Anbindung der orthopädischen Praxis Dr. Fischer an die Kliniken Hochfranken

D i p l o m a r b e i t

an der Fachhochschule Hof

Fachbereich Informatik / Technik

Studiengang Wirtschaftsinformatik

Vorgelegt bei

Prof. Dr. Jörg Scheidt

Alfons-Goppel-Platz 1

95028 Hof

vorgelegt von

Matthias Reizammer

Wichernstrasse 3

95615 Marktredwitz

Hof, 27. Juni 2007

Danksagung

Mein Dank gilt Herrn Professer Dr. Scheidt für die Unterstützung bei der Themenfindung, die Betreung während der Arbeit sowie für die Herstellung der ersten Kontakte, Herrn Jürgen Hager für die Vermittlung weiterer Kontakte und die sehr gute Unterstützung bei den zahlreichen Problemen und die Mithilfe bei der Organisation, Herrn PD Dr. med Simank für die Ermöglichung dieser interessanten Diplomarbeit sowie für die Übernahme der praxisinternen Organisation und Kommunikation unten den Kollegen, den Herren Kühl und Heimann von Oehm und Rehbein für die tatkräftige Unterstützung bei der Erarbeitung einer Lösung für die Röntgenbildübertragung sowie Herrn Faatz von der Firma H-M Praxissysteme für die Erarbeitung einer Lösung für das Medistarsystem.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AE Application Entity

AH Authentication Header

DICOM Digital Imaging and Communications in Medicine

DIMSE DICOM Message Service Element

DRG Diagnosis Related Groups

ESP Encapsulating Security Payload

HMAC Hashed Message Authentication Code

IOD Information Object Definition

IPsec Internet Protocol Security

PACS Picture Archiving und Communication System

RIS Radiologie-Informationssystem

SA Security Association / Sicherheitsassoziation

SC Service Class

SCP Service Class Provider

SCU Service Class User

SG Service Group

SOP Service Object Pair

SPI Stateful Packet Inspection

VPN Virtual Private Network

1 Einführung

Der Titel der Arbeit deutet bereits darauf hin, dass sich die Diplomarbeit mit einem medizinischen Thema befasst. Auf den ersten Blick etwas ungewöhnlich für eine Diplomarbeit des Studiengangs Wirtschaftsinformatik, aber bei genauerer Analyse des Themas wird auch ersichtlich, dass es um eine Verbindung zweier Parteien geht. An diesem Punkt kommt die Informatik ins Spiel. Es müssen zwei eigenständige Netzwerke miteinander verbunden werden, die bisher autonomen Informationssysteme müssen miteinander kommunizieren können.

Für die Kommunikation der Informationssysteme ist Wissen aus der Medizininformatik, den Abläufen im medizinischen Umfeld sowie den strengen Datenschutzrichtlinien erforderlich. Informatik im medizinischen Bereich gehört seit einiger Zeit zu meinen favorisierten Themengebieten und ich habe deshalb schon diverse Praktika in einer Klinik absolviert. Aus diesem Grund war ich sehr angetan, als mir dieses Diplomarbeitsthema angeboten wurde, da es mir die Möglichkeit bietet, mich mit neuen Themen auseinanderzusetzen und bereits Bekanntes in der Praxis anzuwenden. Nach ersten Gesprächen wurde sehr schnell deutlich, dass auch viele administrative Dinge zu planen sind, was meinem zukünftigen Berufsziel entspricht.

Der Aufbau der Diplomarbeit orientiert sich an der weitverbreiteten Gliederung für Projekte und beginnt mit der Ausgangssituation sowie den Zielen, die erreicht werden sollen. Anschließend werden in einem Kapitel die theoretischen Grundlagen für das Projekt notwendigen Themengebiete beleuchtet, wobei besonderes Augenmerk auf den DICOM-Standard fällt, da dieser in unterschiedlichen Ausprägungen von jedem Radiologiesystemhersteller implementiert wird.

Im nächsten Kapitel werden Möglichkeiten erörtert, wie die Anbindung der Praxis an die Klinik und die Kommunikation der autonomen Informationssysteme theoretisch erfolgen kann. Das darauffolgende Kapitel erläutert, welche der beschriebenen Lösungsmöglichkeiten auf-

gegriffen wurden, um diese in die Praxis umzusetzen und warum die anderen Optionen nicht realisierbar sind. Im Anschluss daran erfolgt eine kurze Bewertung des Endergebnisses.

Den Abschluss der Arbeit bildet die Spezifikation eines Modells, das der Simulation einer optimalen Anbindung von Klinik und Praxis dient. Es bildet den gängigen Arbeitsablauf ab und konzentriert sich auf die wesentlichen Funktionen. Die dort zu findenden Patientendaten sind reine Testdaten und erfüllen keinen Anspruch auf Vollständigkeit im realen Leben. Das spezifizierte Modell wurde in Java implementiert und steht auf der CD-ROM zur Diplomarbeit zur Verfügung. Eine Installationsanleitung befindet sich in Anhang B.

2 Ausgangssituation

Die Ausgangssituation für das Projekt, die Anbindung der orthopädischen Gemeinschaftspraxis an die Klinik Münchberg, ist aufgrund der vielen Beteiligten auf beiden Seiten als Komplex einzustufen. Auf der Seite Praxis gilt es die Vorstellungen von vier Ärzten zu koordinieren und auf eine mit der Klinik abgestimmte und realisierbare Linie zu führen.

Neben den Ärzten und dem Klinik-EDV Verantwortlichen sind auch noch die Hersteller der Informationssysteme beteiligt. Auf Praxisseite Medistar für das Praxisinformationssystem, vertreten durch H-M Praxissysteme, sowie Oehm und Rehbein für das digitale Röntgen, auf der Klinikseite Agfa Health Care für das Krankenhausinformationsystem Orbis sowie das PACS.

Medistar ist ein speziell für niedergelassene Ärzte zugeschnittenes Informationssystem. Das System besteht aus mehreren Modulen. Die Basis bildet die Verwaltung des Patientenstammes sowie die Dokumentation medizinischer Daten und die dazugehörige Leistungsabrechnung. Mit Hilfe von Zusatzmodulen erleichtert Medistar z.B. die BG-Abrechnung, vereinfacht Hausbesuche durch das Hausbesuchsmodul und die Verwaltung von Labordaten, falls das entsprechende Labor eine Datenfernübertragung unterstützt, wodurch die Daten automatisch in die Kartei des Patienten eingetragen werden können.

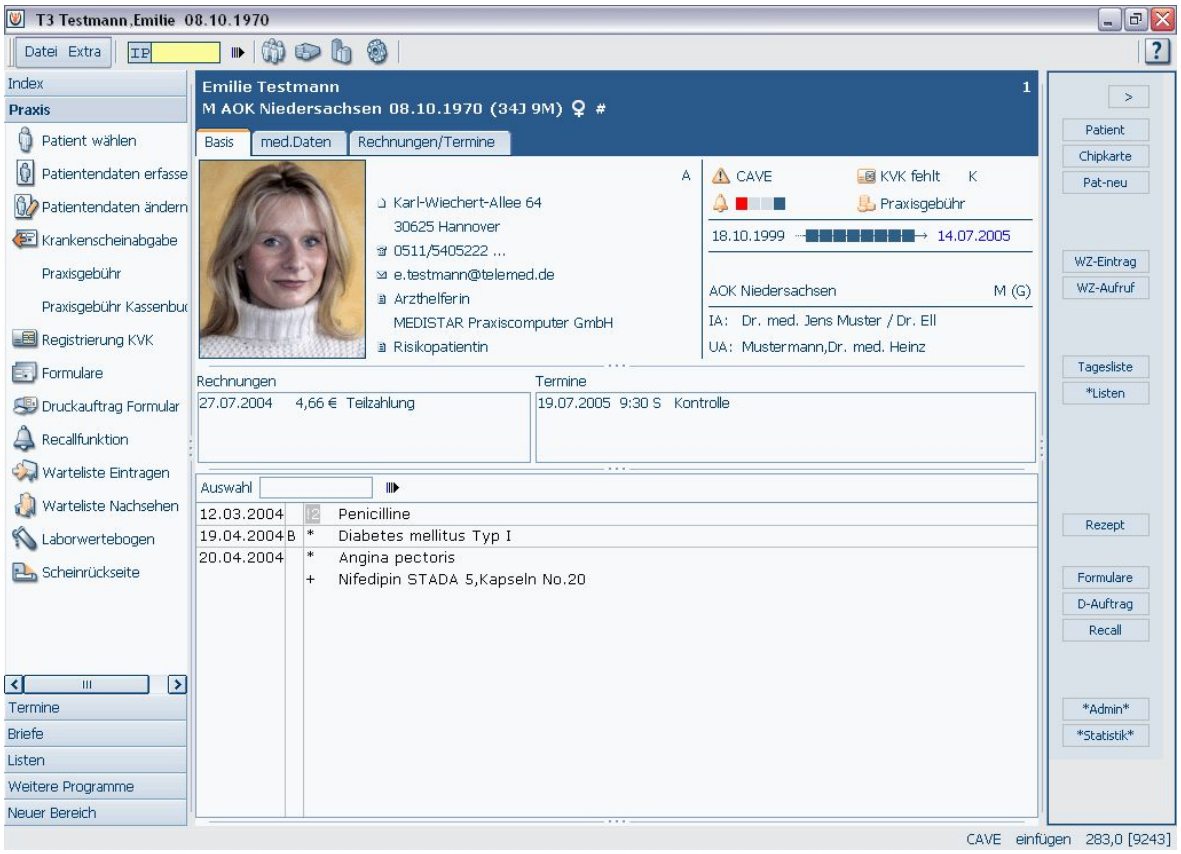

Abbildung 1: Patienteninformation in Medistar[1]

Im Gegensatz dazu ist Orbis für den Klinikalltag entwickelt. Es basiert auf für jede Einrichtung definierbare Workflows und lässt sich somit individuell zuschneiden. Das Basismodul unterstützt den administrativen Prozess Aufnahme - Verlegung - Entlassung - Leistungsdokumentation bis zur medizinischen Dokumentation Anamnese - Untersuchung - Therapie - Verlauf - Arztbriefschreibung.

[1] siehe http://www.medistar.de/images/medistarprg/mdpatinfoauswahl_0705.gif

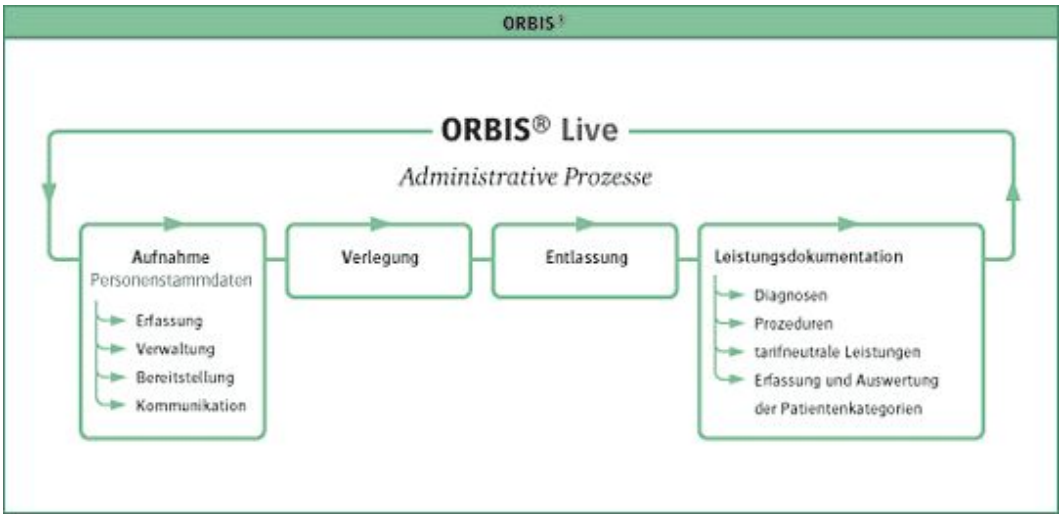

Abbildung 2: Der administrative Prozess in Orbis[2]

Hinzu kommen Funktionalitäten für den Medikationsprozess von der Erfassung der Medikamente bis zur Therapiekontrolle, die elektronische Leistungsanforderung an die Funktionsstellen wie z.B. Labor und Röntgen, Laborbefundpräsentation sowie übergreifende Prozesse wie eine Diktatfunktion, Formulardesign und Textbausteine[3].

Die Basis lässt sich mit Hilfe von Modulen um spezielle Workflows ergänzen. Diese sind in Gruppen aufgeteilt und umfassen medizinische, diagnostische und andere Spezialfälle wie z.B. Intensivmedizin, OP und Anästhesie, Labor, Radiologie und Controlling. Somit lässt sich mit Orbis die gesamte Klinikwelt in einem System abbilden.

Somit liegt eine heterogene Systemlandschaft vor. Eine Kopplung von heterogenen Systemen gestaltet sich grundsätzlich schwierig, da gemeinsame Schnittstellen geschaffen werden müssen, sofern kein einheitlicher Standard zur Verfügung steht, den die beteiligten Systeme implementieren.

[2] siehe http://www.agfa.com/germany/de/binaries/orbis_admin_process_540_tcm181-25893.gif

[3] Detailliertere Informationen über Orbis siehe

http://www.agfa.com/germany/de/he/solutions/krankenhausweite_it/index.jsp

Der derzeitige Arbeitsablauf von der Vorstellung des Patienten in der Praxis über die Einweisung in die Klinik bis zur Leistungsabrechnung nach erfolgter Behandlung hat zu Projektstart folgendes Aussehen:

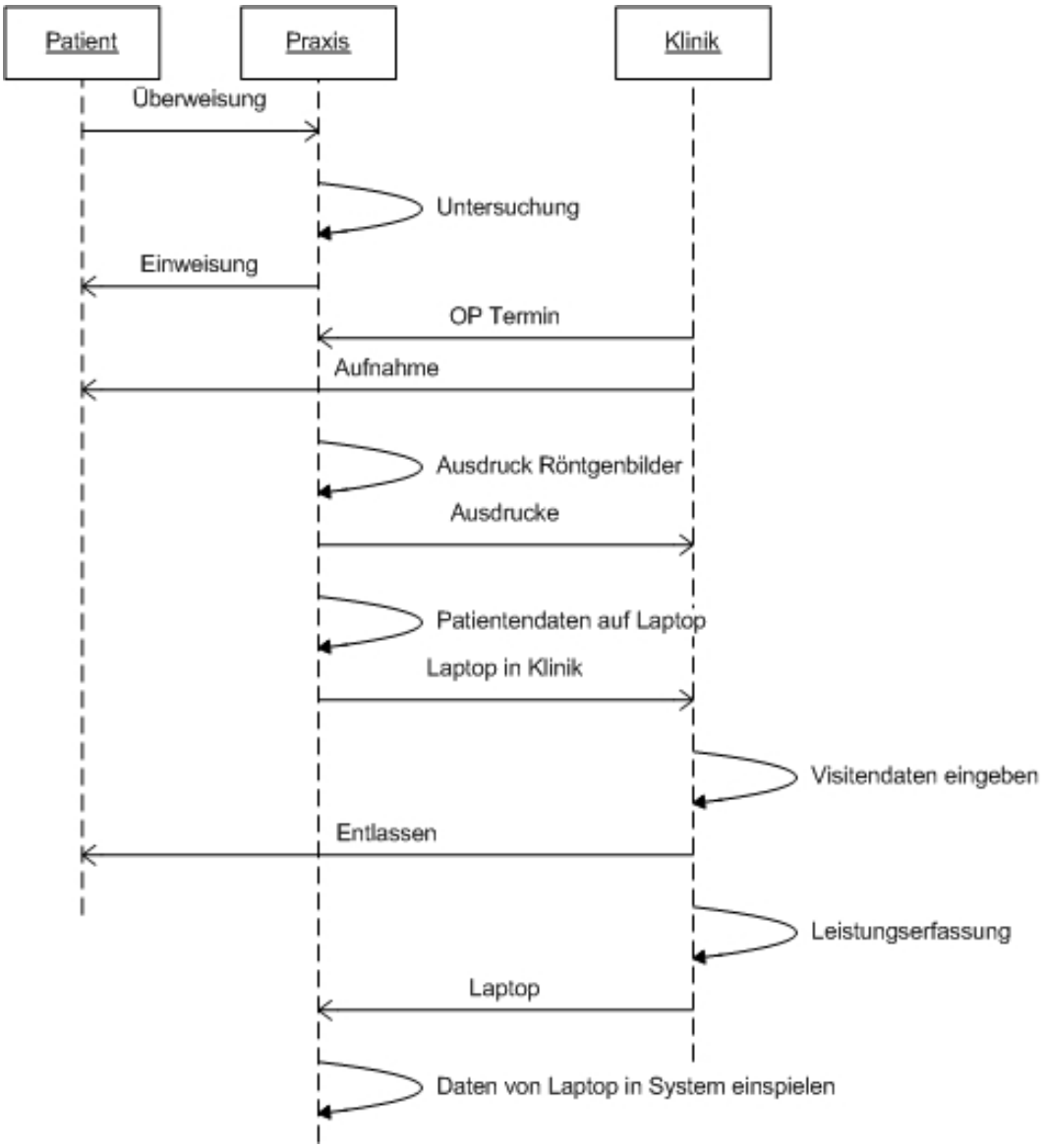

Abbildung 3: Arbeitsablauf bei Projektstart

Nach einer Überweisung des Hausarztes stellt sich der Patient in der Praxis vor. Dort wird dieser im System angelegt, wenn er noch nicht verzeichnet ist und untersucht. Je nach Untersuchungsergebnis erfolgt eine normale Behandlung oder eine stationäre Einweisung. Anschließend wird in der Praxis ein Operationstermins mit der Klinik festgelegt. Die für die Operation benötigten Röntgenbilder werden ausgedruckt, die Patientendaten auf den Laptop

übertragen und beides termingerecht in die Klinik transportiert.

Während des Aufenthalts des Patienten werden die auflaufenden Daten in den Laptop eingegeben. Nach der Entlassung des Patienten erfolgt die Abrechnung der erbrachten Leistungen und der Laptop kommt zur Datensynchronisierung in die Praxis.

Dieser für einen Patienten beschriebene Ablauf wird im Praxisalltag natürlich für mehrere Patienten gleichzeitig durchgeführt. Die Datensynchronisation erfolgt einmal pro Woche vorgenommen, damit die in der Klinik erhobenen Daten relativ zeitnah auch in der Praxis verfügbar sind.

3 Ziele des Projektes

Der oben beschriebene Arbeitsablauf ist an manchen Punkten doch sehr umständlich. Somit ergibt sich als Ziel eine Verbesserung der Abläufe an den Schnittstellen, an denen Daten von der Praxis in die Klinik und umgekehrt transportiert werden müssen. Dabei gibt es zwei Kategorien von zu tranportierenden Daten, zum einen Bilddaten, zum anderen Textdaten.

Oberstes Ziel muss bei diesem Projekt die Erhaltung der Stabilität sowohl der Klinik- als auch der Praxissysteme sein. Für die Bilddaten lassen sich folgende Ziele definieren:

- Verbesserung der Qualität der übertragenen Bilder aus der Praxis
- Verbesserung des Ablaufs des Transportes der Bilder in die Klinik
- Einrichtung eines zuverlässigen Rücktransportes der postoperativen Bilder aus der Klinik in die Praxis

Für die textuellen Daten wie die Patientengeschichte, Befunde und Diagnosen gelten folgende Ziele:

- Ermöglichen eines direkten Zugriffs auf die Daten des Praxissystems
- Vermeidung von doppelter Arbeit bei der Datenerfassung soweit möglich
- Zeitnahe Aktualisierung der Daten in der Praxis

4 Theoretische Grundlagen

In diesem Kapitel folgt die theoretische Beschreibung der Technologien, die für die praktische Umsetzung notwendig sind.

4.1 Virtuelles Privates Netzwerk - VPN

Ein virtuelles privates Netzwerk (engl. Virtual Private Network) ist ein Rechnernetzwerk, das zum Transport privater Daten ein öffentliches Netz (zum Beispiel das Internet) verwendet. Somit wird eine sichere Übertragung über ein unsicheres Netzwerk ermöglicht. Die Teilnehmer eines VPN können Daten wie in einem internen LAN austauschen, das VPN ist somit transparent. Die Verbindung über das öffentliche Netz wird normalerweise verschlüsselt, was aber nicht immer der Fall sein muss. Zwischen den Teilnehmern wird ein Tunnel aufgebaut, über welchen die Datenübertragung stattfindet.
Es gibt mehrere Typen von virtuellen privaten Netzen:

- Site-to-Site

  Dieser Typ wird dazu verwendet, um zwei private Netzwerke miteinander zu verbinden. Dazu wird auf beiden Seiten ein VPN-Gateway installiert. Die Gateways bauen einen Tunnel auf, über welchen Daten in das andere Netz gesendet werden. Diese Methode findet z.B. bei der Verbindung von Firmenstandorten Verwendung.

- Site-to-End

  Will ein Unternehmen Mitarbeitern von außen Zugang zum Firmennetz verschaffen, wird dieser Typ verwendet. Dazu baut der Mitarbeiter eine Verbindung zu dem ihm bekannten VPN-Gateway (Remote-Access-VPN) auf und kann nun so arbeiten, als würde er direkt im Firmengebäude an seinem Arbeitsplatz sitzen

- End-to-End

  Dieses Verfahren dient der Verbindung zweier mobiler Mitarbeiter oder dem Aufbau einer hochsicheren virtuellen Abteilung innerhalb eines lokalen Netzwerks. Diese beiden Szenarien sind derzeit nur mit Hilfe der ViPNet-Technologie möglich, wobei ein

zentraler Kommunikationsserver die Parameter der einzelnen Netzknoten erfasst und diese an die anderen Knoten weiterreicht. Mit diesen Parametern kann dann zwischen Clients ein direkter privater oder ein indirekter Tunnel über den Kommunikationsserver aufgebaut werden.

Um die Sicherheit zu gewährleisten, sollten VPN-Gateways immer eine Firewall mit Paketfilter[4] verwenden. Ansonsten kann Schadcode in das private Netz eingeschleust werden. Zur Authentifizierung der VPN-Gateways können Passwörter, öffentliche Schlüssel oder Zertifikate[5] eingesetzt werden. [6]

4.2 Firewall

Eine Firewall ist eine Netzwerksicherheitskomponente, deren Aufgabe darin besteht, den Netzwerkverkehr zwischen zwei Netzen zu überwachen. Eine Firewall besteht sowohl aus Hardware- als auch aus Softwarekomponenten. Die Hardwarekomponenten sind Rechner mit mindestens zwei Netzwerkschnittstellen, Softwarekomponenten sind Paketfilter, Intrusion Detection Systeme uvm.[7] Im folgenden werde ich nur auf den für VPN notwendigen Paketfilter näher eingehen.

Ein Paketfilter ist eine Software, die dazu dient, den ein- und ausgehenden Netzwerkverkehr zu überwachen und unerwünschte Pakete zu filtern. Ein einfacher Paketfilter prüft jedes Datenpaket auf Senderadresse, Empfängeradresse, Quell- und Zielport und vergleicht diese Daten mit seinem Regelwerk. Ist das Paket erlaubt, wird es weitergeleitet (FORWARD bzw. PERMIT), ist es nicht zulässig wird es entweder ignoriert (DENY) oder zurückgesendet (REJECT).[8]

Eine weitere Technologie eines Paketfilters ist die Stateful Packet Inspection (SPI). Jedes Paket wird hierbei analysiert und in einer dynamischen Zustandstabelle gespeichert. Mit dieser

[4] Siehe Kapitel 4.2
[5] Siehe Kapitel 4.3
[6] vgl. http://de.wikipedia.org/wiki/Virtual_Private_Network
[7] vgl. http://de.wikipedia.org/wiki/Firewall
[8] vgl. http://de.wikipedia.org/wiki/Paketfilter

Tabelle werden weitere Pakete verglichen; aufgrund von Vergleichen wird die Entscheidung zur Weiterleitung oder zum Verwerfen des Paketes getroffen. [10]

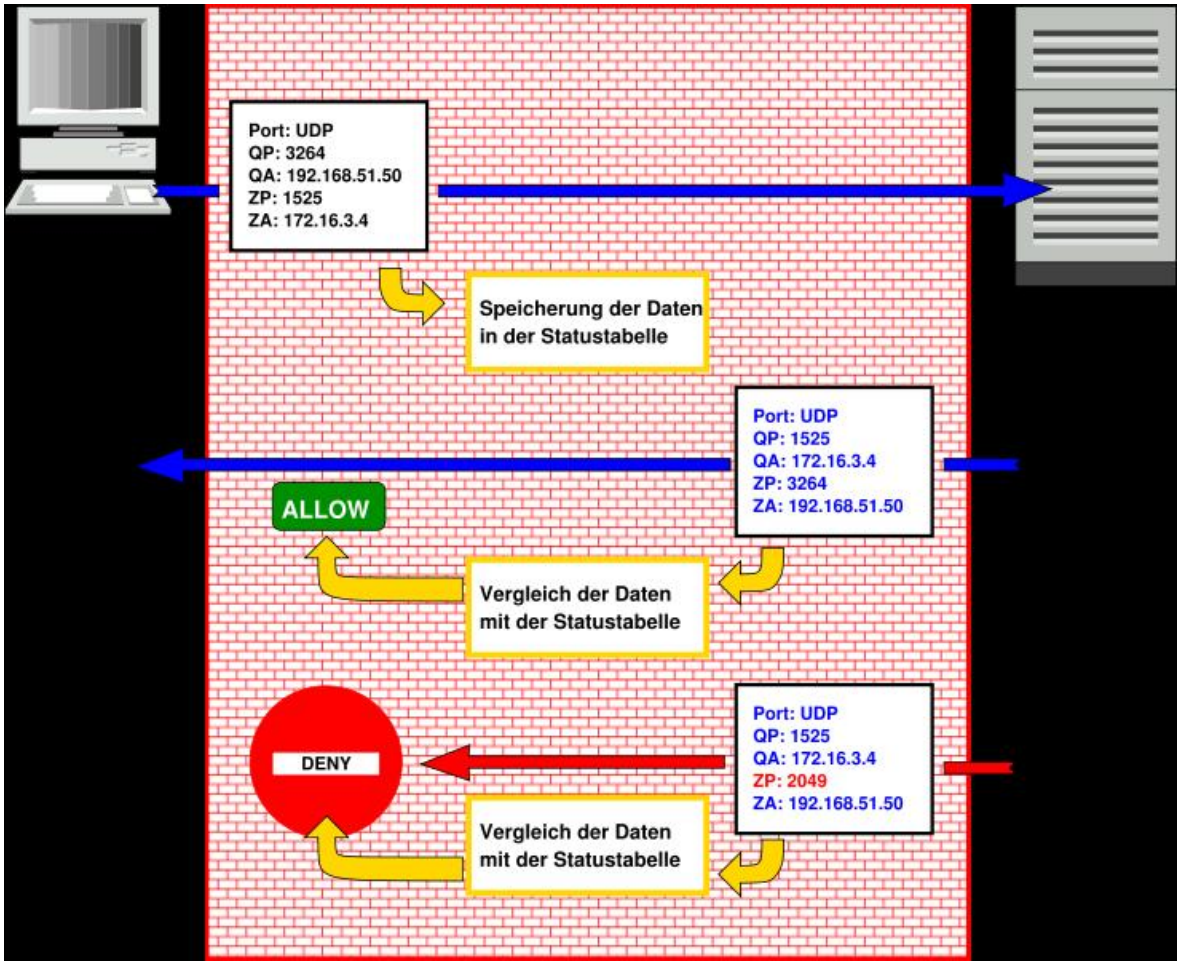

Abbildung 4: Funktionsweise der Stateful Packet Inspection[9]

[9] http://de.wikipedia.org/wiki/Bild:Stateful_inspection_udp.svg

[10] vgl. http://de.wikipedia.org/wiki/Stateful_Packet_Inspection

4.3 IPsec

IPsec steht für Internet Protocol Security und ist eine relativ junge Sicherheitsarchitektur, die erst im Jahr 1998 entwickelt wurde. Die RFCs 2401, 2402 und 2406 beschreiben dessen Architektur. IPsec gewährleistet die Schutzziele Datenschutz, Datenintegrität sowie Schutz vor Replay-Angriffen und basiert auf einer symmetrischen Verschlüsselung, da eine hohe Geschwindigkeit bei der Verschlüsselung benötigt wird.

Obwohl sich IPsec in der IP-Schicht befindet, ist es verbindungsorientiert. Dies erklärt sich dadurch, dass Sicherheit erst durch eine Art Verbindung möglich ist. Im Kontext von IPsec wird eine Verbindung als Sicherheitsassoziation(SA) bezeichnet. Eine SA ist eine Simplex-Verbindung zwischen zwei Endpunkten. Ist sicherer Datenverkehr in beide Richtungen notwendig, müssen zwei SAs aufgebaut werden.

IPsec bietet zwei Betriebsmodi, wovon aber immer nur einer aktiv sein kann.

- Transportmodus
  Im Transportmodus wird der IPsec-Header direkt nach dem IP-Header eingefügt, welcher Sicherheitsinformationen wie die Sicherheitskennung und eine Integritätsprüfung der Nutzdaten des Pakets enthält. „Der Transportmodus wird verwendet, wenn die „kryptographischen Endpunkte“ auch die „Kommunikations-Endpunkte“ sind“[11].

- Tunnelmodus
  Im Tunnelmodus wird das gesamte ursprüngliche IP-Paket (Sender-,Empfängeradresse, Daten) in einem neuen IP-Paket gekapselt. Der Datenteil des neuen IP-Pakets enthält das gesamte „alte“ Paket, der neue Header die Adressen der Tunnelendpunkte. Der Vorteil dieses Modus liegt darin, dass nur auf den Gateways IPsec konfiguriert werden muss und Angreifer nur die Tunnelendpunkte, aber nicht die wirklichen Teilnehmer erfahren, da die Adressen der Teilnehmer in dem neuen IP-Paket gekapselt sind.

[11] http://de.wikipedia.org/wiki/IPSec Kapitel 7

Der Verbindungsaufbau kann unterschiedlich erfolgen. Man unterscheidet zwei Möglichkeiten des Verbindungsaufbaus.
Manual Keying:
Hierbei werden die verwendeten Schlüssel für IPsec auf beiden Tunnelendpunkten fest konfiguriert.
IKE(Internet Key Exchange):
Dieses Protokoll wird für die Verwaltung der Schlüssel verwendet. Zum Aushandeln des Schlüssels kommt das Diffie-Hellman-Schlüsselaustauschverfahren[12] zum Einsatz. Der über diesen Weg ausgehandelte Schlüssel wird dann für die Verschlüsselung benutzt.

IPsec führt zwei neue Header ein, welche unterschiedlich eingesetzt werden. Der AH(Authentication Header) bietet keine Verschlüsselung sondern nur eine Integritätsprüfung und einen Schutz vor Replay-Angriffen. Der ESP(Encapsulating Security Payload)-Header bietet zudem noch eine Verschlüsselung der Daten.

Der AH im Transportmodus stellt sich wie folgt dar:

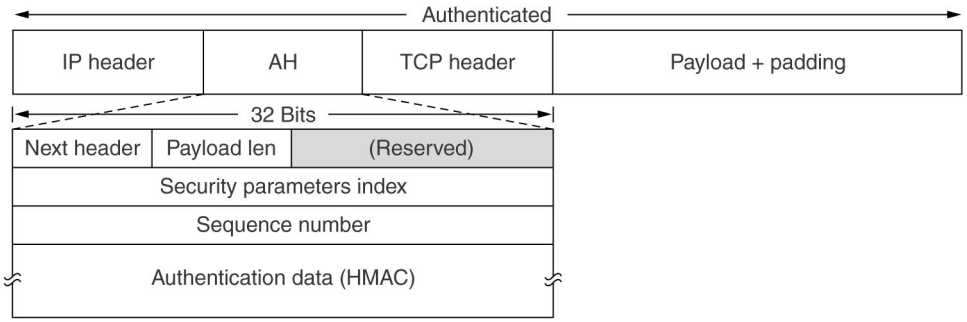

Abbildung 5: Der Authentifizierungsheader von IPsec im Transportmodus[13]

Die einzelnen Felder des AH haben folgende Bedeutung:

- Nächster Header

  Hier befindet sich in den meisten Fällen der Code für TCP

[12] siehe Kapitel Kryptografie
[13] [Tan03] Abbildung 8.27 Seite 835

- Nutzdatenlänge

  Gibt die Anzahl der 32-Bit-Worte im AH-Header minus 2 an

- Sicherheitsparamterindex

  Dieses Feld enthält Informationen über die Sicherheitsassoziation wie den gemeinsamen Schlüssel

- Folgenummer

  Die Folgenummer dient der Nummerierung der Pakete, die auf der SA gesendet wurden. Jedes Paket besitzt eine eindeutige Nummer. Dies dient dem Schutz vor Replay Angriffen, da jede Nummer nur einmal verwendet werden darf.

- Authentifizierungsdaten(HMAC)

  Dieses Feld ist in seiner Länge variabel und erhält die Prüfsumme, welche mit Hilfe des gemeinsamen Schlüssels, der beim Verbindungsaufbau ausgehandelt wurde, erstellt wird. Der Hashwert berechnet sich über das Paket plus den gemeinsamen Schlüssel, wobei keine Übertragung des gemeinsamen Schlüssels stattfindet. Dies wird als HMAC(Hashed Message Authentication Code) bezeichnet.

Der ESP-Header in den beiden Modi von IPsec:

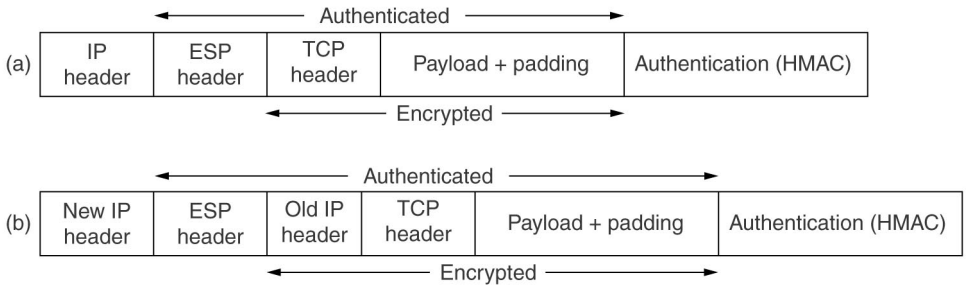

Abbildung 6: Der ESP-Header im (a) Transportmodus (b) Tunnelmodus[14]

Der ESP-Header besitzt die bereits vom AH bekannten Felder Sicherheitsparameterindex, Folgenummer und Authentifizierungsdaten. [15]

4.4 DICOM-Standard

4.4.1 Allgemeines über den Standard

Die Notwendigkeit, einen Standard für Bildkommunikation zu schaffen, entstand um 1970, als die ersten Computertomographen installiert wurden und digitale Bilder verarbeiteten. Jedoch entwickelte jeder Hersteller seine eigenen Bildkommunikationssysteme, welche stark spezialisiert waren und somit die herstellerübergreifende Kommunikation unmöglich machten.

Aus diesem Grund entstand 1983 eine Arbeitsgruppe aus dem American College of Radiology (ACR) und der National Electrical Manufacturers Association (NEMA) mit dem Ziel, Daten von unterschiedlichen Systemen zusammenzuführen. Die Version 1.0 des ACR-NEMA Standards wurde 1985 auf dem RSNA-Kongress (Radiological Society of North America) in Chicago publiziert. In dieser Version wurden Hardware-Interface sowie alle notwendigen

[14] [Tan03] Abbildung 8.28 Seite 836
[15] vgl. [Tan03] Kapitel 8.6.1 Seiten 833-837

Datenelemente und Protokolle für den Datenaustausch spezifiziert. Genau wie die verbesserte Version 2.0 war dieser Standard aber nicht netzwerkfähig und erlaubte nur Punkt-zu-Punkt Verbindungen.

Um die Netzwerkfähigkeit zu erreichen, wären weitreichende Änderungen im ACR-NEMA Standard notwendig gewesen. Deshalb erschien 1993 der neue Standard mit dem Namen DICOM 3.0 (Digital Imaging and Communications in Medicine). Der DICOM Standard ist mit objektorientierten Methoden entwickelt worden und definiert nicht wie ACR-NEMA Bilder, sondern Objekte, die Bilder enthalten. DICOM ist nach dem OSI-Modell (Open System Interconnection Modell) entworfen worden. Dieses Modell erlaubt die Kommunikation zwischen heterogenen Systemen.

DICOM legt ein Informationsmodell fest, welches Datenstrukturen und Messages für eine bestimmte Dienstleistung definiert. Zudem enthält der Standard Beschreibungen für Informationsobjekte (z.B. Bilder), den Übertragungsmechanismus sowie für die Speicherung von Informationen. Darüberhinaus sind noch Services für die Workflow Unterstützung in der Radiologie enthalten.

In DICOM werden Objekte als Informations-Objekte (Information Objects), Aktivitäten (Operations) oder Dienstleistungen(Services) als Service Classes bezeichnet. Dies sind jedoch allgemeine Definitionen. Ein spezifisches Auftreten einer Klasse nennt man Ereignis(Instance).

Mit DICOM lassen sich aber nicht nur Bilder übertragen, sondern auch die daran gekoppelten Daten wie z.B. Befundungsergebnisse, diese können zusammen mit dem Bild im PACS archiviert werden. Zudem ist die Interaktion mit einem RIS/KIS[16] möglich. Leistungsanforderungen werden als HL7-Nachricht[17] gesendet und die Antwort erfolgt als DICOM-

[16]RIS=Radiologieinformationssystem, KIS=Krankenhausinformationssystem
[17]HL-7 siehe Anhang A

Nachricht. Dies ist durch die Intergration der beiden Standards HL7 und DICOM gegeben. [18]

4.4.2 Bestandteile des Standards

Der DICOM Standard ist in sogenannte Parts untergliedert, die klar voneinander getrennt sind. 2005 waren es 18 Parts, wovon manche aber noch nicht ganz vollständig sind. Die ersten neun Parts sind die Basis des Standards mit folgenden Elementen:

- Zielsetzung
- Conformance Statement
- Datenobjekte
- Funktionen(Protokolle)
- binäre Codierung
- Definitionen
- Netzwerk Protokoll (Part 7 =Services)
- Übergabepuffer an TCP/IP (Part 8 = Netzwerkprotokoll)
- DICOM Stacker (zurückgezogen)
- Fileformat
- binäre Codierung und Datentypen-Anforderungen auf Medien
- Medizin
- Punkt zu Punkt drucken (zurückgezogen)

[18] vgl. [Gär05] Kapitel 2.1 - 2.2 Seiten 45 - 49

- konsistente Graustufen-Bilddarstellung

Die restlichen Parts sind Ergänzungen und Modifikationen und wurden erst nachträglich entwickelt. Für die laufende Entwicklung und Fortschreibung sind ungefähr 20 Arbeitsgruppen des DICOM-Kommittees verantwortlich, die Supplements und Correction Proposals erarbeiten:

- Supplements sind Erweiterungen, die neue Dienste oder Objekte hinzufügen oder Teile aus dem Standard entfernen

- Correction Proposals sind kleine Korrekturen am Standard, die Inkonsistenzen beheben oder unklare Formulierungen richtig stellen

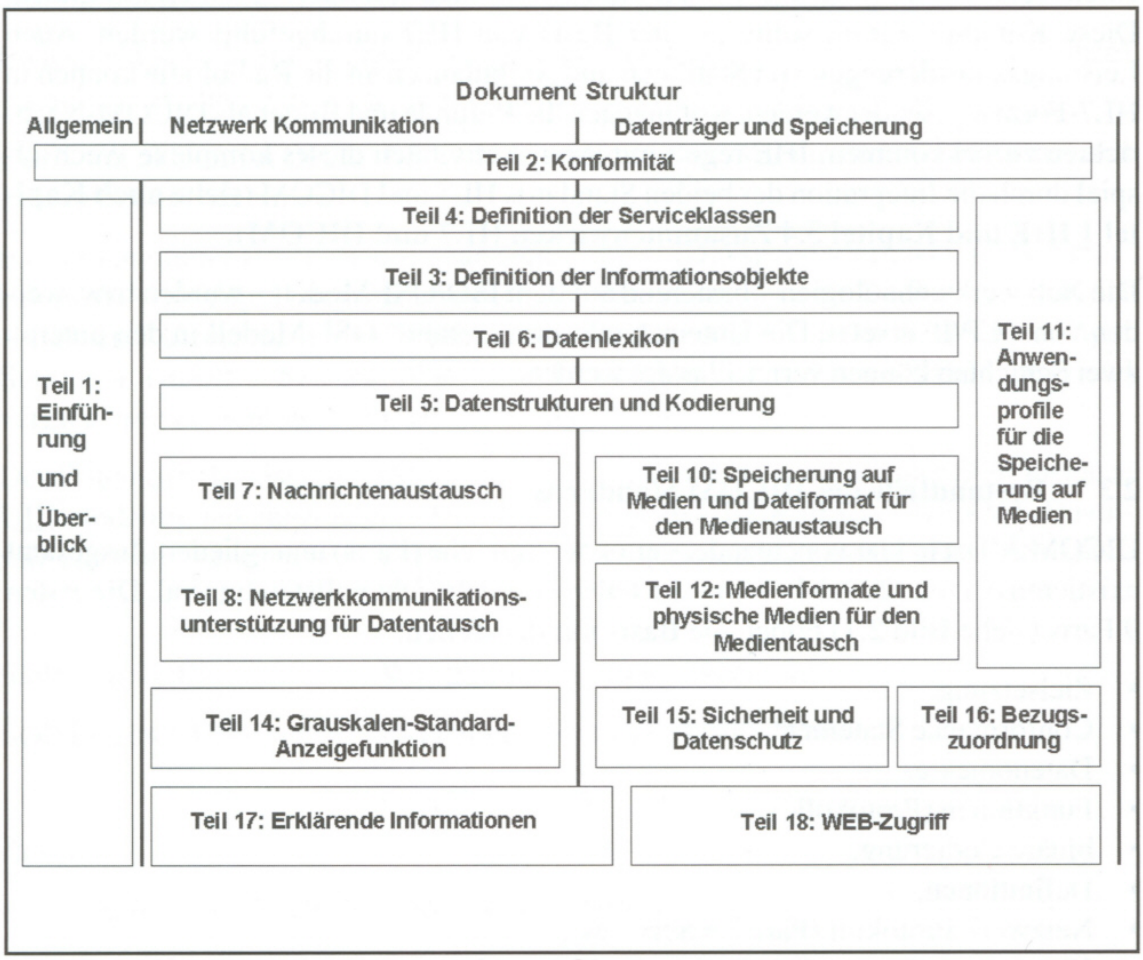

Abbildung 7: Die Bestandteile des DICOM Standards[19]

[19][Gär05] Bild 2.1 Seite 50

Teil 1: Einführung und Überblick

Beschreibung des Entwurfprinzips und Erklärung der verwendeteten Abkürzungen und Fachausdrücke sowie ein grober Überblick über DICOM 3.0.

Teil 2: Konformität

Darstellung der Kriterien, die ein Hersteller erfüllen muss, um sein Gerät DICOM-kompatibel nennen zu dürfen sowie Definitionion der Form der sogenannten Conformance Statements, die jeder Hersteller veröffentlichen muss und die auch in Ausschreibungen als Anforderungsspezifikation verwendet werden können.

Teil 3: Informationsobjekt Definitionen

Die im Standard verwendeteten Informationsobjekte werden hier definiert. Wenn Informationsobjekte ähnliche Eigenschaften besitzen, werden diese in Klassen zusammengefasst, die Klassen werden in die Typen „Normalized“ und „Composite“ eingeteilt.

Teil 4: Serviceklassen Definition

Die mit einem Informationsobjekt durchführbaren Aktionen und Dienstleistungen werden hier definiert. Gleichartige Dienstleistungen bilden eine Serviceklasse. Werden diese Klassen aktiv zur Verfügung gestellt, spricht man von einem Service Class Provider (SCP), werden diese passiv in Anspruch genommen, von einem Service Class User (SCU). Als Beispiel kommunizieren zwei Bildarchive miteinander. Ein Archiv nimmt die Position des SCU ein, indem es dem anderen Archiv, welches als SCP fungiert, ein Bild mit Aufforderung dieses Bild abzuspeichern.

Teil 5: Datenstrukturen und Kodierung

Definitionen der Struktur der Daten und der Ordnung in einem Datensatz für die Übertragung über ein Netzwerk sind hier beschrieben. Der Verbindungsaufbau ist in Teil 7 definiert.

Teil 6: Datenlexikon

Das Datenlexikon ist eine Liste aller gültigen Datenelemente in der miteinander verwandte Elemente in Gruppen zusammengefasst sind. Das Datenlexikon ist eine Übersetzungstabelle die im Standard kodiert sind. Als Beispiel ist der Name des Patienten mit 0010,0010 kodiert.

Teil 7: Nachrichtenaustausch

Definition des Ablaufs und der Protokollstruktur für den Austausch zwischen zwei Geräten. Im Speziellen sind dies die Services DIMSE (DICOM Message Service Element) wie C-Store, C-Move u.a.

Teil 8: Netzwerkkommunikationsunterstützung für Datenaustausch

Hier werden alle notwendigen Systemkomponenten für den DICOM-Nachrichten-Austausch über ein Netzwerk definiert, die sich auf Ebene 5 und 6 des OSI Modells befinden (DICOM Upper Layer). Die Ebenen 1-4 werden hier nicht beschrieben.

Teil 9: Punkt-zu-Punkt Kommunikation

Dieser Teil ist nicht mehr im Standard enthalten.

Teil 10: Speicherung auf Medien und Dateiformat für den Medienaustausch

Hier wird die Art und Weise spezifiziert, wie Daten auf ein bestimmtes Medium geschrieben werden. Teil zehn definiert nur das allgemeine DICOM Speichermedium und das Datenformat, mit dem die Informationsobjekte gespeichert werden. Der Hauptpunkt dieses Teils ist ein Header, der dem Datensatz hinzugefügt wird, der u.a die SOP-Klasse[20] und die Kodierung enthält. Die Teile elf und 12 spezifizieren die Speichermedien selbst, Teil zehn bildet hierfür allerdings die Grundlage.

[20] siehe Kapitel 4.4.4

Teil 11: Anwendungsprofile für die Speicherung auf Medien

Hier werden die Informationen definiert, welche bei bestimmten Applikationen zusätzlich zu den Grundinformationen gespeichert werden müssen, wobei sich die Anforderungen an den klinischen Bedürfnissen orientieren sollen.

Teil 12: Medienformate und physische Medien für den Medienaustausch

Teil 12 spezifiziert physikalische Medien und das medienabhängige Datenformat eines Mediums, das für den Datenaustausch geeignet ist. Dies sind z.B. CD-ROMs, magnetooptische Platten, Bänder und DVD-Medien.

Teil 13: Punkt-zu-Punkt drucken

Dieser Teil ist nicht mehr im Standard enthalten.

Teil 14: Grauskalen-Standard-Anzeigefunktion

Da die Einstellungen, die die optische Erscheinung eines Bildes bestimmen, nicht dauerhaft im Archiv gespeichert werden, wurde ein neues DICOM Objekt mit dem Namen „Presentation State“ definiert, welches einen Parametersatz enthält, wie ein Bild auf dem Monitor dargestellt werden soll. Dieses Objekt enthält nur Referenzen auf das zugrundeliegende Bild und nicht die Bilddaten selbst und ist somit sehr klein, was eine mühelose Speicherung und Übertragung ermöglicht.

Presentation State:

Der eigentliche Datensatz (das Bild) bleibt unverändert, alle Einstellungen am Bildschirm landen in einem getrennten Datensatz. Somit ist es möglich, mehrere Presentation States für ein Bild zu speichern.

Grayscale Standard Display Function:

Diese Funktion gewährleistet eine fast einheitliche Darstellung von Bildern auf unterschiedlichen Medien(Drucker, Bildschirm) durch die Einführung einer genormten Darstellungskurve.

Teil 15: Sicherheit und Datenschutz

Dieser Teil definiert einen Sicherheitsmechanismus, der die Anforderungen bezüglich der Abhörsicherheit sowie der Verifikation von Sender und Empfänger aus dem amerikanischen Datenschutzgesetz (HIPAA) aufnimmt sowie Authentizität über digitale Signaturen und benutzerorientierte Zugriffsrechte ermöglicht.

Teil 16: Bezugszuordnung

Die Bezugszuordnung wird dazu benutzt, Anwenderspezifika in klinischen Konzepten abzubilden. Die Definitionen sind Radiologie-unabhängig und werden durch Vorlagen, sog. Templates, beschrieben.

Teil 17: Erklärende Informationen

Enthält ausführliche Beschreibung spezieller DICOM Objekte oder Tags, die zum Teil aus Teil 3 übernommen wurden.

Teil 18: Web-Zugriff

Spezifikation eines webbasierten Dienstes, der es ermöglicht, DICOM-Objekte mit dem Webbrowser, entsprechende Berechtigungen vorrausgesetzt, weltweit zu betrachten. Zusätzlich wird hierfür ein DICOM-Viewer benötigt. [21]

[21] vgl. [Gär05] Kapitel 2.3 Seiten 49 - 56

4.4.3 Grundzüge der DICOM-Kommunikation

Für eine DICOM-Kommunikation zweier Geräte, im DICOM Kontext als Modalitäten bezeichnet, ist zunächst eine Verhandlung (Association Negotiation) notwendig. Dabei werden folgende Parameter ausgehandelt:

- gewünschte DICOM-Dienste (Service Classes) = Abstract Syntax
- Rollenverteilung: Wer ist SCU und wer SCP
- Kodierung der zu übermittelnden Daten = Transfer Syntax
- Bekanntgabe des AE-Titles[22]

Nach dem erfolgreichen Verbindungsaufbau bildet eine Modalität den Client (Service Class User) und nimmt die ausgehandelte Dienstleistung von der zweiten Modalität, die den Server (Service Class Provider) bildet, in Anspruch.

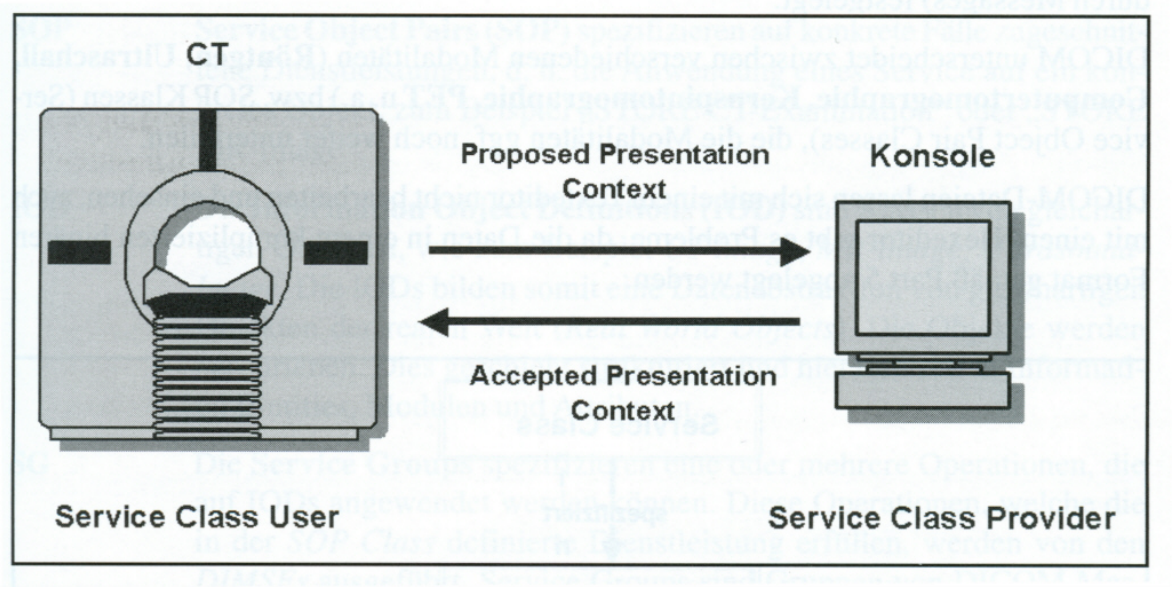

Abbildung 8: Rollenverteilung bei einer DICOM-Kommunikation[23]

[22] AE = Application Entity; Der AE-Title ist der Einstiegspunkt in das Archiv
[23] [Gär05] Bild 2.2 Seite 57

Die Kommunikation beginnt mit einer Liste von Dienstleistungen (Services) und Datendarstellungen (Proposed Presentation Context), welche der User dem Provider vorschlägt. Der Provider antwortet nun mit einer Liste von ihm möglichen und akzeptierten Dienstleistungen und Datendarstellungen (Accepted Presentation Context). Diese „Verhandlung“ (Association Negotiation) ist notwendig, um eine gemeinsame Menge von Dienstleistungen zu vereinbaren. Anschließend können die vereinbarten Dienste genutzt werden, wobei der einfachste Dienst die Verification Service Class ist, mit der ein Verbindungstest durchgeführt werden kann, ob eine DICOM-Kommunikation überhaupt möglich ist. [24]

4.4.4 Das DICOM-Informationsmodell

Im DICOM Informationsmodell werden sowohl die Struktur und Organisation der für die Kommunikation benötigten Informationen als auch die Dienstleistungen, die für die Datenmanipulation notwendig sind, spezifiziert. Die Dienstleistungen, auch als Service Classes bezeichnet, enthalten die Informationen, was mit den Daten passieren soll.

Im DICOM Standard wird zwischen unterschiedlichen Modalitäten (Röntgen, Ultraschall, Computertomographie, Kernspintomographie u.a.) beziehungsweise SOP Klassen unterschieden.

[24] vgl. [Gär05] Kapitel 2.4 Seiten 56 - 57

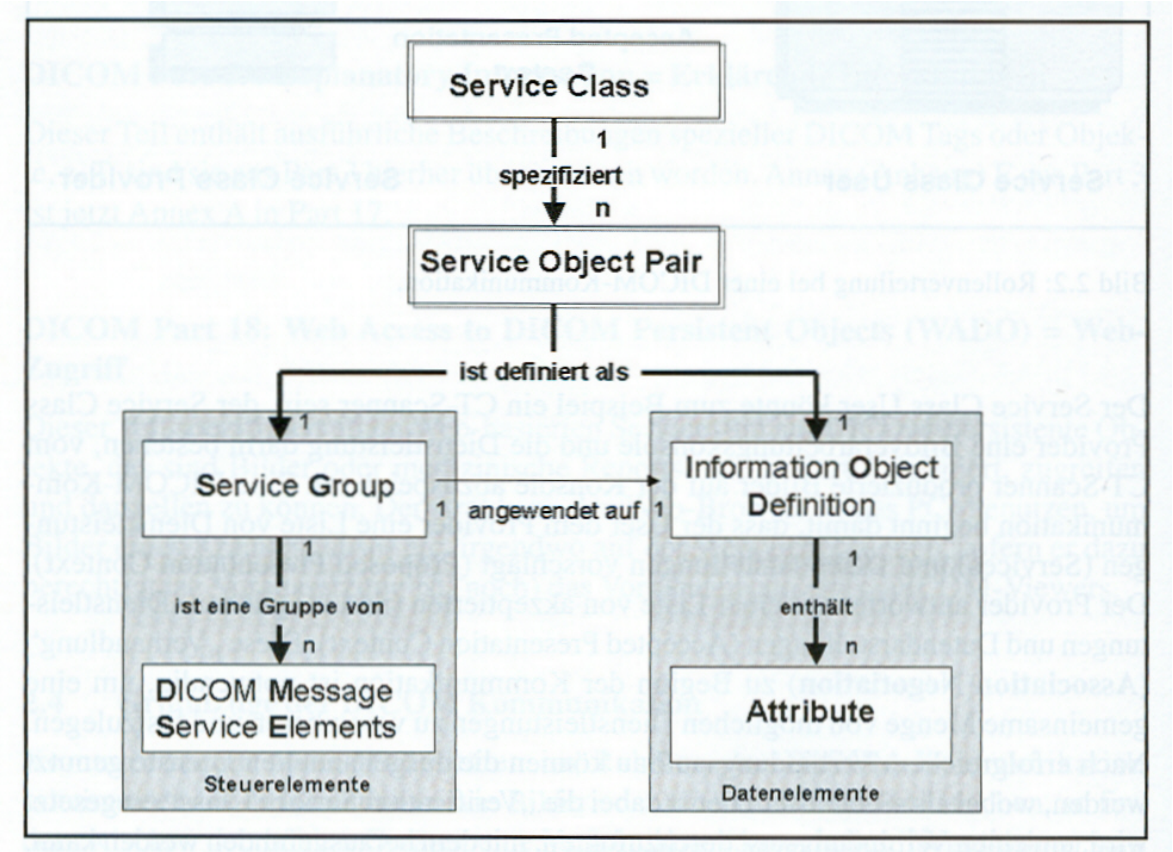

Abbildung 9: DICOM Informationsmodell[25]

Wie in der Abbildung dargestellt enthält das Modell folgende Komponenten, aus denen sich die Datenstrukturen und Messages aufbauen:

- **SC**

  SC steht für **Service Class**. Sie legt die allgemeinen Dienstleistungen fest, die vom Standard unterstützt werden. Beispiele hierfür sind die STORE-Class für die Datenspeicherung oder QUERY-Class zum Anfordern von Informationen.

- **SOP**

  Service Object Pairs sind Unterklassen der SCs und somit definieren sie eine Dienstleistung für ein konkretes Objekt, z.B. STORE MR-Image.

[25] [Gär05] Bild 2.3 Seite 58

- **IOD**

  Information Object Definitions sind Objektdefinitionen von gleichartigen realen Objekten wie CT-Image, MR-Image und stellen somit eine Datenabstraktion dar. Die Definitionen sind strukturiert und hierarchisch in Information Entities, Module und Attribute gegliedert.

- **SG**

  Service Groups definieren Operationen, die auf IODs angewendet werden können. Die Operationen erfüllen die in der SOP Klasse definierte Dienstleistung und werden von den DIMSEs ausgeführt.

- **DIMSE**

  DICOM Message Service Elements bestehen aus Request und Response Operationen und wirken auf IODs. Die Messages bestehen aus einem Befehls- und einem Datenteil. Somit ist eine SOP Class eine präzise Beschreibung einer DICOM Kommunikation, denn in einer SOP Class werden ein oder mehrere DIMSEs mit einer IOD gekoppelt.

Die Service Object Pairs sind das zentrale Element dieses Modells und enthalten neben dem Objekt auch noch die ausführbaren Aktionen dieses Objekts. Jeder Hersteller muss ein Conformance Statement erstellen und veröffentlichen, in dem alle unterstützten SOP-Klassen dokumentiert sind. Für die Kommunikation zweier Geräte müssen beide die gleiche SOP-Klasse unterstützen, allerdings ein Gerät als SCP und das andere als SCU.

Das Datenformat von DICOM

Die DICOM Datenstrukturen basieren auf einem objektorientierten Informationsmodell. Hierfür wird ein Entity-Relationship Modell verwendet, worin die Objekte von den Entitäten dargestellt werden. Der DICOM Standard setzt dieses Modell in Datenstrukturen, wie in folgendem Bild zu sehen ist, um:

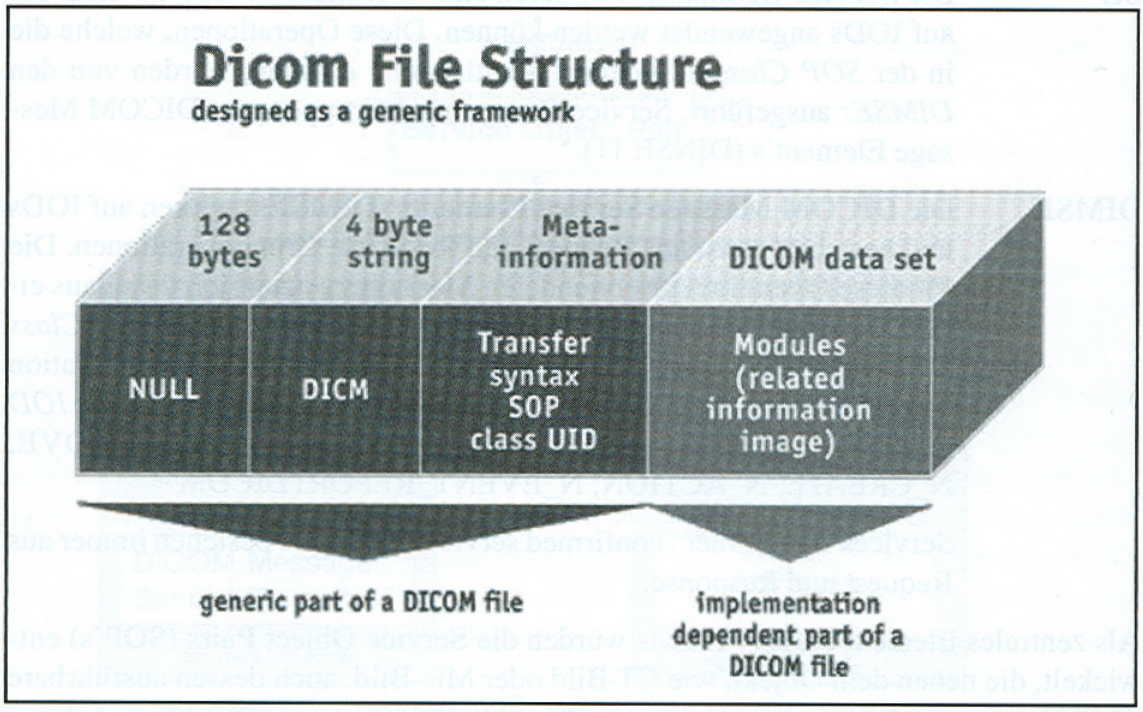

Abbildung 10: DICOM Datenstruktur[26]

[26][Gär05] Bild 2.4 Seite 60

DICOM Datenstrukturen

Information Object Definitions werden durch die Einteilung der Datenelemente in Gruppen, gekennzeichnet durch Gruppennummern, welche aus durch Elementnummern gekennzeichneten Elementen bestehen, beschrieben. Eine Bilddatei besteht aus einem Header und dem eigentlichen Bild. Der Header enthält Informationen über den Patienten, die Untersuchung und die für die Darstellung benötigten Informationen. Der Header wird auch als „Data Set“ bezeichnet.

Es werden folgende Daten benötigt um ein Bild innerhalb einer Untersuchung zu identifizieren und darzustellen:

Gruppennummer	**Inhalt**
0008	Untersuchungsidentifikationsdaten
0010	Patienteninformationen
0018	Aufnahmeinformationen
0020	Informationen über die Beziehung des Bildes zur gesamten Untersuchung
0028	Informationen zur Bilddarstellung
0040	Overlay Informationen, Kommentare, Interpretationen
7FE0	Bildmatrix

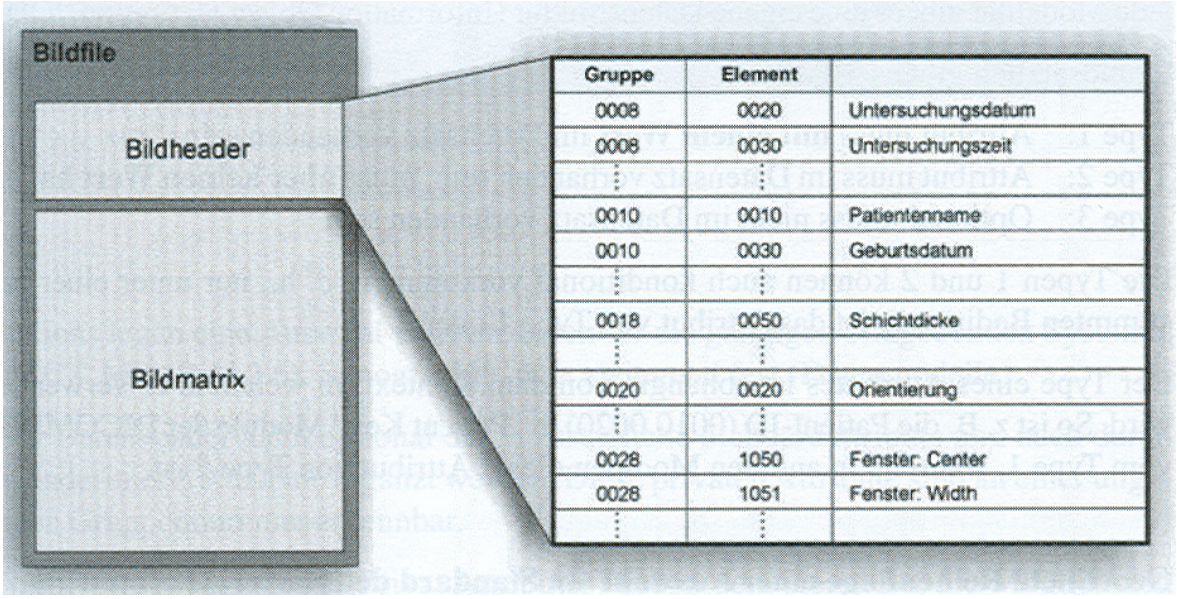

Abbildung 11: Graphische Darstellung eines DICOM-Datensatzes (Bildfile)[27]

Alle Gruppennummern, Elementnummern, Datenformate und Wertebereiche sind im DICOM Data Dictionary spezifiziert. Welche Datenelemente für die Beschreibung einer IOD benötigt werden, ist im Standard genau vorgeschrieben.

Aufbau einer DICOM Bilddatei Ein DICOM-Bild besteht aus einer Liste von Attributen, wobei jedes Attribut aus

- einer eindeutigen Identifikation (attribute tag)
- einem Datentyp (value representation)
- einem Wert

besteht. Die Attribute werden in drei Typen eingeteilt, wobei Typ 1 und 2 konditional sind, also nur unter einer bestimmten Bedingung Typ 1 oder 2 sind. Es gibt folgende drei Typen von Attributen:

[27] [Gär05] Bild 2.5 Seite 61

- Typ 1: Attribut muss mit einem Wert im Datensatz vorhanden sein
- Typ 2: Attribut muss im Datensatz vorhanden sein, aber keinen Wert zugewiesen haben
- Typ 3: Ist optional und muss somit nicht im Datensatz vorhanden sein

DICOM Nachrichten

DICOM Nachrichten enthalten nicht nur Daten, sondern auch Instruktionen über die Verwendung der Daten. Mit diesen Nachrichten werden bei DICOM Daten über Netzwerke ausgetauscht. Die Informationen werden von den im DICOM Standard definierten Informationsobjektklassen aufgenommen, wovon es zwei Typen gibt:

- Normalisierte Informationsobjektklassen enthalten nur die Attribute, die auch in der realen Welt existieren
- Zusammengesetzte Informationsobjektklassen enthalten noch zusätzliche Attribute die mit der realen Welt in Verbindung gebracht werden können aber dort nicht existieren

Eine Nachricht besteht entweder aus einer normalisierten oder zusammengesetzten Informationsobjektklasse. Zusammengesetzte IODs bestehen aus mehreren normalisierten IODs, die zueinander in Relationen stehen. Ein Patient hat zum Beispiel mehrere Serien, Studien und Bilder. Normalisierte Objekte haben keine Relationen.

Aufgrund dieser beiden Objekttypen muss es auch zwei unterschiedliche Dienste geben. Ein C-Dienst muss z.B. einen Datensatz einem bereits vorhandenen Patienten zuordnen, während ein N-Dienst diesen einfach im Puffer eines Druckers speichern kann, ohne auf Abhängigkeiten achten zu müssen.

Wie oben bereits beschrieben werden Nachrichten über sogenannte DIMSEs gesendet. Auch von diesen gibt es zwei Gruppen. Die DIMSE-C Nachrichten sind für folgende Einsatzbereiche:

- Kommunikationsüberprüfung zwischen zwei DICOM-Applikationen
- Bilddatenübertragung
- Bilddatenbankdienste
- Benachrichtigung über Existenz, Inhalt und Standort von Bildern einer Untersuchung

DIMSE-N Nachrichten sind rein für normalisierte Datenstrukturen vorgesehen und decken folgende Einsatzbereiche ab:

- Verwaltung von Patientendaten
- Verwaltung von Untersuchungen
- Verwaltung von Untersuchungsergebnissen
- Ausdruck von Bildern

Die Kommunikationsstruktur von DICOM

Im DICOM Standard werden für die Kommunikation drei Datenkanäle definiert, wobei ein Datenaustausch über jeden dieser Kanäle möglich ist, sofern Sender und Empfänger den gleichen benutzen. Das Wechseln der Datenkanäle ist laut Standard möglich ,ohne die Applikation neu zu implementieren.

<table>
<tr><td colspan="3">Medical Imaging Application</td></tr>
<tr><td colspan="3">DICOM Application Message Exchange</td></tr>
<tr><td>DICOM</td><td>DICOM</td><td>OSI Association Control</td></tr>
<tr><td rowspan="5">Session/
Transport/
Network
(STN)</td><td rowspan="4">Upper Layer protocol
for TCP/IP</td><td>Service Element (ACSE)</td></tr>
<tr><td>OSI Presentation Kernel</td></tr>
<tr><td>OSI Session Kernel</td></tr>
<tr><td>OSI Transport</td></tr>
<tr><td>TCP</td><td>OSI Network</td></tr>
<tr><td>DICOM
Data Link</td><td>IP</td><td>LLC</td></tr>
<tr><td>DICOM
Physical (50-pin)</td><td colspan="2">Standard network physical layers(Ethernet, FDDI, etc.)</td></tr>
<tr><td>Point-to-point
environment</td><td colspan="2">Networked environment</td></tr>
</table>

Abbildung 12: Das DICOM Kommunikationsprotokoll[28]

Im ACR-NEMA Standard in der Version 2.0 war bereits die Kommunikation über Point-to-Point vorhanden. Hierfür gab es, ähnlich dem ISO-OSI Referenzmodell, mehrere Schichten über die die Daten weitergereicht wurden. Der DICOM 3.0 Standard erweiterte die Kommunikationsmöglichkeiten und brachte die Netzwerkfähigkeit mit.

[28] [Gär05] Tabelle 2.4 Seite 69

Als Transportschicht für die Netzwerkkommunikation wurde TCP gewählt, die darunterliegenden Schichten halten sich an das OSI Modell. Die Schichten 5 und 6 realisiert das DICOM Upper Layer Protokoll und die oberste Schicht stellt das eigentliche DICOM Protokoll dar. [29]

4.5 Kryptografie

4.5.1 Symmetrisches Verschlüsselungsverfahren

Bei einem symmetrischen Verschlüsselungsverfahren wird im Gegensatz zu einem asymmetrischen Verfahren der gleiche Schlüssel zum Ver- und Entschlüsseln verwendet. Es gibt zwei unterschiedliche Typen von symmetrischen Verschlüsselungsverfahren.

- Stromchiffrierer
  Ein Stromchiffrierer arbeitet zeichenweise, es wird also immer nur ein Zeichen ver- bzw entschlüsselt.

- Blockchiffrierer
  Der Blockchiffrierer teilt den Text in Blöcke und ver- bzw entschlüsselt blockweise.

Symmetrische Verfahren haben zwei Nachteile. Zum einen verwenden sie den gleichen Schlüssel zum Ver- und Entschlüsseln. Ist der Schlüssel einem Angreifer bekannt, so kann er leicht Informationen fälschen. Das zweite Problem liegt darin, dass im Gegensatz zu asymmetrischen Verfahren der Schlüssel mit übertragen werden muss und dies meist über ein unsicheres Medium(z.B. das Internet) stattfindet. Dieses Problem beheben sogenannte hybride Verfahren, wie das bei IPsec verwendete Diffie-Hellman-Schlüsselaustauschverfahren.

[29] vgl. [Gär05] Kapitel 2.5 Seiten 58 - 70

4.5.2 Diffie-Hellman-Schlüsselaustauschverfahren

Das Diffie-Hellman-Verfahren dient dazu, über einen unsicheren Kanal einen geheimen Schlüssel auszuhandeln, den nur die beiden Partner kennen, ohne dass ein Dritter, der die gesendeten Nachrichten mitliest, den Schlüssel errechnen kann. Der mit Hilfe dieses Verfahrens ausgehandelte Schlüssel wird dann für ein symmetrisches Verschlüsselungsverfahren verwendet. Ohne zusätzliche Schutzmassnahmen ist die einzige Angriffsmöglichkeiten der Man-in-the-Middle Angriff.

Die Funktionsweise des Verfahrens ist verhältnismäßig einfach:

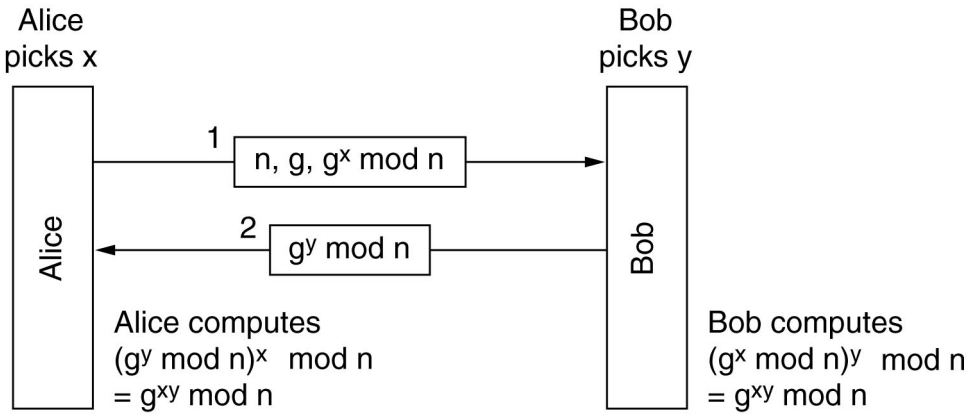

Abbildung 13: Das Diffie-Hellman-Schlüsselaustauschverfahren[30]

1. Alice und Bob einigen sich beide auf zwei große Primzahlen n und g und teilen diese einander mit

2. Beide Kommunikationspartner erzeugen sich jeweils eine Zufallszahl x und y, die jeweils nicht übertragen wird

3. Alice sendet n,g und $g^x \ mod \ n$ an Bob

4. Bob sendet $g^y \ mod \ n$ an Alice

[30][Tan03] Abbildung 8.37 Seite 853

5. Alice berechnet nun $(g^y \ mod \ n)^x \ = \ g^{xy} \ mod \ n$

6. Bob berechnet $(g^x \ mod \ n)^y \ = \ g^{xy} \ mod \ n$

Damit haben Alice und Bob einen gemeinsamen Schlüssel $g^{xy} \ mod \ n$. Die Sicherheit des Verfahrens beruht darin, dass ein Angreifer, der die gesendeten Nachrichten mitgelesen hat, zwar g,g^x und g^y kennt, daraus aber nicht g^{xy} berechnen kann. Dieses Problem wird als Diffie-Hellman Problem bezeichnet und ist verwandt mit dem Diskreten-Logarithmus-Problem. Derzeit existiert kein Algorithmus, der dies zeitnah lösen könnte.

Die einzige Angriffsmöglichkeit, wogegen das Diffie-Hellman-Verfahren nicht immun ist, ist der sogenannte Man-in-the-Middle Angriff. Hierbei verändert ein Angreifer die gesendeten Nachrichten wovon Alice und Bob aber nichts mitbekommen. Alice sendet n,g und

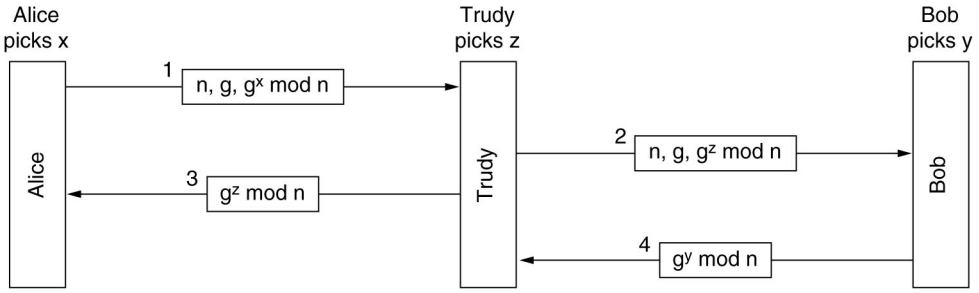

Abbildung 14: Der Man-in-the-Middle Angriff beim Diffie-Hellman-Verfahren[31]

$g^x \ \ mod \ \ n$.Diese Werte erreichen jedoch Bob nicht sondern Trudy modifiziert dies und schickt ihr eigenes $g^z \ mod \ n$ an Bob weiter. Bob antwortet nun wie erwartet mit $g^y \ mod \ n$. Dies modifiziert wiederum Trudy und sendet $g^z \ \ mod \ \ n$ an Alice. Somit hat das Diffie-Hellman-Verfahren zweimal stattgefunden, einmal zwischen Alice und Trudy und einmal zwischen Trudy und Bob. Jedoch gehen Alice und Bob immer noch davon aus, direkt miteinander zu kommunizieren. In Wirklichkeit kann Trudy nun jede Nachricht entschlüsseln, manipulieren und verschlüsselt weitersenden.

[31] [Tan03] Abbildung 8.38 Seite 854

Ein solcher Man-in-the-Middle Angriff kann nur ausgeschlossen werden, wenn die ausgetauschten Nachrichten authentisiert sind. Dies kann mit Hilfe von elektronischen Unterschriften oder Message Authentication Codes geschehen. [32]

[32] vgl. [Tan03] Kapitel 8.7.2 Seiten 852-854

5 Modell einer optimalen Umsetzung

Wie oben bereits erwähnt, konnten einige Dinge wie z.B. die Kopplung der beiden Informationssysteme nicht umgesetzt werden. Aus diesem Grund wird im folgenden ein Modell entwickelt, wie eine Anbindung einer Praxis an eine Klinik optimal aussehen könnte. Dieses Modell basiert auf den Anforderungen, die in Kapitel 3 beschrieben wurden und bildet das Szenario für einen Orthopäden ab. Hierbei steht die Modellierung der Funktion der wesentlichen Punkte im Vordergrund. Die Daten des Modelss erfüllen keinen Anspruch auf Vollständigkeit und dienen rein der Veranschaulichung.

5.1 Anforderungsanalyse

Das nachfolgende Diagramm skizziert die benötigten Funktionalitäten unter den oben beschriebenen Einschränkungen. Die beiden Akteure stehen für aktive bzw. passive Aktionen. System steht für Aktionen, die nach einer Eingabe des Benutzers im Hintergrund automatisiert ablaufen.

Trotz einer Verbindung zwischen Praxis und Klinik sind immer noch zwei eigenständige Informationssysteme vorhanden. Nach erfolgter Einweisung erfolgt die Aufnahme des Patienten im Kliniksystem. Diese Aufnahmen werden registriert und können nun im Praxissystem manuell zugeordnet werden. Zudem muss vom System die Übertragung der entsprechenden Röntgenbilder bewältigt werden, welche mit Hilfe einer DICOM Kommunikation stattfindet.

Der Benutzer muss für die Bildübertragung die gewünschten Bilder des Patienten somit nur noch auswählen. Zudem ist es möglich, alle relevanten Daten vor Ort in der Klinik im Praxissystem zu erfassen. Dies beginnt im Workflow mit der Registrierung der Aufnahme eines Patienten in der Klinik und endet mit dem Erfassen von abrechnungsrelevanten Daten. Während des Klinikaufenthalts können aktuelle Befunde und andere wichtige Daten wie Laborergebnisse erfasst werden.

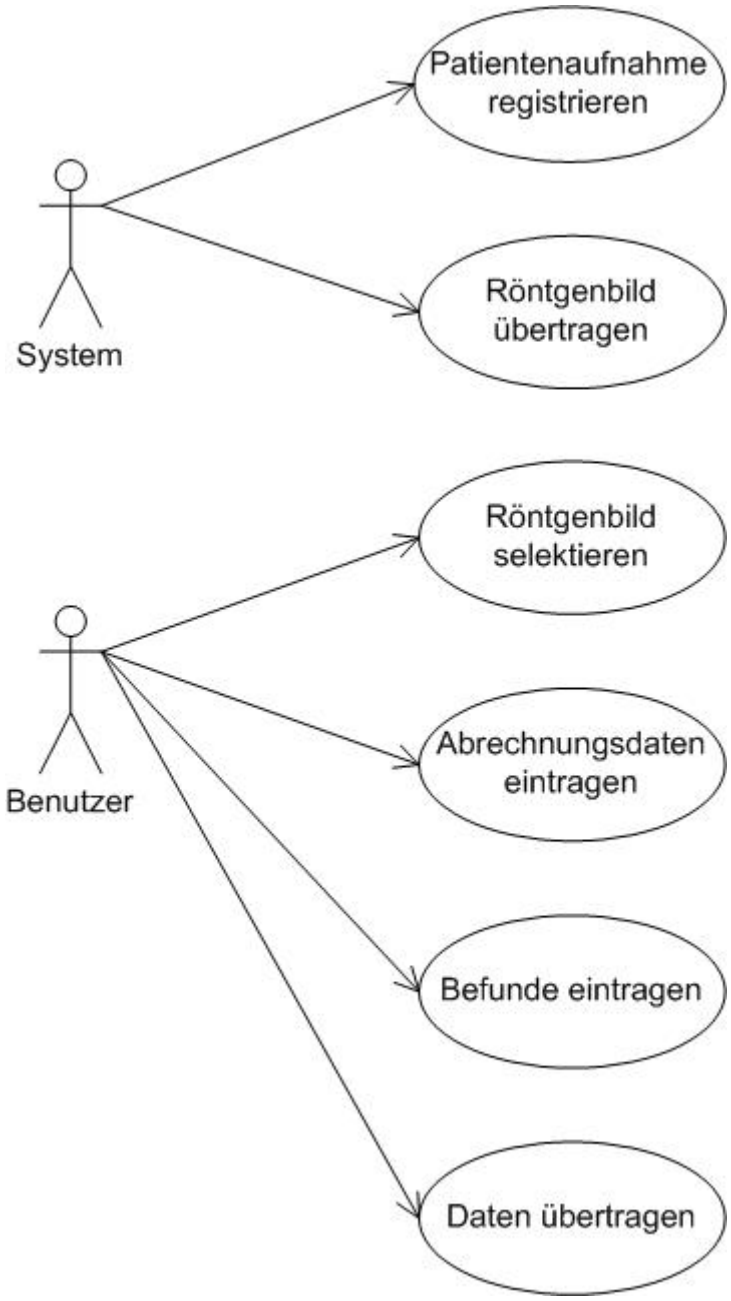

Abbildung 15: Use Case Diagramm

5.2 Spezifikation

Die Spezifikation der Modellsoftware erfolgt mit Hilfe von Datenflussdiagrammen sowie einer Beschreibung der grafischen Benutzeroberfläche.

5.2.1 Datenflussdiagramm

Der Startvorgang des Programms ist in zwei Schritte unterteilt. Im ersten Schritt wird die Konfigurationsdatei ausgewertet, über die der Benutzer das Verhalten des Programms beeinflussen kann. Die entsprechenden Parameter sind in einer einfachen Textdatei gespeichert.

Der zweite Schritt lädt unter Berücksichtigung der im oben beschriebenen Vorgang ausgewerteten Parameter die Modelldaten aus der Datenbank, bereitet die Darstellung vor und zeigt dem Benutzer schließlich die grafische Oberfläche mit den Daten an. Der Ablauf ist in folgendem Diagramm grafisch nocheinmal veranschaulicht.
Zur Laufzeit des Programms gestaltet sich der Datenfluss nach dem immer gleichen Schema.

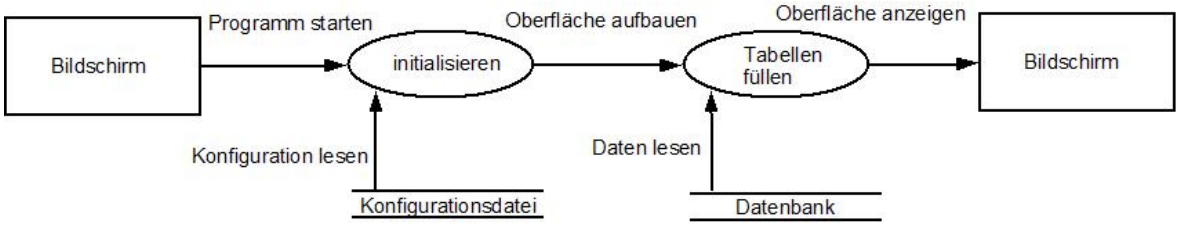

Abbildung 16: Datenfluss während dem Programmstart

Über die grafische Benutzeroberfläche führt der Benutzer eine Aktion aus. Die erforderlichen Daten werden aus der Datenbank gelesen oder es findet auf der Datenbank ein Schreibvorgang statt. Anschließend wird das Ergebnis auf dem Bildschirm ausgegeben.

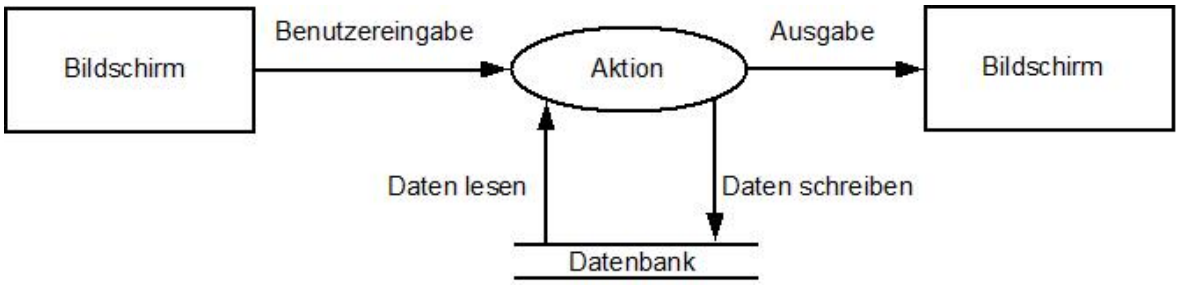

Abbildung 17: Datenfluss zur Programmlaufzeit

5.2.2 Oberflächenbeschreibung

Die grafische Oberfläche des Programms orientiert sich vom Aufbau her an dem üblichen Workflow, der bei der Patientenaufnahme beginnt und mit der Leistungserfassung/Abrechnung endet. Jeder Schritt des Workflow erhält einen eigenen Tab in der Oberfläche. Je nach Aufgabe unterscheidet sich die Gestaltung der Tabs.

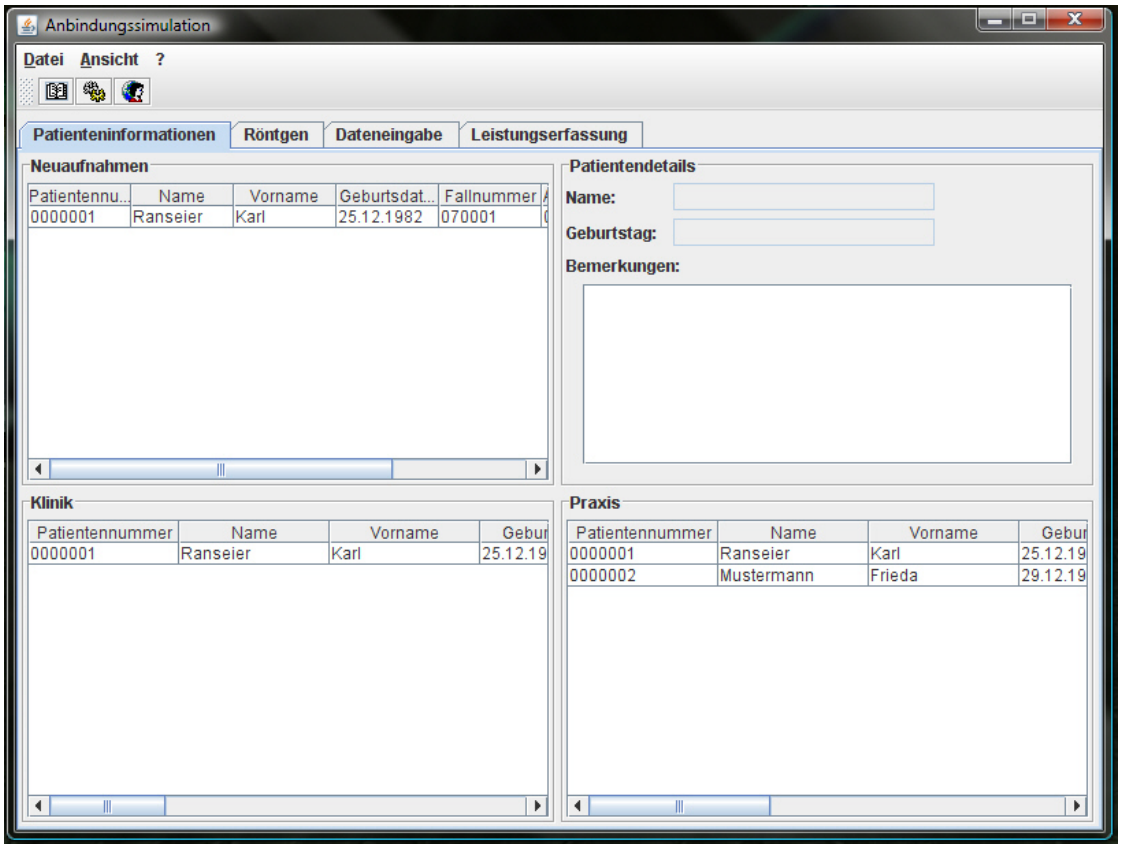

Abbildung 18: Patienteninformation

Der Patienteninfotab ist in vier Teile gegliedert. Die beiden unteren stellen die in der Klinik bzw. in der Praxis vorhanden Patienten dar. Oben links werden die neu aufgenommenen Patienten angezeigt, deren Aufnahme durch eine Funktion in der Praxis registriert werden kann. Aufgrund des oft recht ausführlichen Bemerkungsfeldes des Praxissystems werden oben rechts die Details eines Patienten aus dem Praxissystem angezeigt.

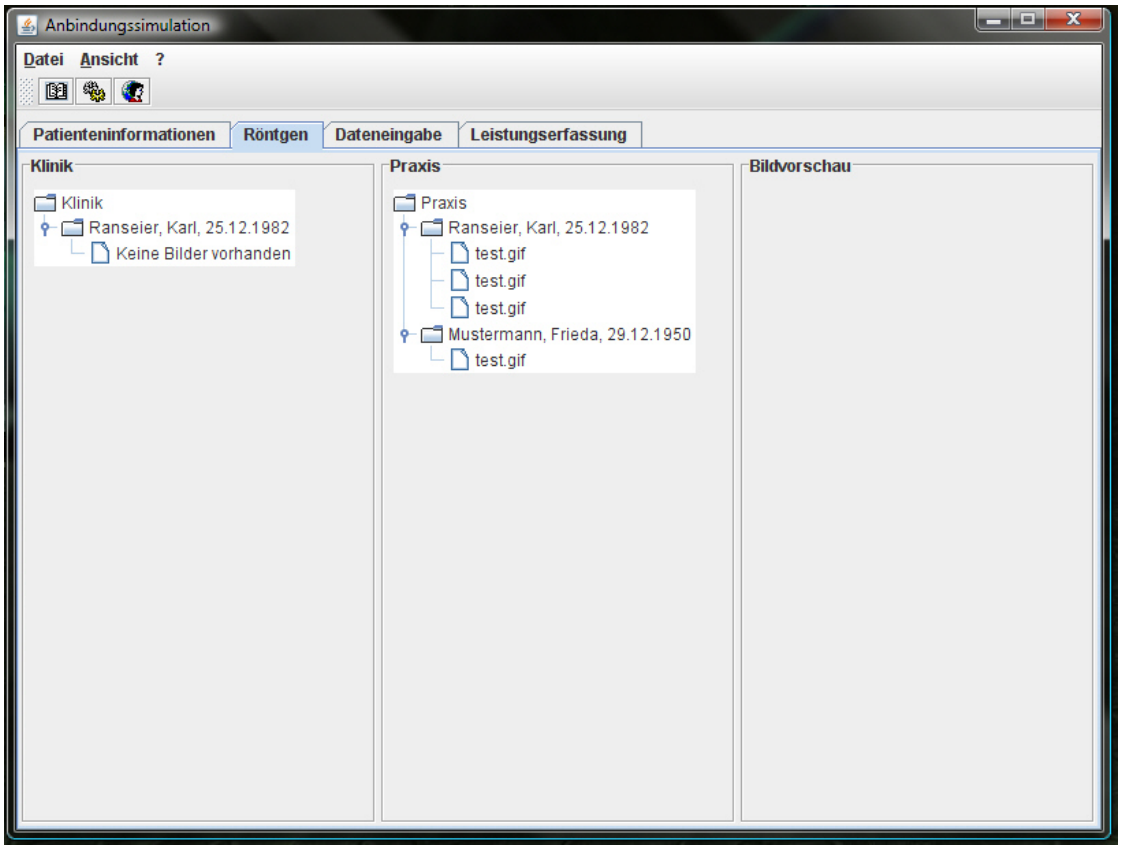

Abbildung 19: Röntgen

Der zweite Tab dient der Übertragung der benötigten Röntgenbilder. Die beiden ersten Panel enthalten jeweils die Patienten der Klinik bzw. der Praxis mit den verfügbaren Röntgenbildern in einer Baumstruktur. Im rechten Teil wid von den Bildern aus der Praxis eine Vorschau angezeigt.

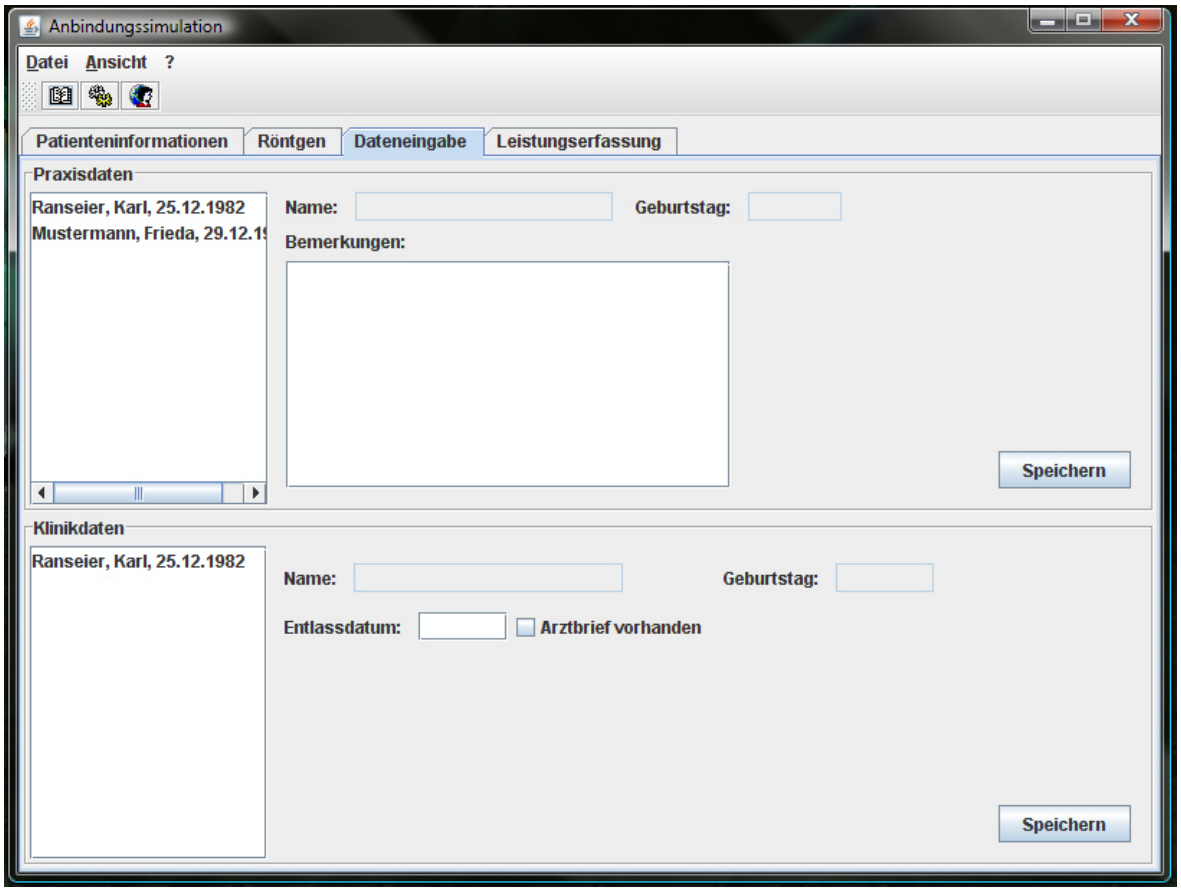

Abbildung 20: Dateneingabe

Der Tab für die Dateneingabe ist horizontal in zwei Teile geteilt. Die obere Hälfte dient der Dateneingabe im Praxissystem, die untere für die Dateingabe im Kliniksystem. Auf der linken Seite werden jeweils die verfügbaren Patienten für Klinik und Praxis in Listenform dargestellt.

Die Leistungserfassung enthält oben links eine Liste mit den verfügbaren Patienten der Praxis. Wird ein Patient ausgewählt, lassen sich neue Leistungen für einen neuen Zeitraum eingeben oder bestehende Eingaben ändern. Über den Button „Neue Leistung“ kann der Tabelle eine neue Zeile hinzugefügt werden.

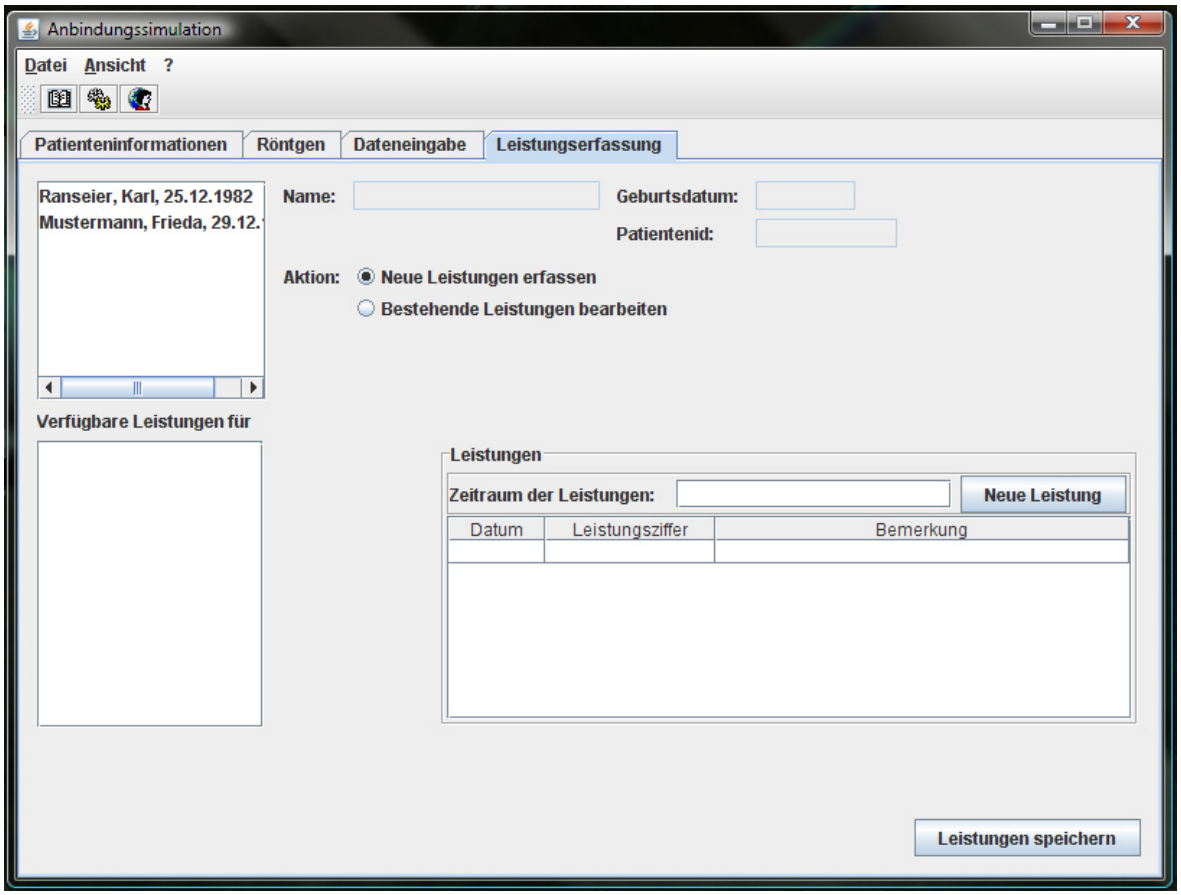

Abbildung 21: Leistungserfassung

5.3 Konstruktion

5.3.1 Komponentendiagramm

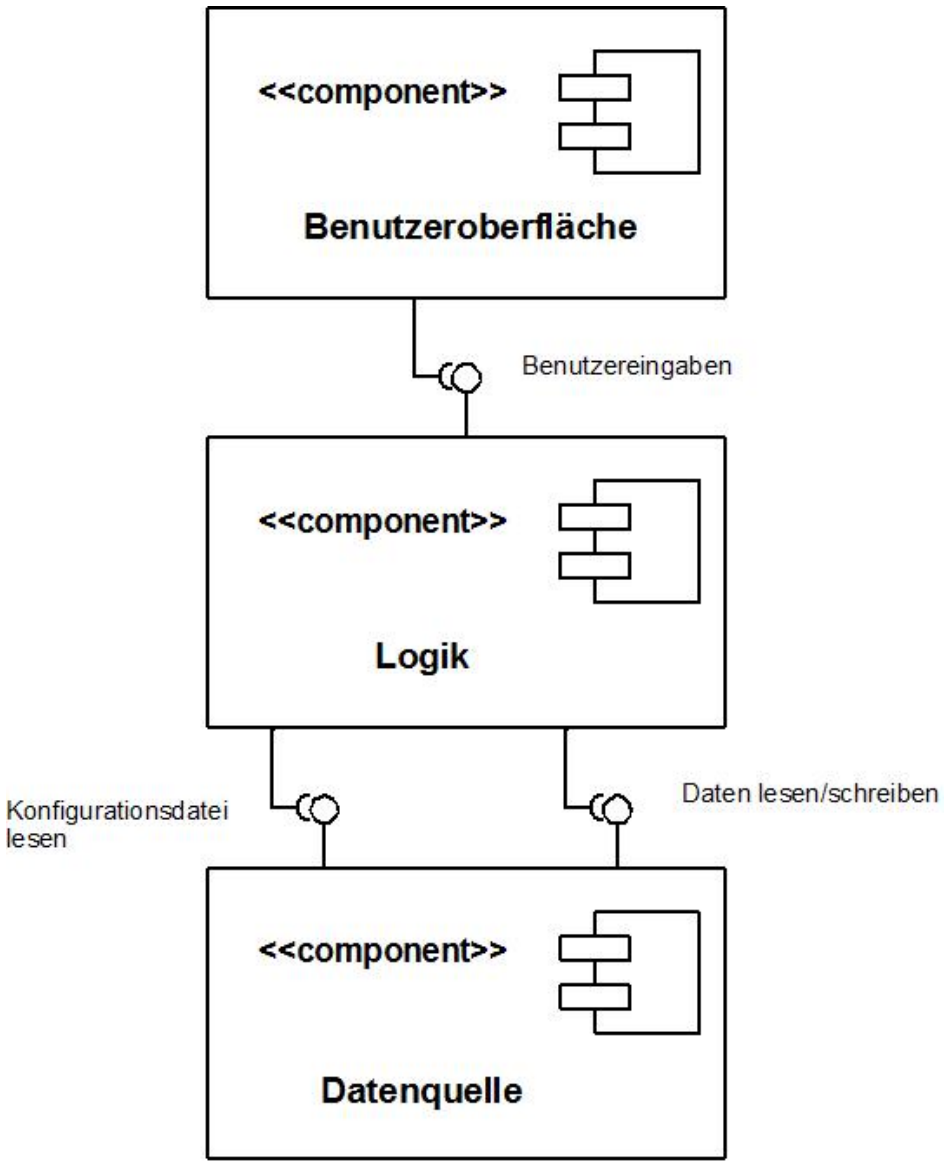

Abbildung 22: Komponenten der Modellsoftware

Die Software ist in einer 3-Schichten-Architektur aufgebaut. In diesem konkreten Fall besteht jede Schicht aus genau einer Komponente. Die Benutzeroberfläche enthält alle Klassen, die für die Darstellung der Daten und die Interaktion mit dem Benutzer zuständig sind. Die Eingaben des Benutzers werden an die Logik weitergegeben, wo falls notwendig, weitere nicht benutzerabhängige Entscheidungen getroffen werden. Anschließend werden die Daten an die unterste Komponente, die Datenquelle, weitergegeben.

Diese führt dann die notwendigen Operationen auf der Datenbank aus. Werden lesende Operationen auf der Datenbank durchgeführt, gehen die Daten den beschriebenen Weg zurück und werden abschließend in der Benutzeroberfläche dargestellt.

5.3.2 Klassendiagramme

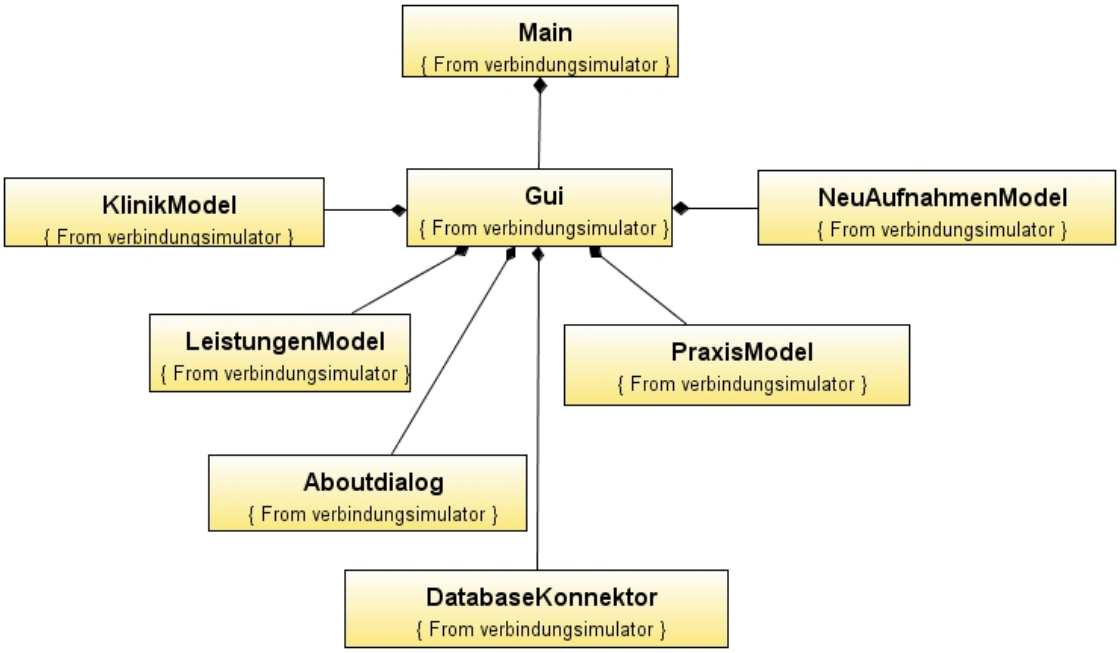

Abbildung 23: Klassendiagramm

Im folgenden werden die Aufgaben der einzelnen Klassen grob skizziert. Eine ausführliche Beschreibung der einzelnen Klassen erfolgt weiter unten.

1. Main

   Die Klasse Main ist für den Programmstart und die Auswertung der Konfigurationsdatei zuständig

2. Gui

   Die Klasse Gui beinhaltet alle Elemente für die Visualisierung und die Funktionalität der Benutzeroberfläche

3. NeuAufnahmenModel

   Diese Klasse definiert das Tabellenmodel für die Neuaufnahmentabelle

4. PraxisModel

   Diese Klasse definiert das Tabellenmodel für die Tabelle der Praxispatienten

5. KlinikModel

   Diese Klasse definiert das Tabellenmodel für die Tabelle der Klinikpatienten

6. LeistungenModel

   Diese Klasse definiert das Tabellenmodel für die Tabelle der Leistungserfassung

7. DatabaseKonnektor

   Die Klasse DatabaseKonnektor erledigt alle Operationen, die auf der Datenbank notwendig sind

8. Aboutdialog

   Die Klasse Aboutdialog erzeugt den Aboutdialog und zeigt diesen bei Bedarf an

Die detaillierten Klassenbeschreibungen befinden sich in Anhang C.

5.3.3 Entity-Relationship Diagramme

Für die Speicherung der Daten kommen zwei getrennte Datenbanken zum Einsatz, um den Charakter zweier getrennter Systeme darzustellen. Die folgenden ER-Diagramme veranschaulichen das Design der beiden Datenbanken. Das Design ist rein für den Zweck des Modells entworfen und erfüllt nicht den Anspruch auf Vollständigkeit für eine echte Anwendung.

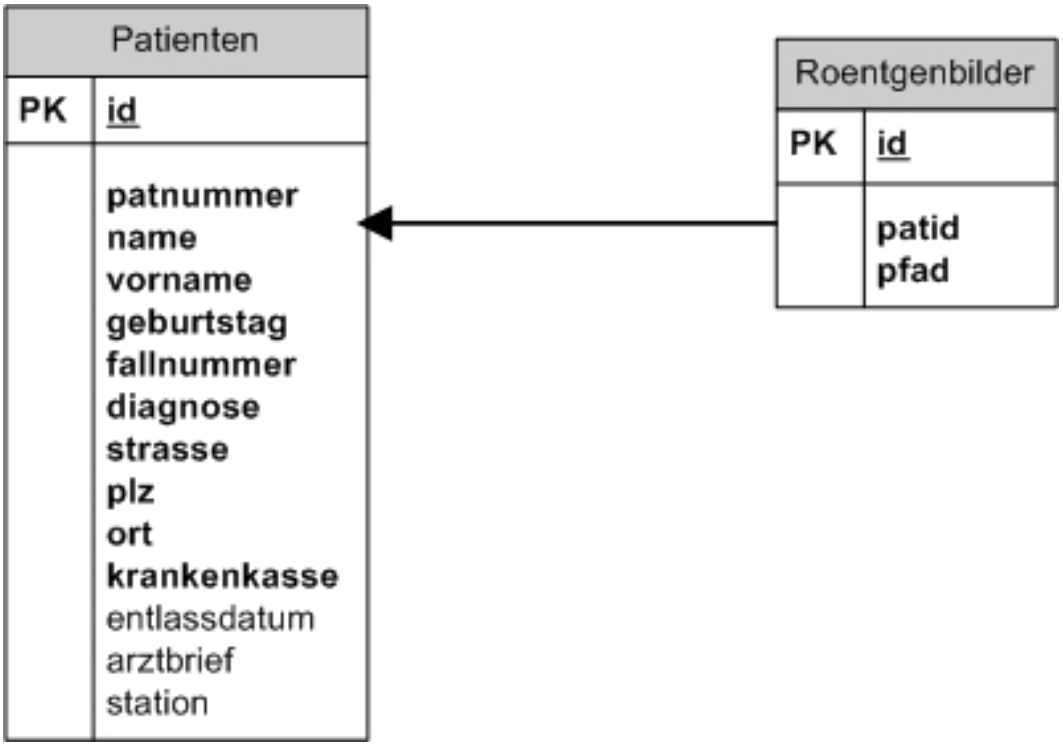

Abbildung 24: Das EntityRelationship Diagramm der Klinikdatenbank

Die Klinikdatenbank enthält zwei Tabellen, zum einen die Patiententabelle, zum anderen die Tabelle für die Röntgenbilder. Die Verknüpfung erfolgt über das Attribut „patnummer“ der Patiententabelle und die „patid“ der Tabelle Roentgenbilder.

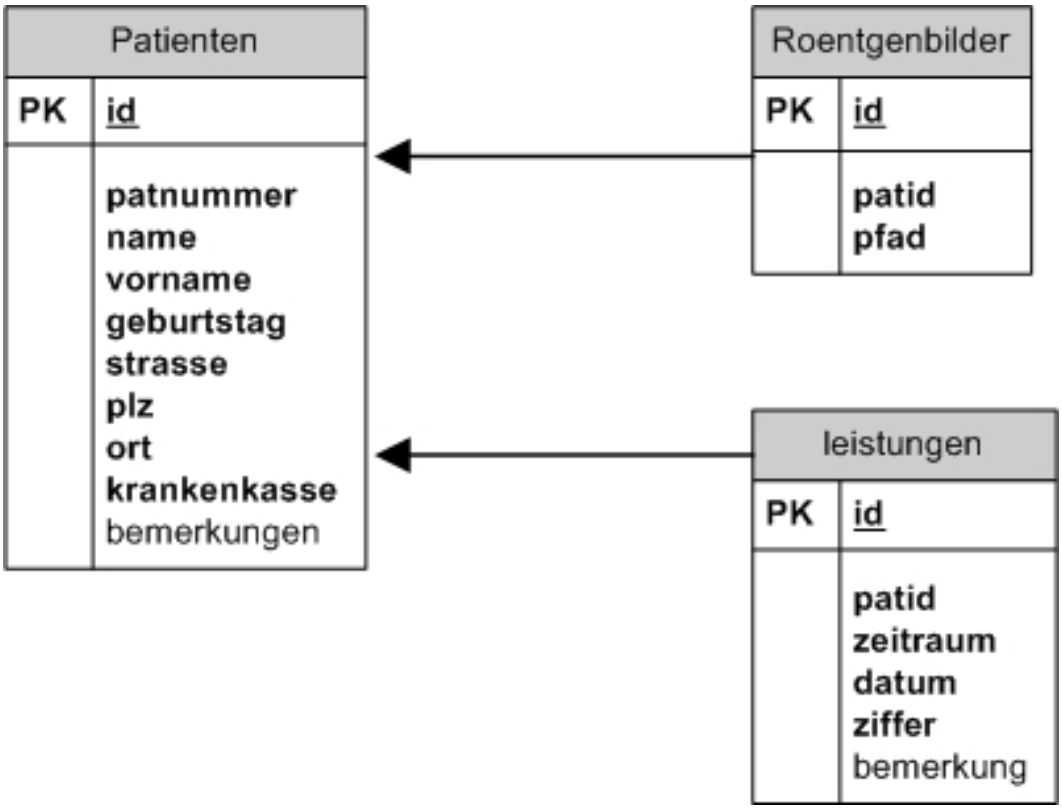

Abbildung 25: Das EntityRelationship Diagramm der Praxisdatenbank

Die Praxisdatenbank enthält noch eine zusätzliche Leistungstabelle für die Leistungserfassung. Die Verknüpfung der Roentgenbilder und Leistungentabelle erfolgt jeweils über das Attribut „patid“ mit dem Attribut „patnummer“ der Tabelle Patienten. Die Attribute der Tabelle Patienten ist geringfügig unterschiedlich zu der der Klinikdatenbank; da in einer Praxis nicht nach Fällen behandelt wird, kann die Fallnummer entfallen.

5.4 Implementierung

5.4.1 Auswertung der Konfigurationsdatei

Die Konfigurationsdatei der Software ist ein sogenanntes Propertiesfile. Java bietet hierfür einen einfachen Mechanismus um eine solche Datei auszuwerten. Die Konfigurationsdatei hat folgendes Aussehen:

```
# Einstellungen für das Modell
# Modelldatenbank neu initialisieren beim Start 0=nein, 1=ja
reset = 1

# Neuaufnahmen automatisch anzeigen 0=nein, 1=ja
neuaufnahmen=1

#Pfad zu den Bildern
pfad=C:\\images\\
```

Die Zeilen die mit einem '#' beginnen, sind Kommentarzeilen und beschreiben jeweils die Funktion des Parameters und die möglichen Werte. In der jeweils folgenden Zeile steht der eigentliche Parameter mit seinem Wert. Diese werden von der Mainmethode der Klasse Main ausgelesen:

```
properties.load(new FileInputStream("einstellungen.properties"));
reset = properties.getProperty("reset");
```

Einmalig wird die Konfigurationsdatei in das Propertiesobjekt properties geladen. Anschließend wird für jeden Konfigurationsparameter die Methode getProperty mit dem Parameternamen aufgerufen. Diese liest den Wert aus und weist ihn der entsprechenden Variable zu.

Im folgenden wird am Beispiel der Praxisdateneingabe der Weg der Daten von der Oberfläche bis zur Datenbank vorgestellt. Die restlichen Funktionen im Programm arbeiten nach dem gleichen Schema.

5.4.2 ListSelection Listener am Beispiel der Praxisdateneingabe

Die actionPerformed()-Methode eines ListSelection Listeners wird immer dann aufgerufen, wenn ein Eintrag der Liste selektiert wird, in diesem Fall ein Patient der Praxis.

```
String patient = (String) praxPatList.getSelectedValue();
String[] elemente = patient.split(", ");
namensFeld.setText(elemente[0] + ", " + elemente[1]);
gebtagFeld.setText(elemente[2]);
bemerkungenFeld.setText(new DatabaseKonnektor().
getBemerkungenPraxis(elemente[0],elemente[1],elemente[2]));
```

Da in der Liste der Patient mit Name, Vorname und Geburtsdatum aufgeführt wird, können diese Daten direkt aus der Liste in die entsprechenden Felder geschrieben werden. Hierfür wird der selektierte Patient aus der Liste gelesen, der String jeweils nach der Zeichenfolge „, “ gesplittet und anschließend die einzelnen Teile in die Textfelder geschrieben. Die nun noch fehlenden Daten, in diesem Beispiel das Bemerkungsfeld des Patienten, werden aus der Datenbank nachgeladen. Zur Identifikation eines Patienten sind in dieser Modellsoftware Name, Vorname und Geburtsdatum ausreichend, in der realen Welt muss das nicht immer der Fall sein.

5.4.3 ActionListener am Beispiel des Speichern-Buttons der Praxisdateneingabe

Die im folgenden Snippet abgebildete actionPerformed()-Methode des ActionListeners wird bei jedem Klick auf den Speichernbutton aufgerufen.

```
String name = namensFeld.getText();
String gebtag = gebtagFeld.getText();
String bemerkungen = bemerkungenFeld.getText();
DatabaseKonnektor db = new DatabaseKonnektor();
db.setBemerkungenPraxis(name,gebtag,bemerkungen);
```

Für den Speichervorgang werden zuerst die einzelnen Textfelder ausgelesen und deren Inhalt in Variablen gespeichert. Diese werden anschließend der entsprechenden Methode des DatabaseKonnektors übergeben, die den eigentlichen Speichervorgang in der Datenbank ausführt.

5.4.4 Speichern-Methode für die Praxisdateneingabe

Die oben aufgerufene Methode setBemerkungenPraxis() des DatabaseKonnektors führt den eigentlichen Speichervorgang aus.

```
Connection con = connectPraxis();
String[] namen = name.split(", ");
 try {
    Statement stmt = con.createStatement();
    stmt.executeUpdate("update patienten set bemerkungen=\'"+
    bemerkungen+"\' where name=\'"+namen[0]+"\' and vorname=\'"
    +namen[1]+"\' and geburtstag=\'"+gebtag+"\'");
    con.close();
 } catch (SQLException ex) {
    ex.printStackTrace();
```

Zuerst wird in diesem Fall die Verbindung zur Praxisdatenbank hergestellt. Anschließend muss der aus Name und Vorname bestehende Übergabeparameter wieder in Name und Vorname gesplittet werden. Um nun ein SQL-Statement absetzen zu können, muss zuerst ein Statement erzeugt werden, auf welchem dann ein Update ausgeführt wird, welches das SQL Statement enthält. Nach erfolgter Ausführung wird die Verbindung wieder geschlossen und der Vorgang ist beendet.

6 Theoretische Lösungsmöglichkeiten

In diesem Kapitel werden die theoretisch möglichen Lösungen für das Projekt beschrieben. Diese wurden während der Planungsphase umfassend diskutiert.

6.1 Verbindung zwischen Praxis und Klinik

Für den Aufbau der Verbindung müssen hohe Sicherheitsanforderungen erfüllt werden, da es um die Übertragung von medizinischen Daten geht. Dadurch ist die Auswahl an Möglichkeiten stark begrenzt. Da die Verbindung der beiden Häuser der Kliniken Hochfranken bereits mit Hilfe eines VPN-Tunnels realisiert wurde und somit die technischen Vorraussetzungen vorhanden sind, auch die Praxis über einen VPN-Tunnel anzubinden. Wichtig hierbei ist eine den Anforderungen entsprechende starke Verschlüsselung der Verbindung. Auf Praxisseite wird neben einem DSL Anschluss noch ein zur Klinikhardware kompatibler VPN Router benötigt.

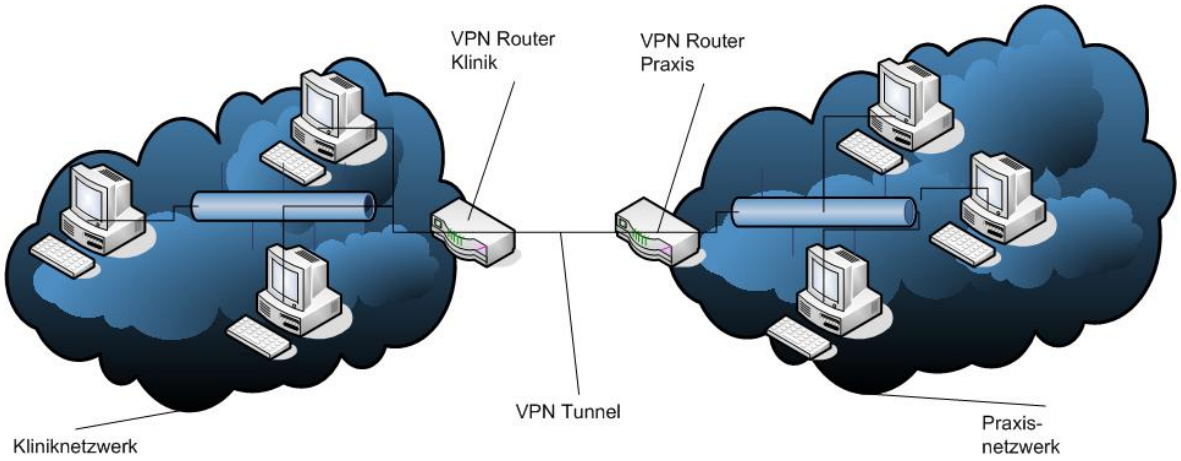

Abbildung 26: Der grobe Aufbau der Verbindung

6.2 Informationssysteme

Die Verbindung der jeweiligen Informationssysteme ist nicht ganz einfach. Dies liegt in der unterschiedlichen Art und Weise, wie die Patienten in den Systemen gepflegt werden. In der Klinik wird nach Fällen gemäß DRG[33] abgerechnet, wofür die Patienten auch nach Fällen geordnet gespeichert sind. Dies ist in der Praxis nicht der Fall, wodurch das Problem einer direkten Kopplung über Schnittstellen in der Identifikation des Patienten liegt.

Als zweite Lösung kommt ein Terminalserver in Frage, auf welchem ein Medistarclient installiert wird. Somit bleibt das Praxisinformationssystem zentral in der Praxis administrierbar. In der Klinik ist keine zusätzliche Softwareinstallation notwendig, denn einen Terminalserviceclient liefert Microsoft Windows bereits mit. Bei eventuellen Verbindungsabbrüchen bricht lediglich die Sitzung ab, die Praxissoftware hat nach wie vor ihren Kontakt zum Server und muss keine eigene Absicherung mitbringen. Der Zugriff auf den Terminalserver kann sowohl von einem festen Punkt im Arztzimmer erfolgen als auch von einem Notebook am Patientenbett. Hierfür ist entweder eine Wireless Lan Infrastruktur notwendig oder es wird eine anderweitige, flächendeckende Verbindung verwendet, wie z.B. UMTS.

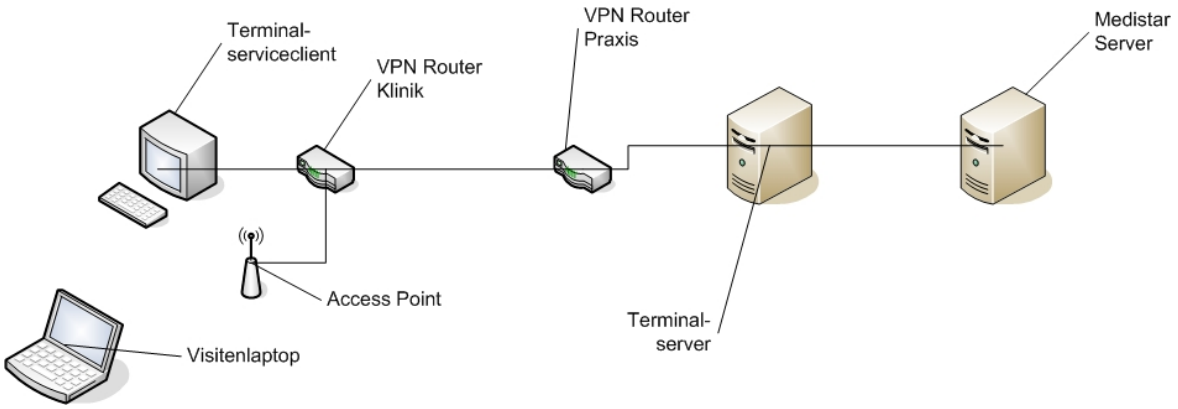

Abbildung 27: Die Terminalserverlösung für Medistar

Eine ähnliche Lösung, die allerdings keinen Terminalserver benötigt, ist die Verwendung des Clients, der auch auf den Praxis-PCs verwendet wird. Da die Verbindung zwischen Praxis und Klinik ein VPN Tunnel ist, ist dies möglich, da die VPN Technik transparent ist und von der Topologie eine Erweiterung des Praxisnetzwerkes darstellt. Bei dieser Variante ist

[33] DRG = Diagnosis Relatad Groups

der VPN Tunnel die Schwachstelle. Falls der Medistarclient über keine Absicherung bei Verbindungsabbrüchen verfügt, können Datenverluste auftreten, die unter keinen Umständen auftreten dürfen.

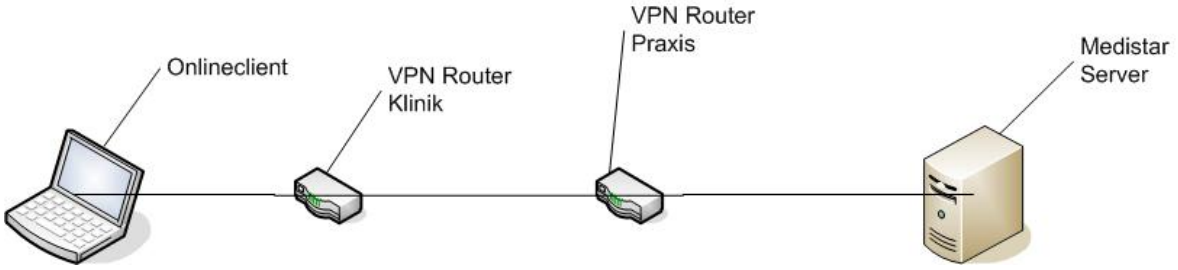

Abbildung 28: Lösung mit Onlineclient

Die letzte Möglichkeit besteht in der Verwendung des bisherigen Offlineclients. Somit würde sich an der Arbeitsweise wenig ändern, nur der Transport des Gerätes in die Praxis würde wegfallen, da ein Datenabgleich nun über die VPN Verbindung erfolgen kann. Diese Option stellt allerdings nur eine Minimallösung dar.

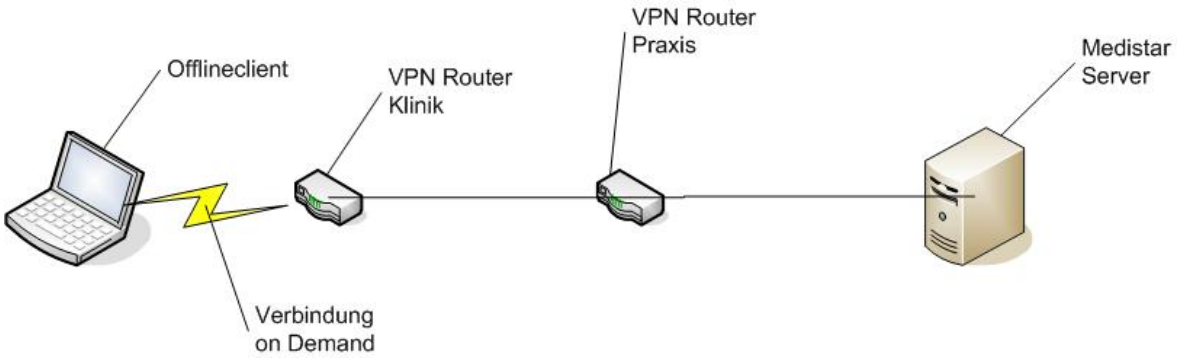

Abbildung 29: Lösung mit Offlineclient

6.3 Röntgenbilder

Um die Röntgenbilder der Praxis in der Klinik verfügbar zu machen, gibt es insgesamt drei Optionen. Diese unterscheiden sich insbesondere in der Belastung der DSL Leitung sowie dem Arbeitsaufwand, der zur Realisierung notwendig ist.

Die erste Möglichkeit belastet die Leitung am wenigsten und ist auch von der Geschwindigkeit die schnellste Option. Auf dem Terminalserver wird ein Viewer installiert, über den die Bilder betrachtet werden können. Die Vorteile hierbei sind die zentrale Positionierung der Komponenten für den Fernzugriff auf einer Maschine, was den Administrationsaufwand sehr gering hält sowie die geringe Belastung der DSL-Leitung, die auf der Uploadseite keine große Bandbreite bietet. Zudem können die Röntgenbilder von dem Bildviewer sehr schnell über das interne LAN geladen werden. Ein Problem dieser Lösung könnte die Bildqualität der Röntgenbilder sein, da ein Terminalserver die Daten, die über die Leitung übertragen werden, stark komprimiert.

Wie in der Abbildung zu sehen ist, ist für diese Lösung in der Klinik nur ein PC mit Netzzugriff notwendig, die restlichen Komponenten befinden sich zentral auf der Praxisseite, somit stellt der Terminalserver den „Single point of failure“ dar.

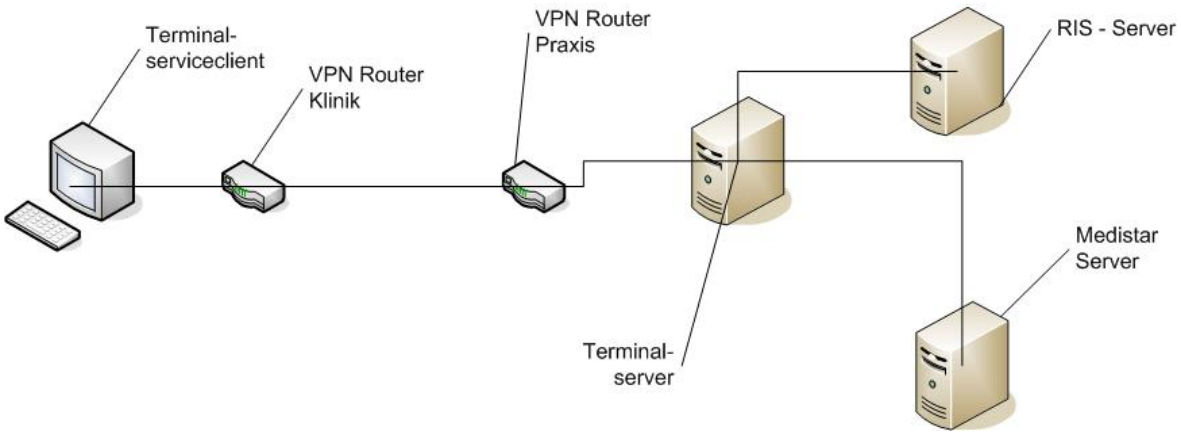

Abbildung 30: Die Terminalserverlösung im Überblick

Die beiden anderen Optionen belasten die Leitung wesentlich mehr als die gerade beschriebene Möglichkeit, da bei diesen beiden eine direkte Übertragung der Bilddaten zur Klinik hin stattfindet. Hierfür ist auf Seiten der Praxis eine DICOM SEND Schnittstelle notwendig, um die Daten Richtung Klinik senden zu können. Auf Klinikseite muss dann entsprechend ein Empfänger vorhanden sein, der die gesendeten Daten verarbeiten kann. Dieser Empfänger kann das PACS der Klinik oder ein PC im Arztzimmer sein.

Allerdings bietet das PACS einige Vorteile gegenüber der PC Lösung. Die Bilder sind mit dem Patienten verknüpft und somit einfach über das Krankenhausinformationssystem oder den Webbrowser abrufbar. Zudem besteht die Möglichkeit, diese auch auszudrucken. Ein weiterer Vorteil ist, dass über diesen Weg auch eine Übertragung von der Klinik in die Praxis stattfinden kann. Der Nachteil an diesen beiden Optionen ist, dass ein größerer Aufwand getrieben werden muss. Es muss vorab festgelegt werden, welche Bilder am nächsten Tag in der Klinik benötigt werden um diese dann auch zeitgerecht übertragen zu können. Zudem ist eine Abstimmung mit der Klinik notwendig, damit dort bekannt ist, dass Bilder für eine bestimmte Abteilung ankommen werden.

Dieses Prozedere ist notwendig, da die Praxis nur über eine asymmetrische DSL Leitung verfügt und die Bilder somit nur mit Uploadgeschwindigkeit Richtung Klinik übertragen werden können.

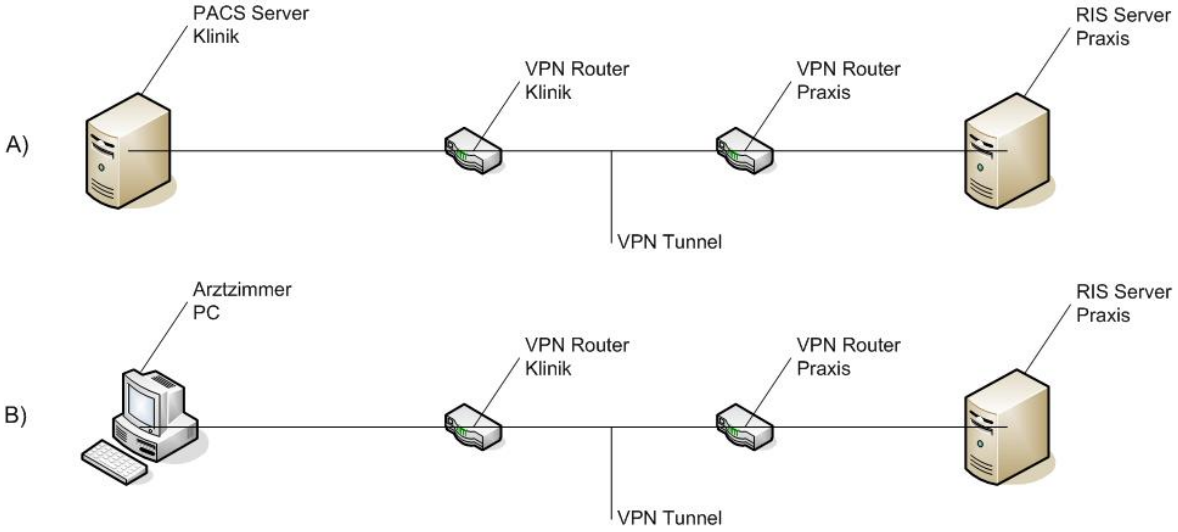

Abbildung 31: a) PACS Lösung b) Speichern auf PC im Arztzimmer

7 Umsetzung

Die oben beschriebenen Alternativen wurden ausführlich diskutiert und auf ihre Realisierbarkeit überprüft. Am Ende fiel die Entscheidung auf jeweils einen Lösungsansatz. Im folgenden werden die getroffenen Entscheidungen sowie die Gründe dafür skizziert. Anschließend werden die umgesetzten Lösungsansätze ausführlich dargestellt.

7.1 Verbindung zwischen Praxis und Klinik

Für die Realisierung der Verbindung gab es von vornherein nur eine Möglichkeit. Diese besteht aus dem Aufbau eines VPN-Tunnels, dessen Endpunkte die Klink in Münchberg auf der einen Seite, die Praxis in Hof auf der anderen Seite bilden. Für die Verschlüsselung der Verbindung kommt IPsec zum Einsatz. Als Algorithmus wird der AES mit einer Schlüssellänge von 256bit verwendet. Als Authentifizierung der beiden Tunnelendpunkte wird ein Preshared Key verwendet.

Auf der Praxisseite muss ein Lancom VPN Router zum Einsatz kommen. Dies hat zum einen technische Gründe, da in der Klinik die Infrastruktur mit Lancom Geräten existiert und durch den Einsatz eines Lancom Routers in der Praxis eine maximale Kompatibilität und somit auch Stabilität der Verbindung gewährleistet ist. Zum anderen stehen hinter dieser Entscheidung Sicherheitsaspekte. Durch den Aufbau einer VPN Verbindung wird ein Zugang zum Kliniknetz geschaffen. Dieser bietet prinzipiell eine Möglichkeit, in das Kliniknetz einzudringen. Damit nur die gewünschten Anwendungen und nicht mehr zugreifen können, wird die VPN Verbindung von der Klinik verwaltet und auch gewartet. Somit hat kein Externer Zugriff auf die Router, um dort Manipulationen durchzuführen.

Seit 6. Juni 2007 sind die Klinik und die Praxis miteinander über einen VPN-Tunnel miteinander verbunden. Die Konfiguration wird im Folgenden näher erläutert.

Den Tunnelendpunkten wurden, wie folgender Grafik zu entnehmen ist, die IP-Adressen 192.168.1.252 auf Klinikseite und 192.168.100.20 zugewiesen. Da die Praxis über keine feste IP-Adresse verfügt, wird der Tunnel immer von Praxisseite aus automatisch aufgebaut, sobald eine Trennung durch den Provider erfolgt.

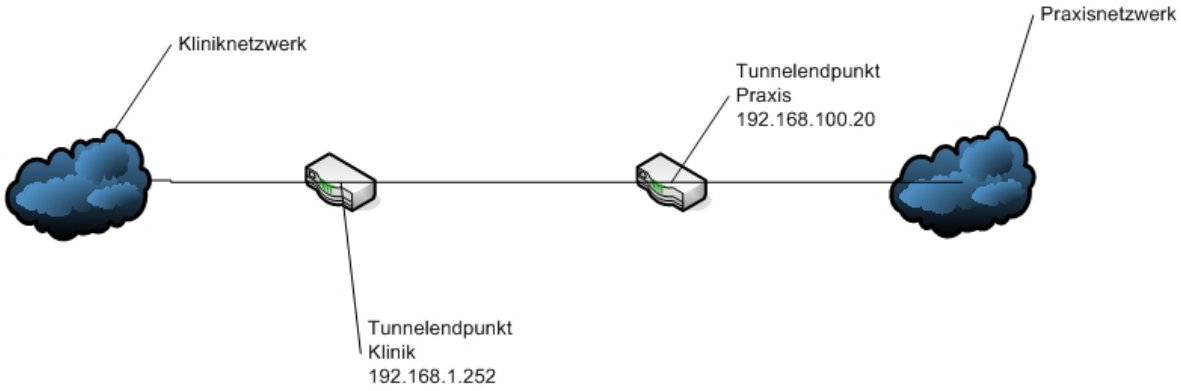

Abbildung 32: VPN Tunnel

In der Praxis existiert eine Infrastruktur für das Wartezimmer TV. Dies ist ein Dienst, der in den Wartezimmern der Praxis ein jeden Tag abwechselndes Filmprogramm abspielen lässt. Hierfür sind folgende Firewallregeln notwendig:

Port	**IP-Adresse**	**Richtung**
21	192.168.100.21, 192.168.100.22, 192.168.100.23	ein- und ausgehend
33	192.168.100.21, 192.168.100.22, 192.168.100.23	ein- und ausgehend
5900	192.168.100.21	eingehend
5901	192.168.100.22	eingehend
5902	192.168.100.23	eingehend

Dieses Regelwerk ist notwendig, damit sich diese drei PCs bei Wartezimmer TV anmelden können und jeden Tag ein neues Filmmaterial aufgespielt werden kann, welches dann in einer Schleife den gesamten Tag abgespielt wird.

7.2 Übertragung der Röntgenbilder

Die Terminalserverlösung ist die kostengünstigste für die Praxis, da nur die Kosten für den Terminalserver anfallen und am Radiologieinformationssystem (RIS) der Praxis keine zusätzlichen Module benötigt werden. Der zusätzliche Wartungsaufwand wäre sehr gering, da er sich rein auf den Terminalserver beschränkt. Dieser Lösungsansatz konnte aber aus einem Grund nicht realisiert werden. Der Hersteller des RIS kann die Befundqualität der Bilder über eine Terminalserververbindung nicht sicherstellen, da der Terminalserver die Bilddaten für bessere Performance komprimiert.

Aus diesem Grund fiel die Entscheidung für die Übertragung der Röntgenbilder in das PACS der Klinik Münchberg. Die Möglichkeit, die Bilder auf einem PC im Arztzimmer zu speichern würde einen zu großen Aufwand bedeuten bezüglich zusätzlicher Softwareinstallation sowie der hausweiten Verfügbarkeit. Auf Praxisseite muss hierfür ein Modul vorhanden sein, welches die DICOM Sendfunktion beinhaltet. Dieses Modul muss im RIS der Praxis noch freigeschalten werden, da der Hersteller die verfügbaren Funktionalitäten mit Lizenzschlüsseln steuert. Somit ist keine zusätzliche Softwareinstallation im herkömmlichen Sinne zu erledigen.

Für den umgekehrten Weg, die Übertragung der postoperativen Bilder, die in der Klinik angefertigt werden, wird die DICOM Receivefunktion im RIS der Praxis benötigt. Die Installation und Einrichtung ist identisch zu dem der oben beschriebenen Sendfunktion.

Wie in folgender Grafik gezeigt, sind an der Übertragung die beiden Bildserver sowie ein Arbeitsplatz der Radiologie in der Klinik beteiligt. Der Arbeitsplatz auf Klinikseite ist notwendig, da das Klinik PACS nicht direkt sendet, sondern diese Aufgabe ein Client übernimmt, der erst die Bilder vom Server lädt und diese dann in die Praxis sendet.

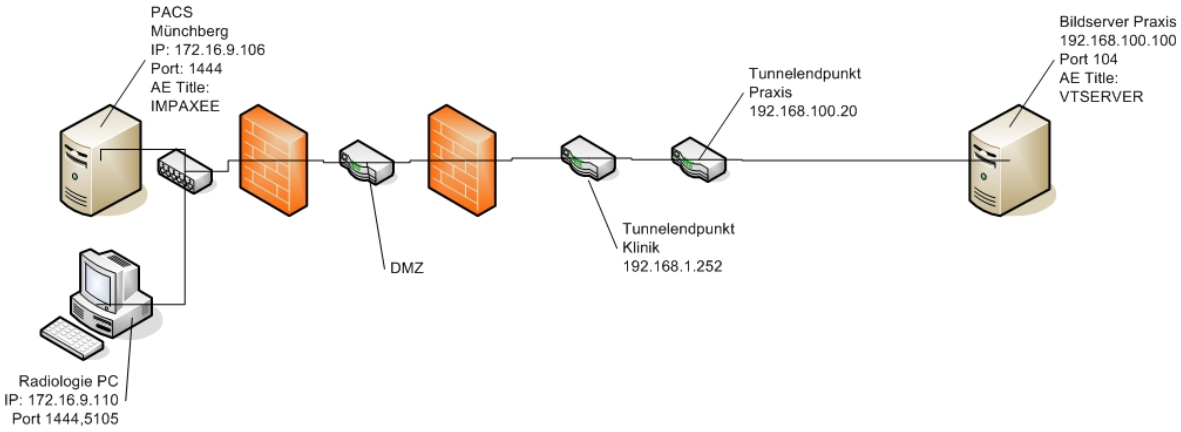

Abbildung 33: Verbindung der Radiologiesysteme

In den Firewalls und Routern mussten die entsprechenden Freigaben für die Teilnehmer eingerichtet werden, damit diese ungehindert kommunizieren können. Zugriffe von außen werden in der Klinik zuerst in die DMZ geleitet. Der PACS-Server befindet sich jedoch außerhalb der DMZ, weswegen hier zusätzliche Firewallregeln eingerichtet werden mussten. Für den Aufbau der DICOM-Verbindung sind, wie schon in Kapitel 4.4.3 beschrieben, die IP-Adresse, der Port sowie der AE-Title des Kommunikationspartners notwendig. Diese Einstellungen waren auf dem jeweiligen System entsprechend einzustellen. Für den alltäglichen Betrieb müssen die zuständigen Personen nur die Bilder für die Übertragung aussuchen, die Übertragung selbst geschieht dann automatisch.

Sendet die Praxis Bilder an die Klinik, bildet das PACS der Klinik den Service Class Provider für die STORE Service Class, der Bildserver der Praxis den Service Class User. Das Service Object Pair besteht somit aus dem Bild und der Aktion Store. Sendet die Klinik an die Praxis, ist die Rollenverteilung genau umgekehrt.

Ein Problem liegt dabei in den ähnlichen Patientenids der beiden Informationssysteme. Diese werden im Header der Bilder mit übertragen. Damit keine falschen Zuordnungen der Bilder auftreten, laufen die Bilder zuerst in einen Pool und müssen dort manuell auf einen Patienten gemappt werden.

Seit 6. Juni 2007 ist die Bildübertragung in Betrieb. An diesem Tagen waren jeweils ein Techniker für die Klinikseite und die Praxisseite vor Ort und haben die Systeme entsprechend konfiguriert. Die Konfiguration ist in obiger Abbildung zu sehen. Der Verbindungstest erfolgte mit Hilfe eines Testbildes, welches einmal von der Praxis in die Klinik und von der Klinik in die Praxis gesendet wurde. Dieses Bild hat jeweils sein Zielsystem erreicht und somit funktioniert die Kommunikation der beiden Systeme.

Das Datenaufkommen von der Praxis in die Klinik ist höher als das von der Klinik in die Praxis. Von der Praxis werden täglich ca. sechs bis sieben Patienten in der Klinik behandelt. Pro Patient sind zwischen zwei und sechs Bilder zu übertragen. Das Aufkommen an postoperativen Bildern, die in die Praxis übertragen werden müssen, ist geringer, da pro Patient nur ein bis zwei Bilder erstellt werden.

7.3 Anbindung des Medistarsystems

Die in Kapitel 5 angedachte direkte Kopplung der beiden Systeme scheiterte an der Politik von Medistar, keine Schnittstellen zu öffnen. Der Hersteller des Kliniksystems war sehr offen für dieses Thema und auch bereit, für diese Kopplung Schnittstelen zu öffnen.

Die Diskussion mit dem zuständigen Systemhaus für das Medistarsystem in der Praxis ergab als einzig mögliche Lösung die Variante mit dem Terminalserver. Bei der Installation eines weiteren Medistarclients, wie er auch in der Praxis verwendet wird, auf dem PC im Arztzimmer besteht das Problem der VPN Verbindung. Sollte diese einmal abbrechen kann dies zu Datenverlusten und ähnlichen Schäden am Praxissystem führen. Dieses Risiko muss auf jeden Fall verhindert werden.

Die Verwendung des bisherigen Hausbesuchsmoduls ist zwar auch über die VPN Verbindung möglich, doch während des Datenabgleichs werden in der Praxis temporäre Datenbanken erzeugt. Während dieser Zeit kann in der Praxis nicht gearbeitet werden, hinzu kommt das

oben genannte Problem bei einem Verbindungsabbruch.

Der Zugriff auf den Terminalserver ist auf zwei Arten realisierbar. Eine Möglichkeit ist ein fester PC im Arztzimmer, auf dem die Daten eingegeben werden. Dies bedeutet aber doppelten Aufwand da bei der Visite erst alle Informationen auf Papier aufgenommen werden und dann erneut von Hand ins System eingegeben werden müssen. Die praktikablere Lösung ist das Laptop am Patientenbett. Hierfür ist aber eine WLAN-Infrastruktur in der Klinik notwendig die es vorerst nicht geben wird. Ein Alternative hierzu ist der Weg über UMTS. Die Geschwindigkeit und Stabilität von UMTS ist ausreichend für diesen Zweck. Ein Für oder Wider hängt rein von den Kosten ab.

Die Erarbeitung einer Detaillösung hierfür wird noch einige Zeit in Anspruch nehmen, da auch ein Austausch des bereits in die Jahre gekommen Praxisservers in Erwägung gezogen wird. Aus diesen Gründen kann die Anbindung des Medistarsystems in der Diplomarbeit nicht berücksichtigt werden.

8 Bewertung des Endergebnisses

Die Bewertung beschränkt sich auf die Bilddatenübertragung, da sich die Anbindung des Medistarsystems noch in der Evaluierungsphase befindet und diese im Rahmen der Diplomarbeit nicht abgeschlossen werden kann. Die Verbindung der beiden Röntgensysteme ist jedoch als sehr gelungen zu bezeichnen. Es wurde eine Lösung gefunden, die von allen Beteiligten akzeptiert wird und die technisch mit geringem Aufwand realisierbar war. Die Ärzte der Praxis haben die benötigten Bilder in Originalqualität vor Ort zur Verfügung, eine nachträgliche Lieferung eines fehlenden Bildes ist in kurzer Zeit zu bewerkstelligen. Die bisherigen Papierausdrucke können entfallen. Weiterhin sind dadurch die Bildlücken in der Praxis verschwunden. Die postoperativen Bilder, die in Klinik angefertigt werden, werden an die Klinik gesendet und sind dort für Nachuntersuchungen verfügbar ohne ein erneutes Bild anfertigen zu müssen, was auf lange Sicht Kosten spart und im Zuge der Gesundheitsreform nicht unwichtig ist.

Der Kostenaufwand zur Erreichung des oben beschriebenen Standes hielt sich für die Praxis in Grenzen. Es fielen lediglich die Lizenzkosten für die beiden DICOM Module sowie für die Vorortinstallation des Routers an. Alle anderen Kosten wie die für den Router, die Einrichtung des VPN Tunnels und die notwendigen Arbeiten am PACS in Münchberg wurden von der Klinik getragen.

9 Zusammenfassung und Ausblick

Unter den in Kapitel zwei beschriebenen Umständen war eine tatsächliche Realisierung der beschriebenen Ziele keine einfache Aufgabe und es musste ein hoher Planungs- und Organisationsaufwand betrieben werden. Es kristallisierte sich sehr schnell heraus, dass die Realisierung der Röntgenbildübertragung einfacher durchzuführen ist, da ein einheitlicher Standard (DICOM) existiert, den alle Hersteller bildgebender Geräte in unterschiedlicher Ausprägung implementieren. Dieser Standard ermöglicht die Kommunikation von Geräten untereinander und es waren auf der Seite der Praxis nur die Module für das Senden und Empfangen von Bildern, auch als Studien bezeichnet, notwendig.

Die am 6. Juni erfolgte Umsetzung erfüllt die Ziele, die in Kapitel drei definiert wurden. Die Bilder aus der Praxis sind in der Klinik in Originalqualität abrufbar und fehlende Bilder sind in wenigen Minuten in die Klinik nachsendbar. Die vor der Umsetzung in der Praxis meist fehlenden postoperativen Bilder aus der Klinik können nun ohne großen Aufwand in die Praxis übertragen werden. Hierdurch verbessert sich die Qualität der Arbeit bei Nachuntersuchungen in der Praxis erheblich.

Die Anbindung von Medistar, dem Praxisinformationssystem, war eine deutlich komplexere Aufgabe. Es gibt auch hier einen Standard, der Kommunikationsabläufe vereinheitlicht, der aber meist für die interne Verknüpfung von Systemmodulen benutzt wird. Für externe Zugriffe müssen die beiden Systemhersteller Schnittstellen vereinbaren. Diese notwendige Zusammenarbeit kam zwischen Agfa und Medistar nicht zu Stande. Agfa war sehr offen für dieses Thema, Medistar hingegen weigerte sich Schnittstellen zu öffnen.

Somit blieb nur die Alternative, von der Klinik aus einen Zugriff auf das Praxissystem zu ermöglichen. Die Verwendung von Onlineclient bzw Hausbesuchsmodul war aufgrund von möglichen Verbindungsabbrüchen und den damit verbunden Datenverlustrisiken nicht möglich. Die letzte Möglichkeit, der Einsatz eines Terminalservers in der Praxis, wurde von

allen Beteiligten angenommen. Mit Hilfe einer WLAN-Infrastrukur in der Klinik hätte diese Möglichkeit einen Datenzugriff am Patientbett ermöglicht. Doch leider wurde das WLAN-Projekt in der Klinik auf unbestimmte Zeit auf Eis gelegt. Ohne einen Onlinezugriff am Patientenbett rechnet sich der finanzielle Aufwand für die Ärzte aber nicht. Derzeit läuft eine Evaluierung eines Zugriffs über UMTS. Diese wird noch einige Zeit in Anspruch nehmen und aus diesem Grund konnte dieser Teil im Rahmen der Diplomarbeit nicht umgesetzt werden.

Fällt die UMTS Evaluierung positiv aus wären die definierten Ziele für diesen Teil ebenfalls erfüllt. Die doppelte Arbeit bei der Datenerfassung lässt sich aufgrund gesetzlicher Regelungen nicht vollständig beseitigen, denn in der Klinik muss für jeden Aufenthalt eines Patienten eine Papierkrankenakte gepflegt werden.

Auch wenn nicht alle Ziele für die Diplomarbeit in die Praxis umgesetzt werden konnten, war diese Diplomarbeit sehr interessant. Neben vieler neuer technischen Dinge, insbesondere die Einarbeitung in den DICOM Standard, konnte ich mein Wissen in den medizinischen Arbeitsabläufen deutlich erweitern und meine vorhanden Erfahrungen in Bezug auf Organisation und Verhandlung mit mehreren Firmen nutzen und weiter vertiefen, was für meinen späteren Beruf sehr wertvolle Kenntnisse sein werden.

A Kurzinformation zum HL-7 Standard

HL7 ist der derzeit wichtigste Kommunikationsstandard zwischen klinischen Informationssystemen und wird für die Übertragung von Patientendaten ohne Bilddaten eingesetzt. Die Kommunikation findet auf Ebene 7 des ISO/OSI Referenzmodells statt.

HL7 ist keine Software, sondern eine sehr detaillierte Anleitung in elektronischer Form. HL7 enthält Beschreibungen, zu welchen Ereignissen Nachrichten mit welchem Aufbau zwischen Bausteinen einer Anwendung im Klinikbereich gesendet werden.

Als abstraktes kleines Beispiel sind zwei Anwendungen A und B gegeben. In A tritt ein Ereignis ein. Der Typ des Ereignisses bestimmt den Nachrichtentyp. Diese Nachricht wird nun an B gesendet. Nach dem Empfang der Nachricht schick B nun eine Bestätigung zurück an A. [34]

[34] vgl. [Gär05] Seiten 91-102

B Installation der Modellsoftware

Für die Installation der Modellsoftware sind im wesentlichen vier Schritte zu erledigen: die Installation des MySQL Datenbankservers, die Installation der Java Laufzeitumgebung, das Entpacken der eigentlichen Modellsoftware sowie die Anpassung der Konfigurationsdateien.

B.1 Installation von MySQL 5

Auf der beigelegten CD-ROM befindet sich ein Ordner **Externe Software**. Dort befindet sich das MySQL MSI Paket. Dieses ist standardmäßig zu installieren. Im zweiten Schritt wird ein Konfigurationsassistent gestartet. Hier ist es wichtig, als Datenbanktyp **Multifunctional Database** zu wählen. Merken Sie sich das Passwort für den Benutzer root, dieses wird später noch benötigt.

B.2 Installation von Java 6

In oben beschriebenem Ordner befindet sich auch der Installer für die Java Laufzeitumgebung. Führen Sie diesen ebenfalls standardmäßig aus.

Hiermit sind die zusätzlichen Softwareinstallationen abgeschlossen.

B.3 Entpacken der Modellsoftware

Entpacken Sie nun die Datei VerbindungSimulator.rar an einen Ort Ihrer Wahl. Stellen Sie sicher, dass der Ordner VerbindungSimulator folgende Dateien und Ordner enthält:

- Ordner lib
- Ordner images

- Datei VerbindungSimulator.jar
- Datei einstellungen.properties
- Datei start.bat

B.4 Anpassen der Konfigurationsdateien

start.bat In der Datei start.bat ist in der ersten Zeile der Pfad anzugeben, wohin die Software entpackt wurde.

einstellungen.properties In der Datei einstellungen.properties sind die Parameter pfad,dbuser,dbpass und dbhost entsprechend anzupassen. Der Parameter pfad muss den Pfad zu dem Ordner images enthalten **mit zwei Backslash am Ende**. Bei den db* Parametern sind die Daten entsprechend einzugeben, die Sie bei der Installation des MySQL-Servers eingeben haben.

Stellen Sie für den ersten Start sicher, dass der Parameter reset den Wert 1 hat. Ein Doppelklick auf die Datei start.bat startet das Programm. Ist dies nicht der Fall, müssen Sie in der Datei start.bat den absoluten Pfad zur Datei java.exe mit angeben.

C Klassendiagramme

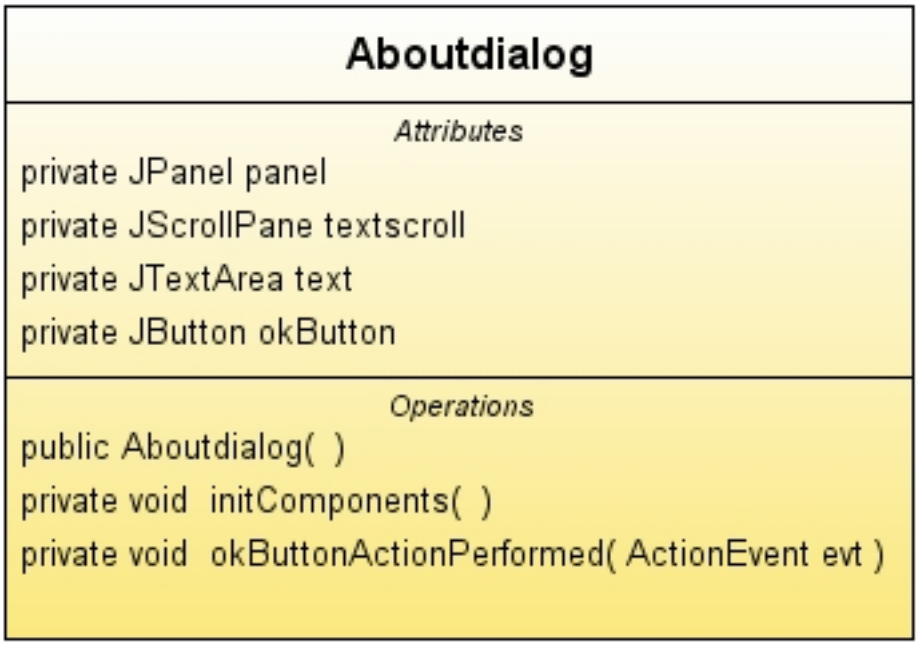

Abbildung 34: Klasse Aboutdialog

Die Klasse Aboutdialog verwaltet den Dialog, in dem Informationen über das Programm enthalten sind. Die Klasse erweitert die Klasse JFrame. Hierfür werden folgende Attribute und Methoden benötigt:
Attribute:

- JPanel panel

  Das Panel enhält die JTextArea sowie den JButton in einem BorderLayout

- JScrollPane textscroll

  Da der Text länger als die Höhe des Fensters ist, muss der Text in eine ScrollPane eingebettet sein

- JTextArea text

  Die Variable Text enthält den eigentlichen Inhalt des Dialogs

- JButton okButton

  Der Button dient dem Schließen des Dialogfensters

Methode:

- Aboutdialog()

  Dies ist der Konstruktor, in dem die Methode initComponents aufgerufen wird

- initComponents()

  Diese Methode initialisiert die oben beschriebenen Variablen und weist die definierten Werte zu

- okButtonActionPerformed(ActionEvent evt)

  Dieser ActionListener sorgt für die Funktionalität des okButtons, in dem er das Fenster sauber schließt und den verwendeten Speicher wieder freigibt

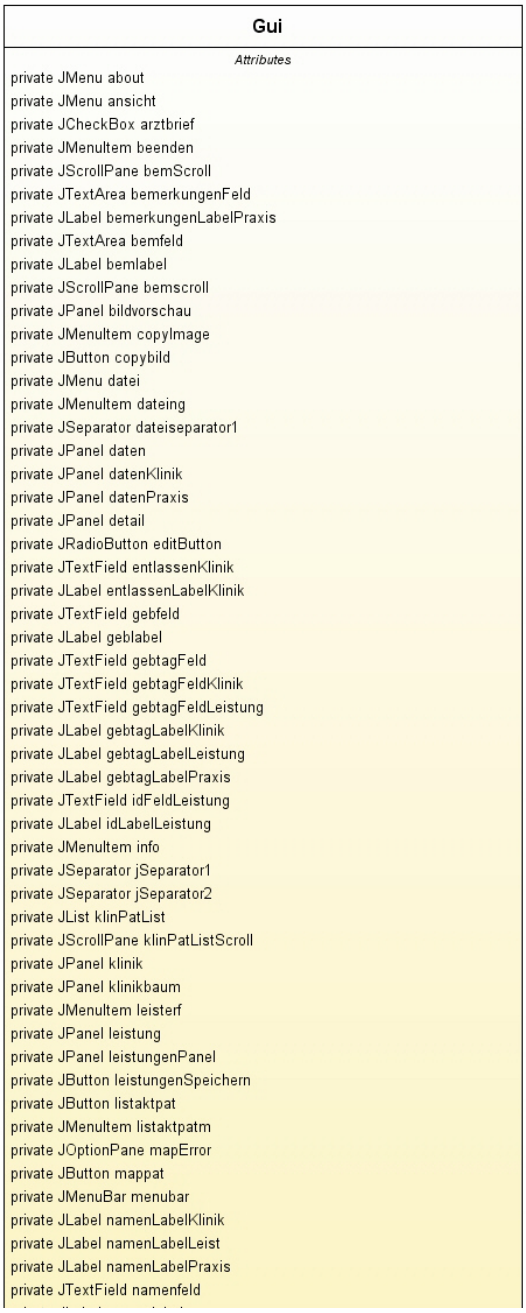

Abbildung 35: Klasse Gui - Teil 1

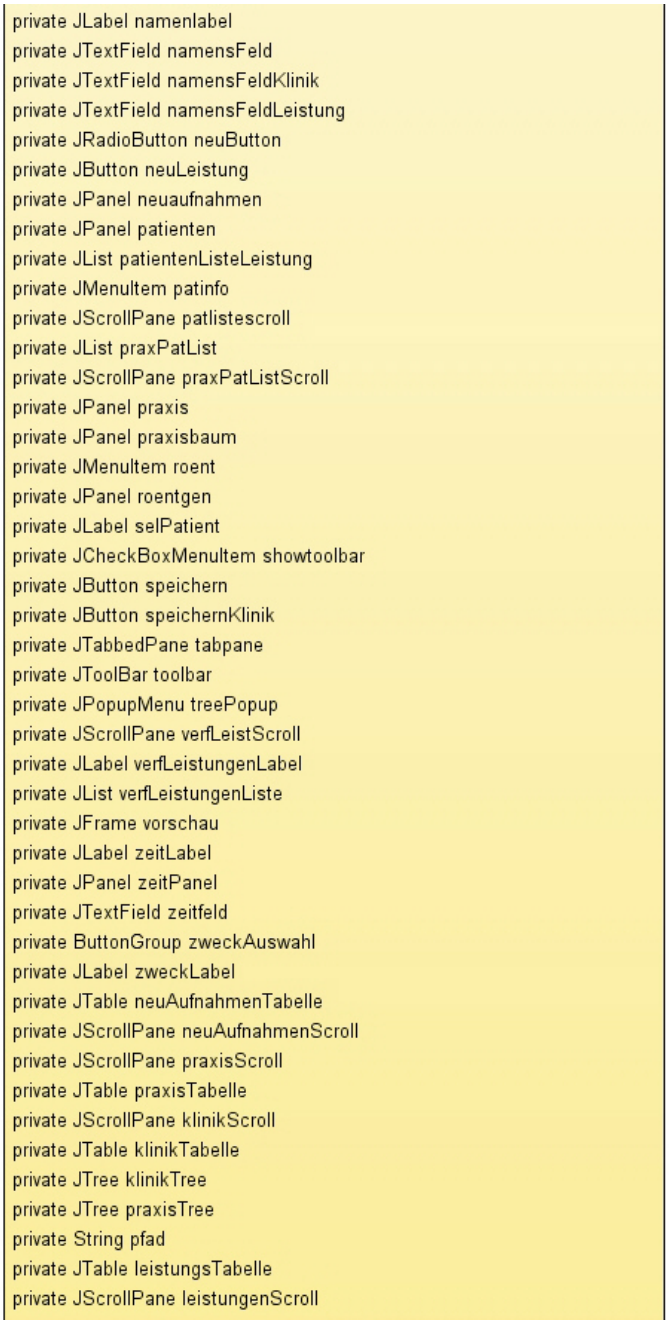

Abbildung 36: Klasse Gui - Teil 2

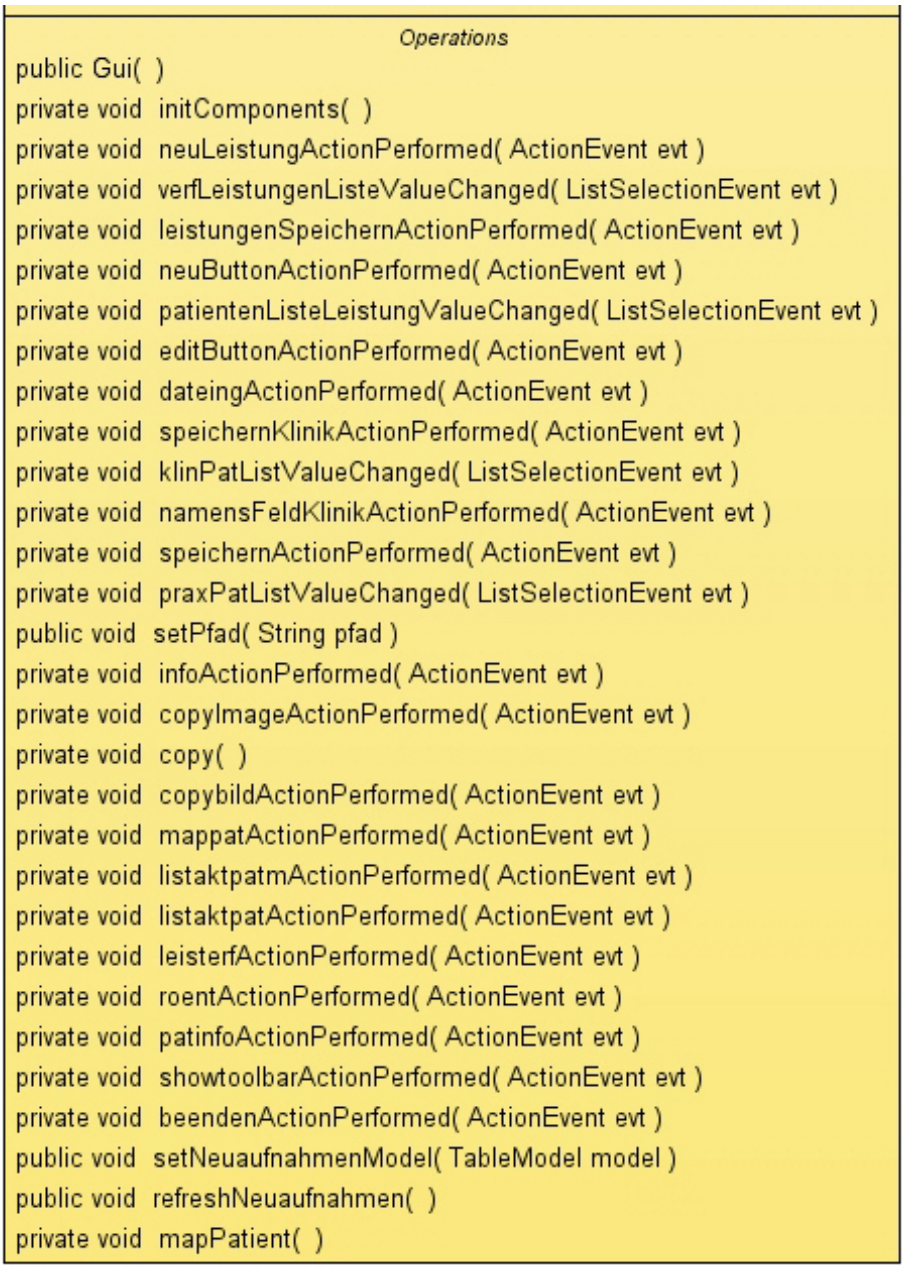

Abbildung 37: Klasse Gui - Teil 3

Die Klasse Gui ist für alle grafischen Elemente und deren Aktionen zuständig. Somit lässt sich das Programm komplett über die grafische Oberfläche steuern. Die Attribute repräsentieren die einzelnen grafischen Elemente aus denen die Oberfläche zusammengesetzt ist. Diese werden hier nicht explizit aufgelistet. Die Methoden sind im wesentlichen Action Listener und zuständig für die Funktionalität der Benutzeroberfläche. Diese sind im Detail:

- Gui()

  Der Konstruktor erzeut eine neue Instanz der Benutzeroberfläche. Er ruft die Methode initComponents() und fügt die einzelnen Elemente zur endgültigen Benutzeroberfläche zusammen

- initComponents()

  Diese Methode initialisiert alle Attribute

- neuLeistungActionPerformed(ActionEvent evt)

  Diese Methode fügt der Leistungstabelle eine neue Zeile hinzu

- verfLeistungenValueChanged(ListSelectionEvent evt)

  Diese Methode füllt die Leistungstabelle bei der Auswahl eines bestehenden Zeitraums die Tabelle mit den entsprechenden Daten

- leistungenSpeichernActionPerformed(ActionEvent evt)

  Diese Methode liest die Eingaben aus und übergibt diese an die entsprechende Methode des DatabaseKonnektors

- neuButtonActionPerformed(ActionEvent evt)

  Wird eine neue Leistung eingegeben wird die Liste mit bestehenden Leistungen disabled

- patientenListeLeistungValueChanged(ListSelectionEvent evt)

  Bei der Auswahl eines Patienten füllt diese Methode die entsprechenden Felder und Listen mit den Werten aus der Datenbank

- editButtonActionPerformed(ActionEvent evt)

  Soll eine bestehender Leistungszeitraum bearbeitet werden wird die Liste enabled

- dateingActionPerformed(ActionEvent evt)

  Diese Methode aktiviert den Dateneingabetab

- speichernKlinikActionPerformed(ActionEvent evt)
  Diese Methode liest die Dateneingaben für das Kliniksystem aus und gibt diese an den DatabaseKonnektor weiter

- klinPatListValueChanged(ListSelectionEvent evt)
  Diese Methode holt bei Auswahl eines Patienten aus dem Kliniksystem die bereits vorhanden Daten aus der Datenbank und setzte diese in die Felder ein

- speichernActionPerformed(ActionEvent evt)
  Diese Methode liest die Dateneingaben für das Praxissystem aus und gibt diese an den DatabaseKonnektor weiter

- praxPatListValueChanged(ActionEvent evt)
  Diese Methode holt bei Auswahl eines Patienten aus dem Praxissystem die bereits vorhanden Daten aus der Datenbank und setzte diese in die Felder ein

- setPfad(String pfad)
  Diese Methode setzt die Pfadvariable für die Röntgenbilder mit dem Wert, der im Propertiesfile angegeben wurde

- infoActionPerformed(ActionEvent evt)
  Diese Methode zeigt den About Dialog an

- copyImageActionPerformed(ActionEvent evt)
  Ruft die Methode copy() auf

- copy()
  Diese Methode kopiert ein Röntgenbild von der Praxis in die Klinik

- copybildActionPerformed(ActionEvent evt)
  Ruft die Methode copy() auf

- mappatActionPerformed(ActionEvent evt)
  Ruft die Methode mapPatient() auf

- listaktpatmActionPerformed(ActionEvent evt)
  Ruft die Methode refreshNeuaufnahmen() auf

- listaktpatActionPerformed(ActionEvent evt)
  Ruft die Methode refreshNeuaufnahmen() auf

- leisterfActionPerformed(ActionEvent evt)
  Diese Methode macht den Leistungserfassungstab sichtbar

- roentActionPerformed(ActionEvent evt)
  Diese Methode macht den Röntgentab sichtbar

- patinfoActionPerformed(ActionEvent evt)
  Diese Methode macht den Patienteninfotab sichtbar

- showtoolbarActionPerformed(ActionEvent evt)
  Diese Methode zeigt oder versteckt die Toolbar

- beendenActionPerformed(ActionEvent evt)
  Diese Methode beendet das Programm

- setNeuaufnahmenModel(TabelModel model)
  Diese Methode setzt das Tabellenmodel für die Neuaufnahmen Tabelle

- refreshNeuaufnahmen()
  Diese Methode holt vom DatabaseKonnektor eine aktuelle Liste mit neuaufgenommen Patienten und stellt diese in Tabellenform dar

- mapPatient() Diese Methode ordnet eine Klinikneuaufnahme einem Patienten in der Praxis zu

Abbildung 38: Klasse DatabaseKonnektor

Der DatabaseKonnektor bildet die Schnittstelle zur Datenbank. Die Methoden der Klasse speichern die Eingaben des Benutzers oder liefern die angeforderten Daten an die Benutzeroberfläche zurück. Folgende Methoden werden hierzu benötigt:

- DatabaseKonnektor()

  Der Konstruktor initialisiert den Driver Manager mit dem MySQL Datenbanktreiber

- Connection connectPraxis()

  Diese Methode baut die Verbindung zur Praxisdatenbank auf und gibt diese zurück

- Connection connectKlinik()

  Diese Methode baut die Verbindung zur Klinikdatenbank auf und gibt diese zurück

- int getAnzahlPatientenPraxis()

  Diese Methode gibt die Anzahl der Patienten in der Praxisdatenbank als Integerwert zurück

- String[][] getPatientenPraxis

  Diese Methode gibt alle Patienten der Praxisdatenbank inclusive ihrer gesamten Daten als mehrdimensionales Stringarray zurück

- int getAnzahlPatientenKlinik()

  Diese Methode gibt die Anzahl der Patienten in der Klinikdatenbank als Integerwert zurück

- String[][] getPatientenKlinik()

  Diese Methode gibt alle Patienten der Klinikdatenbank inclusive ihrer gesamten Daten als mehrdimensionales Stringarray zurück

- int getMappingKandiaten(String name, String vorname, String gebtag, String aufdat)

  Hier werden für die Registrierung einer Klinikaufnahme möglichen Patienten in der Praxisdatenbank gesucht. Wird nur ein möglicher Patient gefunden, erfolgt die Registrierung automatisch, ansonsten gibt es eine Auswahl der möglichen Patienten

- String[][] getPatientenBilderPraxis()

  Diese Methode gibt für jeden Patient der Praxis die Röntgenbilder mit zurück

- String[][] getPatientenBilderKlinik()

  Diese Methode gibt für jeden Patient der Klinik die Röntgenbilder mit zurück

- copyImage(String name, String vorname, String gebtag, String pfad)
  Diese Methode kopiert ein Röntgenbild anhand der übergebenen Daten von der Praxis in die Klinik

- String[][] getNeuaufnahmen()
  Diese Methode gibt die neu aufgenommen Patienten in der Klinik als Stringarray zurück

- String[] getListePatientenPraxis()
  Diese Methode gibt eine Liste aller Praxispatienten mit Name, Vorname und Geburtstag zurück

- String[] getListePatientenKlinik()
  Diese Methode gibt eine Liste aller Klinikpatienten mit Name, Vorname und Geburtstag zurück

- reset()
  Durch den Aufruf dieser Methode wird die Datenbank in den Urzustand versetzt

- String getBemerkungenPraxis(String name, String vorname, String gebtag)
  Diese Methode gibt den Inhalt des Bemerkungsfeldes eines Patienten zurück

- setBemerkungenPraxis(String name, String gebtag, String bemerkungen)
  Diese Methode speichert den Inhalt des Bemerkungsfeldes eines Patienten

- setDatenKlinik(String name, String gebtag, String entlassdatum, Sring brief)
  Diese Methode speichert zusätzliche Informationen zu einem Patienten in der Datenbank

- String getEntlassDatumKlinik(String name, String vorname, String gebtag)
  Diese Methode gibt das Entlassdatum eines Patienten als String zurück

- String getArztbriefKlinik(String name, String vorname, String gebtag)
  Diese Methode gibt zurück, ob ein Arztbrief vorhanden ist oder nicht

- String[] getLeistungszeitZeitraeume(String id)
  Diese Methode gibt die Zeiträume zurück, in denen für einen Patienten bereits Leistungen erfasst wurden

- int getAnzahlZeiträume(String id)
  Diese Methode gibt die Anzahl der Zeiträume zurück, in denen für einen Patienten bereits Leistungen erfasst wurden

- String getPatientenId(String name, String vorname, String gebtag)
  Diese Methode gibt die Id eines Patienten zurück

- setLeistung(String id, String zeitraum, String[][] leistungen)
  Diese Methode speichert die Leistungen eines bestimmten Zeitraums in der Datenbank

- String[][] getLeistungen(String id, String zeitraum)
  Diese Methode gibt die Leistungen eines bestimmten Zeitraums zurück

- int getanzahlLeistungen(String id, String zeitraum)
  Diese Methode gibt die Anzahl der Einzelleistungen eines bestimmten Zeitraums zurück

- updateLeistungen(String id, String zeitraum, String[][] leistungen)
  Diese Methode speichert die Änderungen an bestehenden Leistungen in der Datenbank

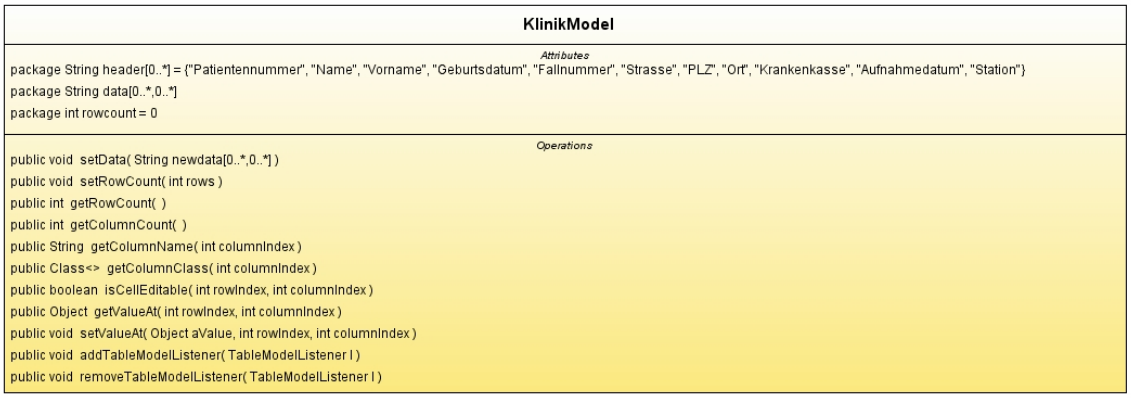

Abbildung 39: Klasse KlinikModel

Die Klasse KlinikModel definiert das Aussehen sowie das Verhalten der Kliniktabelle der Klasse Gui und implementiert das Interface TableModel. Sie enthält die folgenden Attribute und Methoden:
Attribute:

- String[] header

  Dieses eindimensionale Stringarray definiert den Header der Tabelle

- String[][] data

  Dieses mehrdimensionale Array enthält die Daten, die in der Tabelle angezeigt werden

- int rowcount

  Dieser ganzzahlige Wert enthält die Anzahl der Zeilen der Tabelle

Methoden:

- setData(String[][] newdata)

  Diese Methode setzt das Attribut data

- setRowCount(int rows)

  Diese Methode setzt das Attribut rowcount

- getRowCount()

  Diese Methode gibt die Anzahl der Zeilen als Integer zurück

- getColumnCount()

  Diese Methode gibt die Anzahl der Spalten als Integer zurück

- getColumnName(int columnindex)

  Hier wird der Spaltenname in Abhängigkeit des übergebenen Index als String zurückgegeben

- getColumnClass(int columnindex)

  Diese Methode gibt die Klasse der Spalte zurück, deren Index übergeben wurde

- isCellEditable(int rowIndex, int columnIndex)

  Diese Methode gibt einen boolschen Wert zurück, ob die Tabellenzelle, die mit dem Zeilenindex und Spaltenindex übergeben wurde, editierbar ist oder nicht

- getValueAt(int rowIndex, int columnIndex)

  Hier wird der Wert der durch den übergebenen Zeilen- und Spaltenindex bestimmten Tabellenzelle als Object zurückgegeben

- setValueAt(Object aValue,int rowIndex, int columnIndex)

  Diese Methode weist der durch den übergebenen Zeilen- und Spaltenindex bestimmten Tabellenzelle den Wert aValue zu

- addTableModelListener(TableModelListener l)

  Diese Methode ist nicht implementiert

- remove TableModelListener(TableModelListener l)

  Diese Methode ist nicht implementiert

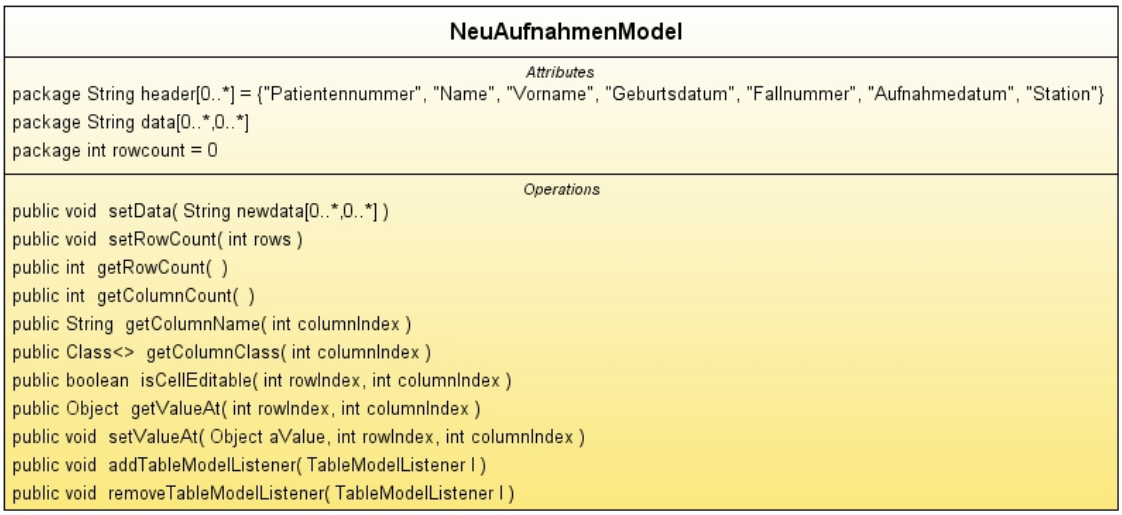

Abbildung 40: Klasse NeuaufnahmenModel

Die Klasse NeuAufnahmenModel definiert das Aussehen sowie das Verhalten der Neuaufnahmentabelle der Klasse Gui und implementiert das Interface TableModel. Sie enthält die folgenden Attribute und Methoden:
Attribute:

- String[] header

  Dieses eindimensionale Stringarray definiert den Header der Tabelle

- String[][] data

  Dieses mehrdimensionale Array enthält die Daten, die in der Tabelle angezeigt werden

- int rowcount

  Dieser ganzzahlige Wert enthält die Anzahl der Zeilen der Tabelle

Methoden:

- setData(String[][] newdata)

  Diese Methode setzt das Attribut data

- setRowCount(int rows)

  Diese Methode setzt das Attribut rowcount

- getRowCount()

  Diese Methode gibt die Anzahl der Zeilen als Integer zurück

- getColumnCount()

  Diese Methode gibt die Anzahl der Spalten als Integer zurück

- getColumnName(int columnindex)

  Hier wird der Spaltenname in Abhängigkeit des übergebenen Index als String zurückgegeben

- getColumnClass(int columnindex)

  Diese Methode gibt die Klasse der Spalte zurück, deren Index übergeben wurde

- isCellEditable(int rowIndex, int columnIndex)

  Diese Methode gibt einen boolschen Wert zurück, ob die Tabellenzelle, die mit dem Zeilenindex und Spaltenindex übergeben wurde, editierbar ist oder nicht

- getValueAt(int rowIndex, int columnIndex)

  Hier wird der Wert der durch den übergebenen Zeilen- und Spaltenindex bestimmten Tabellenzelle als Object zurückgegeben

- setValueAt(Object aValue,int rowIndex, int columnIndex)

  Diese Methode weist der durch den übergebenen Zeilen- und Spaltenindex bestimmten Tabellenzelle den Wert aValue zu

- addTableModelListener(TableModelListener l)

  Diese Methode ist nicht implementiert

- remove TableModelListener(TableModelListener l)

  Diese Methode ist nicht implementiert

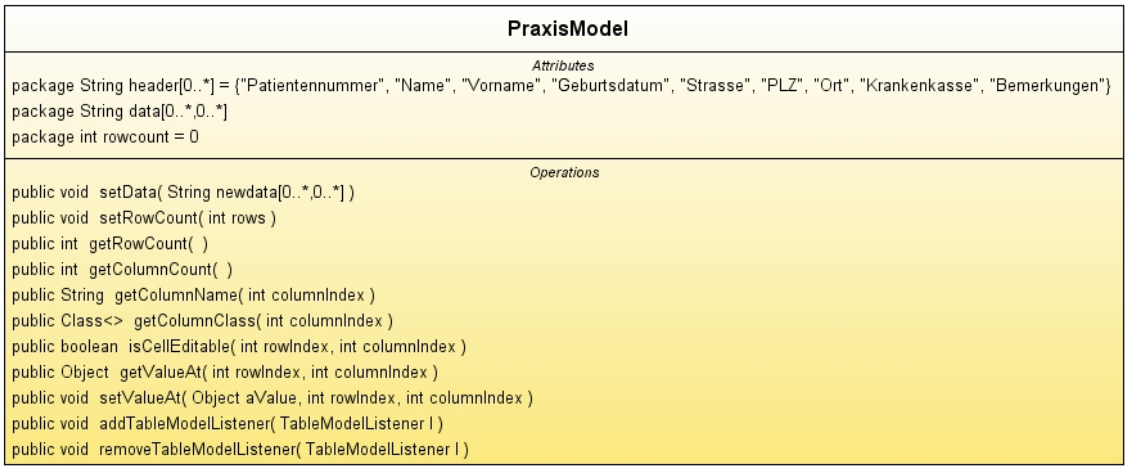

Abbildung 41: Klasse PraxisModel

Die Klasse PraxisModel definiert das Aussehen sowie das Verhalten der Praxistabelle der Klasse Gui und implementiert das Interface TableModel. Sie enthält die folgenden Attribute und Methoden:
Attribute:

- String[] header

  Dieses eindimensionale Stringarray definiert den Header der Tabelle

- String[][] data

  Dieses mehrdimensionale Array enthält die Daten, die in der Tabelle angezeigt werden

- int rowcount

  Dieser ganzzahlige Wert enthält die Anzahl der Zeilen der Tabelle

Methoden:

- setData(String[][] newdata)

  Diese Methode setzt das Attribut data

- setRowCount(int rows)

  Diese Methode setzt das Attribut rowcount

- getRowCount()
  Diese Methode gibt die Anzahl der Zeilen als Integer zurück

- getColumnCount()
  Diese Methode gibt die Anzahl der Spalten als Integer zurück

- getColumnName(int columnindex)
  Hier wird der Spaltenname in Abhängigkeit des übergebenen Index als String zurückgegeben

- getColumnClass(int columnindex)
  Diese Methode gibt die Klasse der Spalte zurück, deren Index übergeben wurde

- isCellEditable(int rowIndex, int columnIndex)
  Diese Methode gibt einen boolschen Wert zurück, ob die Tabellenzelle, die mit dem Zeilenindex und Spaltenindex übergeben wurde, editierbar ist oder nicht

- getValueAt(int rowIndex, int columnIndex)
  Hier wird der Wert der durch den übergebenen Zeilen- und Spaltenindex bestimmten Tabellenzelle als Object zurückgegeben

- setValueAt(Object aValue,int rowIndex, int columnIndex)
  Diese Methode weist der durch den übergebenen Zeilen- und Spaltenindex bestimmten Tabellenzelle den Wert aValue zu

- addTableModelListener(TableModelListener l)
  Diese Methode ist nicht implementiert

- remove TableModelListener(TableModelListener l)
  Diese Methode ist nicht implementiert

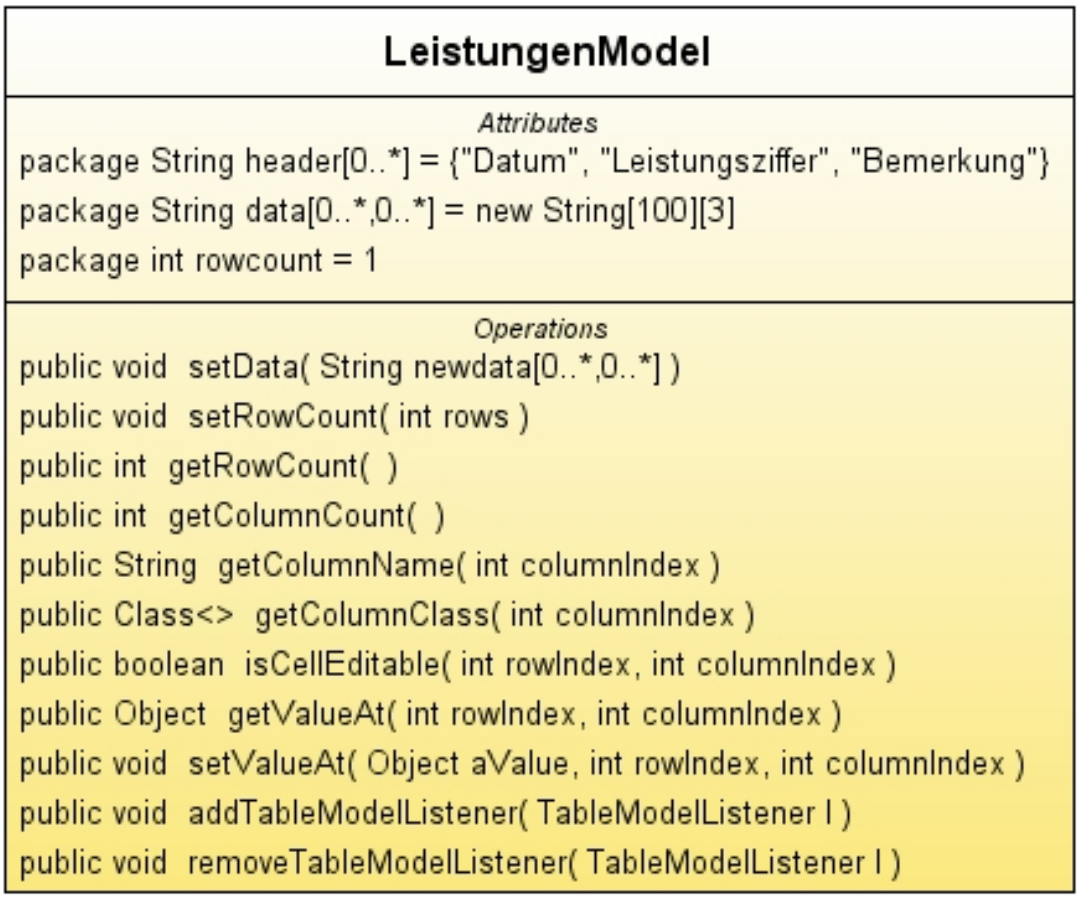

Abbildung 42: Klasse LeistungenModel

Die Klasse LeistungenModel definiert das Aussehen sowie das Verhalten der Leistungstabelle der Klasse Gui und implementiert das Interface TableModel. Sie enthält die folgenden Attribute und Methoden:
Attribute:

- String[] header

  Dieses eindimensionale Stringarray definiert den Header der Tabelle

- String[][] data

  Dieses mehrdimensionale Array enthält die Daten, die in der Tabelle angezeigt werden

- int rowcount

  Dieser ganzzahlige Wert enthält die Anzahl der Zeilen der Tabelle

Methoden:

- setData(String[][] newdata)

  Diese Methode setzt das Attribut data

- setRowCount(int rows)

  Diese Methode setzt das Attribut rowcount

- getRowCount()

  Diese Methode gibt die Anzahl der Zeilen als Integer zurück

- getColumnCount()

  Diese Methode gibt die Anzahl der Spalten als Integer zurück

- getColumnName(int columnindex)

  Hier wird der Spaltenname in Abhängigkeit des übergebenen Index als String zurückgegeben

- getColumnClass(int columnindex)

  Diese Methode gibt die Klasse der Spalte zurück, deren Index übergeben wurde

- isCellEditable(int rowIndex, int columnIndex)

  Diese Methode gibt einen boolschen Wert zurück, ob die Tabellenzelle, die mit dem Zeilenindex und Spaltenindex übergeben wurde, editierbar ist oder nicht

- getValueAt(int rowIndex, int columnIndex)

  Hier wird der Wert der durch den übergebenen Zeilen- und Spaltenindex bestimmten Tabellenzelle als Object zurückgegeben

- setValueAt(Object aValue,int rowIndex, int columnIndex)

  Diese Methode weist der durch den übergebenen Zeilen- und Spaltenindex bestimmten Tabellenzelle den Wert aValue zu

- addTableModelListener(TableModelListener l)

  Diese Methode ist nicht implementiert

- remove TableModelListener(TableModelListener l)

  Diese Methode ist nicht implementiert

D Internetquellen

Literatur

[Gär05] GÄRTNER, Armin: *Medizintechnik und Informationstechnologien Band 2 Bildmanagement*. TÜV Verlag, 2005

[Tan03] TANENBAUM, Andrew S.: *Computernetzwerke*. 4. Auflage. Pearson Studium, 2003

Internetquellen

http://www.agfa.com

http://de.wikipedia.org/wiki/Firewall

http://de.wikipedia.org/wiki/IPSec

http://de.wikipedia.org/wiki/Paketfilter

http://de.wikipedia.org/wiki/Stateful_Packet_Inspection

http://de.wikipedia.org/wiki/Virtual_Private_Network

http://www.medistar.de

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde nach meiner besten Kenntnis bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Hof, den Unterschrift