Karlsruher Institut für Technologie

Institut für Angewandte Informatik
und Formale Beschreibungsverfahren

Terminplanintegration an Kliniken am Beispiel von Betty24 und AGFA ORBIS

Bachelorarbeit
von
Tobias Dillig

an der Fakultät für Wirtschaftswissenschaften

In dem Studiengang
Informationswirtschaft

eingereicht am 31.10.2016 beim
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren
des Karlsruher Instituts für Technologie

Prüfer: Prof. Dr. York Sure-Vetter
Betreuer: Patrick Philipp, Anne Zander

KIT – Universität des Landes Baden-Württemberg und
nationales Forschungszentrum in der Helmholtz-Gemeinschaft

Eidesstattliche Erklärung

Ich versichere hiermit wahrheitsgemäß, die Arbeit und alle Teile daraus selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

Karlsruhe, den 31.10.2016 *Tobias Dillig*

Inhaltsverzeichnis

1 Einleitung und Motivation **1**

1.1 Fragestellung und Zielsetzung 2

1.2 Vorgehensweise 2

2 Grundlagen **3**

2.1 Krankenhausinformationssystem 3

2.1.1 Definition 3

2.1.2 Terminverwaltung 3

2.2 Online-Terminmanagementsysteme 4

2.2.1 Definition 4

2.2.2 Terminverwaltung 4

2.2.3 Schnittstellenproblematik 5

2.3 Ontologie 5

2.3.1 Ontologiebasierte Datenintegration 6

2.3.2 RESTful Linked Data 6

2.3.3 Linked Data Warehousing 6

2.4 Related Work 7

3 Szenario **8**

3.1 Ist-Zustand 8

3.2 Soll-Zustand 8

3.3 Anforderungsanalyse 9

4 Umsetzung **11**

4.1 Schnittstellenbeschreibung 11

4.1.1 Schnittstelle des OTMS Betty24 11

4.1.2 Schnittstelle des KIS AGFA ORBIS 12

4.2 Konzeptueller Aufbau 12

4.2.1 Entwicklung des RDF-Vokabulars 13

4.2.2 Aufbau der Wrapper 14

4.2.3 Linked Data-Fu (ldfu) 15

4.2.4 Terminportal 15

4.3 Technische Implementierung 16
4.3.1 Wrapper 16
4.3.2 Linked Data-Fu 17
4.3.3 Terminportal 18
4.3.4 Anforderungen im Betrieb 19

5 Evaluation **20**
5.1 Validität 20
5.2 Skalierbarkeit 20
5.3 Echtzeitsynchronisation 23

6 Fazit **24**
6.1 Ausblick 25

A Anhang **26**
A.1 Testergebnis Funktionstest 26
A.2 RDF-Vokabular 26
A.3 Testdokumentation 26
A.4 Quellcode 27
A.5 Demo Terminportal 27

Abkürzungsverzeichnis

API	Application programming interface
CSV	Comma-separated values
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
KIS	Krankenhausinformationssystem
ldfu	Linked Data-Fu
OTMS	Online-Terminmanagementsystem
OWL	Web Ontology Language
PIS	Praxisinformationssystem
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
SKK	Städtisches Klinikum Karlsruhe
URI	Uniform Resource Identifier
VM	Virtuelle Maschine

Abbildungsverzeichnis

1 Weboberfläche zur Terminbuchung beim OTMS Betty24 5

2 Ordnerstruktur, über welche Im- und Export in und aus AGFA ORBIS simuliert werden. 12

3 Komponentendiagramm des zu entwickelnden Softwareprodukts 13

4 Klassendiagramm der myfhir-Ontologie. Der Übersicht wegen wird nur der Klassenname ohne weitere Details dargestellt. 14

5 Die Wrapper stellen vereinheitlichte RESTful Linked Data APIs bereit. . . 15

6 Laufzeiten der Integration bei verschiedener Terminanzahl 21

7 Laufzeiten der Integration bei unterschiedlicher Terminanzahl (lokal) . . . 23

8 Grafische Ergebnisdokumentation des Funktionstests (vgl. Kapitel 5.1) . . 26

Tabellenverzeichnis

1 Relevante APIs des OTMS Betty24 11
2 Content Negotiation der Wrapper 16
3 Spezifikation der VM im SKK 19
4 Minimalanforderungen für den Betrieb 19
5 Spezifikation der Test-VM 21
6 Laufzeiten der Integration bei verschiedener Terminanzahl 21
7 Laufzeiten der Integration bei lokaler Ausführung 22

1 Einleitung und Motivation

Die Vereinbarung eines Arzttermins, wie wir sie heute kennen, ist zeitlich aufwendig und ineffizient: Überwiegend telefonisch werden Termindaten zwischen Patient und Klinikpersonal ausgetauscht bis ein geeigneter Termin gefunden wird. Da jeder Beteiligte nur Zugriff auf den eigenen Terminkalender hat, ist es für beide Seiten sehr mühsam einen passenden Termin zu finden. Zudem ist die telefonische Erreichbarkeit abhängig von den Öffnungszeiten der Praxis oder Ambulanz. Alternative Terminbuchungsverfahren sind nur selten anzutreffen (vgl. [Pri15]).

Die Ambulanz der Frauenklinik am Städtischen Klinikum Karlsruhe[1] (SKK) bietet ihren Patienten in Kooperation mit Betty24[2] ein solches alternatives Verfahren an: Bereits seit einigen Monaten können Patienten Termine online buchen. Dabei haben sie, je nach Besuchsgrund, wöchentlich einige Termine zur Auswahl. Diese lassen sich direkt buchen und können online auch jederzeit wieder storniert oder umgebucht werden. Darüber hinaus kann der Patient den gebuchten Termin mit einem Klick in seinen digitalen Kalender übertragen oder zu einem wählbaren Zeitpunkt per E-Mail daran erinnert werden.

Neben all den Vorteilen für die Patienten profitiert auch die Ambulanz vom neuen Terminbuchungsverfahren: Da weniger Termine vor Ort oder telefonisch vereinbart werden, wird das Personal entlastet und hat mehr Zeit für andere Tätigkeiten.

Bei all den Vorteilen gibt es derzeit aber auch noch Probleme: die online gebuchten Termine müssen im Kalender der Ambulanz manuell eingetragen werden. Durch eine Automatisierung dieses Transfers könnten, anstatt wenigen ausgewählten Terminen, alle verfügbaren Termine online angeboten werden. Erst durch diesen Schritt ist eine großflächige Nutzung für alle Patienten des SKK möglich.

[1] `http://www.klinikum-karlsruhe.com/`
[2] `https://betty24.de/`

1.1 Fragestellung und Zielsetzung

Im Rahmen der Bachelorarbeit soll evaluiert werden, inwiefern sich das Schnittstellenproblem am SKK lösen lässt. Hierfür wird ein Softwareprodukt zur Terminplanintegration entwickelt und evaluiert. Schwerpunkte der Arbeit werden dabei folgende Fragen sein:

- Ausgehend vom Ist-Zustand: Wie wäre der Soll-Zustand? Welche Anforderungen müssen hierfür erfüllt sein? Ist dieser Optimalzustand überhaupt erreichbar?
- Ist eine bidirektionale Integration und Synchronisation der Terminplandaten möglich? Bidirektional bedeutet hierbei, dass Online-Terminbuchungen in den Kalender der Ambulanz übertragen werden und telefonisch oder vor Ort gebuchte Termine auch online geblockt werden und somit nicht zur Buchung verfügbar sind.
- Wie performant ist die Integration der Terminplandaten? Gibt es zeitliche Verzögerungen? Wie verhält sich das System bei sehr vielen Terminen?

1.2 Vorgehensweise

Um die eigentliche Problematik korrekt und vollumfänglich zu erfassen, wird zunächst der Ist-Zustand am SKK erfasst. Anschließend wird der Soll-Zustand beschrieben und Anforderungen daraus abgeleitet (s. Kapitel 3.3). Für die Schnittstellenlösung wird der Ansatz einer ontologiebasierten Integration mithilfe von Linked Data-Fu[3] (vgl. [SSHS13]) evaluiert, wofür in Kapitel 4.2.1 ein RDF-Vokabular entwickelt wird. Ergänzend wird eine dritte Softwarekomponente implementiert um die integrierten Daten anzuzeigen und einen Ausblick auf weitere Forschungs- und Entwicklungsmöglichkeiten zu offerieren.

Nach Abschluss der Entwicklung des Prototypen wird dieser auf Basis der Anforderungsanalyse evaluiert.

[3] `http://linked-data-fu.github.io/`

2 Grundlagen

In diesem Kapitel werden die wichtigsten Grundlagen erläutert, welche für das Verständnis der weiteren Arbeit dienen. Zu Beginn werden die Begriffe *Krankenhausinformationssystem* und *Online-Terminmanagementsystem* definiert. Anschließend wird das Themenfeld um Ontologien, *semantic data* und *linked data* strukturiert und erläutert.

2.1 Krankenhausinformationssystem

2.1.1 Definition

Bereits 1973 definierte Köhler den Begriff Krankenhausinformationssystem (KIS) wie folgt:

„Die Funktion des Krankenhauses besteht aus der Integration aller vorhandenen Mittel und aller Handlungen der in ihm tätigen Personen mit dem Ziel der Gesundung von Patienten; das Krankenhaus-Informationssystem ist eines dieser Mittel.“ [Kö82]

Im Laufe der Jahre hat sich diese sehr freie und umfangreiche Definition konkretisiert und mehr auf die technischen Komponenten fokussiert:

„Ein KIS besteht aus verschiedenen informationsverarbeitenden Systemen, die über Schnittstellen miteinander verbunden sind. Hierzu gehören bspw. das klinische Arbeitsplatzsystem, das klinische Kommunikationssystem, die elektronische Patientenakte oder das Radiologieinformationssystem.“ [OHG+15]

Über die in der Definition bereits genannten Systeme hinaus bietet ein KIS weitreichende Funktionen zum Patientenmanagement, der Behandlungsdokumentation und der Abrechnung von Leistungen gegenüber Krankenkassen und Privatpersonen (vgl. [OHG+15]). In der Regel besteht ein KIS aus heterogener Software verschiedener Hersteller mit vielen Schnittstellen. Das übliche Datenformat zum Austausch von Informationen zwischen den einzelnen Systemen ist der HL7-Standard[4]. Ein Teil eines KIS, die Terminverwaltung, wird im folgenden Kapitel genauer betrachtet.

2.1.2 Terminverwaltung

KIS haben in der Regel eine sehr weitreichende Terminmanagementfunktion. So wird über die Kalenderfunktionalität zum Beispiel oft auch die Raumverwaltung vorgenommen. Zudem gibt es zahlreiche Kalender für unterschiedliche Teams und behandelnde Ärzte. Termine bestehen dabei nicht nur aus Name, Zeit und Ort, sondern sind mit weiteren Daten verknüpft. Dem Termin sind in der Regel direkt der Patient (sofern er bereits in der Klinik behandelt wurde), als auch der Kontext zum Besuch (Fallbezug) zugeordnet. So kann das Klinikpersonal und die behandelnden Ärzte sofort die Patientenakte aufrufen

[4] `http://www.hl7.org/`

und die Behandlungsdokumentation des letzten Termins nachlesen.

Ist ein Patient bei der Terminvereinbarung noch nicht im System erfasst, wird ein Freitexttermin ohne Patienten- und Fallbezug angelegt. Erscheint der Patient dann zu seinem Termin, wird der Patientenbezug vor Ort verknüpft. Der Fallbezug wird im weiteren Behandlungsverlauf hergestellt und steht somit für Folgetermine zur Verfügung.

Am SKK kommt das KIS *ORBIS* des Herstellers *Agfa HealthCare GmbH*[5] zum Einsatz. Dieses bietet eine solche Terminmanagementfunktion.

2.2 Online-Terminmanagementsysteme

In diesem Abschnitt wird der Begriff *Online-Terminmanagementsystem* definiert und auf die typischen Eigenschaften eingegangen. Anschließend wird die Hauptproblematik beim Einsatz von Online-Terminmanagementsystemen herausgearbeitet.

2.2.1 Definition

Ein Online-Terminmanagementsystem (OTMS) ist ein Softwaresystem, welches die asynchrone Terminvereinbarung im Internet ermöglicht. Hierfür muss sich der Patient im Regelfall online registrieren, bevor er bei einer teilnehmenden Arztpraxis Termine buchen kann. Im Gegensatz zum klassischen Terminvereinbarungsprozess per Telefon oder vor Ort ist die Online-Terminbuchung asynchron, ortsunabhängig, zeitunabhängig und aufgrund der vollständigen Automatisierung des Prozesses skalierbar.

Am deutschen Markt gibt es bereits einige Anbieter von OTMS. In dieser Arbeit wird immer Bezug auf das OTMS Betty24[6] genommen.

2.2.2 Terminverwaltung

Aus Patientensicht bietet das OTMS eine Weboberfläche zur Buchung von Terminen an. Der teilnehmende Arzt kann hierbei genau festlegen, welche Terminslots für welchen Besuchsgrund angeboten werden. Abbildung 1 zeigt die Weboberfläche des OTMS Betty24. Nach der Buchung des Termins durch den Patient erhält dieser die Möglichkeit den Termin zu seinem Kalender hinzuzufügen. Der teilnehmende Arzt erhält, sofern er dies wünscht, eine E-Mail mit den Termindaten. Alternativ kann er den gebuchten Termin über die API des Anbieters automatisch zu seinem Kalender hinzufügen lassen. Dies ist Thema des nächsten Kapitels.

[5] `http://www.agfahealthcare.com/`
[6] `https://www.betty24.de/`

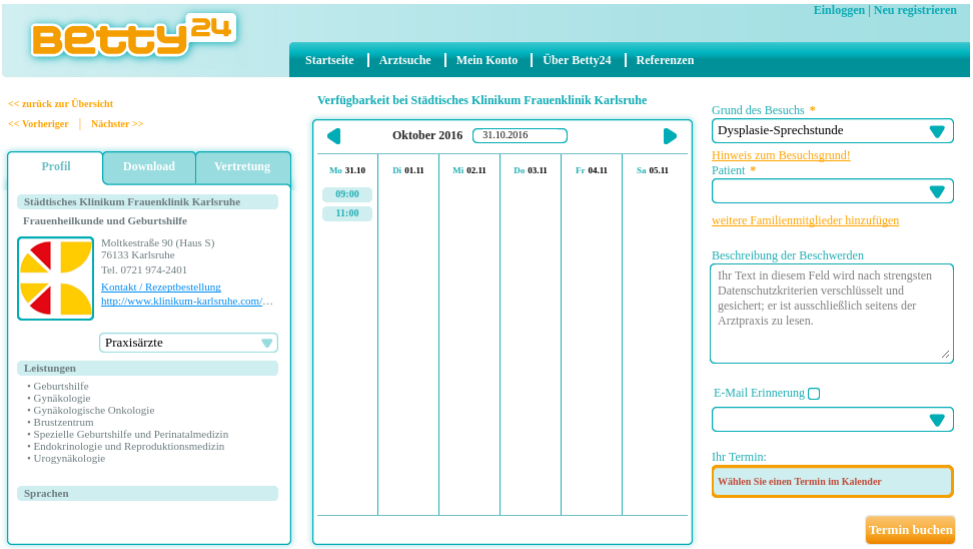

Abbildung 1: Weboberfläche zur Terminbuchung beim OTMS Betty24

2.2.3 Schnittstellenproblematik

Wie bereits beschrieben bietet das OTMS Betty24 Schnittstellen zur Integration ihres Produktes in bestehende Praxisinformationssysteme (PIS) und KIS an. Die Anbindung von Drittsystemen ist bei den meisten KIS allerdings eher problematisch: Neben dem hohen Entwicklungsaufwand für passende Softwarelösungen fallen häufig hohe Lizenzkosten an. Dies führt zu dem Problem, dass die Verwendung eines OTMS aus den oben genannten Gründen sehr wünschenswert, die Integration in die bestehende Softwarelandschaft oft aber sehr kompliziert und teuer ist. Diese Arbeit soll auf Basis einer ontologiebasierten Integration (vgl. Kapitel 2.3.1) den Komplexitätsgrad reduzieren und eine einfache Verwendung des Prototypen ermöglichen.

2.3 Ontologie

Der Begriff *Ontologie* stammt ursprünglich aus der Philosophie und bedeutet *die Lehre vom Sein* (vgl. [BHL+14]). In der Informatik gibt es viele Definitionen, die sich stark unterscheiden. [BHL+14] haben verschiedene Definitionen zusammengefasst und definieren Ontologie als „formale Definition von Begriffen und deren Beziehungen als Grundlage für ein gemeinsames Verständnis“. Im Kontext von semantischen Daten wird Ontologie oft auch als Äquivalent für Wissensbasis genutzt (vgl. [HKRS08]) und beschreibt ein Dokument, welches in einer Ontologiesprache geschrieben wurde. Für diese Arbeit sind die Ontologiesprachen RDFS [GB14] und OWL [HHPS04] relevant.

2.3.1 Ontologiebasierte Datenintegration

Bei der ontologiebasierten Integration werden Daten mit Hilfe von einer oder mehreren Ontologien integriert. Hierfür müssen unstrukturierte und semi-strukturierte Daten „geliftet“, also zu semantischen und damit strukturierten Daten umgewandelt werden. Dank des semantischen Kontext können die Daten dann, auch wenn sie unterschiedliche Vokabulare verwenden, integriert werden, sofern es eine entsprechende Verbindung zwischen den Vokabularen gibt. Nutzen sie die gleichen Vokabulare, ist die Integration der semantischen Daten trivial.

2.3.2 RESTful Linked Data

Unter *Linked Data* versteht man die Benutzung von Webtechnologien, um verteilte semantische Daten und ihre Definitionen zu verknüpfen. Berners-Lee erarbeitete 2006 vier Grundregeln für das Veröffentlichen von Linked Data (auch „Linked Data principles“ genannt, vgl. [BHBL09]):

1. Benutzung von URIs zur Benennung von Objekten
2. Benutzung des HTTP-Protokolls zum Abrufen der Objekte
3. Bereitstellen von nützlichen Informationen im richtigen Format (RDF, SPARQL)
4. Verlinken weiterer Objekte

Diese Grundregeln sollen die Erschaffung eines einzigen globalen Datenraums ermöglichen, in dem die Objekte miteinander verlinkt sind. Zusammenhänge ergeben sich hierbei nicht durch die Benutzung der gleichen Terminologie beziehungsweise des gleichen Vokabulars, sondern durch Verlinkungen (z.B. *owl:sameAs*, [HHPS04]).

RESTful Linked Data ist eine Erweiterung der „Linked Data principles“ um die REST-Paradigmen. [RA13] fasst diese in Form der REST Maturity Models in drei Level zusammen:

1. Benutzung von URIs für Ressourcen
2. Benutzung der HTTP Methoden (GET, POST, PUT, DELETE, ...) zum Abruf und zur Manipulation von Ressourcen
3. Verlinken weiterer Ressourcen während der Interaktion mit der Schnittstelle

2.3.3 Linked Data Warehousing

Bei der Verarbeitung von Linked Data steht man vor der Herausforderung, dass nicht nur eine Datenquelle existiert, sondern beliebig viele verteilte Datenquellen welche über

HTTP abgerufen werden können. *Linked Data Warehousing* beschreibt den Ansatz, alle Daten zu aggregieren und auf dem aggregierten Datensatz zu arbeiten. Dies bringt unter anderem den Nachteil mit sich, dass alle Daten heruntergeladen und aggregiert werden müssen, bevor mit der Auswertung begonnen werden kann. Zudem ist es ohne hohen Aufwand nicht möglich, zusätzliche Informationen hinzuzufügen oder Daten, abhängig von ihrem Ursprung, häufiger oder seltener zu aktualisieren.

Ein alternativer Ansatz ist das Arbeiten mit *Linked Data Streams* (vgl. [HKS+13]), wodurch auch Echtzeitzugriff auf die Ursprungsdaten ermöglicht wird.

2.4 Related Work

Da OTMS eine relativ neue Entwicklung sind, gibt es wenig wissenschaftliche Literatur zum Thema, insbesondere zur Integration von Terminplandaten. [Pri15] stellt in seiner Masterarbeit *Online-Terminmanagementsysteme als Optimierungs- und Marketinginstrument für Arztpraxen in Deutschland* einen Überblick zum aktuellen Ablauf von Terminbuchungen dar. Darüberhinaus geht der Autor insbesondere auf Grundlagen, Begrifflichkeiten, Struktur und Optimierungspotentiale von und durch OTMS ein und leitet daraus letztendlich einen Leitfaden für eine erfolgreiche Entwicklung und den Betrieb eines OTMS ab. Er hebt die Rolle der Datenintegration als wichtige Anforderung zur Benutzung eines OTMS hervor, liefert jedoch keinen Lösungsansatz dazu.

[MSK11] zeigt die Relevanz semantischer Technologien für alle Bereiche, in denen Daten verarbeitet werden. Die Autoren stehen ebenfalls vor der Herausforderung, Daten in heterogenen Datenformaten und verschiedenen Standards auf Basis einer ontologiebasierten Integration zu aggregieren. Sie modellieren hierfür ein eigenes Vokabular in RDF und RDFS, führen die Integration von Daten aber nicht durch. Auch findet Linked Data keine Beachtung: es wird also von rein lokal vorliegenden Daten ausgegangen.

[KWO+14] präsentieren ein System zur automatischen Integration und Auswertung von Finanzanalyseinformationen. Auch wenn ihr System dabei nicht auf ldfu basiert, ist die Funktionsweise vergleichbar mit dem Vorhaben, Termindaten aus unterschiedlichen Quellen zu integrieren. Auch hier findet die Integration auf Basis semantischer Daten (Linked Data) statt, wenn auch durch andere Software unterstützt und ohne Synchronisation aller Datenquellen.

In [TKSA12] befassen sich die Autoren mit der Integration von Daten verschiedener APIs. Zum Umwandeln der API-Rückgaben in strukturierte, semantische Daten (Linked Data) verwenden die Autoren *Karma*[7], ein Webservice zum automatischen „*Liften*“ und Integrieren von Daten aus Dateien, APIs oder Datenbanken. Der Fokus liegt hier insbesondere auf einer grafischen Oberfläche zum Modellieren des „*lifting*“-Prozesses und weniger auf der Integration und Synchronisation von Daten.

[7] `http://usc-isi-i2.github.io/karma/`

3 Szenario

Im folgenden Abschnitt wird der Ist-Zustand am SKK erfasst. Anschließend wird das Soll-Szenario beschrieben und daraus eine Anforderungsanalyse abgeleitet.

3.1 Ist-Zustand

Aktuell bietet das SKK ausschließlich in der Gynäkologischen Ambulanz der Frauenklinik[8] Online-Terminbuchungen an. Der geringe Verbreitungsgrad erklärt sich durch den Aufwand, welchen das Pflegepersonal für die manuelle Synchronisation der Termine aufbringen muss: Bucht ein Patient über das OTMS einen Termin, werden automatisch alle relevanten Daten per E-Mail an das Pflegepersonal übermittelt. Jeden Morgen vor Öffnung der Ambulanz werden alle online gebuchten Termine in das KIS AGFA ORBIS übgetragen: Hierfür werden, mangels eines zweiten Bildschirms am Arbeitsplatz, alle eingegangenen E-Mails ausgedruckt und händisch als Termine in AGFA ORBIS angelegt. Durch das manuelle Erstellen der Termine kann der Patientenbezug in AGFA ORBIS direkt hergestellt und, falls vorhanden, auch der Fall verknüpft werden.

Online buchbar sind nur ausgewählte Termintypen[9]: *Dysplasie-Sprechstunde*, *Inkontinenz-Sprechstunde*, *Endometriose-Sprechstunde* und *Anmeldung zur Geburt (bei der Hebamme)*. Die Termintypen wurden dahingehend ausgewählt, dass der zeitliche Aufwand sehr gut abschätzbar ist und es meist Folgetermine sind, was den Patienten die Online-Terminbuchung erleichtert.

Zu jedem Termintyp sind feste wöchentliche Terminslots zur Online-Buchung freigegeben. Diese lassen sich ausschließlich über das OTMS buchen, sodass eine Terminüberschneidung trotz fehlender Echtzeitsynchronisation zwischen OTMS und KIS ausgeschlossen werden kann. Sind die Termine für die nächsten Wochen online ausgebucht, kann das Pflegepersonal bei Bedarf weitere Online-Termine freischalten, welche dann im KIS manuell geblockt werden müssen.

3.2 Soll-Zustand

Um einen erfolgreichen und großflächigen Einsatz eines OTMS am SKK zu ermöglichen, muss das OTMS vollständig in die bestehende IT-Infrastruktur integriert werden. Termine und Patientendaten müssen möglichst in Echtzeit zwischen dem OTMS und dem KIS im SKK integriert und synchronisiert werden. Mit der Echtzeitsynchronisation wäre es möglich, alle Termine eines Termintyps sowohl online, als auch per Telefon und vor Ort zur Buchung bereitzustellen. Aufgrund der hohen Datenschutzanforderungen muss die Integration und Synchronisation innerhalb des SKK ablaufen: Es dürfen keine

[8] `http://www.klinikum-karlsruhe.com/medizin/kinder-und-frauenkliniken/frauenklinik/allgemeine-vorstellung.html`

[9] `https://www.betty24.de/DoctorDetails.aspx?did=146873`, Stand 10.10.2016

patientenbezogenen Daten nach außen übermittelt werden. Der Wartungsaufwand muss aufgrund des heterogenen Softwarebestands im SKK möglichst gering bleiben. Auch bei großen Datenmengen (Patienten und Termine) muss die Schnittstellensoftware performant und stabil funktionieren.

3.3 Anforderungsanalyse

Auf Basis des Sollzustandes lassen sich Anforderungen an die Umgebung, mit welcher die Software später interagiert, ableiten:

- **Schnittstelle des KIS:** Das KIS muss Schnittstellen zum Im- und Export von Termindaten und Patientendaten bereitstellen. Die Daten müssen mindestens semi-strukturiert und damit automatisiert lesbar sein.
- **Schnittstelle des OTMS:** Das OTMS muss Schnittstellen zum Im- und Export von Termindaten und Patientendaten bereitstellen. Die Daten müssen mindestens semi-strukturiert und damit automatisiert lesbar sein.

Es lassen sich auch Anforderungen an das Softwareprodukt ableiten:

- **Wartung:** Der Wartungsaufwand muss so gering wie möglich sein um eine zusätzliche Arbeitsbelastung zu vermeiden.
- **Datenschutz:** Aufgrund der Datenschutzanforderungen dürfen keine patientenbezogenen Daten das SKK verlassen. Die Integration der Daten muss also innerhalb des SKK stattfinden. Kontakt nach außen darf ausschließlich per Pull-Request (die Softwarelösung baut in der Rolle des Clients eine Verbindung nach außen auf) und über den zentralen Proxy-Server des SKK stattfinden.
- **Skalierbarkeit:** Die Synchronisationslösung muss skalierbar sein und auch bei großen Datenbeständen performant und zuverlässig funktionieren.
- **Datenformate:** Die Schnittstellenlösung muss mit verschiedenen Datenformaten und -formen umgehen können. Es muss möglich sein, zusätzliche Formate zu implementieren.

Darüber hinaus soll die Software gewisse Qualitätsansprüche erfüllen. In der Literatur gibt es viele Qualitätsmerkmale. Im Folgenden wird auf die Qualitätsmerkmale nach ISO/IEC 9126[10] [SWB08] eingegangen, welche - obwohl die Norm mittlerweile von ISO/IEC 25000[11] ersetzt wurde - auch heute sowohl in der Literatur, als auch in der Praxis eine große Rolle spielt.

- **Funktionalität (functionality):** Das Softwareprodukt soll alle benötigten Anforderungen angemessen und genau erfüllen. Dies schließt auch eine gewisse Flexibilität der Software mit ein. Zudem soll es die Daten vor unautorisiertem Zugriff schützen und Standards und Konventionen (hier z.B. datenschutzrechtliche Aspekte) einhalten.

- **Zuverlässigkeit (reliability):** Das Softwareprodukt erfüllt eine gewisse Reife und Fehlertoleranz. Das bedeutet, dass es bei Fehlern oder Fehlbedienung der Softwareschnittstellen nicht direkt versagt. Zudem darf bei Fehlern kein Datenverlust auftreten.

- **Benutzbarkeit (usability):** Das Softwareprodukt soll für den Benutzer verständlich und leicht bedienbar sein. Die Einlernzeit soll für Benutzer wie Administratoren in einem angemessenen Rahmen bleiben. Die Software soll für den Anwender *attraktiv* sein.

- **Effizienz (efficiency):** Das Softwareprodukt erfüllt unter gewissen Bedingungen ein gewisses Leistungsniveau bezogen auf die eingesetzten Ressourcen. Dabei können zeit- oder ressourcenbezogene Werte erfasst und verglichen werden.

- **Wartbarkeit (maintainability):** Das Softwareprodukt soll änderungsfähig sein: Es soll erweiterbar, aktualisierbar, analysierbar und testbar sein. Anpassungen sollen zu jeder Zeit durchführbt werden können.

- **Portabilität (portability):** Das Softwareprodukt soll auch für veränderte oder neue Umgebungen anpassbar und in diesen stabil betreibbar sein. Umgebungen können hierbei die Hardware, auf der die Software läuft oder auch Software, mit der es interagiert, sein.

Die meisten Qualitätsmerkmale lassen sich nur schwer, beziehungsweise unvollständig messen [SWB08], weswegen im Kapitel Evaluation vor allem Bezug auf die Effizienz und Leistung des Systems genommen wird. Andere Aspekte der Qualitätsmerkmale wurden während des Entwicklungsprozesses beachtet, werden in dieser Arbeit aber nur indirekt angesprochen.

[10] `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=16722`

[11] `http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=64764`

4 Umsetzung

In diesem Kapitel werden die Schnittstellen des KIS und des OTMS vorgestellt. Anschließend wird der konzeptuelle Aufbau des Softwareprodukts vorgestellt und letztendlich die technische Implementierung beschrieben.

4.1 Schnittstellenbeschreibung

4.1.1 Schnittstelle des OTMS Betty24

Betty24 stellt Termin- und Patientendaten über eine RESTful API bereit. In der Schnittstellenbeschreibung [Sae14] werden alle verfügbaren APIs ausführlich erläutert und teils grafisch verdeutlicht. Die Schnittstelle kann mittels Content Negotiation (vgl. [FR14b]) Daten als JSON oder XML ausgeben. Für die weitere Arbeit sind folgende Schnittstellen relevant:

Tabelle 1: Relevante APIs des OTMS Betty24

Methode	**Pfad**	**Beschreibung**
POST	/account/register	Generierung des {Authentication-Tokens}, welcher im weiteren Verlauf für alle Anfragen an die API mitgeschickt werden muss.
GET	/calendars	Gibt alle Kalender aus, die mit den Zugangsdaten verknüpft sind. Ein Kalender entspricht einem behandelnden Arzt.
GET	/calendars/{id}/appointment_updates/{updated_since}	Gibt alle neuen oder geänderten Termine des Kalenders mit der ID {id} aus. Der Referenzzeitpunkt, ab dem alle neuen oder geänderten Termine ausgegeben werden sollen, ist frei wählbar und wird als UTC-Zeitstempel (vgl. [KN02]) übergeben.
POST	/calendars/{id}/block	Blockt die als POST-Daten übertragenen Terminspannen für Online-Buchungen. So können Termine aus dem KIS im OTMS geblockt werden.

4.1.2 Schnittstelle des KIS AGFA ORBIS

AGFA ORBIS bietet eine HL7-Schnittstelle zum Im- und Export von Termin- und Patiententendaten. Hierüber wäre eine vollständige Integration von Termin- und Patiententendaten möglich. Termine könnten, sofern der Wrapper aus dem KIS genug Patienteninformationen erhält um die Zuordnung vornehmen zu können, mit oder ohne Patienten- und Fallbezug angelegt werden. Da das SKK keine Lizenz zur Nutzung der HL7-Schnittstelle hat, kann diese Schnittstelle leider nicht genutzt werden.

Die Alternative ist ein Importformular in AGFA ORBIS selbst, durch welches CSV-Dateien manuell importiert werden können. Unter gewissen Umständen lässt sich dieser Importvorgang auch automatisiert ausführen. Damit ist ein Import der Termine aus dem OTMS möglich, ein Export von Termin- und Patiententendaten allerdings nicht.

Im Rahmen der Bachelorarbeit wird daher die Im- und Exportkomponente über eine Ordnerstruktur simuliert, sodass die zweite Datenquelle für eine Integration vorhanden ist. Abbildung 2 zeigt diese Ordnerstruktur: im Ordner *in* werden zu importierende Termine angelegt, im Ordner *out* Termine, die aus dem KIS kommen und im OTMS angelegt werden müssen.

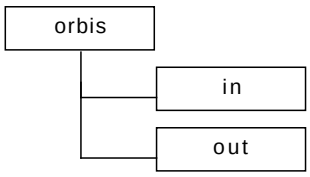

Abbildung 2: Ordnerstruktur, über welche Im- und Export in und aus AGFA ORBIS simuliert werden.

Die Termine liegen in Form von CSV-Dateien vor, wobei jede Datei einem Termin entspricht. Jede Datei liegt im folgenden Schema vor:

```
Datum, Zeit, Dauer, Kalendername, Kategorie, Nachname, Vorname,
    Geburtsdatum, Bemerkung, Typ
```

Alle Daten müssen im Format *dd.mm.yyyy* vorliegen, die Uhrzeiten im Format *hh:mm*. Dauer ist die Termindauer in Minuten. Der Termintyp kann „F“ oder „G“ annehmen, wobei „F“ einem Freitexttermin entspricht und „G“ einem Gruppentermin.

4.2 Konzeptueller Aufbau

Um den Anforderungen aus der Anforderungsanalyse gerecht zu werden, soll die Schnittstellenlösung die Termin- und Patientendatenintegration auf Basis semantischer Technolo-

gien durchführen. Dies ermöglicht die Integration verschiedener Datenformate unabhängig proprietärer Datenstandards (vgl. [MSK11]).

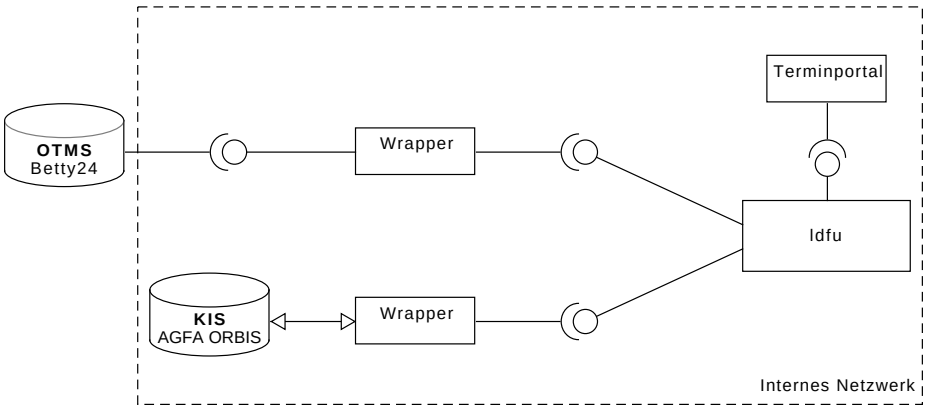

Abbildung 3: Komponentendiagramm des zu entwickelnden Softwareprodukts

Das Softwareprodukt besteht aus vier Komponenten: zwei Wrapper, welche mit den Programmschnittstellen des KIS und OTMS kommunizieren, ldfu, welches die Wrapper regelbasiert aufruft, sowie dem Terminportal, das die integrierten Daten in grafisch aufbereiteter Form darstellt. Die Wrapper wandeln die von KIS und OTMS empfangenen Daten in semantische Daten um und stellen sie mittels einer RESTful Linked Data API zur Verfügung. Hierfür wurde ein eigenes Vokabular entwickelt, welches im nächsten Abschnitt näher beschrieben wird.

4.2.1 Entwicklung des RDF-Vokabulars

Da es noch kein publiziertes RDF-Vokabular für Termin- und Patientendaten im medizinischen Umfeld gibt, wurde ein eigenes Vokabular in den Ontologiesprachen RDF [WLC14], RDFS [GB14] und OWL [HHPS04] entwickelt. Zwar unterstützt der HL7 FHIR-Standard[12] die Repräsentation von Ressourcen in RDF[13]; da sich der Standard allerdings noch im Entwurfsstatus (DSTU2: *Draft Standard for Trial Use*) befindet und die eingesetzte Software in Krankenhäusern für gewöhnlich ältere HL7-Spezifikationen nutzen, findet dieser Standard keine direkte Verwendung. Dennoch stützt sich das selbst entwickelte Vokabular in seiner Terminologie auf den HL7 FHIR DSTU2 v1.02[14].

Die erste Version des Vokabulars wurde mit dem Open-Source-Editor Protégé[15] der Stanford-Universität entwickelt. Alle weiteren Versionen wurden mit Hilfe eines Texteditors direkt in N3 geschrieben.

Das Vokabular lässt sich unter der im Anhang angegebenen Adresse herunterladen.

[12] `https://www.hl7.org/fhir/`
[13] `https://www.hl7.org/fhir/rdf.html`
[14] `http://hl7.org/fhir/directory.html`
[15] `http://protege.stanford.edu/`

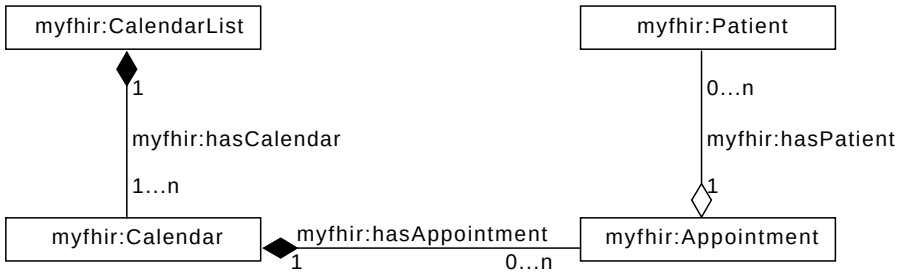

Abbildung 4: Klassendiagramm der myfhir-Ontologie. Der Übersicht wegen wird nur der Klassenname ohne weitere Details dargestellt.

Wie in Abbildung 4 dargestellt, enthält das eigene Vokabular *myfhir* vier Klassen (*owl:Class*). Die Aggregation und Komposition werden im Vokabular als *owl:ObjectProperty* abgebildet. Die *properties* sind in dieser Vereinfachung nicht dargestellt, ergeben sich aber aus dem Vokabular.

4.2.2 Aufbau der Wrapper

Die Wrapper verbinden ldfu mit den eigentlichen Datenquellen für die Integration. Für ldfu stellen die Wrapper eine RESTful Linked Data API bereit, bauen selbst (im Falle des Betty24-Wrappers) aber auch wieder eine Datenverbindung zur Betty24-API auf. Die Wrapper agieren sowohl als Client, als auch als Server. Im Falle des KIS-Wrappers werden durch diesen CSV-Dateien in der in Kapitel 4.1.2 beschriebenen Ordnerstruktur verarbeitet. Wie in Abbildung 5 ersichtlich, findet die Kommunikation zwischen ldfu und den Wrappern ausschließlich im RDF-Format statt, wohingegen die Wrapper auf Textdateien und eine JSON-API zugreifen. Die Konvertierung der Daten zu RDF findet automatisch statt und wird *lifting* bezeichnet (vgl. [HKS+13]): Die Daten werden von semi-strukturierten Daten zu semantisch definierten Daten *gehoben*, erfahren also eine Qualitätssteigerung. Umgekehrt können aber auch semantische Daten im RDF-Format an den Wrapper geschickt werden, welche der Wrapper in der entsprechenden semi-strukturierten Form an das OTMS oder KIS weiterleitet.

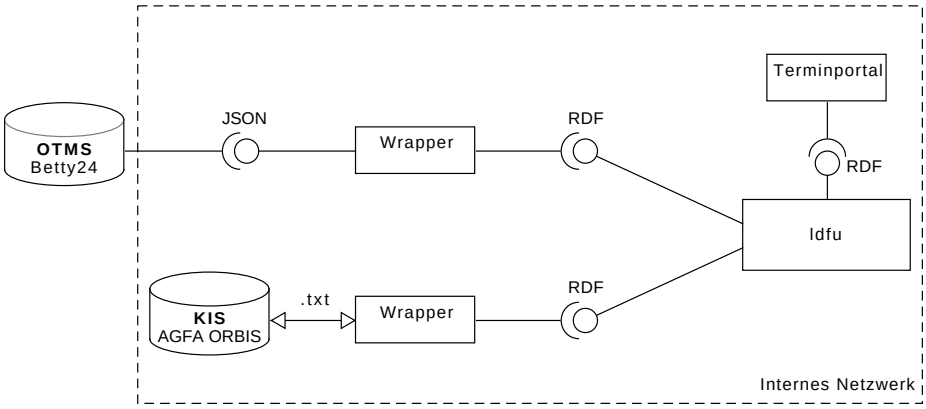

Abbildung 5: Die Wrapper stellen vereinheitlichte RESTful Linked Data APIs bereit.

4.2.3 Linked Data-Fu (ldfu)

Linked Data-Fu (ldfu) ist ein Datenverarbeitungssystem, welches regelbasierte Programme (*linked programs*) ausführt (vgl. Webseite[16]). *Linked programs* sind in N3-Notation (vgl. [BLC11]) geschriebene Regelsätze. Dabei kann ldfu auf lokale wie entfernte Daten zugreifen, diese aggregieren und Abfragen über sie ausführen. Zudem lassen sich auf Basis bestehender, beziehungsweise aggregierter Daten HTTP-Anfragen triggern.

Im hier vorliegenden Fall verbindet ldfu die Wrapper und nimmt die eigentliche Integration der Daten vor. Dabei werden zuerst die verfügbaren Kalender der Wrapper abgefragt und anschließend deren Termine. Auf Basis der nun aggregierten Daten werden diese Termine einzeln an das Terminportal und an den jeweils anderen Wrapper geschickt, sodass letztendlich an allen drei Endpunkten die gleichen Termindaten vorhanden sind.

4.2.4 Terminportal

Das Terminportal dient in erster Linie der visuellen Aufbereitung der integrierten Termine. Da es durch ldfu alle Termine erhält und sie in einer relationalen Datenbank, sowie auch in Form von *turtle*-Dateien abspeichert, ist es sehr gut geeignet um Datenanalysen auszuführen. Es zeigt auch, wie gering der Aufwand ist zusätzliche Systeme an den Integrationsprozess anzuschließen und dass die Programmiersprache der Komponenten aufgrund der losen Kopplung durch REST-APIs keine Rolle spielt. Das Terminportal stellt eine Webseite bereit, auf welcher in einer Kalenderansicht die integrierten Termine angezeigt werden. Zusätzlich werden unaufwendige statistische Analysen durchgeführt.

[16] `http://linked-data-fu.github.io/`

4.3 Technische Implementierung

Im folgenden Abschnitt ist die Implementierung des Wrappers beschrieben. Anschließend folgen Informationen über die Einbettung von ldfu in das Softwareprodukt. Letztendlich werden die Anforderungen zum Betrieb des Softwareproduktes beschrieben.

4.3.1 Wrapper

Die Wrapper übernehmen eine sehr wichtige Rolle im Integrationsprozess. Aus diesem Grund müssen sie sehr performant und robust sein, weshalb die Wahl der Programmiersprache auf Java fiel. Java ist eine sehr etablierte Sprache mit vielen Bibliotheken und Frameworks hoher Qualität, was auch eine kurze Entwicklungszeit ermöglicht.

Die Basis der Wrapper ist das Framework Jersey[17], welches die Entwicklung von RESTful APIs in Java vereinfacht und dabei vor allem das Routing und Mapping der URIs auf den entsprechenden Programmcode vornimmt. Es unterstützt auch das Serialisieren und Parsen diverser Datenformate, sodass hierüber auch *Content Negotiation* vgl. [FR14b] implementiert werden konnte. Das Datenformat der Wrapperantwort hängt somit vom *Accept*-Header der Anfrage ab (s. Tabelle 2).

Tabelle 2: Content Negotiation der Wrapper

Accept-Header der Anfrage	**Format der RDF-Daten der Antwort**
application/rdf+xml	RDF/XML
text/plain	RDF/XML
text/turtle	TURTLE
text/rdf+n3	N3
application/n-triples	N-TRIPLE

Zum Parsen und Serialisieren von RDF in Java wird die RDF-API von Apache Jena[18] genutzt. Apache Jena ist eines der meistgenutzten RDF-Frameworks[19] in Java und zeichnet sich als modulares, freies Open-Source-Framework mit geringen Anforderungen aus.

Als HTTP(S)-Client für ausgehende Verbindungen zum OTMS kommt Unirest[20] zum Einsatz.

[17] `https://jersey.java.net/`
[18] `https://jena.apache.org/`
[19] `http://db-engines.com/en/ranking/rdf+store`
[20] `http://unirest.io/`

Um den Build- und Deploy-Prozess zu automatisieren wird Apache Maven[21] verwendet. Maven automatisiert alle Schritte vom Auflösen der Abhängigkeiten bis zum Kompilieren und Installieren der Applikation auf dem Apache Tomcat[22]-Server.

4.3.2 Linked Data-Fu

ldfu bietet mehrere Arten zum Ausführen der in N3-Notation geschriebenen *linked programs*:

- **Konsolenaufruf**: ldfu wird je nach Plattform über ein Shell-/Batchscript in regelmäßigen Abständen aufgerufen (z.B. als Cronjob unter Linux/Cygwin[23]).
- **Eingebettet in Java-Projekt**: ldfu wird als Java-Bibliothek geladen und über die Java-API verwendet.
- **Standalone-Modus**: ldfu startet sich im Servermodus und kann über eine RESTful API gesteuert werden.

Aufgrund der zu erwartenden Änderungen in der Java-API und im Standalone-Modus wurde für diese Arbeit die Konsolen-Variante bevorzugt. In regelmäßigen Abständen wird ldfu mit dem *linked program* myfhir.n3 gestartet:

```
$ bin/ldfu.sh -p myfhir.n3 rulesets/rdf.n3 rulesets/rdfs.n3
```

Das *linked program* ist dabei wie folgt aufgebaut:
Der folgende Quelltext schickt Anfragen an beide *CalendarList*-Ressourcen der Wrapper. Diese enthalten ihrerseits wieder Verlinkungen auf *Calendar*-Ressourcen.

```
{ [] http:mthd httpm:GET ;
        http:requestURI <http://localhost:8080/betty2rdf/rest/
            CalendarList> . }

{ [] http:mthd httpm:GET ;
        http:requestURI <http://localhost:8080/rdf2csv/rest/
            CalendarList> . }
```

Durch folgende Regel wird jede *Calendar*-Resource der beiden Wrapper abgerufen und die enthaltenen Kalendereinträge nachgeladen.

```
{ ?x myfhir:hasCalendar ?cal . }
=>
{ [] http:mthd httpm:GET ;
http:requestURI ?cal . } .
```

[21] https://maven.apache.org/
[22] http://tomcat.apache.org/
[23] https://www.cygwin.com/

Werden in den abgerufenen Daten bestimmte Bedingungen erfüllt, greifen nun weitere Regeln um die Termine an die entsprechenden API-Endpunkte zu schicken. Nachfolgende Regel zeigt dies beispielhaft für Termine, die aus dem KIS an das OTMS übertragen werden:

```
{
        ?app rdf:type myfhir:Appointment ;
        myfhir:orbisIdentifier ?orbis ;
        myfhir:calendarId      ?calId ;
        myfhir:end             ?end ;
        myfhir:start           ?start .
} => {
[] http:mthd httpm:POST ;
http:requestURI <http://localhost:8080/betty2rdf/rest/Calendar
   /{?calId}> ;
        http:body {
                ?app rdf:type myfhir:Appointment ;
                myfhir:orbisIdentifier    ?orbis ;
                myfhir:calendarId         ?calId ;
                myfhir:end                ?end ;
                myfhir:start              ?start .
        } .
} .
```

Die Regel selektiert aus den bisher aggregierten Daten alle Termine, welche die Properties *ordisIdentifier, calendarId, end* und *start* besitzen und schickt diese per POST-Anfrage an den Wrapper des OTMS.

4.3.3 Terminportal

Das Terminportal ist eine optionale Komponente, welche zwei RESTful APIs bereitstellt. Eine API dient dem Speichern, Aktualisieren und Löschen von Termindaten und wird von ldfu genutzt, die andere API gibt die gespeicherten Termine im JSON-Format aus um den Kalender in der Browseransicht des Benutzers zu befüllen. Es basiert auf der Scriptsprache PHP[24], sowie HTML, Javascript und CSS und nutzt EASYRDF[25] zum Parsen und Serialisieren der RDF-Daten. Das Terminportal hält alle Termindaten in einer MySQL[26]-Datenbank, sowie als *turtle*-Dateien vor. Diese doppelte Speicherung der Daten ermöglicht hohe Flexibilität für weitere Datenverarbeitungsprozesse.

[24] `https://php.net`
[25] `http://www.easyrdf.org/`
[26] `https://mariadb.org/`

4.3.4 Anforderungen im Betrieb

Trotz des dreiteiligen Aufbaus sind die Anforderungen zum Betrieb des Softwareproduktes sehr gering. Im Zeitalter der Virtualisierung ist es für Institutionen und Firmen sehr einfach und kostengünstig eine virtuelle Maschine bereitzustellen. Das Softwareprodukt wurde im SKK auf einer virtuellen Maschine mit folgenden Daten erfolgreich ausgeführt und getestet:

Tabelle 3: Spezifikation der VM im SKK

Virtualisierungsform	VMware[27]
Plattform	Windows Server 2016
CPU	2 Kerne (Intel XEON E7-4870)
RAM	4 GB DDR3-RAM
Internetzugriff	aus DMZ über Firewall mit SSL-Proxy

Zum Betrieb des Softwareproduktes (Tomcat-Server inkl. Applikationen und ldfu) werden folgende Anforderungen benötigt[28]:

Tabelle 4: Minimalanforderungen für den Betrieb

Plattform	Windows oder Linux
CPU	1 Kern
RAM	mind. 1 GB RAM, 2 GB empfohlen
Schnittstellen	vgl. Kapitel Anforderungsanalyse
Internetzugriff	Internetzugriff auf API des OTMS

[28] Disclaimer: Die Minimalanforderungen basieren auf Erfahrungsberichten und eigener Erfahrung Je nach verwendeter Hardware, ggf. Virtualisierungstechnologie und Software können die Werte abweichen.

5 Evaluation

In Kapitel 3.3 wurden einige Anforderungen an den zu entwickelnden Prototypen ausgearbeitet. Eine Auswahl dieser Anforderungen sollen nun in diesem Kapitel evaluiert werden. Hierfür wird zunächst das Softwareprodukt auf Validität und Skalierbarkeit überprüft. Abschließend wird evaluiert, ob eine Echtzeitsynchronisation möglich ist.

5.1 Validität

Testbeschreibung
Um die Validität des Softwareproduktes sicherzustellen werden 30 OTMS-Termine sowie 30 KIS-Termine aus jeweils 2 Kalendern integriert. Je 2 Termine aus OTMS und KIS überschneiden sich.

Erwartetes Ergebnis
Als Ergebnis sollen die Termine des KIS im OTMS geblockt sein und die OTMS-Termine im KIS vorhanden. Alle 60 Termine müssen im Portal einsehbar sein. Dort müssen auch beide Terminüberschneidungen rot markiert sein.

Tatsächliches Ergebnis
Wie in Abbildung 8 im Kapitel Testdokumentation ersichtlich, wurde die Integration der Daten korrekt durchgeführt: die Datenbank enthält 60 Termine, die Überschneidungen sind rot markiert und alle weiteren Daten (Name, Geburtstdatum, ...) sind korrekt übertragen worden. Alle Termine des OTMS wurden zudem als CSV-Dateien für das KIS angelegt, ebenso wurden die Termine aus dem KIS im OTMS geblockt. Die weiteren Ergebnisdaten (SQL-Dump der Datenbank, CSV-Dateien für den Import in das KIS, ...) stehen zusammen mit den Ursprungsdaten im Anhang bereit.

5.2 Skalierbarkeit

Testbeschreibung
Um die Skalierbarkeit des Softwareproduktes sicherzustellen, wird ein Lasttest durchgeführt. Unter fest definierten Umweltbedingungen werden 10, 100 und 500 Termine integriert. Anschließend werden die Laufzeiten verglichen. Da es nicht möglich war, automatisch eine hohe Anzahl Termine im OTMS zu generieren, dient hier das KIS als einzige Datenquelle. Die Daten sind dennoch sehr aussagekräftig und geben sogar eher einen pessimistischeren Wert an, da die Latenz zum OTMS höher ist als zur Ordnerstruktur im internen Netzwerk. Tabelle 5 gibt die Spezifikation der VM an, welche für die Tests genutzt wurde.

Tabelle 5: Spezifikation der Test-VM

Virtualisierungsform	KVM/Qemu[29] (Proxmox[30])
Plattform	Linux (Ubuntu 16.04[31])
CPU	2 Kerne (Intel Xeon CPU E3-1245V2[32] inkl. HT (4 Threads))
RAM	6 GB DDR3-RAM

Tatsächliches Ergebnis

Die Laufzeiten des Lasttests können Tabelle 6 entnommen werden.

Tabelle 6: Laufzeiten der Integration bei verschiedener Terminanzahl

Anzahl Termine	**10**	**50**	**100**	**500**
Gesamtlaufzeit in ms	4.298	12.440	26.528	117.579
Laufzeit pro Termin in ms	429,8	248,8	265,28	235,16

Auffallend sind insbesondere zwei Dinge:

(i) Bei vielen Terminen sinkt die Laufzeit pro Termin rasch auf ca. 250 ms.

(ii) Die Laufzeit bleibt auch bei der Integration von 500 Terminen bei ca. 250 ms pro Termin.

Abbildung 6 verdeutlicht diese beiden Erkenntnisse.

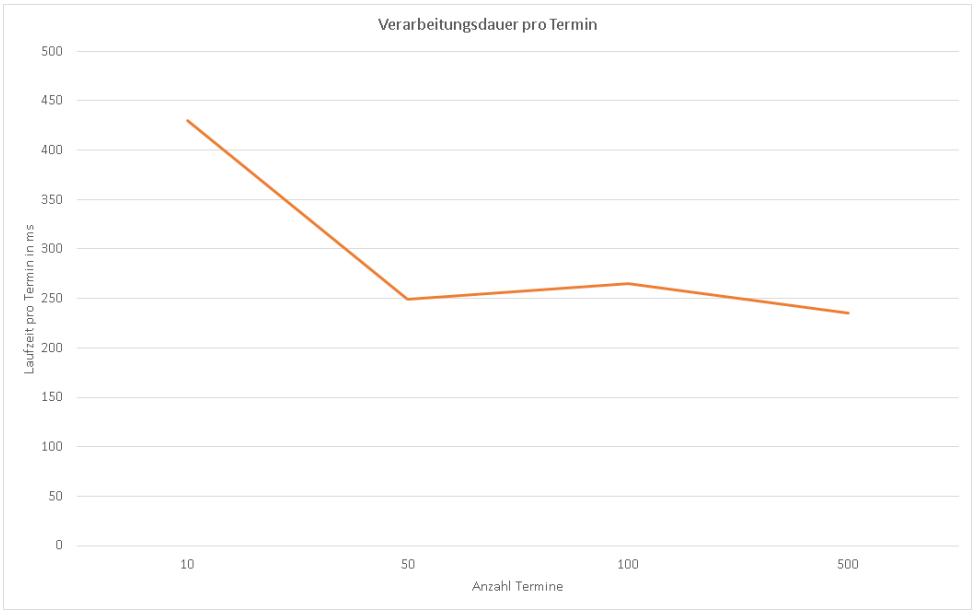

Abbildung 6: Laufzeiten der Integration bei verschiedener Terminanzahl

Der gute Skalierungsfaktor in Höhe von 1,83 beim Vergleich der Laufzeiten pro Termin von 10 und 500 Terminen lässt sich durch mehrere Gründe erklären:

- **HTTP Keep-Alive** (vgl. [FR14a]): Der HTTP Keep-Alive-Header ermöglicht es, bereits offene Verbindungen für eine bestimmte Zeitspanne aufrechtzuerhalten. Bei weiteren HTTP-Requests an den gleichen Server kann diese bereits offene Verbindung dann erneut genutzt werden, wodurch ein mehrfacher Verbindungsaufbau entfällt. Dies ermöglicht mehr Anfragen pro Zeiteinheit und erklärt zum Teil den Skalierungsfaktor.

- **Multithreaded ldfu**: Da ldfu *multithreaded* ist, können große Datenmengen parallel auf mehreren Prozessorkernen verarbeitet werden (vgl. [SSHS13]). Der zusätzliche Aufwand zum Aufspalten der Daten auf die verschiedenen Prozessorkerne ist bei großen Datenmengen vernachlässigbar, wodurch ebenfalls eine bessere Performance erreicht werden kann.

- **Multithreaded HTTP-Client in ldfu**: Da die ldfu-Komponente, welche ausgehende HTTP-Verbindungen ausführt, *multithreaded* ist (vgl. [HKS+13]), können Daten parallel abgerufen und verteilt werden. Auch dadurch wird, insbesondere bei vielen beteiligten Servern, die Performance deutlich gesteigert. Im vorliegenden Fall wurde pro Zielserver zwar nur eine Verbindung aufgebaut, da aber mehrere Server an der Integration beteiligt sind, wurden dennoch parallele Verbindungen genutzt.

Die erstaunlich hohe Laufzeit pro Termin - auch bei einer sehr hohen Terminanzahl - lässt sich durch die Latenz zum OTMS erklären. Laut den ldfu-Logs beträgt die Laufzeit durchschnittlich ca. 230 ms pro HTTP-Anfrage zum OTMS. Diese Laufzeit enthält die Latenz zum Wrapper, die Ausführung des Wrappers und die Latenz vom Wrapper zur Schnittstelle des OTMS. Um reine Performancedaten für ldfu zu erhalten wurde der gleiche Testaufbau auch rein lokal durchgeführt. Dabei entfällt die Ausführungszeit des Wrappers und die Latenz vom Wrapper zum OTMS. Da die Wrapper auf der gleichen VM wie ldfu installiert sind, ist die Latenz vernachlässigbar. Tabelle 7 zeigt die Ergebnisse des abgewandelten Tests. Abbildung 7 verdeutlicht die Zahlen.

Tabelle 7: Laufzeiten der Integration bei lokaler Ausführung

Anzahl Termine	**10**	**50**	**100**	**500**
Gesamtlaufzeit in ms	222	478	720	2.380
Laufzeit pro Termin in ms	22,2	9,56	7,2	4,76

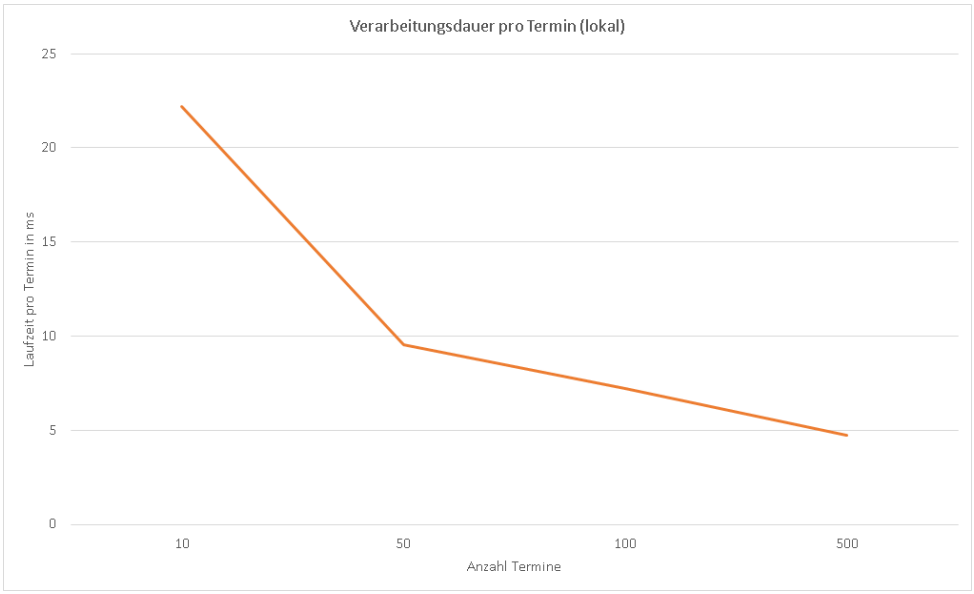

Abbildung 7: Laufzeiten der Integration bei unterschiedlicher Terminanzahl (lokal)

Auch für diesen Test sind die verwendeten Daten im Anhang verlinkt.

5.3 Echtzeitsynchronisation

Auf Basis des Lasttests aus vorherigem Kapitel ist ersichtlich, wie performant die Integration mit ldfu ist. Da im vorliegenden Fall nur neue Termine integriert werden, ist es problemlos möglich die Integration jede Minute oder kürzer durchzuführen. Die Wahrscheinlichkeiten für Überschneidungen bei Terminbuchungen lassen sich somit minimieren. Eine wirkliche Echtzeitintegration ist allerdings nicht möglich. Zwar unterstützt ldfu auch höhere Frequenzen (20-30 Durchläufe pro Sekunde sind möglich, vgl. [KKS+14]), die Limitierungen liegen dann aber bei den Latenzen zum OTMS. Zudem würde eine derartige Menge an HTTPS-Requests die API des OTMS unnötig belasten und unter Umständen sogar als Angriff auf die Serverinfrastruktur klassifiziert und geblockt werden. Darüberhinaus wäre ein eventgetriebener Ansatz möglich: hier müsste bei jeder Terminbuchung die Ausführung der Integration angestoßen werden. Dieser Ansatz ist jedoch allein aus Sicherheitsgründen nicht möglich, da bei der Buchung eines Termins im OTMS eine Verbindung zu ldfu hergestellt werden müsste. Da die Integration innerhalb des Intranets des SKK stattfinden muss ist eine solche Verbindung von außen aus Sicherheitsgründen nicht möglich.

6 Fazit

Ziel dieser Arbeit ist es zu evaluieren, inwiefern die Schnittstellenproblematik zwischen KIS und OTMS gelöst werden kann. Hierfür wurde der Ansatz einer ontologiebasierten Datenintegration erarbeitet und anhand eines Prototypen durchgeführt. Abschließend wurde evaluiert, ob der Prototyp die zuvor erarbeiteten Anforderungen erfüllt.

Die Arbeit hat gezeigt, dass die ontologiebasierte Integration mit ldfu performant und skalierbar ist. Bei einer rein lokalen Integration konnten extrem gute Werte von wenigen Millisekunden pro Termin erreicht werden. Beim Test unter Realbedingungen ist die Laufzeit pro Termin abhängig von der langsamsten API-Anfrage. Dennoch konnten auch hier sehr gute Werte erzielt werden.

Eine Echtzeitintegration ist aufgrund des Pull-Ansatzes nicht vollständig möglich. Dank der kurzen Laufzeit des Integrationsprozesses lässt sich die Frequenz der Integration so erhöhen, dass der Unterschied zur Echtzeitintegration kaum ins Gewicht fällt. Ein anderer Ansatz, mit dem eventuell eine Echtzeitintegration möglich wäre, scheidet aus Sicherheitsgründen aus.

Sollte wider Erwarten doch eine Terminkollision auftreten, so ist diese im Terminportal rot markiert und damit direkt erkennbar.

Da die Datenintegration innerhalb der Infrastruktur des SKK stattfinden kann, werden die Datenschutzanforderungen erfüllt: Keine Patientendaten verlassen das interne Netzwerk. Zudem ermöglicht der datengetriebene Poll-Ansatz eine Integration ohne eingehende Datenverbindung in einem abgeschotteten Netzwerkbereich, sodass sicherheitstechnisch keine Bedenken auftreten.

Durch die Nutzung von ldfu kann der Integrationsprozess sehr einfach erweitert oder angepasst werden: Ein einfaches Austauschen der Regeldatei genügt. Auf Basis der aktuell bestehenden Regeln ist eine Erweiterung auch ohne Vorwissen möglich. Auch weitere Datentypen und Wrapper können ergänzt werden.

Durch die periodische Ausführung hält sich der Wartungsaufwand in Grenzen. Da alle benötigten Dienste beim Start des Betriebssystem automatisch starten, kann die Integration bei Stromausfall oder Defekten nach Wiederherstellung des Normalzustanden automatisch fortgeführt werden. Ein Datenverlust ist nicht zu erwarten.

Problematisch hingegen ist die Schnittstelle des KIS. Da es keine Möglichkeit gibt Termin- oder Patientendaten aus dem KIS zu exportieren, musste die Exportfunktion simuliert werden um das gewünschte Ergebnis zu erzielen.

Mit dieser Einschränkung kann der Soll-Zustand erreicht werden.

6.1 Ausblick

Dieses Kapitel soll abschließend einige Ansätze umschreiben, auf deren Grundlage weitere Forschungsprojekte oder Abschlussarbeiten durchgeführt werden können:

Auf Basis der integrierten Daten im Terminportal könnten weitere statistische Analysen realisiert werden. Neben der reinen Vergleichsfunktion der Ergebnisse könnten diese auch zur Prozessverbesserung genutzt werden.

Denkbar wäre auch, den Wrapper des KIS mit einem HL7-Wrapper zu ersetzen. Dadurch wäre - bei Erwerb der Lizenz zur Nutzung der HL7-Schnittstelle - eine noch tiefere Integration in die Infrastruktur des SKK und auch der Patienten- und Terminexport möglich. Somit könnte die Schnittstelle bidirektional Daten integrieren und synchronisieren.

Bei Terminkollisionen könnte ein Algorithmus implementiert werden, welcher selbstständig versucht diese Kollision zu beheben. Dies wäre zum Beispiel durch den Abgleich der Buchungszeitpunkte möglich. Dem Patienten des später gebuchten Termins könnte automatisch abgesagt und alternative Terminvorschläge unterbreitet werden.

Sehr interessant wären auch Prognosen auf Basis historischer oder zusätzlich integrierter Daten. Für Patienten nach einer OP könnten, gestützt auf die allgemeinen Erfahrungen und der individuellen Patientenakte, bei einem hohen Risiko für Komplikationen direkt Folgetermine freigehalten werden.

Auch patientenübergreifend könnten Prognosen erstellt werden: Beispielsweise könnten Veranstaltungsdaten oder Wetterdaten in Verbindung mit historischen Daten zur Vorhersage des Patientenaufkommens in der Folgewoche genutzt werden.

Diese Liste ist natürlich nicht abschließend und zeigt vermutlich auch nur einen kleinen Teil der möglichen Aspekte auf. Sie verdeutlicht aber, wie viel Potenzial in diesem Themengebiet noch schlummert und wie interessant weitere Arbeiten gestaltet werden können.

A Anhang

A.1 Testergebnis Funktionstest

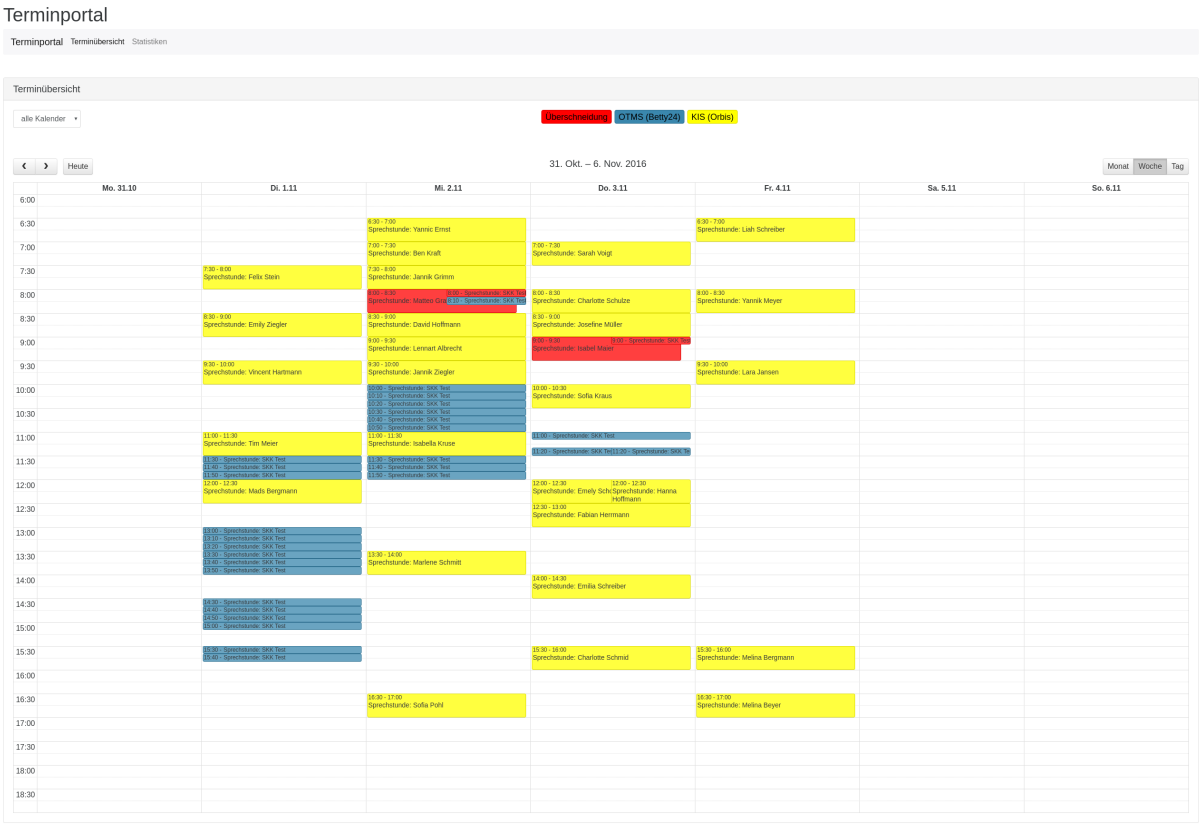

Abbildung 8: Grafische Ergebnisdokumentation des Funktionstests (vgl. Kapitel 5.1)

A.2 RDF-Vokabular

Das verwendete RDF-Vokabular steht unter folgender Adresse zum Download bereit:
`https://ba.tdillig.de/myfhir#`

A.3 Testdokumentation

Die Ausgangsdaten für alle Tests und die Testergebnisse der Evaluation sind unter folgendem Link abrufbar:
`https://ba.tdillig.de/test`

A.4 Quellcode

Der Quellcode der Wrapper, des Terminportals und die ldfu-Regeldatei sind ebenfalls online verfügbar:
`https://ba.tdillig.de/`

A.5 Demo Terminportal

Eine Demo des Terminportals ist unter folgender Adresse verfügbar:
`https://ba.tdillig.de/portal/`

Literatur

[BHBL09] Bizer, Christian, Tom Heath und Tim Berners-Lee: *Linked Data-the story so far*. International Journal on Semantic Web and Information Systems, 5(3):1–22, 2009.

[BHL+14] Busse, Johannes, Bernhard Humm, Christoph Lübbert, Frank Moelter, Anatol Reibold, Matthias Rewald, Veronika Schlüter, Bernhard Seiler, Erwin Tegtmeier und Thomas Zeh: *Was bedeutet eigentlich Ontologie?* Informatik-Spektrum, 37(4):286–297, 2014.

[BLC11] Berners-Lee, Tim und Dan Connolly: *Notation3 (N3): A readable RDF syntax*. W3C Team Submission, W3C, März 2011. https://www.w3.org/TeamSubmission/n3/.

[FR14a] Fielding, R. und J. Reschke: *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. RFC 7230, RFC Editor, June 2014. `http://www.rfc-editor.org/rfc/rfc7230.txt`.

[FR14b] Fielding, R. und J. Reschke: *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231, June 2014. `http://www.rfc-editor.org/rfc/rfc7231.txt`.

[GB14] Guha, Ramanathan und Dan Brickley: *RDF Schema 1.1*. W3C Recommendation, W3C, Februar 2014. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[HHPS04] Hayes, Patrick, Ian Horrocks und Peter Patel-Schneider: *OWL Web Ontology Language Semantics and Abstract Syntax*. W3C Recommendation, W3C, Februar 2004. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/.

[HKRS08] Hitzler, Pascal, Markus Krötzsch, Sebastian Rudolph und York Sure (Herausgeber): *Semantic Web : Grundlagen*. eXamen.pressSpringerLink : Bücher. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[HKS+13] Harth, Andreas, Craig Knoblock, Steffen Stadtmüller, Rudi Studer und Pedro Szekely: *On-the-fly Integration of Static and Dynamic Sources*. In: *Proceedings of the ISWC 2013 workshop on Consuming Linked Data*. CEUR-WS, 2013.

[KKS+14] Keppmann, Felix Leif, Tobias Käfer, Steffen Stadtmüller, René Schubotz und Andreas Harth: *High Performance Linked Data Processing for Virtual Reality Environments*. Poster & Demo Proceedings of the 13th International Semantic Web Conference (ISWC), Oktober 2014.

[KN02] Klyne, G. und C. Newman: *Date and Time on the Internet: Timestamps*. RFC 3339, July 2002.

[KWO+14] Kämpgen, Benedikt, Tobias Weller, Sean O'Riain, Craig Weber und Andreas Harth: *The Semantic Web: Trends and Challenges: 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, Kapitel Accepting the XBRL Challenge with Linked Data for Financial Data Integration, Seiten 595–610. Springer International Publishing, Cham, 2014.

[Kö82] Köhler, Claus O.: *Ziele, Aufgaben, Realisation eines Krankenhausinformationssystems*. Medizinische Informatik und Statistik ; 36. Springer, Berlin [u.a.], 1982.

[MSK11] Martini, Daniel, Mario Schmitz und Martin Kunisch: *Datenintegration zwischen Standards in der Landwirtschaft auf Basis semantischer Technologien*. In: *Qualität und Effizienz durch informationsgestützte Landwirtschaft, Fokus: Moderne Weinwirtschaft. Referate der 31. GIL-Jahrestagung, 24. - 25. Februar 2011 in Oppenheim, Germany*, Seiten 133–136, 2011.

[OHG+15] Ohlmeier, C, F Hoffmann, K Giersiepen, H Rothgang, R Mikolajczyk, H-J Appelrat, A Elsässer und E Garbe: *Verknüpfung von Routinedaten der Gesetzlichen Krankenversicherung mit Daten eines Krankenhausinformationssystems: Machbar, aber auch „nützlich "?* Das Gesundheitswesen, 77(02):e8–e14, 2015.

[Pri15] Prinz, Simon: *Online-Terminmanagementsysteme als Optimierungs- und Marketinginstrument für Arztpraxen in Deutschland*. Masterthesis, KIT - Kalsruher Institut für Technologie, April 2015.

[RA13] Richardson, Leonard und Mike Amundsen: *RESTful Web APIs : [services for a changing world]*. OReilly, Beijing, 1. ed. Auflage, 2013.

[Sae14] Saewa UG (haftungsbeschraenkt): *Betty24 Integration - Technical Integration Document*, 1.0 Auflage, jul 2014.

[SSHS13] Stadtmüller, Steffen, Sebastian Speiser, Andreas Harth und Rudi Studer: *Data-Fu: A Language and an Interpreter for Interaction with Read/Write Linked Data*. In: *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, Seiten 1225–1236, New York, NY, USA, 2013. ACM.

[SWB08] *Lehrbuch der Software-Technik*, Band [2]: Softwaremanagement. Spektrum, Akad. Verl., Heidelberg, 2. Aufl. Auflage, 2008.

[TKSA12] Taheriyan, Mohsen, Craig A. Knoblock, Pedro Szekely und José Luis Ambite: *Rapidly Integrating Services into the Linked Data Cloud*. In: *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*, ISWC'12, Seiten 559–574, Berlin, Heidelberg, 2012. Springer-Verlag.

[WLC14] Wood, David, Markus Lanthaler und Richard Cyganiak: *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation, W3C, Februar 2014. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.