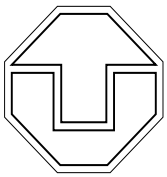

Eine generische Dienstarchitektur für das Gesundheitswesen

Dissertation

---

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von

Dipl.Inf. Thomas Pucklitzsch
geboren am 03.07.1976 in Halle/Saale

am 29. April 2010

Gutachter:

Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Prof. Dr. Olaf Schlimpert

Tag der Verteidigung: 6. Juli 2010

Inhaltsverzeichnis

1 Einführung

1.1 Vernetzung der Gesellschaft

Das Internet hat unsere Gesellschaft bereits mehr verändert als jede andere Erfindung des zwanzigsten Jahrhunderts. Dabei sind Netzwerke, ohne dass diese Tatsache den Menschen vielleicht immer bewusst ist, nicht erst seit Erfindung des Internets Bestandteil unserer Gesellschaft. Denken wir beispielsweise an soziale Netzwerke, die sich jeder Mensch in seinem Umfeld aufbaut. Wie im Kapitel über das „Small World Phänomen“ auf Seite 89 beschrieben wird, verbinden diese Netzwerke die Menschen enger miteinander, als man sich gewöhnlich vorstellen kann. Sie dienen dazu, uns abzusichern und im Notfall auf die Hilfe anderer zurückzugreifen. Sie sind aber auch wichtig für die Beschaffung von Informationen. Menschen mit ähnlichen oder gleichen Interessen treffen sich und tauschen sich über Hobby oder Beruf aus. Durch derartige Kontakte entstehen Verbindungen zu anderen Menschen, die wieder ähnliche Interessen haben oder über bestimmtes Wissen verfügen, auf das im Bedarfsfall zugegriffen werden kann. Als zu dieser Arbeit passendes Beispiel sei hier eine Linux Usergroup erwähnt. Verbindungen zwischen Menschen können innerhalb der sozialen Netze durch eigene Aktivität entstehen, wie das Beitreten zu besagter Linux Usergroup. Es können aber auch zufällige Verbindungen zwischen Personen entstehen, welche sich beispielsweise an ihrem Arbeitsplatz kennen lernen. Menschen sind in der Lage, andere Personen zu kategorisieren und ihnen bestimmte Eigenschaften zuzuordnen. Auf diese Art und Weise können bei der Suche nach Informationen Anfragen gezielt an bestimmte Personen aus dem sozialen Umfeld gestellt werden, die bestimmten Kriterien entsprechen. Das führt oftmals zu guten Suchergebnissen. Nachteile der Kommunikation innerhalb sozialer Netze sind ungenaue Übertragungen. Das zeigt sich unter anderem in Geschichten, die als bewiesene Tatsachen weitergegeben werden, obwohl sie nicht der Wahrheit entsprechen. Solche Geschichten spielen meist in der Gegenwart und werden als Urbane Legenden bezeichnet. Die starke Verbreitung dieser Erzählungen steigert fälschlicherweise deren Glaubwürdigkeit.

Das Internet ist eine Struktur, die Querverbindungen, sogenannte Shortcuts, zwischen Personen herstellt, welche sonst keine direkte Verbindung zueinander hätten. Interessengemeinschaften finden sich in Foren und auf Internetplattformen auch über räumlich große Distanzen. Im Abschnitt über strukturierte P2P-Netze auf Seite 86 wird der Begriff des Overlay-Netzwerkes definiert. Man kann das Internet mit seinen verschiedenen Diensten als ein solches Overlay-Netzwerk verstehen, welches die Strukturen der sozialen Netzwerke erweitert. Dieses Netzwerk ermöglicht das Erstellen von Verbindungen, die ohne diese Struktur nicht zustande gekommen wären. Das Internet wird langfristig dazu führen, dass manche Teile des sozialen Netzwerkes an Bedeutung verlieren, da das Auffinden von Informationen, Produkten und Dienstleistungen besser und schneller durch Dienste des Internets zu bewerkstelligen sind. Antiquariate sind beispielsweise für die Suche nach einem vergriffenen Buchtitel der Suche im Internet unterlegen. Dennoch wird zukünftig nicht auf die vorhandenen sozialen Netze verzichtet werden können. Viele Abläufe in unserer Gesellschaft werden auch in Zukunft durch die Struktur und Beschaffenheit unserer sozialen Netzwerke bestimmt.

1.2 Vernetzung im Gesundheitswesen

Inmitten dieser Strukturveränderungen befindet sich auch das Gesundheitssystem im Wandel. Die Versorgung von Patienten nimmt durch gestiegene Möglichkeiten bei der Diagnostik und Behandlung immer komplexere Formen an. Das führt dazu, dass die Versorgung nur noch durch die Zusammenarbeit unterschiedlicher Dienstleister und Institutionen erfolgen kann, was die Kosten des Gesundheitssystems in die Höhe treibt. Diese weitestgehend voneinander unabhängigen Instanzen benötigen Informationen von anderen Dienstleistern, um eine optimale Behandlung des Patienten gewährleisten zu können. Die Menge derartiger Daten nimmt dabei so rasant zu, dass ein Auffinden der relevanten Informationen in den unterschiedlichen Systemen sehr aufwändig werden kann. [Dog09] Derartige Informationen können beispielsweise die Diagnosen eines Patienten, der Versicherungsstatus, neue Behandlungsmethoden oder die Weiterbehandlungsmöglichkeiten in anderen Einrichtungen sein. Die besondere Herausforderung dabei besteht darin, dass ein Teil der betreffenden Daten sehr sensibel ist und der Umgang damit strengen gesetzlichen Bestimmungen genügen muss. Das ist auch der Grund dafür, dass Daten von Patienten nicht wie Artikel oder Dateien in Portalen zum Download bereitgestellt werden können.

Trotz dieser Entwicklung existieren auch innerhalb der Dienstleister im Gesundheitswesen soziale Netze und Beziehungsgeflechte. Nach wie vor ist ein Krankenhaus auf ein gutes Verhältnis zu seinen einweisenden Ärzten angewiesen. Patienten bauen ein Vertrauensverhältnis zu ihrem Hausarzt auf. Die Instanzen dieses Gesundheitsnetzwerkes sind dabei nicht zentral organisiert und gesteuert. Dienstleister sind größtenteils selbstständige Institutionen. Dennoch ist ein einzelner Hausarzt nicht in der Lage, seine Patienten in ausreichendem Maße zu behandeln. Eine Zusammenarbeit der einzelnen Dienstleister ist dafür unverzichtbar. Um diese Zusammenarbeit mit Hilfe der Informationstechnologie zu unterstützen, werden vielfältige Bemühungen unternommen. Viele Länder haben deshalb große IT-Projekte ins Leben gerufen. Nicht nur Deutschland plant seit geraumer Zeit die Einführung der Elektronischen Gesundheitskarte und der dazugehörigen Telematikinfrastruktur. [HRNea08] Während in den Industrieländern meist die Senkung der immer mehr ansteigenden Kosten [ATT08] im Vordergrund steht, verfolgen Entwicklungsländer oftmals andere Ziele. Dort wird in der Informationstechnologie eine Chance gesehen, Menschen in dünn besiedelten Gebieten besser und schneller medizinisch zu versorgen. [Mal07] Dazu werden vielerorts keine Kosten und Mühen gescheut, um schwer zugängliche Regionen mit Hilfe von Satellitentechnik an das Internet anzubinden [CVMea07]. So wurde auf Kreta eine Plattform getestet, die es erlaubt, mittels Instant Messaging zu kommunizieren und parallel auf Patientendaten zuzugreifen. [CCMea03]

Die Beispiele zeigen, welche hohe Wertigkeit der Austausch von Daten im Gesundheitswesen über Netzwerke, welcher Art auch immer, für die Gesellschaft besitzt. Dieser Austausch wird jedoch durch die Heterogenität der Dienstleister im Gesundheitswesen, die Sensibilität der Patientendaten, die Dynamik der Abläufe und nicht zuletzt durch den Umstand erschwert, dass es im Gesundheitswesen nicht um die Verarbeitung von Produkten oder den Umschlag von Waren geht, sondern um die Behandlung von Menschen. Entscheidungen, die auf falschen Informationen beruhen, können fatale Folgen für den Patienten haben. Das ist der Grund, weshalb die natürlichen Vernetzungen und Beziehungen

von den verschiedenen Personen, die im Gesundheitswesen arbeiten, oftmals bevorzugt werden. Der Anruf des niedergelassenen Arztes beim Krankenhausarzt wird dem Portal in bestimmten Situationen vorgezogen.

1.3 Die wissenschaftliche Fragestellung

In dieser Arbeit soll untersucht werden, wie eine Software aufgebaut werden müsste, welche den Austausch von Informationen zwischen so weit als möglich gleichberechtigten Knoten erlaubt. Es soll gezeigt werden, wie Abläufe definiert werden können, ohne dass diese Abläufe von zentraler Stelle gesteuert werden. Dabei muss im besonderen Maße auf die Einhaltung der Datenschutzbestimmungen geachtet werden. Es soll der Frage nachgegangen werden, wie das Overlay-Netzwerk einer Infrastruktur im Gesundheitswesen beschaffen sein müsste, damit Informationen schnell gefunden werden können und Abläufe reibungslos funktionieren. Wichtig dabei ist, dass die Struktur dieses Netzes ohne hohen Aufwand an die aktuellen Erfordernisse angepasst werden kann. Damit soll der hohen Dynamik der Prozesse im Gesundheitswesen Rechnung getragen werden. Konkret sollen folgende Thesen überprüft werden:

- Für den Austausch der wichtigsten Informationen im Gesundheitswesen genügt eine begrenzte Zahl von Basisobjekten.
- Einrichtungsübergreifende Behandlungsabläufe lassen sich abbilden, ohne dass diese durch eine zentrale Instanz gesteuert werden.
- Mit Hilfe von P2P Technik kann nach Patientendaten gesucht werden.
- Bei dieser Suche können die Anforderungen des Datenschutzes erfüllt werden.
- Eine solche übergreifende Infrastruktur lässt sich an vorhandene Informationssysteme im Gesundheitswesen anbinden.

1.4 Motivation

1.4.1 Ressourcen teilen

Das Gesundheitswesen ist ein komplexes Gebilde verschiedener Dienstleister. Viele Aufgaben sind nur mit spezieller Software oder Hardware zu bewältigen. Beispielsweise müssen Röntgenbilder in naher Zukunft von Softwareprogrammen vorselektiert werden, da die Flut von Bildern nicht mehr von den Radiologen mit der nötigen Sorgfalt betrachtet werden kann. Dies trifft besonders auf Schnittbilder zu. Die Abrechnung mit den Krankenkassen erfolgt über Fallpauschalen, welche zu Gruppen zusammengefasst und dann mit den Kassen abgerechnet werden. Diese Gruppierung erfolgt nach komplexen Regelsystemen und kann nur mit bestimmter Software optimiert werden. Diese Aufgaben können oftmals nicht von jedem Dienstleister hinreichend bedient werden. Die Infrastruktur soll die Möglichkeit schaffen, Ressourcen und somit auch Funktionen anderen Einrichtungen zur Verfügung zu stellen und so Synergien zu erzeugen.

1.4.2 Workflow zwischen verschiedenen Knoten

Die Fähigkeit, Funktionen anderer Systeme über Services zu nutzen, ermöglicht es, den Workflow beispielsweise zweier Krankenhäuser zu verzahnen. Wird im System eines Krankenhauses ein Patient in eine andere Klinik verlegt, kann über einen Service die Anmeldung in der anderen Klinik automatisch ausgelöst werden. Röntgenbilder und Befunde, die zur aktuellen Episode des Patienten gehören, können automatisch an die weiterbehandelnde Klinik gesendet werden. Auch die Terminvergabe kann so geregelt werden. Wird eine Untersuchung angeordnet, die im eigenen Haus nicht möglich ist, dann kann automatisch ein Termin ausgehandelt werden, der daraufhin in das eigene Patientenmanagementsystem übernommen wird.

1.4.3 Statistische Daten

Die einzelnen Dienstleister im Gesundheitswesen müssen sich regelmäßigen Kontrollen durch Behörden und Institutionen der Selbstverwaltung unterziehen. In einem bestimmten Intervall müssen Daten an die Ärztekammer und andere Einrichtungen übermittelt werden. Dabei müssen diese Daten anonymisiert werden. Durch einheitliche Schnittstellen und Datenobjekte wäre es möglich, diese Übermittlung zu automatisieren und mit Hilfe bestimmter Regeln klar zu definieren, welche Daten in welcher Form übertragen werden dürfen.

1.4.4 Doppeluntersuchungen

Ein Faktor, welcher die Kosten im Gesundheitswesen in die Höhe treibt, sind Doppeluntersuchungen. Wird ein Patient verlegt, so werden oftmals Untersuchungen in der weiterbehandelnden Einrichtung nochmals durchgeführt. Das verursacht Kosten und kann gerade bei Röntgenuntersuchungen für die Patienten gesundheitsschädlich sein. Gründe dafür sind oft Unkenntnis über die Existenz bereits stattgefundener Untersuchungen, Inkompatibilität der Datenformate oder ein hoher Aufwand bei der Einrichtung von Übertragungswegen über das Internet. Hilfreich wäre eine Suche, die das Vorhandensein von Untersuchungen ermitteln kann sowie die anschließende Übertragung der Ergebnisse mittels standardisierter Objekte.

1.5 Bestandteile des Gesundheitswesens

In diesem Kapitel wird ein Überblick über die einzelnen Bestandteile des Gesundheitswesens gegeben. Abbildung 1 veranschaulicht die Zusammensetzung des Gesundheitssystems und zeigt beispielhaft Kommunikationsvorgänge zwischen verschiedenen Einrichtungen.

1.5.1 Definition

> „Das Gesundheitssystem oder Gesundheitswesen eines Landes umfasst alle Personen, Organisationen, Einrichtungen, Regelungen und Prozesse, deren Aufgabe die Führung und Erhaltung der Gesundheit sowie die Vorbeugung und Behandlung von Krankheiten und Verletzungen ist.“[Jan10]

1.5.2 Finanzdienstleister

Jedem Gesundheitssystem liegt ein bestimmtes Finanzierungssystem zugrunde. Man unterscheidet dabei die Finanzierung durch Steuermittel und die Finanzierung durch Krankenversicherungen. In manchen Staaten besteht eine Versicherungspflicht wie z.B. in Deutschland. Durch eine teilweise Privatisierung des Gesundheitssystems existieren in Deutschland derzeit über 200 Krankenkassen. Der Gesetzgeber regelt, welche Leistungen von den Krankenkassen bezahlt werden müssen und welche bezahlt werden können. In Deutschland sind ca. 90 % der Bevölkerung pflichtversichert. Etwa 9 % der Bürger haben sich privat versichert und ca. 1 % sind ohne Versicherungsschutz. [Sta08] Die Leistungserbringer rechnen die erbrachten Leistungen nun mit den privaten und gesetzlichen Krankenkassen ab. Jede Leistung muss mit genau der Krankenkasse abgerechnet werden, bei welcher der Patient zum Zeitpunkt der Behandlung versichert war. Da gewisse Krankenkassen mehr Leistungen bezahlen als andere und die Anforderungen an den Nachweis der Notwendigkeit unterschiedlich sind, werden oftmals nicht alle erbrachten Leistungen bezahlt. Die Krankenkassen sind für die Finanzierung von medizinischen Behandlungen und den dazugehörigen Pflegekosten zuständig. Die Pflege in Altenheimen wird von den Pflegekassen übernommen. Die Abrechnung mit den Krankenkassen ist jedoch völlig anders, als die mit den Pflegekassen. Während mit den Krankenkassen über Fallpauschalen abgerechnet wird, bezahlen die Pflegekassen Pflegesätze. Die Krankenkassen und Pflegekassen sind Körperschaften des öffentlichen Rechts und haben eine Selbstverwaltung. Da die Kassen jedoch vom Staat gesteuert werden, sind diese verpflichtet, bestimmte Informationen an die Gesundheitsbehörde weiterzugeben.

1.5.3 Ambulanter Bereich

In Deutschland arbeiten ca. 134000 niedergelassene Ärzte. Sie sind in der Kassenärztlichen Vereinigung organisiert, welche die Verteilung der einzelnen Fachärzte auf bestimmte Gebiete organisiert. Die niedergelassenen Ärzte rechnen mit den Krankenkassen über Fallpauschalen ab. Die Abrechnung erfolgt jedoch über ein anderes System, als es bei den stationären Einrichtungen verwendet wird. Für den Patienten ist die erste Anlaufstelle die Ambulanz. Erst ein ambulanter Arzt kann einen Patienten in eine stationäre Einrichtung überweisen. Notfälle bilden hier die Ausnahme.

1.5.4 Stationärer Bereich

Die stationäre Versorgung von Patienten geschieht losgelöst von der ambulanten Versorgung. Abgesehen von einer Überweisung, die ein Patient von einem niedergelassenen Arzt benötigt, ist die Behandlung im Krankenhaus völlig anders organisiert als die ambulante Versorgung. Die Trennung dieser beiden Bereiche wird von Kritikern als eine Ursache für mangelnde Effizienz bei der Behandlung angesehen. Eine bessere Zusammenarbeit zwischen den beiden Bereichen ist dringend notwendig. Die Krankenhäuser kommunizieren mit den Krankenkassen, von denen sie finanziert werden. Außerdem sind die Krankenhäuser verpflichtet, an die Ärztekammer und von ihr beauftragte Institute statistische Informationen zum Zwecke der Qualitätssicherung zu liefern. In Deutschland existieren ca. 2160 Kliniken, von denen manche staatlich, andere gemeinnützig

und wieder andere privat organisiert sind. Krankenhäuser werden in Grundversorger, Regelversorger sowie Schwerpunkt- und Maximalversorger unterteilt. Je nach Status dürfen von der Einrichtung bestimmte Behandlungen vorgenommen werden und andere nicht. Das genaue Leistungsmengengerüst wird jedes Jahr neu für jedes Krankenhaus zwischen dem Verband der Krankenkassen und der Klinik verhandelt. Dafür ist ein intensiver Informationsaustausch zwischen Krankenkassen und Klinik notwendig.

1.5.5 Apotheken

Im Arzneimittelgesetz ist geregelt, dass Medikamente, welche eine Beratung des Patienten voraussetzen, nur durch Apotheken verkauft werden dürfen. Bei verschreibungspflichtigen Medikamenten muss außerdem das von einem Arzt ausgestellte Rezept oder ein Arztausweis vorgelegt werden. Jede Bestellung eines verschreibungspflichtigen Medikamentes muss also durch einen Arzt autorisiert werden.

1.5.6 Gesundheitsbehörde

Der Staat nimmt auf vielerlei Art und Weise Einfluss auf das Gesundheitssystem. Durch die Fallpauschalen, welche von einem staatlich beauftragten Institut entwickelt und gepflegt werden, kann Einfluss auf das gesamte System genommen werden. So werden immer mehr Operationen ambulant durchgeführt. Indem für bestimmte Eingriffe nur noch eine niedrige Fallpauschale gezahlt wurde, verpflichtete man die Krankenhäuser, vermehrt Operationen im Sinne der Wirtschaftlichkeit, ambulant durchzuführen. Darüber hinaus prüft der Staat die Dienstleister im Gesundheitswesen auf vielerlei Weise.

1.6 Technische Voraussetzungen im Krankenhaus

1.6.1 Interne Strukturen

In einem modernen Krankenhaus arbeiten heute viele verschiedene Informationssysteme. Es existieren für die unterschiedlichen Bereiche im Krankenhaus vollkommen getrennte Systeme - teilweise von verschiedenen Herstellern. In diesem Kapitel sollen verschiedene Informationssysteme und deren Zusammenwirken innerhalb eines Krankenhauses vorgestellt werden. Dabei wurde sich am Beispiel des Diakoniekrankenhauses Chemnitzer Land (DKC) in Hartmannsdorf orientiert. Die wichtigsten Informationssysteme wurden in Abbildung 2 visualisiert um zu illustrieren, wie diese Systeme untereinander und mit anderen Einrichtungen Daten austauschen.

Die Gesamtheit von Informationssystemen bezeichnet man als Krankenhausinformationssystem (KIS). Um diese Gesamtheit sinnvoll zu strukturieren, muss eines dieser Systeme eine führende Rolle übernehmen. Dieses System ist in der Regel das Patientenmanagementsystem - das sogenannte PMS. Das PMS dient zur Verwaltung der Patientendaten und deren einzelnen Episoden. Ein Finanzbuchhaltungssystem ist oft an das PMS gekoppelt. Auch die Behandlungsdokumentation und Teile der Pflegedokumentation werden oftmals mit Hilfe des PMS erfasst. Nach Paragraph 301 des SGB V sind die Krankenhäuser dazu verpflichtet, ihre Abrechnung mittels elektronischem Datenaustausch an die Krankenkassen zu senden. Somit wird das PMS zum integralen Bestandteil eines Kran-

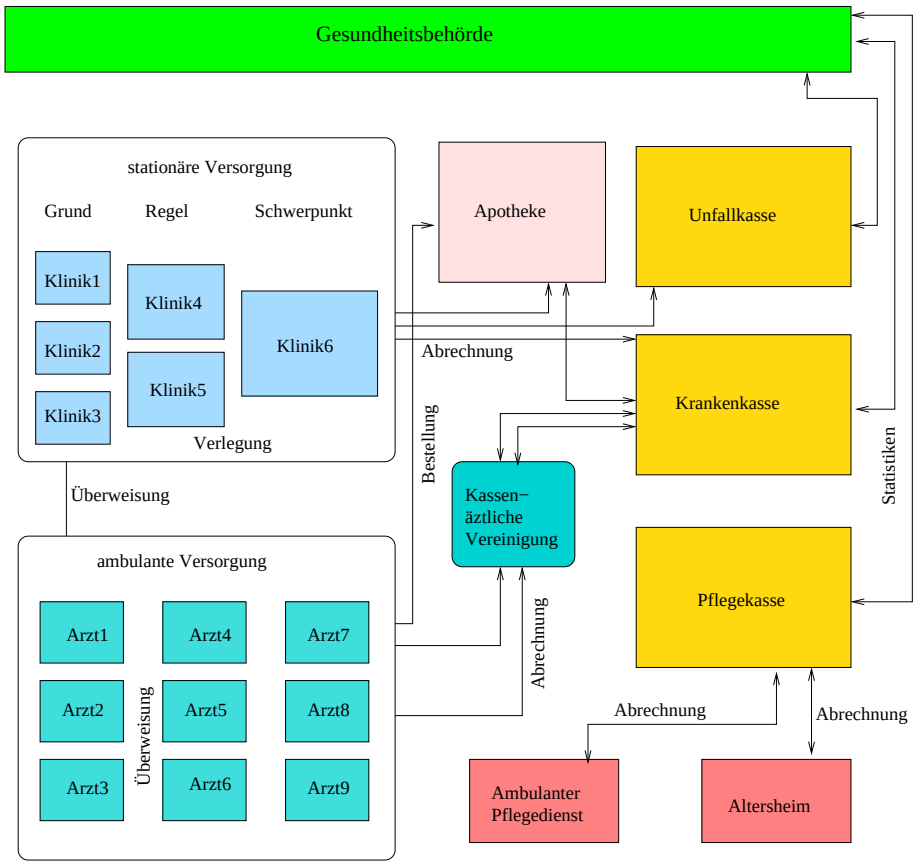

Abbildung 1: Kommunikationsvorgänge im Gesundheitssystem

kenhauses. Über eine Schnittstelle wird Software eingebunden, mit deren Hilfe die Fallpauschalen - die DRGs - nach bestimmten Regeln gruppiert und validiert werden. Die Abkürzung DRG steht für „Diagnoses Related Groups“. Dahinter verbirgt sich ein Abrechnungssystem, welches Pauschalpreise für die Behandlung einer bestimmten Krankheit definiert. Die daraus resultierenden Rechnungsinformationen werden den Krankenkassen in elektronischer Form übermittelt und fließen gleichzeitig in die Finanzbuchhaltung ein. Systeme, die dem PMS untergeordnet sind, sogenannte Subsysteme, bekommen administrative Daten der Patienten über eine Schnittstelle aus dem Patientenmanagementsystem und liefern Befunde an eine elektronische Patientenakte zurück.

Für die Radiologie existieren zwei Systeme. Das Radiologie Informationssystem (RIS) dient dazu, Termine für die Untersuchungen zu planen, wichtige Kenndaten einer Untersuchung (Strahlendosis etc.) und den Befund zu erfassen. Das RIS stellt Geräten wie dem MRT, dem CT oder Sonographiegeräten über das DICOM-Protokoll eine Worklist zur Verfügung. Die Abkürzung DICOM steht für „Digital Imaging and Communications in Medicine“. Die administrativen Daten eines Patienten werden also lediglich im PMS erfasst und anschließend an das RIS übermittelt. Dort kann der Patient ausgewählt und einem Gerät zugeordnet werden, wo er dann in der Worklist erscheint. Die Daten

der bildgebenden Untersuchungen werden anschließend vom Gerät an das Picture Archiving and Communication System (PACS) übermittelt. Dieses System dient dazu, Röntgenbilder im DICOM-Format zu archivieren und bei Bedarf zu suchen.

In ähnlicher Weise wie das RIS arbeitet das Laborsystem. Patientendaten und Aufträge gehen über eine Schnittstelle aus dem PMS in das System. Blutproben werden über Barcodes den einzelnen Aufträgen zugeordnet und die Ergebnisse an das KIS rückübermittelt. Die Anbindung der Geräte im Laborbereich erfolgt auch heute noch fast ausschließlich über serielle Schnittstellen. Weitere Verbindungen bestehen zu einem Küchensystem. Jede Patientenaufnahme und -entlassung wird dort übermittelt. Das Gleiche gilt für das Verwaltungssystem der Patiententelefonie. Alle Patienten werden dort aus dem PMS übernommen. Ein Bereich, der an Bedeutung gewinnt, ist die Qualitätssicherung. Aus dem PMS heraus wird dafür eine Fremdsoftware aufgerufen und die entsprechenden Patientendaten werden auf Anwendungsebene übergeben. Die ausgefüllten Bögen werden dann in einer separaten Datenbank hinterlegt.

Alles in allem kann über die IT-Landschaft gesagt werden, dass sie sich im Krankenhaus sehr inhomogen gestaltet. Administrative Daten werden mehrfach gehalten und durch Insuffizienz mancher Schnittstellen entstehen im Laufe der Zeit gewisse Inkonsistenzen zwischen den einzelnen Systemen. Für die Anbindung von Subsystemen existiert ein Standard namens Health Level 7 (HL7). Darin sind bestimmte Nachrichtentypen definiert, über die Patientendaten innerhalb eines Krankenhauses ausgetauscht werden können. Allerdings ist HL7 ein offener Standard und es existieren hunderte Dialekte. Die Anbindung von Systemen erfordert deshalb immer gewisse Anpassungen. [CCF06] Trotz der durch diese Anpassungen entstehenden Kosten, werden für die Archivierung der unterschiedlichen Untersuchungsdaten mehr und mehr spezialisierte Systeme eingesetzt. Wurden zu Beginn der Entwicklung Sonographiebilder meist im PACS abgelegt, werden heute verstärkt spezielle Archivierungssysteme für diese Art der Untersuchung eingesetzt. Es ist also damit zu rechnen, dass die Anzahl der Schnittstellen innerhalb eines Krankenhauses in Zukunft stark ansteigen wird.

1.6.2 Externe Kommunikation

Das Krankenhaus betreibt zahlreiche Schnittstellen nach außen. Die Form der Kommunikation wird oftmals von den Empfängern bestimmt und ist deshalb sehr vielfältig. Die Abrechnung mit den Krankenkassen erfolgt über spezielle ASCII-Dateien, welche auf einer X400 Mailbox der Telekom abgelegt werden. Das vom Fraunhofer Institut und der Kassenärztlichen Vereinigung entwickelte Doctor to Doctor (D2D) wird für die Abrechnung mit der Berufsgenossenschaft und den Versand der dazugehörigen Berichte eingesetzt. Alle Bögen der Qualitätssicherung werden in regelmäßigen Abständen manuell exportiert und per E-Mail an die zuständige Stelle versandt. Für die Kommunikation mit niedergelassenen Ärzten werden neuerdings Portallösungen eingesetzt, welche über eine Webschnittstelle den Zugriff auf Patientendaten erlauben. Da die Behandlung und Abrechnung im niedergelassenen Bereich vollkommen anders organisiert ist als im stationären Bereich, sind auch die Softwaresysteme vollkommen unterschiedlich strukturiert. Auch die Schnittstellen der Systeme in den beiden Bereichen sind vollkommen inkompatibel. So ist HL7 ausschließlich im stati-

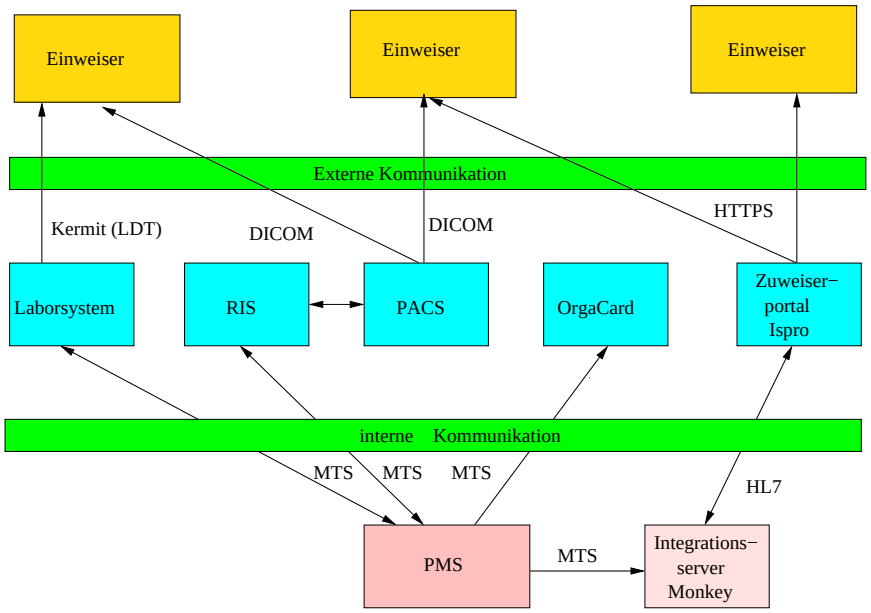

Abbildung 2: Kommunikation im Krankenhaus am Beispiel des DKC

onären Bereich verbreitet. Deshalb sind Portallösungen meist mit proprioritären Schnittstellen ausgestattet. Ist das Portal, welches ein Krankenhaus den niedergelassenen Ärzten zur Verfügung stellt, vom gleichen Hersteller wie das Praxissystem der Praxis, so können Daten aus dem Portal direkt in das Praxissystem übernommen werden. Ansonsten können sich die Ärzte nur über eine gesicherte Website über ihre Patienten informieren. Im Laborbereich ist der „Labor Daten Transfer Standard“ (LDT) verbreitet. Dieser definiert ein Textformat, in welchem Labordaten übertragen werden können. Über das vom Freistaat Sachsen geförderte Projekt „Sachstelemed“ wurde ein Teleradiologieverbund zwischen verschiedenen Krankenhäusern aufgebaut. Die Kommunikation zwischen den Einrichtungen erfolgt mittels des DICOM-Protokolls. Zuerst wurden Verbindungen über ISDN-Router aufgebaut, was zu sehr langen Übertragungszeiten führte. Mittlerweile findet die Übertragung mittels IPSec-Tunnel statt. Die Resonanz von niedergelassenen Ärzte war jedoch sehr gering, sodass an dem Projekt fast ausschließlich Krankenhäuser beteiligt waren.

1.7 Übergreifende Dienstarchitektur

Als Ergebnis der Arbeit soll die Definition von Schnittstellen für ein offenes System vorliegen, an dem sich Dienstleister im Gesundheitswesen selbstständig anbinden können. Eine Middleware soll so an die Anforderungen des Gesundheitswesens angepasst werden, dass sie als Grundlage für die Implementation der verschiedensten Telematikinfrastrukturen im Gesundheitswesen dienen kann. Die besondere Herausforderung dabei besteht darin, alle gesetzlichen Bestimmungen bezüglich des Datenschutzes genauestens einzuhalten. Für die erwähnten Schnittstellen ist deshalb ein Sicherheitskonzept nötig. Es muss garantieren, dass nur von berechtigten Personen auf Daten zugegriffen werden kann und der

Patient sein Mitbestimmungsrecht behält. Gleichzeitig soll das System aber so offen wie möglich sein. Ähnlich dem Internet muss es möglich sein, dass jeder Dienstleister sich selbstständig mit vertretbarem Aufwand an das System ankoppeln kann. Die Bereitstellung und Nutzung neuer Dienste muss auf Grundlage der Schnittstellendefinitionen und der zugrundeliegenden Middleware ohne hohe Kosten flexibel und schnell realisierbar sein. An den Systemen innerhalb eines Krankenhauses müssen keine aufwändigen Anpassungen durchgeführt werden. Diese arbeiten in gewohnter Weise über Schnittstellen zusammen. Lediglich ein Gateway kommt hinzu, über welches die Systeme Daten in eine elektronische Patientenakte einspeisen. Die Abläufe innerhalb einer Einrichtung, beispielsweise eines Krankenhauses, müssen vollkommen anderen Anforderungen genügen als die zwischen verschiedenen Einrichtungen. Auch die Werkzeuge, die für die interne Organisation von Abläufen nötig sind, unterscheiden sich von denen, die für eine übergreifende Kommunikation zweckmäßig sind. [TGLea09] Deshalb ist die interne Organisation von Arbeitsabläufen nicht Inhalt dieser Arbeit, die sich mit der übergreifenden Kommunikation auseinandersetzt.

Die Einrichtungen, welche an der Dienstarchitektur angeschlossen sind, können dann auf die Daten der elektronischen Patientenakte einer anderen Einrichtung zugreifen. Jeder Teilnehmer stellt seinerseits den anderen Teilnehmern Dienste zur Verfügung. Das können von Teilnehmer zu Teilnehmer vollkommen unterschiedliche Dienste sein, jedoch müssen bestimmte Dienste immer bereitgestellt werden. Das Gateway kann mit den Systemen innerhalb der Einrichtung kommunizieren. Die vorhandenen Systeme können das Gateway via Web-Services ansprechen. Die Authentifikation der Teilnehmer, die Zuteilung der Rechte und die Verschlüsselung wird vom Dienstsystem vollkommen transparent realisiert. Das Gateway ist die Schnittstelle zwischen den Prozessen, Protokollen und Systemen innerhalb einer Einrichtung zur übergreifenden Dienstarchitektur, welche die einzelnen Einrichtungen verbindet. Nach außen stellen sich die Einrichtungen lediglich als Menge von Diensten dar, welche der jeweilige Knoten bereitstellt. Jeder Knoten muss in der Lage sein, selbstständig Geschäftsprozesse zu definieren. Diese müssen entweder statisch geplant werden oder können auch dynamisch verzweigen. So muss beispielsweise die Möglichkeit bestehen, einen Prozess abhängig von diversen Untersuchungsergebnissen dynamisch zu ändern. Die Dienstarchitektur muss die Kommunikation zwischen ganz unterschiedlichen Knoten ermöglichen. Viele dieser Knoten haben jedoch teilweise ähnliche Anforderungen an die Kommunikation. So muss die Bestellung eines Medikaments genauso vom Arzt autorisiert werden wie die Beauftragung einer CT-Untersuchung. Aus diesem Grund sollen in der Dienstarchitektur Dienste und Funktionen in Klassen unterteilt werden. Für Dienste einer bestimmten Klasse müssen jeweils gewisse Rahmenbedingungen gelten. Diese Dienste können dann als Grundlage für ganz unterschiedliche Vorgänge zwischen den einzelnen Knoten dienen.

1.8 Gliederung der Arbeit

In diesem Abschnitt wird der Aufbau der Arbeit grob umrissen. In Kapitel 2.1 und 2.2 werden beispielhaft einige teilweise bereits im produktiven Einsatz befindliche Telematiksysteme im Gesundheitswesen vorgestellt. Dieses Kapitel soll verdeutlichen, auf welchem Stand der Entwicklung sich das Gesundheitswesen mit seinen unterschiedlichen Facetten im Bereich der Informationstechnologie

befindet. Die Kapitel 2.3, 2.4 und 2.5 stellen Werkzeuge aus den Bereichen Serviceorientierte Architektur, Grid-Computing und Peer-to-Peer Netzwerken vor und bewertet diese bezüglich ihrer Verwendbarkeit bei der Implementierung einer generischen Infrastruktur für das Gesundheitswesen. Im Kapitel 3 wird zunächst die Gliederung der Software in verschiedene Arten von Diensten und Modulen beschrieben und das Zusammenspiel mit anderen Systemen innerhalb der IT- Landschaft im Gesundheitswesen skizziert. In diesem Kapitel wird weiter eine Menge von Basisobjekten eingeführt, welche die relevanten Daten kapseln und ein einheitliches Format für die Übertragung darstellen. Wie die Übertragung der Objekte vonstatten geht, wird in einem Abschnitt dieses Kapitels ebenfalls beschrieben. Es wird anschließend dargelegt, wie ein komplexer Behandlungsprozess über die Grenzen einer Einrichtung hinaus damit gesteuert werden kann. In Kapitel 4 werden die daraus resultierenden Dienste in Klassen unterteilt und Beispiele für Dienste dieser Klassen vorgestellt. Kapitel 5 setzt sich mit der Realisierung einer verteilten Suchfunktion innerhalb dieser Infrastruktur auseinander und das Kapitel 6 beschreibt das Datensicherheitskonzept. Im Kapitel 7 wird dargelegt, wie eine Anbindung der Infrastruktur an vorhandene Systeme umgesetzt wird. Die Validierung durch die Implementation einer prototypischen Anwendung wird in Kapitel 8 vorgestellt und durch zwei Szenarien untermauert. In Kapitel 9 werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick auf weiterführende Forschungsthemen gegeben.

2 Grundlagen

2.1 Systeme im Gesundheitswesen

2.1.1 D2D

D2D [Fra05] ist ein von der Kassenärztlichen Vereinigung Nordrhein-Westfalen entworfenes System zum Austausch von Dokumenten. Es basiert auf dem vom Fraunhofer-Institut entwickelten PatDoc. Dieses System arbeitet nach dem Store-and-Forward Mechanismus und nutzt ein Netzwerk von Datenaustausch-Servern zum temporären zwischenspeichern. PatDoc verwendet ein patentiertes Doppelschlüsselverfahren, mit dem es möglich ist, erst nach dem Versenden einer Nachricht den Empfänger zu definieren. Damit soll der Forderung Folge geleistet werden, dass der Patient sich seinen Arzt selbst wählen kann. Das PatDoc System versucht den Versandvorgang von Arztbriefen zu automatisieren. Dabei arbeitet das System ähnlich wie ein E-Mail-System. Nachrichten mit beliebigem Inhalt werden auf einem Kommunikationsserver im Postfach des Empfängers abgelegt und später dort abgeholt. Dabei werden drei verschiedene Versandarten unterschieden. Erstens der adressierte Versand, zweitens der gerichtete Versand und drittens der ungerichtete Versand. Die Daten werden auf dem Kommunikationsserver verschlüsselt abgelegt. Nur der Empfänger hat die Möglichkeit, die Daten zu entschlüsseln.

Die Kommunikationssicherheit soll mit der Verwendung von direkten ISDN-Verbindungen gesichert werden. Als Zugangskontrolle nutzen die Einwahl-Router einen Radiusdienst, der nur registrierten PatDoc-Anwendern Zugang gewährt. Beim gerichteten Versand wird das Dokument verschlüsselt und signiert auf dem Kommunikationsserver abgelegt und mit einer elektronischen Vorgangskennung verknüpft. Diese Kennung kann der Patient manuell als Barcode oder auf einer Chipkarte gespeichert zum Arzt seiner Wahl mitnehmen. Der Arzt liest die Vorgangsnummer ein und kann damit das Dokument abholen. Der Schlüssel, mit dem das Dokument chiffriert ist, wird genau wie die Vorgangskennung dem Patienten mitgegeben. Die Dokumente, welche ausgetauscht werden, sind im Extensible Markup Language (XML) Format abgelegt.

Im Rahmen des D2D Projektes wurden verschiedene SCIPHOX (Standardized Communication of Information Systems in Physician Offices and Hospitals using XML) basierte Anwendungen realisiert. Der ungerichtete Versand dient dazu, Dokumente mehreren Empfängern zur Verfügung zu stellen. Die Dokumente werden für die Zeit der Behandlung auf dem Kommunikationsserver gelagert und können von Ärzten abgerufen werden. Diese Dokumente bilden die so genannte PatDoc-Netzkrankenakte. Derjenige, der diese Akte angelegt hat, fungiert als Moderator. Das Sicherheitskonzept von D2D wird unter anderem von Thomas Maus [Mau05] in starkem Maße hinterfragt. Aus einem Zitat in seinem Vortrag geht hervor, dass Ticket-Systeme eine gute Lösung sind. Das System des Fraunhofer Instituts sei jedoch zu kompliziert und berge daher Sicherheitsrisiken. Weiter ist die Verwendung von ISDN-Verbindungen mit PAP oder CHAP und Rückruf zu kritisieren. Wegen Problemen mit Telefonanlagen ist oft der Rückruf nicht realisierbar und so wurde eine Authentifikation mit PAP oder CHAP implementiert. Diese Protokolle sind jedoch nicht sicher. Zusammenfassend kann man sagen, dass D2D mit PatDoc ein Kommunikationssystem ist, das mit einem speziellen E-Mail-System vergleichbar ist. Es bedarf immer der

manuellen Zuweisung von Informationen. Aßerdem sind spezielle Kommunikationsserver nötig und damit eine gesonderte Infrastruktur.

2.1.2 Die österreichische Gesundheitskarte

Österreich ist dabei, die so genannte E-Card auszurollen. Diese Gesundheitskarte löst den Krankenschein ab, den die Arbeitnehmer bisher vom Arbeitgeber abholen mussten, wenn sie zum Arzt gehen wollten. Die Karte enthält Speicherbereiche, in denen Notfalldaten und das elektronische Rezept gespeichert werden können. Die Kommunikation wird mittels sogenannter GINA-Boxen durchgeführt, auf denen ein gehärtetes Linux läuft. Die Karte selbst enthält neben Speicherplatz einen Mikroprozessor, der für die Authentifikation genutzt wird. Sie ist so aufgebaut, dass die elektronische Signatur auch für Verwaltungsaufgaben verwendet werden kann. Bei Auslieferung der Karte enthält sie jedoch nur die Patientendaten, wie es bei der deutschen Krankenkassenkarte schon seit Jahren der Fall ist. Die E-Card soll Ärzten die Möglichkeit geben, auf Patientendaten, welche auf zentralen Servern abgelegt sind, zuzugreifen. Der Zweck der E-Card ist aber in erster Linie zu prüfen, auf welche Leistung der Patient Ansprüche hat. Mit anderen Worten: Es kann noch vor Beginn der Behandlung geprüft werden, ob der Patient krankenversichert ist und die Leistung im Anschluss abgerechnet werden kann. Andere Anwendungen sind vorgesehen, aber noch nicht umgesetzt.

2.1.3 Medigrid

Medigrid [KBBea09] ist ein Forschungsprojekt im Rahmen des D-Grid. Es dient dazu, gemeinsam aufgebaute und genutzte Datenpools zu vernetzen. Schwerpunkt ist die Unterstützung einer kollaborativen Verbundforschung und die Verbesserung der interdisziplinären, standortunabhängigen Zusammenarbeit in der biometrischen Forschung. Das Medigrid bietet Dienste an, welche die Forschung an hochdimensionalen Daten mit korrelativen Verknüpfungen von vielfältigen genotypischen und phänotypischen Daten beschleunigt. Das Projekt ist in verschiedene Module untergliedert. Diese Module sind Bioinformatik, medizinische Bildverarbeitung und Ontologie. Für diese Dienste existieren bereits einige Anwendungen, die über ein Portal aufgerufen werden können.

Im Bereich der Bioinformatik sind das z.B. der Augustus Gensequenz Analyzer, mit dem man genetische Sequenzen analysieren und einer bestimmten Spezies zuordnen kann. Eine zweite Anwendung dieses Moduls ist SNP (Single Nucleotide Polymorphism). SNP's sind Punktmutationen, die sich innerhalb einer gewissen Population durchgesetzt haben. Diese können als Suchmuster verwendet werden, um bestimmte Abschnitte der DNA zu finden. Außerdem korrelieren SNP's mit bestimmten Eigenschaften, wie z.B. dem Risiko bestimmter Krankheiten. Der Dienst SNP ermöglicht den Zugriff auf eine große Datenbank bekannter SNP's. Im Bereich der medizinischen Bildverarbeitung existieren derzeit drei verschiedene Anwendungen. Ein Dienst ermöglicht es, Ultraschallbilder, welche während einer Biopsie der Prostata entstehen, in dreidimensionale Abbildungen umzuwandeln und auszuwerten. Der Dienst „Virtuelle Gefäßchirurgie“ ist in der Lage, Aufnahmen verschiedener Modalitäten auszuwerten. Aus den Bilddaten kann die Anwendung eine Simulation errechnen, welche die Strömungsgeschwindigkeiten und die daraus resultierende Belastung

der Blutgefäße dreidimensional darstellt.

Ein weiterer Dienst des Moduls medizinische Bildverarbeitung ist „Functional Magnetic Resonance Imaging“ (FMRI). Er befasst sich mit der Auswertung sogenannter funktioneller Magnetresonanztomographien. Dabei wird ein Proband mehrfach von einem MRT gescannt und teilweise Reizen ausgesetzt. Durch die Unterschiede der Aufnahmen können Rückschlüsse auf Stoffwechselvorgänge geschlossen werden und mittels Wahrscheinlichkeiten die räumlichen Ursprünge dieser Vorgänge berechnet werden. Diese Methode dient vornehmlich zur Bestimmung aktiver Bereiche im Gehirn. FMRI ist derzeit als Demoversion über das Portal des Medigrid aufrufbar.

Das Modul Ontologie umfasst verschiedene Dienste, die es ermöglichen, Informationen und Daten zu kategorisieren und miteinander zu verknüpfen. Die Basis bildet das Standard Grid Datei Interface (OGSADAI). Beispielsweise die Gene-Ontologie ist eine Domäne des Moduls Ontologie, welche 3 Subontologien beinhaltet. Sie umfasst ca. 19000 Terme, welche über IS_A und PART_OF Beziehungen miteinander verknüpft sind. Das Portal des Medigrid bildet die Schnittstelle, um Begriffe und Muster innerhalb der verschiedenen Ontologien zu suchen.

unterstützen soll. Das D-Grid soll zukünftig in das EU-DataGrid integriert werden. [SDL+03] [BCF+03] Aufgrund seiner Verwandtschaft zum D-Grid unterstützt das Medigrid Schnittstellen der Middleware Globus-Toolkit und Unicore. Ziel des Projektes ist nicht die Behandlung von Patienten zu unterstützen, sondern Forschern eine Möglichkeit zur Zusammenarbeit zu schaffen.

2.1.4 Artemis

Artemis [EAT06] ist ein von der EU gefördertes Forschungsprojekt, das die Verbesserung der Interoperabilität zwischen Krankenhäusern vornehmlich im KIS-Bereich erreichen will. Dazu werden Ontologien [DLS+] zwischen verschiedenen Standards beschrieben, mit deren Hilfe eine Umsetzung zwischen diesen erfolgen kann. Die Kommunikation wird über semantische Webservices realisiert. Diese Webservices sind mit Metadaten ausgestattet, die es erlauben, die Dokumente und Daten in andere Standards zu konvertieren. Im Rahmen des Projektes wurde ein Protokoll entwickelt, welches die Suche von Patientendaten ermöglichen soll. Dabei wurden P2P-Ansätze verwendet. Die einzelnen Knoten werden dabei in Gruppen unterteilt und diese mit Hilfe eines Super-Peers gekoppelt. Anfragen werden dann mittels Broadcast an andere Einrichtungen weitergeleitet. Da das Projekt international angelegt ist und es in vielen Ländern keine eindeutigen Patientennummern gibt, muss anhand der demographischen Daten des Patienten gesucht werden. Um den Datenschutz zu gewährleisten, bedient sich die Patientensuche einer Kontrollnummer, wie sie auch bei anderen Informationen, so dem Krebsregister in Niedersachsen, eingesetzt wird. Dazu werden dem Patienten Kontrollnummern zugeordnet, die sich aus den demographischen Daten ergeben. Da diese Daten von Land zu Land unterschiedlich aussehen, werden die Daten zuerst gesplittet, dann standardisiert und anschließend in Kontrollnummern umgewandelt. Zusätzlich können von den demographischen Daten noch phonetische Zeichen erzeugt werden, z.B. in Soundex. Bei Soundex handelt es sich um einen phonetischen Algorithmus zur Indizierung von Wörtern und Phrasen nach ihrem Klang. Da diese Kodierung stark länderspezifisch ist, werden meist Kontrollnummern der demographischen Daten einmal mit und

einmal ohne phonetische Beschreibung erzeugt. Die Umwandlung in Kontrollnummern geschieht nun mittels Message Digest Algorithm 5 (MD5) , Secure Hash Algorithm (SHA-1) oder RACE Integrity Primitives Evaluation Message Digest (RIPEMD-160).

Die ausgetauschten Nachrichten werden dann mit Hilfe eines auf allen Knoten bekannten Schlüssels chiffriert. Die Records können nicht in die demographischen Daten des Patienten umgewandelt werden. Es ist jedoch möglich, die eingehenden Records mit den vorhandenen Datensätzen der Patienten zu vergleichen. Das ist sehr aufwändig und erfordert bei jeder Anfrage die Neubildung und den Vergleich der Kontrollnummern aller Patienten. Aus diesem Grund werden sogenannte Blockvariablen übertragen. Diese dienen dazu, die Anzahl der zu vergleichenden Patienten einzuschränken. Dazu werden Komponenten wie der Tag der Geburt oder die Postleitzahl unverschlüsselt übertragen. Dabei muss jedoch beachtet werden, dass diese Blockvariablen den Personenkreis nicht soweit einschränken, dass ein Unbeteiligter Rückschlüsse auf die Identität des gesuchten Patienten ziehen kann. Im Netzwerk existieren dazu verschiedene spezielle Knoten, die unterschiedliche Aufgaben wahrnehmen. Der Record Linkage Server (RLS) beispielsweise empfängt Anfragen und vergleicht die darin enthaltenen Records mit einer Liste, welche er von verschiedenen Repositories bekommen hat. Diese Liste wurde auf Grund der Blockvariablen und des Session Key, der für jede Anfrage neu erzeugt wird, zusammengestellt. Da der RLS nicht im Besitz des Session Key ist, hat er keine Möglichkeit, Rückschlüsse auf den Inhalt der demographischen Daten zu ziehen. Die sogenannte Trusted Third Party (TTP) ist ein Netzwerkknoten, der den Kommunikationspartnern einen Session Key zur Verfügung stellt. Er ermöglicht, dass zu Beginn der Suche keine direkte Kommunikation zwischen zwei Knoten stattfinden muss. Der TTP muss eine vertrauenswürdige Komponente im Netzwerk sein. Das Ergebnis einer Anfrage enthält eine URL, welche auf die lokale ID des Patienten im anderen Knoten verweist. Das Auffinden eines Repositories, welches Records eines Patienten enthält, ist die erste Phase der Kommunikation. In der zweiten Phase müssen die Daten angefordert, übertragen und dargestellt werden. Das wird mittels des „Retrieval Information for Display“ (RID) Protokolls ermöglicht. Es erlaubt die sichere Übertragung von Patientendaten über Organisationsgrenzen hinweg und ermöglicht die Darstellung dieser Daten im jeweiligen Kontext. Artemis ist kein reines P2P-Netzwerk. Es existieren vertrauenswürdige Peers, die als Gateway für die Kommunikation dienen. Die Suche erfolgt über einen Broadcast. Eine Anonymisierung erfolgt über Hashwerte, die von einzelnen Komponenten der demographischen Daten gebildet werden. Der Schwerpunkt bei Artemis liegt auf der Interoperabilität der verschiedenen Systeme. Die demographischen Daten, mit deren Hilfe gesucht wird, werden - wie auch die gesuchten Daten selbst - mit Hilfe semantischer Ontologien in eine einheitliche Form gebracht.

2.2 Die Elektronische Gesundheitskarte

2.2.1 Einführung

[Gem08] Eines der größten IT-Projekte weltweit ist die Einführung der Elektronischen Gesundheitskarte (EGK) und deren Infrastruktur in Deutschland. In diesem Kapitel soll deren Architektur kurz vorgestellt und von der in dieser Arbeit entworfenen Infrastruktur abgegrenzt werden. Die EGK ist eine mit

einem Mikrochip ausgestattete Karte, welche mittels asymmetrischer Kryptographie eine eindeutige Signatur erstellen kann. Es ist also damit möglich, eine Authentifizierung des Patienten durchzuführen. Bei der dahinterliegenden Infrastruktur handelt es sich jedoch nicht um ein existierendes System. Der Gesetzgeber hat in §291a des SGB V die Eckpunkte für die EGK festgelegt. Die Gematik hat diese Anforderungen verfeinert und bestimmte Standards und Rahmenbedingungen definiert, die in diesem Kapitel betrachtet werden. Die EGK-Infrastruktur sieht vor, dass alle Praxisverwaltungssysteme, Krankenhausinformationssysteme und Apothekensysteme um eine spezielle Schnittstelle erweitert werden. Diese Schnittstelle wird mit einem Konnektor verbunden. Dieser Konnektor ermöglicht die Verbindung zu den Diensten und zur Public Key Infrastruktur der EGK. Die Gesundheitskarte soll den Zugang zu sogenannten Pflichtanwendungen gewährleisten. Das sind Anwendungen, die in jedem Fall umgesetzt werden müssen und für deren Verwendung die Zustimmung des Patienten nicht notwendig ist. In erster Linie handelt es sich dabei um die Speicherung administrativer Daten. Darüber hinaus sollen sogenannte „Freiwillige Anwendungen“ implementiert werden. Diese sollen den Arbeitsablauf einer Behandlung unterstützen. Freiwillige Anwendungen sind beispielsweise der Zugang zu einer elektronischen Patientenakte, die Speicherung von Notfalldaten auf der EGK oder eine Arzneimitteltherapiesicherheitsprüfung. Weiter besteht die Möglichkeit, sogenannte Mehrwertanwendungen zu implementieren. Diese Anwendungen sind nicht innerhalb der Spezifikation beschrieben, können jedoch die EGK nutzen. Mehrwertanwendungen müssen zuerst von der Gematik abgenommen und auf ihre Konformität zu den Anforderungen der EGK geprüft werden.

2.2.2 Hintergrund

An die Architektur der Gesundheitskarteninfrastruktur werden diverse Anforderungen gestellt. Viele dieser Anforderungen ergeben sich aus dem Gesetz für die Modernisierung des Gesundheitswesens. An dieser Stelle soll ein kurzer Überblick über dessen Inhalt gegeben werden. Datenzugriffe müssen protokolliert werden, und mindestens die letzten 50 dieser Zugriffe müssen von den dazu berechtigten Teilnehmern jederzeit überprüft werden können. Dem Patienten muss es möglich sein, seine Daten zu veröffentlichen oder zu verbergen. Er hat die Kontrolle über seine Daten. Im Falle des Ablaufes einer Karte muss gewährleistet sein, dass der Zugriff auf bestimmte gesundheitsrelevante Daten des Patienten weiterhin möglich ist. Die Teilnehmer an der Telematikinfrastruktur haben die Aufgabe, so wenig wie möglich personenbezogene Daten zu erheben. Die Gematik muss beim Entwurf der Architektur auf Interoperabilität und Kompatibilität zu anderen Systemen achten. Dabei muss die Einführung von Mehrwertdiensten von Anfang an mit vorgesehen werden, auch wenn diese zu Beginn noch nicht zu realisieren sind. Die Gematik muss Spezifikationen zu den Schnittstellen erarbeiten, mit denen sich die Primärsysteme an die Infrastruktur anschließen können. Diese Systeme sind ansonsten unabhängig und werden von der Spezifikation nicht berührt. Der Versicherte muss selbst entscheiden können, wer Zugriff auf welche Daten bekommt. Der Patient muss seine Zustimmung zur Bereitstellung seiner Daten geben und darf diese Berechtigungen jederzeit wieder zurücksetzen. Das Anwendungsgateway, welches auch Broker genannt wird, muss in der Lage sein, Patientendaten zu anonymisieren. Die Infrastruktur

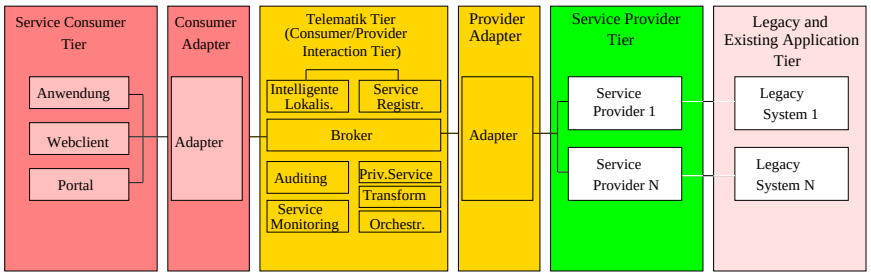

Abbildung 3: Kommunikation über Intermediäre

muss die Möglichkeit erhalten, die Daten der EGK regelmäßig zu prüfen. Dies geschieht mindestens einmal im Quartal durch den Kostenträger. Zum Kommunikationsvorgang innerhalb der Infrastruktur muss es ein Ausfallkonzept geben, welches die Versorgung der Patienten auch bei technischen Problemen gewährleistet. Die Nachrichten werden innerhalb der Telematikinfrastruktur mittels Simple Object Access Protocol (SOAP) übertragen. Der Zugriff auf die Anwendungen der Telematikinfrastruktur muss rollenbasiert erfolgen. Je nach Rolle des Anwenders werden bestimmte Daten sichtbar oder auch nicht. Die beteiligten Netzwerke werden ausschließlich über VPN-Verbindungen miteinander gekoppelt. Dazu dient unter anderem der Konnektor. Die in der Infrastruktur hinterlegten Daten müssen revisionssicher gespeichert werden. Es muss jederzeit der gesamte Verarbeitungsprozess nachvollziehbar überprüft werden können. Daten müssen generell verschlüsselt abgelegt werden. Das ist vorgeschrieben, damit weder Systemadministratoren oder sonstige unberechtigte Personen Zugriff auf die Daten erlangen können. Allerdings muss auch ein Konzept erstellt werden, wie auf die Daten zugegriffen werden kann, falls beispielsweise die Karte defekt ist. Im ersten Schritt wird die Authentifizierung des Senders durch nachrichtenbasierte Verfahren gewährleistet. Es muss jedoch ebenfalls die Möglichkeit einer Sitzung für spätere Mehrwertdienste vorgesehen werden. Zertifikate müssen online auf ihre Gültigkeit überprüft werden können. Es werden dabei X.509 und CV-Zertifikate unterstützt.

2.2.3 Architektur der EGK

Die Gesamtarchitektur der Gesundheitskarteninfrastruktur sieht Lösungen für folgende Anforderungen vor: Zuerst muss ein Management der Infrastruktur ermöglicht werden. Hierzu zählen Service Level Agreements, Kapazitätsplanung und ein Monitoring von Diensten. Weiter wird innerhalb der Gesamtarchitektur ein übergreifendes Sicherheitskonzept definiert. Die Nachrichten, welche ausgetauscht werden, orientieren sich an einem Nachrichtenkonzept. Das Auffinden von Services, das sogenannte Discovery, wird ebenfalls von der Gematik festgelegt. Ein Konzept zur Orchestrierung von Services entspricht nicht den Anforderungen an die Infrastruktur. Die Gesamtarchitektur sieht zwei Formen der Kommunikation vor. Zum einen den Austausch von Nachrichten über einen sogenannten Intermediär. Die zweite Form der Kommunikation findet zwischen Consumer und Provider direkt statt. Dabei soll die Kommunikation mittels Intermediäre die Option einer Orchestrierung von Webservices offenhalten. Außerdem kann die Bereitstellung von Notfalldaten und Arzneimitteltherapiesicherheit besser

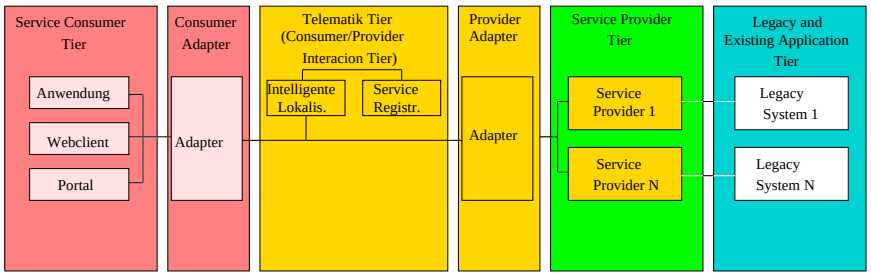

Abbildung 4: Direkte Kommunikation

durch eine Lösung mittels Intermediäre abgebildet werden. In Abbildung 3 sind die verschiedenen Schichten einer Kommunikation mittels Intermediär abgebildet. Die Verwendung einer direkten Kommunikation ist derzeit nicht für die Implementation vorgesehen. Eine solche Kommunikation würde ohne Broker eine direkte Verbindung zwischen den Knoten erlauben, wie in Abbildung 4 zu erkennen ist. Die angestrebten Dienste sollen also über eine Kommunikation der Variante mit Intermediär abrufbar sein. Die Gesamtarchitektur der EGK sieht in erster Linie ein Kommunikationsmuster vor. Die von der Gematik entwickelten Gateways unterstützen das Request/Response - Kommunikationsmuster. Es besteht allerdings die Möglichkeit, die Architektur zu erweitern und andere Kommunikationsmuster, wie beispielsweise „Store and Forward“ oder „Publish and Subscribe“ zu realisieren. Hierfür müssten neue Gateways definiert werden, was bisher noch nicht geplant ist. Eine Authentifizierung findet auf Basis von Nachrichten statt. Der Gesetzgeber verlangt eine informationstechnische Trennung von Pflichtanwendungen und Mehrwertanwendungen. Aus ökonomischen Gesichtspunkten ist es allerdings sinnvoll, eine maximale Anzahl von Komponenten wiederzuverwenden. Aus diesen Gründen werden die Mehrwertanwendungen in unterschiedliche Typen untergliedert. Mehrwertanwendungen vom Typ 2 weisen ein wesentlich niedrigeres Sicherheitsniveau auf. Deshalb müssen diese sehr stark von der Infrastruktur der Pflichtanwendungen getrennt werden. Mehrwertanwendungen vom Typ 4 haben vergleichbare Sicherheitsanforderungen wie die Dienste der Pflichtanwendungen. Deshalb werden diese in die Infrastruktur der Pflichtanwendungen integriert. Die einzelnen Komponenten der Telematikinfrastruktur werden in Schichten, sogenannte Tiers, untergliedert. Es existiert die „Service Consumer Tier“ auf der Seite des Client. Über einen Adapter ist diese Tier mit der „dezentralen Telematik Tier“ verbunden. Die Kommunikation findet dann über eine „zentrale Telematik Tier“ statt. Mittels eines Adapters kann diese Tier wiederum mit der „Service Provider Tier“ kommunizieren. Die „Legacy and Existing Application Tier“ beherbergt bereits im Netzwerk vorhandene Anwendungen. Der Connector Service stellt die Verbindung zwischen den Primärsystemen der Leistungserbringer, welche sich innerhalb der „Service Consumer Tier“ befinden, und der Telematikinfrastruktur her.

2.2.4 Sicherheitsanforderungen der EGK

Einen besonderen Stellenwert beim Entwurf der Infrastruktur der EGK hat das Sicherheitskonzept. Diverse Schutzziele müssen zwingend realisiert werden. Als erste und wichtigste Regel gilt, dass nur die Personen Zugriff auf die Daten ei-

nes Patienten bekommen dürfen, die der Patient explizit dazu berechtigt hat. Jegliche Daten, die entweder online oder direkt auf der EGK gespeichert wurden, müssen verschlüsselt werden. Die Daten, welche online gespeichert werden, müssen hybrid verschlüsselt werden. Mit einem symmetrischen Schlüssel werden die Daten chiffriert. Dieser Schlüssel wird wiederum mit dem öffentlichen Schlüssel auf der EGK verschlüsselt. Personen haben Zugriff auf Daten eines Patienten nur innerhalb eines bestimmten Zeitraumes. Um die Daten auch nach dem Verlust der EGK nicht zu verlieren, muss jedoch ein Umschlüsseln der Daten durch einen vertrauenswürdigen Service möglich sein. Der Patient muss die Möglichkeit bekommen, einzelnen Leistungserbringern Zugriff auf seine Daten zu gewähren. Diese Leistungserbringer müssen auch in Abwesenheit des Patienten Zugriff auf dessen Daten erhalten.

Die Sicherheitsarchitektur der EGK richtet sich nach Standards von Architekturen aus dem Bereich von E-Goverment-Anwendungen. Diese orientieren sich ihrerseits an der International Norm ISO 7498-2, Security Architecture. Für medizinische Informationssysteme gilt im allgemeinen, Patientendaten immer nur einem klar definierten Nutzerkreis zugänglich zu machen. Dieses Rechtemanagement setzt eine Authentifizierung voraus. Diese ist die Basis für eine Autorisierung entsprechend dem Rechtemanagement. Der nächste Aspekt des Sicherheitskonzeptes betrifft die Integrität der Daten. Der Ersteller muss eindeutig feststellbar sein und die Daten müssen nachweislich unverändert weitergegeben werden. Dabei kann der Ersteller eine Person oder auch ein Gerät sein, welches einen Messwert aufgenommen hat. Weiter muss sichergestellt sein, dass ein Empfänger das angeforderte Dokument erhalten hat. Weder der Sender darf glaubhaft abstreiten können, dass er das Dokument gesendet hat, noch der Empfänger, dass er das Dokument erhalten hat. Die Bearbeitungsschritte eines Dokuments müssen revisionssicher nachvollziehbar sein. Defizite in diesem Bereich können im Fall eines Prozesses vor Gericht zur Beweislastumkehr zu Gunsten des Patienten führen. Eine solche Dokumentation ist in erster Linie Aufgabe der Primärsysteme. Die EGK-Infrastruktur darf diese Primärsysteme jedoch diesbezüglich nicht beeinträchtigen.

Bei der Verarbeitung von Gesundheitsdaten agiert jeweils ein Datenbearbeiter, der von einer Person, welche die Datenhoheit über die Daten besitzt, autorisiert wurde. Diese Person ist meist der Patient. Um die Datenverarbeitung abzubilden, wird in der Infrastruktur der EGK ein hybrides rollen- und identitätsspezifisches Rechtekonzept verwendet. Die Autorisierung der Datenbearbeiter geschieht durch ihre Rolle, während die Autorisierung für ein spezielles Datenobjekt durch die Identität des Versicherten geschieht. Dieses Rechtekonzept zieht sich durch alle Tiers. Zwei Ausnahmen bilden der Versichertenstammdatendienst und das sogenannte „Card Management System“. Die Kommunikation über das Internet erfolgt über IPSec-Verschlüsselung. Da das für die Telematikinfrastruktur der EGK entworfene IPSec-Netz keine durchgängige Ende-zu-Ende-Verschlüsselung implementiert, wird auf der OSI-Schicht 4 ein Integritätsschutz durch Transport Layer Security (TLS) und Secure Socket Layer (SSL) gewährleistet. Es findet dabei sowohl vom Server als auch von Client eine Authentifizierung statt. Mit Hilfe von Webservice-Security werden Nachrichten von bestimmten Teilnehmern signiert. Ein Request muss mindestens durch die Datenautorität signiert werden. Das ist in der Regel der Patient. Wenn er keine Zustimmung gibt, kann niemand die Daten abrufen. Wenn der Datenbearbeiter nicht identisch mit der Datenautorität ist, dann muss der Request

auch durch den Datenbearbeiter signiert werden. Der Response wird von dem entsprechenden Service signiert, um die Unveränderheit der Daten zu gewährleisten. Rollenbasierte Policies werden in der Beschreibungssprache eXtensible Access Control Markup Language (XACML) definiert.

Die Zugriffe auf Patientendaten müssen protokolliert werden. Dies wird durch einen Datenzugriffsauditservice realisiert. Zugriffe auf Services und Objekte bekommen Teilnehmer, die nicht im Besitz der EGK sind, mittels eines Tickets erteilt. Um eine Liste verfügbarer Daten zu erhalten, benötigt ein Teilnehmer ein Service-Ticket. Will der Teilnehmer dann Daten abrufen, muss er ein Objekt-Ticket vorweisen, welches ihn zum Zugriff auf die spezifischen Daten berechtigt. Die Integrität der Daten wird nach dem Prinzip eines Bussystems sichergestellt. Jeder Knoten muss selbst entscheiden, in welchem Arbeitsschritt Daten festgeschrieben werden können. Wenn ein Verarbeitungsprozess fehlschlägt, muss ein Teilnehmer selbstständig die entsprechenden Änderungen rückgängig machen. Damit dies möglich ist, müssen schreibende Zugriffe auf Datenobjekte zuletzt durchgeführt werden („write last“). Die Suche nach Services kann mittels eines Dienstes namens „Service Discovery Service“ erfolgen. Technische Grundlage dieses Dienstes ist der UDDI v2 Standard. Da die Services statische Schnittstellen zur Verfügung stellen, wurden die dynamischen Bestandteile von Universal Description, Discovery and Integration (UDDI) in Service Discovery Service (SDS) nicht implementiert. Die EGK sieht verschiedene Arten der Kommunikation vor. Während des Basis-Rollouts wird ausschließlich direkt auf die Daten der Karte zugegriffen. Später sollen dann noch die Kommunikationsmuster Card-to-Card, Webservice und Card-to-Server dazukommen. Dabei ist der direkte Zugriff eine Ausnahme, die nötig ist, da zum Zeitpunkt des Rollouts noch keine Konnektoren zur Verfügung stehen. Bei der direkten Kommunikation greift das Primärsystem einer Einrichtung über ein Kartenterminal direkt auf die Karte zu. Dabei können nur die Versicherungsdaten aus dem ungeschützten Bereich der Karte ausgelesen werden. Die Schnittstellen zwischen Diensten werden mit Hilfe von XML beschrieben.

2.2.5 Zusammenfassung zur EGK

Zusammenfassend kann die Elektronische Gesundheitskarte als Schlüssel für eine Infrastruktur betrachtet werden, welche die Gematik spezifiziert. Diese Spezifikation definiert Anforderungen an mögliche Dienste und gibt Standards und Lösungsansätze vor. In der ersten Ausbaustufe soll die Gesundheitskarte lediglich als Speicher für die Versicherungsdaten benutzt werden. Allerdings sind beim geplanten Rollout im Jahr 2006 Verzögerungen aufgetreten, die dazu geführt haben, dass die Gesundheitskarte frühestens 2010 flächendeckend eingesetzt werden kann. Anschließend muss die Infrastruktur aufgebaut werden. Durch die Fülle an Anforderungen und die Integration verschiedenster Standards und Kommunikationsmodelle wird der Aufbau der EGK-Infrastruktur sehr aufwändig und nur in ferner Zukunft realisiert werden können. Geplant sind zentrale Server, die als elektronische Patientenakte fungieren und auf welche von allen Knoten zugegriffen werden kann. Die Erfahrungen in Österreich haben gezeigt, dass eine solche sehr zentrale Lösung schnell überlastet sein kann. Da die EGK-Infrastruktur um ein vielfaches größer als die Österreichs ist, muss mit Engpässen gerechnet werden. Darüber hinaus sind Instanzen und Knoten, die Gesundheitsdaten verschiedener Einrichtungen sammeln, aus datenschutz-

rechtlicher Sicht bedenklich. Die Einführung der EGK wird nicht nur durch offene Finanzierungsfragen behindert. Auch der Umstand, dass spezielle Hardware entwickelt und von den Einrichtungen gekauft werden muss und Primärsysteme von den Softwareherstellern angepasst werden müssen, erschweren ihre Einführung. Darüber hinaus stellt die Variantenvielfalt der von der Gematik entworfenen Architektur ein Hindernis bei der Umsetzung dar. Durch die beschriebenen Probleme wurden bei der Einführung viele Kompromisse in Sachen Datensicherheit eingegangen. Dieser Umstand wird die spätere Einführung einer EGK-Infrastruktur und der dazugehörigen Dienste erschweren. Das in dieser Arbeit entworfene Konzept geht im Gegensatz zur EGK-Infrastruktur davon aus, dass Gesundheitsdaten immer zuerst dort gespeichert werden, wo sie angefallen sind. Nur im Bedarfsfall können diese Daten von anderen Knoten lokalisiert und angefordert werden. Weiter wird die Infrastruktur nicht in das Primärsystem einer Einrichtung integriert, sondern über Schnittstellen angebunden. Die EGK kann wie jede andere Smartcard als Schlüssel für die in dieser Arbeit entworfene Infrastruktur verwendet werden.

2.3 Verteilte Systeme und SOA

Die Abkürzung SOA steht für Serviceorientierte Architektur. [BSI09] Die Entwicklung der Netzwerktechnologie und des Internets hat es ermöglicht, komplexe Anwendungen auf verschiedene Rechner zu verteilen. Solche Anwendungen stellen Softwareentwickler vor ganz neue Herausforderungen, wie die Synchronisation von Speicherbereichen verschiedener Knoten oder den Umgang mit Verbindungsabbrüchen und Latenzzeiten. Um eine Basis für die Implementation verteilter Anwendungen zu schaffen, wurden zahlreiche Paradigmen entworfen und viele Softwarebibliotheken entwickelt. Eines dieser Paradigmen ist die Serviceorientierte Architektur. Dabei handelt es sich um ein Architekturkonzept, welches entwickelt wurde, um unterschiedliche Teilaufgaben einer Anwendung in Dienste mit klar definierten Schnittstellen zu kapseln. Diese Dienste können dann zu komplexeren Anwendungen zusammengesetzt werden. Die Organization for the Advancement of Structured Information Standards (OASIS) definiert SOA wie folgt: „SOA ist ein Paradigma für die Strukturierung und Nutzung verteilter Funktionalität, die von unterschiedlichen Besitzern verantwortet wird.“ [LEMea06] Anwendungen, die nach dem serviceorientierten Paradigma entwickelt wurden, können unterschiedliche Technologien, wie beispielsweise Corba, verwenden. Meist kommen jedoch Webservices zum Einsatz, welche über das HTTP-Protokoll XML-Dateien übertragen. In den folgenden Abschnitten sollen unterschiedliche Middleware-Systeme vorgestellt werden, mit deren Hilfe serviceorientierte Anwendungen erstellt werden können. Weiter werden diese Systeme auf ihre Eignung zur Lösung der in dieser Arbeit diskutierten Aufgabenstellung bewertet.

2.3.1 Ruby on Rails

Ruby on Rails ist ein von David Heinemeier Hansson in Ruby geschriebenes Web-Framework. Es basiert auf dem Prinzip „Don’t Repeat Yourself“ (DRY) und beinhaltet Konventionen über die Anwendungskonfiguration („Convention over Configuration“). Es unterstützt damit eine zügige Entwicklung von leistungsfähigen Anwendungen. Die eigentliche Programmlogik übernimmt ein

Controller, dessen öffentliche Methoden aufgerufen werden können. Die Ergebnisse werden von einer View-Komponente ausgegeben, die mittels Templates gesteuert werden kann. So erreicht man die Trennung von Datenbank, Anwendungslogik und Darstellung.

Rails ist nach der Model View Controller Architektur entwickelt worden. Es besteht aus fünf Modulen, von denen Action Support, eine Ruby-Erweiterung von Rails, das Erste ist. Das zweite Modul namens Active Record bildet die Objektabstraktionsschicht, welche auf dem objektrelationalen Muster basiert. Dieses Modul kapselt relationale Datenbankstrukturen in Objekte und ermöglicht so den Zugriff auf Datenbankstrukturen ohne die Verwendung von SQL. Dadurch können völlig transparent verschiedene Datenbanklaufzeitsysteme verwendet werden, ohne den Programmcode zu modifizieren. Datenbanktabellen können mittels eines „foreign key“ miteinander verknüpft sein. Ist das der Fall, so wird in einem Objekt vom Typ Active Record ein Array hinterlegt, in dem alle dazugehörigen Objekte der verknüpften Tabelle abgelegt sind. Auf diese Art und Weise können verschachtelte Objekte entstehen, die auf eine objektrelationale Struktur abgebildet werden. Das dritte Modul nennt sich Action Pack und ist für die Requestbehandlung und die Responseausgabe verantwortlich. Anfragen werden durch öffentliche Methoden des Controllers behandelt. Action Mailer - das vierte Modul ist für den Versand und Empfang von E-Mails zuständig. Das fünfte Modul zur Erzeugung von Webservices ist Action Webservice. Mit Hilfe dieser Bibliothek erstellte Services können mittels SOAP oder XML-Remote Procedure Call (XML-RPC) aufgerufen werden.

Ab Version 2.0 ist Representational State Transfer (REST) direkt in ActionRessource integriert. Dabei handelt es sich um einen Softwarearchitekturstil, welcher davon ausgeht, dass die meisten Anwendungen mit einer eingeschränkten Anzahl von Methoden abzubilden sind. REST wurde in einer Dissertation von Roy Fielding im Jahr 2000 erstmals beschrieben. Darin wird jeder Ressource eine eigene URL zugeordnet. Man geht davon aus, dass für jede dieser Ressourcen standardisierte Methoden, wie „GET“, „PUT“ oder „DELETE“ existieren. Damit passt REST gut in das Konzept von Rails. Ruby on Rails bietet darüber hinaus eine Reihe von Werkzeugen, welche das Erzeugen einer webbasierten Datenbankanwendung beschleunigen. Ein Beispiel dafür ist das sogenannte Scaffolding. Durch mitgelieferter Skripte kann eine einfache Weboberfläche erzeugt werden, mit deren Hilfe man die Daten der dazugehörigen Datenbank modifizieren kann. So werden Funktionen zum Einfügen, Lesen, Verändern und zum Löschen von Ressourcen erzeugt. Dieser Funktionsumfang wird mit der Abkürzung CRUD beschrieben, welche für die Worte Create, Read, Update und Delete steht. Das Scaffolding kann auch im Zusammenhang mit Webservices verwendet werden. Rails generiert automatisch Formulare, deren Inhalte mittels XML-RPC oder SOAP an den Webservice übergeben werden. Dies ist in erster Linie für die Fehlersuche und den Test hilfreich.

Mit Hilfe der Migrationswerkzeuge kann ein vorhandenes Datenbankschema exportiert oder ein neues Schema definiert werden. Mittels der Migrationswerkzeuge ist es somit möglich, ein datenbankunabhängiges Datenbankschema anzulegen. Rails legt in der Datenbank die Tabelle schema_info an, in der es Versionsinformationen des aktuellen Datenbankschemas speichert. Das ist nützlich bei Updates eines vorhandenen Datenbankschemas. Mit Hilfe der Versionsinfos kann Rails ein Rollback des Datenbankschemas durchführen. Eine wichtige Fähigkeit von Rails ist der Model-Callback. Diese Funktionalität ermöglicht es,

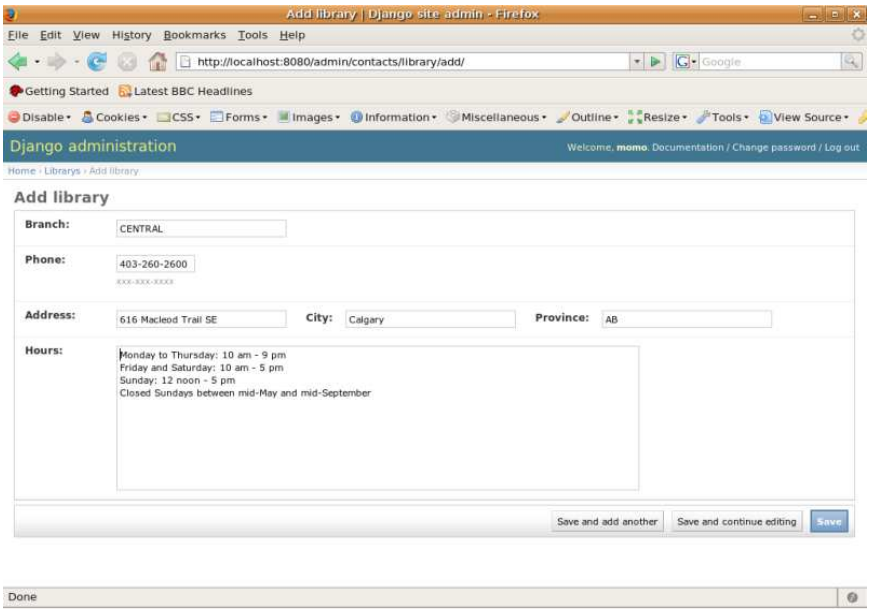

Abbildung 5: Admin-Interface von Django

bei bestimmten Zustandsänderungen des Datenbankmodells Methoden aufzurufen. Wird zum Beispiel ein Datensatz gelöscht, so kann eine Funktion aufgerufen werden, die einen anderen dazugehörigen Datensatz löscht. Im konkreten Fall der Infrastruktur im Gesundheitswesen ermöglicht es diese Funktionalität, Aktionen auszulösen, falls ein untergeordnetes System Änderungen an der Datenbank vornimmt. Ruby on Rails eignet sich zur zügigen Implementierung von Webanwendungen und bietet komfortable Möglichkeiten zur Entwicklung von Webservices nach dem serviceorientierten Paradigma.

2.3.2 Django

Django ist ein in Python implementiertes Web Development Framework. Ähnlich Rails ermöglicht es eine schnelle dynamische Entwicklung von Webanwendungen. Es wurde nach dem DRY Paradigma entwickelt und bietet im wesentlichen die Funktionalität von Rails. Im Unterschied dazu unterstützt es jedoch nur drei Datenbankmanagementsysteme - SQLite, MySQL und Postgres. Die entwickelten Oberflächen werden bei Django mit Hilfe des Apache Webservers veröffentlicht, wozu Apache mit dem Modul mod_python ausgerüstet sein muss. Mit Django ist es auch möglich, Webservices zu implementieren. Ähnlich dem Scaffolding bei Rails bietet Django ein dynamisches Admin-Interface. Mit wenigen Zeilen Code stellt Django eine Schnittstelle zur Verfügung, in welcher auf die Tabellen der Datenbank zugegriffen werden kann und verschiedene Konfigurationen durchgeführt werden können. Abbildung 5 zeigt den Screenshot einer solchen dynamisch generierten Administrationsoberfläche. Ein berechtigter Benutzer kann sich am Interface authentifizieren und anschließend z.B. Nutzer, Gruppen und Rechte verwalten. [New05] Mit Apache und mod_python ist Django etwas performanter als der in Rails integrierte Webserver Webrick. Je-

doch besteht die Möglichkeit, auch Rails mit Apache laufen zu lassen. In diesem Fall ist kein Unterschied bei der Performance beider Frameworks feststellbar. Mit Hilfe eines Moduls URLconf können in Django URLs bestimmten Python-Funktionen zugeordnet werden. Dadurch entstehen gut lesbare URLs ohne lästige Parameterangaben. Django ist insgesamt vergleichbar mit Ruby on Rails. Es unterstützt allerdings nur wenige Datenbankmanagementsysteme. Ein weiterer Nachteil von Django ist seine geringe Verbreitung, weshalb nicht so viele Informationen über das Framework verfügbar sind, wie beispielsweise für Ruby on Rails.

2.3.3 BPEL - Business Process Execution Language

BPEL ist eine Beschreibungssprache zur Strukturierung von Webservices. Mit ihrer Hilfe ist es möglich, verschiedene Webservices unterschiedlicher Systeme zu einem komplexen Geschäftsprozess zu verknüpfen. BPEL ist keine Programmiersprache um Anwendungen zu implementieren, sondern ist für das Programmieren im Großen gedacht (programming in the large). Es existieren bei der Beschreibung von Webservices in BPEL zwei verschiedene Paradigmen. Zum einen existiert die Beschreibung eines Prozesses mit BPEL Executable Processes. Dabei geht BPEL vom Ansatz der Orchestrierung aus, d.h. eine zentrale Komponente steuert den Ablauf des Geschäftsprozesses. Dieses Paradigma eignet sich für die Steuerung von Prozessen im Gesundheitswesen wenig, da die Teilnehmer am Behandlungsprozess vollkommen unabhängige Instanzen und Einrichtungen sind. Zum anderen unterstützt BPEL auch ein anderes Paradigma, nämlich das der Choreographie von Webservices. Dafür eignet sich die Beschreibung der Prozesse in Form von BPEL Abstract Processes. Hierfür existieren jedoch auch Beschreibungssprachen, welche explizit diesen Ansatz verfolgen.

2.3.4 WS-CDL - Web Service Choreography Description Language

Im Gegensatz zu BPEL Executable Processes wurde WS-CDL nach dem Paradigma der Choreographie von Webservices entwickelt. Anders als bei der Orchestrierung von Webservices beschreibt die Choreographie das Zusammenwirken unabhängiger Instanzen, welche im Zusammenhang mit WS-CDL Dancer genannt werden. Diese Dancer können somit unabhängige Firmen sein, die zusammenarbeiten. Ein klassisches Beispiel dafür ist das Online-Reisebüro, welches seinen Kunden über ein Onlineportal Flüge und Urlaubsreisen anbietet. Die Preise werden dynamisch über das Aufrufen von Webservices der einzelnen Fluggesellschaften ermittelt und im Portal angezeigt. Bucht ein Kunde eine Reise, so kann die Buchung der Flüge und Hotels wiederum über Webservices erfolgen. Ein solches Szenario ist in Abbildung 6 dargestellt. Kunden können über einen Travel Agent Reisen buchen. Dieser ruft Webservices der verschiedenen Dancer auf und stellt so die Bestandteile der Reise zusammen. WS-CDL beschreibt dabei die von außen beobachtbaren Kommunikationsvorgänge. Alle beteiligten Instanzen verfolgen parallel den Verlauf des Workflow und beeinflussen dessen strukturierte Abarbeitung. Ein in WS-CDL definierter Geschäftsprozess ist kein ausführbares Programm, sondern eine deklarative Beschreibung von Interaktionsmustern. [Bra08] [ZCXH05] Das W3C-Consortium hat die Anforderungen an ein Modell definiert, welches in WS-CDL weitestgehend umgesetzt worden ist. Ein wichtiges Ziel von SOA allgemein und somit auch eines Choreographie-

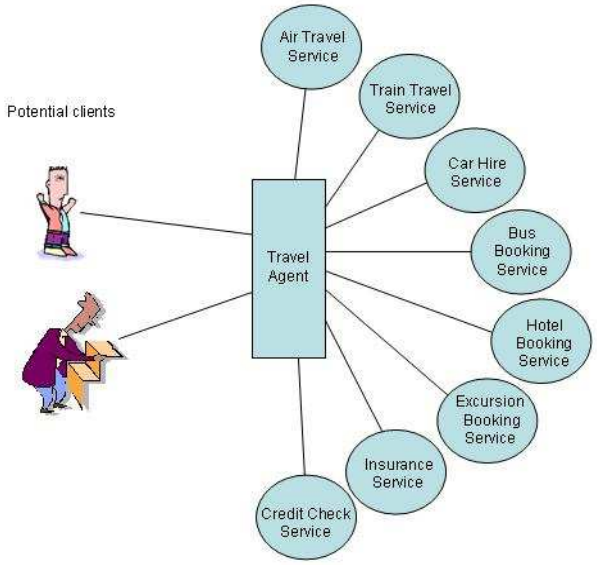

Abbildung 6: Choreographie von Webservices verschiedener Dancer

modells ist die Wiederverwendbarkeit. Eine Choreographie muss in verschiedenen Kontexten verwendbar sein und soll den Nachrichtenaustausch von zwei oder mehr vollkommen unabhängigen Teilnehmern oder Prozessen beschreiben. Wie die Orchestrierung mit BPEL ist auch das Erstellen eines Geschäftsprozesses mit WS-CDL ebenso aufwändig und zeitraubend. Für die Entwicklung von Abläufen im Gesundheitswesen ist eine schnelle und einfache Implementation von Prozessen nötig, da durch sich ständig ändernde Vorgaben, eine flexible Anpassung jederzeit möglich sein muss. Ein Ansatz, um diese Implementierung von Geschäftsprozessen mit Hilfe von Choreographie anwenderfreundlich zu realisieren, ist die Gruppierung von IHE-Profilen. (Integrated Healthcare Enterprise) [DKTea08] Diese Profile sind mit BPEL realisiert und werden vordefiniert. Wichtig ist, dass die gleichen Profile auf allen beteiligten Knoten verfügbar sind. Während die Gruppierung mit einem graphischen Werkzeug realisiert werden kann, sind für die Erstellung des Profiles BPEL-Kenntnisse erforderlich. [BK04]

2.4 Grid-Architekturen

Eine andere Art verteilter Anwendungen sind Grid-Architekturen. [ABK+04] Dahinter steckt die Idee, Ressourcen wie Speicherplatz, Rechenzeit oder Sensoren, anderen Knoten innerhalb eines Weitverkehrsnetzes zur Verfügung zu stellen. Der Name orientiert sich am Energienetz, an welches sich jeder anschließen und Energie daraus beziehen kann. Das Ziel des Grid-Computing ist es, Rechenleistung und Speicherplatz ähnlich der Elektroenergie des Stromnetzes zur Verfügung zu stellen. Jeder, der Ressourcen benötigt, greift darauf zu. In den folgenden Abschnitten werden Middleware-Systeme vorgestellt, die der

Entwicklung von Grid-Anwendungen dienen. Ziel dabei ist es, den Ansatz der Grid-Architektur, unter dem Gesichtspunkt einer Verwendung als Basis, für die Kommunikation in einer generischen Infrastruktur im Gesundheitswesen zu betrachten.

2.4.1 Unicore

Unicore [Hub] ist ein Open Source Projekt. Seine Architektur läßt sich in drei Schichten untergliedern. Die erste Schicht ist die Clientschicht. Es existiert eine Clientsoftware, die man sich testweise downloaden kann. Die zweite Schicht ist die des Servers. Ein Server stellt die Schnittstelle zwischen Clients und der dritten Schicht, dem Zielsystem, dar. Dieses Zielsystem stellt Ressourcen zur Verfügung, die von Clients genutzt werden können. Die Client-Seite besteht aus zwei Komponenten, dem Job Preparing Agent (JPA) und der Job Monitor Component (JMC). Der JPA ist dazu da, einen Job zu erstellen, während man diesen mit dem JMC überwachen kann. Jobs werden mittels der Abstract Java Object - Klassen definiert und überwacht. Die Seite des Service-Anbieters besteht ebenfalls aus zwei Teilen, dem Network Job Supervisor (NJS) und dem Target System Interface (TSI). Der NJS kontrolliert alle Jobs, verwaltet die Zugriffsrechte der Nutzer und wertet Zertifikate aus. Das NJS sendet Jobs an das TSI. Dieses ordnet die Jobs in seine Warteschlange ein. Unicore unterstützt ein Single Sign On mittels X.509V3 Zertifikaten. In etlichen Projekten findet Unicore seine Anwendung, so z.B. im BioGrid, im MeteoGrid oder im CAE-Grid.

2.4.2 GridBus

GridBus ist ähnlich wie Globus ein modular aufgebautes System. Es steht unter einer Open Source Lizenz. Das Besondere an GridBus ist die Abstraktionsschicht, die es ermöglicht, mit anderen Middleware-Systemen zusammenzuarbeiten. GridBus kann so Unicore und Globus-Komponenten vereinen. Hinter GridBus steckt die Idee einer computational economy - d.h. ein Client muss für eine Ressource zahlen. Natürlich geschieht diese Bezahlung nicht in Euro, sondern mit Ressourcen, die ein Client seinerseits wieder bereitstellt, oder mit bestimmten Rechten. So kann ein Knoten nur so viele Jobs an andere Knoten schicken, wie es sein Budget zulässt.

2.4.3 Legion

Legion [NNTHG01] ist kein Open Source Projekt. Es existiert eine kommerzielle Version. Die Idee dahinter ist, verschiedene Prozessorarchitekturen, Betriebssystem-Plattformen und Speichermedien über eine Netzwerkverbindung zu einem virtuellen Rechner zusammenzufassen. Legion ist aus verteilten Objekten aufgebaut, deren Methoden über einen „remote method invocation service“ aufgerufen werden können. Alle Software- und Hardwarekomponenten werden durch Legion - Objekte repräsentiert. Das Objektmodell von Legion ist mittels einer IDL (Interface Definition Language) beschrieben. Diese Objekte können mit anderen Objekten so kommunizieren, als ob sie auf derselben Maschine mit dem gleichen Kontext liefen. Da Legion das Modell eines virtuellen Rechners implementiert, kann man verschiedene Standardobjekte bestimmten Bestandteilen eines Rechners zuordnen. Host Objekte repräsentieren z.B. Prozessoren,

während Vault Objekte persistenten Speicher darstellen. Die Objekte werden mittels eines dreistufigen Namenssystems adressiert. Zum Ersten die Kontext Namen. Diese bestehen aus lesbaren Zeichenketten. Diese Namen sind eindeutig einem zweiten Namen, dem Legion Object Identifier (LOI), zugeordnet. Ein Binding Agent wandelt diesen Identifier in die Legion Object Address (LOA) um, welche die dritte Stufe des Namenssystems darstellt. Legion stellt eine API für C++ zur Verfügung. Legion benutzt für die Verwaltung von Dateien ein verteiltes Filesystem namens LegionFS. [aMWHG] Damit ist es möglich, ein Dateisystem über ein Weitverkehrsnetz, verschiedenen Knoten einer Gridinfrastruktur zur Verfügung zu stellen. Projekte, die Legion als Middleware nutzen, sind z.B. Nimrod-G oder Centurion.

2.4.4 Das Globus Toolkit

[For02] [FKT01] Das Globus Toolkit ist eine Implementation, der in der Open Grid Services Architecture (OGSA) festgelegten Standards. Dieser Standard wurde geschaffen, um verteilte Ressourcen logisch zu vereinigen. Dabei können diese aus Festplattenspeicher, Daten, Netzwerken, Sensoren oder Rechenleistung bestehen. [DFK+03] Das Toolkit ist eine Sammlung von Werkzeugen, die aus konkreten Projekten herausgelöst wurden, um sie bei anderen Fragestellungen einzusetzen. Es sollte also eine Sammlung von Werkzeugen entstehen, die ganz allgemein für die Realisierung verteilter Anwendungen nötig ist.[Sot05] Globus stellt Container bereit, welche mit einer Anwendung gefüllt werden können. Dabei muss die Anwendung nicht lokal auf jedem Knoten des Grid installiert werde. Globus übernimmt die Verteilung von Services. Diese Eigenschaft macht ein solches Grid flexibel. Man kann Services in ein komplexes System einpflegen, die man beim Entwurf noch nicht berücksichtigen konnte. Ein Service kann nach Bedarf von einem Client angefordert werden. In diesem Fall wendet der Client sich an einen sog. Factory-Service. Dieser erzeugt eine Instanz des gewünschten Service. Jetzt kann der Client mit dem Service kommunizieren. Das Globus Toolkit bringt eine Reihe von Services mit, die es ermöglichen, das Grid zu managen. Für die spezifischen Probleme, welche man mit einem Grid bewältigen will, kann man nun eigene Services definieren. Es existiert eine API für Java, C und Python. Dabei ist eine Entwicklung in Java am komfortabelsten [CS06], ein Service in C am performantesten und ein Service in Python am klarsten strukturiert. Seit Version vier basiert die Kommunikation des Globus Toolkit auf Webservices. Der Vorteil der Verwendung von Webservices besteht darin, dass dafür sehr verbreitete Protokolle wie HTTP verwendet werden. Der Globus Toolkit soll in der Lage sein, möglichst viele Knoten zu verbinden. Durch die Verwendung von HTTP ist dies möglich. Da HTTP und damit auch die Verwendung von Webservices verbindungslos sind, lässt sich ein Grid nicht ausschließlich mit Hilfe von Webservices abbilden. Das heißt, zwischen einem Aufruf eines Service und dem nächsten Aufruf werden keine Statusinformationen gespeichert. Um Statusinformationen der Webservices zu speichern wird das Web Service Resource Framework verwendet. Dieses von der OASIS entwickelte Framework verleiht Webservices Statusinformationen. Diese Informationen können aus einer URL, aber auch aus komplexen XML-Dokumenten bestehen. Im Request des Clients wird eine Ressource adressiert, die den Status des Webservice darstellt. Der Globus Toolkit unterstützt auch die Replikation von Datenbeständen und die Suche nach Replikaten. [VTF01] Damit ist es möglich, Daten redundant in verschiede-

nen Knoten zu speichern und im Bedarfsfall das Replikat zu finden, welches am schnellsten übertragen werden kann. Der Globus Toolkit verfügt außerdem über ein Sicherheitskonzept [BF06], das eine sichere Kommunikation, Authentizität und Integrität gewährleistet. Das ist eine wichtige Vorraussetzung für den Aufbau virtueller Organisationen. Darüber hinaus ermöglicht der Globus Toolkit die Delegation von Rechten. [Glo05]

2.5 Peer-to-Peer-Architekturen

Eine andere Architektur von verteilten Systemen ist die Peer-to-Peer-Architektur. Im Gegensatz zu einer Client-Server Architektur sind die Peers einer P2P-Anwendung gleichberechtigt. P2P-Architekturen dienen in erster Linie dem Austausch von Dateien und deren Lokalisation, ohne dass ein zentraler Knoten vorhanden sein muss. [MRHea09] Die Hauptmotivation für die Entwicklung leistungsstarker P2P-Algorithmen war der illegale Austausch von geistigem Eigentum wie Musikdateien oder Videos. Da diese Raubkopien nicht auf einem öffentlichen Server abgelegt werden durften, wurden Technologien entwickelt, die es erlaubten, solche illegalen Daten direkt von Client zu Client zu übertragen. Dabei bieten P2P-Netze erhebliche Vorteile. Durch den Verzicht auf zentrale Komponenten kann der Ausfall eines Peer kompensiert werden. In den folgenden Abschnitten werden drei Ansätze von P2P-Software dargestellt und deren Eignung für eine Suche innerhalb der Knoten, einer Infrastruktur im Gesundheitswesen, beleuchtet. [DP04]

2.5.1 Freenet

[CSWH01] Freenet ist ein P2P-System, welches Teilnehmern erlaubt, anonym Daten in das System zu laden und anderen zur Verfügung zu stellen. Es eignet sich auch für Instant Messaging und zum Versenden von Nachrichten ähnlich einem Email-System. Freenet geht auf die von Ian Clarke 1999 veröffentlichte Abhandlung unter dem Titel „Distributed Anonymous Information Storage and Retrieval System“ zurück. Nach dessen Veröffentlichung entstand ein Open Source Projekt, welches als Ziel hatte, ein anonymes verteiltes Datenspeichersystem zu entwickeln. Dieses sollte mit bestimmten Suchfunktionalitäten ausgestattet sein. Freenet hat sich indes stark verbreitet. Es handelt sich dabei um ein reines P2P System. Trotzdem werden bei Suchen keine Nachrichten an alle Knoten gesendet. Das System lernt also selbst dazu. Wird ein Peer an das Netz angeschlossen, muss lokal eine Software installiert und mit mindestens einem anderen Knoten ein Kontakt hergestellt werden. Dieser Knoten wird Seednode genannt. Bei der ersten Anfrage wird diese selbstverständlich an diesen Seednode geleitet. Über dieses Peer entstehen Kontakte mit anderen Peers. Die Dateinamen werden mit der 160 bit SHA-1 Funktion gehasht, wodurch ein Schlüssel, der sogenannte content-hash-key (CHK), entsteht, der eindeutig auf die Datei verweist. Es existiert ein zweiter Schlüssel namens Signed-Subspace Key (SSK). Mit dessen Hilfe kann sich ein Teilnehmer einen Namensraum schaffen, in welchem nur er Dokumente ablegen darf. Mittels eines asymmetrischen Kryptoverfahrens wird ein Namensraum angelegt, in welchem jeder lesen, aber nur der Besitzer schreiben kann. Dazu wird ein zufälliges Schlüsselpaar erzeugt. Anschließend wird die Beschreibung des Namensraumes und der öffentliche Teil des SSK unabhängig voneinander gehasht, dann verkettet und die Verkettung

Abbildung 7: Verteilung der Anfragen und Schüssel bei Freenet

ein weiteres Mal gehasht. Auf diese Art und Weise wird eine Beziehung zwischen dem Namensraum und dem öffentlichen Schlüssel des Autors hergestellt. Legt der Teilnehmer eine Datei an, so ist im SSK eine Verknüpfung zum CHK der eigentlichen Datei hergestellt. Der SSK wird mit dem geheimen Schlüssel signiert. Über den SSK kann so die CHK der Datei aufgerufen werden. Sucht man eine Datei, benötigt man die Beschreibung des Namensraumes und den öffentlichen Schlüssel, aus denen man sich den SSK erzeugen kann. Mit Hilfe des so gewonnenen Schlüssels fordert man die SSK-Datei an und prüft die Signatur mit Hilfe des öffentlichen Schlüssels. So kann man mit dem öffentlichen Schlüssel Dateien abrufen. Um neue Dateien anzulegen, muss man die SSK-Datei modifizieren, wozu der geheime Schlüssel nötig ist. Eine Änderung von Dateien ist möglich, indem man eine neue Version dieser Datei hochlädt und anschließend den CHK mit dem SSK verknüpft. So bleibt die referenzielle Integrität erhalten. Man findet also unter derselben Beschreibung ein neues Dokument, das vom Besitzer des geheimen Schlüssels verändert wurde. Die Datei wird dann auf einem Peer gespeichert, welcher andere Daten mit ähnlichen Schlüsseln beinhaltet. Dateien werden in den Peers immer verschlüsselt abgelegt. Der Sinn besteht darin, dass ein Teilnehmer nicht darüber entscheiden kann, welche Daten auf seinem Computer abgelegt werden. Da die Dateien verschlüsselt sind, kann ein Teilnehmer glaubhaft bestreiten, dass er von den Daten auf seinem PC Kenntnis besessen hat und ist damit von der Verantwortung für die Inhalte entbunden.

Das Routing einer Suchanfrage funktioniert ohne das Fluten des gesamten Netzes. Eine Anfrage wird nicht an alle benachbarten Knoten gesendet. Die erste Version von Freenet verwendete ein sehr einfaches Routing. Der Grundgedanke davon war folgender: Wenn ein Knoten einmal für die Suche nach einem bestimmten Schlüssel gute Ergebnisse geliefert hat, dann wird er auch in Zukunft für ähnliche Schlüssel gute Ergebnisse liefern. Dieser Routingmechanismus führt nach einiger Zeit zu einer Spezialisierung der Peers. Peers, die für eine Anfrage gut geeignet sind, werden auch zum Speichern von ähnlichen Schlüsseln bevorzugt verwendet. In Abbildung 7 ist links die Verteilung der Schlüssel, die angefragt werden, und rechts die Verteilung der auf dem Peer gespeicherten Schlüssel zu sehen. Die Verteilungen korrelieren. Dieser Routingalgorithmus erwies sich als erstaunlich effizient. Anfragen werden solange von einem Peer zum anderen geleitet, bis der gesuchte Schlüssel gefunden ist oder der sogenannte „Hops to live“-Wert (HTL) erreicht ist. Dieser ist bei Freenet auf 20 voreingestellt. Abbildung 8 zeigt, wie Suchanfragen jeweils an einen Nachbarknoten weitergegeben werden und der HTL-Wert dabei dekrementiert wird. Dabei verändern die Peers die HTL zufällig. Das heißt, eine Suchanfrage liefert ein negatives Ergebnis, falls ca. 20 Knoten durchlaufen wurden, aber der gesuchte Schlüssel nicht gefunden wurde. Ohne HTL würde die Suche nach einem nicht vorhandenen Schlüssel nie

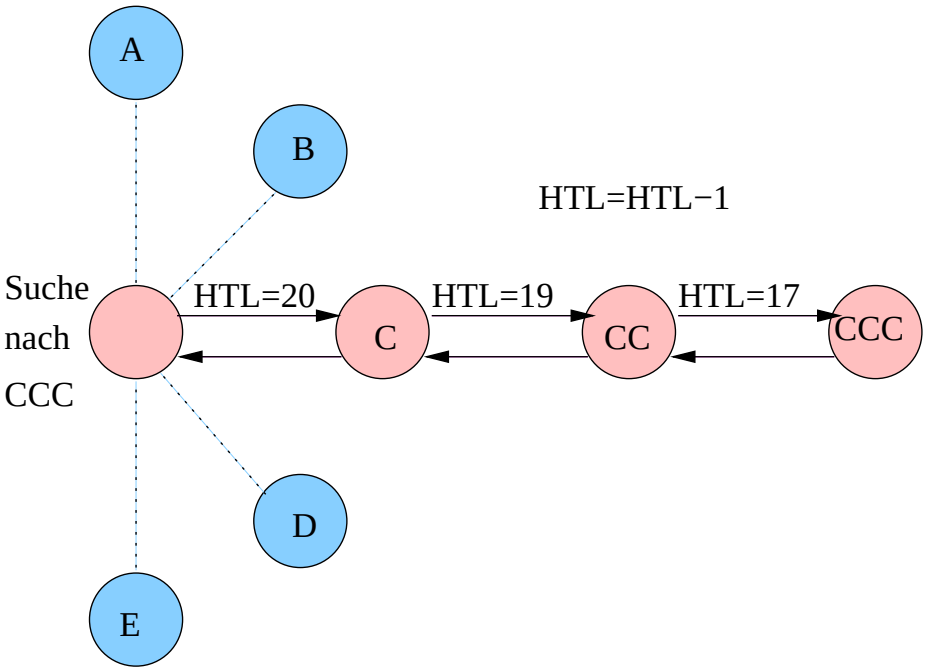

Abbildung 8: Suche in Freenet

enden. Wird der gesuchte Schlüssel gefunden, liefert Freenet die Datei auf dem gleichen Weg zurück, den die Anfrage genommen hat. Dadurch wird sichergestellt, dass der Empfänger nicht sicher wissen kann, von welchem Peer die Datei geschickt wurde. Es ist möglich, dass das Peer, an welches die Anfrage gesendet wurde, die Datei selbst gespeichert hatte. Es kann jedoch ebenfalls sein, dass dieses Peer die Datei wieder von einem anderen Peer bekommen hat. Die Anonymität, die damit erreicht werden soll, ist auch der Grund, weshalb die Peers den HTL-Wert zufällig verändern. Ansonsten könnte ein Peer, welches eine Anfrage mit dem HTL-Wert 19 weiterleitet, im Falle des Erfolges wissen, dass die Datei vom angefragten Peer stammt, da dieses die Anfrage sonst verworfen hätte. Indem die Peers die HTL-Werte verändern, kann ein Peer nie die Quelle einer Datei mit Sicherheit feststellen.

Der Suchalgorithmus von Freenet berücksichtigt jedoch nicht andere Eigenschaften eines Peers, wie die Verfügbarkeit, die Bandbreite, mit welcher das Peer angebunden ist oder die Latenzzeit, die ein Peer zur Beantwortung einer Anfrage benötigt. Deshalb wurde 2005 das Next Generation Routing (NGR) entwickelt. Dieser Algorithmus sammelt in einer Routingtabelle verschiedene Informationen über die Peers und verwendet diese, um eine sogenannte Datareplay-Abschätzung durchzuführen. Das bedeutet, dass der Routingalgorithmus abschätzt, wie viel Zeit ein Knoten benötigt, um die gesuchte Datei zu liefern. Dabei ist wichtig, dass der Routingalgorithmus unbekannte Peers sinnvoll einschätzen kann. Er muss dynamisch auf Veränderungen reagieren können, darf aber nicht sensibel auf Fluktuationen reagieren, die vom Durchschnittswert abweichen. Der Algorithmus muss außerdem scalenfrei sein. Das bedeutet, dass er in der Lage ist, auch ein Peer, das nur für einen kleinen Bereich des Schlüssel-

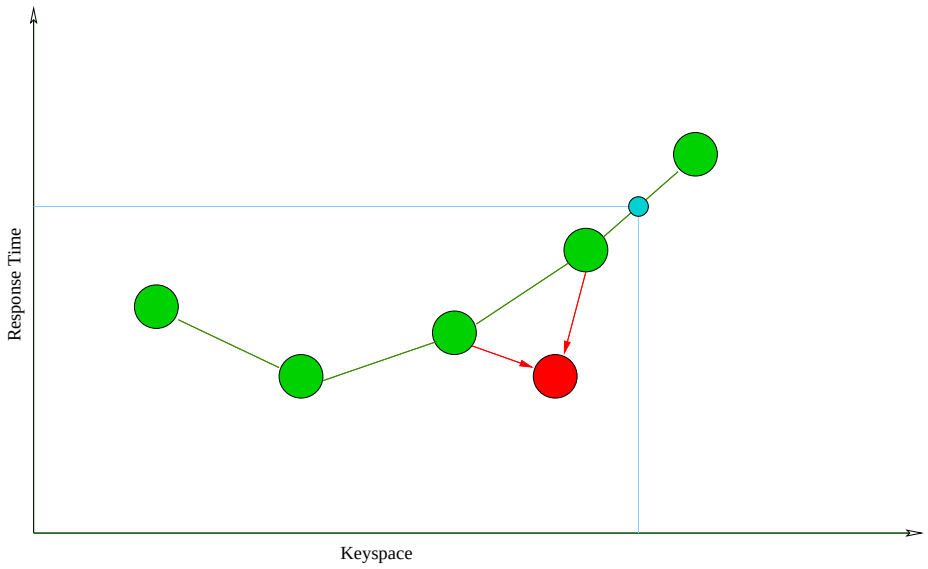

Abbildung 9: NGR - Annäherung der Referenzpunkte an eine Laufzeit

raumes zuständig ist, richtig einzuschätzen. Dieses Verhalten erreicht der Algorithmus, indem er n Referenzpunkte unterhält, die anfangs gleichmäßig auf den Schlüsselbereich verteilt sind. Wird eine Datei gesucht und eine Zeit gemessen, bis diese übertragen wurde, dann werden die im Schlüsselraum benachbarten Referenzpunkte um einen bestimmten Betrag verschoben, wie in Abbildung 9 zu sehen ist. Der Wert dieses Betrages entscheidet, wie schnell das System vergisst. Da die Zeit bis zum Ende der Übertragung einer Datei auch von deren Größe abhängt, wird bei der Abschätzung eine durchschnittliche Dateigröße angenommen. Manche Suchen liefern kein Ergebnis zurück. Wird eine Datei nicht gefunden, so wird eine DataNotFound-Nachricht zurückgeliefert. Diese kann bedeuten, dass entweder die Datei mit dem gesuchten Schlüssel nicht vorhanden ist oder nicht gefunden wurde. Um diese Ereignisse richtig zu approximieren, wird als Anzahl der legitimen DNF-Nachrichten, also derjenigen, die auftreten, wenn keine Daten vorhanden sind, die Anzahl der DNF-Nachrichten des Peers verwendet, der die wenigsten DNF-Nachrichten geliefert hat. Für den Rest der DNFs wird davon ausgegangen, dass die Daten vorhanden sind und nach Erhalt der DNF-Nachricht die Suche über ein anderes Peer fortgesetzt werden muss. Diese approximierten Werte werden nun in die Abschätzung mit einbezogen. Damit die Phase des Anlernens verkürzt wird, kann ein Peer zu Beginn eine Liste von Informationen eines anderen Peers übernehmen und mit Hilfe dieser Werte seine Arbeit aufnehmen. Im Laufe der Zeit passt das Peer dann die Routingtabelle gemäß seinen Erfahrungen an. Die andere Möglichkeit, die ein Peer für die Entdeckung anderer Peers besitzt, ist das Feld „DataSource“. Dieses wird von einem beliebigen Knoten in der Kette der verschiedenen Knoten zurückgeliefert. Es durchläuft alle Peers, bis es schließlich beim Sender der Anfrage ankommt. Jedes Peer kann die Routinginformationen anpassen, falls es bereits andersartige Erfahrungen mit dem betreffenden Peer gemacht hat. So kann vermieden werden, dass Routinginformationen von einem Peer manipuliert und ungefiltert

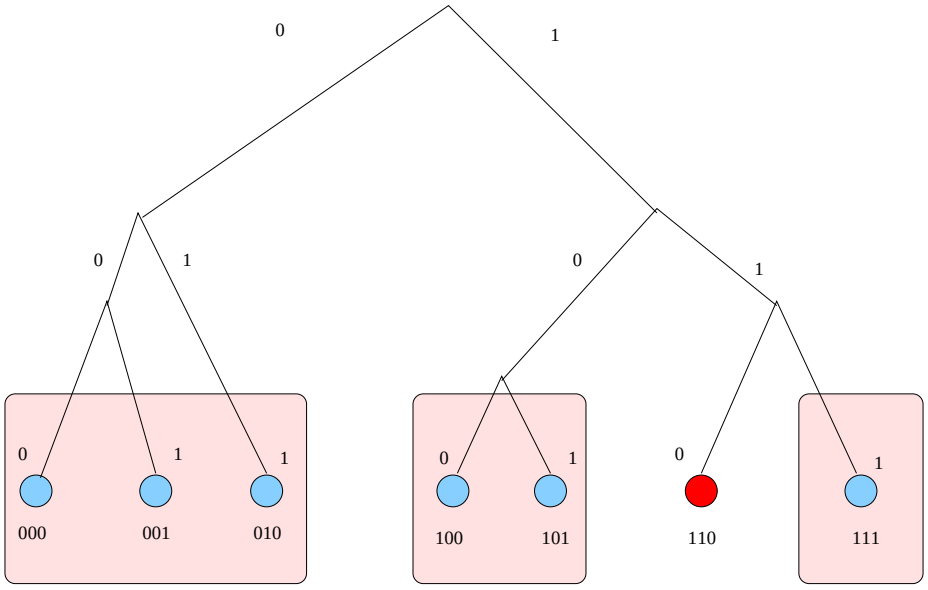

Abbildung 10: K-Buckets in Kademlia

an ein anderes Peer übertragen werden. In der jüngeren Vergangenheit wurde in Freenet ein sogenanntes Darknet implementiert. Dabei handelt es sich um einen Bereich des Netzwerkes, zu dem man nur durch eine Einladung eines schon angemeldeten Benutzers beitreten kann. Freenet ist für das Thema dieser Arbeit ein sehr interessantes Beispiel, da im Gesundheitswesen hohe Anforderungen an den Datenschutz gestellt werden. Dabei ist unter anderem wichtig, dass Anfragen nicht in jedem Fall einem konkreten Knoten zugeordnet werden können. Freenet bietet hier bemerkenswerte Ansätze.

2.5.2 Kademlia

Kademlia ist ein P2P-Algorithmus, der auf Basis verteilter Hashtabellen arbeitet. Er wurde von Petar Maymoukov und David Mazieres entwickelt. Ein Knoten berechnet zu Beginn eine zufällige Node-ID. Anschließend muss der Knoten einen Prozess durchlaufen, welcher „Bootstrapping“ genannt wird. Dabei erhält der neue Knoten die IP-Adresse eines anderen Knotens, von dem er wiederum Informationen über andere Knoten erhält. Die Node-IDs der Knoten stellen den Bezug zu den Hashwerten der Daten her, die von dem betreffenden Knoten verwaltet werden. Eine Suche erfolgt in mehreren Schritten. Mit jedem Schritt kommt die Suche dem Ergebnis näher. Schließlich wird der Knoten erreicht, der einen Verweis auf die gesuchten Daten enthält. Falls kein Knoten existiert, dessen Node-ID näher am gesuchten Hashwert liegt, war die Suche erfolglos. Die Anzahl der maximal durchlaufenen Knoten verhält sich dabei logarithmisch zur Gesamtzahl der Knoten im P2P-Netz. Ein Vorteil von Kademlia wurde erst nach seiner Entwicklung festgestellt. Das Netz ist sehr resistent gegen Denial-of-Service-Angriffe. Fällt ein Teil der Knoten aus, so bildet sich eine Struktur am Rand dieses Loches, und die Verfügbarkeit wird nicht sehr stark beeinträchtigt. Bei der Suche wird die Distanz zwischen dem Hashwert des gesuchten Schlüssels

und der Node-IDs der Nachbarn gebildet. Die Suchanfrage wird dann an den Knoten gesendet, dessen ID die geringste Distanz zum Hashwert aufweist. Diese Distanz wird gebildet, indem der Hashwert und die Node-ID mittels XOR verknüpft werden. Der daraus resultierende Wert wird als Integer interpretiert und ergibt die Distanz.

Das Routing von Kademlia nutzt sogenannte K-Buckets. Dabei handelt es sich um Listen. Für jedes Bit der Node-ID existiert eine dieser Listen. Jede beinhaltet mehrere Einträge. Jeder Eintrag enthält jeweils für einen Nachbarknoten alle Informationen, die notwendig sind, um diesen Nachbarknoten zu erreichen. Meist handelt es sich dabei um die IP-Adresse, den Port und die Node-ID. Jede dieser Listen beinhaltet alle Knoten, die eine bestimmte Distanz zur Node-ID des Knotens besitzen. Da die Anzahl der Einträge in den Listen mit der Größe des Netzwerkes sehr schnell ansteigen kann, wird die Anzahl der möglichen Einträge nach oben hin auf K Elemente begrenzt. Da die Menge der Knoten mit einer sehr ähnlichen Node-ID überschaubar ist, werden diese vollständig in den K-Buckets gespeichert. In den Buckets, welche die Node-IDs mit größerer Abweichung beinhalten, kann nur ein kleiner Teil der Knoten hinterlegt sein. Da jedoch der Raum der IDs weit größer sein muss als die Anzahl der vorhandenen Nodes, bleiben manche Buckets leer. Das betrifft ID's, welche eine sehr kleine Distanz zur Node-ID besitzen. Abbildung 10 zeigt grafisch die Verteilung eines möglichen Schlüsselraumes auf verschiedene K-Buckets. Ist ein K-Bucket mit K Einträgen gefüllt und ein neuer Knoten wird gefunden, welcher in diesen Bucket gehört, dann wird der Knoten ermittelt, der aus diesem Bucket die längste Zeit nicht mehr in Erscheinung getreten ist. Dieser Knoten wird angepingt. Falls dieser nicht antwortet, wird er durch den neuen Knoten ersetzt. Im anderen Fall wird der neue Knoten in eine andere Liste eingetragen - den sogenannten „replacement cache“. In diesem werden Knoten so lange hinterlegt, bis ein anderer Knoten nicht mehr zu erreichen ist. Der wird dann durch einen Knoten aus dem „replacement cache“ ersetzt.

In Kademlia sind vier Nachrichtentypen definiert. Erstens die PING-Nachricht. Sie dient ausschließlich dazu, zu überprüfen, ob ein Knoten noch erreichbar ist. Der zweite Nachrichtentyp ist die STORE-Nachricht. Sie hat den Zweck, einen Wert mit dem entsprechenden Schlüssel auf dem richtigen Knoten zu hinterlegen. Die FIND_NODE-Nachricht fragt einen Knoten nach den K Einträgen des K-Bucket, welcher dem gesuchten Schlüssel am nächsten kommt. Der vierte Nachrichtentyp ist die FIND_VALUE-Nachricht. Diese Nachricht liefert die gleichen Rückgabewerte wie FIND_NODE, falls der Schlüssel auf dem Empfängerknoten nicht vorhanden ist. Ansonsten wird der Wert zum gesuchten Schlüssel zurückgegeben. Kademlia beinhaltet kein Protokoll zum Austausch von Dateien. Es dient ausschließlich dazu, Daten zu lokalisieren. Diese müssen dann durch andere Protokolle von der Quelle zum Empfänger übertragen werden. Kademlia ist ein sehr leistungsstarker P2P-Algorithmus, mit dessen Hilfe Daten sehr performant gefunden werden können. Der Ansatz, Referenzen auf Daten in den Keybuckets und damit auf anderen Knoten zu deponieren, entspricht jedoch nicht ganz dem Ziel, jeden Knoten unabhängig von anderen Knoten betreiben zu können.

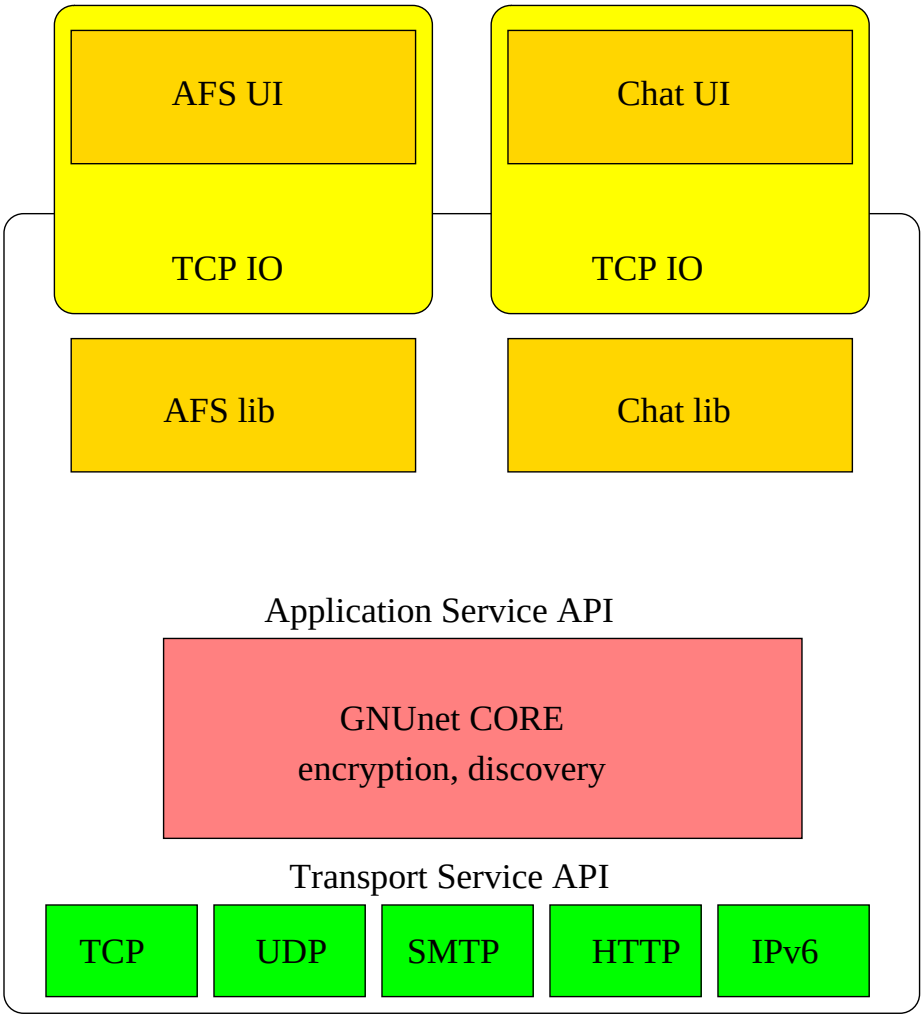

Abbildung 11: Aufbau von GNUnet

2.5.3 GNUnet

GNUnet ist ein Framework, welches dazu verwendet werden kann, anonyme P2P-Netzwerke aufzubauen. Das Projekt steht unter der Geral Public License (GPL) und soll als Plattform für neue dezentrale Internetprotokolle dienen. Die Designziele für GNUnet sind in erster Linie der Schutz gegen Angriffe und den Missbrauch von Daten, sowie der Schutz der Privatsphäre der Nutzer. Aus diesem Grund wurden in GNUnet keine Mechanismen integriert, welche die Analyse und die Auswertung von Kommunikationsvorgängen ermöglichen. GNUnet unterstützt kein spezielles Protokoll, sondern ganz verschiedene P2P-Technologien. Es ist mit Hilfe von Plugins erweiterbar. Da Sicherheit und Effizienz oftmals im Gegensatz zueinander stehen, erlaubt GNUnet es dem einzelnen Benutzer, ein hohes Maß an Sicherheit gegen Effizienz zu tauschen. Es ist einem Teilnehmer jedoch nicht möglich, die Sicherheit anderer Knoten zu gefährden. Die Kommunikation zwischen Peers geschieht fast ausschließlich zwischen gegenseitig authentifizierten Partnern. Ein Peer erzeugt einen Sitzungsschlüssel, signiert diesen mit seinem geheimen Schlüssel und verschlüsselt diesen mit dem öffentlichen Schlüssel des anderen Peers. Dieser Sitzungsschlüssel wird dann für eine mit AES verschlüsselte Verbindung verwendet. Die IP-Adresse oder der Port spielen bei der Authentifizierung keine Rolle. Entscheidend ist lediglich der öffentliche Schlüssel eines Peers. In P2P-Netzen ist es wichtig, dass jeder Knoten auch Daten zur Verfügung stellt. Sogenannte Freeloader, die Daten herunterladen, ohne selbst welche bereitzustellen, schaden dem P2P-Netz. Um derartige Knoten herauszufiltern, führt jeder GNUnet-Knoten ein Logbuch für jeden Knoten, mit dem er kommuniziert hat. Antwortet ein Knoten auf eine wichtige Anfrage, so steigt seine Priorität. Wenn der Netzwerktraffic steigt, werden nur die Knoten bedient, die eine hohe Priorität besitzen. Ein wichtiger Aspekt bei der Entwicklung von GNUnet ist die Anonymität der Teilnehmer. Diese wird erreicht, indem Kommunikationspakete im Traffic anderer Knoten versteckt werden. Es ist so für einen Angreifer, der eine Reihe von Knoten unter Kontrolle hat, schwer, zu unterscheiden, ob ein Paket von einem speziellen Teilnehmer stammt oder nur weitergeleitet wurde. Außerdem haben alle Pakete die gleiche Größe. Diese richtet sich nach der Maximum Transmission Unit (MTU) des entsprechenden Transportprotokolls.

In GNUnet sind verschiedene Protokolle implementiert, welche für die Kommunikation in P2P-Netzen und im Client-Server-Bereich gedacht sind. Abbildung 11 zeigt den Aufbau von GNU-Net. Im Zentrum der Grafik befindet sich der Kern von GNU-Net. Im oberen Teil der Abbildung sind Bibliotheken dargestellt, die den Zugriff auf verteilte Dateisysteme und Instant Messaging-Funktionen erlauben. Im unteren Bereich sind die unterschiedlichen Komponenten der Transport Service API dargestellt. Sie ermöglichen eine Kommunikation über unterschiedliche Protokolle. Im P2P-Bereich handelt es sich dabei um Funktionen des GNUnet-Kerns, das GAP-Protokoll, das RPC-Protokoll und das DHT-Protokoll. Im Kern sind Nachrichten enthalten, welche jeder Node unterstützen muss. Das sind beispielsweise eine PING-Nachricht und die dazugehörige Antwort, die PONG-Nachricht. Wichtig ist auch die sogenannte SETKEY-Nachricht. Diese dient zum Austauschen eines symmetrischen Schlüssels zwischen den Peers. Der Schlüssel wird verwendet, bis vom anderen Peer eine HANGUP-Message empfangen wird. Diese weist das Peer an, seinen Sessionkey zu verwerfen. Um Nachrichten übertragen zu können, deren Größe die

MTU des entsprechenden Transportprotokolls übersteigt, werden FRAGMENT-Nachrichten definiert. Sie enthalten nur einen Teil der Nachricht. Die sogenannten NOISE-Nachrichten werden verwendet, um eine Analyse des Traffic zu erschweren. Diese Nachrichten enthalten keine Daten und sollen so die wirklichen Pakete verstecken. Das GAP-Protokoll dient zum Senden von Querys an einen anderen Knoten. Es sind zwei Nachrichtentypen definiert. Zum einen die P2P_Gap_Query-Nachricht und zum Anderen die GAP_Result-Nachricht. Der erste Nachrichtentyp dient zum Senden einer Anfrage, und der zweite beinhaltet die Antwort auf die Anfrage. Dabei wird eine Anfrage in drei Schritten verarbeitet. Zuerst wird eine Entscheidung darüber getroffen, ob die Suchanfrage überhaupt verarbeitet wird. Das hängt beispielsweise von der Auslastung des Peers ab. Ist das Peer überlastet, die Anfrage veraltet oder wurde die Anfrage schon einmal beantwortet, so wird die Anfrage nicht bearbeitet. Wird das Query verarbeitet, dann führt das Peer im zweiten Schritt eine lokale Suche durch. Ist diese erfolgreich, so sendet das Peer eine Antwort an den Sender zurück. Im anderen Fall kommt es zu Schritt drei und es wird ein Routing durchgeführt, der Wert „time-to-live“ dekrementiert und die Anfrage an einen Knoten weitergeleitet. Die Auswahl wird mit einem Algorithmus getroffen, der zufällig ein anderes Peer auswählt, wobei die Wahrscheinlichkeit für ein bestimmtes Peer gewichtet ist. Die Peers, mit denen gute Erfahrungen für den Bereich des gesuchten Schlüssels gemacht wurden, werden mit einer höheren Wahrscheinlichkeit ausgewählt. Dieser Algorithmus kann jedoch auch geändert werden. Wird der in GNUnet implementierte Algorithmus an spezielle Anwendungsfälle angepasst, so bleiben die Knoten dennoch kompatibel. Peers können eine Antwort, welche sie von einem Peer bekommen und weiterleiten, zwischenspeichern. Es ist auch möglich, dass für eine Suchanfrage mehrere Ergebnisse gefunden werden. Jeder Knoten merkt sich deshalb die Ergebnisse, die er auf eine Anfrage weitergeleitet hat. So kann vermieden werden, dass ein Peer zweimal das gleiche Ergebnis überträgt. Das GAP-Protokoll benutzt den ECRS-Kern, wobei ECRS für „Encoding for Censorship-Resistant Sharing “ steht, [GG03] um Suchergebnisse der dazugehörigen Anfrage zuzuordnen.

GNUnet unterstützt auch Remote Procedures Calls. Dafür ist in GNUnet ein RPC-Protokoll integriert. Es erlaubt den Aufruf einer Funktion auf einem anderen Peer. Dieser Zugriff erfolgt immer von einem Peer direkt zu einem anderen. Ein Peer, welches RPC-Anfragen nicht unterstützt, verwirft eine empfangene Anfrage einfach. Das RPC-Protokoll unterstützt die drei Nachrichtentypen REQ, RES und ACK. Andere Protokolle von GNUnet bauen auf diesen RPC-Nachrichten auf. GNUnet implementiert ein DHT-Protokoll für den Einsatz mit verteilten Hashtabellen. Es beinhaltet eine Reihe von Nachrichtentypen. Einer der wichtigsten Nachrichtentypen bei DHT in GNUnet ist die Discovery-Nachricht. Sie dient dazu, einem Peer mitzuteilen, dass DHT unterstützt wird und wichtige Informationen über die Topologie des Netzwerkes zur Verfügung zu stellen. Neben den Protokollen für die Kommunikation in P2P-Netzwerken unterstützt GNUnet auch die Client-Server-Architektur. Diese Protokolle dienen in GNUnet jedoch in erster Linie dem Management des GNUnet-Dienstes. Ein Client kann sich mit dem GNUnet-Deamon verbinden und diesen konfigurieren und steuern. Er kann weiter Statistiken der Daten und des Traffic abrufen. Die Client-Server-Architektur beinhaltet jedoch auch das FS-Protokoll, welches für Filesharing benutzt werden kann. Als Transportprotokolle können in GNUnet beispielsweise TCP, UDP, HTTP oder auch SMTP benutzt werden. Es ist auch

möglich, andere Protokolle für den Transport zu verwenden. Um andere Protokolle einzubinden, stellt GNUnet eine Transport-API zur Verfügung. [FGR03b] [FGR03a] Damit kann auch ein Transport über ein bislang nicht unterstütztes Transportprotokoll verwendet werden. So könnte beispielsweise ein Transport mittels Webservices umgesetzt werden. GNUnet weist im Ansatz gewisse Parallelen zu Freenet auf. Es handelt sich dabei jedoch mehr um eine sehr flexible Plattform für die Entwicklung eines P2P-Netzwerkes.

2.5.4 Fazit

In den vorhergehenden Abschnitten wurden unterschiedliche Ansätze für die Implementierung von verteilten Systemen betrachtet. Zum einen wurde auf die Middleware-Systeme Unicore, Gridbus, Legion sowie Globus-Toolkit eingegangen, die für die Entwicklung gridbasierter Anwendungen verwendet werden. Jedes dieser Projekte verwendet einen anderen Ansatz, wie beispielsweise Legion, wonach die Ressourcen der einzelnen Knoten zu einem virtuellen Rechner zusammengefasst werden. Jedes dieser Middleware-Systeme ist jedoch dazu entwickelt worden, Hardwareressourcen anderer Knoten für die Lösung von komplexen Problemen oder die Speicherung großer Datenmengen nutzen zu können. Die in dieser Arbeit entworfene Softwarearchitektur soll jedoch weniger der Nutzung von Hardwareressourcen anderer Knoten dienen, als vielmehr einem übergreifenden Workflow. Dies ist mit Mitteln der Service Orientierten Architektur realisierbar, wovon mit Ruby on Rails und Django zwei Webapplications-Frameworks vorgestellt wurden, mit deren Hilfe es möglich ist, Webservices zu implementieren. Weiter wurden aus dem Bereich der Serviceorientierten Architektur BPEL und Web-Services Choreography Description Language (WS-CDL) betrachtet, wovon BPEL in erster Linie für die Orchestrierung von Webservices entwickelt wurde. Dieser Ansatz passt nicht zu der in dieser Arbeit vorgestellten Architektur, da für die Orchestrierung eine zentrale Instanz notwendig ist. Interessanter ist der Ansatz der Choreographie von Webservices, der mit BPEL ebenfalls realisiert werden kann und für dessen Einsatz WS-CDL entwickelt wurde. Bei genauer Betrachtung wurde jedoch schnell klar, dass eine Implementierung mit Hilfe von Choreographie sehr aufwändig ist und damit eine Entwickung von neuen Diensten durch einzelne Knoten nicht zu leisten ist. Unter diesem Aspekt wurde das Werkzeug Ruby on Rails ausgewählt, um die Steuerung der Dienstarchitektur mit Hilfe von Webservices prototypisch zu entwickeln. Mit Freenet und Kademlia wurden zwei Algorithmen vorgestellt, welche dem Bereich der P2P-Netzwerke zuzuordnen sind. Dieser Ansatz passt gut zum Konzept, da er eine Kommunikation von gleichberechtigten unabhängigen Knoten, die im P2P-Bereich Peers genannt werden, gewährleistet. Diese Technologie bietet die Möglichkeit, eine Suche zu realisieren, die ohne zentrale Instanzen arbeitet. Ausschließlich mit Werkzeugen der Service Orientierten Architektur ist diese P2P-Suche schwer zu realisieren. Deshalb wurde die Suchfunktion als modularer Bestandteil integriert, der mittels Webservices angesprochen werden kann. Ein solches Modul könnte mit Hilfe der Open Source Bibliothek GNUnet realisiert werden, die ebenfalls im Abschnitt über P2P-Technologien betrachtet wurde. Aufgrund der Struktur im Gesundheitswesen sind etliche Aspekte von Freenet für einen Suchalgorithmus innerhalb des Gesundheitswesens beachtenswert. Die Konzepte von Freenet ermöglichen einen hohen Standard an Datensicherheit und Anonymität der Patienten bei vertretbaren Suchergebnissen.

3 Architektur-Konzept

3.1 Grundkonzept

Komplexe Arbeitsabläufe und Kommunikationsvorgänge setzen sich meist aus einer Menge wiederkehrender Basisdienste zusammen. Diese sollen in den folgenden Abschnitten definiert und anschließend komplexere Dienste mittels dieser Basisdienste entworfen werden. Basisdienste werden mit Hilfe von Webservices realisiert. Diese Technologie bietet Plattformunabhängigkeit und eignet sich besonders für die Verknüpfung heterogener, lose gekoppelter Systeme, wie im Kapitel 2.3 beschrieben wurde. Im Gesundheitswesen existiert eine solche heterogene IT-Landschaft. Aus diesem Grund ist die Verwendung von Webservices als Grundlage der Kommunikation das Mittel der Wahl. Die Prozesse - speziell die Behandlungsprozesse, die im Gesundheitswesen ablaufen - sind jedoch stark verzahnt - oder sollten es zumindest sein. Die Verwendung von Webservices reicht an dieser Stelle nicht aus. Es ist nötig, die einzelnen Basisdienste zu komplexeren Abläufen zu verbinden und den Diensten Statusinformationen anzuhängen. Dieses Problem wird in anderen Bereichen mit Beschreibungssprachen wie BPEL und dem Konzept der Orchestrierung vom Webservices gelöst. Das ist jedoch bei den Prozessen im Gesundheitswesen mangels einer zentralen Einheit, welche die Prozesse überwacht und steuert, nicht sinnvoll. Jeder Knoten sollte hier unabhängig sein. Ein Konzept muss sich also in diesem Bereich mehr an der Choreographie von Webservices orientieren. Wie in Abbildung 12 erkenn-

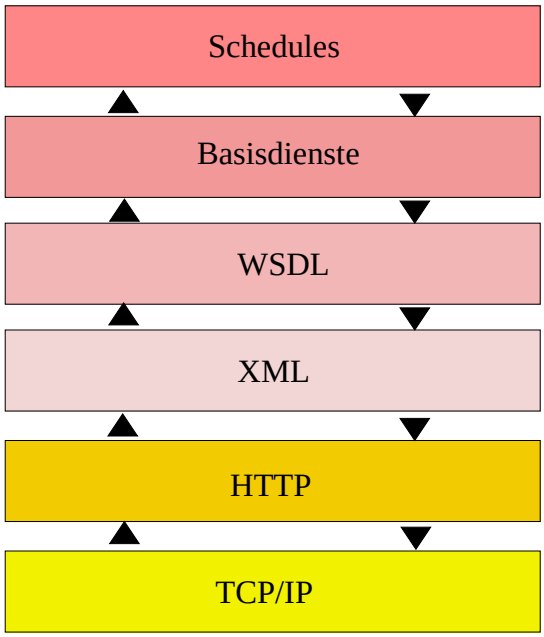

Abbildung 12: Kommunikationsschichten

bar, lässt sich die Kommunikation innerhalb der Dienstarchitektur in Schichten untergliedern. Die Basis bildet TCP/IP, auf dem das HTTP-Protokoll aufsetzt. Mit Hilfe dieses Protokolls werden XML-Dateien übertragen, deren Struktur wiederum mit Hilfe der Webservice Definition Language (WSDL) beschrieben

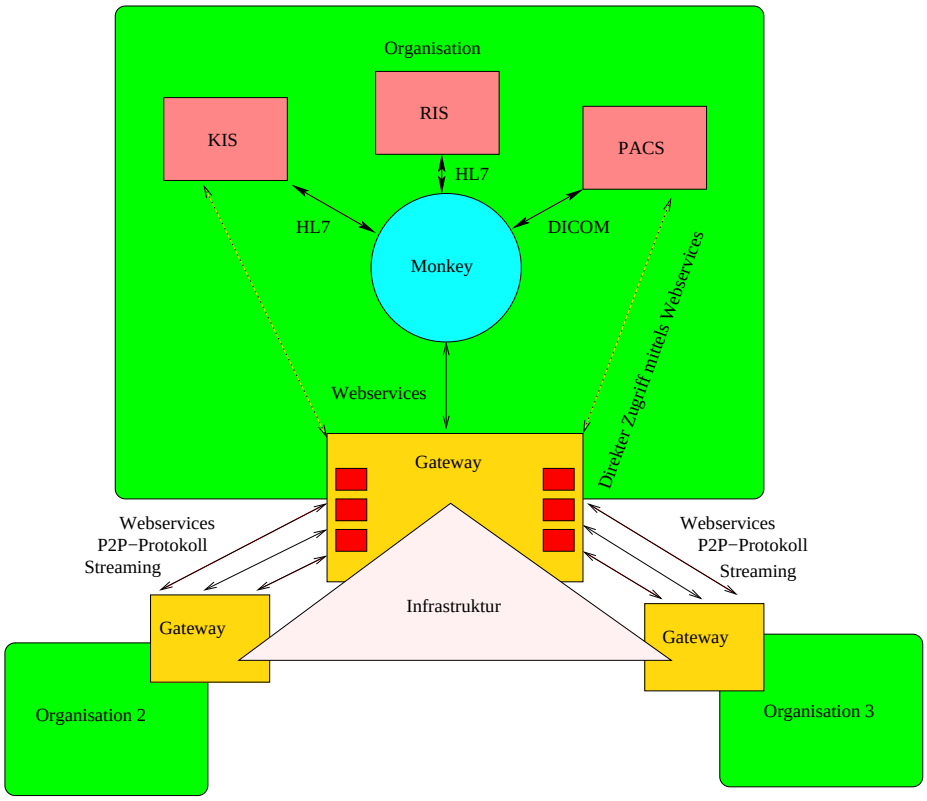

Abbildung 13: Integration der Software in KH-Struktur

wird. Auf diesen Webservices setzen die Basisdienste auf, die in Kapitel 3.6 definiert werden. Diese bilden die Grundlage für die Schedules, welche in Kapitel 3.5.1 eingeführt werden. Wichtig dabei ist, dass auf einfache Art und Weise neue Dienste schnell und ohne Programmieraufwand implementiert werden können. Das wird durch die Komposition von komplexen Diensten aus einer Menge von fest vorgegebenen Basisdiensten erreicht. Dabei werden Basisdienste als solche Dienste definiert, die für ihre eigene Ausführung keine Statusinformationen benötigen. Der Aufruf eines Basisdienstes besteht demzufolge aus genau einer Anfrage, die vom Dienstanbieter beantwortet wird. Mittels Webservices können Datensätze übermittelt werden, die aus einer Menge von Variablen unterschiedlichen Typs zusammengesetzt sind. Eine Sequenz solcher Webservices wird nun in eben einer solchen Datenstruktur abgelegt und kann mittels eines Webservices an einen anderen Knoten übermittelt werden, so dass beide Knoten nun über denselben Ablaufplan verfügen. So entstehen komplexe Workflows über die Grenzen von Einrichtungen hinweg. Im folgenden Kapitel soll der Aufbau der Software und die dafür notwendigen Technologien und Serviceklassen skizziert werden.

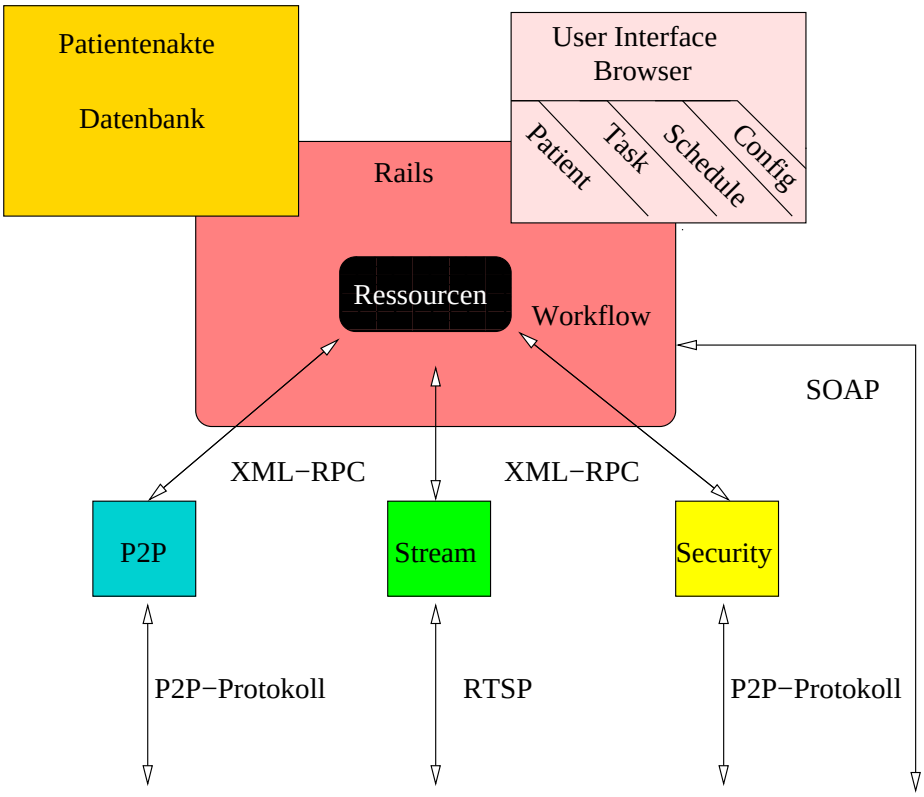

Abbildung 14: Integration von Modulen

3.2 Gesamtarchitektur von Diensten

Die Dienstarchitektur dient dazu, unterschiedliche Einrichtungen des Gesundheitswesens zu verknüpfen. Interne Strukturen sind aus der Sicht anderer Knoten nicht relevant. Es kommuniziert immer eine Organisation mit einer anderen. Dadurch entsteht ein Netz verschiedener Knoten, die miteinander kommunizieren können. Jeder dieser Knoten bietet Dienste zu Systemen innerhalb der Einrichtung und Dienste nach außen an. Abbildung 13 zeigt, dass die Software, welche die Kommunikation mit anderen Knoten durchführt, auch mit Systemen innerhalb einer Einrichtung Daten austauschen muss. Diese Systeme können Informationssysteme wie das PMS, das RIS oder das PACS sein, auf die bereits in Kapitel 1.6.1 eingegangen wurde. Diese Kommunikation wird mit Hilfe von Webservices realisiert. Bei der Betrachtung möglicher Dienste im Gesundheitswesen wird jedoch deutlich, dass nicht alle denkbaren Anwendungen auf Basis von Webservices implementiert werden können. Die Suche über P2P-Technologien beispielsweise lässt sich wesentlich komfortabler mit anderen Transporttechnologien und Werkzeugen realisieren. Für andere Anwendungen, wie Streaming von Videodaten, sind Webservices sogar vollkommen ungeeignet. Aus diesem Grund muss die Anwendung offen implementiert sein. Dadurch können Module angebunden werden, die Kommunikationsaufgaben übernehmen, welche mit Webservices nicht umsetzbar sind. Diese Module werden über einheitliche Schnittstellen angesprochen. An dieser Stelle kommen wiederum Webservices zum Einsatz. Ein solches Modul kann dann eine Skype-Verbindung aufbauen, Dateien aus dem Internet downloaden oder Roboter für ferngesteuerte Operationen ansteuern. Abbildung 14 zeigt schematisch den Aufbau der Softwarearchitektur. Zum einen besteht diese aus einer Datenbank, in welcher sich Patienteninformationen befinden. Zum anderen existiert ein webbasiertes Benutzerinterface, mit deren Hilfe die Anwendung gesteuert werden kann, und die Zugriff auf die in der Datenbank befindlichen Informationen erlaubt. Der Kern der Anwendung stellt Webservices zur Verfügung, die mit den Systemen innerhalb einer Einrichtung und mit anderen Knoten kommunizieren. Die Ressourcen stellen eine Verbindung zwischen den Daten der Patientenakte und den Diensten her, wie in Abschnitt 3.5.3 genauer beschrieben wird.

3.2.1 Interne Dienste zur Steuerung der Anwendung

Im vorigen Kapitel wurde erläutert, dass die Anwendung nicht nur mit anderen Knoten, sondern auch mit Systemen innerhalb der Organisation Daten austauschen muss. Ohne diesen Datenaustausch wäre die gesamte Anwendung nutzlos, denn die übergreifende Dienstarchitektur dient zum Austausch von Daten, die sich in Informationssystemen innerhalb des Local Area Network (LAN) der Organisation befinden. Diese Kommunikation zwischen den internen Informationssystemen wird mittels XML-RPC umgesetzt. Wie in Kapitel 7 näher erläutert wird, kann diese Anbindung mit Hilfe eines Integrationsservers durchgeführt werden, der bereits vorhandene Schnittstellen in XML-RPC-Aufrufe umsetzt. Solche Schnittstellen können beispielsweise im Krankenhausbereich verbreitete HL7-Nachrichten sein. Listing 1 zeigt eine solche Nachricht von Typ ADT/A01, die für die Übermittlung von Patientenstammdaten üblich sind. Die Felder einer solchen Nachricht müssen dann von einem Integrationsserver in ein Basisobjekt der Infrastruktur umgewandelt und mittels eines XML-RPC-Aufrufes an das

Gateway übertragen werden.

Listing 1: HL7 Nachricht

```
MSH|~^&|MONKEY|||Portal|201001261221||ADT^A01|73|P|2.3|73
EVN||||501|ROESSLER
PID||86|17||Max^Muster||1984|2|||Bergstr.9^^DD^^|||||L|
OR||||||Burgstaedt|||D
PV1||I|BCH1^^^DCL^A01^^1|I|||ZEUG^Rst^^^^Dr.med.||3|||
```

Im Vergleich zu SOAP ist der Aufruf eines Webservice mit Hilfe von XML-RPC performanter. In Tabelle 3.2.1 sind die Laufzeiten verschiedener Webservices gegenübergestellt. Die Werte wurden mit Hilfe des in Rails integrierten Scaffolding durchgeführt. Die Webservice-API, die Rails mit Hilfe von Action-Webservice bereitstellt, kann analog über SOAP und XML-RPC aufgerufen werden. Die Tests wurden lokal durchgeführt. Bei einem Aufruf der Services über ein Weitverkehrsnetz ist damit zu rechnen, dass die Differenz der Laufzeiten noch größer ist als bei den durchgeführten Versuchen. Bei Diensten wie „remote_control“ ist der Unterschied weniger signifikant als beim Aufruf des Dienstes „put_patient“. Die Ursache dafür ist die Menge der Daten, die übertragen wird. Das Patientenobjekt beinhaltet ein Foto des Patienten, welches in Base64-Format in den Request eingebunden wird. Die Datenmenge dieses Fotos beträgt lediglich 37 Kilobyte. Die gemessenen Ergebnisse lassen jedoch die Vermutung zu, dass die Übertragung von großen Datenmengen zu noch auffälligeren Differenzen der Laufzeit von XML-RPC und SOAP führt. Der Grund für die schnellere Verarbeitung von XML-RPC-Aufrufen liegt im geringeren Overhead dieses Protokolls. Im folgenden Listing 2 ist der Request des Dienstes „remote_control“ über XMP-RCP dargestellt.

Listing 2: XML-RPC-Request des Service remote_control

```
<?xml version="1.0" ?>
<methodCall>
<methodName>RemoteControl</methodName>
<params><param>
<value><string>12345</string></value>
</param><param>
<value><string>D_NEUROKONSIL</string></value>
</param></params>
</methodCall>
```

Da im Normalfall mehr Objekte innerhalb einer Einrichtung übertragen werden müssen als dann auch nach außen gesendet werden, ist es vorteilhaft, die internen Dienste über das performantere XML-RPC abzubilden. Der Overhead einer Kommunikation mit XML-RPC ist unter anderem deshalb geringer als beim SOAP-Protokoll, da XML-RPC weniger Möglichkeiten für die Gewährleistung der Datensicherheit und Verschlüsselung besitzt. Darauf wird im Kapitel 6 genauer eingegangen. Aus diesem Grund ist es wichtig, dass die Webservices über XML-RPC auch wirklich nur von Knoten aus dem lokalen Netz aufgerufen werden können. Das kann mit Hilfe eines HTTP-Filters erreicht werden. Dieser darf von außen nur Requests über das SOAP-Protokoll erlauben. Diese Aufgabe kann beispielsweise ein Proxyserver, wie die Open-Source-Software Squid, übernehmen. Eine andere Möglichkeit besteht darin, XMP-RPC-Aufrufe über einen anderen Port als SOAP entgegenzunehmen. Der XML-RPC-Port kann dann von einer Firewall geblockt werden.

Service	SOAP	XML-RPC
remote_control	0.030359 s	0.018213 s
put_value	0.092438 s	0.062801 s
put_data	0.207302 s	0.061444 s
put_patient	0.279159 s	0.065937 s

Tabelle 1: Vergleich der Latenzzeiten zwischen XML-RPC und SOAP

Die internen Dienste dienen in erster Linie dazu, Basisobjekte, die in 3.4 definiert werden, zu übertragen. So kann der Stammdatensatz eines Patienten mit Hilfe des Webservice „put_patient“ an die Patientenakte der übergreifenden Infrastruktur übertragen werden. Um eine Kommunikation auszulösen, existiert der Webservice remote_control. Ähnlich einer Fernbedienung soll dieser Dienst nur aus der Nähe - also von Systemen innerhalb der zugehörigen Organisation - aufgerufen werden können und explizit einen Kommunikationsvorgang auslösen. Wenn beispielsweise das PMS die Verlegung eines Patienten veranlassen möchte, dann kann es diesen Webservice aufrufen und als Parameter den Master Patient Index (MPI) und die entsprechende Funktion übergeben.

Listing 3: Ausschnitt aus WSDL-Beschreibung

```
<xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="urn:data/wsdl">
<xsd:complexType name="Value_OJ">
<xsd:all>
<xsd:element name="unit" type="xsd:string"/>
<xsd:element name="value" type="xsd:double"/>
<xsd:element name="device" type="xsd:string"/>
<xsd:element name="cert" type="xsd:string"/>
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="serialno" type="xsd:string"/>
<xsd:element name="ceiling" type="xsd:double"/>
<xsd:element name="oid" type="xsd:int"/>
<xsd:element name="typ" type="xsd:int"/>
<xsd:element name="assistant" type="xsd:string"/>
<xsd:element name="lower" type="xsd:double"/>
<xsd:element name="mpi" type="xsd:string"/>
<xsd:element name="creation" type="xsd:dateTime"/>
<xsd:element name="institution" type="xsd:string"/>
<xsd:element name="episode" type="xsd:int"/>
</xsd:all>
</xsd:complexType>
</xsd:schema>
</types>
<message name="PutValue">
<part name="param0" type="typens:Value_OJ"/>
<part name="param1" type="xsd:string"/>
<part name="param2" type="xsd:string"/>
</message>
</definitions>
```

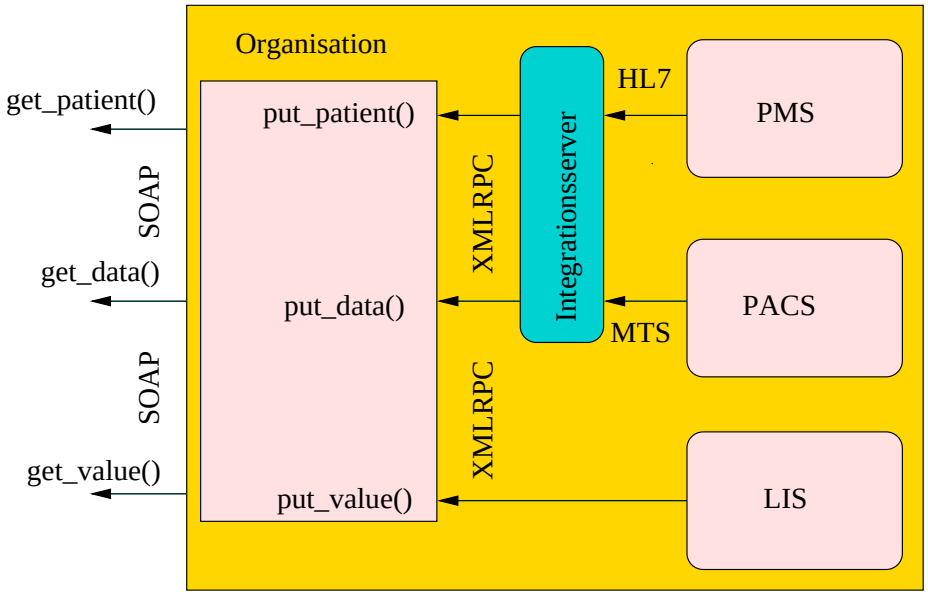

Abbildung 15: Kommunikation nach außen und innnen

3.2.2 Externe Dienste

Eine zweite Möglichkeit der Steuerung bilden sogenannte Callbacks. Diese veranlassen den Aufruf bestimmter Funktionen bei Änderungen am Datenmodell. Rails unterstützt die Definition von Regeln, nach welchen Änderungen am Datenmodell vom Anwender definierte Funktionen auslösen. Soll beispielsweise in einem Klinikverbund automatisch jeder Patient an das System der anderen Klinik übermittelt werden, so kann diese Aufgabe mittels Callback gelöst werden. Wird über den Service „put_patient“ ein neuer Patientendatensatz in die Datenbank eingebracht, dann ruft das System einen Dienst auf, der den Patienten mittels der externen Services verlegt. Der eigentliche Nutzen der in dieser Arbeit entworfenen Software ist die Kommunikation über die Grenzen einer Einrichtung hinweg. Im vorigen Abschnitt wurden Dienste vorgestellt, die für den Datenaustausch zwischen der Infrastruktur und den Informationssystemen innerhalb einer Einrichtung verwendet werden. Im Gegensatz zu diesen Diensten werden die Webservices, die eine Einrichtung nach außen zur Verfügung stellt, mit Hilfe des SOAP-Protokolls realisiert. Abbildung 15 zeigt schematisch die XML-RPC-Aufrufe innerhalb der Einrichtung, mit denen die gleichen Basisobjekte übermittelt werden, wie bei der Kommunikation mit anderen Knoten. Dazu wird allerdings das SOAP-Protokoll verwendet. Dieses Protokoll bietet mehr Möglichkeiten, die damit zu übertragenden Daten zu sichern. Weiterhin erzeugt Rails für die SOAP-Schnittstelle eine Beschreibung in WSDL, die von einem anderen Knoten heruntergeladen werden kann. In Listing 3 ist ein Ausschnitt der Beschreibung des Services „put_value“ dargestellt, der für die Übertragung von Labor- und sonstigen Werten in Form eines Objektes vom Typ „Value“ verwendet werden kann. Wie auch die Dienste innerhalb einer Einrichtung, werden mit den externen Services Basisobjekte übertragen, welche in den folgenden Kapiteln genau definiert werden. Darüber hinaus existieren auch Dienste, die zur

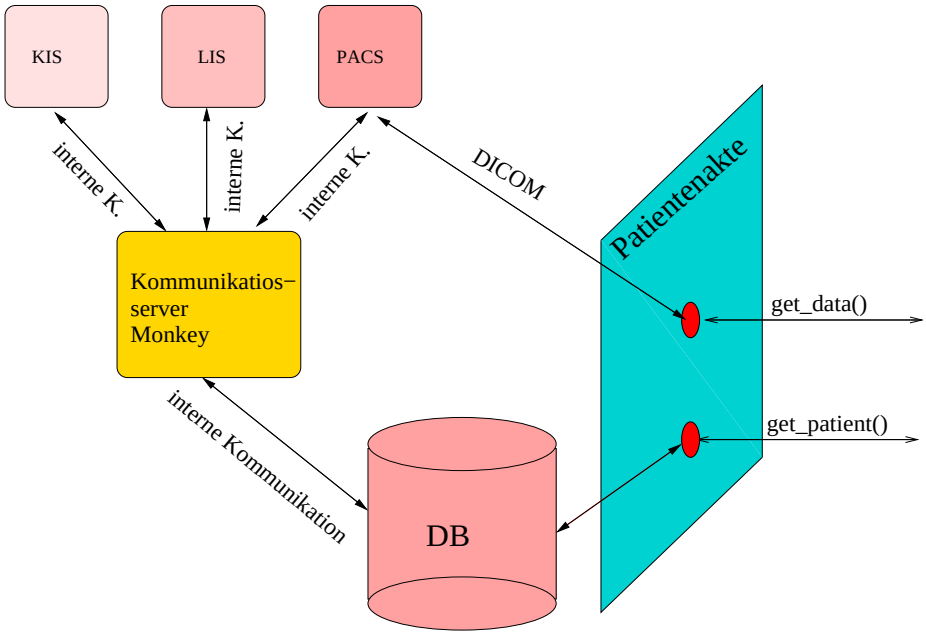

Abbildung 16: Elektronische Patientenakte

Steuerung der Anwendung verwendet werden. Der Dienst „activate“ beispielsweise aktiviert einen Knoten bezüglich eines Vorgangs, dessen weitere Bearbeitung der aufgerufene Knoten fortsetzen soll. Weiter existiert ein Dienst namens „get_service“. Dieser teilt einem anderen Knoten mit, welche Dienstleistungen ein Knoten für andere Teilnehmer anbietet. Dieser Dienst liefert Objekte vom Typ „service“ zurück. Welche Basisdienste benötigt werden, wird detailliert im Abschnitt 3.6 beschrieben.

3.2.3 Datenstrukturen

Intern fallen im Krankenhaus Daten auf verschiedene Weise an. Manche werden in Systemen gespeichert, von denen die Daten später über Protokolle wie DICOM wieder abgerufen werden können, so zum Beispiel Röntgenbilder, die mittels Query/Retrieve angefordert und übertragen werden. Andere Daten, wie eine Patientenaufnahme, werden in der Praxis oft als Nachricht an alle Subsysteme gesendet. Besonders das Patientenmanagementsystem als führendes System besitzt oftmals nur wenige Schnittstellen für eine Abfrage. Solche Systeme senden bestimmte Ereignisse oft nur ein einziges Mal an alle Subsysteme. In der HL7-Spezifikation werden zwar Query- und Response-Nachrichten definiert, jedoch unterstützen nicht alle Subsysteme derartige Abfragen. Als Grundlage für Services, die eine Einrichtung anderen Teilnehmern zur Verfügung stellt, ist eine Datenbasis nötig, in welcher alle Informationen, die dafür relevant sind, verfügbar gemacht werden. Manche dieser Daten werden einmalig über einen Integrationsserver oder direkt an alle Subsysteme gesendet. Es ist deshalb sinnvoll, das Gateway ebenfalls als Subsystem in das KIS zu integrieren. Das Gateway muss dazu über eine Datenbank verfügen, in welcher Patientendaten hinterlegt werden können. In der Datenbank werden die Datenobjekte als Tabellen abge-

legt. Diese Tabellen haben eine Entsprechung bei den Datenobjekten, die von den Webservices übertragen werden. So entsteht eine elektronische Patientenakte, die als Datenbasis für die Kommunikation in der Dienstarchitektur dient. Manche Daten werden direkt in der Patientenakte gespeichert, während andere bei Bedarf aus den Subsystemen angefordert werden. Dazu ist die Entwicklung einer Softwarezwischenschicht analog zum objektrelationalen Mapping von Rails denkbar. Beispielsweise müssten Röntgenbilder dazu als Objekte in der Datenbank hinterlegt sein, welche nicht die eigentlichen Daten enthalten, sondern lediglich eine sogenannte Accession Number. Anhand dieser Zeichenkette lässt sich mittels des DICOM-Protokolls ein Datensatz aus dem PACS anfordern. Soll ein Objekt an einen anderen Knoten übertragen werden, so muss diese Zwischenschicht mittels Query/Retrieve das Bild aus dem PACS abrufen und als Objekt vom Typ „Data“ weitervermitteln. Die Query/Retrieve-Anfrage muss für den Nutzer vollkommen transparent sein. Abbildung 16 zeigt das Gateway, welches Daten über standardisierte Webservices bereitstellt. Diese Daten können in der Datenbank des Gateway hinterlegt sein oder bei Bedarf aus den Subsystemen, wie beispielsweise dem PACS, abgerufen werden. Mindestens ein Hashwert muss in der elektronischen Patientenakte des Gateway für jeden Aufenthalt in der Datenbank gespeichert werden. Anhand dessen kann entschieden werden, ob bei einer Suchanfrage zum gesuchten Schlüssel Daten vorhanden sind. Für die Kommunikationspartner erscheint der Knoten als eine lokale, vollständige elektronische Patientenakte. Zur Laufzeit werden die angeforderten Daten entweder aus der Datenbank oder über entsprechende Protokolle aus den Subsystemen angefordert.

3.3 HSML - Health Statistic Markup Language

Für manche Dienste ist es notwendig, dass auf einem Knoten gewisse Auswertungen stattfinden. Das ist teilweise aus rechtlichen Gründen erforderlich. Beim Aufruf spezieller Dienste müssen bestimmte Daten im jeweiligen Knoten auf eine vorgegebene Weise verdichtet werden und dürfen nur das Ergebnis zurückliefern. Will man beispielsweise über den Verlauf bestimmter Krankheiten Informationen erhalten, um die Ergebnisse für eine wissenschaftliche Arbeit zu verwenden, so dürfen dadurch keine Patientendaten den entsprechenden Knoten verlassen. Trotzdem müssen für die Ergebnisse Beziehungen zwischen den einzelnen Episoden hergestellt werden, die nach außen nicht sichtbar sein dürfen. Es ist deshalb notwendig, dass die Knoten selbstständig bestimmte Auswertungen durchführen, welche ihnen von außen vorgegeben werden. Um das zu erreichen, existieren verschiedene Ansätze: Bei der Verwendung von manchen Programmiersprachen wie z.B. Ruby wäre es denkbar, einem Knoten ein Codesegment zu übermitteln, das dieser zur Laufzeit in den Programmablauf einfügt und ausführt. Ein solches Vorgehen hätte jedoch enorme Nachteile. Zum einen birgt das Einfügen von fremden Programmcode hohe Sicherheitsrisiken in sich. Eine Überprüfung wäre sehr aufwändig und zugleich lückenhaft. Zum anderen würde ein solcher Code natürlich voraussetzen, dass alle Knoten mit Hilfe der gleichen Werkzeuge implementiert und vollkommen identisch strukturiert sind. Eine Ruby-Klasse kann natürlich auch nur in eine Software, welche in Ruby implementiert ist, eingefügt werden. Ziel der Dienstarchitektur und Grund für die Verwendung von Webservices ist jedoch die Plattformunabhängigkeit. Gleichzeitig sind die Anforderungen im Gesundheitswesen für eine Dienstarchitektur sehr

speziell, sodass nicht die Mächtigkeit einer Programmiersprache benötigt wird. Deshalb wird an dieser Stelle eine Beschreibungssprache mit Hilfe von XML definiert, welche auf die Erfordernisse im Gesundheitssystem zugeschnitten ist. Mit Hilfe dieser Beschreibungssprache können Listen mit statistischen Informationen von einem Knoten abgerufen werden. Sie dienen als eine Art Bestellliste für Daten und Anweisungen bezüglich deren Verpackung. Diese Beschreibungssprache ist so konzipiert, dass sie ausschließlich ein Array von numerischen Werten zurückliefert. Eine Anfrage nach Diagnosen, Namen, Befunden oder ähnlichem ist für statistische Zwecke nicht nötig und aus datenschutzrechtlichen Gründen nicht zulässig. Folgende Anforderungen sind dabei zu erfüllen:

- Verwendung der Basisobjekte der Infrastruktur
- Suche von Objekten nach allen Attributen
- Verknüpfung von Objekten
- Zählen von Objekten
- Vergleichen von Objekten und Attributen
- Grundrechenarten
- Ermitteln von Maximum, Minimum und Durchschnitt
- Übersichtlich und einfach strukturiert

Listing 4: HSML-Beschreibung

```
<average>
   <diff>
       <object episode>
         <attr>end</attr>
         <cond param=episodenid>
           <object name=drg>
              <attr>episodenid</attr>
              <or>
                <cond param=drg>K35.0</cond>
                <cond param=drg>K35.1</cond>
              </or>
           </object>
        </cond>
      <object>
        <object episode>
          <att >begin</attr>
          <cond param=episodenid>
            <object name=drg>
              <attr>episodenid</attr>
              <or>
                <cond param=drg>K35.0</cond>
                <cond param=drg>K35.1</cond>
              </or>
            </object>
          </cond>
        <object>
    </diff>
</average>
```

Die Beschreibungssprache ist folgendermaßen aufgebaut: Das Tag, welches alles umschließt, steht für das Ergebnis des Jobs. Handelt es sich beispielsweise um das Tag < **count** >, so soll das Ergebnis die Anzahl der darin beschriebenen Werte darstellen. Jedes Tag repräsentiert somit eine Liste aus Attributen verschiedener Objekte des gleichen Typs. Eine Anfrage wird in Form eines wohlgeformten XML-Dokumentes gestellt. Dabei werden zwei verschiedene Arten von Tags verwendet. Zum einen Tags, mit denen Basisobjekte selektiert werden können, und zum anderen solche, die für die Verdichtung der Attribute von Objekten benutzt werden können. Das Tag < **object** > kann analog zum SELECT relationaler Datenbankmanagementsysteme verstanden werden. Es wählt Objekte mit der im Attribut „name“ spezifizierten Bezeichnung aus. Innerhalb dieses Tags muss sich das Tag < **attr** > befinden. Es gibt an, welche Attribute der ausgewählten Objekte zurückgeliefert werden. Das Tag < **cond** > beschreibt eine Bedingung bei der Auswahl der Objekte. Das Attribut „para“ gibt das Attribut an, welches den Wert einnehmen muss, der vom Tag< **cond** > umschlossen wird. Befindet sich innerhalb des Abschnittes < **cond** >< /**cond** > ein Objekt-Tag, so ist die Bedingung für alle Werte erfüllt, welche das Objekt-Tag umschließt. Auf diese Art und Weise können Objekte miteinander verknüpft werden. Die Bedingung ist optional. Fehlt diese im Objekt-Abschnitt, so werden alle Objekte vom betreffenden Typ in die Auswahl übernommen. Die Bedingung kann selbst von den Tags < **and** >, < **or** > oder < **xor** > umschlossen sein. Das ist sinnvoll, wenn mehrere Bedingungen für die Auswahl der Objekte eine Rolle spielen und gibt an, wie die Bedingungen verknüpft werden.

Die zweite Gruppe von Tags dient dem Verdichten der Daten. Die Grundrechenarten werden mit den Tags < **add** >, < **diff** >, < **prod** > und < **frak** > abgebildet. Dabei können die Argumente Listen sein. In diesem Fall kann man die Listen als Vektoren betrachten, deren Elemente miteinander verknüpft werden. Addition und Subtraktion können wie die Operationen bei Vektoren durchgeführt werden. Division und Multiplikation werden je Element des Vektors einzeln durchgeführt, d.h. jedes Element wird mit dem Element der anderen Listen unter dem gleichen Index verrechnet. Die Tags < **max** >, < **min** > und < **average** > erlauben es, das Maximum, Minimum oder den Durchschnitt der Werte einer Liste zu ermitteln. In Listing 4 wird die durchschnittliche Verweildauer von Patienten, die wegen einer Blinddarmentzündung behandelt wurden, ermittelt. Auf diese Weise ist es möglich, Informationen von einem Knoten abzurufen, ohne dass dieser konkrete Objekte übermittelt. Trotzdem ist dabei enorm wichtig, dass die Abfragen einer genauen Kontrolle unterzogen werden. Eine Suche nach Fällen von Patienten mit einem bestimmten Nachnamen könnte auch implizit Patientendaten übermitteln. Wird eine Abfrage gestartet, die eine Zählung aller Episoden enthält, welche Patienten mit einem sehr seltenen Namen zugeordnet sind, dann kann das Ergebnis die Menge der Krankenhausaufenthalte eines Patienten enthalten. Die Abfragen müssen deshalb ebenso wie die Ergebnisse einer Kontrolle unterzogen werden. Bevor eine Anfrage bearbeitet wird, muss geprüft werden, ob die Berechtigungen für die in der Anfrage gesuchten Daten vorhanden sind. Nach der Bearbeitung muss das Ergebnis nochmals geprüft werden. So könnte beispielsweise eine berechnete Summe von Fällen nur dann übertragen werden, falls ein Wert größer als 20 ermittelt wurde. Diese Einschränkungen muss die Anwendung durch den Einsatz von Policies ermöglichen.

Objekt	Beschreibung
Patient	Patientenstammdaten
Episode	Informationen über Episoden
Diagnose	Diagnosen und ICD
Value	numerische Werte und entsprechende Referenzwerte
Text	Textdokumente wie Arztbriefe in Unicode
Findings	Befund in ASCII-Format
Date	Termin im Bezug auf einen Auftrag
Prescription	Elektronisches Rezept
Data	binäre Daten verschiedener Formate
Money	elektronisches Zahlungsmittel
Job	XML-File in HSML
Institution	Dienstleister im Gesundheitswesen
Order	Auftrag für eine Behandlung
Edifact	Abrechnungsdaten im Edifact-Format
Basics	Medizinische Grunddaten
Connection	Verbindungsdaten für Modul
Schedule	Ablaufplan
Service	Arbeitsschritt eines Schedules
Ressource	Datenobjekte eines Schedules
Error	Fehlermeldung eines Schedules

Tabelle 2: Basisobjekte

3.4 Basisobjekte

Den Basisdiensten liegen sogenannte Basisobjekte zugrunde. Bei diesen Objekten handelt es sich um Datenstrukturen, die verschiedene zusammengehörige Informationen bündeln. Dabei wurde angestrebt, mit einer möglichst kleinen Anzahl von Objekten möglichst viele Anwendungsfälle abzubilden. Je größer die Anzahl unterschiedlicher Datenstrukturen ist, desto wahrscheinlicher ist das Auftreten von Inkompatibilitäten. Bei den Basisobjekten werden administrative Datenobjekte, wie Patientenstammdaten und medizinische Daten, wie beispielsweise Röntgenbilder oder Labordaten unterschieden. Die Menge der möglichen medizinischen Daten wächst ständig an. Sehr oft werden neue Untersuchungsmethoden, Laboruntersuchungen und neue Geräte eingeführt, die ganz neue Messergebnisse liefern. Um dem Minimalitätsprinzip bei der Anzahl der Basisobjekte gerecht zu werden, wurden die medizinischen Daten in Objekte gekapselt, welche sich lediglich durch das Datenformat unterscheiden. Ein Foto einer Wunde ist ebenso ein Objekt vom Typ „Data “ wie eine Röntgenaufnahme. Das Objekt besitzt jedoch Informationen darüber, um welches Format es sich bei dem Datensatz handelt. Die Liste der Basisobjekte erhebt keinen Anspruch auf Vollständigkeit. Allerdings sollte bei der Definition von zusätzlichen Objekten das Prinzip beachtet werden, so wenig wie möglich Basisobjekte einzuführen. Momentan sind folgende Basisobjekte definiert.

3.4.1 Patient

Das Objekt „Patient“ beinhaltet die wichtigsten Stammdaten eines Patienten, welche für die Behandlung eines Patienten nötig sind. Es enthält die Stammdaten eines Patienten, die zum gegenwärtigen Zeitpunkt auf der Krankenversicherungskarte der Patienten und zum Basis-Rollout der EGK auf dieser gespeichert sind. Wird ein Patient verlegt, können diese Daten an die weiterbehandelnde Einrichtung weitergegeben werden. Wichtig dabei ist, dass alle Patienten eine einrichtungsübergreifende eindeutige Patienten-ID, also einen MPI, besitzen. Alle Objekte, die Patientendaten enthalten, können über den MPI, das Institutionskennzeichen und die Episodennummer der entsprechenden Einrichtung, eindeutig unterschieden werden. Listing 5 zeigt den Rubycode des Objektes Patient_OJ, welches die Repräsentation der Patientenstammdaten für die Übermittlung mittels Webservices darstellt.

3.4.2 Episode

Das Objekt „Episode“ umfasst eine Liste aller Daten, die bei einem Behandlungszyklus innerhalb einer Einrichtung zu einem Patienten angefallen sind. Es ist die zentrale Komponente, um Informationen über den Aufenthalt eines Patienten an eine weiterbehandelnde Einrichtung weiterzugeben. Das Objekt enthält Verweise auf sämtliche Daten, die während eines Aufenthaltes von Patienten gespeichert wurden. Bei Bedarf können diese Daten dann von den anderen Einrichtungen angefordert werden. Werden Patientendaten eines Patienten gesucht, so werden zuerst einmal alle Episodenobjekte aus anderen Einrichtungen gesammelt. Aus diesen Objekten kann jede Einrichtung entnehmen, welche Untersuchungen und Behandlungen mit dem Patienten in anderen Einrichtungen durchgeführt wurden. Wird ein Datensatz benötigt, kann die Einrichtung direkt angefragt werden.

Listing 5: Objekt patient_oj

```
class Patient_OJ < ActionWebService::Struct
  member :id, :string
  member :mpi, :string
  member :name, :string
  member :givename, :string
  member :suffix, :string
  member :title, :string
  member :birthname, :string
  member :birthday, :datetime
  member :birthtown, :string
  member :insurance, :string
  member :street, :string
  member :zip, :string
  member :city, :string
  member :gender, :string
  member :phone, :string
  member :email, :string
  member :image, :binary
  member :public_key, :string

  def get_patient_db
    p = PatientDb.new
    p.mpi = self.mpi
```

```
    p.name = self.name
    p.givename = self.givename
    p.suffix = self.suffix
    p.title = self.title
    p.birthname = self.birthname
    p.birthday = self.brithday
    p.insurance = self.insurance
    p.street = self.street
    p.zip = self.zip
    p.town = self.town
    p.gender = self.gender
    p.phone = self.phone
    p.email = self.email
    p.image = self.image
    p.public_key = self.public_key
    p.save
    return p
  end
end
```

3.4.3 Diagnose

Das Objekt „Diagnose“ beinhaltet eine Diagnose nach dem DRG-System. Bezieht diese sich auf eine bestimmte Untersuchung, zu der ebenfalls Daten vorhanden sind, so ist die ID dieser Datensätze im Objekt hinterlegt. Der Diagnoseschlüssel bezieht sich auf die Systematik des Deutschen Institutes für Medizinische Dokumentation und Information. Die darin kategorisierten Diagnosen können für Verzweigungen innerhalb des Workflows verwendet werden. So kann festgelegt werden, dass bei bestimmten Diagnosen der Patient verlegt wird oder zusätzliche Untersuchungen anberaumt werden müssen. Jedes Objekt vom Typ Diagnose enthält auch das Institutionskennzeichen (IK) der Einrichtung, in der die Diagnose gestellt wurde. Außerdem ist der Arzt hinterlegt, welcher die Diagnose gestellt hat. Weiter befindet sich im Diagnose-Objekt eine kurze Beschreibung und der Typ des Objektes. Dieses Attribut ist ein Integer, dessen Werte zwischen eins und vier liegen. Dabei bedeutet der Wert eins, dass es sich bei der Diagnose um eine Hauptdiagnose handelt. Ist der Wert Typ zwei, so liegt eine Nebendiagnose vor. Die Werte drei und vier weisen die Diagnose als eine Aufnahme- bzw. Entlassungsdiagnose aus.

3.4.4 Value

Das Objekt „Value“ dient dazu, einen Messwert aufzunehmen. Dieses Objekt hat verschiedene Attribute, die angeben, welches Gerät den Messwert gemessen hat, wer das Gerät bedient hat und in welcher Einrichtung die Messung vorgenommen wurde. Außerdem beinhaltet das Objekt noch Vergleichswerte. Diese können wieder im Ablaufplan komplexer Dienste zu Verzweigungen führen. Dabei ist das Objekt absichtlich ganz allgemein gehalten. So kann jeder beliebige Laborwert mit Hilfe dieses Objektes abgebildet werden. Es kann sich jedoch dabei auch um den Blutdruck oder andere Daten in Form von numerischen Werten handeln. Um den Wert interpretieren zu können, wird im Attribut „unit “ die Einheit hinterlegt.

3.4.5 Text

Derzeit existieren im Gesundheitswesen eine ganze Reihe unterschiedlicher Datenaustauschformate, mittels derer unterschiedliche Datensätze zwischen Systemen und Einrichtungen ausgetauscht werden. So kommen Formate wie LDT oder DTA zum Einsatz, um unterschiedliche Patienten- und sonstige Daten auszutauschen. Das Objekt „Text“ dient dazu, Daten in solchen Formaten, mittels eines Basisobjektes der Infrastruktur, zu übermitteln. Es dient unter anderem zur Abwärtskompatibilität. Mit Hilfe dieses Objektes können Daten in Form bestehender ASCII-Formate übertragen und an bereits existierende Softwaresysteme übergeben werden. Zur Beschreibung der Daten wird das Attribut „format“ benutzt. Dort wird der Name des verwendeten Datenformates wie beispielsweise LDT angegeben. Bei manchen Schnittstellen spielt der Name der übertragenen Datei eine entscheidende Rolle. Deshalb sieht das Objekt „Text“ ein Attribut „filename“ vor, welches diesen Namen beinhaltet. Darüber hinaus wird die Quell- und Zielapplikation und die Institution, in welcher der Datensatz erstellt wurde, hinterlegt.

3.4.6 Findings

Das Objekt „Findings“ beschreibt einen Textbefund oder Arztbrief in ASCII-Format. Es besteht aus Informationen über den Ersteller des Textes, den Betreff und nicht zuletzt aus dem eigentlichen Text. Darüber hinaus wird der Patient mit Hilfe des Master Patient Index referenziert. Jeder Befund wurde von einem Arzt diktiert. Im Attribut „doctor“ ist die Arztnummer des Arztes hinterlegt, der den Befund erstellt hat. Ein weiteres Feld „typist“ beinhaltet den Namen der Schreibkraft, die den Brief verfasst hat. Wichtig ist noch das Datum der Entstehung des Befundes und die Unterschrift des Arztes in Form einer elektronischen Signatur. Die Einrichtung, in welcher der Befund entstand, ist ebenfalls im Objekt enthalten.

3.4.7 Date

Das Objekt „Date“ beschreibt einen Termin, den ein Patient bekommt. Es hat folgende Attribute: Zum einen die ID des Objektes „Order“, welches den Auftrag beschreibt, für dessen Ausführung der Termin vorgesehen ist. Ein Attribut von Typ timestamp beschreibt den Zeitpunkt des Termins. Ein Termin besitzt darüber hinaus einen Status. So kann beispielsweise ein Termin versäumt oder vorher abgesagt worden sein. Das Objekt besitzt ebenfalls ein Attribut namens „Condition“, welches Voraussetzungen für das Wahrnehmen des Termins beschreibt. Zu manchen Untersuchungen muss ein Patient beispielsweise nüchtern erscheinen. Ein weiteres Attribut heißt „place“ und gibt den Ort des Termins an. Das letzte Attribut enthält das IK der Einrichtung, welche den Termin vereinbart hat.

3.4.8 Prescription

Das Objekt „Prescription“ beschreibt ein Rezept, welches für einen Patienten ausgestellt wurde. Es enthält alle wichtigen Informationen, die in ein Rezept gehören. Dazu zählen der Name des Arztes, der das Rezept ausgestellt hat, und

dessen LAN-Nummer, die Episode, in deren Zusammenhang das Rezept ausgestellt wurde und die Angaben zum Medikament, wie der Name, der Wirkstoff, die Dosierung, die Packungsgröße und die Menge. Außerdem können noch Hinweise zur Anwendung integriert werden. Wichtig sind auch noch zwei Flags. Das Attribut „out_idem“ gibt an, ob an Stelle des Medikamentes auch ein Generika mit dem gleichen Wirkstoff gegeben werden kann. Ist der Patient von der Zuzahlung befreit, so wird das Attribut „fees_free“ auf den Wert True gesetzt. Das Objekt „Prescription“ beinhaltet außerdem ein Zertifikat des Ausstellers und ist mit dessen geheimen Schlüssel signiert. Darüber hinaus ist die Einrichtung, in welcher der Aussteller tätig ist, hinterlegt. Diese Einrichtung kann die elektronische Unterschrift des Arztes bestätigen.

3.4.9 Data

Ein Data-Objekt dient zum Übertragen von binären Daten. Dies kann z.B. ein Röntgenbild oder ein Diktat sein. Neben den Angaben zu Patient, Datentyp und Format, findet sich im Objekt ein Attribut vom Typ „binary“ namens „data“, welches die eigentlichen Binärdaten enthält. Das Datenbankobjekt kann diese Daten in Form eines „Binary Large Object“ (BLOB) abspeichern. Manche Datenbankmanagementsysteme verlieren jedoch an Performance, wenn man große Mengen an Daten in Form von BLOBs in ihnen ablegt. Für diesen Fall ist es möglich, im Attribut „data“ einen Pfad zu hinterlegen, der auf eine Datei verweist, welche die zum Objekt gehörigen Daten enthält. Somit bildet das Data-Objekt eine Ausnahme, da die Repräsentation in der Datenbank und bei der Übermittlung durch den Webservice voneinander abweichen können. Wie die Daten abgelegt werden, ist vom verwendeten Datenbankmanagementsystem abhängig und für andere Knoten völlig transparent. Mit Webservices ist es möglich, Binärdaten zu kodieren und mittels XML zu übertragen. Dazu werden die Daten in eine Base64-Codierung überführt, wodurch das Volumen der zu übertragenden Daten um etwa ein Drittel anwächst. [SS07] Für viele Daten, die im Gesundheitswesen ausgetauscht werden, stellt dieser Overhead bei heute üblichen Bandbreiten kein Problem dar. Die Vorteile, welche durch die Plattformunabhängigkeit entstehen, wiegen diesen Nachteil auf. Bei gewissen aufwändigen bildgebenden Verfahren, wie MRT-Untersuchungen (Magnetresonanztomografie) müssten zum gegenwärtigen Zeitpunkt geeignetere Verfahren gefunden werden.

3.4.10 Money

Das Objekt „Money“ soll die Umsetzung einer Computational Economy ermöglichen. Da die komplette Infrastruktur auf der Kommunikation über ein Netzwerk basiert, kann davon ausgegangen werden, dass jeder Knoten über eine Netzwerkverbindung verfügt. Aus diesem Grund ist die Verwendung eines Online E-Cash Systems möglich. Das Objekt Money kann demzufolge eine Quittung enthalten, welche dem Empfänger mitteilt, dass eine Transaktion ausgelöst wurde. Denkbar wäre hier allerdings auch das Übermitteln von elektronischen Schecks, die mit dem geheimen Schlüssel des Senders signiert und mit dem öffentlichen des Empfängers verschlüsselt sind. Dieser könnte dann bei einem Kreditinstitut eingelöst werden. Sogenanntes „anonymes elektronisches Geld“ soll ein Pandon zu Bargeld, in Form von Münzen, darstellen. Es soll Fälschungssicherheit, Uni-

versalität, Offline-Verwendbarkeit, Einmalverwendbarkeit, Anonymität, Übertragbarkeit und Teilbarkeit gewährleisten. Bislang existiert noch kein Protokoll, welches diese Forderungen realistisch umsetzt. [PS99] Die Angst vor Missbrauch und die fehlenden Möglichkeiten, das Geld zurückzuverfolgen, würden jedoch vermutlich dazu führen, dass sich solche Systeme nicht durchsetzen könnten. Sollte die Entwicklung in Richtung anonymes elektronisches Geld gehen, so könnten die Zahlungen zwischen den Einrichtungen auch mittels dieser Technologie erfolgen. Zum gegenwärtigen Zeitpunkt ist das Objekt „Money“ jedoch ausschließlich online verwendbar. Das Problem solcher Zahlungssysteme besteht immer darin, dass elektronische Daten kopiert werden können. Stellt eine Einrichtung ein Money-Objekt aus, so kann der Empfänger dieses Objekt problemlos kopieren. Es wäre theoretisch möglich, dass ein solches Objekt gleichzeitig bei zwei verschiedenen Kreditinstituten eingelöst wird, die voneinander nichts wissen. Seit dem 1. Januar 2007 wird das Gesundheitssystem ausschließlich über den Gesundheitsfond finanziert. Das heißt, alle Beiträge fließen in diesen Fond und werden dann durch die Krankenkassen an die entsprechenden Dienstleister ausgezahlt. Somit existiert eine zentrale Stelle, die jeder Geldstrom im Gesundheitswesen durchläuft. Dieser Gesundheitsfond kann somit als zentrale Bank für das Gesundheitswesen dienen, über welche alle Zahlungen abgewickelt werden. Diese zentrale Stelle könnte also für ein Online-Zahlungssystem verwendet werden.

3.4.11 Job

Das Job-Objekt beinhaltet einen Job in Form einer XML-Datei im HSML-Format. Diese XML-Datei kann vom Empfänger abgearbeitet werden und liefert einen verdichteten Wert für statistische Zwecke zurück. Das Objekt kann dazu verwendet werden, um aus Knoten bestimmte Informationen zu ermitteln, ohne konkrete Daten von Patienten zu übertragen. So muss von den Kassen jedes Jahr der Landesbasisfallwert ermittelt werden. Dafür werden von einigen Knoten Daten erhoben, die als Grundlage für dessen Berechnung dienen. Mit dem Job-Objekt können solche Daten automatisch ermittelt werden.

3.4.12 Institution

Das Objekt „Institution“ beschreibt eine Einrichtung. Es enthält alle wichtigen Informationen über einen Knoten. Dazu gehört der öffentliche Schlüssel der Einrichtung. Weiter enthält das Objekt das Institutionskennzeichen der Einrichtung und den geographischen Standort sowie die Adresse des Knotens. Das ist wichtig, um Transportwege abschätzen zu können. Eine wichtige Information ist die Klasse einer Einrichtung. Hier ist hinterlegt, ob es sich bei der Institution um ein Krankenhaus, eine Arztpraxis, eine Krankenkasse oder eine Apotheke handelt. Darüber hinaus existiert noch ein Attribut, welches die Einrichtung genauer beschreibt. Beispielsweise kann ein Krankenhaus ein Schwerpunkthaus oder ein Regelversorger sein. Bei einer Arztpraxis kann es sich ebenfalls um eine Facharztpraxis einer bestimmten Fachrichtung handeln. Weiter enthält das Objekt Informationen über die IP-Adresse und den Port, auf dem die Einrichtung angesprochen werden kann.

3.4.13 Order

Um nicht nur Daten zu übertragen, sondern auch einen Workflow abbilden zu können, ist es unerlässlich, einem anderen Knoten Aufträge erteilen zu können. Das Order-Objekt übernimmt diese Aufgabe. Es übermittelt ein Attribut namens „application“. Dieses Attribut beschreibt die Anwendung, die auf dem Zielknoten aufgerufen werden soll. Diese Anwendungen werden von Knoten für andere Knoten bereitgestellt und können in einen Workflow eingebunden werden. Aufträge kann man in verschiedene Gruppen unterteilen. Für die Erfüllung mancher Aufträge ist die Anwesenheit des Patienten nötig. Dies ist beispielsweise bei Untersuchungen der Fall. Andere Aufträge setzen die Anwesenheit nicht voraus. Hier wäre eine Laboruntersuchung zu nennen, bei der lediglich eine Blutprobe transportiert werden muss. Bei manchen Aufträgen verbleibt die Behandlungsverantwortung beim Auftraggeber, während wieder bei anderen nur eine Dienstleistung erbracht wird. Das Objekt „Order“ ist in dieser Beziehung allgemein gehalten. Um welchen Typ von Auftrag es sich handelt, ergibt sich aus dem Zusammenhang. Verbleibt die Verantwortung für den Patienten beispielsweise beim Auftraggeber, so wird innerhalb des Prozesses kein Patient-Objekt übertragen. Wichtig für einen Auftrag sind lediglich die Fragestellung, der Name der Anwendung und die Dringlichkeit. Darüber hinaus stellt das Objekt „Order“ ein Attribut zur Verfügung, welches es erlaubt, zusätzliche wichtige Parameter als Freitext zu übergeben. Das Attribut „require_date“ zeigt dem Empfänger eines Order-Objektes an, ob der Kommunikationspartner einen konkreten Termin benötigt oder nicht. Hat das Attribut den Wert False, so benötigt der Auftraggeber keinen Termin. Der Auftrag soll so schnell wie möglich abgearbeitet werden. Im anderen Fall verlangt der Auftraggeber ein Objekt vom Typ date_oj zurück.

3.4.14 Basics

Wenn ein Auftrag an eine andere Einrichtung erteilt wird, sind oftmals bestimmte medizinische Grunddaten für die Ausführung dieses Auftrages nötig. Eine MRT-Untersuchung kann beispielsweise nicht stattfinden, wenn der Patient ein Metallimplantat eingesetzt bekommen hat. Auch bestimmte Allergien müssen für manche Untersuchungen bekannt sein. Diese Informationen sind in der Regel dann von Bedeutung, wenn der Patient in einer anderen Einrichtung untersucht werden soll. Behält der Auftraggeber die Verantwortung für den Patienten, dann müssen keine Patientenstammdaten in Form des Objektes „Patient“ an den Auftragnehmer übermittelt werden. Trotzdem sind die medizinischen Grunddaten unter Umständen sehr wichtig. Um diese Daten übertragen zu können, wurde das Objekt „Basics“ definiert. Es enthält keine Daten, anhand derer ein Patient eindeutig identifiziert werden könnte, sondern ausschließlich solche Daten, die für die Vorbereitung einer Untersuchung wichtig sein könnten. So wird in diesem Objekt kein Geburtsdatum übertragen, sondern lediglich das Geburtsjahr, welches für die Bestimmung des Alters nötig ist. Das Alter ist für viele Untersuchungsmethoden ein wichtiger Parameter. Weiter besteht das Objekt aus Attributen, welche medizinische Daten des Patienten beschreiben. Diese ändern sich im Normalfall sehr selten. So wird die Blutgruppe, die Existenz von Metallimplantaten, bekannte Allergien, das Geschlecht und ob es sich bei dem Patienten um einen Raucher handelt, sowie das Vorhandensein

einer Diabeteserkrankung mit dem Objekt „Basics“ übertragen. Eine Ausnahme bildet die aktuelle Medikation. Diese kann sich sehr häufig ändern, ist aber dennoch im Objekt „Basics“ enthalten. Diese Informationen können für manche Untersuchungen von entscheidender Bedeutung sein. Soll beispielsweise eine Gastroskopie durchgeführt werden, so sollte die Einnahme eines blutverdünnenden Medikamentes dem untersuchenden Arzt bekannt sein.

3.4.15 Edifact

Im Gesundheitswesen wird bereits seit dem Jahr 2000 elektronisch mit den Kostenträgern abgerechnet. Dafür wird das Format UN/Edifact verwendet, welches vom „United Nations Economic Commission for Europe“ standardisiert wurde. [Uni08] Die verschiedenen Versionen von Edifact werden Verzeichnisse genannt. Der Standard wird ständig weiterentwickelt, um immer neue Nachrichten aufzunehmen. Es handelt sich dabei lediglich um ein Datenformat, das unabhängig vom Transportprotokoll definiert wurde. Das Objekt „Edifact“ beschreibt eine solche Nachricht mit bestimmten Zusatzinformationen. Es beinhaltet ein Attribut, welches das Verzeichnis der Nachricht beschreibt. Ein anderes enthält das Datum der Nachricht. Die Nachricht befindet sich ebenfalls in einem Attribut des Objektes „Edifact“. Da Edifact-Daten oft durch sogenannte Annahmestellen entgegengenommen und an die eigentlichen Kostenträger weitergeleitet werden, beinhaltet das Objekt „Edifact“ ein Attribut namens „destination“. Darin wird die IK-Nummer des Empfängers der Nachricht hinterlegt. Die Annahmestelle kann die Nachricht an diesen Empfänger weiterleiten.

3.4.16 Connection

Das Objekt „Connection“ beschreibt eine Verbindung zu einem anderen Knoten. Mittels der darin befindlichen Informationen kann eine Verbindung zu einem anderen Knoten aufgebaut werden. Das Konzept der Module wurde in Kapitel 3.2 unter anderem deshalb eingeführt, um Kommunikationsprozesse, welche nicht mit Hilfe von Webservices abgebildet werden können, zu realisieren. Das Objekt enthält wichtige Parameter, mit deren Hilfe ein solches Modul eine Verbindung über das Protokoll TCP/IP zu einem Modul auf einem anderen Knoten aufbauen kann. Als Beispiel soll in dieser Arbeit eine Videokonferenz dienen, die über das Modul Hstream realisiert wurde. Zu diesen wichtigen Verbindungsinformationen gehören die IP-Adresse und der Port, über den die Verbindung aufgebaut werden soll. Weiter ist das Protokoll von Bedeutung, mit dessen Hilfe die Kommunikation stattfinden soll. Anhand dieses Attributes kann die Anwendung entscheiden, welches Modul für die Kommunikation verwendet werden muss. Dieses kann von Einrichtung zu Einrichtung variieren. Mit dem Objekt „Connection“ werden außerdem die Attribute „speed“, „encryption“ und „key“ übergeben. Das Attribut „speed“ gibt dabei die benötigte Bandbreite an. Das Attribut „encryption“ zeigt an, ob für die Kommunikation eine Verschlüsselung verwendet werden soll und wenn ja, so kann ein Schlüssel im Attribut „key“ hinterlegt werden. Die Verarbeitung dieser Werte ist jedoch sehr stark vom verwendeten Protokoll abhängig. Die Module müssen die Verschlüsselung unterstützen. Andernfalls müssen die beiden Parameter ignoriert werden.

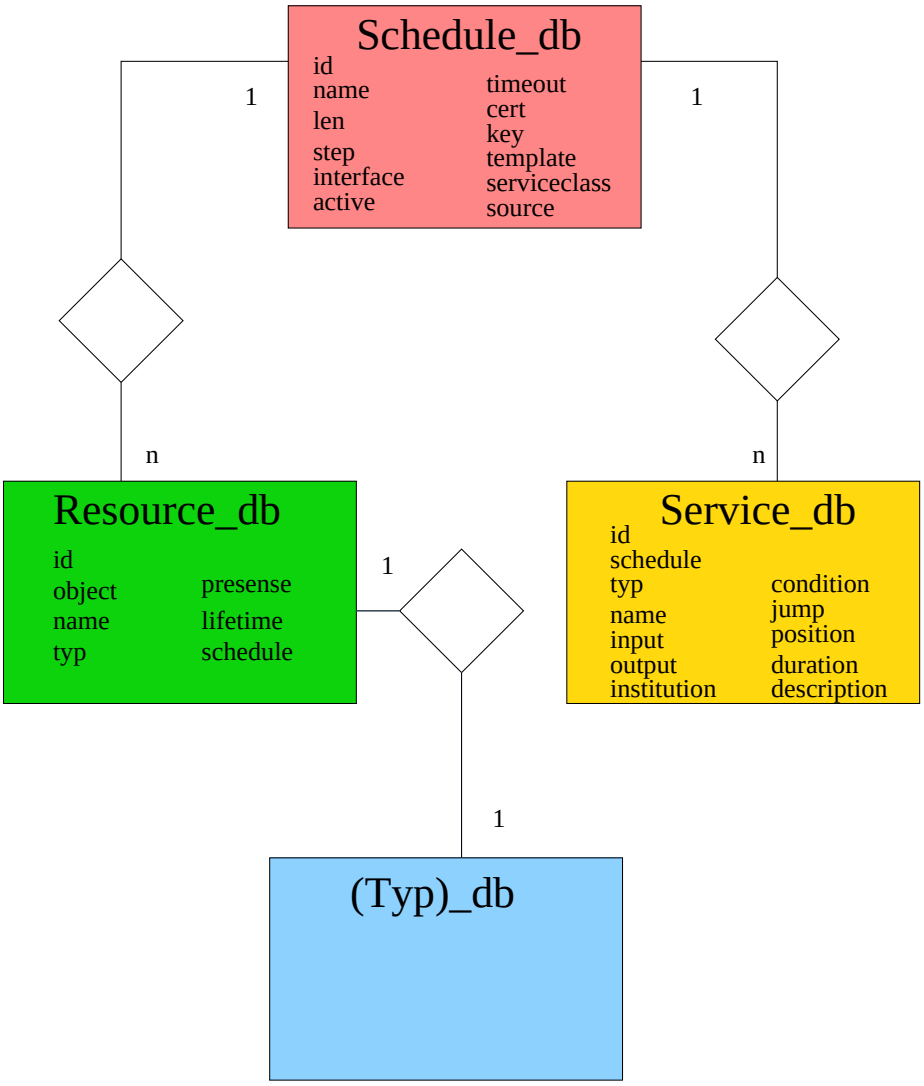

Abbildung 17: ER-Diagramm Schedule

3.5 Basisobjekte zur Steuerung

Die folgenden drei Basisobjekte bilden eine Einheit. Abbildung 17 zeigt mit Hilfe eines Entity-Relationship-Diagramms, in welcher Beziehung die drei, in den folgenden Abschnitten beschriebenen, Basisobjekte stehen. Dabei können einem Schedule n Ressourcen und ebenfalls n Services zugeordnet sein.

3.5.1 Schedule

Moderne Informationssysteme im Krankenhaus unterstützen das Arbeiten mit Behandlungspfaden, welche oft auch Clinical Pathways genannt werden. Solche Pfade dienen dazu, standardisierte Abläufe der Behandlung zu beschreiben und zu steuern. Durch Behandlungsabläufe und klinische Pfade kann die Qualität der Behandlung stark verbessert werden. Darüber hinaus ist es damit möglich, die Dauer der Behandlung zu steuern und gegebenenfalls zu straffen. Das ist wichtig, da durch das DRG-System für jede Behandlung eine untere Grenzverweildauer festgelegt ist. Wird ein Patient vorher entlassen, muss mit Einbußen beim Erlös gerechnet werden. Bleibt ein Patient zu lange in stationärer Behandlung, verursacht er Kosten, die nur in Ausnahmen zu Mehrerlösen führen. Eine Kontrolle und Steuerung der Verweildauer im Krankenhaus ist daher unbedingt notwendig. Die Clinical Pathways werden individuell von den Ärzten entwickelt und ständig an den aktuellen Stand der Forschung angepasst.

Das Objekt „Schedule“ soll es ermöglichen, ähnliche Ablaufpläne aus den Basisdiensten zu erstellen und so komplexere Abläufe zu entwerfen. Ziel ist es, Ärzten zu ermöglichen, Schedules selbstständig zu entwerfen und damit Prozesse zu definieren, die über die eigene Einrichtung hinausgehen. Dabei steht jedoch nicht wie bei den klinischen Pfaden die Steuerung der Behandlungsdauer im Vordergrund, sondern die reibungslose Zusammenarbeit mehrerer Einrichtungen. Das Objekt „Schedule“ dient jedoch nicht ausschließlich der Steuerung von Abläufen, sondern hat auch eine wichtige technische Bedeutung. Es soll mehrere Aufrufe von Webservices verbinden und einen Zusammenhang zwischen den einzelnen Anfragen herstellen. Deshalb werden alle Dienste, welche Daten zwischen verschiedenen Einrichtungen übertragen, mit Hilfe eines Schedule-Objektes abgebildet. In diesem Objekt können dann Statusinformationen eines Vorgangs hinterlegt werden. Dazu enthält jeder Schedule eine eindeutige Schedule-ID. Die Eindeutigkeit wird dadurch sichergestellt, dass die Schedule-ID sich aus dem IK der Einrichtung, die den Schedule erstellt hat und einer fortlaufenden Nummer zusammensetzt. Das Objekt „Schedule“ wird an die beteiligten Knoten gesendet. Jeder Aufruf eines Basisdienstes wird einem Schedule-Objekt zugeordnet, indem dabei eine Schedule-ID übermittelt wird. Ein Objekt vom Typ Schedule beschreibt einen Dienst, der einer bestimmten Klasse zugeordnet werden kann. Diese Klasse bestimmt, welche Datenobjekte innerhalb der Ausführung des Schedules benutzt werden dürfen. Die Klassen von Diensten werden in Kapitel 4 ausführlich beschrieben. Ein Objekt vom Typ Schedule wird an einen anderen Knoten mittels des Webservice „put_schedule“ übertragen. Dieser kann dann prüfen, ob der Schedule abgearbeitet oder aus Gründen der Sicherheit oder des Datenschutzes abgelehnt wird.

Beim Übertragen eines Schedule-Objektes werden gleichzeitig wichtige kryptographische Informationen ausgetauscht. Beispielsweise enthält dieses Objekt ein signiertes Zertifikat, anhand dessen der Kommunikationspartner eindeutig

und zweifelsfrei identifiziert werden kann und einen Schlüssel, mit dessen Hilfe die den Schedule betreffende Kommunikation chiffriert wird. Die Struktur eines Schedules wird, wie alle Basisobjekte, in einer relationalen Datenstruktur abgebildet. Das Attribut „active“ gibt an, ob sich der Schedule auf dem betreffenden Knoten gerade in der Ausführung befindet. Jedes Objekt vom Typ Schedule hat einen Zeitstempel, der angibt, zu welchem Zeitpunkt das Objekt übertragen wurde. Weiter existiert ein Attribut „timeout“. Dieses gibt an, wie lange ein Schedule gespeichert werden soll. Ist das Verfallsdatum des Objektes erreicht, muss man davon ausgehen, dass die anderen beteiligten Knoten die Verarbeitung dieses Prozesses bereits abgebrochen haben und das Objekt wird gelöscht. Zu einem Schedule gehört eine Liste von Ressourcen. Diese bilden den Datenbereich des Prozesses. Alle während der Verarbeitung benötigten Daten werden im Vorfeld festgelegt. Während der Abarbeitung der einzelnen Basisdienste des Schedules werden die Ressourcen dann mit Datenobjekten verknüpft. Das Attribut „template“ gibt an, ob es sich beim Datenbankobjekt um ein Template oder ein konkretes Objekt handelt, dem ein Schedule in Ausführung zugeordnet ist.

3.5.2 Service

Die Service-Objekte definieren die Schritte, welche bei der Ausführung eines Schedules abgearbeitet werden. Ein Service-Objekt beinhaltet neben dem Namen, dem Typ, seiner Position im Schedule und seiner Dauer auch Informationen darüber, welches Objekt der Services als Eingabe benötigt und welches Objekt von ihm zurückgegeben wird. Hier sind die Namen der Ressourcen hinterlegt. Weiter enthält das Service-Objekt eine Sprungbedingung. Ist diese erfüllt, wird der Befehlszähler nicht inkrementiert, sondern auf den Wert gesetzt, der im Attribut „Jump“ hinterlegt ist. Außerdem enthält das Service-Objekt das Attribut „institution“, welches angibt, welche Einrichtung den Service ausführen soll. Das Attribut „timestamp“ gibt an, wann ein Service ausgeführt werden soll. Besitzt es nicht den Wert null, so referenziert es eine Ressource vom Typ Date. Erst ab diesem Zeitpunkt wird der Service bei der nächstmöglichen Gelegenheit abgearbeitet. Knoten können Serviceobjekte mittels der Webservice „get_service “ anderen Knoten zur Verfügung stellen. Diese können die Objekte dann importieren, um sie in individuell gestaltete Schedules zu integrieren.

3.5.3 Ressourcen

Die Kommunikationstechnik, mit deren Hilfe die einzelnen Knoten Dienste abrufen, sind Webservices. Webservices basieren auf HTTP und besitzen deshalb keine Statusinformationen. Manche komplexeren Dienste bedürfen aber einer längeren Bearbeitungszeit. Es ist deshalb notwendig, dass ein Dienst Statusinformationen ablegt. Wird beispielsweise ein Termin für eine Untersuchung in einer anderen Einrichtung vergeben, dann muss das Ergebnis der Untersuchung später dem richtigen Auftrag zugeordnet werden. Verschiedene Aufrufe werden mittels einer „sched_id“ einem Vorgang zugeordnet. Wird beim Aufruf eines Basisdienstes eine solche ID angegeben, so kann ein Zusammenhang zwischen Basisdiensten eines Schedules hergestellt werden. Diesem Ablauf müssen aber auch Datensätze zugeordnet werden können, welche durch den Vorgang bearbeitet werden sollen. Dazu wird auf jedem Knoten eine Datenbanktabelle

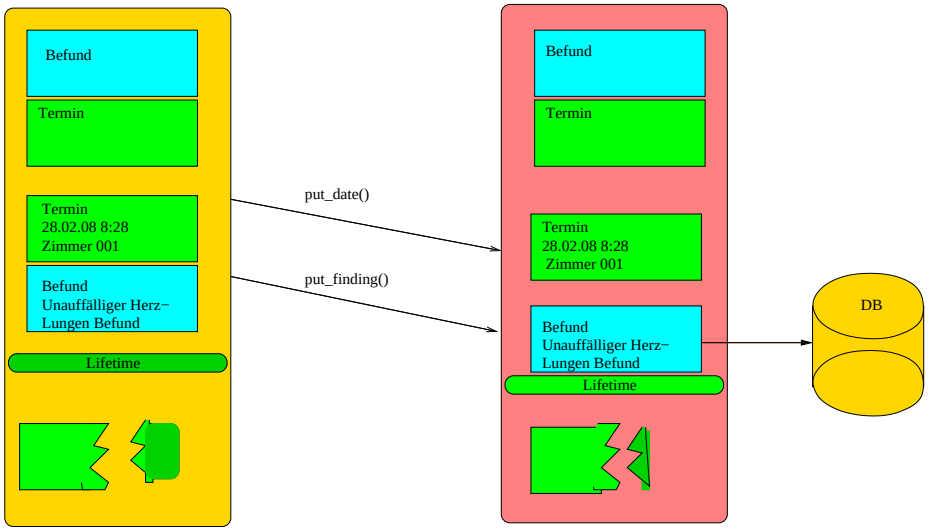

Abbildung 18: Ressourcen

„resource_dbs“ eingerichtet, in der Objekte vom Typ „Resource“ abgelegt werden. Diese Tabelle besitzt sieben Attribute. Das erste Attribut ist die ID der jeweiligen Ressource. Diese ID ist lokal eindeutig. Zweitens besitzt eine Ressource einen Namen, mit dessen Hilfe sie während der Bearbeitung referenziert werden kann. Beim dritten Attribut handelt es sich um den Typ des Objektes, auf welches die Ressource verweist. Ist dieses Objekt beispielsweise ein Datenobjekt, dann hat es den Wert „data“. Das vierte Attribut ist die ID des Objektes. Dieses Attribut beinhaltet eine Referenz auf ein bestimmtes Datenobjekt. Als fünftes Attribut beinhaltet die Tabelle die Spalte „schedule“, welche auf das dazugehörige Schedule-Objekt verweist. Somit ist es möglich, einem Vorgang ein oder mehrere konkrete Objekte zuzuordnen. Das sechste Attribut einer Ressource heißt „lifetime“. Es gibt an wie, lange eine Ressource auf einem Knoten verbleiben darf.

Im Normalfall wird die Ressource nach Ausführung des Schedules gelöscht. Die dazugehörigen Datenobjekte können jedoch noch vorhanden bleiben. Das Ressourcen-Objekt stellt also den Bezug eines Datensatzes zu einem in Ausführung befindlichen Dienst dar. Wird ein Schedule-Objekt an einen anderen Knoten geschickt, dann werden die entsprechenden Ressourcen angelegt. Diese verweisen teilweise noch nicht auf konkrete Daten. Im Verlauf der Verarbeitung werden die Daten mit den entsprechenden Ressourcen verknüpft und stehen dann während der Abarbeitung des Vorgangs zur Verfügung. Wie das geschieht wird mit Hilfe des siebten Attributes namens „presence“festgelegt. Es kann die Werte eins bis drei annehmen, von denen die eins bedeutet, dass die Ressource überhaupt nicht auf andere Knoten übertragen werden darf. Hat das Attribut den Wert zwei, so kann ein Knoten, der in die Verarbeitung eines Ablaufplanes involviert ist und damit im Besitz des dazugehörigen Schedule-Objektes ist, eine Ressource gezielt anfordern, sobald diese für die Bearbeitung benötigt wird. Hat „presence “den Wert zwei, so bedeutet dies, dass das zur Ressource gehörige Objekte sofort an alle beteiligten Knoten übertragen wird. Ein Röntgenbild, das zu einer

Ressource gehört, deren Attribut „presence“ auf drei gesetzt ist, wird sofort nach dessen Archivierung an alle beteiligten Knoten gesendet. Abbildung 18 zeigt, wie die Ressourcen auf beiden Knoten angelegt wurden. Sobald eine Ressource in einem der beiden Knoten mit einem Datenobjekt verknüpft wurde, wird dieses Objekt an den anderen Knoten übertragen und dem dortigen Schedule-Objekt zugeordnet. Die Grafik stellt den zeitlichen Verlauf von oben beginnend dar.

3.5.4 Error

Das Error-Objekt enthält Informationen über Fehler, die während der Verarbeitung eines Schedules auftreten können. Das Objekt besitzt eine eindeutige ID. Außerdem beinhaltet es die ID des Schedules und dessen Namen, sowie die Position des Arbeitsschrittes, bei dessen Verarbeitung der Fehler verursacht wurde. Weiter kann eine Fehlerklasse angegeben werden. Damit können Fehler in Schutzverletzungen, Timeouts, Bedienfehler, Modulfehler oder Kommunikationsfehler kategorisiert werden. Das Objekt besitzt außerdem ein Attribut, in dem der Zeitpunkt hinterlegt ist, zu welchem der Fehler auftrat. Darüber hinaus kann ein Modul hinterlegt werden, welches den Fehler verursacht hat. Das Attribut „rule“ beschreibt eine Regel, bei der eine Schutzverletzung aufgetreten ist. Weiter beinhaltet das Objekt eine verbale Beschreibung und die Kontaktdaten eines Ansprechpartners der Einrichtung, welche den Fehler verursacht hat.

3.6 Zugriff auf Basisobjekte

Die Basisdienste sind Dienste, die selbst keinen Status besitzen. Sie bestehen lediglich aus dem Aufruf eines Webservices. Es gibt unterschiedliche Arten von Basisdiensten. Wie in den Kapiteln 3.2.1 und 3.2.2 beschrieben wurde, kann man sie in Dienste unterteilen, welche mit anderen Knoten kommunizieren und solche, die mit Modulen innerhalb des Knotens Daten austauschen. Weiter werden Dienste zur Steuerung und solche zur Übertragung der in den vorigen Abschnitten vorgestellten Basisobjekte unterschieden. In Abbildung 19 sind die verschiedenen Basisdienste optisch gruppiert. Die Funktionalität letzterer Dienste beschränkt sich darauf, Basisobjekte entweder anzufordern oder zu versenden. Dazu existiert für jedes Objekt eine get- und eine put-Methode - z.B. get_patient und put_patient. Beim Aufruf eines Basisdienstes wird eine sched_id übergeben. Ist diese ID gleich null, dann handelt es sich um einen direkten Aufruf eines Basisdienstes. Die meisten Basisdienste dürfen jedoch nur innerhalb einer Einrichtung direkt aufgerufen werden. Patientendaten dürfen beispielsweise nur dann übertragen werden, wenn die Kommunikationspartner sich vorher gegenseitig authentifiziert haben. Dies geschieht im Rahmen der Abarbeitung eines Schedules. Hat die sched_id einen Wert ungleich Null, so gibt sie das Schedule-Objekt an, auf welches sich der Aufruf bezieht. Der Service put_schedule wird mit der sched_id null aufgerufen, da dieser Service zu Beginn der Ausführung eines zusammengesetzten Dienstes aufgerufen wird. Eine Einrichtung nutzt diesen Service, um einer anderen Einrichtung den Ablaufplan für einen komplexen Vorgang zu übermitteln. Dieser liegt dann beiden Knoten vor und wird abgearbeitet.

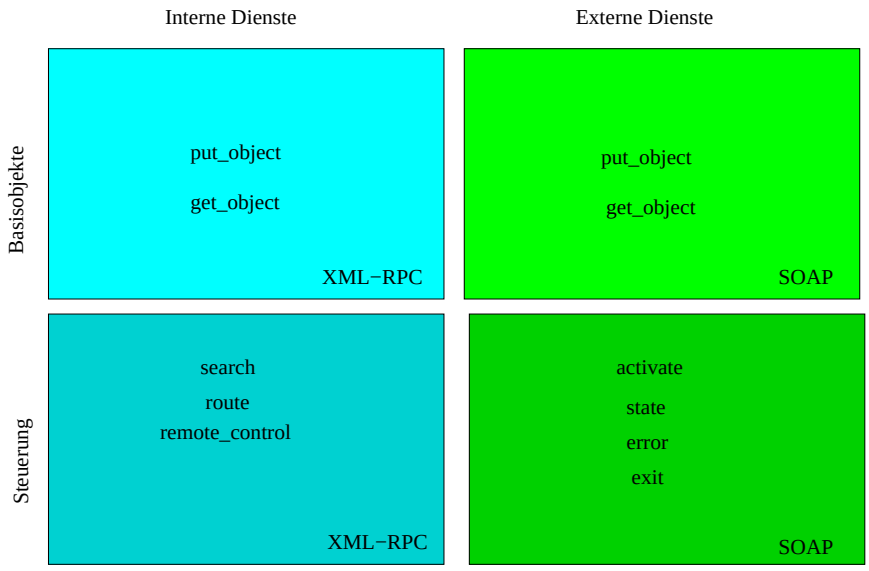

Abbildung 19: Basisdienste

3.7 Externe Steuerungsdienste

Um die Schedules abzuarbeiten sind Dienste für die Steuerung nötig. Der Dienst „activate“ aktiviert einen Schedule auf einem beteiligten Knoten. Der Dienst benötigt zwei Argumente. Zum einen die ID des Schedules und zum zweiten den Prozessschritt, welcher als nächstes zur Ausführung kommen soll. Ein anderer Dienst dient zum Abruf von Statusinformationen. Mit „state“ kann der Bearbeitungsstatus eines Schedules an andere beteiligte Knoten gesendet werden. Dieser Dienst überträgt ebenso wie der Dienst „activate“ eine sched_id und den aktuellen Prozessschritt. Allerdings dient der Aufruf dieses Dienstes als Information. Am Prozess beteiligte Knoten können so den Status der Bearbeitung verfolgen. Ein dritter Dienst zur Steuerung ist „exit“. Dieser Dienst veranlasst einen Knoten, den Prozess zum dazugehörigen Schedule-Objekt zu unterbrechen und dieses Objekt samt allen damit verbundenen Ressourcen zu löschen. Der Dienst unter dem Namen „error“ übermittelt an einen Knoten eine Fehlermeldung. Dabei wird ebenfalls eine sched_id übertragen, die den Bezug zum Prozess herstellt, bei dessen Verarbeitung der Fehler aufgetreten ist. Weiter wird der Arbeitsschritt, in dem es zum Fehler kam, und ein Fehlercode samt dazugehöriger Fehlerbeschreibung, übertragen. Der Erhalt eines Error-Objektes soll dem Kommunikationspartner Aufschluss über das Auftreten und die Ursache eines Fehlers geben. Beispielsweise könnte eine Ressource, die während der Bearbeitung eines Schedules adressiert wird, nicht mehr vorhanden sein. Wird ein Prozess auf Grund eines internen Fehlers auf einem Knoten abgebrochen, so muss der Dienst „error“ auf allen beteiligten Knoten aufgerufen werden.

3.8 Interne Steuerungsdienste

3.8.1 Basisdienst für die Authentifikation

Ein für die Datensicherheit wichtiger Basisdienst ist der Dienst „auth“. Dieser Dienst fragt bei einem Modul nach der Identität eines anderen Teilnehmers. Der Dienst soll universell einsetzbar sein und sich deshalb sowohl auf Einrichtungen, als auch auf konkrete Nutzer anwenden lassen. Als Argument wird ein digitales Zertifikat übergeben, welches Informationen über den Teilnehmer enthält. Unter anderem beinhalten diese Informationen auch den öffentlichen Schlüssel der Identität, als die sich der Kommunikationspartner ausgibt. Als Rückgabewert wird eine Datenstruktur übertragen, in welcher für jede in der Infrastruktur definierte Rolle ein Wert vom Typ Boolean hinterlegt ist. Anhand dieser Liste kann ein Knoten erkennen, welche Berechtigungen der Teilnehmer besitzt. Wird eine Liste ausschließlich aus dem Wert False bestehend als Ergebnis zurückgeliefert, so bedeutet das, dass der Teilnehmer unbekannt ist und deshalb keine Berechtigungen erhält. Teilnehmer ohne jegliche Berechtigung, werden nicht benötigt und existieren damit in der Infrastruktur nicht. Dieser Basisdienst wird als interner Dienst verwendet und an ein Modul geschickt, welches das Zertifikat überprüft. Ist ein Zertifikat ungültig, weil beispielsweise das Gültigkeitsdatum abgelaufen ist oder die Signatur nicht stimmt, so wird eine Fehlermeldung zurückgegeben.

3.8.2 Basisdienst für die Suche

Der Basisdienst „search“ dient dazu, Patientendaten in anderen Einrichtungen zu lokalisieren. Der Dienst wird als interner Dienst realisiert, der auf ein Modul zugreift, das die Suche durchführt. Ein Aufruf des Dienstes liefert Objekte vom Typ „Institution“ zurück, die auf Einrichtungen verweisen, in welchen sich Patientendaten befinden. Dabei kann das Modul eine Suche via LDAP durchführen oder mittels P2P-Technik Knoten ermitteln, auf denen sich Patientendaten befinden. Die Funktionsweise des Moduls wird in Kapitel 5 ausführlich beschrieben. Damit das Modul eine Suche durchführen kann, benötigt es bestimmte Parameter. Zum einen muss der, mit Hilfe einer Einwegfunktion verschlüsselte Master Patient Index des Patienten, kombiniert mit einem Autorisierungsschlüssel, der Suchfunktion übergeben werden. Bei den Autorisierungsstufen handelt es sich um drei Pins, die in Kapitel 6.10 eingeführt werden. Demzufolge besitzt dieses Attribut, welches „mpi_hash“ genannt wird, drei verschiedene verschlüsselte Werte. Der MPI wird mit jeweils einer PIN kombiniert und verschlüsselt. Eine Einrichtung, die im Besitz der ID des Patienten ist, kann erkennen, dass nach diesem speziellen Patienten gesucht wird, indem sie die Hashwerte aller MPIs mit den ihr vorliegenden Autorisierungsschlüsseln kombiniert, den Hashwert bildet und mit dem Hashwert der Suchanfrage vergleicht. Andere Knoten können den Hashwert aus MPI und Autorisierungsschlüssel nicht bilden und sind deshalb nicht in der Lage zu ermitteln, welcher Patient gesucht wurde.

Es muss darüber hinaus auch gewährleistet werden, dass nur die Knoten eine Suche durchführen können, welche gerade mit der Behandlung des Patienten betraut worden sind. Bei einer Suche ausschließlich mittels des Hashwertes könnte eine Einrichtung, in welcher der Patient vor längerer Zeit behandelt wurde, auch nach Jahren noch nach dem Patienten suchen und würde als Ergebnis eine Liste aller Institutionen erhalten, in welcher sich dieser Patient aufgehalten

hat. Das wäre ein schwerer Verstoß gegen die Grundsätze des Datenschutzes. Auch wenn es mit diesen Informationen nicht möglich wäre, die Daten selbst zu empfangen, so könnte die Auskunft darüber, dass sich der Patient in einer bestimmten Einrichtung befunden hat, für den Patienten bereits beträchtlichen Schaden hervorrufen. Deshalb muss als weiterer Parameter ein Zertifikat - signiert mit dem geheimen Schlüssel des Patienten - übergeben werden, welches das aktuelle Datum enthält. Der Dienst übermittelt dieses Zertifikat im Attribut „cert“. Ein Knoten kann damit prüfen, ob eine Suchanfrage von einem Knoten stammt, der innerhalb eines bestimmten Zeitfensters mit der Behandlung des Patienten beauftragt war.

Im Kapitel 5 wird ausführlich diskutiert, welche Algorithmen für eine Suche nach Patientendaten gute Ergebnisse liefern, ohne zentrale Knoten, für die Verwaltung von Indizes, einzusetzen. Da Daten von Patienten nicht beliebig anfallen, sondern meist in Einrichtungen, die in irgendeiner Beziehung zum Patienten stehen, können bestimmte Informationen des Patienten wie der Wohnort hilfreich für die Suche sein. Dazu wird ein Parameter namens „target“ übergeben, der beschreibt, in welcher Region sich Patientendaten befinden können. Ein weiterer Parameter, der hilfreich für die Suche sein kann, ist der Parameter „diagnose“. Dieser wird mittels des gleichnamigen Basisobjektes aus Kapitel 3.4.3 an das Modul übergeben. Da es sich um einen internen Dienst handelt, wird eine Suchanfrage mittels XML-RPC an das Modul übergeben, das die Suche durchführt.

3.9 Verarbeitung von Schedules

Die in den vorigen Abschnitten vorgestellten internen und externen Basisdienste dienen als Grundlage für komplexere Dienste. Wichtig dafür sind die Service-Objekte und die Ressourcen-Objekte, die bereits eingeführt wurden. Diese beschreiben einen Ablaufplan, der einem Schedule-Objekt zugeordnet ist und von verschiedenen Knoten gemeinsam abgearbeitet werden kann. Dabei beschreibt der Schedule das sogenannte „observable behavior“, also das Verhalten der unterschiedlichen Knoten, wie es ein Beobachter von außen beschreiben würde. Das bedeutet, dass ein Schedule und die zugehörigen Service-Objekte die genauen Arbeitsschritte nicht detailliert beschreiben. Es wird nur festgelegt, welche Einrichtung eine Aufgabe durchführen muss, welche Eingangswerte sie dazu benötigt und welche Ergebnisse erwartet werden. Jede Einrichtung muss in der Lage sein, sich Schedules selbst zusammenzustellen. Manche dieser Schedules müssen zwar in allen Einrichtungen definiert sein, andere können sich jedoch von Einrichtung zu Einrichtung, je nach Bedarf in Struktur und Namen unterscheiden. Aus diesem Grunde ist es nicht sinnvoll, dass eine Einrichtung einen Schedule eines anderen Knotens mittels der Methode „get_schedule“ anfordert. Statt dessen muss jeweils die Einrichtung, welche einen Schedule erstellt, diesen an andere Knoten mittels der Methode „put_schedule“ übermitteln. Eine Einrichtung, die einen Schedule an einen anderen Knoten sendet, wird im folgenden Initiator genannt. Dabei läuft die Verarbeitung eines Schedules in fünf Phasen ab.

Die erste Phase ist die Übertragung des Schedule-Objektes. Initiiert ein Knoten die Ausführung eines Schedules, so wird das für den gewünschten Schedule erstellte Template in der Datenbank samt allen Service-Objekten und Ressourcen geklont und mit einer neuen ID versehen. Dieses Objekt wird dann

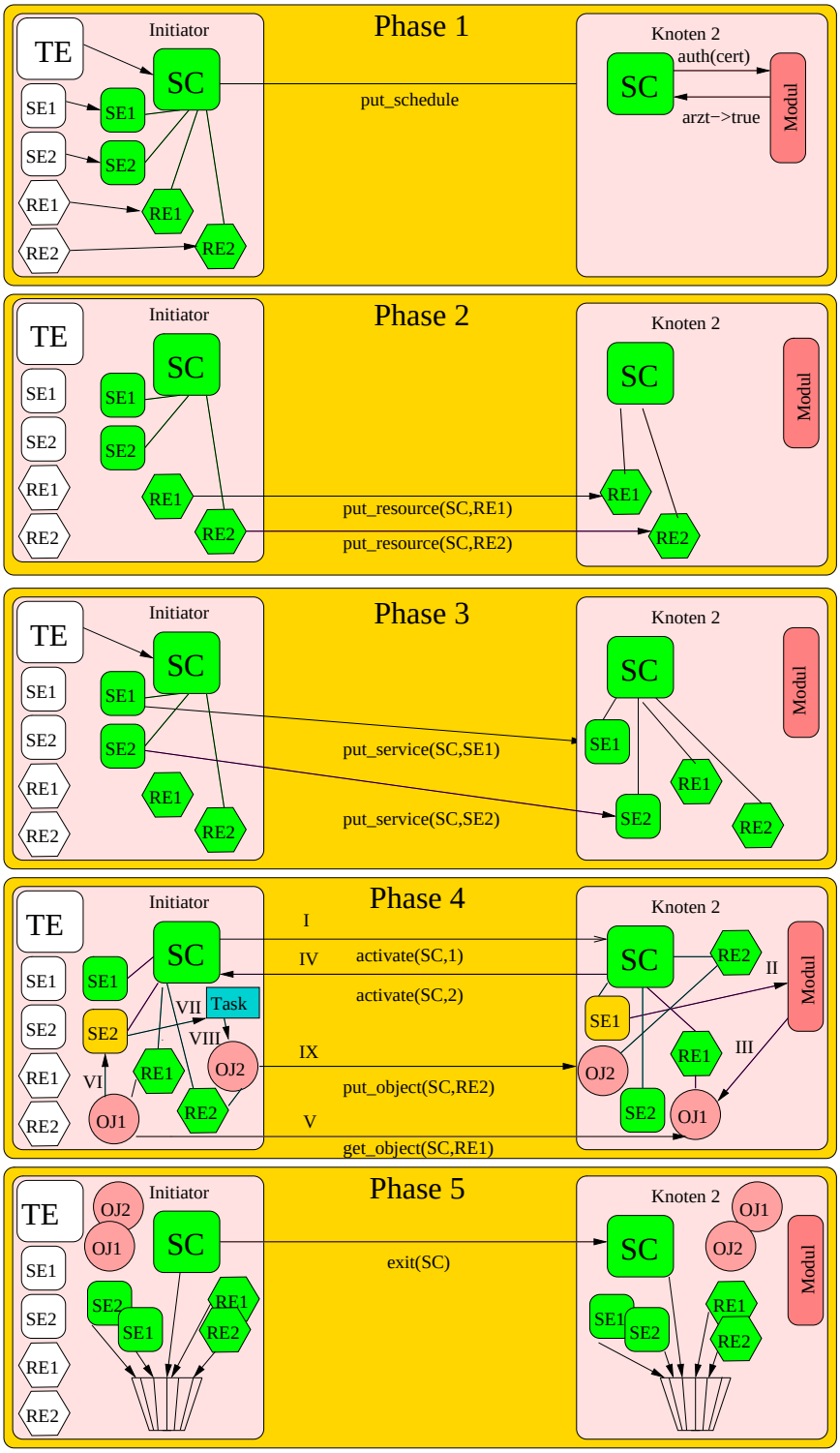

Abbildung 20: Verarbeitung eines Schedules mit zwei Schritten

an alle in den Service-Objekten definierten Einrichtungen mittels der Methode „put_schedule“ gesandt. Diese empfangen das Schedule-Objekt und prüfen, ob der Schedule verarbeitet werden darf. Dabei wird das Zertifikat der Einrichtung geprüft. Abhängig von der Rolle, mit welcher der Schedule ausgeführt werden soll, wird außerdem die Signatur des Patienten überprüft. Ist das Zertifikat verifiziert, entspricht die Rolle des Schedules einer Rolle, welche die Einrichtung einnehmen darf und wurde die Patientensignatur verifiziert, so wird als Rückgabewert ein „OK“ zurückgeliefert. Im anderen Fall gibt die Funktion ein Error-Objekt zurück. Wurde der Schedule erfolgreich übertragen, so beginnt die Phase zwei, bei welcher nun die Ressourcen übertragen werden. Der Initiator sendet diese mit Hilfe der Methode „put_resource“ an alle anderen beteiligten Knoten. Dabei wird geprüft, ob die Ressourcen zur Serviceklasse des Schedule-Objektes passen. Wird ein Schedule als eine reine Dienstleistung implementiert, darf beispielsweise kein Objekt vom Typ Patient_OJ übertragen werden. Die dritte Phase besteht darin, die Service-Objekte, welche zum Schedule gehören, an die beteiligten Knoten zu übertragen. Dabei wird überprüft, ob die Ressourcen, die als Eingabewerte und Ausgabewerte innerhalb der Service-Objekte definiert sind, auf dem jeweiligen Knoten vorhanden sind. Weiter muss eine Einrichtung, die ein Service-Objekt erhält, prüfen, ob die Services, welche sie bearbeiten soll, auch in ihrer Datenbank definiert sind. Anderenfalls kommt es zum Fehler und ein Error-Objekt wird zurückgegeben.

Sind alle Services an die beteiligten Knoten übertragen, startet Phase vier. Diese wird durch den Initiator ausgelöst, der einen anderen Knoten mittels des Basisdienstes „activate“ aktiviert. Zu Beginn der Verarbeitung eines Schedules ist dieser immer auf dem Knoten, der als Initiator fungiert, aktiv. Während der Phase vier werden die Aufgaben der einzelnen Service-Objekte sequenziell abgearbeitet. Immer dann, wenn ein Knoten während der Verarbeitung zu einem Schritt gelangt, der in einer anderen Einrichtung verarbeitet werden muss, dann aktiviert er diese Einrichtung. Die Services können auf den einzelnen Knoten mit einem Modul verknüpft sein. Dazu existiert eine Datenbanktabelle mit dem Namen „stms“, die den Namen eines Service-Objektes der ID eines Moduls zuordnet. Diese Tabelle ist jedoch nur intern von Bedeutung und kann auch anders benannt werden. Existiert eine Zuordnung des aktuellen Arbeitsschrittes zu einem Modul, wird das Basisobjekt, welches der Verarbeitungsschritt als Eingabe verlangt, an das Modul übertragen und dieses liefert ein Ergebnis in Form eines anderen Basisobjektes zurück. Existiert kein Modul für die Verarbeitung eines Service-Objektes, so ist eine Interaktion mit einem Benutzer der Einrichtung nötig. In diesem Fall wird eine sogenannte Task erzeugt, die von einem Benutzer der Einrichtung abgearbeitet werden muss. Dies kann sich darauf beschränken, dass der Benutzer die Erledigung einer bestimmten Aufgabe quittiert. Verlangt der Arbeitsschritt als Rückgabewert eine Ressource, so kann der Benutzer manuell einen Datensatz auswählen. Dieser wird dann mit der Ressource verknüpft. Die Ressourcen werden je nach Wert des Attributes „presence“ an die anderen Knoten übertragen oder können von diesen bei Bedarf mittels der Methode „get_object“ abgerufen werden. Dadurch wird vermieden, dass Patientendaten an andere Einrichtungen geschickt werden, obwohl dies nicht in jedem Fall nötig ist. Dies ist insbesondere sinnvoll, wenn mehr als zwei Knoten an der Verarbeitung eines Schedules beteiligt sind. Die letzte Phase der Verarbeitung beginnt, wenn ein Knoten den letzten Arbeitsschritt abgearbeitet hat. Dann aktiviert dieser den Initiator. Der ruft auf allen beteiligten Knoten den Basisdienst

„exit“ auf. Diese löschen daraufhin alle den Schedule unter der betreffenden ID und alle Services und Ressourcen. Die Objekte, welche den Ressourcen zugeordnet sind, werden nur unter bestimmten Umständen gelöscht. Allerdings kann auf die Objekte dann mangels passender Ressourcen nicht mehr durch andere Einrichtungen zugegriffen werden, sobald der Schedule gelöscht ist.

Die fünf Phasen der Verarbeitung eines Schedule sind in Abbildung 20 für einen Schedule, der aus zwei Arbeitsschritten besteht und dazu zwei Ressourcen verwendet, dargestellt. Dabei sind die Ressourcen als Sechsecke dargestellt, Schedules, Module und Service-Objekte als Rechtecke mit abgerundeten Ecken, Datenobjekte als Kreis und Tasks als Rechteck dargestellt. Linien zwischen Objekten bedeuten die Zugehörigkeit eines Objektes zu einem anderen, wie die Verbindung zwischen dem Schedule und den dazugehörigen Ressourcen und Service-Objekten. Die Übertragung von Daten ist als Pfeil dargestellt. In Phase eins wird das Template des Schedules geklont und der dadurch entstehende Schedule an den Kommunikationspartner übertragen. Der Empfänger prüft die Signatur des Zertifikates der anderen Einrichtung mit Hilfe eines Moduls. Die Phasen zwei und drei dienen dazu, die beiden Ressourcen und Service-Objekte an den anderen Knoten zu übertragen. Die eigentliche Verarbeitung findet in Phase vier statt. In Abbildung 20 ist für die Phase vier die Reihenfolge der Kommunikation in Form von römischen Zahlen angegeben, die sich jeweils auf den Pfeil unterhalb der Ziffer beziehen. Zuerst aktiviert der Initiator den Schedule auf Knoten zwei. Dieser muss dann den Service SE1 abarbeiten. Dazu ruft er ein Modul auf, dass dem Service-Objekt SE1 zugeordnet ist. Dieses Modul liefert das Objekt OJ1 zurück, was nun der Ressource RS1 zugeordnet wird. Anschließend aktiviert Knoten zwei den Initiator, der mit Service SE2 die Bearbeitung fortsetzt. Da für SE2 als Parameter das Objekt der Ressource RE1 nötig ist, fragt der Initiator Knoten zwei nach dieser Ressource und erhält das dazu passende Objekt OJ1 zurück. Da dem Service-Objekt SE2 kein Modul zugeordnet ist, wird eine Task erzeugt, die von einem Benutzer manuell bearbeitet werden muss. Nach Bearbeitung dieser Task steht das Objekt OJ2, welches der Ressource RE2 zugeordnet ist, zur Verfügung. Da diese Ressource im Parameter „presence “ den Wert zwei besitzt, wird das Objekt, sobald es vorhanden ist, an alle beteiligten Knoten mittels der Methode „put_object“ gesendet. Phase fünf wird mit Erreichen des letzten Arbeitsschrittes des Schedules gestartet. Der Initiator ruft daraufhin den Dienst „exit“ auf allen beteiligten Knoten auf. Diese löschen nun alle Service-Objekte, Ressourcen und den Schedule. Die Datenobjekte verbleiben jedoch auf dem Knoten. Diese stellen das Ergebnis des Dienstes dar.

3.9.1 Schedules mit mehr als zwei Knoten

Viele Abläufe spielen sich zwischen zwei Knoten ab. Die Schedules, welche auf dem Initiator als Templates abgelegt sind, beziehen nur solche Einrichtungen ein, die dem Initiator eines Schedules bekannt sind. Die Mehrzahl der Abläufe findet ohnehin zwischen benachbarten Einrichtungen statt. Ein Arzt überweist seine Patienten in der Regel in ein nahegelegenes Krankenhaus. Wird ein Schedule zwischen mehreren Einrichtungen abgearbeitet, ist es möglich, dass sich beteiligte Knoten nicht kennen und ein Knoten keine Informationen über einen anderen Knoten besitzt, der in den Schedule involviert ist. In diesem Fall kann der Knoten die entsprechenden Daten in Form eines Institution-Objektes beim

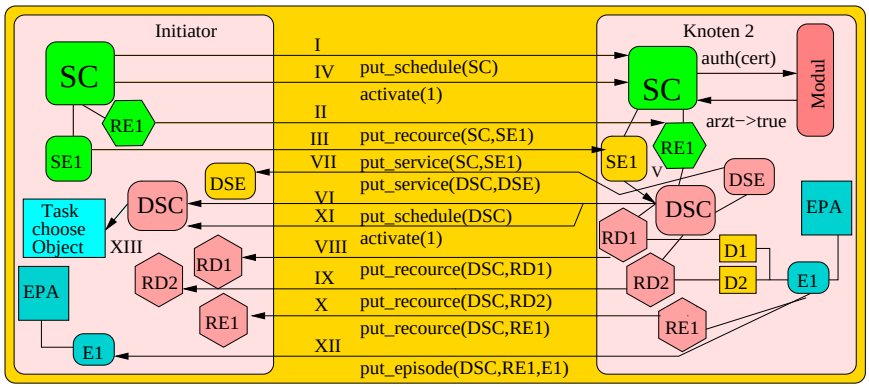

Abbildung 21: Dynamisch erzeugte Schedules

Initiator des Schedules mit Hilfe der Methode „get_institution“ anfordern. Der Initiator stellt diese Daten als Ressource unter dem Institutionskennzeichen der Einrichtung zur Verfügung, ohne dass diese explizit definiert werden müß. Empfängt ein Knoten ein Institution-Objekt vom Initiator eines Schedules, so kann er dieses mit Hilfe des Basisdienstes „auth“ überprüfen, indem er das Zertifikat der Institution an das entsprechende Modul übergibt. Ist diese Verifikation erfolgreich und der Knoten besitzt die Berechtigung, die für den Schedule benötigte Rolle einzunehmen, so kann diese Institution mit Hilfe der Schedule-ID auf die Ressourcen der anderen Einrichtung zugreifen. Der Initiator trägt dafür die Verantwortung. Er muss dafür sorgen, dass kein unbeteiligter Knoten Zugriff auf die Ressourcen des von ihm initiierten Schedules erhält.

3.9.2 Dynamische Schedules

Die bisher beschriebenen Fälle der Verarbeitung bezogen sich auf Schedules, die als Template statisch definiert wurden. Alle Ressourcen und Services waren vor Beginn der Abarbeitung bereits definiert. Es gibt jedoch auch Anwendungsfälle, in denen ein Schedule nicht statisch vordefiniert werden kann, sondern dessen Template nach dessen Klonierung verändert werden muss. Dies ist insbesondere der Fall, wenn im Vorfeld nicht klar ist, welche Ressourcen für die Verarbeitung des Schedules nötig sind. In einem solchen Fall ist es möglich, Schedules nach Bedarf dynamisch zu generieren. Insbesondere für die Übertragung von Datensätzen aus unbekannten Einrichtungen, die bei der Suche nach Patientendaten gefunden wurden, ist dies nötig. Weder die Einrichtung, in der sich die gesuchten Patientendaten befinden, noch die, welche die Daten anfordert, kann im Vorfeld entscheiden, welche Ressourcen benötigt werden. Für eine Übertragung von Patientendaten muss ein Schedule für jeden Patienten individuell dynamisch erstellt werden können.

Die Initiative geht jedoch von dem Knoten aus, der Daten empfangen möchte. Dieser kann jedoch auch nicht zum Zeitpunkt der Suche entscheiden, wie viele Ressourcen benötigt werden. Deshalb schickt dieser Knoten der Einrichtung, in welcher Patientendaten gefunden wurden, zuerst einen Schedule mit einer Ressource. Diese Ressource verweist auf ein Schedule-Objekt. Außerdem gehört zum Schedule ein Service-Objekt mit Namen „patakte“. Dieses Service-Objekt

muss auf allen Knoten definiert sein, von welchen Patientendaten angefordert werden sollen. Anschließend wird der Zielknoten aktiviert und beginnt den Arbeitsschritt „patakte“ abzuarbeiten. Dies tut der Knoten, indem er einen Schedule erstellt, der die Episodenobjekte und Datensätze des gesuchten Patienten enthält. Dabei besitzt das Attribut „presence“ für die Ressourcen der Episoden-Objekte den Wert zwei. Die Episoden werden also sofort verteilt. Die Datensätze des Patienten hingegen besitzen den Wert eins für das Attribut „presence“. Diese werden nur übertragen, wenn die Einrichtung, welche den Patienten behandelt, diese Objekte anfordert. Nun aktiviert der Knoten, auf dem sich die gesuchten Datensätze befinden, den Knoten, der die Suchanfrage gestellt hat. Der dynamisch erstellte Schedule besitzt ebenfalls ein Service-Objekt. Es trägt den Namen „choose_object“. Der Knoten erzeugt eine Task, die den Nutzer auffordert, eine Ressource auszuwählen, welche dann mit Hilfe der Methode „get_object“ übertragen werden kann. Der Knoten, welcher die Suchanfrage gestellt hat, muss nun die Task nicht löschen und den Arbeitsschritt nicht weiterschalten. Solange dies nicht geschehen ist, wird der Schedule nicht beendet und der Knoten kann noch über die Schedule-ID Daten des Patienten anfordern. Nach der Verfallszeit des Schedules wird der Basisdienst „exit“ aufgerufen und der Schedule samt allen Ressourcen gelöscht. Werden jetzt noch Patientendaten benötigt, so muss der Vorgang von neuem gestartet werden.

Es sind unterschiedliche Formen von dynamisch erzeugten Schedules nötig. Zum einen können, wie im Beispiel, Ressourcen dynamisch erstellt werden. Zum anderen ist es bei etlichen Anwendungsfällen nötig, Institutionen, auf denen Arbeitsschritte stattfinden sollen, dynamisch zu verändern. Dazu werden Attribute des Schedules oder der Services in eckigen Klammern angegeben. Wird bei der Abrechnung in einem Service der Wert [patient.inshurance] im Attribut „Institution“ hinterlegt, so wird dieser Wert zur Laufzeit durch die Krankenversicherung des Patienten ersetzt. Abbildung 21 zeigt den Vorgang der dynamischen Erzeugung eines Schedules schematisch. Die Symbolik entspricht der aus Abbildung 20. Der Initiator sendet ein Schedule-Objekt an den Knoten, der in seiner elektronischen Patientenakte Daten des gesuchten Patienten gespeichert hat. Dieser Knoten prüft den Schedule. Anschließend aktiviert der Initiator Knoten zwei. Wurde der Schedule erfolgreich validiert, erzeugt dieser dynamisch einen Schedule, zu welchem er für die Episoden- und Datenobjekte des Patienten eine Ressource anlegt. Dieser Schedule wird dann an den Initiator zurückgeschickt und aktiviert. Das Episodenobjekt E1 wird sofort an den Initiator mittels des Basisdienstes „put_episode“ übertragen. Jetzt kann der Initiator die Objekte anfordern, auf welche sich die Ressourcen beziehen.

3.9.3 Fehlerbehandlung

Das System soll unterschiedlichen Einrichtungen ermöglichen, ihre Abläufe zu verknüpfen. Damit dies gelingen kann, muss eine transparente Behandlung von Fehlern erfolgen. Administratoren unterschiedlicher Einrichtungen richten Dienste ein, die von anderen Einrichtungen aufgerufen werden. Dabei kann die Ausführung eines Schedules in jeder der fünf Phasen jeweils in jedem beteiligten Knoten scheitern. Durch die Validierung zu Beginn der Bearbeitung werden die einzelnen Arbeitsschritte nach bestimmten Regeln geprüft. Verbietet eine Regel die Ausführung eines Schrittes, beispielsweise weil in einem Schedule-Objekt Patientendaten übermittelt werden sollen, für welches das nicht erlaubt ist, so

muss der gesamte Ablaufplan verworfen werden. An dieser Stelle ist es nötig, dem Initiator eine genaue Beschreibung des Fehlers zu senden. Dazu wurde in Kapitel 3.5.4 das Error-Objekt eingeführt. Dieses Objekt wird mit der entsprechenden Schedule-ID an alle Knoten gesendet, wenn bei der Verarbeitung ein Fehler aufgetreten ist. Die Fehlermeldungen werden eine bestimmte Zeit in der Datenbank gespeichert, um eine Analyse des Fehlers durchführen zu können. Im System kann ein Retry hinterlegt werden. Dieser gibt an, nach welcher Wartezeit ein Schedule-Objekt erneut an die beteiligten Knoten gesendet werden soll und wie oft dies geschehen soll. Trat ein Fehler wegen eines temporären Problems der Netzwerkverbindung auf, so kann der Schedule beim zweiten oder dritten Versuch erfolgreich abgearbeitet werden.

4 Domänenspezifische Anwendungsdienste

Um Dienste so spezifisch wie nötig, aber gleichzeitig so allgemein wie möglich zu gestalten, ist es sinnvoll, die Dienste in Klassen zu unterteilen. Dabei haben die an dem Schedule beteiligten Einrichtungen und ihre spezifischen Aufgaben einen wesentlichen Einfluss auf die Zuordnung der Dienste zu einer dieser Klassen. Je nach Klasse eines Dienstes unterscheiden sich die Anforderungen an den Datenschutz teils erheblich. Nicht in jeder Klasse müssen alle Arten von Datenobjekten übertragen werden. So kann schon aufgrund der Klasse eines Dienstes bereits eine Sicherheitsüberprüfung stattfinden. Gleichzeitig werden nur die Sicherheitsmaßnahmen für die Kommunikation zwischen Knoten gefordert, die für die jeweilige Klasse auch wirklich nötig sind. Innerhalb der Klassen können nun von den Knoten eigene Abläufe in Form von Schedules definiert werden. Diese werden individuell auf die Bedürfnisse der einzelnen Einrichtungen und deren Kommunikationspartner abgestimmt. Dennoch ist es notwendig, bestimmte Abläufe für alle Knoten vorzudefinieren. Dabei handelt es sich insbesondere um die in Kapitel 3.9.2 eingeführten dynamischen Schedules, die für verschiedene Knoten vordefiniert werden. Wird beispielsweise eine Suche durchgeführt, so ist nicht bekannt, auf welchem Knoten Daten zu dem betreffenden Patienten gefunden werden. Diese Suche nach Daten hat zur Folge, dass mit hoher Wahrscheinlichkeit eine Kommunikation mit einem bis dahin unbekannten Knoten erforderlich wird. Um dies zu ermöglichen, müssen bestimmte Abläufe jeder Klasse von allen Teilnehmern verarbeitet werden können. Jede Klasse beinhaltet deshalb eine Menge von Diensten, welche von allen für diese Klasse relevanten Knoten bedient werden müssen und solche, die individuell von einzelnen Knoten definiert werden können. In Abbildung 22 sind die vier Klassen von Diensten dargestellt. Das Sechseck in der Mitte der Abbildung stellt die Menge von Diensten dar, die von allen Knoten angeboten werden müssen. Dabei muss es sich um dynamische Schedules handeln. Die äußeren Bereiche der Rechtecke sind Dienste, die frei definiert werden können. Die zugehörigen Schedules können als statische Templates definiert werden. In den folgenden Abschnitten werden vier Klassen von Diensten definiert und jeweils Beispiele für Dienste dieser Klassen beschrieben.

4.1 A - Abrechnung

Die erste Klasse in dieser Reihe ist die Klasse der Dienste, welche für die Abrechnung notwendig sind. Sie ermöglichen eine Kommunikation zwischen den Erbringern medizinischer Dienstleistungen (Krankenhäuser, Arztpraxen), den Kostenträgern, dem Medizinischen Dienst der Krankenkassen (MDK) und gegebenenfalls den Kreditinstituten, bei denen sich die entsprechenden Konten befinden. Im Gesundheitswesen ist die elektronische Abrechnung bereits üblich. Hier findet das Objekt „Edifact“ Verwendung, welches Nachrichten im Format UN/Edifact beinhaltet. Bei den Informationen, welche dabei ausgetauscht werden, handelt es sich um patientenbezogene Daten. Deshalb müssen sich die Kommunikationspartner authentifizieren und die Nachrichten müssen verschlüsselt übertragen werden. Außerdem muss der Knoten, der eine Kommunikation auslöst, eine Signatur des Patienten vorweisen um zu zeigen, dass er mit der Behandlung des Patienten beauftragt ist. Diese Informationen werden bei der Übertragung des Schedule-Objektes ausgetauscht.

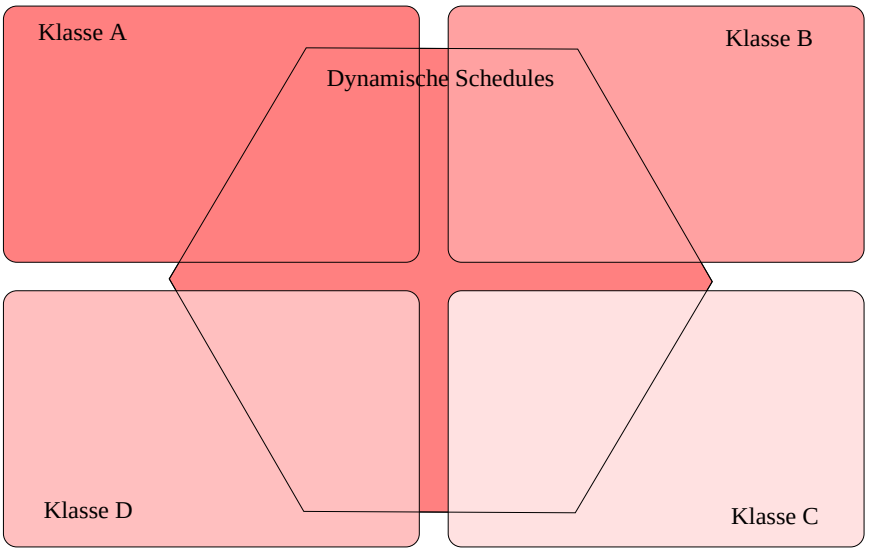

Abbildung 22: Klassen von Diensten

4.1.1 A_PAYROLL

Der Schedule A_PAYROLL bildet die Abrechnung mit der Krankenkasse ab. Dabei bezieht sich ein solcher Prozess immer auf einen Patienten. Als ersten Schritt muss vom Leistungserbringer geprüft werden können, ob ein Patient beim betreffenden Kostenträger versichert ist oder nicht. Dazu sendet der Leistungserbringer ein Schedule-Objekt an den Kostenträger, der mit einem Zertifikat des Patienten versehen ist. Sendet dieser ein Error-Objekt, so ist der Patient nicht über diesen Kostenträger versichert. Wird ein OK gesendet, so besteht ein Versicherungsverhältnis und der Schedule kann verarbeitet werden. Damit ein Fall überhaupt abgerechnet werden kann, muss die Aufnahme des Patienten in einer stationären Einrichtung dem Kostenträger innerhalb einer bestimmten Frist gemeldet werden. Bereits bei der Übertragung des Schedules wurde dessen ID mit dem Datensatz des Patienten beim Kostenträger verknüpft. Nun sendet der Leistungserbringer ein Objekt vom Typ „date_oj“, welches den Zeitpunkt und die Zeit der Aufnahme im Krankenhaus angibt. Dann wird der Patient behandelt. Ist die Behandlung beim Leistungserbringer abgeschlossen, so muss wiederum die Entlassung gemeldet werden. Der Leistungserbringer muss nun wieder innerhalb einer bestimmten Frist die Entlassung des Patienten melden. Das geschieht ebenfalls mit Hilfe des Objektes vom Typ „date_oj“. Im Anschluss sendet der Leistungserbringer die Rechnung in Form eines Edifact-Objektes und ein Episoden-Objekt. Letzteres beinhaltet Verknüpfungen zu allen verfügbaren Daten des Patienten, indem dafür jeweils eine Ressource an den Kostenträger gesendet wird. Diese Ressourcen werden dynamisch erstellt, wie im Kapitel 3.9.2 beschrieben wurde. Werden vom Kostenträger zur Prüfung des Falles medizinische Daten des Patienten benötigt, so kann dieser die Daten direkt beim Leistungserbringer über den Basisdienst „get_object“ abrufen. Außerdem existiert eine Ressource, die mit den Stammdaten des Patienten verknüpft ist. Dies kann der Kostenträger abrufen, um die Daten des Patienten zu aktualisieren. Am En-

ID	Name	Klasse	Len	Rolle	
2	A_PAYROLL	A	4	management	
ID	**Name**	**Objekt**	**Lifetime**	**Verteilung**	**Typ**
6	Aufnahmedatum	0	144000	2	date
7	Entlassdatum	0	144000	2	date
8	Rechnung	0	144000	2	edifact
10	Kostenübernahme	0	144000	2	edifact
11	Befunde	*	144000	1	findings
12	Untersuchungsdaten	*	144000	1	data
23	Episode	0	144000	1	episode
24	Stammdaten	0	144000	1	patient
Name	**Input**	**Output**	**Spr.**	**Bed.**	**Institution**
Aufnahme	Aufnahmedatum		0	False	Diakomed
Entlassung	Entlassdatum		0	False	Diakomed
Rechnung		Rechnung	0	False	Diakomed
Kostenübernahme		Kostenübernahme	0	False	Techniker KK

Abbildung 23: Konfiguration des Dienstes A_PAYROLL

de der Verarbeitung sendet der Kostenträger eine Kostenübernahmeerklärung an den Leistungserbringer. Dies geschieht mittels eines Edifact-Objektes. Ein Dienst der Klasse A bezieht sich immer auf einen speziellen Patienten. Dadurch kann jeder Abrechnungsfall einzeln betrachtet werden und es läßt sich leicht ein Überblick über noch nicht erfolgreich abgerechnete Fälle verschaffen. Abbildung 23 zeigt beispielhaft die Konfiguration des Dienstes A_PAYROLL. In der obersten Spalte der Tabelle ist der Name und einige Informationen zum Schedule dargestellt. Die nächsten Spalten beschreiben die Ressourcen, die für seine Abarbeitung nötig sind. Die Ressourcen „Befunde“ und „Untersuchungsdaten“ müssen dynamisch für jeden Befund und jedes Untersuchungsergebnis angelegt werden, damit die Krankenkasse diese bei Bedarf abfordern kann. Die letzten Spalten zeigen die Arbeitsschritte, welche durchlaufen werden müssen. Diese Struktur ist in der Datenbank abgelegt und kann über ein Wizard erstellt werden.

4.2 B - Behandlungsprozess

Der wichtigste und entscheidendste Prozess im Gesundheitswesen ist der Behandlungsprozess. Wird eine Person ernsthaft krank und damit zum Patienten, so sind meist mehrere Stationen bis zu seiner Heilung zu durchlaufen. Ein Hausarzt stellt die Diagnose und weist ins Krankenhaus ein. Dort werden Untersuchungen und Behandlungen durchgeführt. Bei schweren Fällen ist oftmals die Verlegung in eine Spezialklinik nötig. Am Ende einer Behandlung steht die Anschlussheilbehandlung. Die Dienste der Behandlungsklasse müssen diesen Weg des Patienten vorbereiten und wichtige diagnostische und organisatorische Daten auf die nächste Station übertragen. Sie sollen dafür sorgen, dass Informa-

ID	Name	Klasse	Len	Rolle	
3	B_REMITTANCE	B	4	doc	
ID	**Name**	**Objekt**	**Lifetime**	**Verteilung**	**Typ**
13	Aufnahmetermin	0	14400	2	date
14	Auftrag	0	14400	2	order
15	Meddaten	0	14400	1	basics
16	Aufnahmediagnose	0	14400	1	diagnosis
17	Stammdaten	0	14400	1	patient
Name	**Input**	**Output**	**Spr.**	**Bed.**	**Institution**
Terminvergabe	Auftrag	Aufnahmetermin	0	False	Chemnitz
Entlassung	Aufnahmetermin		0	False	Diakomed
Aufnahme	Stammdaten		0	False	Chemnitz
Ankunft			0	False	Chemnitz

Abbildung 24: Konfiguration des Dienstes B_REMITTANCE

tionen und Daten den Patienten auf seinem Weg begleiten und überall dort, wo sich der Patient gerade aufhält, eine optimale Behandlung gewährleistet werden kann. Der Aufruf eines B-Dienstes wird immer von der Einrichtung initiiert, in welcher sich der Patient gerade befindet. Das ist wichtig, da für die Übermittlung von Patientendaten die Einwilligung des Patienten erforderlich ist. Diese Einwilligung kann zum Beispiel mit Hilfe einer Smart-Card technisch realisiert werden. Beim Aufruf eines Dienstes der B-Klasse ist eine Identifikation des Patienten anhand eines MPI Voraussetzung. Es ist nicht sinnvoll, alle jemals aufgezeichneten Daten eines Patienten zu übermitteln. So sollten alle relevanten Daten für den jeweiligen Fall bei einer Verlegung an die Einrichtung gesandt werden, in welche der Patient verlegt werden soll. Welche Daten für eine Weiterbehandlung wichtig sind, muss vom Personal individuell entschieden werden. Im Gegensatz zu den Schedules der Dienstklasse C werden bei der Verarbeitung von Abläufen der Klasse B ausschließlich die Daten ein und desselben Patienten übertragen.

4.2.1 B_REMITTANCE

Der Dienst B_REMITTANCE dient zur Überweisung eines Patienten in eine andere Einrichtung, beispielsweise nach dem Besuch beim Hausarzt in eine Klinik. Dabei muss sich der Aufrufende authentifizieren. In diesem Fall ist nicht notwendig, dass der Knoten, welcher den Patienten überweisen möchte, beweist, dass der Patient in seiner Einrichtung in Behandlung ist, da vom Zielknoten keine Patientendaten übertragen werden. Damit der Patient verlegt werden kann, muss die Zieleinrichtung einen Termin vergeben, zu welchem der Patient aufgenommen werden kann. Dieser Termin wird von einem Arbeitsschritt „Terminvergabe“ erzeugt, der als Eingabe einen Auftrag in Form des Basisobjektes vom Typ „offer“ benötigt. Der Termin wird wiederum als Eingabewert für den Arbeitsschritt „Entlassung“ benötigt. Ist dieser Arbeitsschritt erledigt, kann der Zielknoten aktiviert werden und den Patienten aufnehmen. Als Eingabe dafür

dienen die Stammdaten des Patienten. Weitere Ressourcen für die medizinischen Grunddaten des Patienten und die Aufnahmediagnose werden ebenfalls angelegt und an den Zielknoten übertragen, sobald dieser die Daten anfordert. Nun kann der Patient mittels eines Krankentransportes verlegt werden. Sobald der Patient dort angekommen ist, wird der Arbeitsschritt „Ankunft“ abgearbeitet. Der Status des Schedules wird weitergeschaltet, und im letzten Schritt wird der Schedule auf beiden Knoten gelöscht. In Abbildung 24 ist die Konfiguration des Dienstes B_REMITTANCE als Screenshot dargestellt.

4.2.2 B_SEND_PRESCRIPTION

In Deutschland sind viele Medikamente nicht frei verkäuflich, sondern bedürfen einer Verschreibung durch den Arzt. Dieser Vorgang wird durch das Rezept abgebildet, welches vom Arzt ausgestellt und unterschrieben wird. [Gem10a] Eine der ersten Anwendungen im Rahmen der EGK und deren Infrastruktur ist die Einführung eines elektronischen Rezeptes. Um dieses elektronische Rezept abzubilden, werden von der Gematik zwei Konzepte diskutiert. Eine Möglichkeit ist, die Rezeptdaten auf den geplanten Servern der EGK-Infrastruktur abzulegen [Gem10b] und den Apotheken Zugriff darauf zu gewähren. Die zweite Möglichkeit ist die Speicherung auf der Chipkarte. In der in dieser Arbeit beschriebenen Infrastruktur werden die Rezeptdaten nicht auf einer Chipkarte gespeichert und mangels zentraler Datenserver auch dort nicht abgelegt. Die Daten sollen bei Bedarf sicher an den Knoten übertragen werden, an welchem diese benötigt werden.

Diese Aufgabe übernimmt der Dienst B_PRESCRIPTION. Er ermöglicht, dass ein Rezept für den Patienten erstellt und hinterlegt werden kann. Wenn der Arzt dem Patienten ein Rezept ausstellt, dann darf er dieses selbstverständlich nicht an eine Apotheke schicken. Der Patient kann selbst entscheiden, in welche Apotheke er geht. Da das Ausstellen eines Rezeptes ein Vorgang ist, bei dem ein Arzt, die Krankenkasse und die Apotheke involviert sind, muss zwischen diesen Instanzen eine Kommunikation stattfinden. Jedoch ist zum Zeitpunkt der Verschreibung noch nicht klar, welche Apotheke in den Vorgang involviert sein wird. Welche Krankenkasse die Zahlung an die Apotheke leisten muss, ist jedoch bei der Verschreibung bereits definiert. Deshalb werden die Rezepte zur entsprechenden Krankenkasse geschickt. Zum Zeitpunkt der Verschreibung lässt sich dem Patienten genau eine Krankenkasse zuordnen. Das Rezept kann dann von der Apotheke angefordert werden.

Abbildung 25 zeigt die Abfolge der Kommunikation. Während auf herkömmliche Art und Weise das Rezept durch den Patienten vom Arzt zur Apotheke gelangt und diese anschließend mit der Krankenkasse des Patienten abrechnet, ist es innerhalb einer übergreifenden Infrastruktur sinnvoller, das Rezept im System der Krankenkasse zu hinterlegen. Die Apotheke kann dieses Rezept dann von dort abrufen. Welche Apotheke vom Patient aufgesucht wird, ist zu Beginn des Vorgangs jedoch nicht klar. Deshalb muss das Senden des Rezeptes und das Abrufen desselben in zwei unterschiedlichen Schedules definiert werden. Damit hat der Patient die Möglichkeit, sich die Apotheke auszusuchen. Es enthält die elektronische Signatur des Arztes und ist somit gültig. Das Rezept wird in Form eines Prescription-Objektes übertragen. Darin sind alle relevanten Informationen für die Apotheke enthalten. Dieses Objekt kann die Apotheke von der Krankenkasse abrufen. Da die Prescription-Objekte bei der Kranken-

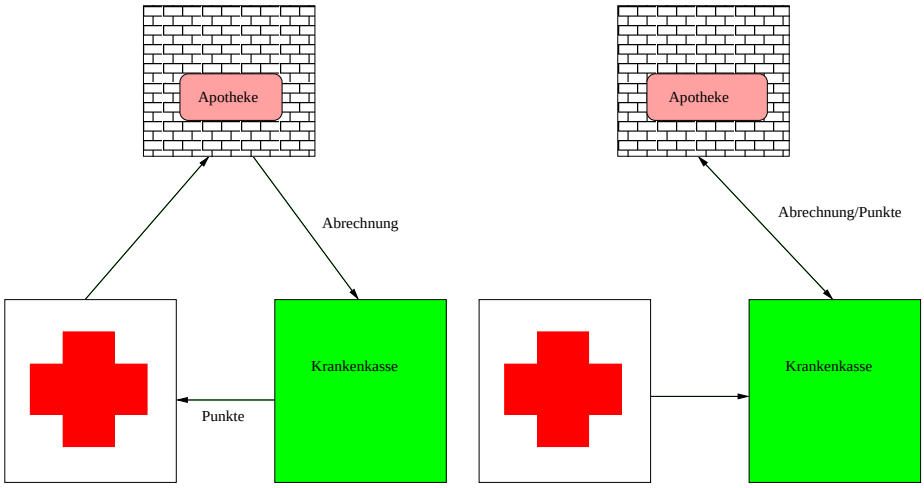

Abbildung 25: Kommunikation aktuell und innerhalb der Dienstarchitektur

kasse hinterlegt werden, ist diese auch in der Lage, schädliche Wechselwirkungen zwischen verschiedenen Medikamenten zu entdecken.

Da durch solche Wechselwirkungen der Krankenkasse auch erhebliche Kosten entstehen können, hat diese ein großes Interesse, Wechselwirkungen und Medikamentenunverträglichkeiten zu erkennen. Ein wichtiges Prinzip bei der Einführung von IT-Systemen sollte immer sein, dass die Verantwortung für die Einführung einer Funktionalität bei dem liegt, der auch eine Motivation dazu hat. Beim Erkennen von Wechselwirkungen ist eine solche Motivation der Krankenkasse gegeben. Deshalb sollte diese Funktionalität auch dort angesiedelt werden. Im Fall einer Wechselwirkung sendet der Kostenträger ein Error-Objekt an die erstellende Einrichtung und bricht den Prozess ab. Diese kann dann anhand der verbalen Beschreibung des Error-Objektes erkennen, dass eine Wechselwirkung zwischen einem bereits verordneten Medikament besteht. Eine Unverträglichkeit wird deshalb schon bei der Verschreibung entdeckt und kann somit korrigiert werden.

4.2.3 B_GET_PRESCRIPTION

Der Schedule B_GET_PRESCRIPTION findet zwischen einer Apotheke und einem Kostenträger statt. Beim Kostenträger werden Rezepte hinterlegt, die von der Apotheke abgerufen werden können. Kommt ein Patient in die Apotheke, so kann die Apotheke mit Hilfe der EGK des Patienten den zuständigen Kostenträger ermitteln. Wie bei allen anderen Kommunikationsvorgängen wird mit Hilfe dieser Chipkarte eine Signatur erzeugt, mit dessen Hilfe die Apotheke nachweisen kann, dass der Patient dort Medikamente abholen möchte. Diese Signatur und das signierte Zertifikat der Apotheke autorisieren den Knoten, das Rezept in der Rolle „Apotheke“ abzurufen. Ist das Schedule-Objekt zur Krankenkasse übertragen, so sendet diese das Objekt vom Typ „prescription“ an die Apotheke. Die Apotheke sendet ihrerseits die entsprechende Rechnung in Form eines Edifact-Objektes an die Krankenkasse. Diese veranlasst dann die Bezahlung der Medikamente. Das kann durch ein Money-Objekt geschehen, welches die Kran-

kenkasse beim Gesundheitsfond einlöst. Im Gegensatz zu anderen Prozessen im Gesundheitswesen ist bei der Verarbeitung dieses Schedules die Delegation des Zertifikates der Chipkarte des Patienten nicht nötig. Solange sich die Chipkarte im Kartenleser befindet, kann eine Verbindung mit dem Kostenträger aufgebaut werden. Sobald der Patient die Apotheke verlässt, ist dies nicht mehr möglich.

4.2.4 B_GET_DATA

Der Hauptnutzen einer Vernetzung verschiedener Knoten im Gesundheitswesen ist die Vermeidung von Doppeluntersuchungen durch die Verfügbarkeit von Daten, die in anderen Einrichtungen gespeichert sind. Bei einer Suche soll ausschließlich von dazu berechtigten Knoten ermittelt werden können, in welchen Einrichtungen Daten des betreffenden Patienten vorhanden sind. Die eigentlichen Daten müssen später mittels eines Schedules übertragen werden. Oft sind jedoch nicht alle auf dem gefundenen Knoten vorhandenen Datensätze für die gegenwärtige Behandlung von Bedeutung. Es muss zuvor eine Auswahl getroffen werden. In Kapitel 3.9.2 wurde die Übertragung von Patientendaten bereits als Beispiel für dynamisch erstellte Schedules aufgegriffen. An dieser Stelle wird der Dienst zur Übertragung von Patientendaten noch einmal detailliert beschrieben. Der Knoten, welcher eine Anfrage an eine Einrichtung sendet, erzeugt die Kopie des Templates B_GET_DATA. Nun muss die Institution in den Service-Objekten dynamisch durch die Institution ersetzt werden, an welche die Anfrage gerichtet werden soll. Der Schedule wird dann zusammen mit einem Service namens „patakte“ und einer Ressource vom Typ „Schedule“ an den Kommunikationspartner geschickt. Dieser prüft das Zertifikat der Einrichtung und die Signatur des Patienten. Die Rolle des Schedules muss den Wert „Arzt“ besitzen, da sonst nicht auf die Daten des Patienten zugegriffen werden darf. Wurde der Schedule erfolgreich verifiziert, wird dieser aktiviert und führt den Dienst „patakte“ aus, indem er einen weiteren Schedule erstellt. Dieser erbt die Rolle des übergeordneten Schedules. Zu dem dynamisch erstellten Schedule gehört ein Service-Objekt und für jeden Aufenthalt und jeden Datensatz eine Ressource. Das Template dieses Dienstes besteht aus Ressourcen jeden Typs, deren Attribut „object“ den Wert „*“ besitzt. Dieser Wert bedeutet, dass für jedes Objekt vom Typ der Ressource, welches dem Patienten zugeordnet ist, eine Ressource erzeugt werden muss. Dabei werden die Episoden des Patienten, auf welche die Einrichtung zugreifen darf, einer Ressource zugeordnet, deren Wert für das Attribut „presence“ gleich zwei ist. Deshalb werden die Episoden sofort an den Kommunikationspartner gesendet. Weiter werden die Ressourcen der unterschiedlichen Patientendatensätze ebenfalls an den anderen Knoten gesandt. Nun wird der Initiator aktiviert und erzeugt eine Task zum Service „choose_object“. Solange diese Task aktiv ist, kann ein Nutzer des Knotens eine Ressource auswählen, deren Objekt dann mit Hilfe des Basisdienstes „get_object“ angefordert werden kann.

4.3 C - Controlling

In den neunziger Jahren kam das Bild des Arztes als Unternehmer auf. Dennoch ist eine Praxis oder ein Krankenhaus kein gewöhnliches Unternehmen auf einem freien Markt. Der Handlungsspielraum eines Dienstleisters im Gesundheitswesen wird durch den Gesetzgeber, die ärztliche Selbstverwaltung und die

ID	Name	Klasse	Len	Rolle
4	C_QUALI	C	2	management

ID	Name	Objekt	Lifetime	Verteilung	Typ
18	Bögen	0	14400	2	text
19	Protokoll	0	14400	2	text

Name	Input	Output	Spr.	Bed.	Institution
Auswertung	Bögen	Protokoll	0	False	QUANT
Import	Protokoll		0	False	Diakomed

Abbildung 26: Konfiguration des Dienstes C_QUALI

Krankenkassen bestimmt. Beispielsweise versucht der Gesetzgeber, durch die ständige Anpassung der Fallpauschalen, auf die aktuellen Gegebenheiten, Einfluss auf Entwicklungen im Gesundheitswesen zu nehmen. Dafür wurde eigens ein Institut namens DIMDI gegründet, welches den Katalog der DRGs und deren Bewertung anpasst und veröffentlicht. DIMDI ist eine Abkürzung und steht für „Deutsches Institut für medizinische Dokumentation und Information“. Um diese Aufgabe zu erfüllen, benötigt das DIMDI eine Rückkopplung mit den Einrichtungen. Welche DRGs kommen wie oft vor und welche Komplikationen sind wahrscheinlich? Solche und ähnliche Fragen müssen beantwortet werden. Andere Aufgaben der Kontrolle obliegen der ärztlichen Selbstverwaltung. Die Krankenhausgesellschaft ist beispielsweise für das Qualitätsmanagement zuständig und hat für die Auswertung dieser Daten Unternehmen beauftragt. Die entsprechenden Daten müssen von den Krankenhäusern an jenes Unternehmen gesandt werden. Dort werden die Daten ausgewertet und die Ergebnisse zur Krankenhausgesellschaft geschickt. Dieser Kommunikationsprozess könnte durch Dienste vereinfacht und automatisiert werden. Bei dieser Art der Dienste geht es um anonymisierte, pseudonymisierte und teilweise quantifizierte Daten. Es ist für die Qualitätssicherung nicht entscheidend, bei welchem Patienten eine Infektion auftrat. Dort ist nur der Zusammenhang zwischen der Häufigkeit einer Infektion und einer bestimmten Diagnose wichtig. Bei verschiedenen Daten des Qualitätsmanagements ist es aber auch nötig, eine Zuordnung zu konkreten Patienten und Fällen herstellen zu können. Das ist notwendig, falls Rückfragen zu bestimmten Positionen gestellt werden. Um dies zu gewährleisten, werden die Daten pseudonymisiert.

4.3.1 C_QUALI

Der Prozess C_QUALI kann von Krankenhäusern benutzt werden, um ihre Qualitätssicherungsdaten an die BQS, die Bundesgeschäftsstelle Qualitätssicherung GmbH, zu übertragen. Nachdem sich die Kommunikationspartner authentifiziert haben, wird das Objekt „Text“ dazu verwendet, die Daten zu übertragen. Dadurch können die derzeit verwendeten Datenformate vorerst weiter benutzt

werden. Der Schedule ist anschließend solange aktiv, bis eine Antwort ebenfalls in Form eines Text-Objektes zurückgesandt wurde. Dieses kann dann in das Qualitätsmanagementsystem eingespielt werden. Abbildung 26 zeigt die Konfiguration des Dienstes C_QUALI mit zwei Arbeitsschritten. Im ersten Schritt wird von einem Softwaremodul des Empfängers die Auswertung der Qualitätssicherungsbögen durchgeführt. Die Protokolldatei wird dann vom Initiator des Schedules importiert.

4.3.2 C_STATISTIK

In Kapitel 3.4.11 wurde das Job-Objekt beschrieben. Der Prozess C_STATISTIK dient dazu, ein solches Objekt an einen Knoten zu übermitteln. In bestimmten Fällen ist es notwendig, dass ein Wert zu einem bestimmten Zeitpunkt gesucht wird. Ist beispielsweise die Belegung am 1. Januar eines Jahres von Interesse, so wird innerhalb des C_STATISTIK-Prozesses ein Objekt vom Typ „Date“ übermittelt. Dieses gibt an, zu welchem Zeitpunkt die Information ermittelt werden soll. Doch bevor diese Objekte übertragen werden, müssen sich die Kommunikationspartner authentifizieren. Das geschieht durch das Schedule-Objekt, das ein signiertes Zertifikat des Initiators enthält. Im Anschluss wird ein Date-Objekt übertragen und anschließend das Objekt vom Typ „Job“. Der Knoten, der die Anfrage bekommen hat, ermittelt nun den Wert anhand der Definition in Form von HSML 3.3 und sendet den Wert an den anfragenden Knoten in Form eines Text-Objektes zurück.

4.4 D - Dienstleistung

Die vierte Klasse beinhaltet Aufgaben, für die ein externer Dienstleister in Anspruch genommen wird. Das kann beispielsweise ein anderes Krankenhaus sein, das die Diagnose eines Röntgenbildes erstellt, ohne dass der Patient in dieses Haus verlegt wird. Aber auch ein Labor, welches das Blut eines Patienten untersucht. Dazu ist es jedoch nicht nötig, dass der Name des Patienten an den Dienstleister übermittelt wird. Im Rahmen eines Schedule-Objektes wird ein Auftrag an einen Dienstleister geschickt. Ist dazu die Anwesenheit des Patienten nötig, so wird dieser zum Dienstleister transportiert. Der Auftrag wird dem Patienten in der ausführenden Einrichtung durch einen Barcode zugeordnet. Über die sched_id wird vom Auftraggeber das Ergebnis dem Patienten zugeordnet. Der Dienstleister muss keine Kenntnis der Stammdaten des Patienten bekommen. Zum Beispiel für die Untersuchung am MRT muss der Patient in die entsprechende Einrichtung gebracht werden. Er kann natürlich nicht vollkommen anonym untersucht werden, denn er wird dort gesehen und spricht mit dem Personal. Dennoch müssen keine elektronischen Spuren des Patienten beim Dienstleister verbleiben. Der Behandlungsvertrag besteht in diesem Fall mit der Klinik, in welcher der Patient liegt, und nicht mit der Einrichtung, welche die Dienstleistung erbringt. In Schedules der Klasse D dürfen deshalb keine Stammdaten von Patienten übermittelt werden. Die Aufträge werden anonym übertragen und die Ergebnisse vom Auftraggeber dem richtigen Patienten mittels der ID des Schedules zugeordnet. Dienstleistungen werden meist von Einrichtungen in Anspruch genommen, die sich in geographischer Nähe befinden und mit welchen man in der Vergangenheit gute Erfahrungen gesammelt hat. Aus diesen Grund können die meisten Dienste der Klasse D in Form von statischen Schedu-

ID	Name	Klasse	Len	Rolle	
5	D_WRITING	D	2	doc	
ID	**Name**	**Objekt**	**Lifetime**	**Verteilung**	**Typ**
20	Diktat	0	14400	1	data
21	Text	0	14400	2	text
22	Bezahlung	0	14400	2	money
Name	**Input**	**Output**	**Spr.**	**Bed.**	**Institution**
Freigabe	Text	Bezahlung	0	False	Diakomed
Schreiben	Diktat	Text	0	False	Schreibbüro

Abbildung 27: Konfiguration des Dienstes D_WRITING

les abgebildet werden. Die Dienstleistungen werden von jeder Einrichtung selbst definiert und können die unterschiedlichsten Bereiche betreffen. Die in diesem Kapitel beschriebenen Schedules dienen demnach lediglich als Beispiel.

4.4.1 D_WRITING

Der Schedule D_WRITING dient dazu, Schreibarbeiten auf komfortable Weise nach außen zu vergeben. Bei der Übertragung des Schedules ist nicht von Bedeutung, ob sich ein Patient beim Auftraggeber in Behandlung befindet. Mittels eines „data_oj“ Objektes wird eine DDS-Datei versandt. Selbstverständlich darf die Aufnahme nicht den Namen und das Geburtsdatum des Patienten enthalten. Eine externe Schreibkraft kann den Brief dann eintippen oder eine Spracherkennungssoftware transkripieren. Das fertige Dokument wird dann als Antwort unter der ID des Schedules zurückübertragen. Dazu wird das Objekt „text_oj“ verwendet. Moderne Arztbriefschreibsysteme fügen Daten wie den Namen und die Adresse des Patienten direkt aus der Datenbank des Patientenverwaltungssystems in den Brief ein. Diese Informationen befinden sich im Brief immer an der gleichen Stelle und müssen nicht diktiert werden. Abhängig von der Länge des Dokumentes kann dann mittels Computational Economy abgerechnet werden, indem ein Objekt vom Typ „money“ an das Schreibbüro übermittelt wird. Es ist durchaus denkbar, dass Einrichtungen zusammenarbeiten, indem sie sich gegenseitig unterstützen. Wenn in einem Krankenhaus ein Engpass entsteht, dann werden die Briefe in einer anderen Einrichtung geschrieben, welche wiederum Dokumente zum Schreiben schickt, falls dort Bedarf entsteht. Der Brief kann dann in die elektronische Krankenakte eingefügt werden. Wie bei jedem Brief muss der Arzt diesen dann noch freigeben. Erst dann hat der Brief seine Gültigkeit. Abbildung 27 zeigt beispielhaft die Konfiguration des Schedules D_WRITING. Er besteht aus zwei Arbeitsschritten. Zuerst wird der Brief von der Schreibkraft geschrieben, wozu das Schreibbüro die DDS-Datei anfordern muss. Als Ergebnis wird ein Text-Objekt erzeugt, welches an den Auftraggeber gesandt wird. Wurde der Brief freigegeben, erfolgt die Bezahlung mit Hilfe eines

Money-Objektes.

4.4.2 D_FINDING

Bei bestimmten Untersuchungen wie der Mammographie ist eine Zweitbefundung vorgeschrieben. Aber auch bei anderen radiologischen Untersuchungen erhöht eine Zweitbefundung die Qualität des Befundes und bringt so einen Gewinn für den Patienten. Der Dienst D_FINDING sendet mittels eines anonymisierten „data_oj“ Objektes ein Röntgenbild an eine andere Einrichtung. In dieser kann sich der Arzt die Untersuchung ansehen, hat aber keinen Zugriff auf Stammdaten des Patienten. Er befundet das Bild unvoreingenommen und verfasst einen Befund. Dieser kann in Schriftform oder im DDS-Format erstellt werden. Anschließend wird der Befund passend zur ID an den Auftraggeber zurückgesandt. Dazu eine Signatur des Befunders, welches diesen als Arzt ausweist. Eine Abrechnung erfolgt mittels der Dienste der Computational Economy.

4.4.3 D_REND

Bei Untersuchungen mit Hilfe eines Computertomographen oder eines MRT ist eine Anzahl von 1000 Schnitten keine Seltenheit. Mittels intelligenter Software werden aus diesen Schnitten dreidimensionale Darstellungen erzeugt. Über die Dichte werden die einzelnen Volumenelemente den Organen des menschlichen Körpers zugeordnet. Daraus können dann 3D-Darstellungen einzelner Teile des Menschen, z.B. des Skelettes, erzeugt werden. Dies erfordert enorme Rechenleistung und den Erwerb entsprechender Softwaremodule. Ist eine Klinik nicht in der Lage, mit Hilfe ihrer Anwendungen ein CT in eine 3D-Darstellung umzuwandeln, so kann diese die Rohdaten mittels eines „data_oj“ Objektes einem Knoten senden, der die Erstellung von 3D-Darstellungen als Dienstleistung anbietet. Dieser rendert die gewünschten Abbildungen oder Videos und sendet diese wiederum als „data_oj“ Objekte zurück. So ist es möglich, solche Berechnungen dreidimensionaler Objekte auszulagern und so Rechenleistung und Softwarelizenzen anderer Einrichtungen zu nutzen. Erwähnenswert ist an dieser Stelle, dass dieser Dienst auch mit Techniken der Computer Aided Diagnosis, die mit CAD abgekürzt wird, verwendet werden kann. Eine Software kann das Röntgenbild beispielsweise nach Herden untersuchen. Mittels neuronaler Netze können damit bereits erstaunlich gute Ergebnisse im Bereiche einer Vorbefundung erreicht werden. Eine solche Software kann beispielsweise kritische Bereiche markieren, wodurch der Arzt auf diese Bereiche aufmerksam gemacht wird. Da eine solche Software keinen Befund erstellen darf, fällt ein solcher Dienst in den Bereich der medizinischen Bildverarbeitung und könnte mit dem Prozess „D_REND“ realisiert werden. Ein Röntgenbild würde an den Dienstanbieter gesendet und dieser würde die bearbeitete Version des Bildes zurückliefern.

4.4.4 D_KONS

Der Dienst D_KONS soll die Konsultation eines Arztes aus einer anderen Einrichtung ermöglichen. Der Dienst handelt die zeitliche und organisatorische Abfolge einer Konsultation aus. So kann der Patient, wenn er transportfähig ist, in die andere Einrichtung transportiert werden. Alternativ können Daten des Patienten wie Laborwerte zum Arzt übermittelt werden, der konsultiert werden soll. Wird eine Konsultation für mehrere Patienten angefordert, so kann

der Arzt auch zum Patienten kommen. Beide möglichen Vorgehensweisen sind jeweils mit einer Bewertung hinterlegt, und aufgrund der Regeln der Computational Economy kann entschieden werden, welche Form wann bevorzugt wird. Als Rückgabewert dient ein Befund, der mit einer Signatur des Konsultanten versehen ist.

4.4.5 D_OPER

Der Dienst D_OPER soll eine ferngesteuerte Operation ermöglichen. Immer dann, wenn für einen Eingriff Spezialwissen erforderlich ist, das zur Zeit vor Ort niemand besitzt, kann eine Operation ferngesteuert stattfinden. Dazu ist eine große Menge an Vorbereitungen nötig. Die benötigten Materialien und Werkzeuge müssen geplant und beschafft werden. Zeiten müssen abgestimmt werden. Eine Operation durch einen ferngesteuerten Roboter kann nur dann realisiert werden, wenn die entsprechend sichere und stabile Netzwerkverbindung besteht. Die Operation selbst wird dann mittels spezieller Module realisiert, welche von den Webservices der Infrastruktur gesteuert werden. Aber auch die Anleitung eines vor Ort befindlichen Operateurs ist denkbar. Der Dienst muss vorbereitend auf die Operation die Stabilität der Verbindung prüfen.

4.4.6 D_LAB

Nicht jedes Krankenhaus besitzt ein eigenes Labor. Die Häuser, welche ein Labor innerhalb ihrer Einrichtung haben, bilden meist mit niedergelassenen Ärzten eine Laborgemeinschaft. Der Dienst D_LAB dient zum Anfordern von Laboraufträgen. Diese werden mittels eines Order-Objektes an das Labor übertragen. Die Proben können mit einem Barcode versehen werden, welcher der ID des Schedules entspricht. Anhand der sched_id können die Ergebnisse der Untersuchung dem richtigen Patienten zugeordnet werden. Die Ergebnisse werden in Form eines oder mehrerer Value-Objekte übertragen.

4.4.7 D_MRT

Nicht jede Einrichtung kann und darf einen Computertomographen vorhalten. Wiederum muss eine Einrichtung, die ein solches Gerät besitzt, dafür sorgen, dass es ausgelastet ist. Der Dienst D_MRT handelt einen Termin für eine radiologische Untersuchung aus. Nachdem der Schedule in der Rolle „Arzt“ übertragen wurde, wird ein Auftrag in Form eines Order-Objektes an den Dienstleister geschickt. Dieser bestätigt den Auftrag mit einem Date-Objekt. Anschließend wird ein Order-Objekt an einen Transportdienst gesandt. Dieser kann Informationen über die Zieleinrichtung vom Initiator anfordern. Der Transportdienst muss vier Service-Objekte abarbeiten. Zuerst muss er quittieren, dass er den Patienten abgeholt hat. Sobald der Patient beim Dienstleister angekommen ist, quittiert der Transportdienst auch diese Task. Anschließend wird der Dienstleister aktiviert und muss nun die Untersuchung durchführen. Ist das geschehen, wird der Transportdienst wieder aktiviert und muss nun den Patienten abholen und beim Initiator abliefern. Durch die Service-Objekte, welche die verschiedenen Stationen des Schedule markieren, kann jede Einrichtung erkennen, wie der Status des Schedules ist und wo sich der Patient gerade befindet. Das Ergebnis der Untersuchung wird mit einer Ressource verknüpft, die für das Attribut

„presence“ den Wert zwei besitzt. Damit werden die Bilder an den Initiator geschickt, sobald sie im Pacs angekommen und in der elektronischen Patientenakte des Gateway mit der Ressource verknüpft sind. Eine Bezahlung kann mit Hilfe der Übertragung eines Money-Objektes erfolgen. Beim Dienstleister sind die Bilder keinem Patienten zugeordnet. Die Bilder werden unter der Schedule-ID im PACS abgelegt, die der Patient als Barcode zum Dienstleister mitbekommen hatte. Ist der Schedule abgearbeitet, löscht der Dienstleister die Bilder, da sie keine Relevanz für ihn besitzen.

5 Suche

5.1 Suche über LDAP

Das Gesundheitssystem besteht aus verschiedenen Dienstleistern, welche unabhängig und doch gemeinsam für die Gesundheit des Patienten sorgen sollen. Damit beispielsweise die Daten eines Patienten aus früheren Episoden herangezogen werden können, muss zuerst eine Patientensuche durchgeführt werden. Erst wenn klar ist, in welchen Einrichtungen ein Patient bereits behandelt wurde, kann gezielt nach Informationen gefragt werden. Dafür ist eine verteilte Suche notwendig. Eine Möglichkeit um eine solche Suche zu realisieren ist der Einsatz eines Verzeichnisdienstes. Verzeichnisdienste greifen auf hierarchisch strukturierte verteilte Datenbanken zu. Deren besondere Eigenschaft besteht darin, dass ein lesender Zugriff auf Datenobjekte sehr schnell erfolgen kann, während eine Schreiboperation wesentlich mehr Zeit in Anspruch nimmt. Bei Anwendungsfällen wie Nutzerdatenbanken oder Adressbüchern wird in der Mehrzahl der Fälle lesend zugegriffen. Der Einsatz eines Verzeichnisdienstes ist unter diesen Umständen gegenüber herkömmlichen relationalen Datenbanken von Vorteil. Ein Protokoll, das sich dafür eignet, ist LDAP. Dabei handelt es sich sowohl

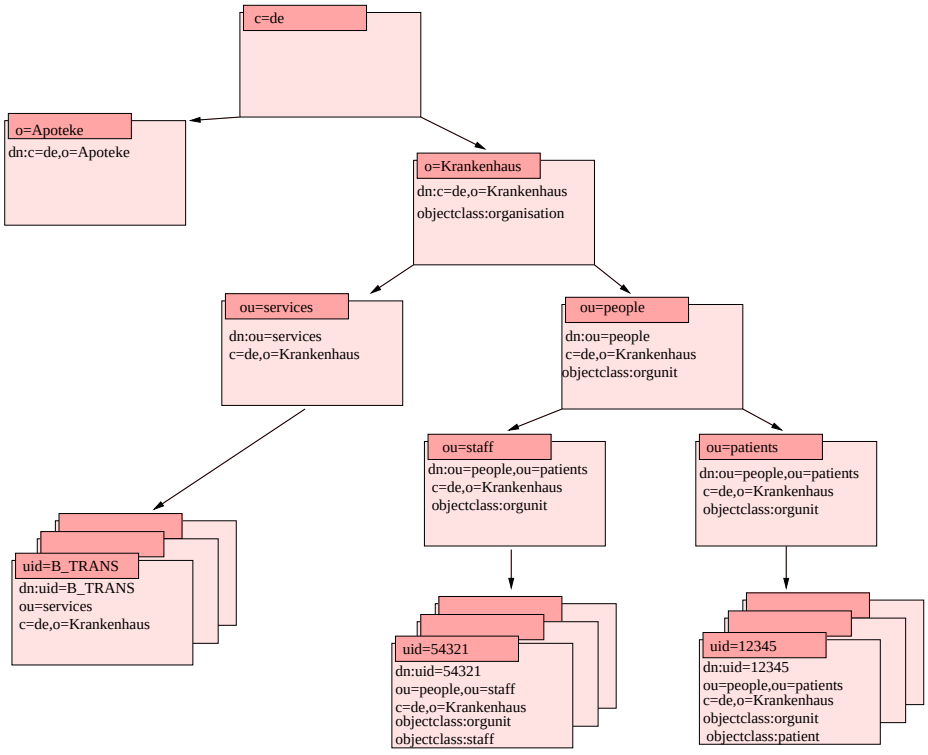

Abbildung 28: Directory Information Tree - DIT

um eine Datenstruktur, als auch um ein Protokoll, um auf diese Struktur zuzugreifen. Eine freie Implementierung dieses Protokolls ist Open-LDAP. Realisiert man die Suche nach Patientendaten mit Hilfe von LDAP, muss eine Instanz einen sog. Directory-Server vorhalten, in welchen die anderen Knoten ihre Informatio-

nen eintragen. LDAP unterstützt auch den Aufbau von verteilten Datenbanken. Somit könnte jeder seine eigenen Daten in sein Verzeichnis einfügen. LDAP ist jedoch hierarchisch aufgebaut. Auch im Falle einer verteilten Struktur sind nie alle Knoten gleichberechtigt. Der Directory Information Tree besitzt eine Wurzel, von der aus alle Zweige erreichbar sind, wie in Abbildung 28 beispielhaft dargestellt ist. Diese Wurzel muss von einem der Knoten verwaltet werden. Ein solcher Knoten, auf dem sich die Wurzel des DIT befindet, stellt ein enormes Sicherheitsproblem dar. Wird diese Wurzel kompromittiert, ist der gesamte DIT unsicher. Ein Vorteil der Datenstrukturen in einem Verzeichnis ist die gute Performance beim Lesen von Datenobjekten. Um später Daten eines Patienten zu finden, muss für jeden Datensatz ein Eintrag im Verzeichnis geschrieben werden. Eine große Anzahl dieser Einträge wird vielleicht nie wieder oder erst nach langer Zeit abgefragt. Dadurch verschiebt sich das Verhältnis von Schreib- und Lesevorgängen zu Ungunsten der LDAP-Datenstruktur. LDAP-Verzeichnisse eignen sich hervorragend für den Einsatz innerhalb einer Organisation. Im Gesundheitswesen arbeiten viele ganz unterschiedliche Organisationen zusammen. Das erschwert den Einsatz eines gemeinsamen Verzeichnisses oder macht ihn nahezu unmöglich.

5.2 Strukturierte P2P-Netze

Ein anderer Ansatz, der für die Suche von Patientendaten Verwendung finden kann, ist der Ansatz des Peer-to-Peer-Netzwerkes. Ursprünglich für den Austausch von Dateien und deren Suche entwickelt, sind Peer-to-Peer-Technologien heute weit verbreitet. Sie erlauben es, in einer Menge von vollkommen gleichberechtigten Knoten zu suchen und mit diesen zu kommunizieren. [WMS06] P2P-Netzwerke kann man in unterschiedliche Kategorien unterteilen. So unterscheidet man strukturierte und unstrukturierte Netze. Die strukturierten P2P-Netze gehören zu den sogenannten Netzen der ersten Generation. [Bis03] Sie sind dadurch gekennzeichnet, dass der Raum der IDs, welche ein Peer verwaltet, durch die Topologie des Netzes bestimmt wird. Jedes Peer hat eine ID, die es eindeutig definiert. Nun verwaltet jedes Peer eine bestimmte Menge von Tupeln, welche aus einem Schlüssel und einem zugehörigen Wert bestehen. Sucht ein Peer den Wert mit Hilfe eines Schlüssels, der von einem anderen Peer verwaltet wird, so muss das Peer ein anderes Peer anfragen. In einem strukturierten P2P-Netzwerk kann eine Verbindung vom Schlüssel zu der ID des Peers hergestellt werden, auf dem das gesuchte Tupel gespeichert ist. Dieser Zusammenhang kann beispielsweise durch verteilte Hashtabellen berechnet werden. Durch die Topologie des Netzes können Peers, welche eine Suchanfrage empfangen, diese Anfrage nach bestimmten Regeln an einen Peer weiterleiten, der dem gesuchten Peer näher ist und so kommt die Anfrage an das Ziel. Diese Struktur des P2P-Netzwerkes, in welchem jedes Peer neben seiner IP-Adresse eine ID bekommt und die Regel, nach welchen Anfragen weitergeleitet werden, wird auch Overlay-Netzwerk genannt. Es handelt sich dabei um eine logische Topologie, die auf der darunter liegenden Netzwerkstruktur, die auch Underlay genannt wird, aufsetzt. Dieses Underlay-Netzwerk kann beispielsweise die TCP/IP Netzwerkstruktur sein. Durch das Overlay-Netzwerk ist ein Fluten des P2P-Netzes nicht nötig und Anfragen können gezielt weitergeleitet werden. Im Idealfall steigt die Anzahl der notwendigen Weiterleitungen, die auch Hops genannt werden, mit Erhöhung der Anzahl der Peers logarithmisch. Dabei besitzt kein Knoten ein

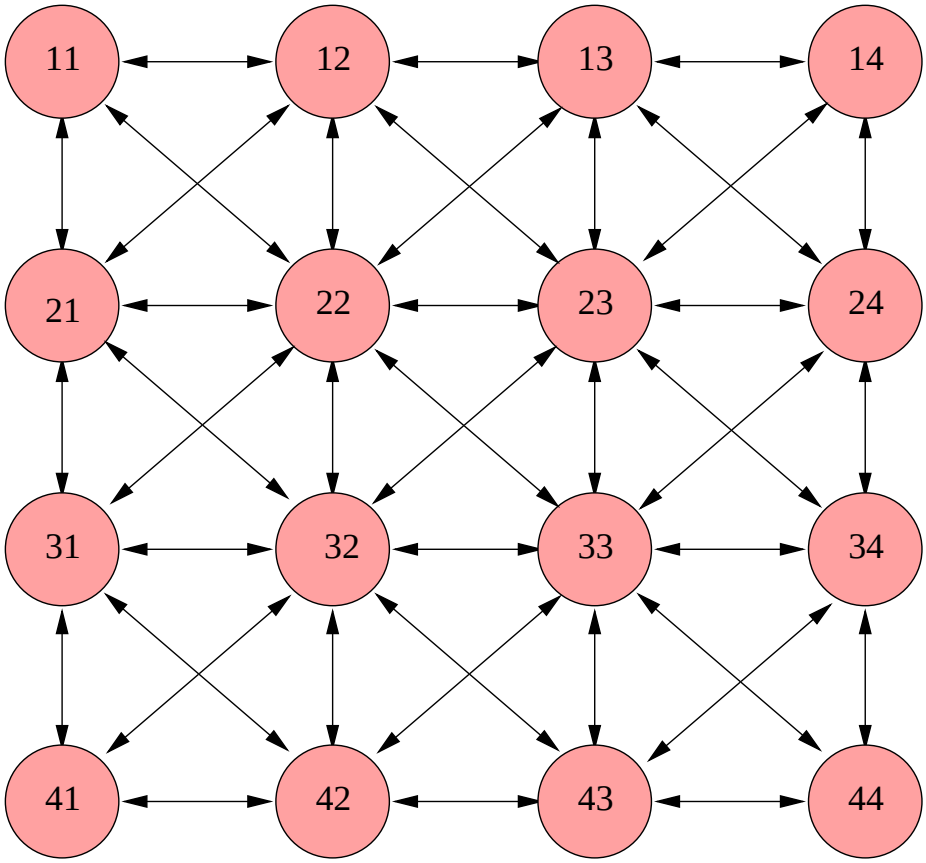

Abbildung 29: Strukturiertes P2P-Netz

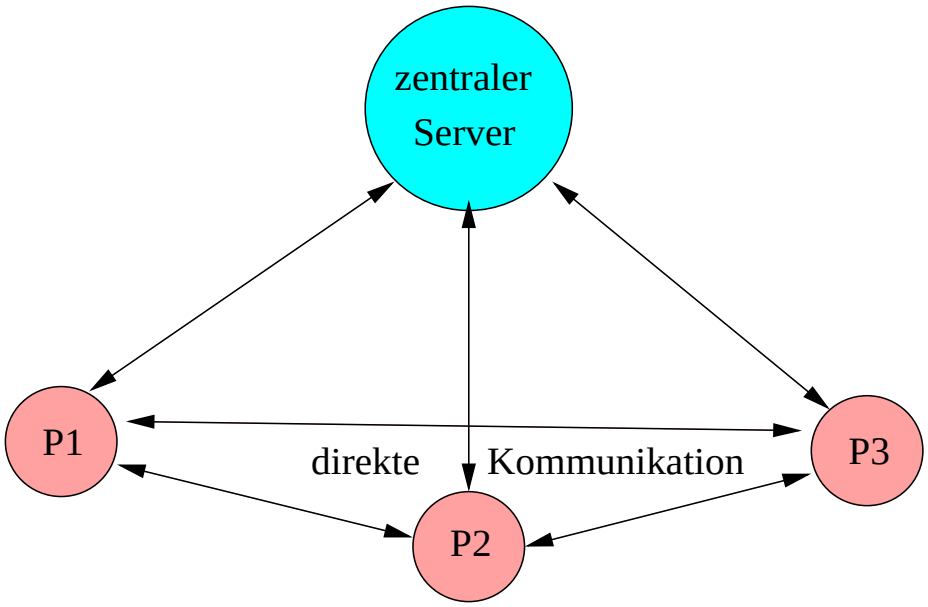

Abbildung 30: Zentralisiertes P2P-Netz

globales Wissen über die Struktur dieses Overlay-Netzwerks, was die Überwachung solcher Netzwerke erschwert. [LMKM06] Das in Abbildung 29 dargestellte P2P-Netzwerk ist strukturiert. Gehen wir davon aus, dass jeder Schlüssel einen Präfix besitzt, der die ID des Knotens angibt. Sucht Peer 11 beispielsweise den Wert 4499, so leitet Peer 11 die Anfrage an Peer 22, dieser an Peer 33 und dieser wiederum an das Ziel weiter. Es sind also in einem derart strukturierten Netz mit 4^2 Peers maximal zwei Weiterleitungen nötig, um den Zielknoten zu finden. Bei einem Netz von 5^2 Peers sind maximal 3 Weiterleitungen notwendig. Die Berechnung der maximalen Anzahl von Hops, abhängig von der Anzahl der Peers X, wäre also im Beispiel $log(X) - 2$. Diese Formel wird von der Topologie des Netzes bestimmt. So gibt es Netze, die ringartig aufgebaut sind. Andere Netze bilden einen n-dimensionalen Raum. Desto mehr Nachbarn jedes Peer hat, umso kürzere Wege existieren im Netz. Eine Suche führt dann also mit weniger Hops zum Erfolg. Außerdem ist ein strukturiertes Netz, in welchem die Peers viele Nachbarn besitzen, stabiler. Der Ausfall eines Peers kann leichter kompensiert werden.

5.3 Unstrukturierte P2P-Netze

Unstrukturierte P2P-Netze sind solche, in denen es keine klare Zuordnung der ID zu der Topologie des Netzes gibt. Auch die Tupel aus Schlüssel und Wert sind nicht auf die Peers aufgeteilt, sondern können in allen Peers vorkommen. Die unstrukturierten Netze kann man wiederum in zentralisierte Netze und reine P2P-Netze untergliedern. Zentralisierte P2P-Netze nutzen einen zentralen Server, an welchem sich die Peers anmelden und über den ein Peer mit anderen Peers Verbindung aufnehmen kann. In Abbildung 30 ist ein solches Netz skizziert. Die eigentliche Kommunikation findet dann zwischen den Peers direkt

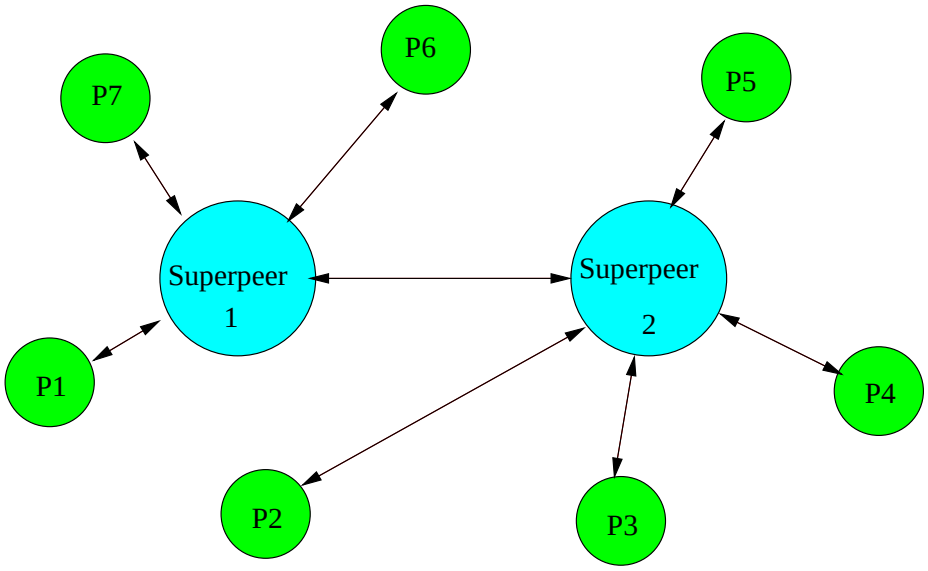

Abbildung 31: Hybrid P2P-Netz

statt. Zentralisierte und strukturierte P2P-Netze bezeichnet man als Netze der ersten Generation. Reine unstrukturierte Netze bezeichnet man als Netze der zweiten Generation. Zwischen diesen verschiedenen Varianten von P2P-Netzen existieren auch Zwischenlösungen, die auch Hybridnetzwerke genannt werden. Das P2P-Netz aus Abbildung 31 ist ein solches Hybridnetz. Es existieren mehrere zentrale Peers. Diese vereinen Vorteile verschiedener P2P-Ansätze. So existieren beispielsweise Lösungen, in denen mehrere Peers eine hervorgehobene Rolle spielen. Diese auch Superpeers genannten Knoten, erfüllen die Funktion eines Servers, wie im zentralisierten P2P-Netz. Nur existieren verschiedene Superpeers, die sich gegenseitig ersetzen können. Ein Beispiel für ein zentralisiertes P2P-Netz ist die Internettauschbörse Napster. Eine besondere Form der reinen P2P-Netzwerke sind friend-to-friend Systeme, die auch web-of-trust-Netzwerke genannt werden. Ein Beispiel hiefür ist das serverlose soziale Netzwerk Retroshare. Mittels RSA-Schlüssel können sich Freunde authentifizieren. Ein Peer kann mit anderen Peers kommunizieren, wenn es in der Freundesliste des Kommunikationspartners eingetragen ist. Die Kommunikationsverbindungen sind von Ende zu Ende mit einer modifizierten Variante von OpenSSL verschlüsselt. Damit können Dateien und Nachrichten innerhalb abgeschlossener Gruppen, ohne die Kontrolle durch den ISP oder außenstehende Personen, ausgetauscht werden. Für das Gesundheitswesen kommen verschiedene P2P-Algorithmen in Frage. Zum einen wäre eine Lösung mit Hilfe eines strukturierten P2P-Netzes denkbar. Kademlia arbeitet beispielsweise mit verteilten Hashtabellen. In diesem Fall müssen Verknüpfungen zu Episoden auf anderen Knoten hinterlegt werde. Durch die spezifische Struktur der Knoten im Gesundheitswesen bietet sich jedoch auch ein unstrukturiertes P2P-Netzwerk ähnlich Freenet an. Der Vorteil hierbei wäre, dass keine Informationen über einen Aufenthalt des Patienten eine Einrichtung verlassen, ohne dass nach dessen Daten gesucht wird. Solange der Patient nicht in einer anderen Einrichtung erneut behandelt wird, bleiben die

Daten im Knoten, ohne dass ein Angreifer außerhalb des Knotens eine Information über den Aufenthalt des Patienten erlangen könnte.

5.4 Small-World-Phänomen

[OS05] Der Psychologe Stanley Milgram führte folgendes Experiment durch: Er ließ 160 Briefe mit der Bitte verteilen, sie an einen befreundeten Aktienhändler weiterzuleiten. Er gab vor, nicht im Besitz der genauen Adresse zu sein. Bei der Auswertung stellte sich heraus, dass die Briefe durchschnittlich nicht mehr als sechs Zwischenadressen durchlaufen hatten. Weiter wurden dem Aktienhändler fast alle der ankommenden Briefe von nur dreien seiner Freunde weitergeleitet. Um die Aussagekraft des Experimentes zu untermauern, wiederholte Milgram den Versuch 1970, indem er Briefe von weißen Teilen der Bevölkerung an Menschen aus der schwarzen Bevölkerung weiterleiten ließ. Dieses Experiment bestätigte die Vermutung, dass der Netzwerkdurchmesser in der Gesellschaft etwa bei sechs liegt. Allerdings ging bei dem Experiment ein nicht unerheblicher Teil der Briefe verloren. Bei einem Test, bei dem ein vermeintlich verlorengegangener Pass verschickt wurde, konnte der Verlust verringert werden. Das ließ den Schluss zu, dass ein entscheidendes Kriterium für den Erfolg der Wert ist, den die beteiligten Personen der Sendung beimessen. Der Wissenschaftler Duncan Watts führte im Jahr 2003 ein ähnliches Experiment mit E-Mails durch. Dazu bat er 61168 Freiwillige aus 166 Ländern, eine Email an einen von 18 Empfängern aus 13 Ländern weiterzuleiten. Den Teilnehmern waren dazu der Name, der Beruf und der Wohnort der Empfänger bekannt. Auch dabei stellt sich heraus, dass die Anzahl der gestarteten Kettenmails, die ihr Ziel erreichten, sehr gering war. Die Kettenmails, welche jedoch ankamen, hatten im Durchschnitt 4,05 Stationen durchlaufen. Nun gab es verschiedene Möglichkeiten, diese Zahlen zu interpretieren. War die durchschnittliche Länge der Kette nur deshalb so kurz, weil alle längeren Ketten abrissen? Der Vorteil eines Experimentes in einem Computernetzwerk ist, dass die Wissenschaftler den Weg der Mail verfolgen konnten. Dadurch konnten die Wissenschaftler herausfinden, dass die Wahrscheinlichkeit dafür, dass eine Person die Mail nicht weiterleitete, an jeder Station nach der ersten 0.65 betrug. Wenn **A** das Ereignis beschreibt, dass eine Mail am Ziel angelangt und **l** die Länge der Kette ist, dann kann man die Wahrscheinlichkeit **P** für das Ereignis **A** folgendermaßen beschreiben.

$$\mathbf{P}(A|L = l) = \prod_{i=0}^{l-1}(1 - r_i) \tag{1}$$

Dabei ist r_i die Wahrscheinlichkeit dafür, dass eine Email nicht weitergesendet wird. Mit steigender Anzahl Weiterleitungen wird die Wahrscheinlichkeit dafür, dass die Email ihr Ziel erreicht, immer kleiner. Im Experiment hatte keine der Emails mehr als 11 Stationen durchlaufen. Der Durchschnitt aller E-Mail-Ketten lag bei sieben Weiterleitungen. Eine Abschätzung, wie lang eine Beziehungskette maximal sein würde, ist schwierig. Die Wissenschaftler fanden jedoch heraus, dass die Hälfte der Mails nach sieben Weiterleitungen ihr Ziel erreicht. Zusammenfassend stellten die Wissenschaftler folgendes fest:

> „Our results suggest that if individuals searching for remote targets do not have sufficient incentives to proceed, the small-world hypothesis will not appear to hold, but that even a slight increase in

> the incentives can render searches sucessful under broad conditions. More generally, the experimental approach adopted here suggests that empirically observed network structure can only be interpreted in light of the actions, strategies, and even preceptions of the individuals embedded in the network: Network structure alone is not everything."

Die magische Zahl von sechs Schritten zwischen zwei Knoten konnte durch das Experiment von Watts nicht glaubhaft bestätigt werden. Es lässt sich aber anhand des Experiments vermuten, dass eine Suche in einem P2P-Netzwerk stark abhängig von der Verhaltensweise der einzelnen Knoten und der Wahrscheinlichkeit ist, mit der eine Nachricht richtig weitergeleitet wird. Aus der Perspektive der Mathematik gesehen handelt es sich bei dem Small-World-Phänomen um eine Aufgabe der Graphentheorie. Die Aufgabenstellung hierbei ist die Messung der Entfernung zwischen zwei Eckpunkten, die dem kürzesten Pfad durch einen Graphen entspricht. Dabei ist die maximale Distanz zwischen zwei Punkten der Durchmesser des Graphen. In zufälligen Graphen sind **V** Punkte jeweils mit einer Wahrscheinlichkeit **P** miteinander verbunden. Ab einer bestimmten Wahrscheinlichkeit **P** steigt der Durchmesser der Graphen mit Anzahl der Punkte logarithmisch. Solche zufälligen Graphen sind jedoch selten gute Modelle für Netzwerke der realen Welt. Eine Eigenschaft, welche viele Netzwerke gemeinsam haben, ist das sogenannte Clustering. Das heißt, Punkte eines Graphen, die einen gemeinsamen Nachbarn besitzen, sind mit einer höheren Wahrscheinlichkeit verbunden, als zufällig ausgewählte Punkte. Ein Wert, der das beschreibt, ist der Clusterkoeffizient **C**. Er gibt die Wahrscheinlichkeit an, mit der zwei Punkte, die einen gemeinsamen Nachbarn besitzen, verbunden sind. Für einen vollständigen Graphen ist **C** beispielsweise 1, für einen Baum 0 und für einen zufälligen Graphen **p**. Solche Graphen lassen sich als sogenannte „Nearest Neighbor" Graphen konstruieren. Diese bestehen aus einem Gitter von Knoten, die in einer Linie jeder mit seinem Nachbarn verbunden sind. Nun wird jeder Knoten mit **k** seiner nächsten Nachbarn verbunden. In Abbildung 32 ist ein solcher Nearest Neighbor Graph skizziert. Jeder Knoten ist mit seinem nächsten und übernächsten Nachbarn verbunden. Wenn **d** die Dimension ist, so berechnet sich der Clusterkoeffizient wie folgt:

$$\mathbf{C} = \frac{3(k-2d)}{4(k-d)} \tag{2}$$

Netze, in denen das Small-World-Phänomen [MSVD] zu beobachten ist, haben meist beide Eigenschaften. Sie besitzen einen geringen Durchmesser und weisen eine starke Clusterung auf. Im Modell von Watts und Strogatz wird mit der Konstruktion eines Nearest Neighbor Graphen begonnen. Anschließend wird ein Anteil der Verbindungen zwischen den Knoten zufällig umgelegt, sodass diese Verbindungen, denen eines zufälligen Graphen entsprechen. Damit bekommt der Graph beide Eigenschaften. Im Experiment von Milgram wurde nicht nur gezeigt, dass die Wege zwischen verschiedenen Personen existieren und sehr kurz sind, es wurde auch festgestellt, dass die Menschen mit Hilfe von wenigen Informationen Nachrichten effektiv weiterleiten können. Ein solches Verhalten ist für die Suche in P2P-Netzen besonders nützlich und es bleibt die Frage, welche Eigenschaften eines Graphen dem Knoten diese Fähigkeit verleihen. John Kleinberg definierte im Jahr 2000 eine andere Familie von semizufälligen Graphen.

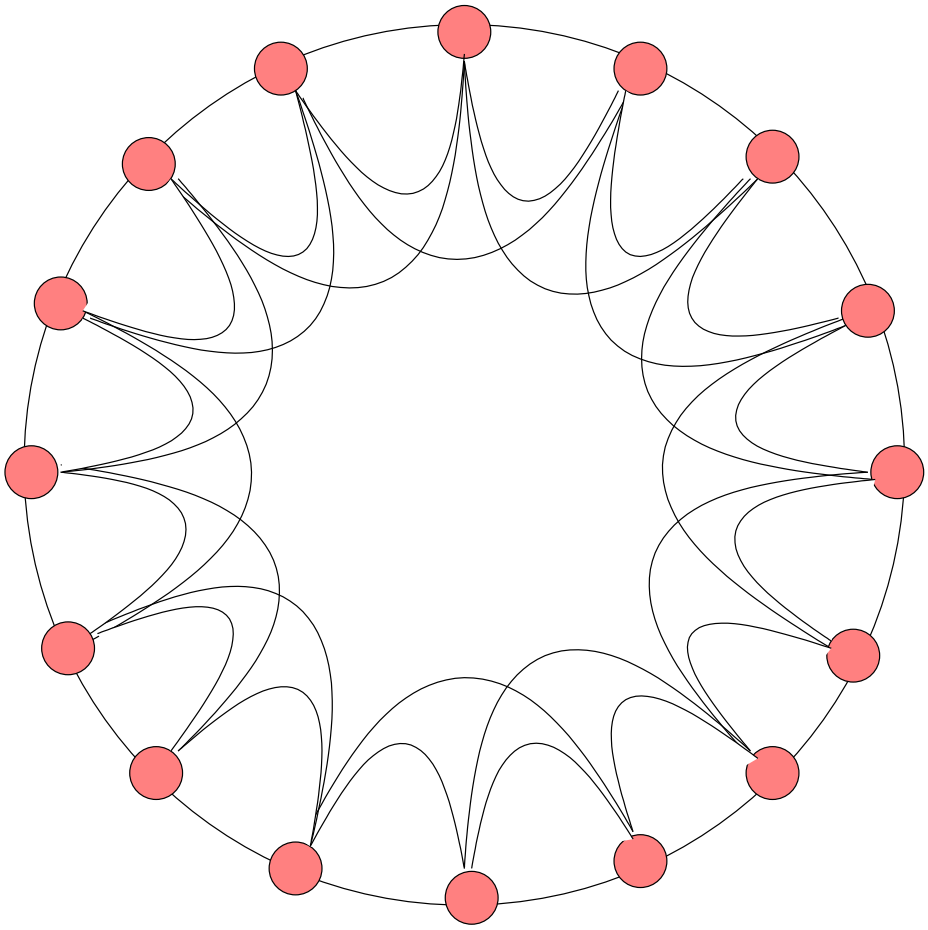

Abbildung 32: Nearest Neighbor Graph mit einer Dimension von 1

Er konstruierte diese mit Hilfe eines darunter liegenden Gitters und knüpfte zufällige Verbindungen zwischen den Knoten, so wie es bereits Newman und Steinberg getan hatten. In seinem Modell ist die Wahrscheinlichkeit dafür, dass zwei Knoten zufällig miteinander verbunden sind, abhängig davon, wie groß die Entfernung zwischen beiden Knoten im Gitter ist. Desto weiter zwei Knoten auseinander liegen, umso unwahrscheinlicher ist eine Verbindung zwischen beiden. Die Wahrscheinlichkeit ergibt sich dabei aus folgender Formel:

$$\frac{\mathbf{d}(x,y)^{-\alpha}}{\Delta_\alpha} \tag{3}$$

Dabei ist $\mathbf{d}$ die Distanz, welche zwischen zwei Knoten liegt, während es sich bei Δ_α um einen Parameter handelt, der darüber entscheidet, ob in einem Netz dieser Klasse effizient gesucht werden kann. Kleinberg fand heraus, dass genau dann, wenn α der Dimension des Netzes entspricht, die Anzahl der Schritte für eine Suche mit der Größe des Netzes polylogarithmisch wächst. Ein solches Netz kann also nach bestimmten Regeln aufgebaut werden, und Shortcuts bei der Initialisierung eines Netzes nach bestimmten Regeln angelegt werden. Ein zweiter Ansatz ist, ein Netz als einfaches Gitter zu implementieren und während der Suche nach Daten das Netz durch Umlegen bestimmter Shortcuts zu optimieren. In diesen Algorithmus können spezifische Eigenschaften des Netzwerkes eingebaut und das Netz damit für den konkreten Anwendungsfall optimiert werden.

5.5 Der Routingalgorithmus

Zu einer erfolgreichen Suche gehört jedoch auch ein entsprechender Routingalgorithmus. Ein Routingalgorithmus ist dann dezentral, wenn die Entscheidung, an welchen Knoten eine Nachricht weitergeleitet wird, ausschließlich vom Koordinatensystem der Gitterstruktur, von der Lage des Knotens, der Lage des Zieles und der Lage aller Nachbarn abhängig ist. Ein einfacher Algorithmus ist das Greedy Routing. Es funktioniert, indem ein Knoten eine Nachricht jeweils an den Nachbarknoten sendet, der dem Ziel am nächsten ist. Die Erfolgsquote einer Suche mit Hilfe des Greedy-Algorithmus ist wiederum von der Beschaffenheit der Verknüpfungen zwischen den verschiedenen Knoten abhängig. Für eine erfolgreiche Suche ist also erstens eine bestimmte Struktur der Verknüpfungen zwischen den Knoten notwendig. Zweitens bedarf es eines passenden Algorithmus, der die Strukturen optimal nutzt und eine Anfrage auf möglichst kurzem Weg zum Ziel führt. Ein Algorithmus muss nun in jedem Knoten eine Routingentscheidung treffen können, deren Summe einen optimalen Pfad durch den Graphen ergibt. Dazu muss der Algorithmus jeweils den Nachbar eines Knotens ermitteln, der dem Ziel am nächsten ist. Netzwerke werden als navigierbar bezeichnet, wenn ein Routing, beispielsweise über den Greedy-Algorithmus, möglich ist. Wichtig dabei ist, dass der Greedy-Routingalgorithmus nicht auf eine Sackgasse stößt. Das kann beispielsweise geschehen, wenn ein Knoten erreicht wird, dessen Nachbarn alle ebenso weit vom Ziel entfernt sind wie der Knoten selbst. Ein weiteres Problem besteht darin, dass es in realen Netzen nicht immer leicht zu entscheiden ist, welcher Nachbar dem gesuchten Knoten am nächsten ist.

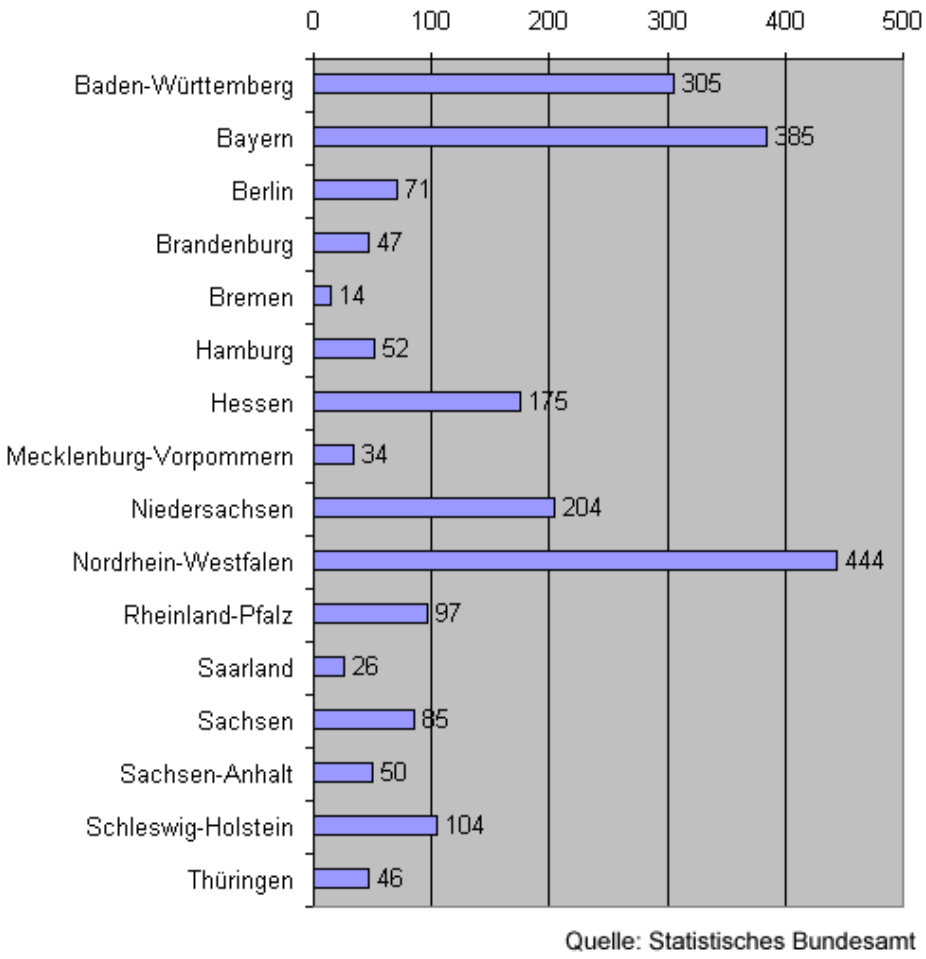

Abbildung 33: Krankenhäuser in Deutschland 2005

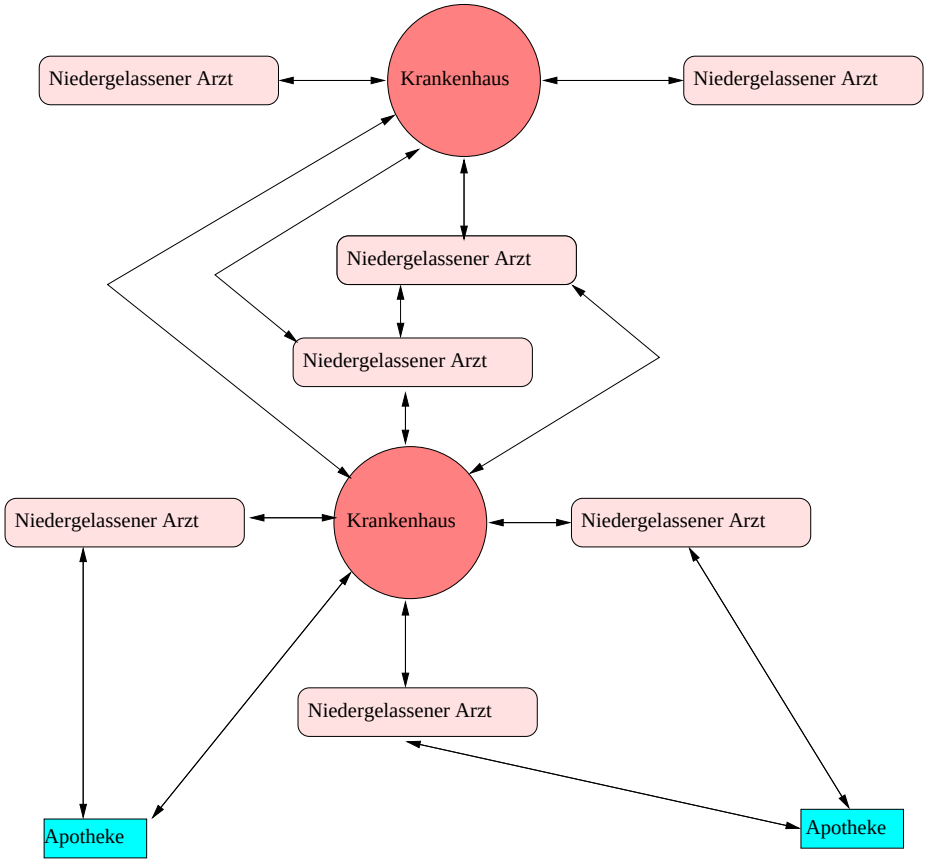

Abbildung 34: P2P-Struktur zwischen Krankenhaus und Arzt

5.6 Struktur eines P2P-Graphen im Gesundheitswesen

Bei der Betrachtung von Small-World-Netzwerken in Kapitel 5.4 wurde deutlich, dass für die effektive Suche in einem unstrukturierten P2P-Netzwerk zwei Aspekte besonders wichtig sind. Zuerst die Struktur des Netzes. Zu viele Verbindungen zwischen Knoten können sich ebenso wie zu wenige Verbindungen, für den Erfolg einer Suche negativ auswirken. Der zweite Aspekt besteht in der Routingentscheidung, die ein Knoten trifft, um eine Anfrage weiterzuleiten. Hier folgt zunächst eine Betrachtung der Struktur im Gesundheitswesen. In Deutschland existieren 2087 Krankenhäuser (Stand 2007). Abbildung 33 zeigt die Verteilung der Krankenhäuser auf die Bundesländer in Deutschland. Dem gegenüber stehen im ambulanten Bereich ca. 150.000 Kassenärzte. Im Jahre 2008 kamen noch ca. 1000 Medizinische Versorgungszentren dazu. In Deutschland existieren rund 21500 Apotheken. Die Zahl der Krankenkassen ist in Deutschland rückläufig und bewegte sich im Juni 2008 um die 218. Die in dieser Umgebung entstehende Struktur ist sehr begrenzt und die Knoten haben ganz unterschiedliche Profile und Aufgaben. Weiter besteht zwischen den an der Behandlung beteiligten Knoten eine starke Clusterung. Die Schnittmenge der Patienten zweier Knoten ist größer, je näher sich zwei Knoten räumlich sind. In dieser Arbeit sollen in erster Linie zwei verschiedene Arten von Suchen betrachtet werden. Zum einen die Suche nach Episoden in anderen Einrichtungen. Und zum anderen die Suche nach Dienstleistungen, die von anderen Knoten in Anspruch genommen werden können. Diese Suchen finden vorwiegend zwischen Krankenhäusern und Arztpraxen statt. Deshalb kann für eine Simulation ein Netzwerk angenommen werden, welches aus den 150 000 Knoten der Arztpraxen und den 2000 Krankenhäusern besteht. In der Praxis bestehen zwischen diesen Knoten natürliche Verknüpfungen. Beispielsweise gibt es für ein Krankenhaus eine Menge von einweisenden Ärzten, die meist in der Region des Krankenhauses angesiedelt sind. Abbildung 34 verdeutlicht, dass Verbindungen zwischen Peers meist zwischen räumlich benachbarten Einrichtungen bestehen.

Für die Zusammenarbeit von Krankenhäusern gilt ähnliches. Dennoch ist es möglich, dass Krankenhäuser oder Arztpraxen Patienten auch über weitere Entfernungen einweisen. Das ist besonders der Fall, wenn es um ganz spezielle Untersuchungen und Therapien geht. Wie die einzelnen Knoten vernetzt werden, wird durch einen Algorithmus bestimmt, der Verknüpfungen zwischen Knoten erstellt. Sind zwei Knoten verknüpft, so gelten diese als Nachbarn. Ein Knoten wird Nachbar, wenn er eine bestimmte Anzahl an Patienten in einer definierten Zeit überweist. Diese Anzahl hängt wiederum von der Entfernung des Knotens ab. Das ist wichtig, damit das dadurch entstehende P2P-Netz die gewünschte Clusterung erhält. Eine Spezialklinik, die sich in weiter Entfernung befindet, wird weniger Patienten überweisen als ein niedergelassener Arzt in der Region. Trotzdem ist eine solche Spezialklinik als Nachbar sinnvoll. Solche Verknüpfungen können als Shortcuts zwischen den Clustern dienen.

5.7 Routing im P2P-Netzwerk des Gesundheitswesens

Das im Gesundheitswesen entstehende Overlay-Netzwerk hat, wie im vorigen Kapitel beschrieben, eine typische Struktur, welche durch die Zusammenarbeit verschiedener Dienstleister bei der Behandlung von Patienten entsteht. Die Verteilung der Daten eines Patienten auf verschiedene Knoten ist von dieser Struk-

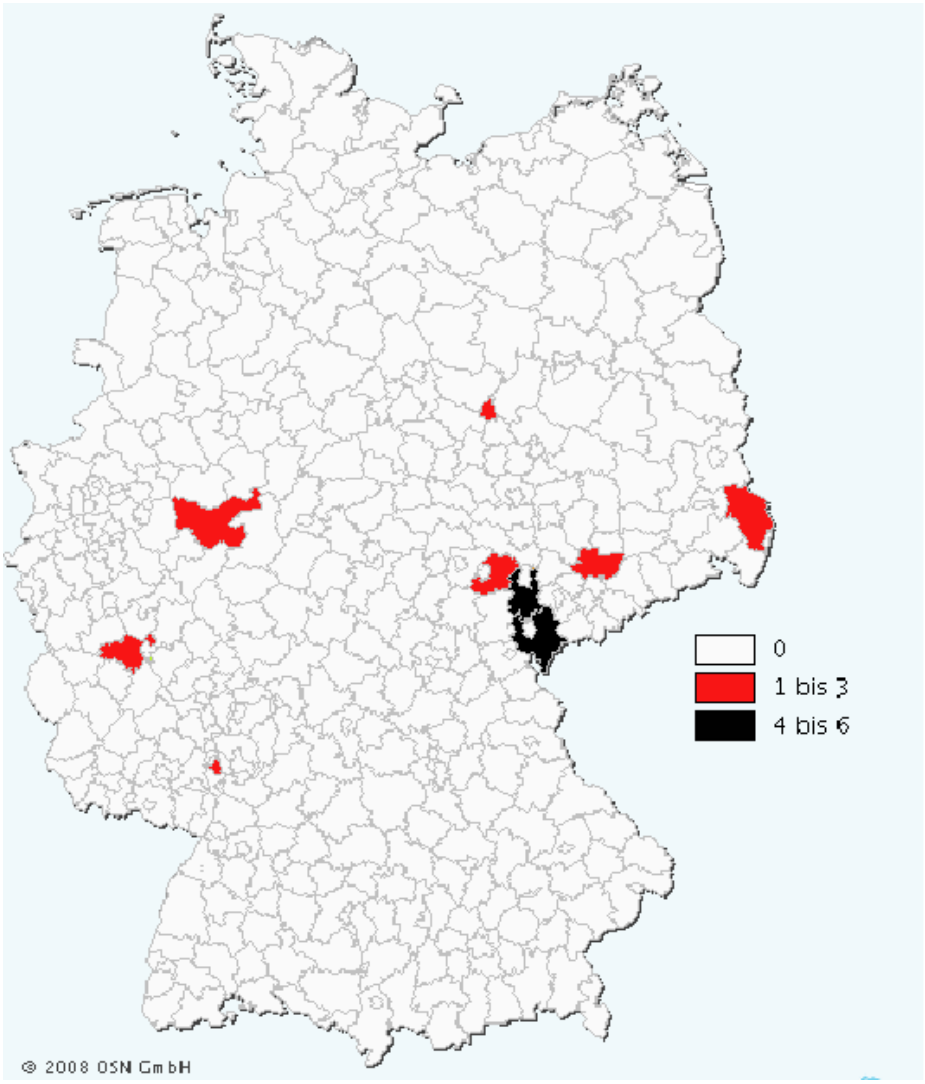

Abbildung 35: Verteilung von Namen

tur abhängig. Ein Patient, der in ein bestimmtes Krankenhaus eingeliefert wird, war mit hoher Wahrscheinlichkeit auch bei Ärzten in dieser Region in Behandlung. Es ist nicht notwendig, alle Krankenhäuser eines Landes zu durchsuchen, um mit einer hinreichend hohen Wahrscheinlichkeit die vollständige Krankenakte eines Patienten zu finden. Diesen Umstand macht sich der hier beschriebene P2P-Suchalgorithmus zunutze. Dieser leitet eine Suchanfrage solange weiter, bis diese auf einen Knoten trifft, der Daten des Patienten gespeichert hat.

Eine solche Suchanfrage besteht aus einer Datenstruktur, in welcher verschiedene Informationen hinterlegt sind, die Einfluss auf die Suche und deren Beantwortung haben. Sollte der Algorithmus in der Praxis eingesetzt werden, so muss der Knoten, der eine Antwort auf eine Suchanfrage zurücksendet, die Antwort verschlüsseln. Dazu benötigt die Datenstruktur einen öffentlichen Schlüssel des Senders. Die Antwort wird dann auf dem gleichen Weg zurückgesendet, den die Suchanfrage genommen hat. Nur der Knoten, welcher die Suche initiiert hat, kann die Antwort entschlüsseln. Es sollte für jede Suchanfrage ein neues Schlüsselpaar erzeugt werden, da sonst ein Zusammenhang zwischen dem öffentlichen Schlüssel und dem Knoten, der die Suche ausgelöst hat, hergestellt werden kann. Außerdem ist eine elektronische Signatur des Patienten mit Zeitstempel nötig. Das ist wichtig, damit nicht jeder Knoten herausfinden kann, wo ein Patient behandelt wurde, sondern nur der, bei dem dieser Patient gerade in Behandlung ist. Das wird noch ausführlicher in Kapitel 6 beschrieben. Weiter muss die Datenstruktur mehrere Identifikatoren enthalten, anhand derer der Patient von Einrichtungen, auf denen dieser bereits behandelt wurde, erkannt wird. Der Grund dafür, dass mehrere Identifikatoren benötigt werden liegt darin, dass dadurch eine Differenzierung möglich ist und nicht jede Suche die gleichen Ergebnisse zurückliefert. In Kapitel 6.10 wird darauf genauer eingegangen. Diese Identifikatoren können mit Hilfe einer Hashfunktion aus den Daten des Patienten gewonnen werden und sollen deshalb Hashwerte genannt werden. Besitzt jeder Patient einen MPI, so kann dieser mit Hilfe einer Einwegfunktion verschlüsselt werden. Der Übersichtlichkeit halber, wird vorerst nur ein Hashwert verwendet. Diese bisher vorgestellten Informationen reichen jedoch für ein Routing nicht aus. In Abschnitt 5.6 wurde deutlich, dass die Verteilung der Daten eines Patienten von der Lokalisation der Knoten abhängig ist. Um in einem solchen Overlay-Netzwerk ein Routing zu ermöglichen, werden in der Datenstruktur der Anfrage Zielkoordinaten hinterlegt. Diese können zuerst einmal anhand des Wohnortes des Patienten ermittelt werden. Es ist jedoch auch leicht möglich, aus anderen Informationen die Zielkoordinaten zu bestimmen.

Listing 6: Definition der Klasse Query

```
class Query:
  def __init__(self,hash,zkx,zky,public_key,sig):
    self.public_key = public_key
    self.sig = sig
    self.id = random.randint(0,1000000)
    self.hash = hash
    self.zkx = zkx
    self.zky = zky
    self.deep = random.randint(28,32)
    self.htl = 1
    self.htl_blue = 1
    self.dist = 4
```

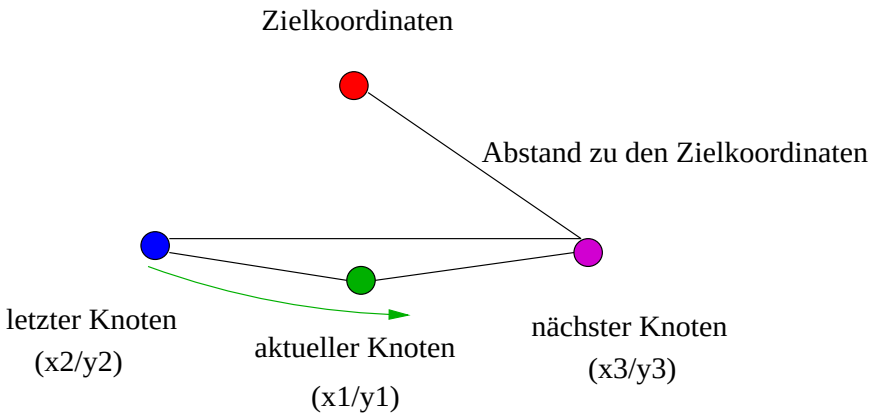

Abbildung 36: Routing in Phase zwei

Abbildung 35 zeigt die Verteilung des Nachnamens Pucklitzsch auf dem Gebiet der Bundesrepublik. Das Ermitteln der Verteilung eines Namens kann als eine Einwegfunktion verstanden werden, da der Name anhand der Karte nicht eindeutig herzuleiten ist. Bei seltenen Namen kann diese Verteilung Zielkoordinaten für eine Suche liefern. Nicht zuletzt ist es möglich, dass der Nutzer die Zielkoordinaten selbst wählt. Beispielsweise ist es denkbar, dass der Patient grob Auskunft darüber geben kann, in welcher Region er bereits in einem Krankenhaus behandelt wurde. Weiter muss jede Suchanfrage eine eindeutige Nummer beinhalten, anhand derer sie von anderen Anfragen unterschieden werden kann. Die zwei Attribute namens „htl“ und „htl_blue“ geben an, wie viele Knoten eine Anfrage bereits durchlaufen hat. Außerdem wird mit jeder Anfrage eine Obergrenze für diesen Wert mitgeschickt. Für das Routing ist noch die Distanze zu den Zielkoordinaten von Bedeutung. Dieser Wert wird ebenfalls in der Datenstruktur hinterlegt. In Listing 6 ist der Konstruktor der Klasse Query abgebildet, in der die relevanten Daten einer Suchanfrage hinterlegt sind. Ein solches Objekt wird nun an einen anderen Knoten gesendet. Die Suche verläuft in vier Phasen und startet mit Phase eins. Die Phasen eins und zwei bilden, wie drei und vier, eine Einheit. Diese beiden Einheiten können sich temporär überschneiden. Sendet ein Knoten eine Suchanfrage an einen anderen Knoten, so übergibt er ein Objekt vom Typ Query und die Phase in Form eines Attributes „typ“, in welcher sich die Suche gerade befindet. Der Routingalgorithmus läuft dann wie folgt ab: Zuerst wird überprüft, ob die Anfrage bereits in der aktuellen oder einer vorhergehenden Phase bearbeitet wurde. Wird eine Anfrage beispielsweise in Phase drei empfangen, wurde aber bereits während der Phase zwei bearbeitet, so wird diese Anfrage verworfen. Das ist nötig, um Schleifen zu vermeiden. Im zweiten Schritt speichert der Knoten den Hashwert des Patienten und dazu den Nachbarn, von dem er die Anfrage erhalten hat. Außerdem wird der Identifikator der Anfrage gespeichert und dazu die Phase, in welcher die Anfrage bearbeitet wurde. Im dritten Schritt wird überprüft, ob die Zielkoordinaten der Anfrage denen des Knotens entsprechen. Ist dies der Fall und die Anfrage befindet sich in Phase eins, so geht die Anfrage in Phase zwei über. Im vierten Schritt wird geprüft, ob der Hashwert des Patienten dem Knoten bereits bekannt ist. In diesem Fall wird parallel zu Phase eins oder zwei eine

Anfrage an den Knoten gesendet, von dem bereits eine Anfrage bezüglich des Patienten gestellt oder eine Antwort geliefert wurde. Diese Anfrage gehört dann zu Phase drei. In Schritt vier wird überprüft, ob der gesuchte Patient bereits in der Einrichtung des Knotens aufgenommen wurde. Ist das der Fall, so wird eine Antwort generiert und an den Sender der Anfrage geschickt. Weiter wird wieder eine neue Anfrage generiert und der Phase vier zugeordnet. Im nächsten Schritt findet ein Routing statt. Der Routingalgorithmus muss nun aus der Menge seiner Nachbarn einen auswählen, an den die Anfragen weitergeleitet werden. Dieser trifft abhängig von der Phase, in der sich eine Anfrage befindet, folgende Entscheidungen:

1. Anfragen, die sich in der Phase eins befinden, werden mit Hilfe des Greedy-Algorithmus, der in Kapitel 7 beschrieben wurde, weitergeleitet. Anhand der Zielkoordinaten kann der Knoten entscheiden, welcher seiner Nachbarn dem Zielgebiet am nächsten ist. In Abbildung 37 entspricht die rote Linie der Phase zwei des Routings.

2. Eine Anfrage, welche sich in Phase zwei befindet, wird geroutet, indem der Nachbarknoten ausgewählt wird, der sich in einem bestimmten geographischen Abstand zu den Zielkoordinaten befindet. Sinn der Sache ist, die nähere Umgebung nach Knoten zu durchsuchen, die eventuell bereits Kontakt mit dem gesuchten Patienten hatten. Dabei wird ein Minimum gesucht, welches sich aus der Summe der Abweichung zwischen dem gewünschten Abstand und dem tatsächlichen Abstand des Nachbarknotens zu den Zielkoordinaten und einem Wert Δ zusammensetzt. Dabei berechnet sich Δ wie folgt:

$$\Delta = \|k3 - (k2 + k3)\| \tag{4}$$

   Dabei gilt:

$$k3 = sqrt(x2 - x3)^2 + (y2 - y3)^2 \tag{5}$$

$$k1 = sqrt(x1 - x2)^2 + (y1 - y2)^2 \tag{6}$$

$$k3 = sqrt(x1 - x3)^2 + (y1 - y3)^2 \tag{7}$$

   Die Werte $(x1, y1), (x2, y2), (x3, y3)$ sind die Koordinaten des aktuellen Knotens, des Knotens von dem die Anfrage kam und des Knotens wohin die Anfrage geroutet werden soll. Durch das Ermitteln des Minimums der beiden Werte wird erreicht, dass die Anfrage an einen Knoten geleitet wird, der einen bestimmten Abstand zu den Zielkoordinaten besitzt. Gleichzeitig läuft die Suche etwa in der gleichen Richtung weiter. Die gewünschte Distanz zu den Zielkoordinaten wird zufällig erhöht, was zur Folge hat, dass der Algorithmus - sofern es das Overlay-Netzwerk erlaubt - die Anfrage eine spiralförmige Bahn beschreiben lässt. Dies ist gut auf Abbildung 37 zu erkennen. Die grüne Linie zeigt das Routing in Phase zwei. Abbildung 36 visualisiert die Knoten, deren Position bei der Routingentscheidung in Phase zwei eine Rolle spielt.

3. Eine Anfrage der Phase drei wird an den Knoten gesendet, von welchem für den gesuchten Hashwert bereits früher eine Anfrage gesendet oder eine Antwort empfangen wurde. Eine Einrichtung sendet nur dann eine Anfrage nach einem Patienten, wenn dieser sich in der Einrichtung

zur Behandlung befindet. Deshalb muss sich am Ende einer Anfragekette immer ein Knoten befinden, der Daten des Patienten gespeichert haben könnte. Diese Anfragekette wird in Phase drei zurückverfolgt. Hierzu wird nicht der HTL-Wert aus Phase eins und zwei verwendet, da dieser bereits verbraucht ist und deshalb die Suche mit einer hohen Wahrscheinlichkeit abbrechen würde, bevor das Ziel erreicht ist. In Abbildung 37 ist das Routing während Phase drei als blaue Linie zu sehen. Es folgt dem Abschnitt der Route einer vorhergehenden Anfrage, die sich offensichtlich in Phase zwei des Routingalgorithmus befunden hat.

4. Eine Anfrage der Phase vier bedeutet, dass einer oder mehrere der Nachbarn, von dem oder zu dem der Patient überwiesen wurde, Daten des Patienten besitzen muss. In dieser Phase wird die Anfrage an alle Nachbarn weitergeroutet, zu denen der Patient eine Verbindung besitzt. Die Phase vier terminiert, indem ein Knoten, der die Anfrage bereits in Phase vier bearbeitet hat, die Anfrage verwirft. Anderenfalls würde die Phase vier in jedem Fall zu einer Endlosschleife führen. Abbildung 37 zeigt die Phase vier mit Hilfe der orangen Linien. Gut zu erkennen ist, wie der Algorithmus die Anfrage an Nachbarknoten leitet, in denen sich definitiv Daten des Patienten befinden.

Listing 7: Routingalgorithmus

```
def query(self,query,peer,typ):
 a = q = []
 if str(query.hash) in self.patients:
   pflag = True
   print "Datensatz␣gefunden␣nach:"
 else:
   pflag = False
 if str(query.id) in self.queries:
   qflag = False
   last_query = self.queries[str(query.id)]
   self.queries[str(query.id)] = typ
   print 'Query␣schon␣bearbeitet!'
 else:
   qflag = True
   last_query = ''
   self.queries[str(query.id)] = typ
 if str(query.hash) not in self.routes:
   rflag = False
   self.routes[str(query.hash)] = {}
   self.routes[str(query.hash)]['admit'] = []
   self.routes[str(query.hash)]['query'] = []
   self.routes[str(query.hash)]['answer'] = []
 else:
   for i in self.routes[str(query.hash)]['query']:
     q.append(i)
   for i in self.routes[str(query.hash)]['answer']:
     a.append(i)
   rflag = True
 if self.x == query.zkx and self.y == query.zky:
   typ = 2
 if peer != None and typ < 3:
   print "Heimat␣des␣Patienten␣erreicht"
   typ = 2
```

```
if peer != None and typ < 3:
  self.routes[str(query.hash)]['query'].append(peer)
hot = query.hot()
hot_blue = query.hot_blue()
if hot:
  if typ < 3:
    min = 200
    n = None
    for p in self.neighbours:
      x0 = query.zkx
      y0 = query.zky
      x1 = self.x
      y1 = self.y
      x2 = self.neighbours[p].x
      y2 = self.neighbours[p].y
      x3 = peer.x
      y3 = peer.y
      k0 = math.sqrt((x1 -x0)*(x1 -x0)\
        + (y1-y0)*(y1-y0))
      k1 = math.sqrt((x1 -x2)*(x1 -x2)\
        + (y1-y2)*(y1-y2))
      k2 = math.sqrt((x1 -x3)*(x1 -x3)\
        + (y1-y3)*(y1-y3))
      k3 = math.sqrt((x2 -x3)*(x2 -x3)\
        + (y2-y3)*(y2-y3))
      k4 = math.sqrt((x2 -x0)*(x2 -x0)\
        + (y2-y0)*(y2-y0))
      delta = math.fabs(k3 - (k1+k2))
      if typ == 2:
        distance = query.dist
        if min > (math.fabs(k4 - distance) + delta)\
          and peer != self.neighbours[p]:
          min = (math.fabs(k4 - distance) + delta)
          n = p
        else:
          if min > k4 and peer != self.neighbours[p]:
            min = k4
            n = p
        query.increment()
        if typ == 2:
          query.inc_dist()
        self.neighbours[n].query(query,self,typ)
if rflag:
  if not qflag and last_query == 3:
    print "Query␣schon␣in␣Phase␣3␣durchgefuehrt"
  else:
    for k in self.routes[str(query.hash)]['admit']:
      if str(k) in self.neighbours:
        self.neighbours[str(k)].query(query,self,3)
        if qflag and hot_blue:
          query.increment_blue()
          for k in q:
            k.query(query,self,3)
          for k in q:
            k.query(query,self,3)
```

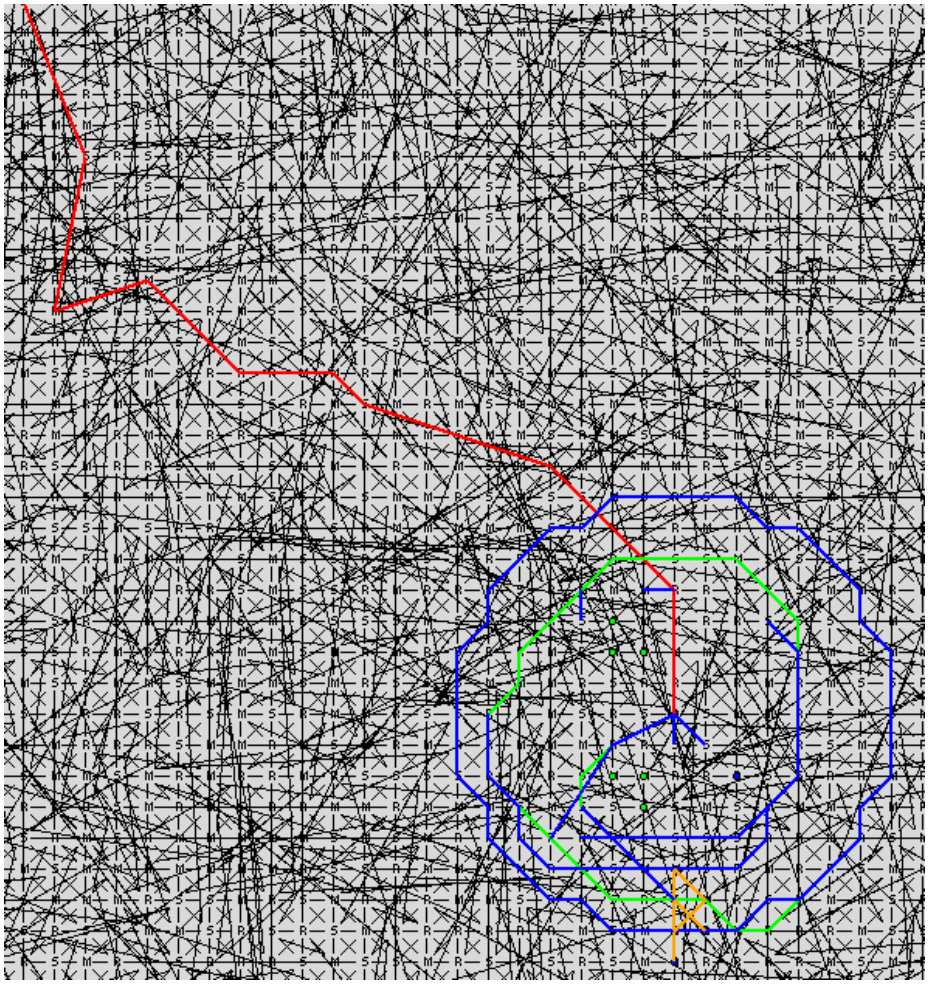

Abbildung 37: Graphische Darstellung der Route einer Suchanfrage

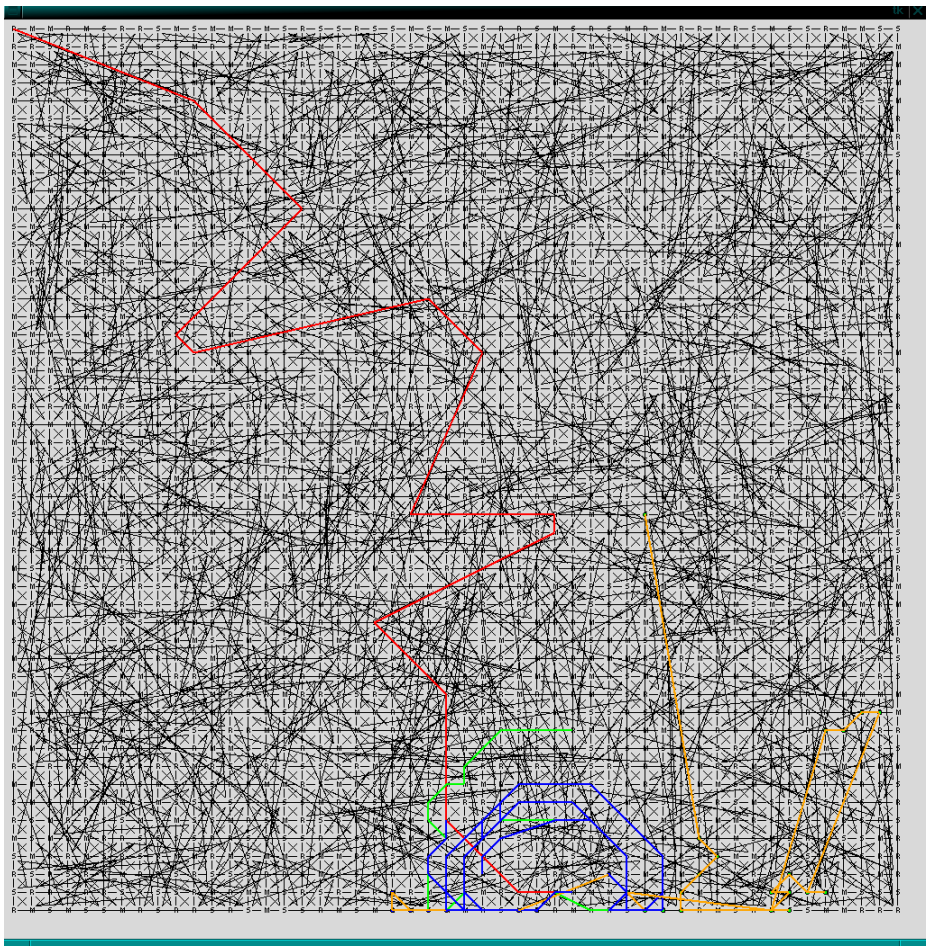

Abbildung 38: Graphische Ausgabe von Hsearch

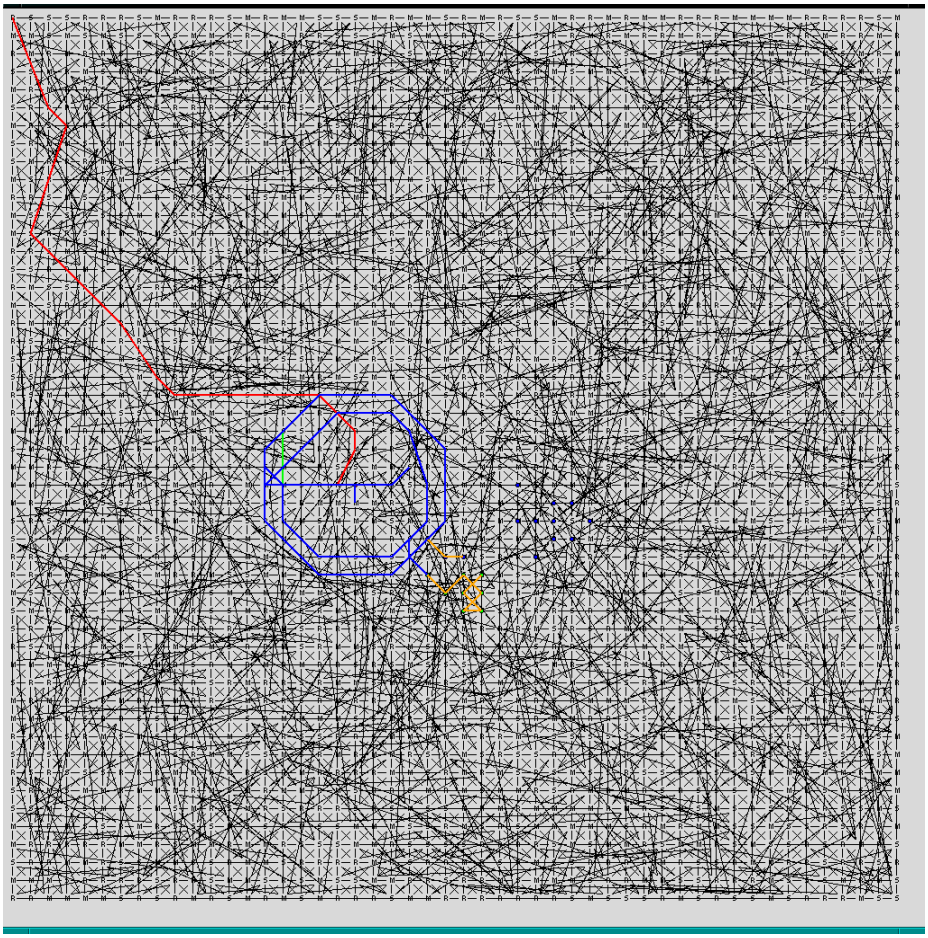

Abbildung 39: Teilweise erfolgreiche Suche

5.8 Testprogramm zum Test des Algorithmus

Abbildung 38 zeigt die graphische Ausgabe einer Implementation des Routingalgorithmus in der objektorientierten Scriptsprache Python. Die Software namens Hsearch simuliert ein Netz, welches aus 2500 Knoten besteht. [YLHea09] Die Anzahl entspricht in ihrer Größenordnung in etwa der Anzahl der Krankenhäuser in Deutschland. Diese Knoten sind in einer quadratischen Matrix angeordnet. Jeder der Knoten besitzt eine Liste, die Zeiger zu all seinen Nachbarn beinhaltet. Bei der Initialisierung wird zuerst jeder Knoten mit seinen acht unmittelbaren Nachbarn verbunden. Anschließend wird für jeden Knoten ein zufälliger Shortcut erzeugt. Dieser wird mittels der Random-Bibliothek ermittelt. Die Länge in X und Y-Richtung ist jeweils normalverteilt, wobei der Erwartungswert null und die Standardabweichung fünf betragen. Dadurch entsteht ein Overlay-Netzwerk, welches auf vereinfachte Art und Weise die Verbindungen zwischen Einrichtungen im Gesundheitswesen abbildet. Jeder Knoten kennt dort seine unmittelbaren Nachbarn und pflegt vereinzelte Beziehungen zu weiter entfernten Knoten.

Nachdem die Software das Netzwerk generiert hat, erzeugt sie einen Patienten, dessen Wohnort zufällig aus einem der 2500 Knoten ausgewählt wird. Anschließend wird simuliert, dass dieser Patient in zwei zufällige Knoten in der Nähe seiner Heimat eingewiesen wird. Dafür werden wieder mit Hilfe der Random-Bibliothek zwei normalverteilte Werte ermittelt, welche die X- und Y-Abweichung von den Koordinaten der Heimat des Patienten darstellen. Dazu wird der Erwartungswert -3 und die Standardabweichung 3 verwendet. Dadurch wird der Patient seltener in dem Knoten eingewiesen, der mit seinen Koordinaten übereinstimmt. Die Einweisung geschieht mit Hilfe der Methode „admit“ des Knotens, welcher eine Instanz der Klasse „Peer“ darstellt. Diese Methode speichert den Hashwert des Patienten in einer Liste. Außerdem verlangt diese Methode einen Parameter names „decision“ der einen Integerwert beinhaltet. Dieser Parameter ist mit dem Wert sieben initialisiert. Die Funktion „admit“ erzeugt einen zufälligen Wert zwischen null und zehn und addiert diesen mit dem Parameter „decision“. Ist die Summe kleiner oder gleich zwei, so wird der Patient an einen Nachbarn, der zufällig ausgewählt wird, überwiesen. Vorher wird der Wert „decision“ dekrementiert. Dadurch wird der Patient in mindestens fünf Knoten aufgenommen. Die Wahrscheinlichkeit sinkt anschließend mit jeder Aufnahme. So werden die Wege eines Patienten durch das Overlay-Netzwerk des Gesundheitswesens simuliert. Wenn ein Knoten in dieser Kette von Überweisungen gefunden wird, so führt das Routing zu allen anderen Knoten, in welchen der Patient behandelt wurde. Wenn die beiden Episoden des Patienten auf die Knoten verteilt sind, so startet die Anwendung eine Suche. Hsearch gibt das entstandene Netzwerk, die Knoten, in denen sich Patientendaten befinden, die Zielkoordinaten und die Route der Suchanfrage graphisch aus. Ein Screenshot dieser Ausgabe ist in Abbildung 8.7 zu sehen. Dabei werden die Knoten, in denen sich Daten befinden, als blaue und grüne Punkte dargestellt. Der Knoten, in dem sich die Heimat des Patienten befindet, wird durch einen roten Punkt hervorgehoben. Während der Suche zeichnet die Software die Route, auf der die Anfrage nach dem Ziel sucht. Dabei wird die Suche während Phase eins rot dargestellt. In Abbildung 8.7 ist gut zu sehen, wie der Algorithmus mit Hilfe des Greedy-Routings die Anfrage immer an den Nachbarknoten weiterleitet, der den Zielkoordinaten am nächsten ist. Sind diese erreicht, dann schaltet der Algorithmus die Suche auf die Phase zwei um. Jetzt wird die Umgebung in ei-

Ergebnis	1. Suche (30)	1. Suche (60)	3.Suche (60)
keine Daten gefunden	30 %	7 %	0 %
Daten gefunden	70 %	93 %	100 %
alle Daten gefunden	21 %	43 %	95 %

Tabelle 3: Ergebnisse des Suchtests

nem bestimmten Abstand durchsucht. Die blaue Linie beschreibt ein Routing der Phase drei. Wurde für einen Patienten vorher keine Suche durchgeführt, so kommt es beim Routing nicht zur Phase drei. Die orange Linie zeigt die Anfragen, die gezielt an Knoten gesendet wurden, von oder zu welchen der Patient überwiesen wurde.

5.9 Bewertung des Routingalgorithmus

Mit Hilfe der im letzten Abschnitt vorgestellten Software wurden Tests durchgeführt, die Anhaltspunkte für die Tauglichkeit des Algorithmus bei der Suche innerhalb des Gesundheitswesens geben sollen. Dazu wurden jeweils 100 Versuche durchgeführt. Beim ersten Test wurde die Obergrenze des HTL-Wertes zufällig zwischen 28 und 32 gewählt, was im Durchschnitt einen Wert von 30 ergibt. Der zweite Test wurde mit einer Obergrenze von durchschnittlich 60 realisiert. Im dritten Test wurde der Versuchsaufbau von Test zwei modifiziert, indem vor der Suche zweimal von zufällig ausgewählten Knoten nach dem Patienten gesucht wurde. Dadurch konnte die Phase drei des Routingalgorithmus genutzt werden. Die Testergebnisse sind in Tabelle 5.9 dargestellt. Die Ergebnisse aller drei Tests zeigen, dass ein solches Overlay-Netzwerk navigierbar ist, d.h. man kann mit Hilfe des Greedy-Algorithmus eine Routingentscheidung treffen. Da die Suche immer vom Nullpunkt aus durchgeführt wurde, verliefen im ersten Test manche Suchen erfolglos. Die Ursache dafür ist der HTL-Wert, welcher manchmal die Obergrenze erreicht hatte, bevor ein Knoten mit Daten gefunden wurde, oft aber bevor alle Knoten gefunden wurden. Die Erhöhung der HTL-Obergrenze brachte in Test zwei bereits erhebliche Verbesserungen mit sich. Die HTL-Obergrenze muss also an die Größe und Struktur des Overlay-Netzwerkes angepasst werden. Möglicherweise kann eine höhere Anzahl von Shortcuts die notwendige Obergrenze senken, da die Knoten in diesem Fall präzisere Routingentscheidungen über eine größere Distanz treffen können. Die vorangestellte Suche nach den Patientendaten konnte die Erfolgswahrscheinlichkeit noch einmal erheblich steigern. In den Fällen, in denen auch in Test drei Daten nicht gefunden wurden, waren diese Daten sehr weit von den Zielkoordinaten entfernt. Dies kann in Abbildung 39 nachvollzogen werden. Die blauen Punkte befinden sich mehr als zehn Knoten entfernt. Aus diesem Grund und durch die Anordnung der Shortcuts in diesem Versuch haben alle drei Suchen diesen Knoten nicht gefunden. Daraus lässt sich schlussfolgern, dass die Wahl der Zielkoordinaten eine große Rolle spielt. Die in diesem Test verwendete Umgebung entspricht nicht exakt der Struktur im Gesundheitswesen. In der Realität werden Patienten nicht zufällig verteilt und stochastisch an andere Einrichtungen überwiesen. Hinter den realen Vorgängen stecken Absichten. Weiter wird in einer realen Infrastruktur ein Patient selten nur zweimal behandelt und es wird auch nicht nur

zweimal nach seinen Daten gesucht. Aus diesen Gründen ist zu vermuten, dass der Algorithmus in einer echten Umgebung mindestens ebenso gute Ergebnisse liefern wird, wie in der hier beschriebenen Testumgebung.

6 Datensicherheitskonzept

6.1 Einleitung

Die Infrastruktur muss ein Datensicherheitskonzept beinhalten, das die Identifikation, Authentifikation und die Zuteilung von Rechten der jeweiligen Kommunikationspartner gewährleistet. Da eine Authentifizierung jeder einzelnen Person, die im Gesundheitswesen tätig ist, schwer schwer ist, wurde im Rahmen der EKG eine Institutionskarte eingeführt. Analog dazu wird bei einer Kommunikation zwischen zwei verschiedenen Einrichtungen, innerhalb der in dieser Arbeit vorgestellten Infrastruktur, immer nur eine Einrichtung authentifiziert und nie eine konkrete Person. Ausnahmen bilden Objekte, welche die elektronische Unterschrift einer Person benötigen. Beim Austausch eines Schedules wird jedoch immer nur geprüft, ob es sich beim Kommunikationspartner um die korrekte Einrichtung handelt. Die Identität von Nutzern muss im jeweiligen Knoten durch das dort eingesetzte Identity Management geprüft und verwaltet werden. Ob eine Kommunikation zwischen zwei Knoten und damit die Übertragung von einen Patienten betreffenden Daten stattfinden darf, ist nicht nur von der Identität der Einrichtung abhängig. Wenn ein Patient sich in ein Krankenhaus begibt, so schließt er durch die Anmeldung einen Behandlungsvertrag mit der Einrichtung ab. Eine Krankenkasse hat einen Vertrag mit ihren Patienten und mit den behandelnden Einrichtungen. Das Vorhandensein solcher Beziehungen entscheidet darüber, ob die Übertragung von Patientendaten legitim ist oder nicht. Im Falle eines Dienstes der in dieser Arbeit definierten Klasse A muss geklärt werden, ob ein Patient bei einer bestimmten Krankenkasse versichert ist. Besteht zwischen der Krankenkasse und dem Versicherten ein Vertragsverhältnis und hat das Krankenhaus eine Rechnung an die Krankenkasse gesendet, die sich auf den betreffenden Patienten bezieht, so hat die Krankenkasse das Recht, die Krankenakte des Patienten anzufordern. Bei Diensten der Klasse B muss sichergestellt werden, dass ein Behandlungsvertrag zwischen dem Patienten und der Einrichtung besteht. Ist das der Fall, dann darf der Dienst einer anderen Einrichtung relevante Daten zurückliefern. Die Daten werden dann dem KIS der anfordernden Stelle übergeben. Für den Zugriffsschutz innerhalb der Einrichtung müssen die lokalen Systeme sorgen. Die Dienste der Klasse C sind weniger kritisch, da hierbei nur anonymisierte Daten übermittelt werden. Dennoch muss verifiziert werden, ob der anfordernde Knoten berechtigt ist, bestimmte Daten abzufragen. Die Beziehungen zwischen Krankenhäusern und den Behörden, die sich mit der Qualitätssicherung und anderen Informationen auseinandersetzen sind eher statischer Natur und deshalb einfach herzustellen. Bei der Klasse D sind die Beziehungen zwischen den Knoten eher wirtschaftlicher Natur. Die Patientendaten sind anonymisiert. Es geht ausschließlich darum, Ressourcen anderer Knoten zu nutzen. Um einen Anreiz für die einzelnen Einrichtungen zu schaffen, selbst Dienste anzubieten, empfiehlt es sich, eine Art „computational economy“ zu implementieren, wie sie bei GridBus eingesetzt wird. Bietet ein Haus selbst Dienste an, kann es auch Dienste anderer Einrichtungen mit der gleichen Wertigkeit aufrufen. Bei der Entscheidung, ob eine Einrichtung einen Dienst nutzen darf oder nicht, spielt also hier eine wirtschaftliche Beziehung die Hauptrolle.

6.2 Datenschutzrechtliche Grundprinzipien

[Gar06] Unter den Bedingungen moderner Datenverarbeitung wird der Schutz des Einzelnen gegen unbegrenzte Erhebung, Speicherung, Verwendung und Weitergabe persönlicher Daten von dem allgemeinen Persönlichkeitsrecht, das in Artikel 2. Absatz 1 in Verbindung mit Artikel 1 des Grundgesetzes beschrieben wird, geregelt. „Das Grundrecht gewährleistet insoweit die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen." Dieser Leitsatz des Bundesverfassungsgerichtes vom 15. Dezember 1983 weist das Recht auf informationelle Selbstbestimmung als Bestandteil der verfassungsmäßigen Ordnung aus. Der durch den Gesetzgeber garantierte Schutz vor unbegrenzter Erhebung von Daten beinhaltet folgende Aspekte: Zum einen müssen die Daten rechtmäßig erhoben werden. Daten, die beispielsweise durch Folter von Personen gewonnen wurden oder für deren Erhebung falsche Tatsachen vorgetäuscht wurden, dürfen nicht gespeichert werden. Die Richtigkeit der gespeicherten Daten muss nicht nur bei der Erhebung überprüft werden, sondern auch in regelmäßigen Abständen erneut untersucht werden. Ein Krankenhaus beispielsweise ist dazu verpflichtet, die Stammdaten eines Patienten zu aktualisieren, falls diese sich ändern. Daten dürfen nur für den Zweck benutzt werden, für den sie erfasst worden sind. Eine spätere Nutzung der Daten mit dem Zweck, andere Erkenntnisse zu gewinnen als ursprünglich Ziel der Erfassung war, ist somit untersagt. Das Erstellen von Kundenprofilen oder der Verkauf von Adressen bestimmter Kunden lässt sich mit Datenschutzbestimmungen in keinster Weise vereinbaren. Dem Bürger wird das Recht eingeräumt, zu erfahren, wer welche Daten von ihm gespeichert hat. Auch Sicherheitsbehörden haben die Pflicht, Auskunft zu erteilen, falls eine Person wissen möchte, welche Daten über sie gespeichert werden. Das Gleiche gilt natürlich auch für Dienstleister im Gesundheitswesen. Aber auch bei den Daten selbst gibt es Unterschiede. Daten, deren Veröffentlichung dazu führen kann, dass Menschen diskriminiert werden, dürfen gar nicht oder nur unter großen Einschränkungen erhoben werden. Solche Daten sind beispielsweise Informationen über das Sexualleben einer Person oder deren religiöse oder weltanschauliche Einstellung. Solche Daten sind aber im Gesundheitswesen oft von großer Relevanz. Beispielsweise lehnen Anhänger bestimmter Glaubensgemeinschaften eine Bluttransfusion mit fremdem Blut ab. Solche Patienten können vor einer Operation eine Eigenblutspende leisten und dadurch auf eine Blutkonserve einer fremden Person verzichten. Um solche Befindlichkeiten zu berücksichtigen, werden auch diese Daten beim Arzt abgefragt. Wenn ein Patient bestimmte sexuelle Gewohnheiten hat, dann kann das dazu führen, dass dieser einer Risikogruppe zugeordnet werden muss. Solche Informationen verbessern die Diagnose und die daraus resultierende Behandlung maßgeblich und sind deshalb relevant. Diese persönlichen Daten, aber auch die diagnostizierten Krankheitsbilder des Patienten selbst, können ein Grund sein, weshalb eine Person unter Diskriminierung zu leiden hat. Daher müssen persönliche Daten und Gesundheitsdaten besonders geschützt werden. Das heißt, diese Daten dürfen nur dann erfasst werden, wenn die betreffende Person explizit eingewilligt hat oder ein Gesetz die Erfassung ausdrücklich erlaubt.

Neben diesen datenschutzrechtlichen Aspekten sind noch andere Schutzziele zu erfüllen. Beispielsweise müssen die Daten von Patienten so gespeichert werden, dass diese Daten nicht verloren gehen können. Ein anderer Aspekt,

der im BDSG verankert ist, befasst sich mit der Verarbeitung von Daten. Es dürfen durch Softwaremodule keine automatischen Entscheidungen aufgrund persönlicher Daten getroffen werden, die für die betreffende Person ungünstige rechtliche oder gesellschaftliche Auswirkungen haben. Die automatische Ermittlung der Kreditwürdigkeit, aufgrund von gesammelten persönlichen Daten, ist deshalb nicht erlaubt. Im Gesundheitswesen gibt es gerade bei der Befundung von Röntgenbildern Softwaremodule, die in solchen Bildern auffällige Veränderungen mit Hilfe von Bilderkennungsalgorithmen finden, und diese markieren. Solche Programme werden teilweise zur Zweit- oder Drittbefundung verwendet. Der Gedanke liegt nahe, dass vielleicht solch eine Software irgendwann automatisch einen Befund erstellt. Ein solcher Befund kann über die weitere Behandlung entscheiden. Schon aus datenschutzrechtlichen Gesichtspunkten wird es im derzeitigen Rechtssystem keine automatische Befundung geben.

6.3 Gesetzliche Grundlagen

Die ärztliche Schweigepflicht wird im Wesentlichen durch zwei Gesetze untermauert. Zum einen durch §35 des Sozialgesetzbuches [sgb04]. Darin werden die Sozialdaten von Personen, die Sozialleistungen in Anspruch nehmen, geschützt. Die Leistungsträger dürfen danach nur die Daten erheben und verarbeiten, zu deren Kenntnisnahme sie befugt sind. Dieser Aspekt ist für den Entwurf einer Telematikinfrastruktur nicht relevant, da ein Verstoß schon bei der Erfassung der Daten vorgenommen wird und nicht erst bei der Weitergabe. Weiter äußert sich das Sozialgesetzbuch aber auch darüber, dass die erfassten Daten auch innerhalb einer Einrichtung nur den Personen zugänglich gemacht werden dürfen, die ausdrücklich dazu berechtigt sind. Zum anderen findet die Weitergabe von Geheimnissen auch im Strafgesetzbuch Erwähnung [stg06]. Im §203 des StGB werden in Absatz 1 explizit Ärzte, Apotheker und Angehörige anderer Heilberufe genannt. Ihnen droht bis zu einem Jahr Freiheitsstrafe, falls sie ein Betriebsgeheimnis oder ein zu einem persönlichen Lebensbereich gehöriges Geheimnis offenbaren. Nach §53a der Strafprozessordnung [spo04] unterliegt auch jemand, der die Tätigkeit eines Arztes unterstützt oder vorbereitet, einer Schweigepflicht und kann nach §203 des StGB zur Rechenschaft gezogen werden. Das ist Grundlage dafür, dass sich auch die Verwaltung eines Krankenhauses oder die IT-Abteilung, welche mit einer IT-Infrastruktur die Arbeit des medizinischen Personals unterstützt, dafür verantworten muss, falls Daten in die Hände unberechtigter Personen fallen. Eine Übermittlung von Daten an andere Leistungserbringer kann demzufolge dann als Ordnungswidrigkeit oder sogar als Straftat gewertet werden, wenn nicht zweifelsfrei sichergestellt ist, dass der Empfänger die Daten mit Zustimmung des Patienten und ausschließlich zu Behandlungszwecken erhält.

6.4 Schutzbedürftigkeit unterschiedlicher Daten

Zum Sicherheitskonzept einer Infrastruktur im Gesundheitswesen gehören eine Analyse der Daten, welche verarbeitet werden sollen und eine Eingruppierung dieser Daten in Gruppen unterschiedlicher Sensibilität. Eine Klasse von Daten bilden die Stammdaten. Diese Daten eines Patienten bestehen aus Informationen wie der Adresse und dem Geburtsdatum einer Person. Sie sind teilweise öffentlich zugänglich, z.B. durch das Telefonbuch. Da der Zweck einer Telema-

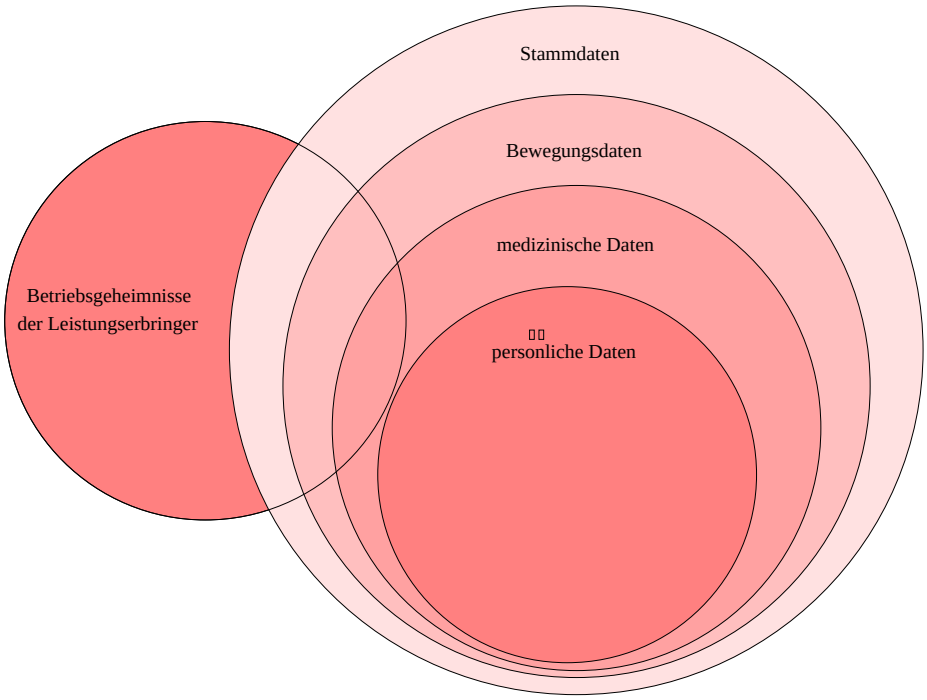

Abbildung 40: Schutzschichten der Daten

tikinfrastruktur die Kommunikation ist, fallen dort Bewegungsdaten an. Diese Bewegungsdaten bilden die zweite Klasse von zu schützenden Daten. Sie sind sensibler als die Stammdaten. Sie können eine Aussage darüber zulassen, wo eine Person behandelt wurde. Dies kann für den Patienten im konkreten Falle schon zu Nachteilen führen. [SS08] Das ist beispielsweise der Fall, wenn ein potentieller Arbeitgeber Informationen über regelmäßige Aufenthalte in einer Klinik mit bestimmtem Profil erhält, und eine Personalentscheidung durch die Kenntnis solcher Bewegungsdaten beeinflusst wird. [NXNea09] Die dritte Klasse von Daten sind medizinische Daten. Diese Daten sind noch sensibler, da sie direkte Schlussfolgerungen auf den Gesundheitszustand des Patienten erlauben. Eine vierte Klasse stellen persönliche Daten dar. Solche werden direkt vom Patienten abgefragt und beinhalten Informationen wie beispielsweise die Religionszugehörigkeit. Die unsachgemäße Verwendung solcher Daten kann dazu führen, dass eine Person Diskriminierungen ausgesetzt wird. Ein fünfte Klasse von Daten schließlich betreffen nicht den Patienten, sondern eine Einrichtung. Die Rechnungsdaten, welche ein Krankenhaus an die Krankenkasse sendet, sind Betriebsgeheimnisse, die konkurrierenden Einrichtungen nicht zugänglich gemacht werden dürfen. Solche Daten können außerdem noch Informationen über Patienten enthalten. Wie die verschiedenen Klassen von Daten ineinander übergeben, ist in Abblildung 40 visualisiert.

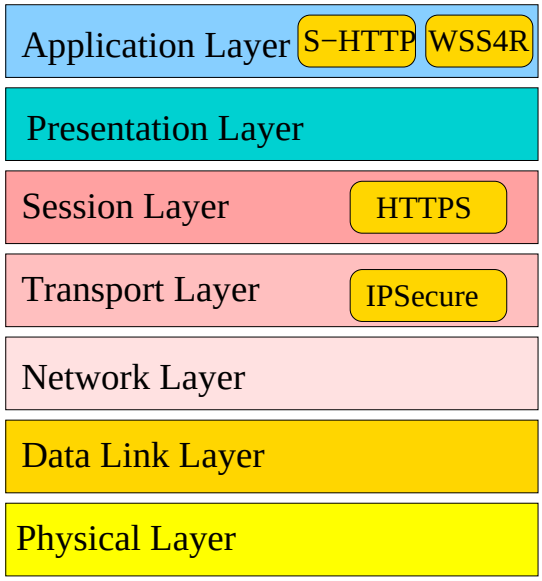

Abbildung 41: Protokolle im OSI-Schichtmodell

6.5 Transportsicherheit und Authentizität

Um Patientendaten vor Zugriffen Dritter zu schützen, müssen die Kommunikationsverbindungen verschlüsselt werden. Die in dieser Arbeit entworfene Dienstarchitektur basiert auf der Kommunikation durch Webservices. Diese Aufrufe werden meist mit Hilfe des HTTP-Protokolls realisiert, welches wiederum auf TCP/IP aufsetzt. Um HTTP-Pakete zu verschlüsseln, kann man sich verschiedener Technologien bedienen, welche auf unterschiedlichen Schichten des ISO/OSI-Referenzmodells [MEK03] aufsetzen. An dieser Stelle sollen vier dieser Technologien auf ihre Vor- und Nachteile bei der Verwendung im Rahmen der Telematik-Infrastruktur untersucht werden. [Res01] Abbildung 41 zeigt die Schichten des ISO/OSI-Referenzmodells und ordnet die vier im Folgenden vorgestellten Technologien jeweils einer Schicht zu.

6.5.1 IPSecure

Eine mögliche Technologie zur Sicherung der Transportschicht ist die Verwendung von IPSecure (IPSec). IPSec ermöglicht die Verschlüsselung von IP-Paketen innerhalb der Schicht 3 des ISO/OSI-Referenzmodells, welche auch Vermittlungsschicht genannt wird. Mittels des Internet Key Exchange Protokolls (IKE), welches auf dem Internet Security Association and Key Management Protokoll (ISAKMP) basiert, werden in regelmäßigen Abständen Schlüssel ausgetauscht. Bevor Schlüssel ausgetauscht werden und die Knoten sich auf einen Verschlüsselungsalgorithmus einigen, müssen sich die Kommunikationspartner gegenseitig authentifizieren. Hierfür existieren zwei Methoden. Die erste nennt sich „pre-shared key“ und setzt voraus, dass im Vorfeld über einen sicheren Kanal ein Schlüssel ausgetauscht wurde und dieser auf beiden Knoten fest installiert ist. Die zweite Methode funktioniert mit Hilfe von Zertifikaten. Das IKE-Protokoll verwendet den Diffie-Hellmann-Algorithmus, um über ein unsicheres

Netz Schlüssel auszutauschen. Mit Hilfe dieser Schlüssel werden die Pakete mit einem Authentification Header (AH) ausgestattet, an dem der Kommunikationspartner die Authentizität des Paketes erkennen kann. AH basiert direkt auf dem IP-Protokoll und verwendet die IP-Protokoll Nummer 51. Es schützt die Verbindung auch vor Replay-Angriffen und stellt die Authentizität von IP-Protokollen sicher. Die Daten werden dann im Encapsulating Security Payload (ESP) in verschlüsselter Form übertragen. ESP setzt ebenfalls direkt auf dem IP-Protokoll auf und verwendet die IP-Protokoll-Nummer 50. Es garantiert die Integrität und Vertraulichkeit der Pakete. Den Zusammenhang zwischen beiden Mechanismen bildet die Security Association (SA). Mittels der ausgetauschten Schlüssel wird eine SA zwischen zwei Adressen hergestellt. Zwischen diesen Knoten können dann Daten in verschlüsselter Form übertragen werden.

6.5.2 Secure Socket Layer

Secure Socket Layer (SSL) ist konzeptionell IPSec sehr ähnlich. Es werden die gleichen Schutzziele wie Authentizität, Integrität und Vertraulichkeit erreicht und es gibt ähnlich der Kommunikation mit IPSec verschiedene Phasen der Verbindung. Zuerst werden Schlüssel ausgetauscht, Protokolle festgelegt und die Kommunikationspartner authentifizieren sich. Anschließend werden Pakete verschlüsselt übertragen. Im Gegensatz zu IPSec arbeitet SSL jedoch in der Session Layer, welche im ISO-OSI-Schichtmodell die fünfte Schicht darstellt. SSL stellt der Anwendung eine API für sichere Sockets zur Verfügung. Will eine Anwendung SSL verwenden, dann muss diese Anwendung angepasst werden. Die Anpassung ist jedoch durch die Analogie zur Socket-API nicht besonders aufwändig. Mit IPSec arbeitet jede Anwendung auch ohne eine Modifikation zusammen. Es erfordert jedoch einen veränderten IP-Stack. Man kann also sagen, dass für die Verwendung von SSL die Anwendung dafür ausgelegt sein muss, während IPSec ein Betriebssystem voraussetzt, welches IPSec unterstützt. Da SSL auf der Session-Layer angesiedelt ist, kann man eine SSL-Verbindung auch über einen Proxy-Server aufbauen. Das ist mit IPSec nicht möglich, da dieses auf der IP-Schicht aufsetzt. Die TCP-Pakete sind dem Proxyserver verborgen. Somit kann er die IPSec-Pakete nicht zusammensetzen, speichern und weiterversenden, wie er es im Normalfall tut. Es ist ebenfalls nicht möglich, eine IPSec-Verbindung durch einen Router, welcher Network Address Translation (NAT) durchführt, aufzubauen. Für SSL ist das kein Problem. Die SSL-Pakete werden durch NAT nicht verändert. Dafür kann man mit Hilfe von IPSec nicht nur den Datenverkehr über TCP/IP, sondern beispielsweise auch UDP-Pakete übertragen. Das ist mit SSL nicht möglich.

6.5.3 Secure-HTTP

Eine dritte Möglichkeit, HTTP-Pakete sicher zu übertragen, ist Secure-HTTP oder auch S-HTTP genannt. Dieses Protokoll setzt auf der auch als Anwendungsschicht bezeichneten ISO-OSI-Schicht sieben auf. Es handelt sich dabei um ein modifiziertes HTTP-Protokoll. Während bei SSL zu Beginn der Kommunikation mehrere Nachrichten übertragen werden müssen, um sich auf Schlüssel und Kryptoalgorithmus zu einigen, sowie sich gegenseitig zu authentifizieren, stecken diese Informationen bei S-HTTP im HTTP-Header. Das Protokoll ist nachrichtenorientiert und ermöglicht, abhängig von den Nutzerdaten, verschiedene Kryp-

tomechanismen einzusetzen. Das S-http-Message-Format basiert auf „Cryptographic Message Syntax“, kurz CMS. In CMS sind fünf Typen von Nachrichten definiert - Data, SignedData, EnvelopeData, EncryptedData und DigestedData. In S-HTTP finden davon nur zwei Typen eine Anwendung, nämlich SignedData and EnvelopedData. SignedData - Messages bestehen aus einem Datensatz und einer oder mehreren Signaturen. Die Nachricht kann außerdem den öffentlichen Schlüssel des Senders enthalten, mit dem der Empfänger nach einer Prüfung dieses Schlüssels die Echtheit der Nachricht testen kann. Eine EnvelopedData-Message enthält verschlüsselte Daten. Die Daten sind mit einem zufällig erzeugten symmetrischen CEK, dem „content encryption key“, verschlüsselt. Dieser Key wird in verschlüsselter Form im Header an den Empfänger mitgeschickt. Der CEK kann auf verschiedene Arten verschlüsselt werden. Eine Möglichkeit besteht darin, den Key mit dem öffentlichen Schlüssel des Empfängers zu verschlüsseln. Es ist jedoch auch möglich, dass der CEK mit Hilfe eines symmetrischen Schlüssels, der durch den Diffie-Hellmann-Algorithmus ausgetauscht wurde, verschlüsselt wird. Es wird jedoch nicht die Nachricht selbst mit dem DH-Schlüssel verschlüsselt. In diesem Fall müsste die Nachricht für jeden Empfänger neu verschlüsselt werden. Wird nur der CEK damit verschlüsselt, kann eine Broadcast-Message mit neu verschlüsseltem CEK ansonsten unverändert an alle Empfänger gesendet werden. In CMS existiert kein Nachrichtenformat, das Signatur und Verschlüsselung enthält. Will man beides erreichen, so muss man die verschiedenen Nachrichtenformate rekursiv benutzen. Man signiert eine Nachricht, um sie dann in eine EnvelopedData-Nachricht zu verpacken. S-HTTP benutzt den gleichen Port wie HTTP. Stattdessen enthält der Header von S-HTTP-Nachrichten Informationen darüber, dass es sich um eine verschlüsselte Nachricht handelt.

Listing 8: S-HTTP-Header

```
Secure * Secure-HTTP/1.4
Content-Type: message/http
Content-Privacy-Domain: CMS
```

Da bereits die Request-URI, sensible Informationen über den Kommunikationsvorgang enthalten kann, wird diese durch einen Stern im Header ersetzt. Dieses Zeichen zeigt dem Webserver, der die Anfrage erhält, dass die eigentliche URI in die Nachricht eingebettet ist. Der Abschnitt „Content-Privacy-Domain“ enthält im Beispiel die Information, dass es sich beim Body um eine mit CMS verschlüsselte Nachricht handelt. Das ist nötig, da S-HTTP auch noch ein anderes Format unterstützt. Dieses Format nennt sich „MIME Object Security Services“ oder kurz MOSS. Mit dem Response verhält es sich ähnlich wie mit dem Request. Er entspricht im Wesentlichen dem Response des HTTP-Protokolls. Da jedoch auch der Status der Antwort schon eine Information liefert, hat ein S-HTTP Response in jedem Fall den Status OK. Ein Link auf eine mit S-HTTP geschützte URL enthält verschiedene Optionen, welche angeben, wie eine Abfrage der URL verschlüsselt sein muss. Ein solcher Link könnte beispielsweise wie in Listing 9 abgebildet aussehen. Das Attribut CN beinhaltet hierbei den Namen des Servers. Der DNS-Name des Servers spielt für die Authentifikation keine Rolle, da jede Nachricht für sich verschlüsselt und signiert wird. So ist es auch möglich, Anfragen an verschiedene Virtual Hosts zu starten. Sendet der Server im HTTP-Header den gleichen CN, so wird er als der gleiche Server erkannt.

Zusammenfassend kann man S-HTTP folgendermaßen mit HTTPS vergleichen: S-HTTP bietet mehr Flexibilität. Während HTTPS nur die zwei Zustände „verschlüsselt“ oder „unverschlüsselt“ abbilden kann, ist es mit S-HTTP möglich, jedes Dokument entsprechend seiner Anforderungen zu verschlüsseln oder zu signieren. Dafür ist der Implementationsaufwand für die Anwendung höher.

Listing 9: S-HTTP Link

```
<CERTS FMT=PKCS-7>
Certificate
</CERTS>
<A DN="CN=Test␣Server,␣O=RTFM,␣Inc.,␣C=DE"
CRYPOOPTS="
SHTTP-Privacy-Enhancements:␣recv-required=encrypt;
SHTTP-Key-Exchange-Algorithms:␣recv-reqired=RSA;
SHTTP-Symmetric-Content-Algorithms:␣recv-required=DES-EDE3-
    CBC"
HREF="shttp://www.rtfm.com/test.html">
Hier Klicken </A>
```

6.5.4 Webservices Security for Ruby

Eine vierte Möglichkeit, Webservices sicher zu implementieren, bietet Webservices Security for Ruby (WSS4R). Dieses Modul für Rails arbeitet mit „ActionWebservices“ zusammen und implementiert die von der OASIS definierten Spezifikationen für die sichere Kommunikation mittels Webservices. Der Vorteil von WSS4R ist unter anderem, dass die damit implementierten Services kompatibel zu den in .NET oder Java programmierten Webservices sind. Die von Roland Schmidt implementierte Bibliothek wird von ihm selbst als „proof of concept“ bezeichnet und beinhaltet nicht alle Funktionalitäten, die von der OASIS für Webservice Security definiert wurden. Dabei wurde sich bei der Implementation auf Verschlüsselung und das Erstellen sicherer Signaturen konzentriert. Das Modul setzt auf der bekannten Bibliothek „soap4r“ auf und kann in den Applikationsserver von Rails integriert werden. WSS4R greift dafür auf die Openssl Bibliothek zu. Um Zertifikate und Schlüssel einzubinden, werden sogenannte Resolver-Objekte benutzt. Sie laden die privaten Schlüssel passend zu einem Zertifikat. Es existieren zwei Implementationen dieses Resolvers. Die erste lädt die Schlüssel und Zetifikate aus einer Datei und die zweite aus einer Datenbank. Zu diesem Zweck kommt eine sqlite3-Datenbank zum Einsatz. Der Rumpf einer SOAP-Nachricht kann wahlweise mit TripleDES oder AES verschlüsselt werden. Diese zwei Algorithmen werden derzeit von WSS4R, unterstützt. Verschlüsselung kann mit einer Signatur kombiniert werden. Dabei kommt es darauf an, in welcher Reihenfolge dies geschieht. Werden die Daten zuerst verschlüsselt, dann wird eine Signatur für die verschlüsselten Daten angefertigt. Umgekehrt bezieht sich die Signatur auf die unverschlüsselten Daten. Der Empfänger muss die Reihenfolge kennen, um die Daten in der richtigen Reihenfolge zu entpacken. Eine sichere Kommunikation im Zusammenhang mit Webservices ist sowohl mit IPSec als auch mit SSL möglich. IPSec würde jedoch voraussetzen, dass spezielle Routen in den einzelnen Knoten hinterlegt werden und VPN-Verbindungen aufgebaut werden. Das ist ein hoher administrativer Aufwand, welcher innerhalb von Konzernen und großen Firmen mit verschiedenen Außenstellen betrieben werden kann. Im Gesundheitswesen muss

die Kommunikation organisationsübergreifend stattfinden. Eine sichere Kommunikation in diesem Bereich würde mit IPSec am Administrationsaufwand scheitern. Die zweite Möglichkeit ist SSL. Das in dieser Arbeit als Framework eingesetzte Rails kann auch mit SSL verwendet werden und so seine Kommunikationsverbindungen über sichere Socket-Verbindungen aufbauen. Der Nachteil davon ist die mangelnde Flexibilität. Die Anwendung hat keinen Einfluss auf die verwendeten Sicherheitsmechanismen. S-HTTP bietet diese Flexibilität. Jedoch ist S-HTTP nicht sehr verbreitet und somit in Rails und anderen Frameworks nicht implementiert. Somit scheidet S-HTTP aus. Als einzige Alternative im Zusammenhang mit Rails bleibt WSS4R welches jedoch von seinem Autor nur als „proof of concept“ bezeichnet wird. Da die Implementation der Dienstarchitektur ebenfalls ein „proof of concept“ ist, eignet WSS4R sich, um die Webservices mit Signaturen auszustatten und zu verschlüsseln. Die Kompatibilität zu anderen WS-Clients eröffnet die Möglichkeit, das in dieser Arbeit vorgestellte und mit Rails getestete Konzept später auch in anderen Sprachen und mit anderen Frameworks zu implementieren.

6.6 Webservices Security

[Law06] Im vorigen Abschnitt wurden verschiedene Mechanismen zur sicheren Kommunikation im Internet vorgestellt. Es wurde begründet, weshalb die Bibliothek WWS4R verwendet werden sollte. Diese Bibliothek setzt die Standards, welche von der OASIS definiert und als Webservice Security veröffentlicht wurden, in Programmcode um. Dieser Abschnitt stellt kurz die von der OASIS definierten Standards und Anforderungen vor. WSS ist in erster Linie eine Erweiterung des SOAP-Standards, welcher Webservices beschreibt, die bestimmten Sicherheitsstandards genügen. Mit WSS lässt sich eine breite Palette von Sicherheitsmodellen wie beispielsweise SSL, SAML, Kerberos oder PKI in Webservices integrieren. Außerdem wird das SOAP-Protokoll mit Hilfe von WSS um Verschlüsselungs- und Signaturfunktionen erweitert. Diese basieren auf XML-Signature und XML-Encryption. WSS beschreibt, wie die entsprechenden Kopfzeilen in den SOAP-Header eingefügt werden. Dabei ist es möglich, viele verschiedene Signaturen einzufügen, die sich auf ganz unterschiedliche Bereiche der SOAP-Nachricht beziehen. So kann eine Kopfzeile von einer anderen Authority signiert sein als der Body der Nachricht. WSS besteht aus sechs verschiedenen Bestandteilen, welche in Abbildung 42 dargestellt sind. WS-Policy ist ein Framework, mit dessen Hilfe Sicherheitsanforderungen und Parameter von Webservices beschrieben werden können. Das ist notwendig, da in WSDL Sicherheitsaspekte von Webservices nicht berücksichtigt sind. WS-Trust beinhaltet ein Vertrauensmodell, mit dessen Hilfe die Behauptung eines Kommunikationspartners verifiziert werden kann. Mit WS-Privacy ist es möglich, die Nachrichten zu verschlüsseln. WS-Secure-Conversation stellt ein Modell für die sichere Kommunikation dar, WS-Federation beschäftigt sich mit den Vertrauensverhältnissen in heterogenen Systemen und WS-Authorisation beinhaltet ein Autorisierungsmodell. WSS definiert ein „Message Security Model“. Dieses Modell arbeitet mit sogenannten „security tokens“. Dabei handelt es sich um Deklarationen, die einen Zusammenhang zwischen einem Schlüssel und einer Identität herstellen. Solche Tokens können ihrerseits von einer Authority signiert sein, um die Echtheit des Token zu gewährleisten. Nachrichten beinhalten Behauptungen, sogenannte Claims. Diese Claims können vom Empfänger mittels eines Trust-WS verifi-

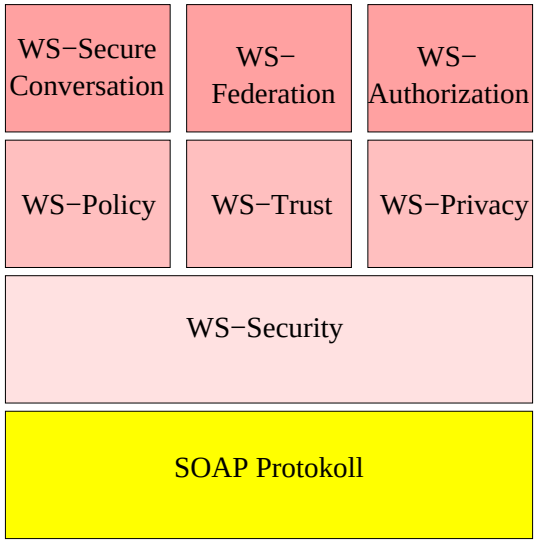

Abbildung 42: Komponenten von WSS

ziert werden. Im Header der Nachricht befindet sich eine URL, unter welcher ein Security-Token abgerufen werden kann. Der dient zur Authentifizierung der Nachricht. Die Security Informationen werden in einer SOAP-Nachricht, in den sogenannten Security Header Block untergebracht. Dieser gilt immer für einen Empfänger. Eine SOAP-Nachricht kann mehrere Security Header Blocks enthalten, dann aber immer für verschiedene Empfänger. Im Security Header Block wird entweder ein Attribut „role“ oder „actor“ definiert. Diese entscheiden, für welche Empfänger der Security Header Block bestimmt ist. Befindet sich dort kein solches Attribut, gilt der Security Header Block für alle Emfänger. Im Security Header Block können auch Tokens enthalten sein, deren Werte binär sind. Das ist z.B. bei X500-Zertifikaten der Fall, welche im Base64Binary-Format kodiert sind. Verschlüsselte Abschnitte werden in das xenc:EncryptedData - Tag eingebettet. Entweder ist ein öffentlicher Schlüssel in die Nachricht eingebettet oder eine SecurityTokenReferenz verweist auf eine URL, unter welcher ein Schlüssel angefordert werden kann. WSS bietet zahlreiche Möglichkeiten, um Benutzer zu überprüfen. Die WSS-Spezifikation beschreibt drei verschiedene Arten explizit:

Listing 10: Schema für UsernameToken

```
<xs:element name="UsernameToken">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Username"/>
   <xs:element ref="Password" minOccurs="0"/>
   </xs:sequence>
  <xs:attribute name="Id" type="xs:ID"/>
  <xs:anyAttribute namespace="##other"/>
 </xs:complexType>
</xs:element>
```

Zum einen eine Authentifizierung über Benutzername und Kennwort, weiter eine Authentifizierung mittels PKI über X.509-Zertifikate und schließlich die Authentifikation mittels Kerberos. Am häufigsten wird die Authentifikation mittels Username und Passwort verwendet. Diese basiert auf einem UsernameToken-Element. Das Schema für dieses Element ist in Listing 10 abgebildet. Dieses Fragment verweist auf zwei weitere Typen: Den Username und das Passwort, welches im SOAP-Header folgendermaßen eingefügt wird.

Listing 11: SOAP-Header

```
<wsse:UsernameToken>
 <wsse:Username>Sven</wsse:Username>
 <wsse:Password Type="wsse:PasswordText">Kennwort</
    wsse:Password>
</wsse:UsernameToken>
```

Wird diese Methode mit Klartext-Passwörtern verwendet, ist sie natürlich sehr unsicher. Deshalb existiert die Möglichkeit, das Passwort mittels Digesthash zu verschlüsseln. Sicherer ist die Authentifikation mittles X.509-Zertifikaten. Mit Hilfe einer Pubic Key Infrastruktur kann dieses Zertifikat einem Nutzer zugeordnet werden. Die Nachricht selbst sollte mit dem öffentlichen Schlüssel des Absenders signiert sein. Das Zertifikat wird dann mittels eines Binary Security Token in die Nachricht eingebunden. Das Zertifikat selbst wird in Base64-Codierung in die Nachricht integriert. Ein solches Fragment richtet sich nach folgendem Schema:

Listing 12: Base64 Fragment

```
<xs:element name="BinarySecurityToken">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="Id" type="xs:ID"/>
<xs:attribute name="ValueType" type="xs:QName"/>
<xs:attribute name="EncodingType" type="xs:QName"/>
<xs:anyAttribute namespace="##other"
processContents="strict"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
```

Bei der Verwendung von Kerberos authentifiziert sich der Nutzer bei einem Key Distribution Center. Dieses stellt dem Nutzer ein Ticket aus. Das Ticket kann in eine SOAP-Nachricht eingebettet werden. Mit Hilfe des Tickets kann der Nutzer nachweisen, dass er sich am Key Distribution Center erfolgreich angemeldet hat. Die Authentifizierung am Key Distribution Center kann wiederum mit einer Nutzername-Passwort Kombination oder mit einem X.509-Zertifikat erfolgen. Dieses Ticket ist zuerst nur der Beweis, dass es sich bei dem Besitzer um einen bestimmten Nutzer handelt. Will dieser Nutzer nun auf eine Netzwerkressource zugreifen, so schickt er das Ticket zu einem Ticket Granting Server. Der wiederum erteilt dem Nutzer ein Service Ticket, welches die entsprechenden Zugriffsrechte für den authentifizierten Nutzer beinhaltet. Damit kann der Nutzer nun auf die Netzwerkressource entsprechend seiner Berechtigung zugreifen. Alle drei dieser Authentifizierungsmechanismen können dazu

verwendet werden, eine Nachricht zu signieren. Ein wichtiges Schutzziel ist auch die Verschlüsselung von Nachrichten. Zu diesem Zweck wurde XML-Encryption in den WSS-Standard aufgenommen. XML-Encryption unterstützt sowohl symmetrische Kryptoverfahren als auch asymmetrische Verfahren. Bei der Verwendung eines symmetrischen Kryptoverfahrens kann der Schlüssel beispielsweise mit Hilfe eines Kerberos-Tickets ausgetauscht werden. Asymmetrische Verfahren können den öffentlichen Schlüssel eines X.509-Zertifikates verwenden, um die Nachricht zu verschlüsseln. Eine mit XML-Encryption verschlüsselte Nachricht kann folgende Form haben:

Listing 13: XML-Encryption

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<soap:Header
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
<wsu:Timestamp>
<wsu:Created
wsu:Id="Id-3beeb885-16a4-4b65-b14c-0cfe6ad26800">
2002-08-22T00:26:15Z</wsu:Created>
<wsu:Expires
wsu:Id="Id-10c46143-cb53-4a8e-9e83-ef374e40aa54">
2002-08-22T00:31:15Z</wsu:Expires>
</wsu:Timestamp>
<wsse:Security soap:mustUnderstand="1">
<xenc:ReferenceList>
<xenc:DataReference
URI="#EncryptedContent-f6f50b24-3458-41d3-aac4-390f476f2e51"
    />
</xenc:ReferenceList>
<xenc:ReferenceList>
<xenc:DataReference
URI="#EncryptedContent-666b184a-a388-46cc-a9e3-06583b9d43b6"
    />
</xenc:ReferenceList>
</wsse:Security>
</soap:Header>
<soap:Body>
<xenc:EncryptedData
Id="EncryptedContent-f6f50b24-3458-41d3-aac4-390f476f2e51"
Type="http://www.w3.org/2001/04/xmlenc#Content">
<xenc:EncryptionMethod Algorithm=
"http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
<KeyName>Symmetrischer Schl"ussel</KeyName>
</KeyInfo>
<xenc:CipherData>
<xenc:CipherValue>
InmSSXQcBV5UiT...␣␣Y7RVZQqnPpZYMg==
</xenc:CipherValue>
</xenc:CipherData>
</xenc:EncryptedData>
</soap:Body>
</soap:Envelope>
```

Immer dann, wenn der Security Header Block von einer Instanz aufgelöst wird, ersetzt diese den betreffenden Teil der Nachricht durch die entschlüsselten Daten und sendet die Nachricht an den nächsten Empfänger. Mit WS-Security werden die wichtigsten Schutzziele wie Authentizität, Integrität und Vertraulichkeit mit standardisierten Protokollen wie XML-Authentification und XML-Encryption erreicht. Dabei können unterschiedliche Mechanismen wie symmetrische oder asymmetrische Kryptoverfahren verwendet werden. Im konkreten Fall der übergreifenden Infrastruktur für das Gesundheitswesen sind eine Authentifizierung mit Hilfe von Kerberos und die Verwendung symmetrischer Kryptoverfahren von Vorteil. Die WSS dient dazu, Einrichtungen zu authentifizieren und die Kommunikationskanäle zu verschlüsseln.

6.7 Zugriffsmatrix

Um Daten vor dem Zugriff unberechtigter Personen zu schützen, muss eine Zugriffsmatrix entwickelt werden, welche die unterschiedlichen Datenklassen bestimmten Rollen und Personengruppen zuordnet. In Abbildung 44 sind die fünf Klassen von Daten, verschiedenen Institutionen und Personen zugeordnet. Der Patient hat auf alle Daten uneingeschränkten Zugriff. Dabei ist die Realisierung eines Rechtemanagements im Gesundheitssystem nicht trivial, da die Zugehörigkeit zu einer bestimmten Gruppe nicht automatisch dazu berechtigt, sensible Patientendaten einzusehen. Beispielsweise darf nicht jeder beliebige Arzt Einsicht in die Krankenakte eines Patienten nehmen, sondern nur der, welcher mit der Behandlung des Patienten auf irgend eine direkte oder indirekte Art und Weise betraut ist. Die verschiedenen Beziehungen zwischen einem Patienten und den unterschiedlichen Einrichtungen sind im Abbildung 43 dargestellt. Jede dieser Beziehungen, die durch die Pfeile abgebildet sind, berechtigen die betreffende Instanz zur Einsichtnahme ganz bestimmter Daten. Eine solche Beziehung zwischen zwei Teilnehmern ist nicht immer eindeutig definiert und muss deshalb vereinfacht werden. Bestimmte Annahmen dienen dazu, eine solche Beziehung zwischen Arzt und Patient, Krankenkasse und Patient oder Arzt und Apotheke klar zu definieren und von bestimmten Umständen abhängig zu machen. Im Zentrum dieser Betrachtungen steht selbstverständlich der Patient. Er hat die Hoheit über die ihn betreffenden Daten. Obgleich ein Patient möglicherweise nicht immer in der Lage ist, seine Krankenakte richtig zu interpretieren, hat er immer uneingeschränktes Zugriffsrecht. Die Gesundheitsdaten eines Patienten sind also nicht die Daten des Arztes sondern gehören dem Patienten. [SAWea09] Auch die Rechnung, die ein Arzt oder ein Krankenhaus an die Krankenkasse stellen, darf ausdrücklich vom Patienten eingesehen werden. Ein Patient hat beispielsweise ein Recht darauf, eine Patientenquittung, auf sein Verlangen hin, ausgestellt zu bekommen. Das soll der Transparenz der Prozesse im Gesundheitswesen dienen. Ein Patient hat also prinzipiell das Recht, alle ihn betreffenden Daten einzusehen. Außerdem kann der Patient entscheiden, welche Personen Zugriff auf die Daten seiner elektronischen Patientenakte erhalten sollen. An dieser Stelle muss jedoch etwas vereinfacht werden. Es ist für den Patienten meist schwer einzuschätzen, welche Daten für den behandelnden Arzt relevant sein können. Deshalb muss diese Entscheidung dem Patienten abgenommen werden. Das System muss auf Grund von Beziehungen zwischen Patient und Dienstleister entscheiden können, welche Daten ein Beteiligter sehen muss, damit dieser seine Dienstleistung durchführen kann. Minimalvoraussetzung für einen Arzt,

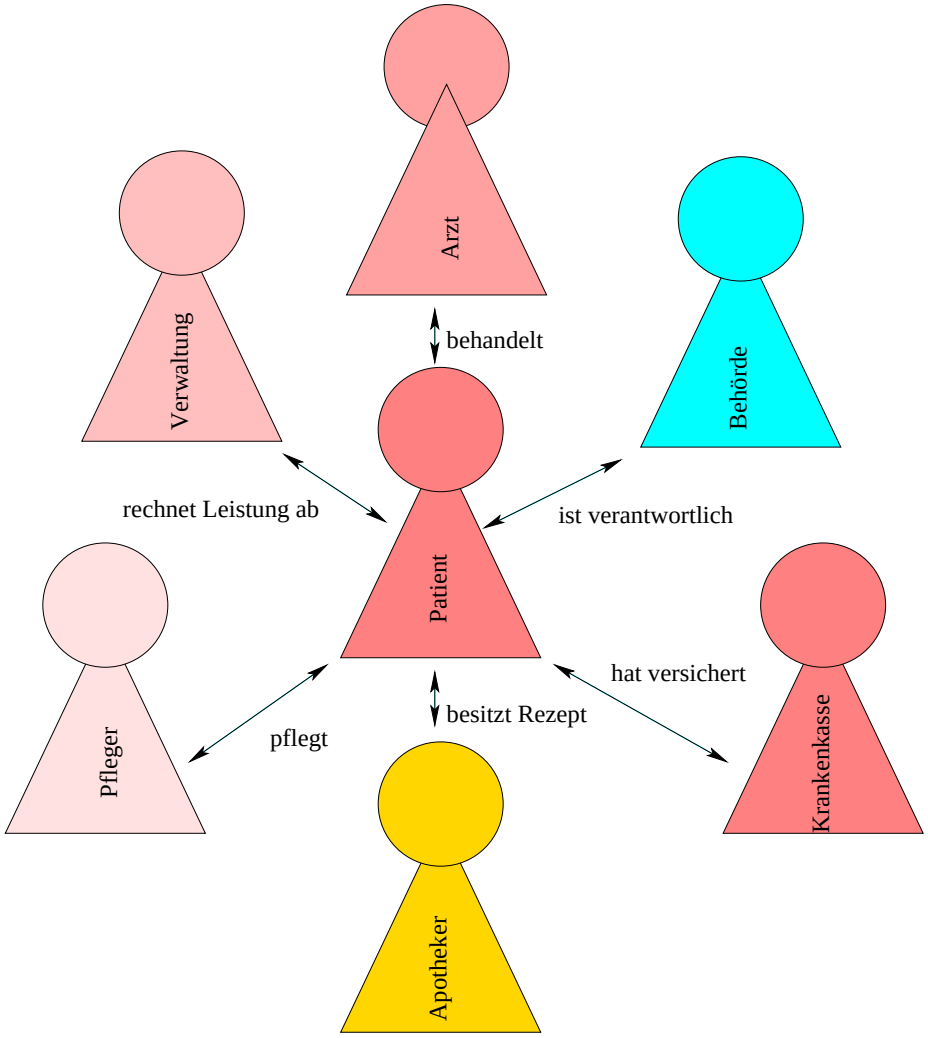

Abbildung 43: Beziehungen zwischen Arzt und Dienstleister

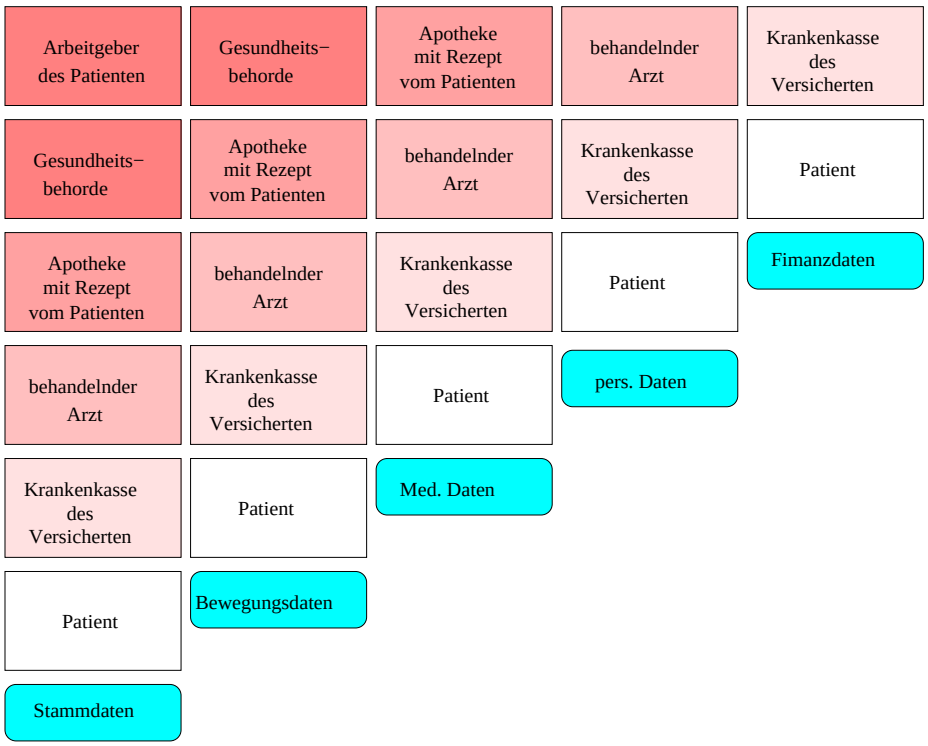

Abbildung 44: Zugriffsmatrix

ist der Zugriff auf medizinische Grunddaten wie beispielsweise die Blutgruppe und die Ergebnisse und Befunde aller Untersuchungen der aktuellen Episode. Diese Daten darf ein Arzt dann abrufen, wenn er bzw. sein Arbeitgeber mit dem Patienten einen Behandlungsvertrag eingeht. Welchen Zugriff eine Person bekommt, hängt also davon ab, zu welcher Gruppe sie gehört, und welche Beziehung diese Person zum Patienten besitzt. Wenn beispielsweise eine Apotheke einem Patienten Medikamente liefert, benötigt sie nicht seine Röntgenbilder. Die Liste aller Medikamente, welche der Patient gerade einnimmt, ist jedoch relevant, um Unverträglichkeiten zu entdecken. Jedoch darf die Apotheke nur dann auf die Daten zugreifen, wenn sie ein Rezept des Patienten besitzt. Um Zugriff auf bestimmte Daten zu erlangen, muss ein Nutzer also zwei Dinge nachweisen: Erstens, dass er zu einer bestimmten Gruppe gehört und zweitens, dass er in einem bestimmten Verhältnis zum Patienten steht, dem die Daten zugeordnet sind. Die Beziehung zum Patienten entscheidet, ob ein Nutzer Zugriff auf Daten erhält, und die Zugehörigkeit zu einer Gruppe ist ausschlaggebend dafür, auf welche Daten der Nutzer zugreifen darf.

6.8 Prüfen der Identität einer Einrichtung

Um den öffentlichen Schlüssel eines Kommunikationspartners zu prüfen, ist in jedem Fall eine vertrauenswürdige Instanz nötig. In der Regel nehmen Trustcenter diese Rolle ein, welche die Identität einer Person oder Einrichtung überprüfen und ein Zertifikat ausstellen, mit dessen Hilfe eine Einrichtung ihre Identität beweisen kann. Die in dieser Arbeit entworfene Dienstarchitektur soll nach Möglichkeit ohne zentrale Knoten auskommen. Für die Suche von Daten wurde deshalb in den vorhergehenden Kapiteln eine P2P-Suche entworfen. Die Rolle der benötigten vertrauenswürdigen Instanz, nimmt in der Dienstarchitektur eine Untermenge der Nachbarknoten ein. Im Netzwerk der Dienstleister im Gesundheitswesen existieren bereits Verbindungen, die bei der Suche nach Daten ausgenutzt werden können. Bei der Prüfung von elektronischen Signaturen können diese Verbindungen zwischen den Knoten ebenfalls nützlich sein. Hierfür wird die zu prüfende elektronische Signatur mittels eines internen Basisdienstes an ein Modul übergeben, welches die Signatur überprüft. Dieses Modul hat eine bestimmte Anzahl an öffentlichen Schlüsseln gespeichert. Von der Richtigkeit der Identität der dazugehörigen Knoten hat sich die Einrichtung überzeugt. So kann beispielsweise ein Krankenhaus mit einer bestimmten Anzahl von niedergelassenen Ärzten und Krankenhäusern aus seiner Region Schlüssel austauschen. Wird nun eine Signatur eines unbekannten Knotens empfangen, so leitet das Modul die Signatur an mehrere der vertrauenswürdigen Nachbarknoten weiter. Mittels der P2P-Struktur kann ein Knoten gefunden werden, der die Identität der zum Zertifikat gehörigen Einrichtung bestätigen kann. Es entsteht eine Vertrauenskette. Wichtig dabei ist, dass ein Knoten, der eine Anfrage von zwei Knoten erhält, nur einmal eine Bestätigung sendet. Ansonsten könnte eine Anfrage über zwei verschiedene Wege von einem Knoten bestätigt werden. Der Knoten, welcher die Signatur prüfen möchte, empfängt die Bestätigungen von zwei seiner Nachbarknoten und sieht die Signatur als von zwei vertrauenswürdigen Knoten bestätigt an. Kann die Identität des Erstellers nicht bestätigt werden, so muss dessen Identität auf anderem Wege nachgeprüft werden. Wird von einem vertrauenswürdigen Knoten die Identität des Erstellers des Zertifikates bestätigt, so wird dieser Ersteller für den Knoten ebenfalls zu einem vertrauenswürdigen

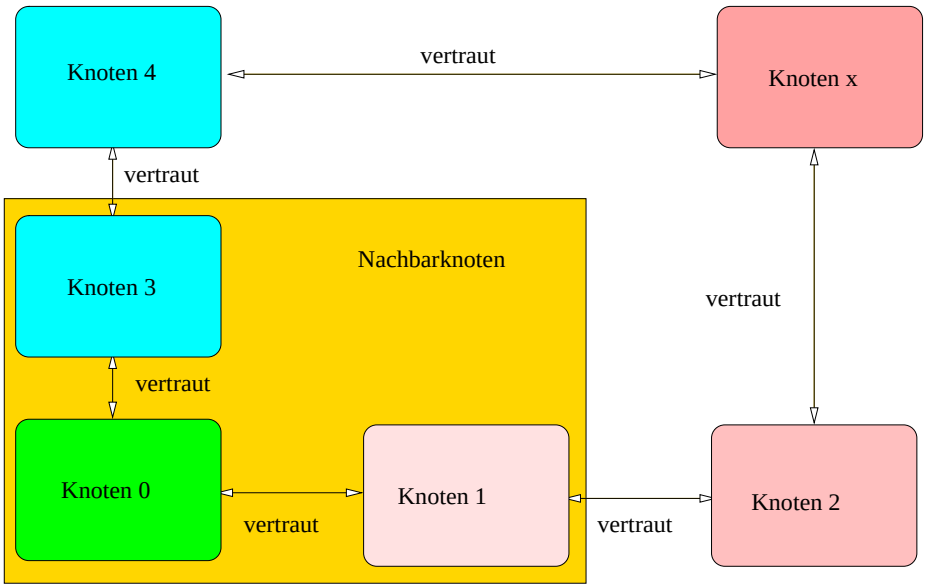

Abbildung 45: Vertrauenskette

Knoten. Ähnlich der Funktionsweise eines Darknet wird den Knoten vertraut, denen ein vertrauenswürdiger Knoten das Vertrauen ausgesprochen hat. In Abbildung 45 ist eine solche Vertrauenskette dargestellt. Erhält Knoten null von Knoten vier eine Anfrage, so kann er dessen Identität prüfen, indem er Knoten drei fragt. Dieser Knoten befindet sich innerhalb der Nachbarschaft von Knoten null. Knoten drei kennt aber wiederum Knoten vier und kann deshalb dessen Identität bestätigen. Diese Vorgehensweise hat gewisse Nachteile. Durch die Kette von vertrauenswürdigen Knoten bekommt die Einrichtung, welche die Signatur prüfen möchte, die Bestätigung von einem Knoten, dem sie bislang nicht vertraute. Knoten können zu vertrauenswürdigen Knoten werden, ohne dass der betreffende Knoten einen Einfluss auf die Prüfung der Identität hat. Somit ist die Vertrauenskette nur so zuverlässig wie der unzuverlässigste Knoten innerhalb dieser Kette. Es ist also darauf zu achten, dass alle Knoten bei der Überprüfung der Identität eines anderen Knotens die gleichen Standards zur Prüfung der Identität eines Knotens verwenden. Weiter sollte die Kette von Knoten, die sich vertrauen, nur eine bestimmte Länge haben. So kann die Wahrscheinlichkeit für Fehler innerhalb der Kette reduziert werden. Durch die Bestätigungen verschiedener Knoten kann die Sicherheit wiederum erhöht werden. Erkennt auch nur ein Knoten eine Fälschung, so wird dem Zertifikat nicht vertraut. Ein anderer Aspekt ist der Austausch der Schlüssel, wenn diese abgelaufen sind. Ein neuer Schlüssel muss wieder auf alle Knoten verteilt werden. Automatisch erlernte Schlüssel werden verworfen, sobald die Gültigkeit abgelaufen ist. Die Vorteile einer solchen verteilten Prüfung von Signaturen sind der Verzicht auf einen zentralen Knoten und damit einen Flaschenhals. Durch die Unabhängigkeit von zentralen Komponenten erhöht sich die Redundanz. Durch die Prüfung einer Signatur von zwei verschiedenen Knoten wird die Sicherheit erhöht. Ein Angreifer müsste somit zwei Knoten manipulieren, um ein Zertifikat zu fälschen.

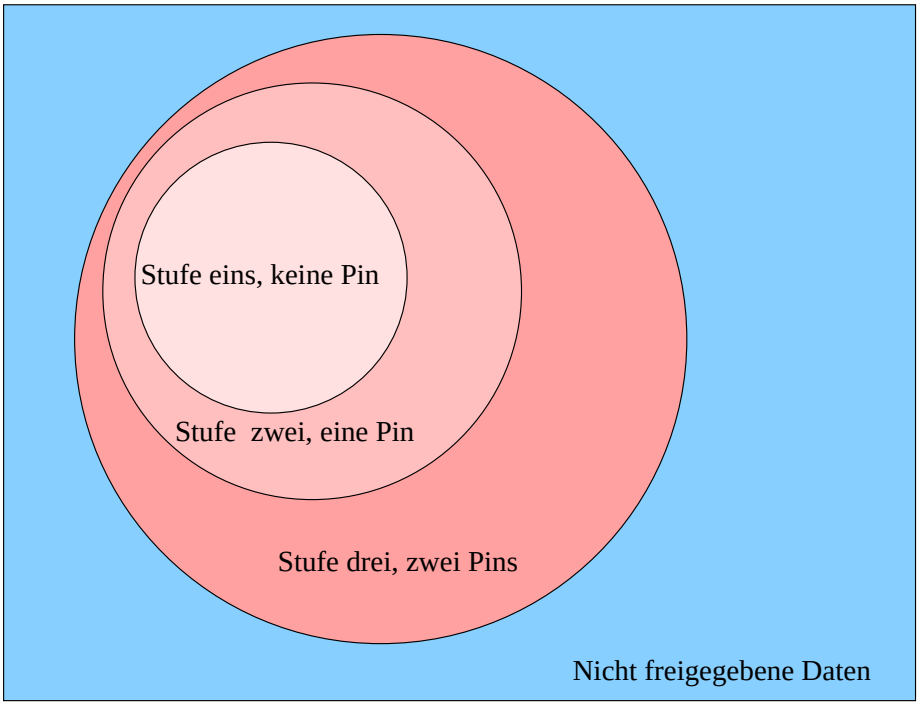

Abbildung 46: Verschiedene Autorisierungsstufen

6.9 Elektronische Unterschrift

Im Rahmen der EGK soll auch der elektronische Heilberufsausweis eingeführt werden. [Mar08] Nach Angaben des Statistischen Bundesamtes waren zum 31. Dezember 2007 rund 4.4 Millionen Menschen im Gesundheitswesen tätig. Das sind ca. zehn Prozent der Bevölkerung. Dazu zählen natürlich nicht nur Ärzte und Krankenschwestern. Es ist handelt sich dabei unter anderem auch um Apotheker, Rettungssanitäter, Medizintechniker, Therapeuten und Sozialarbeiter. Diesen Personenkreis mit Heilberufsausweisen und der entsprechenden Infrastruktur auszustatten erweist sich in der Praxis als äußerst schwierig. Vor allem dann, wenn eine zentrale Behörde die Karten ausgeben und die Daten verwalten muss, werden dabei möglicherweise Kapazitätsgrenzen erreicht. In bestimmten Fällen werden personifizierte elektronische Unterschriften allerdings dringend benötigt. Beispielsweise muss ein Rezept von einem Arzt unterschrieben werden. Dazu kann eine Einrichtung selbst für jeden Mitarbeiter ein Schlüsselpaar erzeugen oder gegebenenfalls Smartcards ausgeben. Erzeugt ein Mitarbeiter eine Signatur mit diesem Schlüssel, so kann ein anderer Knoten diese Signatur prüfen, indem er sich von der Einrichtung die Echtheit des Zertifikates bestätigen lässt. Ist diese Einrichtung eine vertrauenswürdige Instanz, kann sie somit als Trustcenter für den jeweiligen Mitarbeiter fungieren.

6.10 Autorisierungsstufen

Die Selbstbestimmung des Patienten steht beim Datenschutz im Gesundheitswesen in Vordergrund. Es ist nicht nur zu vermeiden, dass ein Teilnehmer Zugriff

Pat.	KH	Praxis	KK	Apo.	Pfl.	Behörde
Pat.	Pat.	-	-	-	-	-
Arzt	Arzt	Arzt	-	-	-	-
Pfl.	Pfl.	Pfl.	-	-	Pfl.	-
Apo.	Apo.	-	-	-	-	-
Man.	-	-	Man.	-	-	Man.
-	-	-	-	-	-	Beamter

Tabelle 4: Rollen die Institutionen einnehmen dürfen

auf Daten erlangt, ohne im Besitz der Gesundheitskarte zu sein. Wichtig ist außerdem, dass ein Patient selbst entscheiden kann, wer Zugriff auf welche Daten erlangen darf. Dafür werden sogenannte Autorisierungsstufen definiert. Alle Daten, die der ersten Autorisierungsstufe zugeordnet sind, können von allen Knoten gefunden werden, welche im Besitz der Gesundheitskarte sind. Daten, die unter die zweite Autorisierungsstufe fallen, können nur mit der Gesundheitskarte und einer speziellen Pin gefunden und abgerufen werden. Für die Daten der Stufe drei gilt das gleiche, jedoch sind zwei Pins nötig, um Daten zu finden. Die Zuordnung kann der Patient selbst treffen. Hatte er beispielsweise eine Lungenentzündung, so kann er diese Erkrankung der Stufe eins zuordnen. Damit kann jeder weiterbehandelnde Arzt die Daten abrufen. Muss sich der Patient in einer psychiatrischen Einrichtung behandeln lassen, so kann er diese Episode der Stufe zwei zuordnen. Ist der Patient der Meinung, sein Allgemeinarzt benötige für die Behandlung einer Grippe diese Informationen nicht, muss er diesen nicht mittels der Pin autorisieren. Der Arzt findet die Daten zu dieser Episode in diesem Fall gar nicht erst, da er ohne PIN nur Zugriff auf Daten der Stufe eins hat. Im anderen Fall kann der Patient den Arzt für den Zugriff auf Daten der Stufe zwei freischalten, indem er die Pin eingibt. Dieser Arzt hätte dann Zugriff auf Daten der Stufe zwei. Wird der Patient wegen Alkoholentzug behandelt, möchte er diese Informationen vielleicht stärker schützen als andere. Er kann diese Daten der Stufe drei zuordnen. Nun kann ein Arzt oder eine andere Einrichtung die Daten nur dann finden, wenn der Patient beide Pins eingegeben hat. Falls er das tut, hat ein Arzt Zugriff auf alle autorisierten Daten des Patienten. Wird eine Episode keiner dieser Autorisierungsstufen zugeordnet, so können die Daten von anderen Knoten nicht gefunden und auch nicht abgerufen werden. Abbildung 46 zeigt, wie die Autorisierungsstufe drei die beiden anderen beinhaltet, während die Stufe zwei nur Daten der Stufen zwei und eins sichtbar macht. Daten, die ohne Pin abgelegt sind, können allein mit Hilfe der Gesundheitskarte gefunden werden. Die Autorisierungsstufen werden realisiert, indem der MPI des Patienten mit den PIN-Nummern verknüpft und anschließend ein Hashwert davon gebildet wird. Die einzelnen Episoden werden unter diesem Hashwert abgelegt. Ohne den MPI des Patienten und die PIN-Nummern können diese Hashwerte nicht gebildet und die Daten dem Patienten nicht zugeordnet werden. Ein Finden der Episodeninformationen ist somit für unberechtigte Personen nicht möglich.

6.11 Rollen

Jede Einrichtung ist selbst für die Zugriffsmatrix der eigenen Mitarbeiter verantwortlich. [MNLea08] In den entsprechenden Informationssystemen werden Nutzer angelegt, welche bestimmte Berechtigungen besitzen. Wie in Abschnitt 6.8 schon erwähnt, tritt ein Kommunikationspartner immer als Einrichtung auf. Die Berechtigungen, welche die Nutzer in dem speziellen Kontext ihrer Einrichtung zugewiesen bekommen haben, können von anderen Knoten nicht überprüft werden. Die Kommunikation findet deshalb immer zwischen Einrichtungen statt. Jedoch kann jede Einrichtung für einen Kommunikationsvorgang verschiedene Rollen einnehmen. Findet eine Kommunikation zwischen zwei Krankenhäusern statt, bei der auf medizinische Daten zugegriffen werden kann, so nimmt das Krankenhaus beispielsweise die Rolle „Arzt“ ein. Wird der Prozess von einer Verwaltungskraft ausgelöst, tritt das eine Krankenhaus dem anderen Haus gegenüber als „Manager“ auf. Dementsprechend können nur Dienste einer bestimmten Klasse aufgerufen werden. Eine Einrichtung kann somit zu unterschiedlichen Zeiten ganz verschiedene Rollen einnehmen. Entsprechend unterschiedlich sind auch die Zugriffsrechte. Je nach Einrichtung sind bestimmte Rollen sinnvoll oder auch nicht. Eine Apotheke kann beispielsweise nie die Rolle eines Arztes einnehmen, denn in einer solchen Einrichtung wird kein Patient behandelt. Mit welcher Rolle eine Einrichtung auftritt wird bei dem Austausch des Schedule-Objektes festgelegt. Da innerhalb eines Schedules auch Dienste vorkommen können, welche verschiedenen Rollen zugeteilt sind, können auch mehrere Rollen gleichzeitig eingenommen werden. Der Zusammenhang zwischen den konkreten Nutzern und den Rollen, die ein Knoten einnehmen darf, muss von den Einrichtungen selbst hergestellt werden.

6.12 Delegation

In Kapitel 6.7 wurde dargelegt, dass für die Erlangung von Zugriffsrechten zwei Voraussetzungen nötig sind. Erstens muss ein Dienst eine entsprechende Rolle einnehmen, die es dem aufrufenden Knoten erlaubt, auf die Daten zuzugreifen. Zweitens muss eine Einrichtung die Beziehung zum Patienten nachweisen. Ein Krankenhaus muss beweisen können, dass der Patient gerade in ihr behandelt wird. Dieses Problem soll mit der Delegation von Rechten gelöst werden. [MS91] Die Elektronische Gesundheitskarte wird mit einem Chip ausgestattet sein, welcher für kryptographische Anwendungen verwendet werden kann. Diese Karte kann also Daten signieren. Auf diese Weise kann ein Knoten einem anderen beweisen, dass er im Besitz der Gesundheitskarte des Patienten ist und deshalb die zweite Bedingung für den Zugriff auf Daten erfüllt. Jedoch muss ein Knoten, auch nachdem der Patient die Einrichtung verlassen hat, noch auf Daten des Patienten zugreifen. Ein niedergelassener Arzt beispielsweise, welcher den Patienten in ein Krankenhaus eingewiesen hat, bekommt nach erfolgter Behandlung des Patienten einen Arztbrief und verschiedene andere Informationen vom Krankenhaus zugeschickt. Das ist für die Weiterbehandlung nötig, die wiederum Aufgabe des niedergelassenen Arztes ist. Zum Zeitpunkt, zu dem der Arzt nun die Daten aus dem System des Krankenhauses abruft, muss der Patient nicht anwesend sein. Der Arzt möchte sich möglicherweise vor der Vorstellung des Patienten in der Praxis bereits auf dessen Behandlung vorbereiten. Somit muss es möglich sein, dass ein Arzt, nachdem er im Besitz der Karte eines Patienten

war, eine gewisse Zeit auf dessen Daten zugreifen kann. Im Krankenhaus wäre es theoretisch möglich, dass ein Patient seine Gesundheitskarte bei der Anmeldung abgibt und wieder abholt, sobald er sich abmeldet. Übliche Praxis ist das jedoch derzeit nicht. Darüber hinaus müsste die Gesundheitskarte bei jedem Datenzugriff auf externe Datenquellen mit dem System verbunden sein. Das hieße, man müsste hunderte von Kartenlesegeräten vorhalten oder diverse Roboter einsetzen, welche die Karten bei Bedarf in ein Kartenlesegerät einführen. Solche Geräte müssten erst entwickelt werden. Demzufolge ist auch im Krankenhaus ein Arbeitsablauf zu gewährleisten, der nach einmaligem Einlesen der Karte Zugriff auf externe Patientendaten für einen bestimmten Zeitraum ermöglicht. Ein solches Verhalten wird mit der Delegation von Rechten erreicht. Beim Einlesen der Karte wird ein Zertifikat erzeugt, welches mit der Gesundheitskarte signiert ist. Dieses Zertifikat hat eine bestimmte Gültigkeitsdauer. Solange das Zertifikat gültig ist, kann ein Knoten damit beweisen, dass der Patient in einem bestimmten Zeitraum eine Behandlung in dieser Einrichtung bekommen hat. Ein Knoten, der Daten an einen anderen Knoten übermitteln soll, bekommt dieses Zertifikat, das von der Gesundheitskarte des Patienten signiert ist. Die Echtheit der Gesundheitskarte kann ein Knoten prüfen, da der Knoten den Patienten bereits behandelt und dessen Gesundheitskarte ausgelesen hat. Der öffentliche Schlüssel liegt einer solchen Einrichtung also bereits vor. Abbildung 47 zeigt die Voraussetzungen für einen Zugriff auf Patientendaten anderer Knoten. Das interne Identitätsmanagement des Knotens muss den aktuellen Nutzer autorisieren, eine bestimmte Rolle einzunehmen. Mit dem von der EGK signierten Zertifikat kann der Knoten dann beweisen, dass sich der Patient bei seiner Einrichtung in Behandlung befindet. Sind diese beiden Voraussetzungen gegeben, kann ein Datenaustausch stattfinden.

6.13 Protokollierung

In bestimmten Situationen ist es nicht sinnvoll, den Zugriff auf Informationen zu sehr zu reglementieren. Schwebt ein Patient in Lebensgefahr, so hat der Datenschutz für den Betreffenden in dieser Situation oft nicht die höchste Priorität. Es kann also in bestimmten Situationen durchaus im Interesse des Patienten sein, einem Arzt auch ohne genaue Überprüfung seiner Berechtigung Zugriff auf seine Gesundheitsdaten zu gewähren. Im Zweifelsfall sollte ein System also eher einige nicht unbedingt nötige Daten preisgeben, statt wichtige Informationen im Interesse des Datenschutzes zu verbergen. Jeder Dienstleister im Gesundheitswesen hat eine Verantwortung und Angestellte im Krankenhaus werden von ihrem Arbeitgeber zur Verschwiegenheit verpflichtet. Jeder ist dafür verantwortlich, dieser Pflicht nachzukommen. Um Personen dafür zu motivieren, ist es wichtig, dass eventuelle Verstöße und potentieller Missbrauch von Daten protokolliert werden, damit Verantworliche später zur Rechenschaft gezogen werden können. Jeder Knoten muss also genaustens protokollieren, wer in welcher Rolle und wann, welche Daten angefordert hat und ob diese Daten tatsächlich weitergeleitet wurden. Diese Protokolle müssen in Verbindung mit den Daten, auf welche zugegriffen wurde, gespeichert werden. Die Zugriffsprotokolle gehören sozusagen mit zu den medizinischen Daten selbst. Wie in Abschnitt 6.7 beschrieben, hat der Patient die Hoheit über die ihn betreffenden Daten. Er kann alle seine Daten abrufen. Diese Daten werden über ein Internetportal für einen Laien verständlich und übersichtlich bereitgestellt. [VPP05] Damit hat er auch Zugriff auf

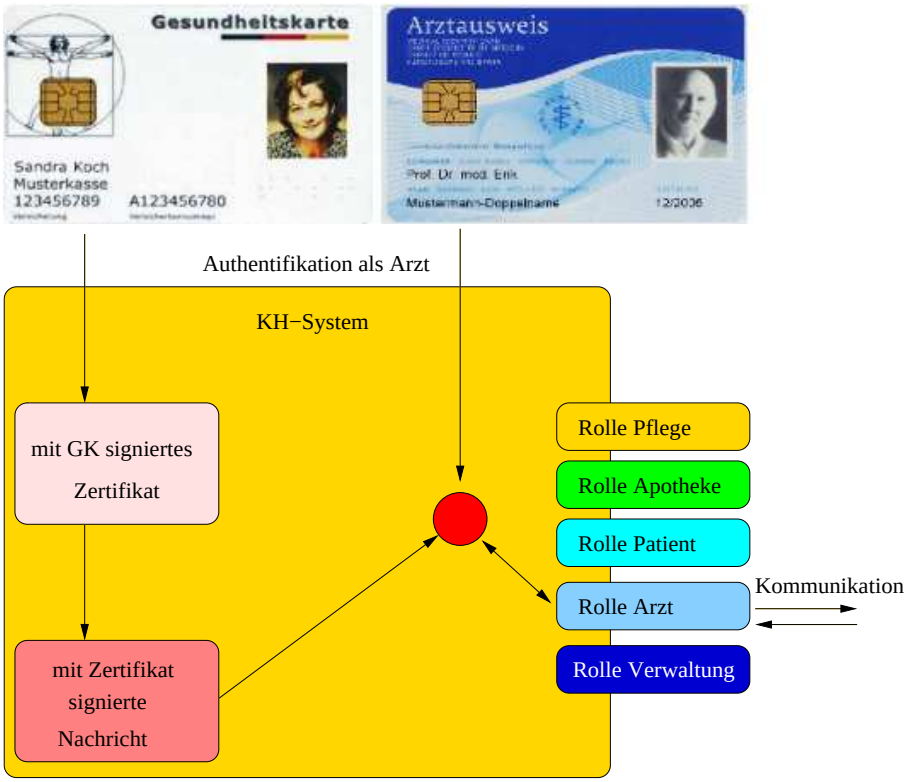

Abbildung 47: Delegation

die Zugriffsprotokolle. Nur so kann die Möglichkeit zur Entdeckung potentiellen Missbrauches von Daten gegeben werden. Entdeckt ein Patient, dass von einer Radiologischen Praxis Daten des Hautarztes abgerufen wurden, so kann der Patient nachfragen, wozu diese Daten benötigt wurden. Realisiert wird der Zugriff auf die Logdateien über eine Web Frontend. Wird Zugriff auf die Logs verlangt, so kann der Knoten eine Checksumme mit dem öffentlichen Schlüssel des Patienten verschlüsseln. Schickt der Kommunikationspartner die entschlüsselte Checksumme im Klartext über eine gesicherte Verbindung zurück, so wurde die Identität des Patienten festgestellt und die Sitzung bekommt Zugriff auf die Log-Einträge.

6.14 Zusammenfassung des Datensicherheitskonzeptes

In den vorigen Abschnitten wurde deutlich, dass die Sicherheit einer generischen Infrastruktur zum Austausch von Patientendaten unterschiedliche Technologien notwendig macht. Manche Aufgaben muss dabei die Anwendung selbst übernehmen. Andere Probleme lassen sich hingegen nur mit Hilfe zusätzlicher Werkzeuge lösen. So muss das Objekt Schedule sicher an einen anderen Knoten übertragen werden. Hierzu ist eine Public-Key-Infrastruktur (PKI) oder das Module Hsecurity notwendig. Im Schedule-Objekte sind unter anderem ein Schlüssel und zwei Zertifikate hinterlegt. Mit deren Hilfe kann die Anwendung selbst die Identität des Kommunikationspartners prüfen und entscheiden, ob dieser einen bestimmten Patienten behandelt. Außerdem kann die Anwendung nun mit Hilfe eines symmetrischen Kryptoalgorithmus und des Schlüssels, der mit dem Schedule gesendet wurde, die Basisobjekte chiffrieren. Dieser Schlüssel kann für jeden Schedule neu generiert werden, wodurch das Risiko eines erfolgreichen Angriffs sinkt. Zusätzlich sollte die Kommunikation über SSL oder WSS4R, wie in Kapitel 6.5 beschrieben wurde, gesichert werden. Die Sicherheit bei der P2P-Suche hat ganz andere Anforderungen. Eine Anfrage darf nicht verschlüsselt werden, da jeder Knoten diese benutzen muss, um zu entscheiden, ob er die gesuchten Daten besitzt. Natürlich muss der MPI des gesuchten Patienten mittels einer Einwegfunktion unkenntlich gemacht werden, sodass unbeteiligte Knoten den gesuchten Patienten nicht identifizieren können. Eine potentielle Antwort hingegen muss verschlüsselt werden, damit keine Rückschlüsse darauf gezogen werden können, ob ein Datensatz gefunden wurde. Das ist mit Hilfe eines asymmetrischen Kryptoverfahrens möglich. Eine PKI ist nicht notwendig, da lediglich sichergestellt werden muss, dass nur der Knoten die Antwort entschlüsselt, welcher die Anfrage gestellt hat. Abbildung 48 zeigt die Kommunikationsarten und die dazugehörige Verschlüsselung. Die Pfeile geben die Richtung an, in welche Daten versendet werden. An den Pfeilen ist jeweils notiert, ob diese Daten mit einem symmetrischen Algorithmus, einem asymmetrischen Algorithmus oder überhaupt nicht verschlüsselt werden müssen.

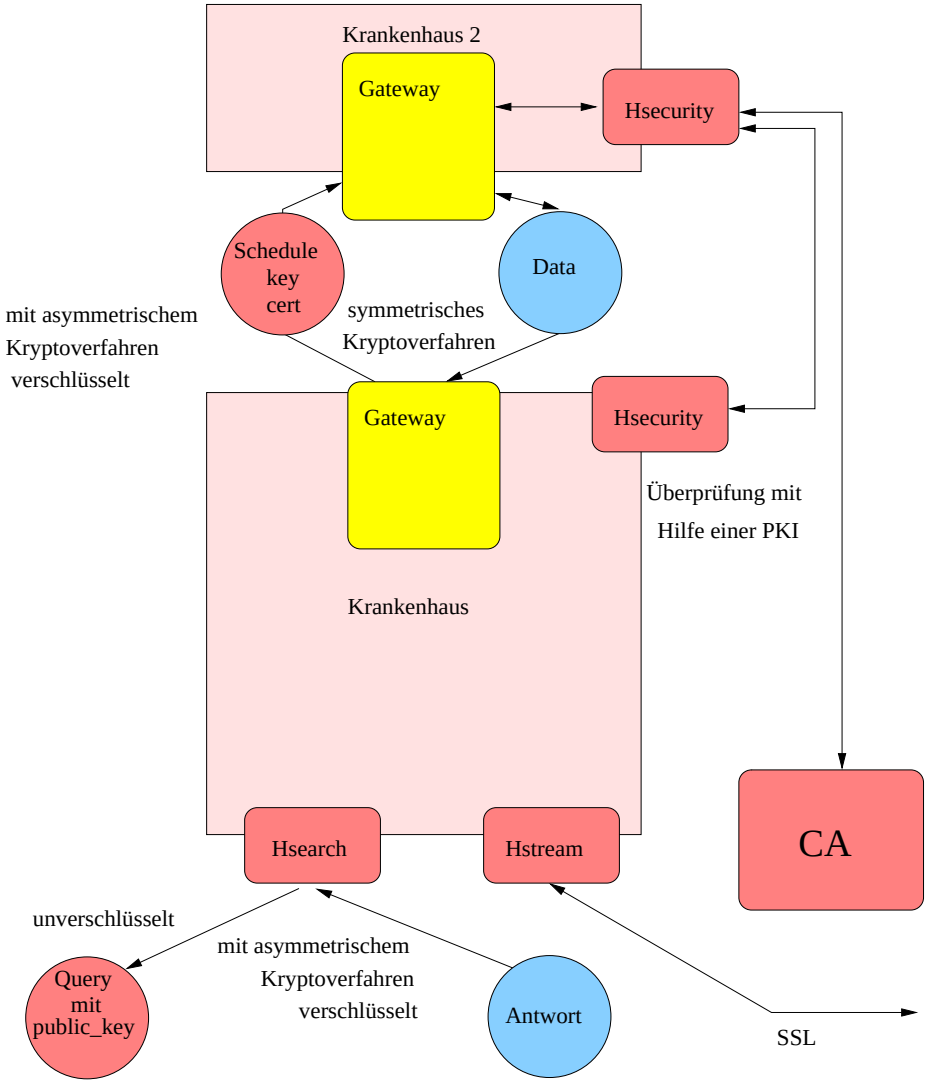

Abbildung 48: Übersicht über die Verschlüsselung der Kommunikation

7 Anbindung an vorhandene Systeme

Im Krankenhaus gibt es zur Zeit eine Vielzahl an Systemen, die durch Schnittstellen miteinander verbunden sind. Um die vorhandenen Systeme mittels einer Middleware mit anderen Einrichtungen zu verbinden, bedarf es einer Komponente, welche die Nachrichten der einzelnen Systeme an die Infrastrukturkomponenten weitergibt. Um die Komplexität der Kommunikation zwischen den verschiedenen Systemen zu verkleinern, werden vermehrt Integrationsserver eingesetzt. Alle Systeme bauen eine Verbindung zu diesem System auf. Der Integrationsserver verteilt nun alle eingehenden Nachrichten nach bestimmten Regeln an die verschiedenen Subsysteme. Viele Integrationsserver beherrschen HL7, DICOM und andere verbreitete Protokolle. Einige unterstützen auch Webservices. Ein solcher Integrationsserver muss als Adapter zwischen den vorhandenen Systemen und den Komponenten der Infrastruktur eingesetzt werden. Hierfür soll der im Rahmen dieser Arbeit entworfene Integrationsserver namens Monkey verwendet werden. Das Konzept, welches hinter diesem Integrationsserver steht, hat zum Ziel, beliebige Schnittstellen ohne zusätzlichen Implementationsaufwand abbilden zu können. Dazu werden die Schnittstellen mit Hilfe einer XML-Datei beschrieben. Diese Beschreibung beinhaltet die Reihenfolge der Felder, Trennzeichen zwischen Feldern und Segmenten, sowie mögliche Angaben zu Wiederholungen. Das XML-File, welches die Schnittstelle beschreibt, bildet so eine Baumstruktur. Um Schnittstellen möglichst allgemein zu beschreiben, arbeitet der Algorithmus von Monkey sequentiell. Die Nachricht wird von Feld zu Feld durchsucht, beginnend am Anfang der Nachricht. Der Baum der Schnittstellenbeschreibung wird parallel dazu durchlaufen und so jedes Feld dem richtigen Namen zugeordnet. Das hat den Vorteil, dass beliebige Trennzeichen verwendet werden können. Beispielsweise kann das Trennzeichen zwischen zwei Segmenten dasselbe sein wie innerhalb dieser Segmente. Dadurch lässt sich die Beschreibung einer Nachricht besser strukturieren. Monkey beherrscht derzeit beliebige TCP-Verbindungen, den Zugriff auf Dateien, das Versenden und Empfangen von E-Mails und den direkten Zugriff auf PostgresSQL-Datenbanken. Außerdem ist es möglich, Daten mittels XML-RPC zu versenden, was für die Kommunikation mit der Infrastruktur von Bedeutung ist. Auf diese Weise können HL7-Nachrichten in XML-Objekte umgewandelt und an den Knoten der Infrastruktur übergeben werden. Für das Versenden von Nachrichten über TCP/IP können Treiber eingebunden werden, welche die Kommunikation über verschiedene Protokolle wie Minimal Lower Layer Protocol (MLLP) oder MTS der Firma Siemens ermöglichen. Je nach Protokoll werden die Nachrichten dann in den passenden Protokollrahmen verpackt und an den Kommunikationspartner versendet. In Abbildung 49 sind die Komponenten dieses Integrationsservers visuell dargestellt.

7.1 Konfigurierbarkeit

Ein Motto von Ruby on Rails ist „Convention over Configuration“. Der Hintergrund dieses Satzes ist die Tatsache, dass viele Konfigurationsmöglichkeiten auch oft zu Konfigurationsfehlern und zu einer schwierigen Bedienung führen können. Konfigurationen können entfallen, wenn stattdessen Konventionen festgelegt werden. Diese Konventionen schränken dann zwar die Flexibilität einer Software ein, steigern aber die Bedienbarkeit. Die Entwicklung von Anwendun-

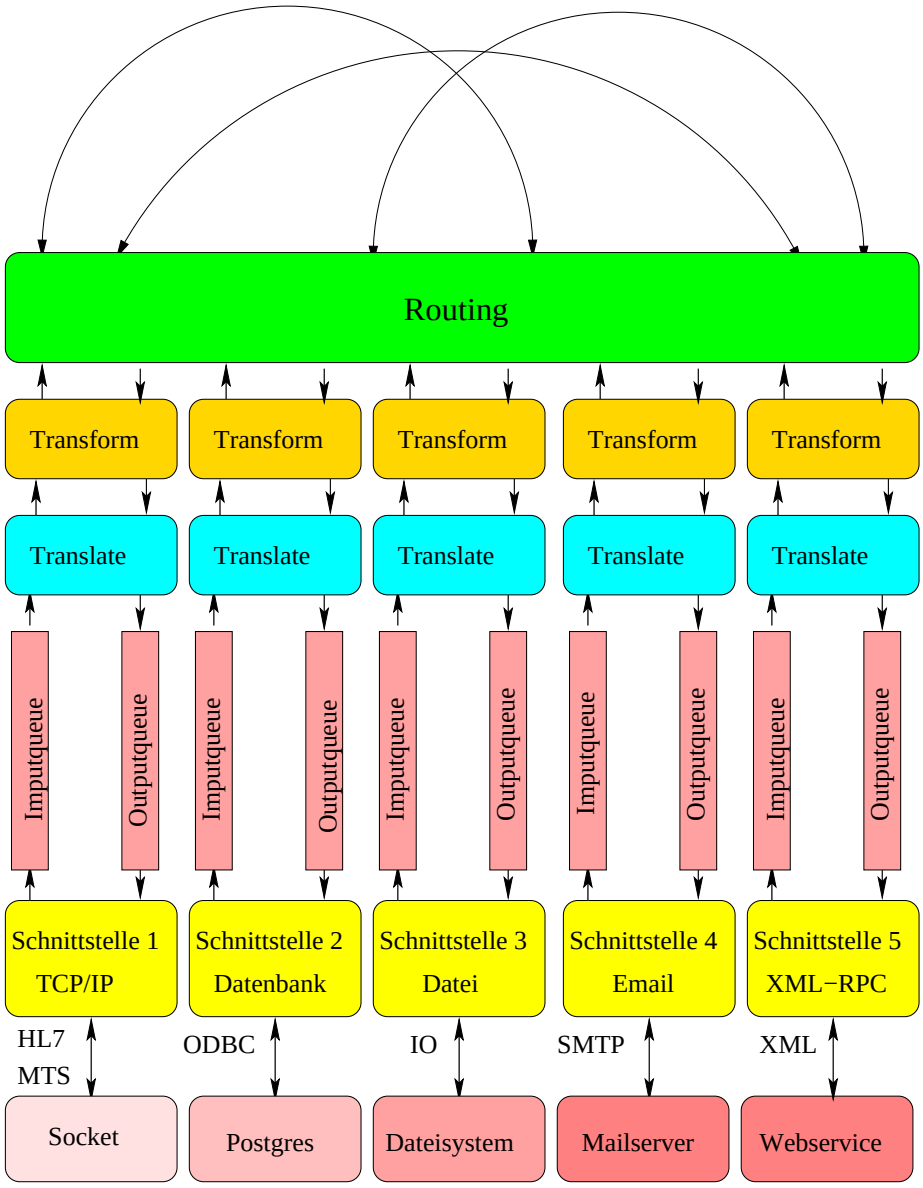

Abbildung 49: Softwarearchitektur des Integrationsservers Monkey

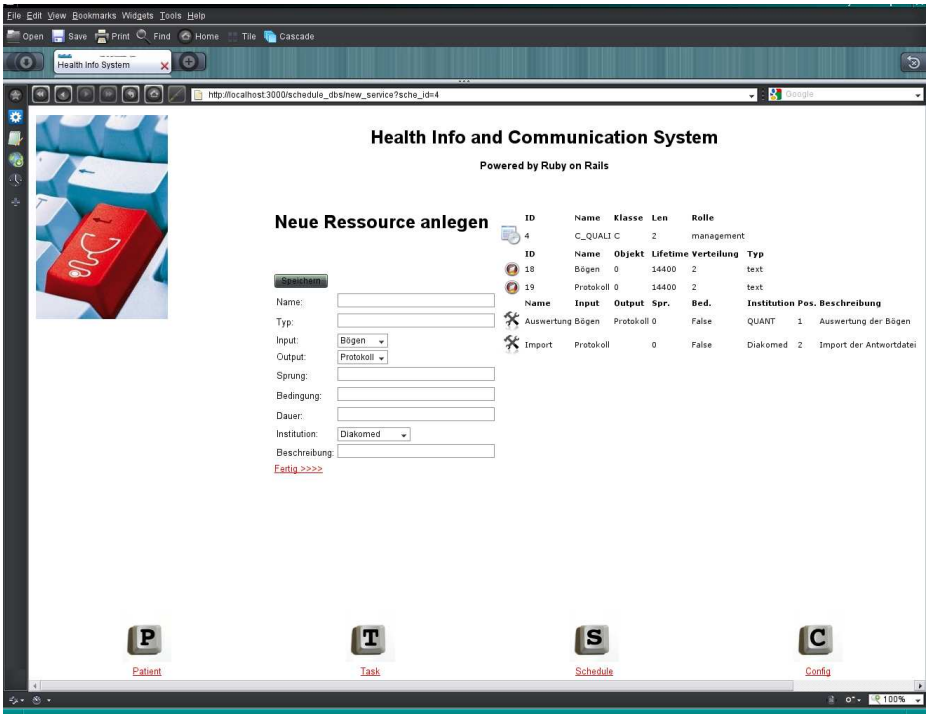

Abbildung 50: Screenshot Schedulekonfiguration

gen mit Hilfe von Serviceorientierter Architektur bietet zwar eine hohe Flexibilität, jedoch bedarf die Implementierung oftmals eines langen Reifeprozesses, bis die Anwendung für geschäftskritische Aufgaben eingesetzt werden kann. [SIVE08] Ziel muss es sein, ein Optimum aus einfacher Bedienung und für den konkreten Anwendungsfall hinreichender Flexibilität zu erreichen. Aus diesem Grund wurden in der Dienstarchitektur ganz konkrete Objekte definiert, die als Konvention verstanden werden können. Der Vorteil dabei ist, dass diese Objekte vom Nutzer nicht definiert werden müssen, sondern von vornherein jedem Knoten zur Verfügung stehen. Die Aufgabe des Nutzers besteht nun darin, mit Hilfe dieser vordefinierten Objekte seine Prozesse und Abläufe abzubilden. Zentrales Element dabei ist die Definition von Templates für Schedule-Objekte. Als Parallele zu diesen Objekten können Clinical Pathways, die auch Guidelines genannt werden, im Krankenhausbereich betrachtet werden. Bei diesen Guidelines hat sich herausgestellt, dass die effektivsten Pfade durch das in der entsprechenden Institution befindliche Personal erstellt wurden. [urb04]. Deshalb ist es sehr wichtig, dass medizinisches Personal ohne spezielle Vorkenntnisse der IT, in die Lage versetzt wird, sich Schedules selbst zu erstellen.

Da die Infrastruktur auf Basis von Webtechnologie entworfen wurde, dient ein Webinterface zur Konfiguration der Schedules. Die Erstellung des Templates eines neuen Schedules erfolgt über einen Wizard, bei dem der Benutzer drei Masken ausfüllen muss. In der ersten Maske werden die Daten des Schedule-Objektes, wie dessen Namen oder Serviceklasse eingegeben. Werden diese Daten gespeichert, gelangt der Nutzer in eine andere Maske, in welcher Ressourcen angelegt werden. Hier kann der Nutzer die Datenobjekte anlegen, die bei der

Verarbeitung des Schedules später benötigt werden. Es können hier beliebig viele Ressourcen angelegt werden. Hat der Nutzer alle Ressourcen erstellt, so kann er mit der Schaltfläche, die mit der Aufschrift „weiter “ versehen ist, in die nächste Maske gelangen. In dieser werden die Arbeitsschritte angelegt. Hier können die im vorhergehenden Schritt angelegten Ressourcen aus einem Drop-Down Menu ausgewählt und jeweils als Input oder Output des Arbeitsschrittes hinterlegt werden. Dabei können beliebig viele Arbeitsschritte zu einem Schedule angelegt werden, wie das auch bei den Ressourcen der Fall ist. Der Vorgang wird abgeschlossen, indem die Schaltfläche mit der Aufschrift „Fertig“ betätigt wird. Abbildung 50 zeigt die Maske zum Anlegen eines Arbeitsschrittes. Dabei wurde darauf geachtet, dass nur die Werte angefragt werden, die vom Benutzer unbedingt eingegeben werden müssen. Die Anzahl der Arbeitsschritte beispielsweise ergibt sich, genau wie die Position eines Service-Objektes, von selbst und muss nicht vom Nutzer eingegeben werden. Damit wird dem Prinzip „Convention over Configuration“ Rechnung getragen.

8 Implementierung und Validierung

Um die in dieser Arbeit vorgestellten Konzepte zu validieren, wurde eine Anwendung auf Basis von Ruby on Rails entwickelt. In diesem Kapitel wird die Implementierung dieser Anwendung vorgestellt und auf besonders interessante Problemlösungen eingegangen. Im ersten Abschnitt werden die Softwarekomponenten vorgestellt, welche für die Implementierung der Anwendung eingesetzt wurden. Rails ist nach dem Three-tiers Prinzip aufgebaut wie in Kapitel 2.3.1 erläutert wurde. Deshalb werden anschließend in drei Abschnitten diese Tiers vorgestellt. Es wird beschrieben, welche Aufgaben sie in der Beispielanwendung übernehmen. In Abschnitt 8.5 wird erläutert, wie es mit Rails auf einfache Art und Weise möglich ist, einen Applikationsserver zu realisieren. Der Abschnitt 8.6 befasst sich mit Testprogrammen, die für den Test der Anwendung entwickelt wurden. Im letzten Abschnitt wird auf die Implementierung des Testprogramms eingegangen, das für die Validierung des in dieser Arbeit entworfenen P2P-Algorithmus entwickelt wurde.

8.1 Softwarekomponenten

Die Anwendung wurde mit Hilfe des Web-Applikationsframeworks Ruby on Rails entwickelt, welches eine performante Entwicklung leistungsfähiger Software ermöglicht. Diese Framework unterstützt von Haus aus REST für die Implementation von Webservices. In Kapitel 3.2.1 wurde erläutert, weshalb XML-RPC und SOAP für die Kommunikation eingesetzt werden sollen. Um diese Protokolle mit Hilfe von Rails zu implementieren, bedarf es der Rails-Komponente ActionWebservice, die optional installiert werden kann. Rails bietet die Möglichkeit, unterschiedliche Webserver wie Apache für die Clientkommunikation zu verwenden. Da er für Testzwecke ausreichende Leistungsfähigkeit bietet, wurde der in Rails integrierte Webserver Webbrick benutzt. Rails arbeitet mit unterschiedlichen Datenbankmanagementsystemen zusammen. Ein sehr leistungsfähiges Open Source Datenbankmanagementsystem ist PostgresSQL. Diese Datenbank unterstützt SQL und Transaktionen und bietet sehr gute Möglichkeiten, einen Dump der Datenbank zu erstellen. Darüber hinaus kann man auch binäre Daten in einer Postgres-Datenbank ablegen. Das bietet der Anwendung die Möglichkeit, wirklich alle Daten in der Datenbank abzulegen. Aus diesen Gründen wurde eine Entscheidung für Postgres als Datenbankbackend getroffen.

8.2 Model

Rails unterstützt ein sogenanntes Objektrelationales Mapping wie in Kapitel 2.3.1 bereits ausgeführt wurde. Diese Funktionalität ermöglicht den Zugriff auf die Tabellen der Datenbank, ohne SQL-Befehle zu verwenden. Zu jeder Datenbank wird eine Klasse angelegt, welche von ActiveRecord::Base abgeleitet ist. Die Datenstruktur, welche diese Klasse benötigt, damit auf die Felder der Datenbank zugegriffen werden kann, wird aus der Datenbank importiert. Wird das Datenmodell geändert, so muss die entsprechende Klasse nicht angepasst werden. Das erleichtert den Umgang mit dem Datenmodell erheblich. Jedes Objekt einer Klasse repräsentiert eine Zeile in der dazugehörigen Datenbanktabelle. Da in Ruby Mehrfachvererbung nicht möglich ist, musste für jedes Basisobjekt

Objekt	Datenbankobjekt	WS-Objekt
Patient	Patient_db	Patient_oj
Episode	Episode_db	Episode_oj
Basics	Basics_db	Basics_oj
Diagnose	Diagnose_db	Diagnose_oj
Findings	Findings_db	Findings_oj
Text	Text_db	Text_oj
Data	Data_db	Data_oj
Value	Value_db	Value_oj
Money	Money_db	Money_oj
Edifact	Edifact_db	Edifact_oj
Date	Date_db	Date_oj
Order	Order_db	Order_oj
Job	Job_db	Job_oj
Prescription	Prescription_db	Prescription_oj
Institution	Institution_db	Institution_oj
Connection	Connection_db	Connection_oj
Schedule	Schedule_db	Schedule_oj
Service	Service_db	Service_oj
Resource	Resource_db	Resource_oj

Tabelle 5: Datenbank- und Webserviceobjekte der Basisobjekte

nicht nur eine Datenbanktabelle und die dazugehörige Klasse angelegt werden, sondern noch eine zweite Klasse, welche von ActionWebService::Struct abgeleitet ist. Tabelle 8.2 zeigt die Klassen, welche für die Basisobjekte angelegt wurden. Als Beispiel ist in Listing 14 und Listing 15 das Datenbankobjekt des Objektrelationalen Mappings und die Struktur des gleichen Objektes für die Kommunikation via Webservice dargestellt. Da Objekte dieser Klassen ineinander umgewandelt werden müssen, ist in den Klassen eine Methode definiert, die das gegensätzliche Objekt zurückliefert. Beispielhaft wurde eine Methode in die Klasse Date_OJ integriert, die ein Verschiebechiffre der Felder durchführt. Dieser Algorithmus ist natürlich nicht sicher und dient nur zur Demonstration. Wird ein Datenbankobjekt in ein Webserviceobjekt umgewandelt, so werden dessen Felder mit dem Schlüssel des Schedules, zu welchem das Objekt mittels einer Ressource gehört, verschlüsselt. In der Praxis muss diese Funktion dann durch einen hinreichend sicheren Algorithmus ersetzt werden.

Listing 14: Listing Date_DB

```
class DateDb < ActiveRecord::Base
 def validate
  if DateDb.find_by_id(id)
   errors.add(:name,"is␣alrady␣in␣db")
  end
 end
 def get_date_oj( key )
  d = Date_OJ.new
  d.oid = crypt( self.id.to_s , key )
  d.ord = crypt( self.ord.to_s , key )
  d.time =crypt( self.time.to_s , key )
```

```
  d.status =crypt( self.status.to_s , key )
  d.condition =crypt( self.condition.to_s , key )
  d.institution =crypt( self.institution.to_s , key )
  d.place =crypt( self.place.to_s , key )
  d.description =crypt( self.place.to_s , key )
  return d
 end
 def crypt( str , key)
   if key == ''
    return str
   end
   while key.length < str.length
    key = key << key
   end
   ret = ''
   i = 0
   str.each_byte do |c|
   k = c - (key[i])[0]
   if k < 0
     k = k + 256
   end
  end
  return ret
 end
end
```

Listing 15: Listing Date_OJ

```
class Date_OJ < ActionWebService::Struct
 member :oid, :string
 member :ord, :string
 member :time, :string
 member :status, :string
 member :condition, :string
 member :institution, :string
 member :place, :string
 member :description, :string

 def get_date_db( key )
  @dat = DateDb.new
   if DateDb.find(:all).length == 0
    @dat.id = 1
   else
    @dat.id =  DateDb.maximum('id') + 1

   end
   @dat.ord =  decrypt( self.ord , key )
   @dat.time = Time.parse(decrypt( self.time , key ))
   @dat.status =  decrypt( self.status , key )
   @dat.condition =  decrypt( self.condition , key )
   @dat.institution = decrypt( self.institution , key )
   @dat.place = decrypt( self.place , key )
   @dat.description = decrypt( self.description , key )
   @dat.save
   return @dat
 end

 def decrypt( str , key )
    if key == ''
```

```
      return str
     end
     while key.length < str.length
       key = key << key
      end
      ret = ''
      i = 0
      str.each_byte do |c|
      k = c + (key[i])[0]
      if k > 255
       k = k - 256
      end
      ret = ret << k.chr
     end
    return ret
 end
end
```

8.3 Controller

Im Controller steckt die eigentliche Programmlogik von Rails. Wird über den Webserver ein Dokument unter einer bestimmten URL angefordert, so wird zuerst der Controller aufgerufen. Dabei handelt es sich um ein Rubyscript. Dieses führt dann alle wichtigen Funktionen aus, um anschließend der View-Komponente Listen und Variablen für die Darstellung der Ergebnisse zu hinterlassen. In Listing 16 ist eine Funktion des Cron-Controllers der Anwendung dargestellt, der die wichtigste Funktionalität beinhaltet. Wird die Methode „index“ aufgerufen, so wird von dem Controller geprüft, welche Schedules sich gerade in Bearbeitung befinden und welcher der Arbeitsschritte als nächstes von welchem Knoten erledigt werden muss. Außerdem prüft dieser Controller, ob Ressourcen mit einem Objekt verknüpft wurden, die sofort an einen anderen Knoten übertragen werden müssen. Die Anwendung besitzt für jedes Basisobjekt einen Controller. Darin sind Funktionen zum Speichern und Ändern sowie die Webservices definiert, mit deren Hilfe die Objekte übertragen werden können.

Listing 16: Ausschnitt aus dem Cron-Controllers

```
require "wss4r/aws/utils"
require "activerecordresolver"
class CronController < ApplicationController
 def index
  @schedule_dbs = ScheduleDb.find_all_by_template('f')
  @re = ''
  @schedule_dbs.each do |s|
   @resource_dbs = ResourceDb.find_all_by_schedule(s.id)
   if s.step > s.len
    @resource_dbs.each do |rec|
    rec.destroy
   end
   @service_dbs = ServiceDb.find_all_by_schedule(s.id)
   @service_dbs.each do |ser|
   t = Task.find_by_service(ser.id)
   if t != nil
    t.destroy
   end
   ser.destroy
```

```
  end
  @pts = PatientSchedule.find_by_schedule(s.id)
  if @pts != nil
   @pts.destroy
  end
   s.destroy
  else
   @resource_dbs.each do |rec|
  if rec.presense == 2 and not rec.object == 0\
  and not Transmit.find_by_resource(rec.id).sendet:
   @r = broadcast(rec , s )
   @re = @re + s.key + '␣' + s.name + '␣' + s.id.to_s
  end
  end
  if s.active == true
   @ret = check( s )
  end
 end
end
end
```

8.4 View

Terminiert der Controller, so wird die dazugehörige View-Komponente aufgerufen. Diese besteht aus einem RHTML-Dokument. Dabei handelt es sich um in HTML eingebetteten Rubycode. Durch die Verarbeitung dieses Codes entsteht ein HTML-Dokument, welches an den Client übertragen wird. Listing 17 zeigt die View-Komponente des im vorigen Abschnitt beschriebenen Cron-Controllers.

Listing 17: Listing RHTML-Dokument

```
<h1>Shedule</h1>
<table>
 <tr>
  <td><%=s.name%></td>
  <th align=right>Aktive</th>
  <th align=right>Fortschritt</th>
</tr>
<% @schedule_dbs.each do |c|  %>
<tr>
 <td><%=c.name%></td>
 <%if c.active%>
  <td border=1><img  src=/Images/active.gif height=10px></td
     >
 <%else%>
  <td border=1><img  src=/Images/notactive.gif height=10px><
     /td>
 <%end%>
  <td width=100 bgcolor=grey>
  <img  src=/Images/bar_green.gif height=10px \
  width=<%=100*ic.step/c.len%>px></td>
  </tr>
 <%end%>
</table>
```

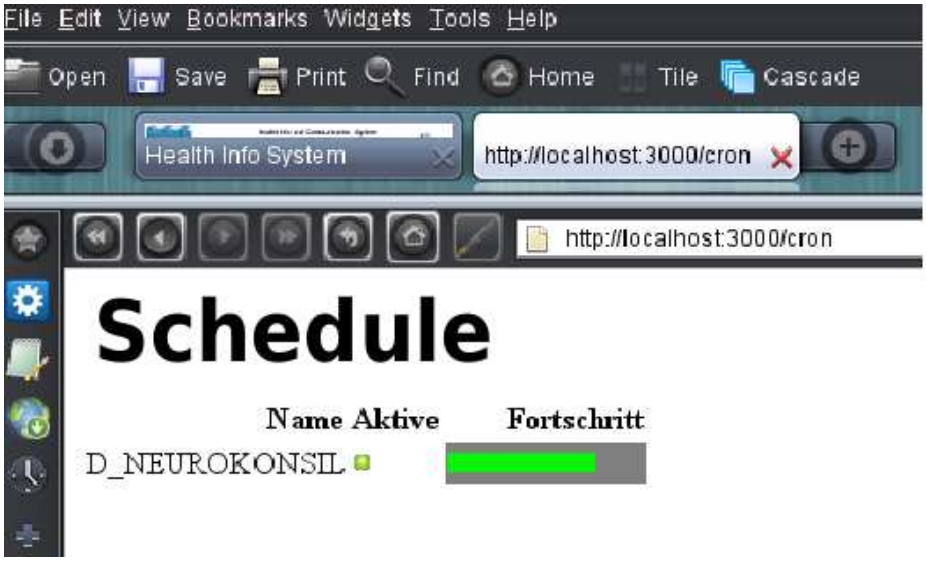

Abbildung 51: Darstellung des Status der aktuellen Schedules

Sie liefert eine Übersicht aller Schedules und zeigt an, wie weit deren Abarbeitung vorangeschritten ist. Abbildung 51 zeigt, wie der Browser das aus RHTML generierte HTML-Dokument darstellt. Die View-Komponente enthält nicht alle Elemente eines HTML-Dokumentes. Der Grund dafür ist, dass die Seite, die der Client erhält, aus einem sogenannten Layout, das im Controller definiert wird, und der Ausgabe der View-Komponente zusammengesetzt wird. Dadurch ist es möglich, der Anwendung ein einheitliches Aussehen zu verleihen. Rails erlaubt es in die RHTML-Dokumente alle Elemente einzubinden, die auch in HTML verwendet werden können. Außerdem wurden sogenannte Helper-Klassen in die RHTML-Dokumente der Anwendung eingebunden. Mit ihrer Hilfe können Formulare erstellt, Javascript-Programme eingebunden oder Stylesheets verwendet werden. Mit Hilfe der Helper-Klassen wurden beispielsweise Javascript-Funktionen integriert, die es erlauben, Patientendaten in Form eines Baumes darzustellen und daraus Elemente auszuwählen. Das ist in Abbildung 52 als Screenshot zu sehen.

8.5 Cronjob

Ruby on Rails ist kein vollwertiger Applikationsserver. Immer dann, wenn ein Client eine URL vom Webserver anfordert, wird der Controller und damit eine Aktion ausgelöst. Rails sieht jedoch nicht vor, einen Controller automatisch regelmäßig aufzurufen. Für die Anwendung ist das jedoch notwendig, denn diese muss den Status der Schedules prüfen und gegebenenfalls selbst Aktionen auslösen. Für die Lösung dieses Problems gibt es mehrere Ansätze. Einer besteht darin, die Methode des Cron-Controllers bei jeder beliebigen Interaktion aufzurufen. Das führt jedoch zu schlechten Antwortzeiten der Anwendung. Falls mehrere Anfragen parallel empfangen werden, kann es darüber hinaus zu ungewollten Effekten kommen. Der Controller wird dann mehrfach gestartet und jede Instanz greift auf die Schedules zu, was zu unvorhersehbaren Zuständen

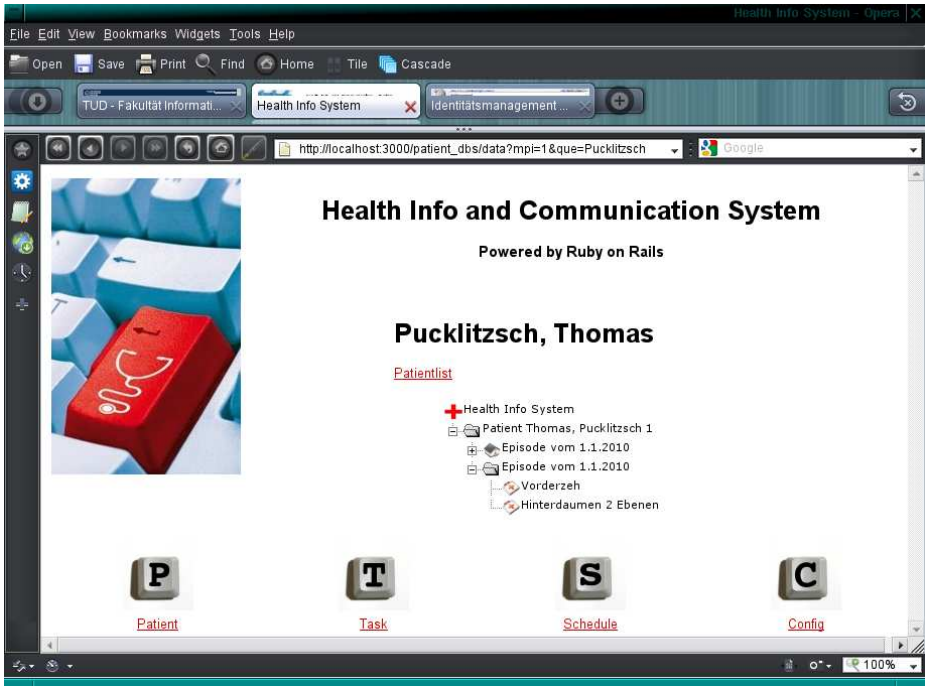

Abbildung 52: Auswahl von Patientendaten mittels Javascript-Baum

führen kann. Deshalb wurde eine andere Lösung gefunden. In Listing 18 ist ein Shellscript abgebildet, welches durch die Startscripte der Anwendung, als Prozess im Hintergrund gestartet wird. Dieses prüft, ob auf dem Rechner, auf welchem es läuft, ein Firefox-Browser läuft. Ist dies der Fall, so wird der Browser ferngesteuert dazu veranlasst, die URL des Cron-Controllers aufzurufen. Im anderen Fall wird diese Seite mit Hilfe des Shell-Befehles wget geladen. Dadurch wird der Cron-Controller aufgerufen und die regelmäßig notwendigen Routinen werden gestartet. Das Script führt diesen Vorgang aller zwei Sekunden solange aus, bis es durch die Stopscripte beendet wird.

Listing 18: Shellscript zum Aufruf des Cron-Controllers

```
#!/bin/bash
path=../
sleep=2
while [ 1 ]
do
firefox=$(ps -ef | grep firefox | \
sed -n '/[0-9][0-9][ ]\/usr\/lib\/iceweasel\/firefox-bin/p')
test -z "$firefox
if␣[␣$?=1␣]
then
firefox␣-remote␣"openurl(http://localhost:3000/cron)"
else
wget␣-O␣$path/hstatus.html␣http://localhost:3000/cron
fi
sleep␣$sleep
done
```

8.6 Dummys

Um die Anwendung zu testen, muss eine Kommunikation mit anderen Knoten und mit Modulen stattfinden. Damit dies möglich ist, wurden Dummyprogramme entwickelt. Sie simulieren das Verhalten anderer Knoten oder Module. Mit dieser Methode wurde der Schedule D_KONSI getestet. Ein in Ruby implementiertes Script simuliert einen anderen Knoten, an welchen Röntgenbilder und Labordaten geschickt werden. Dieses Script antwortet dann mit einem Termin und einem Connection-Objekt. Das wird wiederum an einen Dummy namens Hstream übergeben. Dieser simuliert den Aufruf eines Moduls, mit dessen Hilfe eine Videokonferenz gestartet werden kann. Sobald die letzte Task in der Anwendung bearbeitet ist, wird Hstream das Connection-Objekt über XML-RPC geschickt, und es öffnet sich ein Fenster, welches die Verbindung zu einem entfernten Arzt symbolisiert.

8.7 Hsearch

Eines der Dummyprogramme, welches für den Test der Anwendung entwickelt wurde, ist Hsearch. Es simuliert ein Modul, an welches die Rails-Anwendung eine Suchanfrage stellen kann. Diese Software wurde zu einer Testumgebung des in Kapitel 5.5 beschriebenen Routingalgorithmus weiterentwickelt. Die Software wurde in Python implementiert, da diese Scriptsprache sehr performant ist. Beim Start der Anwendung werden eine große Menge Objekte angelegt, welche Knoten in einem P2P-Netzwerk simulieren. Dies wurde bereits ausführlich in Kapitel 5.5 beschrieben. In diese Objekte können nun andere Objekte eingefügt werden, die Patientendaten simulieren. Nun kann eine Suche nach diesen Patienten mit Hilfe des Routingalgorithmus durchgeführt werden. Die besondere Schwierigkeit bei der Implementierung bestand darin, dass die Suche durch einen rekursiven Aufruf der Funktion query() realisiert wurde. Dies hat zur Folge, dass der Routingalgorithmus die Funktion query() erst dann abgeschlossen hat, wenn alle rekursiven Aufrufe der Funktion beendet waren. Das führte dazu, dass sich - während des Aufrufes dieses einen Befehls des Routingalgorithmus - die Umgebung des Knotens ändern konnte. Dadurch kam es zu unerwünschten Effekten, die nicht mit den Vorgängen in einer realen P2P-Struktur übereinstimmten. Deshalb mussten alle Datenstrukturen die global in der Klasse Peer definiert waren, kopiert werden, und in einer nur in der Methode query() gültigen Struktur abgelegt werden. Dadurch wurde es möglich, das Verhalten eines P2P-Netzwerkes zu simulieren und die Ergebnisse aus Kapitel 5.9 zu erhalten.

8.8 Abbildung der Architektur auf konkrete Szenarien

In der vorliegenden Arbeit wurde ein Konzept erarbeitet, das eine Kommunikation zwischen gleichberechtigten Knoten innerhalb des Gesundheitswesens ermöglicht ohne die Notwendigkeit zentraler Serverkomponenten, auf denen Daten oder Indices abgelegt sind, oder die eine Steuerung von Abläufen durchführen. Dazu wurden zwei wichtige Konzepte eingeführt. Zum einen das Konzept der Schedules und zum anderen ein P2P-Algorithmus, der eine Suche nach Patientendaten ermöglicht. In diesem Kapitel werden zwei Szenarien skizziert, welche die Anwendungsmöglichkeiten dieser Konzepte anhand praktischer Beispiele verdeutlichen.

8.9 Szenario 1

Im ersten Szenario wird der Patient vom Hausarzt mittels des Schedules B_REMITTANCE an das Krankenhaus überwiesen. Dabei werden die medizinischen Grunddaten des Patienten, eine Diagnose des Arztes und ein Auftrag an das Krankenhaus übertragen. Der Patient bekommt vom Hausarzt den Termin mitgeteilt, den das Krankenhaus für dessen Aufnahme vergeben und an den Arzt gesendet hat. Ist der Patient im Krankenhaus angekommen, wird der Schedule A_PAYROLL an die Krankenkasse geschickt und sofort die Aufnahme des Patienten gemeldet. Anschließend wird der Patient verschiedenen Untersuchungen unterzogen. Eine MRT-Untersuchung muss von einer radiologischen Praxis durchgeführt werden, da das Krankenhaus kein eigenes MRT besitzt. Dazu wird der Schedule D_MRT verwendet, der den Krankentransportdienst mit integriert. Der Patient bekommt einen Barcode, wird vom Krankentransport abgeholt und zur Untersuchung transportiert. Die Mitarbeiter des Krankenhauses können in ihrer Anwendung den Status des Schedules erkennen und daraus ableiten, wo sich der Patient gerade befindet. Ist die Untersuchung beendet, werden die Untersuchungsdaten an das Krankenhaus via Webservice übertragen und anhand der ID des Schedules dem richtigen Patienten im Krankenhaus zugeordnet. Anschließend soll der Patient entlassen werden. Zuvor wird mittels des Schedules D_WRITING eine DDS-Datei zu einem Schreibbüro gesendet und der geschriebene Text an das Krankenhaus übermittelt. Ist der Patient entlassen, so wird die Entlassungsmeldung und die Rechnung mittels des bereits gestarteten Schedules A_PAYROLL an die Krankenkasse gesendet. Diese schickt dann die Kostenübernahmeerklärung an das Krankenhaus zurück. Der Hausarzt kann mit Hilfe des P2P-Modules die Episode des Patienten lokalisieren und mittels des Schedule B_GET_DATA dessen MRT-Bilder herunterladen. Daraufhin stellt er dem Patienten ein Rezept aus, das elektronisch mittels B_SEND_PRESCRIPTION an die Krankenkasse gesendet wird, wo es für eine spätere Abholung hinterlegt wird. Die Apotheke, die der Patient wählt, holt sich das Rezept mit Hilfe des Dienstes B_GET_PRESCRIPTION und rechnet anschließend mit der Krankenkasse ab. Abbildung 53 zeigt diese Vorgänge schematisch. Die Pfeile zeigen die Richtung an, in welcher die Scheduleobjekte versendet werden. Die römischen Ziffern zeigen die Reihenfolge.

8.10 Szenario 2

In Szenario zwei soll mit Hilfe der Infrastruktur ein möglicher Ablauf bei der Behandlung von Schlaganfallpatienten abgebildet werden. Diese Behandlung ist sehr zeitkritisch. Es dürfen nur wenige Stunden vergehen, bis gezielte Maßnahmen eingeleitet werden. Zuerst wird der Patient einer Untersuchung mit Hilfe eines Computertomographen unterzogen. Anschließend wird eine Laboruntersuchung in einem Fremdlabor durchgeführt. Dazu wird der Schedule D_LAB an das Labor geschickt und parallel dazu eine Blutprobe des Patienten, versehen mit dem Barcode der Schedule-ID. Anhand dieser ID kann das Value-Objekt, welches das Labor zurückschickt, dem Patienten zugeordnet werden. Die Daten der CT-Untersuchung werden mit Hilfe des Schedules D_REND an eine andere Klinik gesendet, welche die Bilder mit Hilfe einer Software bearbeitet und die veränderten Bilder im Rahmen des Schedules D_REND an den Auftraggeber zurücksendet. Diese Bilder werden dann im PACS des Krankenhauses als

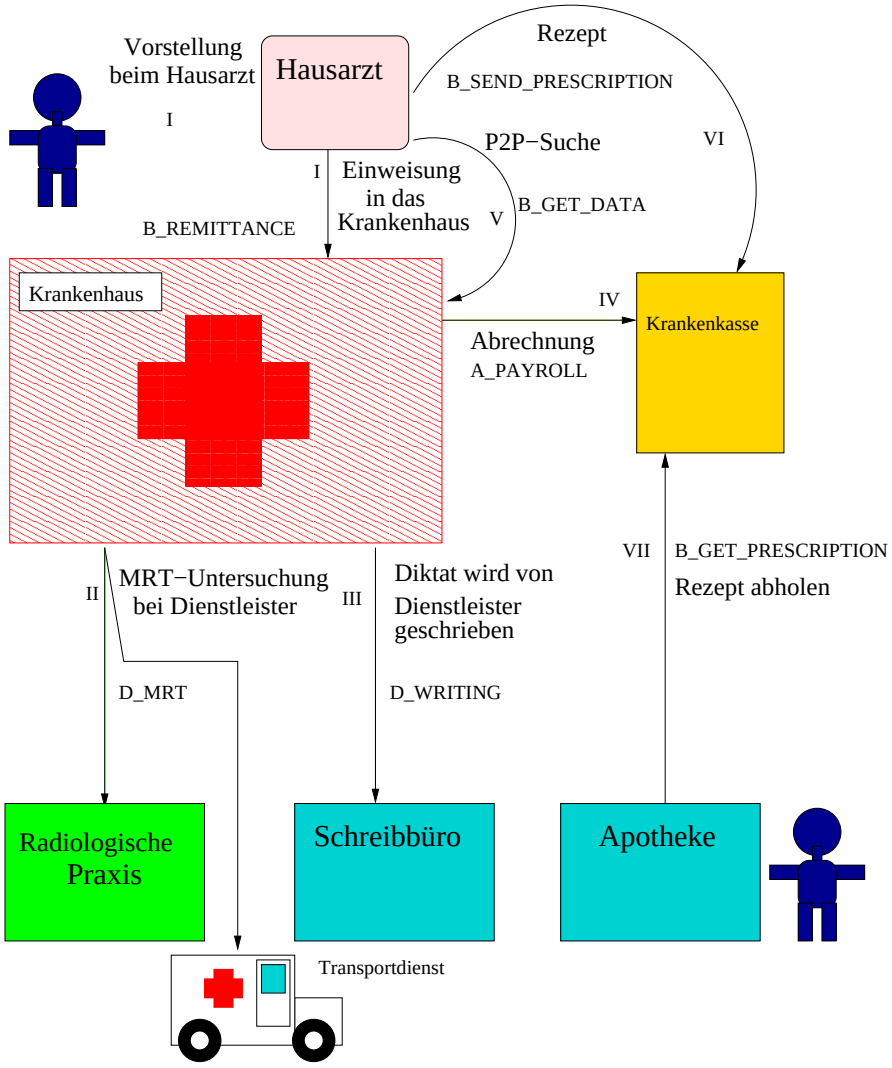

Abbildung 53: Szenario 1

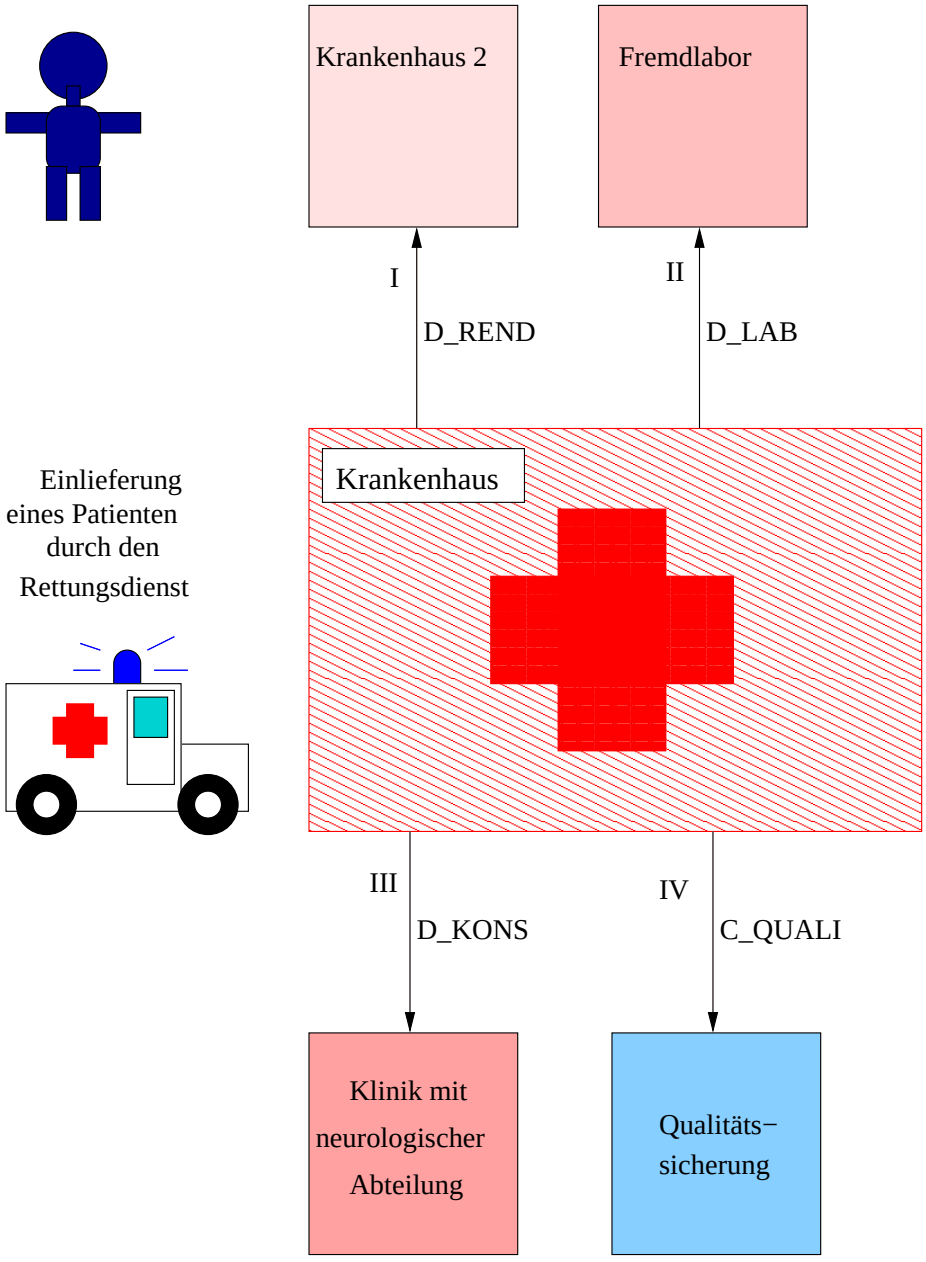

Abbildung 54: Szenario 2

sogenannte Secondary-Capture-Objekte, also als Kopie der eigentlichen Bilder, abgelegt. Bei der Behandlung von Schlaganfallpatienten in einer dafür speziell ausgerüsteten Abteilung namens Stroke Unit muss eine Betreuung durch einen Neurologen stattfinden. Krankenhäuser ohne eine neurologische Abteilung können eine sogenannte Stroke Unit Light einrichten. Dafür muss die Einrichtung jedoch mit einer Klinik für Neurologie eines anderen Krankenhauses zusammenarbeiten. Um dies zu ermöglichen, soll der Schedule D_KONS verwendet werden. Dieser sendet die Bilder der CT-Untersuchung sowie die Laborwerte des Patienten an den Neurologen. Der vergibt einen Termin, an welchem die Konsultation stattfinden kann und sendet außerdem ein Objekt vom Typ „Connection“, welches die Zugangsdaten für eine Videokonferenz beinhalten. Dieses Objekt kann dann an ein Modul übergeben werden, das eine Videoverbindung aufbaut. Dabei können sich die Ärzte vor Ort mit dem Neurologen austauschen und anhand der Untersuchungsdaten, die auf beiden Seiten der Verbindung vorliegen, eine Behandlungsstrategie entwickeln. Der letzte Schritt in Szenario zwei ist das Erstellen und Versenden eines Qualitätssicherungsbogens. Bei bestimmten Erkrankungen müssen solche Bögen ausgefüllt werden. Welche Krankheiten das betrifft, kann sich innerhalb eines Jahres mehrmals ändern. In Szenario zwei soll angenommen werden, dass ein solcher Bogen ausgefüllt und an die zuständige Stelle geschickt werden muss. Dieser Vorgang kann mit Hilfe des Dienstes C_QUALI abgebildet werden. Eine zentrale Stelle ist für alle Bögen zuständig. Deshalb kann hier ein statischer Schedule verwendet werden. Um den Vorgang zu automatisieren, wird ein Verzeichniss angelegt, in welchem die Bögen abgelegt werden. Mit Hilfe des Integrationsservers Monkey werden die Daten in ein Objekt vom Typ „text “ eingefügt und an das Gateway geschickt. Im Gateway wird ein CallBack eingerichtet, der nach einer Änderung am Datenmodell prüft, ob neue Bögen zum Senden bereitstehen. Ist das der Fall, wird der Schedule C_QUALI an die verantwortliche Stelle geschickt, welche die Daten bearbeitet und anschließend wird der Schedule aktiviert. Über den Schedule erhält das Krankenhaus eine Antwortdatei, die wiederum mittels des Integrationsservers in ein Verzeichnis abgelegt werden kann. Von dort aus kann diese Datei dann eingelesen werden. In Abbildung 54 ist die Abfolge der Schedules visualisiert. Die Symbolik entspricht der aus Abbildung 53.

9 Zusammenfassung

9.1 Schlussfolgerung

Durch seine Inhomogenität und die Sensibilität der Patientendaten gestaltet sich der Entwurf einer übergreifenden Dienstarchitektur für das Gesundheitswesen sehr schwierig. Das ist der Grund, weshalb sich bislang lediglich vereinzelte Lösungen wie D2D, welches in Kapitel 2.1.1 vorgestellt wurde, im produktiven Einsatz befinden. Deshalb haben Regierungen einiger Länder, zu denen auch Deutschland gehört, IT-Großprojekte gestartet, im Zuge derer das Gesundheitswesen modernisiert und eine Infrastruktur für die Kommunikation und die Übertragung von Gesundheitsdaten entwickelt werden soll. Die Architektur dieser Projekte benötigt zentrale Komponenten, die diese Infrastruktur verwalten und Daten speichern, die dann von anderen Knoten abgerufen werden können.

In dieser Arbeit wurde ein Konzept vorgestellt, welches eine Kommunikation von Knoten im Gesundheitswesen ohne zentrale Komponenten ermöglicht. Um dieses Ziel zu erreichen, wurden in den ersten Kapiteln zahlreiche Middleware-Produkte untersucht, die für eine solche Infrastruktur als Grundlagen dienen könnten. Angefangen von Toolkits aus dem Grid-Computing über Web-Applications-Frameworks bis hin zu Peer to Peer Technologien wurden die Vor- und Nachteile analysiert. Das Ergebnis zeigt, dass eine solche Infrastruktur zuerst eine Kommunikationsebene benötigt, die zur Steuerung der Anwendung verwendet wird. Diese Kommunikationsebene wurde mit Ruby on Rails implementiert und arbeitet mit Webservices und dem SOAP-Protokoll. Es wurde das Konzept der Schedules entwickelt, welches ein einfaches, aber für den Anwendungsfall ausreichendes System zur Steuerung von Abläufen darstellt. Bei genauerer Betrachtung wurde jedoch klar, dass im Gesundheitswesen auch viele Anwendungsfälle denkbar sind, die mit dieser Technologie nur unzureichend abgebildet werden können. Aus diesem Grund wurde die Anwendung so entworfen, dass Module eingebunden werden können, die Aufgaben erfüllen, für welche sich Webservices nicht eignen. Mit diesen Modulen wird wiederum über Webservices und XML-RPC kommuniziert. Da das System ohne zentrale Komponenten auskommen soll, ist eine Lösung für das Problem der Suche nach Patientendaten nötig. An dieser Stelle wurde das Konzept der Module angewandt und eine P2P-Komponente entworfen. Diese führt eine Suche durch, indem sie andere Knoten nach Daten des Patienten durchsucht und Anfragen nach einem Routingalgorithmus weiterleitet. Dazu wurden Ideen unterschiedlicher Algorithmen wie beispielsweise Freenet aus Kapitel 2.5.1 adaptiert und an die Erfordernisse im Gesundheitswesen angepasst.

Der Routingalgorithmus wurde so entworfen, dass er die natürlichen Wege von Patienten wie Überweisungen nutzt und so die Spur der Daten verfolgt, um die gesuchten Informationen zu lokalisieren. Mit Hilfe eines Testprogramms konnte gezeigt werden, dass der Algorithmus in einer Testumgebung mit 2500 Knoten für die Suche nach Patientendaten sehr gute Ergebnisse liefert. Um Daten zu übertragen, wurde eine absichtlich begrenzte Zahl von Basisobjekten definiert, mit deren Hilfe unterschiedliche Daten in standardisiertem Format an andere Knoten übertragen werden können. Diese Objekte orientieren sich deshalb mehr an den verlangten Datenformaten als am Inhalt der Daten, was zu mehr Flexibilität führt. Durch die Implementierung einer Beispielanwendung wurde gezeigt, dass mit Hilfe von Web-Technologien wie Javascript und Ruby

on Rails eine komfortable Anwendung entwickelt werden kann, welche die in dieser Arbeit entworfenen Konzepte umsetzt. Weiter wurde gezeigt, wie sich die Infrastruktur an das vorhandene Krankenhausinformationssystem anbinden lässt.

In Kapitel 6 wurde demonstriert, wie eine verteilte Anwendung ohne zentrale Komponenten den Bestimmungen des Datenschutzes genügen kann. Die wichtigsten Erkenntnisse dieser Arbeit lassen sich folgendermaßen zusammenfassen: Eine Steuerung von Abläufen über die Grenzen einer Einrichtung hinweg ohne eine zentrale Instanz, welche diese Abläufe steuert, ist mit dem Konzept der Schedules realisierbar. Dabei können die Schedules auf einfache Art und Weise erstellt werden. Das Problem einer Suche nach Daten lässt sich in einer Umgebung wie der des Gesundheitswesens zufriedenstellend mit einer P2P-Suche und dem in dieser Arbeit entwickelten Algorithmus lösen. Dabei werden Informationen über den Aufenthalt eines Patienten nur an Knoten weitergegeben, die diese Informationen benötigen. Der Einsatz dieser Konzepte hat Vorteile gegenüber existierenden Infrastrukturen, da nur die Daten eine Einrichtung verlassen, von denen es gewünscht ist.

9.2 Ausblick

Aufgrund der Heterogenität des Gesundheitswesens und der notwendigen Komplexität einer IT-Infrastruktur, die einen Großteil der Kommunikationsprozesse abbilden kann, bietet das in dieser Arbeit behandelte Thema noch Raum für weiterführende Forschungsarbeiten. Insbesondere sollen hier drei Fragestellungen aufgezeigt werden. Das in dieser Arbeit entworfene Schedule-Konzept soll das Erstellen eines Ablaufplanes und dessen automatische Verarbeitung ermöglichen. An dieser Stelle könnte untersucht werden, wie dieses Konzept um eine automatische Optimierung von Abläufen erweitert werden kann. Wäre es möglich, dass das System unter Berücksichtigung anderer Schedules einen Ablaufplan so optimiert, dass Einsparungen zu erzielen sind? Ist beispielsweise der Transport eines Patienten in eine andere Einrichtung so zu organisieren, dass ein zweiter Patient, der die gleiche Untersuchung benötigt, mit Hilfe desselben Transportmittels zur Untersuchung befördert werden kann? Welchen Einfluß hätte eine solche Optimierung auf den Datenschutz? Könnten durch die dynamische Veränderung eines Schedules Rückschlüsse auf die Behandlung anderer Patienten gezogen werden?

Die zweite Fragestellung bezieht sich auf die Suche im P2P-Netz. Der hier entworfene Routingalgorithmus wird durch verschiedene Parameter wie die Suchtiefe oder die Anzahl der Nachbarknoten konfiguriert. Dieser Algorithmus könnte untersucht werden und ein Optimum dieser Parameter ermittelt werden. Dazu wäre es jedoch nötig, eine genaue Analyse der Patientenströme und der Verteilung der Patienten auf verschiedene Einrichtungen durchzuführen. Aus den Ergebnissen könnte man ein Netzwerk entwickeln, mit dessen Hilfe man die Parameter für die P2P-Suche optimieren könnte. Dabei sind besonders die Phasen eins und zwei von Interesse. Die dritte Fragestellung beschäftigt sich mit der Prüfung von Signaturen, indem Nachbarknoten als Certification Authority verwendet werden. Welche Sicherheitsrisiken bringt ein solches Verfahren mit sich? Wie kann man eine solche Vorgehensweise verbessern? So wurde in dieser Arbeit davon ausgegangen, dass mehr als ein Knoten befragt wird. Wieviele Knoten sollte man für die Prüfung einer Signatur anfragen? Wo liegt das Optimum

zwischen höchst möglicher Sicherheit und vertretbarem Aufwand? Diese Fragen können als Grundlage für weitere Untersuchungen dienen und ihre Beantwortung könnte die in dieser Arbeit entworfene Infrastruktur weiter verbessern.

Literatur

[ABK+04] Parvin Asadzadeh, Rajkumar Buyya, Chun Ling Kei, Deepa Nayar, and Srikumar Venugopal. Global Grids and Software Toolkits: A Study of Four Grid Middleware Technologies, 2004.

[aMWHG] Brian S. White anf Michael Walker, Marty Humphrey, and Andrew S. Grumshaw. LegionFS: A Secure and Scalable File System Supporting Cross-Domain High-Performance Applications. Presented at Supercomputing 2001.

[ATT08] Y. Akemastu, M. Tsuji, and F. Taoka. An empirical analysis of reduction of medical expenditures by the eHealth system: Further results. 10th International Conference on e-health Networking, Applications and Services, 2008.

[BCF+03] Diana Bosio, James Casey, Akos Frohner, Leanne Guy, Peter Kunszt, Erwin Laure, Sophie Lemaitre, Levi Lucio, Heinz Stockinger, Kurt Stockinger, William Bell, David Cameron, Gavin McCance, Paul Millar, Joni Hahkala, Niklas Karlsson, Ville Nenonen, Mika Silander, Olle Mulmo, Gian-Luca Volpato, and Giuseppe Andronico. Next-Generation EU DataGrid Data Management Services, 2003.

[BF06] Tobias Staub Bartol Filipovi. Grid Security Infrastruktur - ein Überblick, 2006.

[Bis03] Udo Bischof. Strukturierte P2P Netze, 2003.

[BK04] David Burdett and Nickolas Kavantzas. WS Choreography Model Overview, 2004.

[Bra08] Iris Braun. Vorlesungsscript SOA - Komposition von Webservices, 2008.

[BSI09] BSI. SOA-Security-Kompendium, 2009.

[CCF06] C.E. Chronaki, F. Chiarugi, and R. Fischer. OpenECG: Medical device interoperability as a quality label for ehealth services. Computers in Cardiology, pages 465 – 468, 2006.

[CCMea03] C. Chronaki, F. Chiarugi, E. Mavrogiannaki, and et. al. An eHealth platform for instant interaction among health professionals. Computers in Cardiology, 2003, 2003.

[CS06] Lisa Childers and Borja Sotomayor. Globus Toolkit 4 - Programming JAVA Services. Elservier, 2006.

[CSWH01] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. Lecture Notes in Computer Science, 2009, 2001.

[CVMea07] C. Chronaki, Kontoyiannis V., Mytaras M., and et. al. Evaluation of Shared EHR Services in Primary Healthcare Centers and their Rural Community Offices: the Twister Story. 29th Annual International Conference of the Engineering in Medicine and Biology Society, 2007.

[DFK+03] A. Demichev, D. Foster, V. Kalyaev, A. Kryukov, M. Lamanna, V. Pose, R. B. Da Rocha, and C. Wang. OGSA/Globus Evaluation for Data Intensive Applications, 2003.

[DKTea08] Asuman Dogac, Yildiray Kabak, Namli Tuncay, and et al. Collaborative Business Process Support in eHealth: Integrating IHE Profiles Through ebXML Business Process Specification Language. Information Technology in Biomedicine, 2008.

[DLS+] A. Dogac, G. Laleci, Kirbas S., Kabak Y., Sinir S., and Yildiz A.and Gurcan Y.

[Dog09] Erdogan Dogdu. Semantic web in ehealth. Proceedings of the 47th Annual Southeast Regional Conference, 2009.

[DP04] O. Delannoy and S. Petiton. A peer to peer computing framework: design and performance evaluation of YML. Third International Workshop on Parallel and Distributed Computing, 2004.

[EAT06] Marco Eichelberg, Thomas Aden, and Wilfried Thoben. A Distributed Patient Identification Protocol bsed on Control Numbers with Semantic Annotation. Int. Journal on Semantic Web and Information Systems, 2006.

[FGR03a] Ronaldo Ferreira, Christian Grothff, and Paul Ruth. A Transprot Layer Abstraction for Peer-to-Peer Networks. Proceedings of GP2PC 2003, 2003.

[FGR03b] Ronaldo A. Ferreira, Christian Grothoff, and Paul Ruth. A Transport Layer Abstraction for Peer-to-Peer Networks, 2003.

[FKT01] Ian Forster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Emabling Scalable Virtual Organizations. Intl J. Supercomputer Applications, 2001, 2001.

[For02] Ian Forster. What is the Grid? A Three Point Checklist, 2002.

[Fra05] Frauenhofer Institut Biomedizinische Technik. D2D-Technisches Handbuch des Kommunikationskonzeptes für patientenbezogene medizinische Daten, 2005.

[Gar06] Hans Jürgen Garstka. Informationelle selbstbestimmung, 2006.

[Gem08] Gematik. Gesamtarchitektur der EGK-Infrastruktur, 2008.

[Gem10a] Gematik. Begriffe und Abkürzungen der Telematik im Gesundheitswesen, 2010.

[Gem10b] Gematik. Das Elektronische Rezept, 2010.

[GG03] Chrisian Grothoff and Krista Grothoff. An Encoding for Censorship-Resistant Sharing, 2003.

[Glo05] Globus Toolkid Team. Delegation Service, 2005.

[HRNea08] Karina Harno, Pekka Ruotsalainen, Pirkko Nykaenen, and et al. Migration from Regional to a National eHealth Network. 2008.

[Hub] Valentina Huber. UNICORE: A Grid Computing Environment for Distributed and Parallel Computing. Springer Berlin / Heidelberg.

[Jan10] Kurt Jansson. Wikipedia - das Gesundheitssystem, 2010. Seite besucht am 23.04.2010.

[KBBea09] D. Krefting, J. Bart, K. Beronov, and et. al. Medigrid: Towards a user friendly secured grid infrastructure. Future Gener. Comput. Syst., 25(3):326–336, 2009.

[Law06] Kelvin Lawrence. Web Services Security: SOAP Security 1.1, 2006.

[LEMea06] K. Laskey, J. Estefan, F McCabe, and et. al. Reference Model for Service Oriented Architecture 1.0. OASIS SOA Reference Model, 2006.

[LMKM06] E. Le Merrer, A. Kermarrec, and L. Massoulie. Peer to peer size estimation in large and dynamic networks: A comparative study. 15th IEEE International Symposium on High Performance Distributed Computing, 2006.

[Mal07] Asif Zafar Malik. Telemedicine Country Report-Pakistan. 9th International Conference on e-Health Networking, Application and Services, 2007.

[Mar08] Tobias Martin. Website des Bundesministeriums für Gesundheit, 2008. Seite besucht am: August 8, 2008.

[Mau05] Thomas Maus. Elektronische Gesundheitskarte und Gesundheitstelematik, 2005.

[MEK03] Günter Müller, Torsten Eymann, and Michael Kreuzer. Telematik- und Kommunikationssysteme in der vernetzten Wirtschaft. Oldenburger Wissenschaftsverlag GmbH, 2003.

[MNLea08] Lorenzo Martino, Qun Ni, Dan Lin, and et. al. Multi-Domain and Privacy-aware Role Based Access Control in eHealth. Second International Conference on Pervasive Computing Technologies for Healthcare, pages 131–134, 2008.

[MRHea09] D. ManojPrasadh, A. Reddy, N. Hemachandar, and et. al. BCOS EPL : Building Co-Operative Systems for Extending Peer to Peer System Lifetime. International Conference on Signal Processing Systems, pages 215–218, 2009.

[MS91] Jonathan D. Moffett and Morris S. Sloman. Delegation of authority, 1991.

[MSVD] Olof Mogren, Oskar Sandberg, Vilhelm Verendel, and Devdatt Dubhashi. Adaptive Dynamics of Realistic Small-World Networks. Proceedings of the European Conference on Complex Systems 2009.

[New05] Sam Newman. A comparison of Django with rails, 2005.

[NNTHG01] Anand Natrajan, Anh Nguyen-Toung, Marty A. Humphrey, and Andrew S. Grumshaw. The Legion Grid Portal. Submitted to Grid Computing Environments 2001, Concurrency and Computation: Practice and Experience, 2001.

[NXNea09] Yoshiaki Nemoto, Lin Xiaodong, Kato Neu, and et al. Sage: a strong privacy-preserving scheme against global eavesdropping for ehealth systems. Selected Areas in Communications, IEEE Journal on , vol.27, 2009.

[OS05] Oskar Oskar Sandberg. Searching in a Small World. PhD thesis, Göteborg University, 2005.

[PS99] Klaus Pommerening and Marita Sergl. Kryptographische Protokolle, 1999.

[Res01] Eric Rescorla. SSL and TLS Designing and Building Secure Systems. Addison-Wesley, 2001.

[SAWea09] MC Schraefel, Paul Andre, Ryen White, and et al. Interacting with eHealth: towards grand challenges for HCI. Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, 2009.

[SDL+03] Heinz Stockinger, Flavia Donno, Erwin Laure, Shahzad Muzaffar, Peter Kunszt, Giuseppe Andronico, and Paul Millar. Grid data management in action: Experience in running and supporting data management services in the eu datagrid project. ECONFC0303241, TUAT007, 2003.

[sgb04] Sozialgesetzbuch der Bundesrepublik, 2004.

[SIVE08] T. Schepers, M. Iacob, and P Van Eck. A lifecycle approach to SOA governance. Symposium on Applied Computing, 2008.

[Sot05] Borja Sotomayor. The Globus Toolkit 3 Programme's Tutorial, 2005.

[spo04] Strafprozessordnung der Bundesrepublik, 2004.

[SS07] Alexander Schill and Thomas Springer. Verteilte Systeme. Springer, 2007.

[SS08] Daniel Slamaning and Christian Stingl. Privacy Aspects of eHealth. Third International Conference on Availability, Reliability and Security, 2008.

[Sta08] Statistische Bundesamt. Versichertenstatus in Deutschland, 2008.

[stg06] Strafgesetzbuch der Bundesrepublik, 2006.

[TGLea09] S. Taneva, G. Grote, E. Law, and et. al. Breaks in Continuity of Surgical Care: Considerations for eHealth Systems Design. eTELEMED '09. International Conference on eHealth, Telemedicine, and Social Medicine, 2009.

[Uni08] United Nations Economic Commission for Europe. UN/Edifact Verzeichnisse, 2008.

[urb04] Interne Leitlinien und Patientenpfade, 2004.

[VPP05] E. Vasilyeva, M. Pechenizkiy, and S. Puuronen. Towards the Framework of Adaptive User Interfaces for eHealth. Computer-Based Medical Systems, 2005.

[VTF01] Sudharshan Vazhkudai, Steven Tuecke, and Ian Foster. Replica Selection in the Globus Data Grid, 2001.

[WMS06] Fang Wang, Yamir Moreno, and Yaoru Sun. Structure of Peer-to-Peer Social Networks. Physical Review E, 73:036123, 2006.

[YLHea09] Junfeng Yu, Zhitang Li, Jun Hu, and et. al. Using Simulation to characterize Topology of Peer-to-Peer Botnets. Computer Modeling and Simulation, 2009. ICCMS '09. International Conference, pages 78 – 83, 2009.

[ZCXH05] Quin Zongyan, Cai Chao, Zhao Xiangpeng, and Yang Hongli. Exploring into the Essence of Choreography, 2005.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Glossary

BLOB Binary Large Object, 53

BPEL Business Process Execution Language - Eine Beschreibungssprache mit deren Hilfe Webservisces orchestriert werden können, 24

BQS Bundesgeschäftsstelle Qualitätssicherung GGMBH, 78

CAD Computer Aided Diagnosis, 81

CEK content encryption key, 114

CHK Content-hash Key, 28

CMS Cryptographic Message Syntax, 114

CRUD Abkürzung - Create, Read, Update, Delete, 22

D2D Doctor to Doctor ist ein System zum sicheren Versand von Arztbriefen, 9

DICOM Digital Imaging and Communication in Medicine, 7

DIMDI Deutsches Institut für medizinische Dokumentation und Information, 78

DIT Director Information Tree, 85

DKC Diakoniekrankenhauses Chemnitzer Land, 6

DNF DataNotFound - Nachricht die zuruckgeliefert wird falls eine Freenetsuche erfolglos endet, 31

DRG Diagnoses Related Groups - Modell zum Abbilden von Erkrankungen und Behandlungen auf Pauschalen, 7

ECRS Encoding for Censorship-Resistant Sharing, 36

EGK Elektronische Gesundheitskarte, 15

FMRI Functional Magnetic Resonance Imaging, 14

GPL Geral Public License, 35

HL7 Health Level 7 ist ein offener Standard für die Kommunikation zwischen Systemen im Gesundheitswesen, 8

HTL Hops to live - Suchtiefe bei welcher Abgebrochen wird, 29

IHE Integrated Healthcare Enterprise, 25

IK Institutionskennzeichen, 51

IKE Internet Key Exchange, 112

ISAKMP Internet Security Association and Key Management Protocol, 112

JMC Job Monitor Component, 26

JPA Job Preparing Agent, 26

KIS Krankenhaus-Informationssystem - Gesamtheit aller Informationssysteme in Krankenhaus, 6

LAN Local Area Network, 41
LDT Labor Daten Transfer - Format zur Ubertragung von Labordaten, 9

MD5 Message Digest Algorithm 5 - kryptographische Hashfunktion mit 128 Bit, 15
MDK Medizinischer Dienst der Krankenkassen - Gemeinschaftseinrichtung der Krankenkassen, welche die Rechnungen der Einrichtungen uberpruft, 71
MLLP Minimal Lower Layer Protocol, 132
MOSS MIME Object Security Services, 114
MPI Master Patient Index, 43
MTU Maximum Transmission Unit, 35

NAT Network Adress Translation, 113
NGR Next Generation Routing - Verbessertes Routing in Freenet, 30
NJS Network Job Supervisor, 26

OASIS Organization for the Advancement of Structured Information Standards, 21
OGSA Open Grid Services Architecture, 27
OGSADAI Standard Grid Datei Interface, 14

PACS Picture Archiving and Communication System - dient zur Archivierung von Röntgenbildern, 7
PKI Public-Key-Infrastruktur, 130
PMS Patientenmanagementsystem - Informationssystem zur Verwaltung der Patientendaten, 6

REST Representational State Transfer, 22
RID Retrival Information for Display, 15
RIPEMD-160 RACE Integrity Primitives Evaluation Message Digest - offener Kryptoalgorithmus zum Erzeugen von Hashwerten, 15
RIS Radiologie Informationssystem, 7
RLS Record Linkage Server, 15

SA Security Association in IPSec, 113
SCIPHOX Standardized Communication of Information Systems in Physician Offices and Hospitals using XML, 12
SDS Service Discovery Service, 20

SHA-1 Secure Hash Algorithm - Sammlung verschiedener Hashalgorithmen, 15
SOA Serviceorientierte Architektur, 21
SOAP Simple Object Access Protocol, 17
SSK Signed-subspace Key - Schlüssel zum Erstellen eines Namensraumes in Freenet, 28
SSL Secure Socket Layer, 113

TLS Transport Layer Security, 19
TSI Target System Interface, 26
TTP Trustet Third Party, 15

UDDI Universal Description, Discovery and Integration, 20

WS-CDL Web Services Choreography Description Language, 37
WSDL Webservice Definition Language, 38
WSS4R Webservices Security for Ruby, 115

XACML eXtensible Access Control Markup Language, 20
XML Extensible Markup Language, 12
XML-RPC Remote Procedure Call, 22

Eigenständigkeitserklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet. Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Dresden, den 28.04.2010