3 Grundlagen

Das Internet dient dem Daten- und Informationsaustausch und stellt die Gesamtheit aller weltweit zusammengeschlossenen Computer-Netzwerke dar, die nach standardisierten, technischen Verfahren miteinander kommunizieren. Dieses Grundlagenkapitel führt in die übergeordnete Organisation des Internets ein und stellt dabei die Organe und Strukturen vor, die Standards für den Datenaustausch entwickeln. Diese Organisatoren verfolgen einen gemeinnützigen Zweck und stellen dadurch die lizenzfreie Nutzung der standardisierten Verfahren und Definitionen sicher. Die in dieser Dissertation vorgestellten und erarbeiteten Methoden basieren ausdrücklich auf dem Internet, deswegen ist es hilfreich einen fundierten Einblick in die geltenden Internetstandards (in Kapitel 3.3) zu erhalten. Nur dem Verständnis dieser Grundlagen werden die Vorzüge der hierauf aufbauenden Methodik eines angestrebten *Web-Services* greifbar. Das Grundlagenkapitel führt über die Vermittlung der Inhalte einzelner Bausteine zu dem daraus entwickelten Konstrukt des *Web-Services*. Jede softwaretechnische Umsetzung einer Methodik in der Informatik basiert auf einer Programmiersprache. In Abschnitt 3.9 werden die verwendete Programmiersprache, sowie der benutzen Funktionsbibliotheken vorgestellt, auf die zur prototypischen Implementierung zurückgegriffen wurde. Für den konkreten Anwendungsbezug mit der radiologischen Lehrsammlung wird ebenfalls das zu adaptierende Informationssystems der radiologischen Klinik vorgestellt.

3.1 Internet-Society (ISOC)

Die *ISOC* [40] ist die Dachorganisation einer Reihe auf die Entwicklung formeller Internet-Standards spezialisierter Arbeitsgruppen und Organisationen. Die *ISOC* trägt somit die Verantwortung für die Entwicklung und die Publikation von Internet-Standards. Die wesentlichen Organisationen, die der *ISOC* angehören sind das *Internet-Architecture-Board (IAB)*, die *Internet-Engineering-Task-Force (IETF)* und die *Internet-Engineering-Steering-Group (IESG)*.
Das *IAB* [38] spezifiziert die Gesamtarchitektur des Internets. Daneben wählt das *IAB* den Vorsitzenden des *IETF* und die Mitglieder der *IESG*. Gleichzeitig nimmt sich das *IAB* auch Beschwerden über unzureichende Implementierungen von Internet-Standards an. Schließlich gibt das *IAB* Empfehlungen und Ratschläge an die *ISOC*.

Die *IETF* [35] wird geleitet von der *IAB*. Inhaltlicher Schwerpunkt der *IETF* ist die Entwicklung von Internet-Protokollen und für die Veröffentlichung der *Request-For-Comments-Dokumente (RFC's)* zuständig. Hierzu ist jedoch die Zustimmung der *IESG* nötig.
Die *IESG* erhält von der *IETF* Vorschläge für *RFC*-Dokumente und ernennt den Vorschlag zu einem Internet-Standard, wenn die Spezifikation technisch kompetent, ausgereift, stabil und verständlich dargestellt ist, einen erkennbaren Nutzen hat und zwei unabhängige Implementierungen und Interoperabilitätstest vorgelegt werden.

3.2 World-Wide-Web Consortium (W3C)

Das *W3C* [94] ist ein internationales Industriekonsortium, das gemeinsam vom *Massachusetts-Institute-Of-Technology-Laboraory-Of-Computer-Science (MITLCS)* [54] in den USA, dem *Institut-National-De-Recherche-En-Informatique-Et-Automatique (INRIA)* [37] in Frankreich und der Keio-Universität [42] in Japan geführt wird. Dem Konsortium gehören daneben circa 520 Organisationen als Mitglieder an. Vom *W3C* werden einheitliche Protokolle entwickelt zur Sicherstellung der Interoperabilität im World-Wide-Web. Die anführenden Organisationen bündeln und publizieren die Ergebnisse der Arbeitsgruppen, die sich aus den Mitgliedsorganisationen bilden.

3.3 Internet-Standards

Verbindliche Spezifikationen für Internet-Protokolle sind die Niederlegungen in den *„Request For Comment" (RFC)*-Publikationen zu finden. Die *RFC*-Nummerierung ist fortlaufend, daher sind Erweiterungen oder Veränderungen, die ein Protokoll betreffen, nicht an einer aufeinanderfolgenden Nummerierung zu erkennen. Die Pflege und Weiterentwicklung eines *RFC*-Standards ergibt sich aus der Fachdiskussion, die sich in den Anmerkungen (Comments) des Fachpublikums widerspiegeln. Verwaltet werden die *RFC*-Dokumente von der *IETF* und der *IESG* (siehe Kapitel 3.1).

3.3.1 Internet-Protocol (IP)

Das im September 1981 spezifizierte *IP* [65] definiert die Adressierung und den Transport von Datenpaketen innerhalb des Internets. Der Transport der Daten ist im Rahmen dieses Protokolls nicht abgesichert, diese Absicherung wird mittels des Transmission Control Protocols

(*TCP*), realisiert. In der derzeit noch gebräuchlichen Version 4 des *IP* (IPv4) , steht zur Adressierung der Kommunikationspartner ein 32-Bit-breiter Adressraum zur Verfügung. Dieser Adressraum ist auf logischer Ebene in eine Netzwerk-ID und eine Rechner-ID unterteilt. Laut Definition sind bis zu fünf der oberen Bits des Adressraums zur Bezeichnung sogenannter Adressklassen reserviert. Diese Klassen, bezeichnet von Klasse A bis Klasse E, sind die übrigen Bits zur Codierung der Netzwerk-ID und der Rechner-ID vorgesehen. Dementsprechend lassen sich je nach Adressklasse eine bestimmte Anzahl von Rechnern identifizieren. Die Einteilung in Adressklassen spiegelt die Anzahl der in das Internet einzubindenden Teilnetze wieder. Die Struktur von Teilnetzen kann benutzt werden, um für eine bestimmte Gruppe angeschlossener Computern die Verbindungswege zu anderen Netzsegmenten separat zu definieren, bzw. diese von anderen Netzen zu isolieren. Diese Art der Teilnetze werden als *Intranet* bezeichnet und können bereits auf der Netzwerkebene die Organisationsstruktur einer geschlossenen Benutzergruppe abbilden. In der medizinischen Anwendung können damit beispielsweise Datenschutzvorgaben sichergestellt werden, da keine oder ausschließlich kontrollierbare Verbindungswege zum übrigen Internet existieren. Gegebenenfalls kann in einer derartigen Netzwerkstruktur auf die Anonymisierung der personenbezogenen Daten in den Lehr- und Diskussionsmaterialien verzichtet, und eine Kopplung zu den übrigen Krankenhausinformationssystemen genehmigt werden.

3.3.2 Transport Control Protocol TCP

Das *TCP* ist seit dem Jahr 1981 in *RFC*-793 [66] spezifiziert und sichert die Übermittlung der Datenpakete zwischen Sender und Empfänger. Zur Datenintegrität wird zu jedem Datenpaket eine „Checksumme“ errechnet und im Rahmen der *TCP*-Verbindung geprüft. Im Falle eines Datenfehlers oder Datenverlusts kann der Sender mittels entsprechender *TCP*-basierter „Non-Acknowledge“ (NAK) Signalisierung zum erneuten Senden des Datenpakets veranlasst werden. Die erfolgreiche Übertragung eines Datenpakets wird hingegen mit einer „Acknowledge“ (ACK) Meldung quittiert. Da auch diese Signalisierungen auf dem Datenverbindungsweg verloren gehen oder durch Störungen unbrauchbar werden könnten, startet der Sender einen Zeitmesser, nach dessen Ablauf auch ohne „NAK“-Signal davon ausgegangen wird, dass die Übertragung fehlgeschlagen ist. Eine mit *TCP* gesicherte Datenverbindung bildet die Grundlage für eine robuste Kommunikation und bietet die Plattform für auf diese Sicherungsprämisse aufsetzenden Verfahren, die nachfolgend vorgestellt werden.

3.3.3 Hypertext Transport Protocol (HTTP)

Auf den Protokollen *IP* und *TCP* basierend ist das *HTTP* definiert und 1996 als *HTTP/1.0* im *RFC*-1945 [9] spezifiziert. Es ist bereits in der Version 0.9 seit Anfang 1990 im Einsatz, als *RFC*-Spezifikation wurde es jedoch erst zeitgleich mit der Version 1.0 in *RFC*-1945 niedergelegt. Seit 1997 existiert eine Erweiterung in Form der Version *HTTP/1.1* als *RFC*-2068 [10]. Unterschiede in den Versionen des Standards bestehen nicht im grundlegenden Kommunikationsschema sondern im Leistungsumfang der definierten Anfrage- und Antwort-Codes, die im Rahmen der *TCP/IP* gestützten Verbindung wechselseitig gesandt werden. Die zwei wesentlichen Befehle, die der Sender mittels *HTTP* senden kann sind *HTTP-GET* und *HTTP-POST*. Mittels des *HTTP-GET* erwartet der Sender in der Position des Clients ein Dokument als Teil der Antwort des Servers, während mittels des Befehl *HTTP-POST* Daten zur serverseitigen Bearbeitung gesandt werden. Das *HTTP*, als Verfahren zum Transport von Nachrichten, hat sich bereits beim *World-Wide-Web*-Dienst (im folgenden WWW-Dienst) im Internet bewährt und ist daher prädestiniert für den Einsatz bei den ebenfalls auf Nachrichtenaustausch basierten *Web-Services*. Der WWW-Dienst und der *Web-Service* werden in einem der nachfolgenden Kapitel vorgestellt.

3.3.4 Uniform Resource Locator (URL) / - Identifier (URI)

Der *URL* und der *URI* werden zur Adressierung von Ressourcen im Internet verwendet. Der *URL* wird in *RFC*-1738 [8], der URI in *RFC*-2396 [88] definiert. Dieser Form der Adressierung bedient sich das *HTTP* bei der Identifizierung von Internetobjekten. Ein relativer *URI* ist ein *URL* ohne Protokollbezeichnung und Serveradresse [80]:

http://www.ukbf.fu-berlin.de/medinf/index.htm

Der kursiv gestellte Bereich ist demnach der *URL*, das gepunktet Unterstrichene ist der relative *URI*-Abschnitt. Mittels einer *URL/URI*-Angabe ist sowohl eine Ressource im Internet eindeutig adressiert, als auch das zum Zugriff auf die Ressource einzusetzende Protokoll spezifiziert, im obigen Beispiel das mittels *HTTP*. Die eindeutige Referenzierung von Ressourcen und Diensten im Internet wird in der Methodik dieser Arbeit für den Zugriff auf Lehrmaterialien eingesetzt werden und dient zudem zur Adressierung der Dienste und Programme, die eine Kommunikationsverbindung herstellen müssen.

3.3.5 Simple Mail Transport Protocol (SMTP)

Ebenfalls auf *TCP* und *IP* aufsetzend ist das *SMTP*, ein in *RFC*-821 [67] beschriebenes Verfahren zum Nachrichtenaustausch, von E-Mails. Der im Rahmen dieser Arbeit relevante Unterschied zum *HTTP* ist die Asynchronität der Kommunikation durch Zwischenspeicherung der Nachrichten in zentralen Mailservern. *SMTP* stellt als Transportprotokoll eine Alternative dar, und ist als Option in der Spezifikation eines *Web-Services* gegen *HTTP* austauschbar.

3.4 Extensible Markup Language (XML)

Die *Extensible Markup Language (XML)* definiert ein Datenformat für strukturierte Dokumente. Der Standard [89] besteht seit 1998 und wird vom *W3C* betreut. Die wesentlichen Vorzüge in *XML* und damit die Eigenschaften, die *XML* zu einer verstärkten Akzeptanz als universelles Hilfsmittel zur Darstellung und Speicherung von Daten verholfen haben sind: (1) die Möglichkeit zur Definition individueller Strukturabschnitte (*Tags*) mit eigenen Attributen, (2) die strukturerhaltende Speicherung und Bereitstellung von Datenelementen und die konsequente Trennung von Daten und deren Darstellung im Browser. Die Anweisungen zur Darstellung von *XML*-Daten sind entsprechend der Notation der *Extensible Style-Sheet Language (XSL) [1]* zu verfassen. Die Definitionen von Datentypen, insbesondere von individuellen, komplexen Datentypen, werden entsprechend der *Document Type Definition (DTD)* beschrieben. Die Trennung von Daten, deren Typen und Darstellung impliziert die separate Speicherung. So enthält eine *XML*-Datei die Sprachanweisungen, eine *XSL*-Datei die Formatierungsanweisungen und die *DTD*-Datei die Definition der individuellen Tags. Alle Dateien basieren auf Klartextanweisungen, werden also nicht in einem binären Format vorgehalten und sind mit einem simplen Texteditor bearbeitbar und lesbar. Zur maschinellen Verarbeitung werden die *XML*, *XSL* und *DTD*-Dateien einem Interpreter-Programm zugänglich gemacht. Dieser Interpreter verarbeitet sequentiell jede Anweisung aus den Dateien, prüft die Gültigkeit von Anweisungen auf ihre Konformität zum *XML*-Standard und führt schließlich die Anweisungen aus. Dieser serielle Vorgang wird als ‚parsing' bezeichnet. Der Begriff des *XML*-Parser ist gleichbedeutend mit dem *XML*-Interpreter. *XML*-Parser sind oftmals bereits integrativer Bestandteil vorhandener Applikationen. So ist im Programm *Internet Explorer* von *Microsoft* ab der Version 5 ein *XML*-Parser enthalten und auch die Programmiersprache *PHP*, die im Rahmen dieser Arbeit Verwendung findet, besitzt ein *XML-Parser*-Modul. Dieses XML-Parser-Modul wird in der im Rahmen dieser Arbeit erfolgten

Parser-Modul wird in der im Rahmen dieser Arbeit erfolgten Implementierung zur Dekodierung von Daten eingesetzt, die mittels des im folgenden Kapitel beschriebenen *Simple Object Access Protocol's* übertragen werden.

3.5 Simple Object Access Protocol (SOAP)

Das *"Simple Object Access Protocol" (SOAP)* ist ein vom *W3C* betreuter Standard, der 1999 zunächst in der Version 1.0 [13] ratifiziert wurde. Im Jahr 2000 folgte die Version 1.1 [14], mit der wesentlichen Änderung, dass in dieser Version auch andere Transportprotokolle als *HTTP* zugelassen sind. *SOAP* definiert ein Standardprotokoll für den Aufruf von Objekten, basierend auf Internetstandards. In der vorliegenden Arbeit wird *SOAP* für die standardisierte Kommunikation zwischen den Programmen eingesetzt und definiert die Verfahrensweise des Aufrufs von entfernten Methoden (*Remote-Procedure-Calls, RPC*), d.h. Aufrufe von Funktionen die Bestandteil separater Programme sind. *SOAP* ist damit methodisch ein wesentlicher Baustein der im Kapitel 3.8 beschriebenen *Web-Service* Spezifikation und dient in der konkreten Anwendung dem Transport von Nachrichten, die durch *RPC*'s eine Interoperabilität herstellen sollen. Die Definition von *SOAP* integriert die bereits beschriebenen Standards, so findet für die Datenkodierung *XML* Verwendung, als Transportprotokoll sind neben *HTTP* auch andere, insbesondere *SMTP* vorgeschlagen. Die *SOAP*-Spezifikation erstreckt sich auf drei Bereiche, auf die in den folgenden Unterkapiteln eingegangen wird:

1. *SOAP-Envelope*
2. *SOAP-Encoding*
3. *SOAP-RPC*

3.5.1 SOAP-Envelope

Im Abschnitt *SOAP-Envelope* wird der Container für *SOAP*-Nachrichten spezifiziert. Die mittels *SOAP* gesendete Nachricht ist ein *XML*-codiertes Dokument, das aus zwei Teilen, dem Umschlag (*SOAP*:Envelope), und dem Rumpf (*SOAP*:Body) besteht. Zusätzlich kann dem *SOAP*:Envelope ein weiteres Element, der *SOAP*:Header, zugefügt sein. Die Struktur wird nachfolgend detailliert beschrieben:

3.5.1.1 SOAP:Header

Laut *SOAP*-Standard ist das *SOAP:Header*-Element optional, sofern jedoch angewandt, muss es ein unmittelbares Element des *SOAP*:Envelope-Elements sein. Für den Fall, dass das Attribut „mustUnderstand“ im *Header*-Element den Wert „1“ trägt, ist die Auswertung zwingend. Das *SOAP*:*Header*-Element kann selbst eine Reihe von *Header*-Elementen bestehen, jedes Element kann beliebige Informationen zu Transport oder Anwendung enthalten. Bezogen auf konkrete Anwendungen könnte in einem *Header*-Element eine Transaktionsnummer kodiert sein. Mit einer Transaktionsnummer lassen sich mehrere, sich gegenseitig bedingende, Prozeduraufrufe gruppieren.

3.5.1.2 SOAP:BODY

Dieses Element ist in einer *SOAP*-Nachricht genau einmal vorhanden und ist unmittelbar vom *SOAP*:Envelope-Element umschlossen. Das *SOAP*-BODY-Element enthält den eigentlichen Inhalt der *SOAP*-Nachricht in Form von *XML* und stellt den Rahmen für Elemente dar, die zur Realisierung des Aufrufs entfernter Proceduren (*Remote-Procedure-Calls, RPC*) dienen. Die konkrete Ausprägung derartiger Funktionsaufrufe bezüglich *SOAP* ist im Kapitel 3.5.3 beschrieben.

3.5.1.3 SOAP:Fault

Dieses Element liefert Status- oder Fehlermeldungen an den Absender einer *SOAP*-Nachricht zurück. *SOAP:Fault* muss ein Teilelement von *SOAP:Body* sein und darf höchsten einmal vorkommen. Innerhalb des *SOAP-Fault*-Elements sind vier Teilelemente vordefiniert: „*faultcode*“ bezeichnet die Fehlerursache in maschinenlesbarer Form, „*faultstring*“ für die Klartextform, „*faultactor*“ als Angabe des Verursachers, insbesondere bei einem Auslösen des Fehlers durch eine Zwischenstation auf der Transportroute, „*detail*“ als in *XML*-codierte Erklärungen zur genauen Fehlerursache.

3.5.2 SOAP-Encoding

Die Kodierungsvorschrift von Daten wird laut der *SOAP*-Spezifikation durch ein *XML-Schema (XSD) [26]* definiert. Die Definition umfasst die wichtigsten Datentypen, die auch in den aktuellen Programmiersprachen verwendet werden. So sind alle gebräuchlichen Basistypen ebenso wie zusammengesetzte Werte, Strukturen und Arrays vordefiniert. Aufgrund der

eindeutigen Referenzierung des eingesetzten *XML-Schema* [95] über eine *URL* (vgl. Kapitel 3.3.4), ist die in *SOAP* verwendete Kodierungsvorschrift flexibel erweiterbar.

3.5.3 SOAP-RPC-Definition

In diesem Teil der *SOAP*-Spezifikation wird eine Sammlung von Regeln definiert, entsprechend denen die Prozedurfernaufrufe (*Remote-Procedure-Calls, RPC*) auf der Basis von *SOAP* zu implementieren sind. Daneben definiert dieser Abschnitt die Bindung des *SOAP-RPC* an das Transportprotokoll *HTTP* (vgl. Kapitel 3.3.3). Wesentliche Inhalte aus diesem Teil der Spezifikation sind:

(1) auf welche Weise und an welche Adresse die *RPC*-Anforderungen gesandt werden,
(2) wie *RPC*-Aufrufe strukturiert sein müssen,
(3) wie Argumente von Methoden bei Anforderung und Antwort strukturiert zu sein haben und
(4) wie das Verhalten im Fehlerfall festgelegt ist.

Zu (1) ist definiert, dass die Remote-Schnittstelle über ein eine *HTTP-URI* identifiziert wird. Zu (2) ist festgelegt, dass *RPC*-Aufrufe in Form von Strukturen dargestellt werden. Bezüglich (3) ist definiert, dass alle im *SOAP-Body*-Element (vgl. Kapitel 3.5.1.2) integrierten Argumente eines *RPC*-Aufrufs als separate Elemente aufzuführen sind. Die Reihenfolge der Argumente hat bei Anforderung und Antwort identisch zu sein. Entsprechend zu (4) ist niedergelegt, dass im Fehlerfall ein *SOAP:Fault*-Element zu senden ist und auf *HTTP*-Ebene die *HTTP*-Antwort den Wert „500“ haben muss, um einen Serverfehler zu signalisieren.

3.6 Web-Service Description Language (WSDL)

Die *WSDL* [90] ist eine *XML*-Grammatik und dient der Definition von elektronischen Diensten. Aktuell ist die Version 1.1 beim *W3C* seit März 2001 eingereicht, derzeit jedoch noch nicht als offizieller Standard bestätigt. Charakteristisch für *WSDL* sind die Definitionen von sogenannten Ports und Diensten. *WSDL* ist grundsätzlich nicht an ein bestimmtes Nachrichtenprotokoll, etwa an *SOAP*, gebunden. Die Protokollbindung wird stattdessen in jedem *WSDL*-Dokument eigens definiert und spezifiziert damit das Protokoll für die Schnittstelle eines Kommunikationspartners. Die aktuell zuordbaren *Bindings* an das jeweilige Nachrichtenprotokoll sind *SOAPv1.1*, sowie *HTTP:GET/POST* und die *Multipurpose-Internet-Media-*

Extensions (MIME) [12]. Letzteres wird vorwiegend in der E-Mail Kommunikation eingesetzt. In der Terminologie der *WSDL*-Spezifikation ist ein elektronischer Dienst (<*service*>) eine Menge von Kommunikationspunkten (*Ports*). Jeder dieser Kommunikationspunkte (*Port*) hat eine Adresse sowie eine Bindung, zur Deklaration eines ausgewiesenen Datenformats und der Protokolldetails. Daneben werden sogenannte *Port-Types (<portType>)* definiert, die eine abstrakte Menge von Operationen beschreiben, die von den *Ports* auf die Nachrichten (<*message*>) angewandt werden. Aktuell sind vier *Port-Types* definiert:

1. *One-way*, charakterisiert den Empfang einer Nachricht,
2. *request-response*, beschreibt das Empfangen einer Nachricht und das Versenden einer Antwort,
3. *solicit-response*, für den Versand einer Nachricht und den Empfang einer Antwortnachricht,
4. *notification*, für den Versand einer Nachricht über den Port.

Formal ist ein *WSDL*-Dokument nur vollständig und gültig, wenn die folgenden Elemente, ausgewiesen als *XML*-Tags, mindestens einmal darin enthalten sind: <definitions>, <message>, <portType>, <binding>, <service>. Das *XML-Tag* <definitions> umfasst als Wurzelelement den gesamten Definitionsbaum, die übrigen *XML-Tags* korrespondieren zu den bereits oben beschriebenen Elementen der *WSDL*-Spezifikation.

3.7 Verzeichnisdienst

Zur Nutzung von Web-Services im Internet ist das Auffinden dieser Dienste notwendig. Die üblichen Suchmaschinen-Dienste sind ungeeignet für das Auffinden von Web-Services, da sie auf die manuelle Stichwortsuche nach *WWW*-Dokumenten ausgerichtet sind. Aus diesem Grund war es notwendig, Szenarien und Protokolle zu entwerfen, die der Anforderung der programmorientierten Suche entspricht. Eine eigens mit dieser Intention entwickelte Spezifikation wird im nachfolgenden Kapitel beschrieben.

3.7.1 Universal Description, Discovery And Integration (UDDI)

Die Spezifikation der *UDDI* [81] entstand im September 2000 und beschreibt Ressourcen im Internet, kategorisiert diese und bietet über entsprechende Schnittstellen die Möglichkeit, per

Schlüsselwort, Anbietername oder Eigenschaft zu suchen. *UDDI* baut auf die Ebenen-Architektur auf, die durch *SOAP* und *WSDL* bereits vorgegeben sind. Die Absicht der Entwickler ist bereits in der Bezeichnung widergegeben: Im Vordergrund stand die Beschreibung (*Description*) zur Verbesserung der Kommunikation wirtschaftlich verkehrender Unternehmen, die Verbesserung der Suchmöglichkeiten (*Discovery*) und die nahtlose Einbindung von Diensten (*Integration*). Wesentliches Merkmal des *UDDI* ist das Verzeichnis (*Registry*), in dem Metainformationen zu den Ressourcen bzw. Diensten hinterlegt sind. Abbildung 2 zeigt das Datenstrukturdiagramm des *UDDI*-Registers, dargestellt in der Notation *Unified-Modelling-Language (UML).*

Hervorzuheben sind die Einträge <*businessEntity*>, <*businessService*>, <*bindingTemplate*> und das <*tModel*>. Im <*businessEntity*> wird das Unternehmen, bzw. die Institution benannt (<*name*>), beschrieben (<*description*>), kategorisiert (<*categoryBag*>) und gegebenenfalls identifiziert (<*identifierBag*>), zum Beispiel über eine Handelsregisternummer im Falle eines Unternehmens. Vom *UDDI*-Register wird zudem ein eindeutiger Schlüsselwert generiert (<*businessKey*>). Die Registereinträge <*businessEntity*> und <*businessService*> sind miteinander durch Verweise in den Attributen <*businessServices*> bzw. <*businessKey*> verbunden. Der Einträge im <*businessService*> beschreiben den Dienst, den der <businessEntity> erbringt. Auch hier kann eine Kategorisierung mittels <*categoryBag*>-Attribut vorgenommen werden, ebenso wird der Dienst benannt (<*name*>) und beschrieben (<*description*>). Der Dienst erhält bei der Eintragung in das Register ebenfalls einen eindeutigen Schlüssel zugewiesen (<*serviceKey*>). Das <*bindingTemplate*> beschreibt die Kommunikationschnittstelle des Dienstes (<*businessService*>). Die Verbindung zwischen <*businessService*> und <*bindingTemplate*> besteht durch die Verweise als <*bindingTemplates*> bzw. <*bindingTemplate*>*:*<*serviceKey*>. Der Eintrag in <*hostingRedirector*> erlaubt Verweise auf ein anderes <*bindingTemplate*>. Bezüglich der *Web-Service* -Thematik ist das <*tModel*> elementar, denn es repräsentiert die Beschreibung (<description>, <name>) der Schnittstelle zum *Web-Service*. Der Eintrag in <*overviewDoc*> enthält die *URL* auf das *WSDL*-Dokument des *Web-Service.*

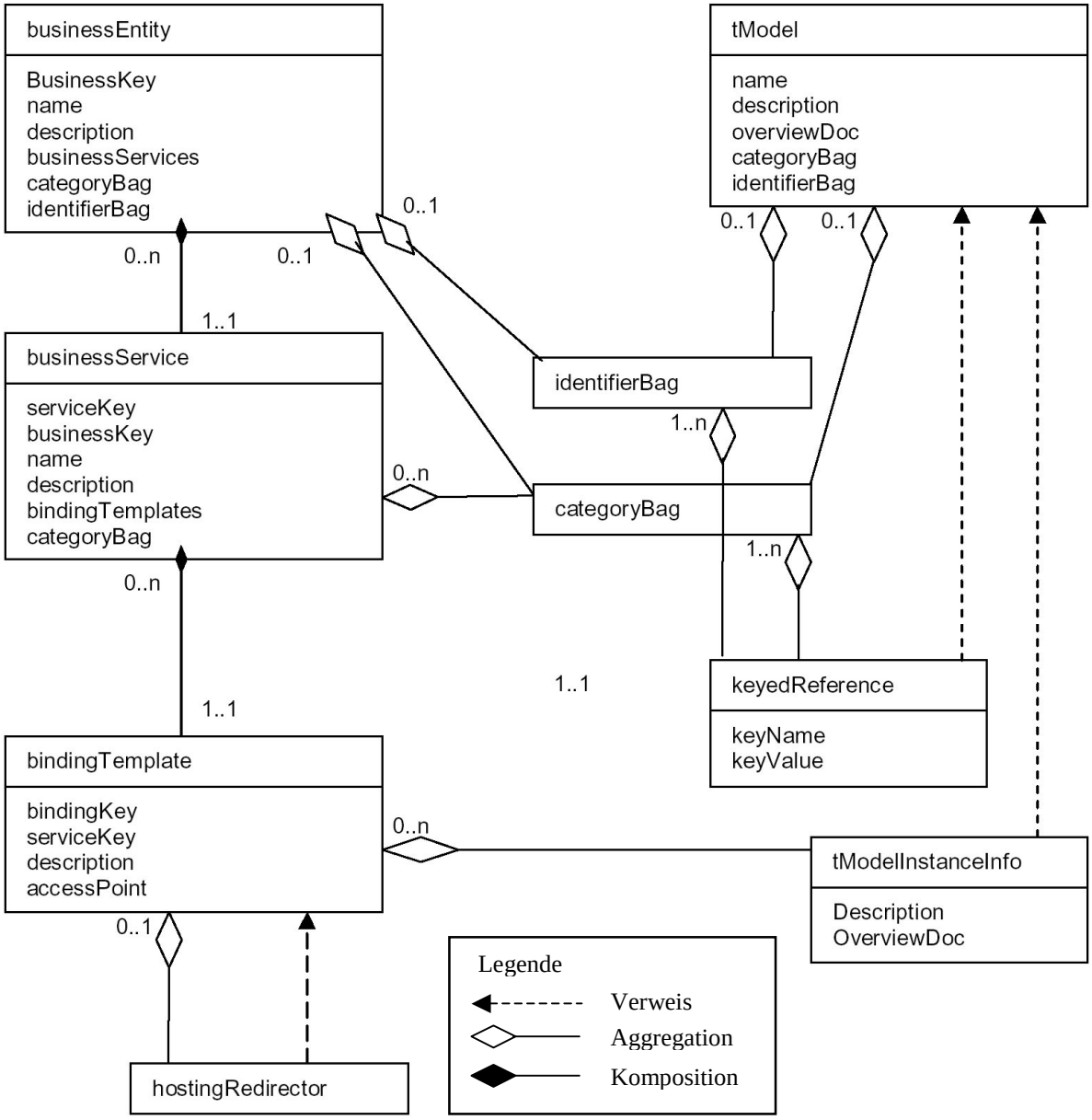

Abbildung 2: *UML*-Darstellung der Datenstruktur des *UDDI*-Registers aus der offiziellen Spezifikation des Standards in [81]. Die gestrichelt dargestellten Verweise sind Bindungen zwischen den Objekten. Die leere Raute stellt die einfache Aggregation dar, aus der die Zusammensetzung eines Objekts hervorgeht. Die mit der ausgefüllten Raute dargestellte Komposition ist eine strenge Form der Aggregation, entsprechend der Teilobjekte abhängig vom Aggregat sind. Wird ein Aggregat im Rahmen einer Komposition gelöscht, so werden auch dessen Teilobjekte gelöscht.

Um das *UDDI*-Register auf unterschiedliche Nutzerprofile vorzubereiten, ist es in drei logische Bereiche unterteilt. Das sind die sogenannten *White- Yellow-* und *Green-Pages,* die nachfolgend erläutert werden.

Die *White-Pages* enthalten allgemeine Angaben über den Anbieter, so zum Beispiel eine Firmeninfo, Kontaktinfo, allgemeine Dienstbeschreibung oder Identifikationsnummern, wie zum Beispiel Steuernummern.
Der Bereich der *Green-Pages* ist den technischen Beschreibungen vorbehalten. Dazu zählen E-Business-Regeln, technische Dienstbeschreibungen, Angaben zu Dienstaufrufen und Datenbindung.
Die *Yellow-Pages* repräsentieren einen Dienst- und Produktindex, die Zuordnung zu einem Industriezweig und die geographische Einordnung.
Die Interaktion mit einem *UDDI*-Register, zum Beispiel Publizieren oder Anfragen von Informationen, erfolgt über *SOAP*-Nachrichten. Das Register stellt somit selbst einen *Web-Service* dar. Listing 1 zeigt eine beispielhafte *SOAP*-Nachricht, die bei einem *UDDI*-Register nach der Bezeichnung „UKBF“ (<name>) als eingetragene Unternehmung anfragen würde.

```
<find_business generic="1.0" xmlns="urn:uddi-org:api">
  <findQualifiers>
         <findQualifier>caseSensitiveMatch</findQualifier>
  </findQualifiers>
  <name>UKBF</name>
</find_business>
```

Listing 1: *SOAP*-Nachricht zur Anfrage bei einem *UDDI*-Registers nach einem *Business*-Eintrag

3.7.2 Öffentliche UDDI-Register

Exemplarische *UDDI*-Register werden aktuell von den Firmen *IBM* [34] und *Microsoft* [53] zur kostenfreien Nutzung angeboten. Zu diesen Registern existieren sowohl Schnittstellen auf Applikationsebene mittels *SOAP*, als auch *Human-Computer-Interfaces* (*HCI´s)*. Die Schnittstellen erlauben das Eintragen, Ändern, Löschen und Suchen innerhalb des Registers. Eine einmalige Benutzerregistrierung ist bei beiden Anbietern obligatorisch. Die Benutzerregistrierung bei der Firma *Microsoft* stellt insofern ein Sonderfall dar, als sich die persönliche Registrierung sich nicht auf den *UDDI*-Dienst beschränkt sondern für sämtliche, von Microsoft angebotene Dienste gilt. Die Benutzung der *HCI* ist in beiden Fällen ähnlich, da die zu erhebenden Dateneingaben bereits durch die *UDDI*-Spezifikation vorgegeben sind. Zudem ist es zweitrangig, welcher Anbieter für die Eingabe der Daten bevorzugt wird, da täglich eine Datensynchronisation zwischen beiden *UDDI*-Registern durchgeführt wird. Eine bebilderte und

bis auf Einzelschritte detaillierte Anleitung zur Bedienung des *HCI* des *IBM-UDDI*-Dienstes ist online [30] bei *IBM* verfügbar.

3.8 Web-Services

Den strukturelle Zusammenhang der in den vorausgegangenen Kapiteln beschriebenen Komponenten zur Bildung eines *Web-Services* zeigt Abbildung 3. Es wird deutlich, dass ein *Web-Service* eine kombinierte Funktion verschiedener Netzwerk- und Protokollspezifikationen darstellt. Die Bezeichnung *Web-Service* gilt als Synonym für Internetbasierte Dienste mit einem aktiven, in *XML* definierten, *Application Programming Interface* (*API*). Während *SOAP* den Netzwerkprotokollstapel und die Datenrepräsentation nach *XML* vereinigt, *WSDL* die Bausteine *XML* und *SOAP* integriert, *UDDI* wiederum auf *WSDL* und *SOAP* basiert, vereinigt ein *Web-Service* diese ineinander geschachtelten Spezifikationen in einer Anwendung. Damit repräsentiert ein *Web-Service* eine technologische Basis für internetfähige Applikationen. Zugleich enthält ein *Web-Service* über *SOAP*-basierte Schnittstellenfunktionen aktive Dienste, die über das Internet von anderen Applikationen eingebunden werden können. Aufgrund der flexiblen Verbindung mittels Austauschs von *SOAP*-Nachrichten zwischen Applikation und dem *Web-Service*, wird die spontane Vernetzung hinsichtlich einer Integration von Diensten, vereinfacht.

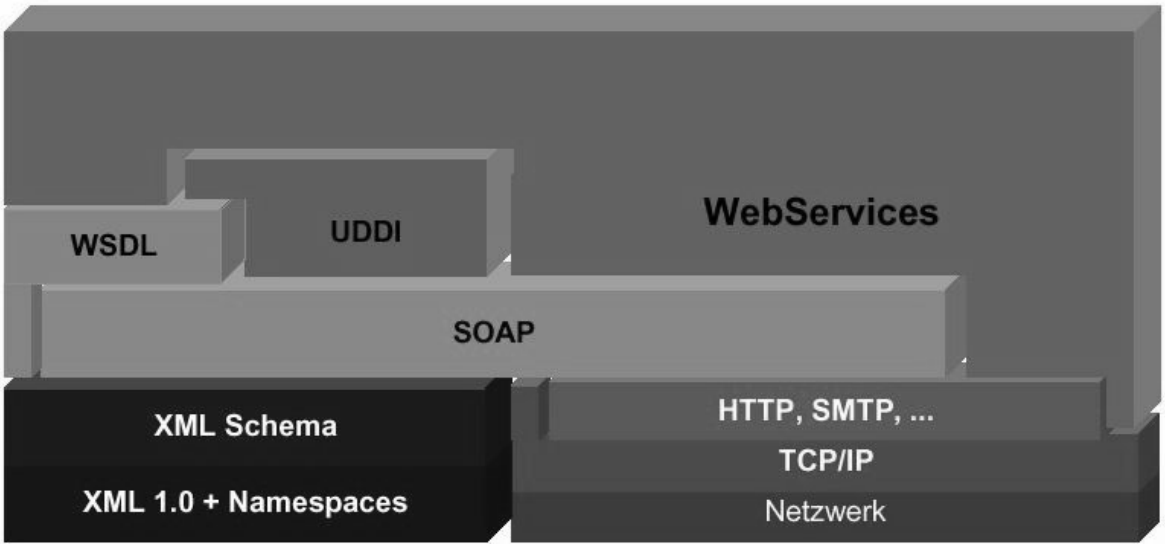

Abbildung 3: Darstellung der gestapelten Definitions- und Protokollschichten aus [41]. Web-Services bilden ein vereinigendes Fundament für Applikationen. Der Baustein SOAP stellt ein zentrales Brückenelement zwischen XML und den Netzwerk- und Transportprotokollschichten dar.

3.9 Programmiersprache

3.9.1 PHP Hypertext Preprocessor

PHP [61] ist eine Scriptsprache, die meist serverseitig zur Generierung dynamischer Webseiten eingesetzt wird, wie jede andere Programmiersprache jedoch auch zur Lösung von Programmieraufgaben clientseitig eingesetzt werden kann. Die erste Version entstand 1994 von Rasmus Lerdorf. Die Bezeichnung der Scriptsprache ist eine Abkürzung für *„PHP HyperText Preprocessor“* und enthält eine innere Rekursion. *PHP* ist verfügbar als *Open-Source*-Lizenz und wird damit als Gemeingut der Entwicklergemeinschaft stetig weiter entwickelt. Die hier eingesetzte Variante der Scriptsprache *PHP* ist die der Version 4 (*PHP-4)*. Die mittels *PHP* entwickelten Scripte können sowohl als einzeln interpretierbare Programme ausgeführt werden, sie sind jedoch auch in ein *HTML*-Dokument integrierbar. Entsprechend konfigurierte *WWW-Server*-Applikationen interpretieren den eingebetteten Scriptteil beim Versenden der Ausgabe an den anfordernden Browser. Das folgende Beispiel erzeugt ein *HTML*-Dokument, das bei der Ansicht mittels eines *WWW-Browsers* auf dem *Client* die Ausgabe des aktuellen Datums inklusive Uhrzeit sichtbar macht:

```
<html>
  Das aktuelle Datum / Uhrzeit ist:
<?php echo date („M d, Y h:m:s A”) ?>
</html>
```

Listing 2: Beispiel einer in einer *HTML*-Datei eingebetteten *PHP*-Funktion zur Ausgabe der Zeitinformation des *Servers*.

Es handelt sich bei der Zeitinformation aufgrund der serverseitigen Ausführung des eingebetteten Scripts, um die des Servers.

Die Vielzahl der verfügbaren funktionalen Erweiterungen zu *PHP* geben dem *Software*-Entwickler ein universelles Programmierwerkzeug an die Hand: In der ursprünglich beabsichtigten Funktion, der Generierung von *HTML*-Dokumenten, existieren *PHP*-Funktionen zur graphischen Repräsentation von Daten, die unmittelbare Steuerung von *mySQL*-Datenbanken, als auch das Generieren von Dokumenten entsprechend des *„Portable Document Formats“ (PDF)*, welches von der Firma *Adobe* [2] entwickelt wurde. Aufbauend auf den *PHP*-Funktionsumfang zu *XML*, sind Funktionsbibliotheken entstanden, die sich dem Bereich *Web-*

Services widmen, in der die vorliegende Implementierung eingesetzt und nachfolgend beschrieben werden.

3.9.2 PHP-Bibliothek SOAPx4

Implementierungen des *SOAP*-Standards in *PHP* sind selten aufzufinden. *SOAPx4* [6] ist von Dietrich Ayala entwickelt und als *Open-Source*-Lizenz frei gegeben. Die Implementierung, die den *XML-Interpreter*-Kern der Basis von *PHP-4* nutzt, lässt sich wahlweise in *Server*- oder *Client*-Position einsetzen. Hinsichtlich der *Server*-Funktionalität, steht eine validierte Interoperabilität mit *Client*-Implementierungen im Mittelpunkt des Interesses. Die Ergebnisse [64], [7] der durchgeführten und Überprüfungen mit einer Vielzahl von *Client*-Implementierungen bescheinigen der *SOAPx4*-Bibliotheks-Implementierung eine stabile und standardkonforme Interoperabilität.

3.9.3 PHP-Plattform PHProjekt

Die Implementierung *PHProjekt* [62] stellt ein Rahmenwerk für Gruppenarbeitsfunktionalität dar und bietet bereits in der Grundausstattung eine Reihe anwendbarer Module. Zu diesen Modulen gehören ein Terminkalender, die Verwaltung und Speicherung gemeinsamer Daten, ein Kommunikationssystem zum synchronen Austausch Text-Nachrichten, sogenannter „*Chats*“. Die praxisnahen Einsatzmöglichkeiten sind in Kapitel 5.2 dargestellt. *PHProjekt* ist als *Open-Source* frei gegeben und lässt sich dementsprechend durch individuelle Implementierung erweitern und in diese integrieren.

3.9.4 API-Bibliothek PocketSOAP

PocketSOAP [27] ist eine im Jahr 2001 von Simon Fell entwickelte Funktionsbibliothek, zur *client*-seitigen Kopplung an einen *SOAP-Web-Service*. Die als Komponente integrierbare Funktionsbibliothek ist kostenlos für die Implementierungs- bzw. Betriebssystemplattformen Windows von Microsoft verfügbar. Ursprünglich wurde die Komponente für das von *Microsoft* entwickelte Betriebssystem „*PocketPC2002*“ der sogenannten *Personal Digital Assistants (PDA).* Vor diesem Hintergrund entstand die Bezeichnung der Komponente. Inzwischen ist die Komponente für alle aktuellen Betriebssystemplattformen von *Microsoft-Windows* ab Version *Windows95* verwendbar und bietet damit dem Programmierer eine identische funktionale Schnittstelle im Umfeld von *PDA* und Desktopsystemen. Im Rahmen dieser Arbeit wird

diese Komponente als Verbindungsstück zwischen einer *Office*-Applikation und dem *Web-Service* eingesetzt (vgl. Kapitel 4.1)

3.10 World Wide Web Server (WWW-Server)

Zentrale Speicherorte, von denen die Informationsträger für das *„Word-Wide-Web“ (WWW)* abgerufen werden, sind die *WWW-Server*. Ein *WWW-Server* ist ein Programm, dass im Internet erreichbar ist und den *HTTP*-Standard (vgl. Kapitel 3.3.3) funktional implementiert. Es existieren für sämtliche Computerplattformen entsprechende Implementierungen. Im nachfolgenden Kapitel 3.10.1, wird eine Implementierung vorgestellt, die kostenlos und im Quellcode erhältlich ist und die dementsprechend für eine Reihe von Plattformen verfügbar ist.

3.10.1 Apache-Implementierung

Bereits im Jahr 2000 wird der *Apache*-Implementierung von *WWW-Servern* ein Anteil von mehr als schätzungsweise 60% [57] eingeräumt. Die *Apache*-Implementierung [4] ist als *Open-Source*-Lizenz verfügbar und wird von einer großen Anzahl von freien Entwicklern stetig weiterentwickelt. Schwerpunkte der Implementierung lagen in den Bereichen Robustheit, Leistungsfähigkeit und Sicherheit. Der hohe Anteil von *Apache* an aktiven Systemen im Internet belegt die erfolgreiche Umsetzung der gestellten Anforderungen.

3.10.2 Verbindung zu PHP

Die Basis der *Apache*-Implementierung ist durch ein System von Erweiterungsmodulen offen gehalten, um eine Anpassung an spezielle funktionale Anforderungen zu gewährleisten. Im Umfeld der Anforderung, *Script*-Programme serverseitig auszuführen, prädestiniert sich das Modul „*mod_php*“, welches einen *PHP-Interpreter* in den *Apache-WWW-Server* einbettet. Erzielt wird dadurch eine höhere Leistungsfähigkeit im Vergleich zu einer separaten Abarbeitung des *PHP-Scriptcodes* mittels eines externen *PHP-Interpreter*, der für jeden Scriptblock auf dem Server erneut gestartet werden müsste.

3.11 Teaching File Server der Radiologie

Gegenstand für die Integration und Adaption in das methodische Konzept dieser Arbeit, ist der web-basierte, radiologische *Teaching File Server* [33] der Klinik und Poliklinik im

UKBF. Kern dieses *Servers* ist eine auf Datenbankeinträge basierende aktive Programmlogik, die sich über einen *Browser* steuern lässt und dynamisch zur Anfrage passende *HTML*-Seiten generiert. Inhaltlich verfügt dieser *Server* über eine digitale Lehrbildsammlung, in der Bild- und Textmedien über eine entsprechende Datenbankabfrage flexibel verknüpft sind. (Abbildung 4).

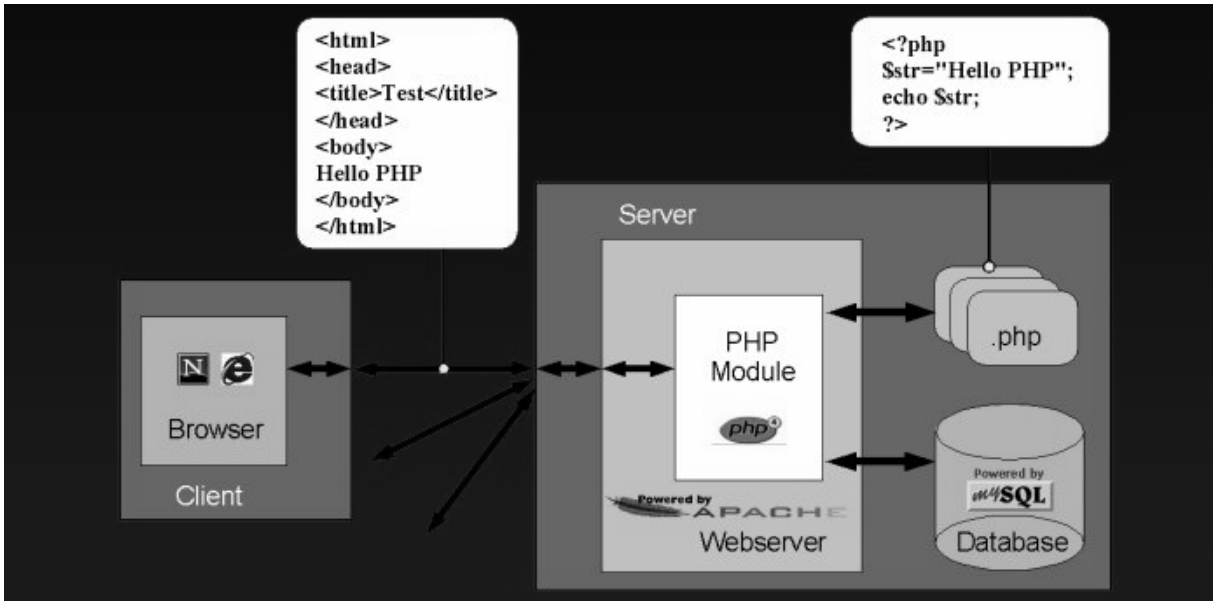

Abbildung 4: Schema der Struktur des radiologischen *Teaching File Server*. Das Servermodul besteht aus einer *mySQL*-Datenbank, *PHP*-Programmen und einem *Apache-Web Server*. Für den *Client* werden mittels des *PHP*-Moduls dynamisch *HTML*-Seiten generiert, die mittels Browser zur Anzeige gebracht werden.

Einsatzziel dieses Servers ist die telematisch betriebene Fort- und Weiterbildung von Ärzten und Studenten über das Internet. Die radiologischen Bilddaten sowie zugehörige Anmerkungs- und Befundtexte sind in einer Datenbankapplikation gekapselt. Die zugrunde liegende Datenbank basiert auf der kostenlosen SQL-Datenbankapplikation „mySQL“. Die Datenbankapplikation ist über eine WWW-Benutzerschnittstelle sowohl über das Intra- als auch über das Internet für den Anwender nutzbar. Zur Vermeidung finanzieller und lizenzrechtlicher Barrieren sind ausnahmslos kostenfreie und frei verfügbare Software-Komponenten eingesetzt. Zusätzlich sind die Mindestanforderungen hinsichtlich der Software-Anforderung niedrig gehalten. Dementsprechend wurde auf aktive Softwarekomponenten auf Anwenderseite verzichtet, die neben einem entsprechenden Ressourcenbedarf unter gewissen Umständen auch zusätzliche Sicherheitsvorkehrungen zum Datenschutz erforderlich gemacht hätten. Der Benutzer interagiert über die im *Browser* dargestellte Schaltelemente, wie nachfolgend als Bildschirmfoto dargestellt:

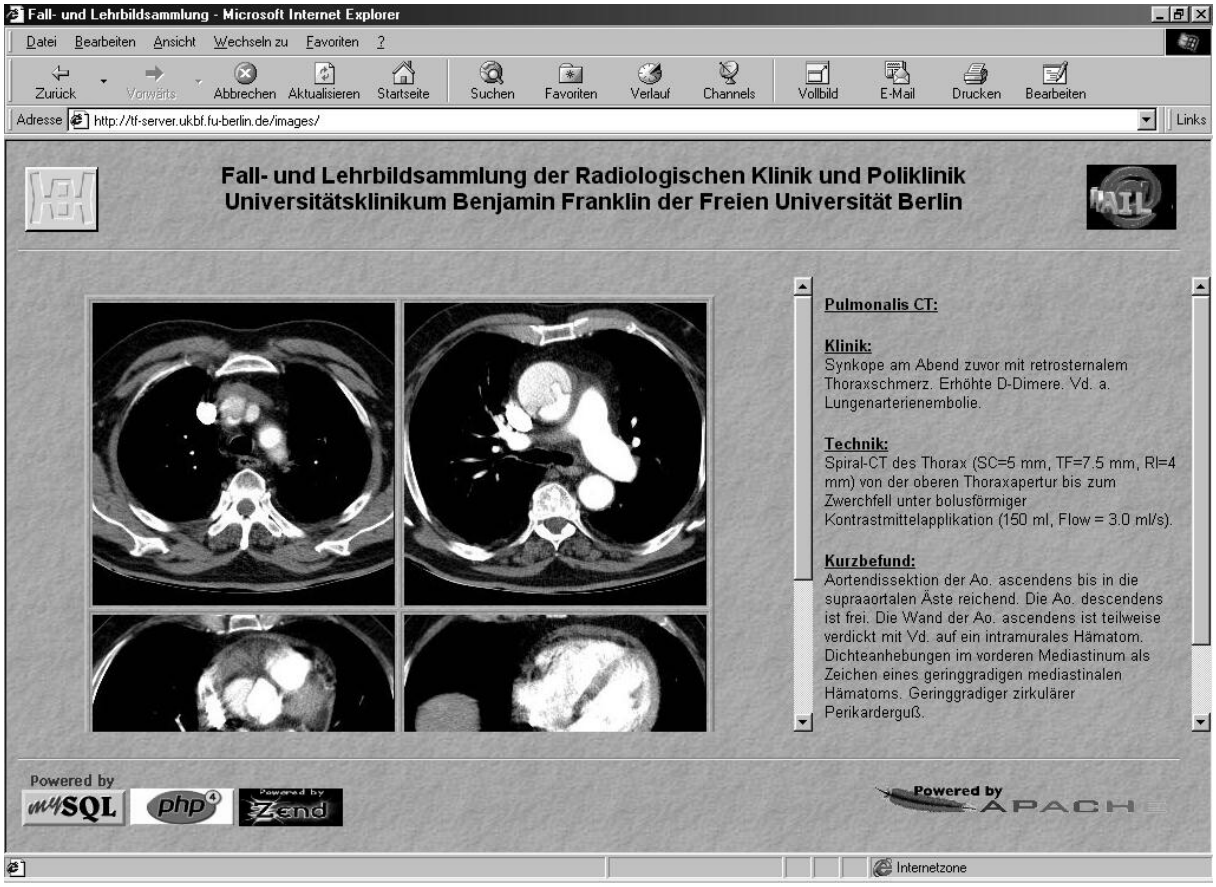

Abbildung 5: Darstellung eines Ergebnisses der Suche nach einer „Aortendissektion“ im Rahmen der *HTML*-basierten Benutzerschnittstelle im Internet-*Browser* des Anwenders. Die grafische Darstellung und die Interaktionsmöglichkeiten werden dynamisch mittels in PHP implementierter Programmlogik auf dem Server generiert.

Die Programmlogik zur dynamischen Generierung der Benutzeroberfläche und die softwaretechnische Datenbankanbindung wurde serverseitig konzentriert, als Implementierungssprache wurde *PHP* herangezogen. Diese Ebene der Programmlogik markiert den Verknüpfungspunkt zu dem im Rahmen dieser Arbeit entwickelten *Web-Service*. Konzeptionell wird der *SOAP*-Adapter zur Interoperabilität eingesetzt und offeriert eine Schnittstelle, die in Kombination mit der grafischen Benutzeroberfläche einen gezielten und in *XML* standardisierten Zugriff auf die in der Datenbank gekapselten Information ermöglicht.