzt, die unter Beteiligung des Autors dieser Arbeit entstanden sind bzw. entstehen. Studentische Arbeiten, die vom Autor betreut oder co-betreut wurden, sind mit dem Präfix „E“ gekennzeichnet (siehe Anhang E).

2 Stand der Technik und Forschung

2.1 Definition von Interoperabilität

Der Begriff *Interoperabilität* ist nicht einheitlich definiert. Während die Basis der verfügbaren Definitionen sehr ähnlich ist, unterscheiden sie sich vor allem in der Tiefe der Anforderungen und Struktur der Definition. So definiert etwa die US-amerikanische Zulassungsbehörde *Food and Drug Administration (FDA)* Interoperabilität in ihrer Richtlinie zu interoperablen Medizingeräten als „die Fähigkeit, Informationen über elektronische Schnittstellen mit anderen medizinischen und/oder nichtmedizinischen Produkten, Systemen oder Geräten auszutauschen und zu nutzen“ [A 56].

Die für den Europäischen Wirtschaftsraum maßgebliche *Medizinprodukteverordnung* (*Medical Device Regulation (MDR)*) spezifiziert in den Begriffsbestimmungen Interoperabilität als „Fähigkeit von zwei oder mehr Produkten – einschließlich Software – desselben Herstellers oder verschiedener Hersteller, a) Informationen auszutauschen und die ausgetauschten Informationen für die korrekte Ausführung einer konkreten Funktion ohne Änderung des Inhalts der Daten zu nutzen und/oder b) miteinander zu kommunizieren und/oder c) bestimmungsgemäß zusammenzuarbeiten“ [A 10]. Punkt b) bleibt von der Stärke der Forderung hinter der Definition der *FDA* zurück. In Punkt c) deutet die *MDR* an, dass die Zusammenarbeit von Medizinprodukten zum Teil der Zweckbestimmung werden kann. Dies geht über die *FDA*-Formulierung hinaus.

Der Interoperabilitätsbegriff unterliegt auch einer stetigen Weiterentwicklung. So hat die *Healthcare Information and Management Systems Society (HIMSS)* erst 2019 ihre dreistufige Definition um eine Stufe erweitert und präzisiert [A 57–59]:

Level 1: Grundlegende Interoperabilität *Foundational Interoperability* ermöglicht es, Daten von einem System zu einem anderen zu übermitteln oder von diesem zu empfangen. Dieses Level beschreibt also die reine Fähigkeit, Daten auszutauschen.

Level 2: Strukturelle Interoperabilität *Structural Interoperability* definiert das Format und die Syntax der ausgetauschten Daten. Damit wird eine Interpretierbarkeit auf Datenfeldebene ermöglicht.

Level 3: Semantische Interoperabilität *Semantic Interoperability* beschreibt die Fähigkeit von zwei oder mehr Systemen, Informationen auszutauschen, diese Informationen korrekt zu interpretieren, zu nutzen und dem Nutzer ein Verständnis der Bedeutung zu vermitteln. Es werden sowohl die Struktur der Daten genutzt (siehe Level 2) als auch die Beschreibung der Daten durch Codes aus standardisierten und zugänglichen Nomenklatur-Systemen (*Kodifizierung der Daten*).

Level 4: Organisatorische Interoperabilität Die neu hinzugefügte *Organizational Interoperability* bezieht darüber hinaus auch nichttechnische Aspekte mit ein. Durch die Hinzunahme von Governance-, rechtlichen, organisatorischen und sozialen Aspekten soll eine nahtlose Kommunikation und Verwendung von medizinischen Informationen auch zwischen verschiedenen Organisationen ermöglicht werden. Somit erfolgt auch eine Integration in die Endnutzerprozesse und -arbeitsabläufe und in die Informationssystemprozesse im Unternehmen und darüber hinaus.

Robkin et al. [A 60] übertragen das *Levels of Conceptual Interoperability Model (LCIM)* [A 61] mit sechs Leveln auf die Medizingerätedomäne. Die dort beschriebenen Level 1 – 3 sind auf die entsprechenden Level der *HIMSS*-Definition abbildbar. Hingegen können die Level 4 (*Pragmatic Interoperability*) und Level 5 (*Dynamic Interoperability*) als Teil des *HIMSS* Level 3 angesehen werden. Robkin et al. beschreiben sie als die Fähigkeit, den Kontext von Daten zu verstehen (*LCIM Level 4*) und als die Nutzung eines interpretierbaren Zustandsmodells (*LCIM Level 5*). Beides kann als Teil der *HIMSS semantischen Interoperabilität* angesehen werden, da sie Voraussetzungen für die korrekte Nutzung der Informationen sind. Die jeweils höchsten Level beider Definitionen können wiederum als kompatibel angesehen werden.

Im Zuge dieser Arbeit wird die Begriffsdefinition der *HIMSS* genutzt. Es wird gezeigt, wie die *IEEE 11073 SDC*-Normenfamilie, zusammen mit geeigneten Nomenklatur-Standards, die Level 1 – 3 abdeckt und damit auch den Anforderungen von *FDA* und *MDR* entspricht. Es werden Beiträge zum Erreichen der *semantischen Interoperabilität* von Medizingeräten geleistet. Die *organisatorische Interoperabilität* (Level 4 bzw. *LCIM* Level 6) geht über den Umfang dieser Arbeit hinaus. Einzelne Aspekte solcher prozessbasierten Anforderungen werden beispielsweise durch das *OR.NET-Session-Konzept* (siehe Kapitel 6) adressiert.

2.2 Verfügbare kommerzielle OP-Integrationslösungen

Einige der führenden Medizingerätehersteller haben Lösungen für die Vernetzung von chirurgischen Medizingeräten im OP-Saal in ihrem Produktportfolio. In sogenannten integrierten OP-Sälen können die Medizingeräte Informationen und ggf. Steuerbefehle in einem vernetzten System austauschen. Laut verschiedener Studien stellt die OP-Integration einen stetig wachsenden Marktbereich dar [A 62–67]. Den größten Marktanteil haben derzeit die Firmen Stryker (iSuite™), Karl Storz (OR1™), STERIS (Harmony iQ®), Olympus (ENDOALPHA), Getinge/Maquet (Tegris) und IntegriTech (Clarity).

Die verfügbaren Integrationslösungen stellen monolithische Systeme auf der Basis von proprietären Protokollen und Hardware dar. So wird beispielsweise der *Karl Storz Communication Bus (SCB)* mittels der *Controller Area Network (CAN)*-Feldbus-Technologie realisiert, die Firma Olympus nutzt den *FlexRay*-Feldbus, während das *Tegris*-System auf *RS232* basiert[2.1]. Die Grundtechnologien sind häufig standardisierte Technologien. Die aufsetzenden Protokolle sind hingegen zumeist ebenso proprietär wie die genutzten Hardwareverbindungen.

Im Bereich der anästhesiebezogenen Medizingeräte zeigt sich ein ähnliches Bild: Ist eine Vernetzung vorhanden, so erfolgt diese meist proprietär über das Patientenmonitoring oder über ebenfalls proprietäre Lösungen wie die der Firma Capsule Technologies, Inc.

Offene Standards und herstellerübergreifende Interoperabilität von Medizingeräten sind nicht im Fokus der meisten Hersteller. Ein Austausch von Daten und Steuerbefehlen kann folglich nur zwischen den Geräten des bereitstellenden Herstellers untereinander und mit ausgewählten, in das System integrierten, Geräten weiterer Hersteller erfolgen. Hierbei stellt die Anbindung jedes Fremdgerätes einen hohen Aufwand für die beteiligten Firmen dar. Aufgrund der Vielzahl der genutzten Protokolle ist die Integration für die beteiligten Firmen kostspielig. Daher ist der Umfang der integrierten Geräte außerhalb des eigenen Produktportfolios meist sehr überschaubar.

Solche geschlossenen Systeme führen zu einer geringen Flexibilität für Klinikbetreiber und Ärztinnen – es kommt zum bereits beschriebenen *Vendor-Lock-in-Effekt* (siehe auch Kapitel 1.1.2). Aus medizinischer Sicht kann dieser dazu führen, dass die Akteure im OP-Saal potentiell nicht optimal unterstützt werden. Außerdem entstehen durch die eingeschränkte Produktauswahl höhere Kosten für die Klinikbetreiber. Der *Vendor-Lock-in-Effekt* sorgt zusätzlich für Markthemmnisse insbesondere für kleine und mittlere Unternehmen (KMUs). Neue Produkte ohne eine Integration in die proprietären Systeme der großen Hersteller haben potentiell höhere Hürden für eine weite Verbreitung. Des Weiteren stehen die aufgewandten Kosten für die Integration in proprietäre Systeme nicht für neue Innovationen zur Verfügung.

Zusammenfassend muss festgestellt werden, dass derzeit keine Vernetzungslösungen für OP-Säle am Markt existieren, die auf offenen Standards basieren. Herstellerübergreifende Interoperabilität existiert praktisch nicht. Die große Mehrheit der Hersteller behandelt Interoperabilität einerseits mit einer

[2.1] Die Liste stellt lediglich einen kleinen Ausschnitt der am Markt genutzten Technologien dar, in die der Autor aufgrund persönlicher Erfahrungen Einblick hatte. Die Technologien stellen spezifisches Firmen-Know-how dar, sodass referenzierbare Veröffentlichungen kaum verfügbar sind oder für mehrere Tausend Euro erworben werden müssen.

niedrigen Priorität und als rein kostspieliges Unterfangen und andererseits haben die Nutzer bisher die Vorteile nicht ausreichend verstanden und gewürdigt [A 68].

Interoperabilität sollte dabei für die Firmen als Zukunftsinvestition und Schlüsseltechnologie für neue Innovationen angesehen werden. Marktkenner gehen davon aus, dass die Nutzerinnen bzw. Klinikbetreiber kein Geld für eine Vernetzung zum Selbstzweck ausgeben werden, aber dass sie bereit sein werden, Geld für neue klinische Applikationen auszugeben, die durch die Interoperabilität ermöglicht werden [A 69].

2.3 Das Leuchtturmprojekt OR.NET

Die vorliegende Arbeit ist im Zuge der Forschungsprojekte *OR.NET*, *MoVE* und *PoCSpec* entstanden. Aufgrund der maßgeblichen gegenseitigen Beeinflussung wird das Projekt *OR.NET* in diesem Abschnitt besonders vorgestellt.

Von 2012 bis 2016 wurde deutschlandweit im Forschungsprojekt *OR.NET* an herstellerübergreifender Medizingeräteinteroperabilität geforscht. Vom Mittelgeber, dem *Bundesministerium für Bildung und Forschung (BMBF)*, wurde *OR.NET* als sogenanntes *Leuchtturmprojekt* eingestuft. Beteiligt waren etwa 50 geförderte Partnerinstitutionen und zusätzlich rund 40 weitere assoziierte Partner. Diese Projektgröße ermöglichte es, dass alle relevanten Akteure/Interessengruppen das Thema gemeinsam bearbeiteten und ihre Expertise einbrachten: Die Anwendungsfälle und grundlegenden Bedarfe wurden von den Ärzten und dem Pflegepersonal getrieben, sodass die Ingenieure und Informatikerinnen für ihre technischen Umsetzungen entsprechende Lösungen entwickeln konnten. Ergebnisse wurden direkt von den Anwendern evaluiert. Klinikbetreiber brachten ihre wirtschaftliche und prozessorientierte Sicht ein. Durch die Beteiligung von Normungsorganisationen und Zertifizierungsstellen konnten die Ergebnisse zeitnah in die Standardisierung überführt und auf ihre Zulassungsfähigkeit hin geprüft werden. Die starke Industriebeteiligung von Großkonzernen und KMUs ermöglichte es, im Zusammenspiel mit den Forschungseinrichtungen innovative und zukunftsweisende Lösungen zu entwickeln und zu demonstrieren.

Neben der technischen Realisierung konnten somit auch viele andere Themenbereiche und Problemstellungen adressiert werden, die für den Einsatz der neuen Vernetzungstechnologie in realen Medizinprodukten zur Patientenversorgung relevant sind. So wurden etwa regulatorische Aspekte des *Inverkehrbringens* / der Zulassungsfähigkeit bearbeitet. Ebenso standen neue Konzepte für ein adäquates Risikomanagement, das den Herausforderungen einer dynamischen und herstellerunabhängigen Vernetzung genügt, im Fokus [A 70, 71]. Die geschaffenen technischen Vernetzungsmöglichkeiten erlauben neue Applikationen, die teils grundlegend neue Benutzerschnittstellen erfordern. Solche innovativen Mensch-Maschine-Interfaces (MMIs) wurden etwa für integrierte Arbeitsstationen oder universell vernetzte Fußschalter entwickelt [A 72–74]. Des Weiteren wurden Betreibermodelle und Strategien entwickelt, die sich mit dem Einsatz der neuen Technologien im Krankenhaus beschäftigen [A 75].

Für die unmittelbare Evaluation der Ergebnisse und um die Fachöffentlichkeit direkt teilhaben zu lassen, lag ein Schwerpunkt auf der Implementierung von Demonstratoren. Diese wurden auf mehreren Messen gezeigt und werden als permanente Demonstratoren an verschiedenen Standorten weiter genutzt [A 76], [B 14].

2.3.1 Zeitliche Entwicklung

In das deutschlandweite *OR.NET*-Projekt mündeten mehrere Vorprojekte, die in kleineren Konsortien bearbeitet wurden. In Abbildung 2.1 veranschaulichen die blauen Balken die Historie und aktuell laufende Forschungsprojekte. Die Vorprojekte, wie etwa *smartOR* [A 77, 78] oder *Dienst-orientierte*

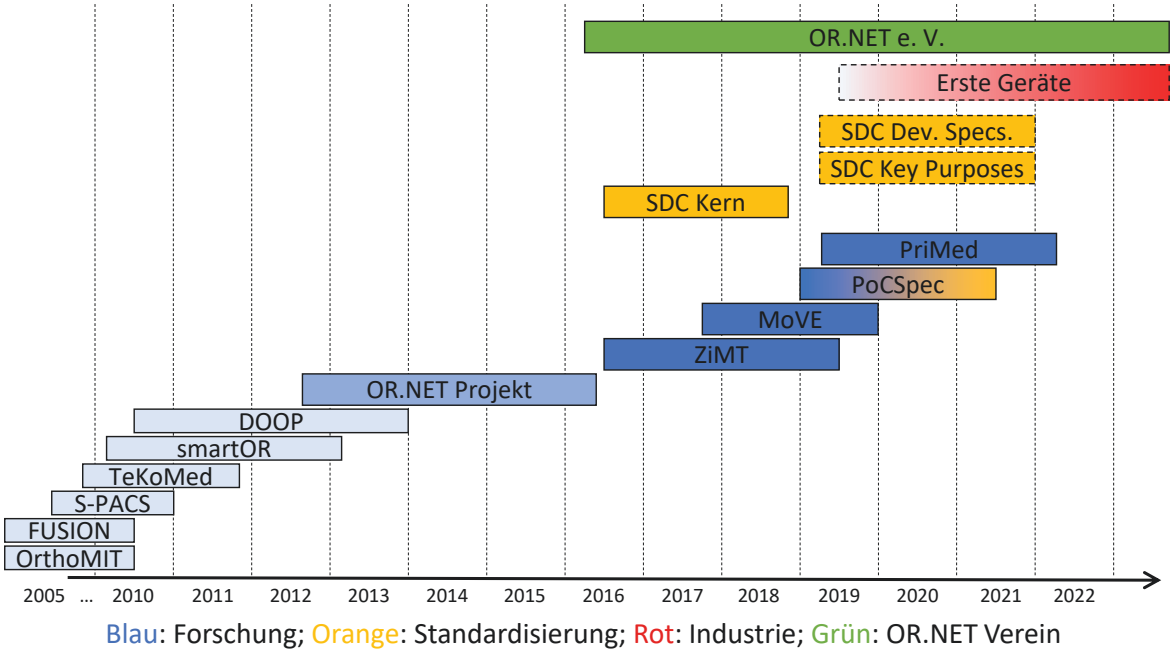

Abbildung 2.1: Zeitliche Übersicht *IEEE 11073 SDC*-bezogener Forschungsprojekte und Standardisierungsaktivitäten.

OP-Integration (DOOP) [A 79], fokussierten hauptsächlich die technischen Realisierungen, während *OR.NET* die beschriebene Breite abdecken konnte.

Aus dem *OR.NET*-Projekt ist der *OR.NET e. V.* hervorgegangen. Der gemeinnützige Verein kümmert sich aktuell um Aspekte wie etwa die Verbreitung der Technologie durch Messeauftritte, Weiterführung der Standardisierung, Vernetzung mit Zulassungsbehörden, Internationalisierung, Koordination von Forschungsprojekten etc. Ebenso bietet er ein Netzwerk für alle interessierten Akteure, wie Unternehmen oder Forschungseinrichtungen.

Im Anschluss an das *OR.NET*-Projekt wurden weitere Forschungs- und Entwicklungsprojekte initiiert, die sich mit bestimmten Teilbereichen beschäftigen. Unter direkter Beteiligung des Autors sind die Projekte *Modular Validation Environment for Medical Device Networks (MoVE)* und *Modular Specialisations for Point-of-Care Medical Devices (PoCSpec)* zu nennen. *MoVE* beschäftigte sich mit der Entwicklung einer Testplattform für dynamisch vernetzte Medizinprodukte, während im Zuge von *PoCSpec* sogenannte *Gerätespezialisierungen* (*IEEE 11073 SDC Device Specializations*, *DevSpec*) entwickelt werden. Der Autor der vorliegenden Arbeit ist auch Co-Initiator des *PoCSpec*-Projekts.

In den auf das Bundesland Nordrhein-Westfalen beschränkten Forschungsprojekten *Zertifizierbare integrierte Medizintechnik (ZiMT)* und *Prozessoptimierung durch integrierte Medizintechnik in OP und Klinik (PriMed)* standen bzw. stehen die Entwicklung von Zulassungsstrategien und der Aufbau einer zentralen, offen vernetzten Chirurgie-Arbeitsstation mit angrenzenden Schnittstellen zur Anästhesie, OP-Pflege und zum OP-Management im Fokus [A 80], [C 2].

Zusätzlich veranschaulicht Abbildung 2.1 die Standardisierungsaktivitäten. Die orangen Balken repräsentieren die Arbeit in der internationalen Projektgruppe der *IEEE*, d. h. von der Annahme des sogenannten *Project Authorization Request (PAR)* bis zum erfolgreichen bzw. prognostizierten Abschluss des Abstimmungsverfahrens (*Ballot*). Neben den bereits verabschiedeten *IEEE 11073 SDC-Kernstandards* (*Core Standards*, *IEEE 11073-10207*, *-20701* und *-20702*) sind die sogenannten *Participant Key Purpose (PKP)*-Standards (*IEEE 11073-10700* bis *-10703*) und die ersten *IEEE 11073 SDC Device Specializations* (*IEEE 11073-10720* bis *-10725*) dargestellt. An diesen Standardisierungsarbeiten war und ist der Autor beteiligt. Für eine Einordnung der einzelnen Standards sei auf den Abschnitt 4.1 verwiesen.

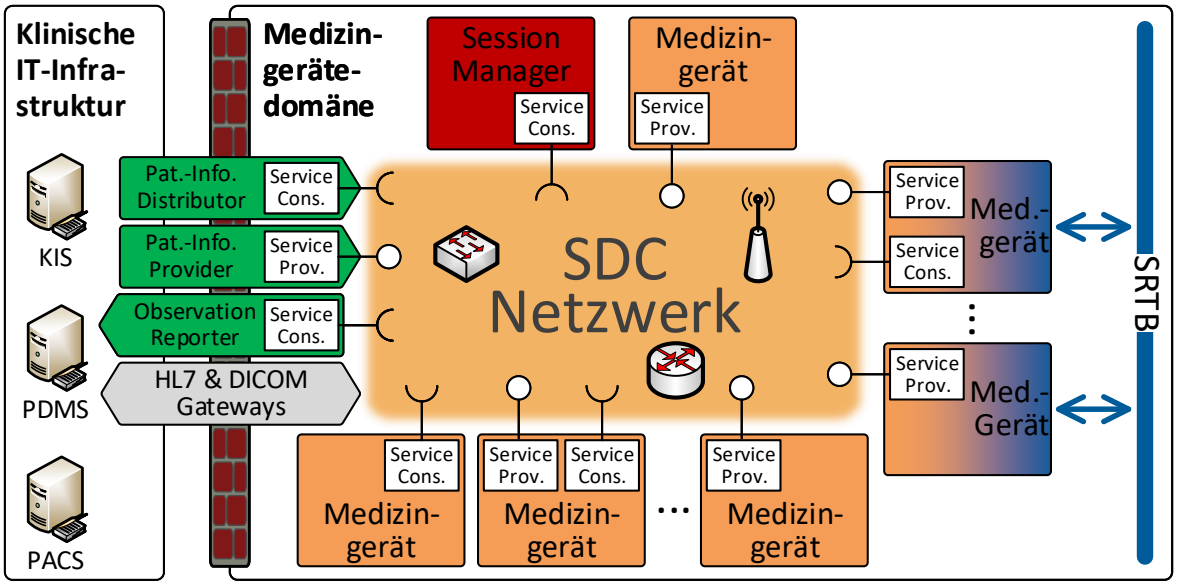

Abbildung 2.2: Technisches Gesamtkonzept der verschiedenen im *OR.NET*-Projekt bearbeiteten Aspekte der Medizingerätekommunikation. Abkürzungen: SRTB – Surgical Real-Time Bus; Prov. – Provider; Cons. – Consumer; Pat.-Info. – Patienteninformationen (inkl. Auftrags-/Eingriffsinformationen); KIS – Krankenhausinformationssystem; PDMS – Patientendatenmanagementsystem; PACS – Picture Archiving and Communication System; HL7 – Zusammenfassung von Health Level 7 Standards wie HL7 v2/v3 oder FHIR; DICOM – Digital Imaging and Communications in Medicine. (Übersetzung von *Figure 1* aus [B 7].)

Abbildung 2.1 verdeutlicht, dass große Anstrengungen über einen langen Zeitraum unternommen wurden. Von den ersten (Vor-)Projekten bis zu den ersten Medizingeräten in der Patientenversorgung sind rund 15 Jahre vergangen.

2.3.2 OR.NET-Architektur und Einordnung dieser Arbeit

Im Zuge des *OR.NET*-Projekts wurde eine technische Gesamtarchitektur entwickelt, die sowohl die Geräte-zu-Geräte-Kommunikation als auch die Verbindung der Medizingeräte zu den angrenzenden klinischen Informationssystemen abdeckt. Abbildung 2.2 veranschaulicht diese Gesamtarchitektur.

2.3.2.1 Geräte-zu-Geräte-Kommunikation Die vorliegende Arbeit fokussiert sich auf die Geräte-zu-Geräte-Kommunikation mittels *IEEE 11073 SDC*. Dieser Teil ist in Abbildung 2.2 orange dargestellt: das *IEEE 11073 SDC*-Netzwerk und die teilnehmenden Medizingeräte. Die entsprechenden *IEEE 11073 SDC*-Konnektoren, für die Rollen *Service Provider* und *Consumer*, sind weiß dargestellt. Einen detaillierten Einblick in die Konzepte und Technologien liefern die Kapitel 3 und 4.

Gelten für den gegebenen Anwendungsfall harte Echtzeiteigenschaften, so wird für diese Kommunikation der *Surgical Real-Time Bus (SRTB)* genutzt. Die blau dargestellten Elemente im rechten Teil von Abbildung 2.2 illustrieren diesen Teil der Gesamtarchitektur. Der *SRTB* folgt nicht dem Paradigma der *Service-Oriented Medical Device Architecture (SOMDA)* (siehe Kapitel 3) und nutzt nicht die *IEEE 11073 SDC*-Normenfamilie. Zur Erfüllung der Anforderungen wird auf die Dynamik und Flexibilität verzichtet. Im Projekt wurden zwei Implementierungen des *SRTBs* entwickelt. Sie nutzen die Echtzeitprotokolle *Ethernet POWERLINK* bzw. *EtherCAT* aus der Domäne der Industrieautomation. Für weiterführende Informationen sei auf die Publikation [B 13] verwiesen, in deren Kontext sich der Autor dieser Arbeit insbesondere mit der Verbindung zwischen *SRTB* und *IEEE 11073 SDC* beschäftigt. Das dort beschriebene Grundkonzept sieht vor, dass die Kommunikation ohne harte Echtzeitanforderungen, wie etwa die Konfiguration von Geräteparametern, über *IEEE 11073 SDC* abläuft. Im Gegensatz dazu werden hart echtzeitkritische Fernsteuerungs- und

Fernauslösekommandos mittels *SRTB* übertragen. Daher sind die entsprechenden Medizingeräte in Abbildung 2.2 orange und blau dargestellt. Für den Betrieb der Echtzeitprotokolle sind *Echtzeit-Master*-Komponenten notwendig (in der Abbildung nicht dargestellt). Die Konfiguration des *Masters* erfolgt mittels *IEEE 11073 SDC* [B 13].

2.3.2.2 Anbindung der klinischen Informationssysteme Im linken Teil von Abbildung 2.2 wird die Anbindung an die klinischen Informationssysteme dargestellt. Für die vorliegende Arbeit wird nicht zwischen den verschiedenen Systemen unterschieden. Der Begriff Informationssysteme wird als Sammelbegriff von Systemen wie dem Krankenhausinformationssystem (KIS), dem Patientendatenmanagementsystem (PDMS), dem Picture Archiving and Communication System (PACS) etc. genutzt.

Einerseits sollen Patientendaten aus den Informationssystemen für die Medizingeräte verfügbar gemacht werden. Das Ziel ist es, die relevanten Daten so bereitzustellen, dass beispielsweise keine zeitaufwändigen und fehleranfälligen mehrfachen Eingaben von Patientendaten an verschiedenen Geräten notwendig sind. Weitere Beispiele sind die Nutzung präoperativer Informationen zur (Vor-)Konfiguration von Geräteparametern oder zur Mitteilung von potentiellen Risiken an die klinischen Akteure.

Andererseits ist insbesondere für Dokumentationszwecke der Datenrücktransport von den Medizingeräten in die Informationssysteme wichtig. Auf der Basis von *IEEE 11073 SDC* werden die Informationen eindeutig einer Patientin zuzuordnen sein und zusätzlich über semantische Annotationen verfügen. Beide Aspekte stellen entscheidende Verbesserungen zur aktuellen klinischen Realität dar. Des Weiteren können intelligente Assistenzsysteme entwickelt werden, die bei der zeitaufwändigen Dokumentationsarbeit unterstützen und somit Ärztinnen und Pflegepersonal entlasten.

Zur technischen Realisierung der Bereitstellung der Patienten- und Eingriffsinformationen für die Medizingeräte wurden zwei Konzepte entwickelt: der *Distributor* und der *Provider* (siehe obere grüne Komponenten in Abbildung 2.2). Diese unterscheiden sich hinsichtlich der implementierten logischen Rolle des *Service-Oriented Architecture (SOA)*-Paradigmas (siehe auch Kapitel 3). Der *Provider* stellt in seiner Rolle als *Service Provider* die Informationen jedem interessierten Vernetzungspartner zur Verfügung. Der *Distributor* agiert als *Service Consumer* und verteilt die Informationen an alle Medizingeräte, die entsprechende *Service Operationen* anbieten. Der *Distributor* verteilt die Informationen also aktiv im Geräteensemble. In der zweiten Realisierungsmöglichkeit werden die Informationen vom *Provider* bereitgestellt und der aktive Part liegt bei den Medizingeräten.

Der *Observation Reporter* (untere grüne Komponente in Abbildung 2.2) sammelt während des Eingriffs Informationen im Geräteensemble, bereitet diese ggf. auf und leitet sie an die Informationssysteme weiter.

IEEE 11073 SDC wurde und wird nicht mit dem Ziel entwickelt, in Konkurrenz zu bestehenden, etablierten oder derzeit ebenso im Entwicklungs- und Standardisierungsprozess befindlichen neuen und vielversprechenden Standards zu treten. So sind etwa *Health Level 7 (HL7) Version 2* (*HL7 V2*), *HL7 Version 3* (*HL7 V3*), *Digital Imaging and Communications in Medicine (DICOM)* oder *HL7 Fast Healthcare Interoperability Resources (FHIR)* zu nennen. Vielmehr schließt *IEEE 11073 SDC* die bestehende Lücke im Bereich der Geräte-zu-Geräte-Kommunikation. Daher sind für die Nutzung dieser Standards entsprechende *Gateways* vorgesehen (siehe graue Komponente in Abbildung 2.2). Für eine Analyse und Abgrenzung von *IEEE 11073 SDC* und *HL7 FHIR* sowie Konzepte für ein zukünftiges Zusammenspiel der Standards sei auf Kasparick et al. [B 10] verwiesen.

Die Kommunikation zwischen dem Medizingeräteensemble und den Informationssystemen ist nicht Teil dieser Arbeit. Einen detaillierten Einblick gibt die Publikation [B 18] des Autors.

2.4 Internationale Forschungsprojekte

Im Folgenden werden die zwei maßgeblichen internationalen Forschungs- und Entwicklungsprojekte bzw. -verbünde und deren Arbeiten vorgestellt.

2.4.1 MD PnP – Medical Device „Plug-and-Play“ (USA)

Das „*Medical Device ‚Plug-and-Play‘ (MD PnP) Interoperability Program*“ wurde 2004 – und damit parallel zu den Aktivitäten in Deutschland – ins Leben gerufen. In diesem US-amerikanischen Forschungsverbund wurde und wird in verschiedenen Projekten an Konzepten und Umsetzungen für integrierte klinische Umgebungen der Zukunft gearbeitet. Die Koordination erfolgt hauptsächlich durch das Massachusetts General Hospital, Boston, USA, insbesondere durch den Direktor des Programms, Julian M. Goldman [A 81].

Der *MD PnP*-Forschungsverbund hat den Begriff des *Integrated Clinical Environment (ICE)* vorangetrieben. Ein *ICE* ist eine klinische Umgebung einer Patientin in einem potentiell hoch kritischen Zustand, in der mehrere, heterogene Medizingeräte verschiedener Hersteller interoperabel zu einem medizinischen Gerätesystem zusammengeführt werden [A 9]. Die grundlegenden Anforderungen und das konzeptuelle Modell für ein *ICE* werden im sogenannten *ICE-Standard ASTM F2761* [A 9] definiert. Dieser Standard kann als eines der wichtigsten Ergebnisse des *MD PnP*-Konsortiums betrachtet werden. Ebenso wurde maßgeblich an der Erstellung des Standards *ANSI/AAMI/UL JC2800* „Standard for Safety for Medical Device Interoperability“ [A 82] mitgearbeitet.

Zusammen mit den Nutzern wurden im *MD PnP*-Forschungsverbund eine Reihe von klinischen Anwendungsszenarien für eine herstellerübergreifende Medizingeräteinteroperabilität entwickelt. Diese sind an verschiedenen Stellen veröffentlicht, etwa im *Clinical Scenario Repository* [A 83] oder in Anhang B des *ICE-Standards* [A 9].

Neben den Standardisierungsaktivitäten stellt *MD PnP* das Open-Source-Softwareframework *OpenICE* zur Verfügung. *OpenICE* wird als Referenzimplementierung des *ICE-Standards* angesehen. Auf der Basis von *OpenICE* wurden umfangreiche Demonstratoren realisiert und der Öffentlichkeit vorgestellt. Beispielweise wurde Ende 2014 ein Szenario zur Behandlung von hoch ansteckenden Ebola-Patienten umgesetzt und präsentiert. Es konnte gezeigt werden, wie durch die herstellerübergreifende Vernetzung von Medizingeräten auf der Intensivstation (ITS) die Pflege für die Patientinnen verbessert und das Ansteckungsrisiko für das Pflegepersonal verringert werden können. So wurde beispielweise ein Remote-Monitoring der Patientinnen und eine Fernsteuerung der angeschlossenen Geräte realisiert. So kann die Zeit, die die Pflegenden im Quarantänebereich verbringen müssen, reduziert werden [A 84].

2.4.1.1 Die *ASTM F2761*-Standards Der *ICE-Standard* „ASTM F2761-09(2013), 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“ [A 9] definiert das Konzept für eine Architektur einer zuverlässigen, patientenbezogenen, integrierten klinischen Umgebung. Inzwischen wurde der Standard von der *American Society for Testing and Materials (ASTM)* zur *Association for the Advancement of Medical Instrumentation (AAMI)* überführt und trägt nun die Bezeichnung *ANSI/AAMI 2700-1:2019* [A 86]. In dieser Arbeit wird weiterhin die Bezeichnung *ASTM F2761* genutzt, da von einer höheren Bekanntheit ausgegangen wird.

Das Architekturkonzept ist in Abbildung 2.3 dargestellt. Innerhalb des *ICE* werden die folgenden Rollen beschrieben [A 85]:

- beliebig viele *ICE-Interfaces* zur Anbindung von Medizingeräten und externen Ressourcen, wie den Krankenhausinformationssystemen,

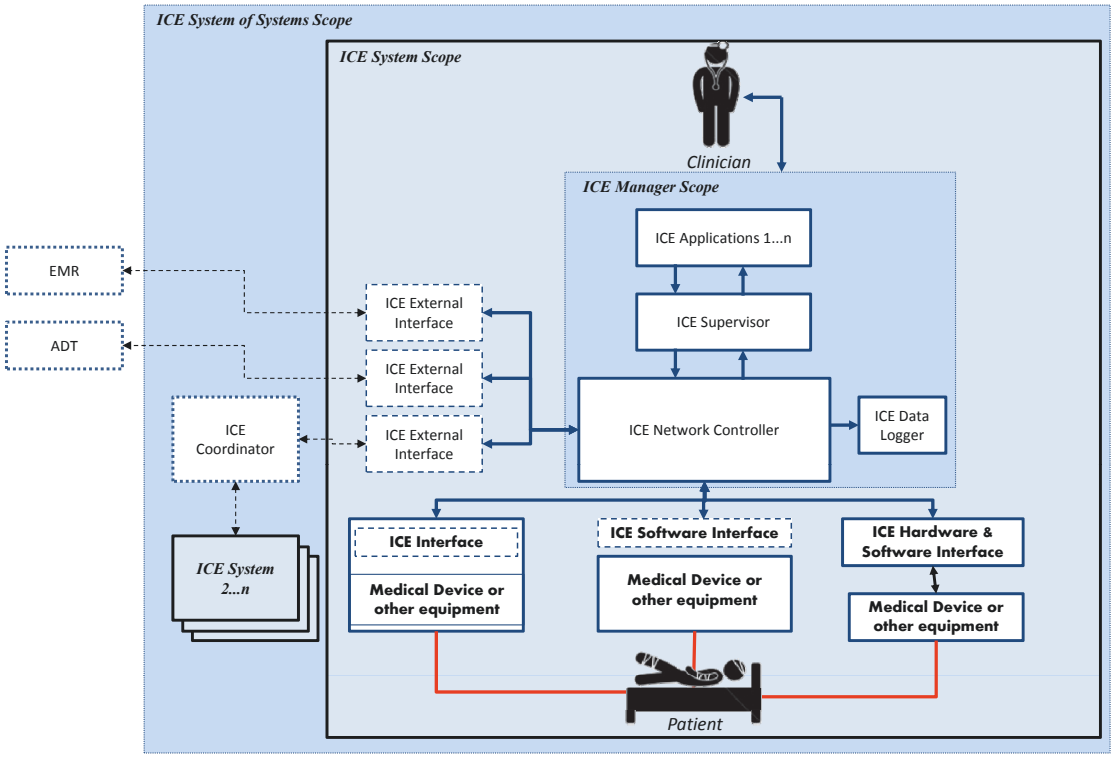

Abbildung 2.3: Systemarchitekturkonzept des *ICE-Standards*. (Abbildung entspricht *Figure 1* aus [A 85], nach [A 9].)

- ein *Netzwerk-Controller*, der den Datenverkehr koordiniert,
- ein *Supervisor*, der die Kommunikation überwacht und Anwendungen bereitstellen kann,
- ein Daten-Logger zur Aufzeichnung, Fehlerbehebung und forensischen Analyse und
- ein Koordinator zur Synchronisierung zwischen mehreren *ICE*-Systemen, um etwa das reibungslose Verlegen einer Patientin vom OP-Saal auf die ITS zu ermöglichen.

Für die Integration von medizinischen oder anderen Geräten werden im unteren Teil von Abbildung 2.3 drei Arten von *ICE-Interfaces* dargestellt (von links nach rechts) [A 85]:

- für Geräte, die ein natives *ICE-Netzwerkinterface* besitzen,
- für Geräte, die eine Standardnetzwerkschnittstelle besitzen, aber eine Softwareerweiterung außerhalb der Kernfirmware benötigen und
- für Geräte, die nur über proprietäre Hard- und Softwareschnittstellen verfügen, sodass ein separater Konnektor zur Anbindung zum Einsatz kommt.

Der *ICE-Manager*, der im oberen mittleren Teil von Abbildung 2.3 abgebildet ist, besteht aus vier Komponenten: dem *Netzwerk-Controller*, dem *Supervisor*, dem *Daten-Logger* und einer beliebigen Menge an Anwendungen. Der *Netzwerk-Controller* leitet Daten und Befehle zwischen *ICE*-Geräten und -Anwendungen weiter und stellt die Quality of Service (QoS) sicher. Er ist agnostisch von den tatsächlichen Anwendungsfällen, Geräten und Anwendungen. Typischerweise wird diese logische Rolle als *Middleware* umgesetzt, so auch in der Implementierung *OpenICE*. Damit ist der *Netzwerk-Controller* nicht zwingend eine physikalische Instanz, sondern kann auch durch eine verteilte *Middleware* realisiert werden. Es ist dabei nicht ausgeschlossen, dass einzelne Aufgaben von einer zentralen Instanz übernommen werden [A 85, 87].

Der *ICE-Supervisor* stellt eine Plattform für die funktionale Integration zwischen den *ICE-Teilnehmern* bereit. Er kann Anwendungen/Services bereitstellen, die vom Gesamtsystem oder mehreren Teilnehmern benötigt werden. Beispiele sind eine zentrale Benutzerschnittstelle oder ein Patientenidentifizierungsmanagement. Zudem hat der *ICE-Supervisor* sicherzustellen, dass die funktionalen und nicht-funktionalen Anforderungen eingehalten werden. Ist dies nicht der Fall, so wird ein technischer Alarm ausgelöst [A 9]. Im Fall einer echtzeitfähigen Middleware-Realisierung stellt der *ICE-Supervisor* ein Echtzeit-Scheduling sowie Daten- und Zeitpartitionierung bereit [A 87].

Über das Architekturkonzept hinaus definiert der Standard *ASTM F2761* [A 9] generelle Anforderungen an ein *ICE*. Im Wesentlichen werden hier etablierte Standards herangezogen, etwa *ISO 14971* [A 88] für den Risikomanagementprozess, *IEC 62304* [A 89] für die Entwicklung und den Lebenszyklusprozess von medizinischer Software und *IEC 60601-1-8* [A 6] für Alarmsysteme. Darüber hinaus werden *(Erstinbetriebnahme-)Tests* für die *Basissicherheit* (*Basic Safety*), die *wesentlichen Leistungsmerkmale* (*Essential Performance*) und *ICE*-Kompatibilität von *ICE-Interfaces* gefordert. Für die Kommunikation sind die *Basissicherheit* und die *wesentlichen Leistungsmerkmale* im Normalfall und im *Erst-Fehler-Fall* (*Single Fault Condition*) sicherzustellen.

Für den *ICE-Standard* sind mindestens fünf Teile geplant. Neben dem hier betrachteten Teil 1 („General requirements and conceptual model“ [A 9]) sollen die Teile 2 bis 5 die Anforderungen an den *Netzwerk-Controller* und die *Geräte-Interfaces*, die *Gerätemodelle*, den *Supervisor* und an die sichere und zuverlässige Integration definieren. Bisher ist lediglich der erste Teil verfügbar.

2.4.1.2 Die *ANSI/AAMI/UL JC2800*-Standards

Die gemeinsamen Arbeiten der AAMI und der Underwriters Laboratories Inc. (UL) in ihrem Joint Committee for Medical Device Interoperability (JC) an der *ANSI/AAMI/UL JC2800*-Standardfamilie sind ebenfalls stark durch das *MD PnP*-Projekt beeinflusst. Das Ziel von *ANSI/AAMI/UL JC2800* ist die Definition von Sicherheitsanforderungen (*Safety* und *Security*) für den Aufbau integrierter Systeme aus heterogenen Komponenten. Diese Anforderungen betreffen Architektur, Middleware, Schnittstellen, Netzwerkinfrastruktur, Implementierung etc. der im System beteiligten Vernetzungspartner. Die Standards beziehen sich dabei nicht auf bestimmte Technologien oder Schnittstellenspezifikationen oder schreiben diese vor. Vielmehr soll ein technologieunabhängiger Rahmen spezifiziert werden. Die Definition von Anforderungen auf System- und Komponentenebene soll es den Herstellern ermöglichen, objektive Nachweise zu erbringen, die zeigen, dass ihre Komponenten und integrierten klinischen Systeme sicher in heterogenen Umgebungen arbeiten [A 82, 90].

Die einzelnen Standards der *ANSI/AAMI/UL JC2800*-Normenfamilie sollen sich aufeinander beziehen. Die Organisationsstruktur orientiert sich an der *IEC 60601*-Serie. Allgemeine Anforderungen werden in *AAMI/UL 2800-1* [A 82] spezifiziert. Für architekturspezifische Anforderungen ist die *AAMI/UL 2800-2-x* Serie und für anwendungsspezifische Anforderungen die *AAMI/UL 2800-3-y* Serie vorgesehen. *AAMI/UL 2800-2* bzw. *AAMI/UL 2800-3* sollen entsprechend technologie- bzw. anwendungsfallübergreifende Aspekte abdecken. Die *AAMI/UL 2800-4-x-y* Standards kombinieren aufbauend Architekturen und Anwendungen [A 90]. Bisher wurde lediglich die Norm *ANSI/AAMI/UL JC2800-1:2019 – Standard for Safety for Medical Device Interoperability* [A 82] veröffentlicht.

Die *ANSI/AAMI/UL JC2800*-Standards spezifizieren das Verhalten eines interoperablen Systems, schränken aber nicht dessen Implementierung ein. Die Normenfamilie stellt keine technischen Spezifikationen für eine Interoperabilitätslösung bereit. Damit stehen diese Normen nicht in Konkurrenz zur *IEEE 11073 SDC*-Normenfamilie. Vielmehr ergänzt die *ANSI/AAMI/UL JC2800*-Familie die technische Spezifikation *IEEE 11073 SDC* und stellt den Herstellern Richtlinien für eine sichere Umsetzung von herstellerübergreifend interoperablen Medizingeräteensembles bereit.

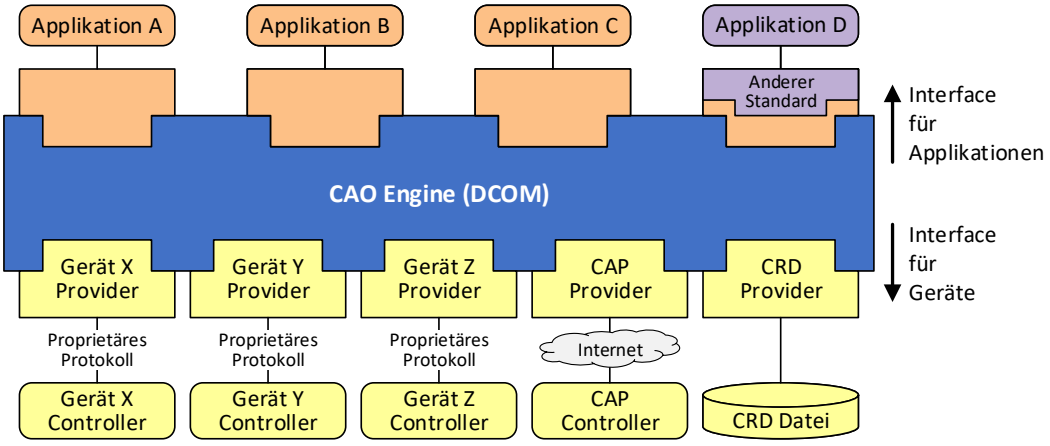

Abbildung 2.4: *ORiN*-Systemarchitektur. Abkürzungen: CAO – Controller Access Object; DCOM – Distributed Component Object Model; CAP – Controller Access Protocol; CRD – Controller Resource Definition. (Abbildung nach *Figure 1* aus [A 92] und *Figure 4* aus [A 93].)

Zum besseren Verständnis erfolgt eine Diskussion der Ergebnisse von *MD PnP* und eine Abgrenzung zu *OR.NET* nach der Einführung in die *IEEE 11073 SDC*-Normenfamilie in Abschnitt 4.8.

2.4.2 SCOT – Smart Cyber Operating Theater (Japan)

Im Zuge des *Smart Cyber Operating Theater (SCOT)*-Projekts arbeiten Wissenschaftler, unter der Führung der *Tokyo Women's Medical University* (Japan), an der Interoperabilität von Medizingeräten im OP-Saal. Die herstellerübergreifende Informationsbereitstellung und die Möglichkeiten zur Fernsteuerung stehen ebenso im Fokus wie in den Projekten *OR.NET* und *MD PnP*. Basierend auf verschiedenen medizinischen Anwendungsfällen wurde und wird ein realer Prototyp eines integrierten OP-Saals entwickelt, der sogenannte *Hyper SCOT*. Der *Hyper SCOT* ist ein Hybrid-OP, in dem intraoperativ Bildgebung auf der Basis von Magnetresonanztomographie (MRT) zum Einsatz kommt. Er befindet sich an der *Tokyo Women's Medical University*. Zusätzlich existieren Versionen mit einem geringeren Funktionsumfang: *Basic SCOT* (Hiroshima University Hospital) und *Standard SCOT* (Shinshu University Hospital) [A 91]. Vernetzungsbasiert werden beispielsweise ein chirurgisches Cockpit, ein Navigationssystem mit Entscheidungsunterstützungsfunktionalität und ein roboterbasierter OP-Tisch und OP-Mikroskop realisiert [A 92].

Technisch basiert die Vernetzung auf dem Industrieautomationsstandard *Open Robot/Resource interface for the Network (ORiN)*, auf dem die sogenannte *OPeLiNK*-Middleware aufsetzt.

2.4.2.1 ORiN Die *Open Robot/Resource interface for the Network (ORiN)*-Middleware wird seit 1999 unter der Führung der *Japan Robot Association (JARA)* entwickelt und wurde 2002 in der Version 1 veröffentlicht. Derzeit wird die Version 2.1 genutzt [A 93–96]. Im Jahr 2011 wurde *ORiN* in den Anhang des Standards *ISO 20242-4* [A 97] aufgenommen.

In Abbildung 2.4 wird die Architektur von *ORiN* dargestellt. Die *Controller Access Object (CAO) Engine* verbindet die Applikations- und die Geräteebene durch Abstraktion und Bereitstellung entsprechender Schnittstellen (Application Programming Interfaces (APIs)). Über Gateways können zusätzlich weitere Technologien eingebunden werden.

ORiN nutzt drei Basistechnologien, um eine herstellerübergreifende Vernetzung zu ermöglichen (siehe auch Abbildung 2.4) [A 92–96, 98]:

Controller Access Object (CAO) *CAO* wurde auf der Basis der *Microsoft*-Technologie *Distributed Component Object Model (DCOM)* entwickelt. Implementierungen stehen für diverse Program-

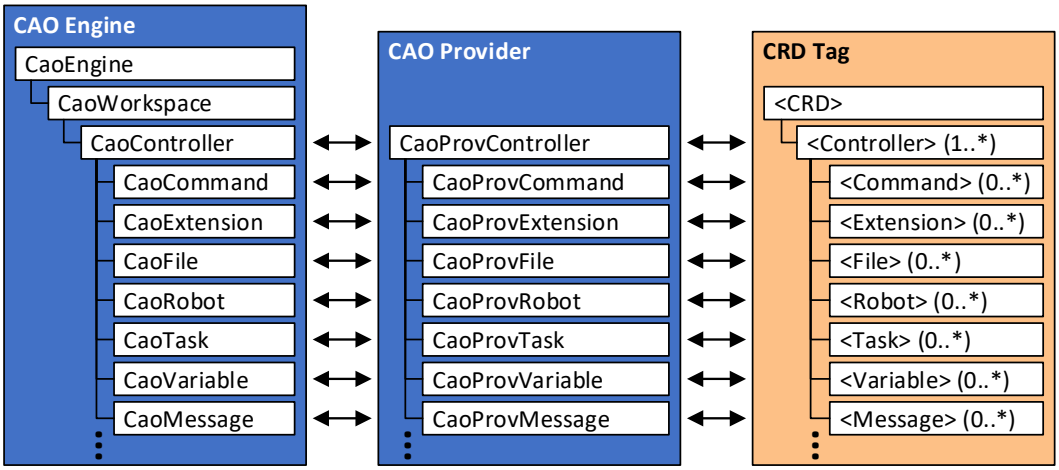

Abbildung 2.5: *ORiN*-Modell: *Controller Access Object (CAO)* und *Controller Resource Definition (CRD)*. (Abbildung nach *Figure 5-1* aus [A 93].)

miersprachen, beispielsweise C#, C++, Java, Visual Basic, Delphi oder LabVIEW, zur Verfügung. *CAO* besteht aus dem *CAO Provider* und der *CAO Engine*. Die *CAO Engine* stellt alle nötigen Schnittstellen zwischen Konsumenten bzw. Applikationen und den *CAO Providern* dar. Der *CAO Provider* bildet dabei die Funktionalitäten und Eigenschaften der integrierten Medizingeräte ab. Das Ziel ist es, verallgemeinerte, geräteunabhängige Funktionen anzubieten. Daher werden Abstraktionen der Geräte bereitgestellt. Die *CAO Engine* verfügt über kein dynamisches Discovery, sodass die Netzwerkadressen aller Teilnehmer bekannt sein müssen.

Controller Resource Definition (CRD) *CRD* ist ein allgemeines, XML-basiertes Datenformat zur Beschreibung der statischen, herstellerabhängigen Geräteinformationen.

Controller Access Protocol (CAP) Das *CAP* ermöglicht es, auf *CAO Provider* über das Internet zuzugreifen. *CAP* nutzt bekannte Technologien wie *SOAP* und *Web Services Description Language (WSDL)*. Für ressourcenbeschränkte Teilnehmer wurden zusätzlich das *Embedded CAP* (e-CAP), das die Informationen mittels *HTTP POST*-Kommandos überträgt, und das *Binary CAP (b-CAP)*, das die Informationen binär codiert per *TCP* überträgt, spezifiziert.

ORiN ermöglicht mittels des *CAO-Objekt-Modells* und der *Controller Resource Definition (CRD)* eine Teilnehmerbeschreibung. Abbildung 2.5 stellt die Beziehung zwischen dem *CAO-Objekt-Modells* von *CAO Engine* und *CAO Provider* und zugehöriger *CRD* dar. Der *CAO Provider*, der eine Abstraktion der realen Geräte darstellt, repräsentiert sich in der *CAO Engine* als ein sogenannter *Controller*. Beschreibungen der Objekte des *CAO Providers* erfolgen über die *CRD*. Die Applikationen nutzen die Schnittstellen der *CAO Engine*.

Im *CAO Interface* und in der *CRD* existieren beispielsweise Elemente zur Beschreibung des Controllers, von Kommandos, Variablen, Nachrichten, Dateien, Robotern etc. Verglichen mit dem *IEEE 11073 SDC*-Datenmodell kann die Ausdrucksfähigkeit als eingeschränkt betrachtet werden. Beispielweise fehlen Möglichkeiten zur semantischen Annotation, der Spezifikation von Einheiten, Messqualität, Auflösungen etc.

2.4.2.2 OPeLiNK Für die Nutzung im OP-Saal wurde die *OPeLiNK*-Middleware auf der Basis von *ORiN* entwickelt [A 92]. *OPeLiNK* stellt grundlegende Anwendungen und Geräteanbindungen bereit. Derzeit sind etwa das Patientenmonitoring, das chirurgische Navigationssystem oder das HF-Chirurgiegerät sowie *HL7*- und *DICOM*-Anbindungen implementiert. Als zentrale Dienste stellt *OPeLiNK* Zeitsynchronisation und Datenarchivierung zur Verfügung. Auf der Anwendungsebene

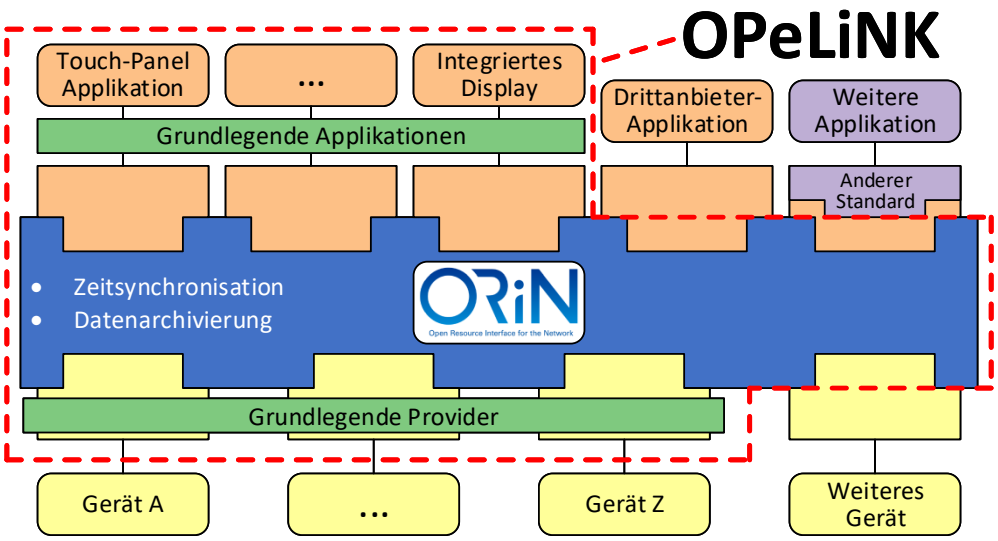

Abbildung 2.6: *OPeLiNK*-Systemarchitektur auf der Basis von *ORiN*. (Abbildung nach *Figure 2* aus [A 92].)

sind beispielsweise die integrierte Arbeitsstation oder das Navigationssystem mit Entscheidungsunterstützungsfunktionalität zu nennen. Über diese Geräte und Anwendungen hinaus kann das System zukünftig erweitert werden.

Abbildung 2.6 veranschaulicht die *OPeLiNK*-Systemarchitektur. Es ist zu erkennen, dass die Basismedizingeräte und -applikationen bereits direkt Teil des Systems sind. Erweiterungen können entsprechend hinzugefügt werden.

2.4.2.3 MRLink *OPeLiNK* besitzt keine harte Echtzeitfähigkeit [A 92]. Für den Anwendungsfall von sogenannten *Master-Slave-Robotern* wird daher an der *MRLink*-Middleware gearbeitet [A 99, 100]. Bei einem *Master-Slave-Roboter* folgt der *Slave* den Bewegungen des *Masters*. Der Chirurg steuert dabei typischerweise das *Master*-System, während der *Slave*-Roboter die Aktionen und damit die eigentliche Operation ausführt. Einer der derzeit bekanntesten chirurgischen *Master-Slave-Roboter* ist der *da Vinci* des US-amerikanischen Herstellers *Intuitive Surgical Inc.* Bei kommerziell verfügbaren Systemen sind die *Master*- und *Slave*-Komponenten untrennbar. Die Forschungsaktivitäten haben hingegen das Ziel, eine freie Kombination vom *Master* und *Slave* zu ermöglichen. Entsprechend interoperable Schnittstellen sind eine grundlegende Voraussetzung [A 99, 100].

Die sichere Steuerung des *Slave*-Roboters stellt folglich hohe Anforderungen an die Kommunikationsverbindung zwischen *Master* und *Slave*, etwa in Bezug auf das Laufzeitverhalten (Latenz) und dessen Determinismus (Jitter). Die *MRLink*-Middleware wird mit dem Ziel entwickelt, diese zu erfüllen. Iwamoto et al. [A 99] stellten einen *MRLink*-basierten Demonstrator vor, der es ermöglicht, den *Slave*-Roboter eines *ZEUS*-Chirurgierobotersystems[2.2] mit einem selbstentwickelten *Master* zu steuern. Präzise Aussagen zur harten Echtzeitfähigkeit von *MRLink* wurden nach Kenntnis des Autors bisher nicht veröffentlicht.

Zum besseren Verständnis erfolgt eine Diskussion der Arbeiten von *SCOT* und eine Abgrenzung zu *OR.NET* nach der Einführung in die *IEEE 11073 SDC*-Normenfamilie in Abschnitt 4.8.

[2.2] Das *ZEUS Robotic Surgical System* wurde vom US-amerikanischen Hersteller *Computer Motion Inc.* entwickelt und vertrieben. Nach der Fusion mit *Intuitive Surgical Inc.* wurde das System zugunsten des *da Vinci* Systems eingestellt.

2.5 Die „klassische“ IEEE 11073-Normenfamilie

Die neuen Standards für die herstellerübergreifende Interoperabilität von Medizingeräten im Krankenhaus werden als Teil der *IEEE 11073*-Normenfamilie entwickelt. Die *IEEE 11073*-Standardfamilie ist aus der *IEEE 1073*-Normenreihe („Standard for Medical Device Communications“) hervorgegangen. Mit der Gründung des zuständigen *IEEE Standards Committee* im Jahr 1984 [A 101] haben die Aktivitäten bereits eine lange Historie.

Die *IEEE 11073*-Standardfamilie wird in zwei Bereiche unterteilt: *„Health informatics – Point-of-care medical device communication“* und *„Health informatics – Personal health device communication“*. Eine einheitliche Definition von *Point-of-Care (PoC)*-Medizingeräten existiert derzeit nicht. Im Sinne der *IEEE 11073* grenzen sich *PoC*-Medizingeräte und persönliche Medizingeräte, sogenannte *Personal Health Devices (PHDs)*, wie folgt gegeneinander ab: *PoC*-Medizingeräte werden in der Regel von (ausgebildeten) Leistungserbringern, wie Ärzten, Pflegepersonal etc., patientennah zur Versorgung eingesetzt, z. B. Patientenmonitore, Beatmungsgeräte, Endoskopie-Geräte etc. Die *PoC*-Medizingeräte werden typischerweise, meist nach einer entsprechenden Aufbereitung, für die Behandlung vieler verschiedener Patientinnen genutzt.

PHD-Medizingeräte werden hingegen direkt von einem Menschen für sich selbst genutzt und sind in der Regel auch nur einer Person zugeordnet. Die Nutzung erfolgt oft im häuslichen/privaten Umfeld. Die Kommunikations- und Sicherheitsanforderungen sowie die Komplexität der Geräte unterscheiden sich entsprechend diesen Aufgaben bzw. Nutzungsszenarien. Für die vorliegende Arbeit wird die beschriebene Abgrenzung im Sinne der *IEEE 11073* von *PoC* und *PHD* genutzt.

Zusätzlich werden die Standards innerhalb der *IEEE 11073* anhand ihrer Nummern thematisch gegliedert. Beispielsweise befinden sich in der *IEEE 11073-1010X*-Serie Nomenklaturen, in der *-1020X*-Serie Domänen-Informations-Modelle, in der *-104XX*-Serie Gerätespezialisierungen (Standards, die sich mit konkreten Klassen von Geräten beschäftigen, z. B. Pulsoximeter) für persönliche Medizingeräte, in der *-20XXX*-Serie Applikationsprofile, in der *-30XXX*-Serie Transportprofile etc. Schmitt et al. [A 102] geben einen guten Überblick über die Standards und ordnen diese auch dem bekannten *ISO/OSI-Referenzmodell* zu. Es sei angemerkt, dass die Arbeiten der *-103XX*-Serie für Gerätespezialisierungen von *PoC*-Medizingeräten nicht in verabschiedeten Standards gemündet sind.

Die Verbreitung der einzelnen Standards ist sehr unterschiedlich. Die *Integrating the Healthcare Enterprise (IHE) Patient Care Device (PCD)*-Domäne nutzt viele Teile der *IEEE 11073*-Standards. Meist werden aber nur das *Domänen-Informations-Modell* (*IEEE 11073-10201*), bzw. die vereinfachte Form aus dem *PHD*-Applikationsprofil (*IEEE 11073-20601*) und die Nomenklatur (*IEEE 11073-1010X* Serie) genutzt. Dies liegt darin begründet, dass die zugrundeliegenden Transportmechanismen die Anforderungen der aktuellen Architekturen von verteilten Systemen von Medizingeräten nicht erfüllen [A 103].

Die neu erstellten Normen der *IEEE 11073 SDC*-Familie sind mit ihrer modernen Technologie auf der Basis der *SOA* angetreten, bestehende Anforderungslücken zu schließen. Insbesondere die Fähigkeit zur interoperablen Vernetzung von Verbünden aus mehreren Geräten, die wechselseitig Informationen und Steuerbefehle mit mehreren anderen Geräten austauschen (Multi-Punkt-zu-Multi-Punkt-Verbindungen), ist für Anwendungsszenarien im Krankhaus unerlässlich.

2.6 Diskussion

Herstellerübergreifende Interoperabilität von Medizingeräten ist derzeit nicht Stand der Technik. Bei auf dem Markt befindlichen Produkten besteht meist keine Möglichkeit, Informationen oder Steuerbefehle zwischen Medizingeräten mehrerer Hersteller auszutauschen. Bestehende Vernetzungslösungen basieren auf proprietären Schnittstellen und sind in der Regel isolierte, monolithische Systeme eines Herstellers.

Weltweit wird in der Wissenschaft daran gearbeitet, die Medizingeräteinteroperabilität voranzutreiben. International sind die Konsortien *MD PnP* in den USA und *SCOT* in Japan zu nennen. In Deutschland ist es vor allem das *BMBF*-Leuchtturmprojekt *OR.NET* mit seinen Vorgänger- und Nachfolgerprojekten, das alle maßgeblichen Akteursgruppen zu diesem Thema vereint. Ein wichtiges Resultat ist die *IEEE 11073 SDC*-Normenfamilie. Sie stellt derzeit die einzige technische Spezifikation für die offene Vernetzung von Medizingeräten dar. Im Forschungsverbund *MD PnP* wurden der *ICE-Standard* und die Referenzimplementierung *OpenICE* entwickelt. Eine resultierende technische Spezifikation, die von Herstellern umgesetzt werden kann, existiert hingegen nicht. Die im *SCOT*-Projekt genutzte Technologie basiert auf einem anerkannten Standard. Der Vernetzungslösung mangelt es jedoch an Flexibilität und Ausdrucksmächtigkeit der Gerätebeschreibung. Daher wird *IEEE 11073 SDC* als die vielversprechendste Technologie angesehen. Ein detaillierter Vergleich zwischen den Ergebnissen von *MD PnP*, *SCOT* und *OR.NET* erfolgt in Kapitel 4.8.

3 Service-Orientierte Architektur für Medizingeräte

Der *IEEE 11073 SDC*-Normenfamilie liegt das Konzept der *Service-Oriented Architecture (SOA)* zugrunde. Die allgemeine Vernetzungsidee der *SOA* wurde zur *Service-Oriented Device Architecture (SODA)* weiterentwickelt bzw. spezialisiert. Unter Berücksichtigung der Anforderungen an eine Vernetzung von (professionellen) Medizingeräten entwickelte sich die *Service-Oriented Medical Device Architecture (SOMDA)*.

Dieses Kapitel beschäftigt sich mit den konzeptuellen Grundlagen der *SOA* und deren Entwicklungen *SODA* und *SOMDA* und beschreibt damit den theoretischen Unterbau der *IEEE 11073 SDC*-Normenfamilie. Zudem wird auf verbreitete Implementierungen der Konzepte eingegangen.

Bei der *SOA* handelt es sich um ein etabliertes Konzept, das in Wissenschaft, Lehre und Praxis verbreitet ist. Während die *SODA* von de Deugd et al. [A 104] geprägt wurde, war der Autor dieser Arbeit an der Weiterentwicklung zur *SOMDA* beteiligt. Die Entwicklung wurde im *OR.NET*-Konsortium vorangetrieben, sodass eine klare Zuordnung zu einzelnen Akteuren kaum möglich ist. Diesem Kapitel der vorliegenden Arbeit liegt das Kapitel „*From SOA and SODA to SOMDA*“ aus [B 7] zugrunde.

3.1 Service-Oriented Architecture (SOA)

Die *SOA* ist ein Entwurfs- oder Designprinzip, um heterogene und verteilte Systeme miteinander zu vernetzen. Die Grundidee ist, dass sämtliche Fähigkeiten der Vernetzungspartner, wie etwa Funktionalitäten, Methoden oder Applikationen, als Services (Dienste) zur Verfügung gestellt werden. Eine einheitliche und vollständig konsistente Definition der *SOA* existiert derzeit nicht. Daher bezieht sich diese Arbeit auf die weithin anerkannten Grundprinzipien, die aus [A 105–108] abgeleitet werden können:

- *Auffindbarkeit* (*Discoverability*) von Services: Services müssen dynamisch zur Laufzeit von Vernetzungsteilnehmern gefunden werden können, ohne bzw. mit sehr geringem Umfang an Vorwissen/Vorkonfiguration.
- Standardisierte *Service-Verträge* (*Service Contracts*): *Service-Beschreibungen* auf Basis standardisierter Dokumente, die die Schnittstelle zur Interaktion mit dem Service spezifizieren.
- *Service-Abstraktion*: Der Service abstrahiert von der zugrundeliegenden Logik. Somit ist einerseits kein Wissen über interne Vorgänge für die Servicenutzung notwendig und andererseits muss der *Service Provider* keine, ggf. geschützten, Informationen preisgeben.
- *Service-Wiederverwendbarkeit* (*Reusability*): Services sollen so entwickelt werden, dass sie wiederholt über Modalitäten-, ggf. sogar über Domänengrenzen hinweg, eingesetzt werden können.
- *Service-Autonomie*: Ein möglichst geringer geteilter Zugriff auf Ressourcen und hohe physikalische Isolation erhöhen die Fähigkeit des Services, eigenständig zu agieren. Dies verbessert zusätzlich die Vorhersehbarkeit des Laufzeitverhaltens.
- *Zustandslosigkeit* (*Statelessness*): Jede Anfrage wird als unabhängige Transaktion bearbeitet, sodass eine Anfrage insbesondere keinen Bezug auf eine frühere Anfrage hat. Es erfolgt etwa keine Nutzung von Sitzungsinformationen.
- *Service-Komponierbarkeit* (*Service Composability*): Durch die Zusammenführung und Kombination (*Komposition*) einzelner Services können neue Services erzeugt werden.
- *Lose Kopplung* (*Loose Coupling*): Möglichst minimale Abhängigkeiten zwischen logischen Einheiten.

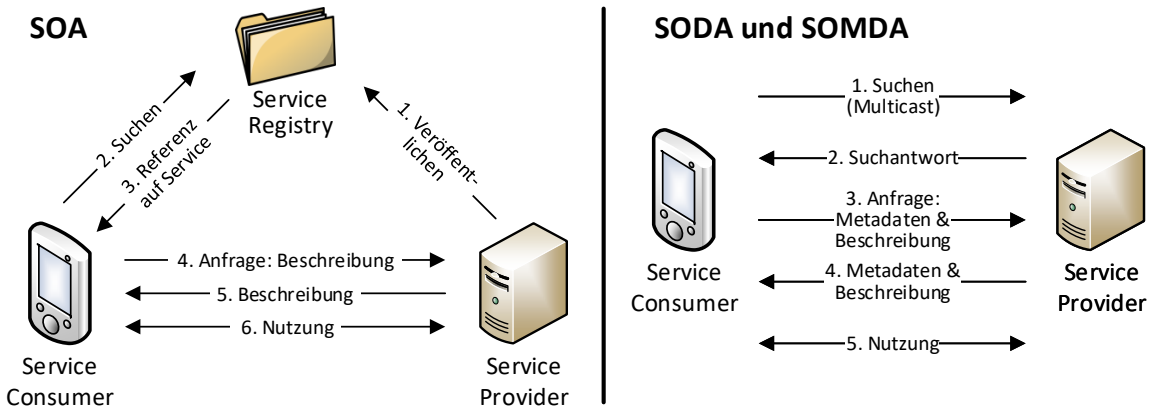

Abbildung 3.1: Gegenüberstellung der Rollen und des *Discovery*-Prozesses der *SOA* und der *SODA* bzw. *SOMDA*. (Abbildung nach *Figure 2* aus [B 7]; linker Teil nach *Abbildung 2.2* und *Abbildung 4.1* aus [A 107].)

Einige der genannten Prinzipien bauen aufeinander auf. So sind beispielsweise die *Auffindbarkeit*, standardisierten *Service-Verträge* und *Service-Autonomie* Voraussetzungen für eine *Lose Kopplung*.

In einer *SOA* existieren drei grundlegende, logische Rollen, die in Abbildung 3.1 schematisch dargestellt werden:

- *Service Provider* (*Dienstanbieter*, auch *Device*),
- *Service Consumer* (*Dienstnutzer*, auch *Client*) und
- *Service Registry* (*Dienstregistrierung*, auch *Service Broker*).

Der *Service Provider* stellt seine Fähigkeiten in Form von Services für die Vernetzungsteilnehmer zur Verfügung. Der *Service Consumer* kann die angebotenen Services, entsprechend den bestehenden Anforderungen und Bedarfen, nutzen. Ein *Service Consumer* stellt aktiv keine Services oder Informationen im Netzwerk zur Verfügung. Die aktive Bereitstellung ist allein der Rolle des *Service Providers* vorbehalten. Ein physikalisches Gerät kann zur Erfüllung seiner Aufgaben beide logischen Rollen implementieren. In komplexen Anwendungsszenarien, wie Medizingeräten, wird dies in der Realität häufig vorkommen.

In der Literatur und dem allgemeinen Sprachgebrauch wird für den *Service Provider* auch der Begriff *Device* und für den *Service Consumer* der Begriff *Client* genutzt. Dies führt gelegentlich zu Unklarheiten, da die logischen Rollen *Device* bzw. *Client* mit einem physikalischen Gerät bzw. mit der Bezeichnung für einen allgemeinen Netzwerkteilnehmer verwechselt werden können.

Die am weitesten verbreitete Umsetzung des *SOA*-Prinzips sind *Webservices*. Das *World Wide Web Consortium (W3C) Web Service Glossary* [A 109] definiert einen *Webservice* als ein Softwaresystem zur interoperablen Maschine-zu-Maschine-Kommunikation über ein Netzwerk. Das Softwaresystem stellt seine Schnittstellen in einer von Maschinen verarbeitbaren Form bereit. Dies erfolgt unter Nutzung der *Web Services Description Language (WSDL)*. Andere Systeme interagieren mit dem *Webservice* über *SOAP*-Nachrichten. *SOAP*-Nachrichten werden typischerweise als *Extensible Markup Language (XML)* serialisiert und mittels *Hypertext Transfer Protocol (HTTP)* ausgetauscht. Für eine Vernetzung auf Basis von *Webservices* kommt zur Realisierung der Dienstregistrierung oft das sogenannte *Universal Description, Discovery and Integration (UDDI)* zum Einsatz.

3.2 Service-Oriented Device Architecture (SODA)

Die Vernetzung auf Basis der *SOA* bietet diverse Vorteile, wie etwa Flexibilität, Plug-and-Play-Fähigkeit, standardisierte und selbstbeschreibende Schnittstellen, geringen Implementierungs- und

Wartungsaufwand etc. Diese Eigenschaften sind auch von hohem Interesse für die Vernetzung von (ressourcenbeschränkten) Geräten, wie Sensoren und Aktoren. De Deugd et al. [A 104] haben daher das Konzept der *Service-Oriented Device Architecture (SODA)* eingeführt. Die Grundidee ist, dass Geräte, bzw. deren Fähigkeiten, als Services modelliert werden. Die *SODA* hat dabei das Ziel, sowohl Geräte untereinander zu vernetzen (horizontale Integration), als auch die verschiedensten physischen Geräte im Feld mit den Enterprise-IT-Systemen zu verbinden (vertikale Integration). Aufbauend führen Mauro et al. [A 110] eine Reihe von Entwurfsmustern (Patterns) ein bzw. verfeinern bestehende Patterns für die *SODA*: *Service Encapsulation*, *Legacy Wrapper*, *Dynamical Adapter* und *Auto-Publishing*. Diese Entwurfsmuster werden bereits von Mauro et al. für den Einsatz im medizinischen Umfeld diskutiert und positiv bewertet.

Die *SODA* basiert auf den beschriebenen Grundprinzipien der *SOA*. Insbesondere Eigenschaften wie *Lose Kopplung*, *Wiederverwendbarkeit* oder *Komponierbarkeit* ermöglichen es, alle Komponenten eines vernetzten Systems aus Geräten und deren Funktionalitäten so unabhängig und einfach wie möglich zu implementieren. Somit verringert die Nutzung des *SODA*-Prinzips die Komplexität des Systems erheblich und damit auch dessen Implementierungs-, Wartungs- und Betriebsaufwand.

Das *Devices Profile for Web Services (DPWS)* [A 111] ist eine spezifische Umsetzung des *SODA*-Konzepts. *DPWS* stellt eine Implementierung von *Webservices* speziell für eingebettete Systeme dar. Die Entwicklung der Spezifikation geht maßgeblich auf das Unternehmen Microsoft zurück [A 112]. Das Ziel war es, die Plug-and-Play-Eigenschaften von Druckern im Netzwerk zu verbessern. Technisch gesehen setzt *DPWS* auf mehreren *Webservice*-Standards auf, den sogenannten *WS-**-Standards, wie etwa *WS-Discovery* [A 113], *WS-Eventing* [A 114], *WS-Policy* [A 115] etc. Um den Anforderungen von ressourcenbeschränkten, eingebetteten Geräten gerecht zu werden, nutzt *DPWS* Teilmengen dieser *WS-**-Standards und definiert Einschränkungen, Konkretisierungen und Erweiterungen. Für die Geräte-zu-Geräte-Kommunikation ist etwa das *Eventing* nach dem *Abonnementprinzip* (*Publish-Subscribe Pattern*) von großer Bedeutung.

Ein wesentlicher Unterschied zwischen *DPWS* und den „klassischen“ *Webservices* besteht im Suchprozess (*Discovery*). Abbildung 3.1 veranschaulicht die Unterschiede. *DPWS* kommt, unter der Nutzung von *WS-Discovery* [A 113], ohne eine zentrale *Service Registry* (z. B. *UDDI*) aus. Ein *DPWS Service Provider* besteht aus zwei Teilen, dem *Hosting Service* und einer beliebigen Anzahl von *Hosted Services*. Die *Hosted Services* bilden die eigentlichen Funktionalitäten ab und werden meist verkürzt als *Services* bezeichnet. Der *Hosting Service* übernimmt die Aufgaben der zentralen *Service Registry* dezentral für jeden *Service Provider* im Netzwerk und stellt die *Hosted Services* bereit. Obwohl keine zentrale *Service Registry* notwendig ist, schließt *DPWS* deren Nutzung nicht aus, wenn diese für das gegebene Anwendungsszenario sinnvoll ist.

Im *Ad-hoc*-Modus, ohne zentrale *Service Registry*, kann der *DPWS-Discovery*-Prozess sowohl explizit als auch implizit erfolgen. Beim expliziten *Discovery* sucht ein *Service Consumer* mittels *UDP-Multicast* `Probe`-Nachrichten nach potentiellen *Service Providern*. Dies kann unter der Angabe bestimmter *Typen* erfolgen. Jeder passende *Service Provider*, genauer gesagt *Hosting Service*, antwortet mit einer entsprechenden `ProbeMatch`-Nachricht. Anschließend werden benötigte Metadaten und Schnittstellenbeschreibungen ausgetauscht, bevor die Servicenutzung beginnt. Für das implizite Discovery sendet ein *Service Provider* beim Betreten des Netzwerks eine *UDP-Multicast* `Hello`-Nachricht und analog eine `Bye`-Nachricht beim Verlassen [A 103].

Im *WSDL*-Standard [A 116] werden *Services* (im Sinne von *Hosted Services*) als Sammlung von *Network Endpoints / Ports* definiert. *Port Types* sind eine Menge von *Operations*. *Operations* sind wiederum als eine Abstraktion einer beliebigen Aktion eines *Services* definiert. *Messages* stellen abstrakte, typisierte Definitionen der ausgetauschten Daten dar. *Port Types*, *Operations* und *Messages* sind unabhängig von der konkreten Bereitstellung im Netzwerk und dem Datenformat. Erst das *Binding* spezifiziert ein konkretes Protokoll und Datenformat für einen *Port Type*, z. B. das

SOAP-over-HTTP-Binding. Der *Port* ist definiert als die Kombination aus *Binding* und Netzwerkadresse. Auf abstrakter Ebene teilen sich *Services* also in *Port Types* auf, die wiederum *Operations* enthalten, mit Hilfe derer über *Messages* Daten ausgetauscht und Aktionen ausgelöst werden können.

3.3 Service-Oriented Medical Device Architecture (SOMDA)

Die Weiterentwicklung von der *SODA* zur *SOMDA* geht weit darüber hinaus, eine *SOA*-basierte Vernetzung von Medizingeräten zu realisieren. Im Fokus der *SOMDA* stehen spezifische Eigenschaften, um den Sicherheitsanforderungen[3.1] von Systemen vernetzter Medizingeräte in Umgebungen für Patientinnen in einem potentiell kritischen Zustand gerecht zu werden. Die Entwicklung der *SOMDA* erfolgte hauptsächlich im Zuge des *OR.NET*-Projekts [A 117]. Im Folgenden werden fünf zentrale Eigenschaften einer *SOMDA* beschrieben:

Nutzung normierter Datenmodelle und Nomenklatur-basierter semantischer Annotationen Standardisierte Schnittstellenbeschreibungen gehören zu den Grundprinzipien von *SOA* und *SODA*. Die *SOMDA* erweiterte dieses Prinzip um standardisierte Datenmodelle. Zudem wird gefordert, standardisierte Nomenklaturen zur semantischen Beschreibung der ausgetauschten Daten zu nutzen. Somit wird ermöglicht, über den Zugang zu den Daten hinaus, die Struktur der Daten selbst und deren Bedeutung zu verstehen. In kritischen und dynamischen Systemen kann so eine sichere und korrekte Interpretation der Daten gewährleistet werden. Die Datenmodelle müssen daher sowohl die Selbstbeschreibung der Fähigkeiten als auch des aktuellen Zustands der *Service Provider* ermöglichen.

Fokussierung auf die Patientensicherheit In vernetzten Medizingeräteensembles sind die ausgetauschten Daten oft von hoher Kritikalität für die Patientensicherheit, insbesondere im Fall der Fernsteuerung. In vielen Anwendungsbereichen der *SODA* besteht hingegen keine unmittelbare Gefährdung von Menschen, etwa wenn Gerätedaten in der Industrieautomation zur Prozessoptimierung zusammengeführt werden. In anderen Fällen kann der Gefährdung mit anderen Mechanismen, wie etwa Schutzzonen im Arbeitsbereich von Roboterarmen, begegnet werden.

Verglichen mit der *SODA* sind die Sicherheitsanforderungen an die *SOMDA*-basierte Vernetzung somit deutlich höher. Ein gängiges Beispiel ist die *Erst-Fehler-Sicherheit* (*Single Fault Safety*), die für einen Großteil von Fernsteuerungsanwendungen garantiert werden muss. In heutigen Systemen wird diese typischerweise durch eine zweikanalige Datenübertragung sichergestellt. So erfolgt zum Beispiel die Datenübertragung zwischen einem Fußschalter und einem medizinischen Laser häufig über zwei redundante physikalische Verbindungen. Diese Übertragung nutzt ggf. verschiedene Darstellungsformen, wie etwa Invertierungen. Die Realisierung einer *SOMDA* muss folglich Mechanismen bereitstellen, um solche Anforderungen zu erfüllen.

Kontextsensitivität Im medizinischen Umfeld ist zu beachten, dass dieselben Daten in unterschiedlichen Kontexten unterschiedlich zu interpretieren sind. Wird beispielsweise ein Messwert einem falschen Patienten zugeordnet, kann dies fatale Auswirkungen haben. Aber auch die Interpretation eines Vitalparameters bei ein und demselben Patienten kann sehr unterschiedlich ausfallen, je nachdem, an welcher Körperstelle die Messung erfolgte oder welche Aktivitäten kurz vor der Messung vollzogen wurden. Daher verlangt die Eigenschaft der *Kontextsensitivität*, dass in einer *SOMDA* relevante kontextbezogene Informationen zusammen mit den eigentlichen Kerndaten übertragen werden, wenn dies notwendig ist.

Ensemblefähigkeit In vielen Anwendungsszenarien der *SODA* im Industrieumfeld sind sowohl Art, Anzahl, Position etc. der interagierenden Geräte als auch Art und Umfang der ausgetauschten Daten zum Zeitpunkt von Konzeption und Entwicklung bekannt. Typischerweise

[3.1] Während für die Medizingerätevernetzung beide Teilaspekte von Sicherheit, *Safety* und *Security*, von entscheidender Bedeutung sind, konzentriert sich diese Arbeit auf Aspekte der *Safety* (siehe auch Kapitel 1.3).

ändern sich diese während des Betriebs nicht. Im medizinischen Umfeld ist die Dynamik deutlich höher. Ein und dasselbe Gerät wird häufig in ganz verschiedenen Eingriffen/Operationstypen genutzt. Darüber hinaus wechselt die behandelte Patientin mit jeder Operation. Mobile Medizingeräte kommen zudem an verschiedenen Orten zum Einsatz. Damit ändern sich auch ständig die Vernetzungspartner und die ausgetauschten Daten.

Für einen sicheren Betrieb sind somit Gruppierungen notwendig, um die komplexe Kommunikationsstruktur dynamisch zu organisieren. Daher ist eine der spezifischen *SOMDA*-Eigenschaften die *Ensemblefähigkeit*. Diese beschreibt die Fähigkeit, flexible Ensembles, also Teilmengen, von Geräten aus der Menge der zur Verfügung stehenden Vernetzungsteilnehmer zu bilden. *SOMDA*-Ensembles können dabei eine beliebige Komplexität aufweisen: von Funktionseinheiten, die aus zwei Geräten bestehen, bis hin zu komplexen Geräteverbünden für komplizierte medizinische Eingriffe, wie etwa neurochirurgische Operationen in einem Hybrid-OP. *SOMDA*-Teilnehmer können Ensembles dynamisch zur Laufzeit beitreten und diese auch wieder verlassen. Ein typisches Beispiel hierfür sind mobile Röntgengeräte, die in mehreren OP-Sälen genutzt werden. *SOMDA*-Geräteensembles können hierarchisch organisiert werden, sodass aus existierenden Verbünden neue Gruppierungen zusammengesetzt werden können.

Neben der Organisation von komplexen Geräteverbünden können Ensembles dafür genutzt werden, Herausforderungen der Kommunikationsberechtigungen zu lösen. So ist aus Sicherheitsgründen zu gewährleisten, dass die Geräte nur die zu einem bestimmten Eingriff an einem bestimmten Patienten gehörenden Daten und Funktionalitäten austauschen. Wie ein solches Ensemblemanagement mittels *IEEE 11073 SDC* umgesetzt werden kann, wird in Kapitel 6 beschrieben.

Beachtung regulatorischer und zulassungsbezogener Aspekte von Medizingeräten Während die ersten vier *SOMDA*-Eigenschaften einen technischen Fokus haben, betrachtet die letzte Eigenschaft die regulatorischen Anforderungen, die an Medizinprodukte gestellt werden. Die Zulassungsfähigkeit, bzw. die Möglichkeit, ein Medizinprodukt nach den geltenden Gesetzen und Regularien in den Markt zu bringen, beeinflusst die ersten vier Aspekte maßgeblich. Daher kann dieser Aspekt nicht klar von den anderen getrennt werden.

Zur Veranschaulichung diene folgendes Bespiel: Es ist sicherzustellen, dass jedes *Point-of-Care (PoC)*-Medizingerät weltweit eindeutig identifizierbar ist. Hierfür kann etwa die *Unique Device Identification (UDI)* genutzt werden, die auf Initiative der US-amerikanischen *Food and Drug Administration (FDA)* eingeführt wurde und im Zuge der *Medical Device Regulation (MDR)* auch in der EU eingeführt wird. Eine *SOMDA*-Implementierung muss folglich diese eindeutige Identifikation der Medizingeräte in dynamischen Systemen beinhalten und diesen Aspekt in den Fähigkeiten der Geräteselbstbeschreibung im Datenmodell (siehe erste Eigenschaft) und der Datenübertragung berücksichtigen.

3.4 Diskussion

In diesem Kapitel wurden die Prinzipien der *SOA* und deren Spezialisierungen *SODA* und *SOMDA* vorgestellt. Im weiteren Verlauf der Arbeit wird gezeigt, wie diese Konzepte für die Realisierung einer herstellerübergreifenden Medizingeräteinteroperabilität genutzt werden können.

Während das Architekturmuster der *SOA* etabliert und weit verbreitet ist, wurden die Eigenschaften der *SOMDA* erstmals 2018 in dieser Form publiziert [B 7]. Eine intensive Auseinandersetzung in der Fachöffentlichkeit, die über das *OR.NET*-Konsortium hinausgeht, steht daher noch aus.

Es ist festzustellen, dass die beschriebenen Eigenschaften der *SOMDA* weniger scharf sind als etwa die Prinzipien der *SOA*. Dies liegt insbesondere an der Ausrichtung auf einen strikt regulierten Anwendungsbereich und den sich daraus ergebenden Anforderungen. Die fünf formulierten Aspekte

spiegeln den Konsens im *OR.NET*-Konsortium wider. Diese können ggf. weiter ausdifferenziert und verfeinert werden. Darüber hinaus werden Realisierungen in der Praxis zeigen, ob weitere Aspekte zugefügt werden müssen.

4 Herstellerunabhängige Medizingerätevernetzung: IEEE 11073 SDC

In diesem Kapitel wird die textitIEEE 11073 SDC-Normenfamilie vorgestellt. Sie ist die erste bekannte, normierte Umsetzung des Konzepts der *Service-Oriented Medical Device Architecture (SOMDA)* (siehe Kapitel 3). Internationale Bestrebungen, wie etwa aus dem US-amerikanischen *Medical Device „Plug-and-Play“ (MD PnP)*-Projekt (siehe Kapitel 2.4.1), haben bisher nicht den Stand einer technischen Spezifikation oder Norm erreicht.

Der Fokus dieses Kapitels liegt auf den sogenannten *IEEE 11073 SDC*-Kernstandards. Dieser Abschnitt legt damit die technischen Grundlagen für die späteren Kapitel. Gleichzeitig war bzw. ist der Autor der vorliegenden Arbeit an der Entwicklung der vorgestellten Normen beteiligt. Den Abschluss des Kapitels bildet eine Gegenüberstellung der Normenfamilie bzw. des *OR.NET*-Projekts mit den Ergebnissen der internationalen Forschungsprojekte *MD PnP* und *Smart Cyber Operating Theater (SCOT)* .

Die *IEEE 11073 SDC*-Normenfamilie wurde und wird in einem über zehn Jahre andauernden Prozess von mehreren öffentlich geförderten Forschungsprojekten unter geförderter wie assoziierter Beteiligung von Industrieunternehmen entwickelt. Somit sind die in diesem Kapitel vorgestellten Technologien nicht allein dem Verfasser der vorliegenden Arbeit zuzuordnen. Im Zuge seiner Forschungstätigkeit wurde jedoch eine Vielzahl von Aspekten in die Standards aufgenommen. Zudem ist der Verfasser einer der Co-Autoren der *IEEE 11073 SDC*-Standards und hat den Standardisierungsprozess intensiv begleitet und vorangetrieben. Als Grundlage dieses Kapitels dienen einige Publikationen des Autors, die sich mit der *IEEE 11073 SDC*-Familie beschäftigen [B 7, 28, 30]. Darüber hinaus wurden Aspekte der Standards bereits von weiteren beteiligten Partnern veröffentlicht, beispielsweise Schlichting und Pöhlsen [A 103], Pöhlsen et al. [A 118–120], Gregorczyk et al. [A 121, 122], Andersen et al. [B 18, 25]. Ebenso sind die Dissertationen von Pöhlsen [A 123] und Gregorczyk [A 124] zu beachten.

4.1 Die Struktur der IEEE 11073 SDC-Normenfamilie

Die *IEEE 11073 SDC*-Normenfamilie gliedert sich in drei Bereiche. Da diese aufeinander aufbauen, sind sie in Abbildung 4.1 übereinander angeordnet. Die Basis bilden die *Kernstandards* (*Core Standards*), die in blau dargestellt sind. Sie stellen die technische Spezifikation für die interoperable Vernetzung dar. Die vorliegende Arbeit und insbesondere dieses Kapitel beschäftigen sich speziell mit den *Kernstandards*. Sie sind bereits verabschiedet und bereits von der International Organization for Standardization (ISO) anerkannt.

Die *Participant Key Purpose (PKP)*-Standards, der grüne Teil von Abbildung 4.1, beschreiben Anforderungen an die Vernetzungsteilnehmer auf der Basis ihrer Rollen. Als Grundlage dient der *Base PKP* (*IEEE 11073-10700*), der allgemeine Anforderungen definiert. *IEEE 11073-10701* beschäftigt sich mit dem Bereitstellen und Konsumieren von Informationen (*Metric Provisioning*). Speziellere Anforderungen an die *Provider* und *Consumer* von Alarmen spezifiziert *IEEE 11073-10702*. Die spezifischen Regeln und Mechanismen für die Fernsteuerung von Medizingeräten werden in *IEEE 11073-10703* beschrieben. Die *PKPs* können als Prozessstandards angesehen werden. Im Fokus steht das Risikomanagement für Geräteensembles mit unbekannten Teilnehmern. Ein Grundkonzept besteht darin, die medizinische Systemfunktion (*System Function*) in die Beiträge der einzelnen Teilnehmer (*System Function Contribution*) zu zerlegen. Auf dieser Basis werden insbesondere die Verantwortlichkeiten zwischen den Vernetzungsteilnehmern definiert.

Geräte mit einer medizinischen Zweckbestimmung werden *PKP*-Standards für die Zulassung bzw. für das Inverkehrbringen einhalten müssen. Es sei angemerkt, dass nicht zwingend jeder *IEEE 11073 SDC*-Teilnehmer ein Medizinprodukt im Sinne der geltenden Gesetze und Regularien sein muss. Systeme, die beispielsweise allein zur Prozessoptimierung genutzt werden und auf deren

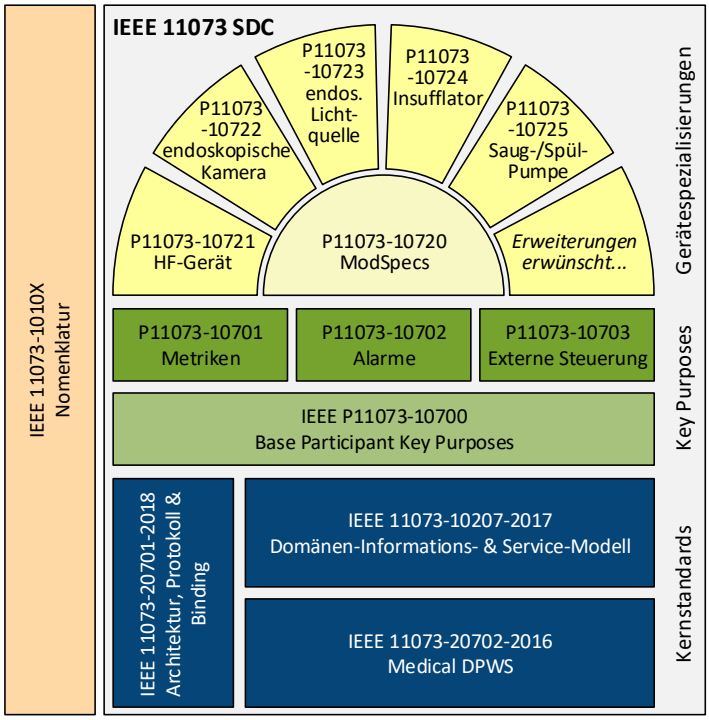

Abbildung 4.1: Übersicht der Teilstandards der *IEEE 11073 SDC*-Normenfamilie. (Abbildung nach *Fig. 1* aus [B 3]. Original © 2019 IEEE.)

Basis keine diagnostischen oder therapeutischen Entscheidungen getroffen werden, müssen folglich diese Ebene der Standardfamilie nicht zwingend erfüllen. Als weiteres Beispiel können Systeme angeführt werden, die Vital- und Geräteparameter für Lehrzwecke zusätzlich zum Videobild in Hörsäle übertragen. Die *PKP*-Standards befinden sich aktuell in der Entwicklung. Während die Zielsetzungen mit der Einreichung der *Project Authorization Requests (PARs)* bei der *IEEE* spezifiziert sind, laufen aktuell die Arbeiten zur Definition der Inhalte.

Die *Kernstandards* und *PKP*-Standards sind unabhängig von konkreten Medizinprodukten. Im Gegensatz dazu beziehen sich die *Gerätespezialisierungen* oder *Device Specializations (DevSpecs)*[4.1] auf konkrete Gruppen von Medizingeräten. In Abbildung 4.1 werden die Normengruppen von unten nach oben immer konkreter und restriktiver. Die *Kernstandards* ermöglichen die Modellierung aller erdenklichen Geräte und erlauben dabei viele Freiheiten bei der Nutzung der definierten Mechanismen. Die *PKP*-Standards schränken diese mit dem Fokus auf eine medizinische Zweckbestimmung weiter ein. Zudem wird auch eine Vielzahl von Anforderungen an das Verhalten von *Service Consumern* gestellt.

Mit dem Fokus auf die Netzwerkrepräsentation beschreiben die *DevSpec*-Standards die Modellierung der Selbstbeschreibung und die Implementierung der Repräsentation des Gerätezustands im Vernetzungskontext. Das Ziel ist es, die Freiheiten der zugrunde liegenden *IEEE 11073 SDC*-Standards so einzuschränken, dass sich verschiedene Geräte von verschiedenen Herstellern desselben Typs so ähnlich repräsentieren und verhalten, dass herstellerübergreifend vernetzte Medizingeräteensembles ermöglicht werden. Während die Grundfunktionalitäten standardisiert werden, sind herstellerspezifische Erweiterungen im Rahmen der *IEEE 11073 SDC*-Spezifikationen explizit erlaubt und erwünscht. Im oberen, gelb dargestellten Teil von Abbildung 4.1 sind die aktuell im Zuge des vom Bundesministerium für Wirtschaft und Energie (BMWi) geförderten Forschungsprojekts *Modular Specialisations for Point-of-Care Medical Devices (PoCSpec)* in der Entwicklung befindlichen Standards zu erkennen: Hochfrequenzchirurgiegeräte, endoskopische Kameras und Lichtquellen, Insufflatoren und

[4.1] Siehe auch Glossareintrag zum Thema *IEEE 11073 SDC Device Specializations*.

Saug-/Spülpumpen (*IEEE 11073-10721* bis *-10725*). Einen Sonderfall stellen die *Module Specifications (ModSpecs)* des *IEEE 11073-10720* Standards dar. In dieser Norm werden Teilspezifikationen zusammengefasst, die von mehreren Geräteklassen genutzt werden können.

Im linken Teil von Abbildung 4.1 sind zusätzlich die Nomenklatur-Standards der *IEEE 11073-1010X*-Serie dargestellt. Diese sind nicht Teil der *IEEE 11073 SDC*-Normenfamilie, ihre Nutzung wird aber normativ vorgeschrieben.

Die *PKPs* und *DevSpecs* befinden sich derzeit im Standardisierungsprozess, sodass sie in Abbildung 4.1 mit dem Präfix „P“ für *Standardization Project* bei der IEEE dargestellt sind. Der Autor dieser Arbeit beteiligt sich intensiv an der Entwicklung dieser Normen.

4.2 Die IEEE 11073 SDC-Kernstandards im Überblick

In diesem Kapitel wird mit den *IEEE 11073 SDC-Kernstandards* eine technische Spezifikation der *SOMDA* beschrieben. Sie bestehen aus drei einzelnen Standards:

- *IEEE 11073-20702* [D 3]: *Medical Devices Communication Profile for Web Services*, kurz *Medical DPWS* oder *MDPWS*.
- *IEEE 11073-10207* [D 2]: Domain Information & Service Model. Dieser Standard ist auch unter dem nicht-normativen Namen *BICEPS*, kurz für *Basic Integrated Clinical Environment Protocol Specification*, bekannt.
- *IEEE 11073-20701* [D 1]: *Service-Oriented Medical Device Exchange Architecture and Protocol Binding*. Umgangssprachlich werden auch die Bezeichnungen *SDC-Glue* oder nur *SDC* genutzt[4.2].

Die Verbindung zwischen den drei Standards wird in Abbildung 4.2 schematisch dargestellt. Der *MDPWS*-Standard (*IEEE 11073-20702*) definiert die Mechanismen zur Datenübertragung und zum dynamischen Finden von Vernetzungsteilnehmern und stellt damit die *grundlegende (foundational) Interoperabilität* her. Dabei ist *MDPWS* vollständig unabhängig von den ausgetauschten Daten, deren Struktur im Standard *IEEE 11073-10207 (BICEPS)* definiert wird (*strukturelle Interoperabilität*). Die Verbindung (*Binding*) zwischen den beiden vorherigen Normen wird vom Standard *IEEE 11073-20701* definiert. Zusätzlich wird in diesem Standard die Gesamtarchitektur beschrieben. Ebenso wird für Aspekte wie Zeitsynchronisation oder *Quality of Service (QoS)* die Nutzung externer Standards spezifiziert. Durch die Nutzung von standardisierten Nomenklaturen, etwa zum semantischen Annotieren von Messwerten und deren Einheiten, wird die *semantische Interoperabilität* hergestellt.

Aufgrund der beschriebenen Trennung zwischen den Daten und der Datenübertragung ist die *IEEE 11073 SDC*-Normenfamilie flexibel für Erweiterungen und damit zukunftssicher. Sollte sich in den kommenden Jahren eine besser geeignete Transporttechnologie entwickeln, so kann ein neuer Standard in die Familie integriert werden. Aus Sicht der Gerätemodellierung und Anwendungsimplementierung würden sich hieraus keine Änderungen ergeben. Für die Integration eines neuen Transportstandards wäre ein neues *Binding* zum *Domänen-Informations- und Service-Modell* notwendig.

4.3 Medical DPWS – IEEE 11073-20702

Der Standard *ISO/IEEE 11073-20702-2016* „Health informatics – Point-of-care medical device communication Part 20702: Medical Devices Communication Profile for Web Services“, kurz

[4.2] Bei der Nutzung der Bezeichnung *SDC* ist darauf zu achten, dass eindeutig klar ist, dass nicht die gesamte *IEEE 11073 SDC*-Normenfamilie gemeint ist, sondern nur der Einzelstandard. Daher wird dir Bezeichnung *SDC* in der vorliegenden Arbeit nicht für den Teilstandard genutzt.

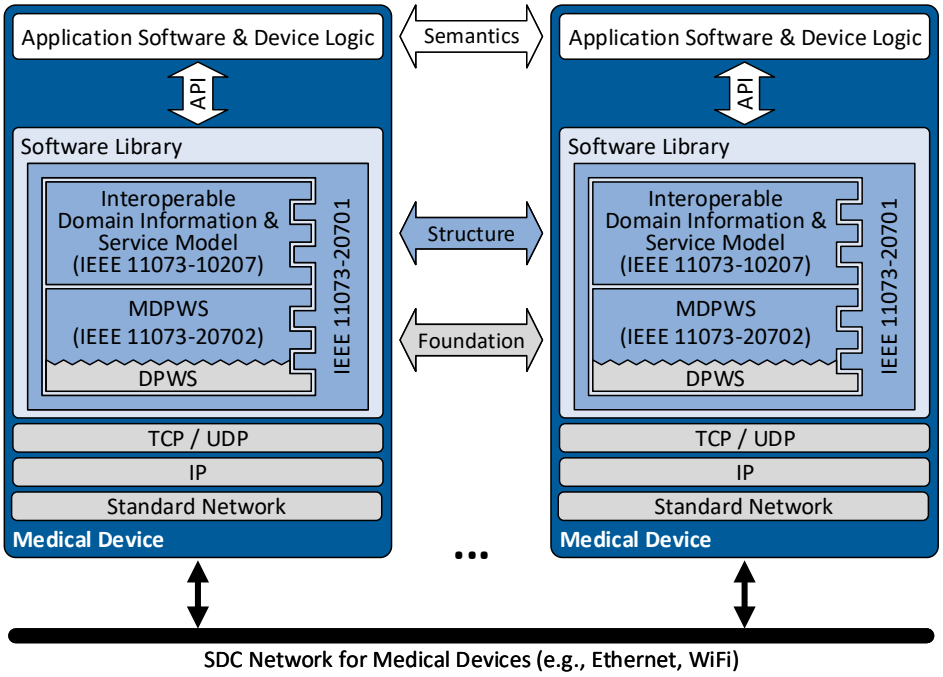

Abbildung 4.2: Verbindung der *IEEE 11073 SDC- Kernstandards*. (Abbildung entspricht *Figure 3* aus [B 7].)

MDPWS oder *Medical DPWS*, realisiert die *grundlegende (foundational) Interoperabilität*. Für die Entwicklung von *MDPWS* wurde auf den existierenden Standard *Devices Profile for Web Services (DPWS)* [A 111] der *Organization for the Advancement of Structured Information Standards (OASIS)* aufgesetzt. *DPWS* ist eine Umsetzung des *Service-Oriented Architecture (SOA)*-Konzepts, speziell für ressourcenbeschränkte, eingebettete Systeme. Von *DPWS* werden Mechanismen zur Geräte-zu-Geräte Kommunikation entsprechend dem Request-Response-Pattern (Anfrage-Antwort-Muster, synchrone Kommunikation) und dem Publish-Subscribe-Pattern (Abonnementprinzip, asynchrone Kommunikation) bereitgestellt (siehe auch Kapitel 3.2).

Auf der Basis des *OASIS*-Standards *WS-Discovery* [A 113] realisiert *DPWS* ebenfalls Mechanismen für das dynamische Finden von Diensten zur Laufzeit (*Dynamic Discovery*). Einerseits wird explizites Suchen, *Explicit Discovery*, ermöglicht, d. h., der *Service Consumer* sucht aktiv *Service Provider* bzw. Services. Andererseits kann implizites Suchen, *Implicit Discovery* genutzt werden, d. h., der *Service Provider* macht sich selbst (unaufgefordert) im Netzwerk bekannt (siehe auch Kapitel 3.2). Durch diese dynamische Suche, zusammen mit den Fähigkeiten zum Austausch von Metainformationen und Schnittstellenselbstbeschreibungen, stellt *DPWS* eine geeignete Grundlage für die Anforderungen an die Vernetzung von dynamischen Medizingeräteverbünden dar.

Die Nutzung des Applikationsschichtprotokolls *DPWS* als Grundlage für *MDPWS* hat weiterhin den Vorteil, dass die Datenübertragung unabhängig vom tatsächlich genutzten Netzwerk ist. Die notwendigen Bedingungen sind die Bereitstellung des *Hypertext Transfer Protocol (HTTP)* und folglich des *Transmission Control Protocol (TCP)* und des *User Datagram Protocol (UDP)* für die Übertragung von *SOAP*-Nachrichten [A 125] (*SOAP-over-HTTP* bzw. *SOAP-over-UDP* [A 126]). Für eine sichere Datenübertragung wird das *Hypertext Transfer Protocol Secure (HTTPS)* genutzt. In medizinischen Umgebungen, wie OP-Saal oder Intensivstation, wird typischerweise *Ethernet* zum Einsatz kommen. Unter der Beachtung entsprechender Sicherheits- und Zuverlässigkeitsmechanismen ist aber auch vom Einsatz drahtloser Technologien wie WLAN (*IEEE 802.11*-Familie) auszugehen. Beispielsweise werden mobile Patientenmonitore, die beim Transport von Patienten im Krankenhaus einsetzt werden, drahtlos angebunden.

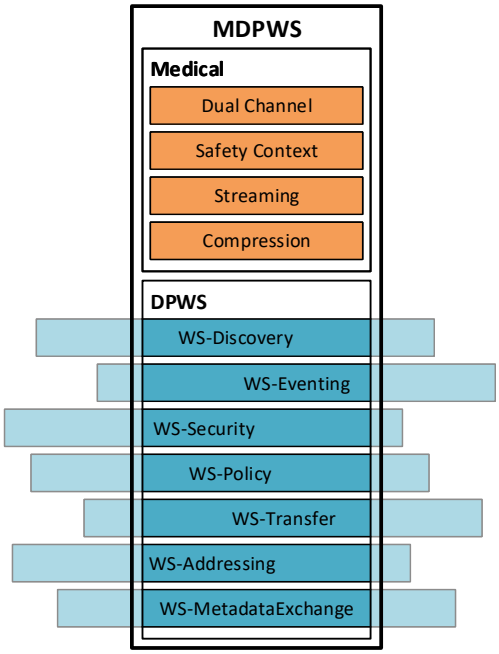

Abbildung 4.3: Beziehungen zwischen *WS-**-Standards, *DPWS* und *MDPWS*-Erweiterungen.

Aufbauend auf *DPWS* spezifiziert *MDPWS* Erweiterungen und Restriktionen, um die speziellen Anforderungen an Medizingeräte und ihre Sicherheitsanforderungen (vor allem *Safety*) zu erfüllen. Abbildung 4.3 veranschaulicht die Beziehungen zwischen den wesentlichen *WS-**-Standards (blau), *DPWS* und *MDPWS* und zeigt die wichtigsten Erweiterungen von *MDPWS* (orange):

- zuverlässige Datenübertragung (*Safe Data Transmission*), bestehend aus zweikanaliger Übertragung (*Dual Channel Transmission*) und *Safety Context*,
- Daten-Streaming und
- komprimierte Datenübertragung (*Compact Transmission*).

Neben den Erweiterungen definiert *MDPWS* auch eine Reihe von Einschränkungen der genutzten Standards. So wird beispielsweise die *HTTP*-Authentifizierung mittels *Client Credentials* zugunsten von *Client*-Authentifizierung mittels *X.509.v3-Zertifikaten* untersagt, während in *DPWS* beide Vorgehensweisen erlaubt sind.

Im Folgenden werden die Grundprinzipien der wichtigsten *MDPWS*-Erweiterungen erläutert. Für weiterführende Informationen sei auf Kasparick et al. [B 28], Schlichting und Pöhlsen [A 103] sowie Pöhlsen et al. [A 120] verwiesen.

4.3.1 MDPWS-Erweiterung zur zuverlässigen Datenübertragung

Der zuverlässige Datentransport ist unerlässlich für vernetzte Medizingeräteensembles. Im Besonderen gilt dies für Fernsteuerungskommandos (*External Control*). Die *Dual Channel Transmission*-Erweiterung ermöglicht eine *Erst-Fehler-Sicherheit* für die Datenübertragung über ein und dasselbe physikalische Transportmedium. Zur Realisierung einer Zweikanaligkeit, unter der Bedingung eines einzigen physikalischen Transportmediums, können eine zeitliche Trennung der Kanäle vorgenommen oder zwei verschiedene Repräsentationen der Daten in derselben Nachricht verschickt werden. Die zeitliche Trennung impliziert ein zustandsbehaftetes Protokoll. Des Weiteren wird eine höhere Netzwerklast und Latenz durch den Austausch mehrerer Nachrichten für dieselbe Information erzeugt. Daher wird für *MDPWS* der zweite Ansatz, die Nutzung von zwei verschiedenen Repräsentationen, verfolgt [A 103, 120].

Der *Service Provider* spezifiziert für eine zweikanalig abgesicherte Datenübertragung einen Mechanismus zur Transformation der Ausgangsdaten in eine kanonische Repräsentation und einen Algorithmus zur Berechnung der Repräsentation des zweiten Kanals (Checksummenberechnung), der auf die kanonische Repräsentation angewandt wird. Zusätzlich wird spezifiziert, welche Informationen entsprechend behandelt werden sollen. Bei der Umsetzung des Verfahrens zur Berechnung bzw. zur Kontrolle der zweiten Repräsentation ist zu beachten, dass die Datenverarbeitungsprozesse wie Serialisierung, Deserialisierung und Parsing eingeschlossen sind. Ein entsprechendes Verfahren zur konkreten Umsetzung schlagen Pöhlsen et al. [A 120] vor.

Die zweite konzeptuelle Erweiterung von *DPWS* zur zuverlässigen Datenübertragung ist der sogenannte *Safety Context*. Er ermöglicht die Einbettung zusätzlicher, kontextbezogener Informationen in den *Header* einer Nachricht. Anwendung wird dieser Mechanismus hauptsächlich im Zusammenhang mit Fernsteuerungskommandos finden. Entsprechend den Sicherheitsanforderungen und dem Risikomanagement kann das zu steuernde Gerät (*Service Provider*) vom steuernden Gerät (*Service Consumer*) verlangen, bestimmte Informationen zusätzlich in den Befehl einzubetten. Sind diese nicht oder nicht korrekt vorhanden, so sollte die Fernsteuerungsanfrage verworfen werden. Das konkrete Verhalten hängt vom Risikomanagement des zu steuernden Gerätes ab. So ist etwa zu bewerten, ob und bis zu welcher Größenordnung Abweichungen im konkreten Anwendungsfall tolerierbar sind oder nicht.

Ein anschauliches Beispiel stellt das Verstellen eines Drucks, etwa an einem Beatmungsgerät oder einer Pumpe/Insufflator, dar. Drücke können in diversen Einheiten angegeben werden, z. B. *Pa*, *bar*, *mmHg*. Für den zuverlässigen Betrieb muss daher sichergestellt werden, dass die Kommunikationspartner bei einer Druckänderung von derselben Einheit ausgehen. Die Einheit eines Parameters wird in der Gerätebeschreibung des *Service Providers* definiert. Geht es um den eigentlichen Wert des Parameters, kann die Einheit aus Effizienzgründen im Allgemeinen weggelassen werden, da sie als bekannt vorausgesetzt werden kann. Daher ist auch bei Verstell-Operationen die Einheit des Wertes im Normalfall nicht Teil des Kommandos (siehe auch Kapitel 4.4.1 für einen tieferen Einblick in die Mechanismen von Gerätebeschreibung und -zustand). In kritischen Fällen bietet der *Dual Channel*-Mechanismus die Möglichkeit, solche Informationen einzubetten, wenn das Risikomanagement dies erfordert. Im Beispiel würde das Beatmungsgerät fordern, dass die Einheit des Zielwertes bei jeder Verstell-Operation im *Safety Context* eingebettet werden muss.

Da verändernde Zugriffe auf ein Medizingerät typischerweise ein höheres Gefährdungspotential für die Patientin aufweisen als der lesende Zugriff, wird diese zusätzliche Absicherung ermöglicht. Ein komplexes Beispiel für die Nutzung des *Safety Contexts* wird in Kapitel 7 beschrieben.

4.3.2 MDPWS-Mechanismen zum Transport von Datenströmen

Ein wichtiger Anwendungsfall in der medizinischen Datenübertagung sind Kurven von Werten (*Waveforms*), wie etwa *Elektrokardiogramm (EKG)*- oder *Elektroenzephalogramm (EEG)*-Kurven. Die visuelle Darstellung der Kurven basiert auf einer Vielzahl von einzelnen Messwerten, die als Datenströme (*Stream*) übertragen werden. *DPWS* nutzt typischerweise das *SOAP-over-HTTP*-Binding, sodass die Datenübertragung *TCP*-basiert erfolgt. Datenströme können mittels der *TCP*-basierten Eventing-Mechanismen realisiert werden. Die Nutzung von *UDP*-Multicast-Nachrichten ermöglicht aber eine effizientere Übertragung von hochfrequenten Daten.

Die *MDPWS*-Mechanismen lassen das Aushandeln beliebiger Protokolle bzw. Transport-Bindings zum Datenstreaming zu. In *MDPWS* wird explizit das *SOAP-over-UDP Multicast Stream Binding* beschrieben, das zur Nutzung für Datenströme empfohlen wird. Es wird dabei auf dem *OASIS*-Standard *SOAP-over-UDP* [A 126] aufgesetzt. Eine *SOAP-over-HTTP*-basierte Übertragung ist aber ebenso möglich.

Es sei kritisch angemerkt, dass die Wahlfreiheit der beiden Mechanismen zu einem Mehraufwand bei der Entwicklung führt, wenn ein System beide Verfahren unterstützen will. Dies gilt insbesondere für universelle *Service Consumer*, die etwa eine zentrale Arbeitsstation im OP-Saal realisieren und zur Laufzeit auf eine Vielzahl von *Service Providern* reagieren müssen, die potentiell sowohl den *SOAP-over-UDP*- als auch den *SOAP-over-HTTP*-Mechanismus implementieren.

Des Weiteren werden derzeit Vor- und Nachteile beider Mechanismen diskutiert: Das *UDP*-basierte Verfahren hat einen geringeren Overhead, was für hochfrequente Daten von Vorteil ist. Andererseits steigt der Aufwand für *Security*-Mechanismen. Diese sind für die *HTTP*-basierte Datenübertragung definiert und müssen für die *IEEE 11073 SDC*-Kommunikation von jedem Teilnehmer implementiert werden. Für *SOAP-over-UDP* besteht einerseits Definitionsbedarf, andererseits müssen zusätzliche Mechanismen implementiert werden. Die Diskussion wird aktuell ergebnisoffen von den Normungsakteuren und implementierenden Firmen geführt.

4.3.3 MDPWS-Mechanismen zur komprimierten Datenübertragung

Die *IEEE 11073 SDC*-Familie sieht für die Datenübertragung die Nutzung des *Extensible Markup Language (XML)*-Datenformats vor. Präzise formuliert: *MDPWS* fordert eine *UTF-8*-kodierte Repräsentation des *XML Information Set*. Dies ist ein logisches Vorgehen, da *SOAP* auf *XML* aufsetzt[4.3]. Sowohl bezogen auf die Datenmenge als auch auf die Verarbeitungszeit weist *XML* einen hohen Overhead auf, verglichen mit anderen Repräsentationsformen [A 127–130], wie etwa *JavaScript Object Notation (JSON)*, *Binary JSON (BSON)*, *Efficient XML Interchange (EXI)*, *FastInfoset* etc.

Wird eine kompakte Datenübertragung benötigt, so sieht der *MDPWS*-Standard die Nutzung von Efficient XML Interchange (EXI) [A 131] vor[4.4]. *EXI* ist ein Standard des *World Wide Web Consortium (W3C)* und wurde speziell für Anwendungen mit leistungsschwachen Geräten entwickelt. *EXI* realisiert eine sehr kompakte Darstellung des *XML Information Set (Infoset)* und wurde mit dem Ziel entwickelt, gleichzeitig die Performanz und die Nutzung der Rechenressourcen zu optimieren. Die Kompression von *EXI* ist dabei verlustfrei.

Das *XML Infoset* wird in einen sogenannten *EXI Stream* überführt. Die strukturgebenden *XML*-Bestandteile werden als sogenannte *EXI Events* repräsentiert, zwischen denen die eigentlichen Nutzdaten in den Strom eingebettet werden. Der Overhead von *XML* wird durch eine möglichst effiziente Binärkodierung minimiert. Die Grundidee besteht darin, dass sowohl für die Bezeichner der strukturgebenden Bestandteile als auch für die Nutzdaten wiederkehrende Inhalte (Zeichenketten) lediglich einmal im *EXI Stream* vorkommen und bei jedem Auftreten eine entsprechende Referenz genutzt wird. Die Zuordnung erfolgt in sogenannten String-Tabellen (*String Tables*). Als Wissensbasis wird eine Sammlung an Grammatiken genutzt, die es ermöglichen, die Länge der Event-Codes anhand ihrer Auftrittswahrscheinlichkeit zu optimieren. Da die Schemata in der Norm *IEEE 11073-10207* (Domäneninformations- und Service-Modell) definiert sind, können diese Informationen als bekannt vorausgesetzt werden. So kann der sogenannte *„schema-informed“*-Modus von *EXI* die Effizienz weiter steigern, da im Schema definierte Bestandteile gar nicht Teil des *EXI Streams* sind und lediglich die Referenzen genutzt werden. Enthält die *XML*-Repräsentation Erweiterungen, die über die Schemata der *IEEE 11073-10207* hinausgehen, so werden diese Informationen entsprechend in den String-Tabellen gespeichert.

4.3 Für die Nutzdaten innerhalb der *SOAP*-Nachrichten könnten auch andere Formate genutzt werden, was für *IEEE 11073 SDC* allerdings irrelevant ist.

4.4 Es sei angemerkt, dass aktuell die Nutzung des *LZ4*-Kompressionsalgorithmus [A 132] diskutiert wird. Dieser verlustfreie Algorithmus wurde von Yann Collet entwickelt. Er basiert auf dem *LZ1* bzw. *LZ77*-Algorithmus von Ziv und Lempel [A 133]. Hinsichtlich Kompressionsrate und Ressourcenverbrauch stellt *LZ4* eine vielversprechende Alternative dar. Die Mechanismen der Einbindung wären etwas anders als bei *EXI*, da es sich bei *LZ4* nicht um eine andere Repräsentation des *XML Infosets* handelt.

Unterstützt ein *MDPWS*-konformer Vernetzungsteilnehmer *EXI*, so *muss* dieser verpflichtend den *„schema-less“*-Modus (alle Informationen sind im *EXI Stream* enthalten) und *sollte* (optional, aber mit hoher Präferenz) ebenso den *„schema-informed“*-Modus unterstützen.

4.3.4 Secure Channel

Der *MDPWS*-Standard nutzt im Wesentlichen die Security-Mechanismen des *DPWS*-Standards. Aufgrund der Wichtigkeit sicherer Kommunikationsverbindungen wird dennoch kurz auf zwei wesentliche Aspekte eingegangen, obgleich *Security* nicht integraler Bestandteil dieser Arbeit ist.

Für eine sichere (*secure*) Kommunikation wird eine *Secure Sockets Layer (SSL)/Transport Layer Security (TLS)*-Verschlüsselung (*HTTPS*) genutzt. Im Standard *IEEE 11073-20701* wird zusätzlich eingeschränkt, dass *TLS 1.2* oder neuer genutzt werden muss.

Wie bereits erwähnt, erfolgt die Authentifizierung mittels *X.509.v3*-Zertifikaten. Die in *DPWS* ebenfalls spezifizierte Nutzung von *Client Credentials* ist ausgeschlossen. Die Fähigkeiten zum Aufbau einer sogenannten *Trust Chain* und zur Bereitstellung von Autorisierungsinformationen sind die entscheidenden Vorteile von *X.509.v3* Zertifikaten.

4.3.5 Interoperabilitätsmechanismen für MDPWS-spezifische Erweiterungen

Für die *MDPWS*-spezifischen Erweiterungen müssen die Interoperabilität und die Plug-and-Play-Fähigkeit gewährleistet werden. Das bedeutet, dass das Bekanntgeben und Aushandeln der Mechanismen zur Laufzeit ermöglicht werden muss.

Grundlegend werden die Mechanismen des *WS-Policy*-Standards [A 115] genutzt. *MDPWS* definiert sogenannte *Assertions* für die spezifischen Funktionalitäten *Streaming* (*mdpws:StreamSource Assertion*), zuverlässige Datenübertragung (*mdpws:SafetyReqAssertion Assertion*) und komprimierte Datenübertragung (*mdpws:Compression Assertion*). Diese Bekanntmachungen der Funktionalitäten in Form von *Assertions* werden direkt in die *Web Services Description Language (WSDL)*-Beschreibung eingebettet, wenn dies gewünscht ist. Gegebenenfalls notwendige, weitergehende Beschreibungen werden Teil der Gerätebeschreibung innerhalb der *Medical Data Information Base (MDIB)* (siehe Kapitel 4.4.1.1)[4.5]. Zusätzlich können sie auch optional in die *WSDL* eingefügt werden.

Wird durch eine *dpws:SafetyReqAssertion* in der *WSDL*-Beschreibung eine zweikanalige Übertragung und/oder ein *Safety Context* durch den *Service Provider* gefordert, so werden zusätzliche Informationen mit Hilfe des *mdpws:SafetyReq*-Elements spezifiziert. Im Fall der *Safety Context*-Nutzung wird hier beispielsweise definiert, welche zusätzlichen Informationen eingebettet werden müssen. Das *mdpws:SafetyReq*-Element wird dann verpflichtend als Erweiterung (*XML Extension*) in die Beschreibung der entsprechenden Fernsteuerungsoperation in die Gerätebeschreibung der *MDIB* eingefügt. Der *Service Consumer* erhält die Informationen somit zur Laufzeit und wird die geforderten Informationen entsprechend in Fernsteuerungskommandos einbetten.

Da ein detaillierteres Wissen über diese Mechanismen für das Verständnis der vorliegenden Arbeit nicht notwendig ist, sei für weiterführende Informationen auf den Standard [D 3] bzw. auf weiterführende Veröffentlichungen [B 28], [A 103, 120] verwiesen.

4.4 BICEPS – IEEE 11073-10207

Die Norm *IEEE 11073-10207* trägt den Titel „Domain Information and Service Model for Service-Oriented Point-of-Care Medical Device Communication“. Häufig ist auch der nicht-normative Name

[4.5] Es sei angemerkt, dass das Einbetten *mdpws:SafetyReq*-Elements in die *MDIB* im Standard *IEEE 11073-20701* beschrieben wird. Zum besseren Verständnis wird hier aber das gesamte Konzept erläutert.

BICEPS (Basic Integrated Clinical Environment Protocol Specification) gebräuchlich. Entsprechend der vorgestellten Definition der Interoperabilitätsstufen (siehe Kapitel 2.1) ermöglicht dieser Standard die strukturelle Interoperabilität. Es werden sowohl die Struktur der zu übertragenden Daten als auch die Möglichkeit zur Interaktion mit den bereitgestellten Informationen und Funktionalitäten definiert. Der Standard ist so spezifiziert, dass er die Anforderungen an heutige und zukünftige Medizingeräte, ungeachtet des Herstellers oder des Einsatzbereiches, erfüllt.

Zum besseren Verständnis sei angemerkt, dass die Bezeichnung Domänen-Informations-Modell, engl. *Domain Information Model (DIM)*, zwar Teil des Titels des *IEEE 11073-10207* Standards ist, diese im Dokument selbst aber keine Rolle spielt. Es werden vier Teilmodelle beschrieben: Das „*Participant Model*“, das „*Communication Model*“, das „*Discovery Model*“ und das *„Extension Model“*. Das *Participant Model* definiert die Selbstbeschreibung der Vernetzungsteilnehmer (*Participants*) und ist in einen Beschreibungs- und einen Zustandsteil gegliedert. Im *Communication Model* werden sowohl die Dienste (*Service Model*) als auch die Nachrichten (*Message Model*) beschrieben. Zusätzlich spezifiziert das *Discovery Model*, dass Vernetzungsteilnehmer ihr Eintreten und Verlassen in die Vernetzung entsprechend bekanntgeben (*implizites Discovery*) und dass Teilnehmer aktiv gesucht werden können (*explizites Discovery*). Das *Extension Model* ermöglicht es, im *Participant Model* und im *Message Model* an vielen Stellen Erweiterungen einzufügen, die über den Standard hinausgehen.

4.4.1 Participant Model

Dieser Teil des *DIMs* ist vom „klassischen“ *IEEE 11073-10201 DIM* [A 134] abgeleitet. Die vorgenommenen Erweiterungen und Veränderungen ergeben sich aus den Anforderungen an Systeme, in denen eine Vielzahl von Medizingeräten mit einer Vielzahl von anderen Medizingeräten, auf der Basis einer *SOA*, verbunden sind (Multi-Point-to-Multi-Point). Der Fokus der „klassischen“ Standards innerhalb der *IEEE 11073*-Normenfamilie liegt hingegen auf der direkten Verbindung zweier Kommunikationspartner (Point-to-Point), insbesondere für den Teil der persönlichen Medizingeräte.

Das *Participant Model* unterteilt sich in zwei Bereiche: *MdDescription* und *MdState*. Die *MdDescription* ist die Selbstbeschreibung der Medizingeräte mit ihren Fähigkeiten. Der *MdState* beschreibt den aktuellen Zustand. In der Modellierung eines Medizingerätes werden beide Teile in der sogenannten *Medical Data Information Base (MDIB)* zusammengefasst. Die Selbstbeschreibung enthält verhältnismäßig statische Informationen, wie die Art und Einheit von Messwerten. Im Gegensatz dazu ändern sich die Inhalte der Zustandsbeschreibung (hoch-)dynamisch, wie etwa der aktuelle Wert einer Messung.

4.4.1.1 Gerätebeschreibung (MdDescription) Die Selbstbeschreibung der Medizingeräte ist in Form eines Baums mit einer (maximalen) Tiefe von vier organisiert, wie es beispielhaft im linken Teil von Abbildung 4.4 dargestellt ist. Die Modellierung in Form eines Baums (*Containment Tree*) wurde aus dem „klassischen“ *IEEE 11073-10201 DIM* übernommen. Vom Wurzelknoten zu den Blattknoten werden die Knoten der einzelnen Ebenen wie folgt bezeichnet: *Medical Device System (MDS)*, *Virtual Medical Device (VMD)*, *Channel* und *Metric*. *MDS*, *VMD* und *Channel* können jeweils beliebig viele (0 bis ∞) *Kindknoten* der entsprechenden Art besitzen. Somit ist es beispielsweise möglich, dass ein *Channel* ohne *Metric* existiert, eine *Metric* kann aber nur als *Kindknoten* eines *Channels* auftreten.

Die Elemente im Beschreibungsbaum heißen *Descriptor*. Jeder *Descriptor* verfügt über ein sogenanntes *Handle*. Dies ist ein innerhalb der *MDIB* eindeutiger Bezeichner. Das *Handle* hat keinerlei semantische Bedeutung. Es dient ausschließlich der Identifizierbarkeit und Referenzierbarkeit von Elementen. Obwohl die Gerätebeschreibung im Vergleich zum Gerätezustand relativ statisch ist, verfügt jeder *Descriptor* über einen Versionszähler (*Version Counter*). Die sogenannte *Descriptor*

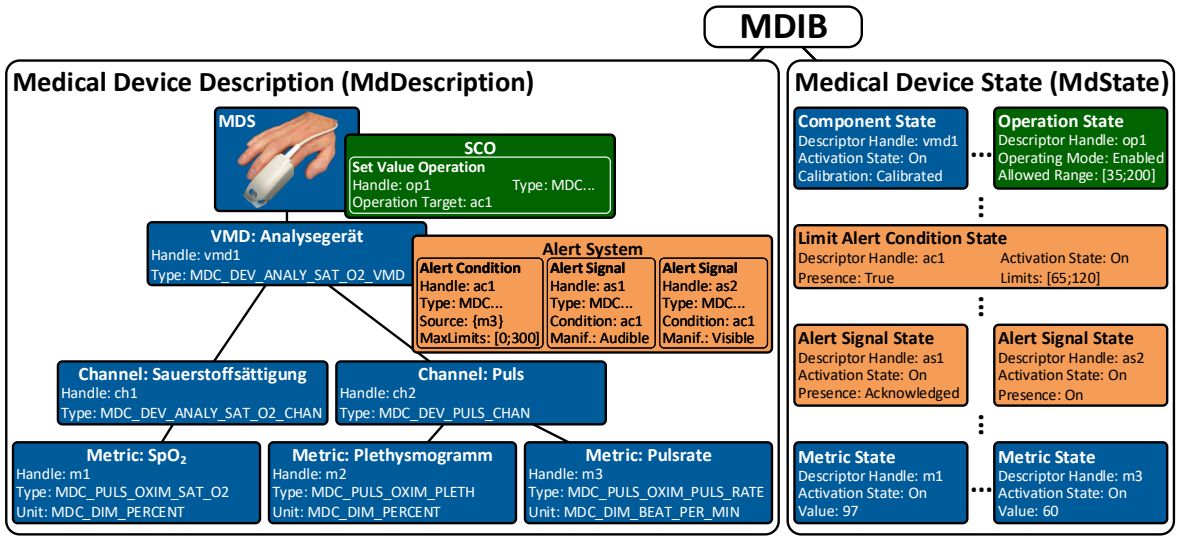

Abbildung 4.4: Beispielhafte und reduzierte Darstellung einer *Medical Data Information Base (MDIB)*: Links Gerätebeschreibung (*MdDescription*) und rechts Gerätezustand (*MdState*). Farbliche Gruppierung: blau: Baumstrukturelemente und Metriken; grün: Fernsteuerung; orange: Alarme. (Abbildung nach *Figure 4* aus [B 7])

Version wird immer dann erhöht, wenn sich etwas an der Beschreibung des Elements ändert[4.6]. Dies ermöglicht eine sichere Interpretation der Informationen, auch wenn Änderungen zur Laufzeit auftreten.

Im Folgenden wird auf die einzelnen Elemente des Beschreibungsbaums detailliert eingegangen.

Metriken Messwerte, Berechnungen, Konfigurationsparameter, Zustände etc. werden als *Metriken* (*Metrics*) modelliert. Metriken bilden die Blattknoten des Beschreibungsbaums. Es können verschiedene Klassen von Metrikbeschreibungen genutzt werden:

- numerische Metriken (`NumericMetricDescriptor`)
- zeichenkettenbasierte Metriken (`StringMetricDescriptor`)
- zeichenkettenbasierte Metriken mit fest vorgegebenen Werten (`EnumStringMetricDescriptor`)
- datenstrombasierte bzw. Array-basierte Metriken (`RealTimeSampleArrayMetricDescriptor` bzw. `DistributionSampleArrayMetricDescriptor`)

Jede Metrik wird semantisch über ihren Typ (*Type*) beschrieben[4.7]. Diese Beschreibung erfolgt durch *Coded Values*, die zu bestimmten Nomenklaturen gehören. Ein *Coded Value* setzt sich aus einem *Coding System* (der Nomenklatur, bzw. einer Referenz auf diese) und dem *Code* (Bezeichner des zu beschreibenden Sachverhalts) zusammen. Die normativ zu nutzende Nomenklatur für die *IEEE 11073 SDC*-Familie ist die *IEEE 11073-1010X*-Serie, z. B. *IEEE 11073-10101* [A 135]. Am Beispiel eines Pulsoximeters (siehe Abbildung 4.4) werden etwa die *Codes* 149.530 (entspricht

4.6 Die *Descriptor Version* ist ein optionales Attribut. Ist es nicht vorhanden, wird der Wert „0“ angenommen (*Implied Value*). Dieser Mechanismus optionaler Attribute, für die beim Nichtvorhandensein ein bestimmter Wert anzunehmen ist, wird in der *IEEE 11073-10207* häufig genutzt. Einerseits führt dies zu einer Reduzierung der Größe der *MDIB*. Andererseits sorgt dies teilweise für Verwirrung beim Lesen der Norm, da entsprechende Attribute nur unter der Bedingung optional sind, dass der *Implied Value* zutreffend ist.

4.7 Generell verfügt jedes beschreibende Element über das *Type*-Element. Stellvertretend für alle Elemente wird hier das Typenkonzept am Beispiel der Metriken erläutert.

2::18.458 bzw. dem symbolischen Bezeichner `MDC_PULS_OXIM_PULS_RATE`)[4.8] für die Herzfrequenz, oder 150.456 (2::19.384; `MDC_PULS_OXIM_SAT_O2`) für die Sauerstoffsättigung, genutzt.

Für die Interpretierbarkeit von Metriken ist zusätzlich das Wissen über die Einheit von entscheidender Bedeutung. Einheiten werden, ebenso wie der Typ einer Metrik, durch *Coded Values* beschrieben. Im gegebenen Beispiel entspricht der *Code* 264.864 (4::2.720; `MDC_DIM_BEAT_PER_MIN`) der Einheit „Schläge pro Minute“ bzw. der *Code* 262.688 (4::544; `MDC_DIM_PERCENT`) der Einheit „Prozent (%)“.

Auszugsweise seien im Folgenden einige weitere beschreibende Eigenschaften von Metriken genannt, die die sichere Interpretierbarkeit und Nutzung der Informationen ermöglichen. Für eine vollständige Beschreibung sei direkt auf den Standard verwiesen [D 2].

- Kategorie: Messung (*Measurement*), Berechnung (*Calculation*), Einstellung (*Setting*), Voreinstellung (*Presetting*), Vorschlag (*Recommendation*)
- Verfügbarkeit: kontinuierlich (*continuous*) oder unterbrochen (*intermittent*)
- Bestimmungsmethode (*Derivation Method*) und -periode (*Determination Period*)
- Maximale Mess- und Berechnungs-/Verarbeitungsverzögerung (`MaxMeasurementTime` und `MaxDelayTime`)
- Maximale Nutzbarkeit des Metrikwertes (`LifeTimePeriod`)
- Sicherheits-/Kritikalitätsklassifikation (*Intended Use*, `SafetyClassification`): Einordnung, für welche Nutzung die Information vorgesehen ist, von informationell (`Inf`, nicht für klinische Funktionen gedacht) bis hin zu hochkritisch (`MedC`, Nutzung der Informationen bzw. eine Entscheidung auf der Basis dieser Informationen kann erheblichen Schaden hervorrufen)
- Auflösung (*Resolution*; nur für numerische Metriken)
- Technisch möglicher Wertebereich (*Technical Range*; nur für numerische Metriken)

Insbesondere für die Modellierung von komplexen Medizingeräten ist die Beschreibung von Beziehungen zwischen Metriken und anderen Elementen des Beschreibungsbaums wichtig. Zur Veranschaulichung dient folgendes, stark vereinfachtes Beispiel (siehe auch Abbildung 4.5): Ein Beatmungsgerät verfügt unter anderem über die Parameter Beatmungsform, Soll-Beatmungsdruck (Zielwert) und Ist-Beatmungsdruck (tatsächlich gemessener Wert). Die zugehörigen Metriken sind in Abbildung 4.5 blau dargestellt. Die ersten beiden Parameter werden jeweils als Metrik der Kategorie *Setting* modelliert, während der Ist-Beatmungsdruck der Kategorie *Measurement* oder ggf. *Calculation* zugeordnet wird. Diese Beziehung nach dem *Soll-Ist-Pattern* kann in der Metrikbeschreibung abgebildet werden. Hierfür wird das Element *Relation* genutzt. Im Beispiel würde im *Descriptor* der Metrik des Soll-Beatmungsdrucks mittels einer *Relation* modelliert werden, dass dieser einen Effekt auf den Ist-Beatmungsdruck hat.

Eine *Relation* wird durch ihre Art (*engl. Kind*) und die Referenzen auf die potentiell betroffenen Elemente des Beschreibungsbaums beschrieben. Falls notwendig, kann die *Relation* über einen *Coded Value* semantisch beschrieben werden.

[4.8] Exkurs: *Coded Values* der *IEEE 11073-1010X* Nomenklaturserie: Die *IEEE 11073*-Nomenklaturen sind in sogenannten *Partitionen* aufgebaut. *Codes* für Metriken befinden sich beispielsweise in *Partition* Nummer 2, *Codes* für Einheiten in *Partition* Nummer 4 etc. Innerhalb der *Partitionen* befinden sich die sogenannten *kontextsensitiven Term Codes*. Die Darstellung erfolgt häufig in der Form *Partitionsnummer*::*kontextsensitiver Term Code*, z. B. 2::18.458, für Pulsrate, gemessen von einem Pulsoximeter. Der *kontextfreie Nomenklatur-Code*, dessen Nutzung für die *IEEE 11073 SDC*-Normenfamilie vorgeschrieben ist, berechnet sich aus 2^{16} * Partitionsnummer + kontextsensitiver Term Code, im Beispiel $2^{16} * 2 + 18.458 = 149.530$. Zusätzlich existiert ein symbolischer Bezeichner, der sogenannte *Reference Identifier (RefId)*, der zu einem gewissen Maße als menschenlesbar bezeichnet werden kann, im Beispiel `MDC_PULS_OXIM_PULS_RATE`.

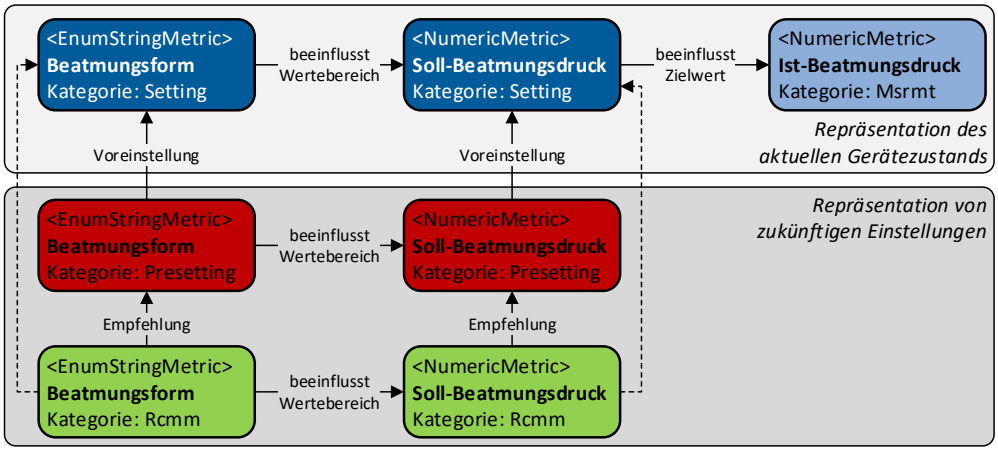

Abbildung 4.5: Zusammenspiel von Metriken verschiedener Kategorien und *Soll-Ist-Pattern*. Die Pfeile repräsentieren Beziehungen der Metriken zueinander, die mittels des Elements *Relation* modelliert werden. Die gestrichelten Pfeile beschreiben eine alternative Modellierungsmöglichkeit, bei der die *Vorschlagsmetriken* (grün) direkt in die *Einstellungsmetriken* (blau) übernommen werden. Abkürzungen: Rcmm – Recommendation; Msrmt – Measurement.

Eine Änderung der Beatmungsform (Metrik mit dem *Handle* `m1`) kann eine Änderung des einstellbaren Wertebereichs des Soll-Beatmungsdrucks (Metrik mit dem *Handle* `m2`) nach sich ziehen. Eine solche Beeinflussung sollte ebenfalls für Vernetzungsteilnehmer bekannt gemacht werden. Wie beschrieben, wird hierfür das Element *Relation* im *Metric Descriptor* genutzt.

Im gegebenen Beispiel könnte durch das Verstellen einzelner Parameter ein inkonsistenter Gerätezustand entstehen, wie etwa ein Soll-Beatmungsdruck, der nicht zur verstellten Beatmungsform passt. Solche Inkonsistenzen können mit Hilfe zusätzlicher Metriken der Kategorie *Presetting* vermieden werden, wie sie in Rot in Abbildung 4.5 veranschaulicht sind. *Presetting*-Metriken ermöglichen es, dass Änderungen der verschiedenen Parameter so vorgenommen werden können, dass sie zunächst keinen Einfluss auf das Geräteverhalten haben. Wurden für alle Parameter die *Presettings* wie gewünscht angepasst und ggf. vom Gerät validiert, kann die Menge der *Presettings* gemeinsam als neue *Settings* übernommen werden. Erst in diesem Moment haben die Änderungen Einfluss auf das Geräteverhalten.

Innovative Geräte können darüber hinaus dem Benutzer Vorschläge für Konfigurationen unterbreiten, die etwa mit einer hohen Wahrscheinlichkeit einen besseren Behandlungserfolg der Patientin versprechen. Solche Vorschläge können mittels Metriken der Kategorie *Recommendation* beschrieben werden (siehe grün gekennzeichneten Bereich in Abbildung 4.5). Die Beziehung zwischen einer Einstellungsmetrik (*Setting*), einer zugehörigen Voreinstellungsmetrik (*Presetting*) und einer ggf. zugehörigen Vorschlagsmetrik (*Recommendation*) sollten somit ebenfalls in der Gerätebeschreibung abgebildet werden. Die beschriebenen Abhängigkeiten sind im Übrigen unabhängig davon, ob diese Einstellungen nur lokal am Gerät vorgenommen werden können oder ob die Möglichkeit der Fernsteuerung durch einen Vernetzungspartner (*Service Consumer*) besteht.

Weitere Elemente des Beschreibungsbaums Die *Channels* bilden die Elternebene der Metrikknoten. *Channels* sind logische oder physiologische Gruppierungen von Metriken. Mehrere *Channels* können wiederum zu einem *Virtual Medical Device (VMD)* gehören. Den Wurzelknoten des Beschreibungsbaums bildet das *Medical Device System (MDS)*, unter dem sich die *VMDs* befinden. Das *MDS* ist eine Abstraktion des Medizingerätes als Gesamtsystem. Ein sehr einfaches Beispiel ist in Abbildung 4.4 anhand eines Pulsoximeters dargestellt. Geräte von wesentlich höherer Komplexität sind etwa Patientenmonitore. *VMDs* repräsentieren Submodule des Medizingerätesystems. Beispielsweise hat ein Patientenmonitor unter anderem je ein *VMD* als Abstraktion des Pulsoximeters, der Blutdruckmessung (mit *Channels* für den invasiv und nicht-invasiv gemessenen Blutdruck), des Thermometers,

des EKGs etc. [A 136]. Einfache Systeme wie ein reines Pulsoximeter kommen hingegen mit lediglich einem *VMD* aus, da eine weitere Unterteilung in Submodule aufgrund der geringen Komplexität nicht sinnvoll ist.

Eine eindeutige Definition, was ein Subsystem eines Medizingerätes ist, und damit als *VMD* oder *Channel* zu modellieren ist, existiert nicht. Daher wird es für einheitliche Modellierungen von komplexen Medizingeräten eine Konsensbildung unter den verschiedenen Herstellern geben müssen, die etwa in standardisierten Gerätespezialisierungen (*Device Specializations*) münden sollten.

Trotz der Bezeichnung Beschreibungsbaum (*Containment Tree*) ist es möglich, dass innerhalb einer *MDIB*, bzw. der zugehörigen Gerätebeschreibung, mehrere *MDS*-Wurzelknoten existieren. Graphentheoretisch gesehen handelt es sich dann um einen *Wald* aus Beschreibungsbäumen. Dies ist zwar technisch möglich, aber in der Realität nicht wahrscheinlich.

Alarme Eine entscheidende Funktionalität von Medizingeräten sind Alarme. Entsprechend der *IEEE 11073-10207* Norm werden Alarme in sogenannten Alarmsystemen (*Alert Systems*)[4.9] zusammengefasst. Diese Objekte können auf der Ebene von *MDS* oder *VMD* in die Gerätebeschreibung eingefügt werden. Ein Alarm unterteilt sich in zwei funktionale Teilbereiche:

- Alarmbedingung (*Alert Condition*): Beschreibt, welche Bedingung eintreten muss, damit ein Alarmzustand gegeben ist.
- Alarmsignal (*Alert Signal*): Beschreibt, wie das Vorliegen eines Alarmzustands dem Nutzer signalisiert wird.

Es wird also strikt zwischen der Detektion und der Signalisierung eines Alarms unterschieden. Alarmbedingungen können physiologische oder technische Aspekte abbilden (*Alert Condition Kind*), sie besitzen eine Priorität (*Priority*) und eine Klassifikation der Sicherheitskritikalität bzw. der intendierten Nutzung (*SafetyClassification*). Alarmbedingungen beziehen sich in der Regel auf Elemente der Gerätebeschreibung. Diese Beziehung zur Ursache des Alarms (*Source*) wird über *Handle*-Referenzen auf die entsprechenden Objekte modelliert. Im Beispiel in Abbildung 4.4 referenziert die Alarmbedingung mit dem *Handle* „`ac1`“ (linkes Element im orangen Kasten der Gerätebeschreibung) die Metrik mit dem *Handle* „`m3`“ (also die Pulsrate) als Ursache. Über die semantische Beschreibung mittels eines *Nomenklatur-Codes* hinaus können mithilfe des Elements *CauseInfo* weitere Angaben zu möglichen Gründen für den Alarm und entsprechende Gegenmaßnahmen in die Beschreibung der Alarmbedingung aufgenommen werden.

Eine Spezialform der Alarmbedingungen sind die Grenzwertalarmbedingungen (*Limit Alert Conditions*). Durch die Angabe von Grenzwerten wird ein Bereich definiert, außerhalb dessen die Alarmbedingung erfüllt ist. In der Beschreibung werden hierfür zunächst die maximal möglichen Grenzwerte angegeben. Im Beispiel von Abbildung 4.4 ist dies das Intervall von 0 bis 300. Über die Beziehung zur Ursache des Alarms ist implizit die Semantik (Pulsrate) und Einheit (Schläge pro Minute) der Grenzwerte gegeben.

Alarmsignale zeigen die Präsenz einer Alarmbedingung an. Das Referenzieren erfolgt über das *Handle* der Alarmbedingung. Es gibt drei verschiedene Erscheinungsformen für Alarmsignale (*Alert Signal Manifestation*): sichtbar, hörbar und haptisch. Dementsprechend können mehrere Alarmsignale einer Alarmbedingung zugeordnet werden. Abbildung 4.4 zeigt beispielhaft die Zuordnung eines akustischen und eines visuellen Alarmsignals (mit den *Handles* „`as1`“ bzw. „`as2`“) zu der Alarmbedingung mit dem *Handle* „`ac1`“ (siehe oranger Kasten im linken Beschreibungsteil der *MDIB*). Alarmsignale werden darüber hinaus durch eine Reihe weiterer Aspekte beschrieben. Exemplarisch seien die Beschreibung verschiedener zeitlicher Verzögerungseigenschaften für die Erzeu-

[4.9] Für eine Definition des Begriffs „Alert“ siehe auch Glossareintrag zum Thema Alert.

gung des Alarmsignals sowie die Quittierbarkeit und das *Latching*-Verhalten[4.10] genannt. Verzögerungen zwischen dem Erkennen der Alarmbedingung und dem Erzeugen des Alarmsignals sind für eine Reihe von medizinischen Anwendungsfällen sinnvoll, um beispielsweise kurzfristig auftretende Fehlalarme, etwa durch Bewegungen des Patienten und damit verbundene Messungenauigkeiten, zu vermeiden.

Neben dem Transport von Alarmen werden die beschriebenen Mechanismen auch für die Kommunikation von Hinweisen und Notifikationen an die Nutzer (*Advisory Signal*) genutzt. In Kapitel 9 wird detailliert auf die Nutzung der *IEEE 11073 SDC*-Alarmfunktionalitäten für klinische, verteilte Alarmierungssysteme eingegangen.

Möglichkeiten der Fernsteuerung (External/Remote Control) Die bisher beschriebenen Mechanismen spezifizieren die Informationen, die ein Medizingerät für den lesenden Zugriff bereitstellt. Die *IEEE 11073 SDC*-Vernetzung ermöglicht zusätzlich auch die Fernsteuerung (*External* bzw. *Remote Control*)[4.11] von Gerätefunktionalitäten. Soll der Gerätezustand von einem entfernten Vernetzungspartner (*Service Consumer*) veränderbar sein, so werden entsprechende Operationen im *Service and Control Object (SCO)* modelliert. Das *SCO* kann als Teil des *MDSs* oder *VMDs* in die Beschreibung eingebunden werden.

Grundsätzlich lassen sich Setzoperationen (*Set Operations*) und Prozedurauslöseoperationen (*Activate Operations*) unterscheiden. *Set Operations* verändern bestimmte Parameter des Medizingerätes, etwa Metriken oder Alarmgrenzen. *Activate Operations* veranlassen hingegen die Ausführung einer beliebig komplexen Aktion des Medizingerätes. Beispiele für einfache Anwendungsfälle sind relative Änderungen von Parametern (z. B. Erhöhung der Intensität einer Lichtquelle auf die nächste Stufe) oder das Ein- und Ausschalten von Gerätefunktionalitäten (z. B. Spültätigkeit einer Pumpe). Ebenso können komplexe Vorgänge ausgelöst werden, etwa das Laden neuer, globaler Konfigurationssets, das (teil-)automatisierte Durchführen von diagnostischen oder therapeutischen Vorgängen, (teil-)automatisierte Dokumentationsvorgänge etc.

Entsprechend ihrer Funktionalität hat jede Operation ein Ziel (*Operation Target*), auf das sie sich bezieht. Die Beschreibung erfolgt über eine *Handle*-Referenz auf das entsprechende Element, z. B. eine Metrik, eine Alarmbedingung bis hin zu einem gesamten (Sub-)System (*VMD* oder *MDS*). Im Beispiel von Abbildung 4.4 referenziert die Operation zum Verstellen von Grenzwerten mit dem *Handle* „`op1`“ (grün gekennzeichnet) die Alarmbedingung mit dem *Handle* „`ac1`“. Somit können mit Hilfe dieser Operation die Alarmgrenzen verändert werden, die wiederum die Pulsrate überwachen.

Auszugsweise seien einige weitere beschreibende Eigenschaften von Operationen benannt: die Klassifikation der Sicherheit bzw. der intendierten Nutzung (*Safety Classification*) oder die maximale Bearbeitungszeit. Insbesondere für *Activate Operations* sind zwei weitere Eigenschaften von Interesse: Mithilfe der Auslösungsablaufzeit (*Invocation Effective Timeout*) kann spezifiziert werden, wie lange der Effekt einer *Activate Operation* anhält. Zusätzlich kann mittels der Wiederauslösbar-Eigenschaft (*Retriggerable*) spezifiziert werden, ob eine erneute Auslösung der *Activate Operation* zu einem Zurücksetzen des *Invocation Effective Timeouts* führt. Zur Verdeutlichung soll das folgende stark vereinfachte Beispiel dienen: Aus Gründen des Risikomanagements sollte die Dauer der Aktivierung einer medizinischen Fräse durch eine *Activate Operation* begrenzt sein, um etwa ein unkontrolliertes Rotieren des Fräskopfs im Fall eines Netzwerkausfalls zu vermeiden. Folglich wird ein *Invocation Effective Timeout* definiert. Da eine unterbrechungsfreie Nutzung der Fräse im Normalbetrieb ermöglicht werden muss, wird die *Activate Operation* als *Retriggerable* modelliert. Damit kann ein *Service Consumer* die *Activate Operation* vor dem Erreichen des Timeouts erneut aufrufen und so die Sicherheitsabschaltung der Fräse verhindern. Die sichere Fernauslösung von kritischen Gerätefunktionalitäten wird ausführlich in Kapitel 7 behandelt.

[4.10] Zum Verständnis dieser Eigenschaften sei auf die Beschreibung der Alarmzustände in Kapitel 4.4.1.2 verwiesen.
[4.11] Siehe auch Glossareintrag zum Thema Fernsteuerung.

Kontexte Die in diesem Kapitel bisher thematisierten Beschreibungselemente spezifizieren das Medizingerät an sich und seine Fähigkeiten. Im Gegensatz dazu beschreiben Kontexte (*Contexts*) die Beziehung des Medizingerätes zu seiner Umwelt. Da sich Kontexte immer auf das gesamte Gerät beziehen, werden sie als Teil des *MDS* beschrieben. Die verschiedenen Kontexte werden im *System Context* zusammengefasst:

Ortskontext (Location Context) Beschreibt den Ort des Gerätes.

Ensemblekontext (Ensemble Context) Ermöglicht Gruppierungen mehrerer Medizingeräte, z. B. Endoskopieturm, Verbund aus Endoskop und Lichtquelle etc. Im *OR.NET*-Projekt wurde der sogenannte *Session Context* entwickelt, der als eine Form des *Ensemble Contexts* umgesetzt wird (siehe Kapitel 6).

Hilfsmittelkontext (Means Context) Bildet die Beziehung zwischen dem Medizingerät und zweckdienlichen Hilfsmitteln ab. Diese Hilfsmittel können physikalisch, z. B. eine Stromversorgung oder ein Haltearm, aber auch virtuell, z. B. Softwareoptionen oder -lizenzen, sein. Im Gegensatz zum *Ensemble Context* werden medizinisch nicht-aktive Hilfsmittel adressiert.

Anwenderkontext (Operator Context) Ermöglicht die Beschreibung, welche(r) Nutzer mit dem Gerät interagieren. Dies ist etwa für Dokumentationszwecke oder das automatisierte Laden von Konfigurationen in Abhängigkeit der Nutzerin von Relevanz.

Patientenkontext (Patient Context) Repräsentiert die Patientin, die dem Medizingerät zugeordnet ist. Zudem können relevante Informationen wie die Patientendaten (*Patient Demographic Data*) und klinische Risiken enthalten sein.

Workflowkontext (Workflow Context) Ermöglicht die Beschreibung einzelner Arbeitsschritte während der medizinischen Behandlung/Diagnostik. Die Granularität kann sich dabei stark unterscheiden. So ist eine Modellierung des gesamten Vorgangs, bis hin zur Unterteilung in feingranulare Teilarbeitsschritte, vorstellbar. Die Modellierung und Erkennung von medizinischen Workflows stellt dabei ein eigenes Forschungsfeld dar [A 137, 138].

Insbesondere die letzten zwei aufgelisteten Kontextarten ermöglichen eine engere Verzahnung der derzeit typischerweise stark getrennten Welten der medizinischen Informationssysteme und Medizingeräte. In heutigen OP-Sälen müssen häufig händisch Namen und Identifikationsnummern von Patientinnen eingegeben werden. Dies ist ein zeitaufwändiger und fehleranfälliger Prozess. Der automatisierte Austausch von Kontextinformationen ermöglicht es, diese Informationen direkt aus den Informationssystemen zu beziehen. Ebenso wird eine (teil-)automatisierte Dokumentation erleichtert, sodass auch der Informationsrücktransport in die klinischen Informationssysteme ermöglicht wird. *IEEE 11073 SDC* bildet hier nur die für die Gerätevernetzung relevante Teilmenge der Informationen ab. Ziel von *IEEE 11073 SDC* ist es nicht, die Mächtigkeit und Aussagekraft von etablierten Standards, wie etwa *Digital Imaging and Communications in Medicine (DICOM)* mit seinen sogenannten *Worklists*, zu erreichen oder mit dem parallel entstehenden *Health Level 7 (HL7) Fast Healthcare Interoperability Resources (FHIR)*-Standard zu konkurrieren [B 10].

4.4.1.2 Gerätezustand (*MdState*)

Der zweite Teilbereich der *MDIB* ist der Gerätezustand (*MdState*). Dieser ist im rechten Teil von Abbildung 4.4 exemplarisch dargestellt. Der *MdState* modelliert den aktuellen Zustand des Medizingerätes, beispielsweise den aktuellen Wert einer Metrik. Für jedes Element der Gerätebeschreibung (*MdDescription*) existiert ein[4.12] Zustand in der Gesamtmenge des Gerätezustands. Die Zuordnung erfolgt über die Referenz auf das *Handle* des *Descrip-*

[4.12] Mit Ausnahme des Konzepts von sogenannten *Multistates* besteht zwischen der Beschreibung und dem Zustand eine bijektive Abbildung, das heißt, dass für jeden Zustand genau eine Beschreibung existiert und umgekehrt. Im Fall von *Multistates* können einer Beschreibung beliebig viele Zustände zugeordnet werden. *Multistates* werden lediglich für die Kontextzustände (*Context States*) genutzt, um beispielsweise eine Abfolge von Arbeitsschritten mithilfe des *Workflow Contexts* abbilden zu können.

tors. Aufgrund dieser engen Verbindung ist es nicht notwendig, statische, beschreibende Informationen als Teil des Zustands zu modellieren. Beispielsweise sind der Typ und die Einheit einer Metrik relativ statisch oder gar unveränderbar im Lebenszyklus des Medizingerätes und sind daher nicht als Teil des Zustands modelliert. Der aktuelle Wert der Metrik kann sich hingegen in sehr kleinen Zeitintervallen ändern. Die Gerätebeschreibung muss vom *Service Consumer* lediglich zum Beginn der Interaktion mit dem *Service Provider*, und im seltenen Fall einer Beschreibungsänderung, abgerufen werden. Hingegen muss eine Zustandsänderung sehr häufig kommuniziert werden. Durch die Trennung zwischen Beschreibung und Zustand wird also eine Datensparsamkeit erreicht, ohne dabei semantische Informationen einzubüßen. Zusätzlich wird der Verarbeitungsaufwand für die Geräte reduziert.

Durch die Abbildung der Zustände auf die Elemente im Beschreibungsbaum kann auf eine Baumstruktur im *MdState* verzichtet werden. Der Gerätezustand ist eine Menge von Teilzuständen. Im Folgenden wird zunächst beispielhaft auf allgemeine Konzepte und auf eine Auswahl von spezifischen Zuständen eingegangen.

Grundlegende Konzepte von Zustandsobjekten Alle Zustände verfügen über einen Versionszähler (*Version Counter*), die sogenannte *State Version*. Ändert sich der Zustand, so wird dieser Zähler inkrementiert. Damit lassen sich beispielsweise veraltete Informationen erkennen, ohne dass eine hochpräzise Zeitsynchronisation zwischen den Vernetzungspartnern vorhanden sein muss. Ist für einen entsprechenden Zustand bereits eine höhere Versionsnummer bekannt, so ist implizit klar, dass ein Problem aufgetreten ist. Die Nutzung der *State Version* zur Realisierung von sicheren Fernauslösungen wird in Kapitel 7 beschrieben.

Die Mehrheit der Zustände verfügt über ein Attribut, das die Funktionsbereitschaft repräsentiert. Entsprechend der Art des Zustands unterscheiden sich die Bezeichnung des Attributs und sein Wertebereich; das Konzept ändert sich hingegen nicht. Für *MDSs*, *VMDs*, *Channels* und *Metriken* wird das Attribut *Activation State* vom Typ *Component Activation* genutzt. Es gibt an, ob die Komponente/-Metrik arbeitet (`On`), sich im `Stand By` befindet, sich im Prozess der Initialisierung befindet oder aus anderen Gründen gerade nicht unmittelbar benutzt werden kann (`Not Ready`), sich im Prozess des Herunterfahrens befindet (`Shutdown`), inaktiv ist (`Off`) oder sich in einem Fehlerzustand befindet (`Failure`). Für Alarme wird beispielsweise das Attribut *Activation State* vom Typ *Alert Activation* genutzt, das lediglich den Wertebereich betriebsbereit (`On`), nicht betriebsbereit (`Off`) oder pausiert (`Paused`) umfasst. Entsprechend dieser Funktionsbereitschaft sind die weiteren Eigenschaften des Zustands zu interpretieren. Beispielsweise ist der Wert einer Mess-Metrik nur dann verlässlich, wenn sie sich im *Activation State* `On` befindet. Mit Hilfe dieses Attributes kann die Funktionsbereitschaft einzelner Metriken, aber auch ganzer Teilkomponenten gesteuert werden. Einen gängigen Anwendungsfall hierfür stellt das Ein- und Ausschalten einzelner Module dar. So kann beispielsweise das Pulsoximeter-Modul (*VMD*) eines Patientenmonitors komplett deaktiviert (z. B. `Off` oder `Stand By`) werden, wenn die Funktionalität nicht benötigt wird. Die Deaktivierung impliziert bzw. erfordert, dass alle Kind-Elemente des Beschreibungsbaums, inkl. der Alarme, ebenfalls deaktiviert werden.

In einigen Fällen konkretisieren bzw. spezialisieren die Eigenschaften eines Zustands die Eigenschaften der zugehörigen Beschreibung. Beispielsweise werden in der Beschreibung technisch mögliche Grenzwerte definiert, während im Zustand die tatsächlich aktiven Grenzwerte beschrieben werden. Im Beispiel in Abbildung 4.4 wird das technisch mögliche Intervall für die der Alarmbedingung mit dem *Handle* „`ac1`“ von 0 bis 300 definiert. Das tatsächlich momentan überwachte Limit ist von 65 bis 120 (siehe orange Boxen in Abbildung 4.4). Offensichtlich dürfen die Eigenschaften des Zustands die der Beschreibung nicht verletzen.

Metrikzustand (*Metric State*) Innerhalb des Zustands einer Metrik ist der aktuelle Wert (*Metric Value*) von besonderem Interesse. Da der Wert allein aber nur bedingt aussagekräftig ist, sind im

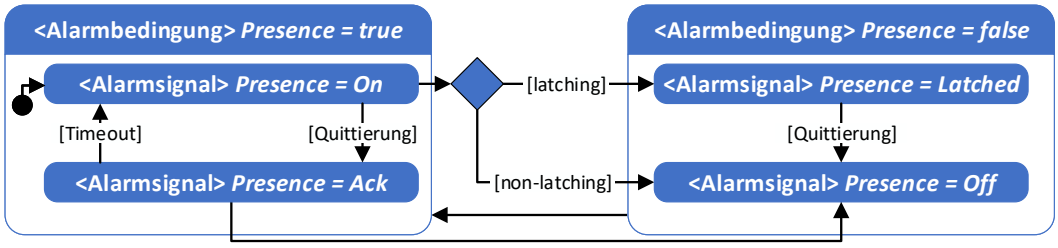

Abbildung 4.6: Zustandsdiagramm zur Beschreibung des Zusammenspiels der *Presence*-Zustände von Alarmbedingung und zugehörigem Alarmsignal.

Metric Value-Objekt weitere Informationen enthalten. Dementsprechend wird die Qualität des ermittelten Wertes modelliert. Zwingend ist die Validität (z. B. *Valid*, *Invalid*, *Measurement Ongoing*, *Calibration Ongoing*, *Overflow/Underflow* etc.) anzugeben. Zusätzlich können der Aufnahmemodus und ein Qualitätsindikator (*Quality Indicator*) angegeben werden. Darüber hinaus können verschiedene Zeitstempel und Annotationen hinterlegt werden, sodass der Wert interpretierbar ist. In Abbildung 4.4 ist der aktuelle Wert der Metrik mit dem *Handle* „`m1`“ 97 und der Wert der Metrik mit dem *Handle* „`m3`“ 60. Über die Referenz zur Gerätebeschreibung sind Semantik, Einheit etc. klar.

Alarmzustand (*Alert State*) Für Alarmbedingungen und -signale gibt das *Presence*-Attribut Aufschluss darüber, ob die Alarmbedingung gerade erfüllt ist oder nicht, und ob ein Alarmsignal gerade propagiert wird oder nicht. Im Folgenden soll zum besseren Verständnis kurz die Beziehung zwischen Alarmbedingung, Alarmsignal und Nutzerbestätigungen beschrieben werden (siehe auch Abbildung 4.6).

Wird eine Alarmbedingung erfüllt, so werden die zugehörigen Alarmsignale erzeugt. An diesem Zeitpunkt haben die *Presence*-Attribute des Alarmbedingungszustands und des Alarmsignalzustands den Wert `true` bzw. `On`. Typischerweise besteht für Alarmsignale die Möglichkeit der Quittierung durch die Nutzerin. Erfolgt eine Quittierung, geht die *Presence* des Alarmsignals für eine definierte Zeit in den Zustand `Acknowledged`. Der Zustand der Alarmbedingung bleibt von diesem Vorgang unbeeinflusst.

Des Weiteren sei der folgende Fall betrachtet: Der *Presence*-Zustand von Alarmbedingung und -signal hat jeweils den Wert `true` bzw. `On`. Zur Verdeutlichung der Relevanz sei angenommen, dass das Alarmsignal vom klinischen Personal bisher nicht wahrgenommen wurde. Im Verlauf ändert sich der von der Alarmbedingung observierte Zustand so, dass die Alarmbedingung nicht mehr erfüllt ist. Entsprechend ändert sich der *Presence*-Wert zu `false`. Für das Alarmsignal bestehen zwei Möglichkeiten: a) Das Alarmsignal hat keine *Latching*-Eigenschaft. Entsprechend geht die *Presence* des Alarmsignals ebenfalls in den Zustand `Off` über. Das klinische Personal erfährt folglich nicht, dass ein Alarmzustand vorlag. b) Für das Alarmsignal ist die *Latching*-Eigenschaft aktiviert. In diesem Fall geht der *Presence*-Zustand nicht in `Off`, sondern in `Latched` über. `Latched` beschreibt also den Zustand, dass ein Alarm aktiv signalisiert wurde, zum aktuellen Zeitpunkt die Alarmbedingung aber nicht mehr erfüllt ist. Damit erfolgt die Signalisierung auch nicht mehr aktiv, aber die Information ist implizit gegeben, dass zwischen der letzten Bestätigung und dem aktuellen Zeitpunkt eine aktive Signalisierung erfolgte.

Aufgrund der begrifflichen Nähe sei darauf hingewiesen, dass das Attribut *Activation State* der Alarmzustände nicht mit dem Attribut *Presence* verwechselt werden darf. Während der *Activation State* die Funktionsbereitschaft der Komponente selbst beschreibt, gibt die *Presence* Aufschluss über den überwachten bzw. angezeigten Alarm. Folgerichtig hat die *Presence* aber nur genau dann eine Aussagekraft, wenn der Wert des *Activation States* `On` ist.

Kontextzustände (*Context States*) Kontexte beschreiben die Beziehung des Medizingerätes zu seiner Umwelt. So kann der Patientenkontext etwa einen Kerndatensatz der Patientendaten enthalten, der Workflowkontext kann Informationen zum Auftrag (*Order*) und klinische Informationen/Risiken transportieren. Kontexte nehmen in ihrem Lebenszyklus verschiedene *Context Association*-Zustände an: nicht assoziiert, prä-assoziiert, assoziiert und dis-assoziiert, wobei nicht zwingend jeder Zustand durchlaufen werden muss.

4.4.1.3 *MBID*-Versionierung

Über die *Version Counter* der Beschreibungs- und Zustandselemente hinaus verfügt die *MDIB* selbst über ein zusätzliches Tripel aus *MDIB Version*, *Sequence ID* und *Instance ID*. Diese *MDIB Version Group* sichert etwa die Informationskonsistenz und Interpretierbarkeit. Das ist beispielsweise in Fällen von Neustarts wichtig, auch wenn diese unvorhergesehen oder im Fehlerfall durchgeführt werden und durch Mechanismen wie `Bye`-Nachrichten nicht erkannt werden können.

Die *MdibVersion* ist ein Zähler, der – etwas vereinfacht beschrieben – mit jeder Änderung in der *MdDescription* oder dem *MdState* um 1 erhöht wird. Setzt ein *Service Provider* die *MdibVersion* zurück, so muss eine neue, eindeutige *Sequence ID* erzeugt werden. Dies ist z. B. typischerweise der Fall, wenn das Gerät neugestartet wird. Auch nach einem Fehlerfall muss verpflichtend eine neue *Sequence ID* erzeugt werden. Beim Zurücksetzen der *MdibVersion* müssen alle *Version Counter* der Beschreibungs- und Zustandselemente ebenfalls zurückgesetzt werden. Die optionale *Instance ID* wird immer genau dann erhöht, wenn sich die *Sequence ID* geändert hat. Somit kann sie genutzt werden, um eine Reihenfolge von *Sequence IDs* herzustellen.

Eine neue *Sequence ID* ist eine entscheidende Information für die beteiligten *Service Consumer*. Sie ist das Signal, dass sich Änderungen in Gerätebeschreibung und -zustand vollzogen haben können, die über nicht über entsprechende *Events* oder die Überwachung des entsprechenden *Version Counter* von *Descriptor* oder *State* detektierbar sind. Da sich bei einer Änderung der *Sequence ID* selbst die *Handles* geändert haben können, müssen *Service Consumer* in diesem Fall die gesamte *MDIB* abrufen, um mit der potentiell veränderten Gerätebeschreibung die Zustände sicher interpretieren zu können. Für eine detaillierte Erläuterung sei direkt auf die Standards verwiesen [D 1, 2]

4.4.2 Communication Model

Das Kommunikationsmodell (*Communication Model*) besteht aus zwei Teilen, dem Nachrichtenmodell (*Message Model*) und dem Servicemodell (*Service Model*). Das *Service Model* beschreibt, welche Interaktionsmöglichkeiten zum lesenden oder verändernden Zugriff auf die *MDIB* eines Vernetzungsteilnehmers bestehen. Die entsprechend ausgetauschten Nachrichten werden im *Message Model* definiert, das damit auch die Verbindung zwischen *Service* und *Participant Model* herstellt.

4.4.2.1 Service Model

Das *Service Model* definiert neun verschiedene Services, die in Abbildung 4.7 dargestellt sind. Jedem Service sind verschiedene Operationen zugeordnet, sodass diese Operationen nach ihren Funktionalitäten logisch gruppiert sind. Die feingranulare Unterteilung in eine Vielzahl von Services hat den Vorteil, dass nur genau die Services implementiert werden müssen, die zum Funktionsumfang des jeweiligen Gerätes passen. Lediglich der *Get Service* ist mandatorisch zu implementieren.

Get Service Der *Get Service* ermöglicht den *Service Consumern* den Abruf von Informationen aus der *MDIB* nach dem *Request-Response-Pattern*. Es existieren Operationen zum Abrufen der gesamten *MDIB* (*GetMdib*), dem Beschreibungs- und dem Zustandsteil (*GetMdDescription* bzw. *GetMdState*). Im Fall der zwei letztgenannten Operationen kann durch die Angabe von *Handle*-Referenzen auch auf einzelne Elemente/Teilbäume lesend zugegriffen werden.

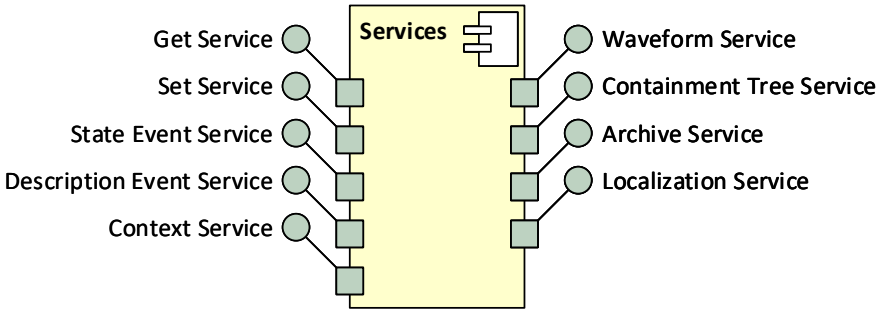

Abbildung 4.7: Services des *IEEE 11073-10207* Service-Modells. (Abbildung nach *Figure 12* aus [D 2].)

Set Service Der *Set Service* gruppiert die Operationen, mit denen der Zustandsteil der *MDIB* durch *Service Consumer* verändert werden kann, um so eine Fernsteuerung des Medizingerätes in seiner Rolle als *Service Provider* zu ermöglichen. Es existieren Setzoperationen für die Werte von Metriken, *SetValue* (für numerische Werte) und *SetString* (für Zeichenketten), und Operationen zum Verändern des gesamten Zustands (sämtliche Elemente und Attribute des Zustands). Des Weiteren gibt es *Activate*-Operationen zum Auslösen vom Funktionalitäten, die über die Veränderung durch das Setzen einzelner Zustände hinausgehen. Zusätzlich enthält der *Set Service* den `OperationInvokedReport`. Dieser informiert den *Service Consumer* nach dem *Publish-Subscribe-Pattern* über den Abarbeitungszustand[4.13] der Operation.

State* und *Description Event Service Diese beiden Services beinhalten ausschließlich Operationen, die den *Service Consumer* über Änderungen des Gerätezustands bzw. der Gerätebeschreibung informieren. Dies erfolgt nach vorherigem Abonnieren nach dem *Publish-Subscribe-Pattern*. Für *State Events* gibt es jeweils periodische (zeitgetriebene) und episodische (durch eine Änderung ausgelöste) Reports. Separate Operationen bestehen für Metrik-, Alarm-, Komponenten- und Fernsteuerungsoperationszustände. Eine Ausnahme bildet der `SystemErrorReport`, der lediglich episodisch versandt wird. Dasselbe gilt für *Description Events*.

Context Service Dieser Service gruppiert die Interaktionsmöglichkeiten mit dem *System Context* des Medizingerätes. Hierzu zählen Operationen zum Lesen und zum Verändern der Kontextzustände sowie zum Abonnieren von periodischen bzw. episodischen Eventreports. Zum Lesen stehen zusätzlich Filtermechanismen zur Verfügung. Kontextinformationen dürfen nur zwischen autorisierten Vernetzungspartnern, unter der Nutzung des verschlüsselten *MDPWS Secure Channels* (siehe Kapitel 4.3.4), ausgetauscht werden.

Waveform Service Der Zustand von datenstrombasierten Metriken (*Real Time Sample Array Metric State*) wird über diesen Service übertragen. Wie in Kapitel 4.3.2 diskutiert, kann das Binding sowohl zum in *MDPWS* definierten *UDP*-basierten Streaming-Mechanismus als auch zum *TCP*-basierten Eventing-Mechanismus genutzt werden.

Containment Tree Service Insbesondere zum gezielten Durchsuchen der Gerätebeschreibung kann der *Containment Tree Service* genutzt werden. Die `GetMdDescription`-Operation des *Get Services* liefert die gesamte Gerätebeschreibung oder entsprechende angefragte Teilbäume

[4.13] Der Abarbeitungszustand (*Invocation State*) einer Operation kann folgende Werte annehmen: *Waiting* (Anfrage erhalten, aber die Bearbeitung hat noch nicht begonnen), *Started* (Bearbeitung hat begonnen), *Finished* (Bearbeitung wurde erfolgreich beendet), *Finished with Modification* (Bearbeitung wurde beendet, aber der gewünschte Zielzustand konnte nicht erreicht werden, sodass der bestmögliche/nächstgelegene Zustand als Zielzustand genutzt wurde), *Cancelled Manually* (Abbruch durch den Benutzer), *Cancelled* (Abbruch durch den *Service Provider*) und *Failed* (Bearbeitung ist fehlgeschlagen). Welche konkrete Reaktion ein Medizingerät auf Fernsteuerungsanfragen zeigt, die beispielsweise nicht erreicht werden können, hängt vom Risikomanagement des jeweiligen Herstellers ab. So kann beispielsweise eine Abarbeitung in Teilen vorgenommen werden und mit *Finished with Modification* quittiert werden oder nicht vorgenommen werden und entsprechend *Failed* zurückgegeben werden.

zurück. Im Gegensatz dazu liefert die Operation `GetContainmentTree` lediglich die Art des Knotens, den Type und Referenzen auf den Elternknoten und die Kindknoten; die Operation `GetDescriptor` liefert genau die Beschreibung des angefragten Knotens des Beschreibungsbaums (ohne Kindknoten). Durch die geringe Menge an Informationen ermöglicht die Nutzung dieses Services ein effizientes Durchsuchen der Gerätebeschreibung. Dies ist beispielsweise sinnvoll, wenn nach dem *Discovery*-Prozess mehrere potentielle *Service Provider* zur Auswahl stehen, aber ein Medizinprodukt mit sehr spezifischen Anforderungen, z. B. einem bestimmten Messwert mit einer bestimmten Qualität, benötigt wird.

Archive Service *Service Provider* können vergangene Gerätebeschreibungen und -zustände speichern. Über den *Archive Service* können *Service Consumer* auf diese Daten unter der Angabe eines Zeitbereichs und/oder eines Bereichs von *Descriptor* bzw. *State Versions* zugreifen.

Localization Service Menschenlesbare Beschreibungen können als Text in verschiedenen Sprachen direkt in der *MDIB* hinterlegt werden. Dies führt bei komplexen, gut dokumentierten Geräten mit mehreren angebotenen Sprachen zu sehr hohen Datenmengen. Alternativ können mehrsprachige Texte als Referenz hinterlegt werden. Mit dieser Referenz kann über den *Localization Service* gezielt der Text in der gewünschten Sprache abgerufen werden.

4.4.2.2 Message Model Das Nachrichtenmodell (*Message Model*) definiert die Nachrichten, die zwischen *Service Consumer* und *Service Provider* auf Basis der Nutzung von Serviceoperationen (siehe Kapitel 4.4.2.1) ausgetauscht werden. Die Zuordnung zwischen Serviceoperationen und Nachrichten erfolgt auf der Basis einer einfachen Namenszuordnungskonvention. So wird beispielsweise einer nach dem *Request-Response-Pattern* genutzten Serviceoperation `X` für die Anfrage (*Request*) die gleichnamige Nachricht aus dem *Message Model* `msg:X`[4.14] zugeordnet. Die zugehörige Antwortnachricht (*Response*) hat den Namen `msg:XResponse`. Z. B.: Zur Serviceoperation `GetMdib` gehören die Nachrichten `GetMdib` und `GetMdibResponse`. Für Serviceoperationen nach dem *Publish-Subscribe-Pattern* werden der Operation `X` die Notifizierungsnachrichten `msg:XReport`, bzw. `msg:EpisodicXReport` und `msg:PeriodicXReport`, falls episodische und periodische Benachrichtigungen existieren, zugeordnet. Für *Streaming*-Operationen erfolgt die Namenszuordnung ohne Suffix (Serviceoperation `X`; Nachricht `msg:X`).

4.4.3 Extension Model

Die Beschreibungsmöglichkeiten des *IEEE 11073-10207* Standards sind so konzipiert und umgesetzt, dass heutige und zukünftige Medizingeräte adäquat modelliert und im Netzwerk repräsentiert werden können. Insbesondere durch die Nutzung der *Coded Values* zur Spezifizierung der einzelnen Beschreibungselemente sind die *IEEE 11073 SDC*-Kernnormen unabhängig davon, welche Geräte später konkret umgesetzt werden sollen. Stößt ein Hersteller an die Grenzen der Aussagefähigkeit des Standards, so können *Extension*-Elemente eingefügt werden. Diese sind für alle Deskriptoren, Zustände und Nachrichten möglich. Während *IEEE 11073-10207* grundsätzlich die Möglichkeit der Erweiterung durch *Extensions* gibt, verbietet der Standard deren Nutzung, wenn der semantische Inhalt durch Mittel transportiert werden kann, die im Standard vorhanden sind. Diese mandatorische Nutzungspflicht erhöht die herstellerübergreifende Interoperabilität.

Aufgrund der strikten Trennung zwischen dem Transportmechanismus, beschrieben im *MDPWS*-Standard (*IEEE 11073-20702*), und den transportierten Daten, kommt dem *Extension*-Mechanismus noch eine zweite Aufgabe zu. *Extensions* werden ebenso dafür genutzt, transportspezifische Informationen in die *MDIB* einzubetten. Werden etwa Mechanismen der zuverlässigen Datenübertragung des *MDPWS*-Standards für Fernsteuerungsoperationen benötigt (siehe Kapitel 4.3.1), so wird

[4.14] Der Namensraum (*Namespace*) `msg` ist für das *Message Model* spezifiziert. URI: http://standards.ieee.org/downloads/11073/11073-10207-2017/message.

das *Extension*-Element des *Operation Descriptors* genutzt, um das *Safety Requirement* einzubetten. Derselbe Mechanismus wird genutzt, um die *Retrievability*[4.15] einer Metrik zu beschreiben, da sie ebenfalls eine Eigenschaft des Transportmechanismus ist und nicht der Daten selbst.

Des Weiteren kann der *Extension*-Mechanismus für standardisierte Erweiterungen anderer Normen oder späterer Versionen dieser Norm genutzt werden, ohne die Abwärtskompatibilität zu verlieren.

4.5 Architektur und Binding – IEEE 11073-20701

Die dritte Norm der *IEEE 11073 SDC*-Kernstandards trägt den Namen „Health Informatics – Device Interoperability – Part 20701: Service Oriented Medical Device Exchange Architecture and Protocol Binding“. Der erste Teil, die Architekturbeschreibung, geht nicht wesentlich über das *Reference Model for Service Oriented Architecture* [A 108] der *OASIS* hinaus (siehe auch Kapitel 3). Im Hauptteil definiert der Standard die Verbindung (*Binding*) zwischen dem Domäneninformations- und Service-Modell (*IEEE 11073-10207*) und dem *MDPWS*-Transportmechanismus (*IEEE 11073-20702*). Darüber hinaus wird die Nutzung der Spezifikationen *Network Time Protocol (NTP)* [A 139, 140] zur Zeitsynchronisation und *Differentiated Services (DiffServ)* [A 141, 142] zur Sicherung von *QoS*-Eigenschaften der Datenübertagung definiert.

Es handelt sich bei dem Standard nicht nur um die Definition eines reinen *Bindings*, es werden vielmehr zusätzlich eine Reihe von Konkretisierungen, Verschärfungen, aber auch Abschwächungen der in den Standards *IEEE 11073-10207* und *-20702* definierten Anforderungen vorgenommen.

4.5.1 Binding: MDPWS und BICEPS

Das Binding innerhalb der *IEEE 11073 SDC*-Normenfamilie erfolgt für die verschiedenen Bereiche des *IEEE 11073-10207* Standards (*BICEPS*). Im Folgenden wird auf ausgewählte Aspekte der einzelnen *BICEPS*-Teilmodelle eingegangen[4.16].

Ein Aspekt des *Participant Model Bindings* ist die Verbindung der *MDPWS*-Mechanismen zur sicheren Datenübertragung (siehe Kapitel 4.3.1) mit den Gerätebeschreibungsmechanismen des *BICEPS*-Standards (siehe Kapitel 4.4.1.1). Wenn ein *Service Provider* für bestimmte Fernsteuerungsoperationen *Safety Requirements* hat, definiert er ein entsprechendes `mdpws:SafetyReq`-Element. Dieses Element spezifiziert, welche zusätzlichen Sicherheitsmechanismen vom *Service Consumer* ergriffen werden müssen, wie etwa die Bereitstellung von Informationen im *Safety Context* und/oder Datenübertragung im *Dual Channel*-Modus. Hierfür wird das `mdpws:SafetyReq`-Element direkt in die Beschreibung der Fernsteuerungsoperation (`pm:AbstractOperationDescriptor`) in Form eines Erweiterungsknotens (*Extension*) eingebettet. So sind Interoperabilität und Plug-and-Play-Fähigkeit gewährleistet.

Für eine sichere (*secure*) Vernetzung ist die vertrauenswürdige Identifizierung der Vernetzungspartner eine entscheidende Voraussetzung. Daher wird definiert, wie bestimmte Informationen aus den *X.509.v3*-Zertifikaten (*MDPWS*) für die sogenannten *Instance Identifier* der Vernetzungspartner (*BICEPS*) genutzt werden.

Für die *Service Operationen* des *IEEE 11073-10207 Service Models* wird ein *Binding* zu *MDPWS*-Operationen definiert. Dies schließt die Abbildung der zugehörigen Nachrichten des *BICEPS Message Models* auf die *MDPWS*-Nachrichten, bzw. deren *Types* als *Input* und *Output Messages* der *MDPWS*-Operationen mit ein. Ebenso wird die Gruppierung in verschiedene *MDPWS Port Types* spezifiziert. Im Wesentlichen handelt es sich um folgende Abbildungen:

[4.15] Die *Retrievability* beschreibt, in welcher Form eine Metrik zur Verfügung gestellt wird: per *Get*-Anfrage, periodischen bzw. episodischen Benachrichtigungen oder als Datenstrom (*Stream*).

[4.16] Für weiterführende und detailliertere Informationen sei direkt auf den Standard *IEEE 11073-20701* [D 1] verwiesen.

- *BICEPS Service* → *MDPWS Port Type*
- *BICEPS Service Operation* → *MDPWS Operation*
- *BICEPS Request* und *Response Messages*, wie sie im *BICEPS Message Model* definiert sind (siehe Kapitel 4.4.2.2) → *MDPWS Input/Output Messages*

Für die Zuordnung zu *MDPWS Hosted Services* werden nur teilweise normative Aussagen getroffen, die beispielsweise fordern, dass einige Services unter einem *MDPWS Hosted Service* zusammengefasst werden müssen. Vorgaben etwa für zu nutzende Bezeichnungen der *Hosted Services* werden in *IEEE 11073-20701* nicht definiert.

Weiterhin wird ein *Binding* für die verschiedenen Nachrichtentypen bzw. -austauschpatterns definiert. Für die Übertragung von *BICEPS*-Nachrichten wird die Nutzung des *MDPWS*-Datentransfers (z. B. *SOAP-over-UDP*, *SOAP-over-HTTP*) spezifiziert, ebenso wie die Nutzung der *MDPWS*-Mechanismen zur Nachrichten-Serialisierung. Zudem werden die Mappings für die Austauschpattern *Request-Response*, *Publish-Subscribe* und *Streaming* definiert. Darüber hinaus werden das *implizite* und *explizite Discovery* des *BICEPS Discovery Models* auf die entsprechenden Mechanismen von *MDPWS* abgebildet.

4.5.1.1 Selektives Discovery In einem Geräteensemble mit vielen Teilnehmern ist es sinnvoll, den *Discovery*-Prozess und die Auswahl des gewünschten Vernetzungspartners möglichst einfach zu gestalten. Die Anzahl der gefundenen Geräte sollte möglichst gering gehalten werden. Hierfür werden mehrere Möglichkeiten des selektiven Suchens von *Service Providern* definiert. Dies reduziert einerseits die Anzahl der Nachrichten im *MDPWS Discovery*-Prozess, andererseits wird die Auswahl des richtigen Vernetzungspartners auf der Basis der Informationen der *BICEPS*-Selbstbeschreibung vereinfacht, da die Menge der zu durchsuchenden Beschreibungen geringer ist.

In *IEEE 11073-20701* werden drei Verfahren definiert, mit denen ein *Service Provider* grundlegende Fähigkeiten und Eigenschaften für den *Discovery*-Prozess bereitstellt. Hierfür können verschiedene *Uniform Resource Identifiers (URIs)* im `dpws:Scopes`-Element der *MDPWS Discovery*-Nachrichten eingefügt werden:

Medizingerätetypen-basiertes Suchen Der *Service Provider* stellt die *Coded Value*-basierte Beschreibung seiner *MDSs* und *VMDs* bereit.

Rollen-basiertes Suchen Der *Service Provider* stellt seine sogenannten *Participant Key Purposes* (*PKPs*, Schlüsselrollen)[4.17] bereit.

Kontext-basiertes Suchen Der *Service Provider* stellt die *BICEPS*-Informationen zu aktuell assoziierten Kontexten, z. B. Ortskontext, Patientenkontext etc. (siehe Kapitel 4.4.1.1), bereit.

Benötigt ein *Service Provider* beispielsweise Informationen eines Pulsoximeters, das an einer bestimmten Patientin in einem bestimmten OP-Saal angeschlossen ist, so werden diese Informationen im `Scopes`-Element der `Probe`-Nachricht spezifiziert. Nur diejenigen *Service Provider*, die über die passenden Eigenschaften verfügen, werden mit einer entsprechenden `ProbeMatch`-Nachricht antworten. Somit wird der Aufwand für *Service Consumer* und *Provider* gering gehalten.

4.5.1.2 Filtermechanismus für Event-Abonnements Im *BICEPS*-Standard werden eine Reihe von Operationen definiert, die nach dem *Publish-Subscribe-Pattern* arbeiten. Ein Beispiel ist der *Episodic Metric Report* des *State Event Services*. Nach dem Abonnieren sendet der *Service Provider*

[4.17] Derzeit finden die Entwicklungs- und Standardisierungsarbeiten zur Definition von *PKPs* statt. Diese werden in die Standards *IEEE 11073-10700* bis *-10703* münden (siehe auch Kapitel 4.1). *IEEE 11073-20702* definiert bisher nur die grundlegendsten Rollen „*SDC Service Provider*“ und „*SDC Service Consumer*“. Spezifischere Rollen sind etwa *Metric Provider*, *Alert Provider*, *External Control Provider*.

dem *Service Consumer* bei jeder Änderung jeder Metrik eine Benachrichtigung (*Notification*). Der *Service Consumer* ist für die Erfüllung seiner Aufgaben ggf. nur an bestimmten Metrikzustandsänderungen interessiert. Ein komplexer *Service Provider* hat potentiell mehrere hundert Metriken. Sendet dieser jedes Update, hat das potentiell einen hohen, unnötigen Overhead zur Folge. Daher wird im *IEEE 11073-20701* Standard ein Filtermechanismus definiert, der auf die bestehenden Mechanismen von *WS-Eventing* [A 114] aufsetzt. So kann im `wse:Filter`-Element der *Subscribe*-Nachricht eine Liste von vollqualifizierten (*fullqualified*) *Handle*-Referenzen[4.18] angegeben werden. Ist eine solche Liste vorhanden, so sendet der *Service Provider* nur Benachrichtigungen, die sich auf das referenzierte Element im Beschreibungsbaum und seine Kindelemente beziehen.

4.5.1.3 Konkretisierungen innerhalb der *IEEE 11073 SDC*-Normenfamilie Wie bereits erwähnt, definiert der *IEEE 11073-20701* Standard eine Vielzahl von Verschärfungen bzw. Konkretisierungen des *MDPWS*- und des *BICEPS*-Standards. Exemplarisch seien einige Beispiele angeführt.

Im *BICEPS Participant Model* sind etwa das *Type*-Element, zur semantischen Beschreibung, und die *Safety Classification*, zur Spezifikation der klinischen Kritikalität und des sogenannten *Intended Use*, optional. Zur Sicherung der semantischen Interoperabilität wird die Angabe des *Type*-Elements verpflichtend für alle Metriken, Alarmbedingungen, Fernsteuerungsoperationen sowie *MDSs*, *VMDs* und *Channels* gefordert. Zusätzlich wird die Bereitstellung der *Safety Classification* für alle Fernsteuerungsoperationen verlangt. Somit werden die Anforderungen des *BICEPS*-Standards verschärft.

Weitere Konkretisierungen betreffen etwa die Generierung der `SequenceId` der *MDIB Version Group* mithilfe des *UUIDv5*-Algorithmus, die Nutzung des *TCP-Ports* `6464`, der bei der *Internet Assigned Numbers Authority (IANA)* für Medizingerätekommunikation reserviert ist, oder die Nutzung des *HTTP-Statuscodes* `413` („*payload too large*“), um anzuzeigen, dass eine Anfrage erfüllt werden könnte, aber die Antwortnachricht die in *MDPWS* spezifizierte Nachrichtenlänge von 4.096 kilobyte überschreitet.

4.5.2 Bindings für weitere, nicht-funktionale Qualitätsmerkmale

Um eine sichere Vernetzung von Medizingeräten zu ermöglichen, werden auch Standards außerhalb der medizinspezifischen *IEEE 11073*-Normenfamilie herangezogen, um nicht-funktionale Eigenschaften sicherzustellen.

4.5.2.1 Cyber-Security Wie bereits in Kapitel 4.3.4 beschrieben, wird für einen sicheren (*secure*) Informationsaustauch *TLS 1.2* oder neuer in Verbindung mit einer Authentifizierung mittels *X.509.v3*-Zertifikaten genutzt.

Abgesehen von Kontextinformationen (hier sind oft patientenspezifische Informationen enthalten) verlangen die *IEEE 11073 SDC*-Kernstandards keine verpflichtende Nutzung einer sicheren (*secure*) Verbindung. Aufgrund des Risikomanagements von Geräten in dynamischen Vernetzungssystemen ist aber davon auszugehen, dass die Kommunikation in der Regel abgesichert erfolgen wird.

Das sogenannte *Trust Establishment*, also der Aufbau einer Vertrauensbeziehung zwischen Vernetzungspartnern, ist insbesondere in herstellerübergreifenden Systemen eine Herausforderung. Hierfür werden Fähigkeiten der *X.509.v3*-Zertifikate genutzt. Es wird definiert, dass das *Common Name (CN)*-Element des *Distinguished Names* des Inhabers (*Subject*) des *X.509.v3*-Zertifikats die primäre *Unique Device Identification (UDI)* des Vernetzungsteilnehmers (in Form einer *UUIDv5*) sein sollte[4.19]. Dies vereinfacht die Identifizierung von Vernetzungspartnern, etwa für die Zugriffskontrolle, basierend auf

[4.18] Unter einer *fullqualified Handle*-Referenz versteht man die Kombination aus `SequenceId`, `InstanceId` (beide Teil der *MDIB Version Group*) und der eigentlichen *Handle*-Referenz.

[4.19] Der Standard definiert diese Anforderung als `SHOULD`, was mit einer Verpflichtung gleichzusetzen ist, wenn nicht relevante und dokumentierte Gründe dagegensprechen.

Black- oder *White-Lists*. Somit wird auch hier eine Verbindung zwischen der Gerätebeschreibung (*primäre UDI*) und den Mechanismen der Datenübertragung hergestellt.

Die *Extended Key Usage (EKU)*-Erweiterung der *X.509.v3*-Zertifikate[4.20] wird dafür genutzt, um die *PKPs* (Schlüsselrollen) für die Identifizierung und Autorisierung zu hinterlegen. Der *Service Provider* soll diese Information nutzen, um Zugriffsmöglichkeiten von *Service Consumern* einzuschränken, falls dies aus Sicht des Risikomanagements notwendig ist. So können beispielsweise alle oder bestimmte, hoch kritische, Fernsteuerungskommandos abgelehnt werden, wenn bestimmte *PKP*-Rollen nicht erfüllt werden. Darüber hinaus ist es ebenfalls gestattet, solche Einschränkungen auf der Basis weiterer Informationen des *X.509.v3*-Zertifikats oder der Zertifikatskette vorzunehmen.

4.5.2.2 Zeitsynchronisation Die zeitliche Synchronisation zwischen den Vernetzungspartnern ist eine entscheidende Voraussetzung für viele klinische Funktionalitäten. Die gemeinsame Zeitbasis bietet einen Bezugsrahmen, der es ermöglicht, Informationen geräteübergreifend zu korrelieren. Die Synchronisationsgenauigkeit ist dabei entsprechend zu beachten. Des Weiteren sind hinreichend genaue Uhren wichtig, um die Validität von Zertifikaten für *TLS*-Verbindungen zu überprüfen.

Zur Erfüllung dieser Anforderungen schreibt der *IEEE 11073-20701* Standard vor, dass Vernetzungsteilnehmer das *Network Time Protocol (NTP)* in der Version 3 [A 139] oder kompatible Versionen, wie etwa *NTP*-Version 4 [A 140], unterstützen müssen. Zur Bereitstellung der Zeitinformationen im *IEEE 11073 SDC*-basierten Netzwerk dient das sogenannten *Clock*-Objekt (`ClockDescriptor` und `ClockState`), das vom *Service Provider* in der Gerätebeschreibung auf MDS-Ebene angeboten werden muss. Die *Clock* stellt Informationen über die Auflösung, die Genauigkeit, das Synchronisierungsprotokoll, die letzte Kalibrierung (Zeitpunkt, Art, Zustand) etc. bereit. Auf der Basis dieser Informationen können andere Vernetzungsteilnehmer die zeitlichen Informationen, die etwa in den Zuständen von Metriken, Alarmen, Kontexten etc. enthalten sind, bewerten und entsprechend nutzen.

4.5.2.3 Quality of Service Wird eine gewisse Dienstgüte (*Quality of Service (QoS)*) für das Erfüllen einer klinischen Aufgabe benötigt, so sieht die *IEEE 11073 SDC*-Familie die Nutzung der *Internet Engineering Task Force (IETF)*-Spezifikation *Differentiated Services (DiffServ)* [A 141–144] vor.

DiffServ ist ein Mechanismus zur Klassifikation von Paketen in einem IP-Netzwerk. Mittels dieser Klassifikation kann eine Priorisierung bei der Paketverarbeitung realisiert werden. Die priorisierte Verarbeitung ist dann wichtig, wenn Pakete aufgrund einer hohen Netzwerkauslastung gepuffert werden müssen und nicht unverzüglich verarbeitet werden können. Die *DiffServ*-Klassen werden im *IP-Header* spezifiziert. *DiffServ* löste die Spezifikation *Type of Service (TOS)* [A 145] ab, nutzt aber das dafür vorgesehene Headerfeld mit einer Breite von acht Bits (im *IPv6*-Header als *Traffic Class* bezeichnet) und ist auch zu einem gewissen Maße abwärtskompatibel. Vom acht Bit breiten Feld, das auch als *Differentiated Services Field (DS Field)* bekannt ist, nutzt *DiffServ* nur die ersten sechs Bits, den sogenannten *Differentiated Services Code Point (DSCP)*. Die letzten zwei Bits werden in dieser Spezifikation nicht genutzt (*„currently unused (CU)“*). Im *DSCP* wird die Paketklasse definiert.

Die Behandlung von Paketen der Prioritätsklassen wird in den sogenannten *Per-Hop Behaviors (PHBs)* beschrieben. Das Verhalten wird also nicht von Endpunkt zu Endpunkt definiert, sondern für jeden einzelnen Netzwerk-Hop. Es seien an dieser Stelle drei *PHBs* erwähnt:

Default PHB Paketweiterleitung erfolgt nach dem *Best Effort*-Prinzip. Entsprechend markierte Pakete haben somit die geringste Priorität. Der *DSCP* ist $000000_{(2)}$.

[4.20] Siehe auch Glossareintrag zum Thema X.509.v3-Zertifikate.

Expedited Forwarding (EF) PHB Die *EF PHB* wird im *RFC 3246* [A 146][4.21] spezifiziert. Im *RFC 3260* [A 147] finden sich einige weitere Erläuterungen. Die *EF PHB* ermöglicht durch ihre strikte Priorisierung eine Optimierung von Latenz und Jitter, da Pakete dieser Klasse stets Vorrang vor Paketen aller anderen Klassen haben. Der *DSCP* ist $101110_{(2)}$. Der *IEEE 11073-20701* Standard schließt die Nutzung dieser *PHB* weitestgehend aus. Dies wird damit begründet, dass der *EF*-Datenverkehr aufgrund seiner harten Priorisierung meist starken Beschränkungen unterliegt[4.22].

Assured Forwarding (AF) PHB Der *RFC 2597* [A 151] spezifiziert eine Gruppe von *AF PHBs*. Diese Gruppe umfasst vier Klassen (1 bis 4) und innerhalb jeder Klasse drei Stufen für die *Drop-Wahrscheinlichkeit*: gering (1), mittel (2) und hoch (3). Pakete einer Klasse müssen unabhängig von den Paketen der anderen Klassen weitergeleitet werden, d. h., dass vier unabhängige *Queues* (Warteschlangen) für die vier Klassen implementiert werden. Für jede Klasse ist ein Minimum an Ressourcen (Puffergröße und Netzwerkbandbreite) zu garantieren, sodass die Unabhängigkeit der Klassen gewährleistet ist. Die Spezifikation weist den Klassen keine Priorität zu. Aufgrund der Ressourcenzuteilung auf die Klassen wird aber in der praktischen Anwendung eine implizite Priorisierung erfolgen. Innerhalb der Klasse erfolgt die Priorisierung anhand der *Drop-Wahrscheinlichkeit*, d. h., je höher die Wahrscheinlichkeit, dass ein Paket verworfen wird, umso geringer seine Priorität.

Der *DSCP* wird wie folgt gebildet: $\texttt{aaabb0}_{(2)}$, wobei `aaa` der Binärdarstellung der Klassennummer entspricht und `bb` der Binärdarstellung der *Drop-Wahrscheinlichkeit*.

In Bezug auf die Nutzung der *DiffServ*-Spezifikation macht der *IEEE 11073-20701* Standard eine Reihe von Vorgaben: Etwa sollen Pakete, die medizinische Alarmzustände transportieren, mit einer geringen *Drop-Wahrscheinlichkeit* markiert werden, medizinische Metriken mit einer mittleren oder geringen *Drop-Wahrscheinlichkeit*, während Pakete ohne eine medizinische *Safety Classification* mit der *Default PHB* (*Best Effort*) markiert werden sollen. Die weitere Nutzung und Klassifizierung unterliegt dem Risikomanagement des jeweiligen Medizingerätes.

In der praktischen Nutzung der *QoS*-Mechanismen ist sicherzustellen, dass die gesamte genutzte Netzwerkinfrastruktur die entsprechenden Spezifikationen umsetzt. Darüber hinaus können durch die Nutzung der *DiffServ*-Spezifikation keine Garantien für spezifische maximale Paketverlustraten oder maximale Latenzen gegeben werden. Es erhöht sich lediglich die Wahrscheinlichkeit für die Einhaltung solcher Vorgaben. Damit können mit der Nutzung der *IEEE 11073 SDC*-Kernstandards allein keine harten Echtzeiteigenschaften garantiert werden. Weiterführende *QoS*-Mechanismen, die für hoch sicherheits- und zeitkritische Anwendungen benötigt werden, gehen über die aktuelle Zielsetzung von *IEEE 11073 SDC* hinaus.

4.6 Erreichung höherer Interoperabilitätslevel

Die *IEEE 11073 SDC*-Familie allein ist nicht ausreichend, um die semantische Interoperabilität herzustellen. Eine noch so wohldefinierte Struktur von ausgetauschten Daten erreicht nicht das Ziel, wenn sie nicht richtig eingesetzt und mit dem falschen Inhalt gefüllt wird. Dies kann aber nicht vom Standard selbst gelöst werden. In Abbildung 4.2 wurde die semantische Interoperabilität daher auf der Ebene der Anwendungssoftware und Gerätelogik abgebildet.

[4.21] *Requests for Comments (RFCs)* stellen de-facto Standards der *IETF* dar, auch wenn die wörtlichen Übersetzung „Bitte um Kommentare“ etwas anderes suggerieren könnte.

[4.22] Aus Gründen der guten wissenschaftlichen Praxis sei darauf hingewiesen, dass dem Autor dieser Arbeit für diese Aussage der Norm *IEEE 11073-20701* keine unabhängigen Quellen bekannt sind. Ein Beleg ist in der englischsprachigen Wikipedia [A 148] zu finden. Entsprechend der Historie des Artikels geht der Abschnitt zur *EF PHB* auf September 2006 zurück und seit Juni 2010 wird das Fehlen einer Referenz kritisiert. In anderen Quellen, wie etwa [A 149] von 2013 findet sich eine nahezu 1:1 Abschrift des entsprechenden Abschnitts des Wikipedia-Artikels. Inwieweit [A 150] von 2016 eine unabhängige Bestätigung der Aussage liefert, vermag der Autor nicht einzuschätzen.

Je mehr Informationen von den Vernetzungspartnern bereitgestellt werden, desto größer ist das Potential zukünftiger, innovativer Produkte für die Lösung der heutigen Probleme. Hersteller sind daher angehalten, nicht nur die als mandatorisch definierten Informationen bereitzustellen, sondern die Netzwerkrepräsentation ihrer Geräte umfassend zu modellieren.

4.6.1 Nomenklaturen

Ein entscheidender Baustein für semantische Interoperabilität sind umfassende, anerkannte, kontrollierte und standardisierte Nomenklaturen. In *IEEE 11073 SDC* wird normativ die Nomenklaturserie *IEEE 11073-1010X* genutzt. Die Mechanismen hierfür wurden bereits in Kapitel 4.4.1.1 beschrieben. Darüber hinaus können ergänzend sogenannte *Meaningful Use*-Kodierungssysteme, wie etwa *Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT)* oder *Logical Observation Identifiers Names and Codes (LOINC)* genutzt werden. Diese bieten etwa durch Verknüpfungsmechanismen, z. B. von Messgrößen, Zeitangaben, Methoden/Vorgängen etc., und Ontologien eine umfassende Aussagekraft. Sie gehen damit von ihrer Mächtigkeit über die Möglichkeiten der *IEEE 11073-1010X* Nomenklatur hinaus.

IEEE 11073 SDC ermöglicht die Vernetzung von *Point-of-Care (PoC)*-Medizingeräten, die bisher nicht im Fokus standen. Daher ist der aktuelle Umfang der *IEEE 11073-1010X* Nomenklatur nicht ausreichend [B 25]. Die Erweiterung um semantische Beschreibungen für diese Geräte ist folglich notwendig. Diese Arbeit erfolgt etwa im Projekt *PoCSpec*, unter Beteiligung des Autors, für einige chirurgische Medizingeräte.

4.6.2 Device Specializations

Aufgrund der hohen Komplexität von Medizingeräten modellieren verschiedene Hersteller ihre funktionell ähnlichen Geräte sehr unterschiedlich. Das hat sich bereits in Forschungsprojekten wie *OR.NET* oder *MoVE* gezeigt. Daraus ergibt sich jedoch das Problem der uneinheitlichen Repräsentation in einem System vernetzter Medizingeräte. Dies erschwert sowohl die herstellerübergreifende Interoperabilität und die Austauschbarkeit von Geräten im Betrieb, als auch die Tests und Zulassung der Geräte erheblich. Die semantische Beschreibung von einzelnen Elementen mittels Nomenklaturen ist nicht ausreichend.

Daher werden im Zuge des *BMWi*-geförderten Projekts *PoCSpec* sogenannte Gerätespezialisierungen (*Device Specializations*, *DevSpecs*) für komplexe Medizingeräte entwickelt. Diese Standards definieren für eine bestimmte Klasse von Geräten Umfang, Struktur und Semantik der im Netzwerk angebotenen Informationen und Funktionalitäten. Dabei sind sowohl Aspekte wie Abhängigkeiten zwischen Parametern, Anforderungen an Fernsteuerungskommandos, Verhalten in verschiedenen regulären Zuständen und Fehlersituationen, als auch Anforderungen an die Vernetzungsinfrastruktur mit eingeschlossen. Unter einer Klasse von Geräten versteht man hierbei nicht die konkreten Produkte einzelner Hersteller, sondern deren Abstraktion, z. B. die Klasse der Beatmungsgeräte, in Abgrenzung zu den verschiedenen Modellen eines oder mehrerer Hersteller.

Die *Device Specializations* geben also einen Rahmen vor, an den sich die Hersteller eines Gerätes zu halten haben, um beispielsweise die Rolle eines Beatmungsgerätes in einem vernetzten System einnehmen zu können. Über diesen Rahmen hinaus sollen aber Hersteller-spezifische Innovationen ermöglicht werden.

Derzeit existieren *Device Specializations* für einfache *Personal Health*-Geräte. Sie sind in der *IEEE 11073-104XX* Serie gruppiert, z. B. Pulsoximeter (-10404), Blutdruckmonitor (-10407), Thermometer (-10408) oder Glukosemeter (-10417). Komplexe Geräte sind bisher nicht betrachtet worden, da die Arbeiten an der *IEEE 11073-103XX* Serie nicht abgeschlossen wurden und derzeit nicht weiterverfolgt werden.

Derzeit werden *Device Specializations* für einige chirurgische Geräte entwickelt (siehe auch Abschnitt 4.1 und Abbildung 4.1). Das Projekt *PoCSpec* beschäftigt sich mit Standards für HF-Chirurgiegeräte, endoskopische Kameras und Lichtquellen, Insufflatoren und Saug-/Spül-Pumpen (*IEEE P11073-10721* bis *-10725*). Aus Gründen des Umfangs dieses Dokuments kann auf diese umfassenden Arbeiten des Autors in diesem Bereich nicht weiter eingegangen werden.

4.7 Diskussion

Mit der vollständigen Veröffentlichung der *IEEE 11073 SDC*-Kernstandards Anfang 2019 ist der Grundstein für die herstellerübergreifende Interoperabilität von Medizingeräten für die Versorgung von Patientinnen im Krankenhaus gelegt.

Erste Firmen haben 2018 und 2019 Medizingeräte mit einer *IEEE 11073 SDC*-Schnittstelle angekündigt und Prototypen vorgestellt [A 152, 153]. Seit 2019 sind erste Produkte im Einsatz am Patienten. Die Firma *Dräger* beschreibt hierzu in ihrer Pressemitteilung [A 154], dass diese ersten Geräte „eine Vorstufe“ der *IEEE 11073 SDC*-Kernstandards implementieren. Es kommt eine *Draft*-Version zum Einsatz, die während des internationalen Abstimmungsverfahrens erstellt wurde. In zukünftigen Produktgenerationen werden die final verabschiedeten Normen genutzt werden, wie etwa aus dem Produktmarketing abgeleitet werden kann [A 155]. *IEEE 11073 SDC* wird im beschriebenen System [A 154] zunächst genutzt, um den Austausch von Daten und Steuerbefehlen (z. B. Alarmmanagement) zwischen dem Patientenmonitoring, dem Anästhesiegerät und dem Gateway zu den Krankenhausinformationssystemen zu realisieren. Das Einbinden von Geräten anderer Hersteller ist zunächst nicht möglich. Dies liegt sowohl darin begründet, dass eine *Draft*-Version der Standards implementiert wurde, als auch darin, dass bisher keine weiteren Geräte anderer Hersteller verfügbar sind. In Zukunft ist von einer entsprechenden Weiterentwicklung auszugehen. Dieses zweistufige Vorgehen zeigt, dass *IEEE 11073 SDC* sowohl für eine interne Gerätevernetzung, als auch für ein herstellerübergreifendes System genutzt werden kann.

Die in diesem Kapitel beschriebenen Mechanismen der *IEEE 11073 SDC*-Kernstandards ermöglichen die Selbstbeschreibung von Fähigkeiten und Zuständen aller derzeitigen und zukünftigen Medizingeräte. Auf technischer Ebene werden damit die Voraussetzungen für eine herstellerübergreifende Interoperabilität geschaffen. Für zulassungsfähige, vernetzte Medizinprodukte und sicheres, herstellerübergreifendes *Plug-and-Play*-Verhalten werden über die *IEEE 11073 SDC*-Kernstandards hinaus aber weitere Prozess- und geräteklassenspezifische Standards benötigt. Mit den *Participant Key Purpose*-Standards (*IEEE 11073-1070X* Serie) und den *Device Specializations* (*IEEE 11073-1072X* Serie) befinden sich diese derzeit in der Erarbeitung und Standardisierung. Der Autor der vorliegenden Arbeit ist an diesem Prozess, u. a. durch das *PoCSpec*-Projekt, beteiligt. Vor dem Abschluss dieser Arbeiten ist nicht mit einer dynamischen, herstellerübergreifenden Interoperabilität im klinischen Umfeld zu rechnen.

Ebenso ist eine Weiterentwicklung und Anpassung der derzeitigen Risikomanagement- und Zulassungsprozesse für offen vernetzte Medizingerätesysteme notwendig. Als Voraussetzung für eine breite Markteinführung kann etwa angesehen werden, dass sich die Prozesse weg von einer Zulassung bzw. von einem Inverkehrbringen konkreter Gerätepaarungen hin zur Zulassung des einzelnen Gerätes als Teil eines Geräteensembles entwickeln. Folglich soll die Zulassung dann gegen Schnittstellen-, Prozess- und Geräteklassenstandards erfolgen [A 51].

Ein weiterer, bisher nicht hinreichend betrachteter Aspekt ist das Zusammenspiel zwischen *Safety*, *Security* und Interoperabilität. In Forschung und Industrie herrscht überwiegend Einigkeit darüber, dass *Safety* nur mit *Security* erreicht werden kann [A 53]. Gerade in einem vernetzten System, das auf Standardtechnologien basiert, muss von einem hohen Bedrohungspotential ausgegangen werden. Einerseits liegt die Herausforderung nun darin, dass *Security*-Mechanismen so entwickelt und umgesetzt werden, dass sie die Schutzanforderungen erfüllen. Andererseits darf der systemübergreifende

Daten- und Steuerbefehlsaustausch nicht so weit eingeschränkt werden, dass keine Interaktion mehr stattfinden kann und damit auch keine Mehrwerte generiert werden können.

Die derzeitigen *IEEE 11073 SDC*-Kernstandards nutzen Standardnetzwerktechnologien. Die Kommunikation erfolgt somit im Wesentlichen nach dem „*Best Effort*"-Prinzip. Zwar ist die Nutzung des *DiffServ*-Standards zur Sicherung der Dienstgüte (*QoS*) beschrieben, gleichzeitig wird aber darauf verwiesen, dass dies nicht ausreicht, um harten Echtzeitanforderungen zu genügen. Insbesondere im Bereich der Fernsteuerung und Fernauslösung von Medizingeräten über ein SDC-Netzwerk werden Produkte (harte) Echtzeitanforderungen an die Vernetzung stellen. Daher sind in Zukunft Mechanismen zu entwickeln, die sowohl die Interoperabilität und Flexibilität der aktuellen *IEEE 11073 SDC*-Standards erhalten, als auch (harte) Echtzeitanforderungen erfüllen.

Ein großes Potential hat die Nutzung der neuen *Time-Sensitive Networking (TSN)*-Standards der *IEEE 802.1*-Normenfamilie. Die Grundidee besteht darin, dass die selbstbeschreibenden Informationen der Medizingeräte, die mittels *IEEE 11073 SDC* zur Verfügung gestellt werden, dafür genutzt werden, um das *TSN*-Netzwerk entsprechend den Echtzeitanforderungen automatisch zu konfigurieren. Erste Umsetzungen dieses Konzepts wurden bereits erfolgreich beschrieben [B 2], [E 2].

Zusammenfassend kann festgehalten werden, dass die *IEEE 11073 SDC*-Kernstandards eine bedeutende Grundlage für eine herstellerübergreifende Medizingeräteinteroperabilität legen. Die ersten Produkte zeigen, dass Hersteller bereit sind, die Standards zu implementieren. Dennoch sind weitere Anstrengungen im Bereich der Standardisierung und der Prozesse für das Inverkehrbringen bzw. Zulassen notwendig. Die Bedarfe für interoperable Systeme werden klar von den Anwenderinnen geäußert. Nichtsdestotrotz bleibt abzuwarten, ob die *IEEE 11073 SDC*-Standards von einer breiten Masse der Marktteilnehmer implementiert werden.

4.8 Gemeinsamkeiten und Abgrenzungen zu MD PnP und SCOT

In Kapitel 2.4 wurden die Projekte *Medical Device „Plug-and-Play" (MD PnP)* und *Smart Cyber Operating Theater (SCOT)* und deren Ergebnisse vorgestellt. In Kapitel 3 und in diesem Kapitel wurden intensiv die architektonischen Grundlagen und die *IEEE 11073 SDC*-Normenfamilie, mit dem Fokus auf den *IEEE 11073 SDC*-Kernstandards, vorgestellt. Zusammenführend sollen nun Gemeinsamkeiten, Unterschiede und Abgrenzungen zwischen den Ansätzen diskutiert werden. Dabei wird über die *IEEE 11073 SDC*-Standards hinaus auch die Arbeit des *OR.NET*-Konsortiums im Allgemeinen betrachtet.

4.8.1 MD PnP und OR.NET

Die Konsortien von *MD PnP* und *OR.NET* verfolgten und verfolgen eine sehr ähnliche Strategie: Ausgehend von den Bedürfnissen der Patientinnen und der Anwenderinnen, wie Ärztinnen und Pflegepersonal, und konkreten Anwendungsfällen werden Lösungen auf der Basis einer herstellerübergreifenden Medizingeräteinteroperabilität entwickelt. Die entwicklungsbegleitende Evaluation der Fortschritte und Ergebnisse durch umfassende Demonstratoren und die Zurverfügungstellung von Open-Source-Software ist eine weitere Gemeinsamkeit.

Während beide Konsortien technische Lösungen präsentieren, konzentrieren sich die Standardisierungsaktivitäten aus *MD PnP* heraus auf die generellen Konzepte und Anforderungen an eine herstellerübergreifende Medizingeräteintegration. So kann etwa der *Integrated Clinical Environment (ICE)-Standard* als Wegbereiter für die *IEEE 11073 SDC*-Normenfamilie gesehen werden. Die technische Umsetzung des *MD PnP*-Konsortiums erfolgt mittels der Referenzimplementierung *OpenICE*. Eine technische Spezifikation im Sinne von Standards, die von Herstellern implementiert werden können, existiert nicht.

Es sei kritisch angemerkt, dass im *ICE-Standard* in Teilen die Grenzen zwischen einer grundlegenden Architektur und allgemeinen Anforderungen einerseits und Ansätzen einer technischen Spezifikation andererseits verschwimmen. So legen beispielsweise Teile der *ICE-Architektur* zentrale Komponenten nahe, die aus technischer Sicht auch verteilt und komplett dezentral realisiert werden können und dennoch alle gestellten Anforderungen einhalten.

Im Gegensatz zum *ICE-Standard* stellt die *IEEE 11073 SDC*-Normenfamilie eine konkrete technische Spezifikation dar. *IEEE 11073 SDC* erfüllt zu großen Teilen den *ICE-Standard*. Im Folgenden wird auf einzelne zentrale Aspekte des *ICE-Standards* und auf Gemeinsamkeiten, Unterschiede und mögliche Umsetzungen mittels *IEEE 11073 SDC* eingegangen. Eine umfassende Analyse der Konformität geht über die Zielsetzung der vorliegenden Arbeit hinaus.

Geräteintegration Die *ICE-Interfaces* zur Integration von Geräten und externen Ressourcen entsprechen den *IEEE 11073 SDC*-Netzwerkschnittstellen. Für Teilnehmer, die Informationen und Funktionalitäten im Netzwerk bereitstellen (*Service Provider* im Sinne einer *SOA*) kann davon ausgegangen werden, dass die *SDC-Gerätebeschreibungen* die Anforderungen der *ICE Device Models* erfüllen. Inwiefern dies ebenfalls für Geräte gilt, die Informationen aus der Vernetzung beziehen (*Service Consumer*) kann aktuell nicht zufriedenstellend beantwortet werden. Der dritte Teil des *ICE-Standards*, der sich speziell mit dem *Device Model* beschäftigen soll, ist bisher nicht verfügbar. Es ist allerdings davon auszugehen, dass auch *Service Consumer* ihre Bedürfnisse im Netzwerk kommunizieren sollen. Eine explizite Bereitstellung von Eigenschaften der *Service Consumer*, wie es für die *Service Provider* erfolgt, ist in einer *IEEE 11073 SDC*-basierten Vernetzung nicht zwingend vorgesehen. Die Bedürfnisbeschreibung der *Service Consumer* erfolgt implizit innerhalb der entsprechenden *Queries*. Die Bereitstellung einer Selbstbeschreibung von *Service Consumern* über die Mittel eines *Service Providers* oder über eine Bereitstellung im Vorfeld ist konzeptuell aber nicht ausgeschlossen. Die konkrete Umsetzung würde der Gerätelogik obliegen.

ICE Netzwerk Controller* und *Supervisor Wie bereits beschrieben werden diese logischen Komponenten in großen Teilen als (verteilte) *Middleware* realisiert [A 85, 87]. *IEEE 11073 SDC* setzt diese Aufgaben und Anforderungen um. Eine abschließende Beurteilung ist ohne die Verfügbarkeit von Teil 2 und Teil 4 des *ICE-Standards*, die sich speziell mit dem *Netzwerk Controller* und dem *Supervisor* beschäftigen, nicht möglich. In *IEEE 11073 SDC* ist keine *Supervisor*-Komponente vorgesehen, da es sich um eine komplett verteilte und dezentrale Middleware-Architektur handelt.

Der *ICE-Supervisor* soll auch eine echtzeitfähige Kommunikation ermöglichen [A 87]. Ansätze für Umsetzungen werden im Konzept *ICE++* [A 156] beschrieben. Um eine echtzeitfähige Kommunikation zu ermöglichen, soll die *Software-Defined Networking (SDN)*-Technologie eingesetzt werden. Bestrebungen hin zu standardisierten Lösungen sind derzeit nicht zu erkennen. In den *IEEE 11073 SDC*-Kernstandards ist derzeit keine echtzeitfähige Kommunikation vorgesehen. Würde diese für die Anwendungsfälle benötigt, so wäre eine entsprechende *Supervisor*-Funktionalität zu ergänzen. Es könnten Konzepte wie *SDN* und *TSN* zum Einsatz kommen. Die entsprechenden Netzwerk-Controller würden die zugehörigen (Teil-)Aufgaben des *ICE-Supervisors* realisieren (siehe auch Abschnitt 10.4 und [B 2]).

***ICE*-Anwendungen und *Daten-Logger* (als Teil des *ICE-Managers*)** Die für die *ICE*-Anwendungen beschriebenen zentralen Aufgaben während der Behandlung können auf der Basis von *IEEE 11073 SDC* realisiert werden. Eine Zuordnung zu einem zentralen *Manager* ist entgegen der *ICE*-Architektur nicht vorgeschrieben. Solche Applikationen werden als *IEEE 11073 SDC*-Teilnehmer realisiert. Der *Session Manager* (siehe Kapitel 6) kann beispielsweise als eine solche Anwendung betrachtet werden. Er stellt eine *ICE-Applikation* dar und wird wie beschrieben als gleichberechtigter Vernetzungspartner mittels *IEEE 11073 SDC*

realisiert. Dasselbe gilt für den *ICE-Daten-Logger*. Für eine technische Realisierung ist dieser nicht notwendig und ist daher auch nicht Teil von *IEEE 11073 SDC*. Zur Erfüllung gestellter Anforderungen von Nachverfolgbarkeit und forensischer Analyse im klinischen Einsatz wäre eine solche Komponente als *IEEE 11073 SDC*-Teilnehmer zu realisieren.

Auf der technischen Ebene unterscheiden sich die Referenzimplementierung *OpenICE* und die Normenfamilie *IEEE 11073 SDC* grundlegend in der Wahl der Übertragungstechnologie: *OpenICE* nutzt *Data Distribution Service (DDS)*, während *IEEE 11073 SDC* auf *DPWS* aufsetzt. Die verschiedenen Demonstratoren haben gezeigt, dass beide Technologien nutzbar sind.

Es kann zusammengefasst werden, dass die Aktivitäten des *MD PnP*-Konsortiums international wegbereitend für herstellerübergreifend, interoperabel vernetzte Medizingeräteensembles sind. Mit dem *ICE-Standard* ASTM F2761 [A 9] sind die generelle Architektur und grundlegende Anforderungen definiert. Die US-amerikanische Food and Drug Administration (FDA) hat den *ICE-Standard* in die Liste der *Recognized Consensus Standards* [A 157] aufgenommen und ihn damit als Stand der Technik hervorgehoben. Eine technische Spezifikation zur Umsetzung existiert hingegen nicht. Diese Lücke schließt die *IEEE 11073 SDC*-Normenfamilie. Somit ergänzen sich die in einem regen Austausch stehenden Konsortien inzwischen mehr, als dass sie in einer direkten Konkurrenz zueinander stehen. Ebenso wurde bereits gezeigt, dass das *IEEE 11073 SDC*-Datenmodell zusammen mit der *DDS*-basierten Übertragungstechnologie des *MD PnP*-Konsortiums arbeiten kann [A 158].

4.8.2 SCOT und OR.NET

Grundlegend verfolgen die Konsortien um *SCOT* und *OR.NET* dasselbe Ziel: herstellerübergreifende Medizingeräteinteroperabilität. *SCOT* fokussiert dabei sehr stark auf die Anwendungen in einem Hybrid-OP, während sich der Anwendungsbereich von *IEEE 11073 SDC* auf den gesamten Bereich der (professionellen) Medizingeräte im Krankenhaus erstreckt.

Auf technischer Ebene unterscheiden sich die Konzepte stark. Zwar kann *Open Robot/Resource interface for the Network (ORiN)* auch *SOAP* und *WSDL* nutzen, dennoch sind die Unterschiede zu *DPWS* bzw. *MDPWS* sehr groß.

Eine Schlüsseleigenschaft von *IEEE 11073 SDC* ist das dynamische Finden (*Dynamic Discovery*) von Vernetzungspartnern. Das *SCOT*-Entwicklerteam stellt in einem umfassenden technischen Überblickspaper [A 92] zu *SCOT* heraus, dass *SCOT* über keine *Plug-and-Play*-Fähigkeiten verfügt. Ein *Dynamic Discovery* ist nicht vorgesehen [A 98]. Die *SCOT-Middleware* arbeitet auf der Basis von fest zugewiesenen und als bekannt vorausgesetzten *IP-Adressen* und *Ports* [A 92]. Derzeit geht man davon aus, dass in den vernetzten OP-Sälen ähnliche Operationen mit demselben Geräteensemble durchgeführt werden, sodass keine dynamischen Änderungen notwendig sind [A 92]. *IEEE 11073 SDC* setzt hingegen auf eine hohe Flexibilität mit wechselnden und selbst zur Laufzeit dynamischen Geräteverbünden.

Das *IEEE 11073 SDC*-Datenmodell (*IEEE 11073-10207*) ermöglicht eine umfassende Selbstbeschreibung der teilnehmenden Medizingeräte. Insbesondere die semantische Interpretierbarkeit und Bereitstellung von zusätzlichen Informationen ermöglichen eine Risikobewertung der Informationen durch die Vernetzungspartner. Dies geht deutlich über die Aussagekraft von *ORiN* bzw. *OPeLiNK* hinaus. Des Weiteren beschreiben Okamoto et al. [A 92] die Notwendigkeit, Anpassungen auf Quellcodeebene vorzunehmen, wenn neue Geräte integriert werden sollen. Die Mächtigkeit der *IEEE 11073 SDC*-Normenfamilie ermöglicht es, dass solche Anpassungen nicht vorgenommen werden müssen.

4.8.3 Verbindung von ORiN und IEEE 11073 SDC

Trotz der beschriebenen Unterschiede zwischen der *OR.NET*- und *SCOT*-Technologie ist es möglich, die beiden Systeme miteinander zu verbinden. So haben Berger et al. [A 98] ein Konzept für eine Schnittstelle zwischen *IEEE 11073 SDC* und *ORiN* entwickelt und prototypisch evaluiert.

Die Grundidee besteht darin, die Strukturen der beiden Standards aufeinander abzubilden. So entsprechen etwa die *Metriken* in *IEEE 11073 SDC* den *Variablen* in *ORiN*. Die hierarchische Struktur der *ORiN-Objekte* (siehe Abbildung 2.5) kann entsprechend auf die *IEEE 11073 SDC*-Baumstruktur abgebildet werden. Beispielsweise werden die Objekte *Robot*, *File* etc. als *Channel* modelliert. Ebenso können die Interaktionskonzepte bzw. Services aufeinander abgebildet werden, z. B. *Set*-Operation und *Put*-Operation oder *Activate*-Operation und *Execution*-Operation. Somit können Informationen zwischen den Systemen übersetzt und ausgetauscht werden.

Als prototypische Schnittstelle zwischen *ORiN* und *IEEE 11073 SDC* wurde die Applikation *GATOR*[4.23] entwickelt. Als Anwendungsbeispiel dient eine OP-Leuchte, die sowohl auf der Basis von *ORiN2* als auch mit *IEEE 11073 SDC* modelliert und implementiert wurde. Es konnte gezeigt werden, dass entsprechend komplementäre Clients mittels der *GATOR*-Schnittstelle Informationen und Steuerbefehle mit der OP-Leuchte austauschen können.

[4.23] Der Name *GATOR* setzt sich aus den englischen Begriffen *Gate* und *OR* (Operating Room) zusammen.

5 Performanzevaluation

Die *IEEE 11073 SDC*-Normenfamilie stellt ein grundlegend neues Kommunikationsparadigma auf der Basis der *SOMDA* dar. Die Fähigkeiten zur semantischen Interoperabilität beliebiger Medizingeräte wurden in Kapitel 4 beschrieben. Für die praktische Nutzbarkeit des Konzeptes muss aber ebenso eine ausreichende Performanz der Kommunikation für medizinische Anwendungsfälle nachgewiesen werden.

Dieses Kapitel basiert in großen Teilen auf der Veröffentlichung „Measuring Latencies of IEEE 11073 Compliant Service-Oriented Medical Device Stacks“ [B 17]. Als Erstautor der Publikation zeigt sich der Autor dieser Arbeit hauptverantwortlich für Konzept, Implementierung, Durchführung und Auswertung der Evaluation. Unterstützung erhielt er dabei von Institutsmitgliedern und einem Projektpartner in den Bereichen Implementierung und Auswertung.

5.1 Zielsetzung und Konzept der Performanzevaluation

Das Ziel der Performanzevaluation besteht darin, die Leistungsfähigkeit der *IEEE 11073 SDC*-Konzepte in Bezug auf Kommunikationslaufzeiten zu untersuchen. Darüber hinaus sollen Verbesserungspotentiale und Handlungsempfehlungen abgeleitet werden. Diese beziehen sich sowohl auf die Nutzung, als auch auf die Entwicklung von Kommunikationsbibliotheken. Die Untersuchungen erfolgen stellvertretend anhand von drei verfügbaren Open-Source-Implementierungen. Diese wurden unabhängig voneinander entwickelt, sodass die Ergebnisse zu einem gewissen Grad verallgemeinert werden können. Eine derartige, beispielhafte Evaluation in einem vereinfachten, realen Versuchsaufbau lässt direkte Aussagen aber nur für die konkreten Implementierungen und genutzten Hardwareplattformen zu.

Aufgrund der Komplexität wurde keine der Softwarebibliotheken vom Autor selbst entwickelt. Dieses Vorgehen ist untypisch für die Art der vorliegenden Arbeit. Es ist darin begründet, dass in den Forschungsprojekten eine Arbeitsteilung zwischen der Konzeptentwicklung und Standardisierung und der Entwicklung der Bibliotheken erfolgte.

Zum Zeitpunkt der Evaluation waren drei Open-Source-Implementierungen der *IEEE 11073 SDC*-Normenfamilie verfügbar[5.1]: *openSDC* (implementiert in *Java*) [A 161], *SoftICE* (implementiert in *Java*) [A 162, 163] und *OSCLib* (implementiert in *C++*) [A 164]. Jeweils zur Version 3.0.0 haben die Bibliotheken *SoftICE* und *OSCLib* ihren Namen in *SDCLib/J* bzw. *SDCLib/C* geändert. Da die Untersuchungen vor diesem Zeitpunkt stattfanden, werden in diesem Abschnitt weiterhin die ursprünglichen Bezeichnungen genutzt.

Die untersuchten Bibliotheken sind prototypische Implementierungen, die im Zuge von Forschungs- und Entwicklungsprojekten entstanden sind. Sie wurden nicht für den Einsatz in zulassungspflichtigen Medizinprodukten entwickelt. Produktiv im Einsatz befindliche bzw. produktreife Implementierungen stehen für die Evaluation bislang nicht zur Verfügung. Bei der Beurteilung der Performanz stehen nicht die einzelnen Implementierungen im Vordergrund, sondern das zugrunde liegende Konzept, das von den drei unabhängigen Bibliotheken umgesetzt wird. Bezogen auf die reine Kommunikationslatenz können die Ergebnisse der Messungen als Obergrenzen angesehen werden, da Optimierungen unter Performanzgesichtspunkten bisher nicht im Fokus standen. Für die Gesamtlatenz realer Systeme ist die Kommunikation hingegen nur ein Teilaspekt.

IEEE 11073 SDC-Kommunikationsbibliotheken sind komplexe Softwaresysteme. Die Architektur und Umsetzung dieser Softwarekomponenten wird im Rahmen der vorliegenden Arbeit nicht detailliert beschrieben, da sie nicht vom Autor selbst entwickelt werden. Es sei an dieser Stelle auf die

[5.1] Inzwischen sind mit der *SDCri*-Bibliothek (Java) [A 159] und der *sdc11073*-Bibliothek (Python) [A 160] zwei weitere freie Implementierung hinzugekommen, die im Jahr 2020 veröffentlicht wurden.

zur Verfügung stehenden Quelltexte [A 161–164] verwiesen. Darüber hinaus beschreiben die Publikationen von Besting et al. [B 11, 12] den Entwurf und die Umsetzung von Kommunikationsbibliotheken am Beispiel der Bibliotheken *SDCLib/C* und *SDCLib/J*, bzw. deren Vorgängerversionen.

Die spezifischen Anforderungen an die Kommunikationsbibliotheken hinsichtlich Latenz, Determinismus, Zuverlässigkeit etc. hängen vom medizinischen Anwendungsfall des Medizingerätes und auch dessen Sicherheitsklassifikation ab. Die Anforderungen reichen von harter Echtzeit über weiche Echtzeit bis hin zu Anwendungen ohne Echtzeitanforderungen; die einzuhaltenden maximalen Antwortzeiten von Sekunden/Minuten (z. B. Dokumentationsaufgaben) bis in den Submillisekundenbereich (z. B. geschlossene Regelschleifen). Daher wird für diese Arbeit eine anwendungsfallunabhängige Evaluation mit generischen *Service Providern* und *Consumern* durchgeführt.

In diesem Kapitel werden zunächst die allgemeinen Anforderungen an die Kommunikationslatenz von *IEEE 11073 SDC*-Implementierungen analysiert. Diese basieren auf einer Literaturrecherche zu menschlichen Reaktionszeiten und der Wahrnehmung von Verzögerungen. Berücksichtigt werden ebenso spezifizierte Anforderungen an medizinische und nicht-medizinische Systeme. Anschließend werden die Evaluationsumgebung (Abschnitt 5.3) und Messmethodik (Abschnitt 5.4) erläutert. In Abschnitt 5.5 wird eine detaillierte Analyse der Messergebnisse vorgenommen. Das Kapitel schließt mit einer Zusammenfassung und Einordnung der Ergebnisse (Abschnitt 5.6).

5.2 Menschliche Reaktionszeiten, Wahrnehmungen und Anforderungen

Reaktions- und Kommunikationszeiten technischer Systeme sind im Kontext des Anwendungsfalls zu betrachten. Die menschliche Reaktionszeit ist im Allgemeinen höher als die Reaktionszeit von Maschinen (ohne Aktorik). Die im Fokus dieser Arbeit stehenden Anwendungsfälle sind dadurch gekennzeichnet, dass sich ein Mensch in der Reaktionskette befindet (*Human-in-the-Loop*). Die Zeit zwischen dem Betätigen einer Taste zum Erhöhen der Lichtintensität und der (wahrnehmbaren) Änderung am Gerät muss also dahingehend bewertet werden, dass der Vorgang für den Menschen ohne eine spürbare Verzögerung („instantan“), bzw. mit einer geringen wahrnehmbaren Verzögerung erfolgt. Soll via *IEEE 11073 SDC* die Auslösung bzw. Beendigung der Auslösung einer sicherheitskritischen Gerätefunktionalität erfolgen, wie etwa die Energieabgabe eines medizinischen Lasers oder eines HF-Chirurgiegerätes, so steigen diese Anforderungen. In jedem Fall addiert sich die Kommunikationszeit des Systems zur Reaktionszeit des Menschen. Somit ist einerseits eine möglichst geringe Kommunikationszeit anzustreben, andererseits ist die Kommunikationszeit im Verhältnis zur menschlichen Reaktionszeit bzw. Wahrnehmungsschwelle zu bewerten. Des Weiteren sind die Verarbeitungszeiten der Medizingeräte zu betrachten. Diese können aufgrund unterschiedlicher physikalischer und mechanischer Eigenschaften stark voneinander abweichen. Daher wird dieser Aspekt in der vorliegenden Arbeit nicht betrachtet.

5.2.1 Kontextunabhängige Reaktionszeitanalysen

Die menschliche Reaktionszeit hängt von einer Vielzahl von Faktoren ab: Art und Intensität des Stimulus, Alter und Verfassung des Menschen, Umgebungsbedingungen, Komplexität der Reaktion etc. Eine umfassende Literaturrecherche des Stands der Wissenschaft stellt Kosinski [A 165] bereit. Für *einfache* Reaktionszeitexperimente (*Simple Reaction Time*), bei denen es nur einen Stimulus und eine mögliche Reaktion gibt, liegt die Reaktionszeit auf akustische Reize bei 140 – 160 ms und auf visuelle Reize bei 180 – 200 ms [A 166–169]. In einer umfassenden Literaturrecherche und eigenen Experimenten ermitteln Woods et al. [A 170] allgemeine Reaktionszeiten zwischen 231 ms und 397 ms.

Eckner et al. [A 171] nutzen einen einfachen Versuchsaufbau für die Bestimmung der Reaktionszeit von College-Football-Spielern: In der Ausgangsposition hält der Versuchsleiter einen Stab senkrecht in die Luft. Der Proband umfasst den Stab mit einer geöffneten Hand an einer definierten Stelle,

ohne den Stab dabei zu berühren. Der Versuchsleiter lässt den Stab fallen und der Proband versucht, diesen im Fallen so schnell wie möglich zu greifen. Mittels des Fallweges wird die Reaktionszeit gemessen. Im Experiment mit 94 männlichen Football-Spielern im Alter von 18 – 23 Jahren (Durchschnitt: 19,9 Jahre; Standardabweichung: +/- 1,5 Jahre) wurde eine Reaktionszeit von durchschnittlich 203 ms (Standardabweichung: +/- 20 ms) ermittelt. Somit wurden die älteren Untersuchungen in ihrer Größenordnung bestätigt.

In der Studie von Eckner et al. [A 171] wurde die Reaktionszeit derselben Probanden ebenfalls mit einem computergestützten System gemessen. Sie lag bei durchschnittlich 268 ms (Standardabweichung: +/- 44 ms). Im Vergleich zum Versuchsaufbau mithilfe eines fallenden Stabs zeigt sich eine signifikant höhere Reaktionszeit. In einem ebenfalls computergestützten Experiment mit 120 Studierenden im Alter von 18 – 20 Jahren ermittelten Jain et al. [A 172] eine einfache Reaktionszeit von 247,60 ms (+/- 18,54 ms) für visuelle Reize bzw. von 228,01 ms (+/- 16,49 ms) für akustische Reize.

Spätestens mit der Endoskop-basierten minimalinvasiven Chirurgie haben computergestützte Methoden Einzug im OP-Saal gehalten. In klassischen, offen-invasiven Eingriffen kommen ebenso technische Systeme wie z. B. HF-Chirurgiegeräte zum Einsatz. Die Charakteristik der Operation ist dennoch eine deutlich andere. Daher ist davon auszugehen, dass Aspekte der computergestützten und nicht-computergestützten Experimente auf den klinischen Alltag übertragbar sind und Unterschiede in die Bewertung einfließen müssen.

5.2.2 Reaktionszeitanalysen im medizinischen Umfeld

Pfister et al. [A 173] untersuchten die Reaktionszeit von Ärztinnen in einem stark vereinfacht simulierten neurochirurgischen Eingriff. Der Proband schaut durch ein chirurgisches Mikroskop auf zwei LEDs. Mit dem Aufleuchten einer roten LED beginnt das Experiment. Der Proband betätigt daraufhin einen Hand- bzw. Fußschalter. Nach einer zufälligen Zeit (zwischen 2 und 15 Sekunden) leuchtet eine grüne LED auf. Die Testaufgabe besteht darin, den Hand- bzw. Fußschalter wieder loszulassen, wenn die grüne LED aufleuchtet.

Das Experiment wurde mit 47 Ärztinnen unterschiedlichen Alters, Geschlechts und Erfahrungslevels durchgeführt. Für das Loslassen des Handschalters wurde eine durchschnittliche Reaktionszeit von 318,24 ms (Standardabweichung: +/- 51,13 ms) und für den Fußschalter von 328,69 ms (Standardabweichung: +/- 48,70 ms) ermittelt.

5.2.3 Latenzen in Multi-Player-Computerspielen und virtueller Realität

Der Einfluss der Netzwerklatenz auf das Spielerlebnis und die Performanz wurde für diverse netzwerkbasierte Multi-Player-Computerspiele untersucht. Die Ergebnisse sind insofern von Relevanz für die vorliegende Arbeit, da explizit die Verzögerung der Netzwerkkommunikation untersucht wird. Die zusätzliche Verzögerung durch die Informationsverarbeitung des Spiels selbst wird nicht betrachtet. Dies gilt analog für die interne Verarbeitungszeit der Medizingeräte, auf die sich die vorliegenden Analysen beziehen.

In Abhängigkeit von der Art des Spiels zeigen sich teils große Unterschiede. Im Bereich der sogenannten *Ego-Shooter*[5.2] kommt beispielsweise Armitage [A 174] für das Spiel „*Quake 3*“ zu dem Schluss, dass Spieler Server mit einer *Ping-Latenz* von unter 150 – 180 ms bevorzugen. Konsistent empfehlen Beigbeder et al. [A 175] für den *Ego-Shooter* „*Unreal Tournament 2003*“ Server mit einer *Ping-Latenz* von unter 150 ms. In weiterführenden Untersuchungen wird festgestellt, dass Verzögerungen von 75 ms von den Spielern wahrgenommen werden, das Spielverhalten ab 100 ms weniger angenehm ist und Latenzen ab 200 ms als störend empfunden werden [A 175].

[5.2] Computerspiel, bei dem der Spieler aus der Ich-Perspektive heraus mit (Schuss-)Waffen Gegner bekämpft.

Im Bereich der Sportspiele untersuchten Pantel und Wolf [A 176] Autorennspiele. Gemessen am Spielerfolg kommen sie zu dem Schluss, dass *Latenzen* unter 50 ms unkritisch sind und *Latenzen* von mehr als 100 ms vermieden werden sollten. Die Befragung nach dem subjektiven Empfinden der Spieler kam zu folgendem Ergebnis: Eine Verzögerung von 50 ms ist kaum wahrnehmbar und das Fahrverhalten ist praktisch unbeeinflusst. 100 ms sind akzeptabel, wenn keine hohen Anforderungen an realistisches Verhalten gestellt werden. Die Verzögerung ist spürbar, hat aber kaum optisch sichtbare Auswirkungen. Bei 200 ms Verzögerung ist diese klar wahrnehmbar, das Auto kann aber kontrolliert werden, wenngleich das Verhalten unrealistisch ist. 500 ms Verzögerung werden als nicht akzeptabel empfunden, da das Auto nicht mehr kontrolliert werden kann und Aktionen und Reaktionen nicht mehr zusammenpassen. Deutlich geringere Anforderungen evaluierten Nichols und Claypool [A 177] für ein *American Football*-Spiel: Die Autoren stellten keinen merklichen Performanzunterschied bei Latenzen bis zu 500 ms fest.

Mithilfe eines vereinfachten Experiments untersuchten Raaen et al. [A 178] wahrnehmbare Verzögerungen zwischen einer Aktion und einem visuellen Feedback von Cloud-basierten Computerspielen. Die Untersuchung mit 41 Teilnehmern im Alter von 19 bis 43 Jahren kommt zu dem Ergebnis, dass im Durchschnitt Verzögerungen unter 51 - 90 ms nicht wahrgenommen werden. Einzelne Teilnehmer konnten hingegen Verzögerungen von 26 - 40 ms wahrnehmen. Aussagen darüber, ob diese Verzögerungen als störend empfunden werden, werden nicht getroffen.

Im Bereich der virtuellen Realität wird untersucht, inwiefern sich Verzögerungen darauf auswirken, ob man sich selbst als Urheber der eigenen Aktion fühlt (englisch „Sense of Agency“ [A 179], auch „Gefühl der Entscheidungsfreiheit“) und ob Aktionen als gleichzeitig empfunden werden. Beispielsweise kommen Waltemate et al. [A 180] zu dem Ergebnis, dass für Verzögerungen bis zu 125 ms mehr als 92 % der Probanden sich als Urheber der Aktionen ihres Avatars fühlen und dabei den Körper des Avatars ihrem eigenen zugehörig fühlen. Über 80 % der Probanden bewerteten ihre eigenen Bewegungen und die des Avatars als gleichzeitig. Die Ergebnisse bestätigen damit Experimente von Farrer et al. [A 181] und Rohde et al. [A 182], die diese Aspekte für einfache motorische Aktionen (Drücken einer Taste) und einfache Reaktionen des Testsystems (Veränderung auf einem Bildschirm) untersuchen.

5.2.4 Allgemeine und medizinische Latenzanforderungen

Für den Anwendungsbereich von Telefonie und Videokonferenzen sind die Anforderungen an die tolerierbare Latenz breit untersucht. So können Menschen bis zu einer Ende-zu-Ende-Verzögerung von 50 ms bei einer Sprachübertragung nicht zwischen einem direkten Gespräch und einem Telefonat unterscheiden. Bei 100 ms besteht ein kaum merklicher Unterschied. Ebenso werden Verzögerungen von 100 ms bei der Videotelefonie als lippensynchron wahrgenommen [A 183].

Für medizinische Anwendungsfälle geht man in der Literatur beispielsweise für die Vitalparameterüberwachung von einer tolerierbaren Latenz von 250 – 300 ms für EKG-Daten und bis zu 1 s für Parameter wie die Herzfrequenz aus [A 184–186].

Verschiedene Studien untersuchen den Einfluss der Verzögerung auf Tele-Operationen mithilfe eines entsprechenden Operationssimulators. Sie kommen zu dem Ergebnis, dass Verzögerungen von maximal 100 – 200 ms [A 187–189] für Tele-Operationen optimal sind und eine *Round Trip Time (RTT)* von $\leq$ 300 ms akzeptabel ist [A 186–188]. Für teleoperative Anwendungen mit haptischem Feedback sind Verzögerungen von unter 10 ms erforderlich [A 189–192]. Die angesetzte maximale Latenz liegt damit unterhalb der evaluierten Reaktionszeit von durchschnittlich über 300 ms von Ärztinnen in einer vereinfachten Simulation einer neurochirurgischen Operation (siehe Abschnitt 5.2.2).

5.2.5 Latenzangaben in Normen

Vorhandene Normen aus dem Bereich der Medizingeräte bzw. deren Vernetzung geben nur wenig Aufschluss über die einzuhaltenden Kommunikationszeiten. Daher werden im folgenden Abschnitt ausgehend von der Recherche indirekt Rückschlüsse auf die Anforderungen an Medizingeräte gezogen.

Der aktuelle Entwurf einer neuen Version der Norm *DIN EN 60601-2-54/A2* [A 193] für Röntgengeräte von 2017 fordert eine maximale Verzögerungszeit für das Beenden der Bestrahlung von 100 ms. Der allgemeinere Prüfgrundsatz *GS-ET-22* [A 194] spezifiziert eine maximale Verzögerung für ein Not-Aus vom 200 ms.

Der *IEEE Standard 1278.2 „Distributed Interactive Simulation (DIS) – Communication Services and Profiles“* definiert Anforderungen für die Kommunikation zwischen Subsystemen einer verteilten interaktiven Simulation. Für die Kommunikationsverzögerung wird eine Latenz (*Transport-Layer-to-Transport-Layer*-Latenz) von 100 ms (*High QoS*) bzw. 300 ms (*Normal QoS*) bei einem maximalen Jitter von 50 ms spezifiziert.

In Abhängigkeit vom jeweiligen Anwendungsfall können, bezogen auf die Summe aus Verarbeitungs- und Kommunikationszeit, folglich maximale Latenzen von 100 ms – 200 ms abgeleitet werden.

5.2.6 Schlussfolgerungen

Aus den verschiedenen Untersuchungen bzw. Vorgaben mehrerer Domänen lassen sich grobe Größenordnungen für die Anforderungen an die Latenz der Kommunikation von Medizingeräten ableiten. In Tabelle 5.1 wird die Literaturrecherche dieses Abschnitts zusammengefasst. Werden verschiedene Zeiten für denselben Aspekt diskutiert, so ist die geringste Zeitdauer angegeben, was einer höheren resultierenden Anforderung entspricht.

Das Ziel in Bezug auf vernetzte medizinische Systeme muss es sein, dass durch die Kommunikationsverzögerung keine negativen Auswirkungen auf die Patientin entstehen und dass das Verhalten des Operierenden nicht negativ beeinflusst wird. Die mögliche Beeinflussung ist in Anwendungsfällen wie der chirurgischen Robotik sehr offensichtlich. Kann die Auslösung des Koagulierens eines HF-Chirurgiegerätes über einen vernetzten Fußschalter zeitlich nicht präzise erfolgen, kommt es ebenfalls zu einer Patientengefährdung. Aber auch von deutlich einfacheren und vom grundlegenden Risikomanagement her weniger risikoreichen Anwendungen, zum Beispiel der Verstellung von Parametern wie der Lichtintensität, können negative Einflüsse ausgehen: Häufen sich (deutlich) wahrnehmbare Verzögerungen, so ist von einer sinkenden Nutzerzufriedenheit auszugehen, was folglich ein höheres Stresslevel nach sich ziehen kann. Zudem können Verzögerungen zu ungewollten Mehrfachbetätigungen von Verstelloperationen und entsprechenden resultierenden Gefährdungen führen.

Ausgehend von den Analysen kann eine Verzögerung von unter 100 ms als wünschenswert für den überwiegenden Anwendungsbereich der *IEEE 11073 SDC*-Normenfamilie abgeleitet werden. In diese Schlussfolgerung fließen mehrere Werte ein: Bei 100 ms beginnen Beeinträchtigungen bei Computerspielen. Gerade bei Endoskop- und Mikroskop-basierten Operationen, bei denen der Operationsraum durch ein technisches System betrachtet wird, können Parallelen zu Computerspielen gezogen werden, bei denen die Interaktion über einen Bildschirm erfolgt. Auch die Untersuchungen zur „Sense of Agency“ im Bereich der virtuellen Realität ($\lessapprox$ 125 ms) sind von Interesse, etwa für die Bewertung von Fernsteuerungseigenschaften, wie beispielsweise dem Verstellen von Geräteparametern. Wie beschrieben ist für diesen Anwendungsfall die Wahrnehmung einer „instantanen“ Reaktion des Systems wichtig. Der Wert von 100 ms liegt ebenso unterhalb der Forderungen für Tele-Operationen, den allgemeinen Reaktionszeiten auf akustische und visuelle Reize und hält ebenso die Anforderungen aus dem Bereich der Telekommunikation und der Norm *DIN EN 60601-2-54/A2* für Röntgengeräte ein. Anwendungsbereiche mit Anforderungen an eine besonders geringe Kommuni-

Tabelle 5.1: Zusammenfassung der recherchierten Reaktions- bzw. Wahrnehmungszeiten und Anforderungen. Bei mehreren Quellen wird jeweils der geringste Wert und folglich die höchste Anforderung angegeben.

	Beschreibung	Zeit	Referenz
durchschnitt. Reaktionszeit	auf akustische Reize	$\gtrapprox 140\,\text{ms}$	[A 166–169]
	auf visuelle Reize	$\gtrapprox 180\,\text{ms}$	[A 166–169]
	mit Augen-Hand-Koordination	$\gtrapprox 200\,\text{ms}$	[A 171]
	bei computergestützter Messung	$\gtrapprox 245\,\text{ms}$	[A 171, 172]
Computer-spiele	„niedrigste“ Wahrnehmbarkeitsschwelle	$\approx 25\,\text{ms}$	[A 178]
	durchschnitt. Wahrnehmbarkeitsschwelle	$\approx 50\,\text{ms}$	[A 175, 176, 178]
	Beginn einer Beeinträchtigung	$\approx 100\,\text{ms}$	[A 175, 176]
	Beginn eines störenden Einflusses	$\approx 200\,\text{ms}$	[A 175, 176]
	virtuelle Realität: gefühlte Aktionskontrolle / „Sense of Agency“	$\lessapprox 125\,\text{ms}$	[A 180–182]
Kommu-nikation	Telefonie (Wahrnehmbarkeitsschwelle)	$\approx 50\,\text{ms}$	[A 183]
	Telefonie (Empfindung direkter Kommunikation)	$\leq 100\,\text{ms}$	[A 183]
	lippensynchrone Videotelefonie	$\leq 100\,\text{ms}$	[A 183]
medizinische Anwendungen	Reaktionszeit neurochirurgischer Eingriff	$\gtrapprox 315\,\text{ms}$	[A 173]
	Beendigung der Bestrahlung von Röntgengeräten	$\leq 100\,\text{ms}$	[A 193]
	Vitalparameterüberwachung (z. B. EKG)	$\leq 250\,\text{ms}$	[A 185, 186, 189]
	Vitalparameterüberwachung (z. B. Puls)	$\leq 1.000\,\text{ms}$	[A 184, 186, 189]
	Tele-Operationen	$\leq 150\,\text{ms}$	[A 187, 188]
	Tele-Operationen mit haptischem Feedback	$\leq 10\,\text{ms}$	[A 189–192]

kationsverzögerung, wie Tele-Operationen mit haptischem Feedback, stehen (derzeit) nicht im Fokus von *IEEE 11073 SDC*.

Verzögerungen bis 200 ms können noch als akzeptabel angesehen werden. Bei diesem Wert beginnt beispielsweise der störende Einfluss der Verzögerung auf Computerspiele. Zudem liegt der Wert unterhalb der beschriebenen Latenz für medizinische Anwendungen (siehe Abschnitt 5.2.4), beispielsweise einer maximalen Verzögerung von 250 ms für Vitalparameter wie dem EKG. Ebenso werden die experimentell ermittelten Reaktionszeiten eingehalten, etwa die ca. 315 ms des neurochirurgischen, Mirkoskop-basierten Experiments.

Einschränkend sei auf zwei nicht berücksichtigte Aspekte der Recherchen dieses Abschnitts hingewiesen: Zum einen werden keine Aussagen darüber getroffen, ob es sich bei den angegebenen Zeitschranken um harte oder weiche Anforderungen im Sinne der Echtzeitdefinition handelt. Zum anderen ist zu beachten, dass die Kommunikationslatenz nur einen Teil der Gesamtverzögerung des Systems darstellt. Die Gesamtverzögerung ist die Summe aus der Kommunikationsverzögerung, der Verarbeitungsdauer, die über die *IEEE 11073 SDC*-spezifische Verarbeitung hinausgeht, ggf. mechanische Reaktionen des Systems etc. Dabei ist zu berücksichtigen, dass im Fall von mechanisch/-physikalisch bedingt langen Zeiträumen bis zu einer gewünschten Systemreaktion typischerweise eine zweigeteilte Abarbeitung erfolgt. Zunächst wird die Annahme eines neuen Soll-Wertes bestätigt, sodass dem Nutzenden in geeigneter Form mitgeteilt werden kann, dass das Kommando verarbeitet wurde. Die Zeit zum Erreichen des neuen Ist-Wertes ist damit nicht Teil der Gesamtverzögerung, die für die Kommunikation von Relevanz ist.

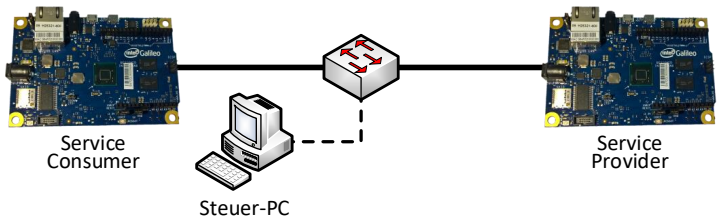

Abbildung 5.1: Schematische Darstellung des Testaufbaus, exemplarisch mit Intel Galileo Boards. (Abbildung nach *Fig. 1* aus [B 17]. Original © 2017 IEEE.)

5.3 Evaluationsumgebung

Die Messungen werden in einer isolierten Testumgebung vorgenommen. Im Kern umfasst sie genau einen *Service Provider* und einen *Service Consumer*. Die Testumgebung wird in Abbildung 5.1 dargestellt. Beide logischen Rollen werden als eigenständige, physikalische Netzwerkteilnehmer realisiert. Verbunden werden sie mittels eines *D-Link DES-1008D* 100 Mbit/s *Fast Ethernet Switches* [5.3] und *CAT5 Ethernet*-Kabeln mit einer Länge von einem Meter.

Zusätzlich wird ein Steuer-PC genutzt, um die Messungen zu starten und die Messergebnisse (Zeitstempel) nach dem Ende der Messungen abzurufen. Während die Messungen stattfinden wird der Steuer-PC physikalisch vom Testnetzwerk getrennt. Hierzu wird das Netzwerkkabel entfernt.

Die isolierte Betrachtung zweier Kommunikationsteilnehmer hat den Vorteil, dass die *Middleware*-bezogenen Latenzen ohne Störeinflüsse und Probleme wie Netzwerk-Overload, Paketverluste etc. evaluiert werden können. Die Netzwerkteilnehmer wurden mit statischen IP-Adressen konfiguriert, sodass keine Einflüsse durch *Dynamic Host Configuration Protocol (DHCP)*-Datenverkehr entstehen.

5.3.1 Evaluationsplattformen

Die Performanzmessungen werden auf verschiedenen Hardwareplattformen durchgeführt, die in Tabelle 5.2 zusammengefasst sind. Es kommt Hardware mit zwei verschiedenen Instruktionssätzen (*x86* und *ARM*) und verschiedenen Abstufungen hinsichtlich der Leistungsparameter Rechenleistung (*CPU*) und Arbeitsspeicher (*RAM*) zum Einsatz: *Intel Galileo Board (GBoard)* sowie *Raspberry Pi (RPi) 1, 2* und *3*. Diese Hardwareplattformen stehen stellvertretend für die Heterogenität der genutzten Systeme in realen klinischen Umgebungen. Professionelle Medizingeräte, die im Fokus von *IEEE 11073 SDC* stehen, sind in Bezug auf Rechenleistung, Speicher, Energieaufnahme etc. typischerweise nicht so stark ressourcenbeschränkt wie Systeme in anderen Domänen oder im Bereich der persönlichen Medizin-/Fitnessgeräte. Daher können insbesondere das *GBoard* und der *RPi 1* als geeignete Repräsentanten für leistungsschwache Vernetzungsteilnehmer angesehen werden.

Für zwei der untersuchten Kommunikationsbibliotheken (siehe Kapitel 5.3.2) ist eine geeignete *Java Virtual Machine (JVM)* erforderlich. Da die *Oracle JVM* und *OpenJDK* die größte Verbreitung haben, wird die Nutzung dieser beiden *JVMs* für *Java 7* und *Java 8* untersucht. Aus Gründen der Security sollte die genutzte *JVM* aktuell sein. Daher werden die zur Publikationszeit der zugrundeliegenden Arbeit [B 17] aktuellen *Stable Release*-Versionen genutzt (siehe Tabelle 5.2)[5.4].

[5.3] Die genutzten Hardwareplattformen verfügen überwiegend lediglich über 100 Mbit/s *Ethernet*-Schnittstellen (siehe Kapitel 5.3.1). Der Raspberry Pi 3 kann die *Gigabit*-Schnittstelle aufgrund seiner Anbindung nicht vollumfänglich nutzen.

[5.4] Es sei angemerkt, dass während der Evaluation Performanzunterschiede auch zwischen *Minor*-Updates der *JVMs*, die lediglich Fehlerbehebungen und Sicherheitsupdates zur Verfügung stellen, festgestellt wurden.

Tabelle 5.2: Übersicht der zur Evaluation genutzten Hardware-Plattformen. *64 MByte reserviert für die *Graphics Processing Unit (GPU)*; **Wird aufgrund des Betriebssystem-Kernels im *32-Bit ARMv7*-Modus betrieben. Abkürzung: Timer Aufl. – Timer Auflösung (Minimal mögliche Zeitspanne zwischen zwei Zeitstempeln in C++ und Java). (Tabelle nach [B 17]. Original © 2017 IEEE..)

Plattform	**Architektur**	**CPU**	**Takt Rate**	**Timer Aufl.**	**RAM**	**Betriebssystem**	**Java Virtual Machine (JVM)**
Intel Galileo Board (*GBoard*)	32-Bit x86 (Intel Pentium-class)	Intel ® Quark SoC X1000 (1 Kern)	400 MHz	$5 \cdot 10^{-3}$ ms	256 MByte	Debian GNU Linux 8.4 Jessie (Kernel 3.8.7)	Oracle JVM 1.8.0_131 Oracle JVM 1.7.0_80 OpenJDK 1.8.0_131 OpenJDK 1.7.0_101
Raspberry Pi 1 Modell B (*RPi 1*)	32-Bit ARMv6	ARM1176JZF-S	700 MHz	$1 \cdot 10^{-3}$ ms	512 MByte*	Raspbian GNU Linux 8.0 Jessie Lite (Kernel 4.4.50-v7)	Oracle JVM 1.8.0_131 Oracle JVM 1.7.0_75 OpenJDK 1.8.0_40 OpenJDK 1.7.0_131
Raspberry Pi 2 Modell B V1.1 (*RPi 2*)	32-Bit ARMv7	ARM Cortex-A7 (4 Kerne)	900 MHz	$6 \cdot 10^{-4}$ ms	1024 MByte*		
Raspberry Pi 3 Modell B V1.2 (*RPi 3*)	64-Bit ARMv8**	ARM Cortex-A53 (4 Kerne)	1200 MHz	$5 \cdot 10^{-4}$ ms			

Sofern im Verlauf der Arbeit keine explizit anderen Plattformeinstellungen beschrieben sind, werden die Standardeinstellungen genutzt, z. B. für den Parameter *JVM*-Mode der Wert `-client` oder für den sogenannten *CPU Governor* die Einstellung `ondemand` für die *RPis* [5.5].

5.3.2 IEEE 11073 SDC-Kommunikationsbibliotheken

Zur Evaluation werden die drei Open-Source-Bibliotheken herangezogen, die im Umfeld des *OR.NET*-Projekts entstanden sind. Ihre Ursprünge liegen zum Teil bereits in Vorgängerprojekten. Seitdem erfolgt eine kontinuierliche Weiterentwicklung. Zum Zeitpunkt der Evaluation waren weder weitere offene Implementierungen bekannt, noch konnten kommerzielle Bibliotheken für eine derartige Evaluation genutzt werden[5.6].

Die Bibliotheken waren und sind in ständiger Weiterentwicklung. Für die Evaluation sollte ein Stand genutzt werden, auf dem alle untersuchten Implementierungen untereinander interoperabel sind, d. h. sich auf demselben Stand des Datenmodells befinden und etwa den gleichen Funktionsumfang besitzen. Die untersuchten Versionen haben ihre Interoperabilität im Zuge umfangreicher Demonstratoren auf den *conhIT*-Messen 2016 und 2017 in Berlin unter Beweis gestellt. Die Implementierungen basieren auf der initialen Version *Draft D6* des Datenmodells aus *IEEE 11073-10207* [5.7].

Folgende Versionen der Bibliotheken werden zur Evaluation genutzt. Zum Zeitpunkt der zugrunde liegenden Publikation [B 17] waren dies die letzten, vollständig kompatiblen Versionen:

5.5 *Frequency Scaling* ist beim *GBoard* nicht möglich.

5.6 Es sei angemerkt, dass *IEEE 11073 SDC* ein grundlegend neues Kommunikationsparadigma in die IEEE 11073 Normenfamilie einbringt. Daher sind andere IEEE 11073 Frameworks, wie etwa das auf IEEE 11073-20601 basierende *Antidote* [A 195], nicht kompatibel.

5.7 Aus der Version *Draft D5* wurde ausschließlich aus Implementierungsgründen das Element `MDIBContainer` übernommen, das aus Modellsicht nicht notwendig ist.

- *openSDC* Version „*OR.NET-BETA_06-SNAPSHOTconhit16*“ [A 196]
- *SoftICE* Version *0.96b* [A 197] (inzwischen umbenannt in *SDCLib/J*)
- *OSCLib* Version *2.0* [A 198] (inzwischen umbenannt in *SDCLib/C*)

Die in *IEEE 11073 SDC* vorgesehenen Verschlüsselungsmechanismen sind in diesen Versionen nicht durchgängig implementiert. Daher kann die Evaluation nur auf der Basis einer unverschlüsselten Kommunikation erfolgen.

5.4 Messmethodik

Für die Latenzanalyse der *IEEE 11073 SDC*-Kommunikationsbibliotheken wird die Kommunikationsumlaufzeit, im Englischen *Round Trip Time (RTT)* genannt, vom Senden einer Anfrage bis zum Empfang der Antwort, gemessen. Zusätzlich werden weitere Messpunkte innerhalb des implementierten Protokollstapels eingefügt, um tiefere Analysen zu ermöglichen.

5.4.1 Kommunikationsumlaufzeit

Die *SOMDA*-basierte Kommunikation unterscheidet zwei Kommunikationspatterns: *Request-Response* und *Publish-Subscribe*. Im ersten Fall ruft ein *Service Consumer* einen Service des *Service Providers* auf (*Request*). Der *Service Provider* sendet eine entsprechende Antwort (*Response*). Im zweiten Fall möchte der *Service Consumer* über den Zustand des *Service Providers* informiert werden. Daher werden die gewünschten Events abonniert (*Subscribe*). Der *Service Provider* sendet dann im Fall von Änderungen, oder auch periodisch, die Zustandsänderungen (*Publish*). *IEEE 11073 SDC* nutzt den *Publish-Subscribe*-Mechanismus für Benachrichtigungen über Zustands- und Beschreibungsänderungen sowie für physiologische und technische Alarme. Für Fernsteuerungsoperationen kommen beide Mechanismen zum Einsatz: Die Fernsteuerungsoperation wird per *Request-Response* ausgelöst, während die nachfolgend ausgelösten Zustandsänderungen per *Publish-Subscribe* kommuniziert werden.

Für die Evaluation wird die Umlaufzeit der Kommunikation (*RTT*) beim Auslösen einer `GetMdState` Operation gemessen. Die Operation fragt den Zustand einer bestimmten Metrik des *Service Providers* an. Entsprechend den in Abbildung 5.2 veranschaulichten Messpunkten ist die *RTT* wie folgt definiert:

$$RTT = t_6 - t_0 \tag{1}$$

Der erste Zeitstempel t_0 wird genommen, direkt bevor die Anwendungssoftware des *Service Consumers* die entsprechende Methode des *Middleware Application Programming Interfaces (APIs)* aufruft, die die Operation auslöst. Die Messung wird beendet (Zeitstempel t_6), direkt nachdem der angefragte Wert in der Anwendungssoftware des *Service Consumers* zur Verfügung steht.

Die Messung der *RTT* hat den Vorteil, dass eine Zeitsynchronisation zwischen den Kommunikationsteilnehmern nicht erforderlich ist, da beide Zeitstempel auf demselben Gerät genommen werden. Durch den Verzicht auf ein präzises Zeitsynchronisationsverfahren ist die Messmethodik von kontrollierten „Laborbedingungen“ auf reale Netzwerke übertragbar und kann ebenfalls genutzt werden, um die Kommunikation in realen Umgebungen zu untersuchen[5.8].

Die beschriebene Messung der *RTT* bildet eine *Request-Response*-Kommunikation ab. Zur Messung der Latenz von asynchronen Events nach dem *Publish-Subscribe*-Pattern ist entweder eine Zeitsynchronisation zwischen *Service Provider* und *Consumer* notwendig, da Zeitstempel auf beiden Geräten

[5.8] Im Fall einer realen Umgebung werden nicht alle Teilnehmer ihren Kommunikationsstack instrumentieren, wie es in Kapitel 5.4.2 beschrieben wird. Entsprechend wird in diesem Fall ggf. lediglich die *RTT* messbar sein.

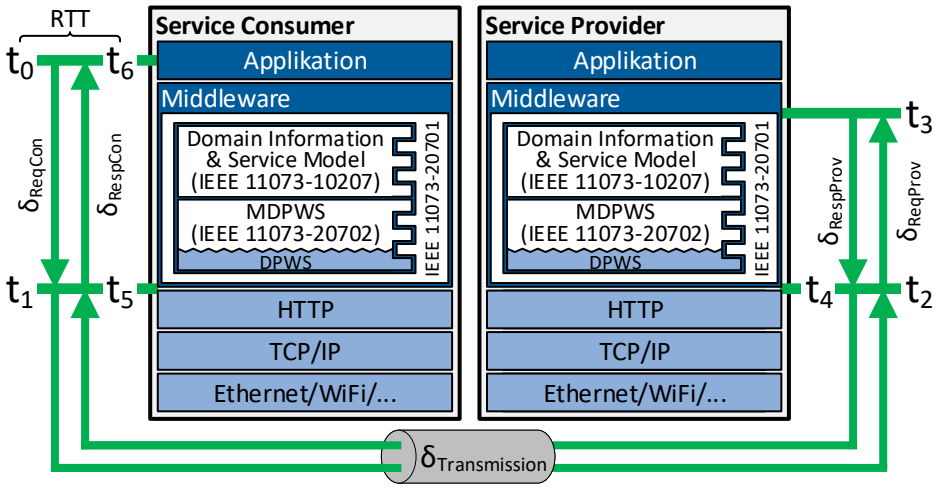

Abbildung 5.2: Visualisierung der Zeitstempel zur Instrumentierung der Kommunikationsbibliotheken. Abkürzungen: RTT – Round Trip Time; $\delta_{Transmission}$ – Übertragungszeit im Netzwerk; *Middleware*-Verzögerung: δ_{ReqCon} – Anfrageerzeugung des *Service Consumers*; $\delta_{ReqProv}$ – Anfrageverarbeitung des *Service Providers*; $\delta_{RespProv}$ – Antworterzeugung des *Service Providers*; $\delta_{RespCon}$ – Antwortverarbeitung des *Service Consumers*. (Abbildung entspricht *Fig. 2* aus [B 17] © 2017 IEEE.)

genommen und subtrahiert werden müssen, oder es kann die *http-Status*-Antwort genutzt werden. Der letzte Fall ist nahezu äquivalent zur Messung der *RTT*. Da Events technisch gesehen dieselben Mechanismen nutzen wie der *Get-Service*, werden sie in dieser Arbeit nicht gesondert betrachtet. Die Ergebnisse und gezogenen Schlüsse sind übertragbar.

5.4.2 Instrumentierung der Kommunikationsstacks

Aus der Messung der *RTT* lassen sich Schlüsse auf die Performanz des Gesamtsystems ableiten. Um einen tieferen Einblick zu erhalten, werden die Kommunikationsstacks mit weiteren Messpunkten (t_1 bis t_5) instrumentiert. Diese werden in Abbildung 5.2 visualisiert.

5.4.2.1 Definition der Messpunkte Nachdem die Anwendungssoftware die entsprechende *API*-Methode aufgerufen hat (Zeitstempel t_0), erzeugt die *Middleware*-Bibliothek die Anfrage, serialisiert die Nachricht und sendet sie an die tieferen Schichten des *Stacks*. Direkt bevor die gesamte Nachricht an die Schicht unterhalb der *IEEE 11073 SDC*-spezifischen Implementierung übergeben wird (*http*), wird der Zeitstempel t_1 genommen. Dies stellt den letzten Zeitpunkt unter der Kontrolle der Kommunikationsbibliotheken dar. Weitere Messpunkte in darunterliegenden Schichten sind somit nicht ohne Weiteres ohne Eingriffe in *Java*-Bibliotheken und Betriebssystemroutinen möglich.

Die Nachrichtenübertragung kann mittels beliebiger IP-basierter Netzwerke vollzogen werden. Im Testaufbau wird ein 100 Mbit/s *switched* Ethernet genutzt. Der nächste Zeitstempel t_2 wird vom *Service Provider* am Übergang zwischen den unterliegenden Schichten und der *IEEE 11073 SDC*-spezifischen *Middleware* genommen. Der Nachrichten-*Header* muss komplett verarbeitet worden sein, um zu entscheiden, ob eine Verarbeitung erfolgt oder nicht. Daher ist t_2 als der Zeitpunkt definiert, an dem die *Middleware* den kompletten *Header* verarbeitet, aber mit der Verarbeitung des Nachrichten-*Bodys* noch nicht begonnen hat.

Anschließend deserialisiert und verarbeitet die *Middleware* die Nachricht. Die drei untersuchten *Middleware*-Implementierungen verwalten den Gerätezustand selbstständig, sodass die Anfrage des Metrik-Zustands keine Interaktion mit der Anwendungssoftware erfordert. Entsprechend wird der Zeitstempel t_3 genommen, wenn die *Middleware* den angefragten Metrik-Zustand gefunden hat. An

diesem Punkt ist die Verarbeitung der Anfrage (*Request*) beendet und die Bearbeitung der Antwort (*Response*) beginnt.

Analog zum Zeitstempel t_1 wird t_4 an dem Zeitpunkt genommen, an dem die Nachricht an die Schicht unterhalb der *IEEE 11073 SDC*-spezifischen *Middleware* des *Service Providers* übergeben wurde. Entsprechend der Definition von Zeitstempel t_2 am *Service Provider*, wird Zeitstempel t_5 vom *Service Consumer* genommen, wenn der Nachrichten-*Header* von der *Middleware* verarbeitet wurde. Die Messung ist beendet, wenn der angefragte Zustand in der Anwendungssoftware des *Service Consumers* zur Verfügung steht (t_6).

5.4.2.2 Genauigkeit und Vergleichbarkeit der Messpunkte Die Zeitstempel t_1 bis t_5 werden innerhalb der *Middleware*-Implementierungen genommen. Solche Implementierungen sind komplexe Softwaresysteme, die ihrerseits externe Bibliotheken nutzen. Daher ist es nicht möglich, die exakt selben Messpunkte in allen Implementierungen einzufügen. Nichtsdestotrotz kann die Genauigkeit als hinreichend groß angesehen werden, sodass die Ergebnisse einerseits zwischen den Bibliotheken vergleichbar sind und andererseits eine absolute Aussagekraft haben.

5.4.2.3 Berechnung der einzelnen Verzögerungen Auf der Basis der definierten Zeitstempel können über die *RTT* (siehe Gleichung 1) hinaus weitere Verzögerungszeiten berechnet werden. Diese Verzögerungszeiten werden in Gleichung 2 dargestellt: die *Middleware*-Verzögerung des *Service Consumers* bei der Anfrageerzeugung (δ_{ReqCon}), die *Middleware*-Verzögerung des *Service Consumers* bei der Antwortverarbeitung ($\delta_{RespCon}$), die *Middleware*-Verzögerung des *Service Providers* bei der Anfrageverarbeitung ($\delta_{ReqProv}$) und die *Middleware*-Verzögerung des *Service Providers* bei der Antworterzeugung ($\delta_{RespProv}$).

$$\begin{aligned} \delta_{ReqCon} &= t_1 - t_0 \\ \delta_{RespCon} &= t_6 - t_5 \\ \delta_{ReqProv} &= t_3 - t_2 \\ \delta_{RespProv} &= t_4 - t_3 \end{aligned} \tag{2}$$

Darüber hinaus kann die Verzögerung δ_{Stack} berechnet werden, die sich aus der Verzögerung unterhalb der *IEEE 11073 SDC*-spezifischen *Middleware* bei *Service Consumer* und *Provider* und der Übertragungszeit im Netzwerk (inkl. *Switch-Delay*; in Abbildung 5.2 als $\delta_{Transmission}$ bezeichnet) zusammensetzt:

$$\begin{aligned} \delta_{Stack} &= RTT - \delta_{ReqCon} - \delta_{RespCon} - \delta_{ReqProv} - \delta_{RespProv} \\ &= (t_5 - t_1) - (t_4 - t_2) \end{aligned} \tag{3}$$

5.4.2.4 Alternative Messmethoden und -punkte Die Nutzung externer Tools wie beispielsweise *wireshark/tshark* [5.9] ist in der gegebenen Umgebung nicht sinnvoll. Der Verarbeitungsoverhead solcher Programme auf ressourcenbeschränkten Systemen hat einen zu großen Einfluss. So wurde bei der Untersuchung der *SoftICE*-Bibliothek auf einem *RPi 1* mit parallel ausgeführtem *tshark*-Programm eine um etwa 17 % erhöhte *RTT* gemessen.

Eine Analyse der Datenpakete am Switch mittels eines *Mirror-Ports*, der alle Pakete für einen passiv agierenden Analyseknoten bereitstellt, bringt keine entscheidenden zusätzlichen Erkenntnisse.

[5.9] Verfügbar unter `www.wireshark.org` (besucht am 27.10.2020).

Mit Hilfe dieser Messpunkte kann sowohl die Gesamtlatenz von *Service Provider* und *Consumer* (inkl. Latenz von Switch und Kabel) gemessen werden, als auch die Latenz, die im *Stack* unterhalb der Messpunkte t_1/t_5 bzw. t_2/t_4 entsteht. Einerseits ist die Verzögerung δ_{Stack} anteilig gering (siehe Kapitel 5.5.6) und andererseits stehen die *Stackschichten* unterhalb der *IEEE 11073 SDC*-spezifischen Implementierungen nicht im Fokus dieser Arbeit. Daher wird auf diese zusätzlichen Messpunkte verzichtet. Des Weiteren wäre Spezialhardware mit entsprechender *Mirror-Port*-Funktionalität notwendig.

Die Ende-zu-Ende-Latenz ist die Summe aus den Software-Latenzen der Applikationssoftware und des Kommunikationsstacks sowie den Hardware-Latenzen des Übertragungsmediums (z .B. Kabel) und der Netzwerkkomponenten (z. B. Switches). Während die Hardware-Latenzen in drahtgebundenen Netzwerken als weitgehend unabhängig vom Anwendungsszenario angesehen werden können, haben Forschungsarbeiten gezeigt, dass die Software-Latenzen stark vom Anwendungsfall und der genutzten Hardware abhängen [A 199]. Die Software-Latenzen unterhalb des *IEEE 11073 SDC*-spezifischen Teils der Kommunikationsstacks wurden bereits in einer Reihe von Publikationen untersucht, beispielsweise in [A 199–204]. Aufgrund dieser umfangreichen bestehenden Vorarbeiten wird für die vorliegende Arbeit auf die detaillierte Betrachtung der unteren Schichten des Kommunikationsstacks verzichtet. Entsprechend werden keine weiteren Messpunkte eingeführt.

5.4.3 Auflösung und technische Realisierung der Messpunkte

Die Zeitstempel an den definierten Messpunkten werden mit Hilfe von Mechanismen genommen, die durch die Programmiersprachen der jeweiligen *IEEE 11073 SDC-Middleware*-Implementierung bereitgestellt werden. Für die *Java*-Bibliotheken *openSDC* und *SoftICE* wird die Methode `java.lang.System.nanoTime()` genutzt, die in einem der grundlegenden *Packages* der *Java-API* bereitgestellt wird. Im Fall der *C++*-Bibliothek *OSCLib* wird die Klasse `std::chrono::steady_clock` genutzt. Als Teil der *Standard Library* können auch die hier zur Verfügung gestellten Mechanismen als Stand der Technik angesehen werden.

Ungeachtet der theoretisch möglichen Auflösung der genutzten Zeitgeber, die in den Dokumentationen beschrieben sind, kann die praktisch nutzbare Auflösung abweichen. In der Realität wird daher eine geringere Genauigkeit erreicht. Dies liegt in der tatsächlich genutzten Hardware und deren Fähigkeiten begründet. Daher wurde die tatsächliche Auflösung der genutzten Plattformen experimentell ermittelt. Die Ergebnisse sind in Tabelle 5.2 aufgeführt. Die Auflösung mit der gröbsten Granularität ist 0,005 ms. Für Messungen im Bereich von mehreren bzw. mehreren zehn Millisekunden ist diese Auflösung hinreichend. Entsprechend werden in der Arbeit die Messwerte in Millisekunden mit einer Nachkommastelle angegeben.

5.4.4 Messvorgang

Für die Evaluation der Latenzen werden 40.000 Messungen für jede Kombination aus *Middleware*-Implementierung, Hardware-Plattform und *JVM* (soweit relevant) durchgeführt. Das bedeutet, dass die präsentierten Angaben zu Median, Standardabweichung etc. auf der Basis von 40.000 Werten berechnet werden. Für die Basisevaluation wurden entsprechend 1.120.000 Messungen durchgeführt, die jeweils aus sieben Zeitstempeln (t_0 bis t_6) bestehen. Darüber hinaus wurden weitere Messreihen durchgeführt, um etwa den Einfluss der XML Schema-Validierung oder den Einfluss der *Java Just-in-Time (JIT)* Kompilierung[5.10] [A 17] zu untersuchen. Diese zusätzlichen Evaluationen resultieren in 400.000 weiteren Messungen.

Im Standardfall wird eine Inter-Messverzögerung von 20 ms genutzt, d. h., dass zwischen dem Zeitpunkt t_6 und damit dem Ende von Messung n und dem Beginn der Messung $n+1$ (Zeitpunkt t_0) eine

[5.10] Siehe auch Glossareintrag zur *Just-in-Time Compilation*.

Wartezeit von 20 ms liegt. Für die Evaluation in Kapitel 5.5.5 wird diese Inter-Messverzögerung für einige Messreihen verändert.

5.4.5 Nachverfolgbarkeit der Ergebnisse

Um die Nachverfolgbarkeit und Reproduzierbarkeit der Ergebnisse zu ermöglichen, werden die Rohdaten der Messungen, die instrumentierten *Middleware*-Implementierungen sowie die genutzten Implementierungen der *Service Consumer* und *Service Provider* öffentlich zur Verfügung gestellt[5.11].

5.5 Evaluation

5.5.1 Tukey Boxplots

„*Tukey Boxplots*“, benannt nach dem US-amerikanischen Statistiker *John Wilder Tukey*, sind Diagramme, die sich zur Darstellung von Verteilungen eignen und damit zur Veranschaulichung von Messdaten der vorliegenden Art. Die *Box* repräsentiert den Bereich zwischen dem 25 %-Quartil und dem 75 %-Quartil, d. h., die „mittleren“ 50 % der Werte liegen innerhalb dieser *Box*. Der *Median* wird als horizontale Linie innerhalb der *Box* dargestellt. Für „*Tukey Boxplots*“ repräsentieren die *Whisker* (vertikale Linien oberhalb und unterhalb der *Box* mit horizontaler Linie an den Enden) den höchsten bzw. geringsten Wert, der sich im 1,5-fachen des Interquartilsabstands, englisch *Interquartile Range (IQR)*, befindet. Der *IQR* ist definiert als die Differenz zwischen dem 75 %-Quartil und dem 25 %-Quartil. Werte außerhalb der *Whisker* werden als statistische Ausreißer angesehen. Sie werden als Kreuze im Diagramm dargestellt. Da ein Boxplot 40.000 Messwerte repräsentiert, erscheinen die Ausreißer ggf. als gepunktete Linien. Mittels der horizontalen *Whisker*-Linienabschlüsse können die Ausreißer klar identifiziert werden.

Die statistische Definition von Ausreißern nach *Tukey* impliziert, dass die Ausreißer in Abhängigkeit der Verteilung der Messwerte berechnet werden. Für reale Anwendungsfälle ist dieses Vorgehen ggf. nicht aussagekräftig. Feste Grenzwerte, die etwa aus der menschlichen Reaktionszeit oder technischen (Echtzeit-)Anforderungen abgeleitet werden, können als zielführender angesehen werden. Für eine anwendungsfallunabhängige Betrachtung wie in dieser Arbeit werden jedoch keine festen Grenzwerte zur Ermittlung von Ausreißern definiert.

5.5.2 Round Trip Time-Messungen

Einen Überblick der gemessenen *RTT*-Werte gibt Abbildung 5.3. In Tabelle 5.3 werden Median, Standardabweichung und Maximum der *RTT*-Messungen dargestellt. Im Gegensatz zu Abbildung 5.3 erfolgt die Gruppierung nach Hardwareplattformen. Zur Komplexitätsreduzierung werden nur die Werte für die *JVM* mit der jeweils besten Performanz gezeigt. Eine Gesamtübersicht aller Messreihen befindet sich in Anhang F. In den Tabellen F.1, F.2 und F.3 können zusätzlich auch das Minimum und die zehn höchsten Werte nachgelesen werden.

5.5.2.1 Abgleich mit gestellten Anforderungen In Abschnitt 5.2 werden Anforderungen an eine interoperable Medizingerätevernetzung in Bezug auf die Kommunikationslatenz abgeleitet. Als Schlussfolgerung aus der Literaturrecherche wird eine Latenz von unter 100 ms als wünschenswert und unter 200 ms als akzeptabel angesehen. Stellt man diese Anforderungen den Messergebnissen gegenüber, so ist zu erkennen, dass *IEEE 11073 SDC*-Implementierungen diese Anforderungen einhalten können. Es sei einschränkend darauf hingewiesen, dass keine Evaluation von verschlüsselter Kommunikation erfolgen konnte.

[5.11] Verfügbar unter: `https://gitlab.amd.e-technik.uni-rostock.de/middleware_latency_measurements/mod_libs_data`.

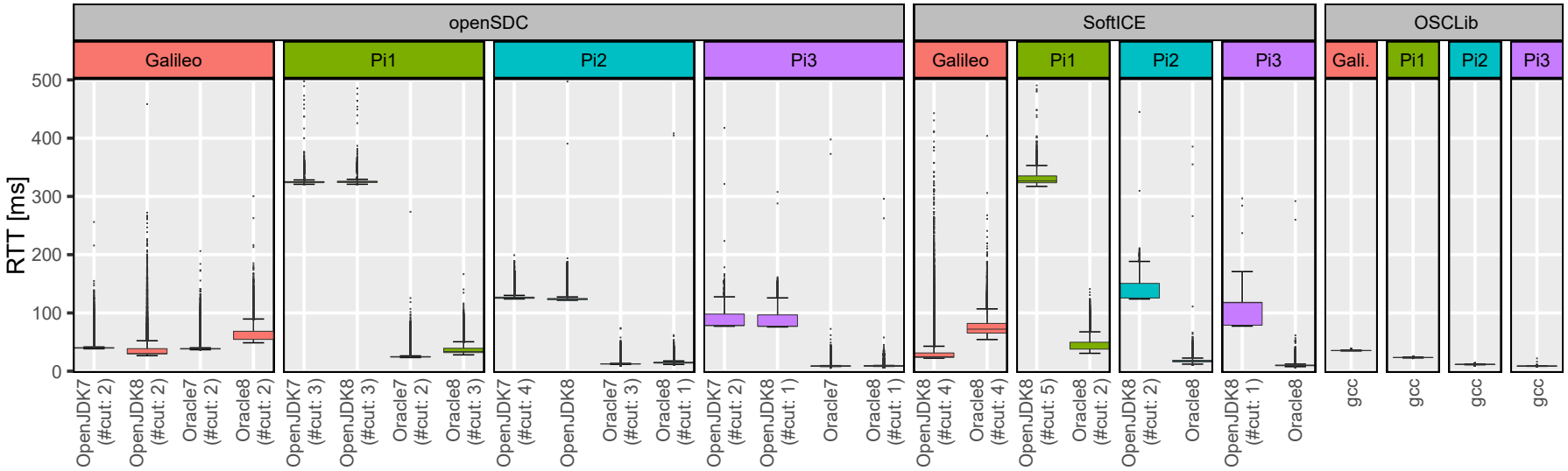

Abbildung 5.3: *„Tukey boxplots“* der *Round Trip Time (RTT)*-Messungen für die Kombinationen aus betrachteten Kommunikationsbibliotheken, Hardwareplattformen und *JVMs*. Zur besseren Lesbarkeit ist die Y-Achse bei 500 ms abgeschnitten; die Anzahl von nicht dargestellten Werten ist als *„#cut“* angegeben. Jede Box entspricht 40.000 Messungen, mit der Einschränkung der Werte über 500 ms, einschließlich aller in Abschnitt 5.5 beschriebenen Aspekte. Das vollständige Diagramm befindet sich in Anhang F in Abbildung F.1.
(Abbildung entspricht *Fig. 3* aus [B 17] © 2017 IEEE.)

Tabelle 5.3: Ergebnisübersicht der *Round Trip Time (RTT)*-Messungen. Werte ohne Klammern: Ergebnisse mit Standardeinstellungen der Schemavalidierung; Werte in Klammern: Ergebnisse für Schemavalidierung bei ein- und ausgehenden Nachrichten. Dargestellt sind nur die Ergebnisse für die *Java Virtual Machine (JVM)* mit der jeweils besten Performanz: [1]OpenJDK8; [2]Oracle 7; [3]Oracle 8. (Vollständige Ergebnisse siehe Tabellen F.1, F.2 und F.3 in Anhang F.) Abkürzungen: Std.-Ab. – Standardabweichung; Max. – Maximum. (Tabelle nach [B 17]. Original © 2017 IEEE..)

	Intel Galileo Board			**Raspberry Pi 1**			**Raspberry Pi 2**			**Raspberry Pi 3**		
	openSDC[1]	**SoftICE**[1]	**OSCLib**	**openSDC**[2]	**SoftICE**[3]	**OSCLib**	**openSDC**[2]	**SoftICE**[3]	**OSCLib**	**openSDC**[2]	**SoftICE**[3]	**OSCLib**
Median [ms]	30.3 (39.2)	24.9 (46.2)	35.4	24.7 (33.7)	42.1 (86.1)	23.5	12.6 (17.0)	17.4 (30.5)	11.7	8.8 (12.2)	9.8 (21.6)	8.8
Std.-Ab. [ms]	20.4 (27.4)	25.5 (37.0)	0.3	10.5 (12.8)	11.8 (15.7)	0.3	5.9 (7.0)	4.9 (6.8)	0.3	4.4 (4.7)	4.1 (6.0)	0.1
Max. [ms]	1784.9 (1458.2)	1194.4 (1339.8)	38.7	1228.9 (1313.4)	859.5 (1001.6)	25.3	630.4 (676.8)	385.5 (854.9)	14.4	397.6 (419.8)	292.0 (803.9)	21.9

Die größte im gesamten Versuchsablauf gemessene *RTT* der *C++*-Bibliothek *OSCLib* liegt bei 38,7 ms auf dem leistungsschwächsten System *(GBoard)*. Die Mediane der *RTT* für die leistungsstärkeren Systeme liegen deutlich darunter: für die *RPi 3*-Plattform mit 8,8 ms im einstelligen Millisekundenbereich. Bei keiner der vier Messreihen auf den vier Plattformen mit je 40.000 Messungen konnten somit *RTT*-Werte gemessen werden, die auch nur in der Nähe der gesetzten Anforderungen von 100 ms bzw. 200 ms liegen. Eine solche Serie von Messungen kann keine Beweise einer *harten Echtzeit*-Eigenschaft erbringen. Die *OSCLib* wurde nicht unter Echtzeitgesichtspunkten entwickelt und ohne Echtzeitbetriebssystem eingesetzt. Daher können die geringen *Standardabweichungen* und maximal gemessenen *RTTs* einen Hinweis darauf geben, dass ein entsprechendes Potential besteht.

Die *Java*-Implementierungen *openSDC* und *SoftICE* erfüllen die gesetzten Anforderungen ebenfalls. Die in Tabelle 5.3 abzulesenden *RTT*-Mediane liegen für alle untersuchten Plattformen deutlich unterhalb der 100 ms-Anforderung. Aufgrund der *Java*-spezifischen Ausreißer, auf die in Abschnitt 5.5.4 eingegangen wird, könnten aber maximal *weiche Echtzeit*-Anforderungen erfüllt werden.

5.5.2.2 Vergleich der Plattformen Die Performanz unterscheidet sich zwischen den verschiedenen Plattformen, der genutzten *JVM* und *Middleware*-Implementierung teils erheblich. In Abbildung 5.3 sind die Messungen anhand der Kommunikationsbibliotheken gruppiert, mit Untergruppierungen anhand der Hardwareplattformen. Betrachtet man die zwei *Java*-basierten Kommunikationsbibliotheken, *openSDC* und *SoftICE*, so sind (teilweise sehr große) Laufzeitunterschiede zwischen den verschiedenen *JVMs* auf derselben Hardwareplattform zu erkennen. Vergleicht man beispielsweise die Mediane der *RTT*-Messungen der *openSDC*-Bibliothek auf einem *RPi 1* unter der Nutzung der *OpenJDK8* (325,0 ms) und der *Oracle7* (24,7 ms) *JVM*, so unterscheiden sich diese um mehr als den Faktor 13.

Für die *RPis*, die auf einer *ARM*-Architektur basieren, kann generell festgestellt werden, dass die *Oracle JVMs* eine deutlich erkennbar bessere Performanz aufweisen als die *OpenJDK JVMs*. Das Verhalten zeigt, dass *Oracle* seine *Closed-Source JVMs* für *ARM*-Architekturen optimiert hat, im Vergleich zu den *OpenJDK Open-Source*-Implementierungen. Einer der Hauptunterschiede zwischen den *JVMs* ist der bei den *OpenJDK JVMs* fehlende, hoch optimierte *JIT*-Compiler für *ARM*-Architekturen. Es gibt aber inzwischen Bestrebungen, einen *JIT*-Compiler, ähnlich zum *Oracle HotSpot*, bereitzustellen [A 205]. Die unter dem *RPi*-Betriebssystem *Raspbian* zur Verfügung

stehenden *OpenJDK*-Versionen verfügen über keinen derartigen *JIT*-Compiler. Im Standardfall wird daher der Interpreter-Modus der *JVM* genutzt, der als `zero` bezeichnet wird.

Vergleicht man *RPi 1, 2* und *3* bezogen auf die Nutzung von *openSDC* und *SoftICE*, sinkt der Unterschied zwischen den verschiedenen *JVMs* mit steigender Systemleistung. Für die *openSDC*-Bibliothek auf den verschiedenen *RPis* zeigt sich ein sehr ähnliches Verhalten für die *Oracle 7* und *Oracle 8 JVM*, wobei die *Oracle 7 JVM* die beste Performanz der untersuchten *JVMs* aufweist. Da die *SoftICE*-Bibliothek *Java 8* benötigt, konnten nur zwei *JVMs* verglichen werden, wobei die *Oracle 8 JVM* die klar geringeren *RTTs* auf den *ARM*-Architekturen aufweist.

Die Ergebnisse für das *GBoard* unterscheiden sich stark, aufgrund der *x86*-Architektur. Obwohl die Performanzunterschiede nicht so klar sind wie für die *RPis*, ist erkennbar, dass die *OpenJDK 8 JVM* die bessere *RTT*-Performanz für diese Plattform aufweist. Da die *OpenJDK* eine *JIT*-Implementierung enthält, die von früheren *Oracle JVM*-Versionen abgeleitet wurde, erscheinen kleine Unterschiede nachvollziehbar. Außerdem legen die Ergebnisse nahe, dass sich die Optimierungen von *Oracle* eher auf leistungsstarke *x86*-Plattformen konzentrieren und daher möglicherweise weniger effektiv für das ressourcenbeschränkte *GBoard* sind.

Für die *OSCLib* wurden auf den Plattformen verfügbare Standardversionen des *gcc*-Compilers genutzt. Entsprechend den steigenden Systemressourcen vom *GBoard* über *RPi 1* und *RPi 2* zum *RPi 3* wurden fallende *RTTs* gemessen. Für die leistungsschwächste Plattform *(GBoard)* lag der Median der *RTT* bei 35,4 ms, für den leistungsstarken *RPi 3* bei lediglich 8,8 ms.

5.5.3 Einfluss der Schemavalidierung

Die drei untersuchten Bibliotheken haben unterschiedliche Standardeinstellungen bezüglich der Schemavalidierung[5.12] der ein- und ausgehenden XML-Nachrichten:

- *OSCLib*: ein- und ausgehende Nachrichten
- *openSDC*: eingehende Nachrichten
- *SoftICE*: keine Schemavalidierung

Während das Standardverhalten der Bibliotheken *openSDC* und *SoftICE* konfiguriert werden kann, bietet die *OSCLib* diese Möglichkeit nicht. Zur besseren Vergleichbarkeit wurden zusätzliche Messungen durchgeführt, bei denen alle Bibliotheken die Schemavalidierung für alle ein- und ausgehenden Nachrichten vornehmen. Die Ergebnisse sind in Tabelle 5.3 in Klammern dargestellt. Betrachtet man die Messungen mit der vollständigen Schemavalidierung, so liefert die *OSCLib* die mit Abstand höchste Performanz. Die *openSDC*-Bibliothek erzielt das zweitbeste Ergebnis. Während der Abstand zur *SoftICE*-Bibliothek auf dem *RPi 1* sehr groß ist (Faktor $\approx 2{,}6$), nehmen der absolute und relative Unterschied mit steigender Systemperformanz für *RPi 2* und 3 (Faktor ≈ 1.8) ab. Entgegen dieser Beobachtung ist der Performanzunterschied auf dem *GBoard* mit einem Faktor von rund 1,2 zwischen den gemessenen Medianen der *RTTs* bei *SoftICE* und *openSDC* am geringsten.

Diese Beobachtungen legen nahe, dass die Softwarelatenz auf Plattformen mit mehr Ressourcen, wie etwa dem *RPi 3*, eher von anderen Faktoren bestimmt wird als von der Verarbeitungskomplexität der medizinischen Interoperabilitätsmechanismen. So haben I/O-Operationen, die von der *JVM*-Implementierung bzw. durch das Betriebssystem verursacht werden, einen größeren Einfluss. Diese Analyse wird durch die Messungen mit dem leistungsschwachen *GBoard* gestärkt: Vergleicht man die *RTTs* der Bibliotheken mit Standardeinstellungen, weist die *OSCLib* die höchste Latenz auf. Da die *OSCLib* im Gegensatz zu den anderen Bibliotheken die Schemavalidierung für ein- und ausgehende Nachrichten durchführt, ist hier die Verarbeitungskomplexität am höchsten.

[5.12] Das Datenmodell ist Teil des *IEEE 11073-10207* Standards und als maschinenlesbare XML-Schemabeschreibung verfügbar. Somit können ein- und/oder ausgehende Nachrichten auf ihre syntaktische Korrektheit validiert werden.

Die Ergebnisse zeigen, dass die Schemavalidierung einen großen Einfluss auf die Performanz hat. Wird diese für die *SoftICE*-Bibliothek (komplett) aktiviert, steigt die Latenz um den Faktor $\approx 1{,}8$ bis 2,2. Für die *openSDC*-Bibliothek, bei der im Unterschied zur Vergleichsmessung nur zusätzlich die ausgehenden Nachrichten validiert werden, beträgt der Faktor $\approx 1{,}3$ bis 1,4. Ob eine Schemavalidierung für alle Nachrichten, nur für eingehende Nachrichten oder gar nicht ausgeführt werden muss, ist im konkreten Anwendungsfall in Abstimmung mit dem Risikomanagement des Herstellers zu entscheiden. In Zukunft definieren ggf. anwendungsfall- bzw. gerätespezifische Normen hierfür Minimalanforderungen.

5.5.4 Standardabweichung und Ausreißer

Zur Beurteilung der Performanz ist die ausschließliche Betrachtung der Median-*RTT* nicht ausreichend. Für viele Anwendungsfälle sind die Standardabweichung bzw. Varianz und die maximale Latenz wichtige Parameter, um zu beurteilen, ob die gestellten Anforderungen erfüllt sind oder nicht. Daher werden in den Tabellen 5.3, F.1, F.2 und F.3 diese Parameter mit angegeben. In den Tabellen im Anhang F werden zusätzlich die zehn höchsten der 40.000 Messwerte dargestellt. Dies erlaubt einen besseren Einblick in die Charakteristik der auftretenden Ausreißer als die Angabe des einen Maximalwertes. Zusätzlich geben die Boxplots in Abbildung 5.3 einen entsprechenden visuellen Überblick.

5.5.4.1 C++-Implementierung Erwartungsgemäß weist die *C++*-Bibliothek *OSCLib* verglichen mit den *Java*-basierten Implementierungen eine geringe Standardabweichung auf. Für die Plattformen *GBoard*, *RPi 1* und *RPi 2* liegt die maximal gemessene Latenz weniger als 3,5 ms über dem Median. Für die *RPi*-Plattformen ist zu erkennen, dass der prozentuale Abstand zwischen Median und Maximalwert mit steigender Leistungsfähigkeit zunimmt. Klare Ausreißer sind insbesondere beim *RPi 3* zu verzeichnen. Arbeiten, wie etwa von Konieczek et al. [A 206], zeigen, dass die Nutzung eines echtzeitfähigen Betriebssystems und die Zuweisung geeigneter Prioritäten solche Probleme lösen.

5.5.4.2 Java-Implementierungen Einer der grundlegenden Mechanismen von *Java* ist die *Garbage Collection (GC)*[5.13] [A 7, 8]. Die *GC* führt, verglichen mit der *C++*-Implementierung, zu wesentlich höheren Standardabweichungen, insbesondere auf den leistungsbeschränkteren Systemen *GBoard* und *RPi 1*.

Abbildung 5.4 verdeutlicht die Auswirkungen der *GC*. Der Übersichtlichkeit halber wird nur ein Ausschnitt der *RTT*-Messwerte gezeigt, die für die *openSDC*-Bibliothek auf einem *RPi 3* unter der Nutzung der *Oracle 7 JVM* aufgenommen wurden. Die Aussagen können verallgemeinert werden. Jeder grüne Datenpunkt entspricht einer *RTT*-Messung mit den Standardeinstellungen für die *GC*. Mithilfe von *JVM*-Parametern wie `-verbose:gc` können Informationen über die Zeitpunkte der durchgeführten *GCs* ausgegeben werden. Solche zusätzlichen Ausgaben zeigen, dass die *GC* vom *Service Consumer* und *Provider* in einer annähernd konstanten Periode durchgeführt wird. Die entstehenden Verzögerungen sind in Abbildung 5.4 zu erkennen. Die orange hervorgehobenen Werte repräsentieren Messungen, die von einer *GC* des *Service Consumers* beeinflusst sind, während die rot gekennzeichneten Messwerte von der *GC* des *Service Providers* beeinflusst sind. Die gekennzeichneten Messungen stellen verhältnismäßig „moderate“ Ausreißer dar.

Alle untersuchten *JVMs* nutzen in den Standardeinstellungen eine zweiphasige *GC*. Im Zuge der „normalen“ *GC* werden hauptsächlich Objekte mit kurzer Lebensdauer entsorgt. Die sogenannte *Full GC*[5.14] findet zusätzlich statt und versucht, Objekte im gesamten *Heap-Speicher* zu entsorgen. Eine Vorhersage des Zeitpunkts und der Dauer der *Full GC* ist mit Standard-*JVMs* kaum

[5.13] Siehe auch Glossareintrag zur *Garbage Collection*.

[5.14] Für die Begriffe *Minor GC*, *Major GC* und *Full GC* existieren keine in der Literatur und Implementierungen einheitlichen Definitionen [A 7]. Daher werden für diese Arbeit die Begrifflichkeiten genutzt, die den Ausgaben entsprechen, die mithilfe von *JVM*-Parametern wie `-verbose:gc` erzeugt werden.

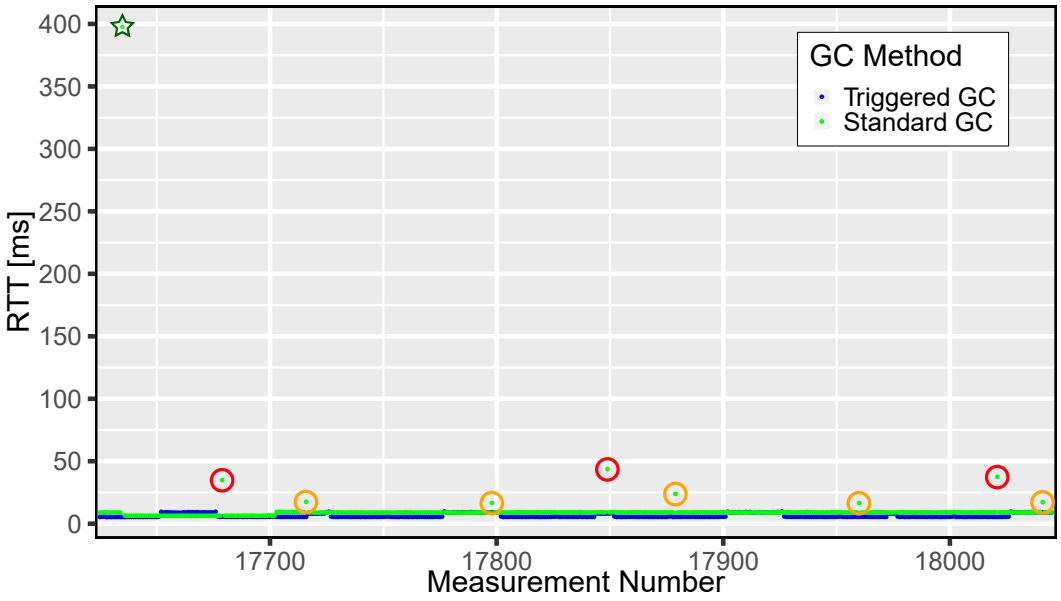

Abbildung 5.4: Latenzcharakteristik von Standard- und Implementierer-gesteuerter *Garbage Collection (GC)*. Dargestellt ist nur ein veranschaulichender Ausschnitt der Messwerte. Die in Abschnitt 5.5.5 beschriebenen Effekte haben hier keinen Einfluss. Hervorhebungen: Stern – *Full GC*; orange Kreise – *GC* des *Service Consumers*; rote Kreise – *GC* des *Service Providers*. (Abbildung entspricht *Fig. 4* aus [B 17] © 2017 IEEE.)

möglich [A 7]. Aufgrund des höheren Aufwands verursacht die *Full GC* eine größere Verzögerung und damit „massive“ Ausreißer. In Abbildung 5.4 ist ein solcher Messwert mit einem grauen Stern (oben links) gekennzeichnet. Im dargestellten Fall handelt es sich um den maximalen *RTT*-Wert bei der Messung mit der Nummer 17.635, mit einem Wert von 397,6 ms. Betrachtet man die zehn höchsten von je 40.000 Messwerten (siehe Anhang F), so ist zu erkennen, dass ein bis drei solcher „massiven“ Ausreißer pro Messreihe auftreten.

Zum Vergleich der präsentierten Ergebnisse wurden Messungen durchgeführt, bei denen aus der Anwendung heraus die *JVM* dazu aufgefordert wird, die *GC* durchzuführen. Der entsprechende Methodenaufruf (`System.gc()`) stellt aber lediglich eine Empfehlung an die *JVM* dar. Das Ergebnis ist in Abbildung 5.4 mit blauen Punkten dargestellt. Es ist zu erkennen, dass keine Ausreißer vorhanden sind. In dieser Serie von 40.000 Messungen lag der Medianwert der *RTT* bei 5,5 ms, bei einer Standardabweichung von +/- 1,8 ms und einem Maximalwert von 34,5 ms[5.15]. Diese Beobachtung lässt sich aber nicht verallgemeinern, da im Verlauf identischer Kontrollmessungen trotzdem *GC*-basierte Ausreißer zu verzeichnen waren. Dies kann im beschriebenen Empfehlungscharakter des *JVM*-Systemaufrufs begründet sein, sodass die *GCs* nicht wie gewünscht durchgeführt wurden. Ebenso wäre es möglich, dass die für die *GC* im Anwendungscode vorgesehene Bearbeitungszeit von einer Sekunde alle 25 Messdurchläufe nicht ausgereicht hat. Des Weiteren herrscht in der Entwicklercommunity weitestgehend Einigkeit darüber, dass das Triggern der *GC* aus dem Anwendungscode heraus, wie es hier experimentell vollzogen wurde, ernst zu nehmende Nachteile hat und nicht eingesetzt werden sollte. Ein solches Vorgehen außerhalb einer experimentellen Umgebung würde weiterhin voraussetzen, dass es möglich ist, vorhersagbar die Zeit für eine *Full GC*[5.16] einzuplanen. Davon kann, je nach Anwendungsfall, nicht ausgegangen werden.

Dennoch liefern die Ergebnisse dieses Teilexperiments einige Hinweise für Verbesserungsmöglichkeiten. So kann etwa das initiale Speichermanagement verbessert werden. Im vorgestellten Experiment wurden die Standardeinstellungen genutzt, sodass beispielsweise die initiale Speichergröße

5.15 Für die Interpretation dieses hohen Maximalwertes sei auf Abschnitt 5.5.5 verwiesen.

5.16 Bei Standard-*JVMs* lässt sich nur die *Full GC* triggern. Diese hat einen höheren Zeitbedarf als die „normale“ *GC*. Technisch gesehen wäre die *Full GC* nicht so häufig notwendig wie im Experiment mangels Alternativen vollzogen.

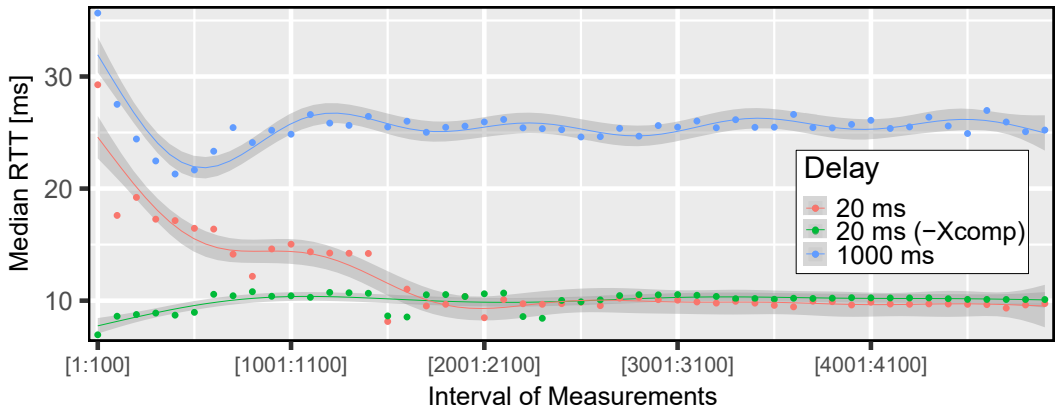

Abbildung 5.5: Latenzentwicklung im zeitlichen Verlauf. Jeder Datenpunkt entspricht dem Median von 100 Messwerten. Dargestellt sind die ersten 5.000 von 40.000 Messwerten. (Abbildung entspricht *Fig. 6* aus [B 17] © 2017 IEEE.)

angepasst werden kann. Um den Herausforderungen der *GC* umfassend zu begegnen, sollte die Implementierung hingegen unter der Nutzung der *Real-Time Specification for Java (RTSJ)* [A 207, 208] erfolgen und eine echtzeitfähige *JVM* zum Einsatz kommen. Die *RTSJ* bietet Möglichkeiten zur deterministischen Speicherverwaltung. Konieczek et al. [A 206] haben dieses Vorgehen beispielsweise erfolgreich für das *Internet of Things (IoT)*-Protokoll *Constrained Application Protocol (CoAP)* gezeigt. Die positiven Effekte der *RTSJ* für eingebettete Echtzeitsysteme werden beispielsweise von Corsaro und Schmidt [A 209] beschrieben. Unabhängig von der Implementierung der *Middleware* (Java, C++ etc.) ist der Einsatz eines echtzeitfähigen Betriebssystems für Anwendungsfälle mit entsprechenden Anforderungen an deterministisches Laufzeitverhalten notwendig.

Die Analysen dieses Abschnitts zeigen in Verbindung mit den Arbeiten von Konieczek et al. [A 206], dass eine *Java*-basierte *Middleware*-Implementierung mit deterministischem Laufzeitverhalten möglich ist. Nach bestem Wissen des Autors ist eine solche Implementierung derzeit nicht verfügbar.

5.5.5 Analyse der RTT-Werte im zeitlichen Verlauf

Im folgenden Abschnitt wird die Entwicklung der *RTT* über die Messdurchläufe betrachtet. Wie bereits beschrieben variieren die Messwerte für die *C++*-Implementierung *OSCLib* um den Median. Dies gilt für den gesamten zeitlichen Verlauf der Messungen. Bei einer detaillierten Betrachtung des Verhaltens der *Java*-Implementierungen *openSDC* und *SoftICE* ergibt sich ein deutlich anderes Bild. Die Beobachtungen sind in Abbildung 5.5 dargestellt: Es ist ein großer Einfluss des *Java Just-in-Time (JIT)*-Compilers[5.17] [A 17] auf das Laufzeitverhalten von Implementierungen nach dem Start der Anwendung („Aufwärmphase“) [A 210] zu erkennen.

Abbildung 5.5 stellt die Veränderung der *RTT*-Messwerte im zeitlichen Verlauf dar. Jeder Punkt im Diagramm repräsentiert den *RTT*-Median eines Messwerteintervalls mit 100 einzelnen Messwerten. So entspricht der erste Punkt dem Median der Messungen mit Nummern 1 bis 100, der zweite Punkt den Messungen mit den Nummern 101 bis 200 usw. Die Messreihe wird also in Zeitbereiche aufgeteilt, um das Verhalten besser darstellen zu können. Eine Betrachtung von Ausreißern wie in Kapitel 5.5.4 steht nicht im Fokus. Sie ist durch die Medianbildung nicht möglich. Um einen besseren Überblick zu erhalten, wurde eine sogenannte *Smooth Line* eingefügt.

Das Diagramm stellt den Ausschnitt der Intervalle [1;100] bis [4901;5000] dar. Dieser erste Teil der jeweils zugrundeliegenden 40.000 Messwerte ist ausreichend, um das Verhalten zu analysieren, da nach dieser dargestellten „Aufwärmphase“ keine für diesen Teilaspekt relevanten Veränderungen zu verzeichnen sind. Die in Abbildung 5.5 dargestellten Messwerte repräsentieren die *SoftICE*-

[5.17] Siehe auch Glossareintrag zur *Just-in-Time Compilation*.

Bibliothek, ausgeführt auf der *RPi 3*-Plattform, unter Nutzung der *Oracle 8 JVM*. Die Beobachtungen sind auf andere Systeme übertragbar, die einen *JIT*-Compiler nutzen [A 210]. Dies gilt für *Oracle JVMs* im Allgemeinen und für *OpenJDK* im Fall von *x86*-basierten Implementierungen.

Der rot dargestellte Teil in Abbildung 5.5 repräsentiert das beschriebene Standardsetup: 20 ms Wartezeit zwischen den einzelnen Messungen und Standardkonfiguration des *JIT*-Compilers. Im Bereich der ersten rund 2.500 Messungen ist ein klarer Abfall der *RTT*-Medianwerte zu erkennen. In diesem zeitlichen Bereich verringern sich die Werte um etwa den Faktor $\approx 2{,}5$. Da sich das Verhalten der *Garbage Collection (GC)* über die Laufzeit nicht verändert (siehe auch Abschnitt 5.5.4), kann dieses Verhalten auf den *JIT*-Compiler und dessen Optimierungen zurückgeführt werden.

JIT-Kompilierung bedeutet, dass die *JVM* nach dem Starten der Applikation damit beginnt, den *Java Bytecode* als *Interpreter* auszuführen. Zur Laufzeit werden aus Performanzsicht kritische Teile des Codes identifiziert und kompiliert. Da im Allgemeinen die Ausführung von kompiliertem Code schneller ist als das Interpretieren, verringert sich die Laufzeit des Systems. Diese Funktionsweise der *JIT*-Kompilierung erzeugt die sogenannte „Aufwärmphase“, die grundlegend ein bekanntes Verhalten von *Java*-Anwendungen darstellt [A 16]. Die beobachtete Dauer von rund 2.500 Kommunikationsvorgängen nach dem *Request-Response*-Verfahren zwischen zwei Vernetzungspartnern stellt hingegen eine unerwartet lang andauernde „Aufwärmphase“ dar. Dies könnte mit der Komplexität der *IEEE 11073 SDC-Middleware*-Implementierungen und entsprechend hohen Komplexität des Optimierungsprozesses begründet werden. So könnte etwa das Identifizieren aller Routinen der kritischen Pfade durch den *JIT*-Compiler eine erhebliche Zeit in Anspruch nehmen, um das Kompilieren auslösen zu können. Die Länge der „Aufwärmphase“ muss sowohl im experimentellen Kontext, aber vor allem für reale klinische Anwendungen berücksichtigt werden. Die Vorhersage der Länge der „Aufwärmphase“ ist derzeit eine offene Forschungsfrage [A 16].

Um die These zu prüfen, dass die sich stark verbessernde Performanz über die Laufzeit vom *JIT*-Compiler verursacht wird, wurde das Experiment mit gesetztem *JVM* Flag `-Xcomp` wiederholt. Diese Konfiguration zwingt die *JVM*, den gesamten Code während des Startens der Applikation zu kompilieren, sodass keine *JIT*-Kompilierung erfolgt. Das Ergebnis ist in Abbildung 5.5 mit grünen Punkten und entsprechender *Smooth Line* dargestellt. Abgesehen von `-Xcomp` *JVM Flag* unterscheiden sich die Systemkonfigurationen nicht im Vergleich zu den rot dargestellten Ergebnissen. Es wird ebenfalls eine Wartezeit von 20 ms zwischen zwei Messungen genutzt. Es ist zu erkennen, dass die grünen Messwerte eine annähernd konstante Charakteristik aufweisen. Weiterhin ist kein Abfall der gemessenen *RTT*-Werte über die Laufzeit zu erkennen. Die zuvor aufgestellte These zum Einfluss des *JIT*-Compilers wird durch diese Beobachtung gestärkt.

Im Allgemeinen wird nicht empfohlen, die *JIT*-Kompilierung mithilfe des `-Xcomp` *JVM Flags* zu deaktivieren. Zum einen verlängert sich die Startzeit der Applikation. Im konkreten Beispiel steigt sie von ≈ 6 s auf ≈ 13 s. Zum anderen kann keine weitere Optimierung auf der Basis von Informationen vorgenommen werden, die zur Laufzeit gesammelt werden. Dies kann im Langzeitverhalten beobachtet werden: Bezogen auf die gesamten 40.000 Messungen liegt der Median der *RTT*-Werte für die Messungen mit *JIT*-Compiler bei 9,8 ms und mit gesetztem `-Xcomp` *JVM Flag*, also ohne *JIT*-Compiler, bei 10,1 ms.

Des Weiteren wurden Messungen mit Standard *JIT*-Einstellungen und einer Wartezeit zwischen zwei Messungen von 1 s durchgeführt. Die Ergebnisse sind in Abbildung 5.5 mit blauen Punkten und zugehöriger *Smooth Line* dargestellt. Vergleicht man die Ergebnisse mit 1.000 ms (blau) und 20 ms (rot) Wartezeit zwischen den einzelnen Messungen, ist zu erkennen, dass die Performanz im Fall von 1.000 ms deutlich geringer ist. Bezogen auf die gesamten 40.000 Messungen liegt der Median der *RTT*-Werte etwa um den Faktor 2,5 auseinander: 25,0 ms (für eine Wartezeit von 1.000 ms) bzw. 9,8 ms (für eine Wartezeit von 20 ms). Dies zeigt, dass die Optimierung im Zuge der *JIT*-Kompilierung im Fall einer hohen Kommunikationsintensität deutlich bessere Ergebnisse erzielt

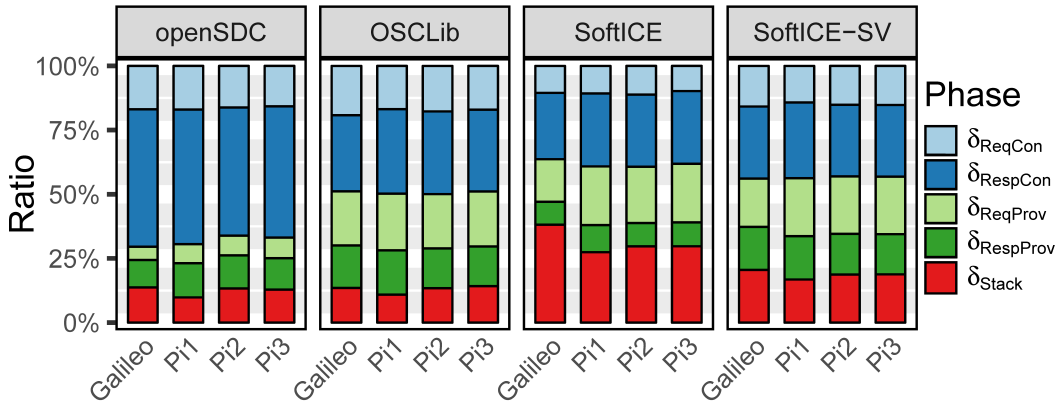

Abbildung 5.6: Anteilige Zusammensetzung der *RTT*. Hellblau – Anfrageerzeugung beim *Service Consumer* (δ_{ReqCon}); dunkelblau – Antwortverarbeitung beim *Service Consumer* ($\delta_{RespCon}$); hellgrün – Anfrageverarbeitung beim *Service Provider* ($\delta_{ReqProv}$); dunkelgrün – Antworterzeugung beim *Service Provider* ($\delta_{RespProv}$); rot – Verarbeitung und Transport unterhalb der *IEEE 11073 SDC*-spezifischen *Middleware* (δ_{Stack}). (Abbildung entspricht Fig. 5 aus [B 17] © 2017 IEEE.)

als bei einer geringen Kommunikationsintensität. Es kann also geschlussfolgert werden, dass *JIT*-Kompilierung unerwartet stark von der Menge und Frequenz der ausgetauschten Daten abhängt.

Die in diesem Abschnitt präsentierten Erkenntnisse legen nahe, dass die Eigenschaften der *JIT*-Kompilierung für den konkreten Anwendungsfall zu untersuchen sind, da sich unter anderem Abhängigkeiten vom Kommunikationsmuster gezeigt haben. Auf dieser Basis sollten Entscheidungen über geeignete Mechanismen, wie etwas das Vorkompilieren, getroffen werden.

5.5.6 Analyse der Zusammensetzung der RTT

Wie in Abschnitt 5.4.2 beschrieben setzt sich die *RTT* aus verschiedenen Sub-Latenzen zusammen. Der Anteil der einzelnen Kommunikationsschritte wird in diesem Abschnitt analysiert. Abbildung 5.6 zeigt die prozentuale Verteilung der Verzögerungen, die in den verschiedenen Phasen entstehen:

- Erzeugung der Anfrage durch die *Middleware* des *Service Consumers* (δ_{ReqCon})
- Verarbeitung der Anfrage durch die *Middleware* des *Service Providers* ($\delta_{ReqProv}$)
- Erzeugung der Antwort durch die *Middleware* des *Service Providers* ($\delta_{RespProv}$)
- Verarbeitung der Antwort durch die *Middleware* des *Service Consumers* ($\delta_{RespCon}$)
- Verarbeitung und Transport unterhalb der *Middleware* von *Service Consumer* und *Provider* (δ_{Stack})

Die verschiedenfarbigen Abschnitte der Balken in Abbildung 5.6 stellen die einzelnen Teilverzögerungen dar. Sie sind farblich und örtlich nach *Service Consumer* (hell- und dunkelblau) und *Service Provider* (hell- und dunkelgrün) gruppiert. Damit entspricht die Schichtung der Balkenabschnitte nicht der zeitlichen Abfolge in der Kommunikation. Die dargestellten und diskutierten Werte beziehen sich jeweils auf die Nutzung der *JVM* mit der besten Performanz. Wie in Abschnitt 5.5.3 diskutiert wird, hat die Schemavalidierung einen großen Einfluss auf die Kommunikationslatenz. Daher wurde ein weiterer Abschnitt in Abbildung 5.6 aufgenommen, der die prozentuale Verteilung der einzelnen Latenzen für die *SoftICE*-Bibliothek mit eingeschalteter Schemavalidierung zeigt (rechte Balkengruppe, „SoftICE-SV“).

5.5.6.1 Untersuchung des *IEEE 11073 SDC*-spezifischen Anteils δ_{Stack} beinhaltet die *HTTP*- und *TCP/IP-Stacks* beider Vernetzungsteilnehmer sowie die *Ethernet*-Übertragungszeit und das

Switch-Delay. In den beschriebenen Experimenten wurde ein *Fast Ethernet*-Switch im *Store-and-Forward*-Modus genutzt. Die Obergrenze für das *Switch-Delay* wurde in anderen Arbeiten am Institut für Angewandte Mikroelektronik und Datentechnik von Schweißguth et al. [A 211] mit 30 µs bestimmt. Bezieht man einen Unsicherheitsfaktor von zehn ein, so kann eine Obergrenze für die gesamte *Ethernet*-Übertragungszeit von 300 µs angenommen werden. Verglichen mit der gesamten *RTT* (siehe beispielsweise Tabelle 5.3) stellt dies eine geringe, quasi konstante Verzögerung im Bereich von 0,3 % bis 3,4 % dar. Die anderen Bestandteile von δ_{Stack} sind stark plattformabhängig.

Betrachtet man das Verhältnis zwischen den höheren, medizingerätespezifischen Teilen der *Middleware* (in Abbildung 5.6 blau und grün dargestellt) und den darunterliegenden Teilen (δ_{Stack}, rot) ist klar zu erkennen, dass der *IEEE 11073 SDC*-spezifische Anteil die Gesamtlatenz dominiert. Der prozentuale Anteil von δ_{Stack} liegt für die Bibliotheken *openSDC* und *OSCLib* bei unter 15 %. Für die *SoftICE*-Bibliothek ist der Anteil größer. Kommt die Schemavalidierung zum Einsatz, sinkt der Anteil von δ_{Stack} bei der *SoftICE*-Bibliothek auf rund 20 %, da der Verarbeitungsaufwand in der *IEEE 11073 SDC*-spezifischen Schicht stark ansteigt (siehe auch Abschnitt 5.5.3).

Aus den beschriebenen Beobachtungen zur prozentualen Verteilung der Latenzanteile kann als Handlungsempfehlung abgeleitet werden, dass der Fokus zukünftiger Optimierungen im oberen, *IEEE 11073 SDC*-spezifischen Teil der *Middleware*-Implementierungen liegen sollte. Da hier die größten Verzögerungen entstehen, ist das Optimierungspotential in diesem Bereich als am größten einzuschätzen. Aufgrund der Unterschiede zwischen *SoftICE* und den beiden anderen Bibliotheken, bezogen auf den Anteil von δ_{Stack}, sollte hier die genutzte *HTTP*-Implementierung untersucht werden.

5.5.6.2 Verteilung zwischen Service Consumer und Service Provider Die Latenz, die von *Service Provider* und *Service Consumer* verursacht wird, ist für die Bibliotheken *OSCLib* und *SoftICE* annähernd gleich. Im Fall der Bibliothek *openSDC* können klare Unterschiede zwischen den Kommunikationspartnern ausgemacht werden: Die vom *Service Provider* induzierte Verzögerung ist um etwa den Faktor 3 bis 4,5 kleiner als die Verzögerung auf der Seite des *Service Consumers*. In realen klinischen Umgebungen werden *Service Provider* oft auf ressourcenbeschränkteren Plattformen implementiert sein. Für *Service Consumer*, die für die Verarbeitung und Anzeige der Informationen höhere Anforderungen haben als Sensoren und einfache Aktoren, ist von Systemen mit mehr Ressourcen auszugehen. Daher wurde bei der Entwicklung von *IEEE 11073 SDC* Wert darauf gelegt, die *Service Provider*, unter Einhaltung der medizinischen Semantik- und Sicherheitsanforderungen, so einfach wie möglich zu halten. Einerseits legen die Ergebnisse nahe, dass während der Implementierung von *openSDC* der Fokus auf der Optimierung der *Service Provider* lag, andererseits zeigt sich ein entsprechendes Potential beim *Service Consumer*.

5.6 Diskussion

In diesem Kapitel wurden Latenzmessungen für *IEEE 11073 SDC-Middleware*-Implementierungen präsentiert und analysiert. Hierfür wurden die drei zum Zeitpunkt der Evaluation offen verfügbaren Bibliotheken *openSDC*, *SoftICE* und *OSCLib* auf vier unterschiedlichen Hardware-Plattformen (*GBoard*, *RPi 1, 2* und *3*) unter der Berücksichtigung verschiedener Parameter untersucht. Die Ergebnisse basieren auf über 1,5 Millionen Messungen mit jeweils sieben Messpunkten.

Nach einer kurzen Einordnung der Messungen und Auswertungen werden Schlussfolgerungen aus den Ergebnissen gezogen und es wird auf potentielle zukünftige Forschungsfragen eingegangen.

5.6.1 Auswertungsmethodik und verwandte Arbeiten

Da *IEEE 11073 SDC* eine neue Technologie darstellt, sind bisher keine ähnlichen Arbeiten zur systematischen Performanz-Analyse von entsprechenden *Middleware*-Implementierungen bekannt.

Dennoch unterstützen viel beachtete Arbeiten im Bereich der Performanz-Untersuchungen, beispielsweise von Georges et al. [A 210] oder von Kalibera und Jones [A 13], die in dieser Arbeit genutzte Methodik. So wird ebenfalls auf die Aspekte verschiedener *JVMs*, das Verhalten der *GC* [A 7] oder den Einfluss der *JIT*-Kompilierung [A 16] eingegangen und deren Auswirkungen differenziert analysiert.

Zusätzlich beschäftigen sich auch praxisnahe Veröffentlichungen mit der Abhängigkeit der Performanz von *JVMs* und deren Parametern. So bezieht etwa Connors [A 212] im *Oracle*-Blog unter anderem auch *ARM*-Architekturen ein, beschränkt sich aber auf *Java 7 JVMs*.

5.6.2 Einordnung der Ergebnisse und Schlussfolgerungen

Die einzelnen Ergebnisse der Performanzevaluationen stellen eine Momentaufnahme zum Zeitpunkt der Evaluation dar. Sie sind zudem beschränkt auf die untersuchten Bibliotheken und Plattformen. Die Analysen haben gezeigt, dass die Performanz von diversen Faktoren abhängt, sodass eine direkte Übertragbarkeit der Ergebnisse nicht ohne Weiteres möglich ist. In dieser Arbeit wurden jedoch eine Reihe von verallgemeinerbaren Herausforderungen herausgearbeitet. An den entsprechenden Stellen wird darauf hingewiesen, dass anwendungsfallspezifische Analysen notwendig sind, um diesen Herausforderungen konkret zu begegnen.

Für die präsentierten Untersuchungen wurde ein generisches Kommunikationsmuster nach dem einfachen *Request-Response-Pattern* zwischen zwei Vernetzungspartnern genutzt. Einleitend wurde gezeigt, dass auf dieser Basis grundlegende Analysen sinnvoll möglich sind. Mit realistischen Kommunikationspatterns würde die spezifische Aussagekraft dennoch weiter steigen.

Trotz dieser Einschränkungen lassen sich aus den Messergebnissen und angestellten Analysen dieses Kapitels zusammenfassend verschiedene Schlussfolgerungen ableiten:

Schlussfolgerung 1 Die Ergebnisse aller drei untersuchten Kommunikationsbibliotheken bestätigen die generelle Praxistauglichkeit der neuen *IEEE 11073 SDC*-Normenfamilie zur Erreichung von herstellerübergreifender Medizingeräteinteroperabilität. Die gestellten Performanzanforderungen einer Kommunikationslatenz von unter 100 ms bzw. 200 ms werden von den Bibliotheken voll bzw. teilweise erfüllt[5.18].

Es sei angemerkt, dass es sich um prototypische und wissenschaftlich geprägte Implementierungen handelt, die bisher nicht hinsichtlich ihrer Performanz optimiert wurden. Substantielle Verbesserungspotentiale einzelner Implementierungen konnten aufgezeigt werden. Die vorhandenen Performanz-Unterschiede geben aber keinen Anlass, das *IEEE 11073 SDC*-Konzept anzuzweifeln.

Schlussfolgerung 2 Die Validierung der ausgetauschten Nachrichten gegen die komplexen XML-Schemata der *IEEE 11073-10207* Norm hat einen starken Einfluss auf die Kommunikationslatenz.

Schlussfolgerung 3 Das größte Optimierungspotential für die Performanz liegt im Bereich der *IEEE 11073 SDC*-spezifischen Implementierung, da dieser Teil des Kommunikationsstacks den größten Anteil an der Kommunikationslatenz hat.

Schlussfolgerung 4 Im Fall von *Java*-basierten *Middleware*-Implementierungen hängt die Performanz stark von der Kombination aus der Hardwareplattform, der genutzten *JVM*, den *JVM*-Einstellungen und den Kommunikationsmustern ab. Daher ist für die Anforderungen eines konkreten Medizingerätes und der konkreten Anwendungsfälle eine geeignete Systemkonfiguration zu finden. Allgemeingültige Vorhersagen und Empfehlungen sind kaum möglich.

[5.18] Die Aussage ist bezogen auf „weiche“ Echtzeiteigenschaften.

Schlussfolgerung 5 Die *JIT*-Kompilierung und damit die Kommunikationslatenz der *Java*-Bibliotheken wird stark von der Charakteristik und der Intensität der Kommunikation beeinflusst. Dies kann sich sowohl auf die sogenannte „Aufwärmphase“, für die sich eine unerwartet lange Dauer gezeigt hat, als auch auf das gesamte Kommunikationsverhalten auswirken.

Schlussfolgerung 6 Die aktuell untersuchten Systeme erfüllen keine (harten) Echtzeitanforderungen. Ein echtzeitfähiges Betriebssystem stellt eine Grundvoraussetzung dar. Im Fall von *Java*-Implementierungen sollte die *RTSJ* in Verbindung mit einer echtzeitfähigen *JVM* und einem effizienten Speicher- und *GC*-Management genutzt werden.

5.6.3 Zukünftige Entwicklungs- und Forschungsfragen

In zukünftigen Arbeiten sollten Performanzuntersuchungen mit weiterentwickelten Implementierungen, insbesondere mit Blick auf (harte) Echtzeiteigenschaften, vorgenommen werden. Soweit dies möglich ist, sollten produktreife, unter den Regeln der Qualitätssicherung implementierte Softwarebibliotheken einbezogen werden.

Die beschriebenen Untersuchungen bilden lediglich generische und vereinfachte Kommunikationsmuster ab. Aufgrund der beschriebenen großen Abhängigkeit vom konkreten Anwendungsfall bieten echte Kommunikationsmuster, die aus realen klinischen Anwendungsfällen abgeleitet sind, einen großen Mehrwert für Forschung und Industrie. Auf der Basis solcher Kommunikationsmuster können tiefere, spezifische und dennoch vergleichbare Analysen in Zukunft durchgeführt werden.

Die Bibliotheken selbst sollten unter Performanzgesichtspunkten weiterentwickelt werden. Die Erkenntnisse der präsentierten Untersuchungen können zur Verbesserung genutzt werden. Das hohe Potential von Optimierungen im *IEEE 11073 SDC*-spezifischen Teil der *Middleware* für die Gesamtperformanz wurde in Abschnitt 5.5.6 dargelegt. In Abhängigkeit der zukünftigen Anwendungsbereiche sollte die Entwicklung unter (harten) Echtzeitgesichtspunkten und der Nutzung entsprechender Technologien erfolgen.

Die *IEEE 11073 SDC*-Normenfamilie spezifiziert derzeit die Datenübertragung mittels *MDPWS* (*IEEE 11073-20702*). Aus Forschungssicht wäre ein Performanzvergleich der aktuellen Implementierungen mit Implementierungen von Interesse, die andere Standards zur Datenübertragung nutzen. So wurde beispielsweise bereits die generelle Machbarkeit der Nutzung des *Constrained Application Protocol (CoAP)* und des *Data Distribution Service (DDS)* gezeigt. Die Entwicklung mit *CoAP* erfolgte durch eine studentische Arbeit unter der Betreuung des Autors [E 3], während die *DDS*-Technologie von Projektpartnern der Universität zu Lübeck [A 158] bearbeitet wurde.

6 Das OR.NET-Session-Konzept

Die Patientensicherheit ist entscheidend für *PoC*-Medizingeräte. Viele Aspekte des Risikomanagements in Bezug auf die Patientensicherheit konnten bisher individuell und autark vom Hersteller betrachtet werden. Dies gilt auch weiterhin für therapeutische und diagnostische Funktionen, die ohne eine Interaktion mit anderen Medizingeräten auskommen. Vernetzungsabhängige Gesichtspunkte bedürfen hingegen neuer Mechanismen. In diesem Kapitel wird auf einen grundlegenden Aspekt in dynamisch und herstellerunabhängig vernetzten Medizingeräteensembles eingegangen: Es ist sicherzustellen, dass die „richtigen“ Geräte Daten untereinander austauschen und dass Fernsteuerungsoperationen an den „richtigen“ Geräten ausgelöst werden.

Gegenwärtig werden typischerweise herstellerspezifische und monolithische Medizingeräteensembles bereits vor der Laufzeit, ggf. schon zur Entwicklungszeit, in Bezug auf die interagierenden Geräte und genutzten Funktionalitäten strikt definiert. Aufgrund der gewollten dynamischen Eigenschaft einer Vernetzung auf Basis einer *SOMDA* ist dies in Zukunft nicht mehr möglich. Die Geräteverbünde ändern sich entsprechend den Anforderungen von Eingriff zu Eingriff, von Patientin zu Patientin und ggf. von Ärztin zu Ärztin. Geräte können während eines Eingriffs hinzukommen oder entfernt werden. Beispielsweise werden einige Geräte, wie etwa mobile Röntgengeräte (sogenannte C-Bögen), zwischen verschiedenen OP-Sälen während der laufenden Behandlung ausgetauscht. Somit besteht die Notwendigkeit, dass dynamisch zur Laufzeit Geräteverbünde definiert werden können, die zum aktuellen Zeitpunkt für die medizinische Versorgung eines bestimmten Patienten genutzt werden. Diese patienten-, behandlungs-/diagnose- und ggf. ortsbezogenen Geräteensembles wurden im *OR.NET*-Projekt als *Session* bezeichnet. Die *Session*-Mitglieder müssen sich nicht zwingend alle an einem Ort befinden. Die zentrale Arbeitsstation einer Intensivstation (ITS) kann beispielsweise zusammen mit den Geräten am Patientenbett zu einer *Session* gehören.

In diesem Kapitel wird erklärt, wie eine solche *Session* erzeugt und verwaltet werden kann, wie mit diesem Mechanismus sichergestellt werden kann, dass ausgetauschte Daten zur richtigen *Session* gehören und dass Fernsteuerungskommandos nur innerhalb der *Session* möglich sind.

Das *Session*-Konzept wurde innerhalb des *OR.NET*-Projekts im Zuge eines Arbeitspakets unter Mitwirkung einer Reihe von Experten entwickelt. Die Veröffentlichung erfolgte durch den Autor der vorliegenden Arbeit zusammen mit mehreren Co-Autoren, in [B 7] im Kapitel „Safety for Dynamic Medical Device Ensembles: The OR.NET Session Concept“. Diese Veröffentlichung bildet die Grundlage dieses Kapitels.

6.1 Erzeugung und Verwaltung von Sessions

Als Vorbedingung für die Erläuterung des Konzepts wird angenommen, dass zwischen den beteiligten Medizingeräten eine *Vertrauensbeziehung* besteht und somit eine Vernetzung zwischen den Medizingeräten aus Sicht der *Security* möglich ist.

Eine *Session* wird technisch gesehen durch einen *Ensemble Context* realisiert, wie er im *IEEE 11073-10207* Standard definiert ist (siehe Kapitel 4.4.1.1). Die verantwortliche Komponente innerhalb der Vernetzungsumgebung für die Verwaltung einer *Session* ist der sogenannte *Session Manager*. In Abbildung 2.2, der Übersichtsgrafik zum *OR.NET*-Projekt, ist der *Session Manager* in Rot dargestellt. Ob dieser als eigenständige physikalische Instanz realisiert wird oder als Teilkomponente eines größeren Systems, ist dabei für das Konzept unerheblich. Innerhalb des *OR.NET*-Projekts wurde der zweite Ansatz vom Projektpartner MEDNOVO Medical Software Solutions GmbH (inzwischen KARL STORZ SE & Co. KG) [A 213] implementiert.

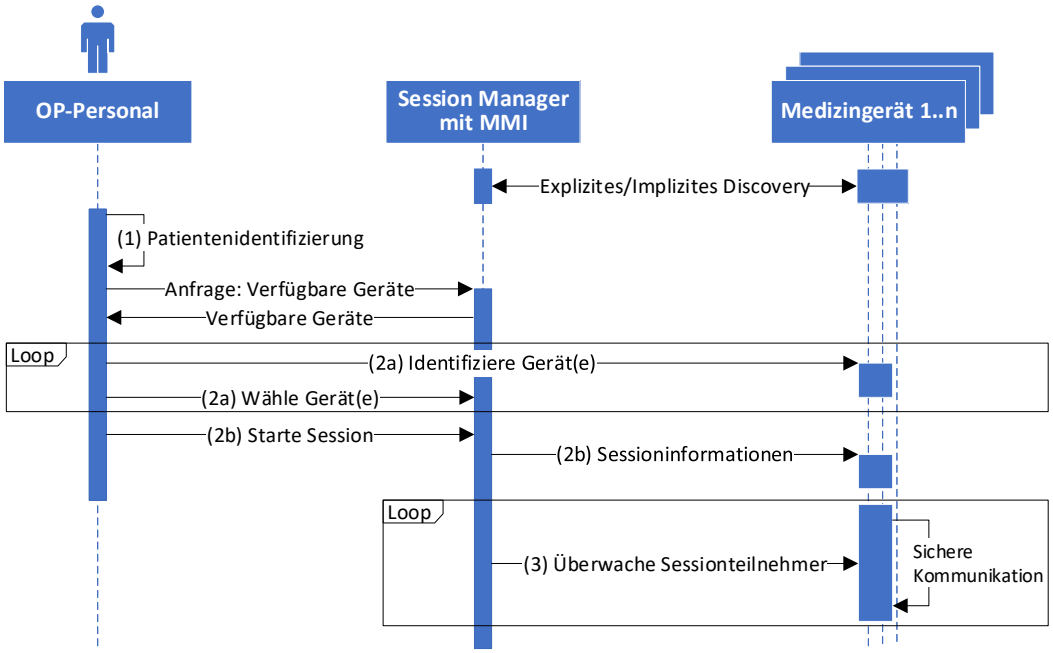

Abbildung 6.1: Sequenzdiagramm der Erzeugung und Verwaltung einer *Session* mittels eines *Session Managers* mit *Mensch-Maschine-Interface (MMI)* zur Geräteauswahl. Die Schritte (4) und (5) des Lebenszyklus, das Hinzufügen/Entfernen von Teilnehmern und das Schließen der *Session*, sind der Übersichtlichkeit halber nicht dargestellt. (Abbildung nach *Figure 5* aus [B 7].)

6.1.1 Der Session-Lebenszyklus

Der Lebenszyklus einer *Session*, der in Abbildung 6.1 veranschaulicht wird, kann wie folgt beschrieben werden:

1. Identifizierung des Patienten und Auswahl des entsprechenden medizinischen, therapeutischen/diagnostischen Vorgangs
2. Erstellung der *Session*
   (a) Zusammenstellung des *Session*-Ensembles: Identifizierung und Auswahl der konstituierenden Medizingeräte
   (b) Start der *Session*: Verteilung der Kontextinformationen an die *Session*-Teilnehmer
3. Überwachung des *Session*-Ensembles
4. Hinzufügen von Teilnehmern zur bzw. Entfernung aus der *Session* (falls notwendig)
5. Schließen der *Session*

Der *Session Manager* bekommt die Patienteninformationen (*Patient Demographic Data*) und Informationen zum medizinischen Eingriff mit Hilfe des *Patienteninformationen-Providers/Distributors* (siehe auch Kapitel 2.3.2.2 und grüne Komponenten in Abbildung 2.2). Für die Interaktion mit den Benutzenden stellt der *Session Manager* ein *MMI* zur Verfügung. Dies kann beispielsweise in Form einer grafischen Benutzeroberfläche, *Graphical User Interface (GUI)*, auf einem PC oder Tablet-Computer, als Barcode-Leser, *Radio-Frequency Identification (RFID)*-Leser etc., oder auch als Kombination mehrerer Formen, realisiert werden. Mit Hilfe dieses *MMIs* werden die folgend beschriebenen Interaktionen mit den menschlichen Akteuren vollzogen.

Nachdem der Patient eindeutig identifiziert wurde, werden im ersten Schritt die entsprechenden Patienten- und Auftragsdaten[6.1] vom OP-Personal ausgewählt. Im kommenden Schritt wird das Geräteensemble zusammengestellt. Hierzu wählt das OP-Personal die benötigten Geräte aus der Menge der zur Verfügung stehenden Geräte aus. Es ist sicherzustellen, dass es sich genau um die Geräte handelt, die intendiert sind. Beim Scannen aller benötigten Geräte mit einem Barcode-Leser ist dies implizit sichergestellt. Realisiert das *MMI* eine entfernte Geräteauswahl, müssen die Geräte geeignete Mechanismen zur eindeutigen Identifizierung bereitstellen. Im *OR.NET*-Projekt wurden hierfür beispielhaft Lösungen aufgezeigt. So können *Activate Operations* angeboten werden, mit denen das Gerät in einen Zustand versetzt wird, der eine Identifizierung erlaubt. Je nach Fähigkeiten des Gerätes kann die Realisierung variieren: Textuelle oder symbolische Nachrichten auf einem Bildschirm, Leuchten einer bestimmten LED, Abspielen eines bestimmten Audiosignals etc. Es bieten sich weiterhin zusätzliche Hilfsfunktionalitäten zum Zusammenstellen des Geräteensembles an, wie etwa das Filtern nach dem Ortskontext oder definierten (Vor-)Gruppierungen, die über andere Instanzen des *Ensemble Contexts* realisiert werden.

Die eigentliche *Session*-Erzeugung erfolgt in Schritt (2b). Der *Session Manager* transferiert den *Session Key* und ggf. weitere kontextuelle Daten (Patienten- und Auftragsinformationen) an die ausgewählten Medizingeräte. Die technische Realisierung des *Session Keys* erfolgt über einen Mechanismus des *IEEE 11073-10207* Standards, dem `pm:Identification`-Element einer Instanz des *Ensemble Contexts* (`pm:EnsembleContextState`). Diese spezifische Instanz des *Ensemble Contexts* definiert die *Session*. Da es potentiell mehrere Gruppierungen von Medizingeräten geben kann, die über verschiedene Instanzen des *Ensemble Contexts* realisiert werden, wird die Semantik über das *Type*-Element der *Ensemble Context*-Beschreibung spezifiziert. Für den *Session Context* ist ein entsprechender Eintrag in der Nomenklatur erforderlich.

Damit der *Session Manager* den *Session Key* an die Medizingeräte übertragen kann, müssen diese den *Ensemble Context* und eine entsprechende *Set Operation* zur Fernkonfiguration bereitstellen. Dies stellt die Minimalanforderung an ein Medizingerät dar, um Teil einer *OR.NET-Session* zu werden. Nur Geräte, die diese Voraussetzung erfüllen, sollten im Schritt (2a) angezeigt werden. Während die *Session* patienten-, auftrags- und ggf. ortsbezogen ist, enthält der zugehörige *Ensemble Context* keine derartigen Informationen. Die Bereitstellung geht über das Konzept der *Session* hinaus. In der Praxis ist aber davon auszugehen, dass Erweiterungen des *Session Managers* entsprechende Funktionalitäten bereitstellen. Benötigt der Vernetzungsteilnehmer derartige Informationen, so können *Patient* und *Workflow Context* genutzt werden. Über die zugehörigen *Patient Demographics Core Data* bzw. *Order Detail*-Objekte können patienten- bzw. auftragsbezogene Daten transportiert werden. Die Verteilung kann beispielsweise analog zum *Session Key* erfolgen.

Der Ausfall der Vernetzungsfunktionalität von Teilnehmern der *Session* kann erhebliche Folgen für den Funktionsumfang wie auch für die Bedienung des Gesamtsystems und damit auch für die Patientensicherheit haben. Daher müssen Geräte, die aufgrund von Verbindungsabbrüchen, Software- oder Hardwarefehlern etc. nicht mehr erreichbar sind, umgehend erkannt werden. Entsprechend überwacht der *Session Manager* in Schritt (3) das Geräteensemble und stellt Informationen und ggf. Alarme bereit, wenn sich Geräte unerwünscht verhalten. Eine solche Überwachungsfunktionalität kann auch von einer unabhängigen Komponente realisiert werden, die nicht der *Session Manager* ist. Unabhängig von der Umsetzung ist die Durchführung einer solchen Überwachung und Zusammenführung der Informationen an einem für die Akteure im OP-Saal geeigneten Ort sinnvoll. Zusätzlich hat jedes Medizingerät ein eigenständiges Risikomanagement und Gegenmaßnahmen für eigene Verbindungsabbrüche bzw. Verbindungsabbrüche von Vernetzungspartnern.

6.1 Ein Auftrag im Sinne eines medizinischen Arbeitsablaufs fasst verschiedene medizinische Informationen für eine gewünschte Diagnostik oder Behandlung eines Patienten zusammen. Neben Informationen zum Patienten und der Diagnostik/Behandlung können etwa Informationen zum Durchführenden oder Risikofaktoren enthalten sein.

Aufgrund der Dynamik von Medizingeräteensembles können Teilnehmer die *Session* gewollt verlassen oder dieser während des laufenden Betriebs beitreten. Entsprechende Funktionalitäten für die Umorganisation werden daher vom *MMI* des *Session Managers* bereitgestellt. Am Ende des medizinischen Eingriffs wird die *Session* geschlossen. Entsprechend den beschriebenen Schritten (2) bis (5) durchläuft der Assoziierungszustand des *Ensemble Contexts*, der die *Session* repräsentiert, die verschiedenen Zustände von nicht bzw. prä-assoziiert bis Schritt (2a), über assoziiert ab Schritt (2b) bis dis-assoziiert ab Schritt (4) bzw. (5). Ob abgelaufene *Session*-Informationen vom Gerät persistiert werden oder nicht, hängt vom Hersteller und den Fähigkeiten des Gerätes ab.

6.1.2 Service Consumer als Session-Teilnehmer

Der beschriebene Mechanismus der Übertragung von Informationen vom *Session Manager* auf die *Session Teilnehmer* setzt voraus, dass alle *Session Teilnehmer* die Rolle eines *Service Providers* einnehmen. Handelt es sich um einen reinen *Service Consumer* ist dies nicht möglich, da ein *Service Consumer* keine Services anbieten kann, die eine vom *Service Manager* ausgehende Datenübertragung ermöglichen. Die Erfahrung aus Projekten wie *OR.NET* und *MoVE* haben jedoch gezeigt, dass die überwiegende Mehrzahl von Geräten mit *Service Consumer*-Funktionalität ebenfalls die Rolle des *Service Providers* implementieren. Dennoch ist es nicht sinnvoll, reine *Service Consumer* dazu zu zwingen, *Service Provider*-Funktionalitäten zu implementieren, nur um eine Teilnahme an einer *Session* zu ermöglichen. Daher muss das Konzept entsprechend erweitert werden.

Um reinen *Service Consumern* die Teilnahme an einer *Session* zu ermöglichen, muss der *Session Manager* die Rolle des *Service Providers* implementieren. In dieser zusätzlichen Funktion stellt er die aktuellen *Session*-bezogenen Informationen, vor allem den *Session Key*, mittels einer Instanz des *Ensemble Contexts* zur Verfügung. Analog können weitere Informationen wie Patienten- und Auftragsinformationen mittels des *Patient* bzw. *Workflow Contexts* bereitgestellt werden. Ein an diesen Informationen interessierter *Service Provider* nutzt die entsprechenden Services des *Session Managers*. Für die praktische Umsetzung sind somit eine klare semantische Selbstbeschreibung des *Session Managers* und damit die Definition eines dedizierten Nomenklatur-Codes notwendig. So kann sichergestellt werden, dass der *Service Manager* von interessierten *Service Consumern* im Netzwerk gefunden wird. Entsprechende normative Rollenspezifikationen und damit verbundene Nomenklatur-Einträge stehen noch aus.

Es ist weiterhin sicherzustellen, dass nur die intendierten *Service Consumer* die *Session Informationen* erhalten. Daher wird der *Session Manager* nur Anfragen von Geräten antworten, die der *Session* zugeordnet werden sollen. Die praktische Umsetzung hängt dabei stark vom Realisierungskonzept des *Session Managers* ab. Werden die *Session*-Teilnehmer etwa mittels eines Barcode-Scanners ausgewählt, so erfolgt dies entsprechend auch für die *Service Consumer*. Im Barcode können identifizierende Merkmale codiert werden, z. B. der *Common Name* des *X.509.v3*-Zertifikatsinhabers. Diese Information kann der *Session Manager* nutzen, um *Session*-bezogene Anfragen zuzulassen bzw. abzulehnen. Setzt das *MMI* des *Session Managers* eine entfernte Geräteauswahl zur *Session*-Teilnehmerauswahl um, so kann etwa auf Vorwissen zurückgegriffen werden, da ein *dynamisches Discovery* von *Service Consumern* nicht möglich ist. Ebenso ist die Möglichkeit einer fernausgelösten Identifizierungsaktion mittels *IEEE 11073 SDC* nicht gegeben.

Abonniert ein Gerät, das ausschließlich über *Service Consumer*-Funktionalitäten verfügt, die Updates der entsprechenden Kontextzustände des *Session Managers*, kann dieser auch die Verfügbarkeit überwachen. Eine entsprechende Pflicht zum Abonnement könnte in einem zu spezifizierenden Standard oder einer Sammlung von *Best Practice*-Verfahren festgehalten werden und zur Laufzeit vom *Session Manager* überprüft werden, bevor dieser *Session*-bezogene Anfragen positiv bearbeitet.

6.2 Session-Safety-Mechanismen

Das Ziel des *Session Contexts* ist es, dass die richtigen Geräte miteinander kommunizieren. Die Kommunikation hat zwei Aspekte:

1. *Service Consumer* sollen Daten von den richtigen Geräten empfangen, auswerten und ggf. anzeigen oder weiterverarbeiten.

2. Fernsteuerungsoperationen dürfen nur an den intendierten *Service Providern* vorgenommen werden.

Beide Bereiche können einen hohen Einfluss auf die Patientensicherheit haben. Wenn beispielsweise Informationen aus einem anderen OP-Saal angezeigt werden, treffen die Ärzte potentiell falsche Entscheidungen. Wenn ein Bedienfeld in einem OP-Saal Geräte in einem anderen ungewollt fernsteuert, kann dies ernsthafte Konsequenzen für die Patientinnen in beiden OP-Sälen haben. Für beide Herausforderungsbereiche können Mechanismen auf der Basis des *OR.NET-Session*-Konzepts umgesetzt werden.

Ein Datenaustauch wird vom *Service Consumer* initiiert, durch eine Anfrage nach dem *Request-Response-Pattern* oder durch ein Abonnement, um Informationen nach dem *Publish-Subscribe-Pattern* zu erhalten. Um sicherzustellen, dass dieser Prozess nur mit anderen Teilnehmern der *Session* abläuft, ruft der *Service Consumer* einerseits vor der Initiierung des intendierten Datenaustauschs die aktuellen *Session Informationen* vom *Service Provider* ab und abonniert andererseits Änderungen der entsprechenden Kontextzustände. Passen die *Session IDs* vom *Service Provider* und *Consumer* zusammen, kann der *Consumer* sicher sein, dass die empfangenen Informationen zum richtigen Patienten, beim richtigen Eingriff, im richtigen OP-Saal gehören.

Zudem ermöglicht das *Session*-Konzept dem zu steuernden Gerät (*Service Provider*) sicherzustellen, dass nur Fernsteuerungskommandos von Geräten (*Service Consumer*) derselben *Session* angenommen werden. Hierzu werden die Mechanismen des *MDPWS Safety Contexts* und des *BICEPS Ensemble Contexts* kombiniert. Der *Safety Context* erlaubt dem *Service Provider* Anforderungen zu definieren, dass zusätzliche, kontextuelle Informationen in einem Fernsteuerungskommando einzubetten sind (siehe auch Kapitel 4.3.1 und 4.5.1). In diesem Fall kann die Einbettung der *Session ID* gefordert werden. Enthält das Kommando die richtige *Session ID*, so wird das zu steuernde Gerät die Anfrage entsprechend bearbeiten. Stimmt die übermittelte *Session ID* nicht oder ist gar nicht enthalten, wird der *Service Provider* das Kommando verwerfen, da nicht sichergestellt werden kann, dass der Absender zum intendierten Geräteensemble gehört.

Wie beschrieben wird die *Session* mittels einer Instanz des *Ensemble Contexts* umgesetzt. Entsprechend den Anforderungen des *IEEE 11073-20701* Standards erfolgt die Übertragung grundsätzlich über eine gesicherte Verbindung. Somit sind die Informationen gegen Manipulation und Vortäuschung geschützt.

6.3 Diskussion

Das vorgestellte *Session*-Konzept weist eine gewisse Komplexität auf. Es müssen sowohl ein *Session Manager* vorhanden, als auch die entsprechenden Kontextfunktionalitäten von den Vernetzungsteilnehmern implementiert sein. Daher sehen andere Konzepte vor, lediglich den Ortskontext (*Location Context*) für das Lösen derselben Problemstellungen zu nutzen. Das hat den Vorteil einer geringeren Komplexität, setzt aber eine sehr präzise und verlässliche Bestimmung des Ortes, auch von mobilen Geräten, voraus. Aus Sicherheitsgründen ist im *Location Context*-Konzept davon auszugehen, dass der Ort der Geräte von menschlichen Akteuren bestätigt werden muss. Erfolgt dies über das *MMI* des Gerätes, so muss eine einfache physikalische Erreichbarkeit der Geräte gewährleistet sein.

Der wohl entscheidende Nachteil des *Location Context*-Konzepts liegt darin, dass mit der alleinigen Nutzung des Ortes die folgenden zwei Anwendungsfälle nicht abgedeckt werden können: Ist es erforderlich, dass nur ein Teil der Geräte, die sich an demselben Ort befinden, miteinander kommunizieren, ist die ausschließliche Nutzung des *Location Contexts* nicht ausreichend. In der Praxis sind beispielsweise häufig in einem OP-Saal mehrere HF-Chirurgiegeräte für unterschiedliche Anwendungsfälle vorhanden. Daher ist sicherzustellen, dass das richtige Gerät in das System integriert wird. Außerdem kann es dazu kommen, dass an demselben Ort mehrere Geräteensembles geformt werden müssen. Dies kann etwa bei Transplantationen sinnvoll sein, bei denen sich zwei Patientinnen und entsprechendes Equipment in einem OP-Saal befinden. Ein weiteres Beispiel sind Polytraumata, bei denen ggf. mehrere OP-Teams zur selben Zeit an demselben Patienten operieren. Sind solche Anwendungsfälle nicht zu erwarten, kann auf die Umsetzung des *Session*-Konzepts zugunsten eines einfacheren Verfahrens verzichtet werden.

Der Vorteil des *Session*-Konzepts besteht darin, dass die verschiedenen Kontextinformationen zusammen betrachtet werden und von einem menschlichen Akteur mit Unterstützung des *Session Managers* validiert werden. Der Ortskontext wird einbezogen, stellt aber nur eine unter mehreren Informationsquellen dar. Der *Session Context* abstrahiert von den einzelnen Informationen und fügt ein Geräteensemble zusammen, in dem sichergestellt ist, dass Informationen und Fernsteuerungsbefehle zum richtigen medizinischen Eingriff, im richtigen OP-Saal und zur richtigen Patientin gehören.

7 Zuverlässige Fernauslösung von Medizingeräten

Die herstellerübergreifende Fernsteuerung ist einer der Kernanwendungsbereiche von *IEEE 11073 SDC*. Die Fernsteuerung kann grob in zwei Bereiche unterteilt werden: Veränderungen von Parametern und Fernauslösung von Gerätefunktionalitäten. Während bei der Veränderung von Geräteparametern etwa Alarmgrenzen oder Zielwerte verändert werden können, werden bei der Fernauslösung Aktionen am zu steuernden Gerät ausgelöst. Beispiele für das Auslösen von Gerätefunktionalitäten sind das Rotieren einer medizinischen Fräse, das Saugen oder Spülen einer Pumpe oder die Energieabgabe eines medizinischen Lasers, HF-Chirurgiegerätes oder Ultraschall-Knochenmessers. Das folgende Kapitel beschäftigt sich mit dem letztgenannten Bereich der Fernauslösung und geht dabei insbesondere auf *Safety*-Mechanismen ein.

Zur Veranschaulichung der technischen Herausforderungen sei folgende, naive Umsetzung der Fernauslösung erwähnt: Der zu steuernde *Service Provider*, z. B. das Ultraschall-Knochenmesser, stellt je eine *Activate Operation* zum Starten und zum Stoppen der Gerätefunktionalität bereit. Die Aktivierung erfolgt vom Triggern der Start-Operation bis zum Triggern der Stopp-Operation durch den *Service Consumer*. Eine solche Realisierung bietet keine Möglichkeit, Verbindungsabbrüche zu erkennen. Das Beenden der Gerätefunktionalität ist im Fall eines Verbindungsabbruchs nicht möglich, sodass eine Verletzungsgefahr für Patient und chirurgisches Team besteht. Aus Sicherheitsgründen sollte folglich von einer solchen Umsetzung Abstand genommen werden.

Dieses Kapitel basiert auf der Publikation „Mechanism for Safe Remote Activation of Networked Surgical and PoC Devices using Dynamic Assignable Controls“ [B 19], die unter der Erstautorenschaft des Verfassers dieser Arbeit entstanden ist. Das Konzept wurde für die Abschlusspräsentation des *OR.NET*-Projekts in Leipzig entwickelt und darüber hinaus in der genannten Publikation entsprechend verallgemeinert.

7.1 Einleitung und Problembeschreibung

Die Auslösung von Gerätefunktionalitäten erfolgt heutzutage fast ausschließlich durch ein dediziertes, fest mit dem Gerät verbundenes Eingabegerät, z. B. einen Fußschalter. Das Eingabegerät ist dabei derzeit über ein Kabel (oder auch drahtlos) mit genau einem Gerät, wie etwa einer Fräse, verbunden. Für komplexe Eingriffe mit einer Vielzahl von verschiedenen Geräten führt diese 1:1 Zuordnung zu einer Vielzahl von Bedienelementen. Insbesondere die steigende Anzahl an Fußschaltern ist eine Herausforderung. In der Praxis kann der Arzt die Fußschalter oft nicht sehen, da sich diese unter dem OP-Tisch und ggf. unter sterilen Tüchern befinden. Die Unterscheidung erfolgt daher häufig durch Ertasten oder Ausprobieren (*Trial-and-Error*). Dies ist fehleranfällig, zeitraubend und ggf. mit Risiken für Patientinnen und Personal verbunden. Zusätzlich steigt mit der Anzahl der Fußschalter die Wahrscheinlichkeit, dass sie durch Verschieben oder Herunterfallen von ggf. genutzten Podesten unerreichbar für den Arzt werden [A 35, 72, 214]. Abbildung 7.1 veranschaulicht einige der beschriebenen Situationen.

Löst man die starre Verbindung von Fußschalter und Medizingerät auf, so wird der Fußschalter zu einem eigenständigen Gerät. Die Gerätefunktionalität wird dann auf Basis der Gerätevernetzung ausgelöst. Heutige Lösungen basieren nicht auf Standardnetzwerken, sondern nutzen spezialisierte Feldbusse wie *Controller Area Network (CAN)*, *PROFINET*, *Ethernet POWERLINK*, *Ethernet for Control Automation Technology (EtherCAT)* etc. [B 13]. Solchen Systemen fehlt es oft an Flexibilität, z. B. fehlen *Plug-and-Play*-Fähigkeit und Skalierbarkeit. Außerdem nutzen Feldbussysteme häufig Spezialhardware. Zusätzlich ist von einer Doppelverkabelung, parallel zum Standardnetzwerk, auszugehen, die aus verschiedenen Gründen vermieden werden sollte: Mehr Kabel können zu mehr Unordnung und mehr Stolperfallen führen; zudem sind Kosten- und Hygieneaspekte von Relevanz. Aus Sicht der Hersteller bedeutet zudem die Doppelimplementierung von zwei Schnittstellen einen erheblichen Mehraufwand.

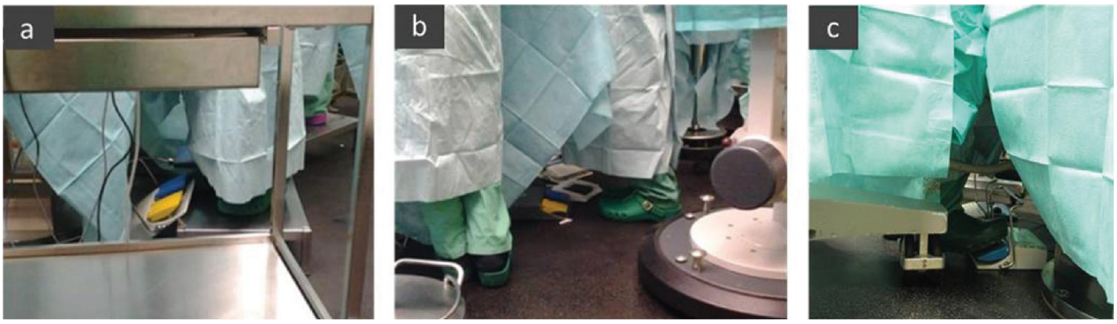

Abbildung 7.1: Fußschalter in alltäglichen OP-Situationen: a) Heruntergefallener Fußschalter; b) Vielzahl von Fußschaltern; b) und c) Verdeckung der Fußschalter durch OP-Tisch und sterile Tücher. (Abbildung entspricht *Figure 4* aus [A 72].)

Lösungen, die auf Standardnetzwerktechnologien aufsetzen und zusätzlich offene Standards nutzen, sind nicht bekannt. Nichtsdestotrotz sind die Grundprinzipien, die in diesem Kapitel vorgestellt werden, bereits etabliert. So werden Mechanismen wie inkrementelle Nachrichten-IDs oder das periodische Senden von Nachrichten/Kommandos bereits in Standards und Spezifikationen beschrieben. Beispiele hierfür sind die Norm *IEC 61784-3* [A 215], die sich mit der funktionalen Sicherheit von Kommunikation beschäftigt, oder die *Prüfgrundsatz-Spezifikation GS-ET-26* [A 216], die entsprechende Vorgaben für „Bussysteme für die Übertragung sicherheitsbezogener Nachrichten“ definiert.

7.2 Sichere Gerätezustände und Systemanforderungen

Im Folgenden werden einige Vorüberlegungen angestellt und Anforderungen abgeleitet, die ein System zur zuverlässigen Fernauslösung über potentiell unzuverlässige Netzwerke erfüllen muss. Standardnetzwerktechnologien wie *Ethernet* oder WLAN können nicht per se als zuverlässig angesehen werden.

7.2.1 Sichere Gerätezustände

Im Betrieb von Medizingeräten bestehen Risiken, die im Zuge des Risikomanagements zu bewerten sind. Ausgehend von dieser Analyse bestehen zwei grundlegende Gegenmaßnahmen: die Minimierung der Eintrittswahrscheinlichkeit und die Minimierung der negativen Auswirkungen. Die einzelnen Hersteller haben das Netzwerk in der Regel nicht unter ihrer Kontrolle. Daher kann die Zuverlässigkeit und damit die Eintrittswahrscheinlichkeit von Verbindungsabbrüchen, hohen Latenzen und Jitter[7.1] nicht optimiert werden. Der Hersteller stellt in der Regel Anforderungen an die Eigenschaften des Netzwerks. Sind diese nicht erfüllt, sollte dennoch sichergestellt werden, dass keine potentiell gefährlichen Situationen entstehen. Somit ist die Minimierung der negativen Auswirkungen Gegenstand des in diesem Kapitel vorgestellten Konzepts. Die Auswirkungen einer gestörten Fernauslösung von Gerätefunktionalitäten lassen sich am effektivsten minimieren, wenn das gesteuerte Gerät in einen, für den Patienten und das klinische Personal, sicheren Zustand übergeht.

Die Charakteristik eines sicheren Zustands der Gerätefunktionalität ist sehr unterschiedlich. Betrachtet man Medizingeräte, die im Normalbetrieb einen gewünschten „destruktiven“ Effekt haben, dann führt die Abschaltung der Gerätefunktionalität zu einem sicheren Zustand. Beispielsweise werden chirurgische Fräsen zum Entfernen von Knochenteilen oder anderem Gewebe genutzt („destruktiv“). Eine unkontrollierte Rotation des Fräskopfs ist im Fehlerfall zu verhindern. Das Abschalten der Gerätefunktionalität führt also zu einem sicheren Zustand, da die Fräse im ausgeschalteten Zustand keinen unerwünschten Schaden am Patienten oder Personal verursachen kann.

[7.1] Jitter ist definiert als die Varianz (bzw. Standardabweichung) der Paketlatenz [A 217]. Ein hoher Jitter bedeutet, dass die Laufzeitunterschiede zwischen den Paketen groß sind.

So würde die Fräse Mechanismen zur Sicherheitsabschaltung einleiten, die etwa in der Norm *IEC 61800-5-2* [A 218] als „*Safe Stop 1*“, „*Safe Stop 2*“ oder „*Safe Torque Off*“ beschrieben sind.

Im Gegensatz dazu gibt es Gerätegruppen, deren sicherer Zustand die eingeschaltete Gerätefunktionalität bzw. der Übergang in einen Notfallbetrieb ist. So sollten lebenserhaltende Geräte, wie ein Beatmungsgerät, ihren Betrieb fortsetzen, wenn Probleme mit der Verbindung zu einem fernsteuernden Vernetzungspartner bestehen. Dazu würde das Gerät ggf. in einen Modus wechseln, der für den unvernetzten Betrieb ausgelegt ist.

Des Weiteren existieren Geräte, bei denen zu jeder Zeit das Starten und Stoppen der Gerätefunktionalität möglich sein muss. So ist die Schneidefunktion eines HF-Chirurgiegerätes der ersten Klasse („destruktive“ Funktionalität) zuzurechnen, während die Koagulationsfunktionalität zum Stillen von Blutungen immer präzise zur Verfügung stehen muss. Es sei angemerkt, dass in der klassischen Risikobewertung auch für die Koagulationsfunktionalität der Zustand *Aus* als sicher angesehen wird. Zur Verdeutlichung der unterschiedlichen Anforderungen wird für die vorliegende Arbeit dennoch diese Charakteristik von Gerätefunktionalitäten als eigenständige Klasse betrachtet.

Entsprechend können drei Klassen von Geräten bzw. Gerätefunktionalitäten unterschieden werden. Diese Klasseneinteilung ist nicht mit der Geräteklassifizierung nach ihrer Zweckbestimmung in die Klassen *I, IIa, IIb* und *III* entsprechend *Medical Device Directive (MDD)* [A 11] bzw. *Medical Device Regulation (MDR)* [A 10] zu verwechseln.

Klasse 1 Sicherer Zustand der Gerätefunktionalität: *Aus*.

Klasse 2 Sicherer Zustand der Gerätefunktionalität: *Ein* bzw. spezielles „Offline-Programm“.

Klasse 3 Die Zustände *Ein* und *Aus* der Gerätefunktionalität müssen zu jeder Zeit erreichbar sein.

Aus der Sicht der Gerätevernetzung und des damit verbundenen Risikomanagements sind die Klassen 1 und 2 äquivalent. Eine Fehlfunktion im Netzwerk muss erkannt werden, um eine geeignete Gegenmaßnahme einzuleiten. Worin diese Gegenmaßnahme besteht, ist Teil der Gerätelogik und unabhängig von der Vernetzung. Im Folgenden wird die Klasse 1 betrachtet. Die Mechanismen lassen sich ohne Beschränkung der Allgemeinheit auf die Klasse 2 übertragen.

Gerätefunktionalitäten der Klasse 3 werden vom beschriebenen Konzept nicht abgedeckt. Für diese Anforderungen ist ein zuverlässiges Netzwerk mit deterministischem Zeitverhalten notwendig. Es bestehen harte Echtzeitanforderungen. Sollen Anwendungen der Klasse 3 über *IEEE 11073 SDC* in Verbindung mit einem Standardnetzwerk realisiert werden, so ist die Funktionalität auf eine der Klassen 1 oder 2 herunterzubrechen. Dies ist aber nur dann möglich, wenn das Erreichen der Klasse 3 im Fehlerfall durch ein geeignetes Rückfallsystem innerhalb einer akzeptablen Zeitschranke sichergestellt werden kann. So kann beispielsweise ein herkömmlich angeschlossener Fußschalter eines HF-Chirurgiegerätes als Rückfallsystem für einen dynamisch vernetzten Fußschalter dienen.

7.2.2 Systemanforderungen

In potentiell unzuverlässigen Standardnetzwerken, die ein nicht-deterministisches Zeitverhalten aufweisen, kann es zu einer Reihe von Problemen kommen. Diese Probleme reichen von einer hohen Latenz und hohem Jitter über Paketverluste bis hin zu kompletten Verbindungsabbrüchen. Vereinfacht dargestellt stellen oft Engpässe (*Bottlenecks*) in Standardnetzwerken und die damit verbundene Nutzung von Warteschlangen (*Buffer*) für Datenpakete [A 217] die Ursache für die erstgenannten Probleme dar.

Das Herausreißen bzw. die Zerstörung von Kabeln ist eine häufige Ursache für Verbindungsabbrüche. Es sei angemerkt, dass auch ein spezialisiertes, echtzeitfähiges Feldbussystem nicht vor diesen Problemen schützt. Daher ist eine Erkennung von Verbindungsabbrüchen zur Einleitung von entspre-

chenden Gegenmaßnahmen generell notwendig. Wie bereits erwähnt ist das periodische Senden von Kommandos gängige Praxis.

Zur Ableitung einer weiteren Systemanforderung wird folgendes verallgemeinerbares Beispiel eines Ultraschall-Knochenmessers betrachtet. Es zeigt, dass periodisches Senden von Kommandos allein nicht ausreichend ist: Die Ärztin aktiviert eine Gerätefunktionalität über einen Fußschalter. Aufgrund einer erhöhten Latenz bzw. Jitter kommen die Pakete zur periodischen Reaktivierung der Schneidefunktion nicht rechtzeitig beim Knochenmesser an. Die Ärztin stellt folglich fest, dass die Schneidefunktion vom Gerät beendet wird. Dieses Verhalten zeigen solche Geräte auch, wenn das Schneiden mit einem zu hohen Druck durch die Ärztinnen ausgeführt wird (ein sogenannter *Overload*). Um die Schneidefunktion nach einem *Overload* erneut nutzen zu können, muss der Fußschalter zunächst gelöst werden. Werden nun die periodischen Reaktivierungspakete vom Knochenmesser nach einer unvorhersehbaren Zeit empfangen und verarbeitet, führt dies dazu, dass die Schneidefunktion ausgelöst wird, obwohl die Ärztin den Fußschalter nicht gedrückt hat. Eine solches Verhalten würde zu einer Situation mit einem hohen, unkalkulierbaren Risiko für Patient und Personal führen. Somit ist zwingend erforderlich, dass verzögert eintreffende (Re-)Aktivierungskommandos klar von intendierten Kommandos unterschieden werden können.

Aus den angestellten Vorüberlegungen lassen sich folgende, zentrale Anforderungen an die zu spezifizierenden Mechanismen stellen:

Anforderung 1 Sichere Erkennung von Verbindungsabbrüchen.

Anforderung 2 Sichere Erkennung von falschen (Re-)Aktivierungskommandos durch hohen Jitter.

7.3 Mechanismen zur zuverlässigen Fernauslösung von Gerätefunktionalitäten

Auf der Basis der Vorüberlegungen und der gegebenen Anforderungen wurden Mechanismen entwickelt, die eine zuverlässige Fernauslösung von kritischen Gerätefunktionalitäten zulassen. Die Mechanismen werden im Folgenden beschrieben.

7.3.1 Periodische Reaktivierung der Gerätefunktionalität

Anforderung 1 kann mittels periodischer Reaktivierung (*Retriggering*) der Funktionalität erfüllt werden. Hierfür spezifiziert der *Service Provider* in der Beschreibung der zugehörigen *Activate Operation* zwei Eigenschaften: Mithilfe des Attributes `InvocationEffectiveTimeout` wird spezifiziert, wie lange die Gerätefunktionalität ausgelöst wird, bevor der Effekt der Operationsauslösung automatisch vom *Service Provider* beendet wird. Des Weiteren wird das Attribut `Retriggerable` auf `true` gesetzt. Somit erhalten *Service Consumer* die Information, dass eine Reaktivierung vor dem Ablauf der definierten Aktivierungszeit (`InvocationEffectiveTimeout`) möglich ist.

Auf der Basis dieser Selbstbeschreibung kann eine kontinuierliche Auslösung der Gerätefunktionalität mittels *IEEE 11073 SDC* realisiert werden. Kommt es zu einem Verbindungsabbruch, erfolgt ein Timeout der Auslösung automatisch. Für eine Jitter-Erkennung entsprechend Anforderung 2 ist ein einfaches *Retriggering*, wie hier beschrieben, nicht ausreichend. Der Mechanismus muss um die Fähigkeit zur Unterscheidung von zwei Situationen erweitert werden:

(a) Unterbrechung des *Retriggerings*, ausgelöst durch Netzwerk-Jitter

(b) Beendigung und Neuauslösung durch den Nutzer

Situation (b) tritt u. a. dann auf, wenn der Arzt den Fuß kurz vom Fußschalter nimmt und unmittelbar wieder herunterdrückt. Dies darf nicht als Jitter erkannt werden, da ansonsten kurzfristig keine erneute Aktivierung erfolgen würde.

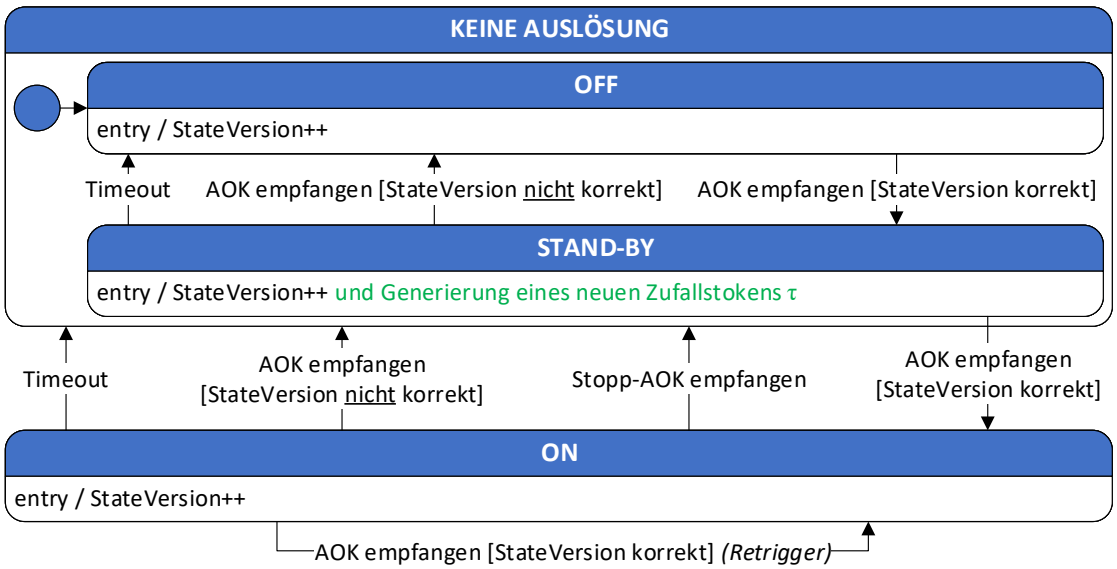

Abbildung 7.2: *UML-Zustandsdiagramm* zur Beschreibung der Auslösung von Gerätefunktionalitäten. Grün gekennzeichnet: *Zufallstoken*-Erweiterung von Kapitel 7.3.2. Abkürzung: AOK – *Activate Operation*-Kommando. (Abbildung nach *Fig. 1* aus [B 19]. Original © 2016 IEEE.)

Zur Jitter-Erkennung, insbesondere am Anfang einer Auslösung der Gerätefunktionalität, wird eine Modellierung mit drei Zuständen genutzt: `OFF`, `STAND-BY` und `ON`. In den Zuständen `OFF` und `STAND-BY` erfolgt keine Aktivierung der Gerätefunktionalität, sodass sie funktionell zu einem Elternzustand `KEINE AUSLÖSUNG` zusammengefasst werden können, wie es in Abbildung 7.2 veranschaulicht ist. Diese Zustände können in Form einer Metrik vom Typ `EnumStringMetric` mit dem entsprechend definierten Wertebereich modelliert werden. Eine ggf. wünschenswerte Abbildung auf den *Activation State* der Gerätekomponente kann nicht unmittelbar erfolgen. Die in der Norm *IEEE 11073-10207* beschriebene Semantik der *ComponentActivation*-Zustände *Off*, *Stand-By* und *Not Ready* (siehe auch Kapitel 4.4.1.2) ist nicht ohne Weiteres auf die in diesem Konzept beschriebenen Zustände `OFF` und `STAND-BY` abbildbar. Entsprechendes wäre in Standards für Gerätespezialisierungen oder *PKPs* zu definieren.

Die Komponente, die den zu aktivierenden Teil des Gerätes repräsentiert, wird in der Beschreibung der *Activate Operation* mittels des Attributs `OperationTarget` referenziert. Dieses referenzierte Objekt verfügt, wie jedes Element der Gerätebeschreibung, über einen Zustand mit entsprechendem Zustandsversionszähler (*State Version*). Die *State Version* wird bei jeder Auslösung der *Activate Operation* durch einen *Service Consumer* um eins inkrementiert. Dies gilt auch für *Retrigger*-Kommandos, die zu einem Verbleib im Zustand `ON` führen, da auch Selbsttransitionen als Zustandsänderung zu interpretieren sind.

Als zusätzlichen Sicherheitsmechanismus fordert der *Service Provider* mit Hilfe des *MDPWS Safety Contexts*, dass die *State Version* des referenzierten *Activate Operation Targets* in jedes *Activate Operation*-Kommando des *Service Consumers* eingebettet werden muss (siehe auch Kapitel 4.3.1, 4.3.5 und 4.5.1).

Die Zustandsübergänge und das entsprechende Inkrementieren der *State Version* sind in Abbildung 7.2 in Form eines Zustandsautomaten in *Unified Modeling Language (UML)*-Notation veranschaulicht. Die Transitionen zwischen den Zuständen sind mit Bedingungen modelliert, die in eckigen Klammern *[Bedingung]* beschrieben werden. Die *entry*-Schreibweise innerhalb eines Zustands beschreibt das Verhalten beim Betreten des Zustands. Dies wird zur Definition des Inkrementierens der *State Version* genutzt. Transitionen können durch empfangene *Activate Operation*-Kommandos des *Service Consumers* oder durch einen internen Timeout im *Service Provider* erfolgen.

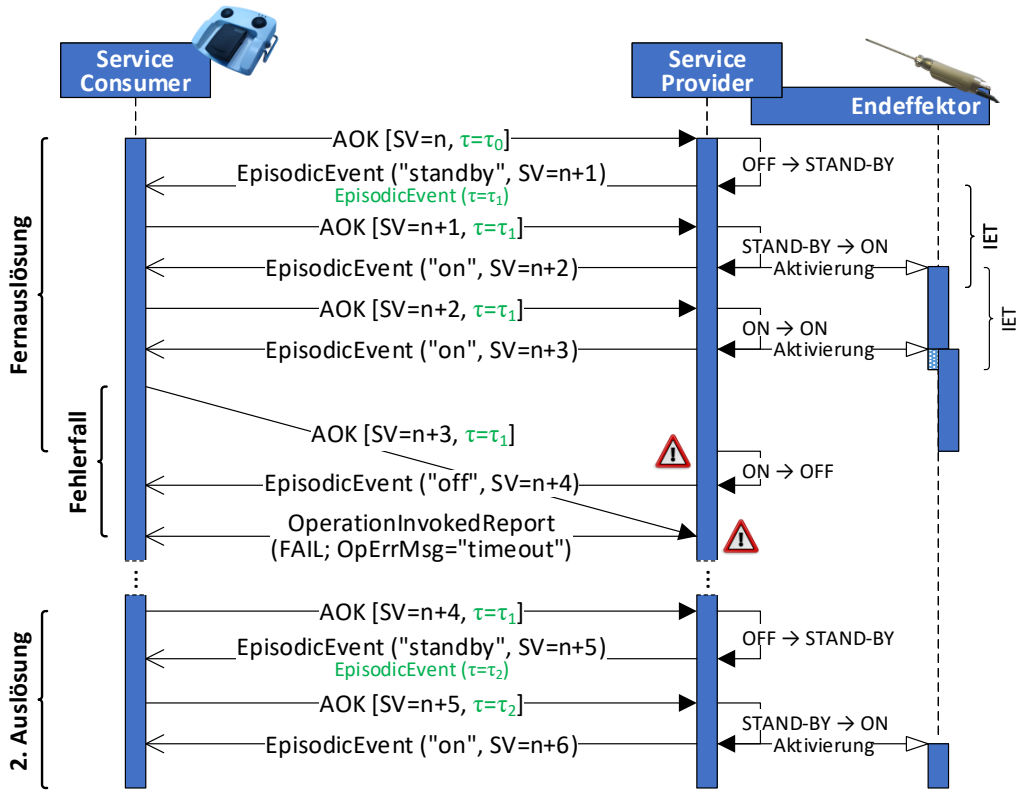

Abbildung 7.3: Sequenzdiagramm der Fernauslösung von Gerätefunktionalitäten, inkl. Timeout mit verzögert ankommendem *Activate Operation*-Kommando (AOK) und einer zweiten Auslösung. Die Zeitachse ist aus Darstellungsgründen nicht äquidistant; so sind beispielsweise die Zeitdauern der Zustandsübergänge und Event-Antworten im Verhältnis zu lang dargestellt. Die grün gekennzeichneten Anteile beschreiben die Erweiterung von Kapitel 7.3.2. Abkürzungen: SV – *State Version*; IET – `InvocationEffectiveTimeout`; τ – *Zufallstoken*. (Abbildung nach *Fig. 2* aus [B 19]. Original © 2016 IEEE.)

Der Prozess der Auslösung der Gerätefunktionalität wird in Abbildung 7.3 als *UML*-Sequenzdiagramm veranschaulicht. Mit der Verarbeitung des ersten *Activate Operation*-Kommandos wird die Transition vom Zustand `OFF` in den Zustand `STAND-BY` ausgelöst. Es erfolgt keine Auslösung der eigentlichen Gerätefunktionalität. Die *Target State Version* wird von `n` auf `n+1` erhöht. Über die entsprechenden Events wird der *Service Consumer* über diese Änderungen informiert.

Das nächste *Activate Operation*-Kommando des *Service Consumers* enthält die *State Version* `n+1` in seinem *Safety Context*. Empfängt der *Service Provider* dieses Kommando vor dem Ablauf der definierten Aktivierungszeit (`InvocationEffectiveTimeout`), so erfolgt der Übergang in den Zustand `ON`. Nun wird auch die Gerätefunktionalität ausgelöst. Im Beispiel wird die Schneidefunktion des Ultraschall-Knochenmessers aktiviert. Wird kein entsprechendes Kommando vor dem Ablauf des Timeouts empfangen, so erfolgt der Übergang in den Zustand `OFF`. In beiden Fällen wird die *State Version* auf `n+2` erhöht.

Der `STAND-BY`-Zustand ist notwendig, um Jitter bei der initialen Auslösung der Geräteaktivierung erkennen zu können. Zur Veranschaulichung sei folgendes Szenario betrachtet: Der Fußschalter wird zur Auslösung des Knochenmessers betätigt. Das entsprechende *Activate Operation*-Kommando hat eine hohe Latenz, sodass keine unmittelbare Auslösung des Knochenmessers erfolgt. Daraufhin löst der Arzt die Betätigung des Fußschalters. Erreicht nun das Kommando das Knochenmesser, soll die Schneidefunktion nicht ausgelöst werden, da der Fußschalter nicht mehr betätigt ist und dies zu einem potentiellen Schaden von Patientin und/oder Personal führen kann. Durch die Einführung des `STAND-BY`-Zustands, in dem die Gerätefunktionalität nicht ausgelöst wird, kann eine solche Situa-

tion erkannt werden. Mit Hilfe dieses Zwischenzustandes kann der *Service Provider* beurteilen, ob das *Activate Operation*-Kommando, das tatsächlich die Aktivierung der Gerätefunktionalität auslöst, innerhalb des definierten Zeitbereichs angekommen ist.

Befindet sich die Gerätefunktionalität des *Service Providers* im Zustand `ON`, so führt jeder weitere Aufruf der *Activate Operation*, der innerhalb der gesetzten Deadline eintrifft, zu einer Selbsttransition in den Zustand `ON`. Damit erfolgt eine kontinuierlich anhaltenden Auslösung der Gerätefunktionalität. Die vom *Service Consumer* in den *MDPWS Safety Context* einzubettende *State Version* wird entsprechend inkrementiert. Das erfolgreiche Initiieren und *Retriggering* wird im oberen Teil von Abbildung 7.3 veranschaulicht.

Im Fall einer erhöhten Netzwerklatenz bzw. Jitter kann es dazu kommen, dass *Activate Operation*-Kommandos den *Service Provider* nach dem spezifizierten Timeout (`InvocationEffectiveTimeout`) erreichen. Dieser Fall ist im mittleren Teil von Abbildung 7.3 dargestellt. Durch die Timeout-getriebene Sicherheitsabschaltung der Gerätefunktionalität geht der Zustand von `ON` in `OFF` über. Das führt zu einer Erhöhung der *State Version*, im Beispiel von `n+3` auf `n+4`. Wird nun das verspätet eingetroffene Kommando verarbeitet, enthält dieses die *State Version* `n+3` im *Safety Context*. Da diese *State Version* nicht zur aktuellen passt, wird das Kommando vom *Service Provider* verworfen. Der *Service Consumer* wird hierüber durch eine Eventbenachrichtigung des Typs `OperationInvokedReport` informiert. Das zugehörige `InvocationState`-Element enthält den Wert `Fail`. Zusätzlich sollten weitere Informationen in den Elementen `InvocationError` und `InvocationErrorMessage` übertragen werden[7.2]. Ebenso ist es sinnvoll, dass der *Service Provider* einen technischen Alarm/Hinweis propagiert, sodass auch weitere Vernetzungsteilnehmer über potentielle Netzwerkprobleme informiert sind. Das kann sowohl für eigene Fernsteuerungskommandos von Relevanz sein, als auch für Logging-Funktionalitäten.

Sendet der *Service Consumer* nach dem Timeout ein neues *Activate Operation*-Kommando, so wird dieses die korrekte *State Version* enthalten. Sie wurde durch den Übergang in den Zustand `OFF` erzeugt und als *Event* propagiert. Im Fall von vorangegangenen Verbindungsproblemen erfragt der *Service Consumer* die *State Version* ggf. aktiv. Im Beispiel im unteren Teil von Abbildung 7.3 ist das die *State Version* `n+4`. Entsprechend dem beschriebenen Mechanismus erfolgt der Übergang vom Zustand `OFF` in `STAND-BY` und die Gerätefunktionalität wird mit dem kommenden *Activate Operation*-Kommando ausgelöst.

Es kann also mit Hilfe des vorgestellten Mechanismus zwischen verspätet eingetroffenen Kommandos und einer erneuten, vom Nutzer intendierten, Auslösung der Gerätefunktionalität unterschieden werden. Somit wird auch die zweite gestellte Anforderung erfüllt.

Zusätzlich zur Beendigung der Auslösung der Gerätefunktionalität durch den beschriebenen Timeout kann der *Service Provider* eine *Stopp Activate Operation* zum direkten Beenden vorsehen. Damit verringert sich die Verzögerung zwischen dem Lösen des Fußschalters und dem Ende der Auslösung. Da diese Operation nur von dem *Service Consumer* genutzt werden sollte, der zuvor die Auslösung vorgenommen hat, verwirft der *Service Provider* Kommandos von anderen *Service Consumern*. Die Authentifizierung erfolgt mit Hilfe der *X.509.v3-Zertifikate*.

Durch die Einführung des `STAND-BY`-Zustands erfolgt die Auslösung der Gerätefunktionalität wie beschrieben nicht unmittelbar mit dem ersten Kommando, sondern erst mit der Verarbeitung des zweiten. Hierdurch sinkt die Reaktivität des Systems. Zieht man die ermittelten Kommunikationslatenzzeiten und die Anforderungen, die sich aus der menschlichen Reaktions- und Wahrnehmungszeit ergeben, aus Kapitel 5 heran, so kann davon ausgegangen werden, dass reale Systeme keine merkliche Verzögerung aufweisen werden. Insbesondere im Fall der Betätigung von Schaltern mit dem Fuß und

[7.2] Die genaue Ausgestaltung des Inhalts dieser zusätzlichen Informationen kann in Standards zu *PKPs* und/oder Gerätespezialisierungen erfolgen.

der Aktivierung von aufgrund ihrer Mechanik trägen Geräten ist von einer hohen Verzögerungstoleranz der Nutzenden auszugehen.

Als Alternative zum zusätzlich eingeführten `STAND-BY`-Zustand kann eine Umsetzung durch zwei separate *Activate Operations* für jede Geräteaktivierung erfolgen. Die erste *Activate Operation* stellt eine „Anfrage zur Geräteaktivierung“ dar. Die zweite *Activate Operation* realisiert die eigentliche Geräteaktivierung und ist meist deaktiviert (`OperatingMode = Disabled`). Das *Activate Operation*-Kommando des *Service Consumers* für die erste *Activate Operation* löst noch keine Geräteaktivierung aus, sondern führt dazu, dass der *Service Provider* die zweite *Activate Operation* im Netzwerk freigibt (`OperatingMode = Enabled`). Der so erreichte Zustand ist analog zum `STAND-BY`-Zustand. Löst der *Service Consumer* innerhalb des definierten Zeitraums die zweite *Activate Operation* aus, so erfolgt die Geräteaktivierung. Nach der Beendigung der Geräteaktivierung wird die *Activate Operation* wieder entsprechend deaktiviert.

Der Vorteil in der Aufteilung der Geräteaktivierung in zwei *Activate Operations* besteht darin, dass die *ComponentActivation*-Zustände für die Beschreibung der Geräteaktivierung direkt genutzt werden können, da der in dieser Arbeit definierte `STAND-BY`-Zustand entfällt. Die Ausdrucksmächtigkeit und die im Zuge dieses Kapitels beschriebenen Mechanismen sowie das Zeitverhalten können als analog angesehen werden. Das Aufteilen der Geräteaktivierung in zwei *Activate Operations* setzt wohl definierte Nomenklatur-Terme zur Beschreibung und Verknüpfung dieser Operationen voraus. Da diese kurzfristig nicht zur Verfügung stehen, wird im Zuge dieser Arbeit die Realisierung mittels des eingeführten `STAND-BY`-Zustands beschrieben. In zukünftigen Standardisierungsprozessen können beide Möglichkeiten verfolgt werden.

7.3.2 Zufallstoken – Maßnahmen gegen das Vorausberechnen von State Versions

Die Mechanismen zur Absicherung der periodischen Reaktivierung der Gerätefunktionalität mittels inkrementeller *State Versions* können als bekannt vorausgesetzt werden. Daher besteht die Möglichkeit, dass ein „bösartiger“ *Service Consumer* nicht die asynchronen Updates des *Service Provider*-Zustands abwartet, sondern die neue erwartete *State Version* selbstständig berechnet. Geschieht dieses in Verbindung damit, dass der *Service Consumer* die *Retrigger Activate Operation*-Kommandos versendet, ohne die Events vom *Service Provider* abzuwarten, kann es dazu kommen, dass eine valide anmutende Sequenz von *Activate Operation*-Kommandos verzögert vom *Service Provider* verarbeitet wird. Dies kann sowohl während des Initialisierungsprozesses (linker Teil von Abbildung 7.4), als auch beim *Retriggering*-Prozess, genau gesagt nach einem Timeout (rechter Teil von Abbildung 7.4), auftreten.

Wie in Abbildung 7.4 mit Hilfe der Aktivitätsbalken von Fußschalter und Endeffektor veranschaulicht ist, kann es durch eine verzögerte Sequenz mit vorausberechneten *State Versions* dazu kommen, dass eine Auslösung der Gerätefunktionalität erfolgt, ohne dass die Ärztin den Fußschalter betätigt. In beiden Abbildungen ist am rechten Rand die aktuelle *State Version* dargestellt. So ist nachvollziehbar, dass alle drei exemplarisch dargestellten *Activate Operation*-Kommandos im linken Teil von Abbildung 7.4 als valide angesehen werden und es entsprechend zu einer Auslösung der Gerätefunktionalität kommt. Im Beispiel des rechten Teils von Abbildung 7.4 wird das Kommando mit der *State Version* `n` zwar verworfen, aber die folgenden Kommandos führen dennoch ebenfalls zu einer Auslösung.

Beide beschriebenen Problemfälle können durch die Einführung eines *Zufallstokens* gelöst werden. Dieses wird in Form einer zusätzlichen Metrik umgesetzt. Der *Service Provider* erweitert die Anforderungen des *Safety Contexts* in der Form, dass das *Zufallstoken* in das *Activate Operation*-Kommando eingebettet werden muss. Ein neuer Zufallswert wird jeweils beim Betreten des Zustands `STAND-BY` erzeugt. Somit ist die Bedingung in den beschriebenen Problemfällen für den Übergang vom `STAND-BY` in den `ON` Zustand nicht erfüllt, da der *Service Consumer* das *Zufallstoken* nicht vorbe-

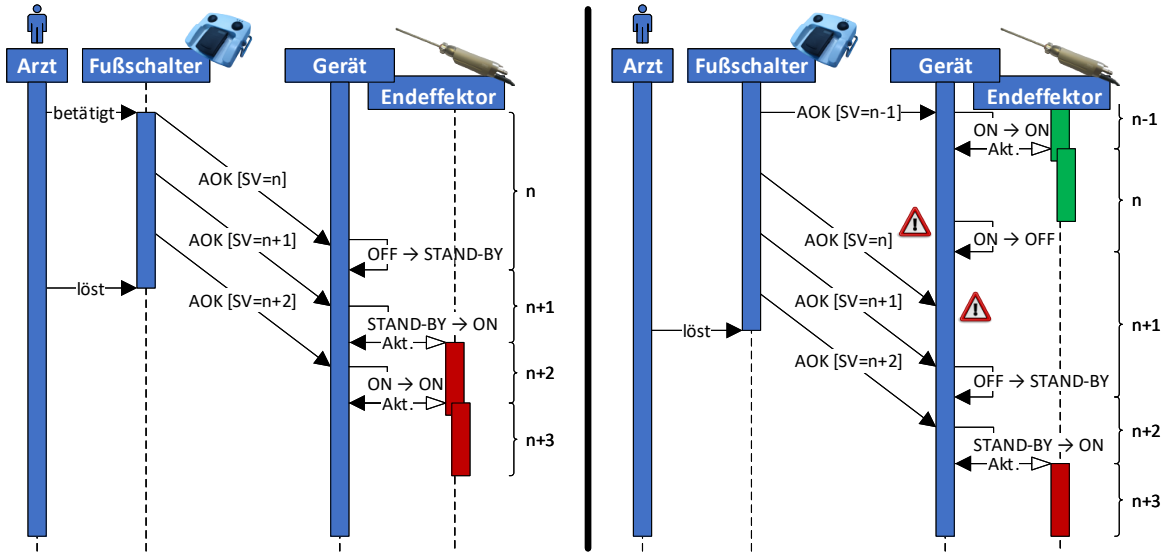

Abbildung 7.4: „Bösartig“ vorausberechnete *State Versions*. Links – Situation bei der Initialisierung: Fußschalter wird gedrückt, aufgrund einer Verzögerung erfolgt keine unmittelbare Auslösung. Aktivierung erfolgt verzögert, ggf. nach dem Lösen des Fußschalters (rot). Rechts – Situation nach einem Timeout: Verzögerung der Kommunikation setzt ein, während der Fußschalter gedrückt ist und die Aktivierung läuft (grün). Timeout wird erkannt und Aktivierung beendet. Es erfolgt eine erneute Aktivierung, ggf. nach dem Lösen des Fußschalters (rot). Abkürzung: AOK – *Activate Operation*-Kommando.

rechnen kann. Diese Erweiterung ist in den Abbildungen 7.2 und 7.3 in Grün dargestellt. Der Overhead für diese Erweiterung ist gering, da das *Zufallstoken* nur beim Betreten des `STAND-BY`-Zustands neu generiert wird, sodass ein entsprechendes Event nur einmal während der Initialisierung der Auslösung übertragen werden muss.

Alternativ zum *Zufallstoken* könnten periodische Selbsttransitionen im `OFF`-Zustand eingeführt werden, mit einer Periodendauer die dem `InvocationEffectiveTimeout` entspricht. Diese würden dazu führen, dass die vorberechnete *State Version* im *Activate Operation*-Kommando nicht mehr zur aktuellen *State Version* passt. Der Nachteil besteht darin, dass das erste *Activate Operation*-Kommando einer eigentlich validen Auslösung fälschlicherweise verworfen werden könnte, wenn die Selbsttransition in der Zeit ausgelöst wird, in der der steuernde *Service Consumer* das *Activate Operation*-Kommando bearbeitet bzw. versendet. In diesem Fall würden die *State Versions* ebenfalls nicht zusammenpassen, obwohl kein Fehlerfall vorliegt.

7.3.3 Maßnahmen gegen Activate Operation Command Flooding

Ein weiteres „bösartiges“ Verhalten von *Service Consumern* kann im Senden einer Vielzahl von *Activate Operation*-Kommandos in kurzer Zeit (*Message Flooding*) bestehen. Zur Veranschaulichung des Problems wird folgendes Szenario betrachtet: Nach dem beschriebenen Initialisierungsvorgang (auch mit *Zufallstoken*) können *Service Consumer* die *State Versions* vorberechnen, da das *Zufallstoken* nur beim Übergang in den Zustand `STAND-BY` neu erzeugt wird. Es kann also eine Vielzahl von validen Kommandos in kurzer Zeit an den *Service Provider* gesandt werden. Ist der *Service Provider* nicht in der Lage, diese Kommandos hinreichend schnell zu verarbeiten, so kann es dazu kommen, dass die Kommandos lokal gepuffert werden, beispielsweise von der genutzten Kommunikationsbibliothek, die *IEEE 11073 SDC* implementiert. Löst nun der Arzt den Fußschalter, verarbeitet der *Service Provider* weiterhin „veraltete“ Kommandos, sodass die Auslösung der Gerätefunktionalität länger anhält als vom Arzt erwartet. Auch versandte *Activate Operation*-Kommandos zum expliziten Stoppen der Auslösung sind betroffen, da sie ebenfalls in die Warteschlange zur Abarbeitung eingereiht werden.

Das Problem kann mit einer internen *Blockierungszeit* (*Blocking Period*) gelöst werden, in der alle eingehenden Kommandos mit der entsprechenden *Handle*-Referenz nicht verarbeitet und stattdessen verworfen werden. Ein solches Aussortieren kann effizient realisiert werden, da lediglich ein Vergleich der *Handle*-Referenz auf die Operation notwendig ist und keine verarbeitende Gerätelogik im eigentlichen Sinne benötigt wird. Diese Erweiterung schützt vor einem *Activate Operation Command Flooding* und kann damit auch in einem gewissen Maße zur Robustheit gegenüber gewollten und ungewollten *Denial-of-Service (DoS)*-Attacken[7.3] beitragen. Aus internen Beschreibungen von Produktschnittstellen ist dem Autor bekannt, dass ein solches Konzept beispielsweise für die Steuerung von OP-Tischen in der Praxis genutzt wird.

Es wird diskutiert, ein weiteres Attribut in das *IEEE 11073-10207* Datenmodell aufzunehmen, um Mechanismen wie die *Blockierungszeit* in der Selbstbeschreibung der Geräte realisieren zu können. Die *RefractoryPeriod* soll zukünftig beschreiben, wie lange eine Fernsteuerungsoperation nach einer Auslösung nicht erneut ausgelöst werden kann. Während dieser *Blockierungszeit* wird der *OperatingMode* der Operation auf nicht aktiv gesetzt.

Die *Blockierungszeit* muss entsprechend den Ressourcen des *Service Providers* gewählt werden. Eine zu lange Zeitspanne kann dazu führen, dass valide Kommandos verworfen werden. Es wird eine empirisch ermittelte *Blockierungszeit* von 25 % der Effektdauer der *Activate Operation* (*InvocationEffectiveTimeout*) vorgeschlagen.

Als Alternative zur *Blockierungszeit* könnte das *Zufallstoken* auch bei jeder Transition und Selbsttransition in den Zustand `ON` aktualisiert werden. Die permanente *Zufallstoken*-Generierung erzeugt aber einen höheren Aufwand für *Service Provider* und *Consumer*. Des Weiteren hat der *Blockierungszeit*-Mechanismus den beschriebenen zusätzlichen positiven Effekt für die Robustheit des Systems. Daher wird die Nutzung der *Blockierungszeit* vorgeschlagen.

7.3.4 Security-Aspekte

In realen OP-Umgebungen gibt es mehrere Fernsteuerungseinheiten: Fußschalter, zentrale Arbeitsstationen, Schaltelemente an Handgriffen von Mikroskopen oder Endoskopen etc. In einer dynamisch vernetzten Umgebung können diese mit demselben zu steuernden Gerät assoziiert werden. Entsprechend dem medizinischen Anwendungsfall ist es mit hoher Wahrscheinlichkeit notwendig, dass die Auslösung der Gerätefunktionalität über einen gewissen Zeitraum nur von einer Fernsteuerungseinheit vorgenommen werden kann. Der zu steuernde *Service Provider* muss also entscheiden können, ob ein Auslösekommando anzunehmen oder zu verwerfen ist. Dies erfolgt auf der Basis des Risikomanagements. Hierfür ist eine eindeutige Identifizierung des Vernetzungspartners (*Service Consumer*) notwendig.

Zur Identifizierung von *Service Consumern* kann das Feld `wsa:From` im Nachrichten-*Header* genutzt werden. Der Nachteil besteht darin, dass dieses Feld optional ist und manipuliert werden kann. Für eine verlässliche Authentifizierung sollte daher der *MDPWS Secure Channel* genutzt werden, sodass eine *HTTPS*-Verbindung zwischen den Vernetzungsteilnehmern aufgebaut wird. Neben der sicheren Authentifizierung stellt dieses Vorgehen auch sicher, dass Steuerkommandos nicht manipuliert werden können. Über die reine Authentifizierung hinaus können die Erweiterungen, der in *MDPWS* vorgeschriebenen *X.509.v3-Zertifikate* dazu genutzt werden, verschiedene Rollen für die Netzwerkteilnehmer, bis hin zu den Nutzenden, zu definieren. So können beispielsweise der Chefärztin mehr Rechte zugewiesen werden als dem Pflegepersonal im OP. Die Rollen ermöglichen den Aufbau von Rechtehierarchien, sodass das zu steuernde Gerät einerseits entscheiden kann, ob eine Fernsteuerung durch die Fernsteuerungseinheit überhaupt zulässig ist und welche Kommandos im Fall eines Szenarios mit mehreren Fernsteuerungseinheiten eine höhere Priorität haben.

[7.3] Angriff mittels einer großen Zahl von Anfragen, mit dem Ziel, durch eine Überlastsituation eine Nichterreichbarkeit des angegriffenen Systems herbeizuführen [A 217].

Abbildung 7.5: Demonstrator (Standort: *ICCAS* in Leipzig) zur zuverlässigen Fernauslösung von Medizingerätefunktionalitäten. Rot: Zu steuernde Medizingeräte: HF-Chirurgiegerät (oben), Pumpe (links) und Fräse (rechts). Grün: Touchscreen-basierte Fernsteuereinheit. (Abbildung nach *Fig. 2* aus [B 19]. Original © 2016 IEEE.)

Technisch gesehen ist es im beschriebenen Anwendungsfall der Fernauslösung von Gerätefunktionalitäten sinnvoll, die sichere Verbindung bereits vor der eigentlichen Fernauslösung aufzubauen und mit der *keep-alive*-Option aufrecht zu erhalten. Dies sollte zu einem Zeitpunkt geschehen, der nicht zeitkritisch ist, da der Aufbau einer sicheren Verbindung einen nicht zu vernachlässigenden Overhead impliziert. So kann der Verbindungsaufbau bereits beim Abrufen der Gerätebeschreibung oder beim Abonnieren der entsprechenden Benachrichtigungen realisiert werden.

7.4 Prototypische Umsetzung

Die beschriebenen Mechanismen wurden in einem Demonstrator mit realen Medizingeräten in einem realistischen OP-Umfeld umgesetzt. Zur Demonstration der zuverlässigen Fernauslösung der Gerätefunktionalitäten wurden drei verschiedene Medizingeräte genutzt: Fräse, Saug-/Spülpumpe und HF-Chirurgiegerät. Als Fernsteuerungseinheit dient stellvertretend für Fußschalter und andere Bedienelemente ein Tablet-PC mit touchbasierter Steuerung. Abbildung 7.5 zeigt den Demonstrator als Teil des *OR.NET*-Demonstrator-OPs am *Innovation Center Computer Assisted Surgery (ICCAS)* der Universität Leipzig. Zu Demonstrationszwecken für ein technikfernes Publikum kam auch eine Steuerung der Geräte mittels eines handelsüblichen Vortragspresenters zum Einsatz. Eine solche Realisierung dient aber lediglich der Veranschaulichung und hat keine medizinisch-technische Relevanz. Die Fernsteuerungseinheit kann beliebig mit einem der drei zu steuernden Geräte assoziiert werden.

Der Demonstrator wurde mit Hilfe von Standardhardware und Standard-*Ethernet* realisiert. Die *IEEE 11073 SDC*-Schnittstelle der Medizingeräte wurde auf der Basis der *openSDC*-Bibliothek [A 161] in Java implementiert. Für die Fernsteuerungseinheit kam die *C++*-Bibliothek *OSCLib* [A 164], inzwischen umbenannt in *SDCLib/C*, zum Einsatz. Für das vorgestellte System wurde ein Timeout für die Funktionsauslösung (`InvocationEffectiveTimeout`) von 250 ms genutzt. Setzt man dieses Timeout in Relation zu den Performanzevaluationen in Kapitel 5, so sei angemerkt, dass im Demonstrator deutlich leistungsfähigere Hardwareplattformen genutzt wurden. Die *IEEE 11073 SDC*-Konnektoren der drei Medizingeräte wurden auf einem Windows-PC mit einem Intel i7-2600 3,4 GHz und 8 GB RAM betrieben. Um trotz der in Kapitel 5.5.4 beschriebenen Performanzausreißer eine kontinuierliche Aktivierung zu erzielen, wurde das im Verhältnis hoch erscheinende Timeout von 250 ms gewählt. Hinzu kommt, dass die Anbindung der Medizin-

geräte im Demonstrator-System über eine Kette von Hard- und Softwarekomponenten erfolgt. Diese Realisierung ermöglicht eine hohe Flexibilität und gute Debugging-Fähigkeiten, die für Forschungsprojekte notwendig sind. In diesem Kontext der prototypischen Konzeptevaluation steht die Performanz hingegen nicht im Vordergrund.

Im Demonstrator erfolgte keine Latenzoptimierung von Kommunikationsstack, Anwendungslogik oder Netzwerkinfrastruktur. Wie in Kapitel 5 beschrieben, können daher in realen Systemen deutlich geringere Werte für den `InvocationEffectiveTimeout` genutzt werden. Als Referenz kann etwa die maximal gemessene *RTT* für die Kommunikation zwischen zwei *OSCLib*-Teilnehmern auf zwei *Raspberry Pi 2* von unter 15 ms zugrunde gelegt werden. Timeouts im Bereich von 50 ms bis 100 ms können ohne erheblichen Optimierungsbedarf als realistisch angesehen werden. Diese Abschätzung enthält einen großzügigen Unsicherheitsaufschlag für relevante Faktoren, wie etwa die höhere Verarbeitungskomplexität als im generischen Evaluationsbeispiel, realen Netzwerkverkehr etc.

Der vorgestellte Demonstrator konnte den Nachweis der Umsetzbarkeit des Konzepts erbringen. Für Geräte mit hoher mechanischer Trägheit und geringer Kritikalität, wie etwa einer medizinischen Pumpe, ist die prognostizierte, bzw. bereits die demonstrierte, Performanz ausreichend. Für kritische Anwendungsfälle, etwa das beschriebene Ultraschall-Knochenmesser, muss die Länge des Timeouts im Zuge des Risikomanagements bewertet werden. Gegebenenfalls sind Performanz-Optimierungen und Netzwerktechnologien mit (harten) Echtzeiteigenschaften notwendig.

Die beschriebenen *Security*-Mechanismen konnten im beschriebenen Demonstrator nicht genutzt werden, da die Kommunikationsbibliotheken diesen Funktionsumfang zum Zeitpunkt der Umsetzung nicht bereitgestellt haben. Inzwischen unterstützen die Kommunikationsbibliotheken den *MDPWS Secure Channel*, sodass die Umsetzbarkeit der beschriebenen *Security*-Mechanismen in anderen Demonstratoren im Zuge anderer Anwendungsfälle gezeigt werden konnte.

7.5 Diskussion

Das vorgestellte und prototypisch evaluierte Konzept erfüllt die gestellten Anforderungen und ermöglicht eine zuverlässige, herstellerübergreifende Fernauslösung von potentiell kritischen Gerätefunktionalitäten auf der Basis von *IEEE 11073 SDC*. Die Grundidee einer periodischen Reaktivierung der Auslösung der Gerätefunktionalität hat eine geringe Komplexität und ist im Bereich der Medizintechnik durch Standards und Prüfspezifikationen [A 215, 216] anerkannt. Die Umsetzung für offen vernetzte Medizingeräte mittels *IEEE 11073 SDC* wurde gezeigt. Zudem wurden in den Abschnitten 7.3.2 und 7.3.3 Mechanismen zum Erkennen von potentiell schädlichem Kommunikationsverhalten von steuernden Netzwerkteilnehmern präsentiert. Ob diese Erweiterungen des Grundkonzepts zur Anwendung kommen, hängt vom Risikomanagement des Herstellers ab. Zukünftige Entwicklungen innerhalb der Normenfamilie, insbesondere die *PKP*-Norm für den Bereich Fernsteuerung, werden Anforderungen enthalten, die die Verantwortung für potentielle Risiken zwischen gesteuertem Gerät (*Service Provider*) und steuerndem Gerät (*Service Consumer*) regeln. Auf der Basis der Zusicherung der Konformität mit dieser *PKP*-Norm kann ggf. auf die beschriebenen Erweiterungen gegen „bösartiges“ Verhalten verzichtet werden. Die Zusicherung der Konformität erfolgt über einen entsprechenden Eintrag im *X.509.v3-Zertifikat* und setzt ein erfolgreiches Konformitätsverfahren voraus.

Die Innovation des vorgestellten Konzepts liegt in der konsequenten und vollständigen Umsetzung der Mechanismen auf der Basis der normierten *IEEE 11073 SDC*-Kommunikationstechnologie. Ein wichtiger Vorteil ist, dass alle notwendigen Informationen für die zuverlässige Funktionsauslösung von den Vernetzungsteilnehmern zur Laufzeit ausgetauscht werden. Hierfür werden die notwendigen Anforderungen des zu steuernden Gerätes in die Selbstbeschreibung eingebettet. Die Fernsteuerungseinheit kann diese ohne a priori Wissen verarbeiten und bettet die spezifizierten, zusätzlichen Informationen in die Fernsteuerungskommandonachrichten ein. Somit wird die *Plug-and-Play*-Fähigkeit

über Herstellergrenzen hinweg möglich. Die notwendigen Informationen für ein umfassendes Risikomanagement der beteiligten Geräte werden maschineninterpretierbar ausgetauscht. Die Notwendigkeit missverständlicher und fehleranfälliger verbal beschriebener Dokumente zur Ermöglichung der Gerätevernetzung wird daher minimiert.

Das vorgestellte Konzept besitzt einen geringen Overhead. Die Hauptinformationsquelle ist die *State Version*. Damit wird ein Element genutzt, das im Datenmodell bereits explizit vorgesehen ist. Dasselbe gilt für die zu übertragenden Eventnachrichten. Somit wird zusätzlicher Overhead in der Gerätebeschreibung, den notwendigen Verarbeitungen und Berechnungen und der ablaufenden Kommunikation vermieden. Lediglich die in Kapitel 7.3.2 und Kapitel 7.3.3 beschriebenen Erweiterungen bringen einen bereits diskutierten Overhead mit sich.

Die Nutzung der *State Version* und des *Zufallstokens* erfordern keine einheitliche Zeitbasis zwischen den Vernetzungsteilnehmern. Daher ist für die vorgestellten Mechanismen keine präzise Zeitsynchronisation notwendig. Dies ist ein Vorteil gegenüber Systemen, die die Erkennung von Latenz bzw. Jitter auf der Basis von Zeitstempeln umsetzen.

Das vorgestellte Konzept ist durch *UML*-Zustands- und Sequenzdiagramme hinreichend spezifiziert. Dies ermöglicht die Entwicklung automatisierter Software-/Protokolltests. In zukünftigen Arbeiten könnte das Konzept zusätzlich durch formale Methoden beschrieben werden. Die Nutzung von Petri-Netzen oder ähnlicher Modelle wäre vorstellbar. Auf dieser Basis könnten formale Verifikationen erfolgen.

Mit dem beschriebenen Verfahren und seinen Erweiterungen können nur die in Kapitel 7.2.1 beschriebenen Klassen 1 und 2 abgedeckt werden. Für Geräte der Klasse 3, deren Gerätefunktionalität immer ein- und ausgeschaltet werden können muss, wird ein zuverlässiges und deterministisches Netzwerk benötigt, wenn keine geeigneten Rückfallmechanismen vorhanden sind, die ohne eine Gerätevernetzung auskommen. Solche Rückfallmechanismen, wie etwa ein fest verbundener Fußschalter, müssen entsprechend dem Risikomanagement in kurzer Zeit verfügbar sein. Dass dies in einem OP-Saal mit einer Vielzahl von Geräten, Schläuchen, Kabeln etc. auf engstem Raum mit den entsprechenden Hygienemaßnahmen realistisch ist, kann jedoch angezweifelt werden. Das heißt, dass das beschriebene Konzept derzeit noch nicht für hochkritische Anwendungen, z. B. die Auslösung von HF-Chirurgiegeräten, geeignet ist. Der Einsatz im Bereich der Auslösung von Saugern, Pumpen, Insufflatoren, Dokumentation etc. kann hingegen als realistisch und gewinnbringend für die Arbeitsabläufe im OP und damit auch für die Patientensicherheit angesehen werden. Es kann folglich ein großer Forschungsbedarf im Bereich der deterministischen, echtzeitfähigen Netzwerke für Medizingeräte auf Basis von Standardnetzwerktechnologien abgeleitet werden. Ist ein entsprechend zuverlässiges Netzwerk vorhanden, kann das beschriebene Konzept vereinfacht werden. Der Mechanismus des `STAND-BY`-Zustands zur *Jitter*-Erkennung ist dann nicht notwendig. Es muss lediglich der komplette Netzwerkausfall betrachtet werden, etwa durch ein herausgerissenes oder beschädigtes Kabel.

Der Bedarf für eine (harte) Echtzeitfähigkeit der *IEEE 11073 SDC*-Kommunikation wird ebenfalls in der Diskussion des Timeout-Wertes für die Funktionsauslösung (`InvocationEffectiveTimeout`) unterstrichen. Diese wurde bereits im Zuge der prototypischen Umsetzung in Kapitel 7.4 geführt. Ein vielversprechender Ansatz hierfür ist die Verbindung von *IEEE 11073 SDC* mit den *TSN*-Standards. Entsprechende Arbeiten erfolgten bereits unter der Beteiligung bzw. Betreuung des Autors dieser Arbeit [B 2], [E 2].

Eine Nutzerschnittstelle für die sichere Konfiguration von dynamisch assoziierbaren Fernsteuerungselementen und die Visualisierung der aktuellen Konfiguration ist nicht Teil der vorliegenden Arbeit. Intuitiv und im Sinne des Risikomanagements sicher benutzbare Lösungen zur Interaktion mit den Nutzerinnen sind unerlässlich für die Transformation vom starren und unflexiblen System in heutigen OP-Sälen, hin zu dynamisch vernetzten, herstellerübergreifenden Geräteensembles. Neben den technischen Aspekten, die in dieser Arbeit beschrieben wurden, ist die Benutzbarkeit ein wichtiges

Kriterium für die Zulassungsfähigkeit solcher Lösungen. Im Zuge des *OR.NET*-Projekts und seinen Nachfolgeprojekten hat sich die Arbeitsgruppe des Helmholtz-Instituts für Biomedizinische Technik der RWTH Aachen mit den Fragestellungen einer adäquaten Nutzerschnittstelle für solche Systeme beschäftigt [A 72, 73, 219, 220].

8 Dynamische Fernsteuerung basierend auf Service-Orchestrierung

In heutigen OP-Sälen sind bereits eine Vielzahl von potentiellen Fernsteuerungselementen wie Schalter an Handgriffen von Endoskopen oder Mikroskopen, Fußschalter etc. vorhanden. Diese sind aber in der Regel nur zur Steuerung des entsprechenden Gerätes selbst nutzbar. Das Konzept der *Service-Orchestrierung* bietet große Potentiale, diese und neue Fernsteuerungselemente effizient und kostengünstig in die herstellerübergreifende, *IEEE 11073 SDC*-basierte Medizingerätevernetzung zu integrieren. Die zu lösenden medizintechnischen Probleme und entwickelten Konzepte werden in diesem Kapitel beschrieben und diskutiert.

Dieses Kapitel beruht auf der Veröffentlichung „Dynamic Remote Control through Service Orchestration of Point-of-Care and Surgical Devices based on IEEE 11073 SDC“ [B 21]. Teile der Arbeit, insbesondere die Validierung des Prototypen, sind in Zusammenarbeit mit Malte Schmitz vom Institut für Softwaretechnik und Programmiersprachen (ISP) der Universität zu Lübeck, der Co-Autor der Veröffentlichung ist, entstanden. Die Publikation erreichte das Finale der „Best (PhD) Student Paper Competition“ im Zuge der „IEEE Healthcare Innovation Point-of-Care Technologies Conference (HI-POCT) 2016“.

8.1 Einleitung und Problembeschreibung

Fehlende Fernsteuerbarkeit von Medizingeräten ist eine große Herausforderung in heutigen OP-Sälen. Das Grundproblem besteht darin, dass eine Vielzahl der genutzten Geräte bzw. deren Bedienelemente vom chirurgischen Team nicht erreicht oder nicht steril bedient werden können. Dies führt dazu, dass weder die Chirurginnen, noch das steril assistierende Personal, Geräteparameter während der OP eigenständig verändern können. Ein alltägliches Beispiel zeigt Abbildung 8.1. Die chirurgischen Geräte sind im OP-Saal verteilt, etwa am linken Bildrand. Die operierenden Personen können bzw. dürfen diese genutzten Geräte nicht erreichen und somit die Parameter auch nicht eigenständig verstellen.

In der Einleitung dieser Arbeit wurden die resultierenden Problemstellungen fehlender herstellerübergreifender Fernsteuerungsmöglichkeiten in heutigen OP-Sälen erörtert (siehe Kapitel 1.1.1). Das chirurgische Team ist auf die Hilfe nicht-steriler Personen, meist des *Springers*, angewiesen. Verzögerungen im OP-Ablauf sind häufig die Folge. Kommunikationsprobleme [A 35, 36] und daraus resultierende Fehlverstellungen sind nicht nur ärgerlich, sondern können die Patientensicherheit gefährden. Beispiele sind das falsche Verfahren des OP-Tisches, wenn eine Änderung der Lagerung benötigt wird [C 2–4] oder das Verstellen von Parametern wie der maximalen Drehzahl von Fräsen oder der maximalen Leistungsabgabe von HF-Chirurgiegeräten und Ultraschalldissektoren [A 214]. Bei diesen Geräten können zu hohe Parametereinstellungen zu einer übermäßigen Schädigung von Gewebe führen, während zu geringe Einstellungen das Erzielen des gewünschten Effekts be- oder verhindern.

Schon in heutigen OP-Sälen sind diverse Fernsteuerungselemente im sterilen und für die Chirurginnen erreichbaren Bereich vorhanden, wie etwa Knöpfe am Handgriff von Endoskopen oder Mikroskopen, sterile Touchscreen etwa am Endoskop etc. Sie können aber lediglich zur Steuerung des Gerätes selbst oder für bestimmte Funktionalitäten innerhalb des proprietären Systems eines Herstellers genutzt werden. Das Potential für eine umfassende Fernsteuerung/Fernkonfiguration der während der Operation genutzten Medizingeräte ist also vorhanden und kann mit Hilfe von *IEEE 11073 SDC* genutzt werden.

Das Ziel des Kapitels ist die Vorstellung von Konzepten, die es ermöglichen, Fernsteuerungselemente so im Netzwerk zur Verfügung zu stellen, dass sie dynamisch konfiguriert und herstellerübergreifend anderen Medizingeräten zugeordnet werden können. Eine Grundvoraussetzung dafür, dass Hersteller ihre Fernsteuerungselemente umfassend innerhalb der Medizingerätevernetzung zur Verfügung stellen, ist ein einfach zu implementierendes Konzept zur Integration der Geräte. Daher steht die

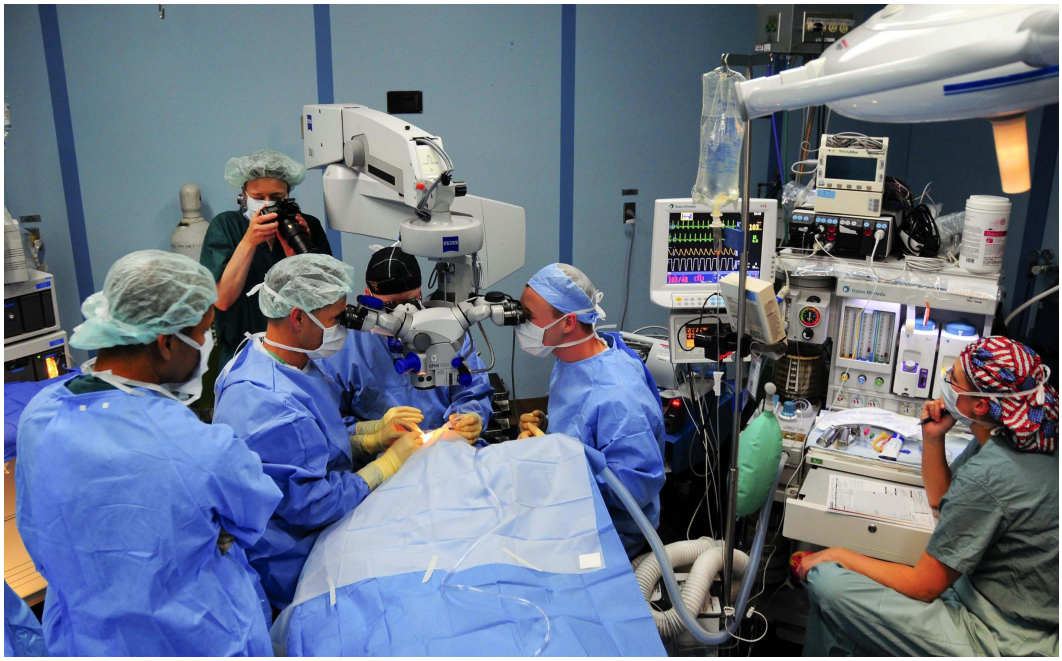

Abbildung 8.1: Alltag im OP-Saal: Chirurgische Medizingeräte sind teilweise schwer zugänglich und dürfen vom sterilen Team nicht betätigt werden. (Foto: Joshua Valcarcel [A 221].)

Einfachheit des Integrationskonzepts für die bereitstellenden Hersteller von Geräten die über Fernsteuerungselemente verfügen im Vordergrund.

Abgeleitet aus den Vorüberlegungen können die folgenden Anforderungen an das Konzept zusammengefasst werden:

- Ermöglichung der dynamischen Assoziierung zwischen Fernsteuerungselementen und anderen Medizingeräten mit Fernsteuerungsoperationen.
- Einfachheit der Umsetzung für die Bereitstellung der Schaltelemente innerhalb der Medizingerätevernetzung.

In Abgrenzung zu Kapitel 7 stehen in diesem Kapitel nicht die Konzepte für die Fernauslösung von Gerätefunktionalitäten im Fokus, sondern die Konfiguration von Parametern eines anderen Gerätes, die Fernkonfiguration. Eine Vielzahl von Konfigurationen können über *Activate Operations* modelliert und im Netzwerk bereitgestellt werden, z. B. das Erhöhen oder Verringern von Parametern, wie etwa die maximale Drehzahl einer Fräse, die maximal abgegebene Leistung eines HF-Chirurgiegerätes oder Ultraschalldissektors, die Intensität einer Lichtquelle etc. *Activate Operations* kommen in der Regel ohne zusätzliche Parameter aus, sodass sie intuitiv auf Schalter und Druckknöpfe abgebildet werden können, die einen binären Zustand haben: entweder gedrückt oder nicht gedrückt. Das Konzept ist aber nicht auf binäre Fernsteuerungselemente beschränkt. Es wird ebenfalls gezeigt, wie Fernsteuerungselemente mit einem größeren diskreten oder (quasi) kontinuierlichen Zustandsraum integriert werden können, wie etwa Fußschalter, die die Drucktiefe kontinuierlich abbilden, Drehknöpfe, Buttons oder Slider auf einem Touchscreen etc.

8.2 Begrifflichkeiten

Zur Beschreibung der einzelnen für das Konzept relevanten Vernetzungsteilnehmer existieren bisher keine feststehenden Begriffe. Für diese Arbeit sollen daher folgende Begriffe eingeführt werden:

Fernsteuerungselement Fernsteuerungselemente bezeichnen alle denkbaren Formen von Schaltern und anderen Interaktionselementen, wie beispielsweise Knöpfe am Handgriff von Endoskopen oder Mikroskopen, Pedale von Fußschaltern, Drehknöpfe, Buttons oder Slider auf einem Touchscreen etc.

Fernsteuerungsgerät Fernsteuerungsgeräte sind Medizingeräte, die Fernsteuerungselemente enthalten. Ist es für den Sachverhalt irrelevant, ob es sich um ein konkretes Fernsteuerungselement oder das zugehörige Fernsteuerungsgerät handelt, werden die Begriffe synonym verwendet.

Medizingerät mit Fernsteuerungsoperation, zu steuerndes/fernsteuerbares Gerät Oberbegriffe für alle Medizingeräte, die Fernsteuerungsoperationen per *IEEE 11073 SDC* zur Verfügung stellen. Für das vorgestellte Konzept handelt es sich hierbei in der Regel um *Activate Operations*, wie beispielsweise das Erhöhen oder Verringern von Parametern oder das Auslösen von Gerätefunktionalitäten, wie etwa die Aufnahme von Bildern zu Dokumentationszwecken, das Auslösen des Weißabgleichs einer Kamera etc.

Assoziierung Zuordnung eines Fernsteuerungselements zu einer Fernsteuerungsoperation eines zu steuernden Gerätes, mit dem Ziel, dass durch eine Betätigung des Fernsteuerungselements die Fernsteuerungsoperation am zu steuernden Gerät ausgelöst wird. Eine dynamische Assoziierung erfolgt im Zuge des medizinischen Vorgangs und kann sich potentiell im Verlauf ändern.

8.3 Konzepte zur dynamischen Fernsteuerung

Ein System zur dynamischen Assoziierung von Fernsteuerungselementen und Medizingeräten mit Fernsteuerungsoperationen kann mittels verschiedener Konzepte umgesetzt werden. Diese unterscheiden sich grundlegend darin, welche Vernetzungspartner welche der beiden logischen Rollen einnehmen, die in einer *SOMDA* spezifiziert sind: *Service Provider* und *Service Consumer*. Im folgenden Abschnitt werden daraus abgeleitete Prinzipien der Auslösung von Fernsteuerungsoperationen vorgestellt und diskutiert.

8.3.1 Grundkonzepte der Verteilung logischer SOMDA-Rollen

Es kann als gegeben vorausgesetzt werden, dass die zu steuernden Geräte die Rolle des *Service Providers* implementieren. Diese Rolle ist zunächst unabhängig von der Fernsteuerungsfunktionalität. Zur Erfüllung der medizinischen Aufgaben werden diese Geräte Informationen und Funktionalitäten im Netzwerk zur Verfügung stellen. Dies definiert die *Provider*-Rolle. Zur Fernsteuerung können die Geräte darüber hinaus Konfigurationsoperationen, wie *Activate Operations*, implementieren.

Für die Fernsteuerungsgeräte können zwei Fälle unterschieden werden: Im ersten Fall sind die Fernsteuerungselemente Teil eines Gesamtsystems, das einen grundlegenden medizinischen Zweck erfüllt. Zum Beispiel können Endoskope oder Mikroskope Knöpfe an ihren Handgriffen zur Steuerung anderer Vernetzungspartner anbieten. Somit haben diese Geräte ebenfalls in jedem Fall die Rolle eines *Service Providers*, um den medizinischen Zweck im Netzwerk abzubilden. Im zweiten Fall handelt es sich um ein reines Fernsteuerungsgerät ohne weitere Funktionalitäten, z. B. einen Fußschalter. Die Rolle ist hier zunächst nicht klar vorgegeben. Sie wird im Folgenden diskutiert.

Um einen Informationsaustausch zu ermöglichen, ist es nach dem Prinzip der *SOMDA* erforderlich, dass im System Vernetzungspartner existieren, die die Rolle des *Service Consumers* implementieren. Abbildung 8.2 veranschaulicht die verschiedenen Herangehensweisen, welcher Vernetzungspartner welche Rolle einnimmt.

Intuitiver Ansatz Das zu steuernde Gerät implementiert in jedem Fall die Rolle des *Service Providers* und bietet Operationen zur Fernkonfiguration an. Aufgrund dieser gegebenen Vorbedingung

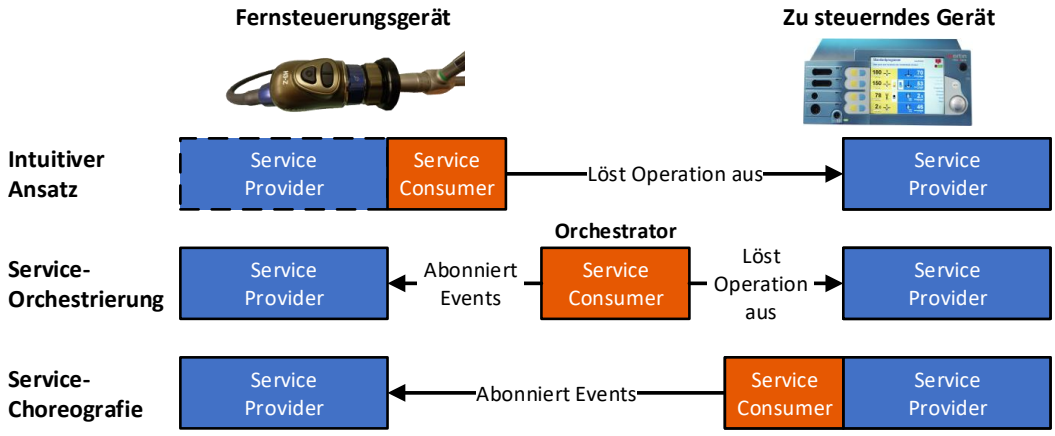

Abbildung 8.2: Grundkonzepte der *SOMDA*-Rollenverteilung für Fernsteuerungen mit dynamischer Assoziierung von Fernsteuerungsgeräten und zu steuernden Geräten.

besteht der *intuitive Ansatz* darin, dass die Fernsteuerungsgeräte die Rolle des *Service Consumers* implementieren. Wird das Steuerungselement betätigt, wird von diesem die assoziierte Fernsteuerungsoperation am zu steuernden Gerät ausgelöst. Im oberen Bereich von Abbildung 8.2 ist die Zuordnung der Rollen dargestellt. Der *Service Provider* ist beim Fernsteuerungsgerät gestrichelt dargestellt, da dieser Teil optional ist. Er wird nur benötigt, wenn die Fernsteuerungselemente Teil eines Medizingerätes sind, das sich unabhängig von diesem Konzept im Netzwerk repräsentiert.

Service-Orchestrierung Sowohl Fernsteuerungsgerät als auch zu steuerndes Gerät implementieren die Rolle des *Service Providers*. Das Fernsteuerungsgerät stellt seinen aktuellen Zustand mittels einer Metrik und zugehöriger Events im Netzwerk bereit. Im einfachsten Fall ist der Zustandsraum `gedrückt` / `nicht gedrückt`. Das zu steuernde Gerät stellt Operationen zur Fernkonfiguration bereit. Der *Service-Orchestrator* ist eine dritte Komponente, die die Rolle des *Service Consumers* einnimmt. Der *Orchestrator* abonniert die Zustandsänderungsevents des Fernsteuerungsgerätes und löst bei einem entsprechenden Zustand bzw. einer Zustandsänderung die Fernkonfigurationsoperation am zu steuernden Gerät aus. (Siehe auch mittlerer Bereich von Abbildung 8.2 und Kapitel 8.3.3, in dem das Konzept detailliert beleuchtet wird.)

Service-Choreografie Das Fernsteuerungsgerät implementiert die Rolle des *Service Providers* und stellt seinen aktuellen Betätigungszustand wie bei der *Service-Orchestrierung* im Netzwerk bereit. Das zu steuernde Gerät implementiert die Rolle des *Service Consumers* und abonniert die entsprechenden Events. Signalisiert das Fernsteuerungsgerät eine Betätigung, reagiert das zu steuernde Gerät, nach der Idee der *Service-Choreografie*, selbstständig auf die resultierende Zustandsänderung mit einer Verstellung des gewünschten Parameters. Das zu steuernde Gerät nutzt Services des Fernsteuerungsgerätes und entscheidet auf der Basis der Informationen, ob es eine Aktion ausführt oder nicht. Technisch gesehen steuert sich das Gerät damit selbst. Damit ist es nicht notwendig, dass das zu steuernde Gerät Fernsteuerungsoperationen in seiner Rolle als *Service Provider* anbietet. (Siehe auch unterer Bereich von Abbildung 8.2.)

8.3.2 Modellierung von Fernsteuerungselementen als Service Provider

In den Konzepten der *Service-Orchestrierung* und *-Choreografie* stellen sich die Fernsteuerungsgeräte als *Service Provider* im Netzwerk dar. Jeder *Service Provider* beschreibt seine Fähigkeiten und seinen Zustand mittels der Mechanismen der *IEEE 11073 SDC*-Normenfamilie. Die Beschreibung von Fernsteuerungselementen im Netzwerk beinhaltet zwei Bereiche, auf die im Folgenden eingegangen wird:

- Darstellung des Betätigungszustands: Zustand des Schaltelements selbst.

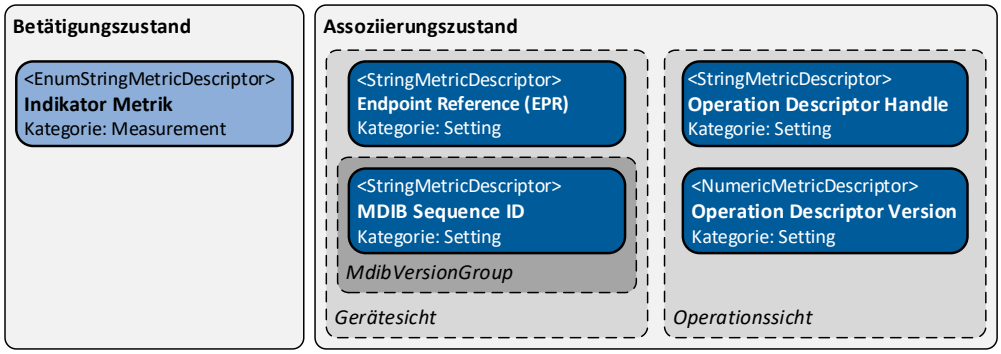

Abbildung 8.3: Modellierung von Fernsteuerungselementen durch Betätigungszustand und Assoziierungszustand (bestehend aus Geräte- und Operationsanteil).

- Darstellung des Assoziierungszustands: Zustand der Zuordnung des Fernsteuerungselementes zu steuerbaren Vernetzungspartnern.

Abbildung 8.3 veranschaulicht die Modellierung. Auf die Visualisierung von *Presetting*- und *Recommendation*-Metriken (siehe auch Kapitel 4.4.1.1) wird aus Gründen der Übersichtlichkeit verzichtet.

8.3.2.1 Modellierung des Betätigungszustands Zur Realisierung der Konzepte der *Service-Orchestrierung* und *-Choreografie* müssen alle Fernsteuerungsgeräte den Zustand ihrer Schaltelemente, z. B. gedrückt, nicht gedrückt, zu 75 % gedrückt etc., und die semantische Beschreibung im Netzwerk anbieten.

Für jedes Fernsteuerungselement wird eine *Indikator-Metrik* modelliert (siehe linker, hellblauer Teil von Abbildung 8.3). Ein Fußschalter mit drei Pedalen würde entsprechend drei *Indikator-Metriken* bereitstellen. Es gibt verschiedene Charakteristiken von Schaltelementen, die etwa binäre, diskrete oder kontinuierliche Zustände annehmen können. Die Art der Metrik wird durch diese Charakteristik bestimmt: Für binäre Schaltelemente kann eine zeichenkettenbasierte Metrik mit festdefinierten Werten (*EnumStringMetric*) für `gedrückt` und `nicht gedrückt` genutzt werden, während für diskrete und kontinuierliche Schaltelemente numerische Metriken genutzt werden. Der Wertebereich ist entsprechend den Fähigkeiten zu wählen. Die kontinuierliche Repräsentation erfolgt entsprechend lediglich „quasi kontinuierlich“, da eine Diskretisierung vorgenommen werden muss. Die *Indikator-Metrik* wird mit einem, noch nicht existierenden, *Coded Value* semantisch beschrieben. Dies erlaubt die sichere Interpretation des Zustands des Fernsteuerungselements durch die Vernetzungspartner. Zustandsänderungen können per *Get Service* oder per Abonnement von beteiligten *Service Consumern* empfangen werden.

Die genaue Ausgestaltung der zu nutzenden *Coded Values* für die semantische Beschreibung der verschiedenen *Indikator-Metriken* und der vordefinierten Werte für die zeichenkettenbasierte Modellierung müssen in entsprechenden Gerätespezialisierungen oder *PKP*-Standards definiert werden.

8.3.2.2 Modellierung von Assoziierungen Der aktuelle Zustand der Zuordnungen zwischen Fernsteuerungselement und auszulösender Operation muss zu jedem Zeitpunkt eindeutig beschrieben sein. Dies ist eine Grundvoraussetzung für ein zuverlässiges System mit dynamisch zur Laufzeit konfigurierbaren Assoziierungen. Ebenso müssen diese Informationen den Nutzerinnen und anderen Vernetzungsteilnehmern zur Verfügung gestellt werden [A 72]. Den Nutzerinnen können diese Informationen über geeignete grafische Schnittstellen bereitgestellt werden, zum Beispiel als Einblendung in das endoskopische oder mikroskopische Bild, als Teil einer zentralen Bedien- und Zustandskonsole etc. Zur Realisierung muss die Assoziierungsinformation im Netzwerk über *IEEE 11073 SDC* publiziert werden. Der rechte Teil von Abbildung 8.3 veranschaulicht die entsprechenden Aspekte.

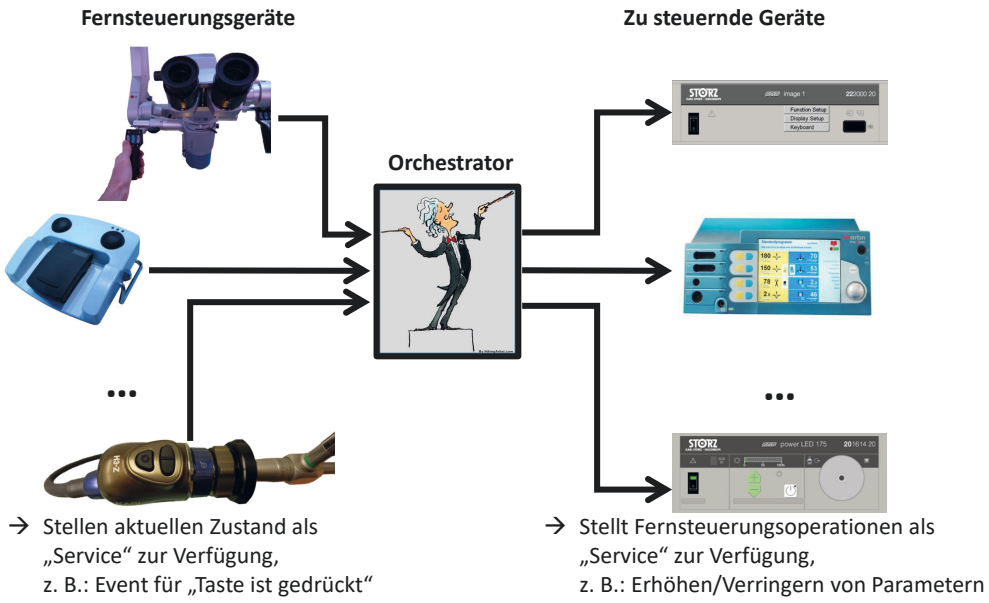

Abbildung 8.4: Veranschaulichung der *Service-Orchestrierung* – Links: Beliebige Anzahl von Fernsteuerungsgeräten. Mitte: *Orchestrator* zur Realisierung der assoziierten Verbindungen. Rechts: Beliebige Anzahl von zu steuernden Geräten.

Die Assoziierung eines Fernsteuerungselements lässt sich durch zwei Informationen eindeutig beschreiben: Den Vernetzungspartner und die Operation, die zugeordnet ist. Der Vernetzungspartner kann durch die *Endpoint Reference (EPR)* des Gerätes (*Hosting Service*) identifiziert werden[8.1]. Zur Identifikation der Operation dient der *Handle* des entsprechenden *Operation Descriptors*. *Handles* müssen per Definition eindeutig innerhalb eines Gerätes sein.

Um die Eindeutigkeit der Assoziierung über die Zeit sicherzustellen, kann zusätzlich die *Descriptor Version* der Operation und die *SequenceId* aus der *MDIB Version Group* des assoziierten Gerätes aufgenommen werden (siehe auch Abschnitt 4.4.1.3). Somit können Änderungen der Gerätebeschreibung des assoziierten Gerätes erkannt werden. Eine geänderte Gerätebeschreibung kann potentiell eine Änderung des *Handles* oder der Semantik der assoziierten Operation beinhalten, was zu einem Risiko für Patient und Personal führen kann. Über die Anforderungen des Standards *IEEE 11073-10207*, insbesondere des dortigen Kapitels 5.2.2, ist mit der *Descriptor Version* der Operation und der *SequenceId* der *MDIB Version Group* sichergestellt, dass solche Veränderungen erkannt werden. Wird eine potentielle Änderung im *Descriptor* erkannt, können entsprechende Risikovermeidungsmaßnahmen ergriffen werden, solange noch kein neuer konsistenter Zustand hergestellt wurde, wie etwa eine Zurückweisung von Fernsteuerungskommandos oder das Aufheben der Assoziierung.

8.3.3 Konzept der Service-Orchestrierung im Fokus

Für das Konzept der *Service-Orchestrierung* ist, wie bereits beschreiben, eine Komponente notwendig, die funktionell unabhängig von den Fernsteuerungsgeräten und zu steuernden Geräten an der Vernetzung teilnimmt. Im Kontext dieser Arbeit wird diese Komponente als *Service-Orchestrator* oder kurz *Orchestrator* bezeichnet. Die englische Bezeichnung, die in der entsprechenden Publikation [B 21] genutzt wird, lautet *Dynamic Orchestration Component (DOC)*. Abbildung 8.4 veranschaulicht das Funktionsprinzip.

8.1 Die *primäre UDI* muss derzeit für verschiedene Instanzen sogenannter *Software as a Medical Device* nicht zwingend eindeutig sein, sodass sie als Identifikator nur bedingt geeignet ist. Der Vorschlag der Nutzung der *EPR* des *Hosting Services* spiegelt den aktuellen Diskussionsstand in der Standardisierung wider. Ggf. wird es notwendig, ein Tupel aus *EPR* des *Hosted Services* und dessen *ServiceID* zu nutzen.

Der *Orchestrator* führt zunächst einen *Discovery-Prozess* im Netzwerk durch, um einerseits Fernsteuerungsgeräte zu finden, die entsprechende *Indikator-Metriken* (siehe Kapitel 8.3.2.1) anbieten und andererseits fernsteuerbare Medizingeräte, die entsprechende Operationen bereitstellen. Der *Orchestrator* abonniert die relevanten Zustandsänderungen der potentiell assoziierbaren Vernetzungsteilnehmer aus beiden Gerätegruppen.

Für den Vorgang der Assoziierung, also der Zuordnung eines Fernsteuerungselements zu einer zu steuernden Funktionalität eines Medizingerätes, stellt der *Orchestrator* eine geeignete Nutzerschnittstelle bereit. Diese kann beispielsweise Touchscreen-basiert für den sterilen Bereich im OP-Saal realisiert werden. Der *Orchestrator* lässt nur zulässige Assoziierungen durch den Benutzer zu. Die Zulässigkeit der Assoziierung erfolgt auf der Basis der Selbstbeschreibungen von *Indikator-Metrik* und Fernsteuerungsoperation. Die entsprechenden Typen müssen kompatibel sein, ebenso wie die Wertebereiche. So kann ein Fernsteuerungselement mit einer binären Repräsentation der *Indikator-Metrik* mit einer *Activate Operation* ohne Parameter assoziiert werden. Eine Assoziierung zu einer *Activate Operation* mit einem numerischen Parameter ist hingegen nicht möglich. Darüber hinaus sind ggf. weitere Aspekte zu überprüfen, beispielsweise: Passen die Zusicherungen des Fernsteuerungsgerätes zur Kritikalität der auszulösenden Operation? Passen die klinischen Zweckbestimmungen zueinander? Bestehen ausreichende Berechtigungen und befinden sich die beteiligten Geräte in derselben *Session* (siehe Kapitel 6)? Dies sollte soweit wie möglich bereits vor einer Nutzung überprüft werden, sodass den Nutzerinnen nur valide Kombinationen angeboten werden.

Auf der Basis der semantischen Beschreibungen durch Typen, Wertebereiche, Einheiten etc. kann der *Orchestrator* automatische Transformationen vornehmen. So ist etwa der Zustand einer *Indikator-Metrik* mit einem Wertebereich von 0 bis 255 (8-Bit-Wert), der beispielsweise die Betätigungstiefe eines Fußschalters modelliert, auf einen Parameter der Einheit Prozent und einem entsprechenden Wertebereich von 0 bis 100, abbildbar. Für solche automatischen Transformationen muss die Korrektheit sichergestellt und nachgewiesen werden.

Wurde die Assoziierung vorgenommen, so löst der *Orchestrator* bei entsprechenden Zuständen, bzw. Zustandsänderungen des Fernsteuerungselements, die zugehörige Operation aus. Wird beispielsweise ein Pedal eines Fußschalters gedrückt, wird die assoziierte Saugleistung einer Pumpe verändert. Das zu steuernde Gerät kann zusätzliche Sicherheitsanforderungen haben, die beispielsweise mittels des *MDPWS Safety Contexts* in Fernsteuerungskommandos einzubetten sind. In diesem Fall ist der *Orchestrator* dafür verantwortlich, diese Informationen korrekt einzuholen und bereitzustellen. Somit können auch Sicherheitskonzepte umgesetzt werden, wie sie in Kapitel 7 beschrieben werden. Der *Orchestrator* trägt damit auch die Verantwortung, nur Assoziierungen zuzulassen, bei denen das Fernsteuerungselement die Anforderungen des zu steuernden Gerätes erfüllt.

Die Assoziierung der Fernsteuerungselemente mit den vernetzten Medizingeräten wird, wie in Kapitel 8.3.2.2 beschreiben, im Netzwerk durch die Fernsteuerungsgeräte bekannt gemacht. Der *Orchestrator* aktualisiert die entsprechenden Metriken, wenn Assoziierungen verändert werden. Diese Informationen können und sollten den Akteuren im OP-Saal in geeigneter Form bekannt gemacht werden. Dies kann beispielsweise über das Nutzerinterface des *Orchestrators* erfolgen. Die im Netzwerk verfügbare Beschreibung des Assoziierungszustands erlaubt es darüber hinaus, dass eine Konfiguration auch von anderen Vernetzungsteilnehmern vorgenommen werden kann. Beispielsweise sind Assistenzsysteme vorstellbar, die auf der Basis einer Situations- bzw. Workflowerkennung zur Laufzeit neue Konfigurationen der Zuordnungen vornehmen, bzw. diese den Nutzern zur Bestätigung vorschlagen [B 9]. Für solche Vorschlagsszenarien werden die Metriken zur Beschreibung der Assoziierung um *Presetting*-Metriken und entsprechende Operationen zur Übernahme erweitert. Somit hat der *Orchestrator* Änderungen der Assoziierungsbeschreibung zu beachten und umzusetzen.

Im hier beschriebenen Konzept stellt der *Orchestrator* einen *Single Point of Failure (SPoF)* dar. In der Praxis sollte daher eine Realisierung genutzt werden, in der der *Orchestrator* redundant ausgelegt oder

Abbildung 8.5: Demonstrator auf dem *OR.NET*-Stand im Zuge der *conhIT*-Messe 2016 in Berlin.

als *verteilter Orchestrator* realisiert wird. Es könnten beispielsweise mehrere leistungsstarke Medizingeräte die *Orchestrator*-Funktionalität implementieren. Alle *Orchestrator*-Komponenten müssen in diesem Fall über das Assoziierungswissen verfügen. Mittels der in Kapitel 8.3.2.2 beschriebenen Mechanismen sind diese Informationen im Netzwerk bekannt. Unter den verfügbaren *Orchestratoren* ist eine verantwortliche Instanz zu wählen. Entsprechende Algorithmen zur *Leader Election* sind Stand der Technik [A 222] und wurden auch für Ad-hoc-Systeme weiterentwickelt [A 223]. Die praktische Umsetzung des verteilten Konzepts geht über den Umfang dieser Arbeit hinaus.

8.3.4 Demonstrator zur Service-Orchestrierung

Zur technischen Evaluation des Konzepts der *Service-Orchestrierung* wurden Demonstratoren mit echten Medizingeräten im Zuge des *OR.NET*-Projekts aufgebaut. Der Aufbau auf der *conhIT*-Messe 2016 in Berlin ist eines von mehreren Realisierungsbeispielen, das in Abbildung 8.5 zu sehen ist. Als Fernsteuerungsgeräte kamen zwei verschiedene Fußschalter von verschiedenen Herstellern mit jeweils drei Pedalen, und ein Kamerakopf eines endoskopischen Kamerasystems mit zwei Tasten zum Einsatz. Diese konnten mit mehr als 50 Fernsteuerungsoperationen von mehr als zehn fernsteuerbaren Medizingeräten von mehr als fünf verschiedenen Herstellern assoziiert werden. Einige Beispiele für genutzte Fernsteuerungsoperationen sind das Erhöhen und Verringern von Geräteparametern wie die Lichtintensität von endoskopischen Lichtquellen, die maximale Rotationsgeschwindigkeit einer Fräse, der Fluss und Druck von chirurgischen Pumpen und Insufflatoren; Parameteränderung eines HF-Chirurgiegerätes; Navigation durch Bilder bzw. Bildserien einer Software zur Betrachtung von *DICOM*-Bildern. Die Fernsteuerung erfolgte überwiegend herstellerübergreifend. So konnten beispielsweise Parameter von endoskopischen Geräten von Hersteller `A` mittels der Tasten am Endoskopkamerakopf von Hersteller `B` verstellt werden.

Für die *IEEE 11073 SDC*-Anbindung wurden drei verschiedene Open-Source-Bibliotheken genutzt: *openSDC* [A 161] (Java), *OSCLib* [A 164] (inzwischen *SDCLib/C*, *C++*) und *SoftICE* [A 162] (inzwischen *SDCLib/J*, *Java*). Die Geräteimplementierungen stammen dabei sowohl vom Autor dieser Arbeit als auch von Partnern im *OR.NET*-Projekt.

Der *Orchestrator* wurde vom Autor dieser Arbeit auf Basis der *openSDC*-Bibliothek prototypisch implementiert. Die Grundfunktionalität der Assoziierung zwischen den Fernsteuerungselementen und den Fernsteuerungsoperationen der zu steuernden Geräte wird vom *Orchestrator* umgesetzt. Zusätzlich stellt er eine prototypische grafische Benutzeroberfläche (*GUI*) zur Konfiguration und Anzeige

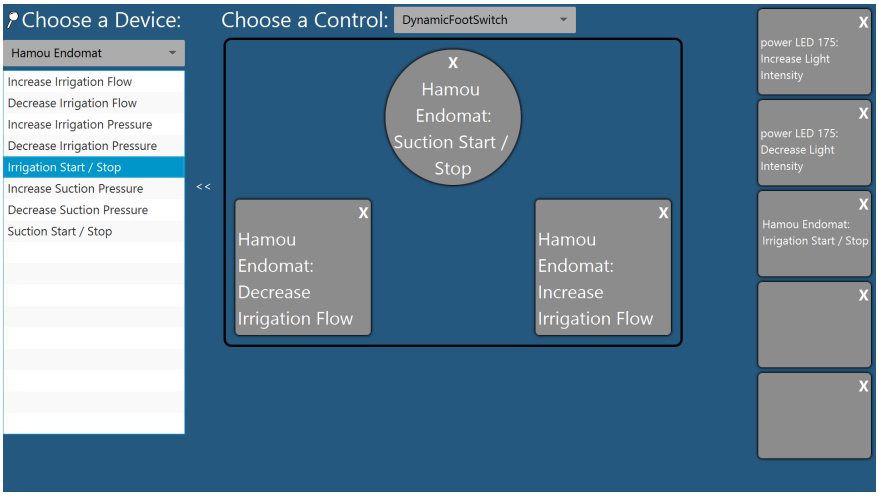

Abbildung 8.6: Prototyp eines Touchscreen-basierten *Graphical User Interface (GUI)* zur dynamischen Assoziierung per *Drag-and-Drop* mit rudimentärer grafischer Darstellung des Fernsteuerungsgerätes. (Abbildung nach *Fig. 2* aus [B 21]. Original © 2016 IEEE.)

der Assoziierungen bereit. Die Touchscreen-basierte Realisierung ist in Abbildung 8.6 dargestellt. Per *Drag-and-Drop* können die Fernsteuerungsoperationen der verfügbaren Geräte (linker Teil der *GUI*) den Fernsteuerungselementen der verfügbaren Fernsteuerungsgeräte (mittlerer Teil der *GUI*) zugeordnet werden.

Der Prototyp verfügt nur über eine rudimentäre visuelle Darstellung des Fernsteuerungsgerätes. In Abbildung 8.6 ist ein Fußschalter mit drei Pedalen dargestellt: links und rechts Pedale mit einer rechteckigen Form und mittig, nach oben versetzt, mit einer runden Form. Die Darstellung wird automatisch aus einer stark vereinfachten Selbstbeschreibung des Fernsteuerungsgerätes generiert. Der Umfang der Beschreibungsmöglichkeit ist auf die Form (rund, eckig) und die Lage (Rasterkoordinaten) beschränkt. Eine effiziente Selbstbeschreibung der baulichen und visuellen Eigenschaften der Schaltgeräte ist nicht Teil dieser Arbeit und muss im Rahmen eigener, zukünftiger Forschungsarbeiten entwickelt werden. Die Notwendigkeit ist darin begründet, dass der *Orchestrator* unabhängig von den zu vernetzenden Kommunikationspartnern entwickelt wird, sodass die Visualisierung zur Laufzeit generiert werden muss.

Die Korrektheit der dynamischen *Service-Orchestrierung* wurde im Demonstrator mit Hilfe des sogenannten *OSCP Swiss Army Knife* [A 224] von Projektpartnern des ISPs der Universität zu Lübeck gezeigt. Die spezifizierte Verhaltensweise wurde mittels *temporaler Logik* beschrieben. Auf dieser Basis konnte das korrekte Verhalten zur Laufzeit vom *OSCP Swiss Army Knife* durch entsprechend generierte Monitore nachgewiesen werden.

8.4 Technische Evaluation

Nach dem Konzept der *Service-Orchestrierung* erfolgt für jede Fernauslösung einer Operation eine *Zwei-Hop-Verbindung*[8.2]: Fernsteuerungsgerät → *Orchestrator* → zu steuerndes Gerät. Die beiden anderen Konzepte nutzen eine direkte *Ein-Hop-Verbindung*. Abbildung 8.8 veranschaulicht die Kommunikation im Vergleich zwischen dem *intuitiven Ansatz* und der *Service-Orchestrierung*.

8.2 Unter der Anzahl von *Hops* einer Kommunikationsverbindung versteht man die Anzahl der Teilverbindungen, die bestehen, um die Übertragung vom Sender zum Empfänger zu realisieren. Sind Sender und Empfänger direkt miteinander verbunden, spricht man von einer *Ein-Hop-Verbindung*. Erfolgt die Kommunikation über genau einen Zwischenknoten, handelt es sich um einer *Zwei-Hop-Verbindung*, usw. Mathematisch ausgedrückt ist die *Hop-Anzahl* gleich der Anzahl der Kanten des genutzten (Kommunikations-)Pfades zwischen Sender und Empfänger.

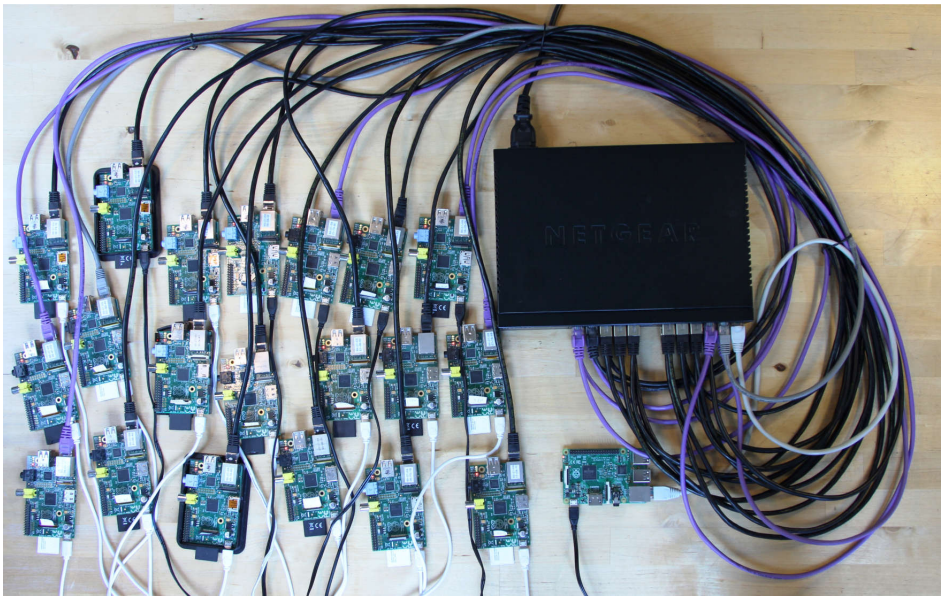

Abbildung 8.7: Testaufbau zur Evaluation des Konzepts der *Service-Orchestrierung*.

Aufgrund des höheren Kommunikationsaufwands im Konzept der *Service-Orchestrierung* muss die Performanz des Systems betrachtet werden. Über die Performanzevaluation eines einzelnen Paares aus Fernsteuerungsgerät und gesteuertem Gerät hinaus muss die Skalierbarkeit des Konzepts für die Verbindung mehrerer Paare über einen Orchestrator evaluiert werden. Da der *Orchestrator* eine zentrale Komponente darstellt, kann dieser einen „Flaschenhals“ darstellen. Daher ist für dieses Konzept die Skalierbarkeit besonders zu untersuchen.

Für die Evaluation werden der *intuitive Ansatz* und das Konzept der *Service-Orchestrierung* verglichen. Da das Konzept der *Service-Choreografie* technisch äquivalent zum *intuitiven Ansatz* ist, wird dieses nicht gesondert betrachtet.

8.4.1 Testumgebung

Für die technische Evaluation wird eine Testumgebung mit verschiedenen Teilnehmerzahlen genutzt. Abbildung 8.7 zeigt den Versuchsaufbau. Alle Vernetzungsteilnehmer werden auf der Basis der Bibliothek *SDCLib/C* [A 164] in der Version `4.4.3`[8.3] implementiert. Zur Ausführung werden verschiedene Hardware-Plattform aus der *Raspberry Pi (RPi)*-Einplatinenrechner-Familie genutzt. Tabelle 8.1 stellt diese zusammen mit ihren Spezifikationen dar. Für alle Netzwerkteilnehmer kommt das gleiche Image mit dem Betriebssystem *Raspbian Buster Lite* (Linux Kernel *4.19.97*) zum Einsatz. Es wird der *GCC*-Compiler in Version *8.3.0-6* genutzt. Die Netzwerkinfrastruktur realisiert ein *NETGEAR GS324* Gigabit-Ethernet (unmanaged) Switch mit 24 Ports.

8.4.2 Messmethodik

Für die Evaluation wird die Kommunikationsumlaufzeit (*Round Trip Time, RTT*) gemessen. Im Folgenden werden die Messpunkte und der Ablauf des Messvorgangs, der sich aus Einzelmessungen zusammensetzt, beschrieben.

8.4.2.1 Messpunkte Die konkreten Zeitpunkte, an denen Startzeitstempel t_s und Endzeitstempel t_e zur Berechnung der *RTT* genommen werden, unterscheiden sich zwischen dem *intuitiven*

[8.3] Master Branch; Commit 48cfa93f83fea35fddb664386ea47846d6830656; als Open Source verfügbar unter `https://github.com/surgitaix/sdclib/commit/48cfa93f83fea35fddb664386ea47846d6830656`.

Tabelle 8.1: Überblick über die genutzten Hardware-Plattformen und ihre Spezifikationen. *64 MByte reserviert für den Grafikprozessor (GPU); **Wird aufgrund des Betriebssystem-Kernels im *32-Bit ARMv7* Modus betrieben.

Plattform	**Architektur**	**CPU**	**Taktrate** [MHz]	**Kerne**	**RAM** [MByte]
Raspberry Pi 1 Modell B (*RPi 1*)	32-Bit ARMv6	ARM1176JZF-S	700	1	512*
Raspberry Pi 2 Modell B V1.1 (*RPi 2*)	32-Bit ARMv7	ARM Cortex-A7	900	4	1024*
Raspberry Pi 3 Modell B+ (*RPi 3B+*)	64-Bit ARMv8**	ARM Cortex-A53	1400		
Raspberry Pi 4 Modell B V1.2 (*RPi 4*)		ARM Cortex-A72	1500		4096*

und dem *Orchestrierungsansatz* konzeptbedingt (siehe auch Abbildung 8.8). Der Startzeitstempel t_s wird jeweils im simulierten Fernsteuerungsgerät genommen. Beim *intuitiven* Ansatz erfolgt dies direkt bevor im *User-Code* die *API*-Funktion der *SDCLib/C*-Bibliothek für das Auslösen der *Activate Operation* des simulierten zu steuernden Gerätes aufgerufen wird. Für diese Evaluation führt die *Activate Operation* am zu steuernden Gerät nach deren Verarbeitung zur Erhöhung eines Wertes einer Metrik. Damit ist die eigentliche Funktionalität des Fernsteuerns eines entfernten Gerätes abgeschlossen. Damit die *RTT* gemessen werden kann, abonniert das Fernsteuerungsgerät zusätzlich die Metrikänderungen des gesteuerten Gerätes. Somit erhält das Fernsteuerungsgerät eine entsprechende Benachrichtigung. Wenn diese Benachrichtigung verarbeitet wurde und die Information im *User Code* zur Verfügung steht, wird der Endzeitstempel t_e genommen. Damit ist in der *RTT* mehr Verarbeitungszeit enthalten als für den eigentlichen Vorgang der Fernsteuerung nötig ist. Dies ist in der Bewertung der Ergebnisse zu berücksichtigen. Der Vorteil der *RTT* liegt darin, dass keine Zeitsynchronisierung zwischen den Netzwerkteilnehmern vorgenommen werden muss, da die beiden Zeitstempel auf demselben Geräte genommen werden. Es sei zusätzlich auf die Argumentation zur *RTT* in Abschnitt 5.4.1 verwiesen.

Wird die Fernsteuerung mittels *Service-Orchestrierung* durchgeführt, so wird der erste Zeitstempel t_s genommen, wenn die Simulation des Fernsteuerungsgerätes die Betätigung des Fernsteuerungselements signalisiert. Dies erfolgt, wie beschrieben, durch die Zustandsänderung einer entsprechenden Metrik. Der Orchestrator empfängt nach dem Abonnementprinzip diese Zustandsänderung und löst die zugeordnete *Activate Operation* am zu steuernden Gerät aus. Für die Evaluation bewirkt dies die Änderung des Wertes einer Metrik. Die eigentliche Fernsteuerungsfunktionalität ist damit vollzogen. Äquivalent zum *intuitiven Ansatz* abonniert das Fernsteuerungsgerät die Änderungen dieser Metrik. Nachdem das entsprechende Event verarbeitet wurde, wird am Fernsteuerungsgerät der zweite Zeitstempel t_e genommen.

Für die Auflösung und technische Realisierung der Messpunkte sei auf den Abschnitt 5.4.3 dieser Arbeit verwiesen. Die Informationen sind auf diese Evaluation übertragbar, da dieselben Mechanismen genutzt werden.

8.4.2.2 Messvorgang Jede Messreihe zwischen einem Paar aus einem Fernsteuerungsgerät und einem gesteuerten Gerät besteht aus 1.000 *RTT*-Messungen von Fernsteuerungsvorgängen. Für die allgemeine Performanzevaluation in Abschnitt 5 dieser Arbeit wurden je 40.000 Einzelmessungen vorgenommen. Die Untersuchungen in Kapitel 5.5.5 haben gezeigt, dass bei der Nutzung der *C++* Implementierung eine deutlich geringere Anzahl von Einzelmessungen ausreichend ist.

Vor jeder Messreihe werden alle Teilnehmer neu gestartet. Dies schließt eine Beeinflussung durch vorangegangene Messreihen aus. So könnten etwa Artefakte im *Discovery-Prozess* oder Mechanismen der Verbindungswiederherstellung durch die Software-Bibliothek zu verzerrendem Verhalten führen. Die Messreihen werden in einem separierten Netzwerk vorgenommen, wie bereits für die Messungen in Abschnitt 5 beschrieben wurde. Das bedeutet, dass der Kontrollrechner zum Starten

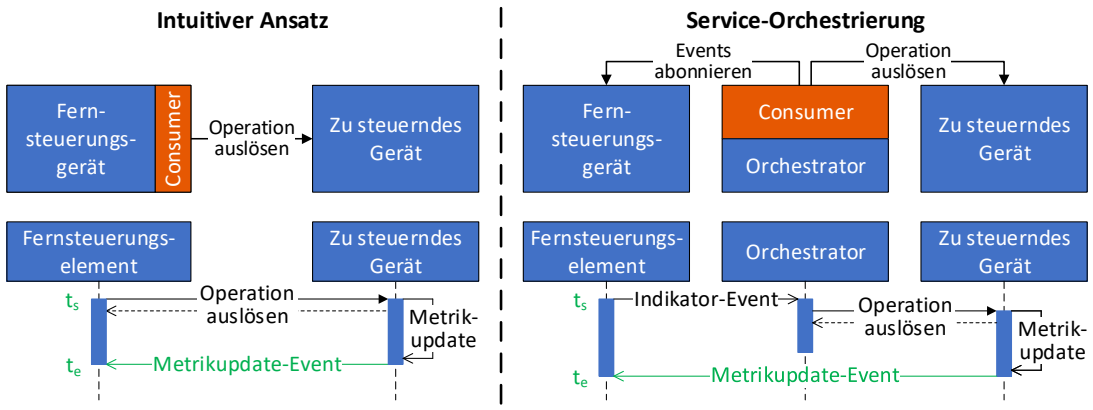

Abbildung 8.8: Vergleich der Konzepte und (vereinfachten) Kommunikationsabläufe im Evaluationssetup. (Abbildung nach *Fig. 1* aus [B 21]. Original © 2016 IEEE.)

der Messreihe und Einsammeln der Messergebnisse aus dem Netzwerk entfernt wird. Die Implementierungen der *IEEE 11073 SDC*-Netzwerkschnittstellen laufen autark in eigenen *Screen Sessions*.

Ebenfalls analog zu den Messungen in Abschnitt 5 wird eine Inter-Messverzögerung von 20 ms genutzt. Um ein realistisches Verhalten für die Kommunikation von mehreren verschiedenen, voneinander unabhängig kommunizierenden Paaren von Fernsteuerungsgeräten und gesteuerten Geräten zu realisieren, wird zusätzlich eine zufällige Verzögerung zwischen 0 ms und 5 ms addiert. Für den Initialisierungswert (*Seed*) des Zufallszahlengenerators werden die letzten Ziffern der *Endpoint Reference (EPR)* genutzt, sodass die Verzögerung nicht bei allen Kommunikationspaaren gleich, aber reproduzierbar ist. Der eigentliche Messvorgang beginnt, indem der *Orchestrator* an allen beteiligten Fernsteuerungsgeräten eine entsprechende Operation als Startsignal auslöst. Es ist davon auszugehen, dass der Auslieferungs- und Verarbeitungsprozess nicht vollständig deterministisch erfolgt. Daher ist die Reproduzierbarkeit des Timings bei einer Versuchswiederholung eingeschränkt.

Laufzeitevaluation Zur Evaluation des Laufzeitverhaltens werden der *intuitive* und der *Orchestrierungsansatz* mittels eines minimalen Testaufbaus verglichen. Für den *intuitiven* Ansatz sind dies zwei Teilnehmer, die ein Paar aus Fernsteuerungsgerät und zu steuerndem Gerät bilden. Zur Untersuchung des Konzepts der *Service-Orchestrierung* kommt als dritter Teilnehmer ein *Orchestrator* hinzu. In allen Testabläufen werden das Fernsteuerungsgerät und zu steuernde Gerät auf *Raspberry Pi (RPi) 1*-Plattformen realisiert. Für den *Orchestrator* kommen in verschiedenen Testläufen ein *RPi 1*, *2*, *3B+* und *4* zum Einsatz. Die Leistungsfähigkeit des *Orchestrators* steigt folglich an.

Skalierbarkeit Die Skalierbarkeit des Konzepts der *Service-Orchestrierung* wird mit einer Testumgebung mit einer ansteigenden Zahl an Paaren aus *Fernsteuerungsgerät* und *zu steuerndem Gerät* evaluiert. Abbildung 8.9 zeigt den Aufbau schematisch. Es werden mehrere Messreihen durchgeführt. Zum einen wird die Anzahl der Fernsteuerungspaare von eins bis zehn erhöht, bei gleichbleibender Hardwareplattform des *Orchestrators*. Zum anderen werden diese jeweils zehn Messreihen mit ansteigender Leistungsfähigkeit des *Orchestrators* durchgeführt: *RPi 1*, *2*, *3B+* und *4*. Für jedes Teilexperiment bleibt die Hardwareplattform von Fernsteuerungsgeräten und zu steuernden Geräten unverändert der *RPi 1*.

Die Assoziierung zwischen Fernsteuerungsgerät und zu steuerndem Gerät durch den *Orchestrator* erfolgt durch eine vorgegebene Zuordnung auf der Basis der Teilnehmer-*EPRs*. Dies ermöglicht eine Testautomatisierung. Die Assoziierung muss so nicht vor jedem Testlauf neu, über eine *GUI* oder ähnliches, vorgenommen werden. Da alle Teilnehmer die gleiche Hardwareplattform und das gleiche Testprogramm mit demselben Quelltext nutzen, stellt dies keine Einschränkung der Messergebnisse dar.

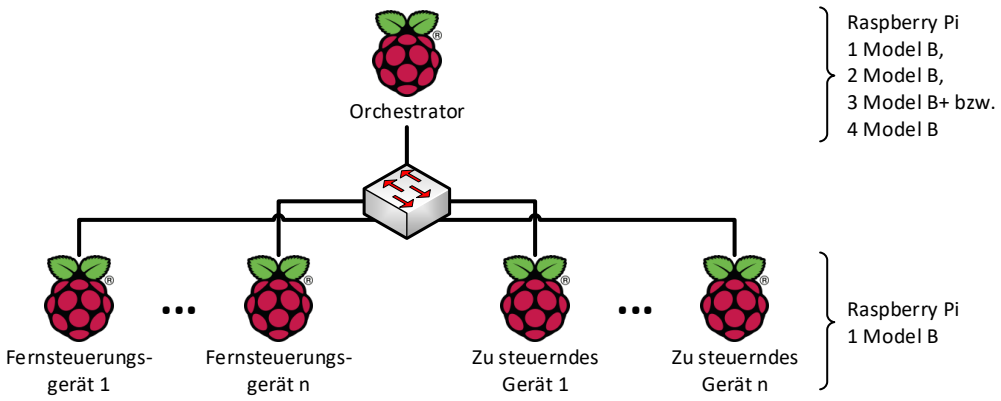

Abbildung 8.9: Schematische Darstellung des Testaufbaus zur Skalierbarkeitsevaluation des Konzepts der *Service-Orchestrierung*.

8.4.3 Nachverfolgbarkeit der Ergebnisse

Die genutzten Implementierungen der Simulatoren von *Fernsteuerungsgerät* und *gesteuertem Gerät* sowie des *Orchestrators* stehen in einem öffentlichen Repository des Instituts für Angewandte Mikroelektronik und Datentechnik (IMDs) der Universität Rostock zur Verfügung[8.4]. Zusammen mit den ebenfalls dort veröffentlichten Rohdaten der Messungen ermöglicht dies die Nachverfolgbarkeit und Reproduzierbarkeit der Ergebnisse.

8.4.4 Ergebnisse und Diskussion

Im Folgenden werden die Messergebnisse vorgestellt, diskutiert und Einschränkungen besprochen. Die Laufzeitevaluation beruht auf vier einzelnen Messreihen mit jeweils 1.000 Messwerten und einer Referenzmessung des *intuitiven Ansatzes* mit ebenfalls 1.000 Messwerten. Für die Skalierbarkeitsevaluation wurden jeweils zehn Messreihen mit ansteigender Anzahl an Fernsteuerungspaaren, mit vier verschiedenen Hardware-Plattformen des *Orchestrators* aufgenommen, sodass in Summe 220.000 Messwerte einfließen.

8.4.4.1 Laufzeitevaluation Die *Zwei-Hop-Verbindung* beim Konzept der *Service-Orchestrierung* ist aufwändiger als die *Ein-Hop-Verbindung* ohne *Orchestrator* des *intuitive Ansatzes*, der somit die Referenzmessung darstellt. Daher sind bei der *Service-Orchestrierung* höhere Laufzeiten zu erwarten. Abbildung 8.10 und Tabelle 8.2 zeigen die Vergleichsmessungen zwischen beiden Konzepten.

Bei den folgend präsentierten und diskutierten Messwerten handelt es sich jeweils um den *RTT*-Median der entsprechenden Messreihe. Der Median-Laufzeit-Referenzwert des *intuitiven Ansatzes* liegt bei 69 ms. Es ist zu erkennen, dass die Laufzeit mit *Service-Orchestrierung* ansteigt. Mit Laufzeit-Medianen zwischen 85,9 ms (*RPi 1*) und 71,0 ms (*RPi 4*) liegt der Overhead zwischen 24,5 % und 2,9 %. Wie zu erwarten nimmt der zeitliche Mehraufwand mit steigender Leistungsfähigkeit der Hardware des *Orchestrators* ab.

Im Messaufbau, in dem alle Komponenten auf der gleichen Hardwareplattform realisiert sind (*RPi 1*), zeigt sich eine *RTT*-Erhöhung von rund 25 %. Betrachtet man die ausgetauschten Nachrichten, inkl. der Events (`OperationInvokedReport`, `EpisodicMetricReport`), so zeigt sich, dass diese Erhöhung etwa dem Mehraufwand an Nachrichten entspricht: Zwischen Beginn und Ende der Messung werden beim *intuitiven Ansatz* vier Nachrichten und bei der *Service-Orchestrierung* fünf Nachrichten ausgetauscht. (Siehe auch Abbildung G.1 in Anhang G.)

[8.4] Link: `https://gitlab.amd.e-technik.uni-rostock.de/imd-ornet-sdc/dynorch-eval`.

Tabelle 8.2: Messergebnisse für den *intuitiven Ansatz* ohne *Orchestrator* und das Konzept der *Service-Orchestrierung* mit ansteigender Leistungsfähigkeit der Hardwareplattform des *Orchestrators*.

Konzept	**Orchestrator**	**Median [ms]**	**Overhead [%]**
intuitiver Ansatz	ohne Orchestrator	69,0	
Service-Orchestrierung	RPi 1	85,9	24,5
	RPi 2	76,4	10,8
	RPi 3B+	73,1	6,0
	RPi 4	71,0	2,9

Der *Orchestrator* ist eine zentrale Komponente im Vernetzungssystem. Daher kann davon ausgegangen werden, dass Ressourcenbeschränkungen wie etwa eine geringe Leistungsaufnahme nicht relevant sind. Somit sind für eine Bewertung der Messergebnisse in Bezug auf eine Übertragung in reale Anwendungsfälle eher die leistungsstärkeren Plattformen heranzuziehen. Für *RPi 3* und *RPi 4* liegt der Overhead bei unter 6,5 % bzw. 3 %. Vergleicht man den beschriebenen potentiellen klinischen Mehrwert der *Service-Orchestrierung* mit dem Laufzeitzuwachs, so überwiegen nach Meinung des Autors die Vorteile des neuen Konzepts.

Die in dieser Evaluation gemessenen absoluten Werte der Laufzeiten können, wie bereits in Abschnitt 5 dieser Arbeit diskutiert, nur im Zusammenhang mit konkreten Anwendungsfällen bewertet werden. Es ist jedoch zu erkennen, dass ein Großteil der in Tabelle 5.1 gesammelten Anforderungen erfüllt werden.

Vergleicht man die Laufzeitmessungen dieser Evaluation mit den Messungen aus Kapitel 5, so ist zu erkennen, dass die absoluten Werte dieser Laufzeitmessungen deutlich höher sind. Die Laufzeitmessung des *intuitiven Ansatzes* kann als technisch nahezu äquivalent zu den Messungen aus Abschnitt 5 mit der *OSCLib* auf der *RPi 1*-Plattform angesehen werden. In Tabelle 5.3 in Abschnitt 5.5.2 liegt der Median der *RTT*-Werte für eine vergleichbare Messung bei 23,5 ms. Verglichen mit dem Median dieser Evaluation des *intuitiven Ansatzes* von 69,0 ms ist dieser um etwa den Faktor 2,9 höher. Ebenso ist im Vergleich eine deutlich höhere Streuung zu erkennen. Dies zeigt sich etwa im Vergleich der *Tukey Boxplots* in Abbildung 5.3 und Abbildung 8.10. Ebenso kann der Vergleich von Median und Maximalwert herangezogen werden:

- **Vergleichsmessung**: Median: 23,5 ms; Maximalwert: 25,3 ms; Faktor: ≈1,08 (Abschnitt 5.5.2)
- ***intuitiver Ansatz***: Median: 69,0 ms; Maximalwert: 124,9 ms; Faktor: ≈1,81

Eine Validierung der Ergebnisse mit eingefügten Messpunkten in den zur *SDCLib/C* gehörigen Beispiel-Applikationen zeigt konsistente Laufzeitmessungen zu den hier präsentierten Ergebnissen. Daher ist davon auszugehen, dass die höheren Laufzeiten durch die Bibliothek verursacht werden und nicht mit mangelnder Qualität bzw. Performanz der vom Autor dieser Arbeit implementierten Messapplikationen zu begründen sind.

Über mögliche Ursachen wurde mit den Entwicklern der *SDCLib/C*-Bibliothek Rücksprache gehalten. Als wahrscheinlichste Ursachen für die Performanzeinbußen sind nach Auskunft der Experten die erhöhten Sicherheits- und Stabilitätsmechanismen zu nennen. So wurde seit dem Softwarestand der *OSCLib*, die in Kapitel 5 genutzt wird, und der Version der *SDCLib/C*, die in dieser Evaluation genutzt wird, die *Thread Safety* maßgeblich erhöht. Solche stabilitätserhöhenden Maßnahmen verringern aber typischerweise die Performanz. Dabei ist zu beachten, dass Synchronisationsmechanismen auf verschiedenen Hardwarearchitekturen unterschiedlich performant sind. So legt Herb Sutter in seinem Vortrag über *Atomics* in *C++11* [A 225] im Zuge der Konferenz „*C++ and Beyond 2012*“ dar, dass diese Mechanismen auf *ARMv8*-Architekturen deutlich effizienter sind. Hierbei ist zu beachten, dass im *RPi 3* und *RPi 4* zwar *ARMv8*-Prozessoren verbaut sind, diese aber

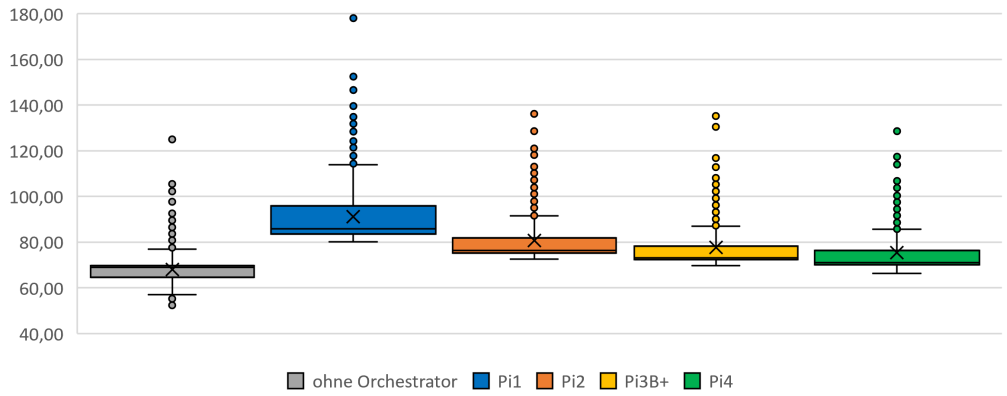

Abbildung 8.10: Messergebnisse im Vergleich zwischen dem *intuitiven Ansatz* ohne *Orchestrator* und dem Konzept der *Service-Orchestrierung* mit ansteigender Leistungsfähigkeit der Hardwareplattform des *Orchestrators*.

aufgrund des Betriebssystem-Kernels im *32-bit ARMv7*-Modus betrieben werden. Ungeachtet der absoluten Laufzeitmesswerte bleibt aber die Aussagekraft der relativen Verhältnisse zwischen *intuitivem Ansatz* und dem Konzept der *Service-Orchestrierung* erhalten.

Für die Einordnung der beschriebenen relativen *Overheads* sollte Folgendes beachtet werden: Die Messmethodik beinhaltete die Laufzeit vom Senden der *Trigger*-Nachricht durch das *Fernsteuerungsgerät* über die Reaktion des *gesteuerten Gerätes*, bis das Ergebnis der Fernsteuerung über *IEEE 11073 SDC* kommuniziert beim *Fernsteuerungsgerät* angekommen und verarbeitet ist. Soll nur der Kern der Fernsteuerungsaktivität betrachtet werden, so wird der *Overhead* durch diese Messmethodik geringfügig unterschätzt. Dies ist darin begründet, dass die eigentliche Fernsteuerung mittels der beschriebenen *Zwei-Hop-Verbindung* über den *Orchestrator* erfolgt. Die Übermittlung des Ergebnisses per Abonnementprinzip erfolgt aber per direkter *Ein-Hop-Verbindung*, wie in Abbildung 8.8 illustriert wird. Soll hingegen auch die Propagierung des Fernsteuerungsergebnisses im Netzwerk mitbetrachtet werden, so spiegelt der angegebene *Overhead* diesen Aspekt komplett wider.

8.4.4.2 Skalierbarkeit Auf der Basis der heutigen Gerätelandschaften ist davon auszugehen, dass die mittels *IEEE 11073 SDC* direkt zum Zweck der Fernsteuerung vernetzten Geräte kurz- bzw. mittelfristig eine geringe zweistellige Anzahl nicht übersteigen werden. Die Maximalgröße des Testaufbaus mit zehn Paaren von *Fernsteuerungsgerät* und *gesteuertem Gerät* und einem *Orchestrator* stellt damit ein realistisches Szenario dar.

Abbildung 8.11 und Tabelle 8.3 geben einen Überblick über die Laufzeitentwicklung des Konzepts der *Service-Orchestrierung* mit steigender Anzahl an Fernsteuerungspaaren. Zusätzlich stellt Abbildung G.2 in Anhang G die Messergebnisse ohne die Messungen mit einer *Orchestrator*-Realisierung mittels *RPi 1* dar. Dies erlaubt eine bessere Darstellung der Messwerte für *RPi 2, 3B+, und 4*.

Die Fernsteuerungsteilnehmer nutzen in allen Messreihen dieselbe Hardware: *RPi 1*. Es ist ein klarer Unterschied zwischen den Messreihen mit einem *RPi 1* als *Orchestrator*-Hardwareplattform und den leistungsfähigeren Realisierungen zu erkennen. Im Fall des *RPi 1-Orchestrators* ist bei einer Steigerung der Fernsteuerungspaare ein Anstieg der Laufzeiten zu verzeichnen, der als *linear* beschreiben werden kann. Im Gegensatz dazu zeigen die Messreihen mit einem *Orchestrator* auf der Basis eines *RPi 3* bzw. *RPi 4* eine als konstant beschreibbare Laufzeitentwicklung. So liegen etwa die Messwerte für den *RPi 4* im Intervall $[70,5\,ms; 71,2\,ms]$ bei einem nicht-monotonen Verlauf.

Für die Messreihen mit einer *Orchestrator*-Realisierung mittels *RPi 2* gilt keine der beiden Analysen uneingeschränkt. Für ein bis sieben Fernsteuerungspaare zeigt sich eine konstante Laufzeitentwicklung. Ab sieben Fernsteuerungspaaren ist ein linearer Anstieg zu erkennen (siehe auch Abbildung G.2 in Anhang G).

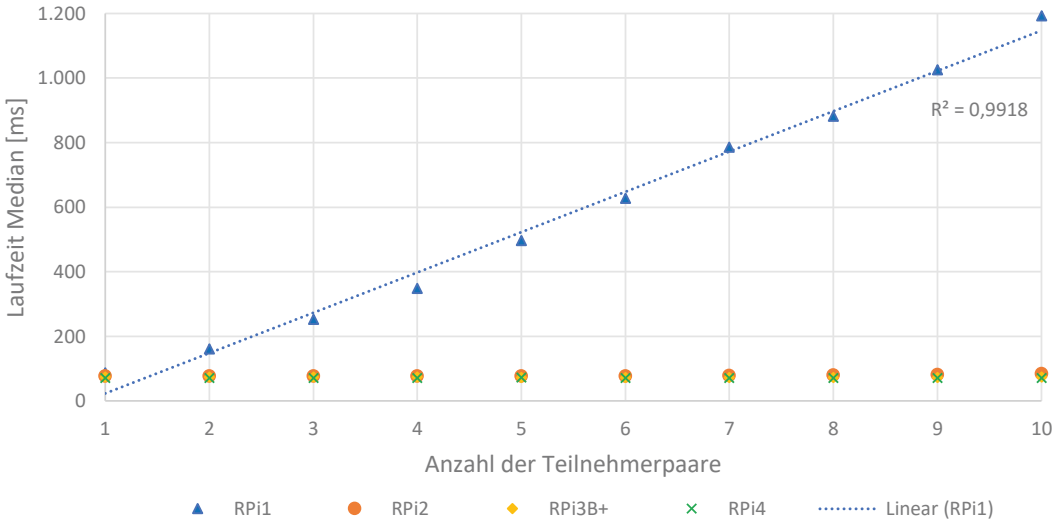

Abbildung 8.11: Skalierbarkeitsuntersuchung der *Service-Orchestrierung*: Laufzeiten mit steigender Anzahl an Fernsteuerungspaaren (immer *Raspberry Pi 1*) und verschiedenen *Orchestrator*-Hardwareplattformen.

Aus den Messergebnissen kann Folgendes abgeleitet werden: Unter der Voraussetzung einer hinreichend leistungsfähigen Plattform des *Orchestrators* hat die Anzahl der Paare aus *Fernsteuerungsgerät* und *gesteuertem Gerät* keinen Einfluss auf die Laufzeit der Fernsteuerungsvorgänge. Bei einer geringen Leistungsfähigkeit des *Orchestrators* zeigt sich hingegen ein linear ansteigendes Laufzeitverhalten. Dies liegt darin begründet, dass der *Orchestrator* die Anfragen nicht hinreichend schnell verarbeiten kann, sodass Wartezeiten entstehen. Der lineare Anstieg erfolgt ggf. erst ab einer Schwellanzahl von Fernsteuerungspaaren, wie beim *RPi 2*. Der *Orchestrator* hat mit steigender Anzahl an Fernsteuerungspaaren mehr parallel eintreffende Events zu verarbeiten und mehr Fernsteuerungsoperationen auszulösen. Entsprechend ist davon auszugehen, dass die Anzahl der Prozessorkerne einen großen Einfluss hat. Der *RPi 1* verfügt über einen Einzelkernprozessor, während die anderen Plattformen über vier Kerne verfügen. Beim *RPi 2* verändert sich das Laufzeitverhalten ab sieben Fernsteuerungspaaren ebenfalls hin zu einem linearen Anstieg. Daraus kann geschlussfolgert werden, dass weitere Faktoren wie die Taktfrequenz einen Einfluss haben.

Die konkrete Dimensionierung der *Orchestrator*-Hardware lässt sich für einen konkreten Anwendungsfall entsprechend experimentell ermitteln. Die absolute Laufzeit sollte ebenfalls in Abhängigkeit vom Anwendungsfall betrachtet werden. Es ist aber zu beachten, dass es zu *Timeouts* bei den Vernetzungsteilnehmern kommen kann, wenn die Verzögerungen zu groß werden.

8.4.4.3 Übertragbarkeit der Ergebnisse Die technische Evaluation wurde nur mit einer Software-Bibliothek vorgenommen. Wie bereits in Kapitel 8.4.4.1 diskutiert, sind die absoluten Messwerte weniger relevant als die relativen Unterschiede zwischen den vorgestellten Konzepten. Da allen Implementierungen dieselben Prinzipien zugrunde liegen, ist davon auszugehen, dass nur geringfügige Unterschiede im relativen Vergleich der Konzepte bestehen. Diese beruhen etwa auf der unterschiedlichen Performanz der Implementierungen der Rollen *Service Provider* und *Consumer*, wie es die Analysen in Kapitel 5.5.6 zeigen. Es kann also davon ausgegangen werden, dass Grundaussagen der technischen Evaluation verallgemeinert werden können. Für die absoluten Messwerte trifft dies hingegen nicht zu.

8.4.5 Zusammenfassung der technischen Evaluation

In der technischen Evaluation wurden das Konzept der *Service-Orchestrierung* und der *intuitive Ansatz*, stellvertretend für *Service-Choreografie*, gegenübergestellt. Im Ergebnis zeigt sich erwar-

Tabelle 8.3: Übersicht der gemessenen Laufzeit-Medianwerte für verschiedene Hardwareplattformen und eine steigende Anzahl an Teilnehmerpaaren.

Anzahl Teilnehmerpaare	**Laufzeit Median [ms]**			
	RPi1	**RPi2**	**RPi3B+**	**RPi4**
1	85,9	76,4	73,1	71,0
2	159,9	76,9	73,5	70,8
3	251,4	76,7	74,0	71,1
4	347,9	75,9	73,1	71,1
5	495,7	76,7	72,4	71,2
6	626,7	76,9	72,1	71,0
7	785,1	78,2	71,9	70,5
8	881,9	78,6	72,2	70,7
9	1024,7	80,7	72,1	70,9
10	1192,5	84,2	73,0	70,8

tungsgemäß ein *Overhead*. Dieser bewegt sich in einem akzeptablen Bereich. Somit überwiegen nach Ansicht des Autors die potentiellen klinischen und konzeptuellen Vorteile. Insbesondere beim Einsatz von leistungsstärkeren Hardware-Plattformen (*RPi 3* und höher) für die Realisierung des *Orchestrators* ist der Anstieg der Laufzeit eines Fernsteuerungsvorgangs gering, je nach Anwendungsfall ggf. sogar zu vernachlässigen.

Die Evaluation verdeutlichte zudem, dass das Konzept der *Service-Orchestrierung* skaliert. Bereits für mittelmäßig leistungsfähige Hardware des *Orchestrators* (*RPi 3* und höher) zeigt sich ein konstanter Laufzeit-Overhead der Fernsteuerungsvorgänge bei steigender Anzahl von Fernsteuerungsteilnehmerpaaren. Es konnte gezeigt werden, dass das Konzept technisch realisiert werden kann und für heutige und zukünftige Anwendungsfälle genutzt werden kann.

8.5 Diskussion

Zur Bewertung der drei in Kapitel 8.3.1 beschriebenen Grundkonzepte, *intuitiver Ansatz*, *Service-Orchestrierung* und *Service-Choreografie*, müssen verschiedene technische Aspekte und Aspekte der Konformitätsbewertung betrachtet werden. Die im Folgenden thematisierten Fragestellungen gehen dabei über die technische Evaluation aus Kapitel 8.4 hinaus, die mit dem Fokus auf Laufzeit des Fernsteuerungsvorgangs und Skalierbarkeit durchgeführt wurde.

8.5.1 Technische und wirtschaftliche Aspekte

Die Implementierungskomplexität der einzelnen Vernetzungsteilnehmer sollte so gering wie möglich sein. Dies senkt die Hürde für Hersteller, ihre Geräte so zu erweitern, dass die zusätzlich benötigten Funktionalitäten für die Teilnahme an einem dynamisch assoziierbaren System zur Verfügung stehen. Die Implementierung einer zusätzlichen *SOMDA*-Rolle stellt dabei einen deutlich höheren Aufwand dar als die Erweiterung der Funktionalität einer bestehenden Rolle. Dies ist einerseits im direkten Implementierungsaufwand begründet. Andererseits sind entsprechende neue Anforderungen der *IEEE 11073 SDC*-Normenfamilie zu erfüllen, insbesondere der *PKP*-Standards, die sich aus der neuen Rolle ergeben.

IEEE 11073 SDC wurde so entwickelt, dass *Service Provider* weniger komplex sind als *Service Consumer*. Die Implementierungskomplexität für die einzelnen Vernetzungspartner ist entsprechend im Konzept der *Service-Orchestrierung* am geringsten, da es ohne Rollenerweiterung der Geräte

auskommt. Sowohl beim *intuitiven Ansatz* als auch bei der *Service-Choreografie* müssen die Fernsteuerungsgeräte bzw. die zu steuernden Geräte zusätzliche *Service Consumer*-Funktionalitäten implementieren. Im Fall der *Service-Orchestrierung* wird dieser Aufwand für das Gesamtsystem an zentraler Stelle geleistet. Für die einzelnen Hersteller, vor allem von Geräten mit geringem Funktionsumfang, die ausschließlich die Rolle des *Service Providers* implementieren, bietet der *Orchestrator* eine große Entlastung. Die Wahrscheinlichkeit, dass Hersteller bereits vorhandene oder auch zukünftige Fernsteuerungselemente innerhalb der Vernetzung bereitstellen, kann daher beim Konzept der *Service-Orchestrierung* als am höchsten eingeschätzt werden. Die Einfachheit der beteiligten Vernetzungskomponenten reduziert außerdem den Aufwand für Betrieb und Debugging.

Der *Orchestrator* ist eine komplexe Komponente mit hohen Anforderungen an Funktions- und Ausfallsicherheit. Somit ist von einem sehr hohen Implementierungsaufwand auszugehen. Einerseits stellt diese Komponente damit ein neues Marktpotential für innovative und intuitiv bedienbare Systeme dar, andererseits ist das Konzept auf deren Existenz angewiesen. Entsprechend besteht die Gefahr eines „Henne-Ei-Problems“. Die Entwicklung einer solchen Komponente stellt für Unternehmen ein erhebliches wirtschaftlich-technisches Risiko dar.

Im Vergleich der drei Konzepte wird bei der *Service-Orchestrierung* der globale Entwicklungsaufwand über alle beteiligten Systeme am geringsten und die Interoperabilität am höchsten angesehen, da nicht jeder Vernetzungsteilnehmer die Logik eigenständig implementieren muss. Aufgrund der Trennung zwischen Fernsteuerungsgerät und zu steuerndem Gerät durch den *Orchestrator* weist die *Service-Orchestrierung* die größte Unabhängigkeit zwischen den Vernetzungspartnern auf. Dies sorgt für ein generisches Verhalten und somit für eine hohe Wiederverwendbarkeit.

Aus den Grundprinzipien der *SOA* (siehe auch Kapitel 3.1) und dem sogenannten *SOA-Manifest* [A 226] kann die Bestrebung zur Aufgaben- und Rollentrennung unter den Vernetzungsteilnehmern abgeleitet werden. Die strikte Aufgabentrennung durch den *Orchestrator* beim Konzept der *Service-Orchestrierung* erfüllt diese Idee umfassend, während die beiden anderen Konzepte lediglich eine logische Rollentrennung innerhalb physikalischer Instanzen vornehmen.

8.5.2 Risikomanagement und Aspekte der Konformitätsbewertung

Das Risikomanagement und die Möglichkeit einer erfolgreichen Konformitätsbewertung (umgangssprachlich Zulassungsfähigkeit) sind, neben der technischen Machbarkeit und Eleganz, entscheidende Aspekte für *PoC*-Medizingeräte. Zulassungsstrategien für dynamisch vernetzte Medizingeräte auf der Basis offener Standards sind nicht im Fokus der vorliegenden Arbeit. Hierfür sei auf Arbeiten von Janß et al. [A 50], Thorn et al. [A 51] oder die *SDC Conformance Principles* [A 227] verwiesen. Zum aktuellen Zeitpunkt gibt es keine konformitätsbewerteten Medizinprodukte, die Hersteller-übergreifend mittels einer *IEEE 11073 SDC*-Schnittstelle kommunizieren[8.5]. Daher kann nicht auf bestehende Erfahrungen und Verfahren zurückgegriffen werden. Nichtdestotrotz sollen einige Aspekte und aktuelle Expertenmeinungen zu den beschriebenen Konzepten vorgestellt werden.

Die Zulassungsfähigkeit der beschriebenen Konzepte erfordert zunächst eine grundlegende Veränderung der Prozesse. Stand heute werden genau spezifizierte Verbünde von Medizingeräten *in den Verkehr gebracht*. In solchen Verbünden ist genau festgelegt, welches Gerät mit welchem Gerät für welche Funktionalität Informationen und Befehle austauscht. Bereits ein Versionsupdate beteiligter Komponenten kann unter Umständen eine Neuzulassung erfordern. Vernetzt ein Betreiber (z. B. eine Klinik) eigenständig Medizingeräte, so besteht die Gefahr, dass der Betreiber die Herstellerhaftung übernehmen muss, wie etwa aus der Medizinprodukte-Betreiberverordnung [A 228] hervorgeht. Daher muss für eine sinnvolle Nutzung des *IEEE 11073 SDC*-Vernetzungskonzepts dieses Vorgehen

[8.5] Das in Abschnitt 4.7 erwähnte und in der Pressemitteilung der Firma *Dräger* [A 154] beschriebene Vernetzungsszenario beschränkt sich auf Geräte der Firma *Dräger*, die über *IEEE 11073 SDC* kommunizieren.

grundlegend geändert werden. Eine Zulassung eines Medizingerätes auf der Grundlage von standardisierten Vernetzungsschnittstellen, Profilen, *PKPs*, Gerätespezialisierungen, Sicherheitsstandards (*Safety* wie *Security*) etc. ist erforderlich. Dieser Veränderungsprozess, der auch eine modulare Risikobewertung und Zulassung beinhaltet, hat bereits begonnen [A 51].

Analog zur Diskussion der Komplexität und des Implementierungsaufwands der einzelnen Komponenten kann festgehalten werden, dass eine Separierung der Aufgaben durch das Konzept der *Service-Orchestrierung* dazu führt, dass Fernsteuerungsgeräte und zu steuernde Geräte im Vergleich zu den beiden anderen Konzepten einen geringeren zulassungsrelevanten Funktionsumfang besitzen. Entsprechend ist von einem einfacheren Zulassungsprozess auszugehen. Die Mechanismen des Risikomanagements eines zu steuernden Gerätes, wie etwa in Kapitel 7 beschrieben, bleiben durch den *Orchestrator* unberührt.

Der *Orchestrator* besitzt, wie beschrieben, eine hohe technische Komplexität, deren sicherer Betrieb zu zertifizieren ist. Nach den Vorgaben von Anhang VIII der *MDR* [A 10] bzw. Anhang IX der *MDD* [A 11] ist davon auszugehen, dass der *Orchestrator* mindestens der höchsten Risikoklasse der zu steuernden Geräte zuzurechnen ist. Für eine umfassende Umsetzung für den OP-Saal bedeutet dies Klasse IIb oder die höchste Klasse III. Die Anforderungen an die Konformitätsbewertung sind entsprechend hoch.

Das Konzept der *Service-Orchestrierung* und explizit die Zulassungsfähigkeit des *Orchestrators* wurde bereits mit verschiedenen Expertinnen erörtert. Diskussionen fanden etwa im Zuge der Präsentation der Publikation [B 21] während der *IEEE HI-POCT* Konferenz 2016, mit Projekt- und assoziierten Partnern der nationalen Projekte *OR.NET* und *MoVE* sowie des US-amerikanischen Projekts *MD PnP*, als auch mit Vertretern der US-amerikanischen *FDA* statt. Während sich die *FDA*-Vertreter und die Mehrheit der wissenschaftlichen Partner positiv zur generellen Zulassungsfähigkeit des Konzepts geäußert haben, bestehen vor allem von Expertinnen aus dem Industrieumfeld Zweifel. Diese bestehen insbesondere hinsichtlich der Komplexität, der Allgemeingültigkeit und des Dokumentationsaufwands des *Orchestrators*. Dies ist verbunden mit der Frage, ob es einen (oder mehrere) Hersteller geben wird, die das Konzept implementieren und zulassen, sodass das bereits angesprochene „Henne-Ei-Problem“ entsteht.

Dokumentierte und verbindliche Aussagen zur generellen Zulassungsfähigkeit des Konzepts der *Service-Orchestrierung* und konkreter Umsetzungen des *Orchestrators* existieren derzeit nicht.

8.5.3 Zusammenfassung

Von den drei beschriebenen Konzepten zur dynamischen Assoziierung von Schaltelementen zu Fernsteuerungsoperationen von vernetzten Medizingeräten wird die *Service-Orchestrierung* als die technisch eleganteste Lösung angesehen. Die Machbarkeit wurde in mehreren Demonstratoren gezeigt. Die Performanzevaluation unterstreicht die praktische Nutzbarkeit. Der Implementierungsaufwand ist so verteilt, dass für die Hersteller von Geräten mit Fernsteuerungselementen und von fernsteuerbaren Geräten ein geringer bzw. kein Mehraufwand besteht. Dies ist ein großer Vorteil im Hinblick auf die praktische Umsetzung in Medizingeräten. Die Realisierbarkeit des notwendigen *Orchestrators* im Sinne der Zulassungsfähigkeit muss hingegen weiter evaluiert werden.

Sollte die *Service-Orchestrierung* nicht oder nicht vollständig in der Praxis umsetzbar sein, behalten andere Teile des vorgestellten Konzepts ihre Gültigkeit und Relevanz. So werden bereits aktuell Teile der Modellierung des Auslöse- und Assoziierungszustands von Fernsteuerungselementen (siehe Kapitel 8.3.2) in die Standardisierung von Gerätespezialisierungen übernommen.

Für die praktische Umsetzung einer dynamischen Assoziierung von steuernden und zu steuernden Vernetzungspartnern sind weitere, über diese Arbeit hinausgehende, Forschungs- und Entwicklungsarbeiten notwendig. Für die Konfiguration der Assoziierungen und die Anzeige des aktu-

ellen Assoziierungszustands werden intuitiv und sicher bedienbare Nutzerschnittstellen benötigt. Zu jedem Zeitpunkt muss für die Akteure im OP-Saal klar sein, welches Schaltelement welche Aktion an einem, potentiell entfernen Gerät, auslöst. Entsprechende Arbeiten erfolgten beispielsweise durch [A 72, 73, 219, 220]. Eine besondere Bedeutung kommt dabei der visuellen Beschreibung von Schaltelementen und Bedienoberflächen von fernsteuerbaren Medizingeräten zu. Dies ist eine Voraussetzung für die dynamische Generierung von Benutzeroberflächen für unbekannte Fernsteuerungsgeräte und stellt eine offene Forschungsfrage dar.

Im beschriebenen Konzept stellt der *Orchestrator* aktuell einen *SPoF* dar. In Kapitel 8.3.3 wurden Erweiterungen für eine redundante bzw. verteilte Realisierung diskutiert. Entsprechende Maßnahmen sollten umgesetzt und in klinischen Systemen genutzt werden.

Im Bereich der Normung sind Erweiterungen der Nomenklatur notwendig. Neue Nomenklatur-Codes zur semantischen Beschreibung von Fernsteuerungsoperationen werden derzeit im Standardisierungsprojekt *IEEE P11073-10107* [D 4] erarbeitet. Des Weiteren sollte das beschriebene Konzept, bzw. Teilaspekte, in Standards zu Gerätespezialisierungen und/oder *PKPs* Eingang finden. Von Relevanz sind insbesondere der *PKP*-Standard *IEEE P11073-10703* [D 7] und Gerätespezialisierungen für Fernsteuerungsgeräte wie Fußschalter, die als Teil von *IEEE P11073-10720* [D 8] geplant sind.

9 Zuverlässige, herstellerunabhängige Alarmierungssysteme

Die regelrechte Flut von Alarmen, die von Medizingeräten im OP-Saal oder der Intensivstation (ITS) generiert werden, ist eine große Herausforderung im klinischen Alltag. Alarmmüdigkeit und Desensibilisierung können zum Tod von Patientinnen führen, der Lärm behindert den Genesungsprozess und schadet dem Pflegepersonal [A 2, 37–40].

Die Forschung der letzten Jahre im Bereich der (intelligenten) Alarmsysteme hat keinen entscheidenden Fortschritt gebracht und hat zumeist nicht den Schritt in die Praxis geschafft [A 40]. Die Vision der „Silent ICU“ beschreibt eine geräuschlose bzw. geräuscharme ITS, in der die Lärmbelastung für Patienten und Personal so gering wie möglich ist. Sie scheint nach wie vor in weiter Ferne zu liegen. Hier setzt das im Folgenden beschriebene Konzept an und legt die Grundlagen für zukünftige innovative Funktionalitäten von Alarmsystemen: Ein System zur zuverlässigen und verteilten Alarmierung auf der Basis der herstellerunabhängigen Medizingerätevernetzung mittels *IEEE 11073 SDC*.

Während sich die *IEEE 11073 SDC*-basierten Anwendungen der Kapitel 7 und 8 mit Aspekten der zuverlässigen und dynamischen Fernsteuerung beschäftigen, wird in diesem Kapitel der Aspekt der Alarmsysteme bearbeitet. Einerseits sind viele Szenarien einer entfernten Bedienung von Medizingeräten ohne entsprechende (verteilte) Alarmsysteme undenkbar. Andererseits kommt die *IEEE 11073 SDC*-Fernsteuerungsfunktionalität für verteilte Alarmsysteme direkt zum Einsatz.

Diesem Kapitel liegt die Veröffentlichung „A Safe and Interoperable Distributed Alarm Notification System for PoC Medical Devices using IEEE 11073 SDC“ [B 15] zugrunde. Neben den Betreuern der Dissertation ist die Arbeit ohne weitere Co-Autoren entstanden. Im Zuge der *IEEE National Institutes of Health Special Topics Conference on Healthcare Innovations and Point of Care Technologies (HI-POCT)* 2017 in Bethesda, Maryland, USA, wurde die Arbeit mit dem *Best (PhD) Student Paper Award* ausgezeichnet.

9.1 Einleitung und Problembeschreibung

Während einer komplexen chirurgischen Operation werden mehr als 70 Alarme pro Stunde erzeugt [A 40]; auf einer ITS sind es pro Patientin und Stunde mehr als 45 Alarme [A 3]. Davon sind zwischen 63 % und 99 %[9.1] als klinisch irrelevant oder gar als „Fehlalarme“ anzusehen [A 1–3, 37, 38, 40]. Die Werte unterscheiden sich zwischen den medizinischen Bereichen, auf die sich die verschiedenen Studien beziehen. Allein diese Zahlen verdeutlichen die Relevanz des Problems. Hieraus resultiert eine große Belastung für Ärzte und Pflegepersonal, was die sogenannte *Alarmmüdigkeit* zur Folge hat [A 2]. Alarmbezogene Probleme stellen ein hohes Risiko für die Patientensicherheit dar. Sie sind häufig auf die *Alarmmüdigkeit* zurückzuführen, aber auch weitere Aspekte wie Fehlkonfigurationen spielen eine Rolle. Im jährlich vom *Emergency Care Research Institute (ECRI)* herausgegebenen Report zu den „Top 10 Health Technology Hazards“ waren Gefahren im Zusammenhang mit Alarmen von 2009 bis 2019 immer vertreten, davon allein achtmal unter den Top 3 [A 229–239].

The Joint Commission (TJC), eine US-amerikanische Non-Profit-Organisation, dokumentiert sogenannte *Sentinel Events*. *Sentinel Events* sind von der *TJC* definiert als unerwartete Ereignisse, die zum Tod oder einem schweren physischen oder psychischen Schaden von Patientinnen führen, die nicht in Verbindung mit dem natürlichen Krankheitsverlauf stehen [A 240]. Für den Zeitraum von Januar 2009 bis Juni 2012 wurden beispielsweise 98 alarmbezogene *Sentinel Events* in den USA registriert, von denen 80 zum Tod führten [A 37].

[9.1] Einige Studien gehen gar von 80 % bis 99 % aus [A 2, 37, 40].

Die US-amerikanische Behörde *Food and Drug Administration (FDA)* führt für den Zeitraum von Januar 2005 bis Juni 2010 in ihrer *Manufacturer and User Facility Device Experience (MAUDE)*-Datenbank 566 alarmbezogene Todesfälle in den USA. Experten gehen von einer noch höheren Zahl aus [A 37]. Auch wenn diese Zahlen im Verhältnis zu den Gesamtbehandlungszahlen als gering erscheinen könnten, zeigen sie doch, dass die Patientensicherheit erheblich verbessert werden kann.

Ein weiteres Problem ist die Lärmbelastung in heutigen Krankhäusern. Alarmtöne mit bis zu 80 db(A), was der Lautstärke von starkem Straßenverkehr entspricht [A 241], haben einen erheblichen Anteil an der Lärmbelastung. In OP-Sälen und Intensivstationen erreicht diese nicht selten einen Wert von 90 db(A) [A 38, 39, 241]. Dieser Lärm schädigt Patienten wie Pflegepersonal [A 38, 39]. Studien haben gezeigt, dass Lärm einen signifikanten Stress auslöst. Dies führt bei Patienten zu Schlafentzug, der das Immunsystem und die Aktivität des Nervensystems negativ beeinflusst und sogar Psychosen verursachen kann [A 38, 40]. Für Ärzte und Pflegepersonal lässt die permanente Lärmbelastung das Risiko für Burn-out-Erkrankungen steigen und verringert die Konzentrationsfähigkeit während der Arbeit erheblich [A 38, 40].

Um diesen Problemen zu begegnen, wird ein System beschrieben, das zuverlässige und verteilte Alarmierungen über Geräte- und Herstellergrenzen hinweg ermöglicht. Zudem stellt es eine sichere Grundlage für weitere Entwicklungen von intelligenten, Computer-assistierten Alarmsystemen dar.

Ein *verteiltes Alarmsystem* besteht hierbei aus den alarmerzeugenden Medizingeräten und beliebigen alarmkommunizierenden Endgeräten. Hierzu zählen beispielsweise mobile Endgeräte, die Ärztinnen und Pflegepersonal am Körper tragen, stationäre Systeme an Leitstellen oder zentralen Arbeitsplätzen etc. Ein solches *verteiltes Alarmsystem* ermöglicht es, die Alarme an dem Ort zu kommunizieren, an dem sich die Pflegenden gerade aufhalten. Auf einer ITS ist dies beispielsweise häufig nicht am Bett der alarmauslösenden Patientin, da mehrere Patientinnen parallel betreut werden müssen. Daher hat die ortsunabhängige Alarmierung einige entscheidende Vorteile: Reduzierung der Alarmgeräusche bei der Patientin, da hier die Alarme in Abwesenheit der Pflegenden nicht kommuniziert werden. Reduzierung der Alarmgeräusche für das Pflegepersonal, da die Lautstärke alarmierender Geräte direkt beim intendierten Empfänger besser abgestimmt werden kann. Reduzierung der Arbeitslast der Pflegenden, da in einem herstellerübergreifend interoperablen System neben den Alarmen selbst auch die ursächlichen Parameter und weitere Informationen auf die mobilen Endgeräte übertragen werden können. Dies ermöglicht eine sichere Einschätzung der Situation auch abseits des Patientenbettes, sodass unnötige Wegstrecken und Unterbrechungen anderer Arbeiten reduziert werden können. Neben mobilen Alarmierungsendgeräten können auch zentrale Arbeitsstationen zur besseren Koordination auf der Basis der *IEEE 11073 SDC*-Standards umgesetzt werden.

Verteilte Alarmsysteme mit mobilen Endgeräten und zentralen Arbeitsstationen sind bereits von verschiedenen Herstellern am Markt verfügbar. Stellvertretend seien Lösungen der Firmen *Philips* [A 242], *GE Healthcare* und *Ascom* [A 243] oder *Dräger* und *tetronik* [A 244, 245] genannt. Ebenso bestehen eine Reihe von Patenten in diesem Bereich. Die Systeme sind geschlossen, oft monolithisch und setzen auf proprietären und herstellerspezifischen Vernetzungslösungen auf. Systeme auf der Basis von Normen sind nicht bekannt.

In den letzten Jahren wurde Forschung mit hohem Aufwand auf dem Gebiet des Alarmmanagements für den OP-Saal und die ITS betrieben. Diese Aktivitäten und Ergebnisse werden in einigen Überblicksartikeln einem umfassenden Review unterzogen und zusammengefasst, beispielsweise [A 2, 3, 38–40]. Es werden etwa die vielfältigen Probleme der *Alarmmüdigkeit* und Desensibilisierung, der hohen falsch-positiv Raten medizinischer Alarme, der Lärmbelastung durch Alarme, der Alarmwahrnehmbarkeit und -identifikation etc. adressiert.

Unter der Überschrift *smarte* oder *intelligente* Alarmsysteme existieren eine Vielzahl von Ansätzen unterschiedlichster Komplexität: Plausibilitätskontrolle von Messwerten durch mehrere Quellen, wie beispielsweise die Herzfrequenz vom Elektrokardiogramm (EKG) und vom Pulsoximeter; adaptive

Alarmeinstellungen in Abhängigkeit von Vorinformationen über den Patienten oder die Phase des chirurgischen Eingriffs; statistische Methoden zur Signalextraktion und -filterung; automatisierte Ursachenanalysen; Trendüberwachung etc. Eine Reihe von Forschungsprojekten setzen dabei auf komplexe Methoden der *künstlichen Intelligenz (KI)* wie künstliche neuronale Netze, Fuzzy-Logik oder Bayes'sche Netze. Nichtsdestotrotz haben Studien gezeigt, dass schon vergleichsweise simple Ansätze, wie etwa Verzögern der Alarmauslösung, eine hohe Wirksamkeit haben. Dabei wird ein Alarm erst ausgelöst, wenn die Alarmbedingung, wie etwa das Über- oder Unterschreiten eines Schwellenwertes, für einen gewissen Zeitraum erfüllt ist und nicht unmittelbar nach dem Erfüllen der Bedingung.

Trotz einer Vielzahl von überzeugenden Konzepten und entsprechenden Studien hat sich die Rate der Fehlalarme bzw. medizinisch irrelevanten Alarme in der klinischen Realität in den letzten Jahren nicht verringert [A 40]. Die Nutzung von intelligenten Alarmsystemen bleibt hinter der allgemeinen Entwicklung und den Innovationen von Medizinprodukten zurück [A 246]. Bedenken bezüglich der Haftungsfrage und Geschäftsinteressen der Hersteller scheinen Gründe hierfür zu sein [A 246]. Die derzeitige „Better-Safe-Than-Sorry-Mentalität" von Herstellern und Zulassungsbehörden führt dazu, dass eine immens hohe Zahl an Fehlalarmen und die daraus resultierenden negativen Folgen eher akzeptiert sind als die Möglichkeit, einen validen Alarm zu verpassen [A 40].

Die betrachteten Überblicksartikel unterstützen den Ansatz zuverlässiger Alarmsysteme auf der Basis von Interoperabilitätsstandards. Borowski et al. [A 38] legen etwa dar, dass sich vernetzte Alarmierungsgeräte positiv auf die Anzahl der Alarme auswirken können. Hierfür ist aber eine Interoperabilität auf der Basis standardisierter und zertifizierter Schnittstellen erforderlich. Dies entspricht aber nicht dem aktuellen Stand der Technik. Insbesondere die Patientensicherheit muss hierbei im Fokus stehen, da eine hohe Zuverlässigkeit notwendig ist und Haftungsfragen bei Alarmsystemen berücksichtigt werden müssen [A 38]. Cvach [A 2] verdeutlicht die Wichtigkeit von zusätzlichen, mobilen Geräten zur Alarmbenachrichtigung, um die Hörbarkeit/Wahrnehmbarkeit und die Identifizierung der Alarmquelle zu verbessern. Insbesondere unterstreicht sie die Notwendigkeit von Alarmsystemen auf der Basis von drahtlosen Technologien [A 2]. Für zukünftige Systeme sind damit die Herausforderungen eines funkbasierten Systems mit Endgeräten ohne externe Stromversorgung zu beachten. Konkani et al. [A 39] kommen zu dem Schluss, dass ein hoher Bedarf an Systemen von Drittanbietern zur Konsolidierung und Notifizierung der Alarme besteht und weisen auf die Wichtigkeit der Standardisierung hin. Sie machen zusätzlich darauf aufmerksam, dass heutige Systeme von Drittanbietern oft einen *Single Point of Failure (SPoF)* aufweisen oder gar jedes Endgerät einen solchen *SPoF* darstellt [A 39].

Zusammenfassend kann festgestellt werden, dass zunächst ein zuverlässiges System für die verteilte Alarmierung benötigt wird. Dies lässt sich aus dem klinischen Alltag, dem aktuellen Stand von Forschung und Technik sowie den aufgezeigten Hindernissen für die Nutzung neuer Entwicklungen in realen Medizinprodukten ableiten. Es ist daher ein Konzept zu entwickeln, dass auf herstellerunabhängigen und standardisierten Kommunikationsschnittstellen aufbaut und den Ansprüchen an Zulassung und Haftung genügt.

9.2 Begrifflichkeiten

Nicht alle Begriffe im Zusammenhang mit Alarmsystemen sind abschließend klar definiert. Zum besseren Verständnis werden daher im Folgenden einige Festlegungen für diese Arbeit getroffen. Diese orientieren sich maßgeblich an der Norm *IEC 60601-1-8* [A 6] [9.2]

[9.2] Für diese Arbeit wird die Norm *IEC 60601-1-8:2006/A1:2012* [A 6] in ihrer zweiten, konsolidierten Auflage von 2006 und dem ersten Amendment von 2012 genutzt. Das zweite Amendment von 2020 stand noch nicht zur Verfügung und konnte daher nicht berücksichtigt werden. In deutscher Sprache ist sie als *DIN EN 60601-1-8* [A 247] verfügbar.

Alarmsystem Im Sinne der Norm *IEC 60601-1-8* [A 6] ist ein *Alarmsystem* ein Teil eines Medizingerätes bzw. Medizingerätesystems, das Alarmbedingungen detektiert und ggf. Alarmsignale generiert.

Verteiltes Alarmsystem Im Sinne der Norm *IEC 60601-1-8* [A 6] ist ein *verteiltes Alarmsystem* ein Alarmsystem, das mehr als eine Komponente eines Medizingerätesystems enthält. Die Komponenten eines verteilten Alarmsystems können potentiell weit voneinander entfernt sein. Einzelne Komponenten können sich auch außerhalb der Patientenumgebung befinden. Die Form des Datenaustauschs zwischen den Teilen des verteilten Alarmsystems ist dabei unerheblich. Die Norm unterscheidet zwischen *verteilten Alarmsystemen* mit und ohne Bestätigungen der Übermittlung von Alarmbedingungen. Systeme ohne Bestätigungen, die damit einen geringeren Grad an Sicherheit aufweisen, werden auch als *verteilte Informationssysteme* bezeichnet.

Verteiltes Alarmierungssystem In dieser Arbeit wird für das eigene Konzept bewusst der Begriff *Alarmierungssystem* genutzt. Dies ist eine sprachliche Abgrenzung gegenüber dem allgemeinen Begriff *Alarmsystem*, im Sinne der Norm *IEC 60601-1-8* [A 6]. Es soll klargestellt werden, dass sich die hier entwickelten Konzepte ausschließlich auf den Bereich der *Alarmsignalgenerierung*, im *verteilten* Fall, bezieht. Aspekte wie die Anforderungen an die Bestimmung von Alarmbedingungen und Zuweisung von Prioritäten, Lautstärken, Farben etc. von Alarmsignalen usw. sind jedoch nicht Teil dieser Arbeit.

Alarmsignalgenerierung Für ein beliebiges Alarmsignal ist die Generierung der Oberbegriff, der beschreibt, dass das Alarmsignal dem Nutzer in der entsprechenden Form mitgeteilt wird. Formen sind etwa das Anzeigen von visuellen Alarmsignalen, das Erzeugen von Alarmtönen für akustische Alarmsignale oder das Erzeugen von wahrnehmbaren Impulsen bei haptischen Alarmsignalen.

Primäres Alarmsystem (PAS) In einem *verteilten Alarmsystem* wird das Medizingerät als *primäres Alarmsystem* bezeichnet, das die Alarmbedingung überwacht und die Verantwortung für die Alarmierung trägt. Typische Beispiele sind Vitalparameterminitore oder Spritzenpumpen. Der Begriff ist nicht normativ definiert, wird aber in den erläuternden Ausführungen zu *verteilten Alarmsystemen* im Anhang A.2 der Norm *IEC 60601-1-8* [A 6] genutzt. Auch der Begriff *Quellalarmsystem* kann genutzt werden.

Entfernter Alarmsignalgenerator (ESG) Der Begriff *entfernter Alarmsignalgenerator* beschreibt einen Teilnehmer in einem *verteilten Alarmierungssystem*, der Alarmsignale generiert und nicht das *primäre Alarmsystem* selbst ist. Die räumliche Distanz ist dabei unerheblich. Entscheidend ist die *externe* Generierung außerhalb des *primären Alarmsystems*. In einem *verteilten Alarmierungssystem* können beliebig viele *entfernte Alarmsignalgeneratoren* existieren. Beispiele sind zentrale Arbeitsstationen oder mobile Endgeräte.
Die Norm *IEC 60601-1-8* [A 6] nutzt Beschreibungen wie „Ausgabe von Alarmsignalen an *entfernt* aufgestellten Geräten“. Daher wird in dieser Arbeit der Begriff *entfernt* genutzt. Als alternative Bezeichnung wäre auch *externer Alarmsignalgenerator* denkbar.

9.3 Anforderungen an zuverlässige, verteilte Alarmierungssysteme

Verteilte Alarmierungssysteme müssen verschiedene Anforderungen erfüllen. Diese ergeben sich einerseits aus dem klinischen Alltag und dem Stand von Forschung und Technik (siehe Abschnitt 9.1) und andererseits aus Sicherheitsanforderungen, die aus Normen wie beispielsweise der IEC 60601-Normenreihe, abgeleitet werden können. Die Norm *IEC 60601-1-8* [A 6] spezifiziert Alarmsysteme für medizinische elektronische Geräte im Allgemeinen, also ohne Bezug zu einer konkreten Geräteklasse. In Kapitel 6.11 setzt sich die Norm mit *verteilten Alarmsystemen* auseinander und formuliert

entsprechende Anforderungen. Für die vorliegende Arbeit wird von diesen Anforderungen abstrahiert, wobei das Konzept die Erfüllung der gestellten Anforderungen voll umfänglich ermöglicht (siehe Abschnitt 9.6).

Anforderungen bestehen aus der Sicht der *PASs*, also der Geräte die direkt am Patienten die Alarmbedingungen überwachen und dort Alarmsignale generieren können. Ebenso existieren Anforderungen, die es den *ESGs* ermöglichen, Alarme zu kommunizieren, die von einem anderen Gerät ursächlich erzeugt werden.

Anforderung 1 Das *primäre Alarmsystem* muss alle Informationen zur Verfügung stellen, die zur korrekten Generierung der Alarmsignale durch die *entfernten Alarmsignalgeneratoren* benötigt werden. Dies sind etwa der Zustand von Alarmsignal und -bedingung, Notifizierungsart etc. Herstellerunabhängige und semantische Interoperabilität muss gewährleistet sein.

Anforderung 2 Das *verteilte Alarmierungssystem* muss eine beliebige Anzahl von *primären Alarmsystemen* und *entfernten Alarmsignalgeneratoren* ermöglichen.

Anforderung 3 Das *primäre Alarmsystem* muss in der Lage sein festzustellen, ob *entfernte Alarmsignalgeneratoren* dazu bereit sind, Alarmsignale zu generieren.

Anforderung 4 Das *primäre Alarmsystem* muss in der Lage sein zu überwachen, ob *entfernte Alarmsignalgeneratoren* Alarmsignale korrekt generieren.

Die erste Anforderung stellt die Basis einer verteilten, offenen, herstellerunabhängigen und interoperablen Lösung dar: Informationen müssen zuverlässig und interpretierbar bereitgestellt werden. Medizingeräteensembles sind komplex und bestehen aus einer Vielzahl an Geräten, die verschiedene Alarme überwachen und verschiedene Alarmsignale generieren können. Ebenso sind oft mehrere Patientinnen von mehreren Personen zu betreuen. Als Konsequenz ergibt sich die zweite Anforderung. Die Anforderungen drei und vier entstammen dem Risikomanagement der *PASs* und der *IEC 60601-1-8* [A 6]. Zunächst einmal muss das *PAS* feststellen können, ob im Geräteensemble Teilnehmer vorhanden sind, die in der Lage sind, die Alarmsignale zu generieren, unabhängig von der aktuellen Notwendigkeit. Man kann von der Bereitschaft zur Generierung von Alarmsignalen des *ESG* sprechen. Im Fall der Alarmierung muss das *PAS* in der Lage sein zu überwachen, dass mindestens ein *ESG* die Alarmierung korrekt vornimmt.

In der Praxis werden häufig drahtlose Technologien zum Einsatz kommen, zumindest für mobile Endgeräte. Daher ist insbesondere eine Robustheit gegenüber störenden Netzwerkeinflüssen, wieVerbindungsabbrüchen, Jitter etc., zu gewährleisten. Als weitere Fehlerquelle für mobile *ESGs* sind Ausfälle aufgrund leerer Akkus wahrscheinlich. Im Fall eines solchen ungewollten bzw. für das *PAS* nicht überwachbaren Verhaltens muss das *PAS* die Alarmsignalgenerierung typischerweise selbst vornehmen. Entsprechend dem Risikomanagement würde damit die *verteilte* Funktionalität durch ein lokales Fallback-Verhalten ersetzt.

Die fünfte Anforderung basiert einerseits auf den Analysen von Konkani et al. [A 39], andererseits spiegelt sie ein Grundprinzip der Ausfallsicherheit wider:

Anforderung 5 Das *verteilte Alarmierungssystem* darf keinen *Single Point of Failure (SPoF)* aufweisen. Existiert auf logischer Ebene ein *SPoF*, so sind entsprechende Redundanzen konzeptuell vorzusehen und umzusetzen.

Die vorliegende Arbeit nutzt die *IEEE 11073 SDC*-Normenfamilie, um die Anforderungen 1 – 5 zu erfüllen. In der zugrundeliegenden *Service-Oriented Architecture (SOA)* wird klar zwischen den Rollen *Service Provider* und *Consumer* unterschieden (siehe auch Abschnitt 3.1). Die Einhaltung dieser logischen Trennung wird daher als zusätzliche Anforderung aufgenommen. *PASs* bieten meist eine Reihe weiterer Informationen im Netzwerk an, die über die Alarmfunktionalität hinausgehen, wie etwa die zugrundeliegenden Messwerte. Ein typisches Beispiel sind Patientenmonitore, die in

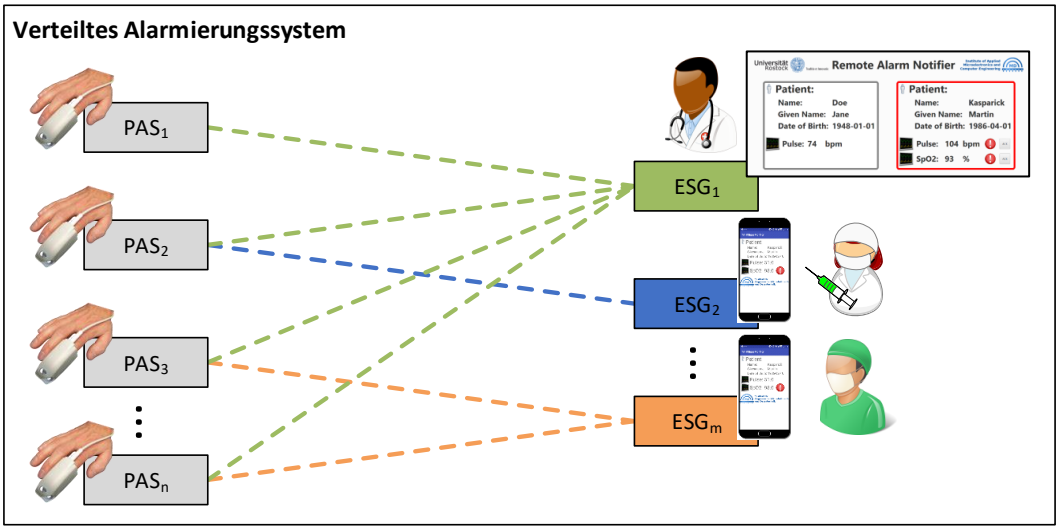

Abbildung 9.1: Beispielhafte Veranschaulichung des Konzepts eines *verteilten Alarmierungssystems*. Links: *primäre Alarmsysteme (PAS)*, die Alarmbedingungen am Patienten überwachen; rechts: *entfernte Alarmsignalgeneratoren (ESG)*, die Alarme dort generieren, wo es sinnvoll ist.

der Rolle des *Service Providers* eine Vielzahl an Messwerten und Alarmen zur Verfügung stellen. Zentrale Arbeitsstationen im OP-Saal oder der ITS, aber auch mobile Endgeräte, werden als ESGs agieren. Aus der Sicht der *SOA* implementieren sie die logische Rolle des *Service Consumers*. Um die Implementierungskomplexität der beteiligten Geräte nicht unnötig zu erhöhen und dem Paradigma der *SOA* gerecht zu werden, wird die sechste Anforderung definiert:

Anforderung 6 Die Sicherheitsmechanismen dürfen nicht erfordern, dass die grundlegenden Rollen des *SOA*-Paradigmas von *primären Alarmsystemen* (*Service Provider*) und *entfernten Alarmsignalgeneratoren* (*Service Consumer*) erweitert werden müssen.

9.4 Konzept: Mechanismen für zuverlässige, verteilte Alarmierungssysteme

Das Konzept zur Realisierung der Sicherheits- und Interoperabilitätsanforderungen mittels der *IEEE 11073 SDC*-Normenfamilie basiert auf zwei Grundideen (siehe auch Abbildung 9.1):

1. Das *primäre Alarmsystem* (*PAS*) definiert für jede Alarmbedingung, die *verteilt* signalisiert werden soll, zwei Alarmsignale. Für ein Signal wird das Attribut `Location` des entsprechenden Zustands auf den Wert `Remote` gesetzt, für das andere auf den Wert `Local`. Das erste Alarmsignal soll von einem *entfernten* Teilnehmer angezeigt werden, während das zweite als *Fallback-Alarmsignal* fungiert.

2. Der *entfernte Alarmsignalgenerator* (*ESG*) teilt dem *PAS* regelmäßig seine Bereitschaft zur Alarmsignalgenerierung, bzw. während der Signalgenerierung deren Zustand, mit.

Theoretisch wäre es auch denkbar, dass das *PAS* ohne lokales *Fallback-Alarmsignal* arbeitet, wenn es technisch nicht in der Lage ist, ein solches zu generieren. Aus der Sicht des Risikomanagements müssten dann andere *Fallback-Mechanismen* vorgesehen werden. Für das Konzept ist die konkrete Maßnahme unerheblich. Der Einfachheit halber erfolgt die Konzepterklärung unter der Annahme, dass das lokale *Fallback-Alarmsignal* existiert.

9.4.1 Modellierung der Alarmfunktionalitäten

Das *PAS* stellt wie beschrieben je zwei Alarmsignale (`Location` Attribut mit den Werten `Local` bzw. `Remote`) für eine Alarmbedingung zur Verfügung. Beide Alarmsignale haben dieselbe Erscheinungs-

form (`Manifestation`): visuell, akustisch bzw. haptisch. Das lokal generierte Alarmsignal fungiert als *Fallback-System* des *PASs* im Fall eines Problems mit der entfernten Alarmsignalgenerierung durch den *ESG*.

Die Alarme werden entsprechend semantisch mittels Termcodes beschrieben, die in *IEEE 11073 SDC* standardmäßig der *IEEE 11073-1010x*-Nomenklatur-Serie entnommen werden. Als Beispiel sei der Code 3::3120 (`MDC_EVT_ECG_TACHY`) zur semantischen Beschreibung einer Tachykardie[9.3] genannt.

Darüber hinaus beschreibt das *PAS* weitere Eigenschaften der Alarmbedingung, wie etwa die Art (physiologisch, technisch), die Priorität, die Generierungsverzögerung etc. Für die Alarmsignale wird spezifiziert, ob sie in einen `Latched`-Zustand[9.4] übergehen können, ob sie bestätigt werden können (*Acknowledgement*), wie lange eine solche Bestätigung anhält, wie das Verzögerungsverhalten zwischen dem Eintritt der Alarmbedingung und dem Generieren des Alarmsignals ist etc. (siehe auch Kapitel 4.4.1.1). Ob sich einzelne Parameter der beiden Alarmsignale voneinander unterscheiden, hängt vom Anwendungsfall ab.

Die umfassende Selbstbeschreibung der Alarme stellt einerseits die herstellerübergreifende Interpretierbarkeit der Informationen in einem heterogenen System sicher. Andererseits macht das *PAS* damit entsprechend seinem Risikomanagement umfassende Vorgaben, wie die Alarmsignalgenerierung im *verteilten Alarmierungssystem* zu erfolgen hat. Damit ist die Anforderung 1 erfüllt.

9.4.2 Systemmodell zur Beschreibung des Verhaltens zur Laufzeit

Um das relevante Verhalten des *verteilten Alarmierungssystems* zur Laufzeit beschreiben zu können, sind fünf Aspekte zu betrachten:

AlConPres Der `Presence`-Zustand der Alarmbedingung gibt an, ob die Alarmbedingung zum aktuellen Zeitpunkt erfüllt (`true`) oder nicht erfüllt ist (`false`):
AlConPres $\in$ {`true, false`}

RemoteAlSigAct Der `ActivationState` des *entfernten* Alarmsignals gibt an, ob das Alarmsignal zum aktuellen Zeitpunkt betriebsbereit (`On`), nicht betriebsbereit (`Off`) oder pausiert (`Paused`) ist:
RemoteAlSigAct $\in$ {`On, Off, Paused`}

LocalAlSigAct Für den `ActivationState` des *lokalen* Alarmsignals gilt analog zum `ActivationState` des *entfernten* Alarmsignals:
LocalAlSigAct $\in$ {`On, Off, Paused`}

RemoteAlSigPres Der `Presence`-Zustand des *entfernten* Alarmsignals gibt den Generierungszustand des Alarmsignals zum aktuellen Zeitpunkt an: Generierung aktiv (`On`), keine Generierung (`Off`), keine Generierung aufgrund einer vorliegenden Bestätigung (`Acknowledged`), Generierung aktiv obwohl die Alarmbedingung nicht mehr aktiv ist (`Latched`):
RemoteAlSigPres $\in$ {`On, Off, Latched, Acknowledged, X`}

LocalAlSigPres Für den `Presence`-Zustand des *lokalen* Alarmsignals gilt analog zum `Presence`-Zustand des *entfernten* Alarmsignals:
LocalAlSigPres $\in$ {`On, Off, Latched, Acknowledged, X`}

Ist ein Alarmsignal nicht aktiv, beschrieben durch den `ActivationState` `Off` oder `Paused`, so ist der `Presence`-Zustand dieses Alarmsignals für das Konzept irrelevant bzw. wird normativbedingt nicht interpretiert. Dies wird durch den Wert `X` in den beschriebenen Wertemengen ausgedrückt.

9.3 Anhaltend beschleunigter Puls; umgangssprachlich auch als Herzrasen bezeichnet.

9.4 `Latched` beschreibt den Zustand, dass ein Alarm aktiv signalisiert wurde, zum aktuellen Zeitpunkt die Alarmbedingung aber nicht mehr erfüllt ist.

Folglich kann der relevante Teil des Systemzustands eines *PAS* durch ein 5-Tupel beschrieben werden, das in Abbildung 9.2 zur Veranschaulichung genutzt wird:

$$PAS = (AlConPres, LocalAlSigAct, LocalAlSigPres, RemoteAlSigAct, RemoteAlSigPres) \quad (4)$$

9.4.3 Systemverhalten zur Laufzeit im regulären Betrieb

Die beschriebene zweite Grundidee besagt, dass ein *ESG* dem *PAS* regelmäßig seinen aktuellen Zustand mitteilt. Entsprechend der Anforderung 6 soll weder ein *ESG* gezwungen werden, die Rolle eines *Service Providers* zu implementieren, noch ein *PAS* die Rolle eines *Service Consumers*. Daher muss das *PAS* in seiner Rolle als *Service Provider* ermöglichen, diese Informationen vom *ESG* (*Service Consumer*) zu bekommen. Hierfür bietet das *PAS* eine entsprechende *Set Alert State Operation* im Netzwerk an. Diese erlaubt es dem *ESG*, relevante Parameter des *entfernten* Alarmsignals zu manipulieren und damit dem *PAS* Informationen bereit zu stellen. Für das Konzept sind die Parameter `ActivationState`, `Presence` und `ActualSignalGenerationDelay`[9.5] von Interesse.

In Abbildung 9.2 wird der konzeptuelle Ablauf dargestellt. Wenn der *ESG* beabsichtigt, das *entfernte* Alarmsignal zu generieren, so teilt er dies dem *PAS* mit. Dies erfolgt, indem er mittels der *Set Alert State Operation* den `ActivationState` auf `On` setzt. Gleichzeitig setzt er das `ActualSignalGenerationDelay` entsprechend seinen Fähigkeiten, aber innerhalb der Vorgaben. Ab diesem Zeitpunkt ist der *ESG* für die Alarmsignalgenerierung verantwortlich. Entsprechend deaktiviert das *PAS* sein *lokales Fallback-Alarmsignal* und teilt dies im Netzwerk mit, indem es den zugehörigen `ActivationState` auf `Off` setzt.

Die Wirksamkeit der ausgelösten *Set Alert State Operation* ist zeitlich begrenzt. Das *PAS* definiert diese Zeitspanne mittels des Parameters `InvocationEffectiveTimeout` in der Operationsbeschreibung als sogenannte maximale *Confirmation Time* (maximale Bestätigungszeit). Innerhalb dieser Zeitspanne löst der *ESG* die Operation erneut aus und bestätigt damit, weiterhin für die Alarmsignalgenerierung verantwortlich zu sein. Mit diesem periodischen Mechanismus kann das *PAS* permanent beurteilen, ob der *ESG* verfügbar ist und entsprechend arbeitet. Die erste Schleife im oberen Teil von Abbildung 9.2 veranschaulicht diesen Ablauf.

Detektiert das *PAS* die Erfüllung der Alarmbedingung, so setzt es das `Presence`-Attribut der Alarmbedingung auf den Wert `true` (*AlConPres* = `true`). Dies wird im Geräteensemble mittels einer entsprechenden Eventnotifikation nach dem *Publish-Subscribe*-Prinzip an alle Teilnehmer, die diese Events abonniert haben, publiziert. Somit hat auch der verantwortliche *ESG* Kenntnis davon, dass die Alarmbedingung erfüllt ist und generiert innerhalb der angegebenen Verzögerung das *entfernte* Alarmsignal. Die erfolgreiche Alarmsignalgenerierung teilt der *ESG* dem *PAS* mit, indem er das `Presence`-Attribut des *entfernten* Alarmsignals auf den Wert `On` setzt (*RemoteAlSigPres* = `On`). Dies erfolgt ebenfalls mittels der *Set Alert State Operation*. Entsprechend dem Grundprinzip wird auch die tatsächliche Generierung periodisch, innerhalb der *Confirmation Time*, bestätigt. Dieser Ablauf ist im mittleren Teil von Abbildung 9.2 dargestellt.

Das *PAS* benötigt für die Risikobewertung ggf. zusätzliche, meist kontextuelle, Informationen vom *ESG*. In diesem Fall nutzt das *PAS* die Mechanismen des *MDPWS Safety Contexts* (siehe auch Abschnitt 4.3.1). Beispielsweise kann das *PAS* fordern, dass der *ESG* Ortsinformationen in das Auslösekommando der *Set Alert State Operation* einzubetten hat. So kann das *PAS* etwa überwachen, dass die Alarmsignalgenerierung an einem bestimmten Ort bzw. in einer akzeptablen Entfernung erfolgt.

[9.5] Gibt die tatsächliche Verzögerung zwischen der Generierung der Alarmbedingung und der Generierung des Alarmsignals zum aktuellen Zeitpunkt an.

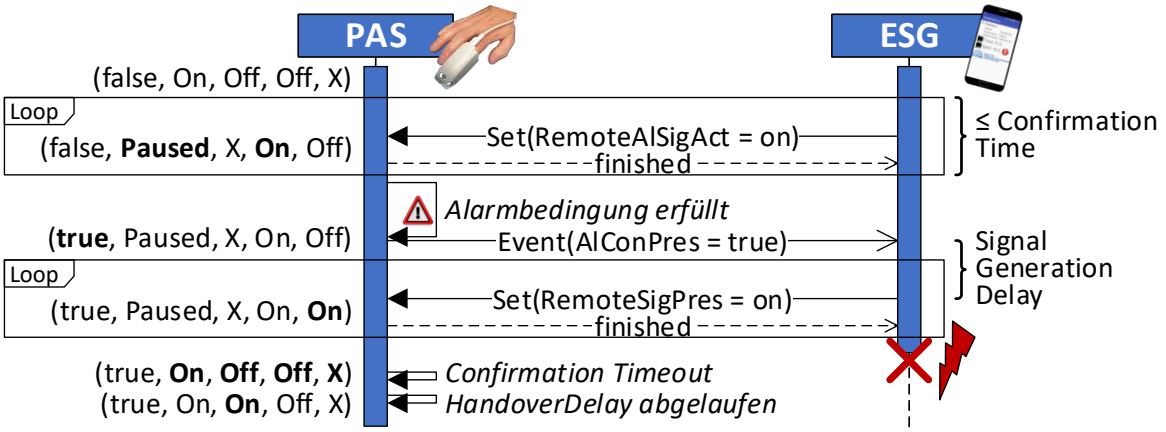

Abbildung 9.2: UML-Sequenzdiagramm zum Konzept des *verteilten Alarmierungssystems*. Systemzustand des *primären Alarmsystems (PAS)* wird im linken Teil als 5-Tupel, entsprechend Formel 4, dargestellt. Geänderte Zustände sind hervorgehoben. Abkürzung: *ESG – Entfernter Alarmsignalgenerator*. (Abbildung nach *Fig. 1* aus [B 15]. Original © 2017 IEEE.)

Entsprechend den allgemeinen Alarmmechanismen der *IEEE 11073 SDC*-Normenfamilie können Alarmsignale auch im *verteilten Alarmierungssystem* von den Nutzerinnen für eine bestimmte Zeit bestätigt werden (*RemoteAlSigPres* = `Acknowledged`) werden. Ebenso kann das Alarmsignal in den Zustand `Latched` übergehen, wenn die Alarmbedingung vom Zustand erfüllt in nicht erfüllt übergeht, ohne dass das Alarmsignal bestätigt wurde (*RemoteAlSigPres* = `Latched`).

Das Zustandsdiagramm in Abbildung 9.3 stellt die relevanten Zustände und Zustandsübergänge des *PASs* dar. Der linke Teil des Zustandsdiagramms beschreibt den Zustand, in dem das *lokale* Alarmsignal aktiv ist, während der rechte Teil die Aktivierung des *entfernten* Alarmsignals darstellt. Vertikal dargestellte Transitionen beschreiben Werteänderungen der `Presence`-Attribute von Alarmbedingung und Alarmsignalen. Das Alarmsignal hat dabei entsprechend der Alarmbedingung zu folgen, unter der Berücksichtigung von ggf. getätigten Alarmbestätigungen. Zustandsübergänge, die horizontal dargestellt sind, verdeutlichen den Wechsel zwischen *entfernter* Alarmsignalgenerierung und *lokaler Fallback*-Alarmsignalgenerierung. Transitionen von links nach rechts werden vom *ESG* angestoßen, indem die entsprechende *Set Alert State Operation* ausgelöst wird. Übergänge von rechts nach links werden durch *Confirmation Timeouts* des *PASs* ausgelöst. Entsprechend erfolgt der Rückfall in die *lokale Fallback*-Alarmsignalgenerierung. Dies stellt zumeist einen Fehlerfall dar, dessen Erkennung im folgenden Abschnitt 9.4.4 beschrieben wird.

9.4.4 Systemverhalten zur Laufzeit in Ausnahmefällen

Um der dritten Anforderung (siehe Abschnitt 9.3) zu genügen, muss das *PAS* in der Lage sein, Fehler und Ausfälle des *ESGs* zu erkennen. Dies gilt ebenfalls für Netzwerkprobleme, wie beispielsweise Verbindungsabbrüche, inakzeptabel hohe Übertragungsverzögerungen etc. Die Erkennung erfolgt mittels der periodischen Bestätigungen und des *Confirmation Timeouts*. Der *Timeout* wird dann ausgelöst, wenn keine entsprechende Nachricht vom *ESG* innerhalb der definierten Zeit vom *PAS* empfangen wird (siehe unteren Teil von Abbildung 9.2). Aufgrund der ausbleibenden Notifizierung muss das *PAS* davon ausgehen, dass ein Fehler vorliegt und entsprechend seinem Risikomanagement handeln. Der typische *Fallback*-Mechanismus besteht darin, die Alarmsignalgenerierung selbst *lokal* vorzunehmen. Entsprechend setzt das *PAS* das `Presence`-Attribut für das *lokale* Alarmsignal auf `On` (*LocalAlSigAct* = `On`) und für das *entfernte* Alarmsignal auf `Off` (*RemoteAlSigAct* = `Off`).

Mit der periodischen Bestätigung des *ESGs* teilt dieser dem *PAS* auch den Zustand der tatsächlichen Alarmsignalgenerierung mittels des `Presence`-Attributes mit. Somit kann das *PAS* die Korrektheit

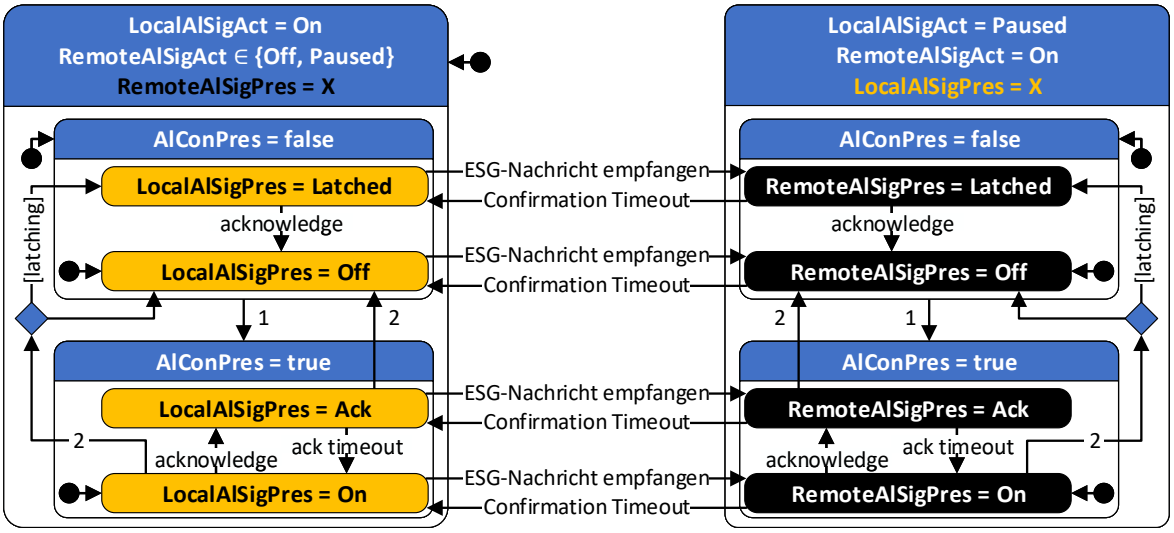

Abbildung 9.3: UML-Zustandsdiagramm des *primären Alarmsystems (PAS)*, das den relevanten Teil der Zustände und Zustandsübergänge beschreibt. Legende: 1 – Erfüllung der Alarmbedingung erkannt; 2 – Alarmbedingung nicht (mehr) erfüllt. Abkürzung: Ack – Acknowledgement (Bestätigung eines Alarmsignals); *ESG – Entfernter Alarmsignalgenerator*. (Abbildung nach *Fig. 2* aus [B 15]. Original © 2017 IEEE.)

der Generierung überwachen. Arbeitet das *ESG* nicht korrekt, kann das *PAS* entsprechend reagieren und die *lokalen Fallback*-Mechanismen auslösen. Damit erfüllt der beschriebene Mechanismus die vierte Anforderung.

Um den Anforderungen der *IEC 60601-1-8* [A 6] zu genügen, implementiert das *PAS* technische Alarme, die das Vorhandensein von Fehlersituationen anzeigen. Dasselbe gilt für die *ESGs*. Wird die periodische Auslösung der *Set Alert State Operation* vom *PAS* nicht bestätigt, so muss der *ESG* von einer Fehlersituation ausgehen und generiert einen entsprechenden technischen Alarm. Aus Gründen der Übersichtlichkeit ist dieses Event in Abbildung 9.2 nicht enthalten.

9.4.5 Geräteensembles mit mehreren ESGs und PASs

9.4.5.1 Verantwortlichkeitsprinzip bei mehreren ESGs In einem Geräteensemble mit mehreren *ESGs* arbeitet nur ein *ESG* strikt nach dem beschriebenen periodischen Benachrichtigungsmechanismus. Dieser wird als *verantwortlicher ESG* bezeichnet. Das *PAS* kann für diesen *ESG* die korrekte Generierung überwachen. Die anderen *ESGs* abonnieren ebenfalls die Events des *PASs*. Sie erhalten folglich ebenso alle Informationen über die Alarmbedingungen und Alarmsignale. Mittels der Events zur Abarbeitung von externen Operationen bekommen die *ESGs* auch indirekt Informationen über das Verhalten des *verantwortlichen ESGs*. Einerseits erlaubt dieser Mechanismus die *entfernte* Alarmgenerierung von mehreren *ESGs*, potentiell an verschiedenen Orten. Andererseits reduziert das Prinzip des einzelnen, *verantwortlichen ESGs* die Komplexität des *PASs*, da nur das Verhalten einer entfernten Instanz überwacht werden muss. Es muss teilweise von (sehr) ressourcenbeschränkten alarmüberwachenden Geräten, wie beispielsweise einem einfachen Pulsoximeter, ausgegangen werden. Daher ist diese Komplexitätsreduktion eine wichtige Eigenschaft des hier vorgestellten Konzepts.

Um den Vorgaben des Risikomanagements bestimmter Anwendungen gerecht zu werden, kann es notwendig sein, dass die Alarmierung durch mindestens zwei (oder mehr) *ESGs* erfolgen muss, die sich ggf. an bestimmten Orten befinden müssen. In einem solchen Fall würde das *PAS* mehrere *entfernte* Alarmsignale für dieselbe Alarmbedingung bereitstellen und überwachen. So können verschiedene *ESGs* die Signalgenerierung in der Rolle eines *verantwortlichen ESGs* übernehmen.

Es sei angemerkt, dass die Verarbeitung von *Acknowledgements*, die von *ESGs* gesandt werden, die nicht die *verantwortlichen ESGs* sind, dem Risikomanagement des *PASs* unterliegt. Stehen dem *PAS* nicht alle notwendigen Informationen zur Verfügung, so kann es den *Acknowledgement*-Befehl zurückweisen. Andernfalls akzeptiert das *PAS* das *Acknowledgement* und publiziert diesen neuen Zustand entsprechend im Geräteensemble.

9.4.5.2 Handover-Mechanismus bei mehreren ESGs Wie im unteren Teil von Abbildung 9.2 veranschaulicht ist, kann das *PAS* die *lokale* Alarmsignalgenerierung nach dem *Confirmation Timeout* verzögern. Diese Verzögerung erlaubt es einem anderen *ESG*, die Verantwortung für die *entfernte* Alarmsignalgenerierung zu übernehmen, ohne dass eine *lokale* Generierung vorgenommen wird. Somit kann auf eine unnötige Alarmierung in Patientennähe verzichtet werden. Das *Handover Delay* kann aus der Signalgenerierungsverzögerung und der maximalen *Confirmation Time* (`InvocationEffectiveTimeout`) der *Set Alert State Operation* berechnet werden. Das *PAS* hat diese Zeit so zu wählen, dass die *lokale Fallback*-Alarmierung trotz *Confirmation Time* und *Handover Delay* innerhalb einer akzeptablen Zeitspanne erfolgt.

9.4.5.3 Geräteensembles mit mehreren PASs In einem *IEEE 11073 SDC*-Netzwerk agieren die *Service Provider* unabhängig voneinander. Daher können in einem Geräteensemble beliebig viele *Service Provider* als *PAS* auftreten. Ebenso kann ein *Service Consumer* für mehrere *PASs* die Rolle des (*verantwortlichen*) *ESGs* übernehmen. Beide Szenarien sind im realen Einsatz als sehr wahrscheinlich anzusehen. Beispielweise kann eine zentrale Arbeitsstation als *ESG* für verschiedene *PASs*, wie etwa Pulsoximeter, Patientenmonitor, Spritzenpumpenturm etc., dienen. Die Mensch-Maschine-Interfaces (MMIs) der *ESGs* haben entsprechend sicherzustellen, dass die Zuordnung der Alarmsignale zu den *PASs* und damit zu den Patientinnen für alle Nutzenden klar ist. Somit erfüllt das Konzept die Anforderung 2 (siehe Abschnitt 9.3).

9.5 Demonstrator

Um die Umsetzbarkeit und Leistungsfähigkeit des beschriebenen Konzepts zu zeigen, wurde ein prototypischer Demonstrator entwickelt, der in Abbildung 9.4 dargestellt ist. Dieser besteht aus zwei *primären Alarmsystemen* (*PASs*), einem Pulsoximeter und einem EKG auf der Basis eines Entwicklungsboards für Vitalparametermessungen, und zwei *entfernten Alarmsignalgeneratoren* (*ESGs*), einer PC-Applikation und einer Smartphone-Applikation inkl. Smartwatch. Über die Präsentation des Papers hinaus wurde der Demonstrator etwa im Zuge der *IEEE HI-POCT* Konferenz 2017 als *Translational Demo Showcase* vorgestellt und mit den internationalen Experten diskutiert.

Das Pulsoximeter stellt die Werte und grenzwertbasierte Alarme für die Herzfrequenz und die Sauerstoffsättigung (S_pO_2[9.6]) bereit. Das EKG-Entwicklungsboard leitet aus der EKG-Kurve ebenfalls die Herzfrequenz ab und publiziert den entsprechenden Alarm und Wert im Netzwerk. Die PC-Applikation steht stellvertretend für zentrale Arbeitsplätze im OP-Saal oder der ITS. Die Smartphone-Applikation repräsentiert mobile Endgeräte, die Ärzte und Pflegepersonal direkt bei sich tragen. Die Smartwatch ist in dieser Implementierung direkt mit dem Smartphone gekoppelt und ist damit kein eigenständiger Teilnehmer im *IEEE 11073 SDC*-Netzwerk. Sie stellt lediglich eine Erweiterung der Benutzerschnittstelle des Smartphones dar. Für die Behandlung von Verbindungsabbrüchen und anderen Fehlern zwischen Smartphone und Smartwatch ist das Smartphone verantwortlich. Für diese Verbindung wird kein *IEEE 11073 SDC* genutzt.

Zur Implementierung der Netzwerkrepräsentation des Pulsoximeters wurde die *openSDC*-Bibliothek [A 161] genutzt. Das EKG-Entwicklungsboard und die PC-Applikation nutzen die Biblio-

[9.6] Pulsoximetrisch gemessene, quasi-arterielle, Sauerstoffsättigung (S_pO_2)

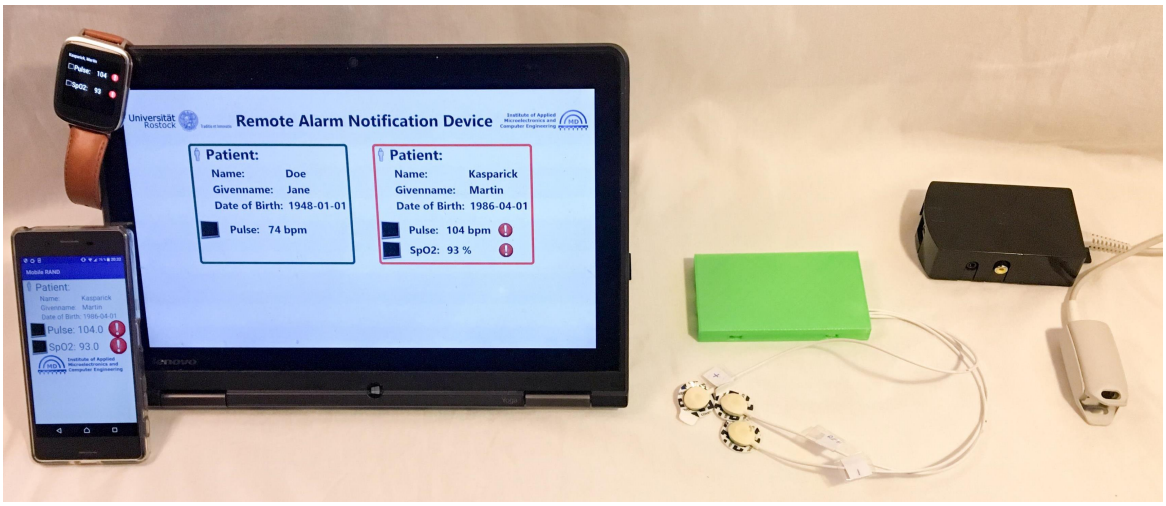

Abbildung 9.4: Demonstrator des *verteilten Alarmierungssystems* – v.l.n.r.: Zwei *entfernte Alarmsignalgeneratoren (ESGs)*, Smartphone inkl. Smartwatch und PC-Applikation, und zwei *primäre Alarmsysteme (PASs)*, EKG-Entwicklungsboard und Pulsoximeter. (Abbildung entspricht *Fig. 3* aus [B 15]. © 2017 IEEE.)

thek *SDCLib/J* [A 162, 163]. Für die Smartphone-Applikation kommt die *JMEDS*-Bibliothek [A 248] zum Einsatz[9.7].

Beide *PASs* haben die Möglichkeit, Alarme akustisch und visuell zu generieren. Entsprechend implementieren sie die *entfernten* Alarmsignale und *lokalen Fallback*-Alarmsignale. Für Demonstrationszwecke wird zusätzlich ein haptisches Alarmsignal bereitgestellt. Dieses kann nur *entfernt* generiert werden, da die *PASs* keine Möglichkeit haben, ein solches Signal *lokal* zu erzeugen. Für den realen Einsatz muss untersucht werden, für welche Anwendungsbereiche haptische Alarmsignale für mobile Endgeräte eingesetzt werden können.

Die technische Evaluation in Form eines Demonstrators hat die Eignung des vorgestellten Konzepts gezeigt. So kann die *entfernte* Alarmsignalgenerierung zuverlässig demonstriert werden. Ebenso lässt sich das Verhalten im Fehlerfall, beispielsweise durch das Stören/Abschalten des drahtlosen Netzwerks oder das Abschalten von *ESGs*, zeigen. Trotz der geringen Komplexität dieses Demonstrators lassen sich die Ergebnisse auf Geräteensembles der heutigen und zukünftigen Komplexität übertragen, da die genutzte *IEEE 11073 SDC*-Vernetzung ihre Eignung für hochkomplexe Geräteverbünde bereits gezeigt hat. So sind beispielsweise die komplexen Demonstratoren mit einer Vielzahl von Geräten von unterschiedlichen Herstellern im Zuge der *conhIT*-, *DMEA*- und *Medica*-Messen oder die Abschlussdemonstratoren des *OR.NET*-Projekts zu nennen.

In Zukunft sollten klinische Evaluationen erfolgen. Bisher wurde lediglich eine unstrukturierte Evaluation mit sechs Ärztinnen (drei Anästhesistinnen/Anästhesisten, zwei Chirurgen und einem Internisten) durchgeführt. Alle gaben positives bzw. sehr positives Feedback und stellten das Potential eines solchen Systems heraus. Eine Befragung einer derart kleinen Benutzergruppe kann aber lediglich einen Hinweis auf die klinische Relevanz und Eignung geben und stellt keine klinische Evaluation dar.

9.7 Die *JMEDS*-Bibliothek implementiert das *Devices Profile for Web Services (DPWS)*, welches *IEEE 11073 SDC* zugrunde liegt. *JMEDS* stellt damit keine medizingerätespezifischen Erweiterungen bereit. Aufgrund der Kompatibilität zum Entwicklungsprozess für Smartphone-Apps auf einem Android-System wurde dieses generische Framework genutzt.

9.6 Abgleich mit den Anforderungen der Norm IEC 60601-1-8

Die Norm *IEC 60601-1-8* [A 6] (hier bezogen auf die Version *Amendment 1* von 2012) stellt eine Reihe von Anforderungen an *verteilte Alarmsysteme*. In diesem Abschnitt erfolgt ein kompakter Abgleich dieser Anforderungen mit den Fähigkeiten des vorgestellten Systems. Es wird dabei nur auf die Anforderungen eingegangen, die in der Norm *IEC 60601-1-8* [A 6] mit direktem Bezug zu *verteilten Alarmsystemen* gekennzeichnet sind. Die generellen Anforderungen an *Alarmsysteme* sind unabhängig von einer vernetzungsbasierten Lösung und daher nicht im Fokus dieser Arbeit.

Da die Norm nicht frei zitiert werden kann, werden im Zuge dieses Dokuments lediglich generische Formulierungen der Anforderung genutzt. Die Anforderungen werden in der Reihenfolge des Auftretens in der Norm betrachtet. Dies spiegelt nicht die Relevanz der einzelnen Anforderungen wider.

Unterabschnitt 6.4 „Offenlegung von Verzögerungen“. Im Speziellen: Unterabschnitt 6.4.2: „Verzögerungen zu oder von einem *verteilten Alarmsystem*“

Anforderung Die Norm fordert die Offenlegung der maximalen bzw. mittleren Verzögerungszeit der Alarmsignalgenerierung in der Gebrauchsanweisung.

Realisierung Die unmittelbare Erfüllung dieser Anforderung liegt außerhalb des Fokus dieser Arbeit, da sie grundlegend in Dokumenten wie dem Benutzerhandbuch erfolgt. Dennoch werden die Informationen über die auftretenden Verzögerungen über die Gebrauchsanweisung hinaus in maschineninterpretierbarer Form im Netzwerk via *IEEE 11073 SDC* als Teil der Geräte- bzw. Gerätezustandsbeschreibung bereitgestellt.

Unterabschnitt 6.7 „Alarmsystem-Sicherheit“

Anforderung Eine Manipulation durch Unbefugte ist zu verhindern.

Realisierung Die *IEEE 11073 SDC*-Normenfamilie, bzw. zugrundeliegende Standards, stellt Mechanismen zur Authentifizierung und Autorisierung bereit. Für weitergehende Einschränkungen, z. B. mittels kontextueller Informationen wie etwa dem Ort eines Teilnehmers, können die Mechanismen des *Safety Contexts* (*IEEE 11073-20702*) genutzt werden.

Unterabschnitt 6.11.1 „Vorhandensein eines *verteilten Alarmsystems*“

Anforderung *Verteilte Alarmsysteme* sind zulässig. Das *verteilte Alarmsystem*, bzw. Teile dieses dürfen sich außerhalb der Patientenumgebung befinden. Die Verbindung der Teilnehmer kann über beliebige, geeignete Kommunikationstechnologien erfolgen.
Notwendige technische Details sind in der technischen Beschreibung offen zu legen.

Realisierung Dieser Unterabschnitt stellt die Grundlage für das vorgestellte *verteilte Alarmierungssystem* dar und erlaubt grundlegend den Einsatz in Medizinprodukten. Über die Anforderung hinaus werden die technischen Details in maschineninterpretierbarer Form zur Laufzeit im Netzwerk via *IEEE 11073 SDC* bereitgestellt.

Unterabschnitt 6.11.2.1 „Quelle und Identifikation von Alarmbedingungen“

Anforderung Die Quelle der Alarmbedingung muss an jedem Ort der Alarmsignalgenerierung identifizierbar sein. Des Weiteren sollten Ursache, Patienten- und Gerätezuordnung kommuniziert werden.

Realisierung Die *IEEE 11073 SDC*-basierte Kommunikation stellt diese Informationen bereit. Die konkrete Ausgestaltung, insbesondere die Umsetzung in der Benutzerschnittstelle obliegt den einzelnen Produkten und kann nicht in einem allgemeinen Konzept betrachtet werden. Es können Mittel der Normenfamilie wie der *Safety Context* genutzt werden, ebenso wie weiterführende Konzepte, z. B. das *Session*-Konzept (siehe Abschnitt 6).

Unterabschnitt 6.11.2.2 „Ausfall der Übermittlung von Alarmbedingungen an *entfernt* aufgestellte Geräte“

Anforderung Ein *verteiltes Alarmsystem* muss sicherstellen, dass der Ausfall der Informationsübermittlung oder ein Ausfall irgendeines Teilnehmers
a) „keinen Teil des *verteilten Alarmsystems* negativ beeinflusst, ausgenommen den Verlust der *verteilten* Funktionalität“, und
b) eine technische Alarmbedingung am betroffenen *PAS* und an jedem betroffenen *ESG* auslöst.

Realisierung Um der Teilanforderung a) zu genügen, ist in der Entwicklung des einzelnen Medizinprodukts sicherzustellen, dass keine Abhängigkeiten vom *verteilten Alarmsystem* andere Funktionaltäten beeinflussen. Das vorgestellte Konzept beinhaltet lediglich die Erkennung von Fehlersituationen, während die korrekte und sichere Reaktion in der Verantwortung der einzelnen Produkte liegt.
Die Grundlage zur Umsetzung der Teilanforderung b) ist die sichere Erkennung von Fehlersituationen. Dies ist für das verantwortliche *PAS* und die *ESGs* beschrieben. Entsprechende technische Alarme, die über die erkannten Fehlersituationen informieren, sind zu implementieren.
Unterabschnitt 6.11.2.2.2 (System ohne Übermittlungsbestätigung) ist für das vorgestellte System nicht zutreffend, da die Bestätigung nach Unterabschnitt 6.11.2.2.1 eines der Grundkonzepte darstellt.
Die spezielle Anforderung nach Unterabschnitt 6.11.2.2.3 (Beendigung einer generellen Deaktivierung von akustischen Alarmsignalen im Fehlerfall) kann auf der Basis der bereitgestellten Informationen erfüllt werden.

Unterabschnitt 6.11.2.3 „Bedienelemente für ein Alarmsystem mit *entfernt* aufgestellten Geräten“

Anforderung In einem *verteilten Alarmsystem* darf ein Zugriff auf Bedienelemente von *entfernten* Teilnehmern vorgesehen werden. Ist dies der Fall, so sind Mechanismen zur Zugriffsbeschränkung vorzusehen.

Realisierung Die *IEEE 11073 SDC*-Mechanismen zur Fernsteuerung von Vernetzungspartnern ermöglichen den entfernten Zugriff auf Bedienelemente. Mechanismen zur Zugriffsbeschränkung sind vorgesehen und sind von den einzelnen Produkten umzusetzen.

Entsprechend dem Abgleich kann zusammengefasst werden, dass das präsentierte Konzept den Anforderungen der *IEC 60601-1-8* [A 6] (Stand 2012) genügt, indem es den Herstellern die entsprechenden Mittel zur Erfüllung bereitstellt.

9.7 Diskussion

In diesem Abschnitt wurde ein zuverlässiges, *verteiltes Alarmierungssystem* auf der Basis von *IEEE 11073 SDC* vorgestellt. Die Eignung für Anwendungen in klinischen Umgebungen wurde mittels eines Demonstrators mit realen Geräten gezeigt. Das beschriebene Konzept ist in den inzwischen verabschiedeten Standard *IEEE 11073-10207* eingeflossen und ist damit direkt von den Herstellern nutzbar.

Das *verteilte Alarmierungssystem* ist in der Lage, Alarmsignale durch *entfernte* Netzwerkteilnehmer zu generieren. Die *entfernte* Alarmsignalgenerierung wird durch *lokale Fallback*-Alarmsignale abgesichert. Das Konzept beinhaltet die Nutzung von mobilen *entfernten Alarmsignalgeneratoren* auch über potentiell unzuverlässige, drahtlose Netzwerkverbindungen. Damit kann die Alarmsignalgenerierung an einem Ort erfolgen, an dem es für Ärzte und Pflegepersonal sinnvoll ist. Auf Intensiv- oder Wachstationen ist dies häufig nicht in unmittelbarer Patientennähe. Folglich leistet das Konzept

einen Beitrag, den grundlegenden Problemen der Alarmmüdigkeit und Desensibilisierung entgegenzuwirken. Somit rückt etwa die Vision einer „Silent ICU“[9.8] in greifbare Nähe. Im OP-Saal kann insbesondere die Möglichkeit der zuverlässigen entfernten Alarmbestätigung Verbesserungen im Alltag bringen.

Es befinden sich bereits kommerzielle Lösungen für *verteilte Alarmsysteme* auf dem Markt. Diese stellen aber geschlossene Systeme auf der Basis von proprietären Schnittstellen dar. Die Innovation des vorgestellten Systems besteht daher in der Interoperabilität und Herstellerunabhängigkeit bei gleichzeitiger Bereitstellung desselben Funktionsumfangs.

Das Konzept setzt auf *IEEE 11073 SDC* auf. Die Realisierung erfolgt ausschließlich auf der Basis von Informationen, die zur Laufzeit zwischen den Vernetzungspartnern ausgetauscht werden bzw. in der Normenfamilie spezifiziert sind. Daher können die teilnehmenden Geräte des *verteilten Alarmierungssystems* herstellerunabhängig entwickelt, *in den Verkehr gebracht* und betrieben werden. Das *primäre Alarmsystem*, das die Alarmbedingungen überwacht und *lokale* Alarmsignale generieren kann, und die *entfernten Alarmsignalgeneratoren* sind damit unabhängig voneinander, sodass das Konzept eine hohe Flexibilität und Erweiterbarkeit ermöglicht.

Im Gegensatz zu den Analysen von Konkani et al. [A 39] weist das präsentierte Konzept keine *Single Points of Failure (SPoF)* auf. Damit sind alle in Abschnitt 9.3 gestellten Anforderungen erfüllt. Darüber hinaus konnte gezeigt werden, dass das System den Anforderungen der *IEC 60601-1-8* [A 6] an ein *verteiltes Alarmsystem* genügt (siehe Abschnitt 9.6).

Das vorgestellte *verteilte Alarmierungssystem* entspricht den Risikomanagement- und Zulassungsvorgaben und widerspricht damit nicht der derzeit vorherrschenden „Better-Safe-Than-Sorry-Mentalität“. Viele vielversprechende Forschungsergebnisse der letzten Jahre auf dem Gebiet der Alarmsysteme haben nicht den Schritt in am marktbefindliche Produkte geschafft und konnten damit die Probleme in heutigen Krankenhäusern nicht lösen [A 40, 246]. Das vorgestellte Konzept löst grundlegende technische Herausforderungen von *verteilten Alarmsystemen*. Daher kann das Potential zur Nutzung in zukünftigen Medizinprodukten als hoch angesehen werden. Zudem kann es als Grundlage für weitere Entwicklungen von Alarmsystemen mit zusätzlichen, intelligenten Assistenzfunktionen dienen.

In zukünftigen Arbeiten können komplexere *verteilte Alarmierungssysteme* mit einem größeren Umfang an teilnehmenden Geräten und Alarmen realisiert werden. Zudem sollte eine umfangreiche klinische Evaluation vorgenommen werden. Auf der Basis des vorgestellten Konzepts können intelligente, computerassistierte Alarmsysteme entwickelt werden, die den Sicherheitsanforderungen genügen und sich an den Bedarfen der Nutzenden und Patientinnen orientieren. Diese Arbeit stellt damit eine Basistechnologie zur Verfügung, die sowohl direkt zum Einsatz kommen kann als auch die Grundlage für weitere Innovationen bietet.

9.8 „Silent ICU“ – sinngemäß für *lautlose Intensivstation (ITS)*.

10 Abschluss

10.1 Zusammenfassung und Ergebnisse

Technisierung und Digitalisierung spielen – wie in vielen Bereichen des Lebens – auch im Krankenhaus eine immer größere Rolle. Im Alltag ist es schon fast eine Selbstverständlichkeit, dass Daten zwischen Handys, Fernsehern, Musikanlagen, Autos, digitalen Bilderrahmen, Kühlschränken etc. im persönlichen Umfeld ausgetauscht werden. Heutige Medizingeräte sind hingegen meist nicht vernetzt oder nur als Teil einer proprietären Insellösung eines Herstellers. Große Potentiale zur Verbesserung von Patientensicherheit und klinischen Arbeitsabläufen bleiben somit ungenutzt. Die vorliegende Arbeit leistet daher einen Beitrag zur zuverlässigen und herstellerübergreifenden Medizingeräteinteroperabilität auf der Basis der neuen *IEEE 11073 SDC*-Normenfamilie. Der Fokus liegt auf dem Operationssaal (OP-Saal) und der Intensivstation (ITS). Die Konzepte sind auf alle *Point-of-Care (PoC)*-Medizingeräte im Krankenhaus anwendbar.

Im Zuge der zugrunde liegenden Arbeiten dieser Dissertation war und ist der Autor an der Erstellung der *IEEE 11073 SDC*-Normenfamilie als Co-Autor beteiligt. Erkenntnisse aus den Forschungsarbeiten fließen direkt und indirekt in die Normung ein. Normen stellen einen wichtigen Transfer aus der Wissenschaft in die Praxis dar, insbesondere wenn sie, wie im vorliegenden Fall, in enger Kooperation zwischen Forschungseinrichtungen und Industrie entstehen und sich an den Bedarfen der Anwender orientieren. Gerade für die deutsche Medizintechnikindustrie, die geprägt ist von kleinen und mittleren Unternehmen (KMUs), stellen normierte Vernetzungstechnologien ein enormes wirtschaftliches Potential dar. Eine Liste der Normen bzw. Normungsprojekte, die unter Beteiligung des Autors entstanden sind bzw. entstehen, befindet sich in Anhang D.

In den Kapiteln 3 und 4 dieser Arbeit werden die Grundlagen der herstellerübergreifenden Medizingerätevernetzung beschrieben. Den konzeptuellen Unterbau liefert die Weiterentwicklung von der *Service-Oriented Architecture (SOA)* zur *Service-Oriented Medical Device Architecture (SOMDA)*. Kapitel 4 beschäftigt sich mit der *IEEE 11073 SDC*-Normenfamilie. Es erfolgt eine intensive Auseinandersetzung mit den *Kernnormen*. Die Performanzevaluation (Kapitel 5) von Open-Source-Bibliotheken, die *IEEE 11073 SDC* implementieren, zeigt, dass die Konzepte der Normenfamilie leistungsfähig genug für eine praktische Umsetzung sind. Mit dem *OR.NET-Session-Konzept* wird in Kapitel 6 ein allgemeines Konzept für die Erzeugung und Verwaltung von spezifischen Medizingeräteensembles für eine spezifische Behandlung einer konkreten Patientin vorgestellt.

Die Sicherheit der herstellerübergreifenden Medizingeräteinteroperabilität im Sinne der *Safety* ist ein zentrales Thema der Arbeit. Die Kapitel 7 und 8 beschäftigen sich mit Aspekten der zuverlässigen Fernsteuerung. Den ersten Themenbereich stellt die Auslösung von Gerätefunktionalitäten dar, wie beispielsweise der Energieabgabe eines HF-Chirurgiegerätes, der Schneidefunktion eines Ultraschall-Knochenmessers, der Saug- oder Spülfunktion einer Pumpe oder der Lichtabgabe einer Lichtquelle. Es werden Mechanismen vorgestellt, die eine solche Auslösung über Geräte- und Herstellergrenzen hinweg zuverlässig ermöglichen, auch wenn von einem potentiell unzuverlässigen Netzwerk auszugehen ist. Über das Grundprinzip einer periodischen Re-Aktivierung hinaus werden Erweiterungen zur Erhöhung der Zuverlässigkeit beschrieben.

Der zweite Fokus im Bereich der Fernsteuerung wird auf die dynamische Zuordnung von Steuerelementen zu Fernsteuerungsoperationen von steuerbaren Geräten und der entsprechenden Auslösung gesetzt. Steuerelemente sind zum Beispiel Fußschalter oder Schalter an Handgriffen von Endoskopen oder Mikroskopen. Zu Fernsteuerungsoperationen zählt etwa die Konfiguration von Geräteparametern, beispielsweise der Fräsendrehzahl, maximal abgegebenen Leistung eines HF-Chirurgiegerätes, Lichtintensität etc. Auf der Basis der *Service-Orchestrierung* wird in Kapitel 8 ein flexibles System beschrieben, das die Verbindung von beliebig vielen Fernsteuerungselementen und zu steuernden Geräten ermöglicht und die Fernsteuerung umsetzt.

Von zentraler Bedeutung für Medizingeräte sind Alarme. Um Problemen wie der Alarmmüdigkeit, der Desensibilisierung oder der Lärmbelastung zu begegnen, können verteilte Alarmsysteme eingesetzt werden. Auf einer ITS ermöglichen diese etwa, die Signalisierung eines Alarms dort vorzunehmen, wo sich zuständige Personen gerade aufhalten. Oft ist dies nicht in unmittelbarer Nähe des Patientenbettes. Daher ist ein akustischer Alarm beim Patienten weder hilfreich für das medizinische Personal, noch ist die Störung förderlich für den Genesungsprozess des Patienten. In Kapitel 9 wird daher ein *verteiltes Alarmierungssystem* auf der Basis der *IEEE 11073 SDC*-Normen beschrieben. Dieses ermöglicht es, Alarme geräte- und herstellerübergreifend zuverlässig zu kommunizieren. Das Konzept berücksichtigt, dass gerade mobile Endgeräte, auf denen Alarmsignale generiert werden sollen, typischerweise drahtlos angebunden sind und keine permanente Stromversorgung haben. Daher müssen Verbindungsabbrüche zwischen dem Gerät, das die Alarmbedingung detektiert und dem entfernten Gerät, das das Alarmsignal für den Nutzer generieren soll, erkannt werden, um entsprechende Rückfallmechanismen zu ermöglichen. Das beschriebene *verteilte Alarmierungssystem* kann direkt für herstellerübergreifende Systeme genutzt werden, hat aber auch das Potential, als Grundlage für innovative Weiterentwicklungen intelligenter Alarmsysteme zu dienen.

Für alle beschriebenen Anwendungsfälle werden prototypische Umsetzungen mit echten Medizingeräten präsentiert. Diese zeigen die Umsetzbarkeit der Konzepte und lassen technische Evaluationen zu. Für die Performanzevaluationen in den Kapiteln 5 und 8.4 werden zur Nachverfolgbarkeit sowohl die genutzten Implementierungen als auch die Rohdaten der Messergebnisse in öffentlich zugänglichen Repositories am IMD der Universität Rostock zur Verfügung gestellt.

Die Innovation der vorliegenden Arbeit liegt in der konsequenten Nutzung der offenen und normierten *IEEE 11073 SDC*-Technologien. Die verschiedenen Beiträge weisen nach, dass die Aufgaben bestehender, proprietärer Systeme ohne Abstriche erfüllt werden können. Der Funktionsumfang existiert bereits innerhalb einiger geschlossener Systeme, die am Markt verfügbar sind. Offene und interoperable Systeme stehen hingegen bisher nicht zur Verfügung. Hinzu kommt, dass alle notwendigen Informationen zur Laufzeit ausgetauscht werden, sodass Hersteller zur Entwicklungszeit kein Wissen über die konkreten potentiellen Vernetzungspartner benötigen. Derzeit notwendige und aufwändige Absprachen zwischen Herstellern können somit entfallen. Dies ermöglicht flexible, interoperable Adhoc-Ensembles, die Medizingeräte über Herstellergrenzen hinweg zusammenführen.

10.2 Praktische Umsetzungen und Demonstratoren

Während der Arbeiten an dieser Dissertation sind diverse Softwarekomponenten entstanden, die sich im praktischen Einsatz befanden bzw. befinden. Es wurden eine Reihe von *IEEE 11073 SDC*-Konnektoren entwickelt, die es ermöglichen, vorhandene Medizingeräte in die interoperable Vernetzung zu integrieren. So wurden beispielsweise Konnektoren für eine Auswahl von mehr als zehn Endoskopie-Geräten des Projektpartners Karl Storz implementiert. Weitere Konnektoren wurden etwa für HF-Chirurgiegeräte, ein Pulsoximeter, ein Vitalparameter-Entwicklungsboard und mehrere Fußschalter entwickelt.

Auf der Seite der *Service Consumer* wurden diverse Komponenten entwickelt, wie etwa die Anzeige von Vital- und Geräteparametern in Form eines Overlays im endoskopischen Bild direkt im Sichtfeld des Anwenders. Abbildung 10.1 zeigt die Overlay-Software im Zuge eines Ausbildungskurses für minimalinvasive Chirurgie im Demonstrator-OP der Universität Heidelberg, der für Operationen an Tieren zugelassen ist. Es sei angemerkt, dass in diesem Prototyp zur technischen Evaluation deutlich mehr Informationen bereitgestellt werden als notwendig und sinnvoll. Zudem sind nicht alle Informationen im Netzwerk vorhanden: Der arterielle Blutdruck und die Körpertemperatur werden nicht bereitgestellt. Ebenso sind die Patienteninformationen nicht verfügbar, da es sich nicht um einen menschlichen Patienten handelt. Die fehlenden Informationen werden entsprechend durch das rote Ausrufezeichensymbol als kritisch markiert.

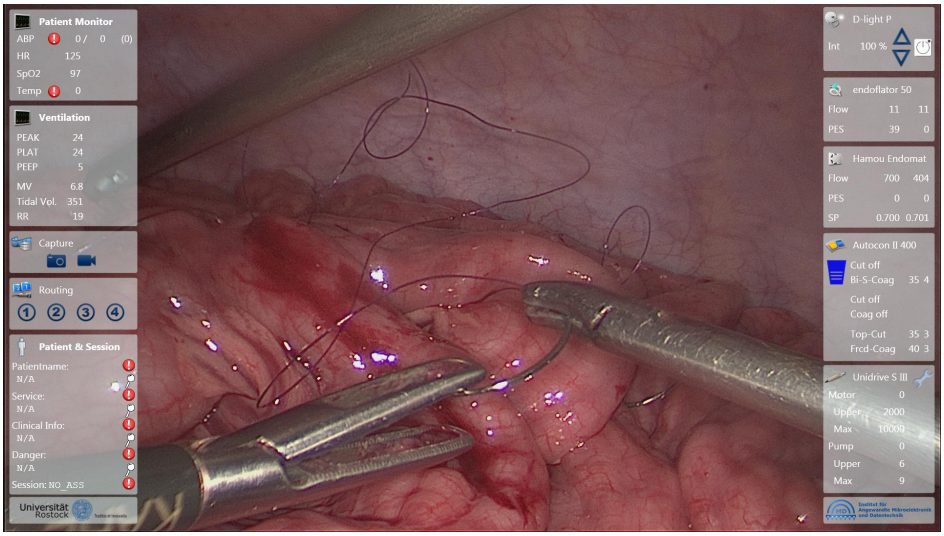

Abbildung 10.1: Endoskopisches Overlay während eines Ausbildungskurses im Demonstrator-OP der Universität Heidelberg. Rechts: Parameter chirurgischer Geräte; links oben: anästhesiologische Vital- und Geräteparameter; links mittig: Dokumentationsauslösung und Anzeigekonfiguration; links unten: vorgesehen für Patienteninformationen.

Diese Implementierungen des Autors waren an mehreren temporären Demonstratoren im Zuge von Messen und Projektpräsentationen im Einsatz. Des Weiteren werden sie aktiv im permanenten Demonstrator-OP am *Innovation Center Computer Assisted Surgery (ICCAS)* der Universität Leipzig genutzt. Stellvertretend für die Messen *CeBIT* 2014 in Hannover, *conhIT* 2015 bis 2017 in Berlin, *XPOMET* 2018 in Leipzig, *DMEA* 2019 in Berlin und *MEDICA* 2019 in Düsseldorf zeigt Abbildung 10.2 den Autor bei einer Vorführung auf der *DMEA*-Messe 2019.

10.3 Arbeiten in Verbindung mit dieser Dissertation

Neben einer Reihe von wissenschaftlichen Konferenz- und Journal-Publikationen (siehe Anhang B) entstanden einige Beiträge für Fachzeitschriften und Konferenzen/Messen mit starkem Anwenderbezug. Diese Transferaktivitäten stellen einen Baustein für den Austausch von Know-how zwischen der Wissenschaft und der Industrie dar. Eine Übersicht befindet sich in Anhang C.

Einige Arbeiten des Autors finden aus Gründen des Umfangs keinen Eingang in diese Dissertation. Exemplarisch seien einige Themenbereiche genannt. Im Fokus des Projektes *Modular Validation Environment for Medical Device Networks (MoVE)* stand das Testen von dynamisch und herstellerübergreifend vernetzten Medizingeräten. Die Herausforderung besteht darin, dass zur Entwicklungs- und Testzeit nicht vollständig klar ist, mit welchen Vernetzungspartnern das zu testende Gerät später interagieren wird. Ebenso steht in der Regel keine breite Auswahl von „Gegenspielern“ zum Testen zur Verfügung Der Autor war an der Entwicklung des Konzepts für die Testplattform maßgeblich beteiligt. Zudem wurde und wird mit dem *Abstract Provider Simulator (APS)* ein Tool entwickelt, das sowohl als Teil der *MoVE*-Testplattform als auch als Einzelkomponente beliebige Medizingeräte simulieren kann. Das *APS*-Tool steht öffentlich zur Verfügung[10.1].

Ein zweiter Schwerpunkt, der nur in wenigen Aspekten im Zuge dieser Arbeit angesprochen wird, ist die Entwicklung von Normen für *IEEE 11073 SDC*-Gerätespezialisierungen. Diese Arbeiten erfolgen im Projekt *Modular Specialisations for Point-of-Care Medical Devices (PoCSpec)*. Der Autor leitet die Entwicklung der Normen für HF-Chirurgiegeräte [D 9] und medizinische Saug-/Spülpumpen [D 13] und ist beteiligt an weiteren endoskopischen Geräten [D 10–12]. Erste Ergebnisse konnten bereits auf wissenschaftlichen Konferenzen vorgestellt werden [B 1, 3].

[10.1] Link: `https://gitlab.amd.e-technik.uni-rostock.de/MoVE/abstractprovidersimulator`

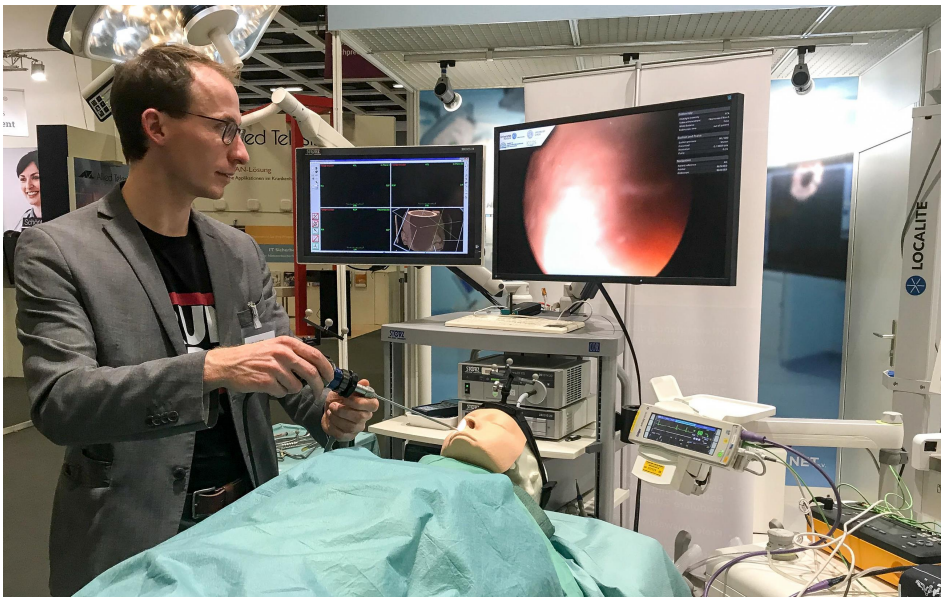

Abbildung 10.2: Demonstrator im Zuge der *DMEA*-Messe 2019 in Berlin: Vorführung durch den Autor der Arbeit.

Ebenso konnten Arbeiten im Bereich der echtzeitfähigen Kommunikation auf der Basis von *IEEE 11073 SDC* [B 2, 13, 22] und Aspekte der Telemedizin [B 29] nicht betrachtet werden.

10.4 Ausblick und zukünftige Forschungsfragen

Neben den Konzepten und Problemlösungen wurden in den verschiedenen Kapiteln der vorliegenden Arbeit bereits einige zukünftige Forschungsfragen und Entwicklungspotentiale aufgezeigt. Im Folgenden werden relevante Aspekte für weiterführende Arbeiten aufgegriffen und über die Anwendungsbereiche hinweg zusammengeführt.

Die ersten Medizingeräte mit einer *IEEE 11073 SDC*-basierten Vernetzung sind bereits in der Patientenversorgung im Einsatz [A 154]. „*Plugfest*“-Veranstaltungen des OR.NET e. V. lassen ein immer breiteres Interesse aus der Industrie erkennen, das über die Grenzen des Konsortiums hinausgeht. Viele klinische Anwender stehen den neuen Technologien und entstehenden Potentialen positiv und erwartungsfroh gegenüber [B 14]. Nichtsdestotrotz können in einem strikt regulierten Marktumfeld mit sicherheitskritischen Produkten neue Technologien kurzfristig nicht alle bestehenden Probleme lösen. Neue Lösungen bringen meist auch neue Herausforderungen mit sich, die ggf. heute noch nicht absehbar sind. Insbesondere die Aspekte der *Cybersecurity*, der sicheren Bedienbarkeit, des Inverkehrbringens und der Organisationsentwicklung in den Krankenhäusern sind essenziell.

Die Hürde für einen *Cyberangriff* auf ein Medizingerät ohne Netzwerkschnittstelle ist hoch. Der Angreifer benötigt physikalischen Zugang zum Gerät. Dies ändert sich für vernetzte Systeme dramatisch. Sichere, im Sinne der *Safety*, vernetzte Systeme können nur mit entsprechenden *Cybersecurity*-Mechanismen existieren [A 53]. Für herstellerübergreifend und dynamisch vernetzte Systeme ergeben sich große Herausforderungen. Die *Security*-Mechanismen müssen wirkungsvoll genug sein, um hoch sensible Patientendaten zu schützen, Manipulationen von Alarmen, Parametern und Fernsteuerungsbefehlen zu verhindern etc. Die Umsetzung darf jedoch die beabsichtigte Kommunikation über Herstellergrenzen hinweg nicht verhindern. Die zu entwickelnden und bestenfalls zu vereinheitlichenden bzw. zu standardisierenden Mechanismen sollten folglich diesen Ausgleich schaffen. Zudem ist die Benutzbarkeit, sowohl im Hinblick auf das medizintechnische Krankenhauspersonal als auch auf Ärztinnen und Pflegende als Endanwender, zu garantieren. Auf jeder organisatorischen Ebene des Krankenhauses müssen die Akteure mitgenommen werden, um die *Cybersecurity* zu gewährleisten.

Aus technischer Sicht ist die Gewährleistung harter Echtzeiteigenschaften ein entscheidender Aspekt für diverse kritische Anwendungsfälle. Die Norm *IEEE 11073-20701* stellt klar, dass die derzeitigen *IEEE 11073 SDC*-Kernstandards keine harte Echtzeitfähigkeit garantieren. In Kapitel 7 werden Mechanismen für eine Fernauslösung von kritischen Gerätefunktionalitäten durch einen menschlichen Akteur mittels eines potentiell unzuverlässigen Netzwerks präsentiert. Für diese *Human-in-the-Loop*-Szenarien wurde bereits thematisiert, dass eine hart echtzeitfähige Vernetzung trotz des bzw. ergänzend zum vorgestellten Konzept wünschenswert wäre. In diversen *Closed-Loop*-Anwendungen, mit hoher Kritikalität und kurzen Zeitanforderungen, ist harte Echtzeit unerlässlich [A 249]. Ein Beispiel für eine solche Anwendung ist ein System, das automatisch die Rotation der Fräse stoppt, wenn sich der Fräskopf in kritischer Nähe zu einer Risikostruktur, wie etwa einem Nerv, befindet. Somit wird sichergestellt, dass keine Schädigung der Risikostruktur erfolgt. Eine solche Anwendung kann beispielsweise auf der Basis eines Neuro-Monitorings des Nervs und/oder über eine genaue Lokalisierung von Risikostruktur und Instrument mittels eines sogenannten Navigationssystems erfolgen [A 250, 251]. Ein weiteres Beispiel aus der Intensivmedizin ist die Vernetzung eines Kompressionsgeräts zur automatisierten Durchführung einer Herz-Lungen-Wiederbelebung mit einem Beatmungsgerät. Hier ist die genaue Synchronisation zwischen Thoraxkompression und Beatmungshub erforderlich.

Im Zuge des *OR.NET*-Projekts wurde vom Autor aufgezeigt, wie sich die Vorteile der dynamischen und interoperablen *IEEE 11073 SDC*-basierten Vernetzung und die einer hart echtzeitfähigen Vernetzung auf der Basis von Feldbussystemen verbinden lassen [B 13]. Der Stand der Forschung setzt aber zwei getrennte Netze voraus (siehe auch Kapitel 2.3.2). Entsprechend ist das Ziel, die harte Echtzeitfähigkeit und die Dynamik und herstellerübergreifende Interoperabilität in einer Vernetzungslösung zusammenzubringen.

Es könnte zukünftig erforscht werden, ob die Nutzung von *IEEE 11073 SDC* in Feldbussystemen möglich und sinnvoll ist. Vielversprechend ist der folgende Ansatz: die Kombination der Normenfamilien *Time-Sensitive Networking (TSN)* (*IEEE 802.1*) und *IEEE 11073 SDC*. *TSN* ermöglicht die Garantie einer harten Echtzeitkommunikation bei gleichzeitig hoher Flexibilität [A 252, 253]. Das Grundkonzept besteht darin, automatisch aus der *IEEE 11073 SDC*-Gerätebeschreibung der Vernetzungsteilnehmer die Echtzeitanforderungen an das *TSN*-Netzwerk abzuleiten. Die grundlegende Machbarkeit wurde bereits unter der Beteiligung bzw. Betreuung des Autors dieser Arbeit gezeigt [B 2], [E 2]. Auf der Basis der abgeleiteten Anforderungen können das *Routing* und *Scheduling* im *TSN*-Netzwerk erzeugt werden. Forschungsarbeiten und Tools werden hierfür beispielsweise am IMD der Universität Rostock entwickelt [A 254–256]. Forschungs- und Entwicklungsarbeiten hin zu einer hart echtzeitfähigen *IEEE 11073 SDC*-Kommunikation sollten fortgeführt und ausgebaut werden. Das Ziel ist, dass auf dieser Basis Medizinprodukte konformitätsbewertet und *in den Verkehr gebracht* werden können.

Dynamisch vernetzte Medizingeräteensembles ermöglichen und erfordern neue Konzepte für *Mensch-Maschine-Interfaces (MMIs)*, wie sie im *OR.NET*-Kontext beispielsweise für universell assoziierbare Fußschalter diskutiert und entwickelt werden [A 72, 73, 219, 220]. Unmittelbar *IEEE 11073 SDC*-bezogene Forschungsfragen sind etwa, wie aus den Gerätebeschreibungen automatisiert sicher und intuitiv benutzbare Benutzeroberflächen eines entfernten Gerätes abgeleitet werden können, welche vorhandenen Mechanismen genutzt werden können oder welche Erweiterungen notwendig sind. Eine weitere Herausforderung für *MMIs* besteht darin, die „richtigen“ Informationen zur „richtigen“ Zeit am „richtigen“ Ort bereitzustellen. Wie erwähnt birgt beispielsweise das Overlay in Abbildung 10.1 in diesem Aspekt erhebliches Optimierungspotential.

Die Normungsaktivitäten der *IEEE 11073 SDC*-Familie sollten in den kommenden Jahren fortgesetzt werden. Während die *Kernstandards* bereits durch die *International Organization for Standardization (ISO)* anerkannt sind, befinden sich die *Participant Key Purposes (PKPs)* und Gerätespezialisierungen noch in der Entwicklung. Gerade im Bereich der Gerätespezialisierungen besteht der Bedarf,

weitere Normen für weitere Gerätegruppen zu entwickeln, die über die aktuellen Normungsprojekte hinausgehen. So sind etwa die Bereiche Anästhesie und Intensivmedizin zu nennen, aber auch weitere chirurgische Geräte.

Die Hersteller von *IEEE 11073 SDC*-fähigen Medizinprodukten können in Zukunft mit der Entwicklung angepasster Strategien für den Prozess der Konformitätsbewertung und mit Konzepten wie einem modularen Risikomanagement unterstützt werden. Die im *OR.NET*-Kontext begonnenen Arbeiten [A 51] sollten fortgesetzt werden, denn nur erfolgreich konformitätsbewertete Geräte können herstellerübergreifend vernetzt werden und so der Patientenversorgung zugutekommen.

Mittelfristig bietet die *IEEE 11073 SDC*-basierte Interoperabilität ein großes Potential für die Entwicklung innovativer Assistenzsysteme zur Unterstützung der klinischen Akteure. Die Grundlage bilden die zukünftig zur Verfügung stehenden, semantisch annotierten Informationen verschiedenster Medizingeräte. Diese können einerseits gesammelt und andererseits direkt genutzt werden. Die Anwendungsbereiche reichen von der einfachen Zusammenführung dieser Informationen bis hin zur Nutzung von Methoden der *künstlichen Intelligenz (KI)* für komplexe Herausforderungen. Gerade letztere profitieren von einer großen Menge gesammelter und vorzugsweise annotierter Daten. Ein Referenzmodell für die Realisierung von *KI*-basierten Systemen auf der Basis von Normen wie *IEEE 11073 SDC* und *HL7 FHIR (Fast Healthcare Interoperability Resources)* wurde bereits vom Autor dieser Arbeit zusammen mit mehreren Co-Autoren vorgestellt [B 6]. Eine Umsetzung des Referenzmodells zur Realisierung einer intelligenten Steuerung einer Arthroskopie-Pumpe konnte ebenso prototypisch gezeigt werden [B 3]. Auf beide Publikationen kann wegen des Umfangs dieser Dissertation nicht direkt eingegangen werden.

Viele Anwendungsfälle können von der Zusammenführung oder der Analyse von Informationen, ggf. unter der Nutzung lernender Systeme oder anderer Methoden der *KI*, profitieren. Stellvertretend seien folgende Beispiele erwähnt: Entscheidungsunterstützungssysteme können Empfehlungen für Therapie und Diagnostik geben [A 257, 258]. Die medizinischen Fachkräfte sollen dabei nicht ersetzt, sondern möglichst gut unterstützt werden. Die mögliche Bandbreite intelligenter Alarmsysteme kann aus Kapitel 9 abgeleitet werden. Einen weiteren Bereich stellen Systeme zur Analyse und darauf basierenden Unterstützung der Arbeitsabläufe im Krankenhaus dar, die auf der Basis von *KI* umgesetzt werden. Im permanenten *IEEE 11073 SDC*-Demonstrator-OP am *ICCAS* der Universität Leipzig werden verschiedene Aspekte der Assistenz gezeigt. Beispielsweise erfolgt eine Abschätzung der verbleibenden Operationszeit, um etwa die zeitlichen Abläufe für den kommenden Patienten zu optimieren; eine Detektion der chirurgischen Arbeitsschritte, die zur Dokumentationsunterstützung genutzt werden kann; eine automatische Steuerung der verschiedenen Lichtquellen oder eine situationsabhängige Zuordnung von Bildquellen (endoskopisches Bild, präoperative Befunde und Bilddatensätze, Navigationssystem etc.) zu Anzeigemonitoren [A 41, 259], [B 9]. Die Beispiele zeigen die vielfältigen Forschungs- und Entwicklungsmöglichkeiten, die durch die neue Vernetzungstechnologie ermöglicht werden.

Die Grundlagen für eine herstellerübergreifende Interoperabilität von Medizingeräten sind gelegt – die vorliegende Arbeit leistet hierzu einen Beitrag. Nun gilt es, eine breite Masse der Hersteller davon zu überzeugen, *IEEE 11073 SDC* in ihren Geräten umzusetzen und aus der Forschung heraus innovative Produkte zu entwickeln. Diese können, in enger Abstimmung mit den Bedarfen der Ärztinnen und Pflegenden, die Patientensicherheit und Behandlungserfolge verbessern, Arbeitsabläufe optimieren und Kosten senken.

A Literaturverzeichnis

[1] Sue Sendelbach und Marjorie Funk. „Alarm Fatigue: A Patient Safety Concern“. In: *AACN Advanced Critical Care* 24.4 (2013), S. 378–386. DOI: 10.1097/NCI.0b013e3182a903f9.

[2] Maria Cvach. „Monitor Alarm Fatigue : An Integrative Review“. In: *Biomedical Instrumentation & Technology* 46.4 (Juli 2012), S. 268–277. DOI: 10.2345/0899-8205-46.4.268.

[3] Ok Min Cho, Hwasoon Kim, Young Whee Lee und Insook Cho. „Clinical Alarms in Intensive Care Units: Perceived Obstacles of Alarm Management and Alarm Fatigue in Nurses“. In: *Healthcare Informatics Research* 22.1 (2016), S. 46. DOI: 10.4258/hir.2016.22.1.46.

[4] Barbara J. Drew, Patricia Harris, Jessica K. Zègre-Hemsey, Tina Mammone, Daniel Schindler, Rebeca Salas-Boni, Yong Bai, Adelita Tinoco, Quan Ding und Xiao Hu. „Insights into the Problem of Alarm Fatigue with Physiologic Monitor Devices: A Comprehensive Observational Study of Consecutive Intensive Care Unit Patients“. In: *PLoS ONE* 9.10 (Okt. 2014). DOI: 10.1371/journal.pone.0110274.

[5] Association for the Advancement of Medical Instrumentation (AAMI). *2011 Summit on Clinical Alarms: A Siren Call to Action - Priority Issues from the Medical Device Alarms Summit*. Techn. Ber. 2011. URL: http://s3.amazonaws.com/rdcms-aami/files/production/public/FileDownloads/Summits/2011_Alarms_Summit_publication.pdf.

[6] International Electrotechnical Commission (IEC). *IEC 60601-1-8:2006/A1:2012 – Medical electrical equipment – Part 1-8: General requirements for basic safety and essential performance – Collateral standard: General requirements, tests and guidance for alarm systems in medical electrical equipment and medical electrical systems*. Nov. 2012.

[7] Nikita Salnikov-Tarnovski und Gleb Smirnov. *Java Garbage Collection Handbook*. Plumbr, 2015, S. 75.

[8] Oracle. *Java Garbage Collection Basics*. URL: https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html (besucht am 27. 10. 2020).

[9] ASTM International (American Society for Testing and Materials). *ASTM F2761-09(2013), 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*. West Conshohocken, PA. DOI: 10.1520/F2761.

[10] Europäisches Parlament und Rat. *Verordnung (EU) 2017/745 des Europäischen Parlaments und des Rates über Medizinprodukte – 'Medical Device Regulation (MDR)'*. Apr. 2017. URL: http://data.europa.eu/eli/reg/2017/745/oj.

[11] Europäischer Rat. *Richtlinie 93/42/EWG des Rates über Medizinprodukte – 'Medical Device Directive (MDD)'*. 1993.

[12] Deutscher Bundestag. *Gesetz über Medizinprodukte (Medizinproduktegesetz – MPG)*. 1994.

[13] Tomas Kalibera und Richard Jones. „Rigorous benchmarking in reasonable time“. In: *ACM SIGPLAN Notices* 48.11 (Dez. 2013), S. 63–74. DOI: 10.1145/2555670.2464160.

[14] John Aycock. „A brief history of just-in-time“. In: *ACM Computing Surveys* 35.2 (Juni 2003), S. 97–113. DOI: 10.1145/857076.857077.

[15] Christian Ullenboom. *Java ist auch eine Insel - Das umfassende Handbuch*. 9. aktualisierte und überarbeitete Auflage. Galileo Press, 2010, S. 1482. ISBN: 978-3-8362-1506-0. URL: http://openbook.rheinwerk-verlag.de/javainsel9/.

[16] Edd Barrett, Carl Friedrich Bolz-Tereick, Rebecca Killick, Sarah Mount und Laurence Tratt. „Virtual machine warmup blows hot and cold“. In: *Proceedings of the ACM on Programming Languages* 1.OOPSLA (Okt. 2017), S. 1–27. DOI: 10.1145/3133876.

[17] T. Cramer, R. Friedman, T. Miller, D. Seberger, R. Wilson und M. Wolczko. „Compiling Java just in time“. In: *IEEE Micro* 17.3 (1997), S. 36–43. DOI: 10.1109/40.591653.

[18] ITU-T Study Group 17. *ITU-T Recommendation X.509 – ISO/IEC 9594-8:2016 – Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*. Techn. Ber. Okt. 2016. URL: http://handle.itu.int/11.1002/1000/13031.

[19] Russell Housley, William Polk, Warwick Ford und David Solo. *IETF RFC 3280 – Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile*. Techn. Ber. 2002. URL: https://www.ietf.org/rfc/rfc3280.txt.

[20] ITU-T Study Group 17. *ITU-T Recommendation X.500 – ISO/IEC 9594-1:2016 – Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services*. Techn. Ber. Okt. 2016. URL: http://handle.itu.int/11.1002/1000/13029.

[21] Jörg Schwenk. *Sicherheit und Kryptographie im Internet*. Wiesbaden: Vieweg+Teubner, 2010. DOI: 10.1007/978-3-8348-9665-0.

[22] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley und William Polk. *IETF RFC 5280 – Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile*. Techn. Ber. 2008. URL: https://www.ietf.org/rfc/rfc5280.txt.

[23] Stan Schneider. „The Internet Of Things Can Save 50,000 Lives A Year“. In: *ElectronicDesign and RTI Whitepaper* (Jan. 2014).

[24] Westhealth Institute. *The Value Of Medical Device Interoperability*. März 2013. URL: https://www.westhealth.org/wp-content/uploads/2015/02/The-Value-of-Medical-Device-Interoperability.pdf.

[25] Kathy Lesh, Sandy Weininger, Julian M Goldman, Bob Wilson und Glenn Himes. „Medical Device Interoperability-Assessing the Environment“. In: *2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP 2007)*. IEEE, Juni 2007, S. 3–12. DOI: 10.1109/HCMDSS-MDPnP.2007.22.

[26] Caprice K. Christian, Michael L. Gustafson, Emilie M. Roth, Thomas B. Sheridan, Tejal K. Gandhi, Kathleen Dwyer, Michael J. Zinner und Meghan M. Dierks. „A prospective study of patient safety in the operating room“. en. In: *Surgery* 139.2 (Feb. 2006), S. 159–173. DOI: 10.1016/j.surg.2005.07.037.

[27] Yue-Yung Hu, Alexander F Arriaga, Emilie M Roth, Sarah E Peyre, Katherine A Corso, Richard S Swanson, Robert T Osteen, Pamela Schmitt, Angela M Bader, Michael J Zinner

und Caprice C Greenberg. „Protecting Patients from an Unsafe System: The Etiology & Recovery of Intra-Operative Deviations in Care“. In: *Annals of surgery* 256.2 (Aug. 2012), S. 203–210. DOI: 10.1097/SLA.0b013e3182602564.

[28] Annetje Guédon, Linda Wauben, Marlies Overvelde, Joleen Blok, Maarten van der Elst, Jenny Dankelman und John van den Dobbelsteen. „Safety status system for operating room devices“. In: *Technology and Health Care* 22.6 (2014), S. 795–803. DOI: 10.3233/THC-140854.

[29] Ruwan A Weerakkody, Nicholas J Cheshire, Celia Riga, Rachael Lear, Mohammed S Hamady, Krishna Moorthy, Ara W Darzi, Charles Vincent und Colin D Bicknell. „Surgical technology and operating-room safety failures: a systematic review of quantitative studies“. In: *BMJ Quality & Safety* 22.9 (Sep. 2013), S. 710–718. DOI: 10.1136/bmjqs-2012-001778.

[30] I. Wubben, J. G. van Manen, B. J. van den Akker, S. R. Vaartjes und W. H. van Harten. „Equipment-related incidents in the operating room: an analysis of occurrence, underlying causes and consequences for the clinical process“. en. In: *BMJ Quality & Safety* 19.6 (Dez. 2010). DOI: 10.1136/qshc.2009.037515.

[31] Ann S Lofsky. „Turn your alarms on“. In: *APSF Newsletter: The Official Journal of the Anesthesia Patient Safety Foundation* 19.4 (2005), S. 43. URL: https://www.apsf.org/wp-content/uploads/newsletters/2004/winter/pdf/APSF200412.pdf.

[32] David Arney, Julian M. Goldman, Susan F. Whitehead und Insup Lee. „Improving Patient Safety with X-Ray and Anesthesia Machine Ventilator Synchronization: A Medical Device Interoperability Case Study“. In: 2010, S. 96–109. DOI: 10.1007/978-3-642-11721-3_7.

[33] Paul Polach. *Photo: Laparoscopic Surgery*. 2007. URL: https://commons.wikimedia.org/wiki/File:US_Navy_070118-N-3455P-003_Fleet_Surgical_Team_5_performs_laparoscopic_surgery_aboard_the_amphibious_assault_ship_USS_Boxer_(LHD_4).jpg (besucht am 27. 10. 2020).

[34] Melanie Blaar, Armin Janß, Jasmin Dell'Anna, Anke Höllig, Klaus Radermacher und Hans Clusmann. „Bottlenecks and needs in human-human and human-machine interaction – a view from and into the neurosurgical OR“. In: *Biomedical Engineering / Biomedizinische Technik* 61.2 (Jan. 2016). DOI: 10.1515/bmt-2015-0059.

[35] Jasmin Dell'Anna, Armin Janss, Melanie Blaar, Anke Höllig, Hans Clusmann und Klaus Radermacher. „Analysis of User-Induced Risks in the Neurosurgical OR“. In: *5th International Conference on Applied Human Factors and Ergonomics AHFE*. July. Kraków, Poland, 2014, S. 352–358.

[36] Amy L. Halverson, Jessica T. Casey, Jennifer Andersson, Karen Anderson, Christine Park, Alfred W. Rademaker und Don Moorman. „Communication failure in the operating room“. In: *Surgery* 149.3 (März 2011), S. 305–310. DOI: 10.1016/j.surg.2010.07.051.

[37] The Joint Commission (TJC). *Sentinel Events Alert - Medical Device Alarm Safety in Hospitals*. Techn. Ber. 50. Apr. 2013. URL: https://www.jointcommission.org/assets/1/18/sea_50_alarms_4_5_13_final1.pdf.

[38] Matthias Borowski, Matthias Görges, Roland Fried, Olaf Such, Christian Wrede und Michael Imhoff. „Medical device alarms“. In: *Biomedizinische Technik/Biomedical Engineering* 56.2 (Jan. 2011), S. 73–83. DOI: 10.1515/bmt.2011.005.

[39] Avinash Konkani, Barbara Oakley und Thomas J. Bauld. „Reducing Hospital Noise: A Review of Medical Device Alarm Management“. In: *Biomedical Instrumentation & Technology* 46.6 (Nov. 2012), S. 478–487. DOI: 10.2345/0899-8205-46.6.478.

[40] Felix Schmid, Matthias S. Goepfert und Daniel A. Reuter. „Patient monitoring alarms in the ICU and in the operating room“. In: *Critical Care* 17.2 (2013), S. 216. DOI: 10.1186/cc12525.

[41] Stefan Franke, Max Rockstroh, Mathias Hofer und Thomas Neumuth. „The intelligent OR: design and validation of a context-aware surgical working environment“. In: *International Journal of Computer Assisted Radiology and Surgery* 13.8 (Aug. 2018), S. 1301–1308. DOI: 10.1007/s11548-018-1791-x.

[42] Stefan Franke, Max Rockstroh, Erik Schreiber, Juliane Neumann und Thomas Neumuth. „Context-aware medical assistance systems in integrated surgical environments“. In: *28th Conference of the international Society for Medical Innovation and Technology (SMIT)*. 2016.

[43] Hang Cheng, Jeffrey W. Clymer, Brian Po-Han Chen, Behnam Sadeghirad, Nicole C. Ferko, Chris G. Cameron und Piet Hinoul. „Prolonged operative duration is associated with complications: a systematic review and meta-analysis“. In: *Journal of Surgical Research* 229 (Sep. 2018), S. 134–144. DOI: 10.1016/j.jss.2018.03.022.

[44] Jacob S. Brady, Stuti V. Desai, Meghan M. Crippen, Jean Anderson Eloy, Yuriy Gubenko, Soly Baredes und Richard Chan Woo Park. „Association of Anesthesia Duration With Complications After Microvascular Reconstruction of the Head and Neck“. In: *JAMA Facial Plastic Surgery* 20.3 (Mai 2018), S. 188–195. DOI: 10.1001/jamafacial.2017.1607.

[45] HIMSS Europe. *Auf den Spuren der Zeitdiebe im Krankenhaus: Die wahre Belastung durch Dokumentation an deutschen Akutkrankenhäusern wird unterschätzt*. Techn. Ber. März 2015.

[46] Duane Bender und Kamran Sartipi. „HL7 FHIR: An Agile and RESTful approach to healthcare information exchange“. In: *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*. Porto: IEEE, Juni 2013, S. 326–331. DOI: 10.1109/CBMS.2013.6627810.

[47] Kaiser Permanente. *Analysis of Implementing Integrated Systems*. Techn. Ber. 2006. URL: http://mdpnp.org/uploads/Impact_HC_6.pdf.

[48] Bundesministerium für Bildung und Forschung (BMBF). *Bekanntmachung – Richtlinie zur Förderung von Zuwendungen für "KMU-innovativ: Medizintechnik"; Bundesanzeiger vom 19.09.2018*. Sep. 2018. URL: https://www.bmbf.de/foerderungen/bekanntmachung-2010.html (besucht am 27. 10. 2020).

[49] Deutschen Industrie- und Handelskammertags (DIHK) und Industrieverbands SPECTARIS - Fachverband Medizintechnik. *Auswirkungen der neuen EU-Medizinprodukte-Verordnung (MDR) sowie der neuen Verordnung für In-vitro-Diagnostika (IVDR) auf die Hersteller in Deutschland: Die Patientenversorgung mit innovativen Medizinprodukten wird schwieriger*. Techn. Ber. Jan. 2019.

[50] Armin Janß, Johannes Thorn, Malte Schmitz, Alexander Mildner, Jasmin Dell'Anna-Pudlik, Martin Leucker und Klaus Radermacher. „Extended device profiles and testing procedures for the approval process of integrated medical devices using the IEEE 11073 communication standard“. In: *Biomedical Engineering / Biomedizinische Technik* 63.1 (Feb. 2018), S. 95–103. DOI: 10.1515/bmt-2017-0055.

[51] Johannes Thorn, Malte Schmitz, Jasmin Dell'Anna, Alexander Mildner und Armin Janß. *Whitepaper – Erweiterung des Lebenszyklus von Medizingeräten um offene Vernetzungsfähigkeit*. Techn. Ber. März 2016.

[52] National Academy of Medicine. *Procuring Interoperability: Achieving High-Quality, Connected, and Person-centred Care*. Washington, DC, 2018.

[53] Dimitrios Serpanos. „There Is No Safety Without Security and Dependability“. In: *Computer* 52.6 (Juni 2019), S. 78–81. DOI: 10.1109/MC.2019.2903360.

[54] CEN/SS F20 Qualitätssicherung. *DIN EN 45020:2007-03 Normung und damit zusammenhängende Tätigkeiten - Allgemeine Begriffe (ISO/IEC Guide 2:2004); Dreisprachige Fassung EN 45020:2006*. 2007. DOI: 10.31030/9832912.

[55] International Organization for Standardization (ISO). *ISO – Frequently Asked Questions (FAQs) – Glossary*. URL: https://www.iso.org/glossary.html (besucht am 27. 10. 2020).

[56] U.S. Food & Drug Administration (FDA). *Design Considerations and Pre-market Submission Recommendations for Interoperable Medical Devices – Guidance for Industry and Food and Drug Administration Staff*. 2017.

[57] Healthcare Information and Management Systems Society (HIMSS). *HIMSS Dictionary of Health Information and Technology Terms, Acronyms and Organizations*. Productivity Press, Jan. 2019. DOI: 10.4324/9781351104524.

[58] Healthcare Information and Management Systems Society (HIMSS). *HIMSS Redefines Interoperability*. Jan. 2019. URL: https://www.himss.org/resource-news/himss-redefines-interoperability (besucht am 05. 06. 2020).

[59] Healthcare Information and Management Systems Society (HIMSS). *What is Interoperability in Healthcare?* 2019. URL: https://www.himss.org/what-interoperability (besucht am 27. 10. 2020).

[60] Michael Robkin, Sandy Weininger, Benjamin Preciado und Julian Goldman. „Levels of conceptual interoperability model for healthcare framework for safe medical device interoperability“. In: *2015 IEEE Symposium on Product Compliance Engineering (ISPCE)*. IEEE, Mai 2015, S. 1–8. DOI: 10.1109/ISPCE.2015.7138703.

[61] Andreas Tolk und James A. Muguira. „The Levels of Conceptual Interoperability Model“. In: *Proceedings of the 2003 Fall Simulation Interoperability Workshop*. 2003.

[62] Grand View Research Inc. *Operating Room Integration Market Size, Share & Trends Analysis Report By Component (Software, Services), By Device Type, By Application, By End Use (Hospitals, Ambulatory Surgical Centers), And Segment Forecasts, 2020 - 2027*. Techn. Ber. 2020.

[63] APEX Market Research. *Global Operating Room Integration System Market By Product Type (High Definition Display System, Recording and Documentation System) And By End-Users/Application (Hybrid OR, General OR) Global Market Share, Forecast Data, In-Depth Analysis, And Detailed O*. Techn. Ber. 2020.

[64] Orian Research. *Global IT Solutions for Integrated Operating Room Market 2019 by Company, Regions, Type and Application, Forecast to 2024*. Techn. Ber. 2019.

[65] Absolute Reports. *Global IT Solutions for Integrated Operating Room Market Growth (Status and Outlook) 2019-2024*. Techn. Ber. 2019.

[66] iData Research. *US Market for Video, High-Tech and Integrated & Hybrid Operating Theatre Equipment*. Techn. Ber. 2016. URL: http://www.idataresearch.com/product/us-video-high-tech-and-integrated-operating-theatre-equipment-market-2016-forecasted-to-2022-medsuite/.

[67] Allied Market Research, Seapee Bajaj und Chandani Poddar. *Global Operating Room Integration Market – Opportunity Analysis and Industry Forecast, 2018-2025*. Techn. Ber. 2018.

[68] Neil A. Halpern. „Innovative Designs for the Smart ICU - Part 3: Advanced ICU Informatics“. In: *Chest* 145.4 (Apr. 2014), S. 903–912. DOI: 10.1378/chest.13-0005.

[69] Tim Gee. *Medical Device Start Ups and Connectivity*. Juni 2008. URL: http://medicalconnectivity.com/2008/06/24/medical-device-start-ups-and-connectivity/ (besucht am 27. 10. 2020).

[70] Armin Janß, Simon Plogmann und Klaus Radermacher. „Human-centered risk management for medical devices – new methods and tools“. In: *Biomedical Engineering / Biomedizinische Technik* 61.2 (Jan. 2016). DOI: 10.1515/bmt-2014-0124.

[71] Peter Knipp und Armin Janß. „Regulatory Approval Route and Strategy for Open Networked Medical Devices“. In: *49. Jahrestagung der Deutschen Gesellschaft für Biomedizinische Technik (DGBMT), BMT 2015*. Lübeck, 2015.

[72] Jasmin Dell'Anna, Armin Janß, Hans Clusmann und Klaus Radermacher. „A Configurable Footswitch Unit for the Open Networked Neurosurgical OR – Development, Evaluation and Future Perspectives“. In: *i-com* 15.3 (Jan. 2016), S. 227. DOI: 10.1515/icom-2016-0031.

[73] Julia Benzko, Lisa Krause, Armin Janß, Björn Marschollek, Paul Merz, Jasmin Dell'Anna und Klaus Radermacher. „Modular user interface design for integrated surgical workplaces“. In: *Biomedical Engineering / Biomedizinische Technik* 61.2 (Jan. 2016). DOI: 10.1515/bmt-2014-0125.

[74] Jasmin Dell'Anna, Armin Janß, Eva-Maria Zeissig, Kathrin Ganser, Klaus Radermacher und Hans Clusmann. „Development of new concepts for safe and usable human-machine-interfaces in the open networked neurosurgical OR“. In: *67th Annual Meeting of the German Society of Neurosurgery (DGNC)*. Frankfurt am Main, 2016. DOI: 10.3205/16dgnc357.

[75] Armin Will, Raluca Pahontu und Björn Bergh. „Vernetzte Medizintechnik im Krankenhaus: Vernetzung von Medizingeräten und weiteren IT-Komponenten“. In: *KU Gesundheitsmanagement* (2015), S. 54–56. URL: http://www.genios.de/fachzeitschriften/artikel/KU/20150105/vernetzte-medizintechnik-im-kranken/3135599939.html.

[76] Max Rockstroh, Stefan Franke, Raluca Dees, Angela Merzweiler, Gerd Schneider, Max Dingler, Christian Dietz, Jonas Pfeifer, Franziska Kühn, Malte Schmitz, Alexander Mildner, Armin Janß, Jasmin Dell'Anna Pudlik, Marcus Köny, Björn Andersen, Björn Bergh und Thomas Neumuth. „From SOMDA to application – integration strategies in the OR.NET demonstration sites“. In: *Biomedical Engineering / Biomedizinische Technik* 63.1 (Feb. 2018), S. 69–80. DOI: 10.1515/bmt-2017-0023.

[77] Marcus Köny, Julia Benzko, Michael Czaplik, Björn Marschollek, Marian Walter, Rolf Rossaint, Klaus Radermacher und Steffen Leonhardt. „The Smart Operating Room: smartOR“. In: *Distributed Networks – Intelligence, Security, and Applications*. Boca Raton, FL, USA: CRC Press, Aug. 2013. Kap. Chapter 12, S. 291–316. DOI: 10.1201/b15282-16.

[78] Marcus Köny, Julia Benzko, Michael Czaplik, Marian Walter, Klaus Radermacher, Rolf Rossaint und Steffen Leonhardt. „Getting Anesthesia Online: The smartOR Network“. In: *International Journal On Advances in Internet Technology* 5.3 and 4 (2012), S. 114–125. URL: http://www.iariajournals.org/internet_technology/.

[79] David Gregorczyk, Timm Bußhaus und Raimund Mildner. „SOA zur Vernetzung medizinischer Geräte - Ein norddeutscher Ansatz“. In: *White Paper* (2012), S. 1–8.

[80] Okan Yilmaz, Dario Wieschebrock, Jan Heibeyn, Klaus Rademacher und Armin Janß. „Development and Evaluation of a Platform-Independent Surgical Workstation for an Open Networked Operating Theatre Using the IEEE 11073 SDC Communication Standard“. In: *Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Posture, Motion and Health. HCII 2020.* 2020, S. 79–92. DOI: 10.1007/978-3-030-49904-4_6.

[81] MD PnP Program. *The Medical Device 'Plug-and-Play' Interoperability Program: advancing safe and secure interoperability to improve patient care*. URL: http://www.mdpnp.org (besucht am 27. 10. 2020).

[82] Advancement of Medical Instrumentation (AAMI) und Underwriters Laboratories Inc. (UL). *ANSI/AAMI/UL 2800-1:2019 – Standard for Safety for Medical Device Interoperability*. 2019.

[83] MD PnP Program. *Clinical Scenario Repository*. URL: http://www.mdpnp.org/MD_PnP_Program_CS_Reposit.php (besucht am 27. 10. 2020).

[84] MD PnP Program. *Open Medical Device and Data Integration Platforms to support the management of Ebola*. 2014. URL: http://www.mdpnp.org/ebola.html (besucht am 27. 10. 2020).

[85] David Arney, Jeffrey Plourde und Julian M. Goldman. „OpenICE medical device interoperability platform overview and requirement analysis“. In: *Biomedical Engineering / Biomedizinische Technik* 63.1 (Feb. 2018), S. 39–47. DOI: 10.1515/bmt-2017-0040.

[86] Advancement of Medical Instrumentation (AAMI). *ANSI/AAMI 2700-1:2019 – Medical Devices and Medical Systems–Essential safety and performance requirements for equipment comprising the patient-centric integrated clinical environment (ICE)–Part 1: General requirements and conceptual model*.

[87] Hamed Soroush, David Arney und Julian Goldman. „Toward a Safe and Secure Medical Internet of Things“. In: *IIC Journal of Innovation* 2 (2016), S. 4–18.

[88] International Organization for Standardization (ISO). *ISO 14971:2007: Medical devices – Application of risk management to medical devices*. März 2007.

[89] International Electrotechnical Commission (IEC). *IEC 62304:2006: Medical device software – Software life cycle processes*. Mai 2006.

[90] Sam Procter, John Hatcliff, Sandy Weininger und Anura Fernando. „Error Type Refinement for Assurance of Families of Platform-Based Systems“. In: 2015, S. 95–106. DOI: 10.1007/978-3-319-24249-1_9.

[91] Yoshihiro Muragaki, Jun Okamoto, Taiichi Saito, Ken Masamune, Hiroshi Iseki, Kaoru Kurisu, Tetsuya Goto und Kazuhiro Hongo. *SCOT Project – The Smart Cyber Operating Theater (SCOT) project – Supporting neurosurgical decision making*. Juli 2017. URL: https://www.wfns.org/newsletter/18 (besucht am 27. 10. 2020).

[92] Jun Okamoto, Ken Masamune, Hiroshi Iseki und Yoshihiro Muragaki. „Development concepts of a Smart Cyber Operating Theater (SCOT) using ORiN technology“. In: *Biomedical Engineering / Biomedizinische Technik* 63.1 (Feb. 2018), S. 31–37. DOI: 10.1515/bmt-2017-0006.

[93] ORiN Consortium. *ORiN2.1 Specifications – Part 1: Outline (Version 2.1.0)*. Nov. 2008. URL: https://www.orin.jp/e/document/.

[94] ORiN Consortium. *ORiN2.1 Specifications – Part 2: CAO (Version 2.1.8)*. Juli 2012. URL: https://www.orin.jp/e/document/.

[95] ORiN Consortium. *ORiN2.1 Specifications – Part 3: CRD (Version 2.1.0)*. Nov. 2008. URL: https://www.orin.jp/e/document/.

[96] ORiN Consortium. *ORiN2.1 Specifications – Part 4: CAP (Version 2.1.1)*. Okt. 2010. URL: https://www.orin.jp/e/document/.

[97] International Organization for Standardization (ISO). *ISO 20242-4:2011 – Industrial automation systems and integration – Service interface for testing applications – Part 4: Device capability profile template*. Dez. 2011.

[98] Johann Berger, Max Rockstroh, Erik Schreiber, Yukishige Yoshida, Jun Okamoto, Ken Masamune, Yoshihiro Muragaki und Thomas Neumuth. „GATOR: connecting integrated operating room solutions based on the IEEE 11073 SDC and ORiN standards“. In: *International Journal of Computer Assisted Radiology and Surgery* (Aug. 2019). DOI: 10.1007/s11548-019-02056-3.

[99] Noriyasu Iwamoto, Atsushi Nishikawa, Toshikazu Kawai, Yuki Horise und Ken Masamune. „A novel medical robot architecture with ORiN for efficient development of telesurgical robots“. In: *CARS 2018 - Computer Assisted Radiology and Surgery: Proceedings of the 32nd International Congress and Exhibition*. 2018. DOI: 10.1007/s11548-018-1766-y.

[100] Ken Masamune, Atsushi Nishikawa, Toshikazu Kawai, Yuki Horise und Noriyasu Iwamoto. „The development of Smart Cyber Operating Theater (SCOT), an innovative medical robot architecture that can allow surgeons to freely select and connect master and slave telesurgical robots“. In: *Impact* 2018.3 (Juni 2018), S. 35–37. DOI: 10.21820/23987073.2018.3.35.

[101] R.J. Kennelly. „The IEEE 1073 Standard for Medical Device Communications“. In: *1998 IEEE AUTOTESTCON Proceedings. IEEE Systems Readiness Technology Conference. Test Technology for the 21st Century (Cat. No.98CH36179)*. IEEE, 1998, S. 335–336. DOI: 10.1109/AUTEST.1998.713466.

[102] L. Schmitt, T. Falck, F. Wartena und D. Simons. „Novel ISO/IEEE 11073 Standards for Personal Telehealth Systems Interoperability“. In: *2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP 2007)*. IEEE, Juni 2007, S. 146–148. DOI: 10.1109/HCMDSS-MDPnP.2007.9.

[103] Stefan Schlichting und Stephan Pöhlsen. „An architecture for distributed systems of medical devices in high acuity environments - A Proposal for Standards Adoption“. In: *IEEE 11073 / HL7 Standards Week*. San Antonio, Texas, USA, 2014. URL: www.hl7.org/documentcenter/public/wg/healthcaredevices/20140116%20An%20architecture%20for%20distributed%20systems%20of%20medical%20devices%20in%20high-acuity%20environments.pdf.

[104] S de Deugd, R Carroll, K E Kelly, B Millett und J Ricker. „SODA: Service Oriented Device Architecture“. In: *Pervasive Computing, IEEE* 5.3 (2006), S. 94–96. DOI: 10.1109/MPRV.2006.59.

[105] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005, S. 792. ISBN: 0131858580.

[106] Thomas Erl. *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007, S. 573. ISBN: 0132344823.

[107] Ingo Melzer. *Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis*. 4. Auflage. Heidelberg: Spektrum Akademischer Verlag, 2010, S. 382. DOI: 10.1007/978-3-8274-2550-8.

[108] OASIS. *Reference Model for Service Oriented Architecture 1.0 - OASIS Standard*. 2006. URL: http://docs.oasis-open.org/soa-rm/v1.0/.

[109] W3C Working Group Note. *Web Services Glossary*. 2004. URL: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/ (besucht am 27. 10. 2020).

[110] Christian Mauro, Ali Sunyaev, Jan Marco Leimeister und Helmut Krcmar. „Standardized Device Services - A Design Pattern for Service Oriented Integration of Medical Devices“. In: *2010 43rd Hawaii International Conference on System Sciences*. Honolulu, HI: IEEE, Jan. 2010, S. 1–10. DOI: 10.1109/HICSS.2010.347.

[111] OASIS. *Devices Profile for Web Services Version 1.1 - OASIS Standard*. 2009. URL: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html (besucht am 27. 10. 2020).

[112] Microsoft Corporation. *Devices Profile for Web Services*. Feb. 2006. URL: http://specs.xmlsoap.org/ws/2006/02/devprof/devicesprofile.pdf (besucht am 27. 10. 2020).

[113] OASIS. *Web Services Dynamic Discovery (WS-Discovery) Version 1.1*. 2009. URL: http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html (besucht am 27. 10. 2020).

[114] World Wide Web Consortium (W3C). *Web Services Eventing (WS-Eventing)*. 2006. URL: http://www.w3.org/Submission/2006/SUBM-WS-Eventing-20060315/.

[115] World Wide Web Consortium (W3C). *Web Services Policy (WS-Policy) 1.5 - Framework - W3C Recommendation*. 2007. URL: http://www.w3.org/TR/2007/REC-ws-policy-20070904/.

[116] World Wide Web Consortium (W3C). *Web Services Description Language (WSDL) 1.1 - W3C Note*. 2001. URL: http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[117] *OR.NET - Sichere dynamische Vernetzung in Operationssaal und Klinik (BMBF-gefördertes Leuchtturmprojekt)*. URL: www.ornet.org.

[118] Stephan Pöhlsen, Stefan Schlichting, Markus Strähle, Frank Franz und Christian Werner. „A Concept for a Medical Device Plug-and-play Architecture Based on Web Services“. In: *SIGBED Rev.* 6.2 (2009), 6:1–6:7. DOI: 10.1145/1859823.1859829.

[119] Stephan Pöhlsen, Stefan Schlichting und Christian Werner. „Praktische Umsetzung einer verteilten Zugriffskontrolle für Webservices anhand ihrer Eigenschaften“. In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 33.2 (Jan. 2010), S. 140. DOI: 10.1515/piko.2010.021.

[120] Stephan Pöhlsen, Winfried Schöch und Stefan Schlichting. „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 and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP 2011)*. 2011.

[121] David Gregorczyk, Timm Bubhaus und Stefan Fischer. „A proof of concept for medical device integration using Web Services“. In: *International Multi-Conference on Systems, Sygnals & Devices*. IEEE, März 2012, S. 1–6. DOI: 10.1109/SSD.2012.6198124.

[122] David Gregorczyk, Timm Bußhaus und Stefan Fischer. „Robust and Semi-automatic Electronic Health Record Dissemination Using the Devices Profile for Web Services“. In: *ICIW 2013, The Eighth International Conference on Internet and Web Applications and Services*. 2013, S. 38–44.

[123] Stephan Pöhlsen. „Entwicklung einer Service-orientierten Architektur zur vernetzten Kommunikation zwischen medizinischen Geräten, Systemen und Applikationen“. Dissertation. Institut für Telematik, Universität zu Lübeck, 2010.

[124] David Gregorczyk. „Technologien für eine interoperable und automatisierte Vernetzung von medizinischen IT-Systemen“. Dissertation. Institut für Telematik, Universität zu Lübeck, 2014.

[125] World Wide Web Consortium (W3C). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) - W3C Recommendation*. 2007. URL: http://www.w3.org/TR/2007/REC-soap12-part1-20070427/.

[126] OASIS. *SOAP-over-UDP Version 1.1 - OASIS Standard*. 2009. URL: http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.html.

[127] Zach Shelby. „Embedded web services“. In: *IEEE Wireless Communications* 17.6 (Dez. 2010), S. 52–57. DOI: 10.1109/MWC.2010.5675778.

[128] Guido Moritz, Dirk Timmermann, Regina Stoll und Frank Golatowski. „Encoding and Compression for the Devices Profile for Web Services“. In: *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2010, S. 514–519. DOI: 10.1109/WAINA.2010.91.

[129] Rumen Kyusakov, Henrik Makitaavola, Jerker Delsing und Jens Eliasson. „Efficient XML Interchange in factory automation systems“. In: *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, Nov. 2011, S. 4478–4483. DOI: 10.1109/IECON.2011.6120046.

[130] Audie Sumaray und S. Kami Makki. „A comparison of data serialization formats for optimal efficiency on a mobile platform“. In: *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication - ICUIMC '12*. New York, New York, USA: ACM Press, 2012, S. 1. DOI: 10.1145/2184751.2184810.

[131] World Wide Web Consortium (W3C). *Efficient XML Interchange (EXI) Format 1.0 (Second Edition) - W3C Recommendation*. 2014. URL: http://www.w3.org/TR/2014/REC-exi-20140211/.

[132] Yann Collet. *LZ4 - Extremely fast compression (Version 1.9.2)*. 2019. URL: https://github.com/lz4/lz4 (besucht am 27. 10. 2020).

[133] Jacob Ziv und Abraham Lempel. „A universal algorithm for sequential data compression“. In: *IEEE Transactions on Information Theory* 23.3 (Mai 1977), S. 337–343. DOI: 10.1109/TIT.1977.1055714.

[134] IEEE Standards Association. „11073-10201-2004 - ISO/IEEE International Standard for Health Informatics - Point-of-care medical device communication - Part 10201: Domain information model“. In: *ISO/IEEE 11073-10201:2004(E)* (Jan. 2004), S. 1–183. DOI: 10.1109/IEEESTD.2004.95742.

[135] „11073-10101:2004(E) - ISO/IEEE Health Informatics - Point-Of-Care Medical Device Communication - Part 10101: Nomenclature“. In: *ISO/IEEE 11073-10101:2004(E)* (2004), 0_1–492. DOI: 10.1109/IEEESTD.2004.95741.

[136] Integrating the Healthcare Enterprise (IHE). *IHE Patient Care Device (PCD) Technical Framework Volume 3 PCD TF-3 (Revision 8.0)*. 2018. URL: https://www.ihe.net/uploadedFiles/Documents/PCD/IHE_PCD_TF_Vol3.pdf.

[137] Thomas Neumuth. „Surgical process modeling“. In: *Innovative Surgical Sciences* 2.3 (Sep. 2017), S. 123–137. DOI: 10.1515/iss-2017-0005.

[138] Stefan Franke, Jürgen Meixensberger und Thomas Neumuth. „Multi-perspective workflow modeling for online surgical situation models“. In: *Journal of Biomedical Informatics* 54 (Apr. 2015), S. 158–166. DOI: 10.1016/j.jbi.2015.02.005.

[139] David Mills. *IETF RFC 1305 – Network Time Protocol (Version 3): Specification, Implementation and Analysis*. Techn. Ber. 1992. URL: https://tools.ietf.org/html/rfc1305.

[140] David Mills, Jack Burbank und William Kasch. *IEFT RFC 5905 – Network Time Protocol Version 4: Protocol and Algorithms Specification*. Techn. Ber. 2010. URL: https://tools.ietf.org/html/rfc5905.

[141] Kathleen Nichols, Steven Blake, Fred Baker und David Black. *IETF RFC 2474 – Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. Techn. Ber. 1998. URL: https://tools.ietf.org/html/rfc2474.

[142] Elwyn Davies, Mark A Carlson, Walter Weiss, David Black, Steven Blake und Zheng Wang. „IETF RFC 2475 – An Architecture for Differentiated Services“. In: (1998). URL: https://tools.ietf.org/html/rfc2475.

[143] Kristof Obermann und Martin Horneffer. *Datennetztechnologien für Next Generation Networks*. Wiesbaden: Vieweg+Teubner, 2009. DOI: 10.1007/978-3-8348-9963-7.

[144] Mirjana Radivojević und Petar Matavulj. *The Emerging WDM EPON*. Cham: Springer International Publishing, 2017. DOI: 10.1007/978-3-319-54224-9.

[145] Jon Postel. *IETF RFC 791 – Transmission control protocol*. Techn. Ber. Sep. 1981. URL: https://tools.ietf.org/html/rfc791.

[146] Victor Firoiu, Anna Charny und Bruce Davie. *IETF RFC 3246 – An Expedited Forwarding PHB (Per-Hop Behavior)*. Techn. Ber. März 2002. URL: https://tools.ietf.org/html/rfc3246.

[147] Dan Grossman. *IETF RFC 3260 – New Terminology and Clarifications for DiffServ*. Techn. Ber. Apr. 2002. URL: https://tools.ietf.org/html/rfc3260.

[148] Wikipedia - the free encyclopedia. *Differentiated services – Section: Expedited Forwarding*. URL: https://en.wikipedia.org/wiki/Differentiated_services#Expedited_Forwarding (besucht am 05. 11. 2018).

[149] Raul Aquino Santos. *Broadband Wireless Access Networks for 4G: Theory, Application, and Experimentation: Theory, Application, and Experimentation*. IGI Global, 2013.

[150] H. W. Barz und G. A. Bassett. *Multimedia Networks: Protocols, Design, and Applications*. Wiley, 2016. ISBN: 9781119090151.

[151] John Wroclawski, Walter Weiss, Juha Heinanen und Fred Baker. *IETF RFC 2597 – Assured Forwarding PHB Group*. Techn. Ber. Juni 1999. URL: https://tools.ietf.org/html/rfc2597.

[152] Simeon Medical. „Mobiler OP-Tisch mit hoher Integrationsfähigkeit“. In: *M&K - Management & Krankenhaus* (Nov. 2018), S. 13. URL: https://www.management-krankenhaus.de/file/track/15720/1.

[153] SCHÖLLY FIBEROPTIC GMBH. *Endoskopie-Kamera mit Option zur OP-Integration*. Mai 2019. URL: https://www.schoelly.de/de/news/endoskopie-kamera-mit-option-zur-op-integration/.

[154] Drägerwerk AG & Co. KGaA. *Anästhesiearbeitsplätze werden interoperabel – Bidirektionaler Austausch statt Daten-Einbahnstraße (Pressemitteilung Nr. 79)*. Nov. 2019. URL: https://www.draeger.com/Corporate/Content/79d_CAP.pdf.

[155] Drägerwerk AG & Co. KGaA. *Medizinische Gerätekonnektivität – basierend auf IEEE 11073 SDC*. 2020. URL: https://www.draeger.com/de_de/Hospital/SDC (besucht am 27. 10. 2020).

[156] Alberto Huertas Celdran, Felix J. Garcia Clemente, James Weimer und Insup Lee. „ICE++: Improving Security, QoS, and High Availability of Medical Cyber-Physical Systems through Mobile Edge Computing“. In: *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE, Sep. 2018, S. 1–8. DOI: 10.1109/HealthCom.2018.8531185.

[157] U.S. Food & Drug Administration (FDA). *FDA Recognized Consensus Standards Database*. URL: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfStandards/search.cfm.

[158] Merle Baake, Josef Ingenerf und Björn Andersen. „Using Data Distribution Service for IEEE 11073-10207 Medical Device Communication“. In: *MobiHealth 2019 - 8th EAI International Conference on Wireless Mobile Communication and Healthcare*. Dublin, Nov. 2020, S. 127–139. DOI: 10.1007/978-3-030-49289-2_10.

[159] David Gregorczyk und Jan Lukas Deichmann. *SDCri – SDC Reference Implementation*. URL: https://gitlab.com/sdc-suite/sdc-ri (besucht am 27. 10. 2020).

[160] Drägerwerk AG & Co. KGaA. *sdc11073 Library – An ISO/IEEE 11073 SDC Implementation: Made in Python for testing purposes and demonstration environments*. 2020. URL: https://github.com/Draegerwerk/sdc11073 (besucht am 27. 10. 2020).

[161] Drägerwerk AG & Co. KGaA. *openSDC Software Library*. URL: https://sourceforge.net/projects/opensdc/ (besucht am 27. 10. 2020).

[162] SurgiTAIX AG. *SDCLib/J Java Software Library*. URL: https://bitbucket.org/surgitaix/sdclib/ (besucht am 27. 10. 2020).

[163] Andreas Besting. *SDCLib/J Java Software Library*. URL: https://bitbucket.org/besting-it/sdclibcontrib (besucht am 27. 10. 2020).

[164] SurgiTAIX AG. *SDCLib/C – Service-oriented Devices Connectivity Library: A reference implementation of the IEEE 11073 SDC Library*. URL: https://github.com/surgitaix/sdclib (besucht am 27. 10. 2020).

[165] Robert J. Kosinski. *A Literature Review on Reaction Time*. Techn. Ber. 2013.

[166] Francis Galton. „Exhibition of Instruments (1) for Testing Perception of Differences of Tint, and (2) for Determining Reaction-Time“. In: *The Journal of the Anthropological Institute of Great Britain and Ireland* 19 (1890), S. 27–29. DOI: 10.2307/2842529.

[167] K. von Fieandt, A. Huhtala, P. Kullberg und K. Saarl. „Personal tempo and phenomenal time at different age levels.“ In: *Reports from the Psychological Institute, No. 2, University of Helsinki* 2 (1956).

[168] A. T. Welford. „Choice reaction time: Basic concepts“. In: *Reaction times* (1980), S. 73–128. URL: https://ci.nii.ac.jp/naid/10008956132/en/.

[169] J. T. Brebner und A. T. Welford. „Introduction: an historical background sketch“. In: *Reaction times* (1980), S. 1–23. URL: https://ci.nii.ac.jp/naid/10016627318/en/.

[170] David L. Woods, John M. Wyma, E. William Yund, Timothy J. Herron und Bruce Reed. „Factors influencing the latency of simple reaction time“. In: *Frontiers in Human Neuroscience* 9 (März 2015). DOI: 10.3389/fnhum.2015.00131.

[171] James T. Eckner, Jeffrey S. Kutcher und James K. Richardson. „Pilot Evaluation of a Novel Clinical Test of Reaction Time in National Collegiate Athletic Association Division I Football Players“. In: *Journal of Athletic Training* 45.4 (Juli 2010), S. 327–332. DOI: 10.4085/1062-6050-45.4.327.

[172] Aditya Jain, Ramta Bansal, Kumar Avnish und K D Singh. „A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students“. In: *International Journal of Applied and Basic Medical Research* 5.2 (2015), S. 124–127. DOI: 10.4103/2229-516X.157168.

[173] Marcel Pfister, Jaw-Chyng L. Lue, Francisco R. Stefanini, Paulo Falabella, Laurie Dustin, Michael J. Koss und Mark S. Humayun. „Comparison of Reaction Response Time between Hand and Foot Controlled Devices in Simulated Microsurgical Testing“. In: *BioMed Research International* 2014 (2014), S. 1–8. DOI: 10.1155/2014/769296.

[174] Grenville Armitage. „An experimental estimation of latency sensitivity in multiplayer quake 3“. In: *The 11th IEEE International Conference on Networks, 2003. ICON2003.* IEEE, 2003, S. 137–141. DOI: 10.1109/ICON.2003.1266180.

[175] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu und Mark Claypool. „The effects of loss and latency on user performance in unreal tournament 2003®“. In: *Proceedings of ACM SIGCOMM 2004 workshops on NetGames '04 Network and system support for games - SIGCOMM 2004 Workshops*. New York, New York, USA: ACM Press, 2004, S. 144. DOI: 10.1145/1016540.1016556.

[176] Lothar Pantel und Lars C. Wolf. „On the impact of delay on real-time multiplayer games“. In: *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video - NOSSDAV '02*. New York, New York, USA: ACM Press, 2002, S. 23. DOI: 10.1145/507670.507674.

[177] James Nichols und Mark Claypool. „The effects of latency on online madden NFL football“. In: *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video - NOSSDAV '04*. New York, New York, USA: ACM Press, 2004, S. 146. DOI: 10.1145/1005847.1005879.

[178] Kjetil Raaen, Ragnhild Eg und Carsten Griwodz. „Can gamers detect cloud delay?“ In: *2014 13th Annual Workshop on Network and Systems Support for Games*. IEEE, Dez. 2014, S. 1–3. DOI: 10.1109/NetGames.2014.7008962.

[179] Werner Stangl. *Stichwort: 'Sense of Agency'. Online Lexikon für Psychologie und Pädagogik.* 2019. URL: https://lexikon.stangl.eu/16744/sense-of-agency/ (besucht am 27. 10. 2020).

[180] Thomas Waltemate, Irene Senna, Felix Hülsmann, Marieke Rohde, Stefan Kopp, Marc Ernst und Mario Botsch. „The impact of latency on perceptual judgments and motor performance in closed-loop interaction in virtual reality“. In: *Proceedings of the 22nd ACM Conference*

on Virtual Reality Software and Technology - VRST '16. New York, New York, USA: ACM Press, 2016, S. 27–35. DOI: 10.1145/2993369.2993381.

[181] C. Farrer, G. Valentin und J.M. Hupé. „The time windows of the sense of agency". In: *Consciousness and Cognition* 22.4 (Dez. 2013), S. 1431–1441. DOI: 10.1016/j.concog.2013.09.010.

[182] Marieke Rohde, Meike Scheller und Marc O. Ernst. „Effects can precede their cause in the sense of agency". In: *Neuropsychologia* 65 (Dez. 2014), S. 191–196. DOI: 10.1016/j.neuropsychologia.2014.10.011.

[183] Yan Chen, Toni Farley und Nong Ye. „QoS Requirements of Network Applications on the Internet". In: *Information Knowledge Systems Management* 4.1 (2004), S. 55–76.

[184] D. Niyato, E. Hossain und J. Diamond. „IEEE 802.16/WiMAX-based broadband wireless access and its application for telemedicine/e-health services". In: *IEEE Wireless Communications* 14.1 (Feb. 2007), S. 72–83. DOI: 10.1109/MWC.2007.314553.

[185] Maulin Patel und Jianfeng Wang. „Applications, challenges, and prospective in emerging body area networking technologies". In: *IEEE Wireless Communications* 17.1 (Feb. 2010), S. 80–88. DOI: 10.1109/MWC.2010.5416354.

[186] Lea Skorin-Kapov und Maja Matijasevic. „Analysis of QoS Requirements for e-Health Services and Mapping to Evolved Packet System QoS Classes". In: *International Journal of Telemedicine and Applications* 2010 (2010), S. 1–18. DOI: 10.1155/2010/628086.

[187] Song Xu, Manuela Perez, Kun Yang, Cyril Perrenot, Jacques Felblinger und Jacques Hubert. „Determination of the latency effects on surgical performance and the acceptable latency levels in telesurgery using the dV-Trainer® simulator". In: *Surgical Endoscopy* 28.9 (Sep. 2014), S. 2569–2576. DOI: 10.1007/s00464-014-3504-z.

[188] Manuela Perez, Song Xu, Sanket Chauhan, Alyssa Tanaka, Khara Simpson, Haidar Abdul-Muhsin und Roger Smith. „Impact of delay on telesurgical performance: study on the robotic simulator dV-Trainer". In: *International Journal of Computer Assisted Radiology and Surgery* 11.4 (Apr. 2016), S. 581–587. DOI: 10.1007/s11548-015-1306-y.

[189] Qi Zhang, Jianhui Liu und Guodong Zhao. „Towards 5G Enabled Tactile Robotic Telesurgery". In: *arXiv preprint* (März 2018). arXiv: 1803.03586.

[190] Mohamad Eid, Jongeun Cha und Abdulmotaleb El Saddik. „Admux: An Adaptive Multiplexer for Haptic–Audio–Visual Data Communication". In: *IEEE Transactions on Instrumentation and Measurement* 60.1 (Jan. 2011), S. 21–31. DOI: 10.1109/TIM.2010.2065530.

[191] Burak Cizmeci, Xiao Xu, Rahul Chaudhari, Christoph Bachhuber, Nicolas Alt und Eckehard Steinbach. „A Multiplexing Scheme for Multimodal Teleoperation". In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 13.2 (Apr. 2017), S. 1–28. DOI: 10.1145/3063594.

[192] Taku Hachisu und Hiroyuki Kajimoto. „Vibration Feedback Latency Affects Material Perception During Rod Tapping Interactions". In: *IEEE Transactions on Haptics* 10.2 (Apr. 2017), S. 288–295. DOI: 10.1109/TOH.2016.2628900.

[193] *DIN EN 60601-2-54/A2:2017-11 – Medizinische elektrische Geräte - Teil 2-54: Besondere Festlegungen für die Sicherheit und die wesentlichen Leistungsmerkmale von Röntgeneinrichtungen für Radiographie und Radioskopie*. 2017.

[194] DGUV Test Prüf- und Zertifizierungsstelle Elektrotechnik. *GS-ET-22 Grundsätze für die Prüfung und Zertifizierung von Elektromechanischen Zustimmungsschaltern und Zustimmungseinrichtungen mit und ohne Anlaufsteuerung*. 2016.

[195] Signove Corp. *Antidote: IEEE 11073-20601 Stack*. 2014. URL: http://oss.signove.com/index.php/Antidote:_IEEE_11073-20601_stack (besucht am 27. 10. 2020).

[196] *openSDC*. URL: https://gitlab.amd.e-technik.uni-rostock.de/opensdc/opensdc-biceps-ornet/commit/a294af33ad4e86830036397f3532a275cb340a4a (besucht am 27. 10. 2020).

[197] SurgiTAIX AG. *SoftICE - Software for the Integrated Clinical Environment (ICE)*. URL: https://bitbucket.org/surgitaix/sdclib/commits/529a75b558a6dd402fafe4fa0132fb08f5461ca1?at=master (besucht am 27. 10. 2020).

[198] SurgiTAIX AG. *Open Surgical Communication Library (OSCLib)*. URL: https://github.com/surgitaix/sdclib/commit/df860802c19522bc59afbd9007e9d06d6c7b174b (besucht am 27. 10. 2020).

[199] Henning Puttnies, Björn Konieczek, Jakob Heller, Dirk Timmermann und Peter Danielis. „Algorithmic approach to estimate variant software latencies for latency-sensitive networking“. In: *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, Okt. 2016, S. 1–7. DOI: 10.1109/IEMCON.2016.7746281.

[200] Alexander Beifus, Daniel Raumer, Paul Emmerich, Torsten M. Runge, Florian Wohlfart, Bernd E. Wolfinger und Georg Carle. „A study of networking software induced latency“. In: *2015 International Conference and Workshops on Networked Systems (NetSys)*. IEEE, März 2015, S. 1–8. DOI: 10.1109/NetSys.2015.7089065.

[201] Lukas M Märdian. „What's New in the Linux Network Stack?“ In: *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)* 2 (2014), S. 6. URL: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2015-03-1.pdf#page=43.

[202] Ken Inoue, Davide Pasetto, Karol Lynch, Massimiliano Meneghin, Kay Muller und John Sheehan. „Low-latency and high bandwidth TCP/IP protocol processing through an integrated HW/SW approach“. In: *2013 Proceedings IEEE INFOCOM*. IEEE, Apr. 2013, S. 2967–2975. DOI: 10.1109/INFCOM.2013.6567108.

[203] Wenji Wu, Matt Crawford und Mark Bowden. „The performance analysis of linux networking – Packet receiving“. In: *Computer Communications* 30.5 (März 2007), S. 1044–1057. DOI: 10.1016/j.comcom.2006.11.001.

[204] Steen Larsen, Parthasarathy Sarangam, Ram Huggahalli und Siddharth Kulkarni. „Architectural Breakdown of End-to-End Latency in a TCP/IP Network“. In: *International Journal of Parallel Programming* 37.6 (Dez. 2009), S. 556–571. DOI: 10.1007/s10766-009-0109-6.

[205] OpenJDK. *AArch32 Port Project*. URL: http://openjdk.java.net/projects/aarch32-port/ (besucht am 27. 10. 2020).

[206] Björn Konieczek, Michael Rethfeldt, Frank Golatowski und Dirk Timmermann. „Real-Time Communication for the Internet of Things using jCoAP“. In: *18th IEEE Symposium on Real-Time Computing (ISORC)*. IEEE-ISORC. 2015, S. 134–141. DOI: 10.1109/ISORC.2015.35.

[207] The Real-Time for Java Experts Group. *The Real TimeSpecification for Java (RTSJ) - Version 1.0.2*. URL: https://www.rtsj.org/ (besucht am 27. 10. 2020).

[208] G. Bollella und J. Gosling. „The real-time specification for Java“. In: *Computer* 33.6 (Juni 2000), S. 47–54. DOI: 10.1109/2.846318.

[209] A. Corsaro und D.C. Schmidt. „Evaluating real-time Java features and performance for real-time embedded systems“. In: *Proceedings. Eighth IEEE Real-Time and Embedded Technology and Applications Symposium*. San Kose, CA, IEEE Comput. Soc, 2002, S. 90–100. DOI: 10.1109/RTTAS.2002.1137384.

[210] Andy Georges, Dries Buytaert und Lieven Eeckhout. „Statistically rigorous java performance evaluation“. In: *Proceedings of the 22nd annual ACM SIGPLAN conference on Object oriented programming systems and applications - OOPSLA '07*. New York, New York, USA: ACM Press, 2007, S. 57. DOI: 10.1145/1297027.1297033.

[211] Eike Schweissguth, Peter Danielis, Christoph Niemann und Dirk Timmermann. „Application-aware industrial ethernet based on an SDN-supported TDMA approach“. In: *2016 IEEE World Conference on Factory Communication Systems (WFCS)*. IEEE, Mai 2016, S. 1–8. DOI: 10.1109/WFCS.2016.7496496.

[212] James Connors. *Comparing Linux/Arm JVMs Revisited*. 2013. URL: https://blogs.oracle.com/jtc/comparing-linuxarm-jvms-revisited (besucht am 27. 10. 2020).

[213] *MEDNOVO Medical Software Solutions GmbH (After a merger on 7/26/2016, KARL STORZ SE & Co. KG is universal successor of MEDNOVO Medical Software Solutions GmbH)*. Berlin. URL: http://www.mednovo.de.

[214] A.-M. von Saucken (geb. Seyffert), S. Donner und M. Kraft. „Ergonomic Problems Originating in the Use of High-Frequency and Ultrasonic Medical Devices“. In: *Biomedical Engineering / Biomedizinische Technik* 57.SI-1 Track-J (Jan. 2012). DOI: 10.1515/bmt-2012-4042.

[215] International Electrotechnical Commission (IEC). *IEC 61784-3: Industrial communication networks – Profiles – Part 3: Functional safety fieldbuses – General rules and profile definitions*.

[216] DGUV Test Prüf- und Zertifizierungsstelle Elektrotechnik. *GS-ET-26 Grundsätze für die Prüfung und Zertifizierung von Bussystemen für die Übertragung sicherheitsbezogener Nachrichten*. 2014.

[217] Andrew S Tanenbaum und David J. Wetherall. *Computer Networks (5th Edition)*. 5. Aufl. Boston, Massachusetts: Pearson Education, Inc., publishing as Prentice Hall, 2010. ISBN: 978-0-13-212695-3.

[218] International Electrotechnical Commission (IEC). *IEC 61800-5-2:2007 – Adjustable speed electrical power drive systems - Part 5-2: Safety requirements - Functional*. 2007.

[219] Armin Janß, Anna Vitting, Benjamin Strathen, Melanie Strake und Klaus Radermacher. „Intraoperative Workflow Optimization by Using a Universal Foot Switch for Open Inte-

grated OR Systems in Orthopaedic Surgery“. In: *CAOS 2017. 17th Annual Meeting of the InternationalSociety for Computer Assisted Orthopaedic Surgery*. 2017, S. 248–253.

[220] Anna Vitting, Armin Janß, Benjamin Strathen, Melanie Strake und Klaus Radermacher. „Further Development and Evaluation of a Universal Foot Switch for Diverse Medical Disciplines within the Framework of an Open Integration Concept for the Operation Theatre of the Future“. In: 2018, S. 438–449. DOI: 10.1007/978-3-319-60483-1_45.

[221] Joshua Valcarcel. *Photo: Cataract Eye Surgery using an Operating Microscope*. 2009. URL: https://commons.wikimedia.org/wiki/File:US_Navy_080607-N-9689V-008_Cmdr._Kenneth_Kubis_and_U.S._Air_Force_Capt._Tighe_Richardson_use_an_operating_microscope_while_performing_cataract_eye_surgery_to_return_sight_to_Marylin_Kansi,_a_12-year-old_girl_from_Cotabato.jpg (besucht am 27. 10. 2020).

[222] Hagit Attiya und Jennifer Welch. *Distributed Computing*. Wiley Series on Parallel and Distributed Computing. Hoboken, NJ, USA: John Wiley & Sons, Inc., Apr. 2004. DOI: 10.1002/0471478210.

[223] Navneet Malpani, Jennifer L. Welch und Nitin Vaidya. „Leader election algorithms for mobile ad hoc networks“. In: *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications - DIALM '00*. New York, New York, USA: ACM Press, 2000, S. 96–103. DOI: 10.1145/345848.345871.

[224] Martin Leucker, Malte Schmitz und Danilo à Tellinghusen. „Runtime Verification for Interconnected Medical Devices“. In: *International Symposium on Leveraging Applications of Formal Methods (ISoLA 2016): Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications*. 2016, S. 380–387. DOI: 10.1007/978-3-319-47169-3_29.

[225] Herb Sutter. „atomic<> Weapons: The C++ Memory Model and Modern Hardware“. In: *C++ and Beyond 2012*. Asheville, NC, 2012. URL: https://onedrive.live.com/?authkey=%21AMtj_EflYn2507c&cid=4E86B0CF20EF15AD&id=4E86B0CF20EF15AD%2124884&parId=4E86B0CF20EF15AD%21180&o=OneUp.

[226] Ali Arsanjani, Grady Booch, Toufic Boubez, Paul C. Brown, David Chappell, John DeVadoss, Thomas Erl, Nicolai Josuttis, Dirk Krafzig, Mark Little, Brian Loesgen, Anne Thomas Manes, Joe McKendrick, Steve Ross-Talbot, Stefan Tilkov, Clemens Utschig-Utschig und Herbjörn Wilhelmsen. *SOA Manifesto*. 2009. URL: http://www.soa-manifesto.org/.

[227] Maximilian Merkel und Stefan Schlichting. *SDC Conformance Principles*. Techn. Ber. 2019.

[228] Deutscher Bundestag. *Verordnung über das Errichten, Betreiben und Anwenden vonMedizinprodukten (Medizinprodukte-Betreiberverordnung – MPBetreibV)*. 1998.

[229] Emergency Care Research Institute (ECRI). „TOP 10 HOSPITAL TECHNOLOGY ISSUES: C-Suite Watch List for 2009 and Beyond“. In: *Health Devices* (Nov. 2008).

[230] Emergency Care Research Institute (ECRI). „Top 10 Health Technology Hazards for 2010“. In: *Health Devices* (Nov. 2009).

[231] Emergency Care Research Institute (ECRI). „Top 10 Health Technology Hazards for 2011“. In: *Health Devices* 39.11 (Nov. 2010).

[232] Emergency Care Research Institute (ECRI). „Top 10 Health Technology Hazards for 2012“. In: *Health Devices* (Nov. 2011).

[233] Emergency Care Research Institute (ECRI). „Top 10 Health Technology Hazards for 2013“. In: *Health Devices* 41.11 (Nov. 2012). URL: https://www.ecri.org/Resources/Whitepapers_and_reports/2013_Health_Devices_Top_10_Hazards.pdf.

[234] Emergency Care Research Institute (ECRI). „Top 10 Health Technology Hazards for 2014“. In: *Health Devices* 42.11 (Nov. 2013). URL: https://www.ecri.org/Resources/Whitepapers_and_reports/2014_Top_10_Hazards_Executive_Brief.pdf.

[235] Emergency Care Research Institute (ECRI). „Top 10 Health Technology Hazards for 2015“. In: *Health Devices* (Nov. 2014). URL: https://www.ecri.org/Resources/Whitepapers_and_reports/Top_Ten_Technology_Hazards_2015.pdf.

[236] Emergency Care Research Institute (ECRI). „Executive Brief: Top 10 Health Technology Hazards for 2016“. In: *Health Devices* (Nov. 2015). URL: https://www.ecri.org/Resources/Whitepapers_and_reports/2016_Top_10_Hazards_Executive_Brief.pdf.

[237] Emergency Care Research Institute (ECRI). „Executive Brief: Top 10 Health Technology Hazards for 2017“. In: *Health Devices* (Nov. 2016). URL: https://www.ecri.org/Resources/Whitepapers_and_reports/Haz17.pdf.

[238] Emergency Care Research Institute (ECRI). „Executive Brief: Top 10 Health Technology Hazards for 2018“. In: *Health Devices* (Nov. 2017). URL: https://www.ecri.org/Resources/Whitepapers_and_reports/Haz_18.pdf.

[239] Emergency Care Research Institute (ECRI). „2019 Top 10 Health Technology Hazards: Executive Brief“. In: *Health Devices* (Nov. 2018). URL: https://www.ecri.org/Resources/Whitepapers_and_reports/Haz_19.pdf.

[240] The Joint Commission (TJC). *Sentinel Event Policy and Procedures*. Juni 2017. URL: https://www.jointcommission.org/sentinel_event_policy_and_procedures/ (besucht am 27. 10. 2020).

[241] P. C. A. Kam, A. C. Kam und J. F. Thompson. „Noise pollution in the anaesthetic and intensive care environment“. In: *Anaesthesia* 49.11 (Nov. 1994), S. 982–986. DOI: 10.1111/j.1365-2044.1994.tb04319.x.

[242] Koninklijke Philips N.V. *Alarm management - from signal to action*. Techn. Ber. 2016. URL: https://images.philips.com/is/content/PhilipsConsumer/Campaigns/HC20140401_DG/Documents/Philips%20hospital%20Alarm%20management%20Solution%20brochure.pdf.

[243] GE Healthcare by amandic. *GE Ascom Patientenalarm-Managementlösung*. 2020. URL: https://www.anandic.com/de/produkte---loesungen/patientenmonitoring/carescape-modulares-monitoring/ge-ascom-verteiltes-alarmmanagement (besucht am 27. 10. 2020).

[244] Drägerwerk AG & Co. KGaA. *Alarmmanagement by Dräger*. 2019. URL: https://www.draeger.com/de_de/Hospital/Application-And-Consulting/Alarmmanagement (besucht am 21. 08. 2019).

[245] tetronik GmbH. *Verteiltes Alarmsystem mit DAKSmed nach IEC60601-1-8*. 2019. URL: https://www.tetronik.com/de/produkte/daksmed.html (besucht am 27. 10. 2020).

[246] Tobey Clark. „Closing the Clinical Alarm Gap“. In: *24x7 Solutions for Healthcare Technology Management* (2010). URL: http://www.24x7mag.com/2010/09/closing-the-clinical-alarm-gap/.

[247] *DIN EN 60601-1-8: Medizinische elektrische Geräte - Teil 1-8: Allgemeine Festlegungen für die Sicherheit einschließlich der wesentlichen Leistungsmerkmale - Ergänzungsnorm: Alarmsysteme - Allgemeine Festlegungen, Prüfungen und Richtlinien für Alarmsysteme*. Apr. 2014.

[248] Materna Information & Communications SE. *JMEDS (Java Multi Edition DPWS Stack)*. URL: https://sourceforge.net/projects/ws4d-javame/ (besucht am 27. 10. 2020).

[249] Jonas H Pfeiffer, Max E Dingler, Christian Dietz und Tim C Lueth. „Requirements and architecture design for open real-time communication in the operating room“. In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Zhuhai: IEEE, Dez. 2015, S. 458–463. DOI: 10.1109/ROBIO.2015.7418810.

[250] Jiaxi Shi, G. Strauss, S. Heininger und T. C. Lueth. „Surgical assistance for instruments' power control based on navigation and neuromonitoring“. In: *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, Aug. 2011, S. 2115–2118. DOI: 10.1109/IEMBS.2011.6090394.

[251] Mathias Hofer, Tim Lueth, Andreas Dietz und Gero Strauss. „Potential of the Navigated Controlled Surgery at the Lateral Skull Base with the Navigated Control Unit (NCU 2.0)“. In: *Studies in Health Technology and Informatics – Medicine Meets Virtual Reality 19: NextMed* 173 (2012), S. 183–185. DOI: 10.3233/978-1-61499-022-2-183.

[252] George A Ditzel und Paul Didier. „Time Sensitive Network (TSN) Protocols and use in EtherNet/IP Systems“. In: *2015 ODVA Industry Conference & 17th Annual Meeting*. Frisco, Texas, 2015.

[253] René Hummen, Stephan Kehrer und Oliver Kleineberg. *White Paper: TSN – Time Sensitive Networking*. 2017. URL: https://www.belden.com/resources/knowledge/white-papers/tsn-time-sensitive-networking-bc-lp?__hstc=167740065.393e9d68e8b4cb000baea658ed44b154.1524229634898.1524229634898.1524229634898.1&__hssc=167740065.1.1524229634900&__hsfp=305609680.

[254] Eike Schweissguth, Peter Danielis, Dirk Timmermann, Helge Parzyjegla und Gero Mühl. „ILP-based joint routing and scheduling for time-triggered networks“. In: *Proceedings of the 25th International Conference on Real-Time Networks and Systems - RTNS '17*. New York, New York, USA: ACM Press, 2017, S. 8–17. DOI: 10.1145/3139258.3139289.

[255] Eike Schweissguth, Helge Parzyjegla, Peter Danielis, Gero Mühl und Dirk Timmermann. „Realtime Publish/Subscribe for the Industrial Internet of Things“. In: *In Proceedings of the 2. KuVS Fachgespräch "Network Softwarization"*. Tübingen, Apr. 2020, S. 2. DOI: 10.15496/publikation-41786.

[256] Eike Schweissguth, Helge Parzyjegla, Peter Danielis, Gero Mühl und Dirk Timmermann. „ILP-Based Routing and Scheduling of Multicast Realtime Traffic in Time-Sensitive

Networks“. In: *In Proceedings of the 26th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. Aug. 2020.

[257] Marcus Köny. „Entscheidungsunterstützungssysteme für den anästhesiologischen Arbeitsplatz der Zukunft auf Basis vernetzter Medizingeräte“. Dissertation. 2015. ISBN: 978-3-8440-4067-8.

[258] B. Middleton, D. F. Sittig und A. Wright. „Clinical Decision Support: a 25 Year Retrospective and a 25 Year Vision“. In: *Yearbook of Medical Informatics* 25.S 01 (Aug. 2016), S103–S116. DOI: 10.15265/IYS-2016-s034.

[259] Stefan Franke und Thomas Neumuth. „Adaptive surgical process models for prediction of surgical work steps from surgical low-level activities“. In: *6th Workshop on Modeling and Monitoring of Computer Assisted Interventions (M2CAI) at the 18th International Conference on Medical Image Computing and Computer Assisted Interventions (MICCAI)*. 2015.

B Wissenschaftliche Konferenz- und Journal-Publikationen des Autors

[1] Björn Andersen, Martin Kasparick, Kathrin Riech, Stephan Klöckner, Anton Keller, Lars Mündermann, Julian Maier-Holzberg, Dirk Timmermann und Josef Ingenerf. „Service-Oriented Medical Device Connectivity: Particular Standards for Endoscopic Surgery“. In: *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. Montréal (virtual): IEEE, Juli 2020, S. 5649–5652. DOI: 10.1109/EMBC44109.2020.9175932.

[2] Benjamin Rother, Martin Kasparick, Eike Schweißguth, Frank Golatowski und Dirk Timmermann. „Automatic Configuration of a TSN Network for SDC-based Medical Device Networks“. In: *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*. Porto: IEEE, Apr. 2020, S. 1–8. DOI: 10.1109/WFCS47810.2020.9114471.

[3] Martin Kasparick, Björn Butzin, Frank Golatowski, Jonas Pabst, Hans-Joachim Cappius, Peter Westerhoff, Björn Andersen und Dirk Timmermann. „From IEEE 11073 SDC Device Specializations to Assistive Systems: Rule-based Data Analysis for Minimal Invasive Surgery“. In: *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*. Sharm El Sheik: IEEE, Dez. 2019, S. 1–7. DOI: 10.1109/SmartNets48225.2019.9069774.

[4] Juliane Neumann, Stefan Franke, Max Rockstroh, Martin Kasparick und Thomas Neumuth. „Extending BPMN 2.0 for intraoperative workflow modeling with IEEE 11073 SDC for description and orchestration of interoperable, networked medical devices“. In: *International Journal of Computer Assisted Radiology and Surgery (Int J CARS)* 14.8 (Aug. 2019), S. 1403–1413. DOI: 10.1007/s11548-019-01982-6.

[5] Frank Golatowski, Björn Butzin, Tim Brockmann, Thorsten Schulz, Martin Kasparick, Yuhong Li, Rahim Rahmani, Aviv Haber, Mustafa Sakalsiz und Ozer Aydemir. „Challenges and Research Directions for Blockchains in the Internet of Things“. In: *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*. IEEE, Mai 2019, S. 712–717. DOI: 10.1109/ICPHYS.2019.8780270.

[6] Martin Kasparick, Björn Andersen, Stefan Franke, Max Rockstroh, Frank Golatowski, Dirk Timmermann, Josef Ingenerf und Thomas Neumuth. „Enabling artificial intelligence in high acuity medical environments“. In: *Minimally Invasive Therapy & Allied Technologies* (Apr. 2019), S. 1–7. DOI: 10.1080/13645706.2019.1599957.

[7] Martin Kasparick, Malte Schmitz, Björn Andersen, Max Rockstroh, Stefan Franke, Stefan Schlichting, Frank Golatowski und Dirk Timmermann. „OR.NET: a service-oriented architecture for safe and dynamic medical device interoperability“. In: *Biomedical Engineering / Biomedizinische Technik* 63.1 (Feb. 2018), S. 11–30. DOI: 10.1515/bmt-2017-0020.

[8] Raluca Dees, Angela Merzweiler, Gerd Schneider, Martin Kasparick, Lars Mündermann, Janko Ahlbrandt, Martin Wagner, Hannes Kenngott, Beat Müller-Stich und Björn Bergh. „Implementing, Connecting, and Evaluating a Standard-Based Integrated Operating Room within a German University Hospital“. In: *ACI Open* 02.01 (Jan. 2018), e10–e20. DOI: 10.1055/s-0038-1639604.

[9] Stefan Franke, Max Rockstroh, Martin Kasparick und Thomas Neumuth. „A Method for the Context-Aware Assignment of Medical Device Functions to Input Devices in Integrated Operating Rooms“. In: *International Workshop on Computer-Assisted and Robotic Endoscopy Workshop on Clinical Image-Based Procedures International Workshop on OR 2.0 Context-Aware Operating Theaters International Workshop on Skin Image Analysis*. Granada, Sep. 2018, S. 12–19. DOI: 10.1007/978-3-030-01201-4_2.

[10] Martin Kasparick, Björn Andersen, Hannes Ulrich, Stefan Franke, Erik Schreiber, Max Rockstroh, Frank Golatowski, Dirk Timmermann, Josef Ingenerf und Thomas Neumuth. „IEEE 11073 SDC and HL7 FHIR - Emerging Standards for Interoperability of Medical Systems“. In: *International Journal of Computer Assisted Radiology and Surgery (2018) 13(Suppl 1): CARS 2018 - Computer Assisted Radiology and Surgery: Proceedings of the 32nd International Congress and Exhibition* 13.Suppl 1 (Juni 2018), S. 135–136. DOI: 10.1007/s11548-018-1766-y.

[11] Andreas Besting, Sebastian Bürger, Martin Kasparick, Benjamin Strathen und Frank Portheine. „Software design and implementation concepts for an interoperable medical communication framework“. In: *Biomedical Engineering / Biomedizinische Technik* 63.1 (Feb. 2018), S. 49–56. DOI: 10.1515/bmt-2017-0012.

[12] Andreas Besting, Dominik Stegemann, Sebastian Bürger, Martin Kasparick, Benjamin Strathen und Frank Portheine. „Concepts for Developing Interoperable Software Frameworks Implementing the New IEEE 11073 SDC Standard Family“. In: *CAOS 2017. 17th Annual Meeting of the International Society for Computer Assisted Orthopaedic Surgery*. Bd. 1. EPiC Series in Health Sciences. Aachen, 2017, S. 258–263.

[13] Jonas H. Pfeiffer, Martin Kasparick, Benjamin Strathen, Christian Dietz, Max E. Dingler, Tim C. Lueth, Dirk Timmermann, Klaus Radermacher und Frank Golatowski. „OR.NET RT: how service-oriented medical device architecture meets real-time communication“. In: *Biomedical Engineering / Biomedizinische Technik* 63.1 (Feb. 2018), S. 81–93. DOI: 10.1515/bmt-2017-0016.

[14] Max Rockstroh, Stefan Franke, Mathias Hofer, Armin Will, Martin Kasparick, Björn Andersen und Thomas Neumuth. „OR.NET: multi-perspective qualitative evaluation of an integrated operating room based on IEEE 11073 SDC“. In: *International Journal of Computer Assisted Radiology and Surgery* 12.8 (Aug. 2017), S. 1461–1469. DOI: 10.1007/s11548-017-1589-2.

[15] Martin Kasparick, Frank Golatowski und Dirk Timmermann. „A safe and interoperable distributed alarm notification system for PoC medical devices using IEEE 11073 SDC“. In: *2017 IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT)*. Bethesda, MD: IEEE, Nov. 2017, S. 257–261. DOI: 10.1109/HIC.2017.8227633.

[16] Manuel Vossel, Benjamin Strathen, Martin Kasparick, Meiko Müller, Klaus Radermacher, Matías de La Fuente und Armin Janß. „Integration of Robotic Applications in Open and Safe Medical Device IT Networks Using IEEE 11073 SDC“. In: *CAOS 2017. 17th Annual Meeting of the International Society for Computer Assisted Orthopaedic Surgery*. Bd. 1. EPiC Series in Health Sciences. Aachen, 2017, S. 254–257.

[17] Martin Kasparick, Benjamin Beichler, Björn Konieczek, Andreas Besting, Michael Rethfeldt, Frank Golatowski und Dirk Timmermann. „Measuring Latencies of IEEE 11073 Compliant Service-Oriented Medical Device Stacks“. In: *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*. Beijing: IEEE, Okt. 2017, S. 8640–8647. DOI: 10.1109/IECON.2017.8217518.

[18] Björn Andersen, Martin Kasparick, Hannes Ulrich, Stefan Franke, Jan Schlamelcher, Max Rockstroh und Josef Ingenerf. „Connecting the clinical IT infrastructure to a service-oriented architecture of medical devices“. In: *Biomedical Engineering / Biomedizinische Technik* 63.1 (Feb. 2018), S. 57–68. DOI: 10.1515/bmt-2017-0021.

[19] Martin Kasparick, Max Rockstroh, Stefan Schlichting, Frank Golatowski und Dirk Timmermann. „Mechanism for safe remote activation of networked surgical and PoC devices using dynamic assignable controls“. In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Orlando, FL: IEEE, Aug. 2016, S. 2390–2394. DOI: 10.1109/EMBC.2016.7591211.

[20] Björn Andersen, Martin Kasparick, Simon Baumhof, Frank Golatowski und Josef Ingenerf. „Definition and Validation of Surgical Device Specialisations for a Service-Oriented Medical Device Architecture“. In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Orlando, FL, Aug. 2016.

[21] Martin Kasparick, Malte Schmitz, Frank Golatowski und Dirk Timmermann. „Dynamic remote control through service orchestration of point-of-care and surgical devices based on IEEE 11073 SDC“. In: *2016 IEEE Healthcare Innovation Point-Of-Care Technologies Conference (HI-POCT)*. Cancun: IEEE, Nov. 2016, S. 121–125. DOI: 10.1109/HIC.2016.7797712.

[22] Björn Konieczek, Martin Kasparick, Michael Rethfeldt, Frank Golatowski und Dirk Timmermann. „Towards a TDMA-based real-time extension for the constrained application protocol“. In: *2016 IEEE World Conference on Factory Communication Systems (WFCS)*. IEEE, Mai 2016, S. 1–4. DOI: 10.1109/WFCS.2016.7496529.

[23] Hannes Grunert, Björn Butzin, Martin Kasparick, Andreas Heuer und Dirk Timmermann. „From Cloud to Fog and Sunny Sensors“. In: *Proceedings of the Conference "Lernen, Wissen, Daten, Analysen"(LWDA)*. Bd. Vol-1670. Potsdam, Sep. 2016, S. 83–88.

[24] Björn Andersen, Martin Kasparick, Hannes Ulrich, Stefan Schlichting, Frank Golatowski, Dirk Timmermann und Josef Ingenerf. „Point-of-Care Medical Devices and Systems Interoperability: A Mapping of ICE and FHIR“. In: *In Proceedings of the IEEE Conference on Standards for Communications and Networking (CSCN)*. Okt. 2016. DOI: 10.1109/CSCN.2016.7785165.

[25] Björn Andersen, Martin Kasparick, Frank Golatowski und Josef Ingenerf. „Extending the IEEE 11073-1010X nomenclature for the modelling of surgical devices“. In: *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*. IEEE, Feb. 2016, S. 244–247. DOI: 10.1109/BHI.2016.7455880.

[26] Björn Andersen, Martin Kasparick, Armin Janß, Max Rockstroh, Stefan Franke, Frank Golatowski, Klaus Radermacher, Thomas Neumuth und Josef Ingenerf. „Unterstützung des Chir-

urgen durch vernetzte Medizingeräte & Computer-assistierte Chirurgie“. In: *Medizin 4.0 – Zur Zukunft der Medizin in der digitalisierten Welt*. Berlin: Hanns Martin Schleyer-Stiftung, 2016.

[27] Max Rockstroh, Stefan Franke, Björn Andersen, Martin Kasparick und Thomas Neumuth. „Sichere und dynamische Vernetzung in Operationssaal und Klinik - Erfahrungen und Ergebnisse des Projektes OR.NET“. In: *Tagungsband des 2. Symposium Medizintechnik Leipzig 2016 - Sichere Anwendung von Medizintechnik im OP*. Leipzig, 2016, S. 15–21. ISBN: 978-3-93-7988-32-0.

[28] Martin Kasparick, Stefan Schlichting, Frank Golatowski und Dirk Timmermann. „Medical DPWS: New IEEE 11073 Standard for Safe and Interoperable Medical Device Communication“. In: *2015 IEEE Conference on Standards for Communications and Networking (CSCN)*. Tokyo, Japan: IEEE, Okt. 2015, S. 212–217. DOI: 10.1109/CSCN.2015.7390446.

[29] Martin Kasparick, Björn Konieczek, Sebastian Unger, Frank Golatowski und Dirk Timmermann. „Making Advanced Telemedicine Affordable“. In: *Global Telemedicine and eHealth Updates: Knowledge Resources* 8 (2015), S. 178–182. ISSN: 1998-5509 (hard copy) 1818-9334 (CD-ROM).

[30] Martin Kasparick, Stefan Schlichting, Frank Golatowski und Dirk Timmermann. „New IEEE 11073 Standards for Interoperable, Networked Point-of-Care Medical Devices“. In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Milan, Italy: IEEE, Aug. 2015, S. 1721–1724. DOI: 10.1109/EMBC.2015.7318709.

[31] Martin Kasparick, Frank Golatowski und Dirk Timmermann. „Cyber-physische Systeme im OP-Saal - Ein Machbarkeitsnachweis.“ In: *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI), Big Data - Mastering Complexity: 44th Annual Meeting of the Society for Computer Science, INFORMATICS 2014*. Bd. P-232. Stuttgart: Gesellschaft fur Informatik (GI), Sep. 2014, S. 1203–1214. ISBN: 978-388579626-8.

[32] Frank Krüger, Martin Kasparick, Thomas Mundt und Thomas Kirste. „Where are My Colleagues and Why? Tracking Multiple Persons in Indoor Environments“. In: *2014 International Conference on Intelligent Environments*. IEEE, Juni 2014, S. 190–197. DOI: 10.1109/IE.2014.35.

[33] Martin Kasparick, Alexander Gladisch, Robil Daher, Martin Krohn und Djamshid Tavangarian. „Self-X Evaluation Model for Wireless Mesh Networks“. In: *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*. IEEE, Mai 2011, S. 1–5. DOI: 10.1109/VETECS.2011.5956652.

C Transferorientierte Publikationen des Autors

[1] Thomas Neumuth und Martin Kasparick. „Herstellerübergreifende Vernetzung von Medizingeräten – Die Interoperabilitätsstandardfamilie IEEE 11073 SDC“. In: *Die deutsche Medizintechnik-Industrie – SPECTARIS Jahrbuch 2019 / 2020*. Berlin: SPECTARIS – Deutscher Industriebverband für Optik, Photonik, Analysen und Medizintechnik e.V, Nov. 2019, S. 48–49. ISBN: 978-3-9817205-7-0.

[2] Frank Beger, Armin Janß, Sebastian Bürger, Martin Kasparick und Hans Clusmann. „PriMed – Interdisziplinäre Arbeitsstationen für den OP 4.0“. In: *mt|medizintechnik* 4 (Aug. 2019), S. 25–28. ISSN: 0344-9416.

[3] Frank Beger, Armin Janß, Martin Kasparick, Sebastian Bürger und Hans Clusmann. „Gerätevernetzung im OP 4.0 wird Standard“. In: *meditronic-journal - Fachzeitschrift für Medizin-Technik* 5 (Nov. 2018), S. 6–10. URL: https://www.beam-verlag.de/app/download/31313846/meditronic-Journal+5-2018+I.pdf.

[4] Frank Beger, Armin Janß, Martin Kasparick und Andreas Besting. „Der Operationssaal OP 4.0 – Für mehr Sicherheit und Effizienz durch offene Vernetzung in Operationssälen und Kliniken der Zukunft“. In: *meditronic-journal - Fachzeitschrift für Medizin-Technik* 4 (2017), S. 20–24. URL: https://www.beam-verlag.de/app/download/29391855/meditronic-Journal+4-2017+V.pdf.

[5] Martin Kasparick und Björn Andersen. „Offene Standards im vernetzten Operationssaal“. In: *ELEKTRONIK PRAXIS* Nr. 14 (2016), S. 62–64. ISSN: 0341-5589.

[6] (alphabetized), Björn Andersen, Björn Bergh, Andreas Besting, Johannes Dehm, Jasmin Dell‘Anna-Pudlik, Max Dingler, Sascha Ermeling, Stefan Franke, Armin Janß, Klaus Radermacher, Martin Kasparick, Franziska Kühn, Angela Merzweiler, Raluca Pahontu, Max Rockstroh, Matthias Röser, Malte Schmitz und Andreas Zimolong. *Booklet: OR.NET - Secure and dynamic networking in operating room and hospital (Update 2016)*. 2016.

[7] (alphabetized), Björn Andersen, Johannes Dehm, Christof Gessner, Armin Janß, Martin Kasparick, Peter Knipp, Angela Merzweiler, Heike Moser, Michael Onken und Matthias Röhser. *Weissbuch – Interoperabilität von Geräten und Systemen in OP und Klinik (2. Version)*. Bd. 2. VDE Verband der Elektrotechnik, Elektronik, Informationstechnik e.V., Nov. 2015.

[8] Martin Kasparick. *Gerätemodellierung & Device-Specialisations – Ein Kochbuch*. 2015. URL: http://ws4d.org/projects/ornet/or_net_kochbuch/.

[9] Jasmin Dell’Anna, Irit Beß-Schanze, Julia Benzko, Andreas Besting, Martin Kasparick, Raluca Pahontu, Christian Kücherer, Markus Birkle, Armin Janß, Johannes Dehm, Sascha Ermeling, Tim Hoppe, Andreas Zimolong, Max Rockstroh, Eva-Maria Zeißig, Laura Weingarten, Klaus Radermacher und Björn Bergh. *OR.NET – Sichere dynamische Vernetzung in Operationssaal und Klinik*. 2015.

D Normen und Normungsprojekte mit Beteiligung des Autors

[1] IEEE Standards Association. *ISO/IEC/IEEE 11073-20701-2020 – International Standard for Health informatics–Device interoperability–Part 20701:Point-of-care medical device communication–Service oriented medical device exchange architecture and protocol binding*. 2020. DOI: 10.1109/IEEESTD.2020.9052096.

[2] IEEE Standards Association. *ISO/IEC/IEEE 11073-10207-2019 – Health informatics–Point-of-care medical device communication Part 10207: Domain Information and Service Model for Service-Oriented Point-of-Care Medical Device Communication*. 2019. DOI: 10.1109/IEEESTD.2019.8675788.

[3] IEEE Standards Association. *ISO/IEEE 11073-20702-2018 – Health informatics – Point-of-care medical device communication – Part 20702: Medical devices communication profile for web services*. 2018. DOI: 10.1109/IEEESTD.2018.8472336.

[4] IEEE Standards Association. *P11073-10107 – Standard for Nomenclature for External Control of Medical Devices*. 2019. URL: https://standards.ieee.org/project/11073-10107.html (besucht am 27. 10. 2020).

[5] IEEE Standards Association. *P11073-10700 – Standard for Common Base Requirements for Participants in a Service-Oriented Device Connectivity (SDC) System*. 2019. URL: https://standards.ieee.org/project/11073-10700.html (besucht am 27. 10. 2020).

[6] IEEE Standards Association. *P11073-10701 – Standard for Metric Provisioning by Participants in a Service-Oriented Device Connectivity (SDC) System*. 2019. URL: https://standards.ieee.org/project/11073-10701.html (besucht am 27. 10. 2020).

[7] IEEE Standards Association. *P11073-10703 - Standard for External Control by Participants in a Service-Oriented Device Connectivity (SDC) System*. 2019. URL: https://standards.ieee.org/project/11073-10703.html (besucht am 27. 10. 2020).

[8] IEEE Standards Association. *P11073-10720 - Module Specifications for a Service-Oriented Medical Device Exchange Architecture*. 2019. URL: https://standards.ieee.org/project/11073-10720.html (besucht am 27. 10. 2020).

[9] IEEE Standards Association. *P11073-10721 - Device Specialization - High Frequency (200 kHz to < 5 MHz) Surgical Equipment*. 2019. URL: https://standards.ieee.org/project/11073-10721.html (besucht am 27. 10. 2020).

[10] IEEE Standards Association. *P11073-10722 - Device Specialization - Endoscopic camera*. 2019. URL: https://standards.ieee.org/project/11073-10722.html (besucht am 27. 10. 2020).

[11] IEEE Standards Association. *P11073-10723 - Device Specialization - Endoscopic light source*. 2019. URL: https://standards.ieee.org/project/11073-10723.html (besucht am 27. 10. 2020).

[12] IEEE Standards Association. *P11073-10724 - Device Specialization - Endoscopic insufflator*. 2019. URL: https://standards.ieee.org/project/11073-10724.html (besucht am 27. 10. 2020).

[13] IEEE Standards Association. *P11073-10725 - Device Specialization - Endoscopic pump*. 2019. URL: https://standards.ieee.org/project/11073-10725.html (besucht am 27. 10. 2020).

E Betreute studentische Arbeiten

[1] Farooq Ahmad. „RSSI-based Indoor Localization of Wireless Sensor Nodes“. Master Thesis. Universität Rostock, 2020.

[2] Benjamin Rother. „Automatische Konfiguration eines TSN-Netzwerks für IEEE 11073 SDC-basierte Medizingeräteverbünde“. Master Thesis. Universität Rostock, 2019.

[3] Zankrut Antani. „CoAP Binding for IEEE 11073 SDC: A new Approach for Medical Device Interoperability“. Specialization Module. Universität Rostock, Sep. 2018.

[4] Asad Ali. „IoT Testbed for Research and Teaching – Concept and Development for OPC UA and CoAP“. Specialization Module. Universität Rostock, 2018.

[5] Syed Ali Zafar. „IoT Testbed for Research and Teaching – Concept and Development for DDS and DPWS“. Specialization Module. Universität Rostock, 2018.

[6] Harikrishnan Vasan. „Concept and Prototypical Development of a Generic Medical Device Simulator for IEEE 11073 SDC-compliant Service Providers“. Specialization Module. Universität Rostock, 2018.

[7] Syed Muhammad Arif Hashmi. „Evaluation of a MICS-Band Transceiver and Concept and Prototypical Development of Interoperability Capabilities for Implants“. Specialization Module. Universität Rostock, 2018.

[8] Michael Nast. „IEEE 11073 SDC Anbindung des BITalino Board Kits“. Masterprojekt. Universität Rostock, 2018.

[9] Sabrina Brossmann. „Konzeption und prototypische Umsetzung eines Systems zur Konfiguration von dynamisch belegbaren Bedienelementen im vernetzten OP-Saal“. Bachelor Thesis. Universität Rostock, 2016.

[10] Marcel Stieringer. „BITalino Board Kit: Hardware-Plattform für Biosignal-Messung“. Masterprojekt. Universität Rostock, 2016.

[11] René Michalski. „Entwicklung eines IEEE 11073-Stacks für OP-Geräte“. Bachelor Thesis. Universität Rostock, 2015.

[12] Ayad Mostafa. „Entwicklung und Evaluierung eines SDN-gestützten echtzeitfähigen Gerätenetzwerkes“. Master Thesis. Universität Rostock, 2015.

[13] Stefan Schuster. „Konzeption und Implementierung einer Lösung zur Echtzeitkommunikation auf Basis von Web Services“. Bachelor Thesis. Universität Rostock, 2015.

[14] Stefan Schuster. „Entwicklung eines generischen DPWS-Clients“. Masterprojekt. Universität Rostock, 2015.

F Performanzevaluation

Tabelle F.1: ***SoftICE*-Bibliothek**: Gesamtüberblick über die *Round Trip Time (RTT)*-Messergebnisse für alle untersuchten *Java Virtual Machines (JVMs)* ohne Schemavalidierung (Standardeinstellung). In der rechten Spalte sind die höchsten zehn von 40.000 Messwerten angegeben; das Maximum ist hervorgehoben.
Abkürzungen: Std.-Ab. – Standardabweichung; Min. – Minimum.

Plattform	**JVM**	**Median [ms]**	**Std.-Ab. [ms]**	**Min. [ms]**	**10 höchste Werte [ms]**
Galileo	Oracle8	72,1	16,5	54,3	230,3; 240,9; 261,5; 267,8; 305,9; 403,9; 636,0; 900,1; 976,7; **1061,9**
	OpenJDK8	24,9	25,5	22,3	384,7; 394,3; 410,4; 411,9; 430,5; 442,8; 534,3; 825,4; 1080,5; **1194,4**
RPi 1	Oracle8	42,1	11,8	30,7	121,3; 122,1; 122,5; 123,3; 124,3; 130,8; 135,6; 141,0; 650,8; **859,5**
	OpenJDK8	326,6	10,1	317,2	448,7; 480; 481,7; 484,3; 490,6; 539,2; 571,8; 770,0; 919,7; **954,4**
RPi 2	Oracle8	17,4	4,9	10,8	55,5; 56,1; 58,6; 58,8; 63,8; 66,3; 111; 266,2; 355,1; **385,5**
	OpenJDK8	126,4	16,2	123,9	204,0; 206,7; 207,3; 207,9; 210,0; 211,1; 309,6; 445,0; 524,3; **772,6**
RPi 3	Oracle8	9,8	4,1	7,1	50,7; 51; 51,5; 51,8; 53,1; 53,3; 56,8; 61,5; 260,1; **292,0**
	OpenJDK8	117,6	21,6	77,4	169,4; 170,4; 170,4; 170,5; 170,7; 170,9; 236,9; 284,2; 296,6; **1299,9**

Tabelle F.2: ***OSCLib*-Bibliothek**: Gesamtüberblick über die *Round Trip Time (RTT)*-Messergebnisse mit Schemavalidierung für ein- und ausgehende Nachrichten (Standardeinstellung). In der rechten Spalte sind die höchsten zehn von 40.000 Messwerten angegeben; das Maximum ist hervorgehoben.
Abkürzungen: Std.-Ab. – Standardabweichung; Min. – Minimum.

Plattform	**Median [ms]**	**Std.-Ab. [ms]**	**Min. [ms]**	**10 höchste Werte [ms]**
Galileo	35,4	0,3	34,8	37,7; 37,8; 37,9; 37,9; 38,0; 38,0; 38,1; 38,3; 38,6; **38,7**
RPi 1	23,5	0,3	22,5	25,1; 25,1; 25,2; 25,2; 25,2; 25,2; 25,3; 25,3; 25,3; **25,3**
RPi 2	11,7	0,3	9,7	13,3; 13,3; 13,3; 13,3; 13,4; 13,4; 13,5; 14,1; 14,1; **14,4**
RPi 3	8,8	0,1	8,3	9,1; 9,1; 9,2; 9,2; 9,6; 9,6; 10,0; 12; 16,3; **21,9**

Tabelle F.3: ***openSDC*-Bibliothek**: Gesamtüberblick über die *Round Trip Time (RTT)*-Messergebnisse für alle untersuchten *Java Virtual Machines (JVMs)* mit Schemavalidierung für eingehende Nachrichten (Standardeinstellung). In der rechten Spalte sind die höchsten zehn von 40.000 Messwerten angegeben; das Maximum ist hervorgehoben. Abkürzungen: Std.-Ab. – Standardabweichung; Min. – Minimum.

Plattform	**JVM**	**Median [ms]**	**Std.-Ab. [ms]**	**Min. [ms]**	**10 höchste Werte [ms]**
Galileo	Oracle7	38,5	13,6	36,6	140,4; 140,7; 140,8; 155,5; 172,2; 173,5; 184,1; 206,3; 1429,4; **1549,5**
	Oracle8	59,0	15,0	48,8	177,7; 178,5; 181,7; 185,4; 213,1; 216,5; 262,7; 300,3; 997,5; **1314,6**
	OpenJDK7	39,9	15,0	38,5	136,9; 138,2; 139,3; 147,0; 150,8; 155; 215,7; 256,1; 1565,5; **1912,8**
	OpenJDK8	30,3	20,4	26,7	254,2; 260,7; 261,6; 264,9; 265,1; 267,1; 272,2; 458,5; 1252,3; **1784,9**
RPi 1	Oracle7	24,7	10,5	23,5	92,7; 95,7; 97,4; 101,6; 107,0; 118,6; 125,6; 273,6; 1095,6; **1228,9**
	Oracle8	33,7	11,5	28,1	108,7; 111,0; 114,0; 116,3; 134,6; 140,5; 166,6; 749,1; 872,0; **976,5**
	OpenJDK7	324,5	8,6	320,5	437,2; 451,1; 457,7; 465,7; 472,9; 489,3; 497,6; 569,9; 1340,9; **1414,9**
	OpenJDK8	325,0	7,7	320,6	439,2; 449,3; 454,1; 463,9; 464,1; 475,9; 485,6; 500,7; 1209,6; **1215,9**
RPi 2	Oracle7	12,6	5,9	9,3	50,1; 50,4; 50,7; 52,6; 57,8; 72,9; 74,0; 550,3; 580,6; **630,4**
	Oracle8	14,7	5,2	10,6	49,1; 49,5; 49,6; 52,5; 53,8; 60,2; 62; 404,7; 408,1; **613,6**
	OpenJDK7	125,7	9,8	124,0	188,1; 188,2; 188,6; 188,9; 188,9; 199,1; 524,3; 576; 696,1; **828,1**
	OpenJDK8	123,6	9,6	121,8	185,7; 186,3; 186,5; 187,1; 187,4; 187,5; 187,6; 193,9; 390,6; **497,1**
RPi 3	Oracle7	8,8	4,4	6,0	43,6; 44,1; 44,4; 45,6; 49,2; 54,8; 61,9; 72,6; 373; **397,6**
	Oracle8	9,1	5,0	6,1	36,4; 36,5; 36,9; 39,3; 41,8; 45,2; 57,8; 262,4; 295,9; **731,8**
	OpenJDK7	78,6	13,8	77,2	158,9; 162,9; 164,6; 166,6; 178,3; 223,5; 321,3; 417,6; 618,8; **1372,6**
	OpenJDK8	77,2	12,2	75,9	154,2; 154,8; 155,8; 156,4; 158,5; 159,2; 161,3; 287,9; 307,6; **691,3**

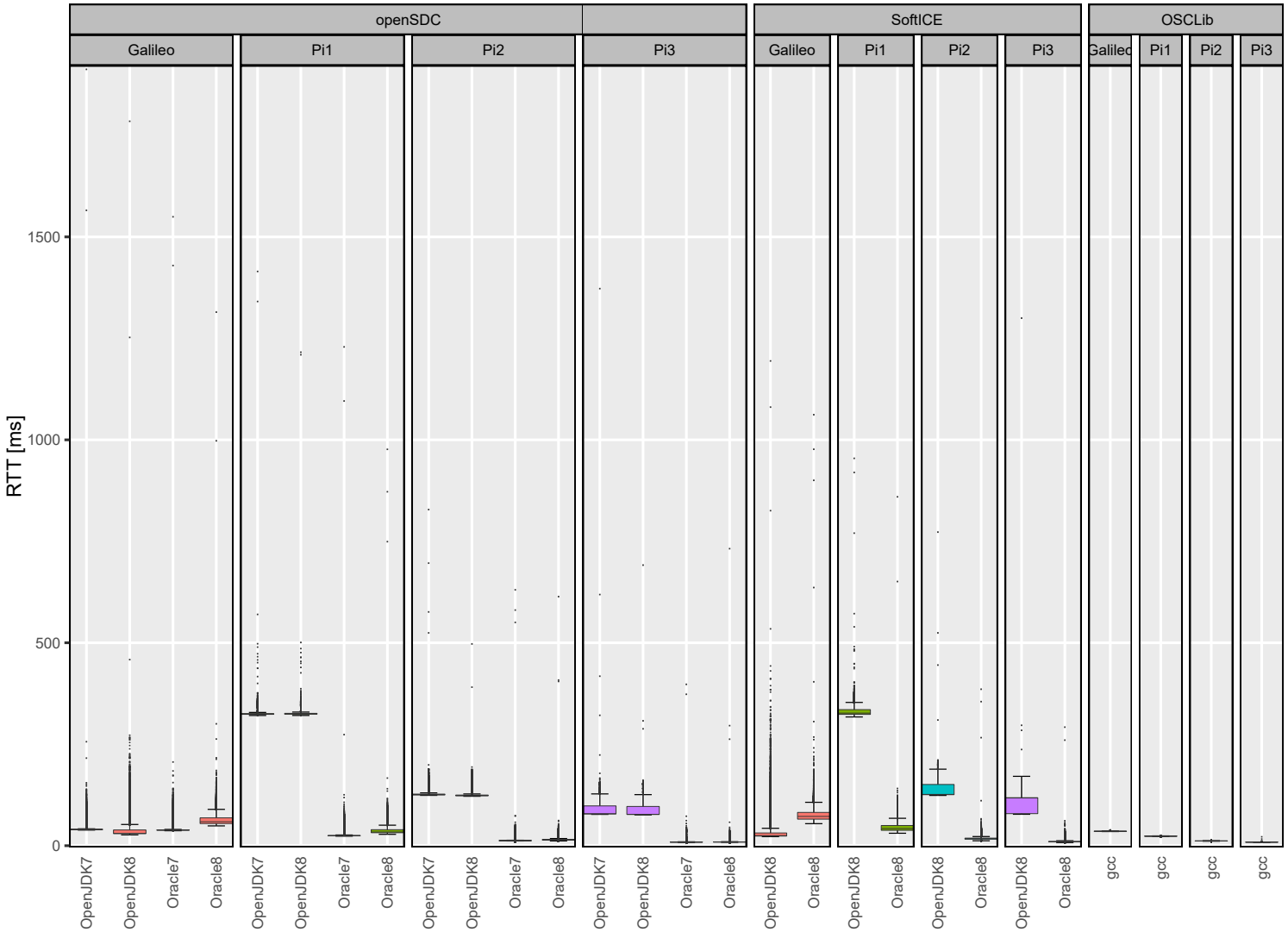

Abbildung F.1: Vollständige Version von Abbildung 5.3

G Dynamische Fernsteuerung basierend auf Service-Orchestrierung

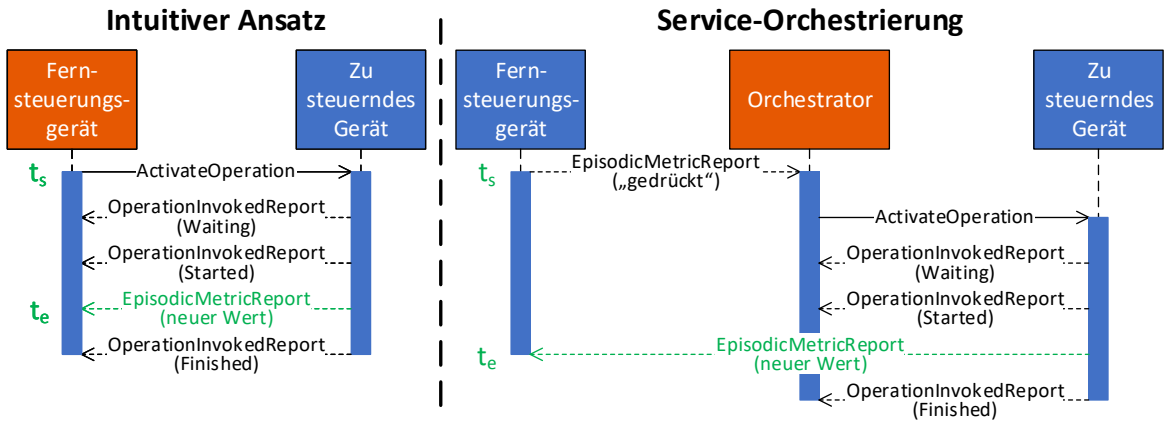

Abbildung G.1: Vergleich der Konzepte bezüglich der ausgetauschten Nachrichten bei einer Realisierung auf Basis der *SDCLib/C*-Bibliothek. Abkürzungen: t_s – Startzeitstempel; t_e – Endzeitstempel.

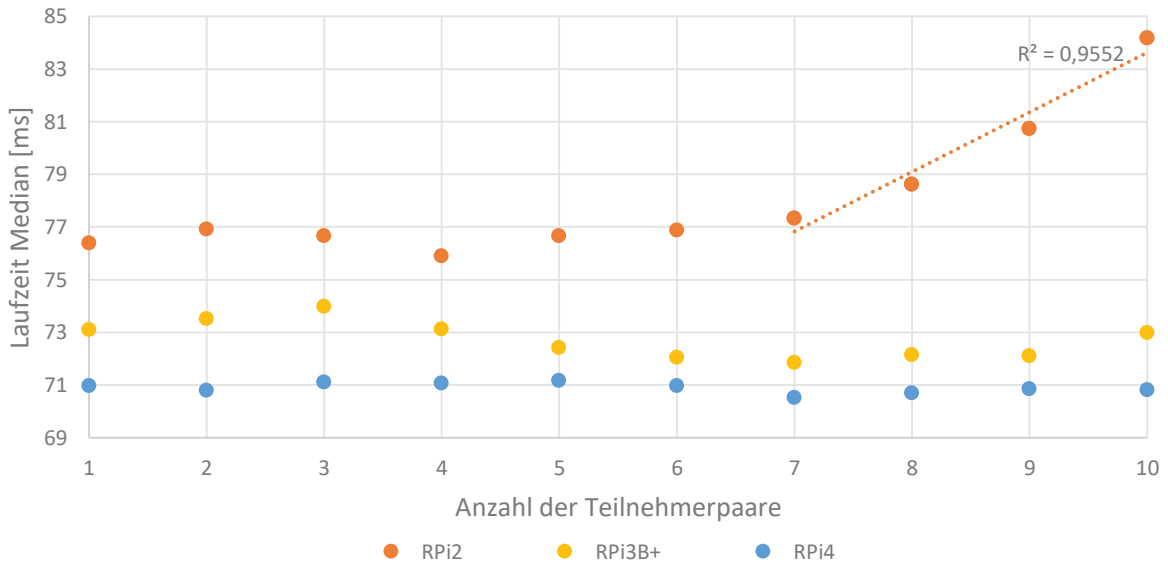

Abbildung G.2: Skalierbarkeitsuntersuchung der *Service-Orchestrierung*: Laufzeiten mit dem Fokus auf die Hardwareplattformen *Raspberry Pi 2, 3B+, und 4* und steigender Anzahl an Fernsteuerungspaaren (immer *Raspberry Pi 1*).