

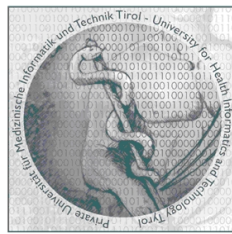
Aus dem Institut für Informationssysteme des Gesundheitswesens

Die XML basierte „Clinical Document Architecture“(CDA) für medizinische Befundberichte und Arztbriefe

Masterarbeit zur Erlangung des Titels Master of Science der Medizinischen Informatik der
Privaten Universität für Gesundheitswissenschaften Medizinische Informatik und Technik
Tirol (UMIT)

vorgelegt von Franz Oberacher, Dr. med.
aus Kitzbühel

Hall in Tirol, 2006



UMIT

Zusammenfassung

Unter kooperativer Patientenversorgung versteht man die kontinuierliche, an den Bedürfnissen des Patienten orientierte Kooperation von Krankenhäusern, niedergelassenen Ärzten und anderen Einrichtungen der Patientenversorgung. Eine wichtige Voraussetzung für eine erfolgreiche kooperative Patientenversorgung ist ein umfassender Informationsaustausch zwischen allen Beteiligten.

In diesem Zusammenhang wird der Verwendung moderner Informations- und Kommunikationstechnologie wichtige Bedeutung zugemessen- zum Beispiel beim elektronischen Austausch von medizinischen Dokumenten.

„Health Level Seven“ (HL7) ist eine Organisation, welche Standards für den elektronischen Austausch, das Management und die Integration von Daten im Gesundheitswesen entwickelt. Die „Clinical Document Architecture“ (CDA) ist ein von HL7 entwickelter Standard für die maschinelle Verarbeitung und den institutsübergreifenden elektronischen Austausch von medizinischen Dokumenten. CDA ist in der so genannten „Extensibel Markup Language“ (XML), einem plattformunabhängigen Textformat implementiert.

Im klinischen Alltag existiert eine Vielzahl verschiedener elektronischer Anwendungssysteme zur Generierung medizinischer Dokumente- wie Arztbriefe und Befundberichte. Diese Dokumente unterliegen einerseits keinerlei Standardisierung hinsichtlich Aufbau und Inhalt und liegen andererseits in proprietären Formaten vor. Maschinelle Verarbeitung und elektronischer Austausch dieser Dokumente werden dadurch erschwert.

Die Motivation für diese Arbeit ist es daher prototypisch für eine konkrete medizinische Fragestellung ein elektronisches Dokumentationssystem zu implementieren, welches automatisch aus den dokumentierten Befunden einen Bericht generiert, welcher dem Standard der Clinical Document Architecture entspricht.

Die medizinische Fragestellung ist das Krampfaderleiden der unteren Extremität des Menschen- auch als Varikosität bezeichnet. Die Implementierung sollte als Webapplikation erfolgen. Als Anhaltspunkt für die zu erfassenden Merkmale und auch für das Layout des zu generierenden Befundberichtes dienen vorliegende Formulare, welche papierbasiert im klinischen Alltag zur Dokumentation verwendet werden.

Für die Durchführung der Arbeit stellen sich drei Probleme. Zunächst muss geklärt werden, welche Daten erfasst werden müssen, um daraus einen CDA konformen Bericht zu generieren. Als nächstes Problem muss aus den erfassten Daten dynamisch schemagültiges XML generiert werden. Schließlich müssen auf das generierte XML Dokument Layout-

Anweisungen angewendet werden, um eine für das menschliche Auge übersichtliche und gut strukturierte Ausgabe der erhobenen Daten in einem Webbrowser zu ermöglichen.

Für die prototypische Implementierung werden folgende Komponenten verwendet: HTML Formulare für die Datenerfassung, die Skriptsprache PHP, eine MySQL Datenbank und ein Apache Webserver. Für die Darstellung des XML Dokumentes wird ein XSLT Stylesheet programmiert.

Das implementierte System generiert einen CDA konformen Befundbericht, wobei durch Anwendung eines validierenden Parser die Standardkonformität gezeigt werden kann. Eine gut strukturierte und übersichtliche Darstellung des generierten CDA konformen Befundberichtes in einem Webbrowser wird durch Verbindung des generierten XML Dokumentes mit dem programmierten Stylesheet erzielt.

Inhaltsverzeichnis

1	EINLEITUNG	1
1.1	GEGENSTAND UND BEDEUTUNG	1
1.2	MOTIVATION UND PROBLEMATIK	4
1.3	PROBLEMSTELLUNG	6
1.4	ZIELSETZUNG	6
1.5	FRAGE UND AUFGABENSTELLUNG	7
1.6	GLIEDERUNG DER ARBEIT	8
2	GRUNDLAGEN	11
2.1	EINFÜHRUNG IN XML	11
2.1.1	Textformat	11
2.1.2	Auszeichnungssprache („Markup“)	11
2.1.3	XML Syntax	12
2.1.4	Weiterverarbeitung von XML	17
2.1.5	Extensible Stylesheet Language (XSL)	18
2.1.6	XML als Speicherformat	19
2.2	CLINICAL DOKUMENT ARCHITECTURE	20
2.2.1	Einführung in CDA	20
2.2.2	Dokumentenparadigma	22
2.2.3	Architektur und Levels	22
2.2.4	Aufbau eines CDA Dokumentes (CDA Level 1 Release 1)	23
2.3	DAS KRANKHEITSBILD DER VARIKOSITAS	31
2.3.1	Definition und Epidemiologie	31
2.3.2	Anatomie	31
2.3.3	Pathophysiologie (primäre Varikose) und Ätiologie	32
2.3.4	Klinik, Stadieneinteilung	32
2.3.5	Diagnostik, Therapie, Prognose	33
3	METHODEN	34
3.1	XML SCHEMA	34
3.1.1	Aufbau und Namensräume	34
3.1.2	Schemavalidierung	35
3.1.3	Elemente und komplexe Datentypen	35
3.2	DATENBANKENTWURF	38
3.2.1	Entity Relationship Model (E/R Model)	38
3.2.2	Entities, Attribute, Relationen	38
3.2.3	Vom E/R Modell zur Relation	39
3.3	HYPertext MARKUP LANGUAGE	40
3.3.1	Formulare	40
3.3.2	Tabellen als Mittel für Web- Seiten- Layouts	42
3.4	PHP SKRIPT	43
3.4.1	Arrays in PHP	43
3.4.2	Verarbeitung von Formularen	44
3.4.3	Session Verwaltung	45
3.4.4	Zugriff auf MySQL Datenbank	46
3.4.5	XML generieren	47
3.5	EXTENSIBLE STYLESHEET LANGUAGE (XSL)	48
3.5.1	XPath	48
3.5.2	Extensible Stylesheet Language Transformations (XSLT)	50

4	DURCHFÜHRUNG	53
4.1	XML SCHEMA FÜR CDA LEVEL 1	53
4.2	DATENBANKENTWURF	58
4.2.1	Entity Relationship Model (E/ R Model)	58
4.2.2	Entities und Attribute für CDA Header	59
4.2.3	Entities und Attribute für CDA Body	60
4.2.4	Vom E/R Modell zur Relation	61
4.3	PROGRAMMIERUNG (HTML UND PHP)	67
4.3.1	Datenerfassung und Speicherung in RDDBS	69
4.3.2	Aus relationalen Daten XML generieren	75
4.4	PROGRAMMIERUNG EINES XSL STYLESHEETS	77
4.5	INSTALLATION UND KONFIGURATION (WIN XP)	83
4.5.1	Apache Webserver 2.0.54 und PHP 5.0.4	83
4.5.2	MySQL 4.1.12	84
4.5.3	phpMyAdmin 2.6.2	84
4.5.4	Java(TM) 2 SDK, Standard Edition Version 1.4.2	85
4.5.5	Xerces-J 2.7.1	85
4.5.6	Xalan 2.7.0	85
4.5.7	Weitere Werkzeuge	86
5	ERGEBNISSE	87
5.1	ERGEBNISSE ZU ZIEL 1	87
5.2	ERGEBNISSE ZU ZIEL 2	88
5.3	ERGEBNISSE ZU ZIEL 3	92
6	DISKUSSION	97
6.1	DISKUSSION DER EIGENEN ERGEBNISSE	97
6.2	AUSBLICK AUF CDA LEVEL II UND LEVEL III	99
7	VERZEICHNISSE	101
7.1	LITERATUR	101
7.2	TABELLEN	106
7.3	ABBILDUNGEN	107
7.4	CODE	107

1 Einleitung

1.1 Gegenstand und Bedeutung

Wir bezeichnen heute das 21. Jahrhundert als das Zeitalter der Informationstechnologie und seine Gesellschaft als die „Informationsgesellschaft“. Heute werden weltweit mehr Computer als Autos verkauft und das Internet hat unsere moderne Arbeitswelt bereits beträchtlich verändert [1]. Ein Zitat aus der US amerikanischen Zeitschrift Popular Mechanics aus dem Jahr 1949 lautet: „Computers in the future may weigh no more than 1.5 tons“. Es vermittelt den Eindruck einer rasanten Entwicklung, welche hier in den letzten 57 Jahren stattgefunden hat [2].

Von dieser Entwicklung haben auch die Medizin und die Gesundheitssysteme in den technisierten Ländern sehr profitiert. Einerseits kann durch den Einsatz moderner Informations- und Kommunikationstechnologien (ICT) die Qualität der medizinischen Versorgung gesteigert werden, gleichzeitig kann aber auch ein signifikanter wirtschaftlicher Nutzen erzielt werden [3],[4]. Der Einsatz von ICT für die Gesundheitsversorgung eröffnet aber auch Chancen für die Menschen in den Entwicklungsländern [5]. Der Fortschritt in den ICT verändert die Medizin, das Gesundheitssystem und die Gesellschaft selbst [6],[7]. Die „Überalterung“ unserer Gesellschaft und der zunehmend massive Kostendruck im Gesundheitssystem stellen auch große Herausforderungen für die Weiterentwicklung und den Einsatz von ICT im Gesundheitswesen dar [8],[9].

Die medizinische Informatik ist eine Disziplin, die sich mit der systematischen Verarbeitung von Daten, Informationen und Wissen in der Medizin und dem Gesundheitswesen beschäftigt [8]. Ein wichtiges Teilgebiet der medizinischen Informatik ist die medizinische Dokumentation. Unter Dokumentation versteht man allgemein die Tätigkeiten des Sammelns, Ordnen und Aufbewahrens von Informationen und Wissen, um beides zu einem späteren Zeitpunkt für ein gegebenes Ziel nutzbar zu machen [10].

Allgemein formuliert ist es das Ziel der medizinischen Dokumentation berechtigten Personen Informationen und Wissen zur Verfügung zu stellen. Und zwar vollständig, ohne Ballast, zum richtigen Zeitpunkt am richtigen Ort und in der richtigen Form. Man spricht in diesem Zusammenhang auch von Informations- und Wissenslogistik [10].

Auch als eine Konsequenz des steigenden Kostendrucks im Gesundheitswesen sind die Entwicklung und Einführung verschiedener Codiersysteme und Klassifikationen zu sehen, die Behandlungsfälle vergleichbar machen, statistische Analysen ermöglichen und so die Gesundheitssysteme transparenter und besser steuerbar machen sollen [10],[11].

Zukünftige Herausforderungen, die in besonderem Maße die medizinische Dokumentation betreffen, sind die patientenzentrierte Aufzeichnung und Wiederverwendung von Daten. In weitere Folge die patientenübergreifende Auswertung dieser Daten zu wissenschaftlichen Zwecken unter Einhaltung der Datenschutzgesetze (z. Bsp. durch Anonymisierung) [8].

Eine weitere wichtige von Experten zur Kosteneindämmung geforderte Maßnahme ist die Reduzierung der Anzahl von Spitalsbetten und der Dauer des stationären Aufenthalts der Patienten. Im Gegenzug dazu sollte ein größerer Bereich der medizinischen Behandlung in den ambulanten Bereich ausgelagert werden. Die Qualität der medizinischen Versorgung sollte durch eine vermehrte Kooperation und verbesserte Kommunikation zwischen den verschiedenen an der Behandlung eines Patienten beteiligten Personen und Institutionen verbessert werden [12],[13].

In diesem Zusammenhang wird der Verwendung moderner Informations- und Kommunikationstechnologie wichtige Bedeutung zugemessen [14].

Eine wichtige Rolle bei der kooperativen Patientenversorgung kommt dem elektronischen Austausch medizinischer Dokumente wie z. Bsp. Arztbriefen und medizinischen Befundberichten zu. Voraussetzung zur erfolgreichen Kommunikation zwischen Sender Empfänger auf der „zwischenmenschlichen Ebene“ ist nicht nur dass beide dieselbe Sprache sprechen, sondern auch dass ein Konsens herrscht bezüglich Inhalt und Bedeutung verwendeter Fachausdrücken und Terminologien [10].

Auf der technischen Ebene sind unter anderem plattformunabhängige Austauschformate, Transportprotokolle und Maßnahmen zur Gewährleistung des Schutzes dieser besonders sensiblen Daten für einen erfolgreichen elektronischen Dokumentenaustausch nötig. Zur automatischen Generierung und maschinellen Verarbeitung medizinischer Dokumente muss ein Dokumentenstandard vorliegen, der Inhalt und Aufbau dieser Dokumente klar regelt. [11],[15].

Die Extensible Markup Language (auf Deutsch übersetzt etwa: erweiterbare Auszeichnungssprache) ist ein einfaches und flexibles Textformat und eine Metasprache, mit der weitere Auszeichnungssprachen definiert werden können. XML wurde in der zweiten Hälfte der 90iger Jahre vom World Wide Web Konsortium (W3C) als ein international anerkannter Standard ins Leben gerufen, um den gestiegenen Anforderungen des Daten- und Dokumentenaustausches im World Wide Web gerecht zu werden[16],[17].

Heute gibt es bereits eine Vielzahl XML basierter medizinischer Dokumente wie Arztbriefe, Befundbereiche und Nachrichtenspezifikationen.

Manchen dieser Dokumente liegt kein Standard - anderen wiederum eine Mischung verschiedener Standards zugrunde. Gerade durch die Möglichkeit, dass mit XML jeder seine eigene Auszeichnungssprache definieren kann, besteht hier die Gefahr der Entstehung einer Vielzahl verschiedener Dokumenttypen anstatt eine Vereinheitlichung zu erreichen und einen gemeinsamen Standard für medizinische Dokumente zu finden.

Die Herausforderung besteht darin einen internationalen Architekturstandard zu schaffen, der die nationalen Unterschiede in den Gesundheitssystemen berücksichtigt, auf einem breiten fachlichen Konsens hinsichtlich Terminologie und Semantik beruht und bisherige Entwicklungen im Bereich XML basierter medizinischer Dokumente nützen kann bzw. stufenweise integrieren kann [18],[19].

Die CDA ist ein Architekturplan für den strukturellen Aufbau und den Inhalt von medizinischen Dokumenten, um diese Dokumente in elektronischer Form zu schaffen, zu speichern und für verschiedene Zwecke automatisiert und flexibel wieder verwenden zu können. Vor allem aber wurde die CDA eingeführt und wird kontinuierlich weiterentwickelt, um medizinische Dokumente im Sinne einer kooperativen Patientenversorgung übergreifend zwischen verschiedensten Institutionen der Gesundheitssysteme elektronisch auszutauschen. Als Format für CDA konforme Dokumente wird XML eingesetzt. XML besitzt eine Reihe von Eigenschaften, die es zum Einsatz für diese Zwecke prädestinieren. Ziel der CDA ist es, einen international gültigen und im klinischen Alltag umsetzbaren Standard für den strukturellen Aufbau, Inhalt und den elektronischen Austausch von medizinischen Dokumenten zu schaffen.

„Health Level Seven“ (HL7) ist eine Organisation, welche Standards für den elektronischen Austausch, das Management und die Integration von Daten im Gesundheitswesen entwickelt [20]. HL7 ist eine vom American National Standards Institute (ANSI) akkreditierte Organisation [21]. Die CDA ist Teil der HL7 Version 3 Familie für Kommunikationsstandards in der Medizin. Alle Teile der HL7 Version 3 Familie basieren auf einem gemeinsamen zugrunde liegenden Informationsmodell- dem „Reference Information Modell“ (RIM) [22]. Das RIM beschreibt alle Objekte, Aktivitäten und deren Relationen in einem generischen Gesundheitsmodell.

In der medizinischen Informatik erhofft man sich von der CDA dem Ziel eines allgemein anerkannten Standards zum Austausch und zur Speicherung medizinischer Daten und so der Vision einer reibungslosen Kommunikation zwischen unterschiedlichen Anwendungen und Organisationen im Gesundheitswesen einen Schritt näher zu kommen [23].

1.2 Motivation und Problematik

Von niedergelassenen Ärzten wird der Kommunikation entscheidende Bedeutung für eine „ideale Zusammenarbeit“ zwischen Klinik und Arztpraxis beigemessen. Arztbriefe und medizinische Befundbereiche aber auch Einweisungsscheine sind in diesem Zusammenhang als wichtige Kommunikationsmittel für die kooperative Patientenversorgung zu sehen [24].

Verschiedene Studien im deutsch- aber hauptsächlich englischsprachigen Raum haben Arztbriefe und Befundbereiche auf ihre Qualität hin untersucht. Dabei konnten einige typische Mängel festgestellt werden. Diese beziehen sich auf Verzögerungen bei der Übermittlung der Dokumente, auf zu großen Umfang der Dokumente sowie schlechte Strukturierung und Unübersichtlichkeit. Der Inhalt der Briefe soll die Bedürfnisse der niedergelassenen Ärzte berücksichtigen und alle relevanten Informationen für eine kontinuierliche Weiterbehandlung enthalten. Klare und nachvollziehbare Diagnosen, Untersuchungsergebnisse, Medikation und Therapievorschlag werden dabei als essentiell betrachtet [25],[26].

In dieser Arbeit soll prototypisch ein Dokumentationssystem für die medizinische Fragestellung Varikosität, also dem Krampfaderleiden der unteren Extremitäten des Menschen implementiert werden. Mithilfe der entwickelten Anwendung soll ein erhobener Befund elektronisch dokumentiert werden können und im Anschluss daran automatisch ein zusammenfassender Befundbericht erstellt werden können.

Es gibt bereits eine Vielzahl verschiedenster elektronischer Dokumentationssysteme für die Erstellung zusammenfassender Befundberichte in unserem Gesundheitswesen, die jedoch vor dem Hintergrund des elektronischen Austausches dieser Dokumente im Sinne einer kooperativen Patientenversorgung zwei typische Schwachpunkte aufweisen. Zum einen basieren diese Dokumente auf kommerziellen proprietären Formaten, zum anderen liegt den generierten Dokumenten kein standardisierter Aufbau zugrunde.

Daher besteht die Motivation für diese Arbeit darin, einen Befundbericht zu generieren, der dem Standard der Clinical Document Architecture entspricht

Die Problematik liegt zunächst darin, dass unbekannt ist, welche Merkmale für die konkrete klinische Fragestellung Varikosität erfasst werden müssen, um daraus einen CDA konformen Befundbericht generieren zu können. Außerdem sollte bei Bedarf auch die Möglichkeit bestehen aus den erfassten Daten statistische Analysen durchzuführen. Weiter ist unklar, wie aus den erfassten Daten ein CDA konformes XML Dokument generiert werden kann. Schließlich muss das generierte Dokument aber noch mit Formatierungen und

Layoutanweisungen versehen werden, damit es für einen Betrachter (zum Beispiel einen weiterbehandelnden Arzt) gut lesbar ist und übersichtlich dargestellt werden kann.

Als Ausgangsmaterial für diese Arbeit liegen Formulare zur Befunddokumentation der Varikosität vor. Diese Formulare sind auch im klinischen Einsatz. Sie dienen als Anhalt sowohl für die Merkmalerfassung als auch für die folgende Gestaltung des Layouts des zu generierenden Befundberichtes.

Außerdem wird zu Beginn beschlossen, dass die prototypische Implementierung des Dokumentationssystems als Webapplikation erfolgen soll und dafür die Komponenten Hypertext Markup Language (HTML), die Skriptsprache Hypertext Preprocessor (PHP) und Apache HTTP Server zum Einsatz kommen sollen. Welche generellen anderen Möglichkeiten zur Implementierung bestehen bzw. die Frage, welche anderen Technologien für die Implementierung als Webapplikation zur Verfügung stehen, ist nicht Gegenstand dieser Arbeit.

1.3 Problemstellung

1.3.1 Problem 1

Es ist unklar, welche Merkmale für eine konkrete medizinische Fragestellung erfasst werden müssen, um daraus einen Befundbericht zu generieren, der dem Standard der Clinical Document Architecture entspricht und ob XML ein geeignetes Speicherformat ist, um aus den erfassten Daten statistische Analysen durchzuführen.

1.3.2 Problem 2

Es ist unklar, wie aus relationalen Daten automatisch ein Befundbericht generiert werden kann, der dem Standard der Clinical Document Architecture entspricht.

1.3.3 Problem 3

Es ist unklar, wie der Inhalt des generierten CDA konformen XML Dokumentes für einen Betrachter übersichtlich und optisch ansprechend dargestellt werden kann.

1.4 Zielsetzung

1.4.1 Zielsetzung zu Problem 1

1.4.1.1 Teilziel 1 zu Problem 1

Es ist klar, welche Merkmale für die konkrete medizinische Fragestellung Varikosität, also dem Krampfadernleiden der unteren Extremitäten des Menschen erfasst werden müssen, sodass in einem späteren Schritt aus den erfassten Daten ein CDA konformer Befundbericht generiert werden kann.

1.4.1.2 Teilziel 2 zu Problem 1

Es ist klar, wie die erfassten Daten gespeichert werden müssen, um daraus bei Bedarf auch statistische Analysen durchführen zu können.

1.4.2 Zielsetzung zu Problem 2

Ziel ist es, aus relationalen Daten dynamisch einen Befundbericht zu generieren, welcher dem Standard der Clinical Document Architecture entspricht.

1.4.3 Zielsetzung zu Problem 3

Ziel ist die Darstellung der erfassten Daten in einem Webbrowser in Form eines übersichtlichen und gut strukturierten Befundberichtes durch Anwendung eines zuvor programmierten Stylesheets auf das generierte CDA konforme XML Dokument.

1.5 Frage und Aufgabenstellung

1.5.1 Frage und Aufgabenstellung zur Erreichung von Ziel 1

- Was ist unter Clinical Document Architecture zu verstehen? (→2.2)
- Welche Daten sind für den Header Teil des CDA Dokuments zu erfassen?(→2.2.3, →3.1, →4.1)
- Welche Merkmale sind für den CDA Body Teil zu erfassen?(→2.2.3, →3.1, →4.1)
- Ist XML als Datenspeicherformat geeignet? (→2.1.6)
- Wie ist das Datenbankdesign zu wählen? (→3.2, →4.1, →4.2)
- Programmierung der HTML Webformulare zur Dateneingabe (→3.3, →4.3)
- PHP Skriptprogrammierung zu Datenspeicherung in MySql Datenbank (→3.4, →4.3)

1.5.2 Frage und Aufgabenstellung zur Erreichung von Ziel 2

- PHP Skriptprogrammierung zur dynamischen Generierung eines schemagültigen XML Dokumentes. (→3.4, →4.3)
- Wie kann die Schemagültigkeit eines XML Instanzdokumentes geprüft werden? (→4.3.2)

1.5.3 Aufgabenstellung zur Erreichung von Ziel 2

- Programmierung eines XSL Stylesheets zur Formatierung und Darstellung des generierten XML Dokumentes in einem Webbrowser. (→3.5, →4.4)

1.6 Gliederung der Arbeit

1.6.1 Kapitel 2: Grundlagen

Die für das Verständnis von Kapitel 3 (Methoden) und 4 (Durchführung) notwendigen Grundlagen werden zusammengefasst.

- Im Grundlagenteil wird zunächst eine Einführung in XML gegeben, um die Inhalte der weiteren Arbeit verstehen zu können: Auszeichnungssprache, Erweiterbarkeit, Textformat und Zeichensatz; Elemente und Attribute, Wohlgeformtheit, Parser; XML Applikationen und XML Schemasprachen, XML und Datenbanken. Im Abschnitt XML und Datenbanken wird dargelegt, warum für die konkrete Webapplikation die Daten zuerst in einem relationalen Datenbanksystem gespeichert werden und erst sekundär daraus XML generiert wird. (→2.1). Auf die Bereiche XPath und XSL Transformation wird im Methodenkapitel näher eingegangen(→3.5, →4.4)
- Als nächstes wird die Clinical Dokument Architecture eingeführt. Motivation, bisherige Entwicklung. Projektphasen Level I bis III, genereller Aufbau mit Header und Body Abschnitt (→2.2)
- Schließlich wird im Grundlagenteil die medizinische Fragestellung erörtert. Es soll kurz auf die Pathophysiologie und Untersuchungsmethoden eingegangen werden, um die Informationsmodellierung für den Body Teil des CDA konformen Befundberichts nachvollziehen zu können. (→2.3)

1.6.2 Kapitel 3: Methoden

Die in Kapitel 4 (Durchführung) verwendeten Methoden werden erläutert.

- Zunächst erfolgt eine Einführung in die XML Schemasprache XML- Schema. Ein grundlegendes Verständnis dafür ist nötig, um das XML Schema für CDA Level 1 soweit zu verstehen, um daraus ablesen zu können, welche Daten (gemeint: Granularität und Qualität) für die konkrete Fragestellung erfasst werden müssen. (→3.1)
- Methoden und Grundlagen zum Datenbankdesign für ein relationales Datenbanksystem werden beschrieben. (→3.2)
- HTML Formulare und HTML Tabellen als Mittel für das Layout von Websites werden beschrieben. Weiter wird auf die HTTP Methoden GET und POST eingegangen, welche ein Client verwenden kann, um Formulardaten an den Server zu übergeben.(→3.3)

- Der Einsatz der Programmiersprache PHP wird anhand der Verarbeitung von Formulardaten vorgestellt. PHP wird in dieser Arbeit auch für den Datenbankzugriff auf eine MySQL Datenbank eingesetzt und verwendet, um XML zu generieren. (→3.4)
- Der Einsatz von XSL Stylesheet zur Formatierung und Darstellung des Inhaltes eines XML Dokumentes wird vorgestellt, wobei hier zur Darstellung in einem Webbrowser die Transformation der Ausgabe in HTML erfolgt. In diesem Abschnitt wird auch auf die Bereiche XPath und XSL Transformation eingegangen(→3.5)

1.6.3 Kapitel 4: Durchführung

Zeigt die Durchführung der eigenen Arbeit und Anwendung der im Kapitel 3 vorgestellten Methoden auf die konkrete Fragestellung. Zusätzlich wird die Installation und Konfiguration der für die praktische Durchführung der Arbeit benötigten Werkzeuge (Betriebssystem: WIN XP) gezeigt.

- Es wird anhand des XML Schemas für CDA Level 1 Release 1 gezeigt, welche CDA Elemente für die konkrete Fragestellung verwendet werden. (→4.1)
- Entwurf des ER Modells und des Datenbankschemas für MySQL Datenbank zur Speicherung der erfassten Daten. (→4.2)
- Beschreibung des Programmablaufs. (→ 4.3)
- Programmierung der HTML Formulare zur Datenerfassung. (→4.3.1)
- Programmierung der PHP Skripte: Formularverarbeitung, Anbindung an MySQL-Datenbank, Generierung von schemagültigem XML. (→4.3.1, →4.3.2)
- XSL Stylesheet- Programmierung, Demonstration der Arbeitsweise eines XSLT Prozessors. (→4.4)
- Installation und Konfiguration der für die praktische Durchführung der Arbeit benötigten Werkzeuge. (→4.5)

1.6.4 Kapitel 5: Ergebnisse

Die Ergebnisse der Arbeit zu den einzelnen Zielsetzungen sollen beschrieben werden.

- Datenerfassung und Datenspeicherung (→5.1)
- Schemagültiges XML aus relationalen Daten generieren (→5.2)
- Formatierung und Darstellung mit XSL Stylesheet (→5.3)

1.6.5 Kapitel 6: Diskussion der Ergebnisse

- Diskussion der eigenen Ergebnisse (→6.1)
- Ausblick auf CDA Level II und III(→6.2)

2 Grundlagen

2.1 Einführung in XML

In diesem Unterkapitel wird eine kurze Einführung in die extensible Stylesheet Language (XML) gegeben, wobei auf die für die weiteren Abschnitte dieser Arbeit relevanten Gebiete etwas näher eingegangen wird. [28]

2.1.1 Textformat

Die Extensible Markup Language (auf Deutsch übersetzt etwa: erweiterbare Auszeichnungssprache) wurde in der zweiten Hälfte der 90iger Jahre vom „World Wide Web Konsortium“ (W3C) als international anerkannter Standard ins Leben gerufen, um den gestiegenen Anforderungen des Daten- und Dokumentenaustausches im World Wide Web gerecht zu werden. [27]

XML ist ein reines Textformat. XML ist unabhängig von einer bestimmten Plattform oder einer bestimmten Applikation, weil es ein reines Textformat ist. Jedes Werkzeug, das in der Lage ist Textdateien zu lesen, kann auch XML Dokumente lesen. XML Dokumente enthalten Unicode- Text. Unicode ist ein Zeichensatz. Ein Zeichensatz bildet ein bestimmtes Zeichen im Rechner auf eine bestimmte Zahl ab. Diese Zahlen werden Code- Punkte genannt. Die Zeichenkodierung wiederum legt fest, wie diese Code Punkte in Bytes repräsentiert werden. Unicode ist ein Zeichensatz, der so groß ist, dass er alle Zeichen heute lebender Sprachen und eine Vielzahl weiterer Sonderzeichen aus verschiedensten Bereichen aufnehmen kann. Die aktuelle Version 4.0.1 (Stand Jänner 2005) umfasst über 95.000 Zeichen aus den meisten heute lebenden Sprachen sowie viele Sonderzeichen. Schätzungen gehen davon aus, dass heute durchschnittlich weniger als jeder tausendste Mensch eine Sprache spricht, die nicht vernünftig in Unicode repräsentiert werden könnte. Unicode soll in Zukunft um weitere Zeichen erweitert werden, sodass diese Zahl noch kleiner wird. Potentiell kann Unicode mehr als eine Million Zeichen aufnehmen. [29]

2.1.2 Auszeichnungssprache („Markup“)

XML ist eine Auszeichnungssprache (im Englischen „Markup Language“). Der Begriff der „Auszeichnung“ („Markup“) kommt aus dem Bereich der Drucktechnik und steht im Prinzip für Markierungen, welche ein Setzer in ein Manuskript einfügt, um es für den Druck vorzubereiten. Markup, das für die konkrete Layoutgestaltung von Text und anderen Medien verwendet wird, nennt man spezifisches Markup. Im Gegensatz zum spezifischen Markup liegt der Schwerpunkt beim generischen Markup auf den semantischen Strukturen von Informationen. Beim semantischen Markup wird Hauptaugenmerk auf die Trennung von Textlayout und Textinhalt gelegt. HTML, die „Hypertext Markup Sprache“ (etwa:

Auszeichnungssprache für Hypertexte, also Webseiten) ist eine Spezifische Markup Sprache. XML im Gegensatz dazu ist als generische Markup Sprache zu bezeichnen. Ein wesentliches Konzept in XML ist, dass Textinhalt eines XML Dokumentes und Anweisungen zu Darstellung dieses Textinhaltes immer voneinander getrennt in zwei verschiedenen Dateien abgelegt werden.

2.1.3 XML Syntax

Die XML Syntax ist einfach, sie muss aber genau eingehalten werden. Programme, welche XML Files weiterverarbeiten sind viel einfacher zu schreiben und können viel effizienter ablaufen, wenn sie sich darauf verlassen können, dass das XML -Dokument syntaktisch richtig ist, und keine automatische Fehlerkorrektur programmiert werden muss.

2.1.3.1 Elemente, Attribute, Baumstruktur

Die Basiseinheit eines XML Dokuments wird Element genannt. Ein Element besteht aus konkreten Daten und wird vom Elementbezeichner umschlossen. Synonym für Elementbezeichner könnten die Begriffe „Tag“ oder „Auszeichnung“ oder „Markierung“ verwendet werden. Konkrete Daten im Element werden durch spitze Klammern von den Tags abgesetzt. Die Summe aller konkreten Daten eines XML Dokumentes stellt den eigentlichen Inhalt eines XML Dokumentes dar. Die Elementnamen sollten möglichst klar auf die inhaltliche Bedeutung der im jeweiligen Element eingeschlossenen Daten hinweisen. Dies ist mit dem Begriff „semantische Auszeichnung“ gemeint. Man sagt auch XML ist „selbst- dokumentierend“.

Die Bezeichnungen „Eltern- und Kindelemente“ dienen dazu die logischen Beziehungen der in einem XML Dokument verschachtelten Elemente zueinander besser zu beschreiben. Ein Elternelement kann mehrere Kindelemente aufweisen- ein Element kann aber nur ein Elternelement haben. Ein Element kann nicht nur konkrete Daten enthalten, sondern selbst wiederum ein- oder mehrere andere Elemente beinhalten. Da sich Elemente nicht überschneiden dürfen und jedes nur ein „Elternelement haben darf, formen XML Dokumente eine Datenstruktur, welche in der Informatik als Baum bezeichnet wird

XML Elemente können Attribute besitzen. Ein Attribut hat einen Namen und einen Wert- beides wird dem Starttag eines Elements zugewiesen. Die Namen werden von den Werten durch ein Gleichheitszeichen getrennt, die Werte müssen in einfachen oder doppelten Anführungszeichen stehen. Ein Element kann auch mehrere Attribute enthalten, wobei diese dann unterschiedliche Namen haben müssen. Es gibt keine eindeutige Regelung oder Empfehlungen, wann Information in XML in Kindelementen oder in Attributen abgelegt werden soll und die Diskussionen dazu sind nicht abgeschlossen.

Allgemein kann man sagen, dass Attribute dazu dienen können, weitere Informationen zu den in einem Element bereits abgelegten Daten zu präsentieren. Es gilt, dass jedes Element nicht mehr als ein Attribut mit einem bestimmten Namen haben darf. Der Wert eines Attributes wird einfach als Text-String betrachtet. Die Informationsmodellierung in einer elementbasierten Struktur ist insgesamt flexibler und erweiterbarer. Typische Beispiele für die Informationsablage in Attributen sind Bild-URLs, Hyperlinks oder Geburts- und Sterbedaten und Zahlencodes.

Aus einem anderen Blickwinkel heraus kann man Elemente eines XML Dokumentes in Container und Datenelemente einteilen. Datenelemente sind alle jene Elemente, welche ausschließlich konkrete Daten beinhalten, aber keine Kindelemente mehr enthalten.

In jedem XML- Dokument gibt es ein Element, welches kein Elternelement hat. Dieses wird als das „Wurzelement“ bezeichnet. Andere Bezeichnungen dafür sind „root- Element“ oder „Document- Element“. Anhand des Wurzelementes erkennt ein weiterverarbeitendes Programm, wann ein XML- Dokument beginnt und, dass es endet, sobald der schließende Tag des Wurzelementes ausgelesen wird.

Leere Elemente beinhalten weder konkrete Daten noch Kindelemente. Im Gegensatz zu HTML müssen selbstverständlich auch leere Elemente einen schließenden Tag aufweisen.

2.1.3.2 Wohlgeformtheit und Parser

Alle XML- Dokumente müssen „wohlgeformt“ sein. Wohlgeformtheit bedeutet, dass eine Reihe von Regeln eingehalten werden muss. Einige dieser Regeln lauten wie folgt:

- Jeder Starttag muss einen dazugehörigen Endtag haben.
- Elemente dürfen geschachtelt sein, sich aber nicht überlappen.
- Es muss genau ein Wurzelement geben.
- Attributwerte müssen in Anführungszeichen stehen.
- Ein Element darf nicht zwei Attributwerte mit dem gleichen Namen besitzen.
- In den Zeichendaten eines Elements oder Attributs dürfen keine ungeschützten Sonderzeichen stehen.

Die Grammatik ist jedenfalls speziell genug, um die Entwicklung von XML Parsern zu ermöglichen. Dabei handelt es sich um Programme, die in der Lage sind, die in den XML Dokumenten ausgezeichneten bzw. „markierten“ Daten auszulesen und die Dokumente auf die Einhaltung dieser Grammatik zu überprüfen.

Entspricht ein Dokument diesem Regelwerk wird es als „wohlgeformt“ bezeichnet, anderenfalls als „nicht wohlgeformtes Dokument“ bezeichnet.

Ein XML Parser weist auf den Ort des Fehlers hin- versucht aber niemals einen Fehler zu reparieren. Mit dieser Strategie versucht man Kompatibilitätsunterschiede- wie sie zwischen Webbrowsern verschiedener Hersteller bestehen- im Vorhinein zu unterbinden. Aktuelle Versionen von Webbrowsern verschiedener Hersteller verfügen über integrierte XML Parser. Die einfachste Methode ein XML Dokument auf Wohlgeformtheit hin zu prüfen besteht daher darin, das Dokument in den Webbrowser zu laden. Ist es wohlgeformt wird es im Browser angezeigt, ansonst erfolgt die Ausgabe einer Fehlermeldung. Bevor ein XML Dokument veröffentlicht wird oder sonst in irgendeiner Weise zur Weiterverarbeitung gelangt, sollte es immer auf Wohlgeformtheit geprüft werden

2.1.3.3 Entwicklung von XML und XML Anwendungen

XML ist ein Abkömmling der „Standard Generalized Markup Language“ (SGML). SGML ist ein reines Textformat, es ist eine semantische und strukturelle Auszeichnungssprache und es ist eine Metasprache. SGML wurde in den 70er Jahren von Mitarbeitern der Firma IBM (Charles F. Goldfarb, Ed Mosher und Ray Lorie) und von weiteren mehreren Hundert Entwicklern auf der ganzen Welt entwickelt. 1986 wurde SGML als ISO Standard 8879 angenommen. Es verbuchte einige Erfolge z. Bsp. beim US- Militär, bei der US- Regierung, auf dem Raumfahrtsektor und anderen Bereichen zur effizienten Verwaltung technischer Dokumente, die Zehntausende von Seiten lang waren. Der größte Erfolg von SGML war HTML, eine Anwendung von SGML. Sir Timothy J. Berners- Lee, der „ Gutenberg des Cyberspace“, gilt als Erfinder der World Wide Web Technologie. Er lehnte die von ihm erfundene Hypertext Markup Language (HTML) syntaktisch an SGML an- etwas später wurde HTML formal als SGML DTD definiert. SGML ist außerordentlich leistungsfähig, aber auch außerordentlich kompliziert. Die offizielle SGML Spezifikation umfasst mehr als 150 sehr technische Seiten. Sie deckt viele Spezialfälle und unwahrscheinliche Szenarien ab. SGML ist so komplex, dass es kaum eine Software gibt, die es vollständig implementiert hat. Programme, die unterschiedliche Teilmengen von SGML implementiert hatten, waren oft inkompatibel zueinander. Aus diesen Gründen begannen 1996 Jon Bosak, Tim Bray, C.M. Sperberg- McQueen, James Clark und verschiedene andere, an einer „Lite- Version“ von SGML zu arbeiten. Das Ergebnis war im Februar 1998 XML 1.0. Diese Version behielt die Leistungsfähigkeit von SGML, verzichtete aber auf viele Funktionen, die sich in den vorangehenden 20 Jahren als wenig nützlich, redundant oder schwierig zu implementieren herausgestellt hatten. [30]

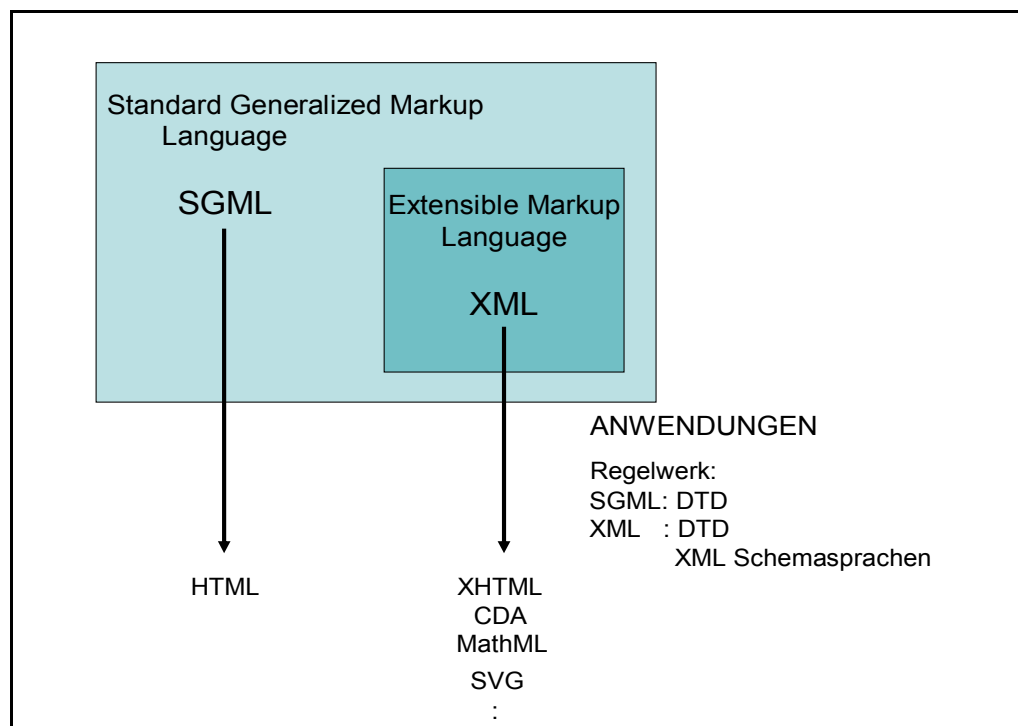


Abbildung 1: Metasprachen SGML und XML

XML ist eine Meta Auszeichnungssprache. XML erlaubt den Entwicklern und Autoren Elemente beliebig nach Ihrem Bedarf (neu) zu definieren.

Dieser Sachverhalt ist mit „erweiterbarer“ Auszeichnungssprache gemeint und ist ein Grund für den flexiblen und bereits weit verbreiteten Einsatz des Sprachstandards XML zur elektronischen Datenverarbeitung in Naturwissenschaften, Kunst, Wirtschaft und Finanzwesen. [28]

Eine XML Anwendung oder „Applikation“ ist die Anwendung von XML auf ein bestimmtes Problemfeld. Für jede XML Anwendung gibt es ein Regelwerk, welches die jeweilige Anwendung spezifiziert. Diese besagt unter anderem, wie die für die jeweilige Anwendung verwendeten Elemente heißen, in welcher Reihenfolge diese in einer konkreten Instanz dieser Anwendung vorkommen dürfen. Außerdem wird für jede XML Anwendung festgelegt, welche Kindelemente und Attribute vorkommen dürfen und so weiter. Dieses Regelwerk wird für jede XML Anwendung in einer XML Schema Sprache geschrieben. Schema Sprache wird an dieser Stelle als Sammelbegriff verwendet. Bekannte XML Schemasprachen sind „Document Type Definition“ (DTD) und „XML Schema“. [31] Daneben gibt es andere Schemasprachen wie etwa „RELAX NG“, „Schematron“, „Hook“ oder „Examplotron“.

Ein konkretes XML Dokument einer XML Anwendung muss alle in einer Schemasprache festgesetzten Regeln für diese Anwendung (erlaubte Elementnamen, Reihenfolge des Auftretens, erlaubte Kindelemente usw.) exakt befolgen.

In Analogie zur objektorientierten Programmierung kann man sagen, dass eine bestimmte XML Applikation eine bestimmte Klasse von XML Dokumenten darstellt und jedes Dokument, welches die jeweilige Grammatik befolgt, eine Instanz dieser Klasse ist.

2.1.3.4 Namensräume

Das Konzept der Namensräume [32] in XML ermöglicht den konfliktfreien Einsatz von Elementen und Attributen gleichen Namens aber verschiedener inhaltlicher Bedeutung in ein- und demselben XML- Dokument. Dieser Fall kann dann eintreten, wenn in einem XML- Dokument gleichnamige Elemente aus verschiedenen XML- Applikationen verwendet werden. Zur maschinellen Weiterverarbeitung dieses Dokuments muss für diesen Fall ein Mechanismus bereitstehen, der erkennt, dass hier Elemente gleichen Namens aber unterschiedlicher inhaltlicher Bedeutung vorliegen. Außerdem muss dieser Mechanismus jedem dieser Elemente eindeutig seine Bedeutung zuordnen können. Ein solcher Mechanismus ist in Form einer eindeutigen Zuordnung von Elementen und Attributen zu bestimmten Namensräumen gegeben. Die Eindeutigkeit der Zuordnung ist durch die Verbindung eines Elementes bzw. Attributes mit einem „Uniform Resource Identifier“ (URI) gewährleistet. [33] Die Technik der Uniform Resource Identifiers erlaubt es eindeutige Bezeichner in einem Netzwerk zu generieren. Die URIs teilen Elemente und Attribute in disjunkte Mengen auf. Elemente mit dem gleichen Namen, aber unterschiedlicher URI sind verschiedene Elemente.

Elemente gleichen Namens und gleicher URI sind gleiche Elemente. Meist verwendet eine XML Applikation einen Namensraum, es gibt aber auch XML- Applikationen, welche mehrere Namensräume verwenden. XSL, die „extensible Stylesheet Language“, verwendet unterschiedliche Namensräume für XSL Transformation und XSL Formatting Objects.

2.1.3.5 DTD und XML Schema

DTD ist in eine SGML Syntax. Im Gegensatz dazu ist XML Schema ebenfalls in XML verfasst. Das heißt jede XML Schemadatei ist ein gültiges, wohlgeformtes XML Dokument. Einige weitere Unterschiede zwischen DTD und XML Schema werden hier kurz beschrieben.

Eine DTD ist Teil des Instanzdokumentes oder wird als externe DTD in einem Instanzdokument referenziert. Ein XML Schema für eine konkrete XML Anwendung wird immer getrennt vom Instanzdokument in eine eigene XML Datei geschrieben. Die Verbindung zwischen Schemadokument und Instanzdokument erfolgt in den Processing Instructions- kurz PI genannt- am Anfang eines XML Dokumentes. PI sind Verarbeitungsanweisungen an einen Parser.

DTD ist die einzige XML Schema Sprache, die innerhalb der XML 1.0 Spezifikation selbst festgelegt wurde.

XML Schema wurde von einer W3C Entwicklergruppe geschaffen, weil DTD einige gravierende Schwachpunkte für bestimmte Anwendungszwecke aufweist. Ein großer Nachteil ist, dass das DTDs das Konzept der Namensräume nur eingeschränkt unterstützt. Dokumenttyp Definitionen geben grundlegende Strukturregeln für Dokumente vor. Für viele Anwendungen ist aber eine mächtigere und ausdrucksstärkere Grammatik nötig. Für narrative, erzählende Dokumente (Webseiten, Buchkapitel, Newsletter) sind die Kontrollmöglichkeiten, welche DTDs bieten, ausreichend. Für die Darstellung mehr datenorientierter Strukturen stoßen DTDs aber auf Grenzen.

XML Schema hingegen ist komplett in XML ausgedrückt. XML Schema ist ausdrucksstärker und genauer als DTD. Namensräume werden problemlos unterstützt. Neben den gängigen Datentypen (integer, string, float, date/ time etc.) werden auch komplexe Datentypen unterstützt. [31]

2.1.3.6 Validierung

Validierung nennt man den Vorgang des Prüfens, ob ein Instanzdokument die Grammatik der jeweiligen Applikation auch exakt befolgt. Die Validierung wird von einem validierenden Parser durchgeführt.

2.1.3.7 Beispiele für XML Anwendungen

Im Folgenden werden einige Beispiele für XML Anwendungen angeführt.

- XHTML (Extensible Hypertext Markup Language); [34] Die Neudefinition von HTML in XML. Entspricht den strengeren Syntax-Regeln von XML und kann deshalb besser von Computer-Programmen weiterverarbeitet werden kann.
- CDA (Clinical Dokument Architecture); [46] Ein standardisiertes Format für den elektronische Repräsentation, Speicherung und Austausch von medizinischen Dokumenten
- Mathematical Markup Language [35]; XML Applikation zum Einbetten von Gleichungen in Webseiten und andere Dokumente
- Scalable Vector Graphics (SVG); XML- Kodierung von Linienzeichnungen [36]

2.1.4 Weiterverarbeitung von XML

Für die maschinelle Weiterverarbeitung von XML Dokumenten stehen zwei verschiedene Konzepte mit spezifischen Vor- und Nachteilen zur Verfügung. Das Document Object Model (DOM) und Simple „API“(Application Programmers Interface) für XML (SAX).

2.1.4.1 Document Object Model (DOM)

Die Entwicklung des Document Object Modell (DOM) reicht in die Zeit vor Einführung des XML Standards (XML 1.0, 1998) zurück. DOM wurde entworfen, um ganz allgemein den Aufbau und den Inhalt von Dokumenten darzustellen- unabhängig vom Dokumenttyp. [37] DOM sieht ein Document als ein Objekt, welches andere Objekte enthält. Beim „Lesen“ eines XML Dokuments legt ein DOM- Parser Schritt für Schritt ein Ebenbild der XML Daten im Arbeitsspeicher in Form einer Baumstruktur an. Um auf die geparsten Daten zugreifen zu können, muss ein XML Dokument zunächst wohlgeformt sein, ansonsten bricht ein Parser seine Arbeit ab und gibt eine Fehlermeldung aus. Weiter muss das gesamte Dokument ausgelesen werden und als Baumstruktur vollständig im Speicher repräsentiert sein, bevor die Daten verarbeitet werden können.

Solange das Programm läuft kann auf die Daten zugegriffen werden- die Baumstruktur kann direkt im Speicher manipuliert werden. Der Zugriff erfolgt über eine Knotensymptomatik und mittels verschiedener Zugriffsfunktionen. Ein Nachteil von DOM ist die hohe Speicherbelastung, wenn sehr umfangreiche XML Dokumenten (>5MB) geparst werden sollen.

2.1.4.2 Simple API for XML (SAX)

Ein anderes Konzept verfolgen SAX Parser. Im Gegensatz zur „baumorientierten“ Sichtweise beim Document Object Modell geben SAX Parser die Daten eines XML- Dokuments während des Parsens aus. In dem Augenblick, in dem ein SAX Parser eine Struktur (zum Beispiel eine Element) in einem XML Dokument als solche identifiziert hat gibt er sie auch schon an das Anwendungsprogramm, welches die Schnittstellen benützt, weiter. SAX bietet die Möglich XML Code einfach, schnell und flexibel zu parsen. [38]

2.1.5 Extensible Stylesheet Language (XSL)

Die extensible Stylesheet Language XSL besteht aus zwei Teilen: Extensible Stylesheet Language Transformations (XSLT) einerseits und Extensible Stylesheet Language Formating Objects (XSL-FO) andererseits. Beide Teile benutzen die so genannte XPath Syntax.

2.1.5.1 XPath

XPath ist eine [39] „Nicht- XML Syntax“, welche eingesetzt wird, um bestimmte Teile von XML Dokumenten zu adressieren. Ein XML Dokument bildet eine Baumstruktur, welche aus Knoten besteht. Mit XPath Ausdrücken können Knotenmengen oder jedes einzelne beliebige Element in einem XML Dokument adressiert werden. XPath identifiziert Knotenmengen oder einzelne Knoten entsprechend der Position, der relativen Position, des Inhaltes und verschiedener anderer Kriterien.

2.1.5.2 XSL Transformations (XSLT)

XSLT ist eine XML Anwendung, welche Regeln formuliert, um ein XML Dokument in ein anderes XML Dokument zu überführen. [40] Außerdem bietet XSLT die Möglichkeit ein XML Dokument in ein anderes Format umzuwandeln, wie etwa in einfachen Text oder HTML. In Kapitel 3 und 4 wird anhand eines konkreten Beispiels näher auf XSLT eingegangen.

2.1.5.3 XSL Formating Objects (XSL-FO)

Ist ein Werkzeug, mit dem detailliert Layout Struktur und Style für die Ausgabe und die Vorbereitung auf den Druck eines XML Dokumentes geregelt werden kann. XSL-FO ist auch geeignet für die Umwandlung von XML in ein binäres Format (z. Bsp. Portable Dokument Format. (*.pdf)).

2.1.6 XML als Speicherformat

XML ist ein reines Textformat. Jedes Werkzeug, welches in der Lage ist Textdateien zu lesen, kann auch XML Dokumente lesen. XML eignet sich daher besonders als Austauschformat zwischen verschiedenen Anwendungen.

Dennoch darf nicht übersehen werden, dass XML nicht immer die beste Lösung für die Repräsentation von Daten ist.

2.1.6.1 XML und Datenbanken

Im folgenden Abschnitt soll auf einem sehr allgemeinen Niveau versucht werden die Verwendung von XML mit Datenbanken zu skizzieren. [41]

Die XML Technologie beinhaltet eine Reihe von Funktionalitäten, welche typischerweise auch in Datenbanksystemen gefunden werden. Datenspeicherung (als XML Dokument), Schemen (DTDs, XML Schema, RELAX NG und andere), Abfrage Sprache (XQuery, XPath, XQL, XML-QL, Quilt), Programmschnittstellen (SAX; DOM) und andere. Es soll hier nicht weiter auf die erwähnten Technologien eingegangen werden. Andererseits fehlen einige sehr wesentliche Eigenschaften echter Datenbanksysteme. Effiziente Speichermechanismen, Indices, Sicherheitskonzepte, Transaktions- und Datenintegrität, gleichzeitiger Zugriff durch mehrere Benutzer und so weiter. XML kann also nicht als Datenbank im eigentlichen Sinn bezeichnet werden.

Es kann unterschieden werden zwischen XML Dokumenten, welche eher gut strukturierte fein granuliert Daten transportieren bzw. beinhalten und solche XML Dokumente, welche eher zusammenhängenden Text enthalten zum Beispiel ganze Dokumente oder Fragmente davon. („Datenorientiertes XML“ versus „Dokumentenorientiertes XML“).

Beispiele für datenzentriertes XML sind: Flugpläne, Bestellungen, Lagerbestände wissenschaftliche Daten. Beispiele für dokumentenzentriertes XML sind: Bücher, Werbeanzeigen, E-Mails. Dokumentenzentriertes XML ist also meist für „menschlichen Konsum“ konzipiert. Nicht immer kann so klar klassifiziert werden oft liegt eine Mischung beider Formen in einer XML Datei vor.

Natürlich immer davon abhängig für welche Zwecke Daten weiterverarbeitet werden kann generell Folgendes gesagt werden. Datenzentriertes XML ist gut geeignet als Austauschformat für die transportierten Daten zum Beispiel zwischen einer Anwendung und einer relationalen Datenbank und schlecht geeignet als Speicherformat für die Daten. Für dokumentenzentrierten Inhalt ist XML an sich aus mehreren Gründen (Textformat, semantische Auszeichnung, Trennung Inhalt Layout usw.) ein sehr vorteilhaftes Format, sowohl zum Dokumentenaustausch als auch zur dauerhaften Speicherung.

Native XML Datenbanken sind Datenbanksysteme, deren fundamentale logische Speichereinheit ein vollständiges XML Dokument ist. (in Analogie dazu würde das einem Tupel in einem relationalen Datenbanksystem entsprechen). Die Bezeichnung „nativ“ ist eher eine Marketing Bezeichnung, es gibt dazu keine formale technische Definition. Native XML Datenbanken beinhalten Funktionalitäten und Konzepte wie Query Languages, Updates und Deletes, Transactions, Application Programming Interfaces und weitere.

2.2 Clinical Dokument Architecture

2.2.1 Einführung in CDA

Die Clinical Dokument Architecture ist ein standardisiertes Format zum elektronischen Austausch klinischer Dokumente. CDA ist in XML implementiert. CDA ist Teil der HL7 Version 3 Familie. Alle Teile der HL7 Version 3 Familie basieren auf einem gemeinsamen zugrunde liegenden Informationsmodell- dem „Reference Information Modell“ (RIM). Das RIM beschreibt alle Objekte, Aktivitäten und deren Relationen in einem generischen Gesundheitsmodell. Zu Beginn der Bemühungen einen gemeinsamen Standard für elektronische klinische Dokumente zu schaffen. Ziel der CDA ist es einen international gültigen und im klinischen Alltag einsetzbaren Standard für die strukturellen Aufbau, Inhalt und den elektronischen Austausch von medizinischen Dokumenten zu schaffen. Literatur zur Einführung in das Thema findet sich unter <http://www.hl7.de/cda2002/>. [42]

Ein Auszug aus der Zieldefinition und den Designanforderungen an den Standard wird wiedergegeben:

- Ziel der CDA ist es die Qualität in der Gesundheitsversorgung zu steigern
- CDA soll eine kostengünstige Implementierung für ein möglichst breites Spektrum verschiedener Systeme ermöglichen
- CDA soll für das menschliche Auge lesbar sein (also kein binäres Format) und zwar auch für Personen ohne wesentliche technische Vorbildung
- Soll der Langlebigkeit (gesetzliche Bestimmungen..) der darin transportierten Information Rechnung tragen
- CDA Dokumente bzw. die darin enthaltenen Daten sollen von einer Vielzahl verschiedener Systeme (Hard und Software) einfach weiterverarbeitet werden können.
- CDA soll Implementierung lokaler Gegebenheiten (gesetzliche Vorschriften etc.) ermöglichen ohne den Standard dafür erweitern zu müssen
- Die technische Barrieren für den Einstieg in den Standard sollen niedrig gehalten werden
- Befriedigung der Ansprüche aller an der Patientenversorgung beteiligten Personen und Institutionen (Softwarehersteller, Versicherungen, Gesetzgeber u. a.)
- CDA soll in der extensibel Stylesheet Language (XML) Implementiert werden.
- Abgeleitet vom „Reference Information Modell“ von „Health Level Seven“ (HL7)

Der erste Name für die Bemühungen einen Standard für klinische Dokumente zu schaffen war „Kona Architecture“ gefolgt von „Patient Record Architecture“ (1996). Im Jahr 2000 wurde die CDA Release 1.0 als erster in XML implementierter Standard im Gesundheitswesen von der ANSI akkreditiert. Mittlerweile (Mai 2005) wurde Release 2.0 zum ANSI Standard. [57]

2.2.2 Dokumentenparadigma

Was ist unter einem klinischen Dokument zu verstehen? Ein klinisches Dokument beinhaltet Beobachtungen und Maßnahmen und hat folgende Eigenschaften:

- Persistenz: Ein klinisches Dokument existiert in unveränderter Form, für einen bestimmten Zeitraum
- Geregelter Verwaltung: Ein klinisches Dokument wird von einer Person oder Organisation verwaltet, welche mit seiner Pflege betraut ist.
- Möglichkeit zur Authentifizierung: Ein klinisches Dokument kann im Rahmen gesetzlicher Bestimmungen authentifiziert werden.
- Ganzheit der Authentifizierung: Die Authentifizierung gilt für das gesamte Dokument und nicht nur für Teile davon
- Lesbarkeit fürs menschliche Auge: Durch Verwendung allgemein verfügbarer XML Browser bzw. Druckertreiber und durch die Anwendung von XSL- Stylesheets

Ein CDA Dokument ist ein definiertes und komplettes Informationsobjekt, welches Texte, Bilder, Klänge und andere multimediale Objekte enthalten kann.

Item	Dokument	Nachricht
Lebensdauer	persistent	temporär
Kommunikation	Mensch- Mensch	Maschine- Maschine
Beziehung zu Arzt	Dokument ist vertraut	Nachrichten nicht vertraut
Gesetzliche Aspekte	hat Bestand vor Gesetz	Unterschrift? Akzeptanz?
Quelle	bekannt Methode	erzeugt im Anwendungsfall

Tabelle 1: Vergleich Dokument- Nachricht

2.2.3 Architektur und Levels

Unter dem Begriff Architektur in CDA ist in etwa folgendes zu verstehen: die vollständige Spezifikation des CDA Standards wird eine hierarchische Menge verschiedener Typen klinischer Dokumente beinhalten repräsentiert durch eine hierarchische Menge von XML Schemadateien. Typen klinischer Dokument sind beispielsweise: Untersuchungsbefunde aus verschiedenen medizinischen Bereichen etwa Ultraschalluntersuchungen des Herzens oder Ultraschallkontrolluntersuchungen während der Schwangerschaft, Entlassungsbriefe oder Kurzarztbriefe zum Transfer eines Patienten in eine andere Abteilung usw.

Die Spezifikation verschiedener Dokumenttypen erfolgt durch intensive Analyse bestehender medizinischer Dokumente durch HL7. Alle Dokumenttypen werden auf einem gemeinsamen Informationsmodell dem Referenz Information Modell (RIM) beruhen. [45]

In der aktuellen Version des Standards ist bis jetzt nur der oberste Knoten der Hierarchie definiert, genannt CDA Level One (Level 1). CDA Levels stellen einen Migrationspfad für Entwickler dar, um iterativ weiteres, feiner granuliertes Markup für klinische Information einzuführen. [46]

Level 1 spezifiziert lediglich den Header Teil eines CDA Dokuments detailliert. Der Body Teil, in dem die klinische Information untergebracht wird, ist in CDA Level 1 lediglich allgemein strukturell spezifiziert. Level 1 soll die technischen Barrieren zum Einstieg in den Standard niederhalten und eine sanfte Einführung in das RIM bieten.

Level 2 ist geplant als ein Set von „Templates“ oder Regeln, welche praktisch wie ein Layer auf Level 1 aufgesetzt werden können. Ein solches Template kann beispielsweise bedeuten, dass ein Dokumenttyp Entlassungsbrief aus einer kardiologischen Abteilung einen obligaten Abschnitt „Untersuchungsbefunde“ enthält, welcher wiederum die Abschnitte „EKG Befund“ und „Thoraxröntgenbefund“ enthält und so weiter. Dieses geplante Vorgehen setzt natürlich detailliertes Wissen in medizinischen Domänen voraus, welches durch Einbeziehen entsprechender Experten und Gesellschaften durch HL7 erreicht wird.

Level 3 wird zusätzliches vom RIM abgeleitetes Markup hinzufügen, um klinischen Inhalt formaler auszudrücken, als dies im RIM ausgedrückt ist, mit der Intention die maschinelle Verarbeitung von CDA Dokumenten noch zu erweitern. So könnten dann zum Beispiel Codes für bestimmte dokumentierte Symptome oder Befunde zur weiteren Verarbeitung aus CDA Dokumenten extrahiert werden oder Daten für die Abrechnung automatisch aus einem CDA Dokument extrahiert werden.

Klinischer Inhalt von CDA Dokumenten bleibt über alle Level hinweg beständig und unverändert. Klinischer Inhalt wird in höheren CDA Level feiner granuliert mit Auszeichnungen versehen und ist somit in einem höheren Grad auch maschinell verarbeitbar.

2.2.4 Aufbau eines CDA Dokumentes (CDA Level 1 Release 1)

Ein CDA Dokument besteht aus einem Header und einem Body. Der CDA Header identifiziert, klassifiziert und authentifiziert das Dokument. Der CDA Header beinhaltet Informationen zum Ereignis (zum Beispiel das Zusammentreffen von Patient und Arzt für eine Untersuchung), Informationen über die Erbringer und Empfänger einer Maßnahme. CDA Body enthält die eigentliche klinische Information.

Im Folgenden wird etwas genauer auf den Aufbau eines CDA Level 1 Dokumentes eingegangen. [46] wie der vorhergehende...]Einen ganz detaillierten Einblick über den vorgeschriebenen Dokumentenaufbau kann man durch Studium des entsprechenden XML-Schemas für die Anwendung CDA Level 1 erhalten.

2.2.4.1 CDA Level 1 Header

CDA Level 1 Header besteht aus vier logischen Komponenten: 1) Dokumenteninformation, 2) Erbringer einer Maßnahme, 3) Empfänger einer Maßnahme, 4) Informationen zum Ereignis. [46]

Im Weiteren werden die vier logische Komponenten und einige Header Elemente gegliedert nach deren Zugehörigkeit zu einer der vier logischen Komponenten vorgestellt. In den folgenden Abschnitten wird immer von Release 1 ausgegangen, obwohl Release 2 der aktuelle Standard ist. (ANSI/ HL7 Da Level1 Release 2, Mai 2005). In der Periode der Programmierung für den praktischen Teil dieser Arbeit war noch Release 1 der aktuelle Standard (2000- 2005), sodass für diese Arbeit Release 1 als Vorgabe benutzt wurde.

Dokumenteninformation

Versionshandling, Bezug zu anderen Dokumenten, Dokumenten Typ Code.

- `<id>`, `<set_id>`, `<version_nbr>`, `<document_relationship>`

Ein CDA Dokument kann ein originales Dokument sein, ein Anhang zu einem existierenden Dokument oder eine Revision eines existierenden Dokumentes ein. Ein angehängtes Dokument („addendum“) ist ein Anhang an ein existierendes Dokument und enthält zusätzliche Informationen. Status und Zustand eines angehängten Dokumentes bleiben unverändert. Die Revision eines Dokumentes („replacement report“) ersetzt das ursprüngliche Dokument und erhält einen neuen global eindeutigen Identifikator („`<id>`“)- es hat aber denselben Wert für „`<set_id>`“ als das Vorgängerdokument und inkrementiert den Wert des „`<version_nbr>`“ Elementes.

- `<fulfills_order>`, `<document_type_cd>`

Dokumente können in Beziehung stehen zu einer oder mehreren Anordnungen („Order“). Das Element „`<fulfills_order >`“ zeigt die Beziehung zwischen dem gegenwärtigen Dokument und dem eindeutigen Identifikator der Anordnung, welche durchgeführt wird.

Jedes CDA Dokument hat einen eindeutigen Code für seine Dokumenttyp, spezifiziert durch das Element „<document_type_cd>“. Die Codes für dieses Element (und auch einige andere Elemente für CDA) stammen aus der LOINC- Datenbank. (LOINC: Logical Observation Identifiers, Names and Codes). Die in der LOINC Datenbank gespeicherten Namen und Codes sollen Austausch und das Zusammenführen von Untersuchungsergebnissen ermöglichen. Die LOINC Datenbank besteht aus einem Teil für Laboruntersuchungen und einem Teil für klinische Untersuchungen

- <confidentiality_cd>

Für das Ganze CDA Dokument oder Teile davon kann durch Einsatz des Elementes „<confidentiality_cd >“ein Vertraulichkeitsstatus festgesetzt werden. (Möglichkeit des Einbeziehens von Datenschutzbestimmungen).

Erbringer einer Maßnahme

An der dokumentierten Maßnahme beteiligten Personen (Beispiel: Untersucher oder Arzt, welcher eine Kopie des Befundes erhält usw.) und Modalitäten (also auch Maschinen und physikalische Phänomene). Alle an der Erbringung einer Maßnahme beteiligten Personen, welche eigenständig Entscheidungen treffen und dafür auch verantwortlich gemacht werden können.

- <authenticator>, <legal_authenticator>

Ein CDA Dokument kann nicht authentifiziert sein, authentifiziert und gesetzlich authentifiziert sein. Nicht authentifiziert bedeutet die Abwesenheit der Elemente „<authenticator>“, „<legal_authenticator>“ im CDA Header. Authentifiziert ist ein CD Dokument, wenn ein oder mehrere Serviceerbringer die Richtigkeit es Dokumentes bestätigten. Das Element <legally_authenticator> beinhalten ein Person, die rechtmäßig für die bestätigte Richtigkeit des CDA Dokumentes verantwortlich ist. Die elektronische Signatur eines CDA Dokumentes ist derzeit nicht teil des Standards, dafür gibt es das Element <signature_cd> mit einem eindeutigen Code als Wert.

- `<originator>`, `< origination_organization >`,
`<originating_device>`, `<provider>`,

Ein CDA Dokument kann von einem Menschen, einer Organisation oder maschinell generiert werden. Die wird durch die Elemente `<originator>`, `< origination_organization >` und `<originating_device>` ausgedrückt. Mindestens eine Person als Erbringer einer Maßnahme muss in einem CDA Dokument im Element `<provider>` genannt werden.

- `<intended_recipient>`

Die Person, welche eine Kopie des CDA Dokumentes zur weiteren Patientenbehandlung erhält.

Empfänger einer Maßnahme

Empfänger einer Maßnahme sind zu Beispiel: Patient, Familienangehörige des Patienten, beteiligte Modalitäten

- `<patient>`, `<birth_dttm>`, `<administrative_gender_cd>`,
`<service_target>`

Das Element `<patient>` hat dieselben Kindelemente wie andere Personen innerhalb eines CDA Dokumentes. Zusätzlich hat es noch die Kindelemente `<birth_dttm>` und ein Kindelement `<administrative_gender_cd>`. Weitere Empfänger einer Maßnahme sind Familienangehörige des Patienten. („`<service_target>`“)

Informationen zum Ereignis

Beschreibt Umstände, in denen das dokumentierte Zusammentreffen stattfindet

- `<encounter_tmr>`, `<practice_setting_cd>`

Zeitpunkt des Zusammentreffens (Untersuchungsdatum), Kategorisierung der Leistung erbringenden Institution (zum Beispiel: Abteilung für Innere Medizin, kardiologische Abteilung, Allgemeinchirurgie etc.)

2.2.4.2 CDA Level 1 Body

CDA Level 1 Body besteht entweder aus einem oder mehreren „`<section>`“ Komponenten oder einem „`<non-xml>`“ Element für Body Inhalt, welcher in einem anderen Format als XML vorliegt.

Das CDA Body Element `<section>` kann wiederum weitere `<section>` Elemente, „Strukturen“ und so genannte „`<coded_entry>`“ Elemente beinhalten.

CDA Strukturen bestehen aus den Elementen „`<paragraph>`“, „`<list>`“, „`<table>`“. Diese Strukturen wiederum beinhalten CDA „entries“. CDA Entries sind die CDA Body Elemente „`<content>`“, „`<link>`“, „`<coded_entry>`“, „`<observation_media>`“, „`<local_markup>`“ oder CDA Entries beinhalten „CDATA“.

Mit dem `<non-xml>` Element wird ein Dokument Body referenziert, welcher nicht XML Format ist. Die Datenspeicherung dieses Nicht XML Dokument Body ,s erfolgt außerhalb des CDA Dokumentes.

Document Strukturen

CDA Strukturen bestehen aus den Elementen sections, paragraphs, lists, und tables.

- `<paragraph>`: kann in Elementen section, item und in Tabellenzellen vorkommen. Hat optionales caption Element gefolgt von einem oder keinem content Element.
- `<list>`: kann in Elementen section, item und in Tabellenzellen vorkommen. Hat optionales caption Element gefolgt von einem oder keinem content Element. Das Attribut `list_type` spezifiziert die Liste als geordnet oder ungeordnete Liste.
- `<table>`: tritt auf in Elementen section oder item. Dient ausschließlich Präsentationszwecken und entrichtet dem strikten XHTML Tabellen Modell.
- `<caption>`: dient als Beschriftung für einen Container. Kann in den Elementen section, paragraph, list, item und table auftreten. Caption kann Text oder Links beinhalten oder auch codiert werden. („`<caption_cd>`“)

Optionale CDA Structure Attribute sind: „`confidentiality`“, „`origination`“ und „`human language`“. Diese Attribute werden im CDA Header deklariert und vom Body aus referenziert. Das Sprachattribut („`xml:lang`“) wird verwendet, um auf die in CDATA Abschnitten verwendete Sprache hinzuweisen.

Document Einträge

Document Einträge kommen in CDA Strukturen vor und beinhalten CDATA. (CDATA: Daten, welche kein Markup enthalten). Dokument Einträge bestehen aus den Elementen `coded_entry`, `content`, `link`, `observation media`.

- `<coded_entry>`: Hier wird von HL7 erkannter oder lokal verwendeter Code verwendet. Kann in den Elementen Section oder Content vorkommen.
- `<content>`: kann in Elementen `local_markup` Tabellenzellen, `item`, `paragraph` und verzweigt in weiteren content Elementen vorkommen. Enthält keine oder mehrere Einträge.
- CDATA: kann vorkommen in Elementen `content`, `local_markup`, `caption`, `link_html` oder Tabellenzellen.
- `<link>`: generischer Referenz Mechanismus, welcher verwendet wird, um Inhalte zu referenzieren, die nicht integraler Bestandteil des CDA Dokumentes sind. Beinhaltet obligates Element `link_html`.
- `<observation_media>`: Inhalt, welcher logischer Teil eines CDA Dokumentes ist, aber außerhalb des CDA Dokumentes gespeichert wird und über Referenz ins Dokument eingebracht wird. Muss verwendet werden, wenn der referenzierte Inhalt integraler Bestandteil des CDA Dokumentes ist. Kann erscheinen in den Elementen `content`, `local_markup` und in Tabellenzellen.

2.2.4.3 CDA Lokalisierung

Ein zentrales Problem beim Austausch von Informationen ist das Spannungsfeld zwischen lokaler Spezialisierung und globaler Generalisierung. Unter Lokalisierung versteht man im Folgenden Bemühungen bzw. Möglichkeiten lokale Begebenheiten und Bedürfnisse in einen globalen Standard zu integrieren.

Es gibt mehrere Möglichkeiten zur Lokalisierung:

- Der CDA Standard wird verwendet, lokale Bedürfnisse werden mit Hilfe des CDA Standard konformen „Lokalem Markup“ („`<local_header>`“ und „`<local_markup>`“) erfüllt.
- In den Anwendungen implementieren Entwickler den lokalen Standard. Für den Dokumentenaustausch wird das Dokument des lokalen Standards in ein CDA Instanzdokument transformiert.

In HL7 V2.x Nachrichten können am Ende dieser Nachrichten neue Segmente und Felder für lokale Bedürfnisse hinzugefügt werden. In Analogie dazu verfügt die CDA Spezifikation über die Elemente „`<local_header>`“ und „`<local_markup>`“. Das Attribut „`descriptor`“ kann zur näheren Beschreibung dieser Elemente verwendet werden.

Ein Code als Attributwert kann von einer lokalen Vokabeldomäne herangezogen werden. Das Attribut „ignore“ teilt einem Empfänger mit das lokale Element zu ignorieren (ignore = „markup“) oder das Element und seinen gesamten Inhalt (ignore = „all“) zu ignorieren.

Ein Beispiel für die Verwendung von lokalen Markup ist das deutsche „SCIPHOX“- Project („Standardized Communication of Information Systems in Physicans Offices and Hospitals using XML“). [19], [54]. Für die elektronische Abbildung von klinischer Information werden im Sciphox Projekt so genannte „Small Semantic Units“ Units (SSU) verwendet. [24]. Durch ein Set von Definitionen kann durch Verwendung von SSU klinische Informationen- mit dem Ziel die maschinelle Verarbeitung in einem höheren Maße zu unterstützen- in Einheiten feinerer Granularität gegliedert werden, als das die CDA Level 1 Standardspezifikation vorsieht. Außerdem wird dadurch auch die Implementierung nationaler Besonderheiten (z. Bsp. Versicherunwesen) erleichtert.

2.2.4.4 Object Identifiers

CDA verwendet so genannte „Object Identifiers“ (OID). OID sind weltweit eindeutige Kennzeichnungen für Objekte und sind von der „International Organisation for Standardization“ (ISO/IEC) normiert. [47]. Objekte sind persistente, wohldefinierte Informationen, Definitionen oder Spezifikationen und werden als Identifikatoren und Kodierungen wiedergegeben. [48].

Wichtig ist zwischen Identifikationen (IDs) und Kodierungen zu unterscheiden. Eine ID deutet auf eine Instanz eines Objektes hin, z. Bsp. eine bestimmte Person (Patient, Arzt), eine konkrete Laboruntersuchung oder ein Röntgenbild. Eine Kodierung deutet hingegen ein Konzept an: Typ des Arztes (z. Bsp. Anästhesist), Typ der Laboruntersuchung (z. Bsp. Leberfunktionsparameter),

Nachrichten und Dokumente des HL7 Version nutzen OID, um Kodierungs- Schemas und Identifikationsbereiche eindeutig zu bezeichnen. Dabei wird die Idee verfolgt, dass jede Identifikation bzw. jedes Kodierschema Teil des Systems ist, in dem sie bzw. es definiert wurde. Beispiele sind Patientennummern, die innerhalb eines Krankenhauses ausgegeben werden, Arztidentifikationsnummern der Ärztekammern, Laboratoriums- Codes für Untersuchungen als LOINC Codes.

Die Kombination aus der eigentlichen Identifikation („Extension“) und der ausgebenden Instanz („Root- OID“) ist dabei zusammen genommen weltweit eindeutig. Abbildung 2 zeigt eine XML- Repräsentation für die Diagnose Appendicitis als ICD- 10 Code (International Classification of Deseases). [49]

```
<value code = "I59.13" codeSystem="2.16.840.1.113883.6.3"/>
```

Abbildung 2: XML Repräsentation eines Object Identifier

OID können von jeder Organisation ausgegeben werden, indem eine eindeutige Wurzel- OID verwendet wird. Einmal zugewiesen, wird ein OID niemals zurückgenommen und bleibt ein gültiger Bezeichner für dasselbe Schema oder Objekt.

The image shows an XML document with several annotations pointing to specific elements:

- Document Information:** Points to the root `<levelone>` element.
- Service Actor:** Points to the `<person>` element within the `<provider>` block.
- Object Identifier:** Points to the `<id EX="123456" RT="2.16.840.1.113883.3.7.2.12345.1.2"/>` element within the `<person>` block.
- Service Target:** Points to the `<person>` element within the `<patient>` block.

A text box on the right side of the XML provides a description of the CDA Level 1 Body structure:

CDA Level 1 Body: lediglich Strukturierung des Inhaltes! „Sections“ (Container), „Structures“ (paragraph, list, item, table) und Entries (content, link, #PCDATA, coded entries.)

Abbildung 3: CDA Level 1 Instanzdokument

2.3 Das Krankheitsbild der Varikositäs

2.3.1 Definition und Epidemiologie

Varizen sind sackförmig oder zylindrisch erweiterte oberflächliche Venen der unteren Extremität. [56] Die Venenerweiterung ist umschrieben oder sackförmig und geht zumindest mit einer Schlängelung und Knäuelbildung einher. (Definition der Weltgesundheitsorganisation WHO). Synonyme Begriffe: Krampfadern (altdeutsch: Krummadern), Varikositäs

Zirka 20% aller Erwachsenen in Mitteleuropa sind von dem Leiden betroffen. Die Prävalenz (Anzahl der erkrankten Individuen in einer Population) nimmt mit dem Alter zu. Frauen sind dreimal häufiger betroffen als Männer. Die Erstmanifestation Erkrankung ist meist im 3. Lebensjahrzehnt.

Einteilung in primäre Varikositäs (95%) und sekundäre Varikositäs (5%). Die sekundäre Varikositäs ist Folge einer Thrombose im tiefen Venensystem der unteren Extremitäten. Die primäre Varikositäs hat keine sicher bekannten Ursachen. Der ICD Code (International Classification of Diseases) für die Varikositäs lautet: I83.9



Abbildung 4: Varizen am linken Unterschenkel

2.3.2 Anatomie

Drei Venensysteme werden an den Beinen unterschieden: Ein oberflächliches und ein tiefes System. So genannte Perforansvenen stellen eine Verbindung zwischen oberflächlichen und tiefen System her.

Durch das tiefe Venensystem fließt 90% des venösen Blutes aus den Beinen zurück zum Herz. Muskel- und Gelenkpumpe sorgen zusammen mit Venenklappen für den gerichteten Blutfluss.

Das oberflächliche System besteht aus der Vena saphena magna (vom Innenknöchel bis zur Leiste), der Vena saphena parva (Fußrücken bis Kniekehle) und Seitenästen. Auch hier verhindern Venenklappen einen pathologischen Blutrückfluss

Die physiologische Flussrichtung in den Perforansvenen ist von außen nach innen und wird von Venenklappen sichergestellt. Drei wichtige Gruppen von Perforansvenen werden unterschieden: Dodd Gruppe (Innenseite mittlerer Oberschenkel), Boyd Gruppe (Innenseite Unterschenkel direkt unterhalb des Knies) und Cockett (Innenseite US im unteren Drittel) Gruppe.

2.3.3 Pathophysiologie (primäre Varikose) und Ätiologie

Schlussunfähigkeit der Venenklappen der oberflächlichen Venen, sodass es zur Strömungsumkehr des Blutes in zentrifugaler Richtung kommt. Am proximalen Insuffizienzpunkt fließt das Blut nicht in die tiefe Beinvene, sondern retrograd bis zum distalen Insuffizienzpunkt. Von dort je nach Lokalisierung des distalen Insuffizienzpunktes in Seitenäste oder über Perforansvenen wieder in das tiefe System. (Rezirkulationskreis)

Die Genese ist multifaktoriell. Genetische Faktoren (in bis zu 50% positive Familienanamnese), Alter, hormoneller Einfluss bei Frauen (zum Beispiel Schwangerschaft), stehende oder sitzende Tätigkeit sind wichtige Faktoren.

2.3.4 Klinik, Stadieneinteilung

Typische Beschwerden sind Müdigkeits-, Schwere-, Spannungsgefühl in den Beinen. (Besserung im Liegen und bei Bewegung). Neigung zu abendlichen Knöchelödemen, eventuell Juckreiz, und Druckgefühl, nächtliche Fuß-, und Wadenkrämpfe. Typischerweise nehmen die Beschwerden venöser Erkrankungen gegen Abend, nach langem Sitzen, langem Stehen oder bei warmem Wetter zu; nicht jedoch nach längerem Gehen (wie zum Beispiel bei der peripheren arteriellen Verschlusskrankheit).

Die Stadieneinteilung der Stammvarikosis der Vena saphena magna (VSM) nach Hach erfolgt nach der Lokalisation des distalen Insuffizienzpunkte (Stadium I bis IV). Die Stammvarikosis der Vena saphena parva (VSP) ist seltener. Es bilden sich hierbei Krampfaderkonvolute an der Dorsalseite der Wade aus.

Die Stadieneinteilung nach Marshal erfolgt nach klinischen Gesichtspunkten (Keine Beschwerden, allenfalls kosmetisch störend- Stadium I, bis Ulcus venosum cruris Stadium IV).

2.3.5 Diagnostik, Therapie, Prognose

Anamnese, Inspektion, Palpation und Venenfunktions-tests werden durch die hohe Aussagekraft von der Duplexsonographie ergänzt und bilden Grundzüge der Diagnostik.

Drei Therapieformen kommen je nach Befund und Stadium der Erkrankung einzeln oder in Kombination zum Einsatz. Konservative Therapie, operative Therapie und Sklerotherapie

Eine konservative Therapie besteht im Tragen von Kompressionsstrümpfen. Laufen und Liegen sind günstig- Stehen und Sitzen ungünstig. Es gibt verschieden operative Methoden. Voraussetzung für alle operativen Methoden ist ein intaktes tiefes Beinvenensystem. Besenreisvarizen (spinnengewebartiges Netz kleinster, intradermaler Varizen), reticuläre Varizen (netzartige Venektasien von wenigen Millimeter Durchmesser) und kleine Seitenastvarizen können, wenn sie kosmetisch störend sind, durch Sklerosierung oder Lasertherapie verschlossen werden.

3 Methoden

In diesem Kapitel werden kurz die im Kapitel 4 Durchführung für den praktischen Teil dieser Arbeit verwendeten Methoden beschrieben

3.1 XML Schema

In diesem Abschnitt wird anhand des XML Schemas CDA Level 1 Release 1 versucht einen Einblick in die XML Schemasprache XML Schema zu geben. Ein grundlegender Einblick in XML Schema ist für das allgemeine Verständnis von CDA und im Besonderen für die praktische Umsetzung der Webapplikation nötig, um bestimmen zu können, welche Merkmale für das geplante Dokumentationssystem erfasst werden müssen.

Der grundsätzliche Sinn und Zweck eines XML Schemas und die Vorteile der XML Schemasprache XML Schema einer DTD gegenüber wurden bereits in Kapitel 2 Grundlagen Abschnitt 2.1 beschrieben. Im Folgenden wird eingegangen auf den Aufbau eines XML Schemadokumentes und den Umgang mit Namensräumen. Dann wird beschrieben was unter Schema Validierung zu verstehen ist. Es wird gezeigt, wie Elemente, komplexe Datentypen und Attribute definiert werden und wie Instanzdokument und Schema verbunden werden. Es werden zur Veranschaulichung dazu jeweils Ausschnitte aus dem XML Schema für CDA Level 1 Release 1 verwendet.

3.1.1 Aufbau und Namensräume

Jede XML Schema Datei ist ebenfalls ein XML Dokument und unterliegt damit den allgemeinen Vorschriften zur Wohlgeformtheit. Ein XML Schema Dokument hat die bekannte Struktur aus ineinander verschachtelten Elementen und Attributen und kann als Baumstruktur dargestellt werden. Eine XML Schemadatei endet auf `.xml` oder `.xsd`. Jede XML Schema Datei beginnt mit der XML Processing- Instruction, welche Aufschluss auf Typ und Version des Dokumentes gibt. Das Wurzelement lautet `schema`.

Allen zu XML Schema gehörenden Elementen wird das Präfix `“xsd”` voran gestellt, welches seinerseits im Attribut `xmlns` (für XML Namespace) dem Namensraum `“http://www.w3.org/2001/XMLSchema”` zugewiesen wird. Alle XML Schema zugehörigen Elemente dienen ihrerseits aber wieder dazu, weitere Elemente und Attribute zu definieren, welche in einem eigenem Namensraum liegen. Um diese Unterscheidung zu ermöglichen, wird ein weiterer Namensraum angelegt, der den Namensraum aller anderer Elemente und Attribute definiert. (für die Anwendung CDA im Namensraum: `“urn:h17-org/cda”`). Dies ist gleichzeitig ein Beispiel für den konfliktfreien Einsatz zweier verschiedener Namensräume innerhalb einer XML Datei. Alle Elemente, welchen nicht explizit die Vorsilbe `xsd` zugewiesen wird gehören nun diesem zweiten Namensraum an.

(„Default Namensraum“). Mit dem Attribut `targetNamespace` wird ausgedrückt, welchem Namensraum das Zieldokument (Instanzdokument) angehört.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="urn:hl7-org/cda"
            targetNamespace="urn:hl7-org/cda">
<!--Inhalt der Schema Datei-->
</xsd:schema>
```

Codelisting 1: Namensräume in XML Schema

3.1.2 Schemavalidierung

Ein XML Schema Validator ist nichts anderes als ein XML Parser, welcher speziell für die Analyse von XML Schema Dokumenten entwickelt wurde. Ein XML Schema Validator vergleicht also, ob die im Schema Dokument angegebenen Vorschriften im Instanzdokument eingehalten werden. Wenn das der Fall ist spricht man von einem „Schema-gültigen Dokument“. Zur Validierung eines Instanzdokumentes gegen sein Schema wird für diese Arbeit die Open Source Software Xerces-J verwendet. Die Installation und Konfiguration der Software wird in Kapitel 4 Durchführung Abschnitt 4.5.5 beschrieben.

Das Programm wird von der Kommandozeile aus aufgerufen und das zu validierende Instanzdokument als Parameter übergeben. Ist das Instanzdokument Schema-ungültig so wird eine detaillierte Fehlermeldung ausgegeben. Diese weist auf Art des Fehlers und auf den Ort des Auftretens des Fehlers im Instanzdokument hin.

3.1.3 Elemente und komplexe Datentypen

Die Deklaration eines Elements in XML- Schema erfolgt mit der folgenden Syntax: `<xsd:element name="elementName" type="xs:elementtyp">`. Das Attribut `name` nimmt den Elementnamen auf, `type` den Datentyp des definierten Elementes. In XML Schema unterscheidet man verschiedene einfache Datentypen für Elemente wie zum Beispiel `integer`, `string`, `float`, `double`, `boolean`, `dateTime` und weitere. Elemente, welchen einer der oben angeführten Datentypen zugewiesen wird, sind Datenelemente. Sie nehmen also keine weiteren Kindelemente mehr auf. Da es sich bei den abgegebenen Typen der Elemente aus Typen aus dem XML- Schema Namensraum handelt, müssen diese Typen bei der Deklaration mit dem XML Schema- Namensraum zugewiesenen Präfix (`xsd`) verbunden werden.

Containerelemente und Elemente, denen Attribute zugeordnet sind werden in Form so genannter komplexer Datentypen in XML- Schema deklariert. Das heißt auch Attributdeklarationen erfolgen innerhalb der Deklaration eines komplexen Datentyps für ein Containerelement.

Die Syntax zur Deklaration eines komplexen Datentyps lautet: `<xsd:complexType name="name_komplexerDatentyp"><!-- Inhalt komplexer Datentyp --></xsd:complexType>`. Innerhalb des Elements `xsd:complexType` erfolgt die Definition des komplexen Datentyps. An dieser Stelle wird also nicht ein Element deklariert sondern ein komplexer Datentyp. Über den Namen des neu deklarierten Typs kann dieser Typ nun bei Bedarf an beliebigen Stellen im XML Schema Dokument neuen Elemente zugewiesen werden. Auf diese Weise entstehen also wieder verwendbare Datentypen. Ein Vergleich mit der Definition von Klassen in objektorientierten Programmiersprachen ist zulässig. Der Inhalt eines komplexen Datentyps besteht grundsätzlich aus zwei Bereichen- nämlich Attributen und Kindelementen.

Zur Festlegung der Reihenfolge des Auftretens der Kindelemente in einem komplexen Datentyp stehen drei Modifikatoren zur Verfügung:

- `Sequence`: Die Elemente müssen genau in der angeführten Reihenfolge stehen
- `Choice`: Nur eines der angegebenen Element darf verwendet werden
- `All`: Die angeführten Element dürfen in beliebiger Reihenfolge stehen

Die Deklaration von Kindelementen innerhalb eines komplexen Datentyps unterscheidet sich nicht von der Deklaration von Datenelementen in XML- Schema. Durch Verwendung so genannter Modifikatoren besteht die Möglichkeit die Elementwiederholungen in einem Containerelement genau festzulegen. Per Default darf ein in einem XML Schema definiertes Element genau einmal vorkommen. Es stehen die Attribute `minOccurs` und `maxOccurs` zur Verfügung. Beide Schlüsselwörter sind optional. Ganzzahlwerte und die Eigenschaft `unbounded` kann ihnen als Wert zugewiesen werden. Neben der Methode Elemente in einem komplexen Datentyp überhaupt neu anzulegen, besteht auch die Möglichkeit innerhalb der Deklaration eines komplexen Datentyps auf ein benötigtes Element zu verweisen, wenn ein solches bereits irgendwo zuvor im Dokument definiert worden ist. Die Syntax dazu lautet `<xsd:element ref="name_benötigtesElement"/>`.

Elemente mit einfachen Typen dürfen in XML- Schema keine Attribute enthalten. Attribute können also in XML- Schema nur für ein Element, welches Teil eines komplexen Datentyps ist, deklariert werden. Die Attributdeklaration in XML- Schema erfolgt mit dem Element `attribute`. Die Deklaration eines Attributs in einem komplexen Datentyp erfolgt immer erst nach der Definition der Kindelemente und immer erst nach dem schließendem Tag eines Ordnungselements wie z. Bsp. `sequence`. Ansonsten ist sie syntaktisch gleich wie die Deklaration eines Elements.

Zusätzlich zum Attribut `name` und `type` des Elements `attribute` in XML Schema gibt es zwei weitere Attribute, welche es erlauben, das Attributelement noch näher zu spezifizieren. Das Attribut `use` kann die Werte `required` und `optional` annehmen. `Required` besagt, dass das Attribut in jedem Fall vom User im Instanzdokument gesetzt werden muss. `Optional` wird oft in Kombination mit der Angabe eines Defaultwertes in einem XML- Schema verwendet. Ein Defaultwert wird dann im Instanzdokument gesetzt, wenn vom User kein anderer Wert explizit gesetzt wird. Soll im Instanzdokument ein fixer, nicht vom User überschreibbarer Wert gesetzt werden, so muss im Schema Dokument das Element Attribut mit dem Attribut `fixed` besetzt werden. `Fixed` wird dann der gewünschte Wert zugewiesen. Eine Attributgruppe ist eine benannte Gruppe von Attributdeklarationen, die in komplexen Datentypen referenziert werden können. Die Attributgruppe wird mit dem Attribut `name` deklariert. Referenzen auf die Gruppe werden in Definitionen von komplexen Datentypen dadurch hergestellt, dass ein Element `xsd:attributeGroup` mit einem Attribut `ref` eingefügt wird, welches den Bezug zur gewünschten Attributgruppe herstellt.

Codelisting 2 zeigt den komplexen Datentyp `clinical_document_header` aus der XML Schema Datei für CDA Level 1 Release 1

```

<xsd:element name="clinical_document_header">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="id"/>
      <xsd:element ref="set_id" minOccurs="0"/>
      <xsd:element ref="version_nbr" minOccurs="0"/>
      <xsd:element ref="document_type_cd"/>
      <xsd:element ref="service_tm" minOccurs="0"/>
      <xsd:element ref="origination_dttm"/>
      <xsd:element ref="copy_dttm" minOccurs="0"/>
      <xsd:element ref="confidentiality_cd" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="document_relationship" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element ref="fulfills_order" minOccurs="0"/>
      <xsd:element ref="patient_encounter" minOccurs="0"/>
      <xsd:element ref="authenticator" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="legal_authenticator" minOccurs="0"/>
      <xsd:element ref="intended_recipient" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="originator" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="originating_organization" minOccurs="0"/>
      <xsd:element ref="transcriptionist" minOccurs="0"/>
      <xsd:element ref="provider" maxOccurs="unbounded"/>
      <xsd:element ref="service_actor" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="patient"/>
      <xsd:element ref="originating_device" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="service_target" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="local_header" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="common_atts"/>
    <xsd:attribute name="HL7-NAME" type="xsd:string"
      fixed="document_service_as_clinical_document_header"/>
    <xsd:attribute name="T" type="xsd:string" fixed="service"/>
    <xsd:attribute name="RIM-VERSION" type="xsd:string" fixed="0.98"/>
  </xsd:complexType>
</xsd:element>

```

Codelisting 2: komplexer Datentyp in XML Schema

Mit der oben gegebenen Einführung in XML Schema ist es möglich aus einem vorliegenden Schema für eine XML Anwendung abzulesen, welche Elemente und Attribute beispielsweise im Instanzdokument enthalten sein müssen oder können und wie die Reihenfolge ihres Auftretens im Instanzdokument zu sein hat.

3.2 Datenbankentwurf

Im Folgenden sollen kurz die Phasen für den Entwurf eines relationalen Datenbanksystems skizziert werden. Die generelle Aufgabe beim Entwurf eines Datenbankschemas besteht darin, eine formale Beschreibung zu finden (Modell) für einen zu modellierenden Teil der realen Welt. Beschreibung durch natürliche Sprache (Pflichtenheft), Beschreibung durch abstrakte graphische Darstellung und schließlich Beschreibung im relationalen Modell sind typische Zwischenstufen.

3.2.1 Entity Relationship Model (E/R Modell)

Das Entity Relationship Modell (E/R Modell) dient dazu, für einen Ausschnitt aus der realen Welt ein konzeptionelles Schema zu erstellen.

Die graphische Darstellung dieses Modells erfolgt in Form eines so genannten Entity Relationship Diagramms. Dabei handelt es sich um ein maschinenfernes Datenmodell auf einem hohen Abstraktionsniveau. Überlegungen zur Effizienz dieses Modells spielen zu diesem Zeitpunkt noch keine Rolle.

Das Entity Relationship Modell muss in weiterer Folge in ein relationales Modell überführt werden. Hierzu bestehen einfache Grundregeln zur Transformation, wobei jedoch die Gewinnung eines effizienten Schemas ein tiefes Verständnis vom Zielmodell erfordert.

3.2.2 Entities, Attribute, Relationen

Elemente des E/R Modells sind Entities (Entitäten), Attribute und Beziehungen („Relationships“) zwischen Entities.

Eine „Entity“ ist ein „Etwas“ ein „Seiendes“, welches eindeutig beschreibbar ist. Entities können Objekte sein wie ein Buch oder ein Auto, eine Person, aber auch Ereignisse wie beispielsweise ein Konzert oder eine Untersuchung beim Arzt.

Attribute sind charakterisierende Eigenschaften von Entitäten. Attribute können einfache Datentypen annehmen, wie Integer- oder Stringwerte. Häufig beschränkt man sich auf die wichtigsten Attribute.

Relationships (Beziehungen) stellen Zusammenhänge her zwischen den verschiedenen Entitäten. Diese Beziehungen können auch eigene Attribute haben.

Die graphische Darstellung dieser Konstrukte aus Entitäten, Attributen und den Beziehungen ist in der Literatur uneinheitlich. Entitätstypen werden meist als Rechtecke, Attribute als Kreise oder Ellipsen dargestellt. Beziehungen werden oft als Rauten mit Kanten zu den beteiligten Entitäten oder nur als Kanten symbolisiert.

Bei den Beziehungen können aufgrund der Funktionalität drei Formen unterschieden werden.

Seien E1 und E2 zwei verschiedene Entitäten, welche zueinander in Beziehung stehen.

- 1:1 Beziehung (one to one Relationship): jede Ausprägung der Entität E1 ist höchstens einer Ausprägung einer anderen Entität E2 zugeordnet und umgekehrt. Beispiel: die „ist verheiratet mit“ Beziehung (in unserem Kulturkreis).
- 1:n Beziehung (one to many Relationship): jeder Ausprägung von E1 sind beliebig viele Ausprägungen von E2 zugeordnet und jeder Ausprägung von E2 ist höchstens eine Ausprägung von E1 zugeordnet. Beispiel: ein Buch wird genau von einem Verlag herausgegeben, aber mehrere Bücher können von genau demselben Verlag verlegt werden.
- n:m Beziehung (many to many Relationship): mehrere Ausprägungen von E1 können zu mehreren Ausprägungen von E2 in Beziehung stehen und umgekehrt. Beispiel: mehrere verschiedene Studenten können verschiedene Vorlesungen besuchen.

Das E/R Modell trifft lediglich eine Aussage darüber, ob ein Objekt zu mehreren Objekten in Beziehung stehen darf oder zu maximal einem Objekt. Es wird im E/R Modell aber keine Aussage darüber getroffen, ob ein Objekt zu mehreren Objekten in Beziehung stehen darf oder zu maximal einem anderen Objekt. (Optionalität). Möchte man die Optionalität von Beziehungen graphisch ausdrücken, so gibt es hierfür auch eigene Notationen.

Schlüssel sind im E/R Modell ähnlich definiert wie im relationalen Modell. Primärschlüssel werden unterstrichen.

3.2.3 Vom E/R Modell zur Relation

Für den Schritt vom E/R Modell zur Relation gibt es einfache Umsetzungsregeln.

Jedem Entity-Typ wird eine Relation zugeordnet. Jedes Attribut einer Entität wird Attribut der Relation. Der Primärschlüssel wird Primärschlüssel der Entität. In weiteren Verlauf können weitere Attribute dazukommen.

Bei den Relationships ist das Vorgehen bei der Umsetzung vom E/R Modell zur Relation abhängig von der Funktionalität der Beziehung:

- 1:n Beziehung: Für die „one to many“- Beziehung wird keine neue Tabelle angelegt. Der Primärschlüssel auf der „one“- Seite der Relation kommt in die Relation der „many“- Seite. Das- oder die neu eingefügten Attribut(e) werden Fremdschlüssel in der Relation der „many“- Seite. Attribute der Relationship werden ebenfalls in die Relation der „many“- Seite eingebracht.
- 1:1 Beziehung: Die beiden Entitäten werden zu einer Relation zusammengefasst. Einer der Primärschlüssel der Entitäten wird Primärschlüsse der neuen Relation. Häufig erfolgt jedoch auch eine Umsetzung wie bei der „one to many“- Beziehung, wobei die Rollen der beteiligten Relationen austauschbar sind.
- n:m Beziehung: Hier erfolgt die Einführung einer zusätzliche Relation, welche den Namen der Relationship erhält. Die Primärschlüssel der Entitäten bilden zusammen einen zusammengesetzten Primärschlüssel der neuen Relation. Attribute der Relationship werden ebenfalls als Attribute in die neue Relation eingefügt.

3.3 Hypertext Markup Language

3.3.1 Formulare

Mit der Hypertext Markup Language HTML besteht die Möglichkeit Formulare zu erstellen. Es gibt verschiedene Typen von Formularen: Wenn ein Formular ausgefüllt ist kann es abgeschickt werden. Die Aufgaben von Formularen reichen vom Einholen strukturierter Auskünfte von Anwendern, über das Suchen in Datenbeständen hin zu individuellen Interaktionen des Anwenders.

Formulare können an beliebiger Stelle innerhalb des Dateikörpers einer HTML- Datei erstellt werden. Ein Formular besteht aus einem einleitenden Tag `<form>` und einem abschließendem Tag `</form>`. Wichtige (Pflicht)-attribute des `form`- Tags sind `action` und `method`.

`Action` gibt an, was mit den Formulardaten passieren soll, wenn ein Benutzer das Formular absendet. Als Wert kann Action beispielsweise eine E- Mail Adresse mit vorangestellten `mailto:` zugewiesen werden, oder aber Action wird als Wert ein URI (Universal Resource Identifier - universelle Quellenbezeichnung) zu einem Programm am Webserver zugewiesen, welches die Formulardaten weiterverarbeitet. Daneben gibt es noch andere Möglichkeiten der Verarbeitung von Formulardaten, welche nicht Gegenstand dieser Arbeit sind.

Die Wertzuweisung an das Attribut `method` bestimmt nach welcher `http` Übertragungsmethode die Formulardaten an ihr Ziel gelangen.

Die Zuweisung des Wertes `GET` (`<..method = "GET"..>`) ist die Default Einstellung: Die Formulardaten werden dabei als Parameter an die Aufrufadresse angehängt. Das verarbeitende Script kann die als Parameter übergebene Zeichenkette zur weiteren Verarbeitung auslesen. Wird `method` der Wert `POST` zugewiesen (`<..method = "POST"..>`), werden die Formulardaten vom Web-Server über den Standardeingabekanal zur Verfügung gestellt. Die Empfehlung des W3C ist dahingehend, `GET` dann zu verwenden, wenn das auswertende Programm die Daten nur zur Ablaufsteuerung benötigt (z.B. zum Weiterblättern). Die Verwendung von `POST` wird dann empfohlen, wenn die Daten über das auswertende Programm hinaus weiterverarbeitet werden (z.B. Speicherung in einer Datenbank).

Es gibt verschieden Typen von Formularen. An dieser Stelle wird kurz auf die Formulartypen Eingabefelder, Eingabebereiche sowie Auswahllisten und Radiobuttons eingegangen.

Einzeilige Eingabefelder werden verwendet für die Aufnahme von einem oder wenigen Wörtern oder einer Zahl. Der Tag `<input>` definiert ein einzeiliges Eingabefeld, das mit dem Tag `<name>` auch einen internen Bezeichner erhalten soll. Der vergebene Namen wird von weiterverbreitenden Scripts benötigt, um auf die Daten des Eingabefeldes zugreifen zu können. Weiter kann bei einzeiligen Eingabefeldern die Anzeigelänge in Zeichen mit `size = ""` sowie die interne Feldlänge mit dem Attribut `maxlength = ""` festgelegt werden.

Mehrzeilige Eingabefelder werden mit Tag `<textarea>` eingeleitet. Ebenso wie einzeilige Eingabefelder sollten mehrzeilige Eingabefelder einen internen Bezeichner mit dem Attribut `name = ""` haben. Angaben zur Höhe und Breite des Eingabebereiches sind obligat. (Attribute `rows = ""`, `cols = ""`). Beendet werden mehrzeilige Eingabebereiche mit dem schließenden Tag `</textarea>`.

Auswahllisten ermöglichen es dem Anwender aus einer vorgegeben Liste eine Auswahl zu treffen. Der Text des ausgewählten Eintrages wird übertragen, sobald das Formular abgeschickt wird. Eine Auswahlliste wird mit dem Tag `<select>` eingeleitet. Mit dem Attribut `name` wird einer Auswahlliste ein interner Bezeichner zugeordnet, der wie bei den anderen Formulartypen für die Auswertung der Formulardaten gebraucht wird. Mit `<option>...</option>` zwischen einleitendem `<select>` und abschließendem `</select>` Tag werden jeweils die Einträge der Auswahlliste definiert. Im Tag `<option>` steht der Text des Listeneintrages.

Radiobuttons werden mit der Syntax `<input type = "radio">` definiert. Jeder Radio Button erhält einen internen Bezeichnernamen. Radiobuttons mit demselben Bezeichnernamen bilden eine Gruppe, aus der ein Anwender genau einen Button markieren kann.

Mit dem Attribut `value` kann für jeden Radiobutton ein interner Bezeichnerwert bestimmt werden, der übertragen wird, wenn das Formular abgeschickt wird.

In HTML stehen zwei Standardbuttons zur Verfügung, um mit Formulardaten umzugehen. Der Button `submit` („Abschicken“) dient dazu, das ausgefüllte Formular abzuschicken, wobei die Formulardaten entsprechend dem zugewiesenen Wert des Attributs `action` im einleitenden `<form>` Tags verarbeitet werden. Mit dem Button Abbrechen werden alle Eingaben verworfen und das Formular wird nicht abgeschickt. Ein Button zum Abschicken der Formulardaten wird mit der Syntax `<input type = "submit" >` erzeugt. Ein Button zum Löschen der eingegebenen Daten wird mit der Syntax `<input type = "reset">` definiert. Mit dem Attribut `value` erfolgt die Beschriftung der Buttons.

3.3.2 Tabellen als Mittel für Web- Seiten- Layouts

Tabellen können in HTML definiert werden, um tabellarische Daten darzustellen, oder um Text und Graphik am Bildschirm zu positionieren. Man kann unterscheiden zwischen Tabellen mit Gitternetzlinien für tabellarische Daten und Tabellen ohne Gitternetzlinien so genannten „blinde Tabellen“ für die Verteilung von Inhalten auf einer Webseite.

Eine Tabelle wird mit dem einleitenden Tag `<table>` definiert. Das Attribut `border` gibt an, ob Gitternetzlinien angezeigt werden sollen (Attributwert für Border => 1) oder eine blinde Tabelle dargestellt werden soll (`border = "0"`). Eine neue Tabellenzeile wird mit `<tr>` eingeleitet und mit `</tr>` wieder abgeschlossen. Eine Tabellenkopfzelle wird mit `<th>` eingeleitet. Der Inhalt einer Kopfzelle wird fett und zentriert ausgegeben. Der Tag `<td>` leitet eine Datenzelle ein, darauf folgt die Eingabe des Zelleninhaltes, `</td>` schließt eine Datenzelle wieder ab. Die Tabelle wird mit dem schließendem `</table>` Tag wieder abgeschlossen.

HTML 4.0 bietet eine Syntax an, um einem Webbrowser am Beginn der Tabellendefinition mitzuteilen, wie viele Spalten eine Tabelle hat und wie breit diese sind. Dadurch kann der Tabellenaufbau im Browser schneller erfolgen. Nach dem einleitenden `<table >` Tag erfolgt mit `<colgroup>` die Definition der Tabellenspalten. Die Attribute `width` und `span` geben einheitliche Spaltenbreite (`width`) sowie Anzahl der Spalten an (`span`). Die Breitenangabe mit `width` erfolgt in Pixel oder Prozent der Gesamtbreite der Bildschirmausgabe. Das `<colgroup>` Element kann aber auch ohne weitere Attribute verwendet werden. Dafür wird im Anschluss an `<colgroup>` für jede gewünschte Spalte ein `<col>` Element definiert. Mit `width` kann die Breite jeder dieser Spalte in Prozent oder Pixel bestimmt werden.

Außerdem gibt es bei dem Attribut `width` noch die Möglichkeit das relative Breitenverhältnis der Spalten untereinander zu bestimmen- unabhängig davon, wie breit die Tabelle im Verhältnis zum Anzeigefenster ist, wobei bei dieser Möglichkeit eine Breite für die gesamte Tabelle im einleitendem `<table>` Tag angegeben werden muss (`<table..width = "100%"..>`)

3.4 PHP Skript

PHP ist die Abkürzung für „Hypertext Preprocessor“. PHP kann auf drei grundlegende Arten eingesetzt werden: PHP wurde ursprünglich als serverseitige Skriptsprache konzipiert, um dynamisch Webinhalte zu erzeugen. PHP kann aber auch an der Kommandozeile ausgeführt werden, um Aufgaben der Systemadministration zu erledigen (Kommandozeilen- Skripting). Schließlich können mit PHP (PHP-GTK) auch vollwertige, plattformunabhängige GUI-Anwendungen entwickelt werden.

Das Hauptanwendungsgebiet für PHP ist aber sicher der Einsatz als Skriptsprache am Webserver, um dynamisch Webinhalte zu erzeugen. Für diese Zwecke wird es auch in dieser Arbeit genutzt. Die Besonderheit von PHP besteht darin, dass der PHP Programmcode direkt in HTML Code eingebettet werden kann, wobei die PHP Anteile speziell markiert werden. PHP wird heute direkt als Modul des Webserver geladen und daher direkt im Kontext es Webserver Prozesses zur Ausführung gebracht.

In dieser Arbeit werden folgende Aufgaben durch Einsatz von PHP als Skriptsprache realisiert: Auswertung von Formulardaten, Datenbankzugriff auf eine MySQL Datenbank. Generieren von XML durch Einsatz von Stringfunktionen. Daher sollen diese Bereiche hier kurz angeschnitten werden.

3.4.1 Arrays in PHP

Arrays sind ein häufig verwendeter Datentyp in PHP, weil Arrays in PHP sehr flexibel eingesetzt werden können. Im Gegensatz zu anderen Programmiersprachen können PHP Arrays nicht nur über eine numerische Referenz angesprochen werden. Ein PHP Array besteht aus Schlüsseln (`key`) und Werten (`value`). Schlüssel können vom Typ Integer (indiziertes Array) oder vom Typ String (assoziatives Array) sein. Werte können von jedem Typ sein, wobei innerhalb ein und desselben Arrays verschiedenen Datentypen als Werte vorkommen dürfen. Die folgende Abbildung zeigt das Erzeugen eines Arrays, Funktionen zum Anzeigen von Arrays sowie das Ansprechen eines Elementes aus einem Array in PHP.

```

<?php //Erzeugung eines Arrays
$neues_array = array('wert1','wert2');
//Schlüssel selbst zuweisen
$neues_array2 = array("erstePosition" => "Wert1",
                    "zweitePosition" => "Wert2");

//Anzeigen mit print_r()
print_r($neues_array);
echo "</p>\n";
/*Ausgabe:
Array ( [0] => wert1 [1] => wert2 ) array(2)
*****/
//Genauere Anzeige mit var_dump()
var_dump($neues_array2);
echo "</p>\n";
/*Ausgabe:
array(2) { ["erstePosition"]=> string(5) "Wert1"
           ["zweitePosition"]=> string(5) "Wert2" }
*****/
//Ansprechen eines Array_ Elementes
echo $neues_array[0];
echo $neues_array2['erstePosition'];
/*Ausgabe
wert1Wert1
*****/ ?>

```

Codelisting 3: Arrays in PHP

3.4.2 Verarbeitung von Formularen

Je nach Einstellung des Wertes für das Attribut `method` im einleitenden Tag `<form>` eines HTML Formulars stehen die abgeschickten Formulardaten in der globalen PHP Variablen `$_POST` oder `$_GET` zur Verfügung. Wird versucht auf einen Array Schlüssel zuzugreifen, der nicht angelegt wurde, weil ein Anwender das entsprechende Formularfeld leer gelassen hat, kommt es zu einer Warnung. Um dies zu vermeiden, kann man mit einer in PHP vorgesehenen Funktion `array_key_exists ()` prüfen, ob das Array (`$ar`) den Schlüssel (`$key`) auch enthält. Der Funktion wird als erster Parameter der zu suchende Schlüssel, als zweiter Parameter das gesamte Array übergeben. Mit einer Hilfsfunktion (`function array_item ($ar, $key)`) kann man dann die gültigen Werte aus dem Array auslesen.

Um Daten, die bereits in ein Formular eingegeben wurden noch einmal zu bearbeiten, gibt es eine praktische Methode, indem an bestimmter Stelle im HTML Formular PHP Code zur Ausgabe des zuletzt in dieses Feld eingegebenen Wertes eingebettet wird. (Codelisting 4)

Um den problemlosen Umgang mit HTML Sonderzeichen in Formulardaten zu gewährleisten, stellt PHP die Funktion `htmlspecialchars()` zur Verfügung. Die Funktion ersetzt die HTML Sonderzeichen `"`, `&`, `<`, `>` durch die Zeichenketten. (`"`, `&`, `<`, `>`.)

Wenn in einem Formular Zeichen erwartet werden, die nicht im latin-1 Zeichensatz (ISO-8859-1) enthalten sind, kommt der Unicode Zeichensatz (ISO-10646) zum Einsatz. Unicode ist ein Zeichensatz, der so groß ist, dass er alle Zeichen heute lebender Sprachen und eine Vielzahl weiterer Sonderzeichen aus verschiedensten Bereichen aufnehmen kann. Trotz der Vorteile der Verwendung von Unicode ist die Unicode Unterstützung durch die verschiedenen Komponenten, welche bei einer Webapplikation zum Einsatz kommen noch nicht fehlerfrei.

3.4.3 Session Verwaltung

Cookies sind ein Mechanismus, über den es für den Webserver möglich ist, Informationen auf dem Client zu speichern und abzufragen. Solche Informationen sind zum Beispiel Einstellungen, die ein Anwender gemacht hat und beim nächsten Versuch wieder vorfinden will. Ein Cookie beinhaltet Informationen zum Gültigkeitsbereich, Gültigkeitsdauer und dem Absender. Die maximale Größe eines Cookies liegt bei vier Kilobyte. Nachteile von Cookies sind die Abhängigkeit von clientseitigen Sicherheitseinstellungen und die Beschränkung nicht mehr als 4 Kilobyte Daten speichern zu können.

Die Session Verwaltung verfolgt hier ein anderes Konzept. Pro Sitzung („Session“) bekommt ein Client eine einmale Kennung (Session- Id), durch die ihn ein Server identifiziert. Die Informationen werden auf dem Server gehalten und unterliegen keinen Größenbeschränkungen. Die Beziehung zwischen Session- Id und Client wird wiederum über ein Cookie hergestellt oder über einen anderen (unsicheren) Mechanismus.

Um Session Variablen zu verwenden, wird am Anfang einer Anwendung mit der Syntax `session_start()` eine Session erzeugt. Sodann können die Variablen, die persistent sein sollen in der globalen Session Variable `$_SESSION` gespeichert werden. Session Variablen können zum Beispiel verwendet werden, um bereits in einem Suchformular eingegebene Daten in einem anderen Formular durch Einbettung von php- Code direkt in einem Formularfeld wieder auszugeben.

```

<html>
<!--.....-->
<body>
<?php
session_start();
?>
<!--Formular-->
<form method="post" action="patArztUntersuchDat.php">
<b>Patientendaten:</b><br>

<input type="text" name="vorname" value=<?php echo $_SESSION['vorname']
;?>>
Vorname des Patienten<br>
<input type="text" name="nachname" value=<?php echo $_SESSION['nachname']
;?>>
Familiename des Patienten<br>
<input type="text" name="gebDat" value=<?php echo $_SESSION['gebDat'] ;?>>
<input type="submit" name="submitbutton" value="in DB eintragen">
<!--
.....-->
</form>
<!--Aufruf PHP Prozessor Datenbankbindung-->

<?php
include("connect_to_DB.php");
/*
.....*/
?>
</body>
</html>

```

Codelisting 4: SESSION Variablen in PHP

3.4.4 Zugriff auf MySQL Datenbank

Für den Zugriff auf MySQL Datenbanken stehen in PHP 5 zwei verschiedene Schnittstellen zur Verfügung. Dies sind die `mysql` Funktionen, die von früheren PHP Versionen bekannt sind und die `mysqli` Schnittstellen. Letztere ist eine objektorientierte Schnittstelle, welche MySQL 4.1 und neuere Versionen voraussetzt.

Für diese Arbeit werden einige der (früheren) `mysql` Funktionen für den Datenbankzugriff verwendet. Dazu muss in der PHP Konfigurationsdatei `php.ini` `php_mysql.dll` aktiviert werden und die benötigte Bibliothek `libmysql.dll` in das Windows Systemverzeichnis kopiert werden.

Für den Verbindungsaufbau zur MySQL Datenbank wird die Funktion `mysql_connect` verwendet, welcher zumindest drei Parameter übergeben werden müssen: Rechnername (Hostname) des MySQL Servers, Benutzername und Passwort. (`$verbindung = mysql_connect("host", "user", "passwd")`). Tritt ein Fehler bei der Verbindung auf so enthält die Variable `$verbindung` den Wert `FALSE` und es wird eine Fehlermeldung ausgegeben. Sobald eine Verbindung hergestellt ist können weitere `mysql` Funktionen ausgeführt werden.

So kann etwa mit der Funktion `mysql_select_db("datenbank_name")` eine Datenbank für die weiteren Aktionen ausgewählt werden. Mit `mysql_close` wird die Verbindung geschlossen.

Mit der Syntax `mysql_query` kann jede Art von SQL Kommando ausgeführt werden (INSERT, UPDATE, DELETE, CREATE TABLE, ALTER TABLE, DROP TABLE). Es gibt noch eine ganze Reihe weiterer nützlicher `mysql`-Kommandos, um sich die Auswirkungen vorher durchgeführter Aktionen anzeigen zu lassen zur Anzeige von Systeminformationen über den Datenbankserver und zur Fehlersuche.

Zur Auswertung von `SELECT` Kommandos wird für diese Arbeit die `mysql` Funktion `mysql_fetch_array` verwendet. Diese Funktion liefert das Ergebnis einer Abfrage als assoziatives Array zurück, sodass auf die gewünschten Spalten des Ergebnisses einfach zugegriffen werden kann. (Codelisting 5)

```
<?php
//...
//pat_id ermitteln
$result_pat_id = mysql_query("SELECT pat_id FROM patient WHERE vorname =
'$vorname'".
" AND nachname = '$nachname' AND gebDat = '$gebDat'");

$result_pat_id = mysql_fetch_array($result_pat_id);
$pat_id = $result_pat_id['pat_id'];
echo "pat_id ist $pat_id<p>\n";
//..
?>
```

Codelisting 5: MySQL Funktionen in PHP

3.4.5 XML generieren

Genau wie PHP dazu verwendet werden kann dynamisch HTML zu erzeugen, kann es auch dynamisch XML erzeugen. Die Daten, die dazu mit XML ausgezeichnet werden, können aus verschiedenen Quellen stammen. Aus einer Datenbankabfrage, aus Formulardaten oder anderen Anwendungen.

Zur Generierung eines XML Dokumentes aus einem PHP Skript muss der MIME Typ (Multipurpose Internet Mail Extensions; gibt Auskunft über die Art der in einem Netzwerk zu übertragenden Daten zwischen zwei Anwendungen) des erzeugten XML Dokumentes mit der PHP Funktion `header()` als `"text/xml"` deklariert werden. Die XML Deklaration kann einfach mit dem `echo` befehl ausgegeben werden. (`<? Echo "<?xml version = "1.0" encoding = "iso-8859-1"?>" ;?>`)

Genau wie es in PHP keine speziellen Funktionen zur Generierung von HTML gibt, gibt es auch keine speziellen Funktionen zur Generierung von XML. Der Code wird einfach mit `echo` ausgegeben. Im Kapitel 4 wird beschrieben, wie in dieser Arbeit durch Einsatz von PHP dynamisch XML generiert wird. (4.3.2).

3.5 Extensible Stylesheet Language (XSL)

Die extensible Stylesheet Language XSL besteht aus zwei Teilen: Extensible Stylesheet Language Transformations (XSLT) einerseits und Extensible Stylesheet Language Formating Objects (XSL-FO) andererseits. XSL kann vor allem als Beschreibungssprache für Formate bezeichnet werden, es enthält darüber hinaus aber auch Konstrukte wie bedingungsabhängige Anweisungen oder Schleifenanweisungen, welche eher an Programmiersprachen erinnern. Beide Teile benutzen die so genannte XPath Syntax. XPath ist eine Nicht XML Syntax, welche eingesetzt wird, um bestimmte Teile von XML Dokumenten zu adressieren. XPath kommt zusätzlich auch noch in anderen XML Technologien zum Einsatz, auf die hier nicht eingegangen wird. XSLT ist eine XML Anwendung, die die Transformation eines XML Dokumentes in ein beliebiges anderes XML Dokument ermöglicht. Außerdem bietet XSLT die Möglichkeit XML ins HTML Format zu transformieren. XSL FO ist eine XML Applikation zur genauen Beschreibung der Anordnung von Text auf einer Seite. XSL FO ist daher die geeignete Methode, wenn die mit XML ausgezeichneten Daten für den Druck aufbereitet werden sollen. Im Folgenden sollen kurz die für diese Arbeit verwendeten Methoden aus XPath und XSLT beschrieben werden.

3.5.1 XPath

Ein XML Dokument bildet eine Baumstruktur, welche aus Knoten besteht. Mit XPath-Ausdrücken können Knotenmengen oder beliebige Daten in einem XML Dokument adressiert werden. Dabei spielt es keine Rolle, ob es sich um ein Element, ein Attribut oder konkrete Daten handelt. XPath identifiziert Knotenmengen oder einzelne Knoten entsprechend der Position, der relativen Position, des Inhaltes und verschiedener anderer Kriterien. XPath bietet folgende Funktionalitäten, die oft in Kombination eingesetzt werden: Angabe von Knotentypen, Achsen und Pfaden; Definition logischer Ausdrücke; zusätzliche Funktionen (Stringfunktionen, einige einfache mathematische Funktionen, Funktionen zur Knotenmanipulation).

3.5.1.1 Adressierung

Für die Adressierung von Anteilen eines XML Dokumentes spricht man in Analogie zur Adressierung auf einem Datenträger (Wurzelverzeichnis, Ordner, Dateien) von Knotentypen, Achsen und Pfaden.

Jeder Bestandteil eines wohlgeformten XML Dokumentes kann als Knoten betrachtet werden. Aus dieser Sichtweise gibt es im XPath Datenmodell sieben verschiedene Knotentypen: Wurzelknoten, Elementknoten, Attributknoten, Textknoten, Namensraumknoten, Verarbeitungsanweisungen, Kommentarknoten. Aus der Baumstruktur eines XML Dokumentes ergibt sich auch, dass diese Knoten in verschiedenen Beziehungen zueinander stehen. So gibt es Knoten, die von anderen Knoten abhängen und solche, die auf der gleichen Ebene stehen. Dafür werden in XPath englische Begriffe verwendet, die aus der Verwandtschaftsterminologie stammen. (self, parent child und andere) und als Achsen bezeichnet werden.

In XPath gibt es zwei syntaktische Mittel, um den Pfad zu einer Knotenmenge oder einem bestimmten Knoten zu notieren, nämlich den einfachen Schrägstrich oder den Doppelpunkt. Im Folgenden wird nur der Schrägstrich verwendet.

Ein vollständiger XPath Ausdruck besteht aus drei Anteilen

- Suchbereich (Achse)
- Gewünschten Knoten (Knotentest)
- optional: Einer einschränkenden Bedingung

Suchbereich [Bedingung] / Knoten [Bedingung]

3.5.1.2 Adressierung mit Positionsangaben und Bedingungen

Einschränkende Bedingungen in XPath sind optional. Durch Einsatz von Bedingungen wird die adressierte Knotenmenge auf die Knoten reduziert, für welche die Auswertung der Bedingung `TRUE` ergibt. Für die Formulierung der Bedingung können (allgemein bekannte) Operatoren (`=`, `!=`, `or`, `and`, `<`, `>`, `<=`, `=>`, `+`, `-`, `div`, `*`, `mod`) verwendet werden, so genannte XPath Funktionen zur Anwendung gebracht werden, der Operatoren und XPath Funktionen miteinander kombiniert werden.

Tabelle 2 gibt ein Auszug aus XPath Funktionen wieder.

XPath Funktion	Beschreibung
text()	gibt den Wert eines Knotens zurück
concat()	fügt die übergeben Argumente zusammen
string-length()	gibt Anzahl der Zeichen zurück
round(zahl)	Zahl wird gerundet
boolean()	Argument als wahr oder falsch bewerten
concat()	Zeichenketten zu einer zusammenfassen
contains()	auf bestimmte Teilzeichenkette überprüfen
current()	aktuellen Knoten ermitteln
normalize-space()	Leerzeichen entfernen
starts-with()	beginnt Zeichenkette mit bestimmter Teilzeichenkette?
position()	Positionsnummer des aktuellen Knotens ermitteln
substring (string, int, int)	schneidet einen String aus

Tabelle 2: XPath Funktionen

3.5.2 Extensible Stylesheet Language Transformations (XSLT)

XSLT ist eine XML Anwendung, die die Transformation eines XML Dokumentes in ein beliebiges anderes XML Dokument ermöglicht. Außerdem bietet XSLT die Möglichkeit XML ins HTML Format zu transformieren. Ein XSLT Dokument ist ein XSLT Stylesheet. Ein solches Stylesheet enthält Anweisungen für die Transformation eines XML Dokumentes.

Ein XSLT Prozessor ist eine Software, welche die im Stylesheet angegebenen Anweisungen ausliest und auf das zu transformierende Dokument anwendet. Ausgehend von der Betrachtungsweise auf ein XML Dokument als Baumstruktur spricht man auch von der Transformation des Quellbaums in den Ergebnisbaum.

3.5.2.1 Templates

Die Steuerung, welche Ausgabe aus welcher Eingabe erzeugt wird, erfolgt in XSLT Stylesheets in Form von Template Regeln. Im Grunde genommen besteht jedes Stylesheet aus einem oder mehreren Templates, die durch das `xsl:template` Element eingeleitet werden. Templates enthalten Regeln für die Ausgabe mit Hilfe von XPath adressierbarer Teile des Quellbaumes in den Ergebnisbaum.

In einem Template wird zunächst mit Hilfe eines XPath Ausdruckes eine Knotenmenge des Eingabebaumes ausgewählt. Die weiteren Anweisungen innerhalb dieses Templates sind Regeln zur Transformation dieser ausgewählten Knotenmenge.

Im Ergebnisbaum wird die ursprünglich ausgewählte Knotenmenge von der transformierten Knotenmenge „überschrieben.“ Werden bei einer Transformation mit XSLT nicht alle Knotenmengen eines Eingabebaumes durch Templates transformiert, so werden die ausgezeichneten Daten der nicht von Template- Regeln transformierten Knoten oder Knotenmengen des Eingabebaumes einfach am Ende der transformierten Ausgabe ausgegeben.

Mehrere Templates innerhalb eines Stylesheets teilen den Quellbaum durch ihre XPath Anweisungen in Unterbäume auf. Auf diese Weise können verschiedene Bereiche des Eingabebaumes komplett unabhängig von einander verarbeitet werden. Durch das Element `xsl:apply-templates` wird die Reihenfolge der Ausgabe der durch die Template Regeln transformierten Bereiche des Quellbaumes im Ergebnisbaum festgelegt. Templates können in einander verschachtelt sein, wobei folgendes gilt: je detaillierter die XPath Anweisung ist, umso höher steht die Regel in der Rangordnung. Die praktische Anwendung der in den letzten Abschnitten beschriebenen Methoden wird in Kapitel 4 (Durchführung) anhand konkreter Beispiele gezeigt. (Abschnitt 4.4).

3.5.2.2 Weitere XSLT- Elemente

Im Folgenden wird noch auf weitere XSLT- Elemente eingegangen, welche in dem programmierten Stylesheet angewendet werden. Das XSLT Element `output` steht am Anfang eines Stylesheets und teilt einem XSLT Prozessor mit, in welchem Format die Ausgabe erfolgen soll. Das `method` Attribut des `output` Elementes kann dabei die Werte `xml`, `html` oder `text` annehmen.

```
<xsl:output method = "html"...>
```

Das XSLT Element `value-of` dient dazu den String Wert eines XPath Ausdruckes zu berechnen und ihn in den Ergebnisbaum einzufügen, wobei der Wert eines Elementes sein Textinhalt ist. Dem Attribut `select` des `value-of` Elementes wird dabei ein entsprechender XPath Ausdruck übergeben. Soll ein Attributwert ausgegeben werden so erfolgt dies nach Referenzieren des entsprechenden Knotens mit der XPath Syntax gefolgt von einem Schrägstrich gefolgt vom dem Zeichen „Klammeraffe“ und einem „V“ (für „Value“).

```
<xsl:value-of select= "cda:birth_dttm/@V" />//Geburtsdatum wird ausgegeben
```

Kontrollstrukturen wie die Elemente `xsl:if` und `xsl:for-each` erinnern auf den ersten Blick eher an Konstrukte wie sie aus Programmiersprachen bekannt sind. Das Element `for-each` ermöglicht es eine Schleife zu konstruieren, welche es erlaubt auf gleich lautende Knoten derselben Ebene eines Quellbaumes selektiv der Reihe nach zuzugreifen.

In Kombination mit dem Element `xsl:if` kann für jeden gleich lautenden Knoten derselben Ebene dann noch eine Bedingung getestet werden.

Das Element `xsl:text` bietet die Möglichkeit statischen Text im Ergebnisbaum zu erzeugen. Zum Beispiel ein Leerzeichen oder auch Programmcode wie etwa Java- Skript.

Daneben gibt es noch einige weitere Elemente für die Verwendung in XSLT. Ein Auszug davon erscheint in Tabelle 3.

XSLT Elementname	Beschreibung
<code>xsl:stylesheet</code>	Stylesheet- Wurzelement
<code>xsl:template</code>	Schablone für Ergebnisbaum definieren
<code>xsl:import</code>	Stylesheets importieren
<code>xsl:include</code>	Stylesheets inkludieren
<code>xsl:output</code>	Erzeugen des Ergebnisbaums kontrollieren
<code>xsl:apply-templates</code>	untergeordnete Schablonen anwenden
<code>xsl:for-each</code>	für jedes Element aus einer Menge wiederholen
<code>xsl:if</code>	wenn- Bedingung
<code>xsl:number</code>	fortlaufende Nummerierung
<code>xsl:value of</code>	Wert ausgeben
<code>xsl:variable</code>	Variable definieren

Tabelle 3: Auswahl von XSLT Elementen

Die praktische Anwendung der in 3.5 vorgestellten Methoden erfolgt in Kapitel 4 (Durchführung Abschnitt 4.4).

4 Durchführung

Das Kapitel Durchführung zeigt die Anwendung der zuvor beschriebenen Methoden in der Praxis. Anhand der erarbeiteten Grundlagen zu XML Schema wird anhand eines zur Verfügung stehenden XML Schemas für CDA Level 1 Release 1 bestimmt, welche Merkmale für die Erstellung eines Befundberichtes für die konkrete medizinische Fragestellung erfasst werden müssen.

Aus Abschnitt 2.1.6 geht hervor, dass Daten, welche für eine statistische Auswertung herangezogen werden besser in einem relationalen Datenbanksystem gespeichert werden sollen als im XML Format. Daher werden die erfassten Daten zunächst in einer MySQL Datenbank gespeichert. Zunächst wird für die angenommene Situation eine E/ R Modell erstellt, welches in dann- wie im Methodenkapitel beschrieben- in ein relationales Modell überführt wird.

Der nächste Abschnitt zeigt die Programmierung der Webapplikation, durch Verwendung der Komponenten HTML, Apache Webserver, PHP Skript und MySQL Datenbank. Detailliert wird beschrieben wie die zuvor dargelegten Methoden eingesetzt werden, um schemagültiges XML zu generieren.

Abschnitt 4.4 zeigt die Programmierung des XSL Stylesheet zur Transformation des generierten Instanzdokumentes ins HTML Format und in Abschnitt 4. 5 wird die Installation und Konfiguration der verwendeten Werkzeuge gezeigt.

4.1 XML Schema für CDA Level 1

Grundlagen für den folgenden Abschnitt sind:

- Aufbau eines CDA Dokumentes (→ 2.2.3)
- XML Schema (→ 3.1)

Für diese Arbeit wird das folgende Situation hypothetisch angenommen: ein Patient konsultiert einen niedergelassenen Arzt für eine Untersuchung. Das Untersuchungsergebnis wird in Form eines Befundberichtes während der Untersuchung elektronisch dokumentiert. Der Patient und sein weiterbehandelnder Arzt erhalten (auf elektronischen Weg) Kopie dieses Befundberichtes.

Als nächster Schritt soll anhand des XML Schemas für CDA Level 1 und eines zur Verfügung stehenden Instanzdokumentes gezeigt werden, welche Daten für das oben beschriebene Szenario erfasst werden müssen.

Tabelle 4 zeigt alle für Implementierung verwendeten CDA Elemente und die Zugehörigkeit der Elemente zu einem der vier logischen Bereiche innerhalb des CDA Header. Weiter kann aus der Tabelle abgelesen werden, welches Attribut den jeweiligen Elementwert aufnimmt, und ob ein Element obligat im Instanzdokument vorkommen muss oder nicht.

Die nächsten beiden Tabellen greifen dem folgenden Abschnitt vor, da hier bereits Felder der Eingabeformulare und Felder aus der erst im nächsten Abschnitt zu entwerfenden Datenbank gezeigt werden:

Tabelle 5 zeigt den Zusammenhang zwischen Autoinkrementfeldern in der Datenbank und den jeweiligen CDA Elementen, in denen diese Daten im Instanzdokument erscheinen.

Tabelle 6 zeigt den Zusammenhang zwischen den in Formularfelder einzugebenden Daten und den CDA Elementen, welche diese Daten im Instanzdokument aufnehmen.

Tabelle 4: zeigt den Zusammenhang zwischen Autoinkrementfeldern in der Datenbank und den jeweiligen CDA Elementen, in denen diese Daten im Instanzdokument erscheinen.

Logischer Bereich / CDA Knoten im Instanzdokument	Elementname	obligat	Attribut
Document Information			
clinical_document_header/	id	x	EX
clinical_document_header/	document_type_cd	x	V
clinical_document_header/	origination_dttm	x	V
Service Actors			
clinical_document_header/intended_recipient/person/person_name/nm/	PFX		V
clinical_document_header/intended_recipient/person/person_name/nm/	GIV		V
clinical_document_header/intended_recipient/person/person_name/nm/	FAM		V
clinical_document_header/intended_recipient/person/addr	STR		V
clinical_document_header/intended_recipient/person/addr	HNR		V
clinical_document_header/intended_recipient/person/addr	ZIP		V
clinical_document_header/intended_recipient/person/addr	CTY		V
clinical_document_header/intended_recipient/person/	telecom		V
clinical_document_header/intended_recipient/person/	telecom		V
clinical_document_header/intended_recipient/person/	telecom		V
clinical_document_header/intended_recipient/person/	telecom		V
clinical_document_header/provider	provider.type_cd	x	V
clinical_document_header/provider/person	id	x	EX
clinical_document_header/provider/person/person_name/nm/	PFX		V
clinical_document_header/provider/person/person_name/nm/	GIV		V
clinical_document_header/provider/person/person_name/nm/	FAM		V
clinical_document_header/provider/person/addr	STR		V
clinical_document_header/provider/person/addr	HNR		V
clinical_document_header/provider/person/addr	ZIP		V
clinical_document_header/provider/person/addr	CTY		V
clinical_document_header/provider/person/	telecom		V
clinical_document_header/provider/person/	telecom		V
clinical_document_header/provider/person/	telecom		V
clinical_document_header/provider/person/	telecom		V

Logischer Bereich / CDA Knoten im Instanzdokument	Elementname	obligat	Attribut
Service Targets			
clinical_document_header/patient	patient.type_cd	x	V
clinical_document_header/patient/person	id	x	EX
clinical_document_header/patient/person/person_name/nm/	GIV		V
clinical_document_header/patient/person/person_name/nm/	FAM		V
clinical_document_header/patient/person/addr	STR		V
clinical_document_header/patient/person/addr	HNR		V
clinical_document_header/patient/person/addr	ZIP		V
clinical_document_header/patient/person/addr	CTY		V
clinical_document_header/patient/person/	telecom		V
clinical_document_header/patient/	birth_dttm		V
clinical_document_header/patient/	administrative_gender_cd		V
Encounter Information			
clinical_document_header/patient_encounter	encounter_tmr		V

Tabelle 4:verwendete CDA Elemente

Autoinkrementfeld in Datenbank	Logischer Bereich / CDA Knoten	Elementname
	Dokument Information	
fallzahl	clinical_document_header/	id
fall_ts	clinical_document_header/	origination_dttm
	Service Actors	
aerzte_id	clinical_document_header/provider/person	id
	Service Targets	
pat_id	clinical_document_header/patient/person	id

Tabelle 5: Beziehung Autoincrementfeldern in Datenbank und CDA Elemente

Tabelle 6 zeigt den Zusammenhang zwischen den in Formularfelder einzugebenden Daten und den CDA Elementen, welche diese Daten im Instanzdokument aufnehmen.

Name Formularfeld	Logischer Bereich / CDA Knoten	Elementname
	Dokument Information	
	Service Actors	
Weiterbehandelnder Arzt:		
Vorname	clinical_document_header/intended_recipient/person/person_name/nm/	GIV
Nachname	clinical_document_header/intended_recipient/person/person_name/nm/	FAM
Straße	clinical_document_header/intended_recipient/person/addr	STR
Hausnummer	clinical_document_header/intended_recipient/person/addr	HNR
Ort	clinical_document_header/intended_recipient/person/addr	CTY
Postleitzahl (PLZ)	clinical_document_header/intended_recipient/person/addr	ZIP
Telefonnummer	clinical_document_header/intended_recipient/person/	telecom
Telefax	clinical_document_header/intended_recipient/person/	telecom
e- Mail Adresse	clinical_document_header/intended_recipient/person/	telecom
	Service Targets	
Patient:		
Vorname	clinical_document_header/patient/person/person_name/nm/	GIV
Nachname	clinical_document_header/patient/person/person_name/nm/	FAM
Geburtsdatum	clinical_document_header/patient/	birth_dttm
Straße	clinical_document_header/patient/person/addr	STR
Hausnummer	clinical_document_header/patient/person/addr	HNR
Ort	clinical_document_header/patient/person/addr	CTY
Postleitzahl (PLZ)	clinical_document_header/patient/person/addr	ZIP
Telefonnummer	clinical_document_header/patient/person/	telecom
	Encounter Data	
Untersuchungsdatum	clinical_document_header/patient_encounter	encounter_tmr

Tabelle 6: Beziehung Formularfeldnamen und CDA Elemente

4.2 Datenbankentwurf

Grundlagen für den folgenden Abschnitt sind: Datenbankentwurf (→ 3.2)

4.2.1 Entity Relationship Model (E/ R Model)

Für diese Arbeit für das folgende Situation hypothetisch angenommen: ein Patient konsultiert einen niedergelassenen Arzt für eine Untersuchung. Das Untersuchungsergebnis wird in Form eines Befundberichtes während der Untersuchung elektronisch dokumentiert. Der Patient und sein weiterbehandelnder Arzt bekommen eine Kopie dieses Befundberichtes.

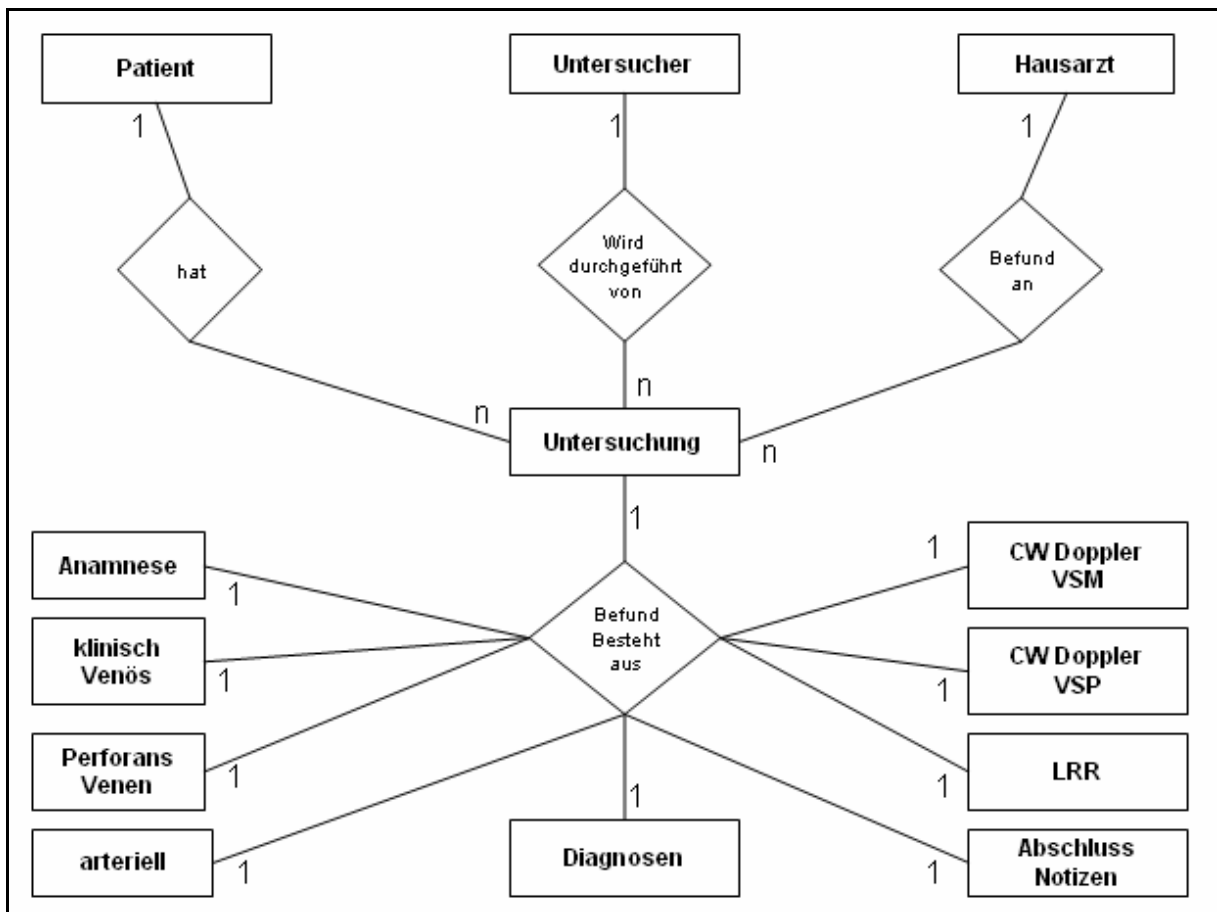


Abbildung 5: Entity Relationship Modell

Abbildung 5 zeigt die Modellierung der oben beschriebenen Situation in Form eines E/R Modells. In dieser Abbildung werden lediglich die Entitäten und ihre Beziehungen zueinander dargestellt. Die Attribute der einzelnen Entitäten werden in den folgenden Tabellen dargestellt.

Die Tabellen 7 und 8 zeigen die für diese Arbeit verwendeten Entitäten und deren Attribute.

4.2.2 Entities und Attribute für CDA Header

Entity	Attribute
Patient	<u>pat_id</u> , Nachname, Vorname, Geburtsdatum, Geschlecht, Straße, Hausnummer, Postleitzahl, Telefonnummer, Ort
weiterbehandelnder Arzt	<u>arzt_id</u> , Nachname, Vorname, Straße, Hausnummer, Ort, Postleitzahl, Telefonnummer, Telefax, e-Mail Adresse
fall	fallzahl, Untersuchungsdatum
Untersucher	<u>untersucher_id</u> , Nachname, Vorname, Straße, Hausnummer, Ort, Postleitzahl, Telefonnummer, Telefax, e-Mail Adresse

Tabelle 7: Entities und Attribute für CDA Header

4.2.3 Entities und Attribute für CDA Body

Entity	Attribute
Anamnese 01	<u>anamnese01_id</u> ; Krankheitsbeginn; Krankheitsdauer; familiäre Belastung; berufliche Belastung; Anzahl Schwangerschaften; Einnahme von Ovulationshemmern; Antikoagulantien
Anamnese 02	<u>anamnese02_id</u> ; Zn. Erysipel; Zn. Thrombophlebitis; Zn. Phlebothrombose; Zn. Kompressionstherapie; Zn. Varizen OP; Zn. Lungenembolie
Anamnese 03	<u>anamnese03_id</u> ; Schmerzhafte Varizen; Müdigkeit Schweregefühl; Wadenkrämpfe; Juckreiz; Beinschwellung
Befund arteriell und andere	<u>arteriell_weitere_id</u> ; arteriell o.B; arteriell path.; Ulcus cruris; Ulcusnarbe; Corona phlebektatica; Zyanose; prim. Lymphödem; sek. Lymphödem; Saphenus Läsion; Suralis Läsion
klinisch venös	<u>klinvenoes_id</u> ; Vena Giacomini; Vena saphena magna saphena parva Anastomose; Seitenäste Oberschenkel; Seitenäste Unterschenkel; Fußvarizen; Perinealvarizen; retikuläre Varizen Oberschenkel; retikuläre Varizen Unterschenkel; Besenreiser Oberschenkel; Besenreiser Unterschenkel
Perforansvenen	<u>klinPerf_id</u> ; Perforans C1; Perforans C2; Perforans C3; Perforans C4; Boyd; Dodd; Bassi 8cm; Perforans lat US 12cm
CW Doppler VSM	<u>CW_VSM_id</u> ; Crosse VSM; Crosse VSM Rezidiv; VSM Stamminsuff. Prox. OS; VSM Stamminsuffizienz prox. Oberschenkel; VSM Stamminsuffizienz dist. Oberschenkel; VSM Stamminsuffizienz prox. Unterschenkel; VSM Stamminsuffizienz dist. Unterschenkel
CW Doppler VSP	<u>CW_VSP_id</u> ; VSP Crosse Insuffizienz; VSP Crosse Insuffizienz Rezidiv; VSP Insuffizienz prox. US; VSP Insuffizienz dist. US
LRR	<u>LRR_id</u> ; LRR normal; LRR pathologisch; LRR normal bei Kompression; LRR pathologisch bei Kompression
Hauptdiagnosen	<u>diagnosen_id</u> ; Diagnose 1; Diagnose 2; Diagnose 3; Diagnose 4
Nebendiagnosen	<u>nebendiagnosen_id</u> ; Nebendiagnosen 1; Nebendiagnosen 2; Nebendiagnosen 3; Nebendiagnosen 4

Tabelle 8: Entities und Attribute für CDA Body

4.2.4 Vom E/R Modell zur Relation

Tabelle 4 und Tabelle 5 zeigen die Entitäten mit ihren Attributen. Jeder Entität wird eine Tabelle zugeordnet, die Primärschlüssel der Entitäten werden Primärschlüssel der neuen Relationen.

Die Umsetzung der „one to many“ Beziehungen erfolgt, in dem der Primärschlüssel der „one“- Seite in die Relation der „many“- Seite eingefügt wird. Das- oder die neu eingefügten Attribut(e) werden Fremdschlüssel in der Relation der „many“- Seite.

Für die Umsetzung der „one to one“ Beziehung ins relationale Modell wird die gleiche Vorgangsweise gewählt wie für die oben beschriebene „one to many“ Beziehung. Der Primärschlüssel der Relation „Untersuchung“ ist „fallzahl“. Dieser wird Fremdschlüssel in allen Tabellen, die über die Beziehung „Befund besteht aus“ mit der Relation „Untersuchung“ in Beziehung stehen.

Das Ergebnis dieses Vorgehens sind 15 Tabellen in der MySQL Datenbank mit dem Namen cda_27082005. Die Definition der Datenbank und der Tabellen erfolgt dabei mit dem MySQL Administrationstool „phpMyAdmin“. In der Datenbank wurde die Tabelle, welche aus der Entität Untersuchung hervorging umbenannt und trägt jetzt den Namen „fall“.

4.2.4.1 Relationen für CDA Header

Feld	Typ	Null	Standard
pat_id	int(11)	Nein	
vorname	text	Nein	
nachname	text	Nein	
gebDat	date	Nein	0000-00-00
geschlecht	text	Nein	
ts_patient	timestamp	Ja	CURRENT_TIMESTAMP
strasse	text	Nein	
hnr	text	Nein	
ort	text	Nein	
plz	text	Nein	
telecom	text	Nein	

Tabelle 9: Relation patient

Feld	Typ	Null	Standard
fallzahl	int(11)	Nein	
untersuchDat	date	Nein	0000-00-00
pat_id	int(11)	Nein	0
aerzte_id	int(11)	Nein	0
fall_ts	timestamp	Ja	CURRENT_TIMESTAMP

Tabelle 10: Relation fall

Feld	Typ	Null	Standard
aerzte_id	int(11)	Nein	
nachname_a	text	Nein	
vorname_a	text	Nein	
hnr_a	text	Nein	
ort_a	text	Nein	
plz_a	text	Nein	
strasse_a	text	Nein	
telecom_a_tel	text	Nein	
telecom_a_fax	text	Nein	
telecom_a_mailto	text	Nein	
ts_a	timestamp	Ja	CURRENT_TIMESTAMP

Tabelle 11: Relation aerzte

4.2.4.2 Relationen CDA Body

Feld	Typ	Null	Standard
anamnPraedisp_id	int(11)	Nein	
fallzahl	int(11)	Nein	0
anamnPraedisp_ts	timestamp	Ja	CURRENT_TIMESTAMP
kh_beginn	varchar(255)	Nein	
kh_dauer	varchar(255)	Nein	
famBel	varchar(255)	Nein	
beruf	varchar(255)	Nein	
nSchwang	varchar(255)	Nein	
ovulHemmer	varchar(255)	Nein	
antikoag	text	Ja	NULL

Tabelle 12: Relation anamnese01 (Prädisposition)

Feld	Typ	Null	Standard
anamnese02_id	int(11)	Nein	
anamnese02_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
znErysipel	varchar(255)	Ja	NULL
znThrombophl	varchar(255)	Ja	NULL
znPhlebothromb	varchar(255)	Ja	NULL
znKomprTh	varchar(255)	Ja	NULL
znVarOP	varchar(255)	Ja	NULL
znVarOPtxt	text	Ja	NULL
znLuEmbolie	text	Ja	NULL

Tabelle 13: Relation anamnese 02 (Vorerkrankungen)

Feld	Typ	Null	Standard
anamnese03_id	int(11)	Nein	
anamnese03_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
schmerzVar	varchar(255)	Ja	NULL
muedigSchwere	varchar(255)	Ja	NULL
wadenkraempfe	varchar(255)	Ja	NULL
juckreiz	varchar(255)	Ja	NULL
beinschwellung	varchar(255)	Ja	NULL

Tabelle 14: Relation anamnese 03 (aktuelle Beschwerden)

Feld	Typ	Null	Standard
klinVenoer_id	int(11)	Nein	
klinVenoer_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
giacomini	varchar(255)	Ja	NULL
vsm_anas_vsp	varchar(255)	Ja	NULL
aeste_OS	varchar(255)	Ja	NULL
aeste_US	varchar(255)	Ja	NULL
fussVar	varchar(255)	Ja	NULL
perinealVar	varchar(255)	Ja	NULL
retVar_OS	varchar(255)	Ja	NULL
retVar_US	varchar(255)	Ja	NULL
besen_OS	varchar(255)	Ja	NULL
besen_US	varchar(255)	Ja	NULL

Tabelle 15: Relation klinVenoer (klinisch venöser Befund)

Feld	Typ	Null	Standard
CW_vsm_id	int(11)	Nein	
CW_vsm_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
crosse_vsm	varchar(255)	Ja	NULL
crosse_vsm_rez	varchar(255)	Ja	NULL
vsmStamm_proxOS	varchar(255)	Ja	NULL
vsmStamm_distOS	varchar(255)	Ja	NULL
vsmStamm_proxUS	varchar(255)	Ja	NULL
vsmStamm_distUS	varchar(255)	Ja	NULL

Tabelle 16: Relation CW_vsm (CW Doppler der Vena saphena magna)

Feld	Typ	Null	Standard
klinPerf_id	int(11)	Nein	
klinPerf_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
perf_c1	varchar(255)	Ja	NULL
perf_c2	varchar(255)	Ja	NULL
perf_c3	varchar(255)	Ja	NULL
perf_c4	varchar(255)	Ja	NULL
boyd	varchar(255)	Ja	NULL
dodd	varchar(255)	Ja	NULL
bassi_8cm	varchar(255)	Ja	NULL
perfLat_12cm	varchar(255)	Ja	NULL

Tabelle 17: Relation klinPerf (klinischer Befund Perforansvenen)

Feld	Typ	Null	Standard
CW_vsp_id	int(11)	Nein	
CW_vsp_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
crosse_vsp	varchar(255)	Ja	NULL
crosse_vsp_rez	varchar(255)	Ja	NULL
crosse_vsp_prox	varchar(255)	Ja	NULL
crosse_vsp_mittl	varchar(255)	Ja	NULL

Tabelle 18: Relation CW_vsp (W Doppler Vena saphena parva)

Feld	Typ	Null	Standard
LRR_id	int(11)	Nein	
LRR_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
LRR_norm	varchar(255)	Ja	NULL
LRR_path	varchar(255)	Ja	NULL
LRR_kompr_norm	varchar(255)	Ja	NULL
LRR_kompr_path	varchar(255)	Ja	NULL

Tabelle 19: Relation LRR (Licht- Reflexions Rheographie)

Feld	Typ	Null	Standard
abschluss_id	int(11)	Nein	
abschluss_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
notiz	text	Ja	NULL

Tabelle 20: Relation abschluss (Abschlussnotizen)

Feld	Typ	Null	Standard
diagn_einfach_id	int(11)	Nein	
diagn_einfach_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
diagnose_01	varchar(255)	Ja	NULL
icd_diagnose_01	varchar(255)	Ja	NULL
diagnose_02	varchar(255)	Ja	NULL
icd_diagnose_02	varchar(255)	Ja	NULL
diagnose_03	varchar(255)	Ja	NULL
icd_diagnose_03	varchar(255)	Ja	NULL
diagnose_04	varchar(255)	Ja	NULL
icd_diagnose_04	varchar(255)	Ja	NULL

Tabelle 21: Relation diagn_einfach (Hauptdiagnosen)

Feld	Typ	Null	Standard
diagn_einfach_weit_id	int(11)	Nein	
diagn_einfach_weit_ts	timestamp	Ja	CURRENT_TIMESTAMP
fallzahl	int(11)	Nein	0
ndiagn_01	varchar(255)	Ja	NULL
icd_ndiagn_01	varchar(255)	Ja	NULL
ndiagn_02	varchar(255)	Ja	NULL
icd_ndiagn_02	varchar(255)	Ja	NULL
ndiagn_03	varchar(255)	Ja	NULL
icd_ndiagn_03	varchar(255)	Ja	NULL
ndiagn_04	varchar(255)	Ja	NULL
icd_ndiagn_04	char(2)	Ja	NULL

Tabelle 22: Relation diagn_einfach_weit (Nebendiagnosen)

4.3 Programmierung (HTML und PHP)

Der Programmablauf der entwickelten Webapplikation lässt sich in drei Abschnitte gliedern:

Zunächst wird ermittelt ob ein Patient und ein weiterbehandelnder Arzt bereits in der Datenbank erfasst sind. Abhängig davon werden die Daten der Personen in die Datenbank neu eingegeben oder die entsprechenden Referenzen der gesuchten Personen ermittelt und nach Eingabe des Datums für diese Untersuchung eine neue eindeutige Nummer für die aktuelle Untersuchung und den daraus hervorgehenden Befund generiert.

Im nächsten Schritt erfolgt die Dokumentation der bei der Untersuchung erhobenen Befunde. (Anamnese, Inspektion und klinischer Untersuchungsbefund der Venen und Arterien, Ultraschallbefund und Licht- Reflexions- Rheographie, Abschlussnotizen und Diagnosen). Nach Abschluss dieses Schrittes sind alle Daten zu den beteiligten Personen und alle erhobenen Befunde dieser Untersuchung in einer MySQL Datenbank gespeichert und stehen für eine statistische Auswertung zur Verfügung.

Im dritten Schritt erfolgt zunächst eine Datenbankabfrage über alle für die durchgeführte Untersuchung gewünschten Daten. Diese stehen als Ergebnis der Datenbankabfrage in mehreren Arrays zur Verfügung. Diese in Arrays zur Verfügung stehenden Daten werden dann in einem CDA Template Dokument durch einen „Suchen und Ersetzen“ Mechanismus an den jeweils passenden Stellen im Vorlage Dokument eingesetzt, sodass als Ergebnis ein CDA gültiges Instanzdokument entsteht, das die Daten der aktuellen Untersuchung beinhaltet.

Tabelle 23 gibt einen Überblick über die verschiedenen Dateien der entwickelten Anwendung und ihre Aufgaben:

Dateiname	Beschreibung
suchePatArzt.php	Patient und weiterbehandelnder Arzt bereits in DB. gespeichert?
untersuchDat.php	Eingabe Untersuchungsdatum
arztUntersuchDat.php	Eingabe Untersuchungsdatum und Daten des weiterbehandelnder Arztes
patUntersuchDat.php	Eingabe Untersuchungsdatum und Daten des Patienten
patArztUntersuchDat.php	Eingabe der Daten des Patienten, des weiterbehandelnden Arztes und des Untersuchungsdatum
cda_b_anamnPraedisposition.php	Eingabe Anamnese: Prädisposition
cda_b_anamnVorerkrankung.php	Eingabe Anamnese: Vorerkrankungen
cda_b_anamnBeschwerden.php	Eingabe Anamnese: aktuelle Beschwerden
cda_b_arteriell.php	Eingabe arterieller Befund und weitere Befunde
cda_b_klinVenoese.php	Eingabe klinisch venöser Befund
cda_b_klinPerf.php	Eingabe klinischer Befund der Perforansvenen
cda_b_CW_vsm.php	Eingabe CW Dopplerbefund der Vena saphena magna
cda_b_CW_vsp.php	Eingabe CW Dopplerbefund der Vena saphena parva
cda_b_LRR.php	Eingabe des Befundes bei der Licht Reflexions Rheographie
cda_b_diagn_einfach.php	Eingabe der Hauptdiagnosen
cda_b_diagn_einfach_weitere.php	Eingabe der Nebendiagnosen
cda_b_abschluss.php	Eingabe Abschlussnotizen
cda_show_var.php	Ausgabe des Abfrageergebnisses aller eingegeben Daten einer Untersuchung
gen_cda_h_b.php	Generierung des CDA Level 1 schemagültigen Befundbereiches
gen_cda_h_b_style01.php	Verbindung des generierten XML Dokumentes mit einem Stylesheet zur Ausgabe in einem Webbrowser

Tabelle 23: Codeaufbau

4.3.1 Datenerfassung und Speicherung in RDBS

Die entwickelte Webapplikation besteht insgesamt aus 20 verschiedenen Dateien, von denen die meisten einen ähnlichen Aufbau aufweisen. Zunächst wird eine Anbindung zur Datenbank hergestellt. Danach werden Daten erfasst und durch ein entsprechendes php-Skript die Formulardaten verarbeitet und in der MySQL Datenbank gespeichert. Über einen Hyperlink wird die nächste Datei geöffnet, bis alle Untersuchungsbefunde in der Datenbank gespeichert sind.

An dieser Stelle soll eines der verwendeten Webformulare zur Eingabe der erhobenen Daten abgebildet werden. Die verwendeten Formulare sind prinzipiell alle ähnlich aufgebaut. Auf ein besonderes Erscheinungsbild der Formulare wurde in dieser Arbeit kein Wert gelegt, da es sich hierbei nur um eine prototypische Implementierung handelt und die Schwerpunkte dieser Arbeit nicht auf der Formulargestaltung liegen.

Anamnese Formular 1:Prädisposition

Krankheitsbeginn:
 vor 30.Lebensjahr nach 30.Lebensjahr

Krankheitsdauer:
 kürzer als 5 Jahre länger als 5 Jahre

familiäre Belastung:
 Mutter Vater keine familiäre Belastung

berufliche Tätigkeit vorwiegend:

Schwangerschaften:

Dauer Einnahme von Ovulationshemmern (Jahre):

Einnahme von Antikoagulationen(Substanz, Einnahmedauer, Indikation etc. oder Feld frei lassen)

[Weiter: Anamneseformular 2](#)

Abbildung 6: HTML Formular zur Dateneingabe

Die Datei `connect_to_DB.php` ist für die Herstellung einer Datenbankverbindung zuständig. Der Code dieser Datei wird über die php-Funktion `include ()` in den einzelnen Dateien verfügbar gemacht. In `connect_to_DB.php` werden zunächst die Variablen `$mysqlhost`, `$mysqluser`, `$mysqlpasswd` und die Variable `$mysqldbname` mit den entsprechenden Werten belegt. Diese müssen dann als Parameter an die Funktion `mysql_connect ()` übergeben werden. Falls die Verbindung nicht hergestellt werden kann wird eine entsprechende Fehlermeldung ausgegeben. Außerdem wird in `connect_to_DB` die Funktion `array_item` definiert, deren Aufgabe bereits im Abschnitt 3.4.2 beschrieben wurde.

```
<?php
//Aufruf PHP Prozessor Datenbankbindung
  $mysqlhost="localhost";
  $mysqluser="root";
  $mysqlpasswd="passwort";
  $mysqldbname="cda_27082005";

$link =
  mysql_connect($mysqlhost, $mysqluser, $mysqlpasswd);
if ($link == FALSE) {
  echo "<p><b>Leider kann keine Verbindung zur Datenbank
    hergestellt werden. Daher können die Ergebnisse
    zur Zeit nicht angezeigt werden. Bitte versuchen
    Sie es später noch einmal.</b></p>\n";
  exit();
}

//Datenbank auswählen
mysql_select_db($mysqldbname);

//Function array_item
function array_item($ar, $key) {
  if(array_key_exists($key, $ar) return($ar[$key]);
  return(''); }
?>
```

Codelisting 6: "connect_to_DB.php"

Die Anwendung beginnt mit der Datei `suchePatArzt.php`, in der ermittelt wird, ob die Daten des Patienten und die Daten des weiterbehandelnden Arztes bereits in der MySQL Datenbank abgelegt sind oder erstmals im Rahmen der aktuellen Untersuchung zu erfassen und zu speichern sind. Vier Fälle können hierbei unterschieden werden:

Die Daten des Patienten und des weiterbehandelnden Arztes sind bereits gespeichert, ein Hyperlink führt weiter zur Eingabe des Datums der aktuellen Untersuchung.

Die Daten beider Personen sind noch nicht gespeichert, ein Hyperlink öffnet ein neues Fenster, indem die Daten der beiden Personen und das Datum der Untersuchung eingegeben werden. Hier wird direkt in das neue Formular php- Code eingebettet und mit Hilfe von Session- Variablen die zuvor im Suchformular eingegebenen Daten der zu suchenden Personen wieder ausgegeben. Das erspart erneutes Eingeben der Daten und verhindert eine fehlerhafte Eingabe.

Es verbleiben jetzt noch die Möglichkeiten, dass entweder noch die Daten des weiterbehandelnden Arztes oder die des Patienten neu in die Datenbank eingegeben werden müssen. Auch hier kommen wieder Session- Variablen in der oben beschriebenen Art und Weise zum Einsatz.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<title>Patient und Arzt bereits in DB gespeichert?</title>
</head>
<body>
<form method="post" action="suchePatArzt.php">
Patient:<br>
<!--Suchformular: Daten von Patient und weiterbehandelndem Arzt-->
Weiterbehandelnder Arzt:<br>
<input type="submit" name="submitbutton" value="Suche starten">
</form>
<!--Aufruf PHP Prozessor Datenbankbindung-->
<?php
//Session Variablen
session_start();
//Datei connect_to_DB einbeziehen; DB Verbinden aufbauen
//und definition der Funktion array_item
include("connect_to_DB.php");
    //Variablenzuweisung
    $submitbutton = array_item($_POST, 'submitbutton');
    if($submitbutton=="Suche starten") {
        $vorname = array_item($_POST, 'vorname');
/*nachname, gebDat, usw.*/
        $_SESSION['vorname'] = $_POST['vorname'];
/* auch als SESSION Variablen verfügbar machen*/
        //Suche Patatient in DB
        $result_pat= mysql_query("Select pat_id,vorname,nachname,gebDat".
        " FROM patient WHERE vorname = '$vorname' AND ".
        "nachname = '$nachname' AND gebDat = '$gebDat'");
        $result_pat = mysql_fetch_array($result_pat);
        $pat_id = $result_pat['pat_id'];
        //var_dump($result_pat);
        //Suche Arzt in DB
        /*analoge Abfrage auch für Daten des weiterbehandelnden Arztes
        //Vier Möglichkeiten:
        //Fall: Weiterbehandelnder Arzt und Patient bereits in DB gespeichert
        if($result_pat== TRUE AND $result_arzt == TRUE){
            echo "Pat bereits in Datenbank gespeichert;pat_id ist:
$pat_id<br>\n";
            echo "Arzt bereits in Datenbank gespeichert;aerzte_id ist:
$aerzte_id</p>\n";
            echo "Bitte folgen Sie dem Hyperlink und geben Sie das Datum
der heutigen Untersuchung an</p>";
            echo "<a href=untersuchDat.php> Eingabe Untersuchungsdatum
</a>";
            //var_dump($_SESSION); }
        //Fall:Patient bereits in DB gespeichert, Arzt nicht in DB
gespeichert
        if($result_pat == TRUE AND $result_arzt == FALSE){...}
        //Fall:Patient ist noch nicht in DB gespeichert, weiterbehandelnder
Arzt bereits in DB gespeichert
        if($result_pat == FALSE AND $result_arzt == TRUE){...}
        //Fall:Patient noch nicht in DB erfasst, Arzt noch nicht in DB
erfasst
        if($result_pat == FALSE AND $result_arzt == FALSE){...}}
        ?></body></html>

```

Codelisting 7: Programmablauf in Datei „suchePatArzt.php“

Nach Eingabe der Patientendaten sowie der Daten des weiterbehandelnden Arztes und des Datums der aktuellen Untersuchung folgt die Eingabe des eigentlichen Befundes. Dabei ist der Primärschlüssel („fallzahl“) der Tabelle „fall“ (entspricht der Entität „Untersuchung“ im ER Modell) Fremdschlüssel in allen weiteren Tabellen, die die eigentlichen Untersuchungsbefunde enthalten. Jeder Untersuchung ist somit genau ein Untersuchungsbefund zugeordnet.

Der Programmaufbau der einzelnen Dateien ist bis auf die Datei `gen_cda_h_b.php` immer der Gleiche. Von den in Abschnitt 3.3.1 beschriebenen HTML Formulartypen kommen hauptsächlich Radiobuttons, Texteingabefelder und Auswahllisten zum Einsatz. Nach Ausfüllen der einzelnen Felder der Formulare werden die Daten durch Klick auf den `submitbutton` mit der Methode `POST` an den Server abgeschickt. Es wird ein SQL-Kommando ausgeführt, um die Fallzahl der aktuellen Untersuchung zu ermitteln. Dabei kann das entsprechende Tupel aus der Tabelle „fall“(fallzahl, pat_id, aerzte_id, untersuchDat) entweder durch Verwendung der `SESSION` Variablen für Patient und weiterbehandelnden Arzt selektiert werden, oder indem einfach das letzte in die Tabelle `fall` eingetragene Tupel ausgewählt wird. Ein weiteres SQL Kommando fügt die in die Formulare eingegebenen Daten zusammen mit der ermittelten Fallzahl in die entsprechende Tabelle ein. Für die Entwicklung der Webapplikation hat sich dabei die Einführung eines Feldes vom Typ „timestamp“ in den verschiedenen Tabellen der Datenbank bewährt; dadurch wird für jedes erfolgreich eingefügte Tupel automatisch Datum und Zeit gespeichert. Wenn die erhobenen Befunde gespeichert sind führt am Ende jedes Formulars ein Hyperlink zur Eingabe des nächsten Befundes. Codelisting zeigt exemplarisch den beschriebenen Programmablauf.

Codelisting 8 zeigt den beschriebenen Programmablauf.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head><title>Anamnese:Prädisposition</title></head><body>
<h4>Anamnese Formular 1:Prädisposition</h4>
<table border="1" cellspacing="" cellpadding=""> <colgroup> <col
width="350"><col width="550"></colgroup>
<form method="post" name="anamnPraedis" accept-charset="ISO-8859-1"
action="cda_b_anamnPraedis.php">
<tr><td>
Krankheitsbeginn:<br>
<input type="radio" name="kh_beginn" value="kh_beginn_lt30">vor
30.Lebensjahr
<input type="radio" name="kh_beginn" value="kh_beginn_gt30"> nach
30.Lebensjahr
</p>
<!--Krankheitsdauer:<br>...-->
<!--familiäre Belastung:<br>...-->
<!--berufliche Tätigkeit vorwiegend:...<br>-->
<!--Schwangerschaften:<br>.., OH Einnahme, Antikoagulantien-->
<input type="submit" name="submitbutton" value="in DB eintragen"></form>
</p>
<!--Aufruf PHP Prozessor -->
<?php
session_start();

//Datei connect_to_DB einbeziehen; DB Verbinden aufbauen
//und definition der Funktion array_item
include("connect_to_DB.php");

//Variablenzuweisung
$submitbutton = array_item($_POST, 'submitbutton');
$kh_beginn = array_item($_POST, 'kh_beginn');
$kh_dauer = array_item($_POST, 'kh_dauer');
$famBel = array_item($_POST, 'famBel');
$beruf = array_item($_POST, 'beruf');
$nSchwang = array_item($_POST, 'nSchwang');
$ovulHemmer = array_item($_POST, 'ovulHemmer');
$antikoag = 'AK-Th: ';
$antikoag .= array_item($_POST, 'antikoag');

//fallzahl stellt Referenz zwischen Fall und konkretem Untersuchungsbefund dar
//ermitteln über DB Abfrage
//fallzahl aus letztem in fall eingetragenen Tupel ermitteln
$result = mysql_query("SELECT fallzahl FROM fall ORDER BY fall_ts DESC
LIMIT 1");
$fallzahl = mysql_fetch_array($result);
$fallzahl = $fallzahl['fallzahl'];
echo $fallzahl;

if($submitbutton=="in DB eintragen") {

mysql_query("INSERT INTO
anamnese01(fallzahl,kh_beginn,kh_dauer,famBel,beruf".
" ,nSchwang,ovulHemmer,antikoag)VALUES
('$fallzahl','$kh_beginn','$kh_dauer','$famBel','$beruf','$nSchwang',".
"'$ovulHemmer','$antikoag' )");

};

```

Codelisting 8: „cda_b_anamnVorerkrank.php“

Nach diesem Schema erfolgt auch die Erfassung und Speicherung aller anderen im Rahmen einer Untersuchung erhobenen Befunde.

4.3.2 Aus relationalen Daten XML generieren

Die Generierung eines CDA gültigen XML Dokumentes erfolgt in der Datei `gen_cda_h_b.php`. Neben den im Abschnitt 3.4.4 bereits beschrieben php- Funktionen für den Zugriff auf die MySQL Datenbank sind in dieser Datei zwei php- Stringfunktionen von besonderer Bedeutung.

- `implode()`: Die Funktion baut einzelne Array Elemente zu einem String zusammen. Sie erwartet zwei Parameter nämlich ein Trennzeichen und Elemente eines Arrays.
- `str_replace()`: Die Funktion sucht in einer Zeichenkette nach allen Vorkommen eines bestimmten Strings und ersetzt diesen durch einen anderen String. Demnach müssen der Funktion drei Parameter übergeben werden, nämlich die zu durchsuchende Zeichenkette, der String, der ersetzt werden soll und der ersetzende String.

Der Programmablauf dieser Datei lässt sich nun in drei Schritte gliedern.

- Zunächst werden alle relevanten Daten aus der Datenbank mit den entsprechenden SQL Kommandos abgefragt. Als Ergebnis dieser Abfragen stehen die Daten in Arrays zur Verfügung und können über ihren numerischen Index angesprochen werden.
- Als nächstes wird mit der Funktion `implode (..$template_cda_h = implode("", file("cda_h_b.xml"));..)` ein CDA Instanzdokument geladen, welches im `htdocs` Verzeichnis des Webservers gespeichert ist. Dieses XML Dokument enthält Platzhalter an genau den Positionen, an denen durch Einsatz der Funktion `str_replace () ($template_cda_h = str_replace("plzh_vorname", $result_patient[1], $template_cda_h);)` die Platzhalter durch die Daten aus der Datenbankabfrage ersetzt werden sollen.

Das verwendete CDA Template ist ein CDA Dokument, das zu Demonstrationszwecken frei erhältlich war. (unter www.hl7.org, Stand Dezember 2004) ist. Um es für diese Arbeit zu nützen wurden einige Modifikationen daran durchgeführt.

- Es werden Strings als Platzhalter eingefügt, die später durch die jeweiligen Daten ersetzt werden. Die Tabellen 4,5 und 6 geben hierbei einen guten Überblick.

- OID Codes werden übernommen, lediglich die Werte bestimmter Extension Attribute werden durch Platzhalter ersetzt, die wiederum später durch Zahlenwerte aus der Datenbank ersetzt werden. (→2.2.3.4, 6.1)
- Der gesamte Body Bereich, der die eigentliche klinische Information aufnimmt wird konsequent in ein Konstrukt bestehend aus den CDA Elementen `section`, `caption`, `list`, `item`, `content` gegliedert und mit entsprechenden Platzhaltern belegt. (→2.2.3.3)

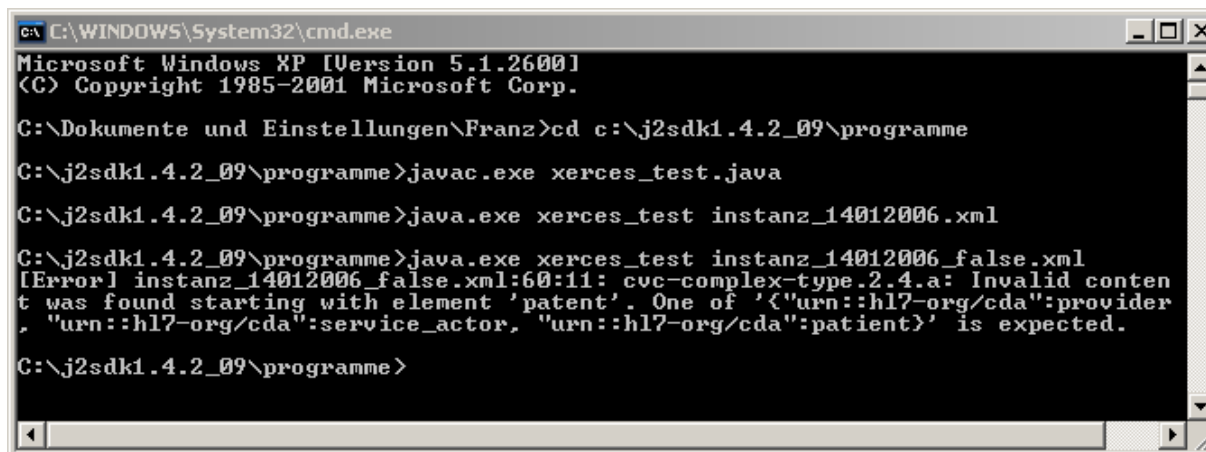
```
<section>
  <caption>Diagnosen</caption>
  <list>
    <item>
      <content>plzh_diagn_01</content>
    </item>
    <item>
      <content>plzh_icd_diagn_01</content>
    </item>
    <item>
      <content>plzh_diagn_02</content>
    </item>
    <item>
      <content>plzh_icd_diagn_02</content>
    </item>
    <item>
      <content>plzh_diagn_03</content>
    </item>
    <item>
      <content>plzh_icd_diagn_03</content>
    </item>
    <item>
      <content>plzh_diagn_04</content>
    </item>
    <item>
      <content>plzh_icd_diagn_04</content>
    </item>
  </list>
</section>
```

Abbildung 7: Ausschnitt aus Body Teil des CDA Templates

Im letzten Schritt wird das Template, nachdem alle Platzhalter durch die Daten der aktuellen Untersuchung ersetzt worden sind, einer Variablen `$output` zugewiesen und nach Ausgabe des XML Headers (`header("Content-type: text/xml");`) mit der php- Anweisung `print` zur Ausgabe gebracht. Im Webbrowser wird nun das CDA Instanzdokument angezeigt, vorausgesetzt es ist wohlgeformt.

Ob das generierte Dokument auch schemagültig ist kann mit dem validierenden Parser Xerces an der Kommandozeile geprüft werden.

Abbildung 8 zeigt Schemavalidierung des generierten Instanzdokumentes mit dem validierenden Parser Xerces, welcher an der Kommandozeile aufgerufen wird.



```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Dokumente und Einstellungen\Franz>cd c:\j2sdk1.4.2_09\programme
C:\j2sdk1.4.2_09\programme>javac.exe xerces_test.java
C:\j2sdk1.4.2_09\programme>java.exe xerces_test instanz_14012006.xml
C:\j2sdk1.4.2_09\programme>java.exe xerces_test instanz_14012006_false.xml
[Error] instanz_14012006_false.xml:60:11: cvc-complex-type.2.4.a: Invalid content
was found starting with element 'patent'. One of '<urn::hl7-org/cda':provider
, 'urn::hl7-org/cda':service_actor, 'urn::hl7-org/cda':patient}' is expected.
C:\j2sdk1.4.2_09\programme>
```

Abbildung 8: Schema Validierung mit Parser Xerces

Das gegen das Schema zu validierende Instanzdokument wird dem Parser an der Kommandozeile als Parameter übergeben. (schemagültig ist Datei: instanz_14012006.xml).

In der Datei instanz_14012006_false.xml wurde das Element `<patient>` umbenannt in „`<patent>`“. Abbildung 3 zeigt die daraus resultierende Fehlermeldung des Parser.

Um das generierte XML Dokument mit Layoutanweisungen zu versehen muss ein so genanntes CDA Stylesheet programmiert werden (Abschnitt 4.4). Instanzdokument und Stylesheet werden miteinander verbunden, indem im Instanzdokument eine Verarbeitungsanweisung an einen Parser mit einem Link auf das entsprechende Stylesheet eingefügt wird. (`<?xml-stylesheet href="cda_stylesheet.xsl" type="text/xsl" ?>`)

4.4 Programmierung eines XSL Stylesheets

Das aus relationalen Daten generierte XML Instanzdokument soll nun durch Transformation ins HTML Format für das menschliche Auge gut lesbar und übersichtlich in einem Webbrowser dargestellt werden.

Dabei soll der gesamte Befundbericht eine Länge von maximalen drei Seiten nicht überschreiten. Auf der ersten Seite sollen übersichtlich die Informationen zum Patienten, dem Untersucher, der Untersuchung und dem weiterbehandelnden Arzt dargestellt werden. Seite zwei und drei sollen den erhobenen Untersuchungsbefund wiedergeben.

Ausgangspunkt für die die Auswahl der zu dokumentierenden Merkmale und der Präsentation der erhobenen Befunde in einem zusammenfassenden Bericht sind Formulare wie sie in einem Tiroler Krankenhaus zur Dokumentation auch verwendet werden.

Anamnese:	Re	Li		
Beginn der Varikose Alter <30a	<input type="checkbox"/>	<input type="checkbox"/>	familiäre Belastung väterlich.	<input type="checkbox"/>
Beginn der Varikose Alter >30a	<input type="checkbox"/>	<input type="checkbox"/>	familiäre Belastung mütterlich	<input type="checkbox"/>
Dauer unter 5a	<input type="checkbox"/>	<input type="checkbox"/>	Beruf stehend	<input type="checkbox"/>
Dauer über 5a	<input type="checkbox"/>	<input type="checkbox"/>	Schwangerschaft –eine	<input type="checkbox"/>
beschwerdefrei	<input type="checkbox"/>	<input type="checkbox"/>	Schwangerschaft mehrere	2
kosmetisch störende Varizen	<input type="checkbox"/>	<input type="checkbox"/>	Weibliche Geschlechtshomone	
Schweregefühl der Beine	<input type="checkbox"/>	<input type="checkbox"/>	Nikotin	<10 / Tag
diffuse Beinschmerzen	<input type="checkbox"/>	<input type="checkbox"/>	Antikoagulantien	<input type="checkbox"/> nein
schmerzhafte Varizen	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/> Cumarine / Heparin
nächtliche Wadenkrämpfe	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/> ASS
Juckreiz der Beine	<input type="checkbox"/>	<input type="checkbox"/>		
Z.n. Kompressionsverband	<input type="checkbox"/>	<input type="checkbox"/>		
„geschwollene Beine“	<input type="checkbox"/>	<input type="checkbox"/>		
Ekzem	<input type="checkbox"/>	<input type="checkbox"/>		
Ulcus cruris	<input type="checkbox"/>	<input type="checkbox"/>		
Z.n. Ulcus cruris	<input type="checkbox"/>	<input type="checkbox"/>		
Z.n. Erysipel	<input type="checkbox"/>	<input type="checkbox"/>		
Z.n. Thrombophlebitis	<input type="checkbox"/>	<input type="checkbox"/>		
Z.n. Phlebothrombose	<input type="checkbox"/>	<input type="checkbox"/>		
Z.n. Lungenembolie	<input type="checkbox"/>	<input type="checkbox"/>		
Z.n. Sklerotherapie	<input type="checkbox"/>	<input type="checkbox"/>		
Z.n. Varizen OP	<input type="checkbox"/>	<input type="checkbox"/>		
			Klinischer Status:(venös, art. Sonstiges)	
			<u>Venös:</u>	Re Li
			VSM-palpabel(Stammvar.)	<input type="checkbox"/>
			VSM-Perkuss.test-positiv	<input type="checkbox"/>
			VS Accesoria med.	<input type="checkbox"/>
			VS Accesoria lat.(Semicirc)	<input type="checkbox"/>
			VSP-palpabel(Stammvar.)	<input type="checkbox"/>
			V. femoropoplitea(Giacomini)	<input type="checkbox"/>
			VSM- VSP Anastomose	<input type="checkbox"/>
			Seitenäste OS	<input type="checkbox"/>

Abbildung 9: Formular zur Varizen Dokumentation

Bei der Programmierung des Stylesheets wurde versucht dieses Layout für die Präsentation des Befundes beizubehalten.

Folgende bereits im Kapitel Methoden beschriebene Techniken werden dabei verwendet.

- Tabellen als Mittel für Webseiten Layout
- XPath Syntax und XPath Funktionen: starts-with(), string-length(), substring()
- XSLT Elemente: xsl:stylesheet, xsl:output, xsl:template, xsl:apply-template, xsl:value-of, xsl:if, xsl:text, xsl:for-each

Nach der XML Deklaration wird das Stylesheet mit dem Element `xsl:stylesheet` eingeleitet, indem auch die Deklaration der Namensräume erfolgt. In diesem Stylesheet werden Elemente aus zwei verschiedenen XML- Anwendungen, nämlich der Anwendung XSLT und der Anwendung CDA im gleichen Dokument, verwendet. Um Konflikte zu vermeiden, müssen daher zwei verschiedene Namensräume definiert werden. Im weiteren Verlauf des Stylesheets muss einem XSLT Prozessor auch mitgeteilt werden, welches Element zu welcher Anwendung gehört. Dies geschieht, indem die verschiedenen Elemente mit dem jeweiligen Namensraum Präfix, welche im einleitenden `xsl:stylesheet`- Element den URI s zugeordnet worden sind, versehen werden.

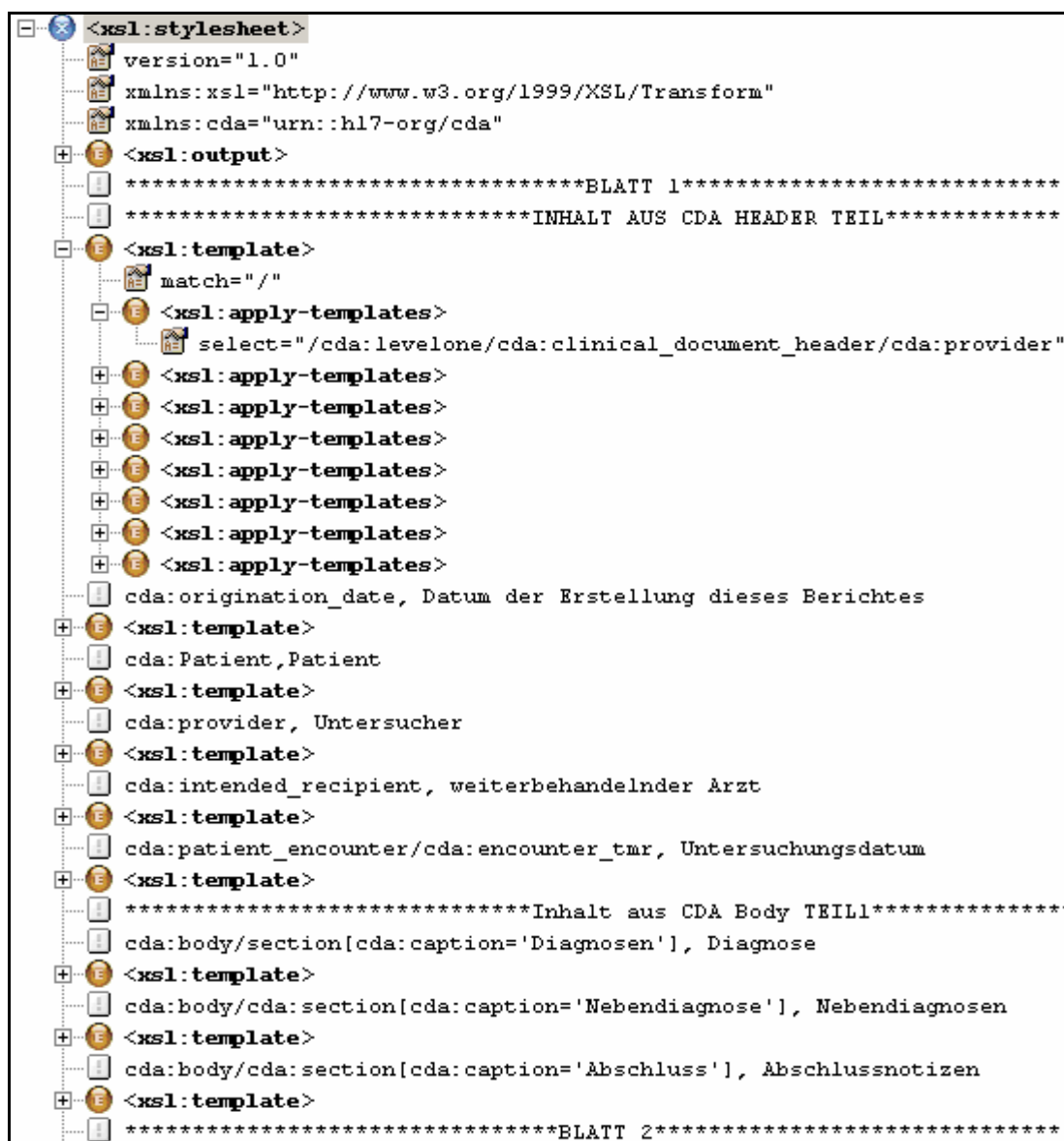


Abbildung 10: Baumansicht XSL Stylesheet

Das vorliegende Stylesheet besteht aus zehn Templates. In Abbildung xxx zeigt einen Ausschnitt des Stylesheets in der Baumansicht. In Form der Kommentare im Stylesheet ist ersichtlich, welche Bereiche im Quellbaum zu welchen Bereichen im Ergebnisbaum transformiert werden.

Das erste Template bezieht sich auf das gesamte Instanzdokument, was durch die XPath Syntax `<xsl:template match="/">` ausgedrückt wird. Innerhalb dieses ersten Template Elementes werden mehrere Elemente namens `xsl:apply-templates` verwendet. Wenn in einem XSL Stylesheet der Quellbaum durch mehrere Templates in den Ergebnisbaum transformiert wird, so lässt sich durch das Konstrukt wie es im ersten Template realisiert ist, die Reihenfolge der Anwendung der in weiterer Folge weiter unten im Stylesheet definierten Templates auf den Quellbaum steuern. Die Reihenfolge ist dann genau so festgelegt, wie die einzelnen `xsl:apply-templates` Elemente in dem Template erscheinen, das auf den gesamten Quellbaum angewendet wird.

Anhand der in Abschnitt 3.5 dargestellten Grundlagen soll im folgenden Abschnitt an einem bestimmten Template exemplarisch gezeigt werden wie XPath und XSLT Elemente in einem Stylesheet kombiniert werden, um das generierte CDA Instanzdokument ins HTML Format zu transformieren und in einem gewünschten Layout im Webbrowser zur Ansicht zu bringen. Die Transformation des folgenden Knoten des generierten Instanzdokumentes soll näher gezeigt werden:

```
/cda:levelone/cda:clinical_document_header/cda:intended_recipient:
```

- Umfasst die Daten des weiterbehandelnden Arztes, also Namen, Adresse, Telefonnummer, Fax sowie Mail Adresse und Homepage. Erscheint im resultierenden Befundbericht auf der ersten Seite gemeinsam mit Daten des Untersuchers, Daten des Patienten sowie Untersuchungsdatum, Diagnosen und Nebendiagnosen und den Abschlussnotizen mit einer Zusammenfassung der Ergebnisse, Therapieempfehlung und gegebenenfalls weiteren Terminen.

```

<!--cda:intended_recipient, weiterbehandelnder Arzt-->
<xsl:template match="/cda:levelone/cda:clinical_document_header/
                    cda:intended_recipient">
  <table border="0" cellspacing="1" cellpadding="1">
    <colgroup>
      <col width="500"></col>
      <col width="200"></col>
      <col width="300"></col>
    </colgroup>

    <tr><td><b>An:</b></td><td></td></tr>
    <tr><td><xsl:value-of select=
      "cda:person/cda:person_name/cda:nm/cda:PFX/@V" />
      <xsl:text> </xsl:text>
      <xsl:value-of select=
      "cda:person/cda:person_name/cda:nm/cda:GIV/@V" />
      <xsl:text> </xsl:text><xsl:value-of select=
      "cda:person/cda:person_name/cda:nm/cda:FAM/@V" /><br/>
      <xsl:value-of select=
      "cda:person/cda:addr/cda:STR/@V" /><xsl:text>
      </xsl:text><xsl:value-of select=
      "cda:person/cda:addr/cda:HNR/@V" />
      <xsl:text>, </xsl:text><xsl:value-of select=
      "cda:person/cda:addr/cda:ZIP/@V" />
      <xsl:text> </xsl:text><xsl:value-of select=
      "cda:person/cda:addr/cda:CTY/@V" /><br/>

      <xsl:for-each select="cda:person/cda:telecom">
        <xsl:if test = "starts-with(@V,'tel:')">
          Tel.:<xsl:value-of select= "substring(@V,5)" />
        </xsl:if></xsl:for-each><xsl:text>, Fax.:</xsl:text>
        <xsl:for-each select="cda:person/cda:telecom">
          <xsl:if test = "starts-with(@V,'fax:')">
            <xsl:value-of select= "substring(@V,5)" />
          </xsl:if></xsl:for-each>
      <!--</td><td></td><td></td></tr>--><br/>

```

Abbildung 11: Ausschnitt aus XSL Template

Am Anfang des Templates wird ein bestimmter Knoten mit einem XPath Ausdruck adressiert. Dann wird eine HTML Tabelle angelegt, mit deren Hilfe die auszugebenden Daten am Blatt positioniert werden können.

Das XSL Element `value-of` dient dazu den Attributwert des mittels XPath Ausdruck adressierten CDA- Elementes auszugeben. Das XSL Element `text` dient hier dazu Leerzeichen auszugeben.

Bei der Ausgabe von Telefonnummer, Telefaxnummer, E-Mail Adresse und Adresse der Homepage besteht das Problem, dass all diese Daten in CDA Elementen mit dem Namen „telecom“ abgelegt sind. Die Lösung besteht darin bestimmte XPath Funktionen und XSLT Elemente zu kombinieren. Für alle Elemente aus einer Liste wird in einer Schleifenkonstruktion (`xsl:for-each`) mit einer XPath Funktion (`starts-with()`) eine Bedingung getestet, um so die gewünschten Ausgaben zu erzielen.

Das Vorgehen zur Transformation der anderen Teile des Instanzdokumentes mittels Templates ist vom Prinzip her identisch mit dem eben Beschriebenen. Adressierung der gewünschten Knoten mit XPath, Manipulation und Ausgabe der gewünschten Daten mit XPath Funktionen und XSLT Elementen, Positionierung der Ausgabe am Bildschirm mit HTML Tabellen sowie weitere Formatierung mit HTML Tags.

Abbildung 12 zeigt schematisch den Programmablauf der implementierten Webapplikation.

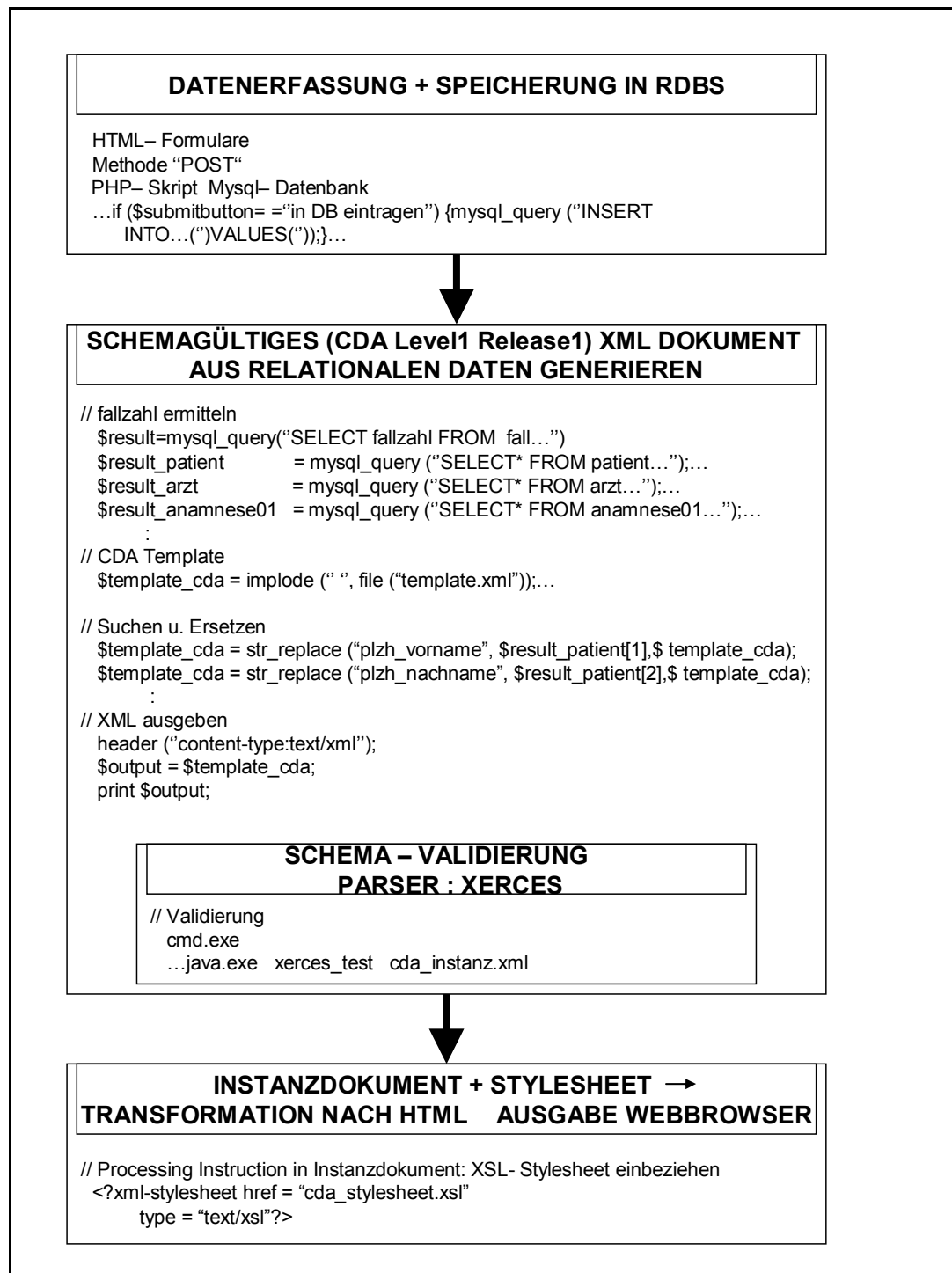


Abbildung 12: Programmablauf schematisch

4.5 Installation und Konfiguration (Win XP)

In den folgenden Abschnitten wird kurz auf die Installation und Konfiguration der für die entwickelte Webapplikation notwendigen Software eingegangen.

4.5.1 Apache Webserver 2.0.54 und PHP 5.0.4

Zur Entwicklung der Webapplikation wird eine Webserver local auf einem Testrechner installiert. Apache steht bei www.apache.org als Installationsprogramm für Windows zur Verfügung. (für diese Arbeit verwendete Version: `apache_2.0.54-win32-x86-no_ssl.msi`).

Die Installation erfolgt nach Doppelklick auf die Installationsdatei automatisch. Im Dialog zur Grundkonfiguration muss ein Domainen Name und ein Server Name eingegeben werden. Nach erfolgter Installation wird Apache unter Windows XP automatisch als Dienst eingerichtet und sofort gestartet. Es erscheint im rechten Teil der Task Leiste ein Icon, über welches der Webserver gestartet und gestoppt werden kann.

Als Test, ob die Installation des Webserver erfolgreich war, kann die Testseite des Webserver in den Webbrowser geladen werden. Diese ist bei erfolgreicher Installation erreichbar unter `http://localhost/` oder `http://127.0.0.1/`. Die Apache Konfigurationsdatei findet sich in dem Verzeichnis `C:\Programme\Apache Group\Apache2\conf`, die entwickelten Webdateien werden im Verzeichnis `C:\Programme\Apache Group\Apache2\htdocs` abgelegt.

Für diese Arbeit wurde von der Website <http://www.php.net/downloads.php> die Zip Datei `php-5.0.4-Win32.zip` zur weiteren Installation verwendet. PHP wird als Erweiterung (Modul) zu Apache installiert. Der Inhalt der Zip Datei wird für diese Arbeit in das neu angelegte Verzeichnis `C:\php5.0.4` extrahiert.

In die Apache Konfigurationsdatei werden nun folgende Zeilen eingefügt: `LoadModule php5_module "c:/php5.0.4/php5apache2.dll"; AddType application/x-httpd-php .php; PHPIniDir "C:/php5.0.4"`. Anschließend muss der Server neu gestartet werden.

Abschließend wird eine Kopie der Datei `php.ini-recommended` erstellt und unter dem Name `php.ini` im Verzeichnis `C:\php5.0.4` abgelegt.

PHP ist nun als Erweiterung zu Apache installiert und kann nun wie folgt getestet werden.

Mit einem (Text)- Editor wird die Datei `phptest.php` mit dem Code `<?php phpinfo(); ?>` erstellt und im `htdocs` Verzeichnis (`C:\Programme\Apache Group\Apache2\htdocs`) angelegt. Anschließend wird die Datei im Webbrowser aufgerufen. Bei erfolgreicher Installation erscheint die PHP Testseite im Browserfenster.

4.5.2 MySQL 4.1.12

Die Installation erfolgt durch Doppelklick auf die Datei `mysql-essential-4.1.12-win32.msi`, welche zuvor von der MySQL Website herunter geladen wurde. Nach der Installation wird automatisch der MySQL Konfigurationsassistent ausgeführt, welcher jederzeit neuerlich gestartet werden kann. Der Assistent richtet die MYSQL Konfigurationsdatei ein und sichert den MySQL Server durch ein Passwort ab.

Um den Erfolg der Installation zu testen, wird der Kommandozeileninterpreter `mysql.exe` aufgerufen. Dabei muss das Passwort eingegeben werden, das während der MySQL Konfiguration verwendet wurde. Bei erfolgreicher Installation erscheint auf das Kommando `status` eine Zusammenfassung der wichtigsten Verbindungsparameter.

4.5.3 phpMyAdmin 2.6.2

Ein Administrationswerkzeug für MySQL ist phpMyAdmin. Damit können Datenbanken und Tabellen erzeugt und verändert und wieder gelöscht werden. Es können damit Datensätze eingefügt, geändert und gelöscht werden sowie Daten bzw. ganze Datenbanken importiert , exportiert sowie weitere Administrationsaufgabe durchgeführt werden.

phpMyAdmin ist ein riesiges PHP Projekt. Die Datei `phpMyAdmin-2.6.2-pl1.zip` wurde von der Website phpMyAdmin Projektes http://www.phpmyadmin.net/home_page/index.php bezogen und dann in das Dokumentenverzeichnis (C:\Programme\Apache Group\Apache2\htdocs) des Webservers extrahiert.

Nun müssen in der Konfigurationsdatei `config.inc.php` einige Einstellungen vorgenommen werden. `$cfg['PmaAbsoluteUri']` muss die Adresse enthalten unter der phpMyAdmin im Netzwerk erreichbar ist. (hier: `$cfg['PmaAbsoluteUri'] = 'http://localhost/phpmyadmin';`). Als nächstes müssen der MYSQL Benutzername und das Passwort für den MySQL Zugang eingetragen werden. (`$cfg['Servers'][$i]['user'] = 'root'; // MySQL user;` und `$cfg['Servers'][$i]['password'] = 'passwort'; // MySQL password`). Hier wurde die einfachste Form der MySQL Authentifizierung verwendet. In der php Konfigurationsdatei `php.ini` muss die Windows Erweiterung `php_mbstring.dll` aktiviert werden, da der Standard Zeichensatz in PHP latin1 ist. Außerdem muss die benötigte Bibliothek `php_mbstring.dll` in das Windows System Verzeichnis (hier: C:\WINDOWS\system) kopiert werden. Die `mbstring` Erweiterung von PHP versteht auch den Umgang mit Mehrbyte Zeichensätzen wie Unicode.

4.5.4 Java(TM) 2 SDK, Standard Edition Version 1.4.2

Die zu installierende Software ist auf der Website von Sun Java Technology unter <http://java.sun.com/> erhältlich. Die Installation erfolgt unkompliziert mit dem Windows Installer durch Doppelklick auf die Datei `j2sdk-1_4_2_09-windows-i586-p.exe`. Wenn die Installation erfolgreich durchgeführt wurde können Java Dateien ohne Probleme kompiliert und ausgeführt werden. Der Pfad `C:\j2sdk1.4.2_09\bin` wird der Systemvariablen PATH als Wert hinzugefügt.

4.5.5 Xerces-J 2.7.1

Xerces ist ein XML Parser der Apache Group. Die Software ist erhältlich auf der Website der Apache Software Foundation unter <http://xerces.apache.org/>. Xerces unterstützt DOM, SAX sowie XML Schema und DTD. Der Inhalt der Zip Datei wird in einen beliebigen Ordner extrahiert. Die Dateien `xerces.jar` und `xml-apis.jar` enthalten die eigentliche Funktionalität des Parsers. Diese beiden Dateien werden in das `lib`- Verzeichnis der Java Installation kopiert. Damit Java die neuen Archive auch findet, muss der Klassenpfad in der Umgebungsvariable `CLASSPATH` angepasst werden. Die Pfade `c:\j2sdk1.4.2_09\lib\xmlapis.jar;c:\j2sdk1.4.2_09\lib\xercesImpl.jar` sowie `c:\j2sdk1.4.2_09\programme` werden der `CLASSPATH` Variable als Wert hinzugefügt. Xerces wird in dieser Arbeit verwendet, um das generierte CDA Instanzdokument auf Schema Gültigkeit hin zu überprüfen.

4.5.6 Xalan 2.7.0

Xalan ist ein Stylesheet Prozessor für die Transformation eines XML Dokumentes in ein anderes XML Dokument oder in das HTML Format. Xalan wird ebenfalls von der Apache Software Foundation als Open Source Software zur Verfügung gestellt. Der Code steht in Java und C++ zur freien Verfügung. Für diese Arbeit wird die aktuelle Version (Dezember 2005) in Java verwendet. Es erfolgt der Download der Zip Datei von der Webseite <http://xml.apache.org/xalan-j/>. Für die Installation muss die Datei `xalan.jar` in das `lib`- Verzeichnis von Java eingefügt werden und anschließend- wie für Xerces oben beschrieben durch Anpassung der Classpath Variablen verfügbar gemacht werden. Xalan wird in dieser Arbeit zu Demonstrationszwecken verwendet. An der Kommandozeile aufgerufen wird der Anwendung eine XML Datei, die mit einem Stylesheet verbunden ist, als Parameter übergeben. Xalan kann dann je nach Anweisung die Transformation durchführen (z. Bsp. In HTML) und das Transformationsergebnis in einer Datei ausgeben.

4.5.7 Weitere Werkzeuge

Als XML Editor wird die Software `Bonfire Studio` verwendet. Es ist kostenlos unter verschiedenen Adressen im World Wide Web erhältlich. Dieser Editor ermöglicht mehrere Dateien gleichzeitig zu bearbeiten. Der Editor prüft auf Wohlgeformtheit und bietet die Möglichkeit verschiedener Ansichten einer Datei, beispielsweise als Baumstruktur.

`HTML Kit` ist ein Webseiten Editor mit vielen Funktionalitäten. Es ist als kostenlose Software erhältlich unter der Adresse <http://www.chami.com/html-kit/>. Das Programm bietet unter anderen Syntax Highlighting für verschieden Programmiersprachen.

5 Ergebnisse

Die Ergebnisse dieser Arbeit sollen dargelegt werden, indem sie in Beziehung zu den Zielen der Arbeit gesetzt werden, welche im Kapitel Einleitung beschrieben worden sind.

5.1 Ergebnisse zu Ziel 1

Die zwei Teilziele zu Problem 1 wurden folgendermaßen definiert:

- Es ist klar, welche Merkmale für die konkrete medizinische Fragestellung Varikositäs, also dem Krampfaderneiden der unteren Extremitäten des Menschen erfasst werden müssen, sodass in einem späteren Schritt aus den erfassten Daten ein CDA konformer Befundbericht aus diesen Daten generiert werden kann.
- Es ist klar, wie die erfassten Daten gespeichert werden müssen, um daraus bei Bedarf auch statistische Analysen durchführen zu können.

Nach Erarbeitung der Grundlagen zu den Themen XML und CDA sowie der Erörterung des medizinischen Krankheitsbildes Varikositäs ist das erste Ergebnis dieser Arbeit, dass klar ist, welche Daten für die konkrete Fragestellung erfasst werden müssen, um daraus in einem späteren Schritt einen CDA konformen Befundbericht zu generieren. Zur Veranschaulichung dieses Ergebnisses dienen die Tabellen 4,5 und 6, welche eine Übersicht über alle für diese Webapplikation verwendeten CDA Elemente gibt und die Zusammenhänge zwischen Eingabefeldern in den Formularen bzw. Autoincrementfeldern in der Datenbank und den jeweiligen CDA Elementen aufzeigt, welche diese Daten im generierten Instanzdokument aufnehmen. (→4.1)

Um bei Bedarf auch statistische Analysen aus den dokumentierten Daten durchführen zu können, erfolgt zunächst die Speicherung in einem relationalen Datenbanksystem. In Abschnitt 2.1.6 wird beschrieben, dass XML gut geeignet ist als Austauschformat zwischen verschiedenen Anwendungen und für dokumentenzentrierte Dokumente, jedoch nicht gut als Ausgangspunkt für statistische Analysen geeignet ist. Aus diesem Grund werden in der in dieser Arbeit entwickelten Webapplikation die erfassten Daten zuerst in einer MySQL Datenbank gespeichert und erst sekundär in ein CDA Dokument eingefügt. Insofern sind Ergebnisse dieses Teilziels das entwickelte E/R Modell und das davon abgeleitete Datenbankschema für diese Anwendung. (siehe Abschnitt 4.2)

5.2 Ergebnisse zu Ziel 2

Ziel 2 wurde folgendermaßen definiert:

- Ziel ist es, aus relationalen Daten dynamisch einen Befundbericht zu generieren, welcher dem Standard der Clinical Document Architecture entspricht.

In Abschnitt 4.3.2 wird das PHP Skript erläutert, welches die aus der Datenbank abgefragten Daten in ein CDA Template Dokument einfügt, wodurch ein XML Dokument mit den Daten aus der durchgeführten Untersuchung entsteht. Dass das auf diese Weise generierte Dokument ein Instanzdokument zu CDA Schema Level 1 Release 1 ist, kann mit dem validierenden Parser Xerces gezeigt werden. (siehe Abschnitt 3.1.2).

Als Beispiel wird ein mit der entwickelten Webapplikation generiertes CDA Level1 Release 1 schemagültiges XML Dokument auf den nächsten Seiten abgedruckt:

Mit dieser XML-Datei sind anscheinend keine Style-Informationen verknüpft. Nachfolgend wird die Baum-Ansicht des Dokuments angezeigt.

```

- <!--
  <?xml-stylesheet href="cda_stylesheet.xml" type="text/xsl" ?>
  -->
- <levelone xsi:schemaLocation="urn:hl7-org/cda levelone_1.0.xsd">
- <clinical_document_header>
  <id EX="303" RT="2.16.840.1.113883.3.7.2.34556.1"/>
  <document_type_cd V="11490-0" S="2.16.840.1.113883.6.1" DN="Befundbericht (kurz)"/>
  <origination_dttm V="2006-02-11 08:45:34"/>
- <patient_encounter>
  <id EX="303" RT="2.16.840.1.113883.3.933"/>
  <practice_setting_cd V="GEFAESS" S="2.16.840.1.113883.3.5.10588" DN="Praxis_Mayer"/>
  <encounter_tmtr V="2006-02-11"/>
</patient_encounter>
- <intended_recipient>
  <intended_recipient.type_cd V="TRC"/>
- <person>
  <id EX="67" RT="2.16.840.1.113883.3.933"/>
- <person_name>
  - <nm>
    <PFX V="Dr. med." QUAL="AC"/>
    <GIV V="Josef"/>
    <FAM V="Huber"/>
  </nm>
  <person_name.type_cd V="L" S="2.16.840.1.113883.5.1xxx"/>
</person_name>
- <addr USE="WP RES">
  <STR V="100"/>
  <HNR V="Stadtplatz"/>
  <ZIP V="0101"/>
  <CTY V="Plötzberg"/>
</addr>
  <telecom V="tel:43(0)2222 3333" USE="WP"/>
  <telecom V="fax:43(0)2222 3334" USE="WP"/>
  <telecom V="mailto:drhuber@aerzte.at" USE="WP"/>
  <telecom V="web:http://" USE="WP"/>
</person>
</intended_recipient>
- <provider>
  <provider.type_cd V="CON"/>
- <person>
  <id EX="123456" RT="2.16.840.1.113883.3.7.2.12345.1.2"/>
- <person_name>
  - <nm>
    <PFX V="Dr. med." QUAL="AC"/>
    <GIV V="Maximilian"/>
    <FAM V="Mayer"/>
  </nm>
  <person_name>
- <addr>
  <STR V="Uferstraße"/>
  <HNR V="10"/>
  <ZIP V="0101"/>

```

```

    <CTY V="Plötzberg"/>
    </addr>
    <telecom V="tel:43(0)2222 4444" USE="WP"/>
    <telecom V="fax:43(0)2222 4445" USE="WP"/>
    <telecom V="mailto:drmayer@aerzte.at" USE="WP"/>
    <telecom V="web:http://www.varizenpraxis.at" USE="WP"/>
  </person>
</provider>
- <patient>
  <patient.type_cd V="PATSBJ"/>
  - <person>
    <id EX="61" RT="2.16.840.1.113883.3.7.2.12345.1.2"/>
    - <person_name>
      - <nm>
        <GIV V="Anna"/>
        <FAM V="Mustermann"/>
      </nm>
    </person_name>
    - <addr>
      <STR V="Hauptstraße"/>
      <HNR V="1"/>
      <ZIP V="0101"/>
      <CTY V="Plötzberg"/>
    </addr>
    <telecom V="43(0)2222 2222" USE="HP"/>
  </person>
  <birth_dttm V="1960-01-01"/>
  <administrative_gender_cd V="W" S="2.16.840.1.113883.5.10285"/>
</patient>
</clinical_document_header>
- <body>
+ <section></section>
+ <section></section>
- <section>
  <caption>Abschluss</caption>
  - <list>
    - <item>
      - <content>
        Bei der Patientin besteht ein Zustand nach Stripping der Vena saphena magna(VSM) li im
        Jahr 2003 mit iatrogener N. saphenus Läsion. Am re Bein zeigt sich eine VSM Insuffizienz
        Grad III nach Hach, sowie ausgeprägte Seitenastkonvolute am re Ober- und Unterschenkel.
        Am re US besteht zusätzlich eine Perforansveneninsuffizienz. Am rechten Bein besteht wird
        die Indikation für ein VSM Stripping, Entfernung der Seitenastkonvolute sowie Ligatur der
        insuffizienten Perforansvenen am re US gestellt. Ein entrechender OP Termin wird mit der
        Patientin in den nächsten Wochen vereinbart. Inzwischen sollte die begonnene
        Kompressionbehandlung fortgeführt werden. Eine Kopie des Befundes geht an die Patientin
        und Ihren Hausarzt Dr. Josef Huber
      </content>
    </item>
  </list>
</section>
- <section>
  <caption>Anamnese01</caption>
  - <list>
    - <item>
      <content>kh_beginn_gt30</content>
    </item>
  </list>
</section>

```

```
</item>
- <item>
  <content>kh_dauer_gt5</content>
</item>
- <item>
  <content>famBel_vater</content>
</item>
- <item>
  <content>stehend</content>
</item>
- <item>
  <content>2</content>
</item>
- <item>
  <content>1-5</content>
</item>
- <item>
  <content>AK-Th:keine</content>
</item>
</list>
</section>
- <section>
  <caption>Anamnese02</caption>
  - <list>
    - <item>
      <content>znErysipel_li</content>
    </item>
    - <item>
      <content>znThrombophl li</content>
    </item>
    - <item>
      <content/>
    </item>
    - <item>
      <content>znKomprTh_bds</content>
    </item>
    - <item>
      <content>znVarOP_li</content>
    </item>
    - <item>
      <content>ZnVarOP:lksdf</content>
    </item>
    - <item>
      <content>ZnLuEmbolie:asdfah</content>
    </item>
  </list>
</section>
+ <section></section>
+ <section></section>
+ <section></section>
+ <section></section>
+ <section></section>
+ <section></section>
+ <section></section>
</body>
</levelone>
```

5.3 Ergebnisse zu Ziel 3

Ziel 3 wurde folgendermaßen definiert:

Ziel ist die Darstellung der erfassten Daten in einem Webbrowser in Form eines übersichtlichen und gut strukturierten Befundberichtes durch Anwendung eines Stylesheets auf das generierte CDA konforme XML Dokument.

Auf den folgenden Seiten wird der Befundbericht abgedruckt so wie er durch Anwendung des programmierten XSL Stylesheets auf das vorne abgedruckte Instanzdokument in einem Webbrowser erscheint.

Praxis für Gefäßkrankheiten
Dr. med. Maximilian Mayer
Facharzt für Allgemein Chirurgie und Gefäßchirurgie
Uferstraße 10, 0101 Plötzberg / Tel.:43(0)2222 4444, Fax.:43(0)2222 4445
eMail:drmayer@aerzte.at, Web.:http://www.varizenpraxis.at

Plötzberg, 2006-02-10 13:31:55

An:

Dr. med. Josef Huber
100 Stadtplatz, 0101 Plötzberg
Tel.:43(0)2222 3333, Fax.:43(0)2222 3334
eMail:drhuber@aerzte.at

BEFUNDBERICHT

Patient:

Name: Anna Mustermann
geb. am: 1960-01-01
Untersuchungsdatum: 2006-02-10

Diagnosen:	ICD Code
VSM Insuff. III rechts	
Perforansinsuff. re	I83.0
Unterschenkel	I83.0
Zn. VSM Stripping li (2003)	

Weitere Diagnosen

arterielle Hypertonie	I10
Diabetes mell. Typ II	E11

Zusammenfassung/Therapieempfehlung/Termine:

Bei der Patientin besteht ein Zustand nach Stripping der Vena saphena magna(VSM) li im Jahr 2003 mit iatrogener N. saphenus Läsion. Am re Bein zeigt sich eine VSM Insuffizienz Grad III nach Hach, sowie ausgeprägte Seitenastkonvolute am re Ober- und Unterschenkel. Am re US besteht zusätzlich eine Perforansveneninsuffizienz. Am rechten Bein besteht die Indikation für ein VSM Stripping. Entfernung der Seitenastkonvolute sowie Ligatur der insuffizienten Perforansvenen am re US gestellt. Ein entprechender OP Termin wird mit der Patientin in den nächsten Wochen vereinbart. Inzwischen sollte die begonnene Kompressionbehandlung fortgeführt werden. Eine Kopie des Befundes geht an die Patientin und Ihren Hausarzt Dr. Josef Huber.

Anamnese	
Item	
Beginn Alter <30.Lj.	<input type="checkbox"/>
Beginn Alter >30.Lj.	<input checked="" type="checkbox"/>
Krankheitsdauer <5 Jahre	<input type="checkbox"/>
Krankheitsdauer >5 Jahre	<input checked="" type="checkbox"/>
fam. Belastg. mütterlicherseits	<input checked="" type="checkbox"/>
fam. Belastg. väterlicherseits	<input type="checkbox"/>
Beruf stehend	<input checked="" type="checkbox"/>
Beruf stehend/sitzend/gehend	<input type="checkbox"/>
Schwangerschaften (Anzahl)	<input checked="" type="checkbox"/> 1
Einnahme OH (Dauer in Jahre)	<input checked="" type="checkbox"/> 1-5
Item	re li
Z.n. Erysipel	<input type="checkbox"/> <input type="checkbox"/>
Z.n. Thrombophlebitis	<input type="checkbox"/> <input checked="" type="checkbox"/>
Z.n. Phlebothrombose	<input type="checkbox"/> <input type="checkbox"/>
Z.n. Kompressionsbehandlung	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Schmerzhafte Varizen	<input checked="" type="checkbox"/> <input type="checkbox"/>
Müdigkeit und Schweregefühl	<input checked="" type="checkbox"/> <input type="checkbox"/>
Juckreiz der Beine	<input type="checkbox"/> <input type="checkbox"/>
Beinschwellung	<input checked="" type="checkbox"/> <input type="checkbox"/>
nächtliche Wadenkrämpfe	<input checked="" type="checkbox"/> <input type="checkbox"/>
Z.n. Varizen OP	<input type="checkbox"/> <input checked="" type="checkbox"/>
Z.n. varizen OP Text: im Jahr 2003, VSM Stripping li, Saphenusläsion li	
Einnahme von Antikoagulationen Text: keine	
Z.n. Lungenembolie Text:	

Klinischer Status: Venös	
Item	re li
florides Ulcus cruris	<input type="checkbox"/> <input type="checkbox"/>
Z.n. Ulcus cruris(Ulcusnarbe)	<input type="checkbox"/> <input type="checkbox"/>
Corona phlebectatica	<input type="checkbox"/> <input type="checkbox"/>
Zyanose	<input type="checkbox"/> <input type="checkbox"/>
primäres Lymphödem	<input type="checkbox"/> <input type="checkbox"/>
sekundäres Lymphödem	<input type="checkbox"/> <input type="checkbox"/>
Z.n. N.saphenus Läsion	<input type="checkbox"/> <input checked="" type="checkbox"/>
Z.n. N.suralis Läsion	<input type="checkbox"/> <input type="checkbox"/>
Vena femoropoplitea (Giacomini):	<input type="checkbox"/> <input type="checkbox"/>
VSM- VSP Anastomose:	<input type="checkbox"/> <input type="checkbox"/>
Seitenäste Oberschenkel:	<input checked="" type="checkbox"/> <input type="checkbox"/>
Seitenäste Unterschenkel :	<input checked="" type="checkbox"/> <input type="checkbox"/>
Fußvarizen:	<input type="checkbox"/> <input type="checkbox"/>
Perinealvarizen:	<input type="checkbox"/> <input type="checkbox"/>
retikuläre Varizen Oberschenkel:	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
retikuläre Varizen Unterschenkel:	<input checked="" type="checkbox"/> <input type="checkbox"/>
Besenreiser Oberschenkel:	<input checked="" type="checkbox"/> <input type="checkbox"/>
Besenreiser Unterschenkel:	<input type="checkbox"/> <input type="checkbox"/>
klinischer Status: Arteriell	
Arteriell o.B.	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Arteriell pathologisch Text	

Klinischer Status: Perforansvenen		CW- Doppler: V.saphena magna (VSM)			
Item	re	li	Item	re	li
Insuffiziente Perforansvenen C1 (6-7 cm):	<input checked="" type="checkbox"/>	<input type="checkbox"/>	VSM Crosse Insuffizienz:	<input type="checkbox"/>	<input type="checkbox"/>
Insuffiziente Perforansvenen C2 (13 cm):	<input type="checkbox"/>	<input type="checkbox"/>	VSM Crosse Insuffizienz Rezidiv:	<input type="checkbox"/>	<input type="checkbox"/>
Insuffiziente Perforansvenen C3 (18 cm):	<input type="checkbox"/>	<input type="checkbox"/>	VSM Stamminuffizienz proximale Hälfte des Oberschenkels:	<input type="checkbox"/>	<input type="checkbox"/>
Insuffiziente Perforansvenen C4 (24 cm):	<input type="checkbox"/>	<input type="checkbox"/>	VSM Stamminuffizienz distale Hälfte des Oberschenkels:	<input type="checkbox"/>	<input type="checkbox"/>
Insuffiziente Perforansvenen (Boyd):	<input type="checkbox"/>	<input type="checkbox"/>	VSM Stamminuffizienz proximale Hälfte des Unterschenkels:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Insuffiziente Perforansvenen (Dodd):	<input type="checkbox"/>	<input type="checkbox"/>	VSM Stamminuffizienz dist Hälfte des Unterschenkels:	<input type="checkbox"/>	<input type="checkbox"/>
Insuffiziente Perforansvenen Unterschenkel außen (Bassi- 8cm):	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CW- Doppler: V.saphena parva (VSP)		
Insuffiziente Perforansvenen Unterschenkel außen (12cm):	<input checked="" type="checkbox"/>	<input type="checkbox"/>	VSP Crosse Insuffizienz:	<input type="checkbox"/>	<input type="checkbox"/>
LRR			VSP Crosse Insuffizienz Rezidiv:	<input type="checkbox"/>	<input type="checkbox"/>
LRR Normalbefund:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	VSP Stammveneninsuffizienz prox. 1/3:	<input type="checkbox"/>	<input type="checkbox"/>
LRR pathologischer Befund:	<input type="checkbox"/>	<input type="checkbox"/>	VSP Stammveneninsuffizienz dist. 1/3:	<input type="checkbox"/>	<input type="checkbox"/>
LRR mit Kompression normaler Befund:	<input type="checkbox"/>	<input type="checkbox"/>			
LRR mit Kompression pathologischer Befund:	<input type="checkbox"/>	<input type="checkbox"/>			

Abbildung 13: Seite 1 bis 3 des Befundberichtes Browseransicht

6 Diskussion

6.1 Diskussion der eigenen Ergebnisse

Alle in Abschnitt 1.4 definierten Ziele wurden erreicht. Das Ergebnis dieser Arbeit ist ein Prototyp eines als Webapplikation implementierten Dokumentationssystems für die medizinische Fragestellung Varikosität. (→2.3). Die Funktionalität der programmierten Anwendung lässt sich in drei Bereiche gliedern.

Die erfassten Daten werden zunächst in einem relationalen Datenbanksystem gespeichert, um daraus bei Bedarf auch statistische Berechnungen durchführen zu können. Dabei soll kritisch angemerkt werden, dass in der jetzigen Version für statistische Abfragen keine entsprechende Funktionalität implementiert ist. Statistische Auswertungen könnten momentan nur mit dem MySQL- Administrations Tool „phpMyAdmin“ durchgeführt werden. Es kann mit Literatur belegt werden, dass XML für die Durchführung statistischer Datenanalysen nicht das optimale Speicherformat ist und daher das gewählte Vorgehen prinzipiell sinnvoll ist. (→2.1.6), [41]. Zum erstellten Datenbankschema ist kritisch anzumerken, dass die im ER Modell angeführte Entität „Untersucher“ nicht wie alle anderen Entitäten des ER Modells ins relationale Modell überführt wurde. (→4.2.4). Für diese Arbeit wurde angenommen, dass immer derselbe Arzt einen Patienten untersucht und entsprechende Befunde dokumentiert. Alle für CDA relevanten Daten dieses Arztes wurde bereits in das CDA Template Dokument eingefügt. Sie müssen also nicht wie die Daten der anderen beteiligten Personen (Patient, weiterbehandelnder Arzt) erfasst, in der Datenbank gespeichert und in einem weiteren Schritt an korrekter Stelle in das CDA Template Dokument eingefügt werden. (→4.3.2).

Im zweiten Schritt wird aus den erfassten Daten schemagültiges XML (CDA Level Release 1) generiert. Dabei werden Platzhalter eines XML Dokumentes, welches als Template dient, durch die bei der Untersuchung erhobenen Befunde ersetzt. Für jedes dokumentierte Merkmal muss dazu die Funktion `str_replace()` ausgeführt werden. (→4.3.2) Als elegantere Lösungsmethode könnte versucht werden eine Schleifen Konstruktion mit einer „for- Schleife“ auszuführen. Dazu müssten die einzelnen Platzhalter durch Integerwerte ersetzt werden und die Arrays der einzelnen Abfragen zu einem gesamten Array verschmolzen werden, so dass die Array Elemente über ihren (linear ansteigenden) numerischen Index angesprochen werden könnten. Wichtig ist, dass das generierte XML Dokument dem CDA Standard entspricht. Dies kann durch die Anwendung eines validierenden Parsers auch demonstriert werden. (→4.3.2)

Für die Darstellung des generierten Instanzdokumentes wurde ein XSL Stylesheet programmiert. Über die Anweisung `<?xml-stylesheet href="cda_stylesheet.xml" type="text/xsl"?>` in der Processing Instruction des generierten Instanzdokumentes wird das Instanzdokument mit dem Stylesheet verbunden und es erfolgt die Transformation ins HTML Format und die gewünschte Darstellung im Webbrowser. Als Vorgabe für die abschließende zusammenfassende Darstellung der dokumentierten Daten wurden papierbasierte Formulare verwendet, die auch in der Praxis eingesetzt werden. Durch das XML- Konzept der Trennung von Daten und Layout ist die Präsentation von Daten, die im XML Format vorliegen, flexibel handhabbar. Der Inhalt ein und desselben XML Dokumentes kann für verschiedene Zwecke oder verschieden Personengruppen durch Anwendung verschiedener Stylesheets komplett unterschiedlich dargestellt werden. [24]

Drei weitere Schwachstellen der aktuellen Version des Prototyps werden beschrieben:

Zunächst muss gesagt werden, dass bis jetzt keine Fehlerbehandlung implementiert wurde.

Es werden auch keine eigenen OID verwendet, sondern OID aus anderen veröffentlichten Instanzdokumenten übernommen. (→2.2.3.4). In den Extension Attributen der Id- Elemente der CDA Knoten `clinical_document_header/intended_recipient/person` und `clinical_document_header/patient/person` werden allerdings die entsprechenden Werte aus der Datenbank (Autoincrementfelder, →4.1) eingefügt. Außerdem wird im Extension Attribut der Id Elemente der CDA Knoten `clinical_document_header` und `clinical_document_header/patient_encounter` der jeweilige Wert für „fallzahl“ (ebenfalls ein Autoincrementwert aus der Datenbank) eingefügt.

Als dritter Schwachpunkt ist zu nennen, dass als Zeichensatz „Latin 1“ (iso-8859-1) verwendet wurde und nicht der XML Default Zeichensatz Unicode UTF-8 (iso-10646). Latin 1 wurde für die vorliegenden Webapplikation verwendet, weil es Hinweise gibt, dass die Unicode Unterstützung der Komponenten, die bei einer Webentwicklung zum Einsatz kommen noch nicht perfekt ist. Der Zeichensatz Latin 1 ist für die Zwecke dieser Arbeit jedoch ausreichend und wird von allen verwendeten Komponenten problemlos unterstützt wird. [50]

Als Beispiel für eine andere Implementierung eines Dokumentationssystems als Webapplikation mit dem Ziel der automatisierten Generierung eines CDA konformen Arztbriefes kann die Arbeit von Paterson GI, Sheperd M, Wang X et al. genannt werden. [18]. Für die Implementierung werden HTML Formulare und die Skriptsprache Java Skript, sowie Java Servlets und eine native XML Datenbank benützt. Das System bietet die Möglichkeit neue Arztbriefe zu erstellen, in der Datenbank abgelegte CDA- Dokumente nach bestimmten Suchkriterien zu suchen, darzustellen und zu verändern.

Abschließend kann zu der im Rahmen dieser Arbeit implementierten Webapplikation noch gesagt werden, dass vorliegende System in der momentanen Version lediglich ein Prototyp ist und nicht dafür geeignet ist in der Klinik oder einer Praxis eingesetzt zu werden. Es kann an dieser Stelle auch bestätigt werden, dass CDA Level 1 einen Einstieg in den Standard ohne großen technischen Aufwand ermöglicht. Literatur sowie Beispiele für CDA Level I Instanzdokumente und CDA Level I Schemadateien sind leicht zugänglich. [52],[45]

6.2 Ausblick auf CDA Level II und Level III

CDA wird mittlerweile von einer großen Anzahl von Ländern in Projekten und zur definitiven IT Versorgung eingesetzt. Die Knochenmarkstransplantations- Dokumentation in Israel ist ein CDA Projekt, Finnland hat CDA als nationale Strategie für die Unterstützung der Gesundheitsprozesse gewählt. In Deutschland wird CDA im so genannten „SCIPHOX“-Projekt („Standardized Communication of Information Systems in Physicans Offices and Hospitals using XML“) für nationale Begebenheiten adaptiert. [19],[52],[54]. Einige weitere internationale Projekte sind: „HYGEIAnet“ (Griechenland), „MERIT-9“ (Japan), „Buenos Aires Project“ (Argentinien), „e- Claims Supproting Document Architecture“ (Cananda), „Dalhousie U, QEII Helath Sci Ctr“ (Canada). In der Mayo Klinik im US Bundesstaat Minnesota werden pro Woche in etwa 30.000 medizinische Dokumente im CDA Format generiert. (etwa 12.000.000 Dokumente in den letzten sieben Jahren).

Wie im Grundlagenkapitel beschrieben stellen die verschiedenen Level des CDA Standard einen Migrationspfad dar, um in jedem weiteren Level iterativ Markup hinzuzufügen. Im gleichen Maße werden die Instanzdokumente höherer Level damit zu einem höheren Grad maschinell verarbeitbar. (→2.2.2.1). Die Entwicklung Template basierter Dokument Typen setzt eine intensive Analyse bestehender medizinischer Dokumente und detailliertes Wissen in medizinischen Domänen voraus, welches durch Einbeziehen entsprechender Experten und Gesellschaften durch HL7 erreicht wird.

Die Architekturparadigmen der „open Electronic Health Record Foundation“ gehen über die CDA Spezifikation hinaus. [53]. Sie basieren auf einer modellgetriebenen Architektur von komponentenorientierten Systemen. Die Komponenten beschreiben statische und dynamische Konzepte, die domänenbezogen durch Constraint Models beschrieben werden. [54]. Gemeinsame Referenzen und Basiskonzepte sowie zumindest teilweise gemeinsame Methodologien und Tools sollen die Kohärenz der verschiedenen Ansätze sichern.

7 Verzeichnisse

7.1 Literatur

1. Haux R, Knaup P, Bauer AW, Herzog W, Reinhardt E et al. Information Processing in Healthcare at the Start of the third Millennium: Potential and Limitations. *Methods Inf Med* 2001;40:156-62.
2. Kay S. Health Informatics: Challenges to Progress. *Methods Inf Med* 1999;38:225-8.
3. International Medical Informatics Association. Recommendations of the International Medical Informatics Association on Education in Health and Medical Informatics. *Methods Inf Med* 2000;39:267-77.
4. Bates DW, Cohen M, Leape L, Overhage M, Shabot MM, Sheridan Th. Reducing the Frequency of Errors in Medicine using Information Technology. *Journal of the American Medical Informatics Association* 2001;8(4):299-308.
5. Kuntalp M, AO. A simple and low- cost Internet- based teleconsultation system that could effectively solve the health care access problems in underserved areas of developing countries. *Computer Methods and Programs in Biomedicine* 2004;75:117-26.
6. Ball MJ, LJ. E-health: transforming the physician / patient relationship. *International Journal of Medical Informatics* 2001;61:1-10.
7. Kulikowski CA. The Micro- Macro Spectrum of Medical Informatics Challenges: From Molecular Medicine to Transforming Health Care in a Globalizing Society. *Methods Inf Med* 2002;41:20-4.
8. Haux R. Health care in the Information Society: What should be the Role of Medical Informatics? *Methods Inf Med* 2002;41:31-5.
9. Haux R, AE, Herzog W, Knaup P. Health Care in the Information Society. A prognosis for the Year 2013. *Int.Journal of Medical Informatics* 2002;66:3-21.
10. Leiner F, GW, Haux R, Knaup- Gregori P, Pfeiffer KP. *Medizinische Dokumentation- Grundlagen einer qualitätsgesicherten integrierten Krankenversorgung*. Stuttgart: Schattauer Verlag; 2003.
11. Kuhn KA, GD. From Hospital Information Systems to Health Information Systems. *Methods Inf Med* 2001;40:275-87.

12. Europäisches Observatorium für Gesundheitssysteme. Website: Gesundheitssysteme im Wandel Österreich. <http://www.euro.who.int/observatory/>; letzter Zugriff: August 2005;
13. Europäisches Observatorium für Gesundheitssysteme. Website: Gesundheitssysteme im Wandel Deutschland. <http://www.euro.who.int/observatory/>; letzter Zugriff August 2005;
14. Ammenwerth E Haux R, Kulikowski C, Bohne A, Brandner R, Brigl B, et al., Medical Informatics and the Quality of Health: New Approaches to Support Patient Care. In:,(Hrsg) 2003. In: Haux R KC, editor. Yearbook of Medical Informatics 2003 Quality of Health Care: The Role of Informatics. D 70040 Stuttgart: Schattauer Verlag; 2003.
15. Bloch P. Nachrichtenstandards verbessern die Kommunikation und steigern die Qualität. Swiss Medical Informatics 2001;47:20-22.
16. World Wide Web Consortium. Website: Word Wide Web Consortium. www.w3.org; letzter Zugriff: September 2005;
17. World Wide Web Consortium. Website: XML Standard des Word Wide Web Consortiums. www.w3.org/xml; letzter Zugriff: September 2005;
18. Paterson GI, Shepherd M, Wang X, Watters C, Zitner D, et al. Using the XML- based Clinical Document Architecture for Exchange of Structured Discharge Summaries. In: Proceedings of the 35th Hawaii International Conference on System Science 2002. p. 1-10., http://users.cs.dal.ca/~shepherd/web_pubs.html; letzter Zugriff: 05.02.2006;
19. Heitmann KU SR, Dudeck J. Discharge and referral data exchange using global standards- the SCIPHOX project in Germany. Int Journal of Medical Informatics 2003;70:195-203.
20. Health Level Seven I. Health Level Seven. <http://www.hl7.org/>; letzter Zugriff: Jänner 2005;
21. The American National Standards Institute ANSI American National Standards Institute. <http://www.ansi.org/>; letzter Zugriff: Jänner 2005;
22. Health Level Seven I. HL7 Data Model Development. <http://www.hl7.org/Library/data-model/index.cfm>; letzter Zugriff: Jänner 2005;
23. Bott OJ. XML- Schlüsseltechnologie der Medizinischen Dokumentation? Forum der medizinischen Dokumentation und Medizin- Informatik 1999;2:9-13.

24. Bludau HB Wolff A, Hochlehnert AJ. Presenting XML- based Medical Discharge Letters According to CDA. *Methods Inf Med* 2003;42:552-6.
25. Renner H SW, Gabor S,. Patients Dismissal letter- No standard is the standard. In: *Proceedings of the TEHRE 2001, 6th Annual European Health & IT- Conference and Exhibition on Electronic health Records*; 2001; London; 2001.
26. Spießl H CC. Qualität und Anforderungsprofile von Entlassungsbriefen und Einweisungsscheinen. *Zeitschrift für ärztliche Fortbildung und Qualitätssicherung ,ZaeQF* 2002;96:257-60.
27. World Wide Web Consortium. Website: World Wide Web Consortium. www.w3.org; letzter Zugriff: September 2005;
28. World Wide Web Consortium. Website: XML Standard des Word Wide Web Consortiums. www.w3.org/xml/; letzter Zugriff: September 2005;
29. The Unicode Consortium. Website: Unicode Home Page. <http://www.unicode.org/>; letzter Zugriff: Februar 2006;
30. World Health Organisation. Website: Overview of SGML Resources. <http://www.w3.org/MarkUp/SGML/>; letzter Zugriff: 06.02.2006;
31. World Wide Web Consortium. Website: XML Schema. www.w3.org/XML/Schema; letzter Zugriff 04.02.2006;
32. World Wide Web Consortium. Website: Namespaces in XML. <http://www.w3.org/TR/REC-xml-names/>; letzter Zugriff: 04.02.2006;
33. World Wide Web Consortium. Website: Naming and Addressing: URIs, URLs, ... <http://www.w3.org/Addressing/>; letzter Zugriff: 04.02.2006;
34. World Wide Web Consortium. Website: XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). <http://www.w3.org/TR/xhtml1/>; letzter Zugriff: 04.02.2006;
35. World Wide Web Consortium. Website: Mathematical Markup Language (MathML™) 1.01 Specification. <http://www.w3.org/TR/REC-MathML/>; letzter Zugriff: 04.02.2006;
36. World Wide Web Consortium. Website: Scalable Vector Graphics (SVG). <http://www.w3.org/Graphics/SVG/>; letzter Zugriff: 04.02.2006;
37. World Wide Web Consortium. Website: Document Object Model (DOM). <http://www.w3.org/DOM/>; letzter Zugriff: 04.02.2006;

38. Source Forge Services. Website: About SAX. <http://www.saxproject.org/>; letzter Zugriff: 04.02.2006;
39. World Wide Web Consortium. Website: XML Path Language (XPath)Version 1.0. <http://www.w3.org/TR/xpath>; letzter zugriff: 04.02.2006;
40. World Wide Web Consortium. Website: XSL Transformations (XSLT)Version 1.0. <http://www.w3.org/TR/xslt>; letzter Zugriff: 04.02.2006;
41. Bourret RP. XML and Databases. <http://www.rpbouret.com/xml/XMLAndDatabases.htm>; letzter Zugriff: 04.02.2006;
42. HL7 Benutzergruppe in Deutschland. Website: First international conference on CDA, Berlin 2002. www.hl7.de/cda2002; letzter Zugriff: 04.02.2006;
43. Dolin RH AL, et al. HL7 Clinical Document Architecture, Release 2.0. <http://www.hl7.de/iamcda2004/fprogram.html>; letzter Zugriff: 04.02.2006;
45. Health Level Seven. Website: HL7 Reference Information Model. <http://www.hl7.org/>; letzter Zugriff: 04.02.2006;
46. Dolin RH, Alschuler L, Beebe C, Biron PV, Lee Boyer S, et al. The HL7 Clinical Document Architecture. JAMIA 2001;8(6):552-69.
47. International Organisation for Standardization. Website: International Organisation for Standardization. <http://www.iso.org/iso/en/ISOOnline.frontpage>; letzter Zugriff: 04.02.2006;
48. HL7 Benutzergruppe in Deutschland. OID-Konzept für das Deutsche Gesundheitswesen. <http://www.hl7.de/downloads/documents/oid-konzept/>; letzter Zugriff: 04.02.2006;
49. World Health Organisation. Website: International Classification of Diseases (ICD). <http://www.who.int/classifications/icd/en/>;
50. Kofler M Öggl B. Buchtitel: PHP 5 & MySQL 5. München, Boston, San Francisco: ADDISON- WESLEY; 2005.3-8273-2158-1:
52. Arbeitsgemeinschaft SCIPHOX GbR mbH. Website: SCIPHOX. <http://www.sciphox.de/index.htm>; letzter Zugriff: 11.02.2006;
53. Bergmann J. openEHR - Die Geschichte eines Baukastensystems für eine gemeinsame Elektronische Gesundheitsakte. Forum der Medizin_Dokumentation und Medizin_Informatik, mdi 2005;7(1):8-15.

54. Blobel B Heitmann K, Görke HJ,. CDA Übersicht.
<http://www.sciphox.de/atwork/cda/uebersicht.htm>; letzter Zugriff:11.02.2006;
55. open EHR Community. Website: HL7 Standards /Towards harmonisation.
http://www.openehr.org/standards/t_hl7.htm; letzter Zugriff: 11.02.2006;
56. Herold G. Varikose. In: Innere Medizin; 2003. p. 674-6. ISBN: HEROLD2003
57. Dolin RH AL, et al. HL7 Clinical Document Architecture, Release 2.0.
<http://www.hl7.de/iamcda2004/fprogram.html>; letzter Zugriff: Dezember 2005;

7.2 Tabellen

Tabelle 1: Vergleich Dokument- Nachricht.....	22
Tabelle 2: XPath Funktionen	50
Tabelle 3: Auswahl von XSLT Elementen	52
Tabelle 4:verwendete CDA Elemente	56
Tabelle 5: Beziehung Autoincrementfeldern in Datenbank und CDA Elemente	56
Tabelle 6: Beziehung Formularfeldnamen und CDA Elemente.....	57
Tabelle 7: Entities und Attribute für CDA Header	59
Tabelle 8: Entities und Attribute für CDA Body	60
Tabelle 9: Relation patient.....	61
Tabelle 10: Relation fall.....	62
Tabelle 11: Relation aerzte	62
Tabelle 12: Relation anamnese01 (Prädisposition).....	62
Tabelle 13: Relation anamnese 02 (Vorerkrankungen).....	63
Tabelle 14: Relation anamnese 03 (aktuelle Beschwerden)	63
Tabelle 15: Relation klinVenoos (klinisch venöser Befund)	64
Tabelle 16: Relation CW_vsm (CW Doppler der Vena saphena magna)	64
Tabelle 17: Relation klinPerf (klinischer Befund Perforansvenen)	65
Tabelle 18: Relation CW_vsp (W Doppler Vena saphena parva)	65
Tabelle 19: Relation LRR (Licht- Reflexions Rheographie).....	65
Tabelle 20: Relation abschluss (Abschlussnotizen)	66
Tabelle 21: Relation diagn_einfach (Hauptdiagnosen)	66
Tabelle 22: Relation diagn_einfach_weit (Nebendiagnosen).....	66
Tabelle 23: Codeaufbau	68

7.3 Abbildungen

Abbildung 1: Metasprachen SGML und XML	15
Abbildung 2: XML Repräsentation eines Object Identifier	30
Abbildung 3: CDA Level 1 Instanzdokument	30
Abbildung 4: Varizen am linken Unterschenkel	31
Abbildung 5: Entity Relationship Modell	58
Abbildung 6: HTML Formular zur Dateneingabe	69
Abbildung 7: Ausschnitt aus Body Teil des CDA Templates	76
Abbildung 8: Schema Validierung mit Parser Xerces	77
Abbildung 9: Formular zur Varizen Dokumentation	78
Abbildung 10: Baumansicht XSL Stylesheet	79
Abbildung 11: Ausschnitt aus XSL Template	81
Abbildung 12: Programmablauf schematisch	82
Abbildung 13: Seite 1 bis 3 des Befundberichtes Browseransicht	95

7.4 Code

Codelisting 1: Namensräume in XML Schema	35
Codelisting 2: komplexer Datentyp in XML Schema	37
Codelisting 3: Arrays in PHP	44
Codelisting 4: SESSION Variablen in PHP	46
Codelisting 5: MySQL Funktionen in PHP	47
Codelisting 6: "connect_to_DB.php"	70
Codelisting 7: Programmablauf in Datei „suchePatArzt.php“	72
Codelisting 8: „cda_b_anamnVorerkrank.php“	74
Codelisting 9: Seite 1 bis 3 des generierten CDA Instanzdokumentes	91

Eidesstattliche Erklärung

Hiermit versichere ich, Franz Oberacher, geboren am 28.09.1969 in Kitzbühel, dass die vorliegende Masterarbeit von mir selbständig verfasst wurde. Zur Erstellung wurden von mir keine anderen, als die angegebenen Quellen und Hilfsmittel verwendet.

Hall in Tirol, am 12.02.2006

(Dr. Franz Oberacher)

Lebenslauf

Daten zur Person:

Geburtsdatum und –ort: 28. September 1969 in Kitzbühel
Familienstand: Ledig
Adresse: Bichling 28, 6363 Westendorf
Telefon: +43 (0)676/ 7000 626

Ausbildung und berufliche Tätigkeiten:

1976 – 1980: Volksschule, 6370 Reith bei Kitzbühel
1980 – 1988: neusprachliches Gymnasium in St. Johann in Tirol
1988 – 1996: Studium der Humanmedizin in Innsbruck
1996 – 1999: Turnusausbildung in Kitzbühel und Innsbruck
1999: Diplom Allgemeinmedizin, Diplom Notararzt
2000: Sekundararzt im KH Natters (Pulmologie)
2001 – 2002: Facharztausbildung Allgemein Chirurgie im KH Kitzbühel
2002 – 2005: Studium Medizinische Informatik UMIT, berufsbegleitend
2002 – 2005: Praxisvertretung in Ordination Dr. Schlegel, Obergurgl
Notarzdienste beim Roten Kreuz im Bezirk Kufstein
Patientenbegleitung bei der Tyrol Air Ambulance, Innsbruck
Seit März 2005: Facharztausbildung für Anästhesie, Intensivmedizin und Notfallmedizin im Bezirkskrankenhaus Kufstein

Danksagung

Bedanken möchte ich mich bei Frau Prof. Ammenwerth für die Übernahme der Betreuung dieser Arbeit und bei Herrn Prof. Haux für die Anregung zum Thema.

Besonderer Dank gilt meiner Freundin Claudia Depauli für die „moralische Unterstützung“ während des gesamten Studiums.

An dieser Stelle möchte ich mich auch bei meinen Eltern und meinem Bruder Klaus für die vielfältige Unterstützung während der letzten Jahre bedanken.

Meinen Studienkollegen möchte ich für Ihre kollegiale Unterstützung und eine bereichernde, schöne gemeinsame Studienzeit auf der UMIT herzlich danken.