Association for Information Systems
AIS Electronic Library (AISeL)

Wirtschaftsinformatik Proceedings 2011 | Wirtschaftsinformatik

2011

SODA@Med – Ein Framework zur serviceorientierten Integration medizinischer Geräte in Krankenhausinformationssysteme

Christian Mauro
Technische Universitat Munchen, mauro@in.tum.de

Jan Marco Leimeister
Universitat Kassel, leimeister@acm.org

Helmut Krcmar
Technische Universitat Munchen, krcmar@in.tum.de

SODA@Med – Ein Framework zur serviceorientierten Integration medizinischer Geräte in Krankenhausinformationssysteme

Christian Mauro
Lehrstuhl für Wirtschaftsinformatik
Technische Universität München
Boltzmannstr. 3
85748 Garching, Germany
mauro@in.tum.de

Jan Marco Leimeister
Fachgebiet Wirtschaftsinformatik
Universität Kassel
Nora-Platiel-Str. 4
34127 Kassel, Germany
leimeister@uni-kassel.de

Helmut Krcmar
Lehrstuhl für Wirtschaftsinformatik
Technische Universität München
Boltzmannstr. 3
85748 Garching, Germany
krcmar@in.tum.de

ABSTRACT
In diesem Beitrag wird ein serviceorientiertes Framework (SODA@Med) vorgestellt, das eine einfachere und bessere Integration medizinischer Geräte in Krankenhausinformationssysteme ermöglichen soll, um informationslogistische Defizite im Krankenhaus zu reduzieren. Das Framework wurde auf Basis von aus der Literatur und Fallstudien erhobenen Anforderungen entwickelt. Es verwendet einen Pattern-basierten Ansatz, der es erlaubt, existierende SOA Design Patterns und Best Practices zu integrieren. SODA@Med besteht aus einem konzeptuellen Design sowie einer zugehörigen prototypischen Implementierung. Das Design beinhaltet a) ein Architekturmodell, das die grundlegende Integrationsstrategie festlegt, b) ein Schichtenmodell, das die Kommunikationswege definiert sowie c) ein Modulmodell, das die logischen Einheiten des Frameworks beschreibt. Die Umsetzbarkeit des Designs sowie die Funktionsfähigkeit des Frameworks konnten durch dessen prototypische Implementierung sowie Labortests belegt werden. Zudem erfolgte eine positive Reflexion der Anforderungen anhand einer kriterienbasierten Evaluation.

Keywords
Service Oriented Device Architecture, SODA, Serviceorientierte Architekturen, SOA, Framework, medizinische Geräte, Integration, Krankenhausinformationssystem.

10th International Conference on Wirtschaftsinformatik,
16th - 18th February 2011, Zurich, Switzerland

1. EINLEITUNG
Die IT-Landschaft in Krankenhäusern ist geprägt durch informationslogistische Defizite, bedingt durch die fehlende Integration aller an den administrativen und medizinischen Prozessen beteiligten Informationssystemen [25]. Dem entgegen steht die Vision des „Seamless Healthcare“, bei der vertikal und horizontal durchgängige Prozesse, Daten und Informationstechnologien es ermöglichen, dass jeder Akteur die richtige Information, zum richtigen Zeitpunkt, am richtigen Ort in der richtigen Qualität und Quantität erhält [22]. Eine wesentliche Hürde zur Realisierung dieser Vision ist die Integration medizinischer Geräte, die aber Teil sehr vieler klinischer Behandlungsprozesse sind und deren Daten meist nicht in die zentralen Krankenhausinformationssysteme übergeben werden. Diese Integration wird u.a. durch eine Vielzahl verschiedener proprietärer Schnittstellen der Medizingeräte erschwert [12]. Im Gegensatz zu anderen Aspekten (wie bspw. Usability oder Qualitätssicherung) ist Interoperabilität keine vom Gesetzgeber geforderte Eigenschaft medizinischer Geräte [12].

Ein aktueller Trend im Bereich der Softwaresysteme im Gesundheitswesen ist der Einsatz serviceorientierter Architekturen (SOA) als Ansatz zur Begegnung der gewachsenen heterogenen IT-Strukturen (siehe bspw. [6, 19, 23]). Ein noch junges Forschungsgebiet versucht diesen Ansatz für die Integration medizinischer Geräte zu adaptieren. Das grundlegende Konzept, Geräte als Services zu kapseln, ist in der Literatur auch unter der Bezeichnung SODA (Service Oriented Device Architecture) bekannt [7].

Wie anhand einer vorhergehenden Literaturanalyse gezeigt werden konnte, erzielten erste prototypische Umsetzungen vielversprechende Ergebnisse hinsichtlich der generellen Eignung sowie der Vorteilhaftigkeit von SODA [17]. Es konnte jedoch auch belegt werden, dass viele Fragestellungen bisher nur ungenügend adressiert wurden. Insbesondere mangelt es an einer nachvollziehbaren Erhebung von Anforderungen an ein serviceorientiertes Integrationskonzept für medizinische Geräte. Auch konnte kein verallgemeinertes Integrationskonzept gefunden werden, das die Charakteristika medizinischer Geräte berücksichtigt. Die wenigen bisher veröffentlichten Arbeiten beschränken sich auf prototypische Umsetzungen von sehr spezifischen Anwendungsfällen ohne eine Verallgemeinerung der Ergebnisse zu diskutieren [17].

Ziel des Beitrags ist die Entwicklung eines serviceorientierten Frameworks (SODA@Med), mit dessen Hilfe die Integration medizinischer Geräte einfacher und besser möglich sein soll, als mit bisherigen Ansätzen. Der grundlegende Forschungsprozess sieht in Anlehnung an Becker [2] Phasen der Analyse (State of the Art Analyse, Identifizierung der Forschungslücken), des Entwurfs (Anforderungserhebung, Entwurf des Framework Designs, prototypische Implementierung), der Evaluation sowie der Diffusion (in Form von Publikationen und Vorträgen) vor (Abbildung 1). Der Aufbau des Beitrags lehnt sich an diesen Forschungsprozess an. Die State of the Art Analyse sowie die Anforderungserhebung wurden bereits in gesonderten Beiträgen publiziert [13-18]. Die Abschnitte 1 und 2 fassen hierzu die wesentlichen Ergebnisse zusammen. Der Fokus dieses Beitrags liegt auf der Entwicklung des Framework-Designs (Abschnitt 3) sowie der zugehörigen prototypischen Implementierung (Abschnitt 4). Labortests sowie eine kriterienbasierte Evaluation in Abschnitt 5 weisen die Funktionsfähigkeit des Frameworks sowie die Erfüllung der Anforderungen nach. Der Beitrag schließt mit einer Zusammenfassung sowie einem Ausblick (Abschnitt 6).

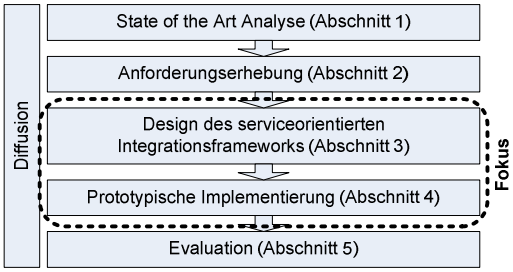

Abbildung 1: Forschungsprozess und Fokus des Beitrags

2. ANFORDERUNGSERHEBUNG

Das Vorgehen zur Erhebung der Anforderungen an das SODA@Med Framework basiert auf der Beobachtung, dass medizinische Geräte, aufgrund der folgenden gemeinsamen Charakteristika, hinsichtlich ihrer Integration als Legacy Systeme betrachtet werden können (vgl. [3, 4]):

- Beschränkte Ressourcen verhindern die Ausführung moderner Kommunikationsframeworks
- Proprietäre Schnittstellen erschweren die Integration mit anderen Systemen
- Die schwierige (und oft unmögliche) Erweiterbarkeit / Änderbarkeit erlaubt keine Modifizierung der unzureichenden Schnittstellen

Für die Integration von Legacy Systemen existieren eine Reihe bewährter Produkte und Konzepte, die sich ebenfalls serviceorientierter Methoden bedienen (vgl. bspw. [24]). Aufgrund der gemeinsamen Charakteristika erscheint es daher schlüssig, diese Konzepte auch auf das Anwendungsfeld der Integration medizinischer Geräte anzuwenden. Jedoch ist zu vermuten, dass medizinische Geräte zusätzliche Eigenschaften aufweisen, die eine Integration erschweren. Diese könnten sich bspw. aus der Mobilität der Geräte ergeben. Aus diesem Grund sieht das Vorgehen zur Anforderungserhebung die Analyse der für medizinische Geräte spezifischen Integrationshürden vor. Aus diesen lassen sich Anforderungen für das SODA@Med Framework ableiten. Damit kann das Framework einerseits auf etablierten Konzepten basieren und muss andererseits die spezifischen Herausforderungen im Kontext medizinischer Geräte berücksichtigen. Die Erhebung der Anforderungen erfolgte zum einen argumentativ und basierend auf Literatur, zum anderen wurden drei Fallstudien an zwei Universitätsklinika durchgeführt (Abbildung 2).

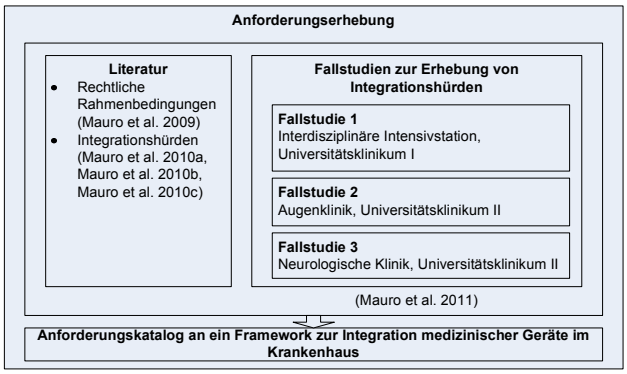

Abbildung 2: Vorgehen zur Anforderungserhebung

Aus den Analysen lässt sich die in Tabelle 1 dargestellte konsolidierte Liste mit neun Anforderungen ableiten, die das Framework erfüllen muss.

Tabelle 1: Anforderungen an ein Framework zur Integration medizinischer Geräte (in Anlehnung an [13-16, 18])

Nr.	Beschreibung der Anforderung
1	Medizinische Geräte müssen mit verschiedensten Systemen gleichzeitig integriert werden können.
2	Medizinische Daten müssen korrekt dem jeweiligen Patienten zugeordnet werden.
3	Medizinische Geräte müssen dynamisch erkannt sowie mit deren unterschiedlichen (meist proprietären) Schnittstellen kommuniziert werden können.
4	Unterschiedlichste Hardware-Schnittstellen müssen unterstützt werden, bspw. serielle Schnittstellen oder Netzwerkschnittstellen.
5	Ein Einsatz des Frameworks darf keine nicht hinnehmbaren Risiken im Sinne der IEC 80001 Norm erzeugen.
6	Die von den Herstellern angegebenen Zweckbestimmungen der Geräteschnittstellen müssen berücksichtigt werden.
7	Streaming muss unterstützt werden.
8	Eine ereignisbasierte Übertragung von Daten muss unterstützt werden.
9	Das Übertragen großer Datenmengen (mehrere hundert Megabyte) muss unterstützt werden.

Anforderung 1 ergab sich aus der Beobachtung, dass die von den medizinischen Geräten erzeugten Daten für unterschiedlichste Systeme genutzt werden können, bspw. für Systeme zur medizinischen Dokumentation, Fakturierung oder mobilen Visite. Anforderung 2 ist wesentlich, um falsche Behandlungsentscheidungen aufgrund einer fehlerhaften Datenzuordnung zu vermeiden. Dieser Aspekt war insbesondere auch in der Literatur zu finden (bspw. [27]). Die Anforderungen 3 und 4 ergaben sich zum einen aus einer Analyse der Hard- und Softwareschnittstellen aller Geräte im Rahmen der durchgeführten

Fallstudien. Hierbei konnte festgestellt werden, dass eine überwiegende Anzahl der Schnittstellen proprietär und teils sehr komplex ist. Zum anderen konnte beobachtet werden, dass medizinische Geräte oft dynamisch ausgetauscht werden, bspw. im Falle einer Fehlfunktion oder zu Wartungszwecken. Auch können Geräte jederzeit ein- oder ausgeschaltet werden. Ein Integrationskonzept muss mit einem solchen dynamischen Verhalten umgehen können. Die Anforderungen 5 und 6 ergaben sich direkt aus den gesetzlichen Rahmenbedingungen (vgl. [16]). Die Anforderungen 7 bis 9 wurden im Rahmen der Fallstudien aus einer Analyse der von den Geräten übertragenen Daten abgeleitet.

3. KONZEPTUELLES FRAMEWORK DESIGN

Im Bereich der serviceorientierten Konzepte existieren – auch im Kontext der Integration von Legacy Systemen – umfangreiche Erfahrungen in Wissenschaft und Praxis, welche sich in Best Practices bzw. SOA Design Patterns manifestiert haben. Aus diesem Grund verfolgen wir zur Erstellung des Lösungsdesigns einen Pattern-orientierten Ansatz, der es (ganz im Sinne des Design Science [11]) erlaubt, die existierende Wissensbasis in das Framework zu integrieren. Dieser Ansatz besteht aus vier Schritten (Abbildung 3). Im ersten Schritt wurden die zuvor erhobenen Anforderungen konkretisiert. Anschließend wurden im zweiten Schritt existierende SOA Design Patterns identifiziert, die zur Erfüllung der erhobenen Anforderungen beitragen können [14]. Für Anforderungen, die nicht mit existierenden Patterns umgesetzt werden konnten, wurden im dritten Schritt neue Patterns entwickelt [14, 15, 18].

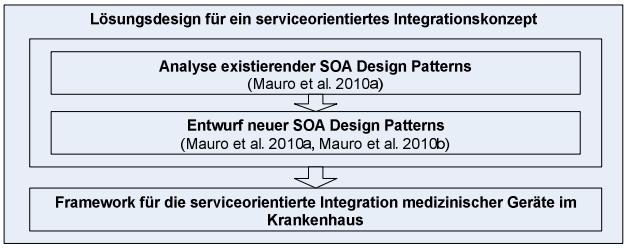

Abbildung 3: Vorgehen zur Entwicklung des Lösungsdesigns

Das konzeptuelle Design des SODA@Med Frameworks basiert auf dem SODA Basiskonzept (Abschnitt 3.1) und beinhaltet ein Architekturmodell (Abschnitt 3.2), ein Schichtenmodell (Abschnitt 3.3) sowie ein Modulmodell (Abschnitt 3.4), welche aufeinander aufbauen (Abbildung 4). Die zuvor identifizierten Patterns fließen in das Design der jeweiligen Modelle ein und werden im Zuge der Implementierung (Abschnitt 4) technisch umgesetzt. Im Rahmen der Beschreibung der einzelnen Modelle wird gezeigt, an welchen konkreten Stellen die Patterns Anwendung finden.

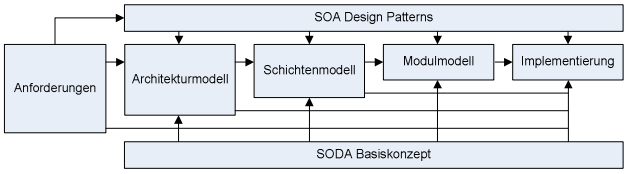

Abbildung 4: Einflussfaktoren bei der Modellentwicklung

3.1 SODA Basiskonzept

Das Basiskonzept für eine serviceorientierte Integration von Legacy Systemen besteht – übertragen auf Geräte (SODA) – aus einer Menge von Servicekonsumenten, einem Geräteservice, einem Gerät sowie einer Service Registry mittels derer die Services publiziert und gefunden werden können (Abbildung 5) [7]. Als Geräteservice bezeichnen wir in diesem Kontext einen Service, der die Funktionalitäten eines Geräts (genauer gesagt seiner Schnittstelle) als Service kapselt. Ein Geräteservice beinhaltet einen Servicevertrag, eine Serviceimplementierung, die den Servicevertrag umsetzt, sowie einen Adapter, der die Kommunikation mit der proprietären Geräteschnittstelle realisiert [18]. Ein Servicevertrag beinhaltet im Wesentlichen eine Beschreibung der Serviceschnittstelle, kann jedoch zusätzliche Elemente, wie bspw. Service Level Agreements, umfassen [9].

Prinzipiell ist die Kapselung eines medizinischen Geräts als Service auf Basis des eben beschriebenen Konzepts realisierbar. Dies konnte durch technische Labortests nachgewiesen werden. Hierzu wurde ein Geräteservice sowie ein Adapter für einen spezifischen Patientenmonitor implementiert, zur Ausführung gebracht und an eine Service Registry publiziert. Anschließend konnte der Geräteservice gefunden und über diesen auf die vom Patientenmonitor angebotenen Daten zugegriffen werden.

Bei genauerer Betrachtung zeigt sich jedoch, dass einige der Anforderungen nicht erfüllt werden können. Bspw. ist das dynamische Erkennen und Nutzen verschiedener Geräte mit dem Basiskonzept nicht abbildbar. Ziel ist daher die Adaption des allgemeinen domänenunabhängigen SODA Basiskonzepts für die Integration medizinischer Geräte (SODA@Med).

3.2 Architekturmodell

Dem SODA@Med Framework liegen Überlegungen hinsichtlich der Gesamtarchitektur zugrunde, die bei einem Einsatz im Krankenhaus angestrebt wird. Eine wesentliche Entscheidung ist hinsichtlich der Publizierung der Geräteservices zu treffen. Hierfür stehen zwei verschiedene SOA Design Patterns zur Auswahl [15]. Das Auto-Publishing Pattern sieht vor, dass ein Service sich beim Aktivieren automatisch selbst bei der Service Registry publiziert bzw. den entsprechenden Eintrag beim Deaktivieren wieder entfernt. Das Decentralized Service Discovery Pattern kommt hingegen ohne Service Registry aus. Das Publizieren und Suchen von Services erfolgt durch das Versenden von Multicast-Nachrichten. Eine Einschränkung hierbei ist, dass Services nur im lokalen Netzwerk auffindbar sind, da Multicast-Nachrichten üblicherweise nicht geroutet werden [26][1]. Wir argumentieren, dass der Einsatz des zweiten Lösungsansatzes (des Decentralized Service Discovery Patterns) im Krankenhausumfeld aus folgenden Gründen zweckmäßiger ist:

1. Ein Single Point of Failure in Form einer zentralen Service Registry wird vermieden.
2. Das direkte Aufrufen von Geräteservices von außerhalb des lokalen Netzwerkes ist nicht zweckmäßig und

[1] Konkrete Technologien wie WS-Discovery ermöglichen eine Netzwerk-übergreifende Suche durch Verwendung von Discovery Proxies [2]. Diese Proxies entsprechen ihrem Wesen nach aber Service Registries und werden daher im Kontext des Decentralized Service Discovery Patterns nicht berücksichtigt.

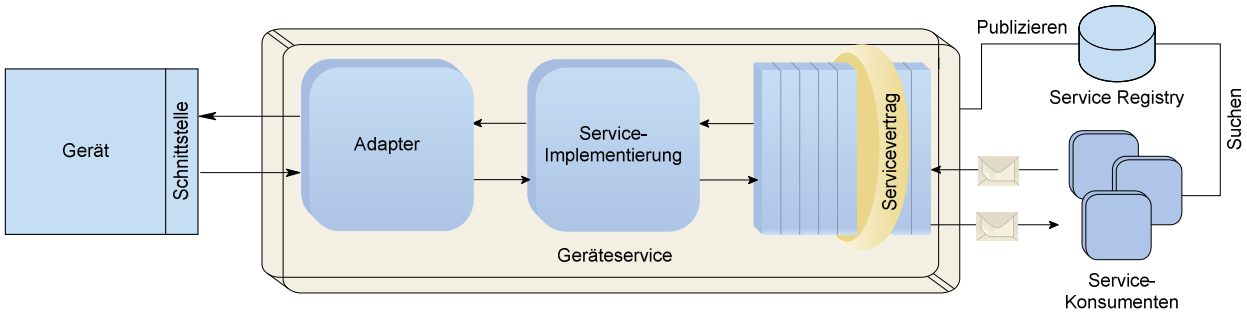

Abbildung 5: SODA Basiskonzept für die serviceorientierte Geräteintegration (in Anlehnung an [18])

zudem rechtlich bedenklich. Daher ist das Publizieren der Services über die Netzwerkgrenze hinaus nicht notwendig.

Hinsichtlich des ersten Grundes stützen wir uns auf Überlegungen von Pöhlsen, der in seiner Arbeit ein Discovery-Konzept für SODA vorstellt [21]. Er argumentiert, dass dezentrale Mechanismen auch bei einem partiellen Netzwerkausfall funktionsfähig bleiben. Zudem bestehe nicht das Risiko des Ausfalls einer zentralen Komponente.

Hinsichtlich des zweiten Grundes weichen wir von den Überlegungen von Pöhlsen ab, dessen Konzept eine netzwerkübergreifende Suche nach Geräteservices unter Verwendung von Discovery Proxies vorsieht. Zwar stimmen wir zu, dass auch Servicekonsumenten außerhalb des lokalen Netzwerks Interesse an den von den Geräten produzierten Daten haben können, jedoch sieht das Architekturmodell in diesem Fall keinen direkten Zugriff auf Geräteservices vor. Wir begründen dies mit zwei Argumenten. Zum einen erscheint der direkte Zugriff auf einzelne Geräteservices meist nicht zweckmäßig. Als Beispiel sei ein Servicekonsument genannt, der regelmäßig die Vitaldaten eines Patienten abrufen möchte. Hierzu müssten die dafür nötigen Geräteservices nicht nur einmalig gefunden, sondern deren Status ständig überwacht werden, da Geräte jederzeit an- oder ausgeschaltet werden können und somit neue relevante Geräteservices hinzukommen bzw. nicht mehr verfügbare Geräteservices wegfallen könnten. Die Implementierung auf Seiten der Servicekonsumenten wäre entsprechend aufwändig und redundant. Das zweite Argument erscheint noch gewichtiger. Gerätenetzwerke im Krankenhaus sind aus rechtlichen Gründen üblicherweise von der restlichen IT logisch getrennt [27]. Der direkte Zugriff auf einen Geräteservice von außerhalb des lokalen Netzwerkes würde dieses Prinzip verletzen.

Aus diesen Überlegungen ergibt sich das in Abbildung 6 dargestellte Architekturmodell. Innerhalb eines lokalen Gerätenetzwerkes mit einer dynamischen Anzahl an Geräteservices erfolgt das Publizieren und Suchen von Services unter Anwendung des Decentralized Service Discovery Patterns. Darüber hinaus findet das Device Concentrator Pattern Anwendung. Dieses Pattern sieht das Hinzufügen einer Abstraktionsschicht zwischen mehreren Geräteservices und Servicekonsumenten vor [14]. Hierzu werden am Rand des Gerätenetzwerks anwendungsspezifische Services platziert, bspw. zum Abruf der Vitaldaten eines Patienten. In Anlehnung an das angewandte Pattern nennen wir solche Services auch Gerätekonzentratoren. Diese haben zum einen Zugriff auf die lokalen Geräteservices, sind aber zum anderen von außen aufrufbar und bieten den Servicekonsumenten eine einfache Schnittstelle zu den von den Geräten erzeugten Daten. Als positiver Nebeneffekt wird die Verwaltung der Zugriffsrechte vereinfacht, da diese nicht auf der Ebene einzelner Geräteservices sondern auf der Ebene fachlicher Services (bspw. einem Vitaldatenservice) erfolgen kann. Dies erlaubt den Einsatz bewährter SOA Sicherheitskonzepte (bspw. die Sicherheitspatterns in [8]). Um unerwünschte Nebeneffekte durch unkontrollierte parallele Zugriffe auf die Geräte zu vermeiden, wird zudem das Trusted Subsystem Pattern [8] umgesetzt. Es empfiehlt, mittels geeigneter technischer Maßnahmen, einen direkten Zugriff auf das Gerät nur dem Geräteservice zu gestatten. Im einfachsten Fall kann dies bspw. ein Passwortschutz sein, wobei nur der Geräteservice das Passwort kennt.

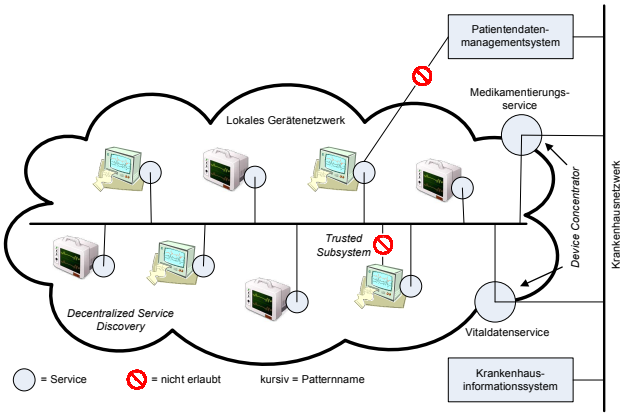

Abbildung 6: Architekturmodell

3.3 Schichtenmodell

Aus den bisherigen Überlegungen lassen sich direkt die notwendigen Schichten des SODA@Med Frameworks ableiten (Abbildung 7). Ausgehend von den Geräten ergibt sich aus Anforderung 4 die Notwendigkeit einer Schicht, die die verschiedenen Hardware-Schnittstellen abstrahiert (Connector Layer). Die Adapter-Schicht wurde aus dem SODA Basiskonzept übernommen, um eine Kommunikation mit den proprietären Geräteschnittstellen zu ermöglichen. Zur Umsetzung von Anforderung 3 findet das Dynamical Adapter Pattern Anwendung. Dieses erfordert eine Schicht, die zur Laufzeit verschiedene Geräte erkennen sowie passende Adapter auswählen und anwenden kann [14]. Die Geräteservices-Schicht beinhaltet analog zum SODA Basiskonzept die Services für die

verschiedenen Gerätetypen[2]. Zur Umsetzung von Anforderung 2 wird ein neues Pattern vorgeschlagen, das Data Enrichment Pattern. Dieses Pattern sieht eine Schicht zwischen den Geräteservices und den Servicekonsumenten vor, um vom Service ausgehende Daten anhand bestimmter Kriterien anzureichern. Im konkreten Fall medizinischer Geräte wird das Pattern genutzt, um bei Bedarf eine fehlende Patienten-Id in einem vom Geräteservice zurückgegebenen Datensatz zu ergänzen. Gerätekonzentratoren greifen auf Geräteservices zu und sind damit eine spezielle Art von Servicekonsumenten und liegen daher außerhalb des Frameworks. Da diese jedoch ein wesentlicher Bestandteil des Architekturmodells sind, stellt das Framework Funktionalitäten bereit, um die Entwicklung von Gerätekonzentratoren zu erleichtern (s. Abschnitt 4.3).

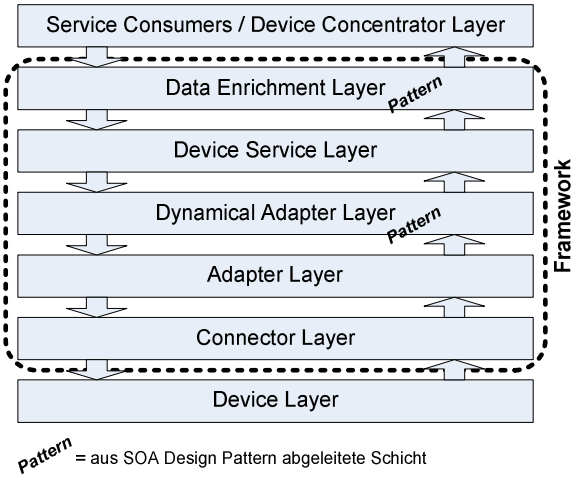

Abbildung 7: Schichtenmodell des SODA@Med Frameworks

3.4 Modulmodell

Ausgehend von den bisherigen Ergebnissen wurde das SODA@Med Framework in ein Kernmodul sowie vier erweiterbare Module unterteilt (Abbildung 8). Der Core Container stellt grundlegende Funktionalitäten, Datenmodelle sowie Möglichkeiten zur Konfiguration bereit. Eine weitere wichtige Komponente (Deployment) ermöglicht das dynamische Installieren und Deinstallieren von Geräteservices. Dies ist notwendig, da die Existenzberechtigung für Geräteservices direkt von der Verfügbarkeit der jeweiligen Geräte abhängt. Zur Umsetzung wird ein neues Pattern vorgeschlagen, das Auto-Deploy Pattern. Dieses Pattern sieht vor, dass eine außerhalb des Services liegende Komponente (bei SODA@Med der dynamische Adapter) die Verfügbarkeit der zur Ausführung des Services benötigten Ressourcen (bei SODA@Med das medizinische Gerät) überwacht und den Service davon abhängig entsprechend automatisiert installiert bzw. deinstalliert.

Die Konnektoren-Schicht wird in Form von Plugins für verschiedene Hardware-Schnittstellen realisiert. Plugins für eine serielle, parallele und TCP/IP-Kommunikation sind im Framework bereits enthalten, weitere können bei Bedarf hinzugefügt werden.

Die Adapterschicht wird umgesetzt, indem für jedes Gerätemodell, das vom Framework erkannt werden soll, ein entsprechender Adapter bereitgestellt wird. Im Rahmen technischer Labortests (vgl. Abschnitt 5.1) wurden Adapter für B. Braun Space® Infusions- und Spritzenpumpen sowie Dräger Infinity® Delta Patientenmonitore umgesetzt. Ein Adapter für Dräger Beatmungsgeräte, die das MEDIBUS-Protokoll verwenden, ist in Planung.

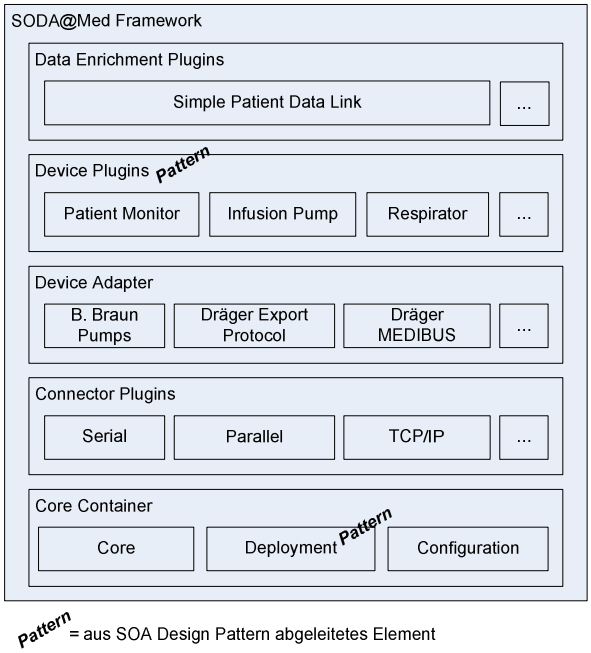

Abbildung 8: Module des SODA@Med Frameworks

Die Schichten der dynamischen Adapter und der Geräteservices werden zusammengefasst. Für jeden Gerätetyp wird ein Plugin benötigt, das einen Geräteservice sowie einen dynamischen Adapter enthält. Zudem definiert das Plugin, wie ein Adapter für diesen Gerätetyp gestaltet sein muss, um mit dem dynamischen Adapter verwendet werden zu können. Wichtig ist die Tatsache, dass es genau ein Geräte-Plugin pro *Gerätetyp* gibt und genau einen Adapter pro *Gerätemodell*, wobei mehrere Gerätemodelle zu einem Gerätetyp gehören können. Dies bedeutet im Umkehrschluss, dass es genau einen Servicevertrag pro Gerätetyp gibt, der unabhängig vom eingesetzten Gerätemodell immer gleich ist. Dies entspricht einer Umsetzung des Legacy Wrapper Patterns [8]. Dieses Pattern empfiehlt, proprietäre Aspekte des Legacy Systems (wie proprietäre Datenmodelle oder systemspezifische Fehlermeldungen) nicht in den Servicevertrag zu integrieren. Stattdessen sollte der Servicevertrag unternehmensweiten / bereichsweiten Standards (hinsichtlich Datenmodell, etc.) entsprechen [8]. Im SODA@Med Framework wird dieses Pattern umgesetzt, indem Geräteservices herstellerunabhängig definiert werden. Sie werden entsprechend nicht für ein Gerätemodell sondern für einen Gerätetyp konzipiert. Auf diese Weise werden Servicekonsumenten von den Geräten entkoppelt und kommen mit den proprietären Geräteschnittstellen nicht in Berührung. Für die Labortests wurden Plugins für Patientenmonitore und

[2] Wir unterscheiden die Begriffe Gerätetyp und Gerätemodell. Ein Gerätetyp ist bspw. der Patientenmonitor. Ein Gerätemodell ist eine konkrete Ausprägung eines Gerätetyps, also bspw. ein Patientenmonitor eines bestimmten Herstellers, aus einer bestimmten Modellreihe in einer bestimmten Version.

Infusions-/Spritzenpumpen entwickelt. Ein Plugin für Beatmungsgeräte ist in Planung.

Auch für die Umsetzung der Schicht zur Datenanreicherung können Plugins bereitgestellt werden. Für die Labortests wurde ein Plugin (Simple Patient Data Link) entwickelt, welches prüft, ob der vom Geräteservice zurückgegebene Datensatz eine Patienten-Id enthält. Falls nicht, versucht es, den Datensatz einem Patienten zuzuordnen und die Patienten-Id zu setzen. Ist dies nicht erfolgreich, wird vom Geräteservice statt des Datensatzes eine entsprechende Fehlermeldung zurückgegeben.

4. PROTOTYPISCHE IMPLEMENTIERUNG

Nachdem in Abschnitt 3 das konzeptuelle Design des Frameworks gezeigt wurde, wird in Abschnitt 4 die technische Umsetzung erläutert. Hierzu wird die Auswahl der verwendeten Technologien, die Klassenstruktur sowie die konkrete Realisierung der einzelnen Module und Patterns vorgestellt.

4.1 Technologieauswahl

Zur technischen Umsetzung des theoretischen Konzepts sind geeignete Technologien und Standards für folgende Aspekte zu wählen:

- Programmiersprache
- Ablaufplattform
- Realisierung der Geräteservices

Das Design des Frameworks ist grundsätzlich mit allen modernen Programmiersprachen realisierbar und ließe sich für eine Vielzahl an Ablaufplattformen anpassen. Da die Implementierung dem Zweck dient, die Güte des Framework Designs zu bewerten, und diese nicht für den produktiven Einsatz im Krankenhaus gedacht ist, wurden als Auswahlkriterien die kostenlose Verfügbarkeit, die Offenheit des Quellcodes für evtl. nötige Anpassungen sowie existierendes eigenes Know-How herangezogen. Als Folge wurde das Framework in der Programmiersprache Java umgesetzt, als Ablaufplattform wurde der Tomcat[3] Anwendungsserver gewählt.

Die Wahl der Technologie für die Realisierung der Geräteservices ist hingegen fundamental und ungleich komplexer als die Wahl der Programmiersprache oder der Ablaufplattform, da die Erfüllung einer Reihe von Anforderungen direkt davon abhängig ist. Grundsätzlich sind verschiedenste Technologien geeignet, um Services zu realisieren. Die wichtigsten sind OSGi, HAVi (Home Audio Video Interoperability), Jini, UPnP (Universal Plug and Play), Web Services und DPWS (Devices Profile for Web Services) [17]. Im Kontext von Geräteservices sind jedoch spezifische Anforderungen zu berücksichtigen. Bohn et al. definieren eine Reihe von Anforderungen an eine Technologie, die eine Realisierung von Services *auf* Geräten ermöglicht. Diese Anforderungen lassen sich auf Technologien übertragen, die Services *für* Geräte umsetzen; die Wichtigsten sind [5]:

- Unterstützung von Plug&Play-Mechanismen (ableitbar aus Anforderung 3)
- Programmiersprachenunabhängigkeit (ableitbar aus Anforderung 1)
- Unabhängigkeit vom Netzwerkmedium (ableitbar aus Anforderung 4)
- Hohe Skalierbarkeit (ableitbar aus Anforderung 9)
- Bereitstellung von Sicherheitsmechanismen (ableitbar aus Anforderung 5)

Wie Bohn et al. zeigen, deckt nur DPWS alle Anforderungen ab. Dies überrascht nicht, da DPWS speziell für die Ausführung von Services auf Geräten konzipiert wurde [5]. Im Kontext medizinischer Geräte lassen sich jedoch zwei weitere Anforderungen ableiten:

- Unterstützung von Streaming (ableitbar aus Anforderung 7)
- Unterstützung ereignisbasierter Kommunikation (ableitbar aus Anforderung 8)

Die Unterstützung einer ereignisbasierten Kommunikation ist zwar von DPWS explizit in Form von WS-Eventing vorgesehen, ein universeller Streaming-Mechanismus ist jedoch nicht enthalten [20]. Dennoch erfüllt DPWS die Anforderungen im Vergleich zu den alternativen Technologien am besten. Ein wesentlicher Vorteil ist zudem, dass DPWS den Standard WS-Discovery referenziert, der das Decentralized Service Discovery Pattern auf Basis von Webservice Technologien realisiert [1, 20]. Für die technische Umsetzung von Geräteservices auf Basis von DPWS wird Axis2[4] in Verbindung mit dem Axis2 Toolkit[5] genutzt. Axis2 ist eine Plattform zur Ausführung von Webservices, die innerhalb des Tomcat Anwendungsservers installiert werden kann. Das Axis2 Toolkit macht Axis2 Webservices „DPWS-fähig“. Die Services verhalten sich also konform zum DPWS Standard. Um die neueste DPWS Version zu unterstützen, wurde das Axis2 Toolkit, welches auf dem DPWS Standard in der Version von 2006 basiert, auf die aktuelle DPWS Version von 2009 angepasst.

4.2 Klassenstruktur

Das in Abbildung 9 dargestellte Klassendiagramm ist direkt aus dem Schichtenmodell sowie der Modulstruktur abgeleitet worden. Aus Platzgründen sind nur die zentralen Klassen dargestellt, auf die Angabe der einzelnen Methoden und Felder wurde ebenso verzichtet. Der Grafik ist zudem die Verteilung der Klassen auf die in Abschnitt 3.4 definierten Module zu entnehmen. Die Erweiterungspunkte des Frameworks sind durch Klassen mit der Bezeichnung „…“ angedeutet. Durch Implementierung entsprechender Erweiterungen lässt sich das Framework auf das jeweilige Einsatzgebiet anpassen. Beispielsweise könnten neue Gerätetypen, Gerätemodelle oder auch andere Mechanismen zum Zuweisen der Patienten-Id zu einem Datensatz hinzugefügt werden. In den folgenden Abschnitten wird auf die einzelnen Klassen Bezug genommen.

4.3 Core Container

Das wichtigste Element im Core Container Modul ist die abstrakte Klasse „DynamicalAdapter“, die das Dynamical Adapter Pattern umsetzt (vgl. Abschnitt 3.3). Dazu enthält die Klasse eine Funktionalität, die bei Ausführung alle ihr bekannten Adapter

[3] http://tomcat.apache.org

[4] http://ws.apache.org/axis2

[5] http://ws4d.e-technik.uni-rostock.de/?page_id=15

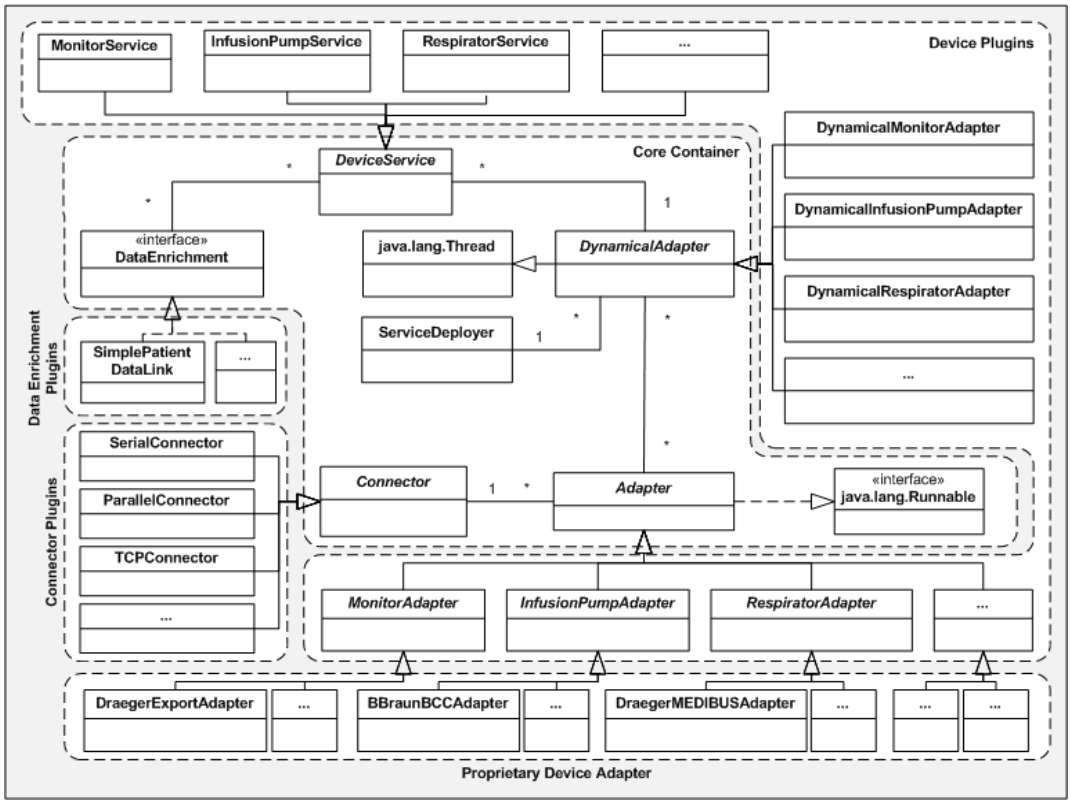

Abbildung 9: Klassenstruktur des SODA@Med Frameworks

testet. Wird ein passender Adapter gefunden, kann die Kommunikation mit dem Gerät begonnen werden. Falls kein Gerät oder kein passender Adapter identifiziert werden konnte, beginnt der DynamicalAdapter erneut mit der Suche nach einem passenden Adapter (Abbildung 10).

Darüber hinaus trägt diese Klasse in Verbindung mit der Klasse „ServiceDeployer“ zur Umsetzung des Auto-Deploy Patterns bei (vgl. Abschnitt 3.4). Im Framework erfolgt das Installieren des Geräteservices, sobald der DynamicalAdapter bereit zur Kommunikation mit dem Gerät ist (Abbildung 10). Beim Auftreten von Kommunikationsfehlern wird der Service automatisch wieder deinstalliert. Das Installieren und Deinstallieren erfolgt unter Verwendung der Klasse ServiceDeployer. Diese packt alle für den Service benötigten Elemente in ein Servicearchiv und übergibt dieses der Axis2-Laufzeitumgebung. Zum Entfernen eines Services löscht die Klasse das entsprechende Servicearchiv, worauf Axis2 den Service automatisch deinstalliert.

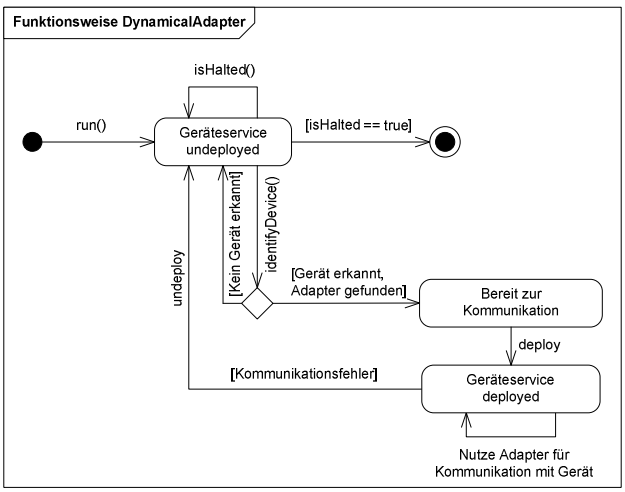

Abbildung 10: Umsetzung der Patterns Dynamical Adapter und Auto-Deploy

Ebenfalls im Core Container angesiedelt sind Hilfsklassen für die Entwicklung von Gerätekonzentratoren. DPWS Services senden sog. Hello- und Bye-Nachrichten sobald sie mit einem Netzwerk verbunden sind bzw. es verlassen. Zudem lässt sich durch das Senden von sog. Probe-, Resolve- und GetMetadata-Nachrichten nach spezifischen DPWS Services suchen sowie deren Metadaten abfragen [1, 20]. SODA@Med enthält Klassen zum Senden und Interpretieren solcher Nachrichten. Diese Funktionalitäten werden von jedem Gerätekonzentrator benötigt.

4.4 Connector Plugins

Konnektoren müssen von der abstrakten Klasse „Connector“ ableiten und somit den Adaptern je einen Stream zum Senden und Empfangen von Daten bereitstellen. Sie bilden damit eine Abstraktionsschicht zwischen Geräten und Adaptern, da ein Adapter den Transportkanal (bspw. seriell oder TCP/IP) nicht kennen muss. Mittels Konnektoren kann zudem das Trusted Subsystem Pattern (vgl. Abschnitt 3.2) unterstützt werden. Evtl. durch technische Maßnahmen zugriffsgeschützte Geräte können über das Bereitstellen eines geeigneten Konnektors angesprochen werden, der entsprechende Authentisierungsmethoden unterstützt.

4.5 Device Plugins

Geräte-Plugins müssen Gerätetyp-spezifische abstrakte Klassen enthalten, die von DynamicalAdapter, DeviceService bzw. Adapter ableiten. Sie befähigen das Framework, einen bestimmten Gerätetyp zu unterstützen. Zur Realisierung des Legacy Wrapper Patterns (vgl. Abschnitt 3.4) und für die technische Umsetzung von Geräteservices folgen wir dem Contract-First Ansatz, bei dem zunächst der Servicevertrag konzipiert wird [8]. Hierfür wurde die Web Service Description Language (WSDL) in Verbindung mit XML-Schema verwendet. Anschließend wurde das Rohgerüst der Implementierung aus dem Servicevertrag generiert. Diese Vorgehensweise unterstützt das Legacy Wrapper Pattern, da die Gestaltung des Servicevertrags manuell erfolgt. Bei Verwendung des Code-First Ansatzes hingegen würde der Servicevertrag aus der Implementierung generiert werden. Dies könnte zur Folge haben, dass Details der Implementierung in den Servicevertrag getragen werden, was einer Verletzung des Legacy Wrapper Patterns entsprechen würde.

4.6 Device Adapter

Geräteadapter müssen von einer Gerätetyp-spezifischen Adapter-Klasse ableiten, die durch ein Geräte-Plugin bereitgestellt wird (vgl. Abschnitt 4.5). Zentral ist hierbei die Implementierung der abstrakten Methode „isApplicable()“. Dynamische Adapter nutzen diese Methode, um zu prüfen, ob der Adapter genutzt werden kann, um mit einem bestimmten Gerät zu kommunizieren. Eine Erläuterung der Implementierung der Geräteadapter zu den von uns genutzten Geräten würde an dieser Stelle zu weit führen. Zudem ist die Umsetzung des vom Hersteller vorgegebenen Protokolls nicht spezifisch für das SODA@Med Framework, sondern muss bei jedem Integrationsansatz vollzogen werden, um eine Kommunikation mit den Geräten zu ermöglichen (vgl. [27]).

4.7 Data Enrichment Plugins

Data Enrichment Plugins müssen die Schnittstelle „DataEnrichment“ implementieren. Im SODA@Med Framework werden diese Plugins in erster Linie genutzt, um die vom Gerät erzeugten Daten korrekt einem Patienten zuzuordnen. In Abhängigkeit von den Prozessen und der IT-Infrastruktur des jeweiligen Krankenhauses sind hierfür verschiedene Strategien möglich (vgl. [27]). Das Szenario für das Testen des Frameworks sieht einen Prozess vor, bei dem die Fallnummer des Patienten mit einem Barcode-Leser ermittelt wird. Durch eine Anfrage am Krankenhausinformationssystem werden alle relevanten Patientendaten geladen und der Patient einem Bett zugeordnet. Für die Zuordnung der Gerätedaten zum Patienten wurde die Klasse SimplePatientDataLink entwickelt. Diese prüft das Vorhandensein der Patienten-Id im vom Geräteservice bereitstehenden Datensatz und setzt diese unter Zuhilfenahme einer internen Tabelle, die eine Zuordnung der Patienten zu den Betten enthält. Diese Tabelle wird durch Abfragen am Krankenhausinformationssystem ständig aktualisiert.

5. EVALUATION

Für die Evaluation des Frameworks wird zum einen dessen Funktionsfähigkeit anhand von Labortests gezeigt. Zum anderen erfolgt die Evaluation kriterienbasiert anhand der Anforderungen.

5.1 Labortests

Um die Funktionsfähigkeit des Frameworks zu testen, wurden verschiedene Szenarien im Labor umgesetzt. Ein ausgewähltes einfaches Szenario soll in diesem Abschnitt verwendet werden, um zu zeigen, wie das Framework genutzt werden kann, um medizinische Geräte auf einfache Weise als Services zu kapseln. Das Szenario sieht vor, dass die Vitaldaten eines Intensivpatienten in regelmäßigen Abständen abgerufen werden sollen. Im Laufe der Zeit wird der verwendete Patientenmonitor für Wartungszwecke durch ein anderes Modell getauscht. Hierbei werden folgende Gerätemodelle verwendet:

- Dräger Infinity® Delta
- Simulierter Patientenmonitor. Da im Labor nur ein realer Monitor zur Verfügung steht, wird das zweite Monitormodell durch eine Software simuliert.

Der Dräger Monitor ist seriell angeschlossen, der simulierte Monitor ist via Netzwerk erreichbar. Für die Integration der Geräte unter Verwendung des SODA@Med Frameworks sind drei Schritte nötig:

1. Entwickeln oder Beziehen der benötigten Plugins
2. Entwickeln oder Beziehen passender Geräte-Adapter
3. Konfigurieren und Ausführen des Frameworks

Die für das Szenario benötigten Plugins sind bereits Bestandteil des Frameworks. Im Zuge der weiteren Entwicklung von SODA@Med wäre es wünschenswert, wenn für alle relevanten Gerätetypen entsprechende Plugins verfügbar wären. Die Entwicklung solcher Plugins ist indes trivial, da die wesentlichen Funktionalitäten bereits durch die abstrakten Klassen des Frameworks bereitgestellt werden. Zur Verdeutlichung: Das Plugin für Patientenmonitore besteht aus fünf trivialen Klassen sowie einer Geräteservice-Implementierung, die aus einer WSDL-Datei generiert wurde (vgl. Abschnitt 4.5). Das Entwickeln des Plugins konnte in ungefähr einer Stunde durchgeführt werden, der größte Aufwand lag hierbei bei der Definition des Servicevertrags.

Der Aufwand für das Entwickeln der Gerätemodell-Adapter in Schritt 2 hängt von der Komplexität der proprietären Schnittstellen ab. Pro Adapter kann dies wenige Stunden bis einige Tage in Anspruch nehmen. Das Implementieren des vom Hersteller vorgegebenen Protokolls ist jedoch unabhängig vom Integrationsansatz unvermeidbar, um eine Kommunikation mit dem jeweiligen Gerät zu ermöglichen (vgl. [27]).

Das Konfigurieren des Frameworks in Schritt 3 erfolgt unter Verwendung des Spring Frameworks[6]. Dieses unterstützt das Prinzip der Dependency Injection. Dies bedeutet, dass Objekte ihre benötigten Objekte und Ressourcen nicht selbst erzeugen, sondern diese von außen „injiziert“ bekommen; auf diese Weise werden Abhängigkeiten reduziert. Im SODA@Med Framework dient dieses Prinzip dazu, die einzelnen Plugins zu einem lauffähigen Gesamtsystem zusammenzusetzen.

Nach Abschluss der Konfiguration wurde im Labor getestet, ob sich das Framework wie geplant verhält. Nach dem Starten des Anwendungsservers begann der dynamische Adapter unter Verwendung der ihm bekannten proprietären Adapter mit dem Versuch, eine Verbindung zum Patientenmonitor aufzubauen. Zu diesem Zeitpunkt war der Geräteservice wie vorgesehen noch nicht verfügbar. Nachdem der Dräger Monitor angeschaltet wurde, konnte dieser durch das Framework gefunden werden.

[6] http://www.springsource.org

Anschließend wurde automatisch der zugehörige Geräteservice installiert und war ab sofort im Netzwerk verfügbar. Durch entsprechende DPWS Anfragen konnte der Service im lokalen Netzwerk gefunden werden. Die Abfrage der aktuellen Vitaldaten war problemlos möglich, die im Datensatz ursprünglich fehlende Patienten-Id wurde korrekt gesetzt. Nach dem Abschalten des Dräger Monitors wurde der zugehörige Geräteservice planmäßig deinstalliert. Anschließend wurde der simulierte Monitor gestartet, um einen Gerätewechsel nachzuahmen. Dieser wurde durch das Framework dynamisch erkannt. Der Geräteservice wurde entsprechend erneut installiert und die Vitaldaten konnten wieder abgerufen werden. Die Funktionsfähigkeit des Frameworks konnte somit belegt werden.

5.2 Reflexion der Anforderungen

Anforderung 1 fordert die Möglichkeit zur Integration der medizinischen Geräte mit verschiedensten Systemen. Dies wird durch die Kapselung der Geräte als DPWS Services realisiert. Solche Services sind ihrer Natur nach für die Kommunikation mit verschiedensten Systemen konzipiert und basieren auf programmiersprachenunabhängigen Standards [9]. Darüber hinaus erleichtert die Anwendung des Device Concentrator Patterns den Zugriff auf die von den Geräten erzeugten Daten.

Anforderung 2, das korrekte Zuordnen medizinischer Daten zu einem Patienten, wird durch Umsetzung eines Data Enrichment Plugins realisiert. Durch die Erweiterbarkeit des Frameworks kann der entsprechende Mechanismus an die Gegebenheiten des jeweiligen Krankenhauses angepasst werden. Das beispielhafte SimplePatientDataLink Plugin hatte die Patienten-Id im Rahmen der technischen Tests korrekt gesetzt.

Anforderung 3 beinhaltet eine Art Plug&Play-Mechanismus und wird durch das Dynamical Adapter Pattern umgesetzt. Das Framework versucht, neu angeschlossene Geräte zu erkennen und einen passenden Adapter zu finden. Ein manuelles Eingreifen durch das medizinische Personal ist dabei nicht notwendig. Die Funktionsfähigkeit der Umsetzung dieses Patterns wurde im Rahmen der technischen Tests belegt.

Anforderung 4, die Unterstützung unterschiedlichster Hardware-Schnittstellen, wird durch das Trennen der Adapterlogik von der Transportlogik realisiert. Die Erfüllung der Anforderung konnte belegt werden, da im Test für einen Geräteservice verschiedene Transportmechanismen (seriell und TCP/IP) genutzt wurden.

Die Prüfung der Erfüllung von Anforderung 5, die Konformität zur IEC 80001, erfordert das Durchlaufen des entsprechenden Risikomanagementprozesses. Dies ist für jedes Einsatzszenario individuell und zudem abhängig von den Gegebenheiten in der jeweiligen Klinik [10]. Durch das Framework werden jedoch Mechanismen bereitgestellt, um das Risiko eines Patientenschadens möglichst gering zu halten. Hierfür wird das Trusted Subsystem Pattern sowie des Data Enrichment Pattern unterstützt, um negative Seiteneffekte durch einen unkontrollierten Direktzugriff auf das Gerät bzw. das falsche Zuordnen von Daten zu verhindern. Darüber hinaus werden die von den Geräten erzeugten Daten maximal in den Adapter-Implementierungen geändert, um diese an das Datenformat des Geräteservices anzupassen.

Anforderung 6 besagt, dass die von den Herstellern angegebenen Zweckbestimmungen der Geräteschnittstellen berücksichtigt werden müssen. Für das Framework ergeben sich hieraus keine direkten Konsequenzen. Jedoch muss bei Verwendung der entsprechenden Geräteservices sichergestellt werden, dass die Zweckbestimmungen nicht verletzt, die Daten also bspw. nicht zur Diagnostik verwendet werden.

Das Übertragen von Streamingdaten (Anforderung 7) ist eine Funktionalität des jeweiligen Geräteservices und betrifft die hierfür verwendete Technologie. Wie in Abschnitt 4.1 angemerkt, unterstützt DPWS aktuell keinen universellen Streaming-Mechanismus, eine solche Funktionalität müsste daher proprietär umgesetzt werden. Es lässt sich jedoch feststellen, dass aktuell kein Standard für Streaming im Kontext von SOA existiert. Im Rahmen einer Diskussionrunde während des 1. TeKoMed-Workshops[7] wurde die Notwendigkeit eines solchen Standards diskutiert und eine Arbeitsgruppe gegründet, die einen entsprechenden Entwurf erstellen soll[8]. Nach Einschätzung dieser Expertengruppe wird ein solcher Standard in einer zukünftigen DPWS Version unterstützt werden.

Anforderung 8, die ereignisbasierte Übertragung von Daten, betrifft ebenfalls die Serviceimplementierung und die hierfür verwendete Technologie. Wie in Abschnitt 4.1 angemerkt, unterstützt DPWS die Verwendung von WS-Eventing und damit die ereignisbasierte Kommunikation.

Anforderung 9, das Übertragen großer Datenmengen, ist überwiegend eine Sache der Auslegung der System- und Netzwerk-Kapazitäten und muss auf den jeweiligen Anwendungsfall angepasst werden. Beschränkungen von Seiten des Frameworks sind hierbei nicht festzustellen. Zudem unterstützt DPWS die Übertragung von Binärdaten durch Verwendung des MTOM Standards (Message Transmission Optimization Mechanism) [20].

6. ZUSAMMENFASSUNG UND AUSBLICK

Mit der Konzipierung und prototypischen Implementierung des SODA@Med Frameworks wurde ein erster Schritt getan, um das allgemeine SODA-Konzept auf das medizinische Anwendungsfeld zu adaptieren und somit die identifizierte Forschungslücke eines fehlenden verallgemeinerten serviceorientierten Integrationskonzepts für medizinische Geräte zu schließen. Es wurde gezeigt, dass die aus der Literatur und drei Fallstudien ermittelten Anforderungen erfüllt werden. Lediglich für Streaming-Funktionalitäten gibt es Einschränkungen, da hierfür noch kein universeller Standard auf Basis von SOA Technologien existiert. Es ist jedoch zu erwarten, dass diese Lücke mit der nächsten Version von DPWS geschlossen wird.

Kritisch zu betrachten ist die Tatsache, dass das Einsatzgebiet medizinischer Geräte sehr breit ist und eine Vollständigkeit der Anforderungen auf Basis von drei Fallstudien nicht angenommen werden kann. Vor diesem Hintergrund ist SODA@Med als erster Entwurf zu sehen, der durch weitere Forschung an ggf. zusätzliche Anforderungen angepasst werden sollte. Weitere Forschung bedarf es auch hinsichtlich der Evaluation sowie der

[7] 1. Workshop Technologische Kompatibilität in der Medizintechnik durch Service-orientierte Architekturen, 1.6.-2.6.2010, Institut für Telematik, Universität zu Lübeck

[8] http://trac.e-technik.uni-rostock.de/projects/ws-streaming

Bewertung des Einsatzes serviceorientierter Konzepte im Kontext medizinischer Geräte. Zwar konnte eine positive Reflexion der Anforderungen gezeigt werden, jedoch muss sich das Framework in einem nächsten Schritt in der Praxis bewähren. Zudem ist systematisch zu analysieren, welche Vor- und Nachteile die Verwendung serviceorientierter Konzepte im Vergleich zu bisherigen Integrationskonzepten bietet. Weiterer Forschungsbedarf ist hinsichtlich der semantischen Interoperabilität und damit hinsichtlich des Service Designs zu sehen. Das Framework selbst ist hiervon jedoch nicht betroffen, da diesem über die Plugins jedes beliebige semantische Modell injiziert werden kann.

7. LITERATUR

[1] Web Service Dynamic Discovery (WS-Discovery) Version 1.1. OASIS ed., 2009.

[2] Becker, J. Prozess der gestaltungsorientierten Wirtschaftsinformatik. in Oesterle, H., Winter, R. and Brenner, W. eds. Gestaltungsorientierte Wirtschaftsinformatik: Ein Plädoyer für Rigour und Relevanz, Infowerk, Nürnberg, 2010, 13-17.

[3] Bennett, K. Legacy Systems: Coping with Success. IEEE Software, 12 (1). 19-23.

[4] Bisbal, J., Lawless, D., Bing, W. and Grimson, J. Legacy information systems: issues and directions. IEEE Software, 16 (5). 103-111.

[5] Bohn, H., Bobek, A. and Golatowski, F., SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains. in International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICN/ICONS/MCL 2006), (2006), IEEE Computer Society.

[6] Bridges, M.W. SOA in Healthcare. Health Management Technology, 28 (6). 6-10.

[7] de Deugd, S., Carroll, R., Kelly, K.E., Millett, B. and Ricker, J. SODA: Service Oriented Device Architecture. Pervasive Computing, IEEE, 5 (3). 94-96.

[8] Erl, T. SOA Design Patterns. Prentice Hall International, Boston, 2009.

[9] Erl, T. SOA Principles of Service Design. Prentice Hall International, Boston, 2007.

[10] Gärtner, A. IEC 80001: Risikomanagement vernetzter medizinischer Systeme. in Jäckel, A. ed. Telemedizinführer Deutschland 2009, Minerva, Bad Nauheim, 2008, 40-44.

[11] Hevner, A.R., March, S.T., Park, J. and Ram, S. Design Science in Information Systems Research. MIS Quarterly, 28 (1). 75-105.

[12] Lesh, K., Weininger, S., Goldman, J.M., Wilson, B. and Himes, G. Medical Device Interoperability – Assessing the Environment Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability, 2007.

[13] Mauro, C., Leimeister, J.M. and Krcmar, H. The Nature of Medical Device Services - A Multiple-Case Study 4th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC), Rome, Italy, 2011.

[14] Mauro, C., Leimeister, J.M. and Krcmar, H. Service Oriented Device Integration – An Analysis of SOA Design Patterns 43st Annual Hawaii International Conference on System Sciences (HICSS-43), Kauai, Hawaii, 2010.

[15] Mauro, C., Leimeister, J.M. and Krcmar, H. Serviceorientierte Integration medizinischer Geräte – ganzheitliche IT-Unterstützung klinischer Prozesse. Informatik-Spektrum, Online First.

[16] Mauro, C., Sunyaev, A., Dünnebeil, S., Leimeister, J.M. and Krcmar, H. Mobile Anwendungen im Kontext des Medizinproduktegesetzes. Fischer, S., Maehle, E. and Reischuk, R. eds. Informatik 2009 - Im Focus das Leben, Köllen Druck+Verlag GmbH, Lübeck, 2009.

[17] Mauro, C., Sunyaev, A., Leimeister, J.M. and Krcmar, H. Service-orientierte Integration medizinischer Geräte - eine State of the Art Analyse Wirtschaftsinformatik 2009 - Business Services: Konzepte, Technologien und Anwendungen, Wien, 2009, 119-128.

[18] Mauro, C., Sunyaev, A., Leimeister, J.M. and Krcmar, H. Standardized Device Services - A Design Pattern for Service Oriented Integration of Medical Devices 43st Annual Hawaii International Conference on System Sciences (HICSS-43), Kauai, Hawaii, 2010.

[19] Melrose, J.P. e-health is the way via SOA. Healthcare Financial Management, 61 (3). 120-122.

[20] OASIS. Devices Profile for Web Services Version 1.1, 2009.

[21] Pöhlsen, S. Entwicklung einer Service-orientierten Architektur zur vernetzten Kommunikation zwischen medizinischen Geräten, Systemen und Applikationen, Universität zu Lübeck, Lübeck, 2010.

[22] Schweiger, A., Sunyaev, A., Leimeister, J.M. and Krcmar, H. Toward Seamless Healthcare with Software Agents. Communications of the Association for Information Systems (CAIS), 19. 692-709.

[23] Shaikh, A., Memon, M., Memon, N. and Misbahuddin, M., The Role of Service Oriented Architecture in Telemedicine Healthcare System. in Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09. International Conference on, (2009), 208-214.

[24] Siebenhaar, M., Lehrig, T., Braun, J. and Görge, T. Entwicklung einer SOA-basierten Webanwendung zur Buchung und Verwaltung von Segeltouren: Proprietäre Software vs. Open Source Wirtschaftsinformatik, 50 (4). 325-329.

[25] Sunyaev, A., Leimeister, J.M., Schweiger, A. and Krcmar, H. Integrationsarchitekturen für das Krankenhaus - Status quo und Zukunftsperspektiven. In: Information Management & Consulting (IMC). Information Management & Consulting (IMC), 21 (1). 28-35.

[26] Tanenbaum, A.S. Computernetzwerke. Pearson Studium, München, 2003.

[27] Zaleski, J. Integrating Device Data into the Electronic Medical Record: A Developer's Guide to Design and a Practitioner's Guide to Application. Publicis Publishing, Erlangen, 2008.