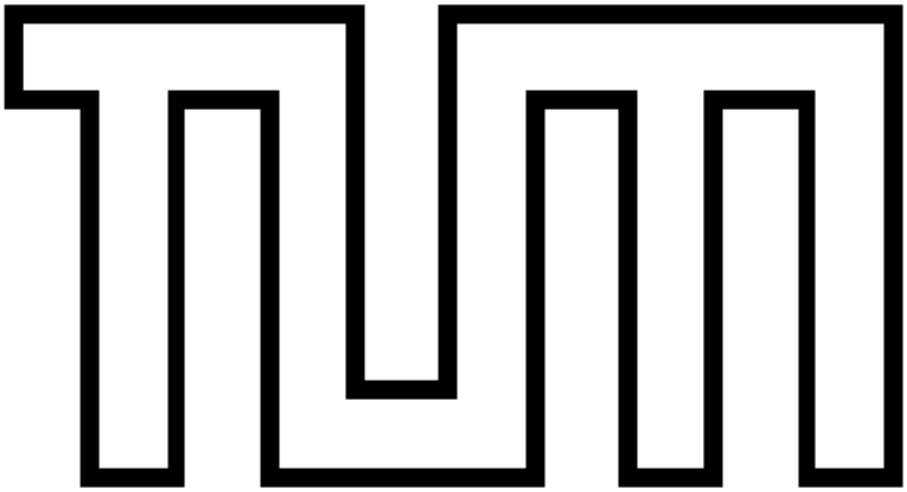


**Technische Universität München**

**Fakultät Elektrotechnik und Informationstechnik**

**Lehrstuhl für Real-Computersysteme**

# Diplomarbeit

Design und prototypische Realisierung einer telemedizinischen Dienstleistungs-Plattform für mobile Schwangerschaftsüberwachung

TECHNISCHE UNIVERSITÄT MÜNCHEN

LEHRSTUHL FÜR REALZEIT-COMPUTERSYSTEME
TECHNISCHE UNIVERSITÄT MÜNCHEN
UNIV.-PROF. DR.-ING. G. FÄRBER

RCS

# Diplomarbeit

## Design und prototypische Realisierung einer telemedizinischen Dienstleistungs-Plattform für mobile Schwangerschaftsüberwachung

| | |
|---|---|
| Diplomand: | Jalal Sefrioui Moudine |
| Betreuer TUM: | Dipl-Ing Robert Diemer |
| Betreuer Trium: | Dipl-Inf Christian Harböck |
| Beginn: | 15.01.2007 |
| Abgabe: | 15.07.2007 |

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Motivation

Der Einsatz von Telematik im Gesundheitswesen, vielfach auch Telemedizin genannt, bringt in vielen Situationen Vorteile sowohl für die Patienten als auch für die beteiligten Ärzte. Der ist heute nicht mehr umstritten. Jedoch ist die Telemedizin in Deutschland, im Gegensatz zu anderen westlichen Ländern wie USA, Großbritannien oder Norwegen, nicht verbreitet.

An vielen Institutionen wird die Telemedizin nur versuchsweise eingesetzt. Die Ursachen liegen in vielfältigen Bereichen: fehlende allgemeine anerkannte technische Standards und Ressourcen, fehlende Regelungen bezüglich der Abrechnung derartiger telemedizinischer Leistungen und juristische Fragen der Verantwortungsübernahme und des Datenschutzes. Zu diesen allgemeinen Fragen kommen noch die fachbezogenen Probleme beim Telemedizineinsatz hinzu. Besonders wichtig ist hier die Frage, wie sich die Telemedizin in die Routine so integrieren lässt, dass deren Nutzung weder eine Störung bzw. Verlängerung der Untersuchungen und Behandlungen darstellt, noch auf eine Inakzeptanz stößt.

Neben der Gynäkologie gibt es auch viele Fachdisziplinen wie Radiologie, Pathologie, Chirurgie, etc., in denen die telemedizinische Methode aufgebaut werden kann. Eine telemedizinische Anwendung in der Radiologie beschränkt sich auf die Übertragung von hochauflösenden Standbildern, wenn auch nicht immer von einzelnen Bildern wie Röntgenbildern, sondern von mehreren zusammenhängenden wie Computertomographien oder Magnetresonanzbildern

Eine telemedizinische Anwendung für die mobile Schwangerschaftsüberwachung legt den Schwerpunkt auf die Übertragung von biomedizinischen Daten (Herzfrequenz, Wehen, Blutdruck etc) anhand von tragbaren Kardiotokographie-Geräten auf eine zentrale Einheit fest. Die übertragenen Messwerte werden in einem telemedizinischen Zentrum von Fachleuten mit Hilfe integrierten Analysealgorithmen interpretiert, so dass gegebenenfalls notwendige Therapiemaßnahmen eingeleitet werden können.

## 1.2 Das Unternehmen

Die Firma *Trium Analysis Online* wurde 1999 von Dr. Martin Daumer und Dipl. Stat Michael Sholz gegründet. Seitdem beschäftigt sie sich mit der Entwicklung von Webapplikationen in der Medizin, sie hat sich auf den medizinischen Bereich spezialisiert und das Motto "Verbesserung der menschlichen Gesundheit" als Unternehmensziel gesetzt. Dafür entwickelt und vermarktet Trium innovative Produkte und Dienstleistungen in den Bereichen Telemedizin und Klinische Studien.

Trium stellt vor allem einfache Plattformen für Durchführung, Management, Analyse und Berichterstellung von Klinischen Studien (CTEngine) zur Verfügung, entwickelt und vermarktet Überwachungssysteme für stationäre Patienten und mobile Lösungen mit telemedizinischer Anbindung für eine ortsunabhängige Fernbetreuung von Patienten.

Trium hat sich frühzeitig für den Bereich Telemedizin eingesetzt, im Lauf des BMBF[1] geförderten Projekt „MMM-Mobile Medical Monitoring" hat Trium das mobile medizinische Überwachungssystem „MedShirt®"zur gesundheitlichen Fernbetreuung von Patienten entwickelt. Dabei werden Vitalparameter wie EKG [2], Blutdruck, Sauerstoff, Temperatur über intelligente Sensoren kontinuierlich aufgezeichnet und per Handy an die Dienstleistunsgzentrale übermittelt. Dort werden mit geeigneten Algorithmen, automatisierte Klassifizierungen, Handlungsempfehlungen und Warnhinweise verarbeitet und weitergegeben. Der Arzt kann die per Mobilfunk übertragene Daten jederzeit abfragen, analysieren und telefonisch eine Medikation oder den Klinikbesuch empfehlen. Mit Hilfe des eingebauten GPS-Modul kann der Patient sogar im Notfall für die Rettungssanitäter sofort zugeordnet werden.

Neben ihrem zentralen Kreissaalüberwachungssystem „*Trium CTG Online*", das für stationäre Patienten ausgedacht ist, hat *Trium* zuletzt eine weitere telemedizinische Anwendung entwickelt und auf den Markt gebracht. Es handelt sich hierbei um das sogenannte „*Trium CTG Mobile*", welches eine mobile Überwachung von Schwangeren zu Hause ermöglicht. Diese Lösung eröffnet den Schwangeren ein größeres Maß an Lebensqualität, Bewegungsfreiheit und Sicherheit (Medizinprodukt Risikoklasse IIa) ohne wiederholte Krankenhaus-Aufenthalte. Nach einer erfolgreichen Studie (Klinikum Rechts der Isar, Klinikum Neuperlach, Krankenhaus Harlaching) wurde das Produkt als medizinisches System zertifiziert. Es ermöglicht die Übertragung mehrerer Messdaten wie der Herzrate des Feten, der Wehentätigkeit der Mutter und deren Blutdruck an eine zentrale Einheit, welche die Daten für die angebundenen Mandanten aufbereitet, darstellt und bedarfsgerecht benachrichtigt. Das Modellkonzept dieser Anwendung soll im Rahmen dieser Diplomarbeit erweitert werden.

[1] Bundesministerium für Bildung und Forschung

[2] Elektrokardiogramm ist die Registrierung der Summe der elektrischen Aktivitäten aller Herzmuskelfasern

# Kapitel 2

# Telemedizin

## 2.1 Definition

Telemedizin ist ein generelles Konzept, das auf diverse Verfahren im Zusammenhang mit der Gesundheit anwendbar ist. Es handelt sich um ein neues, weitgehend noch in Entstehung begriffenes Gebiet, das sich neuer technologischer Entwicklungen bedient und innovative Behandlungsansätze anbietet. Laut Weltgesundheitsorganisation (WHO) umfasst der Begriff Telemedizin die Anwendung der Informations- und Kommunikationstechnologien im Gesundheitssystem zur Unterstützung der direkten oder indirekten medizinischen Behandlung (MM00).

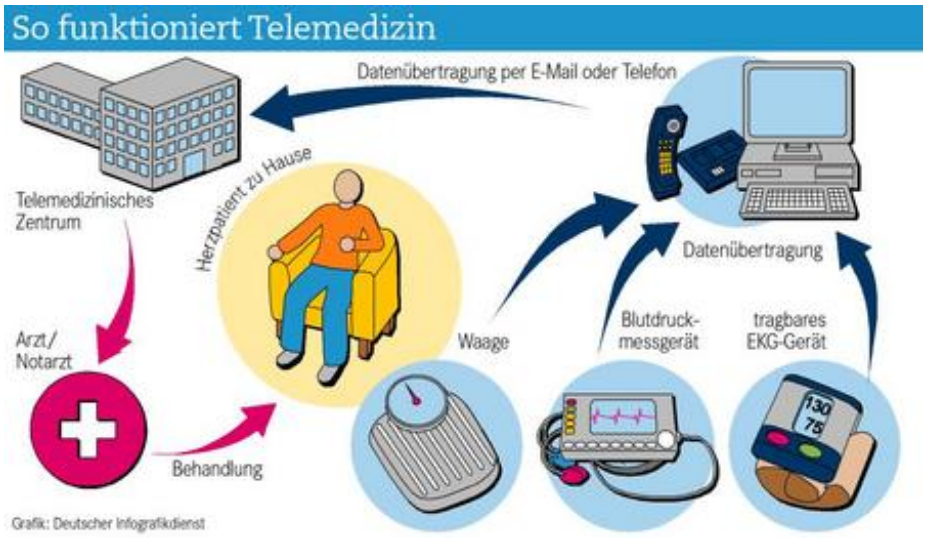



Abbildung 2.1: Szenario der Telemedizin [Quelle (Med)]

Der grösste Vorteil der Telemedizin ergibt sich durch den erleichterten Zugriff auf medizinische Information, die jederzeit und von jedem beliebigen Ort aus möglich wird. Grundkonzept ist dabei die Übertragung von Informationen und/oder Know-how an den Ort, wo

ein Entschluss getroffen oder eine medizinische Handlung durchgeführt werden muss. Die Information kommt zum Patienten und nicht mehr umgekehrt (Abbildung 2.1). Bei der Förderung neuer Entwicklungen der Telemedizin sollte dieses Konzept stets beherzigt werden. Informations- und Kommunikationstechnologien (IKT) spielen dabei die Hauptrolle. Der Informationstransfer erfolgt sowohl über das klassische öffentliche Fernmeldenetz wie über Funk, Internet, oder andere Geräte, wie sie heute innerhalb von Gebäuden oder im Fernverkehr benutzt werden.Es ist durchaus möglich, vertrauliche Daten von Ort zu Ort oder von einer Zentrale an mehrere entfernte Empfänger zu übermitteln (Med).

Besondere Aufmerksamkeit muss der Akzeptanz dieser neuen Techniken durch die Patienten gewidmet werden, aber auch durch die Ärzte, das Pflegepersonal und die Mitarbeiter in der Verwaltung, deren tägliche Arbeit davon entscheidend beeinflusst wird. Ferner muss ein wirksames Datenschutzsystem eingerichtet werden. Dieses muss sowohl sicher als auch effizient und einfach sein (WN04).

## 2.2 Telemedizinsche Anwendungen

Telemedizin-Applikationen werden schon heute in mehreren Bereichen der Medizin eingesetzt. Unter den Verfahren, die sich bereits als nützlich erwiesen haben seien folgende erwähnt [(Die99), (Wik07)]:

1. **Telemonitoring** (Fernüberwachung) Die Fernüberwachung physiologischer Parameter erfolgt entweder direkt in Echtzeit (real time) oder verzögert, falls die Daten gespeichert werden. Das Monitoring kann am Patientenbett eingesetzt werden oder zur Beobachtung von Personen, die einem erhöhten Risiko ausgesetzt sind, sei es krankheitsbedingt oder auf Grund spezifischer Situationen, in der Wohnung(ältere Personen) oder in gefährlichem beruflichen Umfeld.

2. **Telediagnose** Bei der Telediagnose handelt es sich um Anwendungen wie Tele-Elektrokardiogramm,Tele-Dermatologie oder Tele-Endoskopie, bei denen eine ärztliche Untersuchung von, oder gemeinsam mit einem Arzt durchgeführt wird, der sich an einem anderen Ort befindet.

3. **Telekonsultation** Die Telekonsultation kann sich als sehr nützlich erweisen, wenn eine Zweitmeinung eingeholt werden soll, etwa bei komplexen oder seltenen Krankheitsbildern oder bei erhöhtem Risiko. Die Fernbefragung von Experten kann auch während einer Operation stattfinden.

4. **Tele-Arbeitssitzung** Selbst über geringe Entfernungen, zum Beispiel innerhalb einer Klinik, erlaubt die Telemedizin einen effizienten Informationsaustausch zwischen verschiedenen Betreuern. Dadurch kann eine qualitative Verbesserung und eine Beschleunigung der Verfahren und der Entscheidungsfindung erzielt werden, da die Notwendigkeit entfällt, die betroffenen Personen am selben Ort zu versammeln.

5. **Telescreening («Pre-gatekeeping»)** Dank medizinischer Zentralen oder Call-Centers können gewisse Gesundheitszustände aus der Ferne professionell beurteilt werden und Patienten anhand der festgestellten Symptome unverzüglich an den geeignetsten Ort überwiesen werden. Auf diese Weise lassen sich Arztbesuche vermeiden, die Effizienz ärztlicher Konsultationen steigern, und Spitäler werden nur noch falls unbedingt erforderlich in Anspruch genommen.

6. **Teledienste für Notfälle** Den Telephonzentralen von Notfalldiensten kommt ein erhöhter Stellenwert zu, wenn sie die Möglichkeit bieten, in schweren Fällen Experten beizuziehen, die bei der Beurteilung der übermittelten medizinischen Informationen und dem Entschluss über die einzuleitenden Massnahmen behilflich sind. Dadurch ergibt sich eine Optimierung der Gatekeeping-Funktion, und die richtigen Vorbereitungen bis zur Ankunft des Patienten können unverzüglich eingeleitet werden.

7. **Teleoperation** Da normalerweise sowohl Ärzte als auch Chirurgen verfügbar sind, eignen sich Teleoperationen eigentlich nur für Eingriffe auf dem Schlachtfeld mittels fernsteuerbaren Instrumenten (Telerobotik).

8. **Fernschulung («eLearning»)** Nutzung der IKT zur Weiterbildung der Angehörigen der Gesundheitsberufe.

   Die Verbindung oder Konvergenz von Medizin und IKT wird, neben «Telemedizin» oft auch als «Tele-Healthcare», «eHealth», Medizintelematik, Fernbehandlung, Patiententelemonitoring usw. bezeichnet.

## 2.3 Arten der Interaktion

Im Technischen Report ISO/TR 16056 „Health informatics  Interoperability of telehealth systems and networks“ [(MM00), (Org04a), (Org04b)] der International Standards Organisation (ISO) werden für den Bereich des Telehealth die drei Interaktionsarten Store-and-forward, Echtzeitsysteme und Medien-Streaming unterschieden (Abbildung 2.2). In der Praxis finden sich oftmals auch hybride Varianten.

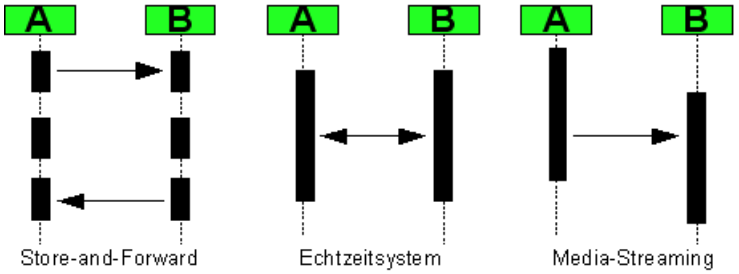



Abbildung 2.2: Telemedizinisches Interaktionsarten zwischen zwei Kommunikationspartnern A und B.

Abbildung 2.2 illustriert diese verschiedenen Kommunikationsarten. Die vertikalen Balken stellen dabei den zeitlichen Verlauf der Kommunikationsprozesse dar.

- Die **Store-and-forward (SF)** Technologie wird dazu verwendet, digitale Daten zu speichern („store“) und an einen entfernten Empfänger zu senden („forward“), wo sie von diesem, in der Regel zeitlich verzögert, eingesehen werden, und eine Befundung an den Sender mitgeteilt wird. SF verwendet eine asynchrone Form der Kommunikation und wird daher nicht für zeitkritische Aufgaben eingesetzt.

- **Echtzeitsysteme (ES)** werden dann eingesetzt, wenn eine interaktive, gleichzeitige Teilnahme aller Partner an einer Telemedizin-Sitzung nötig ist, sie verwenden also eine synchrone Form der Kommunikation. Über eine audiovisuelle Verbindung kann jeder Kommunikationspartner alle anderen Teilnehmer sehen und mit ihnen sprechen. Häufig werden dabei noch Zusatzgeräte eingesetzt, die es den Kommunikationspartnern ermöglichen, gleichzeitig relevante Dokumente, Bilder, etc., einzusehen, oder interaktiv Untersuchungen (Tele-Stethoskop, ...) oder Eingriffe (chirurgische Roboter) am Patienten vorzunehmen. In der Regel ist die für ES benötigte technische Ausstattung aufwendiger als die für die Interaktionsart SF und benötigt, je nach erforderlicher Bildauflösung, höhere Bandbreiten für die Verbindung.

- **Medien-Streaming (MS)** wird dann eingesetzt, wenn Daten unterschiedlichen Typs (Audio, Video, Dokumente, Standbilder, ...) direkt von der Datenquelle oder auch in gespeicherter Form über Netzwerke übertragen werden. Falls parallel mehrere Streams übertragen werden sollen, können diese zeitlich synchronisiert werden (z.B. Dokumente und begleitende Audiodaten).

## 2.4 Rechtliche Rahmenbedingungen

Bei Einsatz eines Telemedizinproduktes ist zu prüfen, ob es ein Medizinprodukt im Sinne des §3 des Medizinproduktgesetzes (MPG) ist (Har01):

*„Medizinprodukte sind alle einzeln oder miteinander verbunden verwendete Instrumente, Apparate, Vorrichtungen, Stoffe und Zubereitungen aus Stoffen oder andere Gegenstände einschließlich der für ein einwandfreies Funktionieren des Medizinprduktes eingesetzten Software, die vom Hersteller zur Anwendung für Menschen mittels ihrer Funktionen zum Zwecke*

*a) der Erkennung, Verhütung, Überwachung, Behandlung oder Linderung von Krankheiten,*
*b) der Erkennung, Überwachung, Behandlung, Linderung oder Kompensierung von Verletzungen oder Behinderungen,*
*c) der Untersuchung, der Erstezung oder der Verändeung des anatomischen Aufbaus oder eines physiologischen Vorgangs oder*

*d) der Empfängnisregelung zu dienen bestimmt sind und deren bestimmmungsgemäße Hauptwirkung im oder am menschlichen Körper werder durch pharmakologisch oder immunologisch wirkende Mittel noch durch Metalbolismus erreicht wird, deren Wirkungsweise aber durch solche Mittel unterstützt werden kann.“*

## 2.5 Datenschutz

Das Bundesdatenschutzgesetz (BDSG) verlangt in § 9 allgemein technische und organisatorische Maßnahmen zur Gewährleistung des Schutzes personenbezogener Daten. Die in der Anlage zu § 9 BDSG beschriebenen Regelungen definieren Sicherheitsmaßnahmen und haben im Wesentlichen die technischen Komponenten von Datenverarbeitungsanlagen zum Gegenstand. Im Folgenden werden die grundlegenden Sicherheitsziele definiert, die von Systemen zur medizinischen Datenverarbeitung gewährleistet werden müssen [(Bun07), (Duf)]:

- **Zutrittskontrolle :** Unbefugten den Zutritt zu Datenverarbeitungsanlagen, mit denen personenbezogene Daten verarbeitet oder genutzt werden, zu verwehren.

- **Zugangskontrolle :** zu verhindern, dass Datenverarbeitungssysteme von Unbefugten genutzt werden können.

- **Zugriffskontrolle :** zu gewährleisten, dass die zur Benutzung eines Datenverarbeitungssystems Berechtigten ausschließlich auf die ihrer Zugriffsberechtigung unterliegenden Daten zugreifen können, und dass personenbezogene Daten bei der Verarbeitung, Nutzung und nach der Speicherung nicht unbefugt gelesen, kopiert, verändert oder entfernt werden können.

- **Weitergabekontrolle :** zu gewährleisten, dass personenbezogene Daten bei der elektronischen Übertragung oder während ihres Transports oder ihrer Speicherung auf Datenträger nicht unbefugt gelesen, kopiert, verändert oder entfernt werden können, und dass überprüft und festgestellt werden kann, an welche Stellen eine Übermittlung personenbezogener Daten durch Einrichtung zur Datenübertragung vorgesehen ist.

- **Eingabekontrolle :** zu gewährleisten, dass nachträglich überprüft und festgestellt werden kann, ob und von wem personenbezogene Daten in Datenverarbeitungssysteme eingegeben, verändert oder entfernt worden sind.

- **Auftragskontrolle :** zu gewährleisten, dass personenbezogene Daten, die im Auftrag verarbeitet werden, nur entsprechend den Weisungen des Auftraggebers verarbeitet werden können.

- **Verfügbarkeitskontrolle :** zu gewährleisten, dass personenbezogene Daten gegen zufällige Zerstörung oder Verlust geschützt sind.

- **Nutzungsfestlegung :** zu gewährleisten, dass zu unterschiedlichen Zwecken erhobene Daten getrennt verarbeitet werden können.

Datenschutz aus Sicht des Patienten ist im Sozialgeseztbuch SGB V §140a Abs. 2 definiert. Hierbei muss eine Einwilligung des Patienten für die Übertragung von Daten vorliegen (Duf). Ein behandelnder Leistungserbringer darf aus dem der gemeinsamen Dokumentation nach § 140 Abs. 3 die den Versicherten betreffenden Behandlungsdaten und Befunde nur dann abrufen, wenn der Versicherte ihm gegenüber seine Einwilligung erstellt hat, die Information für den konkret austehenden Behandlungsfall genutzt werden soll und der Lietsungserbringer zu dem Personenkreis gehört, der nach § 203 des Strafgestezbuches zur Geheimhaltung verpflichtet ist.

## 2.6 Datensicherheit

Unter Sicherheit eines IT-Systems versteht man eine Eigenschaft eines IT-Systems, bei der Maßnahmen gegen die im jeweiligen Einsatzumfeld als bedeutsam angesehenen Bedrohungen der Integrität, der Verfügbarkeit und der Vertraulichkeit in dem Maße wirksam sind, dass die verbleibenden Risiken tragbar sind [(MB02),(Ber04)]. Neben den Datenschutzbestimmungen gibt es zur Zeit kaum anwendbare Normen für dieses Gebiet. Deshalb sind nachfolgend einige Risikobereiche aufgelistet, die sich in den letzten Jahren im Rahmen der praktischen Anwedung der Telemedizin als wichtig herausgestellt haben:

- Verfügbarkeit
  Schutz vor dem Ausfall von Systemen bzw. Komponenten: Systeme und Komponenten sollen so ausgelegt werden, dass deren autorisierte Nutzung in einen definierten Umfang möglich ist. Verfügbarkeitsanforderungen können an Hand prozentualer Werte bzgl. Der Gesamtbetriebsdauer bzw. durch die Angabe von Ausfallszeiten festgelegt werden.

- Integrität
  Zum Schutz vor Manipulation dürfen Daten nie von Personen oder Programmen verändert werden, wenn diese dafür autorisiert sind. Auch hier sollte durch eine Risikoanalyse geprüft werden, welchen Grad von Sicherheit die Autorisierung der Teilnehmr im Netz hat. Von einfachen Autorisierung wie Kennwort und Benutzername bis zu Smartcards ist je nach Applikation alles möglich.

- Vertraulichkeit
  Daten dürfen nur von Personen oder Programmen gelesen werden, wenn diese dafür autorisiert sind, um den Schutz vor unbefugter Kenntnisnahme sicherzustellen. Die Vertraulichkeit ist nicht nur ein technisches Problem. Um dieses Schutzziel zu erreichen, sind nach weitere Maßnahmen - wie in den Datenschutzgesetzen beschrieben - anzuwenden.

- Verbindlichkeit
  Der Schutz vor gefährlicher Identität sowie Schutz vor Abstreitbarkeit wird gelöst, in dem die Durchführung von Aktionen und die Herkunft von Daten eindeutig einer Identität (Person, Programm) zugeordnet werden. Dieser Vorgang wird häufig auch mit dem Begriff Revisionssicherheit verbunden.

# Kapitel 3

# Aufgabenstellung

## 3.1 Medizinischer Hintergrund

Die Anwendung der Kardiotokographie (CTG) stellt derzeit das weltweit eingesetzte Verfahren zur lückenlosen Intensivüberwachung des ungeborenen Kindes auf der Basis verschiedener Methoden dar. Sie bietet, dabei Dank ihrer hohen Sensitivität den entscheidenden Vorteil, im Sinne einer Screening-Methode, einen hypoxiegefährdeten [1] Feten herauszufinden, da eine defizitäre Sauerstoffversorgung in den meisten Fällen ein pathologisches Herzfrequenzmuster nach sich zieht. Somit kann bei unauffälligem CTG-Befund mit relativ großer Sicherheit von fetalem Wohlbefinden ausgegangen werden.

Die Kardiotokographie ist das am häufigsten angewendete Verfahren zur Überwachung von Schwangerschaften und Geburten. Dabei werden biomedizinischen Daten mithilfe eines Fetal-Monitors (CTG Gerät) gemessen, der ermöglicht sowohl die fetale Herzfrequenz (FHF) als auch die Wehentätigkeit der Mutter zu messen. Abhängig vom Gerätetyp können zusätzlich andere Messwerte (Signale) wie fetal/mutterliche Sauerstoffsättigung, Blutdruck, Kindesbewegung, etc erfasst werden. Die meisten dieser Geräte beschränken sich nur auf die Visualisierung und Ausdrucken der Messwerte. Die Beurteilung dieser Aufzeichnungen bleibt den klinischen Fachkräfte überlassen.

Von der fetalen Herzfrequenz werden Beurteilungsparameter bestimmt. Es handelt sich hierbei um lang-, mittel- und kurzfristige Merkmale, die aus der FHF-Verlaufskurve extrahiert werden: Basisfrequenz, Floatingline mit Akzelerationen und Dezelerationen sowie Variabilität (Oszillation). Es wird nach Bandbreite als auch nach Frequenz differenziert. Somit kann danach diese Aufzeichnung in eine der drei Standard Katogerien klassifiziert werden, nämlich normal, suspekt oder pathologisch.

Bei einer normalen Schwangerschaft werden ab der 28. Schwangerschaftswoche bis zur Geburt etwa acht CTG geschrieben. Bei einer Risikoschwangerschaft kann es aber sein, dass das Kind täglich bis zu drei mal überprüft werden muss. In aller Regel kann man aus die, als normal eingestuften FHF-Mustern zuverlässig auf fetales Wohlbefinden schließen. Das Hauptproblem ergibt sich aber daraus, dass die Mehrzahl der nicht als pathologisch

[1] Sauerstoffmangel

eingestuften FHF-Muster fälschlicherweise als normal eingestufft sind, was eigentlich zu eienr großen Gefährdung sowohl für die Mutter als auch das ungeborene Kind sein könnte. All diese Gründe haben dazu beigetragen, eine mobile Kardiotokographie-Lösung auszudeneken, die die Gesundheitssicherheit ohne wiederholte Krankenhausaufenthalt und die Bewegungsfreiheit der Patienten gewährleistet.

## 3.2 Systembeschreibung

*Trium Analysis Online* stellt für die Analyse und Auswertung aufgezeichneter Cardio-Toko-Grammen (CTG-Aufzeichnungen) eine ausgereifte Plattform zur Verfügung, deren Architektur und Datenfluss in den nächsten Abschnitten beschrieben werden.

### 3.2.1 Systemarchitektur

Abbildung 3.1 zeigt die vereinfachte schematische Architektur des Telemonitoring Systems („Trium CTG Mobile“). Der Telemonitoring Service besteht aus einem tragbaren fetal Monitor (CTG Gerät) und einer Übertragungseinheit, ausgestattet mit einem Mobilfunkmodul sowie eine zentrale Softwareverwaltung *CTG Mobile Server*. Zur Zeit wird der Personal Digital Assistant (PDA) als Übertragungseinheit verwendet.

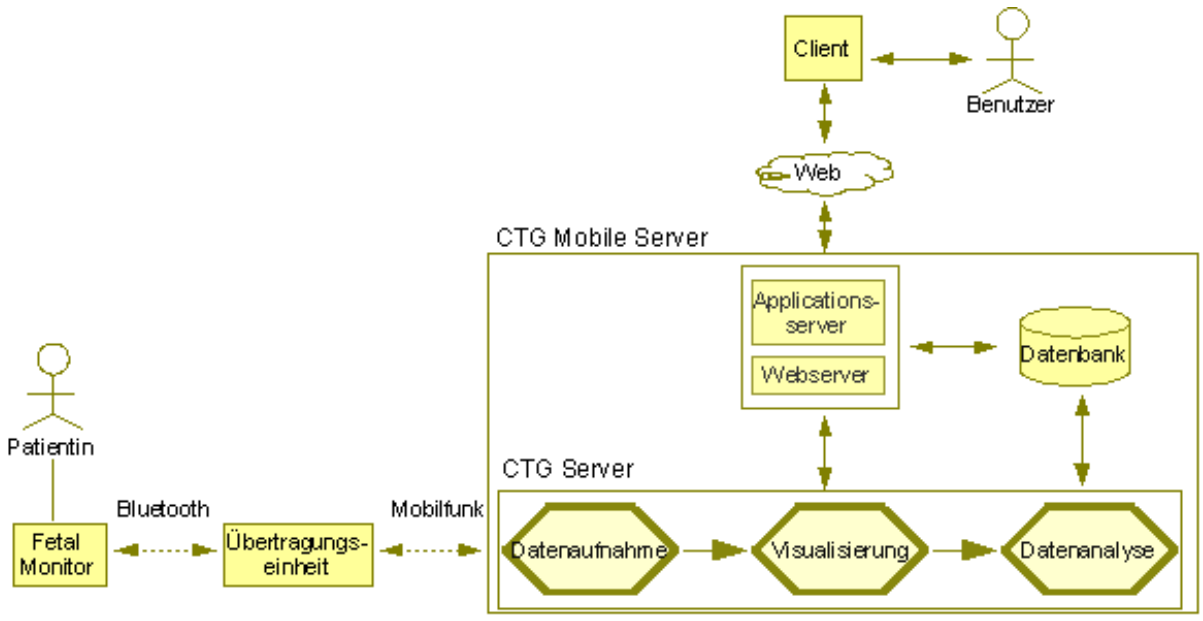



Abbildung 3.1: Schematische Architekturübersicht

Der CTG Mobile Server, wie ihm die Abbildung 3.1 zeigt, umfasst den Applikation Server inklusive Web Server, die Datenbank wie auch den Telemonitoring Server (CTG Server), der wiederum aus drei zentralen Modulen der Fernüberwachung von Schwangeren besteht. Das „Datenaufnahme“ Modul empfängt die übertragene Messwerte von der Übertragungseinheit

und übergibt die neu registrierten Daten an das nachgeschaltete Modul. Anhand des „Visualisierung“ Moduls können die erfassten Daten betrachet und mit Hilfe des Analysemoduls analysiert werden. Das „Datenanalyse“ Modul berechnet die zentrale Werte und erstellt aus diesen eine Klassifizierung des Zustandes („normal“, „suspekt“, „pathologisch“) der Patientin (Har01).

### 3.2.2 Datenfluss

Die Patientin wird zunächst an einem fetal Monitor angeschlossen, der alle benötigten Signale aufnimmt. Um die Signale weiterverarbeiten zu können, müssen die Daten im nächsten Schritt aus dem Gerät ausgelesen und an den CTG Mobile Server, den Kern der Anwendung, weitergegeben werden. Die Übertragungseinheit bildet die Verbindung zwischen CTG Gerät und CTG Mobile Server.

Nachdem das CTG Gerät angeschlosssen und aktiviert wurde, muss dann die Anwendungssoftware auf die Übertragungseinheit gestartet werden, die automatisch eine Verbindung über Bluetooth zum CTG Gerät zur Datenaufnahme aufbaut. Die Applikation bietet zwei Modi zur Übertragung der Daten:

- **Store-and-forward Modus** (siehe 2.3): Der sogenannte Offline Modus, bei dem die erfassten Daten lokal auf die Übertragungseinheit gespeichert und zu einem ausgewählten Zeitpunkt zum Server transferiert werden. Dieser Fall trifft ein, wenn der Anwender das vorab eingestellt hat oder keine Mobilfunkverbindung besteht.

- **Echtzeit Modus** (siehe 2.3): Der Online und Standard Modus in der CTG Mobile Anwendung. Hier wird parallel zu der Verbindung über Bluetooth zum CTG Gerät eine geschützte Verbindung über Mobilfunk (GSM [2], GPRS [3], UMTS [4]) zum zentralen Server und zwar zum „Datenaufnahme“ Modul initiiert.

Die Übertragung der Daten zwischen Übertragungseinheit und den Server basiert für beide Modi auf dem TCP[5]/IP[6] Protokoll.

Sobald eine Mobilfunkverbindung zum Server zur Verfügung steht, werden alle erfassten Messwerte in einer spezifizierten Konfigurationsdatei an das „Datenaufnahme“ Modul übergeben. Ist eine Aufzeichnung abgeschlossen, wird mit der Archivierung der Dateien begonnen. Dabei werden nur Referenzen auf die Dateien mit den Messwerten in der Datenbank gespeichert, sowie Informationen zum aufgezeichneten CTG, wie zum Beispiel die Start- und Endzeit der Registrierung und Übertragungsstatus etc. Die Dateien, die die Messwerte beinhalten, werden in einem externen Speicher abgelegt. Das verantwortliche

[2] Global System Mobile
[3] General Packet Radio Service
[4] Universal Mobile Telecommunications System
[5] Transmission Control Protocol
[6] Internet Protokol

Klinikpersonal wird danach per email, SMS oder Fax über eingehende und abgeschlossene CTG Übertragungen informiert.

Der CTG Mobile Server nimmt die Daten aus der Übertragungseinheit in Empfang, verarbeitet diese und visualisiert die Ergebnisse für die Darstellung auf dem Bildschirm. Die Ergebnisse des CTG Servers werden als Bilder über eine CGI[7]-Schnittstelle an Web-und Applicationserver übergeben, die die Bilder wiederum für den Client sich zur Verfügung stellt. So können die Benutzer auf ihrem Client jederzeit und von jedem Ort über ihre Browser die eingegangenen Daten On- oder Offline als Bilder anzeigen lassen, und haben somit die Möglichkeit Aufzeichnungen online zu bewerten. Der Application Server übernimmt außerdem die Verwaltung der Patienten- und Benutzerdaten sowie weiterer Informationen, die in der Datenbank gespeichert sind.

## 3.3 Ausgangssituation

Die in 3.2.1 vorgestellte Lösung bietet ein effizientes Fernüberwachungssystem, das dem Fachpersonal einen schnellen Überblick über den Zustand der Patientinnen und deren Feten ohne großen Aufwand für alle Beteiligten ermöglicht. Dies rechtfertigt die starke Nachfrage an dem Produkt, was eigentlich sehr gut für das Unternehmen ist, jedoch sind dadurch neue Herausforderungen entstanden.

Das derzeitige Geschäftsmodell ist nur für einzelne Mandanten [8] konzipiert worden. Die entstandenen Daten mehreren Mandanten gleichzeitig bereitzustellen, ist noch nicht möglich. Aus diesem Grund muss für jeden Kunden ein separater Server zur Verfügung gestellt werden, was zu einem hohen Wartung- und Installationsaufwand führt und dadurch hohen Kosten verursacht.

Ein weiteres Hindernis ist die nicht synchrone Auswertung der Daten. Das System kann zwar eine schnelle Übertragung der Messwerte sowie deren gleichzeitigen Visualisierung ermöglichen, jedoch kann es nicht sicherstellen, dass diese Aufzeichnungen innerhalb einer bestimmten Zeitspanne aufgrund der hohen Auslastung des Fachpersonales analysiert werden.

Außerdem ist eine Verwaltungsmöglichkeit aller tragbaren Geräte und SIM-Kartenverträge in diesem Modell nicht enthalten, was jedoch bei der aktuellen Größe des Systems notwendig wäre.

Der technische Hintergrund ist bei dem jetzigen Umfang des Systems nicht mehr ausgereift. Zusätzlich ist die verwendete Software kostenpflichtig. Als Beispiel sind Access, LabView, und Coldfusion zu nennen.

[7] Common Gateway Interface

[8] Klinik, Praxisgemeinschaft, niedergelassener Arzt, etc

## 3.4 Zielsetzung

Eine schnelle Auswertung der Aufzeichnungen und die damit verbundene sofortige Reaktion bei Risikoschwangerschaften lässt sich nur durch die Einführung eines telemedizinischen Zentrums (TMZ) gewährleisten. Das TMZ übernimmt die Realzeit-Überwachung der jeweiligen Mandanten und die dazu gehörigen Patientinnen. Zusätzlich werden die Mandanten durch weitere Dienste (technisch, administrativ, support), welche ebenfalls vom TMZ angeboten werden, noch mehr entlastet.

Das zu konzipierende Geschäftsmodell soll um den Aufwand zu reduzieren auf einem zentralen Server laufen und neue Verwaltungskonzepte, unterstützt von einem Systemalarmmanagement anbieten. Dies wird in Form einer Mutlti-Mandanten und Multi-TMZ Architektur realisiert.

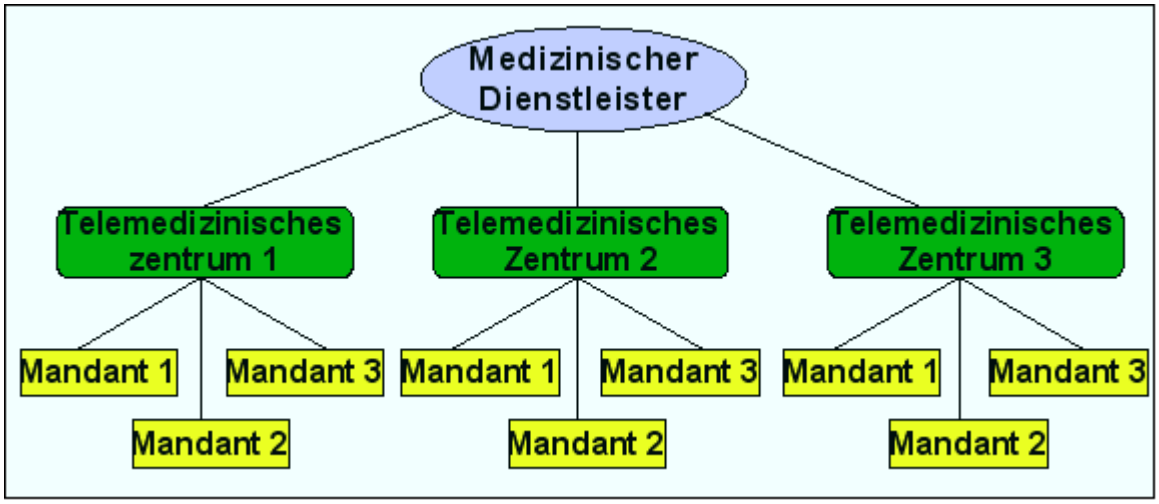



Abbildung 3.2: Konzeptschem der zukünftigen telemonitoring bei mobilen Schwangere

Abbildung 3.2 visualisiert das zu entwickelnde Konzept, in dem mehrere TMZ vom medizinischen Dienstleister (MD) aufgenomen und verwaltet werden können. Diese können wiederum mehrere Mandanten im System einfügen und verwalten. Der Medizinische Dienstleister (MD), in diesem Fall *„Trium Analysis Online“* betreibt den Server, und die damit verbundenen Dienste.

Im Rahmen dieser Arbeit soll diese Dienstleistung-Plattform entworfen werden und im Anschluss ein Prototyp entwickelt werden. Die Plattform soll die Dreieck-Architektur (Multi-Mandanten/TMZ, Abbildung 3.2) verwirklichen. Zusätzlich zu der Kernfunktionalität soll sie folgende Leistungen erbringen:

- Benutzerverwaltung
- Patienten- und Datenverwaltung
- Bereitstellung und Analysemöglichkeit von Aufzeichnungen

- Aufzeichnungsverwaltung
- Geräteverwaltung
- SIM-Kartenverwaltung

Die Implementierung soll allerdings auf die, zur Verfügung gestellten Frameworks (*Formengine*, *JwebbAppCore*) von *Trium* basieren. Dieses bietet den Vorteil bereits implementierte Klassen und Schnittstellen optimal zu wiederzuverwenden und eine ausgereifte Basis für die Weiterentwicklung des Prototypes bereitzustellen.

## 3.5 Aufbau dieser Arbeit

# Kapitel 4

# Geschäftsmodell

## 4.1 Einführung

Die enormen technischen Fortschritte im Bereich der mobilen Kommunikation sind im täglichen Leben allgegenwärtig. Die Weiterentwicklung der entsprechenden medizinischen Infrastruktur sowie des organisatorischen und rechtlichen Umfeldes folgen jedoch den technischen Möglichkeiten hinterher (WN02).

Selbstverständlich bedient sich auch die moderne Medizin neuester Technik und Kommunikationsmittel. Im Mittelpunkt steht der Patient. Schliesslich ist das Ziel klar:
*„Die medizinische Versorgung des Patienten soll stetig verbessert werden, wie es der jüngste Stand der Technologie erlaubt“* (JF02).

Gerade in der Schwangerschaftsüberwachung kann aber davon ausgegangen werden, dass telemedizinische Applikationen zu deutlichen Verbesserung der Handlungsabläufe führen. Den Schwangeren wird ein größeres Maß an Bewegungsfreiheit und Sicherheit durch die kontinuerliche Fernüberwachung gewährleistet, und das ohne wiederholte Krankenhaus-Aufenthalte.

In diesem Kapitel wird das Geschäftsmodell der Fernüberwachung mobiler Schwangeren „Telekardiotokographie“ mit dem eingefügten telemedizinischen Zentrum beschrieben. Zunächst werden die vielfältigen Missionen erklärt, die das TMZ übernehmen könnte, sowie welches Nutzen dadurch entstehen kann.

## 4.2 Szenario der Telekardiotokographie

### 4.2.1 Produkt Hintergrund

Das „Trium CTG Mobile“ Produkt besteht aus Software und einem Hardware Pack. Das Hardware Pack gliedert sich in einen tragbaren fetalen Moniror inklusive Bluetooth Adapter, und ein Mobilfunkmodul (PDA[1]), auf dem sowohl das Software-Modul für „Datenaufnahme“ als auch für „Datenübertragung“( siehe 3.1) bereits installiert ist. Darüber hinaus

[1]Personal Digital Assistant

stellt Trium eine Web-Applikation zur Visualisierung der Daten 3.1 für die Benutzer zur Verfügung. Der Erwerb dieses Produkts kann bis jetzt auf zwei Betriebsart erfolgen.

**Kaufmodell**

Hier kann der Mandant jederzeit Lizenzen für das Hardware Pack erwerben. ES wird somit die definierte Anzahl von Geräte im Vertrag für den Mandant bereitgestellt. Diese Geräte sethen ab dem Zeitpunkt für die Patientinen zur Verfügung. Folglich kann der Mandant die Web-Applikation auch nutzen, um die Geräte und Patietendaten zu verwalten sowie die Aufzeichnunegn online zu analysiseren.

**Mietmodell**

Bei dem Mietmodell hat der Mandant die Möglichkeit die Lizenzen für das Hardware Pack für einen bestimmten Zeitraum auszuleihen. Innerhalb der vereinbarten Zeitspanne darf der Mandant diese Geräte für die Datenaufnahme und Datenübertragung nutzen, zudem hat er den freien Zugang zur Web-Applikation, mit welcher die Daten betrachten kann.

### 4.2.2 Anwendungsablauf

Die Telekardiotokographie wird den Patientinnen nach Untersuchungen von ihrem Arzt verordnet. Damit wird die Schwangere nach Hause entlasstet. Die Patientin muss als nächstes eine Einverständniserklärung unterschreiben, in dem sie erklärt, dass sie die Vorteile, Nachteile und Risiken der Fernüberwachung in häuslicher Umgebung zur Kenntnis genommen hat.

Nach erfolgreicher Registrierung der Patientendaten durch Mitarbeiter des TMZs im System erfolgt die Einweisung der Schwangeren in das CTG Gerät und die Übertragungseinheit (PDA) vom Personal des Mandants beim Kaufmodell oder vom Mitarbeiter des TMZs beim Mietmodell.

Die Schwangere schreibt und sendet die mit ihrem Arzt vereinbarte Anzahl von CTGs. Besteht eine Verbindung in das Mobilfunknetz wird die Aufzeichnung grundsätzlich online übertragen, ansonsten zeitversetzt, sobald eine Verbindung verfügbar ist. Neben dem Echtzeitsmodus bietet das System auch die Möglichkeit die Daten im Store-and-forward Modus zu übertragen (siehe 2.2).

Jeweils nach Abschluss einer Aufzeichnung wird sie direkt von einem Medizinischer Betreuer des TMZs analysiert und diagnostiert. Die patientin wird bei einen auffälligen Befund darüber in Kenntnis gesetzt und es werden gegebenfalls Maßnahmen ergriffen (siehe Abbildung 4.2.2). Bei schwachen Abweichungen individuell festgelegter Grenzwerte werden umgehend therapeutische Maßnahmen mit ihrer Haus- und Klinikarzt eingeleitet. Und im Fall eines normalen CTG-Befundes obliegt es dem Medizinischen Betreuer, ob er die Patientin kontaktieren möchte oder nicht. Dies wird mit der grünen durchgestrichene Linie in Abbildung 4.2.2 illustriert.

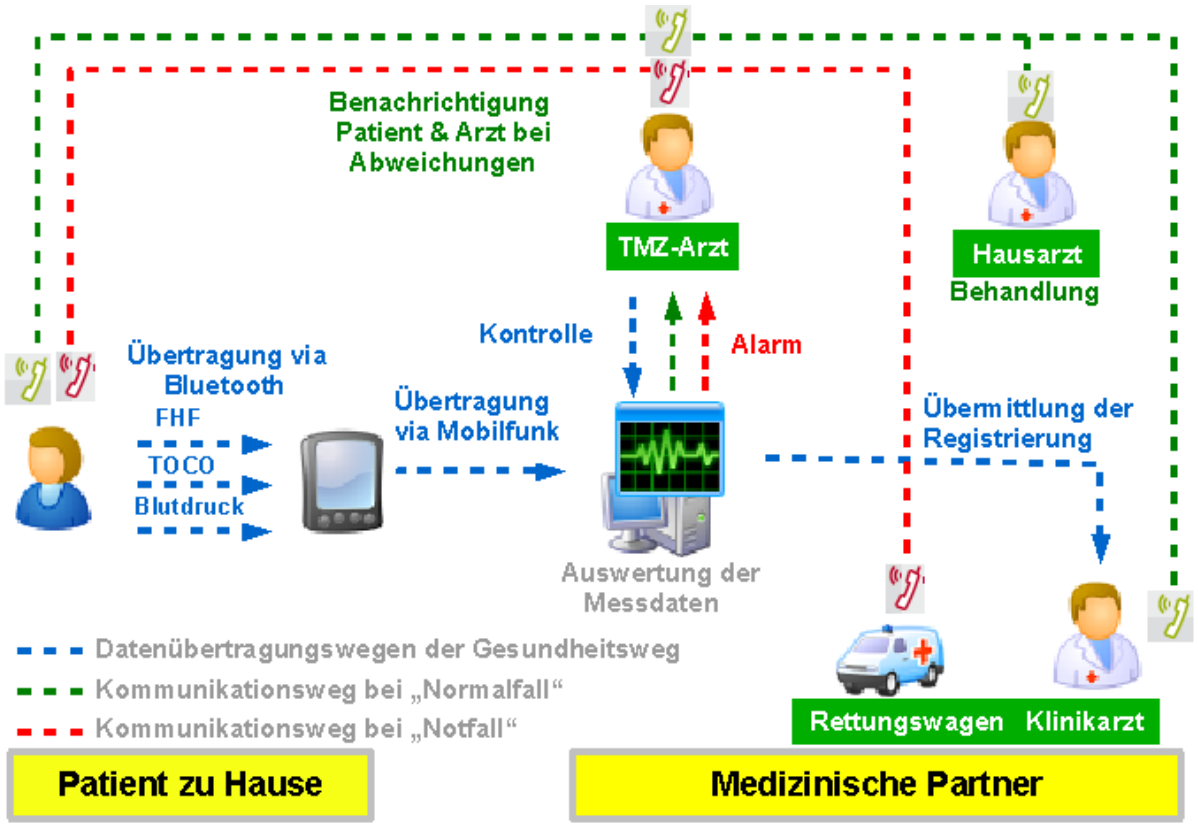



Abbildung 4.1: Szenario der telemedizinischen Betreuung

Bei einem suspekten CTG-Befund wird die Patientin durch ihren medizinischen Betreuer telefonisch aufgefordet, innerhlab der nächsten Stunden erneut ein CTG (Aufzeichnung) zu schreiben. Sollte dieses wiederum einen suspekten Befund ergeben, wird die Schwangere aufgefordert, sich zeitnah in der Klinik vorzustellen. Bei pathologischem CTG-Befund wird die Patientin telefonisch aufgefordert, sich unverzüglich in der Klinik einzufinden. Parallel wird das Mandant-Personal darüber benachrichtigt. Die sowohl suspekten als auch pathologischen Befunde werden mit der roten gestrichelten Linie in Abbildung 4.2.2 dargestellt.

Gleichzeitig wird der jeweilige Mandant auf elektronischem Wege automatisch darüber informiert (Email und/oder Fax). Bestandteil der Benachrichtigung ist der aktuelle Befundbericht, d.h. Ausdruck des CTG-Streifens sowie FIGO Beurteilung [2]. Das übertragene CTG kann jederzeit vom Mandanten und TMZ-Mitarbeiter Online ausgewertet werden.

Das Mandant-Personal legt täglich nach Eingang des CTG-Befundes unter Berücksichtigung der Krankengeschichte und des Ausgangsbefunds (sowohl vom medizinischen Betreuer als auch vom eignen Arzt oder Hebamme) neu fest, ob die Schwangere weiterhin zuhause überwacht werden kann, ob die Therapie geändert werden muss oder ob eine stationäre Einweisung erfolgen bzw. die Entbindung eingeleitet werden muss.

[2] Fédération Internationale de Gynécologie et d'Obstétrique

Nach der Entbindung bzw. bei einer stationärer Wiederaufnahme bei sich verschlimmerndem Befund übergibt die Schwangere das Gerät an ihren Mandanten (Kaufmodell) oder an Ihre TMZ (Mitemodell) zurück. Das Gerät muss anschließend im System wieder erfasst werden und kann erst danach einer neuen Patientin zugewiesen werden.

Der Einsatz des telemedizinischen Zentrums bietet sich daher als zentrales Service- und Informationsinstrument an, so dass durch eine konsequente und kontinuerliche Überwachung des Patienten eine optimierte Therapieführung ermöglicht wird.

## 4.3 Aufgabenkonstellation des telemedizinischen Zentrums

Das Kernstück der Fernüberwachung bei schwangeren Frauen ist das zentrale Telemedizinische Zentrum, welche unterschiedliche Funktionalitäten, insbesondere Patientenakte, Reporting / Analyse / Statistik abbildet. Dieses Zentrum steht unter fachärtzlicher Leitung und ist mit qualifiziertem und geschultem Fachpersonell (Medizin und Technik) so ausgestattet, dass ein ganzjähriger 24-Stdunden-Service garantiert werden kann. Nur so können im Notfall und/oder bei medizinischen Beschwerden auf der Basis optimierter Kommunikation (Hausarzt, Klinikarzt, Rettungssanitäter) entsprechende therapeutische Maßnahmen eingeleleitet werden (siehe Abbildung 4.2).

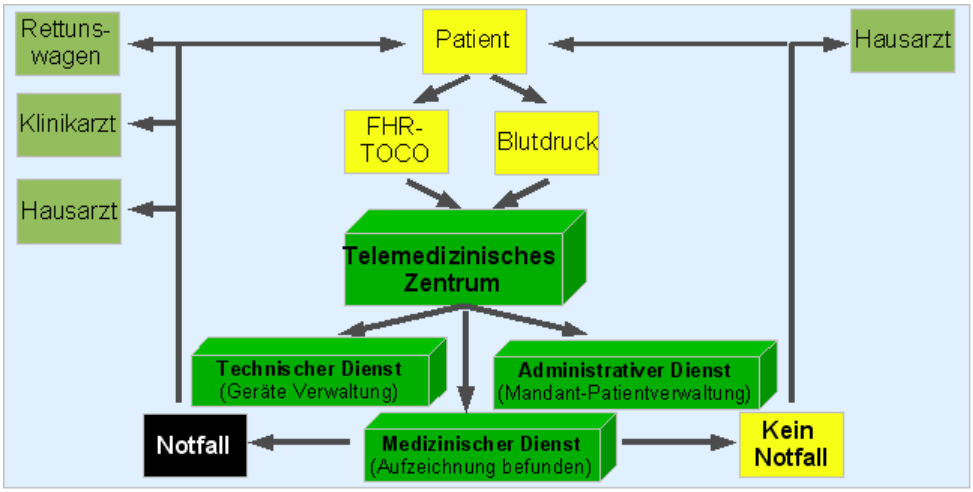



Abbildung 4.2: Schematische Darstellung der Funktionalität und Aufgabenkonstellation des Telemedizinischen Zentrums

Im folgenden wird auf die unterschiedliche Funktionsvielfalt des TMZs einegegangen, welche in vier Kategorien eingeteilt ist.

### 4.3.1 Medizinischer Dienst

Der Medizinische Dienst muss eine konsequente Überwachung des Patienten garantieren, andererseits aber auch eine kontinuerliche Therapieführung und Steuerung bei Patienten ermöglichen. Diser Dienst ist durch den Einsatz vom medizinischen Fachpersonal vertreten. Sie sorgen dafür, dass die Registrierung der Patienten sofort analysiert und bewertet werden, was folglich eine bessere Kontrolle und Überwachung der Patienten ermöglicht. Das medizinische Fachpersonal des TMZs reagiert sofort auf auffällige Messwerte und koordiniert rund um die Uhr entsprechende Maßnahmen: von der Kontaktaufnahme mit dem Patienten über die Absprache mit deren Hausarzt oder Klinikarzt, um die Therapie zu ändern bis zum Einsatz eines Rettungswagens im Notfall. Eine weitere Aufgabe des medizinischen Dienstes ist die Einweisung der Patientinnen in den Fetal-Monitor (CTG Gerät) und die Übertragungseinheit, so dass die Schwangere in Zukunft selber die Geräte anbinden und die Registrierung starten können.

### 4.3.2 Technischer Dienst

Mit steigender Anzahl der Mandanten steigt automatisch die Anzahl der zu verwaltenden tragbaren Geräte und damit der Aufwand. Deswegen wurde der technische Dienst geschaffen, der alles zentral regelt und somit einen gesamten Überblick über alle verwendeten Geräte und ihren Status beibehält. Dieser wird durch ausgebildete tecchnische Fachkräfte repräsentiert, die sich vor allem um die Wartung und Verwaltung der tragbaren Geräte (fetal Monitor, PDA, Bluetooth Adapter) sowie der SIM Karten für die Übertragungseinheit inklusive Verträge kümmern.

### 4.3.3 Administrativer Dienst

In Abbildung 4.2 kann man sehen, dass der administrative Dienst auch ein Teil des zukünftigen TMZ ist. Dieser Service ist für die Mandantenverwaltung und ihre Patientendatenverwaltung wie auch die Zuordnung von Patienten zu den tragbaren Geräte verantwortlich. Nur somit können die empfangenen Aufzeichnungen anhand der Übertragungseinheiten identifiert werden.

### 4.3.4 Support Dienst

Zusätzlich kann das TMZ eine weitere wichtige Aufgabe annehmen, nämlich der Support Dienst. Diese Service kümmert sich sowohl um technische als auch um administrative Fragestellungen, die eine große Bedeutung beim Erfolg solcher Plattform spielen. Somit können die Mandanten direkt bei ihrem TMZ eine Antwort auf ihre Unklarheiten oder Gebrauchanweisungen bezüglich der Funktionalität der Applikation. Derzeit bietet *Trium* kein separaten Support Dienst dafür, sondern es wird direkt vom Entwickler der Anwendung erledigt, was zu einer Beeinträchtigung dessen Arbeitsleistung führen kann. Das in der Zukunft durch das TMZ besser organisiert werden kann.

Durch die Einführung des TMZ-Konzepts bei der Fernüberwachung von schwangeren Frauen können sowohl die Patientinen als auch das Personal von Mandanten sowie der Anbieter der Plattform (Medizinischer Dienstleister - MD) profitieren. An der erste Stelle gewinnen die Schwangere mehr Bewegungsfreiheit und Sicherheit durch die konsequente dauerhafte Überwachung und zusätzliche Betreuung durch das TMZ-Fachpersonal. Einerseits dieses Konzept bietet eine große Auslastung für das Fachpersonal des Mandanten, welches sich nicht mehr um die Patientendateneingabe, Geräteverwaltung oder die rechtzeitige Analyse von Aufzeichnungen kümmern sollte. Denn alles wird nun zentral bei dem TMZ geregelt sein. Andererseits kann der MD davon auch einen Vorteil haben, in dem er vom Support Dienst befreiet wird. Was die Entwickler eine bessere Konzentration auf deren Aufgaben verschaffen kann.

# Kapitel 5

# Vorgenhensmodell

Softwareentwicklung ist ein iterativer Prozess, das aus verschiedenen Stadien von der Modellierung bis zur Systemevaluierung besteht. Es werden so viele Iterationen durchlaufen, bis das Programm die notwendige Reife erreicht. hat.

Das Vorgehen bei der Entwicklung von Software orientiert sich an einem strukturierten Vorgehensmodell oder Prozessmodell, das den allgemeinen Rahmen für den organisatorischen Prozess der Softwareerstellung festlegt (Fär03). Entwicklungsprozesse sind zu komplex, um sie ad hoc bearbeiten zu können. Vielmehr ist es notwendig, ein Erfolg versprechendes Vorgehen zu planen, seine Ausführung zu kontrollieren und bei Bedarf zu reagieren, also steuernd einzuwirken. Unterstützung bieten hierbei Vorgehensmodelle. Aufgrund der häufig wechselnden Situationen, sowie unterschiedlicher Randbedingungen, müssen diese Modelle flexibel sein und während der Entwicklungstätigkeit gegebenenfalls Adaptionsmöglichkeiten bieten.

Vorgehensmodelle geben den Systementwicklern Richtlinien für die Realisierung eines Projektes und Beinhalten Angaben über Verantwortlichkeiten innerhalb des Projekt, regeln den zeitlichen Ablauf, legen die zu erledigenden Aufgaben fest und stellen Methoden zur Erledigung dieser.

## 5.1 Vorgehensmodelle im Überblick

Um ein Projekt erfolgreich durchführen zu können, wird ein Softwareentwicklungsprozess angewendet. Darunter versteht man „die Summe aus einem Vorgehensmodell, den Tätigkeiten und Aktivitäten sowie den angewandten Methoden“. Das Vorgehensmodell „ist eine Beschreibung einer koordinierten Vorgehensweise bei der Abwicklung eines Vorhabens“. Dabei legt das Vorgehensmodell eine Reihe von Aktivitäten fest, sowie deren Input und Output (Artefakte). Weiterhin erfolgt eine „feste Zuordnung von Rollen, die die jeweilige Aktivität ausüben“ [(KH98), (Gna02)]. Die Rollen, und somit die Verantwortung für die Aktivitäten, werden den einzelnen Mitgliedern des Projektteams zugeordnet.

Zu unterschiedlichen Zeiten innerhalb eines Modellierungsprojektes sind unterschiedliche

Maßnahmen anzuwenden. Insofern hat es sich als vorteilhaft erwiesen, die Grundsätze in ein Vorgehensmodell einzubetten. Das Vorgehensmodell besteht aus mehreren, zeitlich aufeinanderfolgenden Stufen mit folgenden vier Grundphasen (BB04): Analysephase, Designphase, Implementierungsphase und Testsphase.

Für die Softwareentwicklung wurden zahlreiche, in Art und Umfang teilweise höchst unterschiedliche Vorgehensmodelle entwickelt. Tabelle 5.1 vergleicht die Eigenschaften von einigen paar bekannten Vorgehensmodellen. Man unterscheidet hier drei Idealstandards für Vorgehensmodelle auch SW-Prozess-Paradigmen genannt (Gna02):

- sequentiell/phasenorientiert („Wasserfall“)
- iterativ, auch evolutionär(„spiralförmig“)
- leichtgewichtig („agil“)

| Modell | Hauptziel | Antreibendes Moment | Benutzer | Charakteristika | Einsatzgebiet |
|---|---|---|---|---|---|
| Wasserfall | minimaler Mangement-aufwand | Dokumente | gering | sequentiell, volle Breite | sehr kleine Projekte |
| V-Modell | Max. Qualität (safe-to-market) | Dokumente | gering | sequentiell, volle Breite, Validation, Verifikation | Großprojekte |
| Prototypen | Risiko-minimierung | Code | hoch | nur Teilsysteme (horizontal oder Vertikal) | Im Rahmen von anderen Modellen |
| Spiral | Risiko-minimierung | Risiko | mittel | iterativ, Entscheidung pro Zyklus über weiteres Vorgehen | Nur für Großprojekte |
| XP (eXtreme Programming) | Min. Entwicklungs-zeit, Risiko-minimierung | Code | mittel | leichtgewichtig, sehr kurze Zyklen | Kleine Projekte |

Tabelle 5.1: Vergleich der Vorgehensmodelle

- Wasserfallmodel : Die erwähnten Grundphasen in Abschnitt 5.1 werden in diesem Modell im Prinizip sequentiell durchlaufen, wobei die Ergebnisse einer Phase als Input für die darauffolgende Phase dient (daher rührt auch die Bezeichnung „Wasserfall“: jede Phase ist quasi eine Kaskade).

- V-Modell : Erweiterung des Wasserfall-Modells durch Integration einer expliziten Qualitätssicherung. Im Gegensatz zu einem klassischen Phasenmodell (Wasserfall) werden im V-Modell lediglich Aktivitäten und Ergebnisse definiert und keine strikte zeitliche Abfolge gefordert.
- Prototypen-Modell : Frühzeitige Erstellung ablauffähiger Modelle (Prototypen) des zukünftigen Produkts zur Überprüfung von Ideen oder zum Experimentieren.
- Spiralmodell : Eine Softwareentwicklung durchläuft mehrmals einen aus vier Schritten bestehenden Zyklus mit dem Ziel, frühzeitig Risiken zu erkennen und zu vermeiden. Pro Zyklus kann dann ein Prozess-Modell oder eine Kombination von Prozess-Modellen zur Erstellung eines Teilprodukts oder einer Ebene eines Teilprodukts festgelegt werden.
- XP (eXtreme Programming) : beschreibt eine agile iterative Vorgehensweise beim Softwareentwicklungsprozess mit leichtgewichtiger Methodologie und wenig Dokumentation und Overhead.

Vor dem Hintergrund der Methodik der Softwarentwicklung war es notwendig, ein Vorgehensmodell für die Entwicklung auszuwählen, das uns bei der Durchführung von dieses Projekts unterstützen wird. Die Auswahl ist im folgenden Kapitel dargestellt.

## 5.2 Auswahl eines Vorgehensmodells

Bei der Auswahl eines Vorgehensmodells für ein Entwicklungsprojekt wird in der Praxis zumeist nicht die Eignung für das spezielle Projekt analysiert, obwohl einige Anforderungen und Rahmenbedingungen von Projektklasse zu Projektklasse und von Projekt zu Projekt erheblich variieren (KH98). In dieser Arbeit wurde das V-Modell als Vorgehensmodell verwendet, das mittlerweile als Standard in der Firma *trium* gilt ist, da es die bessere Qualitätssicherung und ausführlicher Dokumentation anbietet, was bei solchen kritischen Systemen von belang ist.

### 5.2.1 V-Modell

Das V-Modell ist einer der meisten verwendeten und ein weltweit anerkanntes Vorgehensmodell. Dieses Modell wird in der öffentlichen Verwaltung, als auch bei kommerziellen Softwareprojekten eingesetzt. Das V-Modell ist streng organisitionsneutral gehalten und beschränkt sich ausschließlich auf den technischen Entwicklungsgang. Es legt die Vorgangsweise“Was ist zu tun?“, die Methoden „Wie ist etwas zu tun?“ und die Werkzeuganforderungen „Womit ist etwas zu tun?“ exakt fest.

Die Basis für eine Vielzahl von Vorgehensmodellen bildet das allgemeine V-Modell nach Boehm. Das allgemeine V-Modell ist ein logisches Modell, das die während der Softwareentwicklung anfallenden Produkte in Beziehung zueinander setzt. Das V-Modell stellt

jedoch kein Vorgehensmodell wie z.B. das Wasserfallmodell dar, da es keine Aussagen über die zeitliche Reihenfolge der einzelnen Entwicklungsschritte macht (KH98).

Das V-Modell verdeutlicht durch explizite Hervorhebung der konstruktiven und integrativen Tätigkeiten und der prüfenden, d.h. verifizierenden und validierenden Tätigkeiten, dass das Testen eine zur Entwicklung gleichwertige Tätigkeit darstellt. Einigen Phasen des V-Modells wurden dazu geeignete Testszenarien zugeordnet. Auf jeder logischen Ebene des Entwicklungsprozesses findet eine Validierung bzw. Verifikation über entsprechende Anwendungsszenarien bzw.Testfälle statt, wie es die folgende Abbildung 5.1 zeigt.

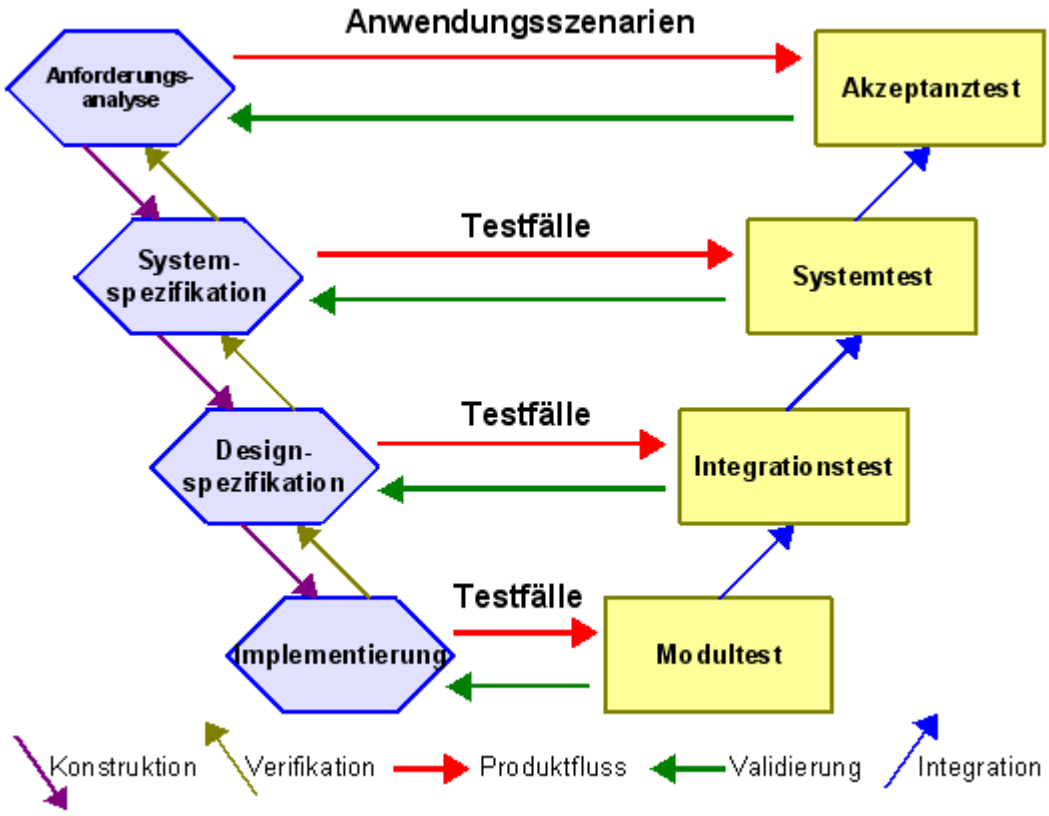



Abbildung 5.1: Schematische und stark vereinfachte Darstellung des V-Modelles [V-Modell 97]

In der oben stehenden Abbildung 5.1 sind auf dem absteigenden Ast die konstruktiven und verifizierenden Tätigkeiten während der Softwareentwicklung mit zunehmenden Detailierungsgrad der dabei entstehenden Artefakte dargestellt. Der aufsteigende Ast zeigt parallel dazu die integrierenden und validierenden Tätigkeiten und verdeutlicht dabei die zunehmende Größe des entstehenden Systems, je weiter oben man sich in der Darstellung befindet. Die Produkte auf der rechten Seite werden gegen die korrespondierenden Anforderungen und Spezifikationen auf der linken Seite validiert.

Bei der Verifikation wird durch geeignete Testfälle geprüft, ob der Entwickler die Software gemäß der Anforderungsspezifikation erstellt, d.h. ob die Software korrekt ist. Bei der Validierung wird über entsprechende Anwendungsszenarien die Eignung der Software bewertet. Dabei stellt sich die Frage, ob die richtige Software entwickelt wurde.

Das V-modell zeichnet sich auch durch die integrierte , detaillierte Darstellung von Systemerstellung , Qualitätssicherung , Konfigurationsmanagement und Projektmanangement aus. Das V-Modell gibt klar vor , in welchen Schritten und mit welchem Methoden und Techniken eine Software zu realisieren ist. Es wird genau festgelegt, welche funktionalen Eigenschaften die Tools aufweisen müssen, damit sie im Rahmen des Softwareentwicklungsprozesses zum Einstaz gelangen können.

### 5.2.2 Einsatz von Werkzeugen und Vorgehen

Es gibt viele unterstützende Programme, die entsprechende Schritten zu automatisieren erlauben und somit die Ablaufphasen des V-Modells überprüfen können. Allerdings wurde in diesem Projekt kein zusätzliches Werkzeug verwendet, denn man braucht ziemlich viel Zeit für die Einarbeitung in solche Tools, je nachdem wie umfangreich sie sind. Zweitens wurde aus Zeitknappheit auf solche Programme verzichtet, um eine prototypische Anwendung rechtzeitig darstellen zu können.

Wir werden in den nächsten Kapiteln gemäß der Abbildung 5.1 die einzelnen Phasen des allgemeinen V-Modells spezifizieren, und zwar bezogen auf die Anforderungen der zukünftigen Plattform der Telekardiotokographie. An dieser Stelle muss erwähnt werden, dass die Verifizierung parallel zur Konstruktionsphasen durch die Entwicklungsabteilung durchgeführt wurde.

# Kapitel 6

# Anforderungsanalyse

Die Anforderungsanalyse ist die erste Phase in dem V-Model (Abbildung 5.1), auf der in diesem Kapitel eingegangen wird. Im Gegensatz zu einem Pflichtenheft ist die Anforderungsanalyse ein zum Teil internes Dokument, bei dem es jedoch nicht unüblich ist, den späteren Benutzer hinzuzuziehen. Ausserdem wird eine Anforderungsanalyse im Gegensatz zum Pflichtenheft während des Software Engineering Prozesses angefertigt, d.h. falls sich Änderungen im System ergeben, so finden sich diese auch in der Anforderungsanalyse wieder.

Eine Anforderungsanalyse soll eine Modellierung eines Systems liefern. Dabei werden alle Aspekte der Anfordungen des Systems aufgegriffen. Die Anforderungsanalyse soll vor allem die Entwickler für die Systemarchitektur vorbereiten, da hier das System von mehreren Seiten mittels Modellierungsmethoden durchleuchtet wird(BB04). Zur Ermittlung der Anforderungen interviewt der Systemanentwickler, ausgehend von den Projektzielen, verschiedene potentielle Anwender und Verantwortliche um einen fundierten Einblick in die Aufgaben des Systems zu erlangen und zu einer ersten Version des Anforderungsdokuments zu kommen.

Die Anforderungsanalyse in diesem Projekt umfasst vor allem die Ermittelung der technischen Anforderungen, in der das Umweld beschrieben wird und Akteure sowie Anwendungsfälle identifiziert werden. Im Anschluß werden sowohl die relevanten technischen wie auch die medizinischen Anforderungen an das Geschäftsmodell definiert.

## 6.1 Teilnehmer und Rollen

Das Entwickeln eines Softwaresystems erfordert die Zusammenarbeit vieler Personen mit unterschiedlischen Hintergrund und verschiedenen Interessen. Man bezeichnt alle am Projekt beteiligten Personen als Teilnehmer („oder Projektteilnehmer“). Einen Satz von Verantwortungsbereichen in einem Projekt bezeichnen wir als Rolle [(BB04)]. Eine Rolle ist mit einem Satz von Aufgaben assoziert, die einem Teilnehmer zugewiesen werden, wobei derselbe Teilnehmer verschiedene Rollen gleichzeitig wahrnehmen kann.

| Rolle | Verantwortlichkeitsbereiche |
|---|---|
| Applikation Administrator (MD_Admin) | Diese Rolle wird an einem Mitarbeiter des MDs vergeben. Rechte: Verwaltung des TMZs und ihre Administrator Benutzerkonten. |
| Business Manager (MD_Bus_Man) | Diese Rolle wird an einem Mitarbeiter des MDs vergeben. Erhält ein aggregierten Überblick über alle CTG Gerät, PDA, Mandanten. Dazu können auch statistische Ergebnisse angezeigt werden (Anzahl der empfangenen Aufzeichnungen pro Mandant etc). |
| Telemedizinischer Administrator (TMZ_Admin) | Diese Rolle wird an einem Mitarbeiter des TMZs vergeben. Rechte: Benutzerverwaltung des eigenen TMZs. |
| Administrative Manager (TMZ_Admin_Man) | Diese Rolle wird an einem Mitarbeiter des TMZs vergeben. Rechte: Mandanten-, Patienten-, Benutzerverwaltung |
| Technik Manager (TMZ_Tech_Man) | Diese Rolle wird an einem Mitarbeiter des TMZs vergeben. Rechte: Geräte Verwaltung (PDA, fetale Monitoren) sowie SIM-karten |
| Telemedizinischer Befunder (TMZ_Bef) | Diese Rolle wird an einem Mitarbeiter des TMZs vergeben. Rechte: Analyse der empfangenen CTG, Abgabe der Kommentaren. Im Notfall die notwendigen Maßnahmen ergreifen. |
| Business Manager (TMZ_Bus_Man) | Diese Rolle wird an einem Mitarbeiter des TMZs vergeben. Erhält ein aggregierten Überblick über alle CTG Gerät, PDA, Mandanten. Dazu können auch statistische Ergebnisse angezeigt werden (Anzahl der empfangenen Aufzeichnungen pro Mandant etc). |
| User (M_User) | Diese Rolle wird an einem Mitarbeiter des Mandantes vergeben. Der Hauptbenutzer der Applikatio. Zur seinen Hauptaufgaben gehören:<br>• Patientenverwaltung<br>• Analyse der empfangenen CTG-Aufzeichnungen<br>• Abgabe der Kommentare<br>er gehört zum Fachkräftekreis der Klinik z.B: Arzt, Assistenzarzt, Hebamme, medizinisches Personal. |
| Zweitbefunder (M_Sec_User) | Wird konsiliarisch zur Bewertung der Befundung hingezogen. |
| Patient | Er schickt die CTG Aufzeichnung anhand des PDAs zum Server. |

Tabelle 6.1: Rollen der telemedizinischen Anwendung für mobile Schwangerschaftsüberwachung

Tabelle 6.1 stellt die existierenden Rollen und ihre Bescheibungen in der Applikation dar, die in drei Kategorien unterteilt sind:

- Lila sind alle Rollen markiert, die von einem Mitarbeiter des medizinischen Dienstleisters angenommen werden können.
- Grün sind alle Rollen markiert, die von einem Mitarbeiter des telemedizinischen Zentrums angenommen werden können
- Gelb sind alle Rollen des Mandant markiert, die von einem Mitarbeiter des Mandantes angenommen werden können

Generell können die Benutzer des jeweiligen Institutes (MD, TMZ) verschiedene Rollen wahrnehmen (d.h. mehrere Rollen annehmen). Allerdings ist das bei den Mandant-Benutzer nicht möglich, die können entweder die Rolle„M_User“ (Hauptuser) oder „M_Sec_User“ (Zweitbefunder) annehmen. Denn es kann nicht sein, dass ein Benutzer gleichzeitig der Hauptuser und der Zeitbefunder vom gleichen Mandant ist.

Welche Operationen diese Akteure im System durchführen können, wird im nächsten Unterkapitel 6.2.1 beschrieben.

## 6.2 Systemanforderungen

Anforderungen spezifizieren einen Satz von Aufgaben, der für einen bestimmten Zweck ausgeführt wird, dass wiederum vom System unterstützt werden muss.

Zunächst werden die technischen Anforderungen bezüglich der gesamten Aufgabenstellung vorgestellt. Danach werden die im telemedizinischen Umfeld typischen Anforderungen erläutert. Zum Schluss werden die fachlichen Ansprüche dargestellt, die jede Anwendung zu erfüllen hat.

### 6.2.1 Technische Anforderungen

Zur Modellierung und Spezifikation der relevanten Prozesse in der Nutzungsumgebung eines angestrebten Software-Produkts, d. h. der funktionellen Rollen und ebenso der Struktur und Funktionalität, die vom jedem zu fordern sind, sind seit Mitte der 90er Jahre sog. „Nutzungsfälle“ (Use Cases) im Bereich der OOSE [1] sehr beliebt [(For07)]. Die zugehörigen Konzepte und Beschreibungsmittel sind nicht zuletzt auch Bestandteil des OO-Spezifikationsrahmens UML [2]. Einschlägige UML-Werkzeuge unterstützen daher die Modellierung und Spezifikation von Use-Cases. Bei der objektorientierten Analyse sind sie mittlerweile ein bewährter Einstiegspunkt zur Erstellung der Systemspezifikation (siehe 7).

Mit der Einführung der Idee der Use cases durch das Buch [(Jac92)] von I. Jacobson und der Übernahme des Konzeptes in UML, gibt es heute eine Vorgehensweise, die es

[1] Objektorientierte Softwareentwicklung
[2] Unified Modelling Language

ermöglicht, Anforderungen übersichtlich, leicht verständlich und doch hinreichend präzise zu erfassen bzw. darzustellen.

Während der Anforderungsanalyse wurden die technischen Anforderungen und somit der Verwendungszweck des Systems anhand der Anwendungsfalldiagramme definiert, die das Systemverhalten in Anwendungsfälle beschreibt. Das Anwedungsfalldiagramm modelliert Anwendungsfälle eines Systems aus der Sicht externer Akteure.Das Ergebnis dieser Aktivität ist eine Beschreibung des Systems mit Akteuren und Anwendungsfälle. Akteure repräsentieren die externen Entitäten, die mit dem System nach erfolgreicher Authentifizierung interagieren können. Anwendungsfällen sind allgemeine Abfolgen von Ereignisse, die mögliche Aktionen zwischen einem Akteur und dem System für eine bestimmte Funktionalität beschreiben.

Es stehen viele Werkzeuge zur Entwicklung von UML-Use Cases zur Verfügung, unter anderem können folgende UML-Tools verwendet werden: Poseidon von Gentleware [(Pos)], ArgoUML von Tigris.org [(Agr)], Enterprise Architect von Sparx Systems [(EA)] etc verwendet. In diesem Projekt wurde allerdings das Microsoft Office Tool Visio (Vis) zur Erstellung von Anwendungsfalldiagrammen eingesetzt, die sind in drei Kategorien unterteilt haben.

#### 6.2.1.1 Anforderungen am Medizinischen Dienstleister

Das Use Case-Modell (Abbildung 6.2.1.1) illustriert möglichst vollständig die gesamte geforderte Funktionalität des zukünftigen Applikationsplattform aus der MD-Sicht (oberste Instanz in der Dreieck-Architektur 3.2) einer oder mehrerer funktionellen Rollen des Nutzungsbereichs. Im unteren modellierten Szenario lassen sich die folgenden Nutzungsfälle rausfiltern:

- **TMZ Verwaltung** für die Registrierung neuer telemedizinischen Zentren sowie zur Bearbeitung Bestehender TMZs im System.
- **TMZ Admin Verwaltung** hier lassen die Admin-Benutzerkonten des jeweiligen TMZs verwalten.
- **Vertretungsdienst** ist für die Durschführung und Bearbeitung der Vertretung eines TMZs von einem anderen Institut zuständig.

  Die drei genannten Anwendungsfälle können vom MD-Mitarbeiter, der in der Rolle „MD_App_Admin“ (Applikation Manager) vertreten ist, durchgeführt werden.
- **Statistische Sicht** bietet den Benutzer eine zusätzliche statistiche Sicht aller zugänglichen Datensektionen. Besipiel: Material (fetale Monitore, PDA, etc), Information (TMZ , Mandanten, ..etc). Ist der MD-Mitarbeiter in der „MD_Bus_Man“(Business Manager) Rolle, wird ihm diese Sicht angeboten.

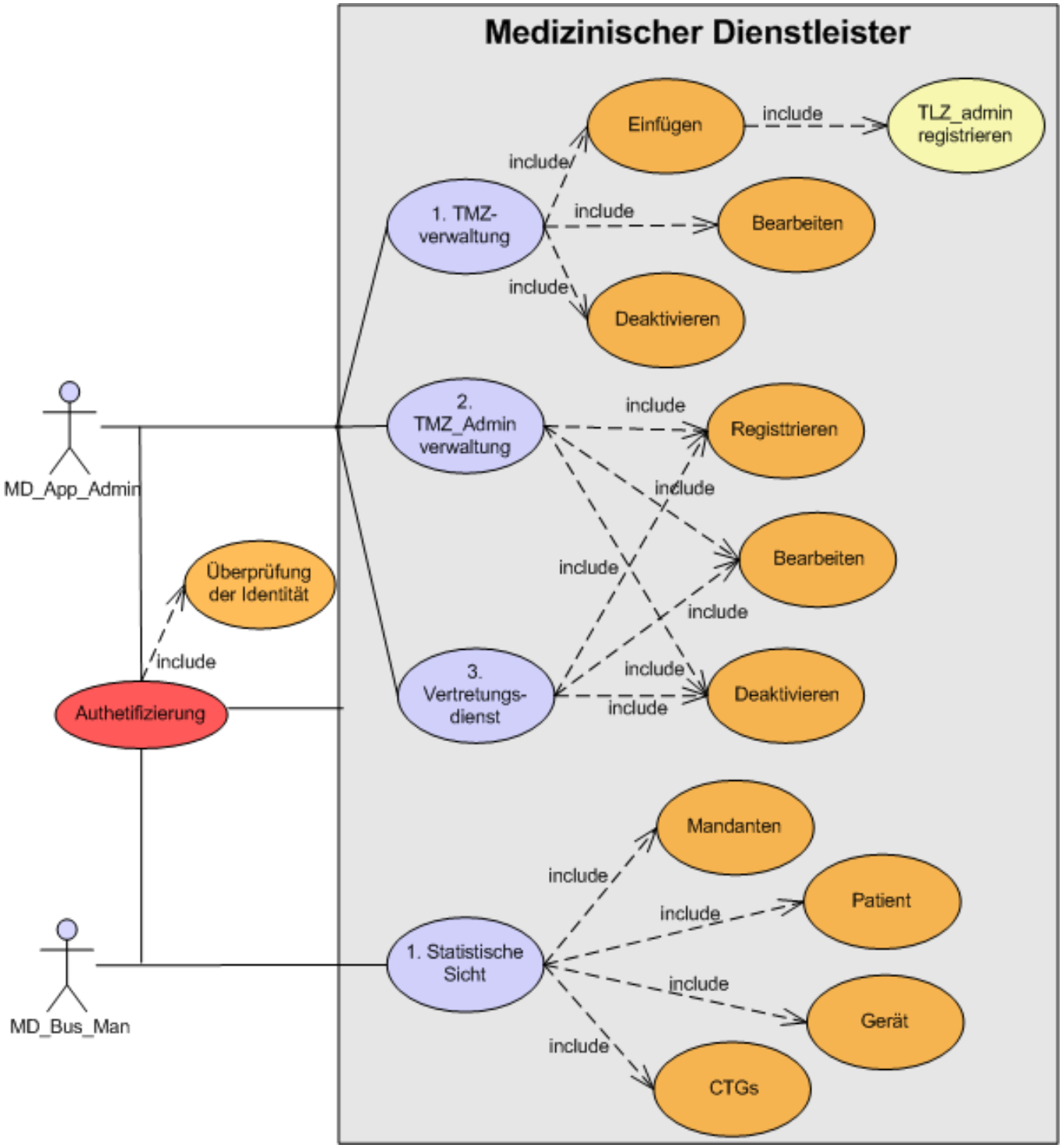



Abbildung 6.1: Medizinischer Dienstleister Sicht

#### 6.2.1.2 Anforderungen am Telemedizinischen Zentrum

Das Use Case-Modell Abbildung 6.2.1.3 beschreibt möglichst vollständig die gesamte geforderte Funktionalität der zukünftigen Applikationsplattform aus der TMZ-Sicht(mittlere Instanz in der Dreieck-Architektur 3.2) einer oder mehrerer Funktionellen Rollen des Nutzungsbereichs. Im unteren modellierten Szenario lassen sich die folgenden Anwendungsfälle unterscheiden:

- **Benutzerverwaltung** zur Verwaltung der eigenen TMZ-Benuzer sowie auch deren Benutzerkonten mit der zugehörigen Rollen. Dieser Anwendungsfall könnte von einem TMZ-Mitarbeiter, der in der „TMZ_Admin“ Rolle ist, übernommen werden.

- **Mandantenverwaltung, Benutzerverwaltung, Patientenverwaltung** umfasst alle organisatorischen Operationen, um vor allem Mandanten, deren Benutzer (sowie deren Benutzerkonten) und deren Patienten im System zu verwalten. Diese Anwendungsfälle können vom TMZ-Administrative Personal, der in der Rolle „TMZ_Admin_Man“ übernommen werden.

- **fetal Monitor-, PDA-, und Vertragsverwaltung** enthalten alle technischen Operationen, um das notwendige Hardware-Material für die Funktionalität des Systems inklusive deren Zubehör (SIM-Karte) in der Applikation zu verwalten. Dies wird vom TMZ-Techniker, der sich in der Rolle „TMZ_Tech_Man“ befindet.

- **CTG Aufzeichnungen** sowie die dazu gehörigen Funktionen, um Aufzeichnungen der betreunden Mandanten (bzw deren Patientinnen) zu bearbeiten, zu analysieren und zu bewerten. Zusätzlich könnten jederzeit Kommentare abgegeben werden. Diese Funktionen können nur vom TMZ-Medizinischer Befunder, der in der Rolle „M_TMZ_Bef“ befindet, ausgeführt werden.

- **Statistische Sicht** bietet den Anwender eine statistiche Sicht aller zugänglichen Datensektionen (je nach Rolle). Beispiel: Material (fetale Monitore, PDA, etc), Information (Mandaten , Patienten, Anzahl der Aufzeichnungen..etc). Wenn der TMZ-Mitarbeiter in der „TMZ_Bus_Man“(Business Manager) Rolle ist, wird ihm diese Sicht angeboten.

All diese Anwendungsfälle können miteinader kombiniert werden, d.h ein Benutzer könnte sowohl für die Patientenverwaltung als auch für die Verwaltung der tragbaren Geräte zuständig sein, Wenn der Benutzer die entsprechenden Zugriffsrechte (Rollen) dafür hat.

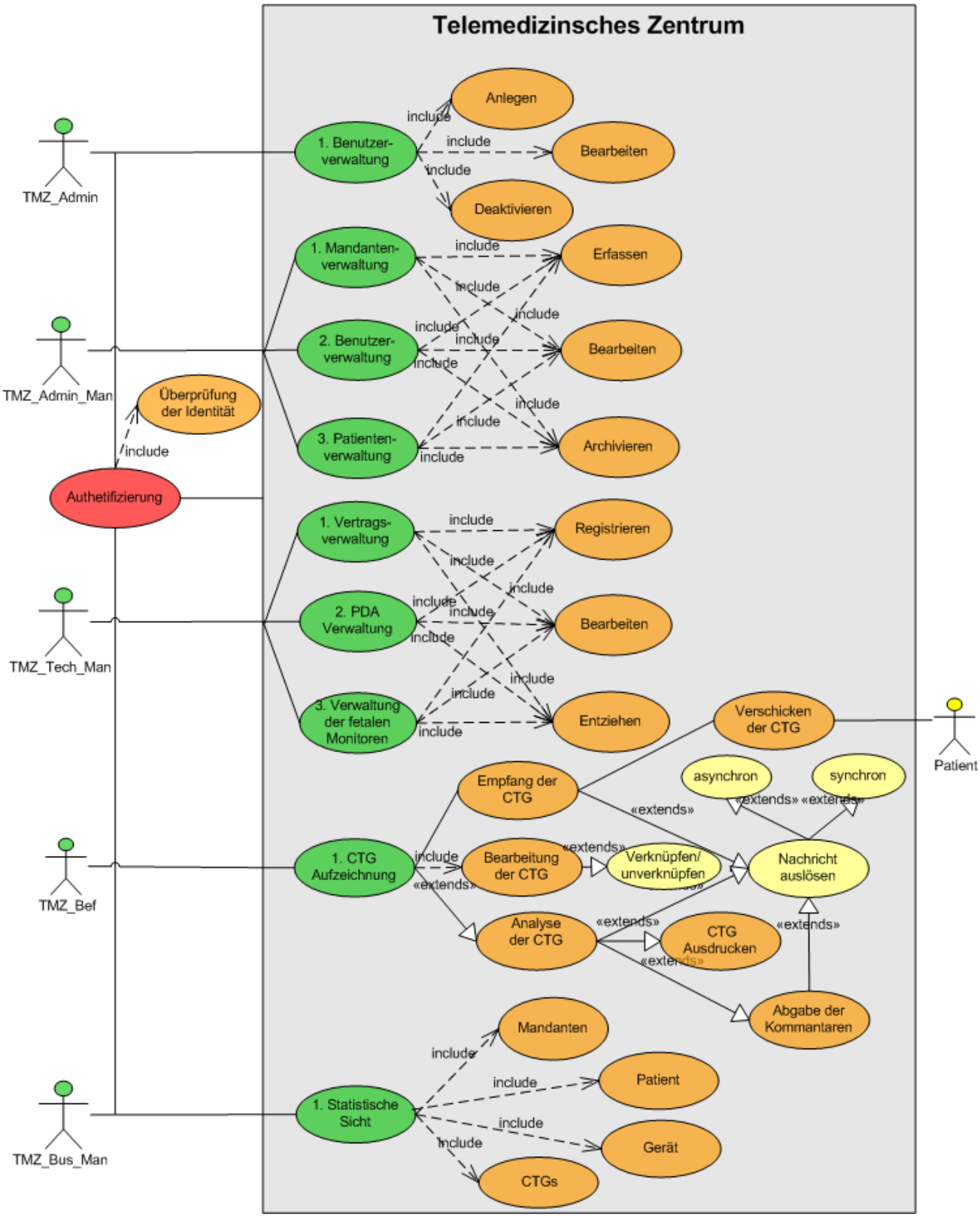



Abbildung 6.2: Telemedizinisches Zentrum Sicht

### 6.2.1.3 Anforderungen am Mandant

Das Use Case-Modell (Abbildung 6.2.1.3) beschreibt möglichst vollständig die gesamte geforderte Funktionalität der zukünftigen Applikationsplattform aus der Mandant-Sicht (unterste Instanz in der Dreieck-Architektur 3.2) einer oder mehrerer funktionellen Rolllen des Nutzungsbereichs. Im oben modellierten Szenario lassen sich die folgenden wichtigen Anwendungsfälle identifizieren:

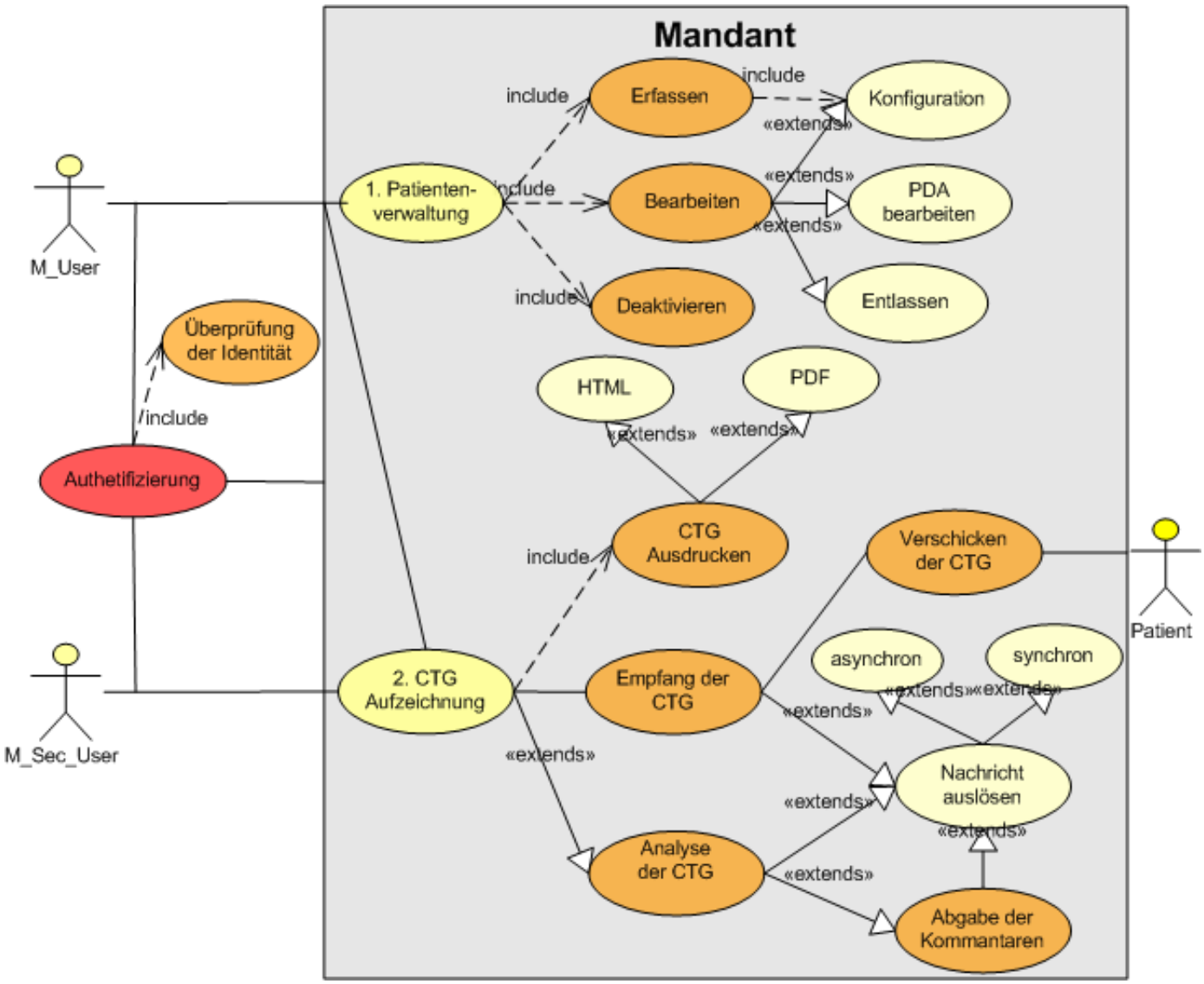



Abbildung 6.3: Manadant Sicht

- **Patientenverwaltung** im System mit allen möglichen Funktionen zur Bearbeitung

von Patientendaten. Dies wird hauptsächlich vom Fachpersonal des Mandanten (Rolle: M_User) übernommen.

- **CTG Aufzeichnung** und die dazu gehörigen Funktionen, um Aufzeichnungen zu bearbeiten, zu analysieren und zu bewerten. Dazu können Kommentare abgegeben werden. Sowohl der M_User als auch M_Sec_User kann diesen Anwendungsfall übernehmen. Der M_Sec_User bearbeitet nur die ihm vom M_User zugewiesenen Aufzeichnungen und gibt einen Befund ab.

### 6.2.2 Mediziniche Anforderungen

#### 6.2.2.1 Maßnahmen zum Datenschutz

Nachfolgend sind einige Maßnahmen beschrieben, die zum Datenschutz innerhalb eines Telemedizinnetzes notwendig sein können (WJF05):

- Im Rahmen eines Telemedizinnetzes muss sichergestellt werden, dass nur derjenige Zugang zu Daten eines Patienten hat, der an der Behandlung oder Versorgung des Patienten beteiligt ist.

- Im Rahmen eines Telemedizinnetzes muss sichergestellt werden, dass nur derjenige Zugang zu Daten eines Patienten erhält, der vom Patienten autorisiert wurde (Berechtigungskonzept).

- Für die Informationsverarbeitung in einem Telemedizinnetz ist eine ausdrückliche Einwilligung des Patienten erforderlich.

- Es dürfen nur Daten verarbeitet bzw . übermittelt werden, die tatsächliche für die Behandungen und Versorgung erforderlich sind (Grundsatz der Erforderlichkeit).

- Im Rhamen der Systemkonzeption muss permanent eine qualifizierte, risikoorientierte IT-Sicheriheitsüberwachung erfogen. Diese IT-Sicheriheitsüberwachung wird beispielweise über einen beauftragten Datenschutzbeauftragten überwacht.

- Die Leistungspartner sind in fallbezogenen Arbeitsgruppen organisiert und treffen untereinander geeignete Vereinbarungen, welche die notwendige Verlässlichkeit schaffen. Die Verlässlichkeit umfasst auch die Festlegung der zu übermittelnden Informationskategorien. Diese Definition der zu transferierenden Information steht dem Patienten auf Verlangen zur Verfügung.

- Die Datenorganisation erfolgt arbeitsgruppenorientiert. Bei ungesicherter Kommunikation bzw. im Abrufsverfahren erfolgt die Bereitstellung von Datendateien anonymisiert.

#### 6.2.2.2 Sicherheitsmaßnahmen

In Telemonitoringsystemen müssen geeignete technische und organisatorische Maßnahmen getroffen werden, damit die erfordliche Vertraulichkeit, Integrität , Verfügbarkeit und Authentizität der Patientendaten gewährleitet bleibt. Ein qualifiziertes IT-Sicherheitskonzept kann beispielweise auf der Basis einer Risikoanalyse erstellt werden. Diese stellt dar, welche Maßnahmen getroffen werden müssen, um auftretende Risiken auszschließen. Ein IT-Sicherheitskonzept deckt beipielsweise (je nach technischer Struktur und Datenstruktur des Netzes) folgende Risikobereiche ab (WJF05):

- Alle Datenübertragungswege zwischen den Kooperationspartnern
- Die Schnittstellen zu den internen Netzen
- Die interne Netze (soweit relevant)
- Die Arbeitsplatzrechner
- Die Nutzer

Folgende Maßnahmen können in Rahmen einer organisationsweiten Sicherheitspolicy getroffen werden:

- Schutz der internen Netze über Brückenkopflösung mit Firewall und Viruswall
- Kommunikationssicherheit durch Verschlüsselung
- Mehrschichtige Zugriffsebenen:Systemebene, Anwenderebene, Datenbankebene
- Mehrschichtiges Berechtigungskonzept je Sicherheitsebene
- Koordination und Vereinhetlichung der Sicherheitsanwendungen
- Rechtliche Maßnahmen gegen interne Scans (Ausspionieren der Struktur)
- Logging und Auditing der Zugriffe
- Analyse sicherheitsgefährdender Software auf Zugangsystemen
- Weitegehende Standardisierung des Zuganges und der Technik (nur ein spezifischer Client, kein Web-basierte Zugang)
- Erstellung von Richtlinien für Endanwender , u. a. Handhabung Kennwörter , keine Durchführung von „Security-Testprogrammen“, Zugang zum System, keine Installation unbekannter Software auf Zugangssystemen
- Elimination „ausführbarer“ Dateinanhänge im e-Mail-transfer einschließlich Bildschirmschonerdateien

### 6.2.3 Fachliche Anforderungen

Neben den technischen Anforderungen soll die zukünftige Applikationsplatttform die Normen zur Software-Qualität einhalten. Vor allem die Softwarequalität gemäß die Norm DIN 66272 bzw. die Reihe ISO/IEC 9126 erfüllen. Man unterscheidet hier sechs fundamentale Qualitätsmerkmale [(Fra07)]:

- Funktionalität : die im Pflichtenheft festgelegten Anforderungen werden durch definierte Funktionen erfüllt. Sie enthält : Korrektheit, Angemessenheit, Interoperabilität, Ordnungsmäßigkeit, Sicherheit.
- Zuverlässigkeit : Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren. Besteht aus : Reife, Fehlertoleranz, Wiederherstellbarkeit
- Benutzbarkeit : erforderlicher Benutzungsaufwand und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe. Inklusive : Verständlichkeit, Bedienbarkeit, Erlernbarkeit, Robustheit
- Effizienz : Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen. Besteht aus : Wirtschaftlichkeit, Zeitverhalten, Verbrauchsverhalten
- Änderbarkeit : erforderlicher Änderungsaufwand. Änderungen können Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen und der funktionalen Spezifikationen einschließen. Sie enthält unter anderem : Analysierbarkeit, Modifierbarkeit, Stabilität, Prüfbarkeit
- Übertragbarkeit : Eignung der Software, von einer Umgebung in eine Andere übertragen zu werden. Umgebung kann die organisatorische Umgebung, Hardware- oder Software-Umgebung einschließen. Inklusive : Anpassbarkeit, Installierbarkeit, Konformität, Austauschbarkeit

# Kapitel 7

# Systemspezifikation

Die Systemspezifikation beschreibt alle funktionalen und nicht-funktionalen Anforderungen an ein Systemelement (System, Unterstützungssystem oder Segment). Zur Erstellung einer Systemspezifikation werden die Anforderungen aus den Spezifikationen übergeordneter Systemelemente, beziehungsweise der Gesamtsystemspezifikation abgeleitet. Die Spezifikation dient als Vorgabe und Hilfsmittel zum Entwurf und Dekomposition der Architektur. Sollten im Laufe der weiteren Entwicklung des Systemelements Änderungen nötig sein, ist zunächst immer die Systemspezifikation anzupassen.

Wesentliche Inhalte der Systemspezifikation sind die Beschreibung der Anforderungen an das Systemelement und die Festlegung der Schnittstellen, die es zu bedienen hat. Zusätzlich wird die Verfeinerung und Zuordnung von Anforderungen und Schnittstellen zu untergeordneten Systemelementen beschrieben.

Im Rahmen der Anforderungsverfolgung wird sichergestellt, dass alle Anforderungen an das Element bei der Verfeinerung auf die nächste Hierarchieebene berücksichtigt werden. Die Erstellung der Systemspezifikationen erfolgt Hand in Hand mit der Designspezifikation des Systems (Abbildung 5.1) bzw. eines Unterstützungssystems.

## 7.1 Spezifikation der Use Cases

Die in der Unified Modeling Language (UML) (Gud06) standardisierte Notation zur grafischen Darstellung der Use Cases (siehe Abschnitt 35) kann zusätzlich als Übersicht über die textuell beschriebenen Use Cases verwendet werden. An dieser Stelle sei erwähnt, daß Use Cases für sich allein gesehen nicht objektorientiert, sondern ablauforientiert sind. Dies zeigt sich daran, daß sie das Verhalten des Gesamtsystems beschreiben. Gerade wegen dieser ablauforientierten Sichtweise bieten sie sich zur Beschreibung von Anforderungen an das dynamische Verhalten bei der Entwicklung komponentenbasierter Softwaresysteme an. In der Phase der Systemspezifikation werden die softwarerelevanten Abläufe eines zu entwickelnden Systems untersucht und in typischen Szenarien, sogenannten Nutzungsfällen

(Use Case), als Text beschrieben. Vorher müssen aber alle infrage kommenden Use Cases von der vorherigen Phase rausgefiltert werden.

### 7.1.1 Aufstellung der Use Cases

Die Tabelle 7.1 listet auf, alle möglichen Anwendungsfälle der zukünftigen Applikation auf.

Tabelle 7.1: Übersicht aller Use Case

| Rolle | Use Case Beschreibung | Zustand |
|---|---|---|
| Einloggen/Ausloggen [EA] | | |
| Alle | Einloggen in die Applikation | vorgelegt |
| Alle | Zwang der Kennwortänderung beim ersten Einloggen | vorgelegt |
| Alle | Manuelles Ausloggen durch den Benutzer | vorgelegt |
| Alle | Automatisches Ausloggen nach einer Periode der Inaktivität | vorgelegt |
| Alle | Sperrung von Benutzerkonten nach mehreren unzulässigen Authentifizierungen | vorgelegt |
| Alle | Eigenes Kennwort ändern | vorgelegt |
| Hauptfenster [HF] | | |
| Alle | Zeigt das entsprechende Hauptfenster je nach Rolle des Benutzers an | vorgelegt |
| Verwaltung der telemedizinschen Zentren [VTZ] | | |
| MD_App_Admin | Hinzufügen eines telemedizinischen Zentrums | vorgelegt |
| MD_App_Admin | Bearbeiten eines telemedizinischen Zentrums | vorgelegt |
| MD_App_Admin | Deaktivieren eines telemedizinischen Zentrums | vorgelegt |
| Verwaltung des TMZ Administrator Accounts [VTA] | | |
| MD_App_Admin | Hinzufügen eines telemedizinischen Administrator Accounts | vorgelegt |
| MD_App_Admin | Bearbeiten eines telemedizinischen Administrator Accounts | vorgelegt |
| MD_App_Admin | Deaktivieren eines telemedizinischen Administrator Accounts | vorgelegt |
| Vertretungsdienst eines Telemedizinischen Zentrums [VT] | | |
| MD_App_Admin | Vertretung eines telemedizinischen Zentrums durchführen | vorgelegt |
| MD_App_Admin | Vertretung eines telemedizinischen Zentrums editieren | vorgelegt |
| MD_App_Admin | Vertretung eines telemedizinischen Zentrums löschen | vorgelegt |
| Administrative Aufgaben [AA] | | |
| TMZ_Admin | Hinzufügen eines TMZ-Benutzerkontos | vorgelegt |
| TMZ_Admin | Bearbeiten eines TMZ-Benutzerkontos | vorgelegt |
| TMZ_Admin | Deaktivieren eines TMZ-Benutzerkontos | vorgelegt |
| Mandantenverwaltung [MV] | | |
| TMZ_Admin_Man | Hinzufügen eines Mandantes | vorgelegt |
| TMZ_Admin_Man | Bearbeiten eines Mandantes | vorgelegt |
| TMZ_Admin_Man | Deaktivieren eines Mandantes | vorgelegt |

Continued on next page

**Tabelle 7.1 – continued from previous page**

| Rolle | Use Case Beschreibung | Zustand |
|---|---|---|
| | Benutzerkontoverwaltung [BV] | |
| TMZ_Admin_Man | Hinzufügen eines Benutzerkontos der Mandanten | vorgelegt |
| TMZ_Admin_Man | Bearbeiten eines Benutzerkontos der Mandanten | vorgelegt |
| TMZ_Admin_Man | Hinzufügen eines Benutzerkontos der Mandanten | vorgelegt |
| TMZ_Admin_Man | Bearbeiten eines Benutzerkontos der Mandanten | vorgelegt |
| | Geräte Verwaltung [GV] | |
| TMZ_Tech_Man | Registrierung der tragbaren Geräte | geplant |
| TMZ_Tech_Man | Bearbeitung der tragbaren Geräte | geplant |
| TMZ_Tech_Man | Entziehung der tragbaren Geräte | geplant |
| | Vertrags-Verwaltung [VV] | |
| TMZ_Tech_Man | Registrierung eines Vertrags | geplant |
| TMZ_Tech_Man | Bearbeitung eines Vertrags | geplant |
| TMZ_Tech_Man | Kündigen eines Vertrags | geplant |
| TMZ_Tech_Man | Sperren einer SIM-Karte | geplant |
| | Patientenverwaltung [PV] | |
| TMZ_Admin_Man, M_User | Hinzufügen einer Patientin zu einem Mandant | vorgelegt |
| TMZ_Admin_Man, M_User | Bearbeiten von Patientindaten | vorgelegt |
| TMZ_Admin_Man, M_User | Entlassen einer Patientin | vorgelegt |
| TMZ_Admin_Man, M_User | Löschen einer Patientin | vorgelegt |
| TMZ_Admin_Man, M_User | Bearbeiten der spezifischen Werte einer Schwanger | geplant |
| | CTG Aufzeichnungen [ACTG] | |
| TMZ_Bef, M_User, M_Sec_User | CTG Aufzeichnung befunden | vorgelegt |
| TMZ_Bef, M_User | Kommentare abgeben | vorgelegt |
| TMZ_Admin_Man, M_User | Zuweisung eines unverknüpften CTGs | geplant |
| TMZ_Admin_Man, M_User | Trennung eines verknüpften CTGs | geplant |
| TMZ_Admin_Man, M_User | Verschiebung eines verknüpften CTGs | geplant |
| | Statistische Sicht [SS] | |
| TMZ_Bus_Man, MD_Bus_Man | Übersicht aller PDAs,CTGs,sowie alle Mandanten | geplant |

### 7.1.2 Beschreibung der Anwedungsfälle

Ein Anwendungsfall wird in der UML grafisch durch eine Ellipse dargestellt, die den Namen des Anwendungsfalles trägt. Zu jeder Ellipse existiert ein Text, der den Anwendungsfall genauer beschreibt (BB04). Tabelle 7.2 zeigt eine Beispielstruktur für diese Beschreibung an, die das Anwendungsfall „Hinzufügen einer Patientin“ spezifiert.

Tabelle 7.2: Übersicht aller Use Case

| **PV01 - Hinzufügen einer Patientin** | |
|---|---|
| Akteure | M_User, TMZ_Admin_Man |
| Auslöser | Neue Patientin wird schwanger |
| Vorbedingungen | Patientendaten sind nicht im System registriert |
| Invarianten | - |
| Nachbedingungen | Neue Patientin wurde gespeichert |
| Ereignisfluss | **Basispfad**<br>1. Der Benutzer wechselt zu dem Bereich „**Patient**“<br>2. Alle Patienten sind hier aufgelistet, die zu dieser Organisation zugewiesen sind. Dabei sind folgende Informationen zu sehen:<br>• Aufnahmenummer<br>• Name der Patientin<br>• Mandant<br>• Geburtsdatum<br>• Kontaktdaten<br>3. Wenn eine Patientin neu eingefugt werden soll, muss der Benutzer auf den Button **“Add“** drücken. Anschließend erscheinen folgende Eingabefelder:<br>• Aufnahmenummer<br>• Nachname (*)<br>• Vorname (*)<br>• Geburtsdatum<br>• Adresse (*)<br>• Stadt |

Continued on next page

**Tabelle 7.2 – continued from previous page**

| **PV01 - Hinzufügen einer Patientin** | |
|---|---|
| Ereignisfluss | • Land<br>• Telefonnummer<br>Nur im Fall des administrativen Managers<br>• zuweisen an Mandant (Auswahlliste aller Mandanten, die diesem TMZ zugewiesen sind.)<br>4. Nachdem alle Pflichtfelder ausgefullt sind, die mit Stern (*) gekennzeichnet sind, kann der Benutzer auf **"Submit"** drücken, um die Patientendaten zu speichern.<br>5. Der Benutzer wird im nachsten Schritt zum Fenster **"Gestation"** umgeleitet, hier werden alle Schwangerschaften dieser Patientin angezeigt . Wenn neue Schwangerschaftsdaten fur diese Patientin eingefügt werden, kann zuerst auf **„Add"** geklickt werden, um diese Eingabefelder auszufüllen:<br>• Gravida (*)<br>• Para (*)<br>• voraussichtlicher Entbindungstermin (*)<br>• Analyse Modus (Threshold, FIGO) (*)<br>• Spezielle Konfiguration<br>• Kommentare<br>• zustandiger Arzt<br>• zustandige Hebamme<br>• zugewiesen an PDA (Auswahlliste zeigt alle PDA , die zu dieser Organisation gehoren.) (*)<br>6. Wenn alle diese Pflichtfelder ausgefüllt sind , die mit Stern (*) gekennzeichnet sind , kann der Benutzer auf den **"Submit"** Button drücken, um diese Schwangerschaft im System zu speichern.<br>7. Anschließend wird zum Überblickfenter gewechselt. |

Continued on next page

**Tabelle 7.2 – continued from previous page**

| PV01 - Hinzufügen einer Patientin | |
|---|---|
| Erweiterungen | 5)<br>Falls der Benutzer die Chekbox „Spezielle Konfiguration“ aktiviert, wird er automatisch zum „Config“-Fenster geleitet, in dem er die spezielle Konfiguration für diese Schwangerschaft eingeben kann. Sonst wird die Standard-Konfiguration gespeichert. (siehe PV04)<br>4,7)<br>Wenn der „Submit“-Button gedrückt wurde und einer dieser Pflichtfelder leer oder mit einem unzulässigen Format ausgefüllt ist (die mit Stern (*) gekennzeichnet sind). markiert das System das entsprechende Eingabefeld und zeigt eine passende Fehlermeldung an. |
| Referenzen | - |

Eine Use Case-Spezifikation ist durch den Akteur, dem ein Use Case zugeordnet ist, durch den Name im Diagramm, sowie durch durchgehend auf das gesamte Use Case-Modell bezogene laufende Nummer, eindeutig identifiziert (siehe Tabelle 7.2). Beschrieben wird in erster Linie folgende Felder:

- **Name und Identifier:** Name des Anwendungsfalles und der zugehörige Identifier im Projekt.
- **Akteure:** beteiligte Akteure (Personen) in diesem Anwendungsfall.
- **Auslöser** Der Grund bzw. die Gründe dafür, dass dieser Anwendungsfall ausgeführt wird.
- **Vorbedingungen:** auch Anfangsbedingungen genannt, beschreibt alle Bedingungen, die erfüllt sein müssen, damit dieser Anwendungsfall ausgeführt werden kann. Gibt es keine Vorbedingungen, so steht hier ”keine”.
- **Invarianten:** Die Bedingungen, die im Rahmen des Anwendungsfalles stets erfüllt sein müssen.
- **Nachbedingungen:** oder Abschlussbedingungen :ist das Ergebnis, dass nach einem erfolgreichen Durchlauf desAnwendungsfalls erwartet wird.
- **Ereignisfluss** oder Ablaufbeschreibungen: hier wird der eigentliche reguläre Interaktionsablauf, durch den ein Anwendungsfall umgesetzt wird, indem jeder Einzelschritt über eine elementare, nicht weiter zerlegbare Aktion, dargestellt wird. Die Ablaufschritte (Aktionen) werden nummeriert und meist in strukturiertem Deutsch bzw. Englisch beschrieben. Mittels der UML können diese Ablaufschritte in Aktivitätsdiagrammen oder anwendungsfall-orientierten Sequenzdiagrammen dargestellt werden.

- **Erweiterungen:** Diese ermöglichen die Formulierung von Erweiterungen der Funktionalität eines Anwendungsfalles unter bestimmten Bedingungen. Beim Eintreten der jeweiligen Bedingung wird dann die Funktionalität eines weiteren Anwendungfalles in Anspruch genommen (der natürlich ebenfalls modelliert sein muss).
- **Referenzen:** Aktivitäten oder „Routine"- bzw. Standardabläufe, die in einem oder in mehreren Use Cases mehrfach zur Ausführung kommen sollen, werden aus dem Ereignisfluss der einzelnen Use Cases ausgelagert und namentlich hier darauf referenziert (siehe Anahang **??**).

Zusätzlih können auch folgende Felder definiert werden:

- **Beschreibung:** kurze Beschreibung des Anwendungsfalles.
- **Ausnahmen und Fehlersituationen**: Beschreibung der fachlichen Ausnahme- und Fehlersituationen, die im Rahmen des Anwendungsfalles auftreten können.
- **Variationen:** Abweichungen und Ausnahmen zum Normalablauf und ggf. Beschreibungen der alternativen Abläufe.

Alle vordefinierten Use Cases aus der Anforderungsanalyse (siehe 7.1.1) werden in diesr Phase textuell gemäß der oben beschriebene Tabelle 7.2 spezifiziert. Die komplette und detaillierte Beschreibung dieser Anwendungsfälle finden sie im Anhang **??**

# Kapitel 8

# Designspezifikation

In der Designphase oder auch Entwurfphase wird ein Lösungskonzept für das bestehende Problem unter Berücksichtigung aller Rahmenbedingungen und Anforderungen entwickelt. Das sogenante Objektorientiertes Design in der OOSE [1] baut auf objektorientierter Analyse-Phase auf und definiert die prinzipiellen Möglichkeiten und Einschränkungen für die Problemlösung (Bal05).

Im Laufe der Designphase werden die in der Analysephase identifizierten Objekte in Klassen umgewandelt. Außerdem werden Datenstrukturen und Prozeduren definiert, sodass diese abstrakten Strukturen in einer objektorientierten Sprache abgebildet werden können.

Die Designphase kann in zwei Phasen unterteilt werden. In der ersten Phase, die Systemarchitektur, wird das System in größere komplexe Komponenten aufegteilt. In der zweiten Phase, Spezifikation des Komponentenmodels, werden auf die einzelnen Komponente des Systems eingegangen. Die zweite Phase stellt eigentlich den Übergang zur Implementierung dar.

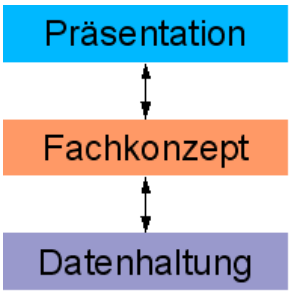
## 8.1 Einleitung

Softwareprojekte werden immer komplexer. Die Umgebung, die Entwicklungszeit, die Produktivität und die Sicherheit werden zu wichtigen Bausteinen in der Entwicklung von umfangreichen Softwareprojekten. Gerade bei umfassenden und komplexen Geschäftsprozessen, welche in der Unternehmenssoftware abgebildet werden müssen, sind gewisse Qualitätsanforderungen in der Entwicklung der Software besonders wichtig. Mehrschichten- und Komponentenentwicklung ermöglichen die dazu nötige Flexibilität.

In einem Schichtenmodell besteht eine Anwendung aus verschiedenen Schichten. Die Anzahl der Schichten ist hierbei abhängig von Art und Komplexität der Anwendung. Heide Balzert schlägt in (Bal05) für kleinere und mittlere Anwendungen drei Schichten vor:

[1] Objektorientierte Software engineering

*„Die GUI-Schicht realisiert die Benutzungsoberfläche einer Anwendung. Dazu gehören die Dialogführung und die Präsentation aller Daten in Fenstern, Berichten usw. Normalerweise löst sie ihre Aufgaben mithilfe eines GUI-Frameworks. Die Fachkonzeptschicht modelliert den funktionalen Kern der Anwendung. Außerdem enthält sie die Zugriffe auf die Datenhaltungsschicht, in der die jeweilige Form der Datenspeicherung realisiert wird, z.B. mit einem relationalen Datenbanksystem.“*




Diese Architektur besteht aus drei Standard-Softwareschichten: der Präsentations-, der Fachkonzept- und der Datenhaltungsschicht. Diese drei Schichten spiegeln die grundsätzlichen Aufgaben von Softwaresystemen wider (siehe (Bal05)) :

- Benutzeroberflächen, um Daten darzustellen und auf Benutzereingaben zu reagieren
- Fachliche Objekte und fachliche Logik, um Geschäftsprozesse abzubilden
- Dienste, die für eine dauerhafte Verwaltung der fachlichen Daten sorgen

In der Praxis hat sich der Einsatz dieses Modells als sehr hilfreich erwiesen. Durch die saubere Trennung der einzelnen Schichten blieb das Programm auch bei steigendem Umfang sehr übersichtlich **??**. Aus diesem Grund wurde in dieser Projekt auch nach diesem Modell orientieren. *Trium* für die erste und zweite Schicht zwei wichtiges Tools zur verfügung stellt. Ein GUI-Framework namens *Formengine*, die verschiedene Klassen und Schnittstellen für die Entwicklung der graphischen Benutzeroberflächen bereitstellt, sowie ein Applikation-Manager namens *JWeppAppCore*, der das Grundgerüst für die Fachkonzeptschicht anbietet, um die Geschäftsprozesse der Applikation umzusetzen. Beide Werkzeuge sind objektorientiert modelliert, und mit Java als objektorientierte Progarmmiersparche implementiert. Es stellt sich die Aufgabe eine geeignete Art der Datenhaltung sowie die Persistenz der Daten auszuwählen.

## 8.2 Persistenz und Datenhaltung

Die Anwendung hat unterschiedliche Informationendaten (Objekte) zu speichern. Typische Objekte sind z.B. Benutzer (Bentutzername, Kennwort, Rolle, Rechte, etc), Patienten (Initialendaten, Aufzeichnungen, etc) , tragbare Geräte (Name, Hersteller, ) usw. Diese Objekte , die vom System erzeugt und aufgerufen werden, sie müssen persistent sein, um eine dauerhafte funktionalität der Applikation zu gewährleisten.

In der Regel sind Objekte in einer Programmiersprache transient (vergänglich), d.h. bei Programmende gehen sie verloren. Allerdings benötigt die Anwendung einen dauerhaften Zugriff auf diese Objekte, auch über die Applikationsende hinaus. Dies bedingt eine Speicherung der Objekte auf nichtflüchtigen Speichermedien, z.B. in Dateien oder Datenbanken. Dieses Vorgehen nennt man Persistenz. Hier muss bestimmt werden, welcher

Persistenzmechanismus für die Anwendung geeignet ist, um alle nötigen Informationen zu sichern.

### 8.2.1 Auswahl des Persistenzmediums

Um diese Daten persistent halten zu können, gibt es dafür verschiedene Vorgehensweisen. Generell gibt es gegenwärtig drei Möglichkeiten für die Speicherverwaltung (Per) :

- Datein : Dateien stellen die Speicherabstraktion dar, die von Betriebssystemen zur Verfügung gestellt werden. Eine Anwendung speichert ihre Daten als eine Folge von Byts und definiert, wie und wann Daten gefunden werden sollen. Das Konzept einer Datei ist ziemlich maschinenorientiert und ermöglicht deshalb den Anwendungen, eine Reihe von Größen- und Geschwindigkeitsoptimierungen vorzunehmen. Dateien verlangen je doch auch, dass die Anwendung viele Probleme zum Beipiel den gleichzeitigen Zugriff und den Verlust von Daten im Falle eines Systemabsturzes, regellt.

- Relationale Datenbank : Eine relationale Datenbank bietet eine Datenabstraktion auf einem höheren Niveau als Dateien. Daten werden hier in Tabellen gespeichert, die einem vordefinierten Typ entsprechen, einem so genannten Datenbankschema. Jede Spalte der Tabelle steht für ein Attribut und eine Reihe repräsentiert ein Datenelement als Tupel von Attributen.

- Objektorientierte Datenbank : Eine objektorientierte Datenbank bietet grundsätzlich dieselben Dienste an wie eine rationale Datenbank . Anders als die letzte erlaubt sie die direkte Speicherung von Klassen und Assoziationen. Durch die Bereitstellung von Klassen auf höheren Abstraktionsebene verringert sich die Notwendigkeit, zwischen Objekten und Speichereinheiten zu übersetzen. Außerdem bieten sie Vererbung und abstrakte Datentypen an. Typische Anfragen sind jedoch wesentlich langsamer als bei relationalen Datenbanken und schwieriger zu optimieren.

### 8.2.2 Auswahlkriterien für das Persistenzmedium

Brügge fasst in seinem Buch (BB04) die Entscheidungsfaktoren zusammen, die bei der Auswahl eines Speicherverwaltungssystems berücksichtigt werden sollen.

***Abwägung zwischen Dateien und relationalen und objektorientierten Datenbanken***

*Wann sollte flache Dateien verwendet werden?*
*+ bei umfangreichen Daten (z.B: Bildern)*
*+ bei temporären Daten (z.B. Kerndateien)*
*+ bei geringer Informationsdichte (z.B. Archivdateien, Aufzeichnungen von Verläufen)*
*Wann sollte relationale oder objektorientierte Datenbanken verwendet werden?*
*+ bei gleichzeitigen Zugriffen*
*+ beim Zugriff auf unterschiedliche Detailstufen*

*+ bei heterogenen Plattformen und Anwendungen*
*Wann sollten relationale Datenbanken verwendet werden?*
*+ bei komplexen Anfragen über Eigenschaften*
*+ bei großen Datensätze*
*Wann sollten objektorientierte Datenbanken verwendet werden?*
*+ beim häufigen Gebrauch von Assoziationen beim Auffinden von Daten*
*+ bei mittelgroßen Datensätzen*
*+ bei irregulären Assoziationen zwischen Objekten*

Dazu kommen auch nichtfunktionale Anforderungen wie etwa: sollen die Objekte schnell wiedergefunden werden können? Muss das System komplexe Anfragen starten, um diese Daten wiederzufinden? Benötigen die Objekte viel Platz im Hauptspeicher?

Neben den großen Datennsätze, die unsere Web-Applikation aufnehmen soll, kann eine lange Verzögerungszeit beim Ausführen von Datenbank-Transaktionen nicht akzeptiert werden, deshalb wurde für eine hybride Lösung aus relationalen Datenbank und flache Dateien entschieden. Diese Entscheidung wurde gemäß der oben genanneten Auswahlkriterien 8.2.2 getroffen. Diese Lösung wird sowohl eine relationale Datenbank benutzen, um alle Dateninformationen bezüglich der Patienten , Benutzer, Geräte, etc persistent speichern zu können, als auch flache Dateien für die Speicherung und Archivierung der empfangenen Aufzeichnungen verwenden. Wobei hier erwähnt werden nmuss, dass der referenzierte Pfad zu diesen Dateien in der Datenbank gespeichert wird, um diese wieder zuordnen zu können.

### 8.2.3 Auswahl der Datenbank

Datenbanken werden von sogenannten Datenbank Mangementsystemen (DBMS) verwaltet. Die zentralen Aufgaben eines DBMS [2] sind im Wesentlichen die Bereitstellung verschiedener Sichten auf die Daten (Views), die Konsistenzprüfung der Daten (Integritätssicherung), die Autorisationsprüfung, die Behandlung gleichzeitiger Zugriffe verschiedener Benutzer (Synchronisation) und das Bereitstellen einer Datensicherungsmöglichkeit, um im Falle eines Systemausfalls zeitnah Daten wiederherstellen zu können (KP05).

Als DBMS wurde das Objektrelational DBMS PostgreSQL [3] ausgewählt. Dieses DBMS wird standardmäßig bei *Trium* eingesetzt. Dabei spielen mehrere Gründe eine wichtige Rolle. Neben die Plattformunabhängigkeit zeichnet sich dieses DBMS durch eine umfassende Unterstützung des SQL-Standards sowie generell durch eine sehr breite Unterstützung gängiger Unternehmensstandards aus. Insbesondere die Techniken Views, Stored Procedures und Trigger sind Bestandteil von PostgreSQL. Weitere technische Fähigkeiten von PostgreSQL sind (PP):

- Referenzielle Integrität

[2] Datenbank Management System
[3] www.postgresql.org

- native Schnittstellen für ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python und Ruby
- Sequences, Verebung
- Outer Joins, Subselects
- Benutzerdefinierte Datentypen
- Unterstützung des ACID-Prinzips [4] zur Transaktionsverarbeitung.

Während PostgreSQL auf fast allen Unix-Varianten verfügbar ist, läuft PostgreSQL ab Version 8.0 auch auf Microsoft NT-basierten Betriebssystemen wie Windows 2000, XP und Server2003. PostgreSQL ist Open-Source und hält sich bei der Implementierung an die gängigen Standards aus der Datenbankwelt. Es steht unter der Lesser GNU Public License LGPL (GNU). Ein wichtiges Auswahlkriterium war die Unterstützung durch das verwendete Object-Relational-Mapping (ORM) Tool, und eine sehr hohe Praxistauglichkeit des Produktes.

Wie bereits erwähnt wurde. Die oberen Schichten (erste und zweite) objektorientiert modelliert und auch impmentiert wurden. Allerdings haben wir uns für eine relationale Datenbank als Persistenzmedium entschieden. Demzufolge liegt das Problem *Mismatch Impedance* der Kopplung von SQL [5] mit der objektorientierten Programmiersprache Java wegen der unterschiedlichen Datenstrukturkonzepte vor: das Konzept der Relation als Menge von Tupeln (eine Zeile in der Tabelle ) bei SQL gegenuüber dem Tupel bzw. dem Objekt als grundlegende Datenstruktur einer objektorientierten Programmiersprache.

Um den Bruch zwischen der objektorientierten Anwendungsentwicklung und den relationalen Datenbanken, die den Datenbank-Standard darstellen, zu überbrücken, wurden in den letzten Jahren viele sogenannte ORM (Object Relational Mapping) oder Persistenz-Framework entwickelt. Diese ermöglichen, den Zustand eines Objekts in einer relationalen Datenbank zu speichern und aus entsprechenden Datensätzen wiederum Objekte zu erzeugen. Hibernate [6] ist ein solcher OR-Mapper für Java, der sehr weit verbreitet ist und mit guten Kritiken überhäuft wird. Ein weiterer Vorteil ist die freie Verfügbarkeit.

### 8.2.4 Auswahl der Persistenz-Framework

Als OR-Mapping Tool wurde hier Hibernate verwendet, denn es ist einer der bekanntesten OR [7] Persistence Services für Java. Ein wichtiges Kriterium für diese Wahl war die Performance, da Hibernate eine Mischung aus Bytecode-Generieung und Core-Reflection benutzt, die zu einem vernachlässigbar kleinen Overhead führt. Bei direkten Datenbank-Kommunikation spiel Hibernate seine Stärken aus. Hier wirken sich primär drei Hibernate-Konzepte positiv aus (RFB06):

[4] Atomicity, Consistency, Isolation, Durability
[5] Structured Query Language
[6] www.hibernate.org
[7] Object Relational

- Spezifische Datenbank-Adapter
- Caching pro Session(First Level Cache)
- Lazy-loading (Load on Demand)

Hibernate ist ein Framework, das eine objektorientierte Sicht auf Tabellen und Beziehungen in relationalen Datenbank-Management-Systemen (RDBMS) zur Verfügung stellt. Statt mit SQL-Statements wird mit Objekten operiert. Dieser Vorgang wird auch als Object-Relational-Mapping (ORM) bezeichnet. Für mehr Details siehe Kapitel 9.2.

*"object-relational mapping is the automated (and transparent)persistence of objects in a Java application to the tables in a relational database, using metadata that describes the mapping between the objects and the database"* (BC05)

Das Hibernate-Projekt wurde 2001 von Gavin King gegründet und bis zum Jahre 2003 als nicht-kommerzielles Open-Source-Projekt geführt (siehe[BK05]). Seit diesem Zeitpunkt ist das Hibernate-Team, unter kommerziellem Aspekt, Teil der JBoss-Corporation [8]. Hibernate ist jedoch weiterhin kostenlos erhältlich und kann auch jederzeit kostenlos in eigenen (auch komerziellen) Projekten verwendet werden. Hibernate steht unter der Lesser GNU Public License LGPL (GNU). Hibernate wurde ursprünglich ausschließlich für die Java-Plattform entwickelt, mittlerweile existiert jedoch auch eine Portierung für .NET unter dem Namen NHibernate. NHibernate befindet sich allerdings noch in einem frühen Entwicklungsstadium und ist derzeit in der Version 1.2 erhältlich, Hibernate bereits in Version 3.2.4 (Pag).

## 8.3 Systemarchitektur

Dieser Abschnitt beschreibt aus welchen Komponenten dieses Software-System aufgebaut ist und wie sich deren hierarchische Struktur darstellt. Die zukünftige Plattform fällt hier grundsätzlich in die Kategorie einer Client-Server-Architektur. Viele Client-Rechner greifen auf einen Server mit der Applikation zu. Nachdem einen Überblick über die Schichten sowie deren Schnittstellen erklärt wurde, aus denen das System besteht, kann die Architektur unserer Plattform schematich anhand der Abbildung 8.1 dargestellt werden. Die Lösung geht von einer klassichen Drei-Schichten-Architektur für die zu entwickelnden Client/Server Anwendung aus. In der GUI-Schicht (Präsentation) sind zusätzliche Unterteilungen vorhanden, da hier das MVC-Designpattern verwendet wird, das unter dem Kapitel 9.3 näher beschrieben wird. In der Fachkonzeptschicht kommt sowohl zudem das Singleton-Design-Pattern zum Einsatz 9.2.2.1.

**Präsentationsschicht**

Die Präsentationsschicht stellt grafisch die Daten, die von der Fachkonzeptschicht bereitgestellt werden, auf einer Benutzeroberfläche (graphical user interface GUI) dar und gibt dem Benutzer die Möglichkeiten mitells Aktionen (Klick, Eintrag, etc) mit der Anwendung

[8] www.jboss.org

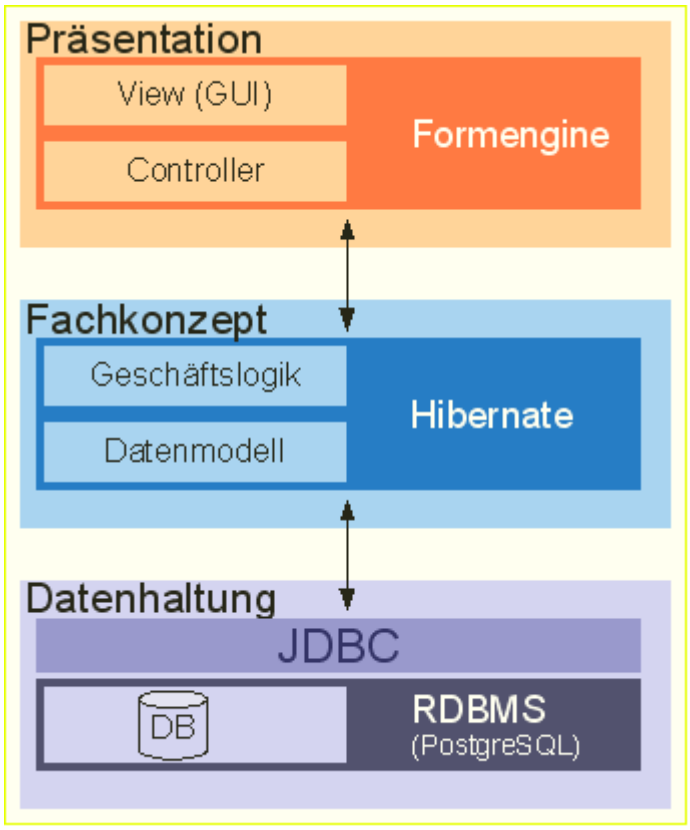



Abbildung 8.1: Schichtenmodell der Anwendung

interzureagieren und Ereignisse auszulösen, welche vom Controller abgefangen und an die Fachkonzeptschicht weitergegeben werden um dort einen Geschäftsprozess auszulösen. Hier wird das GUI-Framework von *Trium* zum Einsatz kommen, das sämtliche Klassen und Schnittstellen zur Implementierung von verschiedenen Benutzeroberfläche zur Verfügung stellt. Im Kapitel 9.3 wird das genauer erläutert.

**Fachkonzeptschicht**

In der Fachkonzeptschicht, auch Geschäftslogikschicht, Anwendungsschicht oder Applikationsschicht genannt, werden sämtliche fachliche Funktionalitäten der Anwendung realisiert. Sie wird von der GUI-Schicht über Zustandsänderungen informiert und führt anhand dieser Änderungen Aktionen aus. Sie nützt die Dienste der bestehenden Komponente und initiiert Zustandsänderungen der GUI.

Die Schicht des Fachkonzeptes wird weiter in das Datenmodell und die eigentliche Geschäftslogik unterteilt. Aufgabe des Datenmodelles ist es die Kapselung der darunter liegenden Datenhaltungsschicht, sie stellt dabei ein logisches Modell der Anwendung dar, welches unabhängig von der Präsentation existiert. Sie enthält vor allem die Fachklassen der Anwendung (persistente Klassen/objektorientierte Datenmodell). Die Objekten des Datenmodelles werden in einer relationalen Datenbank abgespeichert. Also sie stellt eine Abstraktionsschicht zwischen den Objekten der Applikation und der physikalischen

Darstellung der Daten in der Datenbank mithilfe einem Objekt-Relationaler-Mapper her.

Die Geschäftslogik ist Kern der Anwendung und umfasst die Funktionalität zur Realisierung der Anwendungsfälle. Sie sorgt für die Haltung des Objektmodelles durch die verschiedenen persistenten Transaktionen. Einige Methoden der Geschäftslogik bestehen aus einfachen Aufrufen des Objektes vom Datenmodell (Objekt des Datenmodelles), andere enthalten zusätzliche Geschäftsregeln und führen mehrere Operationen auf die Objekte des Datenmodelles aus. Dabei wird mithilfe von rollenbasierter Sicherheit geprüft, ob der aktuell angemeldete Benutzer überhaupt berechtigt ist, bestimmte Aktionen mit der Geschäftslogik auszuführen.

Für diesen Zweck wurde sowohl *Hibernate* als *JWebbAppCore* in dieser Schicht verwendet. Das Hibernate-Framework kapselt die üblichen Daten-Operationen und verwaltet die Datenbank-Transaktionen. Die Persistenzschicht kennt deshalb das Datenbankschema, in dem die fachlichen Daten verwendet werden.

Das selbst entwickelte JWebbCore-Framework von *Trium* stellt abstrakte Klassen zur Sicherung des Zugriffsrechts von Benutzer, sowie Schnittstellen zur Durchführung von unterschiedliche Operationen auf die physikaliche Datenbankschema.

**Datenhaltungsschicht**

Die Datenhaltungsschicht (Persistenzschicht) sorgt dafür, dass die fachlichen Daten dauerhaft in einem Datenbanksystem gespeichert und auch wieder geladen werden können. sie ist also die für Persistenz der Daten und für die Sicherung der Datenkonsistenz zuständig. Da Hibernate das standarisierte JDBC-Schnittstelle zum Zugriff auf die Datenbank verwendet, wurde die JDBC-Schnittstelle zum bessern Verständnis in der Datenhaltungschicht aufgenommen.

Der Hauptvorteil dieser Drei-Schichten-Architektur ist durch die klaren Abgrenzungen der einzelnen Ebenen gegeben (HT01):

- Verständlichkeit: Höhere Ebenen (Systemkomponenten) werden einfacher, weil sie tiefere Ebenen benutzen können.
- verbesserte Projektorganisation bei umfangreichen Projekten: Änderungen auf höheren Ebenen bleiben ohne Einfluß auf tiefere Ebenen.
- Wiederverwendbarkeit,Redundanzfreiheit : Höhere Ebenen lassen sich abtrennen, tiefere Ebenen bleiben trotzdem funktionsfähig (z.B.: der grundlegenden GUI).
- Tiefere Ebenen können getestet werden, bevor die höheren Ebenen lauffähig sind.

Diese Architektur hat allgemein viele Vorteile und ein paar Nachteile. Da die Anwendungslogik auf dem Server liegt, kann die Anwendung zentral verwaltet und gesichert werden weiterhin ist die Anwendung sehr sicher vor ungewollten Änderungen und vor Kopien geschützt. Ein Nachteil ist, dass der Server entsprechend leistungsstark sein muss, um alle Clients zu bedienen.

In diesem Projekt wurde nach dem Bottom-up Vorgehen entwickelt, also mit dem Datenmodell begonnen, dann die Fachklassen und zuletzt die GUI entwickelt. Der Vorteil

bestand darin, dass es einfacher war, die bereits vorhandene Funktionalität zu testen. Wir könnten uns bei der Entwicklung der nächst höheren Schicht weitgehend darauf verlassen, dass die tiefere Schicht funktioniert.

## 8.4 Datenmodell

Wie im vorigen Abschnitt erwähnt wurde, war die Persistenzschicht die erste Schicht, die entwickelt wurde. In diesem Abschnitt wird gezeigt, wie man ein Objektorientiertes Datenmodell auf ein Relationales Datenmodell (Datenbank) abbilden kann.

### 8.4.1 Objektorientiertes Datenmodell

In der folgenden Abbildung 8.2 ist das Anwendungs-Objektmodell zu sehen, in dem alle persistente Klassen sowie deren Assoziationen dargestellt sind. Sie bilden das Geschäftsobjekt der Anwedung, das auf Datenbankschema abgebildet werden.
Bei dem Entwurf von diesem Klassendiagramm wurde die Standard-Notation vom *Trium* berücksichtigt. Die zeigt, dass der Name der Klassen im objetorientierten Datenmodell immer mit dem Prefix DO_ anfangen muss. Bei diesem Klassendiagramm wurde nur wenige Attributen (wesentliche) modelliert, um die Verständlichkeit und die klare Sicht des Datenmodelles zu behalten. Das komplette Datenmodell mit allen Attribute finden Sie in der Abbildung 8.4. Die meisten von der angezeigten Klassen dieses Klassendiagrammes sind aus dem Use-Case-diagramm von der Anforderungsanalyse 6.2.1 abgeleitet worden, wobei alle nummerierten Anwendungsfälle in den Anwendungsfall-Diagramm 6.2.1 der Anforderungsanalyse sind als Klassen verwendet werden und die restlichen Anwendungsfälle als Klassen in der Geschäftslogik umgewandelt werden.

In Abbildung 8.2 kann man leicht erkennen, dass das Objekmodell aus zwei verschiedenen Klassen entsteht. Die grünen Klassen, die innerhlab der JWebAppCore-Framework definiert sind, sowie die restlichen Klassen, die für dieses Projekt extra entwickelt wurde, sind mit der Farbe Lila markiert. Da im Klassendiagramm und auch bei den Datenbankeinträgen englische Bezeichnungen verwendet wurde. Finden Sie im Anhang in der Tabelle 8.4, die deutsche Übersetzungen aufklärt, um Unklarheiten zu vermeiden.

### 8.4.2 Abbildung von Objektmodell auf Datenbankschema

Um Objekte einer objektorientierten Programmiersprache persistent in einem relationalen Datenbanksystem zu speichern, müssen einige Objekttransformationen durchgeführt werden. Objekte des objektorientierten Programmierparadigmas müssen in Strukturen des relationalen Datenmodells transformiert werden (BB04).

Die wichtigste Aufgabe beim Abbilden von Objekten ist es, Objekte der objektorientierten Programmiersprache auf Tabellen (Relationen) und Tupeln (Zeilen) der relationalen Datenbank abzubilden.Wie in [Ambler 99] dargelegt wird, müssen dabei folgende Schritte berücksichtigt werden:

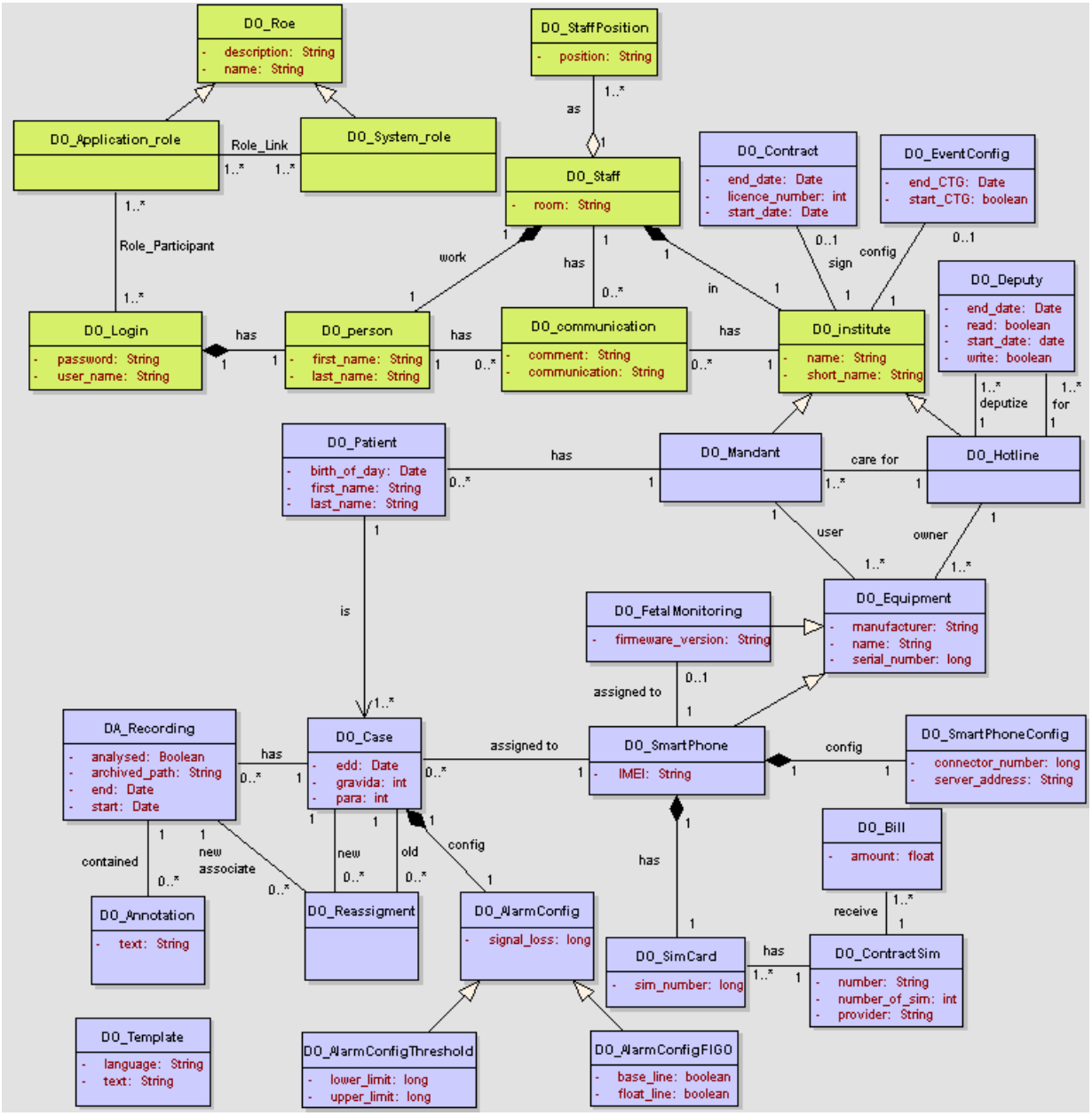



Abbildung 8.2: Klassendiagramm für das Anwendungs-Objektmodell

- Abbildung von Klassen und Attributen
- Abblidung von Assoziationen
- Abbildung von Vererbungsbeziehungen

#### 8.4.2.1 Abbildung von Klassen und Attributen

Beim Abbilden von persistenten Objekten auf relationale Datenbankschema konzentrieren wir uns zuerst auf Klassen und deren Attribute. Für jede persistente Klasse definieren wir eine Tabelle, die den gleichen Namen wie die Klasse erhält. Für jedes Attribut fügen wir der Tabelle eine Spalte mit dem Namen des Attributs hinzu. Als nächstes werden wir uns der Wahl des Primärschlüssles zu, wobei es zwei Möglichkeiten gibt: die erste besteht darin, eine Anzahl von Klassenattributen auszusuchen, die ein Objekt eindeutig identifizieren. Die zweite Möglichkeit ist die Erstellung eines neuen, eindeutigen Attributs zur Identifikation der Objekte.

#### 8.4.2.2 Abbildung von Assoziationen

Nachdem wir jetzt die Klassen auf relationale Tabellen abgebildet haben, beschäftigen wir uns mit der Assoziationen in Datenbankschema, dabei gibt es zwei Arten von Imlementierungen, die von der Multiplizität der Assoziationen abhängen: Eins-zu-Eins (1:1) und Eins-zu-Viele (1:n) Assoziationen werden durch die Verwendung von eingebetteten Fremdschlüssel als sogenannte verdeckte Assoziationen implementiert. Viele-zu-Viele (n:m) Assoziationen werden in einer Separaten Tabelle implementiert (KA04).
**Verdeckte Assoziationen** Assoziationen , die an einem Ende der Multiplizität 1 haben, können durch die Verwendung eines Fremdschlüssels abbgebildet werden. Bei Eins-zu-Viele Assoziationen fügen wir den Fremdschlüssel in die Tabelle der Klassen auf der „Viele“-Seite der Assoziationen ein.
**Separate Tabelle** Eine Viele-zu-Viele Assoziation wird durch eine separate Tabelle mit zwei Spalten, die beide Fremdschlüsssel auf die Klassen der Assoziationen darstellen, abgebildet. Diese Tabelle nennt man auch Assoziationstabelle(siehe role_participant in 8.3).

Es ist zu beachten, dass auch Eins-zu-Eins und Eins-zu-Viele Assoziationen durch Assoziationstabellen - anstelle von verdeckten Assoziationen - realisiert werden können. Die Verwendung von separaten Assoziationstabellen resultiert in leichter zu verändernden Datenbankschema. Ändern wir die Multiplizität einer Assoziation von Eins-zu-Viele in Viele-zu-Viele, so müssen wir das Datenbankschema nicht verändern

#### 8.4.2.3 Abbildung von Vererbungsbeziehungen

Vererbungen werden nicht vom relationalen Datenbankmodell direkt unterstützt. Um Klassen des Objektmodells, die in einer Vererbungsbeziehung stehen, auf das relationale Modell abzubilden, werden in (BB04) und [Keller 97] drei Konzepte vorgeschlagen und erläutert. Zur Erläuterung dieser Konzepte wurde das Beispiel vom Equipment(Oberklasse),SmartPhone und fetal Monitor(Unterklasse) vom Klassendiagramm (siehe 8.2) ausgewählt:

- Vertikale Partitionierung
- Horizontale Partitionierung
- Typisierte/gefilterte Partitionierung

**Vertikale Partitionierung**: Bei der vertikalen Partitionierung wird jede Klasse des Objektmodells auf eine eigene Tabelle abgebildet. Jede Klasse enthält nur ihre eigenen Attribute und ein zusätzliches Attribut, das eine Beziehung zu zugehörigen Zeilen in den Superklassen herstellt (Abbildung 8.1). Bei dieser Variante werden die Daten eines Objektes auf mehrere Tabellen verteilt. Objekte müssen durch komplizierte, multirelationale Suchanfragen unter Verwendung von JOIN-Operationen und mit Hilfe des zusätzlich eingefügten Attributs rekonstruiert werden.

| Equipment | | |
|---|---|---|
| equipment_id | name | manufacturer |
| 1 | fetal 1 | Philips |
| 2 | PDA 1 | Nokia |

| Fetal Monitor | |
|---|---|
| equipment_id | firmeware |
| 1 | V 3.04 |

| Smart Phone | |
|---|---|
| equipment_id | IMEI |
| 2 | 1234 |

Tabelle 8.1: Abbildung der Vererbung über vertikale Partitionierung

**Horizontale Partitionierung**: Die horizontale Partitionierung wird so beschrieben, daß nur die spezialisiertesten Klassen (Unterklasse) samt ihrer geerbten Attribute auf eigene Tabellen abgebildet werden.Gesuchte Objekte können mit einer einfachen monorelationalen Suchanfrage ohne aufwendige JOIN-Operationen ermittelt werden. Nachteilig ist die geringe Flexibilität des Datenbankschemas bei Änderungen (Abbildung 8.2).

| Fetal Monitor | | | |
|---|---|---|---|
| monitor_id | name | manufacturer | firmeware |
| 1 | fetal 1 | Philips | V 3.04 |

| Smart Phone | | | |
|---|---|---|---|
| phone_id | name | manufacturer | IMEI |
| 2 | PDA 1 | Nokia | 1234 |

Tabelle 8.2: Abbildung der Vererbung über horizontale Partitionierung

**Typisierte/gefilterte Partitionierung**: Ein weiteres Abbildungskonzept ist die typisierte/gefilterte Partitionierung. In diesem Fall werden alle in der Vererbungshierarchie vorhandenen Klassen auf eine einzige Tabelle abgebildet. In dieser Tabelle muß ein zusätzliches Attribut (eine zusätzliche Spalte) definiert werden (Type), welches angibt, zu welcher Subklasse das jeweilige Objekt gehört (Abbildung 8.3). Der Vorteil dieses Verfahrens liegt in der hohen Geschwindigkeit, mit der Objekte gefunden werden können, da Anfragen nur an eine Tabelle gestellt werden. Aus diesem Grund wurde dieses Konzept in deiesem Projekt eingesetzt. Alle Vererbungsbeziehungen des Klassendiagrammes (Abbildung 8.2) wurden nach diesem Konzept in dem relationalen datenbankschema (Abbildung 8.4) aufgelöst.

| Equipment | | | | | |
|---|---|---|---|---|---|
| equipment_id | name | manufacturer | type | IMEI | firmeware |
| 1 | fetal 1 | Philips | monitor | | V 3.04 |
| 2 | PDA 1 | Nokia | phone | 1234 | |

Tabelle 8.3: Abbildung der Vererbung mittels getypter Partitionierung

### 8.4.3 Relationales Datenmodell

Abbildung 8.3 ist das Entity-Relationship-Diagramm und in Abbildung 8.4 die Tabellenstruktur der Datenbank dargestellt. Bei dem ER-Modell gibt es mehrere Arten diese darzustellen. Diese Arten werden Notation genannt. Meist sind diese Notationen in graphischer Form, also Diagramme, und werden auch Entity-Relationship-Diagram (ERD) oder ER-Diagramme genannt. Die grafische Darstellung von Entitätstypen und Beziehungstypen dieses ER-Diagrammes ist nach der Martin-Notation (Krähenfuß-Notation)[9] modelliert wurde. Sie stellt ein weit verbreitetes Werkzeug zur Diagramm-Sprache dar.
Anhand der Abbildung 8.3 kann man feststellen, dass das ER-Diagramm aus zwei Sorten von Relationen (Tabellen) besteht. Nämlich die grünen Tabellen, die aus dem *JWebApp-Core* von *Trium* importiert sind, sowie die restlichen Tabellen, die für dieses Projekt extra entwickelt wurden. Diese sind mit der Farbe Lila markiert.
Alle Tabellen sind nach der Standard-Bezeichnung von *Trium* entworfen. Dabei müssen alle Tabellen mit der Bezeichnung tab_ gefolgt mit den Name der Tabelle notiert werden, die eine entsprechende Log-Tabelle besitzen sollen. Wir werden in dem Kapitel 9.1 noch mal auf dieser Punkte zurück kommen. Bevor wir mit den Beziehungserklärungen anfangen, muss darauf hingewiesen werden, dass die Tabelle „Institut“ die Einträge für MD, TMZ, und Mandant umfasst (Typisierte-Partitionierung Konzept für die Abbildung der objektorientierte Vererbungsbeziehungen auf relationale Datenbankschema, siehe Kapitel 8.4.2.3). Bezüglich der Relationen und Kardinalitäten gab es folgende Vorüberlegungen:

- MD kann mehrere TMZ verwalten
- Jedes TMZ kann beliebig viele Mandanten administrieren
- Jedes TMZ kann von mehreren anderen TMZs vertreten werden
- Jedes TMZ besitzt viele Geräte (PDA, fetal Monitor), die von einem Mandant gemietet oder gekauft werden können
- Jedes fetal Monitor kann zu einem einzigen PDA zugeordnet werden
- Jedes PDA hat genau einen Konfigurationseintrag
- Jeder Mandant wird genau von einem TMZ betreuet

[9]Für mehr Informationen: http://de.wikipedia.org/wiki/Er-diagramm

- Jeder Manadnt hat genau eine Konfigurationsdatei
- Jeder Mandant hat eine große Anzahl von Patientinen
- Jede Patientin kann mehrmals schwanger werden
- Jede Schwangerschaft wird genau zu einem PDA zugeordnet, und wird genau auch von einem Arzt, sowie von einer Hebamme beaufsichtigt
- Jede Schwangerschaft hat genau eine Schwellenwerteinstellung
- Jede Schwangerschaft hat mehrere Aufzeichnungen, die wiederum mehrere Kommentare enthalten können
- Jede Aufzeichnung kann von einer Schwangerschaft zu einer neuen Schwangerschaft neu verknüpft werden

Die Abbildung 8.4 dient der Darstellung der Tabellen und ihrer Beziehungen zueinander inklusive aller Attribute. Wobei die im fett vorgehobenen Felder als Pflichtsfelder der jeweiligen Tabelle zu betrachten sind. Die rot markierten Attributen repräsentieren die Primärschlüsselen der jeweiligen Tabelle. Die Felder, die mit der blauen Farbe eingefärbt sind, legen den Fremdschlüssel (referenzielle Integrität) zur anderen Entitäten dar. Sie stellen somit die Beziehnungen (Assoziationen) zur anderen Tabellen.

Aufgrund der hohen Performance der Typisierte Partitionierung 8.4.2.3 wurde dieses Konzept hier eingesetzt. So sind jetzt alle Einträge zur tragbaren Geräte (fetal Monitor, PDA, Bluetooth) in der Entität „tab_Equipment“, wobei das Attribut „equipment_type“ als Unterscheidungsmerkmal für die verschiedene Geräte. Das gleiche gilt auch für die Entität „tab_institute“, in der die Einträge aller Organisationen (MD, TMZ, Manadant) gespeichert sind, aber durch den Diskriminator „institute_type“ können die Instituten unterscheiden werden.

Zur Erstellung des Klassendiagrammes wurde das Tool „Enterprise Architect“ von Sparx Systems (EA) verwendet. Dagegen wurde die lizenzfreie Software „Toad Data Modeler“ von Case Studio (CS) zur Modellierung der relationalen Datenmodell. Die unzählige Datenbanken wie z. B. (Oracle, SQL Server, DB2, Sybase, MySQL, MaxDB, PostgreSQL und viele mehr) sowie automatische Generierung von SQL-Code unterstützt.

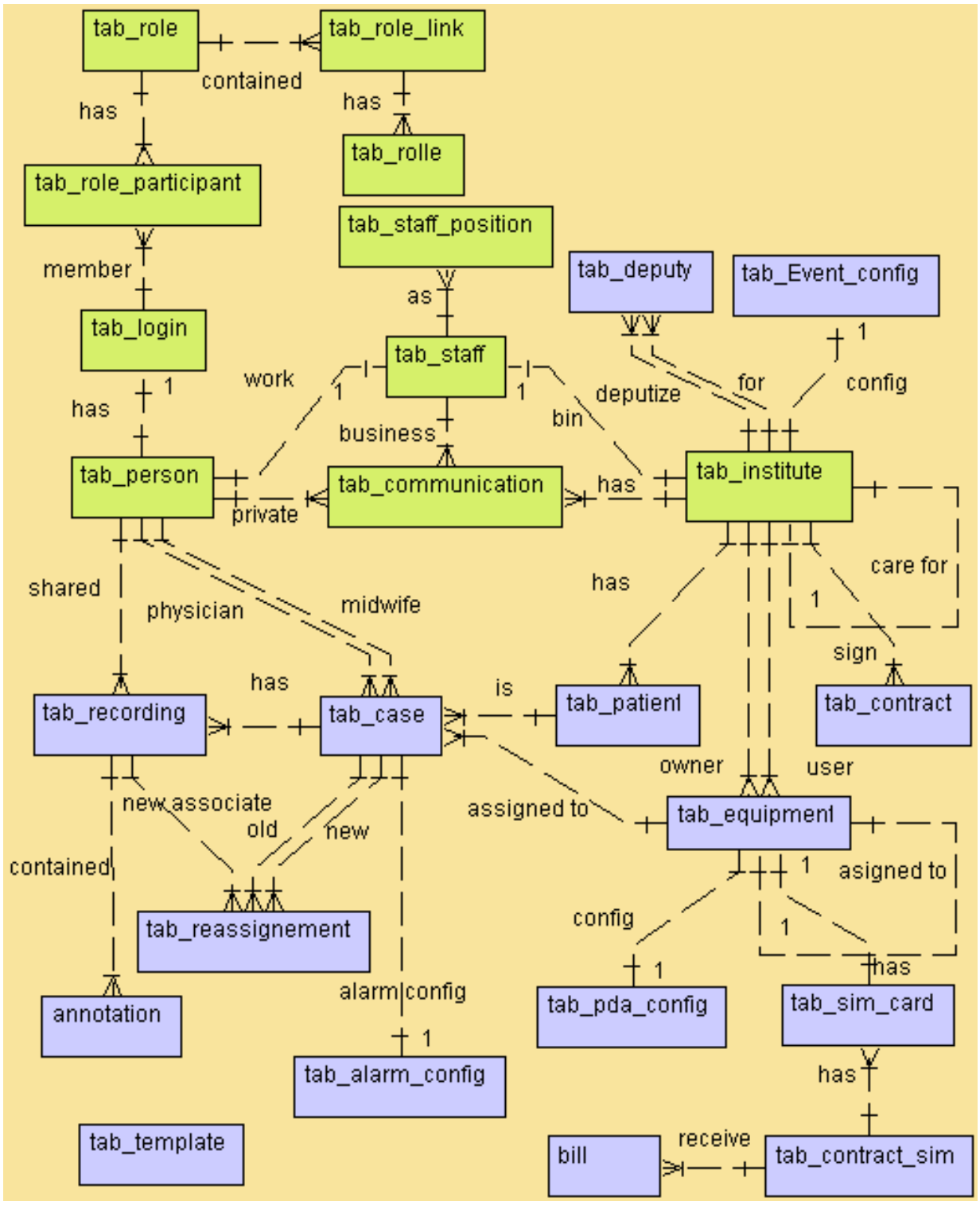



Abbildung 8.3: vereinfachtes Entity Relation Diagramm

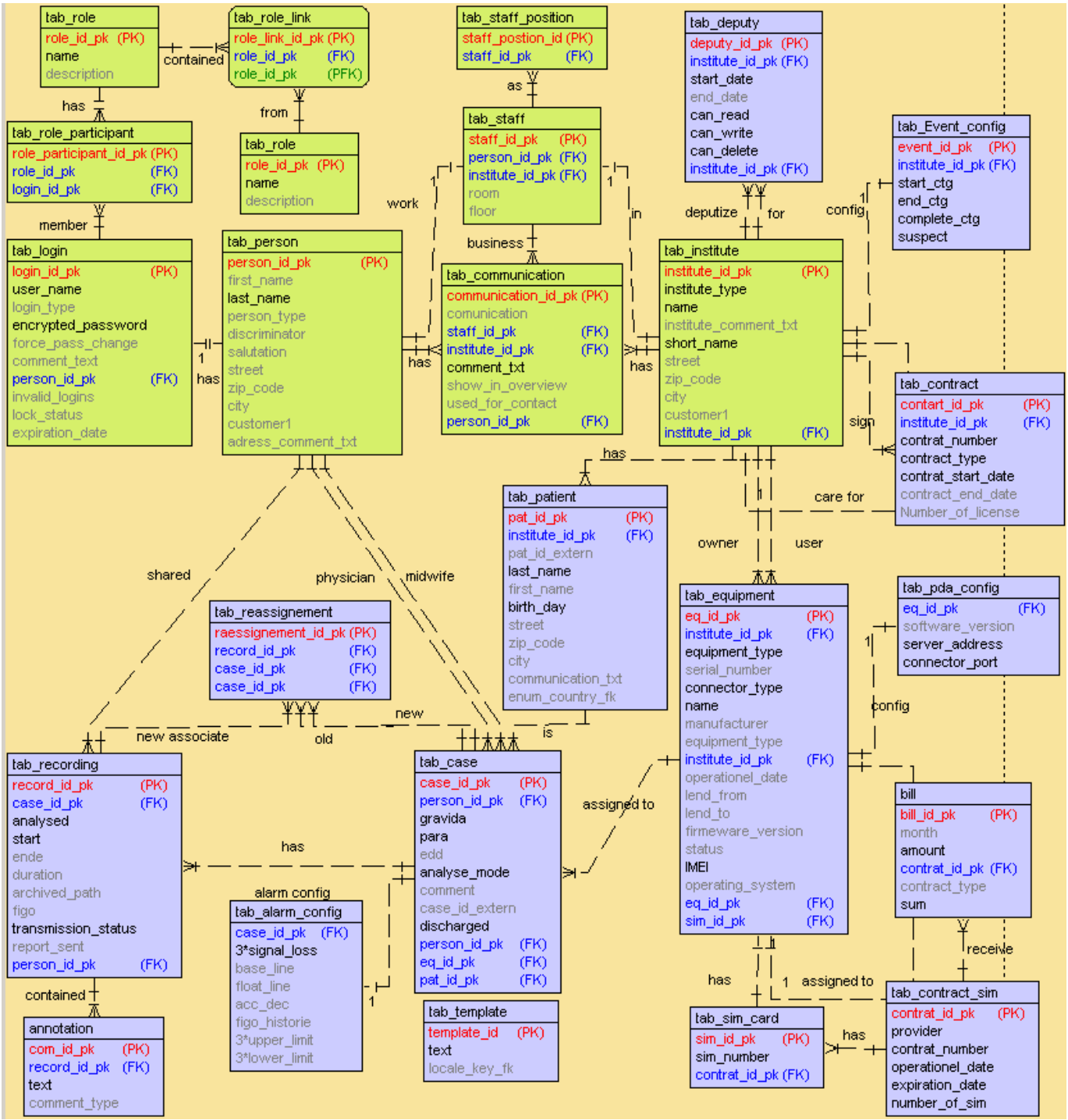



Abbildung 8.4: Tabellenstruktur

# Kapitel 9

# Implementierung

Nachdem die Systemarchitektur in Kapitel 8 beschrieben wurde, wird in diesem Kapitel die Implementierung des Prototyps vorgestellt. Das soll die gewünschten Anforderungen erfüllen und sich die Realisierung an das geplante Design halten. Da die Implementierung ein wesentlicher Arbeitschritt in diesem Projekt ist, werden die wichtigen Bestandteile des gesamten Systems hier im Detail behandelt.

Im Laufe der Implementierungsphase wird genauso wie bei der Designphase 8 nach dem Bottom-up Prinzip vorgegangen. Zunächst wird die Realisierung der Datenhaltungsschicht beschreiben. Anschließend wird die Implementierung der Fachkonzeptschicht erläutert, und zuletzt kommt die Verwirklichung der Präsentationsschicht.

## 9.1 Realisierung der Datenhaltungsschicht

Dieser Abschnitt befasst sich mit der Erstellung des Datenbankschemas und mit der Initialisierung der Datenbank. Für diesen Zweck wurde die Klasse “DBU_SetupCTGMobileSchema“ im Package dbupdate implementiert. Diese erbt von der Klasse „CoreDBUpdateScript“, die vom Framework *JWebbAppCore* zur Verfügung gestellt wird. Die Oberklasse bietet mehrere Funktionen zur Durchführung verschiedener Operationen auf die Datenbank an. Das Anlegen des Datenbankschemas „inklusive aller Tabellen und deren Beziehungen“ wurde mit Hilfe von vier Methoden dieser Klasse duchgeführt, die im Folgenden beschrieben werden.

```
public void createTable(Connection conn, String tableName, String primaryKeyName)
```

Die Funktion wurde verwendet, um Tabellen physikalich auf die Datenbank anzulegen. Ausschließlich Tabellen aus dem ER-Diagramm 8.4, die nicht mit dem Präfix tab_ anfangen werden dadurch erzeugt. Das Attribut „conn“ repräsentiert die Verbindung (Sitzung) zur Datenbank. In „tableName“ wird der Name der zu erzeugenden Tabelle eingegeben, und „primaryKeyName“ wird für den Primärschlüssel verwendet.

```
public void createAuditableTable(Connection conn, String rawTableName)
```

*createAuditableTable* ist eine weitere Funktion zur Erstellung von Tabellen. Die Funktion unterstützt das sogenannte Audit-Trail Konzept, und erzeugt somit automatisch eine Audit-Tabelle sowie ihre entsprechende Trail-Tabelle in der Datenbank.

Zusätzlich zu dem connection Attribut wird das „rawTableName“ für den Namen der zu erzeugenden Tabellen (Audit und Trail Tabellen) verwendet. Eine Audit-Tabelle beginnt mit dem Präfix tab_ gefolgt vom angegeben Namen. Das gleiche gilt für Trail-Tabellen, jedoch mit dem Präfix log_. Alle Tabellen im ER-diagramm 8.4, die mit dem Prefix tab_ anfangen, wurden anhand dieser Funktion erstellt.

Beide Tabellen erhalten zusätzlich zu den eigenen Attributen weitere Attribute. Diese sind:

- changer: steht für den Benutzer, der die Änderungen vornimmt
- rationale: hier wird der Grund des Ausführens einer gewissen Operation eingetragen
- timestamp: notiert das aktuelle Systemdatum
- deleted: Status des Datensatzes

Die Auditing-Funktion ist eine wichtige Komponente für die Sicherheit und Überwachung in einem Datenbank-Management-System. Alle wichtigen Systemaktivitäten werden in den Audit-Trail Tabellen aufgezeichnet. Somit könnenn alle Operationen z.B.(Einfügen, Updaten, Löschen eines Datensatzes), mit Hilfe einer Trigger Funktion protokolliert werden. Die entsprechende Trigger Funktion wird auch automatisch von der „JWebbAppCore“ erzeugt und in der Datenbank angelegt.

Anhand dieses Konzeptes können alle Operationen auf die Datensätze der Tabellen aufgespürt werden. Nur so kann der Administrator der Applikation durch Analyse der Audit-Trail Tabellen Muster erkennen, die Zugriffe auf die Datenbankobjekte untersuchen und die Aktivitäten der einzelnen Benutzer überwachen. Audit-Protokolle können zu den jeweiligen Benutzern zurückverfolgt werden.

```
public void addColumn(Connection conn, String rawTableName,
       String colName, int sqlType, boolean applyToLogTable)
```

Zum Hinzufügen zusätzlicher Spalten in den Tabellen kommt die „addColumn“ Funktion zum Einsatz.Damit können alle einzelnen Attributen der jeweiligen Tabellen 8.4 bis auf den Primärschlüssel mit Hilfe dieser Methode angelegt werden.

Das Attribute „colName“ gibt den Namen der Spalte an. „sqlType“ steht für den Datentyp des Attributes, der den Datentyp in der Tabelleenspalte entspricht. „applyToLogtable“ zum festlegen ob das Einfügen in einer oder in beiden (Audit & Trail ) Tabellen erfolgen muss.

```
public void addFKColumn(Connection conn, String rawTableName,
String colName, String targetTableName, boolean applyToLogTable)
```

Die Klasse „CoreDBUpdateScript“ stellt ausserdem eine wichtige Funktion („addFKColumn“) bereit, die alle Assoziationen (Beziehungen) in Datenbankschema 8.4 abbilden kann. Dafür entminnt sie den Primären Schlüssel von der Quelltabelle und fügt es in der

Zielspalte als zusätzliche Spalte an (Referenzielle Integrität). Das Attribut „rawtableName“ legt den Namen der Quelltabelle fest. „colName“ der Name der Spalte in der Quelletabelle. „targetTableName“ Name der Zieltabelle.

Für die Initialisierung der Datenbank wurden mehrere Klassen implementiert, die wiederum von der Klasse „CoreDBUpdateScript“ erben. Die Datenbank wird mit verschiedenen Daten initialisiert, so wurden z.B. Klassen für das Erstellen der Systemrollen, oder Anlegen des Administrator-Benutzerkontos, sowie das Einfügen anderer Initialdaten in der Datenbank.

## 9.2 Realisierung der Fachkonzeptschicht

Wie bereits in Kapitel 8.3 erwähnt wurde, unterteilt sich die Fachkonzetptschicht in zwei Teile. Das „Datenmodell“ das für die Abbildung der persistenten Klassen des Geschäftsobjektes auf das relationale Datenbankschema zuständig ist, „die Geschäftslogik“ die alle Transaktionen auf die Datenbank verwaltet.

### 9.2.1 Datenmodell

Nachdem das Datenbankschema angelegt wurde, müssen an dieser Stelle die entsprechenden persistenten Klassen implementiert werden. Diese Klassen bilden letztendlich das objektorientierte Datenmodell, das auf das erstellte Datenbankschema abgebildet wird. Dies erfolgt mit Hilfe von *Hibernate*.

#### 9.2.1.1 Mapping Datei

Bei Hibernate werden Datenbanktabellen auf Java-Klassen abgebildet. Jede Java-Klasse verfügt über eine korrospondierende Mapping-Datei, die in XML-Format [1] definiert wird. Die Datei und das enstprechende Java Bytecode werden vom Hibernate-Framework zur Laufzeit ausgewertet. Tabelle 9.1 zeigt ein Ausschnitt einer Mapping-Datei der Klasse DO_Patient (Patient):

Das Wurzelelement jeder Mapping-Datei ist *hibernate-mapping*, aber die eigentliche Abbildungsvorschrift steht in dem *class*-Element, das bestimmt, welche Datenbank-Tabelle auf welche Java-Klasse abgebildet werden soll. Das *name*-Attribut wählt hier die Klasse DO_Patient aus, und das *table*-Attribut (tab_patient) legt fest, in welcher Tabelle unsere Patient-Objekte landen sollen.

Das *id*-Element ist ein spezielles Element. Es definiert, dass das Attribut „id“ Persistent sein und dass es außerdem der Primärschlüssel für die entsprechenden Datensätze in der Datenbank sein soll. Das id-Element des Mappings deklariert die Java-Property patientId (name=”patientId”vom Type long (type=”long”). Diese wird auf die Spalte patient_id_pk gemappt(column name=”patient_id_pk”).

[1] Extensible Markup Language

```
<hibernate-mapping>
<class name="de.trium.ctgmobile.model.tab.DO_Patient" table="tab_patient">
<id name="patientId" column="patient_id_pk" type="long" >
<generator class="sequence"/ >
</id>
<property name="firstName" type="java.lang.String" column="first_name"/>
<property name="lastName" type="java.lang.String" column="last_name"/>
<property name="birthDay" type="java.util.Date" column="birth_day"/>
<set name="cases" lazy="true" >
<key column="patient_fk">
</key>
<one-to-many class="de.trium.ctgmobile.model.tab.DO_Case"/>
</set>
</class>
</hibernate-mapping>
```

Tabelle 9.1: Einfache Mapping-Datei für Patient

Das *generator*-Element legt dabei die Strategie für das Erzeugen der „id“ fest. Dazu stellt Hibernate zahlreiche Möglichkeiten zur Verfügung. Häufig verwendete Methoden sind zum Beispiel der HI/LO [2] -Algorithmus oder ID-Spalten und Sequenzen. Hier wurde das Sequence Algorithmus verwendet.

Das *property*-Element deklariert ein Attribut der Klasse, welches persistent werden soll. Die Mapping-Datei deklariert drei Properties: firstName, lastName und birthDay mit Datentyp String, String und Date an. Die auf die Tabellenspalten first_name, last_name, birth_day in der Tabelle tab_patient gemappt werden.

Die 1:n-Assoziation des Klassendiagramms 8.2 bedeutet aus Patientinsicht, dass eine Patientin mehrmals schwanger werden kann. In der Mapping-Datei wird dies durch das *set*-Element ausgedrückt. Das Java-Property cases (name=”cases”) enthält mehrere Schwangerschaft. Die 1:n-Assoziation zeigt auf die persistente-Klasse DO_Case (`<one-to-many class="DO_Case"/>`), die in der Tabelle durch die Fremdschlüsselspalte patient_fk (column name=”patient_fk”) realisiert ist. Das Attribut lazy legt fest, wann werden die Objekte aus der Datenbank geholt. Bei true wird der Datensatz erst beim ersten Zugriff aus der Datenbank geholt. Eine vollständige Dokumentation aller möglichen Optionen finden sich in der Referenz-Dokumentation(RFB06).

Hibernate unterstützt aller Arten von Beziehungsstrukturen und Vererbungsstruktur.

#### 9.2.1.2 Persistente Klasse

Laut Spezifikation von Hibernate die entsprechende Java-Klasse muss als POJO-Klasse [3] implementiert werden. POJO-Klasse sind Java-Klassen, die neben einem parameterlosen

[2] High/Low-Algorithmus
[3] Plain Ordinary Java Objects

Konstruktor, get()- und set()-Methoden für jede Property besitzen müssen. Allerdings Die Mapping-Dateien können auch automatisch mit Hilfe vom Generierung-Code Engine. Somit wurde das *XDoclet*-Engine [4] verwendet, um aus der Java-Quellcode die Mapping-Dateien automatisch generieren zu lassen. *XDoclet* wertet us den Java-Quellentexten neben dem eigentlichen Quelltext (Klassenname, Methodennamen und Parameter) die Java-Meta-Tags. Über die Java-Meta-Tags werden die Interfaces, Klassen und Methoden mit zusätzlichen Informationen annotiert, die den Generierungsprozess steueren.

Bei der vorliegenden Projektarbeit wurden alle persistenten Klassen mit *XDoclet* Metainformationen versehen Wie das funktioniert, werden wir im nächsten anhand eines konkreten Quellcodeausschnitts (Listing 9.1) erläutern.

```
/** (1)
 * @hibernate.class
 *  table="tab_patient"
 */
public class DO_Patient implements Serializable {
        private long    patientId;
        private String  lastName;
        private Date    birthDay;
        private Set     cases = Collections.EMPTY_SET;

        public DO_Patient() {
        }
        /** (2)
         * @hibernate.id
         *  generator-class="sequence"
         *  column="patient_id_pk"
         */
        public long getPatientId(){
                return patientId;
        }
        public void setPatientId(long patientId){
                this.patientId = patientId;
        }
        /** (3)
         * @hibernate.property
         *  column="last_name"
         */
        public String getLastName() {
                return lastName;
        }
        public void setLastName(String lastName) {
                this.lastName = lastName;
        }
        /** (4)
         * @hibernate.property
         *  column="birth_day"
         */
        public Date getBirthDay() {
                return birthDay;
        }
        public void setBirthDay(Date birthDay) {
```

[4] http://xdoclet.sourceforge.net/xdoclet/index.html

```
                this.birthDay = birthDay;
        }
        /** (5)
         * @hibernate.set role="cases" lazy="true"
         * @hibernate.collection-key  column="patient_fk"
         * @hibernate.collection-one-to-many  class="DO_Case"
         */
        public Set getCases(){
                return cases;
        }
        public void setCases(Set cases){
                this.cases = cases;
                }}
```

Listing 9.1: persistente Klasse mit XDoclet Annotation

Listing 9.1 zeigt die entsprechende persistente Klasse DO_Patient zur vorherigen Mapping-Datei (Tabelle 9.1), dieses mal aber mit XDoclet Annotation. Der Javadoc-Kommentar bei Markierung (1) vor der Klassedeklaration legt die Tabelle, auf die die Klasse D0_Patient gespeichert werden soll, fest. DO_Patient besitzt die Attribute patientId, lastName, BirthDay und eine Liste cases (Scwangerschaft). Der Javadoc-Kommentar zur get-Methode jedes Attributs enthält Informationen über dessen Mapping. Diese Kommentare sind mit Ziffern im Quellcodeausschnitt markiert.

Der Kommentar bei Markierung (2) beschreibt die Eigenschaften des Primärschlüssels in der Tabelle tab_patient. Der erste Tag definiert dabei die Generierungsmethode für den Primärschlüssel, während der zweite, die Eigenschaften der Primärschlüssel-Spalte in der Tabelle bestimmt.

Für alle einfachen Attribute wie in (3) und (4) dargestellt, reicht der Tag @hibernate.property, gefolgt vom gewünschten Spaltennamen damit dieses in die Mapping-Datei der Klasse aufgenommen wird.

Kommentar (5) beschreibt die Umsetzung der 1:n-Beziehung mit der Klasse DO_Case. Die Abbildung einer 1:n-Beziehung in einer relationalen Datenbank wird durch eine Fremdschlüsselbeziehung realisiert. Dabei erhält die Tabelle der aggregierten Elemente einen Fremdschlüssel auf das Eins-Element. In der objektorientierten Welt kann dieser Zusammenhang durch eine Liste, Map, Bag oder Set realisiert werden. Eine ausführliche Dokumentation aller Optionen der XDoclet Annotation für Hibernate finden Sie auf ihre Homepage (XDo).

#### 9.2.1.3 Konfiguration

Nun haben wir jetzt eine persistente DO_Patient-Klasse und ihre korrospondierende Mapping-Datei, die mit *XDoclet* automatisch generiert wurde. Hibernate muss aber eine Verbindung mit einem Datenbankmanagementsystem aufnehmen. Es müssen also mindestens die Angaben gemacht werden, die das vom Hibernate verwendete JDBC benötigt: Den Treiber, die Verbindungs-URL und Authentifizierungsdaten. Da Hibernate verschiedene Datenbankmanagementsysteme verwenden kann und diese sich in der Regel durch den zur Verfügung gestellten SQL-Sprachumfang unterscheiden, muss auch das Datenbankmanagementsystem

(der SQL-Dialekt) angegeben werden. Zusätzlich muss auch in dieser Datei alle persistente Klassen aufgelistet werden, damit Hibernate die auf die entsprechende Tabellen in Datenbakn abbilden kann. Üblicherweise heißt diese Datei *hibernate.cfg.xml*. Tabelle 9.2 zeigt ein Beipiel einer Konfigurationsdatei, die im nächsten Beispiele im Abscnitt 9.2.2 verwenden werden.

```
<hibernate-configuration>
<session-factory>
<!--Datenbank-Verbindung Einstellungen -- >
<property name="hibernate.connection.driver_class">org.postgresql.jdbcDriver</propert
<property name="hibernate.connection.url">jdbc:postgresql://localhost/ctgmobile</pro
<property name="hibernate.connection.username">postgres </property>
<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property
<!--Auflistung der gemappten Klassen -->
<mapping resource="de/trium/ctgmobile/model/tab/DO_Patient.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

Tabelle 9.2: Konfiguration Datei für Hibernate

### 9.2.2 Geschäftslogic

Im vorangegangenen Abschnitt wurde beschrieben, wie das Geschäftsobjekt seine Daten persistent in einer relationalen Datenbank speichert, angelegt und es an *Hibernate* angebunden wird. Es wurden die zugehörige persistente Klassen mit der entsprechenden *XDoclet* Annotation geschrieben sowie Konfigurationsdatei erstellt. Im folgenden Kapitel soll es nun darum gehen wie genau das Arbeiten mit dieser persistente Klassen funktioniert. Unter anderem wird gezeigt, wie das Anlegen neuer Objekte und das Speichern dieser Objekte in der Datenbank, sowie das Manipulieren vorhandener Datenbankeinträge von statten geht. Diese Operationen werden häufig unter dem Kürzel „CRUD“ [5] zusammengefasst.

Dabei spielt die Klasse *Transaction* von *JWebbAppCore* eine zentrale Rolle. sie stellt automatisch ein Objekt vom *UserContext*(enthält Angaben zur eingeloggten Benutzer : LoginId, personId Vorname, Nachname etc) sowie ein *Session*-Objekt von *Hibernate* zur Verfügung, das aus dem *Sessionfactory* gewonnen ist. Unter anderem bietet die Klasse *Transaction* zwei wichtige Methoden an. Die *checkConstraints* Methode, die überprüft ob der Benutzer berechtigt ist, eine bestimmte Operation durchzuführen ist.

*execute* Methode, nur innerhalb dieser Funktion können verschiedene Operationen (Transaktionen) auf die Datenbank ausgeführt werden. Eine Operation kann aus einfachen Zugriff auf einem Geschäftsobjekt bestehen, andere enthalten zusätzliche Geschäftsregeln und führen mehrere Operationen auf die Objekte des Datenmodelles aus. Alle Operationen, die

[5] Create, Read, Update, Delete

innerhalb *execute* Funktionen sind, werden automatisch in einer einzigen Transaktion ausgeführt (ACID Prinzip).

Alle Transaktionen-Klassen müssen aus der *Transaction* oder ihre Unterklasse *UserTransaction*, *SystemTransaction* abgeleitet werden. Dabei wird die *UserTransaction* Klasse, wenn es um eine sichere Transaktion handelt, die unter den eingeloggten Benutzer in Datenbank protokolliert werden, verwendet. Beim „SystemTransaction“ handelt sich um eine Transaktion, die vom System auslöst wird. Listing **??** zeigt den Standardgerüst für die Transaktionen-Klasse.

```
public class TransactionPatient extends Transaction {

        public TransactionPatient() throws Exception {
        /* konstruktor kann verschiedene Attributen
         * enthalten, je nach Transaktion.
         */
        }
        @Override
        protected void checkConstraints() throws Exception {

        /* Hier wird überprüft, ob der Benutzer berechtigt ist,
         * diese Operation durchzuführen
         */
        }
        protected Transaction execute() throws Exception {

        /* Hier folgen die Zugriffsoperationen auf die Datenbank,
         * die innerhalb eine einzige Transaktion ausgeführt werden.
         */
        }
}
```

Listing 9.2: Standardgerüst für Transaktionen

Beim Ausführen von dieser Klasse wird zuerst die *checkConstraints* Methode aufgerufen. Die überprüft, ob der Benutzer berechtigt in seinem Kontext diese Operation durchzuführen. Dafür stellt das *JWebbAppCore* mehrere Schnittstellen und Klassen zur Ermitellung der Benutzerzugriffsrechte auf einzelnen Datensektionen zur Verfügung. Auf die will man nicht genau eingehen. Erst wenn die erste Methode erfolgreich (Keine Exception) ausgeführt wurde, dann wird die nächste *execute* Methode aufgerufen, um die Zugriffsoperationen auf die Datenbank durchzuführen.

#### 9.2.2.1 Initialiserung der Session-Factory

Als nächstes muss eine *SessionFactory* initialisiert werden, dann nur mit ihrer Hilfe können *Session*-Objekte erzeugt werden, über die die eigentlichen Datenbankoperationen laufen, optional als Transaktionen. Sie sollte pro Datenbank als Singleton [6] existieren. Wenn mehrere Benutzer mit der Datenbank verbunden sind bzw. die Applikation ausführen, so sollten

[6] Es stellt sicher, dass zu einer Klasse nur genau ein Objekt erzeugt werden kann und ermöglicht einen globalen Zugriff auf dieses Objekt

sie ein gemeinsames *SessionFactory* Objekt benutzen, um dies zu gewährleisten ist es sinnvoll folgende Hilfsklasse (Listing **??**) einzubinden. Die wurde aber in dieser Projekt nicht eingesetzt. Denn wie es schon bereits erwähnt wurde, die Klasse *Transaction* stellt *Session* Objekt zur Verfügung.

```
public class HibernateUtil {

        private static final SessionFactory sessionFactory;
        //Static Block = eine Art Konstruktor für das Klassenobjekt selbst
        static{
        try{
                //Neue SessionFactory durch hibernate.cfg.xml erstellen
                sessionFactory = new Configuration().configure()
                .buildSessionFactory();
        }catch(Throwable ex){
                throw new ExceptionInInitializerError(ex);
                }
        }
        public static SessionFactory getSessionFactory(){
                return sessionFactory;
        }
}
```

Listing 9.3: Hilfsklasse für SessionFactory

Wie zu erkennen ist, wird in dieser Klasse mit Hilfe des static Blocks genau eine *SessionFactory* erzeugt. Die Methode der Erzeugung ist Standard und wird gewöhnlich immer so gehandhabt. Die Informationen werden der Konigurationsdatei aus Kapitel 9.2.1.3 entnommen. Sollte ein Fehler auftreten wird eine Exception geworfen. Über diese Klasse ist von nun an auf die *SessionFactory* zuzugreifen.

Jetzt können wir mit der *SessionFactory* eine *Session*, die zentrale Schnittstelle zu Hibernate öffnen. Die Hibernate *Session* bietet Funktionen zum Laden, Speichern, Löschen von Objekten an.

```
Session hs = HibernateUtil.getSessionFactory().openSession();
```

Jetzt steht der Benutzung unsere gemappten Klassen nichts mehr im Wege. In den nächsten Abschnitten wird gezeigt, wie mit der *Session* die grundlegenden Datenbank-Operationen ausgeführt werden.

#### 9.2.2.2 Speichern eines Objekts

Zuerst wird gezeigt, wie ein neues Objekt in die Datenbank gespeichert und sein Zustand sich von transient auf persistent ändert **??**. Ein, in Java, mit new angelegtes Objekt, das anschließend mit den entsprechenden gettern und settern einen angepassten Zustand erhält, existiert vorerst nur im Hauptspeicher und ist demnach transient. Persistent wird es, indem mit Hilfe einer Session die Methode *save()* aufgerufen wird. In das Grundgerüst (siehe Listing 9.2) ist demzufolge dieser Code einzufügen um eine neue Patientin zu speichern:

```
                DO_Patient patient = new DO_Patient();
                patient.setLastName(Patient);
                patient.setBirthDay(new Date(1978,01,03));
```

```
session.save(patient)
```

Weil der Primärschlüssel von Hibernate generiert wird, belegen wir nur das lastName-Property, birthDay-Property, während der Wert „0“ für patientId bleibt. Die Methode save fügt das neue Objekt in die Datenbank ein, das dort noch nicht existiert. Dazu erzeugt die Methode ein INSERT- Statement. Da wir automatisch generierte Primärschlüssel (sequence) verwenden, generiert *save* vorher einen Identifier und weist ihn dem patientId-Property zu. Nach erfolgreicher Übertragung steht die Patientin schließlich in der Datenbank, so dass er nicht mehr verloren gehen kann.

#### 9.2.2.3 Laden und Verändern eines Objekts

Um ein Objekt aus der Datenbank in den Speicher zu laden und zu verändern ist im einfachsten Fall der Weg über die Primärschlüssel zu gehen. Eine Patientin mit bekannter patientId lädt man wie folgt:

```
DO_Patient patient=(DO_Patient)hs.get(DO_Patient.class,new Long(patientId));
```

Neben der Methode *session.get()* existiert auch noch die Methode *session.load()*, um ein Objekt aus der Datenbank zu laden. Diese Methode hat jedoch den Nachteil, dass sie, wenn der entsprechende Eintrag in der Datenbank nicht vorhanden ist, eine Exception wirft. Die Methode *get()* hingegen gibt in einem solchen Fall null zurück.

Nachdem ein Obejkt geladen ist, können wir unsere Veränderungen an diesem Objekt vornehmen. und speichern die Änderungen durch *merge()*, was eine UPDATE-Statement in der Datenbank auslöst.

```
DO_Patient patient=(DO_Patient) hs.merge(patient);
```

#### 9.2.2.4 Löschen eines Objekts

Um ein Objekt aus der Datenbank zu löschen, bietet Hibernate die Session-Methode *delete()*. Sie löst ein SQL-DELETE-Statement aus. Eine Patientin mit bekannter patientId löscht man wie folgt:

```
DO_Patient patient=(DO_Patient)hs.load(DO_Patient.class,new Long(patientId));
hs.delete(patient);
```

Das Beispiel lädt eine DO_Patient-Objekt mit dem zugehörigen patientId (Identifier) und löscht es anschließend.

#### 9.2.2.5 Abfragen

Sofern die id’s oder Primärschlüsselen der benötigten Objekte nicht bekannt sind, so ist eine Abrafge (Query) nicht zu umgehen. Abfragen können in Hibernate mittels drei verschiedener Techniken durchgeführt werden:

- SQL-Statements Die Verwendung „normaler„ SQL-Statements bringt allerdings einen großen Nachteile mit sich: Man verliert die Datenbankunabhängigkeit.

- HQL (Hibernate Query Language) Eine objektorientierte Ausdruckssprache, deren Synthax an SQL angelehnt ist. HQL garantiert Datenbankunabhängigkeit Eine einfache Abfrage wird mit Hilfe der Methode *createQuery(String)*, die über ein *Session* Objekt aufgerufen werden kann, erstellt. Die Methode liefert eine Referenz auf ein *Query* Objekt zurück mit dessen Hilfe über die Methode *list()* das Ergebnis der Abfrage in eine *Collection* geladen wird. Sofern zuvor bekannt ist, dass die Query nur genau ein Objekt zurückliefert, steht neben der Methode *list()*, die Methode *uniqueResult()* zur Verfügung. Eine einfache HQL Query sieht beispielsweise folgendermaßen aus:

```
Query qr = session.createQuery(from DO_Patient order by lastName)
List patientinen= qr.list();
```

  Diese Abfrage liefert alle Patientinen nach Name sortiert zurück.

- HCA (Hibernate-Criteria-API) Eine objektorientierte API, deren Klassen Methoden für unterschiedlichste Abfragen kapselt. Vorteil der Criteria-API ist die Typsicherheit. Fehler, die sonst erst zur Laufzeit erkannt werden, führen hier sofort zu einem Compiler-Fehler, was bei umfangreicheren Anwendungen wertvolle Zeit sparen kann. Die Hibernate-Criteria-API garantiert ebenfalls Datenbankunabhänigkei. Das geschieht, in dem man die Methode *createCritria*, die einen Referenz auf ein *Criteria* Objekt zurück gibt. Das sämtliche Funktionen zur Formelierung verschiedener Bedingungen zur Verfügung stellt. Die gleiche Abfrage in HPA sieht folgendermaßen aus:

```
Criteria crit = hs.createCriteria(DO_Patient.class);
.addOrder(Order.asc(lastName));
List patientinen=crit.list();
```

In der Praxis findet man sowohl HQL als auch API im Einsazt, deswegen wurde auch beide Verfahren in dieser Projekt je nach Abfrage verwendet.

## 9.3 Realisierung der Präsentationsschicht

Die Darstellung der, aus der Fachkonzeptschicht kommenden Daten wird in der Präsentationsschicht realisiert. Im Allgemeinen werden Standard Technologien wie z.B. JSP (Java Server Page) oder HTML () verwendet. Im Gegensatz zu der darunter liegenden Schicht, die Objekt-Orientiert ist, bieten diese Technologien eher flache Dateien statt Objekte an was das „Mismatch Impedance"Problem verursacht. Um dieses Problem zu bereinigen, hat *Trium* ein komplettes GUI-Framework basierend auf Objektorientiertem Design entwickelt. Das so genannte *Formengine*Framework wurde in Java als Programmiersprache realisiert und erlaubt mehrere Darstellungsmöglichkeiten der Repräsentationsschicht (z. Z. nur XHTML 1.0 implementiert). Das *Formengine* wurde ähnlich zu den AWT/Swing/MFC Klassenbibliotheken entwickelt, und somit folgt es in der Architektur relativ streng dem MVC [Ref] Entwurfsmuster, das die Stärken der objektorientierten Programmierung voll ausspielt.

Das *Formengine* dessen Architektur in Abbildung 9.1 dargestellt ist, schreibt folgenden Ablauf vor: Nachdem eine Benutzerinteraktion mit der GUI (z. B. Betätigen eines Buttons) erfolgt ist, erhält der Controller ein Event. Für jeden beobachteten Event wird ein FormEventListner (vgl. Java EventListner) registriert, welcher die entsprechenden FormEventHandler (Bearbeiter) Methoden aufruft (vgl. Java EventHandler). Abhängig von der Interaktion des Benutzers (Request) ändert der Controller entweder das Modell oder die View oder beide Komponenten.

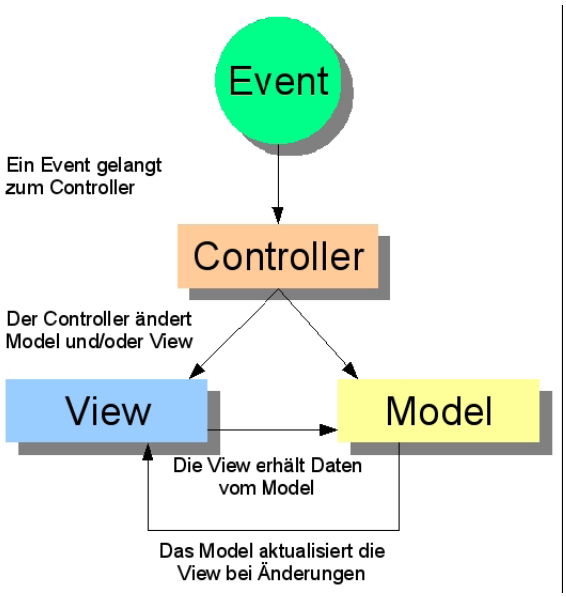



Abbildung 9.1: FormEngine Architektur

Im Folgenden werden die Aufgaben der einzelnen Modulen kurz erläutert:

- Modell: ist durch die in der Fachkonzeptschicht zur Verfügung gestellten Datenobjekte vertreten und ist somit für die Beschaffung aller notwendigen Anwendungsdaten von der Datenbank zuständig

- View: ist für die Visualisierung der Anwendungsdaten zuständig. Dafür stellt das FormEngine zahlreiche Klassen und Schnittstellen zur Entwicklung verschiedener Benutzeroberflächen zur Verfügung. Die vereinfachte Hierarchie der verschiedenen Elemente einer solchen Oberfläche sind in Abbildung 9.2 dargestellt.

- Controller: führt die Initialisierung und die Request-Verarbeitung durch, verwaltet die Datenübergabe und ist für die Steuerung der View zuständig. Dafür wird das Event Handling Prinzip eingesetzt (Abbildung 9.1).

Basierend auf das FormEngine wurden für sowohl MD, TMZ sowie Mandanten und für das Darstellen und editieren der dazugehörigen Datensektionen, abstrakte Benutzeroberflächen konzipiert. Die Darstellung und der Inhalt der GUI's werden je nach Rolle des

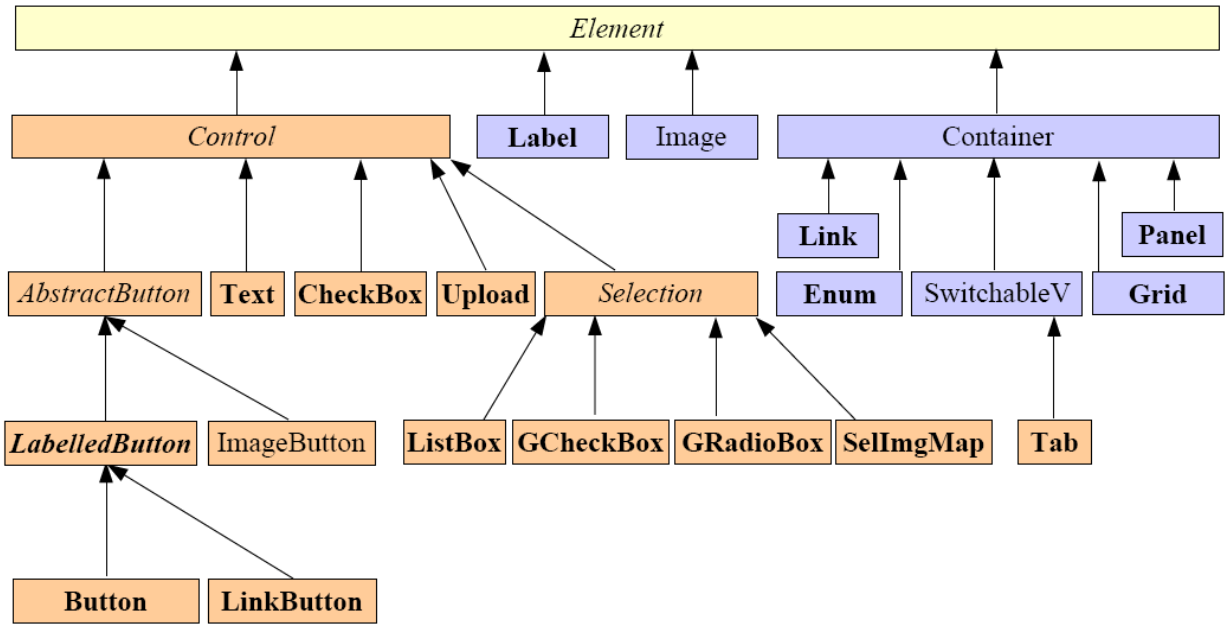



Abbildung 9.2: Übersicht der Formengine-Element

angemeldeten Benutzers angepasst. So werden zum Beispiel Felder je nach Benutzer hinzugefügt oder entfernt. Die GUI Parameter werden entweder von einer Konfigurationsdatei oder direkt aus der Datenbank gelesen. Demzufolge erstellt das FormEngine dynamisch das entsprechende (X)HTML der später im Browser dargestellt wird. Im Folgenden Listing 9.4 ein Abschnitt der Konfigurationsdatei aus der, die Parameter für die Präsentation einer Patienten Datensektion in der GUI eingelesen wird:

```
#####################################################################
# Patient Label Window                                              #
#####################################################################
txt_patient_add                   = Add Patient
txt_patient_edit                  = Edit Patient
txt_patient_label_birthday        = Birthday
txt_patient_label_externpat       = Admission number
txt_patient_label_communication   = Phone
txt_patient_label_comment         = Comment
txt_patient_label_name            = Name
txt_assign_to_mandant             = assign this Patient to Mandant
/*
 * mandatory fields
 */
txt_patient_last_name             = Last Name*
txt_patient_first_name            = First Name*
txt_patient_birth_day             = Birthday*
txt_title_hot                     = Hotline
txt_assign_to_mandant             = After creation,
                                    assign as Patient to:
/*
 * Patient Text label
```

```
 */
txt_pat_ov_title                    = Patient Overview
txt_title_pat                       = Name
txt_title_birth                     = Birthday
txt_title_street                    = Street
txt_patient_extern_id               = patient Id
txt_patient_added                   =The patient has been created,
                                     please specify gestation details
%
```

Listing 9.4: Ausschnitt aus der Konfigurationsdatei

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Literaturverzeichnis


[Agr] AgroUML, Tigris.org: *LaTeX- Use Case Tool*. http://argouml.tigris.org,

[Bal05] Balzert, Heide: *LaTeX- Lehrbuch der Objektmodellierung*. Analyse und Entwurf mit der UML 2. Elsevier, Spektrum Akad. Verl., 2005

[BB04] Bernd Brügge, Allen H. D.: *LaTeX- Objektorientierte Softwaretechnik*. Pearson Studium, 2004

[BC05] Bauer Christian, King G.: *LaTeX- Hibernate in Action*. Manning, 2005

[Ber04] Berg, Wilfried: *LaTeX- Telemedizin und Datenschutz*. 2004

[Bun07] Bundesdatenschutzgesetzes: *LaTeX- Datenschutz*. http://www.datenschutz-berlin.dehttp://www.datenschutz-berlin.de, 2007

[CS] Case Studio, Toad Data M.: *LaTeX- Toad Data Modeler*. http://www.casestudio.com/enu/default.aspx,

[Die99] Dierks, Christian: *LaTeX- Rechtliche und praktische Probleme der Integration von Telemedizin in das Gesundheitswesen in Deutschland*. http://cdl.niedersachsen.de/blob/images/C1231690_L20.pdf, 1999

[Duf] Duftschmid, Georg: LaTeX- Richtlinien zur Planung und Realisierung telemedizinischer Anwendungen / Medizinische Universität Wien, Universitätsklinik für Dermatologie. http://www.meduniwien.ac.at/msi/mias/papers/Duftschmid2005a.pdf, . – Forschungsbericht

[EA] Enterprise Architect, Sparx s.: *LaTeX- Use Case Tool*. http://www.sparxsystems.com.au/ea.htm,

[For07] Forbrig, Peter: *LaTeX- Objektorientierte Softwareentwicklung mit UML*. Hanser, 2007

[Fär03] Färber, Georg: *LaTeX- Software Engineering*. Institute for Real-Time Computersystemes, 2003

[Fra07] Franz, Klaus: *LaTeX- Handbuch zum Testen von Web-Applikationen.* Springer Berlin Heidelberg, 2007

[Gna02] Gnatz, Michael Andreas J.: *LaTeX- Vom Vorgehensmodell zum Projektplan*, Technische Universität München, Diss., 2002

[GNU] GNU.org: *LaTeX- Free Software.* http://www.gnu.org,

[Gud06] Gudenberg, Jürgen Wolff v.: *LaTeX- Software-Entwurf mit UML 2.* Springer, 2006

[Har01] Harböck, Christian: *LaTeX- Richtlinienkonforme Softwareentwicklung von Medizinprodukten am Beispiel eines internetbasierten CTG Monitors mit LabVIEW*, Technische Universität München, Diplomarbeit, 2001

[HT01] Härder Theo, Rahm E.: *LaTeX- Datenbanksysteme.* Konzepte und Techniken der Implementierung. Springer, 2001

[Jac92] Jacobson, Ivar: *LaTeX- Object oriented software engineering.* ACM Press u.a., 1992

[JF02] Jürgen Fleckenstein, Thomas M.: *LaTeX- Telemedizin in Bayern.* München : Bayerisches Staatsministerium für Wissenschaft, Forschung und Kunst, 2002

[KA04] Kemper Alfons, Eickler A.: *LaTeX- Datenbanksysteme.* Oldenbourg, 2004

[KH98] Kaindl Hermann, Tippold P. Lutz Benedikt B. Lutz Benedikt: *LaTeX- Methodik der Softwareentwicklung.* Vieweg, 1998

[KP05] Kleinschmidt Peter, Rank C.: *LaTeX- Relationale Datenbanksysteme.* Springer, 2005

[MB02] Marion Bultmann, Heinz Biermann Jürgen Engels Walter Ernestus Udo Höhn Rüdiger Wehrmann Andreas S. Rita Wellbroc W. Rita Wellbroc: *LaTeX- Datenschutz und Telemedizin - Anforderungen an Medizinnetze* -. http://cdl.niedersachsen.de/blob/images/C1231690_L20.pdf, 2002

[Med] Medizin, Magazin: *LaTeX- Telemedizin: Hilfe aus der Distanz*

[MM00] Marlene Maheu, Allen A. Whitten P. P. Whitten P.: *LaTeX- E-Health, telehealth, and telemedicine.* San francisco : Jossey-Bass A Wiley Company, 2000

[Org04a] Organization(ISO), International S.: LaTeX- Interoperability of telehealth systems and networks - part 1: Introductions and definitions. ISO / TC 215 / TR 16056-1: 2004 / http://www.americantelemed.org/ICOT/ISO-TH-TR-Pt1-Nov-15-2002.pdf. 2004. – Forschungsbericht

[Org04b] Organization(ISO), International S.: LaTeX- Interoperability of telehealth systems and networks - part 2: Real-time systems. ISO / TC 215 / TR 16056-2: 2004 / http://www.americantelemed.org/ICOT/ISO-TH-TR-Pt2-Nov-15-2002.pdf. 2004. – Forschungsbericht

[Pag] Page, Hibernate H.: *LaTeX- object/relational persistence.* http://www.hibernate.org/,

[Per] Persistenz: *LaTeX- Komplexe Datenbank-Anwendungen (Persistenz).* http://www.it-infothek.de/fhtw/semester_8/db_anwendungen_03.html#db_anw_03_01,

[Pos] Poseidon, Gentleware: *LaTeX- Use Case Tool.* http://www.gentleware.com,

[PP] PostgreSQL PostgreSQL, Die V.: *LaTeX- Die Vorteile von PostgreSQL.* http://www.thinx.ch/topic7830/story20763.html,

[RFB06] Robert F. Beeger, Stefan Roock Sebastian S. Arno Haase H. Arno Haase: *LaTeX- Hibernate.* dpunkt-Verlag, 2006

[Vis] Visio, Microsoft O.: *LaTeX- Use Case Tool.* http://germany.trymicrosoftoffice.com/default.aspx,

[Wik07] Wikipedia: *LaTeX- telemedizinische Anweundungen.* , 2007

[WJF05] Wolf-Joachim Fischer, Benjamin Homberg Harald Korb Christian Lührs Thomas Norgall Peter Rumm Silke Scmidt Holger Strehlau-Scwoll Jürgen S. Stefan Hey H. Stefan Hey: LaTeX- Thesenpapier Telemonitoring / VDE Initiative MikroMedeizin. 2005. – Forschungsbericht

[WN02] W. Niederlag, Heinz U. L.: *LaTeX- Telemonitoring und Tele Home Care.* Dresden : Health Academy, 2002

[WN04] Wolfgang Niederlag, Albrecht Hempel Heinz U. L. Bernd Lüderitz L. Bernd Lüderitz: *LaTeX- Telekardiologie.* Dresden : Health Academy, 2004

[XDo] XDoclet: *LaTeX- XDoclet Annotation für Hibernate.* http://xdoclet.sourceforge.net/xdoclet/tags/hibernate-tags.html,