Aus dem

Institut für Medizinische Biometrie und Informatik
Abteilung Medizinische Informatik
(Direktor: Prof. Dr. R. Haux)

und dem

Labor *Computerunterstützte Ausbildung in der Medizin*
(Leitung: Prof. Dr. Dr. h.c. H.-G. Sonntag, Prof. F.J. Leven)

Plattformunabhängige, adaptive Lehr-/Lernsysteme für die medizinische Aus- und Weiterbildung

Inauguraldissertation
zur Erlangung des Doctor scientiarum humanarum (Dr. sc. hum.)
der Medizinischen Fakultät der Ruprecht-Karls-Universität Heidelberg

vorgelegt von
Martin Haag
aus Brettheim
1998

„Der verlorenste aller Tage ist der, an dem du nicht gelacht hast.“
A. DE SAINT-EXUPÉRY

Dekan: Prof. Dr. Dr. h.c. H.-G. Sonntag
Betreuer: Prof. Dr. R. Haux

Zusammenfassung

Lehr-/Lernsysteme können einen wichtigen Beitrag zu einer qualitativ hochwertigen Medizinausbildung liefern. Sie ermöglichen es, Lernerfahrungen in der *realen* Praxis zu vertiefen bzw. die Studierenden besser auf diese Praxis vorzubereiten. Die zur Zeit verfügbaren CBT-Systeme (Computer-based Training) weisen jedoch einige konzeptuelle Schwächen auf, die neben der fehlenden Einbindung in die Curricula für deren vergleichsweise geringen Einsatz in Deutschland mitverantwortlich sind.

In der vorliegenden Arbeit werden u.a. Antworten auf folgende Fragen erarbeitet:

- Welche konzeptuellen Merkmale muß ein Lehr-/Lernsystem aufweisen, um die Schwächen existierender CBT-Systeme zu beseitigen?
- Wie kann eine häufigere Nutzung von Lehr-/Lernsystemen durch die Studierenden erreicht werden?
- Wie kann die Akzeptanz von Lehr-/Lernsystemen bei den Medizindozenten verbessert werden?

Das erarbeitete Lösungskonzept sieht eine Lehr-/Lernsystemshell vor, mit deren Hilfe Fachgebietsexperten ohne Informatikkenntnisse qualitativ hochwertige Lehr-/Lernsysteme erstellen und weltweit verfügbar machen können. Der gewählte Shellansatz ermöglicht die leichte Wiederverwendbarkeit sowohl der Domäneninhalte als auch der Lehr-/Lernsystemfunktionalität. Die Shell basiert auf umfangreichen Datenstrukturen, welche mittels objektorientierter semantischer Datenmodellierung beschrieben werden. Dabei wurde auf möglichst weitgehende Fachgebietsunabhängigkeit geachtet.

Dozenten können mit dem Autorensubsystem der Shell Lehr-/Lernsysteme erstellen, die ihren persönlichen Anforderungen entsprechen. Die Lehr- und Lernsubsysteme der Shell basieren vollständig auf Internet-Anwendungen und Internet-Standards. Sie sind plattformunabhängig. Dadurch ist gewährleistet, daß die erstellten Systeme auf allen Rechnern mit Internet-Zugang nutzbar sind, also auch auf privaten Rechnern der Studierenden und Dozenten durch Einwahl über Modem oder ISDN-Adapter. Zur Lösung der Performance-Probleme internetbasierter Lehr-/Lernsysteme wird eine 7-Schichten-Architektur vorgeschlagen.
Die erstellten Systeme sind adaptiv und adaptierbar, d.h. sie passen sich an ihre Nutzer an, können aber von diesen auch selbst, abhängig von den persönlichen Vorlieben, angepaßt werden.

Die Realisierung der erarbeiteten Konzepte erfolgte bzw. erfolgt im Rahmen des CAMPUS-Projektes (Computerunterstützte Aus- und Weiterbildung in der Medizin durch plattformunabhängige Software). Erste Einsatzgebiete der CAMPUS Lehr-/Lernsystemshell sind Pädiatrie und Infektiologie.

Inhaltsverzeichnis

1 Einführung

1.1 Problematik und Motivation

Bereits Anfang der sechziger Jahre, nach Einführung der zweiten Großrechnergeneration an den Universitäten, wurde mit der Entwicklung von CBT-Systemen (Computer-based Training) begonnen [OWEN, HALL et al. 65]. Obwohl der Preis für Rechnerleistung seither um ein Vielfaches gesunken ist und mittlerweile auf vielen Schreibtischen multimediafähige Computer stehen, hat die computerunterstützte Ausbildung in der Medizin [LEVEN, SCHULZ et al. 95; KLAR, BAYER 90; KALLINOWSKI, MEHRABI et al. 97, BAUR, MICHAELIS (Hrsg.) 90] in Deutschland immer noch keinen sehr hohen Stellenwert erreicht. Die Universität Heidelberg bildet hier keine Ausnahme. Auch dort setzen nur wenige Medizindozenten CBT-Systeme im Rahmen ihrer Lehrveranstaltungen ein. Ein Grund hierfür ist die teilweise fehlende Akzeptanz der computerunterstützten Ausbildung bei den Dozenten, die häufig auf mangelnde Kenntnis über qualitativ hochwertige CBT-Systeme für das eigene Fachgebiet und deren Möglichkeiten und Grenzen zurückzuführen ist. Um diesem Informationsdefizit wirkungsvoll zu begegnen wurde Ende 1993 an der Universität Heidelberg das Labor *Computerunterstützte Ausbildung in der Medizin* [LEVEN, ALLE et al. 95] eingerichtet. Diese von der medizinischen Gesamtfakultät getragene Einrichtung hat sich mittlerweile gut bewährt und das Informationsdefizit bzgl. CBT konnte durch verschiedene Aktivitäten [HAAG 95] reduziert werden. Allerdings fehlt immer noch eine bessere Einbindung von CBT in die Curricula. Auch weisen Dozenten ihre Studenten immer noch sehr selten auf gute Lehr-/Lernsysteme hin. Der hohe Anteil an MC-Tests (Multiple-Choice) bei Physikum und Staatsexamen tut ein übriges, damit das Angebot der Mediotheken oder Showrooms, die den Studenten genauso wie die Bibliotheken als Angebot zur selbständigen Nutzung zur Verfügung stehen, nur wenig in Anspruch genommen wird. Nach Erfahrungen, die an verschiedenen Universitäten gemacht wurden, werden die Mediotheken lediglich von etwa zwei bis fünf Prozent der Medizinstudenten genutzt. Für die Prüfungsvorbereitung nutzen Studenten lieber die Kurzlehrbücher der sogenannten „Schwarzen Reihe“ (z.B. [BOB, ERDINGER 94]), da diese optimal auf die MC-Tests vorbereiten. Die effiziente Vorbereitung auf MC-Tests kann nicht Intention von CBT-Systemen sein. Die Stärke derartiger Systeme, die es gezielt für die medizinische Ausbildung zu nutzen gilt, liegt in der Interaktivität. So ermöglichen computergestützte Fallsimulationen beispielsweise, daß Medizinstudenten das in Lehrveranstaltungen angeeignete theoretische Wissen ohne Belastungen für Patienten an virtuellen Patienten anwenden lernen können, bevor sie mit „realen“ Patienten in Kontakt kommen. Die Forderung der Ärztlichen Approbationsordnung (ÄAppO) §2 Absatz 2 [ÄAPPO 89], daß unzumutbare Belastungen durch Unterricht zu vermeiden sind, kann dadurch erfüllt werden. Medizinstudenten haben durch den Einsatz von CBT-Systemen die Möglichkeit, ihr gelerntes Wissen häufiger anzuwenden und so zu festigen.

Auch der Wissenschaftsrat steht der Verwendung von CBT-Systemen in der Medizinerausbildung positiv gegenüber:

> „Praxisbezug heißt auch Kontakt des Studenten mit praktisch-klinischen Problemen, Diskussionen von Kasuistiken und Lösungsstrategien, von Differentialdiagnosen und Differentialtherapie. Apersonale Medien könnten hier gewinnbringend eingesetzt werden. Audiovisuelle und computergestützte Lernprogramme bzw. Arbeitsplätze können zwar die praktischen Erfahrungen nicht ersetzen, sind aber wertvolle Hilfsmittel zur Vor- und Nachbereitung von Seminaren und zum Wissenserwerb über seltene Krankheiten oder besonders schwierige Gegenstände. Die volle Entfaltung des didaktischen Potentials apersonaler Medien wird jedoch nur gelingen, wenn die Hochschullehrer aktiv an der Erstellung der Programme mitwirken, die jeweilige Erstellung mit dem Curriculum abgestimmt ist und die Studierenden ausreichenden Zugang - z.B. zu einer Mediothek haben.“ [WISSENSCHAFTSRAT 92 S. 46]

In seinen neuesten *Empfehlungen zur Hochschulentwicklung durch Multimedia in Studium und Lehre* schreibt er:

> „Multimediale Studienangebote vertiefen Lernerfahrung in der ‘realen’ Praxis (z.B. Laborarbeit, Untersuchung am Patienten durch eine ‘virtuelle’ Praxis; sie eröffnen neue Möglichkeiten, solche Praxis effizient vorzubereiten und Trainingsmöglichkeiten in simulierten Umgebungen zu unterstützen. In der Medizin gibt es dafür eindrucksvolle Beispiele. Es ist davon auszugehen, daß in allen Bereichen, wo das Training wichtiger Bestandteil der zu vermittelnden Berufsfähigkeit und -fertigkeit ist, Multimedia für Lehre und Studium schnell an Bedeutung gewinnen wird. Dies gilt gerade auch für Gebiete, wo die Ausbildungserfordernisse in realen Umwelten auf Hindernisse und Akzeptanzprobleme stoßen; so kann beispielsweise durch Multimedia-Einsatz auf Tierversuche zu Studienzwecken weitgehend verzichtet werden.“ [WISSENSCHAFTSRAT 98 S. 5]

Die Erfahrungen im Labor *Computerunterstützte Ausbildung in der Medizin* der Universität Heidelberg zeigen, daß Medizinstudenten sehr gerne multimediale Fallsimulationen bearbeiten. Reine Browsingsysteme, insbesondere solche ohne multimediale Bestandteile, finden dagegen bei den Studenten deutlich geringere Beachtung. Leider ist das Angebot an qualitativ hochwertigen Lehr-/Lernsystemen auf dem kommerziellen Markt immer noch nicht groß. Die medizinischen Fachverlage verlassen sich ausschließlich auf die Universitäten und entwickeln praktisch keine CBT-Systeme selbst. Dies ist sicherlich sinnvoll, weil das erforderliche fachliche Know-how dort konzentriert vorhanden ist. Allerdings bedeutet es ebenfalls, daß sich die Benutzungsschnittstellen, auch bei den Angeboten eines einzigen Verlages, meist sehr stark unterscheiden. Dadurch steigt der Einarbeitungsaufwand bei der Nutzung von CBT-Systemen. Im Gegensatz zu Lehrbüchern, bei denen die Verlage mit geringem Aufwand durch Einsatz von Formatvorlagen ein Lehrbuch in das verlagstypische Layout bringen können, sind Änderungen an der Benutzungsschnittstelle von Lehr-/Lernsystemen in der Regel mit großem Aufwand verbunden.
Die Verlage vermarkten die an den Universitäten eingekauften bzw. die dort im Auftrag entwickelten Lehr-/Lernsysteme leider häufig zu für Studierende viel zu hohen Preisen. An dieser Situation wird sich in nächster Zeit auch wenig ändern, weil die Kosten für die Entwicklung von Lehr-/Lernsystemen sehr hoch sind und die Nachfrage auf der anderen Seite relativ gering ist. Ein kostendeckendes Engagement der Verlage ist in diesem Bereich zur Zeit also nur schwer möglich. Die medizinischen Hochschulen müssen deshalb

auch in Zukunft Ressourcen in die Erstellung qualitativ hochwertiger CBT-Systeme investieren, wenn sie solche Systeme in der Ausbildung einsetzen wollen. Dies ist insbesondere auch vor dem Hintergrund der seit vielen Jahren geplanten Neufassung der Approbationsordnung für Ärzte (z.B. [BM GESUNDHEIT (Hrsg.) 93]) erforderlich, die u.a. einen verringerten Anteil der MC-Tests an den Prüfungen vorsieht. Die Nachfrage nach guten CBT-Systemen wird dadurch in Zukunft wahrscheinlich zunehmen.
Erforderlich ist für die Zukunft auch eine noch intensivere Zusammenarbeit bei der Entwicklung von CBT-Systemen zwischen den Hochschulen, um zum einen teure Parallelentwicklungen zu vermeiden und zum anderen Know-how (auch fachübergreifend) auszutauschen. Arbeitsgruppen von medizinischen Fachgesellschaften wie die Arbeitsgruppe *Computergestützte Lehr- und Lernsysteme in der Medizin* der GMDS (Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie) [CONRADI, KREUTZ et al. (Hrsg.) 97; ADLER, DIETRICH et al. (Hrsg.) 98] bieten hierfür ein geeignetes Forum. Auch hochschulübergreifende Projekte wie das VIROR-Projekt (Virtuelle Hochschule Oberrhein) [VIROR], an dem das Labor *Computerunterstützte Ausbildung in der Medizin* beteiligt ist, bieten hervorragende Möglichkeiten für den Informationsaustausch.
Die Einrichtung von WWW-Servern zur Darstellung von CBT-Aktivitäten an der eigenen Universität (Einsatzgebiete von Lehr-/Lernsystemen, aktuelle Entwicklungsprojekte...) und zur Bereitstellung von allgemeinen Informationen zur computerunterstützten Ausbildung in der Medizin ist eine weitere gute Möglichkeit, um den Informationsaustausch zu fördern [CBT-SERVER].
Darüber hinaus bietet das World Wide Web [LOWE, LOMAX et al. 96] völlig neue Perspektiven für die Realisierung von plattformunabhängigen, netzbasierten Lehr-/Lernsystemen. Gegenüber konventionellen CBT-Systemen [AUHUBER 98, HOOPER, J. O'CONNOR et al. 95, PASTERKAMP 91], die auf einer bestimmten Rechnerplattform mit einem dort lauffähigen Autorentool erstellt werden und häufig auch nur auf dieser Rechnerplattform lauffähig sind, bieten solche Systeme wesentliche Vorteile [HAAG, MAYLEIN et al. 98]. So können die im World Wide Web bereitgestellten Systeme auch von den Medizinstudenten zu Hause, ohne daß sie installiert werden müssen, genutzt werden. Dadurch könnte eine häufige Forderung der Medizinstudenten, daß sie Lehr-/Lernsysteme zu Hause auf ihrem eigenen Rechner bearbeiten möchten, erfüllt werden. Netzbasierte und plattformunabhängige CBT-Systeme werden auch in der Lage sein, Teleteaching-Ansätze, wie sie u.a. an den Universitäten Heidelberg und Mannheim zur Zeit entwickelt werden [EFFELSBERG 95], sinnvoll zu ergänzen. Studierende könnten z.B. nach Abruf von aufgezeichneten Vorlesungen in die Lage versetzt werden, das gelernte Wissen sofort an virtuellen Patienten anwenden zu lernen. Und dies unabhängig von der Verfügbarkeit von Dozenten und Patienten sowie von Tageszeit und Ort.

1.2 Probleme

Bisher auf dem Markt verfügbare, konventionelle CBT-Systeme weisen verschiedene konzeptuelle Schwächen auf und leiden bei vielen Medizindozenten unter Akzeptanzproblemen. Unter konventionellen CBT-Systemen werden im Folgenden Lehr-/Lernsysteme verstanden, welche auf stand-alone Computern ohne Netzanschluß lauffähig sind und auf Datenträgern, meist über medizinische Fachverlage, verbreitet werden.
Folgende Probleme, die Einsatz, Entwicklung und Verbreitung von Lehr-/Lernsystemen beeinträchtigen, lassen sich feststellen:

Problem 1: Konventionelle CBT-Systeme sind plattformabhängig. Studenten oder Universitäten benötigen deshalb die „richtige“ Hard- und Softwareplattform, um mit einem Lehr-/Lernsystem arbeiten zu können. Die Portierung von Systemen (beispielsweise von Macintosh auf PC oder umgekehrt) ist für die Entwickler mit zusätzlichem Aufwand und Kosten verbunden und wird deshalb häufig nicht durchgeführt.

Problem 2: Konventionelle CBT-Systeme müssen installiert werden. Die Installation erfordert Zeit und auf der lokalen Festplatte des Rechners wird Speicherplatz benötigt. Weiterhin können Installationen, insbesondere bei Windows-PC’s, zu Instabilitäten führen. Viele Anwender scheuen deshalb davor zurück, lediglich zu Testzwecken ein CBT-System zu installieren.

Problem 3: Das Update von konventionellen CBT-Systemen ist aufwendig, weil an alle Nutzer Updates verschickt und von diesen auf ihren Rechnern aufgespielt werden müssen.

Problem 4: Beim Einsatz konventioneller CBT-Systeme im Selbststudium bleiben die Studierenden auf sich alleine gestellt, wenn sie Fragen mit dem im System enthaltenen Wissen nicht selbst lösen können.

Problem 5: Konventionelle CBT-Systeme sind in der Regel nicht adaptiv. Adaptivität, d.h. die Anpassung des Lehr-/Lernsystems an den Benutzer, ist praktisch nur bei Intelligenten Tutoriellen Systemen (siehe Kapitel 2.8) zu finden. Deren Erstellung ist sehr aufwendig.

Problem 6: Beim Entwurf existierender CBT-Systeme wurde der Wiederverwendbarkeit von Lehr-/Lerninhalten (Domänenwissen, multimedialen Lehr-/Lernmedien usw.) und von Lehr-/Lernsystemfunktionalität in vielen Fällen zu wenig Beachtung geschenkt. Häufig sind Domänendaten und Domänenwissen nicht strikt von der Lehr-/Lernsystemfunktionalität getrennt. Eine Wiederverwendung von Lehr-/Lerninhalten und von Programmcode ist dadurch nur mit hohem Aufwand möglich.

Problem 7: Die Erstellung von Lehr-/Lernsystemen ist sehr zeit- und kostenintensiv und erfordert neben medizinischem Fachwissen auch Informatikkenntnisse. Medizindozenten haben meist weder die erforderliche Zeit noch die notwendigen Informatikkenntnisse, um ohne leistungsfähige Autorentools qualitativ hochwertige, multimediale CBT-Systeme erstellen zu können.

Problem 8: Viele Dozenten besitzen eine kritische Haltung gegenüber CBT-Systemen, insbesondere gegenüber solchen, die nicht unter ihrer Mitwirkung entwickelt wurden.

1.3 Zielsetzung und Fragestellung

Um die oben genannten Probleme zu lösen und dadurch den Einsatz und die Entwicklung von CBT-Systemen zu fördern, ist es erforderlich, Systeme mit grundlegend neuer Architektur zu entwickeln.

Für die Arbeit ergeben sich folgende Ziele und Fragestellungen:

Ziel 1: Erstellung eines innovativen Konzeptes für die Realisierung von in der medizinischen Aus- und Weiterbildung eingesetzten Lehr-/Lernsystemen.

Folgende Fragen sollen in Zusammenhang mit diesem Ziel beantwortet werden:

Frage 1.1: Welche konzeptuellen Merkmale muß ein Lehr-/Lernsystem aufweisen, um die in Kapitel 1.2 beschriebenen Probleme konventioneller CBT-Systeme zu lösen?
Frage 1.2: Wie kann eine häufigere Nutzung von Lehr-/Lernsystemen durch die Studierenden erreicht werden?
Frage 1.3: Wie kann die Akzeptanz von Lehr-/Lernsystemen bei den Medizindozenten verbessert werden?
Frage 1.4: Wie können Dozenten dazu gebracht werden, verstärkt in ihren Lehrveranstaltungen CBT-Systeme einzusetzen und ihren Studenten die Arbeit mit CBT-Systemen zu empfehlen?
Frage 1.5: Inwieweit kann das Konzept fachgebietsunabhängig gehalten werden?

Ziel 2: Exemplarische Umsetzung der entwickelten Konzepte in Form eines Prototypen für die Pädiatrie und die Infektiologie.

1.4 Gliederung der Arbeit

In Kapitel 2 werden alle für die Arbeit relevanten Grundlagen kurz dargestellt. Kapitel 3 beschreibt ausführlich das erarbeitete Lösungskonzept für die in Kapitel 1 beschriebenen Probleme. Kapitel 4 stellt kurz die Realisierung der Konzepte im Rahmen des CAMPUS-Projektes (Computerunterstützte Aus- und Weiterbildung in der Medizin durch plattformunabhängige Software) vor. Zusammenfassung und Diskussion (Kapitel 5) sowie eine Literaturliste (Kapitel 6) schließen die Arbeit ab.

2 Grundlagen

2.1 Begriffsdefinitionen

Computer-based Training (CBT): Computer-based Training ist eine Ausbildungsform, bei der Computer nicht als Lerngegenstand sondern als Lehrmittel eingesetzt werden.

CBT-System: Ein Anwendungssoftwareprodukt und evtl. benötigte Zusatzhardware, das auf einem Computer zur Aus- und Weiterbildung eingesetzt werden kann.

Konventionelles CBT-System: CBT-System, welches auf einem stand-alone Computer ohne Netzanschluß lauffähig ist und auf Datenträgern meist über herkömmliche Vertriebswege (häufig medizinische Fachverlage) verbreitet wird.

WBT-System: CBT-System, welches auf dem Internet bzw. World Wide Web (WWW) und den dort verwendeten Standards basiert.

Web-based Training (WBT): Computer-based Training unter Verwendung von WBT-Systemen.

Lehr-/Lernsystemfunktionalität: Unter Lehr-/Lernsystemfunktionalität werden alle Komponenten verstanden, die für Lehr-/Lernsysteme charakteristisch sind und ein System für die Ausbildung geeignet machen. Hierunter fallen u.a. Lehrkomponenten (implementierte Lehrstrategien), Präsentationskomponenten und Navigationsfunktionen (bei Browsingsystemen).

Lehr-/Lernsystemclient: Teil eines Lehr-/Lernsystems, welcher bei Lehr-/Lernsystemen mit Client/Server-Architektur auf dem *Client*-Computer angesiedelt ist und mit dem Lehr-/Lernsystemserver über Rechnernetz kommuniziert.

Lehr-/Lernsystemserver: Teil eines Lehr-/Lernsystems, welcher bei Lehr-/Lernsystemen mit Client/Server-Architektur auf dem *Server*-Computer angesiedelt ist und mit dem Lehr-/Lernsystemclient über Rechnernetz kommuniziert.

Basistechnik: Technik oder Programmiersprache zur Erstellung von WBT-Systemen.

Lehr-/Lernsystemshell: CBT-System-Hülle, in der lediglich die Lehr-/Lernsystemfunktionalität vorhanden ist und bei der durch Hinzufügen von Domänendaten und Domänenwissen ein CBT-System entsteht. Das Hinzufügen von Domänendaten und Domänenwissen wird dabei durch ein einfach zu bedienendes Werkzeug unterstützt.

2.2 Medizinische Ausbildung

Das medizinische Wissen ist heute so umfangreich, daß selbst bei einer starken Spezialisierung auf ein bestimmtes Gebiet die Menge des verfügbaren Wissens kaum noch zu überblicken ist [WEED 89]. Außerdem ändert sich dieses Wissen relativ schnell, so daß es während der Dauer eines Medizinstudiums praktisch unmöglich ist, den Studenten eine hinreichende Menge medizinischen Wissens zu vermitteln [ARBEITSKREIS MEDIZINERAUSBILDUNG 95 S. 73]. Genau dies wird aber immer noch bei den meisten Medizinstudiengängen in Deutschland versucht. Sie beruhen auf dem auf Vollständigkeit ausgelegten Gegenstandskatalog. Im Gegensatz zu anderen Fachgebieten besteht das Ausbildungsziel in der Medizin noch immer in der umfassenden Vermittlung von Wissen. Von Jurastudenten wird während ihres Studiums nicht verlangt, Gesetzestexte auswendig zu lernen. Vielmehr müssen diese lernen, Gesetzestexte schnell zu verstehen, richtig anzuwenden und dabei Hilfsmittel wie Kommentare richtig einzusetzen.
Es ist unbestritten, daß die Medizinausbildung dringend reformbedürftig ist. Denn alle bisherigen Reformbemühungen haben nicht den gewünschten Erfolg gebracht. Vielfach wird von einer Ausbildungsmisere bzw. Krise der ärztlichen Ausbildung gesprochen [EITEL 93b, AAMC 84]. Es existieren deshalb eine ganze Reihe verschiedener Reformvorschläge [ARBEITSKREIS MEDIZINERAUSBILDUNG 95, WISSENSCHAFTSRAT 92, BUNDESMINISTERIUM FÜR GESUNDHEIT (Hrsg.) 93, MFT-PRÄSIDIALKOMMISSION 96] bei denen sich in vielen Punkten ein Konsens über die erforderlichen Maßnahmen herauszubilden schien [EITEL 93a].
1989 veröffentlichte der sogenannte „Murrhardter Kreis“ (ein von der Robert-Bosch-Stiftung geförderter Arbeitskreis Medizinerausbildung) eine Studie über das *Arztbild der Zukunft*. Diese Studie bildete fortan die Diskussionsgrundlage für die Reformdiskussionen in Deutschland. In überarbeiteter Form wurde die Studie 1995 erneut vorgelegt. Die Kritikpunkte am Medizinstudiums werden dort wie folgt beschrieben [ARBEITSKREIS MEDIZINERAUSBILDUNG 95]:

- Der Schwerpunkt im Medizinstudium liegt bei der Vermittlung von Faktenwissen. Verständnis und die Fähigkeit zum Problemlösen werden weniger intensiv vermittelt.
- Bei den bundeseinheitlich durchgeführten schriftlichen Prüfungen werden zu viele Detailkenntnisse verlangt, die in der beruflichen Praxis häufig keine große Bedeutung haben und schnell wieder vergessen werden.
- Medizinstudenten werden nicht darauf vorbereitet, sich selbständig weiterzubilden, was aufgrund „...der sich schnell ändernden wissenschaftlich-technologischen und gesellschaftlichen Rahmenbedingungen ärztlicher Berufsausbildung“ [ARBEITSKREIS MEDIZINERAUSBILDUNG 95 S. 273] erforderlich wäre.
- Der Umgang mit Patienten und deren Problemen wird nicht hinreichend geübt. Die Praxis kommt in der Medizinausbildung zu kurz.
- Die Nutzung der Informationsverarbeitung wird nicht ausreichend vermittelt. Ebensowenig wie kommunikative Fähigkeiten ausreichend geschult werden.
- Naturwissenschaftliche Aspekte besitzen im Medizinstudium eine Priorität gegenüber psycho-sozialen, ethischen und emotionalen Aspekten. „Im Studium werden Mediziner, nicht Ärzte ausgebildet“ [ARBEITSKREIS MEDIZINERAUSBILDUNG 95 S. 273].
- Die im Studium vermittelte klinische Ausbildung ist für eine Tätigkeit im allgemeinärztlichen Bereich nicht ausreichend.
- Die Lehrinhalte der verschiedenen Fächer und Ausbildungsstufen sind nur schlecht koordiniert.

- Forschung und Krankenversorgung besitzen bei den meisten Hochschullehrern eine Priorität gegenüber der Lehre, was häufig zu einer unbefriedigenden didaktischen Qualität von Lehrveranstaltungen führt.
- Angebote für die Schwerpunktbildung auf naturwissenschaftliche, biomedizinische oder psycho-soziale Aspekte werden von den Studierenden nicht genügend akzeptiert bzw. von den Hochschulen gar nicht angeboten.

Aus den aufgezählten Kritikpunkten ergeben sich die Anforderungen an ein Reformcurriculum:

- Studenten sollten besser für primärarztliche Tätigkeit ausgebildet werden.
- Selbständiges, problemorientiertes und aktives Lernen sollte gefördert werden.
- Die starre Trennung zwischen klinischer und vorklinischer Ausbildung sollte aufgehoben werden.
- Kontakt mit Patienten sollte möglichst früh ermöglicht werden.
- Psycho-soziale und ethische Aspekte sollten im Studium stärker berücksichtigt werden.
- Die Studierenden sollten mehr Wahlfreiheit bei der Studiengestaltung haben.
- In den Prüfungen sollte mehr Praxisbezug herrschen und der Anteil an MC-Prüfungen sollte verringert werden.

2.2.1 BMG-Entwurf

Der Entwurf des Bundesgesundheitsministeriums zur Reform des Medizinstudiums entspricht sicherlich nicht in allen Punkten den oben genannten Anforderungen an ein Reformcurriculum [GÖBEL, REMSTEDT (Hrsg.) 95] und bleibt auch hinter den Empfehlungen des Wissenschaftsrates [WISSENSCHAFTSRAT 92] zurück. Allerdings hatte dieser Entwurf von allen Entwürfen die besten Chancen, in die Praxis umgesetzt zu werden. Der Entwurf im Einzelnen:

- Sechsjähriges Studium, gegliedert in drei Teile (5+5 Semester, PJ), ergänzt durch 1,5-jährige AiP-Zeit.
- Verzahnung von Vorklinik und Klinik.
- Zusammenfassung der Fächer zu Stoffgebieten.
- Studieninhalte sollen so weit als möglich fächerübergreifend und studienbezogen vermittelt werden.
- Verbesserungen in der praktischen Ausbildung z.B. durch Seminare, Unterricht am Krankenbett und gegenstandsbezogene Studiengruppen im ersten Studienabschnitt. Klinische Blockpraktika im 2. Studienabschnitt.
- Problemorientiertes Lernen [ALBANESE, MITCHELL 93; HASMAN 89] als neue Unterrichtsform neben Unterricht am Krankenbett.
- Staatsprüfung in drei Teilen. Erster Teil der Prüfung nach 5 Semestern (Verschmelzung von Physikum und 1. Staatsexamen). Zweiter Teil der Prüfung nimmt an Umfang zu. Dabei werden die mündlich-praktischen Prüfungen stärker gewichtet. Der dritte Teil findet unverändert als mündliche Kollegialprüfung in vier Fächern statt.
- Eine Experimentierklausel soll die Einrichtung von Modellstudiengängen zulassen.
- Reduzierung der Zulassungszahlen um 20%.

Mittlerweile konnte auch dieser Entwurf nicht die erforderliche Zustimmung in den zuständigen Gremien finden. Die 8. Novelle der Ärztlichen Approbationsordnung ist deshalb

nun auf unbestimmte Zeit verschoben. Jedoch wird wahrscheinlich eine Experimentierklausel eingeführt, durch die Modellstudiengänge wie der geplante Reformstudiengang in Berlin ermöglicht werden.

2.2.2 Berliner Reformstudiengang

In Berlin wurde nach einem Studentenstreik im WS 1988/89 eine Arbeitsgruppe *Reformstudiengang Medizin* eingerichtet. Aufgabe dieser Arbeitsgruppe war es, einen Reformstudiengang inhaltlich zu entwerfen und organisatorisch vorzubereiten. An der Gestaltung wirkten neben Hochschullehrern aus den Grundlagenfächern Vertreter der klinischen Fachgebiete ebenso mit wie studentische Vertreter [Haller, Burger et al. 95]. Vorbild bei der Konzipierung des Studiengangs war die McMaster-Universität in Kanada. Das Ergebnis der Arbeitsgruppe wurde 1993 präsentiert. Der Reformstudiengang sollte ursprünglich dann 1995 seinen Betrieb aufnehmen. Noch immer sind allerdings nicht alle rechtlichen (Änderung der Approbationsordnung) und finanziellen Voraussetzungen für einen Start gegeben. Der Berliner Reformstudiengang stimmt mit den Empfehlungen des Wissenschaftsrates und des Murrhardter Kreises überein und geht damit über den vom Bundesgesundheitsministerium vorgelegten Vorschlag hinaus.
An der Verteilung der Wochenstunden (siehe Abbildung 1) können die typischen Charakteristika eines Reformcurriculums erkannt werden.

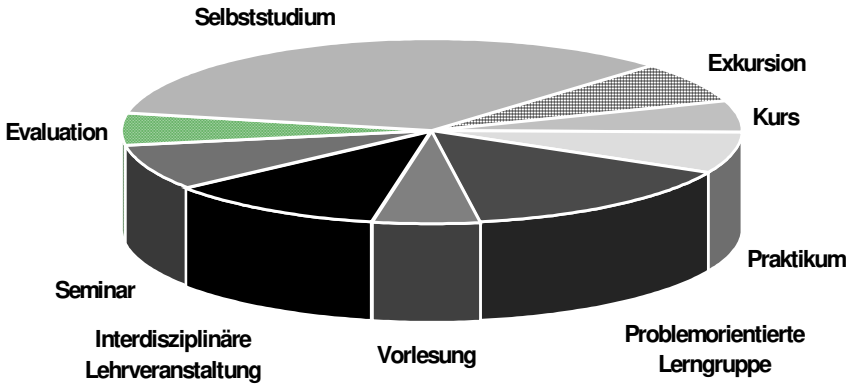

Abbildung 1: Stundenverteilung beim Berliner Reformstudiengang

2.2.3 Studienreform und CBT

Im Gegensatz zu den Lehr-/Lernformen bleiben die Lehr-/Lerninhalte in den Reformcurricula weitgehend unverändert. Bedingt durch Fortschritte in der Medizinischen Forschung sind diese sowieso einem ständigen Wandel unterzogen.
Computer spielen als Lehrmittel in Reformstudiengängen eine wichtigere Rolle als in „herkömmlichen" Studiengängen. Denn die Studenten sollen hier verstärkt durch problemorientiertes Lernen [ALBANESE, MITCHELL 93; HASMAN 89] und einen höheren Anteil des Selbststudiums zu selbständigem Handeln und zur eigenverantwortlichen Weiterbildung befähigt werden. Dabei können Lehr-/Lernsysteme, die in Mediotheken oder auf Servern im Internet bereitgestellt werden, einen wichtigen Beitrag leisten [WISSENSCHAFTSRAT 92, WISSENSCHAFTSRAT 98]. Auch die während der gesamten Berufstätigkeit von Ärzten erforderliche Weiterbildung kann durch den Einsatz von CBT-Systemen erleichtert und unterstützt werden. In den USA können Ärzte der (staatlich vorgeschriebenen) Weiterbildung durch Bearbeitung von Lehr-/Lernsystemen nachkommen. Für jedes vom *Accreditation Council for Continuing Medical Education* zugelassene System können eine bestimmte Anzahl an Weiterbildungspunkten erworben werden (z.B. [UPMC]). Hierzu muß nach der Arbeit mit einem System ein Fragebogen ausgefüllt und eingesandt werden. Falls ein ausreichender Prozentsatz der Fragen korrekt beantwortet wurde, werden die Weiterbildungspunkte gutgeschrieben. Für Deutschland gibt es zur Zeit noch keine staatlich vorgeschriebene Weiterbildungspflicht, jedoch sind auch hier die Ärzte zur ständigen Weiterbildung gezwungen, um ihren Patienten eine dem aktuellen wissenschaftlichen Stand entsprechende Diagnostik und Therapie zukommen lassen zu können.

2.3 Computer-based Training

2.3.1 Interaktionsformen bei Lehr-/Lernsystemen

Bei CBT-Systemen im medizinischen Bereich kann man fünf unterschiedliche Interaktionsformen unterscheiden (siehe Abbildung 2) [HAAG, MAYLEIN et al. 98]. Ein reales Lehr-/Lernsystem kann gleichzeitig auch mehrere der aufgeführten Interaktionsformen aufweisen. Beispielsweise wenn in einem Lehr-/Lernsystem neben einer Fallsimulation (Interaktionsform Simulation, s.u.) auch ein elektronisches Lexikon (Interaktionsform Browsing, s.u.) vorhanden ist, in dem die Anwender Wissen nachschlagen können, das sie für eine erfolgreiche Fallbearbeitung benötigen. Alle fünf Interaktionsformen lassen sich prinzipiell mit den in Kapitel 2.6.2 angesprochenen *Basistechniken* für die Entwicklung von Internet-Applikationen implementieren.

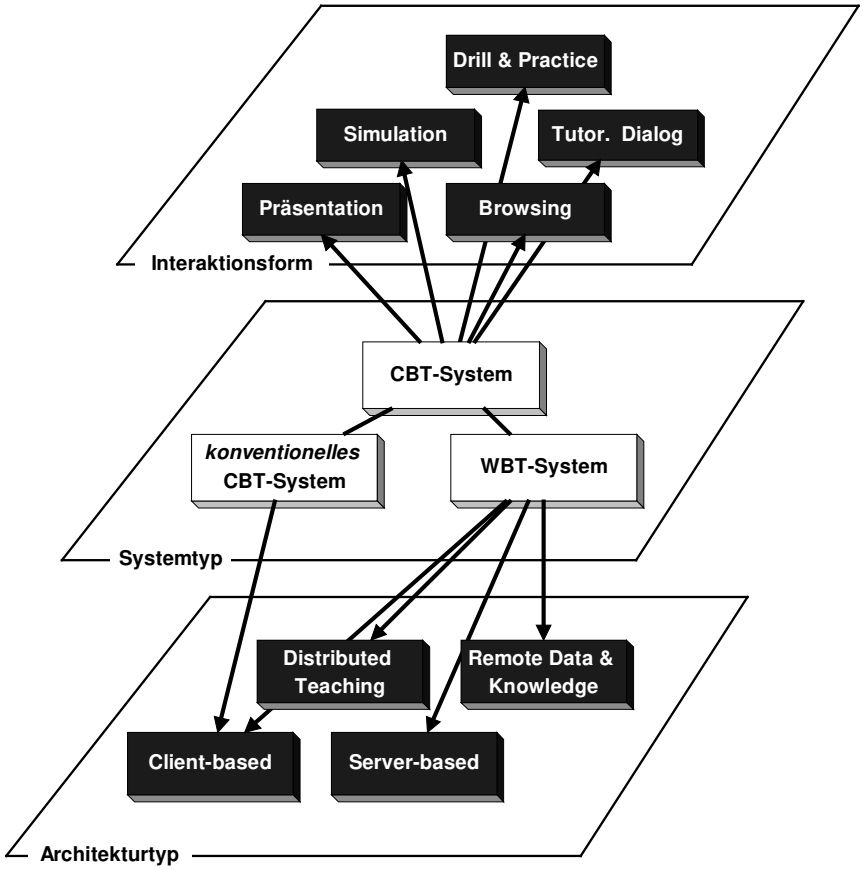

Abbildung 2: Interaktionsformen und Architekturtypen bei CBT-Systemen

2.3.1.1 Interaktionsform *Präsentation*

Bei der Interaktionsform *Präsentation* werden Sachverhalte in linearer Form vom System präsentiert. Es handelt sich dabei um die elektronische Form eines Diavortrags oder einer Tonbildschau. Die Eingriffsmöglichkeiten des Nutzers beschränken sich darauf, in der Präsentation vorwärts bzw. rückwärts zu gehen und bei automatischer Präsentation diese anzuhalten.

2.3.1.2 Interaktionsform *Browsing*

Beim *Browsing* stehen den Anwendern die Lehr-/Lernsysteminhalte in Form eines Hypertextes zur Verfügung, in dem sie frei navigieren können. Die Entscheidung darüber, welche Inhalte überhaupt und in welcher Reihenfolge angeschaut werden, bleibt dem Anwender selbst überlassen. In der Regel ist ein Inhaltsverzeichnis und/oder eine Stichwortliste vorhanden, über die gezielt zu bestimmten Inhalten gesprungen werden kann. Die Inhalte sind durch Links untereinander verknüpft, so daß ein Netzwerk von Informationselementen entsteht.
Elektronische Lehrbücher [KNAUP 94] sind ein typisches Beispiel für diese Interaktionsform.

2.3.1.3 Interaktionsform *Drill & Practice*

Die Interaktionsform *Drill & Practice* dient dazu, um mit anderen Lehrmedien (z.B. Büchern) gelerntes Faktenwissen zu vertiefen. Dies geschieht, indem man vom Lehr-/Lernsystem gestellte Fragen beantwortet oder Übungen bearbeitet. Es erfolgt eine Rückmeldung darüber, ob die Frage korrekt beantwortet wurde oder nicht bzw. ob die Übung korrekt durchgeführt wurde oder nicht. Das System versucht bei dieser Interaktionsform jedoch nicht festzustellen, warum eine Frage falsch beantwortet wurde. *Drill & Practice*-Systeme lassen sich deshalb mit vergleichsweise geringem Aufwand implementieren.

2.3.1.4 Interaktionsform *Tutorieller Dialog*

Im Gegensatz zu den Interaktionsformen *Präsentation* und *Browsing*, bei denen das Lehr-/Lernsystem eine passive Rolle einnimmt, übernimmt das CBT-System bei der Interaktionsform *Tutorieller Dialog* eine aktive Rolle. In der Regel werden nach der Präsentation einzelner Lehr-/Lerneinheiten Wissenskontrollfragen gestellt, mit deren Hilfe das CBT-System festzustellen versucht, ob und wie weit die vorangegangene Lehr-/Lerneinheit verstanden wurde. Werden Fragen falsch beantwortet, dann gibt das System gezielte Hilfestellungen und präsentiert weitere Informationen und weiteres Wissens um das festgestellte Defizit zu beseitigen. Entscheidend für die Kategorisierung als *Tutorieller Dialog* ist, daß im Wesentlichen prozedurales Wissen (Regeln und deren Anwendung) gelehrt wird und nicht Faktenwissen [BAUMGARTNER, PAYR 94]. Der didaktische Anspruch von *Tutoriellen Dialogen* und damit auch der Entwicklungsaufwand ist viel höher als bei *Drill & Practice*, wo das System lediglich feststellen muß, ob eine Frage korrekt beantwortet wurde oder nicht.
Eine Sonderform des Tutoriellen Dialogs stellt der *Intelligente Tutorielle Dialog* dar. Hierbei werden Methoden der künstlichen Intelligenz [WENGER 87] eingesetzt um den Dialog zwischen CBT-System und Nutzer so zu steuern, daß der Lernfortschritt in idealer Weise gefördert wird. Dies geschieht, indem kontinuierlich Lernverhalten, Vorkenntnisse, Vorlieben des Nutzers analysiert werden und auf Basis dieser Informationen der weitere Programmablauf gesteuert wird (z.B. [FONTAINE, LE BEUX et al. 94]). Beim *Intelligenten Tutoriellen Dialog* versucht sich das Lehrsystem also an den Nutzer zu adaptieren.

2.3.1.5 Interaktionsform *Simulation*

Bei der Interaktionsform *Simulation* verhält sich das Lehr-/Lernsystem wiederum passiv. Es präsentiert dem Nutzer das möglichst realitätsgetreue Modell eines Wirklichkeitsausschnittes und zeigt die Reaktionen des Modells auf Aktionen der Nutzer. Das System gibt jedoch keine direkten Rückmeldungen auf die Nutzeraktionen. Unterscheiden kann man zwischen Grundlagensimulationen (z.B. [HIRSCH, BRAUN et al. 93]), und Fallsimulationen (z.B. [PUPPE, REINHARDT 95]).

2.3.2 Entwicklung von CBT-Systemen

Für die Entwicklung konventioneller CBT-Systeme mit den Interaktionsformen Präsentation (siehe Kapitel 2.3.1.1) und Browsing (siehe Kapitel 2.3.1.2) stehen eine größere Zahl verschiedener kommerziell verfügbarer Autorensysteme zur Verfügung. Man kann sie in drei unterschiedliche Kategorien einteilen [HAAG 95]:

Skript- oder kartenbasierte Systeme:
Bei skript- oder kartenbasierten Systemen erstellt der Autor sogenannte *Skripten*, die bei Eintritt von bestimmten Ereignissen (z.B. Anwahl eines Buttons mit der Maus, Eingabe von Text usw.) abgearbeitet werden. Die Skriptsprachen sind meist sehr leistungsfähig und unterstützen die Erstellung von Hypertexten. Unter Windows auf dem PC ist Toolbook sehr weit verbreitet und wird häufig eingesetzt. Auf dem Apple Macintosh werden HyperCard [APPLE 93] und SuperCard sehr häufig für die Erstellung von CBT-Systemen verwendet. Skript- oder kartenbasierte Systeme erlauben in der Regel auch die Erstellung komplexerer Lehr-/Lernsysteme. Allerdings muß bei Verwendung dieser Art von Autorensystemen eine Skriptsprache erlernt werden.

Iconbasierte Systeme:
Iconbasierte Systemen ermöglichen die Erstellung von Lehr-/Lernsystemen, ohne daß im traditionellen Sinne programmiert werden muß. Autoren können den Programmablauf mittels graphische Programmierung erstellen, indem sie Icons positionieren und miteinander verknüpfen. Die verschiedenen Icons repräsentieren dabei jeweils Grundkonstrukte (Schleife, If-Anweisung usw.), wie man sie von Programmiersprachen wie Pascal kennt. Mit dieser Art von Systemen können Fachgebietsexperten ohne große Informatikkenntnisse schnell Präsentationen und einfache Lehr-/Lernsysteme erstellen. Allerdings ist die Flexibilität bei der Programmierung eingeschränkt. Das wohl bekannteste Beispiel für iconbasierte Autorensysteme ist Authorware Professional [MACROMEDIA AUTHORWARE].

Zeitbasierte Systeme:
Zeitbasierte Systeme sind sehr gut für die Erstellung von Präsentationen geeignet. Entlang eines Zeitstrahls können verschiedene „Aktionen“ aufgereiht werden, wie zum Beispiel das Einblenden eines Bildes, eines Eingabefeldes oder das Abspielen eines Soundfiles. Animationen lassen sich sehr leicht erstellen, indem Einzelbilder entlang des Zeitstrahls aufgereiht werden. Eine Steuerung durch Eingaben der Anwender ist möglich. Das wahrscheinlich am häufigsten verwendete zeitbasierte System ist Macromedia Director [MACROMEDIA 94]. Es ist für verschiedene Plattformen erhältlich.

Die angesprochenen kommerziell erhältlichen Autorensysteme besitzen einige typische Schwächen. Autoren können Domäneninhalte und Lehr-/Lernsystemfunktionalität nicht strikt voneinander trennen. Dadurch ist eine Wiederverwendbarkeit der Inhalte kaum oder nur mit vergleichsweise hohem Aufwand möglich. Bei den skript- und kartenbasierten Autorensystemen beispielsweise werden auf einer Karte sowohl die Domäneninhalte eingetragen, als auch Lehr-/Lernsystemfunktionalität in Form von Skripten an den auf der Karte angesiedelten Objekten (Buttons, Textfelder ...) verankert. Desweiteren müssen die erstellten Systeme auf dem Zielrechner meistens installiert werden und ein Update ist nur dadurch möglich, daß das überarbeitete Lehr-/Lernsystem über die alte Version installiert

wird. In vielen Fällen sind die erstellten CBT-Systeme entweder nur auf einer oder lediglich auf wenigen Plattformen lauffähig.

2.3.3 Problemorientiertes Lernen mit CASUS

Eine der derzeit aktivsten Arbeitsgruppen im medizinischen CBT-Bereich in Deutschland arbeitet an der Ludwig-Maximilians-Universität München unter Leitung von Martin Fischer. Mit dem in dieser Arbeitsgruppe entwickelten Lehr-/Lernsystem *CASUS* [FISCHER, GRÄSEL et al. 95] können Medizinstudenten anhand von Lehr-/Lernfällen lernen. Das System besteht aus zwei Komponenten: Einem Autorensystem sowie einem Abspielmodul bzw. Player. Das leicht zu bedienende Autorensystem erlaubt es Autoren mit vergleichsweise geringem Aufwand, Lernfälle zu erstellen. Diese Lehr-/Lernfälle werden wahlweise lokal auf dem Autorenrechner oder zentral in einer relationalen Datenbank gespeichert. Durch Verwendung dieser zentralen Datenbank konnte in einem weiteren Projekt namens *ProMediWeb* [WEICHELT, SCHMIDT 98] die Entwicklung eines Abspielprogramms für das Internet begonnen werden (Architekturtyp *Remote Data & Knowledge*, siehe Kapitel 2.5.1.2). Aufgabe des Players ist neben dem Anzeigen der multimedialen Lehr-/Lernfälle (siehe Abbildung 3 [SCRIBA, MANDL et al. (Hrsg.) 97 S. 18]) das Mitprotokollieren aller Nutzereingaben für spätere Evaluationen.
Der genaue Ablauf und Aufbau eines medizinischen Lehr-/Lernfalles wird vom Autor bestimmt. Das Grundschema sieht wie folgt aus:

- Erster Eindruck vom Patienten
- Anamnese
- Körperliche Untersuchung
- Laboruntersuchung
- Technische Untersuchung
- Diagnose
- Therapie und Verlauf

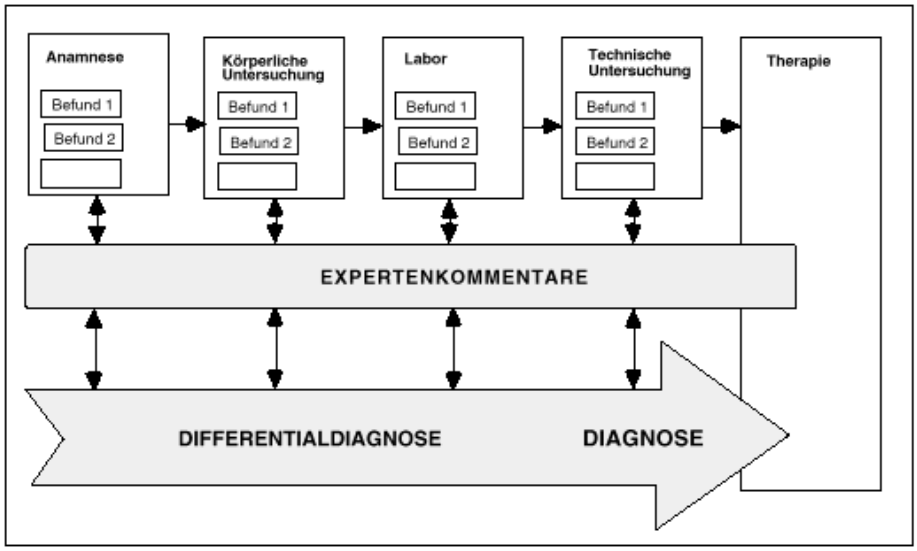

Abbildung 3: Schematische Programmstruktur eines CASUS-Lehr-/Lernfalls

Es kann durch Hinzufügen weiterer Schritte, Schleifen und Sprünge variiert werden. Für die Fallpräsentation stehen den Autoren verschiedene Kartenlayouts (insgesamt 6) zur Verfügung, mit Hilfe derer sie einen Lehr-/Lernfall präsentieren können. Zwischen den einzelnen Präsentationskarten können Karten in den Fallablauf integriert werden, auf welchen die Lernenden Fragen zum Lehr-/Lernfall beantworten müssen. Dadurch, daß die Autoren die (multimedialen) Inhalte der Karten selbst per Freitext eingeben, ist es kaum möglich, einzelne Fallbestandteile ohne großen Aufwand in anderen Lehr-/Lernfällen wiederzuverwenden. Eine solche Wiederverwendung wird vom CASUS-System auch nicht unterstützt. Lehr-/Lernfälle werden also in wenig strukturierter Form in der Datenbank abgespeichert. Dadurch unterscheidet es sich u.a. von dem in dieser Arbeit vorgestellten CAMPUS-System, das auf einem detaillierten Fallmodell (siehe Kapitel 3.5.2) basiert.
Einen zentraler Bestandteil des Autorensystems bildet das Netzwerktool, mit dem Autoren die differentialdiagnostische Vorgehensweise eines Experten abbilden. In einem differentialdiagnostischen Netzwerk werden die Befundergebnisse eines Patienten mit differentialdiagnostischen Hypothesen und den jeweiligen Therapien in Verbindung gebracht. Lernende nutzen die von den Autoren erstellten differentialdiagnostischen Netzwerke dazu, um eigene Überlegungen mit denen des Autors zu vergleichen und Fehler bei der Fallbearbeitung bemerken und korrigieren zu können. Während der Fallbearbeitung können Expertenkommentare abgerufen werden. Diese vom Autor erstellten Kommentare sollen die Lernenden beim Problemlösen lenken, wenn sie nicht ohne Hilfe weiterkommen. Die Kommentare sind optional abrufbar.
Lehr-/Lernfälle können zur Zeit in zwei Datenbanken abgelegt werden: In einer objektorientierten NeoAccess-Datenbank und in einer relationalen Oracle-Datenbank. Der Zugriff auf die NeoAccess-Datenbank erfolgt über einen eigens entwickelten *CASUS-Dataserver*. Auf ORACLE wird mittels des SQL (Structured Query Language) Net Protokolls zugegriffen. Die notwendige Datenbankzugriffsfunktionalität ist im Autorensystem implementiert.

2.4 WWW

Bis Anfang der Neunziger Jahre wurde das Internet auf der ganzen Welt fast ausschließlich von Wissenschaftlern und Technikbegeisterten genutzt. Mit der Verfügbarkeit des World Wide Web [BERNES-LEE, CAILLEAU et al. 92], dessen Entwicklung 1989 am CERN (Europäisches Labor für Teilchenphysik) begonnen wurde, hat sich diese Situation grundlegend geändert. Das World Wide Web macht weltweit auf Internet-Hosts verteilte Dokumente verfügbar und integriert auch die bereits vorher existierenden Internet-Dienste unter einer leicht zu bedienenden Benutzeroberfläche. Es basiert auf dem Hypertext-Prinzip (s.u.) und ermöglicht die Erstellung von Hypertext-Dokumenten, deren einzelne Medienbestandteile (Bilder, Videos usw.) sich auf verschiedenen Rechnern befinden dürfen. 1992 wurde die erste WWW-Software (WWW-Server und WWW-Browser) vorgestellt und im Internet kostenlos zur Verfügung gestellt. Die WWW-Clients waren rein textorientiert und unterstützten keine multimedialen Komponenten. Von 50 WWW-Servern weltweit im Jahr 1993 hatte sich deren Zahl bis 1995 auf mehrere Tausend vergrößert [MAIER, WILDBERGER 95] und seither beobachten wir weiterhin ein starkes Wachstum. Auch die Anzahl der Nutzer im Internet ist in den letzten Jahren stark gestiegen. Immer mehr Menschen besitzen einen Zugang zum Internet und das World Wide Web hat sich geradezu zu einem Massenmedium entwickelt. So gibt es kaum noch Firmen, Institutionen, Fernsehsender usw., welche nicht im WWW präsent sind.
Durch die einfache und intuitive Bedienung der WWW-Browser und die Möglichkeit, multimediale Dokumente anzeigen zu können sowie die Tatsache, daß viele Leute mitt-

lerweile schon einmal einen WWW-Browser bedient haben, gibt es kaum Berührungsängste. Nach eigenen Beobachtungen sind auch computerunerfahrene Nutzer sehr schnell in der Lage, WWW-Browser anzuwenden.
Das World Wide Web bietet eine Reihe von Vorteilen für die Erstellung von Anwendungen im medizinischen Bereich [CIMINO, SOCRATOUS et al. 95 S. 282]. So können bereits existierende Anwendungen in relativ kurzer Zeit unter Verwendung der CGI-Schnittstelle (Common Gateway Interface) im Internet verfügbar gemacht werden. HTML-Dokumente (Hypertext Markup Language) und CGI-Skripten können leicht weitergegeben und auch in anderen Institutionen verwendet werden. Client-Software ist für nahezu alle Plattformen, für Universitäten bzw. Universitätsangehörige meist kostenlos, erhältlich. Bilder (z.B. CT- oder MR-Bilder) können einfach mit HTML in Dokumente eingebunden werden. Der Zugriff auf Informationen und Anwendungen ist von allen Rechnern im Internet weltweit und auch von zu Hause über Modem möglich. Weltweit vorhandene Wissensressourcen im Internet können sehr leicht zugegriffen und in eigene Anwendungen integriert werden.

Den unbestreitbaren Stärken des World Wide Web wie einfache Bedienbarkeit und breite Verfügbarkeit von Software stehen auch einige Schwächen gegenüber:

- Das Hypertext Transmission Protocol (HTTP) ist ein zustandsloses Protokoll. Es wird also keine kontinuierliche Verbindung zwischen Browser und Server aufgebaut, sondern für jede Nutzeraktion eine neue [IBRAHIM, FRANKLIN 95]. Dadurch ergeben sich bei mehrstufigen Mensch-Maschine-Dialogen Probleme, weil der Server keine Informationen über die vorhergehenden Interaktionsschritte besitzt. Als Lösung dieses Problems können versteckte Eingabefelder verwendet werden. Hier legt der Server Informationen in HTML-Dokumenten ab. Allerdings wird durch diese Lösung unnötiger Netzverkehr erzeugt. Ein anderer Lösungsansatz sind *Cookies*. Der WWW-Browser legt auf der Festplatte des Nutzers Informationen ab, die bei Bedarf an den WWW-Server gesendet werden.
- Haben Nutzer in Eingabefelder Daten eingetragen und betätigen zu einem späteren Zeitpunkt den *Back-Button* bzw. gehen über die *History-List* auf die Eingabeseite zurück, so stellen sie fest, daß die eingegebenen Daten immer noch in den Eingabefeldern stehen und können dann evtl. dazu verleitet werden, die Daten erneut abzuschicken. Beim Schutz von WWW-Seiten oder Unterverzeichnissen durch Paßwörter kann es leicht vorkommen, daß bei mehreren Personen, die sich einen Rechner teilen, eine WWW-Seite für eine Person zugänglich ist, weil der vorhergehende Nutzer das erforderliche Paßwort bereits eingegeben hat [CIMINO, SOCRATOUS et al. 95 S. 282].
- Beim World Wide Web kann man nicht feststellen, ob sich eine bestimmte Information im System befindet [RAMM 95], man kann es also nicht als Informationssystem betrachten. Auch wird die mangelnde Qualität vieler Angebote im World Wide Web beklagt (z.B. [STOLL 96]). Ein anderer Autor warnt in diesem Zusammenhang vor der Gefahr, daß sich das WWW leicht zur weltgrößten „vanity press“[1] [LLAURADO 97] entwickeln kann, weil jedermann mit Computer gleichzeitig Autor, Redakteur und Verleger sein kann. Diese Rollen können alle anonym ausgefüllt werden. Die Gefahr für Neulinge besteht in der Schwierigkeit, Nützliches von Schädlichem zu trennen.
- Ein weiteres Problem stellen die URL’s (Uniform Ressource Locator) dar. Bei Umbenennung von Dokumenten oder Servern bzw. Umstrukturierungen zeigen zwangsläufig alle extern gesetzten Verweise auf die betroffenen Dokumente ins Leere. Weltweit zeigen nach Schätzungen bis zu 25% aller Verweise ins Leere.

[1] vanity (engl.) = Nichtigkeit

Das World Wide Web zeigt also klare Schwächen. Weitreichendere Ansätze wie Hyper-G [KAPPE, MAURER 93; DALITZ, HEYER 95], bei denen zumindest die technischen Schwächen entschärft sind, besitzen allerdings momentan keine großen Chancen, weil für das WWW mittlerweile ein riesiges und häufig auch noch kostenloses Softwareangebot existiert.

2.4.1 Hypertext und Hypermedia

Das Hypertext-Konzept wurde als erstes im Jahr 1945 von Vannevar Bush, einem Berater des damaligen US-Präsidenten Roosevelt, beschrieben [NIELSEN 95]. Obwohl das von Bush beschriebene System Memex zum Blättern in großen Textmengen unrealisiert blieb, stellte sich zunehmend heraus, daß die Vision von Bush doch realisiert werden sollte. So zwingt der Umfang von Handbüchern für hochkomplizierte Geräte wie Düsenflugzeuge (Handbuchumfang ca. 300000 Seiten [VENTURA 88, zitiert in SCHULMEISTER 96 S. 207]) dazu, die Dokumente in elektronischer Form abzulegen und den Nutzern verfügbar zu machen.
Nach Meinung von Spiro und Jehng [SPIRO, JEHNG 90, zitiert in SCHULMEISTER 96 S. 248] eignen sich Hypertext-Systeme besonders gut für „ill-structured domains“. Dagegen sind Methoden der künstlichen Intelligenz bisher weitgehend auf gut strukturierte Anwendungsgebiete beschränkt, z.B. Mathematik, Physik oder Programmiersprachen.
Kuhlen vermutet aufgrund bisheriger Studien, „...daß die nicht-linearen Eigenschaften von Hypertext Lernerfolge in komplexen Situationen begünstigen können, zumal dann, wenn ein bestimmtes Vorwissen vorausgesetzt werden kann.“ [KUHLEN 91]. Es kann beobachtet werden, daß Lernende mit größerem Eingangswissen eine effektivere Auswahl in Hypertexten treffen [Schulmeister 96]. Allerdings hängen der Lernerfolg bzw. -mißerfolg von vielen verschiedenen Faktoren ab, z.B. auch vom technologischen und methodischen Stand existierender Hypertext-Systeme. Generalisierende Aussagen sind deshalb momentan noch nicht angebracht.

Hypertext-Systeme können in drei Ebenen unterteilt werden [CAMPBELL AND GOODMAN 1988, zitiert in NIELSEN 95]:

Presentation level (Präsentationsebene):
Auf der Präsentationsebene wird definiert, welche Befehle für den Nutzer zugänglich sind, wie Knoten und Verweise aussehen usw. So können dem Neuling beispielsweise bestimmte Typen von Links vorenthalten werden.

Hypertext Abstract Machine (HAM) level:
Im *HAM-level* ist die Struktur der im System vorkommenden Knoten und Links definiert, z.B. verschiedene Link-Typen, Attribute von Knoten (Besitzer, letzte Änderung...). Für den Import und Export von Hypertexten ist der HAM-level am besten geeignet, da er plattformunabhängig ist (im Gegensatz zur Datenbankebene) und außerdem die Präsentationsebene wiederum von System zu System sehr stark abweicht. Der Austausch von Hypertexten ist allerdings nicht ganz einfach, da ja nicht nur die Texte sondern auch verschiedene Grafik-, Sound- und Videoformate ausgetauscht werden müssen. Außerdem müssen die möglicherweise verschiedenen Linktypen ebenfalls ohne Informationsverlust übergeben werden. Es ist recht unbefriedigend, wenn beim Export weitergehende Informationen zu einem Verweis verloren gehen (z.B. Informationen darüber, auf welches Objekt in einem Knoten ein Verweis zeigt).

Database level (Datenbankebene):
Die Datenbankebene hat wenig Spezifisches mit Hypertext zu tun. Es geht darum, die Daten so zu speichern, daß kleinste Teile davon in sehr kurzer Zeit abgerufen werden können. Auf Datenbankebene sind auch evtl. vorhandene Sicherheits- und Mehrfachzugriffsmechanismen angesiedelt. Die Datenbankebene ist hardwareabhängig.

Derzeit gibt es nur sehr wenige Systeme, welche sich in ihrer internen Struktur an diesem Modell orientieren. Jedoch ist das Modell nützlich für Standardisierungsbemühungen und zeigt interessante Perspektiven für die Zukunft [NIELSEN 95].

2.4.2 Nachteile von Hypertext

„Hypermedia is a non-pedagogical technology, one which is open to learning through browsing, but which must count on the student's own intelligence for learning guidance." [DUCHASTEL 92, zitiert in SCHULMEISTER 96 S. 185]

Hypertextsysteme können demzufolge nur bei Personen eingesetzt werden, die in der Lage sind, ihre eigenen Lernbedürfnisse zu erkennen und systematisch durch Navigation im Hypertext zu befriedigen. Hypertextsysteme können die Navigation und Suche durch verschiedene Ansätze allerdings erleichtern [NIELSEN 95]:

- Guided tours: Autoren legen für bestimmte Nutzergruppen Lernpfade fest. Personen aus der Nutzergruppe können den vorgegebenen Lernpfaden sequentiell folgen und müssen sich dabei nicht mit der Navigation auseinandersetzen. Der Lernpfad kann in der Regel an jeder beliebigen Stelle verlassen werden.
- Backtrack: Es können verschiedene Backtrack-Methoden verwendet werden. Die einfachste Lösung, einfach die Knoten in umgekehrter Reihenfolge nochmals aufzurufen ist sicherlich nicht sonderlich intelligent. Insbesondere wenn ein Nutzer einige Knoten mehrmals besucht hat. Der *single-revisit backtrack* ist eine sinnvolle Alternative. Die Knoten werden in umgekehrter Reihenfolge wieder aufgerufen, jedoch werden Knoten, welche mehrmals besucht wurden, nur einmal angesprungen.
- History lists: In einer Liste sind die letzten x angesprungenen Knoten eingetragen. Der Nutzer kann aus dieser Liste auswählen, wohin er springen möchte.
- Bookmarks: Lesezeichen (bookmarks) ermöglichen es einem Nutzer, Hypertext-Seiten zu markieren. Dadurch kann bei Bedarf schnell auf eine markierte Seite gesprungen werden. Eine evtl. langwierige Suche entfällt.

- Overview Diagrams**:** Übersichtsdiagramme zeigen, z.B. auf verschiedenen Abstraktionsebenen, einen Überblick über die verfügbaren Inhalte. Eine Alternative stellt der sogenannte *fisheye view* dar, bei dem einzelne Bereiche und Knoten um so detaillierter dargestellt sind, je näher sich diese am aktuellen Standort des Nutzers im Hypertext befinden. Bei der *link inheritance* werden Cluster gebildet. Nur noch die einzelnen Cluster sind dann durch Verweise (links) verbunden. Der Benutzer hat so bei größeren Hypertexten einen wesentlich besseren Überblick darüber, wie der Hypertext aufgebaut ist.
- Metaphors: Metaphern sollten im gesamten System konsistent angewendet werden und nicht wechseln, da dadurch der Nutzer verwirrt wird.

Cognitive Overhead

Unter *cognitive overhead* wird die Anstrengung verstanden, die man beim Lesen eines Hypertextes zusätzlich zur Anstrengung des Wissenserwerbs aufbringen muß, um sich im Verweisgeflecht überhaupt noch zurechtzufinden [CONKLIN 87]. Leser eines Hypertextes müssen sich stets Gedanken darüber machen, was sie schon gelesen haben, was sie noch interessiert und wo sie sich gerade befinden.

Lost in Hyperspace

Benutzer eines Hypertextsystems werden nicht geführt, sondern können sich vollkommen frei im Hyperspace bewegen. Dabei kann es leicht passieren, daß ein Nutzer den Überblick darüber verliert, wo er sich gerade befindet. Dieses Phänomen wird mit *lost in hyperspace* bezeichnet [HALASZ 88]. Dabei kann es auch vorkommen, daß er auf neue, ihn interessierende Informationen stößt und dabei vergißt, wonach er ursprünglich gesucht hatte.

2.5 WBT-Systeme

2.5.1 Architekturtypen

Das World Wide Web besitzt eine Client/Server-Architektur [NIEMANN 96], die es ermöglicht, darauf aufbauend verteilte Lehr-/Lernsysteme zu erstellen. Lehr-/Lernsysteme können allgemein in die drei Schichten Präsentationsschicht, Lehrsystemlogikschicht und Domänendaten- & Wissensschicht unterteilt werden. Die Präsentationsschicht stellt die Benutzungsschnittstelle zwischen Benutzer und Lehrsystemlogik dar. Benutzeraktivitäten (Mausaktionen, Eingaben usw.) werden von der Lehrsystemlogik verarbeitet und die Reaktion darauf an die Präsentationsschicht weitergereicht. Die Verarbeitung basiert auf Lehrwissen, welches in der Lehrsystemlogikschicht enthalten ist. Die Lehrsystemlogikschicht enthält also Wissen darüber, wie in angemessener Weise auf Benutzeraktionen reagiert werden kann und welche Art der Rückmeldung für den Lernprozess in bestimmten Situationen förderlich ist. In der Domänendaten- & Wissensschicht sind das in einem Lehr-/Lernsystem verfügbare und zu vermittelnde Domänenwissen sowie multimedialen Informationseinheiten (Bilder, Videos, Sounds usw.) abgelegt. Es kann auf unterschiedlichste Arten dort repräsentiert sein, z.B. in einer Datenbank, in Textdateien usw. Abhängig davon, wie diese Schichten bei einem WBT-System auf Client und Server verteilt werden, ergeben sich unterschiedliche Architekturtypen mit charakteristischen Stärken und Schwächen.

Insgesamt können vier unterschiedliche Architekturtypen bei WBT-Systemen unterschieden werden [HAAG, MAYLEIN et al. 98]. Drei Architekturtypen (*Remote Data & Knowledge, Distributed Teaching* und *Server-based*) lassen sich dabei nur bei WBT-Systemen finden, *Client-based* ist der Architekturtyp konventioneller CBT-Systeme.
Für die Akzeptanz von WBT-Systemen spielt die System-Performance eine wichtige Rolle. Neben der Leistungsfähigkeit von Client- und Server-Computer und der verfügbaren Netzbandbreite wird diese von den verwendeten Basistechniken und der Systemarchitektur maßgeblich beeinflußt.

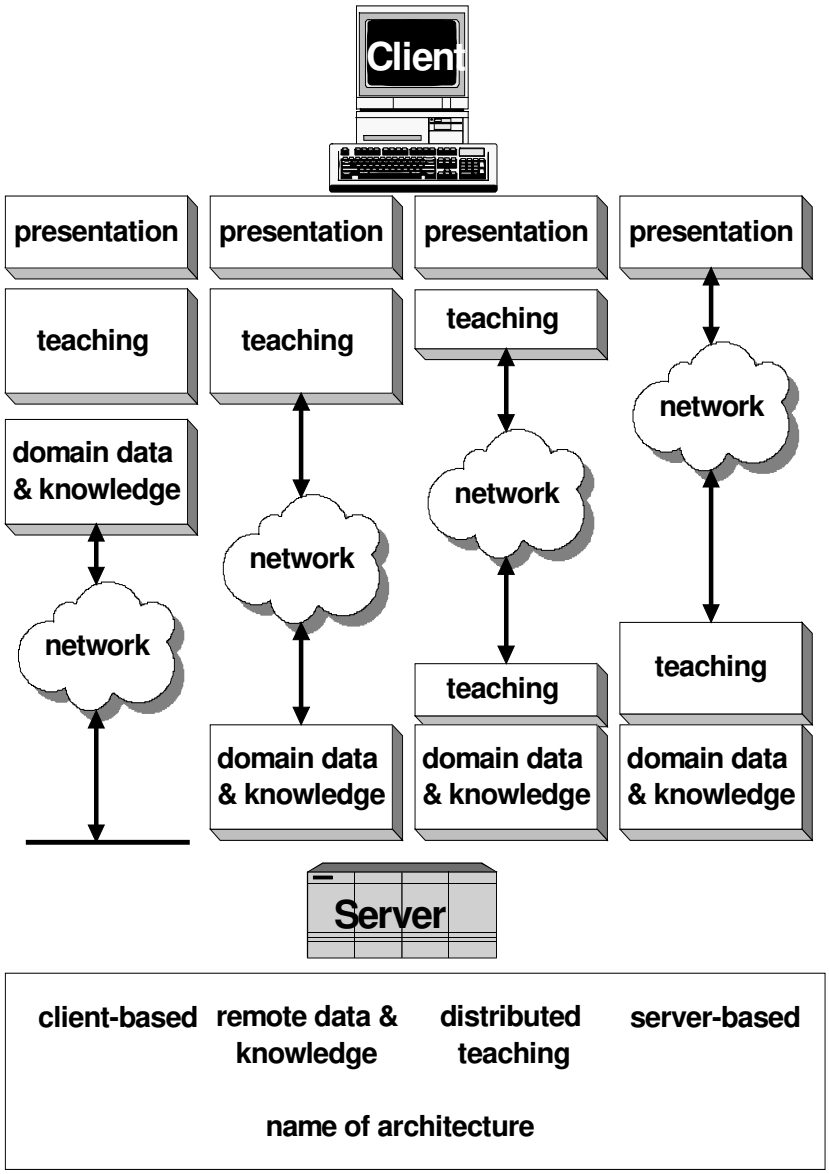

Abbildung 4: Architekturtypen bei WBT-Systemen

Für den Start und die Initialisierung einer notwendigen Laufzeitumgebung (z.B. Browser-Plug-In oder virtuelle Maschine) wird Zeit benötigt, allerdings in der Regel nur wenige Sekunden und vor dem Start des Lehr-/Lernsystems. Dieser Zeitbedarf ist deshalb normalerweise unkritisch. Weiterhin spielt die Performance der verwendeten Basistechnik(en) eine Rolle. CBT-Systeme benötigen jedoch in der Regel keine sonderlich große Rechenlei-

stung, so daß die meisten Basistechniken hinreichende Performance bieten. Eine große Auswirkung auf die Performance hat die Menge des vom Server zum Client zu übertragenden Codes, insbesondere wenn nur eine geringe Netzbandbreite verfügbar ist. Wenn der gesamte System-Code vor dem Start zum Client übertragen wird, dann ergeben sich schnell inakzeptable Wartezeiten. Werden stets nur die aktuell benötigten Teile übertragen, können häufige und längere Wartezeiten während des Programmlaufs die Folge sein. Auswirkungen auf die System-Performance hat auch die Häufigkeit und Dauer von Kommunikationsvorgängen zwischen Lehr-/Lernsytemclient und Lehr-/Lernsystemserver während des Programmlaufs.

2.5.1.1 Client-based Architektur

WBT-Systeme vom Typ *Client-based* besitzen die Architektur *konventioneller* CBT-Systeme. Das komplette WBT-System wird über Rechnernetz auf den Client übertragen und dort ausgeführt. Der Server dient lediglich als Fileserver, er besitzt sonst keinerlei weitere Funktionen. Alle drei Schichten des Lehr-/Lernsystems sind also auf dem Client angesiedelt (siehe Abbildung 4 [HAAG, MAYLEIN et al. 98]). Für die Entwicklung von Client-basierten WBT-Systemen stehen eine Reihe verschiedener Basistechniken zur Verfügung (siehe Kapitel 2.6.2). Der Vorteil besteht darin, daß sich die Entwicklung solcher Systeme kaum von der Entwicklung konventioneller CBT-Systeme unterscheidet. Entwickler von konventionellen CBT-Systemen können deshalb auch schnell Client-basierte WBT-Systeme erstellen. Teilweise können sogar konventionelle CBT-Systeme, die für den Betrieb auf stand-alone-Computern erstellt wurden, mit geringem Aufwand zu einem Client-basierten WBT-System umgewandelt werden. Möglich ist dies z.B. bei konventionellen CBT-Systemen, welche mit dem Autorensystem Macromedia Director erstellt wurden. Diese sind nach der Umwandlung durch ein mitgeliefertes Tool (Afterburner) in WWW-Browsern mit installiertem Shockwave-Plug-In ablauffähig.
Ein Beispiel für diesen Architekturtyp ist „The interactive exploration of the fundus diabeticus“ [DAETWYLER]. Hier werden dem Nutzer Strukturen genannt, die er mit der Maus auf dem Bild eines Augenhintergrundes anklicken muß. Er erhält dabei eine Rückmeldung, ob er diese korrekt erkannt hat oder nicht.
Der Hauptnachteil dieser Art von Systemen liegt darin, daß sie bei ihrem Aufruf grundsätzlich komplett auf den Client-Rechner über Netz geladen werden müssen, was je nach Umfang des Systems und der verfügbaren Netzbandbreite sehr lange dauern kann.

2.5.1.2 Remote Data & Knowledge Architektur

Bei diesem Systemtyp verbleibt die Domänendaten- & Wissensschicht auf dem Server (siehe Abbildung 4). Die Lehrsystemlogik greift während des Programmablaufs unter Anwendung des in der Lehrsystemlogikschicht enthaltenen Lehrwissens darauf zu. Der Vorteil einer solchen Architektur besteht gegenüber der Client-basierten Architektur darin, daß nicht alle im System verfügbaren multimedialen Informationselemente und das komplette vorhandene Domänenwissen sondern nur die tatsächlich benötigten Teile vom Server auf den Client übertragen werden müssen. Außerdem können im Hintergrund, unabhängig von den Aktivitäten des Nutzers, Domänendaten und Domänenwissen prospektiv auf den Client geladen werden. Dadurch kann, insbesondere bei langsamen Netzwerkverbindungen, Performance und Reaktionszeit des Systems stark verbessert werden.
Allerdings wird dadurch, daß Lehrwissen (Lehrsystemlogikschicht) sowie Domänendaten und Domänenwissen (Domänendaten- & Wissensschicht) auf verschiedenen Rechnern angesiedelt sind, mehr Netzverkehr erzeugt, als unbedingt erforderlich. Verglichen mit den anderen Architekturtypen kommunizieren Client und Server häufiger miteinander. Mehr-

stufige Anfragen an die Daten- und Wissensschicht führen zu einer hohen Netzbelastung. Denn der Server muß die Ergebnisse jeder Abfrage zum Client schicken. Erst dort können sie, basierend auf dem vorhandenen Lehrwissen analysiert und neue Abfragen erstellt werden. Diese müssen dann zum Server zurückgeschickt werden usw.
Ein Beispiel für diesen Architekturtyp ist das in Kapitel 2.3.3 beschriebene CASUS/ProMediWeb-System.
Es gibt eine ganze Reihe verschiedener Anbieter, welche Werkzeuge zum transparenten Zugriff auf die Domänendaten- & Wissensschicht anbieten. Die Entwicklung von Systemen mit einer solchen Architektur ist deshalb kaum schwieriger als die einer Client-based-Architektur. Symantec bietet beispielsweise mit der *Visual Database Development Edition* [SYMANTEC] eine Entwicklungsumgebung, mit der leicht auf entfernte relationale Datenbanken zugegriffen werden kann. Als Entwickler muß man sich dabei nicht um die Kommunikation zwischen Datenbankserver und Client kümmern. Clientseitig ist auch kein Datenbanktreiber erforderlich. Eine Umwandlung bereits existierenden, konventioneller CBT-Programme in diese Architektur ist möglich, falls die Domänendaten- & Wissensschicht von der Lehrsystemlogikschicht getrennt ist. Dies ist bei kartenbasierten Systemen, die mit HyperCard oder Toolbook erstellt wurden, nicht der Fall. Solche Systeme sind für die Umwandlung deshalb wenig geeignet.

2.5.1.3 Distributed Teaching Architektur

Bei diesem Architekturtyp ist die Lehrsystemlogikschicht aufgeteilt. Ein Teil der Lehrsystemlogik ist auf dem Client angesiedelt, der andere Teil auf dem Server (siehe Abbildung 4). Anfragen an die Domänendaten- & Wissensschicht können hier in Abhängigkeit von den vorhergehenden Nutzeraktionen auf dem Server erzeugt werden. Auch die Ergebnisse der Abfragen können dort analysiert werden. Dadurch ergeben sich Performance-Vorteile. Mehrstufige Anfragen an die Domänendaten- & Wissensschicht können komplett auf dem Server erzeugt und durchgeführt werden. Lediglich die „Endergebnisse“ solcher Anfragen müssen zum Client übertragen werden.
Die Auswahl einer geeigneten Wissenskontrollfrage aus einer Menge von verfügbaren Fragen ist ein gutes Beispiel zur Verdeutlichung der Vorteile dieser Architektur. Geeignete Fragen an einen Nutzer sollten einen angemessenen Schwierigkeitsgrad besitzen, sie sollten der aktuellen Lernsituation entsprechen und sie sollten für den Nutzer neu sein. Falls bei der ersten Abfrage an die Domänendaten- & Wissensschicht eine solche Frage nicht gefunden werden kann, kann durch weitere Anfragen eine Frage mit bestmöglicher Näherung an die Anforderungen gesucht werden. Falls mehrere Fragen die Anforderungen erfüllen, dann kann basierend auf dem Lehrwissen auf dem Server eine der Fragen ausgewählt und zum Client gesendet werden. Diejenigen Teile der Lehrsystemlogikschicht, welche für die Überwachung der Nutzeraktivitäten und für angemessene Rückmeldungen zuständig sind, können auf dem Client angesiedelt werden. Falls Domänendaten oder Domänenwissen für diese Aufgaben erforderlich sind, genügt es, eine kurze Nachricht zum Server zu senden. Die *Distributed Teaching*-Architektur minimiert den Netzwerkverkehr und bietet große Vorteile, insbesondere wenn nur schmalbandige Netzanbindungen (z.B. über Modem) zur Verfügung stehen.
Wie auch bei den *Remote Data & Knowledge*-WBT-Systemen ist es möglich, prospektiv Domänendaten und -Wissen unabhängig von den Nutzeraktivitäten vom Server zu laden. Werden diese Daten und dieses Wissen benötigt, stehen sie bereits auf dem Client zur Verfügung und brauchen nicht mehr erst vom Server angefordert zu werden. Die Reaktionsgeschwindigkeit eines Systems kann dadurch optimiert werden. WBT-Systeme mit dieser Architektur bieten einerseits die beste Performance und die geringste Netzbelastung aller

Architekturen, sind andererseits allerdings bei der Implementierung auch am aufwendigsten. Es müssen zwei Applikationen, eine Client-Applikation und eine Server-Applikation erstellt werden, welche miteinander über Rechnernetze kommunizieren müssen. Auch diese Kommunikation zwischen Client und Server muß implementiert werden. Als Basistechniken für die Implementierung werden Techniken wie CORBA (Common Object Request Broker Architecture) oder RMI (Remote Method Invocation) benötigt, mit denen entfernte Methodenaufrufe möglich werden (siehe Kapitel 2.5.2).
Eine Umwandlung bereits existierender CBT-Systeme in eine *Distributed Teaching*-Architektur ist sehr aufwendig, da neben der Abspaltung der Domänendaten- und Wissensschicht auch eine Aufteilung der Lehrsystemlogikschicht notwendig ist. Die Kommunikation zwischen den beiden Teilen der Lehrsystemlogikschicht über Netzwerk muß ebenfalls konzipiert und implementiert werden.

2.5.1.4 Server-based Architektur

Bei diesem Architekturtyp ist lediglich die Präsentationsschicht auf dem Client angesiedelt. Lehrwissen sowie Domänendaten und Domänenwissen befinden sich auf dem Server. Die Präsentationsschicht leitet alle Nutzereingaben an den Server weiter. Dort werden sie verarbeitet und das Ergebnis der Verarbeitung wird an den Client zurückgeliefert. Dies geschieht normalerweise in Form eines HTML-Files. Der Vorteil von *Server-based* Architekturen liegt darin, daß auf Server-Seite beliebige Werkzeuge verwendet werden können, ohne die Plattformunabhängigkeit auf Client-Seite zu verlieren. Entwickler können deshalb die Domänendaten- & Wissensschicht sowie die Lehrsystemlogikschicht mit ihnen vertrauten Werkzeugen erstellen und müssen sich nicht erst in neue Werkzeuge einarbeiten. Die Kommunikation zwischen Präsentationsschicht und Lehrsystemlogikschicht wird bei Server-basierten Systemen in der Regel über die CGI-Schnittstelle eines WWW-Servers abgewickelt. Durch diese Schnittstelle kann ein WWW-Server externe Programme starten und diese mit Daten versorgen. WBT-Systeme mit *Server-based* Architektur sind allerdings nicht sehr performant, weil sämtliche Benutzeraktivitäten vom Server ausgewertet werden müssen (der Aufbau einer Netzverbindung ist jedesmal erforderlich) und bei zu vielen gleichzeitigen Nutzern kann leicht der Serverrechner überlastet werden, was zu geringer Performance des Gesamtsystems führt. Ein Beispiel für ein WBT-System mit Server-basierter Architektur ist der „Interactive Patient“ der *Marshall University* [HAYES, LEHMANN 96]. Bei einem Patienten muß hier die komplette Untersuchung, bestehend aus Anamnese, klinische Untersuchung, technische Untersuchung und Laboruntersuchung durchgeführt werden. Danach muß der Anwender eine Diagnose stellen und einen Therapieplan auf Basis der eruierten Daten aufstellen. Das System wertet alle Benutzereingaben aus und gibt Feedback.
Zur Verfügbarmachung bereits existierender Intelligenter Tutorieller Systeme im Internet ist die Server-basierte Architektur sehr gut geeignet. Die Benutzungsschnittstelle eines existierenden konventionellen CBT-Systems muß hierzu durch eine beispielsweise mit HTML erstellte Benutzungsschnittstelle ersetzt und an die vorhandene Lehrsystemlogikschicht angebunden werden. Inferenzmechanismus und Wissensbasis können unverändert auf dem Server verbleiben.

2.5.2 Client/Server Kommunikation

Für die Erstellung von WBT-Systemen mit Distributed Teaching-Architektur müssen Kommunikationsmechanismen zwischen Lehr-/Lernsystemserver und -client entwickelt werden. Geeignet hierfür sind vor allem RMI (Remote Method Invocation) und CORBA (Common Object Request Broker Architecture).

RMI

Für grundlegende Kommunikationsmechanismen stehen in Java sogenannte *Sockets* zur Verfügung. Sockets sind sehr flexibel und zufriedenstellend für die allgemeine Kommunikation. Allerdings muß sowohl auf Client-Seite als auch Server-Seite die zu übermittelnden Nachrichten codiert und decodiert werden. Es muß also ein Protokoll für den Datenaustausch entworfen und implementiert werden.
Eine Alternative stellen *Remote Procedure Calls (RPC)* dar. Dadurch wird das Kommunikationsinterface auf die Ebene der Prozeduraufrufe abstrahiert. Anstatt direkt mit Sockets zu arbeiten, ruft ein Programmierer entfernte Prozeduren auf, als wären diese lokal. Die Argumente eines Aufrufs werden verpackt und an den entfernten Rechner übertragen.
Remote Procedure Calls lassen sich jedoch nicht sehr gut in verteilte objektorientierte Systeme übertragen. Für diese Art von Systemen gibt es die *Remote Method Invocation (RMI)* [SUN]. Darunter versteht man einen Methodenaufruf eines auf einem entfernten Rechner angesiedelten Objektes. Die Syntax zwischen dem Aufruf eines lokalen Objektes und eines entfernten Objektes unterscheiden sich dabei nicht.

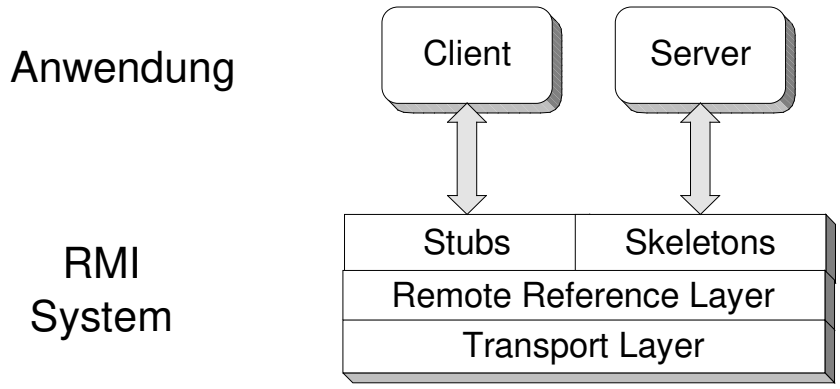

Abbildung 5: RMI-Systemarchitektur

Java-RMI wurde speziell für die Java-Umgebung entwickelt. Während andere RMI-Systeme dahingehend angepaßt werden können, daß sie Java-Objekte handhaben können, haben diese Systeme Nachteile bei der optimalen Integration von Java, da sie die Interoperabilität mit anderen Sprachen erhalten müssen. Dagegen geht Java-RMI von einer homogenen Java-Umgebung aus und kann deshalb oft von den Vorteilen des Java Objektmodells profitieren. Das Java RMI-Modell ist vollkommen in Java integriert. Dadurch wird die einfache Erstellung von zuverlässigen, verteilten Anwendungen unterstützt. Die Sicherheit, die durch die Java Laufzeitumgebung (*runtime environment*) gegeben ist, wird bei Java-RMI bewahrt und die automatische *garbage collection*, wenn ein entferntes Objekt nicht mehr benötigt wird, wird unterstützt. Sogar ein entfernter Methodenaufruf über Firewalls hinweg ist mit RMI möglich. Um die transparente Übertragung von Objekten

von einem Adressraum in einen anderen zu bewerkstelligen, wird die Technik der *object serialization* eingesetzt.

Das RMI-System besteht aus drei Schichten:

1. stub/skeleton layer:
Die stub/skeleton-Schicht besteht auf Client-Seite aus sogenannten *stubs* (proxies) und auf Server-Seite aus *skeletons*. Deren Aufgabe besteht u.a. im Zusammenstellen und Auflösen der Argumente zum *marshal stream*, der zwischen den Rechnern übertragen wird.

2. remote reference layer:
Die remote reference-Schicht bildet die Verbindungsschicht zwischen Transportschicht und stub/skeleton-Schicht. Sie ist unabhängig von diesen Schichten und verantwortlich für die Ausführung der Semantik eines Aufrufs.

3. transport layer:
Die Transportschicht ist verantwortlich für den Verbindungsaufbau und das Verbindungsmanagement. Sie basiert auf dem TCP-Protokoll (Transmission Control Protocol) und verwendet Java Sockets.

Alle drei Schichten sind vollkommen unabhängig voneinander und können einzeln ausgetauscht werden.

CORBA

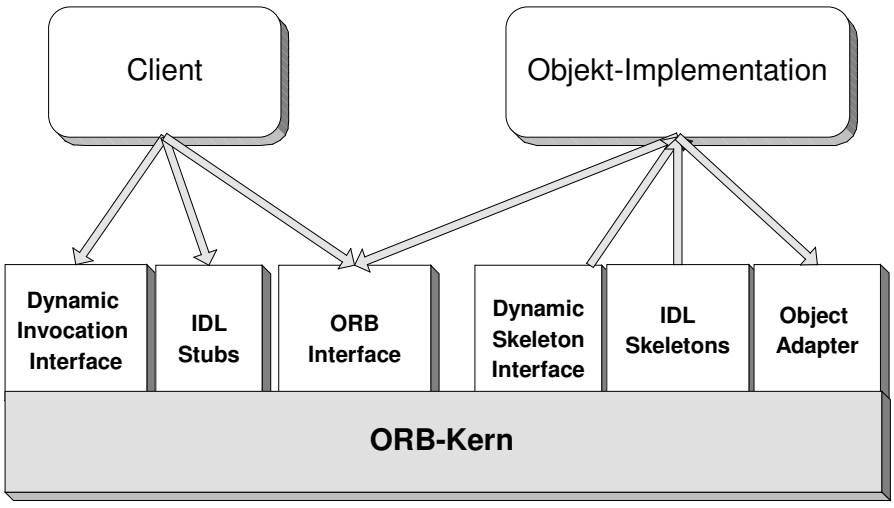

Abbildung 6: CORBA (Common Object Request Broker Architecture)

CORBA (*Common Object Request Broker Architecture*) [REDLICH 96, SAYEGH 97] geht im Gegensatz zu RMI von einer heterogenen, mehrsprachigen Umgebung aus und muß deshalb ein sprachunabhängiges Objektmodell besitzen. Der CORBA-Standard der OMG (Object Management Group) definiert u.a. Schnittstellen zur Kommunikation zwischen verteilten Objekten. Zentraler Bestandteil der CORBA-Architektur ist der sogenannte *Object Request Broker (ORB)*, über den Nachrichten zwischen verschiedenen Objekten aus-

getauscht werden. Durch den ORB wird Zugriffstransparenz erreicht. Das sendende Objekt braucht nicht zu wissen, auf welchem Rechner das Empfängerobjekt angesiedelt ist und mit welcher Programmiersprache dieses erstellt wurde. Der CORBA-Standard definiert außerdem sogenannte *Application Objects*, *Object Services* und *Common Facilities* [KALKBRENNER 96 S. 80f.]. Application Objekts sind Dienste der Anwendung, Object Services sind Dienste wie Erzeugen, Benachrichtigen und Löschen von Objekten. Common Facilities schließlich erlauben das Drucken oder die Präsentation.

Das Erstellen einer CORBA-Anwendung geschieht in mehreren Schritten [REDLICH 96 S. 3] und ist aufwendiger als das Handling von RMI:

1. Interface-Beschreibung für das entfernte Objekt erstellen
2. Interface-Beschreibung übersetzen
3. Server implementieren
4. Server registrieren
5. Klienten implementieren
6. Alle Komponenten aktivieren (Ausführen der Anwendung)

Weil CORBA ein sprachunabhängiges Objektmodell besitzt, müssen die Schnittstellen zwischen Objekten mit einer speziellen Sprache, der sogenannten *Interface Definition Language (IDL)* definiert werden. Aus einer Schnittstellenbeschreibung wird dann die Schnittstelle für die jeweils verwendete Programmiersprache generiert. Die Regeln zur Umsetzung der IDL-Definitionen in semantisch äquivalente Java-Konstrukte werden als *Language Mapping* bezeichnet.

2.5.3 Integration vorhandener WWW-Ressourcen

Im WWW ist eine Vielzahl hochwertiger medizinischer Ressourcen vorhanden, welche teilweise auch speziell für Zwecke der Aus- und Weiterbildung entwickelt wurden. Neben speziell für die Ausbildung konzipierten WBT-Systemen lassen sich auch Atlanten (The Whole Brain Atlas [THE WHOLE BRAIN ATLAS]), Sammlungen von Fallbeschreibungen (Pittsburgh Case Index [PITTSBURGH CASE INDEX]), Handbücher und Referenzwerke (Universtity of Iowa Family Practice Handbook [UNIVERSTITY OF IOWA FAMILY PRACTICE HANDBOOK]), diagnostische und therapeutische Leitlinien (HSTAT [HSTAT]; QUADE, PUSCHEL et al. 96), Literaturreferenz-Datenbanken (PubMed - Free MEDLINE [PUBMED-FREE MEDLINE]), Diskussionsforen (WebDoctor MDForum [WEBDOCTOR MDFORUM]) usw. finden. Darüber hinaus sind umfangreiche virtuelle Bibliotheken in der Entwicklung (z.B. [STANFORD UNIVERSITY DIGITAL LIBRARIES PROJECT, UC BERKELEY DIGITAL LIBRARY PROJECT]), bei denen Texte, Bilder und Videos aus verschiedenen Fachgebieten komfortabel und ortsunabhängig verfügbar gemacht werden sollen.
Die relativ einfache Integrationsmöglichkeit solcher Ressourcen in eigene Systeme stellt einen großen Vorteil des WWW dar (siehe z.B. [OLIVER, WINGERT et al. 96]). Entwickler von Lehr-/Lernsystemen können so ohne großen Aufwand Möglichkeiten für die Lernenden schaffen, das erworbene Wissen individuell zu vertiefen. Die zwangsläufig vorhandenen Grenzen konventioneller Systeme können dadurch stark erweitert werden.
Allerdings bringt die Integration fremdverwalteter Wissensressourcen im WWW in eigene Lehr-/Lernsysteme auch eine Reihe von Problemen mit sich [MAYLEIN, HAAG 98]. Sie sollen nachfolgend diskutiert werden.

2.5.3.1 Unterschiedliche Repräsentationskonzepte

Von traditionellen Hypertextsystemen unterscheidet sich das WWW nicht nur durch seine Größe, die weltweite Verteilung und das häufig redundante Angebot an Wissen und Informationen. Ein wichtiger Unterschied liegt auch in der Vielzahl unterschiedlicher Konzepte, mit denen Informations- und Wissenseinheiten auf Knoten und Kanten der Hypertextbasis abgebildet werden. Die Autoren von WWW-Ressourcen können ohne Einschränkungen ihre eigenen Konzepte umsetzen. Außerdem können sie beliebige Verweise definieren, mit denen auch die Grenzen von unterschiedlichen Repräsentations- und Präsentationskonzepten überschritten werden können. Die Integration einer externen Ressource in ein bestehendes System setzt deshalb ein abstrahiertes Beschreibungsschema voraus, welches sich mehr an den bereits vorhandenen Informations- und Wissenseinheiten orientiert und weniger an einzelne Hypertextknoten.
Der Begriff *Ressource* soll als Sammlung von permanent vorliegenden bzw. dynamisch erzeugten Dokumenten angesehen werden, welche einem einheitlichen Repräsentations- und Präsentationskonzept folgen. Ressourcen stellen darüber hinaus ein Verfahren zum Dokumentenzugriff zur Verfügung.

2.5.3.2 Granularitätsproblem

Bei der Einbindung bestehender Ressourcen müssen den Lernenden die Probleme einer eigenständigen Informations- und Wissenssuche erspart bleiben, welche sich aus der unzureichenden Beschreibung von Objekten im WWW durch Suchhilfen in Verbindung mit der großen Heterogenität der verwendeten Konzepte ergeben.
Obwohl im WWW viel medizinisches Wissen zur Verfügung steht (z.B. [HERSH, BROWN et al. 96], erweist sich die Suche nach qualitativ hochwertigem und relevantem Wissen oftmals als frustrierend [DETMER, SHORTLIFE 97]. Lernende müssen also vom Lehr-/Lernsystem so direkt wie möglich zu den, bezogen auf die jeweilige Lernsituation, relevanten und adäquaten Dokumenten geführt werden. Dies gilt insbesondere für Intelligente Tutorielle Systeme.
Um die Lernenden möglichst gut führen zu können, ist meist eine detaillierte Betrachtung auf Ebene einzelner Dokumente, gegebenenfalls unter Berücksichtigung des ressourcenabhängigen Ressourcenzugriffs, erforderlich. Aber auch eine Betrachtung der Ressource als Ganzes ist notwendig, dient sie doch als Vertrauensbasis in Bezug auf die Qualität des vorhandenen Wissens zwischen Wissenssuchendem und Ressource. Da ein Ressourcenwechsel meist auch mit einem Wechsel der Repräsentations- und Präsentationskonzepte einhergeht, können die Lernenden diesen bewußt vornehmen und sich auf die neuen Konzepte einstellen. Die Gefahr eines durch häufige und unbewußte Ressourcenwechsel ausgelösten *lost in hyperspace*-Syndroms (siehe Kapitel 2.4.2) kann dadurch verringert werden. Keinesfalls dürfen durch Bestrebungen, einen einheitlichen Zugriff auf verschiedene Ressourcen zu ermöglichen, die Heterogenität der Informationen und Zugriffsmechanismen verborgen werden [RAO, PEDERSEN et al. 95]. Ansonsten wird für die Lernenden die Einordnung und das Wiederfinden von Informationen erschwert.

2.5.3.3 Ressourcenqualität

Weil im WWW jede Person mit Computer gleichzeitig Autor, Redakteur und Verleger sein kann [LLAURADO 97] und Qualitätsstandards fehlen, wie sie bei papiergebundenen Medien angewendet werden, ist die Qualität der medizinischen Ressourcen sehr unterschiedlich.
Viele Dokumente sind „unvollständig, irreführend oder fehlerhaft“ und es ist oftmals schwierig, „die Spreu vom Weizen, das Nützliche vom Schädlichen zu trennen“ [SILBERG, LUNDBERG et al. 97]. Vielleicht werden sich aber im WWW in Zukunft Qualitätszertifikate wie *HON Code of Conduct* [HON] oder *Medical Matrix Code of Conduct* als allgemeingültiger Standard oder Maßstab durchsetzen. Zumindest können diese Anhaltspunkte für die Beurteilung der Qualität einer zu integrierenden Ressource bieten.
Die zu erwartende Qualität einer konventionellen Ressource ist das wichtigste Kriterium für deren Konsultation im medizinischen Bereich [COVELL, UMAN et al. 85]. Deshalb muß für das WWW gefordert werden, daß für die Lernenden bestehende Qualitätskontrollen und -zertifikate transparent gemacht werden.

2.5.3.4 Dynamik des WWW

Das WWW weist als Besonderheit eine sehr hohe Dynamik auf. Selbst wenn nur Ressourcen bei der Integration verwendet werden, die sich nur relativ selten ändern, muß trotzdem gewährleistet sein, daß relevante inhaltliche und strukturelle Änderungen erkannt und berücksichtigt werden können. Hierfür gibt es im WWW eine recht große Zahl an technischen Hilfsmitteln (siehe z.B. [LINKS CHECKERS]). Das bloße Erkennen von Änderungen ist jedoch nicht ausreichend. Auf organisatorischer Seite müssen geeignete Reaktionen definiert werden, um bei Änderungen schnell und effizient die erforderlichen Anpassungen am WBT-System durchführen zu können.

2.5.3.5 Grade der Integrierbarkeit

Wie gut externen Ressource überhaupt in WBT-Systeme eingebunden werden können, ist primär vom verwendeten Dokumentenzugriffsverfahren der zu integrierenden Ressource abhängig. Es lassen sich drei Kategorien unterscheiden:

Hierarchische Zugriffsstrukturen (Menüstrukturen):

Diese Gruppe von Ressourcen besteht aus statischen Wissensdokumenten sowie in der Regel aus weiteren hierarchisch angeordneten Dokumenten, welche einen Zugriff per Browsing erlauben. Abhängig vom verwendeten Repräsentationskonzept, können solche Ressourcen leicht integriert werden. Eine direkte Adressierung und Beschreibung einzelner Dokumente ist möglich. Der Aufwand für die Integration ist recht hoch, da zu bestimmten Lernsituationen jeweils die geeigneten Dokumente zugeordnet werden müssen. Gegebenenfalls kann allerdings die Zuordnung auf einer höheren Hierarchieebene erfolgen, bei der ein Dokument stellvertretend für alle untergeordneten Einzeldokumente steht. Dadurch verringert sich der Integrationsaufwand für die Entwickler. Allerdings muß dann darauf geachtet werden, daß das integrierte Dokument für die konkrete Lernsituation nicht zu unspezifisch ist.

Serverseitige Anfragemechanismen:

Bei diesen Zugriffsverfahren erzeugt ein Prozeß auf einem WWW-Server anhand von Anfrageparametern Dokumente. Einzelne Dokumente können nicht direkt adressiert werden. Allerdings können geeignete Abfragen mit bestimmten Lernsituationen verknüpft bzw. auf Basis von dieser generiert werden. Ein Beispiel für diese Form der Integration bietet der *Dermatologische Online Atlas* der Universität Erlangen-Nürnberg [BITTORF, BAUER et al. 95]. Dort können zum aktuellen Atlaseintrag relevante Literaturreferenzen aus der im WWW frei verfügbare Medline-Version *PubMed* der National Library of Medicine (NLM) abgefragt werden. Für die Integration von externen, serverseitigen Anfragemechanismen ist häufig eine Umsetzung zwischen verschiedenen medizinischen Terminologiesystemen bzw. eine Anpassung an die Terminologie der externen Ressourcen erforderlich. Im Dermatologischen Online Atlas wird hierfür das unter anderem zu diesem Zweck entwickelte *Unified Medical Language System (UMLS)* [LINDBERG, HUMPHREYS et al. 93] der NLM (National Library of Medicine) verwendet.

Anwendungsgesteuerte :

Anwendungsgesteuerte Dokumentenzugriffe sind WWW-Anwendungen, welche z.B. auf Programmiersprachen wie Java [RODGERS 96; MIDDENDORF, SINGER et al. 96] oder JavaScript [FLANAGAN, KUHNERT 98] basieren. Allerdings führt auch bereits der Einsatz von HTML-Frames, Plug-Ins oder ActiveX [SCHMITT 96] in der Regel dazu, daß die einzelnen Dokumente nicht mehr direkt beschrieben und angesprochen werden können. Sie sollen deshalb ebenfalls unter dieser Gruppe subsumiert werden. Generell lassen sich solche Ressourcen nur schlecht, und wenn, dann meist nur als Ganzes integrieren. Dadurch ergibt sich der große Nachteil, daß sich die Lernenden mit der Funktionalität der integrierten Ressourcen auseinandersetzen müssen.

2.6 Plattformunabhängigkeit

2.6.1 Portabilität

Für Balzert [BALZERT 96 S. 776] bedeutet Portabilität, daß Konzepte, welche bei der Erstellung der Anwendungssoftware benutzt werden, auf unterschiedlichen Rechnerplattformen (von unterschiedlichen Herstellern) zur Verfügung stehen. Er definiert drei Abstufungen von Portabilität:

Objektcode-Portabilität (binäre Portabilität):

Die lauffähige Anwendung kann ohne weitere Maßnahmen auf verschiedenen Plattformen ausgeführt werden. Java-Applikationen zeichnen sich durch binäre Portabilität aus. Durch Kompilieren von Java-Programmcode wird plattformunabhängiger Byte-Code erzeugt, welcher von einer für fast alle Plattformen verfügbaren *Java Virtual Machine* interpretiert werden kann, ohne daß irgendwelche Änderungen am Programmcode beim Wechsel der Rechnerplattform notwendig wären.

Quellcode-Portabilität:
Der Quellcode einer Anwendung kann von einer Plattform auf andere Plattformen übertragen werden. Dort wird der Quellcode neu compiliert und kann danach ausgeführt werden.

Entwurfs-Portabilität:
Eine Anwendung ist so konzipiert, daß die Konzepte leicht in verschiedene Implementierungen transformiert werden können.

Plattformunabhängigkeit wird im Folgenden immer im Sinne von *binärer Portabilität* verstanden. D.h., plattformunabhängige WBT-Systeme sollen ohne Änderungen am Quellcode und erneute Compilierung auf verschiedenen Rechnerplattformen lauffähig sein.

2.6.2 Basistechniken für die Erstellung plattformunabhängiger WBT-Systeme

Nachfolgend sollen die wichtigsten Basistechniken kurz angesprochen werden, die zur Zeit für die Erstellung von plattformunabhängigen Lehr-/Lernsystemen verfügbar sind.

HTML:
HTML [RAGGETT 97] ist die Seitenbeschreibungssprache des World Wide Web. Sie dient als Grundlage für die Erstellung von WWW-Seiten. Die im Weiteren aufgeführten Basistechniken sind bei WBT-Systemen in HTML-Code eingebettet. Die erstellten Systeme laufen dadurch im WWW-Browserfenster. HTML ist auf praktisch allen Plattformen verfügbar und es existieren sehr leistungsfähige und häufig kostenlose Werkzeuge zur Erstellung von HTML-Dokumenten [SCHULZ, SCHRADER et al. 97]. Auch Personen ohne Programmiererfahrung können damit Präsentationssysteme ohne Probleme erstellen.

Java:
Java [FLAGANAN 98] ist eine neue, objektorientierte Programmiersprache mit „C“-ähnlicher Syntax. Für die Plattformunabhängigkeit sorgt der *byte code virtual machine (VM) interpreter*. Java Source-Code wird vom Java Compiler zu Byte-Code compiliert und von der virtuellen Maschine ausgeführt. Einer der wichtigsten Vorteile von Java ist die ausgezeichnete Netzwerkunterstützung. So kann mit entfernten Objekten einfach über Java-RMI (siehe Kapitel 2.5.2) kommuniziert werden. Der Hauptnachteil von Java im Vergleich zu anderen Basistechniken ist der vergleichsweise hohe Zeitaufwand für die Erstellung von Systemen.

JavaScript:

JavaScript [FLANAGAN, KUHNERT 97] ist eine kompakte, *objektbasierte*[2] Skriptsprache, die Bestandteil der Netscape WWW-Browser seit Version 2.0 ist und in einer Untermenge auch im Internet Explorer von Microsoft verfügbar ist. Die JavaScript-Sprache hat große Ähnlichkeiten mit Java und damit zu C, ist jedoch leichter zu erlernen und zu programmieren, da lediglich die bereits vorhandenen Objekte und Funktionen aufgerufen werden müssen. JavaScript eignet sich nur für relativ beschränkte Problemstellungen, da es nicht beliebig erweiterbar und bei weitem nicht so flexibel wie Java ist. Auch sind die Sicherheitsmechanismen von JavaScript nicht so stark ausgeprägt. JavaScript ist außerdem wesentlich langsamer als Java.

Plug-Ins:

Plug-Ins ermöglichen das Abspielen von externen Programmen in einem WWW-Browserfenster. Ein Beispiel ist das Shockwave-Plug-In von Macromedia. Damit werden Programme, die mit dem Macromedia Director erstellt wurden, auch im WWW-Browserfenster lauffähig. Ein großer Nachteil von Plug-Ins ist, daß sie vor dem ersten Einsatz vom Netz heruntergeladen und installiert werden müssen.

ActiveX:

Microsoft bezeichnet die Internet-Erweiterungen seines *Component Object Model* (COM) mit ActiveX [SCHMITT 96]. ActiveX-Komponenten können recht komfortabel mit visuellen Programmiersprachen wie Visual Basic von Microsoft erstellt werden. Bereits vorhandene Komponenten können ohne großen Aufwand auch im WWW genutzt werden. ActiveX besitzt gleich mehrere große Nachteile. So sind ActiveX-Komponenten zur Zeit nur in Microsofts Internet Explorer auf der Windows-Plattform lauffähig. Von Plattformunabhängigkeit kann also keine Rede sein. Ein weiterer großer Nachteil von ActiveX besteht darin, daß durch fehlende Sicherheitsmechanismen nicht gewährleistet wird, daß keine Daten ausspioniert bzw. Schäden am Computer verursacht werden. Dadurch ist ActiveX für die Erstellung von medizinischen Lehr-/Lernsystemen vollkommen ungeeignet.

CGI:

Das Common Gateway Interface (CGI) ist eine standardisierte Schnittstelle von WWW-Servern. Diese Schnittstelle ermöglicht den Aufruf von externen Programmen auf dem WWW-Server. Dadurch kann die Funktionalität eines WWW-Servers beliebig erweitert werden. Hauptnachteil von CGI-Anwendungen ist, daß sie nicht sehr performant sind, da bei jeder Benutzeraktion eine Verbindung zum Server aufgebaut werden muß. Ihr großer Vorteil besteht darin, daß auf Server-Seite beliebige (plattformabhängige) Programmiersprachen verwendet werden können, ohne daß auf Client-Seite die Plattformunabhängigkeit verloren geht.

[2] JavaScript ist nicht objektorientiert. Es können nur die vordefinierten Objekte verwendet werden. Vererbung ist nicht möglich.

2.7 Adaptivität und Adaptierbarkeit

Unter Adaptivität in Zusammenhang mit CBT-Systemen wird die Anpassung eines Lehr-/Lernsystems an seine Nutzer verstanden. Adaptive Lehr-/Lernsysteme sind folglich Systeme, welche sich im Verlaufe eines Lehr-/Lernprozesses automatisch an die veränderlichen Lernbedürfnisse der Lernenden anpassen. Dabei ist anzunehmen, „...daß der Lernerfolg um so höher bzw. die erforderliche Lehrzeit um so geringer ist, je besser diese Anpassung gelingt.“ [LEUTNER 92 S. 1]. Das Ziel adaptiver Systeme ist es, die Systembenutzung für den einzelnen Nutzer zu erleichtern sowie die Produktivität, im Falle von medizinischen CBT-Systemen den Lernerfolg und die Zufriedenheit der Nutzer zu erhöhen [OPPERMANN (ed.) 94 S. 4].

Adaptive Systeme können in drei Gruppen eingeteilt werden [KROGSÆTER, THOMAS 94]:

- Adaptive Hilfesysteme: Die Hilfe wird entweder an den Aufgabenkontext oder an besondere Nutzervorlieben adaptiert.
- Adaptive Benutzungsschnittstellen: Die Mensch-Computer-Interaktion wird an die besonderen Bedürfnisse und Vorlieben eines Nutzers angepaßt.
- Adaptive Applikationen: Diese Systeme adaptieren ihre Funktionalität oder die interne Arbeitsweise in Abhängigkeit von den vorhergehenden Nutzeraktivitäten bzw. dem aktuellen Zustandsraum.

Bevor näher auf die Adaptionsmöglichkeiten von Lehr-/Lernsystemen eingegangen wird, soll eine Charakterisierung von guten Lehrern gegeben werden:

> „Good instructors are skilled at tailoring their behaviour to the particular skill level of students and to both the quality and quantity of student’s knowledge. This adaptation takes several forms, including:
> - varying the information content of descriptions and explanations;
> - varying the degree of detail of descriptions and explanations;
> - varying the aspects of the problem that they focus on during a given exercise;
> - gradually broadening and deepening the student’s knowledge in a way that builds upon and refines the student’s existing knowledge;
> - overlooking student mistakes that involves principles which the instructor judges the student to be unprepared to comprehend.“ [CHEIKES, RAGNEMALM 95 S. 95]

Ob Lehr-/Lernsysteme allerdings tatsächlich in der Lage sind, sich genauso gut wie ein kompetenter Lehrer an seine Schüler anzupassen, kann durchaus bezweifelt werden [vgl. YETIM 94 S. 34f.]. Auch Wenger [WENGER 97 S. 426] gibt zu, daß die Adaptivität selbst in den besten derzeit verfügbaren Lehr-/Lernsystemen im Vergleich zu menschlichen Lehrern eher bescheiden ist. Die Fähigkeit von Lehrern, Diagnose und Didaktik eng miteinander zu verbinden und zwischen dem bei den Lernenden bereits vorhandenen und dem zu lernenden Wissen hin und her zu springen, macht sie den Lehr-/Lernsystemen überlegen.

Nachfolgend soll beschrieben werden, wie Adaptivität bei CBT-Systemen charakterisiert werden kann und welche Adaptionsmaßnahmen bei Lehr-/Lernsystemen überhaupt realisierbar sind. Leutner wählt eine Klassifikation hinsichtlich dreier Facetten [LEUTNER 92]:

Adaptionsrate:

Die Adaptionsrate gibt die Häufigkeit der Anpassungsversuche des Lehrsystems an seine Benutzer an. Unter Mikro-Adaption wird meistens die Adaption innerhalb des laufenden Unterrichts verstanden, während bei der Makro-Adaption nur zu Beginn einer Unterrichtseinheit eine Adaption vorgenommen wird.

Art der Adaptionsmaßnahme:

Beim Lehren können Lehrziel (Lernstoff und zu erwerbende Kompetenz), Lehrmethode und die Lehrzeit adaptiert werden. Bildet man alle acht möglichen Kombinationen aus diesen drei elementaren Adaptionsmitteln so erhält man die nachfolgende Tabelle [LEUTNER 92 S. 10].

Adaptionsmittel			**Adaptionsmaßnahme**
Ziel	**Methode**	**Zeit**	**(Beispiel)**
fixiert	fixiert	fixiert	Fortschreitende Auslese
		variabel	Mastery Learning bezügl. Lehrziel
	angepaßt	fixiert	Innere (Unterr.-) Differenzierung
		variabel	Förderunterricht / Nachhilfeunterricht
angepaßt	fixiert	fixiert	Notengebung nach herkömml. Unterricht
		variabel	Mastery Learning bezügl. Note
	angepaßt (an Ziel)	fixiert	Äußere (Schul-) Differenzierung
		angepaßt (an Methode)	gegliedertes Schulsystem

Tabelle 1: Klassifikation von Adaptionsmaßnahmen

Wenn Ziel, Methode und die verfügbare Zeit fest vorgegeben sind, dann kann die Adaption nur darin bestehen, fortlaufend die Lernenden auszuschließen, welche die angestrebten Lernziele in der vorgegebenen Zeit nicht erreichen werden. Ist die Unterrichtszeit nicht begrenzt, so kann so lange unterrichtet werden, bis alle Schüler das Lernziel erreicht haben (*zielerreichendes Lehren* bzw. *Mastery Learning*). Wenn verschiedene Lehrmethoden verfügbar sind, dann kann bei begrenzter Unterrichtszeit die Lehrmethode verwendet werden, die den Bedürfnissen des Einzelnen am besten entspricht. Ist darüber hinaus auch eine längere Unterrichtszeit möglich, um Defizite zu beseitigen, dann spricht man von *Förder- bzw. Nachhilfeunterricht*. Ist nur eine einzige Lehrmethode verfügbar und kann das Lernziel für jeden Lernenden individuell angepaßt werden, dann handelt es sich um die Notengebung wie sie bei herkömmlichem Unterricht vorkommt. Bei unbegrenzter Unterrichtszeit kann so lange unterrichtet werden, bis das individuell festgesetzte Lehrziel (z.B. Note) erreicht ist.

Werden neben Lernziel auch die Lehrmethoden individuell an den Lernenden angepaßt spricht Leutner von *äußerer Differenzierung*. Kann die Unterrichtszeit in Abhängigkeit von der Lernumgebung verändert werden, wobei auch Lernziele und Lehrmethoden angepaßt werden können, ergibt sich eine Differenzierung, wie sie beispielsweise dem

deutschen Schulsystem mit unterschiedlichen Ausbildungszeiten je Schultyp zugrunde liegen.

Adaptionszweck:

Der Adaptionszweck beschreibt, wozu eine Adaptionsmaßnahme überhaupt eingesetzt werden soll. Salomon [SALOMON 72, zitiert in LEUTNER 92 S. 13] skizziert drei *heuristische Modelle*, um Interaktionshypothesen zu gewinnen. Beim Einsatz einer Adaptionsmaßnahme zur Beseitigung individuelle Defizite bei den Lernvoraussetzungen durch zusätzlichen Unterricht spricht Salomon von *Fördermodell*. Sind Lernvoraussetzungsdefizite zwar diagnostiziert, aber nicht ohne großen Aufwand zu beseitigen, dann besteht beim *Kompensationsmodell* der unmittelbare Zweck der Adaptionsmaßnahme im Geben von Lernhilfen (Kompensation). Davon profitieren allerdings nur diejenigen Lernenden, welche die Lernvoraussetzungsdefizite auch tatsächlich aufweisen. Lernende ohne diese Defizite können durch solche Maßnahmen in ihren Lernfortschritten behindert werden (Langeweile, Demotivation). Beim *Präferenzmodell* besteht der Zweck der Adaptionsmaßnahme darin, die individuellen Stärken eines Lernenden zu nutzen, um dessen Defizite zu kompensieren.

Ein Verfahren für die Adaption der Schwierigkeitsstufe von Übungsbeispielen an den aktuellen Kenntnisstand des Lernenden schlagen Lichtfeld, Driscoll und Dempsey [LICHTFELD, DRISCOLL, DEMPSEY 90, zitiert in LEUTNER 92 S. 53] vor:
Bearbeitet der Nutzer ein Übungsbeispiel korrekt, dann wählt das System bei der nächsten Übung ein Beispiel der nächsthöheren Schwierigkeitsstufe aus. Bearbeitet er die Aufgabe nicht korrekt, dann wird als nächstes ein Übungsbeispiel der nächstniedrigeren Schwierigkeitsstufe ausgewählt. Die Übungen innerhalb einer Schwierigkeitsstufe werden per Zufall ausgewählt. Bei einem Nachtest wurde festgestellt, daß dieses Verfahren gegenüber einer Strategie, bei der alle verfügbaren Übungen einer bestimmten Schwierigkeitsstufe bearbeitet werden müssen bevor die Übungen der nächsthöheren Stufe vorgelegt werden, keine signifikanten Wissensunterschiede ergab. Allerdings wurde signifikant weniger Übungszeit benötigt, weil wesentlich weniger Übungsbeispiele bearbeitet werden mußten.

Viele adaptive Systeme sind nicht nur adaptiv sondern auch adaptierbar. D.h., nicht nur das System paßt sich an seine Nutzer an, sondern diese können das System auch an ihre Wünsche anpassen, falls sie mit der automatischen Adaption nicht zufrieden sind. Oppermann und Simm unterscheiden zwischen Adaptierbarkeit der Funktionalität und der Adaptierbarkeit des User Interfaces [SCHNEIDER-HUFSCHMIDT, KÜHME et al. 93] bei einem System [OPPERMANN, SIMM 94]:

Adaptierbarkeit der Funktionalität:

- Funktionsumfang (nicht alle Funktionen sind jederzeit verfügbar)
- Benutzerdefinierte Kommandos (z.B. Makros)
- Add-Ins (Aufruf dann z.B. durch Menü)
- Default-Werte bei der Ausführung von Funktionen und von Objekt-Attributen
- Trigger: Bei vom Benutzer bestimmten Systemzuständen startet das System Funktionen

Adaptierbarkeit des User Interfaces:

- Zugang zur Funktionalität (einzelne Funktionen können ausgeblendet werden)
- Dialog-Verhalten (z.B. Unterdrücken von Dialogboxen, Einschalten der Hilfe wie *balloon help* beim Macintosh)
- Layout (z.B. Farbe von Fenstern usw.)
- Plattformübergreifende Interface-Adaptierungen (z.B. maximale Zeit für Doppelklick, vom Mauszeiger zurückgelegter Weg in Abhängigkeit des von der Maus zurückgelegten Weges)

Adaptierbarkeit ist ein Lösungsansatz um die Probleme zu bewältigen, die sich aus der unzureichenden Systemanpassung an das Benutzerverhalten ergeben [PAETAU 1990 S. 271]. Nutzer sollten deshalb die Adaption des Gesamtsystems oder einzelner Teile aus- und einschalten können. Anpassungen des Systems sollten zunächst als Vorschläge angeboten werden, welche angenommen oder abgelehnt werden können. Die Benutzer sollten bestimmte Parameter als adaptionsresistent definieren und Erklärungshilfen über die Auswirkungen einer Anpassungsänderung erhalten können, um vor Überraschungen bzw. vor Inkonsistenzen des Systems bewahrt zu werden. Schließlich sollten sie auch die volle Kontrolle über die weitere Verwendung der Protokolle ihrer Aktivitäten und deren Auswertung erhalten.

2.8 Intelligente Tutorielle Systeme

Der Begriff *Adaptivität* wird meist im Zusammenhang mit Intelligenten Tutoriellen Systemen (ITSen) [LUSTI 92; WINKELS, BREUKER 92] verwendet. ITSe sollen deshalb im Folgenden näher betrachtet werden.
In den 70er Jahren wurden Lehr-/Lernsysteme mit dem Anspruch entwickelt, Entscheidungsregeln, d.h. das entscheidungsrelevante Wissen selbst zu programmieren und nicht nur Entscheidungsergebnisse. Bei „nicht-intelligenten“ CBT-Systemen werden die auf Grundlage von fachlichem und didaktischem Wissen getroffenen Entscheidungen fest im System codiert. Dies ist ein wesentliches Unterscheidungsmerkmal zwischen ITSen und anderen CBT-Systemen [vgl. WENGER 87].

Sowohl an „nicht-intelligente“ CBT-Systemen als auch an Intelligente Tutorielle Systeme müssen allerdings auch gemeinsame Anforderungen gestellt werden [LARKIN, CHABAY (eds.) 92 S. 6f.]:

- Förderung des aktiven Lernens durch das Lehr-/Lernsystem: Lernende sollen nicht passiv Informationen aufnehmen, indem sie Texte am Bildschirm lesen oder Animationen beobachten, sondern sich vielmehr Lehr-/Lerninhalte aktiv erarbeiten. Dies kann dadurch geschehen, daß die Lernenden Fragen beantworten oder Aufgaben lösen müssen („Ask, don’t tell“).
- Realisierung angemessener Benutzungsschnittstellen: Die Benutzungsschnittstelle eines Lehr-/Lernsystems soll den zu vermittelnden Inhalten adäquat sein.
- Lernende sollen sich mit relevanten Aufgaben auseinandersetzen und diese angemessen lösen lernen.
- Bei falschen Benutzereingaben ist seitens des Systems ein Feedback erforderlich.
- Die individuellen Kenntnisse von Lernenden müssen vom System berücksichtigt werden.

2.8.1 Aufbau

Intelligente Tutorielle Systeme werden in der Literatur meist in die vier Bestandteile *Expertenmodul*, *Studentenmodul*, *Unterrichtsmodul* und *Kommunikationsmodul* eingeteilt [vgl. z.B. LUSTI 92]. Diese werden nachfolgend kurz charakterisiert.

2.8.1.1 Expertenmodul

Im Expertenmodul ist das Wissen eines menschlichen Experten in einem bestimmten Fachgebiet repräsentiert. Seine Funktion besteht darin, stets den optimalen nächsten Interaktionsschritt zu finden. Die nachfolgend aufgeführten Repräsentationsformen sind verbreitet:

Regeln: In Regeln ist Wissen in der Form *Bedingung(en)* $\rightarrow$ *Aktion(en)* dargestellt. Wenn alle Bedingungen einer Regel erfüllt sind, dann werden alle angegebenen Aktionen durchgeführt. Regeln sind einerseits relativ einfach zu implementieren, z.B. in Prolog, andererseits treten bei der Repräsentation von Wissen in Regeln aber auch Probleme auf [STRASSER 92 S. 21]:

- Mangelnde Flexibilität: Regeln werden für eine gegebene Problemklasse erstellt. Veränderte Problemklassen können nicht mehr gelöst werden, obwohl das dazu nötige Wissen in der Wissensbasis vorhanden wäre. Daraus folgt eine nur mangelhafte Wiederverwendbarkeit der Wissensbasen.
- Inadäquate Erklärungsfähigkeit**:** Erklärungen, welche anhand der Regel-Traces erstellt werden, sind oftmals nicht adäquat.
- Probleme der Wissensakquisitation: Es ist sehr schwierig, ein robustes und vielseitig verwendbares Regelsystem zu erstellen. Dies liegt u.a. auch darin, daß sich Fachexperten oftmals gar nicht mehr bewußt sind, welches Wissen sie für eine konkrete Problemlösung verwendet haben.
- Implizite Kontrollstrukturen: Domänenwissen und Kontrollstruktur sind oftmals untrennbar miteinander verbunden. Darunter leidet die Klarheit und eine Wiederverwendbarkeit der Wissensbasis bei anderen Systemen und ähnlichen Problemstellungen ist kaum möglich. In Regeln ist oftmals „oberflächliches“ Wissen (shallow knowledge) kodiert. Der Übergang von *shallow knowledge* zu *deep knowledge* (Tiefenwissen) gelingt, indem die in der Domäne inhärenten Modelle explizit in der Wissensbasis beschrieben werden. Bei Verwendung von shallow knowledge in einer Wissensbasis sind zwar Reaktionen des Systems bei bestimmten Situationen möglich, allerdings scheitert ein solches System oftmals bereits bei ähnlichen Situationen, auch wenn das zur Problemlösung notwendige Wissen durchaus in impliziter Form in der Wissensbasis vorhanden ist.

Semantische Netze: In semantischen Netzen ist das Wissen in Form eines Graphen dargestellt, welcher aus Knoten und Kanten besteht. Die Knoten sind miteinander durch Kanten verbunden. Kanten können Beschriftungen tragen und dadurch typisiert werden. Als Beispiel soll KL-ONE aufgeführt werden. KL-ONE ist eine auf semantischen Netzen aufbauende Wissensrepräsentationssprache. Objekte werden hier *Konzepte* genannt, deren Eigenschaften *Rollen*. Kernidee von KL-ONE ist die strikte Trennung von definierenden und optionalen Rollen und die Unterscheidung zwischen generischen Konzepten und Instanzen.

Constraints: Mit Hilfe von Constraints können Beziehungen zwischen Quantitäten mittels mathematischer (Un-)Gleichungen spezifiziert werden. Constraints sind Bedingungen, welche von der gesuchten Lösung erfüllt werden müssen. Constraints können nur in Domänen eingesetzt werden, die numerischen Charakter haben.

Frames: Das Frame-Konzept stammt von Minsky [MINSKY 75]. Frames stellen Analogien zum menschlichen Erfahrungswissen dar. *Frames* sind Objekte, die durch *Slots* (Eigenschaften der Frames) und *Facetten* (Eigenschaften der Slots) beschrieben werden. Es gibt Facetten für den Default-Wert, den tatsächlichen Wert, den Wertebereich usw.

2.8.1.2 Studentenmodul

> „Intelligent tutoring systems can be individualized if they are designed to take into account differences between students. The process of doing this is called student modelling. Unfortunately, student modelling is hard, and increasingly researchers are trying to avoid the need.“ [MCCALLA 92 S. 107]

Das Studentenmodul überwacht das Problemlöseverhalten eines Studenten und stellt Veränderungen fest. Hauptbestandteil vieler Studentenmodule ist die Fehlerdiagnose. Durch Vergleich der Lösungen von Expertenmodul und Lernendem wird herauszufinden versucht, ob es sich um einen zufälligen oder einen systematischen Fehler handelt. Trifft letzteres zu, so wird versucht, dem Lernenden gezielt Hilfestellungen und Erklärungen zu geben.
Die wichtigsten Studentenmodelle sind Stereotypen, Überlagerungsmodelle, Theorien von Fehlern, Wiederherstellung von Fehlern, Erzeugung von Fehlern und eine Kombination der vorangegangenen Modelle [REINHARDT, SCHEWE 95 S. 84]
Stereotypen [RICH 89] sind Mengen von Merkmalsausprägungen, die oft gemeinsam bei Personen auftreten. Sie sind wichtig, weil sie es ermöglichen, viele plausible Annahmen mit relativ wenigen Beobachtungen zu treffen. Allerdings muß es möglich sein, diese Annahmen (inferences) durch tatsächliche Beobachtungen zu überschreiben. Deshalb ist es erforderlich, Techniken für nichtmonotones Schließen zu verwenden. Typische Stereotypen sind z.B. Anfänger, Fortgeschrittener oder Experte. Beim Zuordnungsmodell [BODENDORF 90 S. 132] versucht das Lehr-/Lernsystem während der Interaktion mit dem Nutzer diesem Eigenschaften zuzuordnen. Dabei kann entweder aus Teilmengen von sich gegenseitig ausschließenden Merkmalen jeweils genau ein Element ausgewählt werden, oder aber es kann eine Menge von Eigenschaften auf einmal ausgewählt werden (Stereotypenansatz).
Das Überlagerungsmodell (overlay model) sieht das Studentenwissen als Teilmenge des im System enthaltenen Expertenwissens. Da Überlagerungsmodelle kein über das Expertenmodell hinausgehendes Wissen benötigen, sind sie verhältnismäßig leicht zu implementieren. Bei „flachem Wissen“ im Expertenmodell müssen einfach die dem Studenten bekannten Konzepte abgehakt werden. Hier können auch Wahrscheinlichkeiten vergeben werden, die angeben, wie „sicher“ das System ist, daß ein Konzept tatsächlich vom Nutzer verstanden wurde. Schwieriger wird es allerdings, wenn das Expertenmodell Tiefenwissen enthält, wie beispielsweise das NEOMYCIN Expertenmodell für den GUIDON2-Tutor [KASS 89 S. 391]. Ein Problem bei Überlagerungsmodellen ergibt sich auch dann, wenn das Studentenwissen vom Expertenwissen abweicht. Das System kann bei Überlagerungsmodellen lediglich feststellen, daß ein Student eine andere Strategie verwendet, als im Expertenmodell abgelegt. Allerdings ist es nicht in der Lage zu beurteilen, ob die vom

Studenten verwendete Strategie nicht ebenfalls zur Problemlösung geeignet ist oder warum sie ungeeignet ist.
Perturbation models (engl. *to perturbate* = stören, beunruhigen) dienen dazu, die Sicht des Studenten auf das Expertenmodell abzubilden, und dabei eine enge Beziehung zwischen Studenten- und Expertenmodell zu bewahren. Das Studentenmodell unterscheidet sich vom Expertenmodell durch enthaltene Fehler und fehlendes Wissen, was ebenfalls als im Modell enthaltene Fehler angesehen wird. Viele ITSe verwenden diese Art von Modell. Schwierigkeiten ergeben sich dadurch, daß bei Fehlern des Studenten nicht klar ist, ob dafür fehlendes oder falsches Wissen verantwortlich ist [KAAS 89].

Laut Woolf [WOOLF 90] sind Studentenmodelle in den existierenden Systemen noch nicht effektiv integriert. Das Hauptproblem bei der Studentenmodellierung besteht darin, die Charakteristik eines Nutzers lediglich aus dessen Interaktion mit dem System zu erschließen. Häufig werden in den bisher existierenden Prototypen diejenigen Parameter ermittelt, bei denen dies am einfachsten geht [KROGSÆTER, THOMAS 94]. Ein weiteres Problem besteht darin, die ermittelten Nutzercharakteristika auf angemessene Antworten an den Nutzer zu mappen.
Man darf an Studentenmodelle sicherlich keine überzogenen Erwartungen stellen. Self, einer der bekanntesten Forscher auf diesem Fachgebiet, empfiehlt für die Erstellung eines Studentenmodells, sich von Visionen zu verabschieden, realistische Ziele zu wählen und die folgenden Punkte zu beachten [SELF 90]:

- „Avoid guessing - get the student to tell you what you need to know“
- „Don’t diagnose what you can’t treat“
- „Empathize with student’s beliefs, don’t label them as bugs“
- „Don’t feign omniscience - adopt a ‘fallible collaborator’ role“

Bei der Verwendung von Studentenmodellen haben sich zwei Sichtweisen gebildet. Die eine Seite empfiehlt den zurückhaltenden Einsatz von Informationen über die Nutzer eines Systems [BIERMAN, KAMSTEEG et al. 92] da es schwierig ist, vollständige und korrekte Studentenmodelle aufzubauen und diese auch noch korrekt zu verwenden. Durch eine zurückhaltende Verwendung von Informationen über den Nutzer kann die Wahrscheinlichkeit für Fehler des Lehr-/Lernsystems, z.B. die Präsentation ungeeigneter Informationen, verringert werden. Das andere Lager empfiehlt eine intensive Nutzung von Studentenmodellen. Mögliche Fehlannahmen eines CBT-Systems können diesem Standpunkt nach während des Programmlaufs korrigiert werden, wenn das System diese beispielsweise durch Nachfragen des Nutzers erkennt [YETIM 94 S. 52].

2.8.1.3 Unterrichtsmodul

Das Unterrichtsmodul ist unter anderem dafür zuständig, die jeweils geeignetste Lehr-/Lernform zu wählen und den Tutanden bei Bedarf zu unterbrechen.
In DOMINIE (Domain Independent Instructional Environment) [SPENSLEY, ELSOM-COOK et al. 90] wurden die folgenden Lehr- und Beurteilungsstrategien implementiert:

- Cognitive Apprenticeship: Der Lernende schaut hier einem Experten bei der Lösung von Aufgaben zu und kann Fragen stellen. Nach und nach erlaubt der Experte dem Lernenden, kleinere Teilaufgaben selbst zu lösen. Diese Teilaufgaben werden immer komplexer, bis der Lernende schließlich in der Lage ist, eine vollständige Aufgabe erfolgreich zu lösen.

- Successive Refinement: Diese Lehrstrategie ist besonders bei komplexen Wissensdomänen geeignet. Zuerst wird ein allgemeiner Überblick vermittelt, der dann schrittweise immer weiter verfeinert wird.
- Discovery Learning: Ein Lehrer wählt hier eine Lernumgebung, die dem Vorwissen des Lernenden angepaßt ist. Dies ist eine sehr anspruchsvolle Aufgabe. DOMINIE kann lediglich einen Wissensbereich auswählen, auf dem der Lernende Neuling ist, der aber strukturelle Ähnlichkeiten mit einem Bereich hat, den der Student bereits gelernt hat.
- Discovery Assessment: Diese Strategie ist mit entdeckendem Lernen (discovery learning) gekoppelt. Nach Auswahl einer geeigneten Lernumgebung ist es erforderlich, den Studenten zu beobachten und gegebenenfalls Hilfestellung zu geben.
- Abstraction: Der Lernende soll einen Überblick für die Möglichkeiten im System vermittelt bekommen.
- Socratic Diagnosis: Der Lernende hat bereits ein Modell der Wissensdomäne aufgebaut, das allerdings Fehler enthält. Der Lehrer ermittelt diese Fehler und macht dem Lernenden deutlich, daß sein Modell falsche Schlußfolgerungen erzeugt und deshalb fehlerhaft sein muß. Durch einen Dialog soll es dem Lernenden ermöglicht werden, die Fehler selbst zu erkennen und zu beseitigen. In DOMINIE werden die Fehler im Modell des Lernenden durch Befragung ermittelt.
- Practice: Der Lernende wird bei der Lösung einer Aufgabe beobachtet. Weicht er vom „richtigen“ Lösungsweg ab, dann bekommt er Anregungen vom Tutor.
- Direct Assessment: Der Lernende wird direkt befragt. Besonders geeignet sind Multiple-Choice-Fragen und Lückentexte. Problematisch dagegen sind natürlichsprachliche Eingaben.

DOMINIE entscheidet sich jeweils für eine der acht beschriebenen Strategien anhand verschiedener Kriterien. Das System versucht stets eine sinnvolle Aufteilung der zur Verfügung stehenden Zeit zwischen Unterricht und Prüfung zu erreichen. Einerseits soll in möglichst kurzer Zeit möglichst viel Stoff erarbeitet werden, andererseits muß aber auch gewährleistet sein, daß der Lernende den bisher präsentierten Stoff richtig verstanden und sich gemerkt hat. Die jeweils angewandte Lehrstrategie ist dem zu unterrichtenden Fach angemessen und der bisherige Lernerfolg eines Studenten bei verschiedenen Lehrstrategien wird vom System berücksichtigt. Vorlieben eines Studenten werden ebenfalls bei der Wahl geeigneter Lehr- und Beurteilungsstrategien berücksichtigt.

Insgesamt gibt es drei Möglichkeiten, wie man zu einem Unterrichtsmodell gelangen kann [LEUTNER 92 S. 66]:

- Intuition des Programmierers: Die Entwickler eines CBT-Systems verlassen sich auf ihre Intuition und greifen auf eigene Erfahrungen mit Lehr-/Lernsituationen, beispielsweise in ihrer Schulzeit, zurück. Diese Art, zu einem Unterrichtsmodell zu gelangen, ist am weitesten verbreitet, da sie mit dem geringsten Aufwand verbunden ist.
- Befragung von erfahrenen Lehrern: Durchführung einer Befragung, vergleichbar der Befragung und Beobachtung von Experten bei der Erstellung von Wissensbasierten Systemen und Expertensystemen [SPITZER, BÜRSNER 95; CHIZZALI-BONFADIN, ADLASSNIG et al. 97; PUPPE 93]. Dabei ist es leicht möglich, daß die eigene Intuition des CBT-System-Entwicklers lediglich durch die Intuition eines Lehrers ersetzt wird.
- Instruktionstheoretische Theoriebildung: Wahlweise wird eine vorhandene Lerntheorie zu einer Instruktionstheorie erweitert oder eine Instruktionstheorie neu entwickelt.

2.8.1.4 Kommunikationsmodul

Im Kommunikationsmodul ist festgelegt, in welcher Form das System mit den Tutanden kommuniziert.
Ideal ist eine Benutzungsschnittstelle mit freier Eingabe. Die freie Eingabe ist allerdings sehr aufwendig zu implementieren. So erforderte alleine die Implementierung einer natürlichsprachlichen Benutzungsschnittstelle in Sophie I zwei Personenjahre [LUSTI 92 S. 227]. Als leichter zu implementierende Alternativen können dem Nutzer auch Alternativantworten oder Auswahlmenüs angeboten werden. Im Gegensatz zur freien Eingabe haben die Nutzer hier allerdings die Möglichkeit, durch „intelligentes Ausschließen" offensichtlich falsche Antworten auszugrenzen. Dagegen ist die Implementierung eines Nutzerdialogs über Alternativantworten oder Auswahlmenüs wesentlich einfacher.

2.8.1.5 Entwicklungsunterstützung

Die sehr arbeitsintensive Entwicklung von Intelligenten Tutoriellen Systemen kann auf zwei Arten vereinfacht und beschleunigt werden [LUSTI 92 S. 197]. Entwickler können gegenstandsunabhängige Module eines existierenden Lehr-/Lernsystems wie Unterrichtsmodul oder Kommunikationsmodul wiederverwenden und müssen lediglich das Expertenmodul an die veränderte Aufgabenstellung anpassen. Die andere Möglichkeit besteht in der Verwendung einer ITS-Shell [PUPPE, GAPPA et al. 96]. Der Vorteil dieses Ansatzes liegt darin, daß von der Shell automatisch die notwendigen Inferenzmechanismen bereitgestellt werden und nicht selbst implementiert werden müssen. Mit ITS-Shells können Fachgebietsexperten, auch wenn sie keine Informatikkenntnisse aufweisen, in die Lage versetzt werden, Intelligente Tutorielle Systeme zu erstellen.

2.8.2 Kritik an Intelligenten Tutoriellen Systemen

Schulmeister übt in seinem Buch „Grundlagen hypermedialer Lernsysteme" [SCHULMEISTER 96] harte Kritik an Intelligenten Tutoriellen Systemen und deren Forschern und Entwicklern:

> „Das Gebiet der Intelligenten Tutoriellen Systeme ist offenbar in der Phase der Programmatik steckengeblieben: Selbst wenn man nur die Aufsätze einer einzigen Zeitschrift zu dem Thema ITS betrachtet, so wiederholen sich ständig dieselben Aussagen, Systematisierungen und Rückgriffe auf die AI-Literatur. Ich habe selten so viel Redundanz auf einem Haufen gesehen, und selten eine so kleine Gemeinschaft von Forschern, die ständig dasselbe auf demselben Entwicklungsstand veröffentlichen (Anderson, Brown, Duchastel, Jonassen, Lesgold, O'Shea, Self, Sleeman). Selbst dann, wenn ein Programm »an Tausenden von Schülern ausprobiert wurde«, kann man davon ausgehen, daß das Programm nicht, wie Clancey formuliert, »im praktischen Gebrauch« war, sondern daß es in flächenartigen Versuchsserien mit Schülern als »Versuchspersonen« getestet wurde, z.B. DEBUGGY [Burton (1982)]." [SCHULMEISTER 96 S. 188]

Cheikes nennt die folgenden Probleme von Intelligenten Tutoriellen Systemen [CHEIKES 95 S. 1 und 2]:

- Jedes System wird komplett neu entwickelt, ohne alte Komponenten zu nutzen.
- Modularität (zwecks Wiederverwendung) ist bei ITSen nicht sehr verbreitet, im Gegensatz zum kommerziellen Softwarebereich.
- Die Entwicklung wird weithin als iterativer Verbesserungsprozess verstanden.
- Der Einbau von bereits erhältlichen Softwarepaketen wird dadurch verhindert, daß diese nur eine beschränkte oder überhaupt keine externe Softwareschnittstelle anbieten.
- Die Evaluation wird behindert, weil keine vollständigen Systeme zur Verfügung stehen.

Lusti nennt Implementationsferne, Praxisferne und Unbescheidenheit bei der Entwicklung von ITSen als wichtige Probleme [LUSTI 92]. Ein zentrales Problem ist außerdem der mit der Erstellung von Intelligenten Tutoriellen Systemen verbundene hohe Entwicklungsaufwand und die hohen Kosten. Trotzdem lassen sich, obwohl in vielen ITS-Projekten ein sehr hoher Aufwand getrieben wurde und wird, nur relativ einfache Lehrgegenstände abbilden. Häufig unterrichten die ITSe kleine Teilgebiete der Mathematik oder Programmiersprachen. Wenn wie bei JA-Tutor ein komplexeres Gebiet abgedeckt wird, dann sind die Entwickler gezwungen, sich vom Ideal eines Intelligenten Tutoriellen Systems zu entfernen.
Ebenfalls noch ungeklärt sind im Zusammenhang mit ITSen sogenannte *ATI-Effekte* (*Aptitude-Treatment-Interactions*). Darunter versteht man den Einfluß individueller Lernvoraussetzungen auf den Lernerfolg [LEUTNER 92], beispielsweise wie die Reaktion auf verschiedene Lernmedien von Eigenschaften der Studenten wie Intelligenz oder Ängstlichkeit abhängt.
Bisher konnte auch kaum für ITSe bzw. tutoriell nutzbare Expertensysteme der tutorielle Nutzen nachgewiesen werden [ENGLER, FÜHRER et al. 95]. Als Gründe hierfür werden die Komplexität der Systeme und langatmige Dialoge vermutet.

2.8.3 D3 und TRAINER

Als Beispiel für Intelligente Tutorielle Systeme soll D3 [PUPPE, GAPPA et al. 96] und die darauf basierende ITS-Shell TRAINER [PUPPE, REINHARDT 95] dienen.
D3 ist eine Expertensystemshell, die von einer Arbeitsgruppe an der Universität Würzburg unter Leitung von Prof. Puppe entwickelt wurde. Sie unterstützt verschiedene Formen der Wissensrepräsentation und Wissensverarbeitung [REINHARDT, SCHEWE 95]:

- Statistische Diagnostik: Verwendung des Satzes von Bayes. Die benötigten Symptom-Diagnosewahrscheinlichkeiten können aus einer Falldatenbank ermittelt werden.
- Fallbasierte Diagnostik: Suche in Datenbank nach ähnlichen Fällen und Übertragung derer Lösung auf den eigenen Fall.
- Heuristische Diagnostik: Symptom-Diagnose Verkettungen, üblicherweise mit Sicherheitsfaktoren.
- „set covering“-Diagnostik: Basiert auf der Überdeckung oder kausalen Beziehungen von Diagnosen zu Symptomen.
- Funktionale Diagnostik: Modellbasierte Diagnostik. Im Modell ist das Verhalten der einzelnen Komponenten beschrieben.

Zur Zeit sind bereits einige medizinische Wissensbasen für D3 verfügbar, unter anderem für Rheumatologie und Neurologie. Diese sind Voraussetzung dafür, um mit TRAINER Fallsimulationen erstellen zu können.
Studenten können mit dem TRAINER lernen, medizinische Lehr-/Lernfälle zu lösen. Das System bietet hierzu zwei Lernmodi an. Im Einen präsentiert das System die von einem Experten in verschiedene Gruppen eingeteilte Untersuchungen (Anamnese, klinische Untersuchung, Labor usw.) in sequentieller Reihenfolge und der Student muß nach jeder Gruppe eine Verdachtsdiagnose aufstellen. Im anderen Modus werden einige Symptome genannt und der Student muß entscheiden, welche weiteren Untersuchungen er durchführen will. Dabei kann das System die vom Studenten getroffenen Entscheidungen kritisieren. TRAINER besitzt allerdings noch kein Studentenmodell. Daran wird in Würzburg gearbeitet.
Die häufigste Kritik an D3 betrifft dessen Benutzungsschnittstelle. Diese wird häufig als zu umständlich und zu wenig intuitiv in der Bedienung kritisiert.

2.9 Semantische Datenmodellierung

Bereits 1971 rief das X3 Komitee der ANSI (American National Standards Institute) eine Arbeitsgruppe mit Namen SPARC (Standards Planing and Requirements Committee) ins Leben. Ziel der Arbeitsgruppe war es zu eruieren, ob und wo Standards bei Datenbanksystemen definiert werden können. Nach längerer Zeit legte die Arbeitsgruppe einen Zwischenbericht vor, in der eine Drei-Schema-Architektur eines DBMS beschrieben wird. Der ANSI/X3/SPARC-Vorschlag besteht aus dem *externen Schema* (Sicht des Benutzers bzw. Programmierers auf die Daten), dem *konzeptuellen Schema* (Unternehmenssicht der Daten, Beschreibung der realen Welt durch Entitäten, Beziehungen usw.) und dem *internen Schema* (physikalische Datenorganisation auf den Speichermedien). Durch das konzeptuelle Schema soll eine stabile Sicht auf die Daten gewährleistet werden.
Es ist vollkommen unabhängig von konkreten logischen Datenmodellen wie beispielswei-

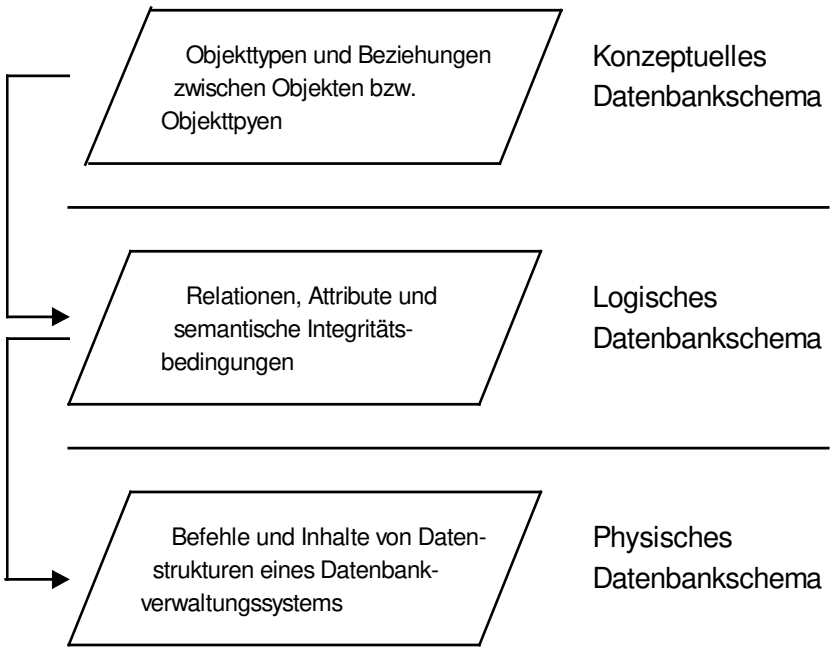

Abbildung 7: Die drei Ebenen der Datenmodellierung

se dem relationalen Datenmodell (siehe Kapitel 2.10.1) und kann gut als Grundlage für die Kommunikation z.B. zwischen Domänenexperten und Entwicklern verwendet werden. Das „implementierte“ konzeptuelle Schema wird häufig in der Literatur als *logisches Schema* bezeichnet [SCHLAGETER, STUCKY 83]. Das logische Schema berücksichtigt das zum Einsatz kommende Datenmodell, es ist aber genauso wie das konzeptuelle Schema vollkommen unabhängig von der verwendeten Hard- und Software.
Mann [MANN 93 S. 9] beschreibt in Anlehnung an [SCHLAGETER, STUCKY 83] drei Ebenen bei der *objektorientierten semantischen Datenmodellierung* (siehe Abbildung 7). Das *konzeptuelle Datenbankschema* stellt ein Modell des betrachteten Realitätsausschnittes dar. Dort werden alle relevanten Objekttypen und Beziehungen zwischen diesen dargestellt. Das *logische Datenbankschema* berücksichtigt zusätzlich das konkret verwendete Datenmodell, im unserem Falle das relationale Datenmodell. Objekttypen und Beziehungen werden auf dieser Ebene durch Relationen, Attribute und semantische Integritätsbedingungen abgebildet. Das *physische Datenbankschema* schließlich enthält die durch Umsetzung des logischen Datenbankschemas erhaltenen Befehle und Inhalte von Datenstrukturen eines konkreten Datenbankmanagementsystems.
Im weiteren Verlauf dieser Arbeit orientiert sich die objektorientierte, semantische Datenmodellierung an diesem 3-Ebenen-Modell.
Die konzeptuelle Datenmodellierung eignet sich sehr gut für die objektorientierte Softwareentwicklung. Sie ermöglicht *real world modeling*, d.h. Objekte können so dargestellt werden, wie wir sie in unserer Umgebung wahrnehmen. Das konzeptuelle Modell kann mit Hilfe eines ER-Modells (Entity-Relationship) [CHEN 76] dargestellt werden. Vom ER-Modell gibt es heute eine Reihe unterschiedlicher Varianten, die sich teilweise lediglich in der Art der Darstellung unterscheiden, teilweise aber auch in ihrer Ausdrucksfähigkeit. Ein Standard existiert bisher nicht.

Ein ER-Schema besteht aus 5 Bestandteilen [RAUH, STICKEL 97 S. 37]:

1. Genau einer Schemabezeichnung
2. Mindestens einer Deklaration von Objektarten
3. Null oder mehr Deklarationen von Beziehungsarten
4. Null oder mehr Integritätsregeln
5. Null oder mehr Ableitungsregeln

Nicht alle Bestandteile können mit graphischen Mitteln dargestellt werden. Deshalb wird das gesamte Schema mittels Text definiert und durch Diagramme verdeutlicht.
Die Schemadefinition in der erweiterten Backus-Naur-Form (EBNF) [VENTER] ist in nachfolgender Tabelle enthalten. Die in [RAUH, STICKEL 97] vorgestellte Definition wurde um Aggregationsbeziehungen erweitert.

```
<schema>                  ::=   schema <schemaname> ;
                                <entity set> { ; <entity set>}
                                {;<relationship set>}
                                {;<integrity rule>}.
<schemaname>              ::=   <name>
<entity set>              ::=   entityset <entity set name>
                                '('attributes: <attribute>
                                {, <attribute>} ;
                                identifier: <identifier>
                                {, <identifier>}')'
                          |     entityset <entity set name>
                                '('<subset declaration>
                                [; attributes: <attribute>{, <attribute>}
                                [; identifier: <identifier>
                                {, <identifier>}]]')'
<subset declaration>      ::=   subsetof <entity set name>
<entity set name>         ::=   <name>
<relationship set>        ::=   relationship <relationship set name>
                                '('participants: <participant>,
                                <participant> {, <participant>}')'
                          |     wholepartassociation <relationship set name>
                                '('participants: <wholeparticipant>,
                                <partparticipant> ')'
<attribute>               ::=   <attribute name> <domain> [not null]
<identifier>              ::=   <attribute name>
<attribute name>          ::=   <name>
<relationship set name>   ::=   <name>
<wholeparticipant>        ::=   <participant>
<partparticipant>         ::=   <participant>
<participant>             ::=   '('<entity set name> [, <role>] , <cardinality>')'
<role>                    ::=   <name>
<cardinality>             ::=   '('<mincard>,<maxcard>')'
<mincard>                 ::=   <nonneg_integer>
<maxcard>                 ::=   <nonneg_integer> | n
<domain>                  ::=   char'('<nonneg_integer>')' | integer | real | date |
                          |     ...
<nonneg_integer>          ::=   0 | 1 | ...
<name>                    ::=   (<letter> | <digit> | _) {<letter> | <digit> | _}
<letter>                  ::=   a | b | ... | z | A | B | ... | Z
<digit>                   ::=   0 | 1 | 2 | ... | 9
```

Abbildung 8: Syntax zur ER-Schema-Deklaration

Da eine Grammatik in EBNF nicht alle inkorrekten Formulierungen verhindern kann, müssen noch einige Zusatzbedingungen hinzugefügt werden [RAUH, STICKEL 97 S. 45]:

1. Alle Objekt- und Beziehungsartennamen (<entity set name>,<relationship name>) dürfen im Schema nur einmal verwendet werden.
2. Attributnamen (<attribute name>) dürfen innerhalb einer Objekt- bzw. Beziehungsart nur einmal verwendet werden.

3. Alle innerhalb einer Partizipantendeklaration (<participant>) einer Beziehungsart genannten Objektartnamen müssen auch als Objektartname (<entity set name>) ein einer Objektartendeklaration (<entity set>) vorkommen.
4. Die Rollennamen (<role>) dürfen bei der Deklaration rekursiver Beziehungsarten nicht weggelassen werden.
5. Wenn in einer Partizipantendeklaration eine Objektart mehrmals vorkommt, dann muß für jedes Vorkommen ein anderer Rollenname gewählt werden.
6. Für die Kardinalitätsangaben (<cardinality>) gilt: Die Untergrenze (<mincard>) muß stets kleiner oder gleich der Obergrenze <maxcard> sein. Ist die Obergrenze n, so kann die Untergrenze jede beliebige Ganzzahl größer oder gleich null einnehmen.

Für die hinzugefügte Aggregationsbeziehung wird folgende Zusatzbedingung definiert: <wholeparticipant> enthält <partparticipant>, aber nicht umgekehrt.

Für die Formulierung von Integritätsregeln in einem ER-Schema soll der leicht abgewandelte Entity-Relationship Calculus (ERC) [RAUH, STICKEL 97] verwendet werden. Nachfolgend seine Beschreibung:

<integrity rule>	::=	<quantification list> (<condition>)
<condition>	::=	<condition> $\leftrightarrow$ <t1>
	|	<t1>
<t1>	::=	<t1> $\rightarrow$ <t2>
	|	<t2>
<t2>	::=	<t2> $\vee$ <t3>
	|	<t3>
<quantification list>	::=	<quantification list> <quantification>
	|	<quantification>
<quantification>	::=	(<quantifier> <variable list>)
<variable list>	::=	<variable list>, <name>
	|	<name>
<quantifier>	::=	$\exists$
<comparison>	::=	<value> <op1> <value>
	|	<entity variable> <op2> <entity variable>
	|	<relationship variable> <op2> <relationship variable>
<value>	::=	null
	|	<text constant>
	|	<attribute variable>
	|	<value expression>
<text constant>	::=	<name>
<attribute variable>	::=	<construct variable> '[' <attribute name> ']'
<construct variable>	::=	<entity variable>
	|	<relationship variable>

<entity variable>	::=	<name>
	|	<relationship variable>: <entity set name>
	|	<relationship variable>: <role>
<relationship variable>	::=	<name>
<value expression>	::=	<value expression> + <a1>
	|	<value expression> - <a1>
	|	<a1>
<a1>	::=	<a1> * <a2>
		<a1> / <a2>
		<a2>
<a2>	::=	(<value expression>)
	|	<attribute variable>
	|	<numerical constant>
	|	<function>(<argument list>)
	|	<aggr function expression>
<numerical constant>	::=	<digit> {<digit>}
<function>	::=	abs | div | mod | max | min
	|	powerof | year | month | day
	|	datediff
<argument list>	::=	<argument list>, <value expression>
	|	<value expression>
<aggr function expression>	::=	<aggr function> <set query>
<aggr function>	::=	min | max | avg | count | sum
<condition>	::=	<condition> ∨ <t3>
	|	<t3>
<t3>	::=	<t3> ∧ <t4>
	|	<t4>
<t4>	::=	¬ <t4>
	|	<t5>
<t5>	::=	<quantification list> (<condition>)
	|	<comparison>
	|	<predicate>
	|	(<condition>)
<set query>	::=	'{'<value list>'|'<condition>'}'
<value list>	::=	<value list>, <value list element>
	|	<value list element>
<value list element>	::=	<value>
		<construct variable> '[' * ']'
<op1>	::=	'=' | '<' | '>' | ≤ | ≥ | ≠
<op2>	::=	'=='
<predicate>	::=	<entity set name> (<entity variable>)
	|	<relationship set name> (<relationship variable>)

Abbildung 9: Entity-Relationship Calculus

Bezüglich der Integritätsregeln wird vereinbart, daß auf eine explizite Formulierung der folgenden Anforderungen verzichtet wird. Sie müssen stets erfüllt sein.

- Werte von Attributen folgen eindeutig aus dem Wert des Identifikators.
- Identifikatoren sind niemals unbestimmt (null).
- Die in der Schemastruktur angegebenen Kardinalitäten der Beziehungsarten sind als Integritätsregeln zu betrachten, die erfüllt sein müssen.

Wie bereits oben erwähnt, existiert kein Standard für das ER-Modell. Die Art der graphischen Darstellung kann also frei gewählt werden. In der Arbeit werden die aus der objektorientierten Programmierung bekannten Klassendiagramme der Unified Modeling Language [BURKHARDT 97] verwendet. Diese graphische Darstellung wurde aus mehreren Gründen gewählt. Zum einen bieten sie die Möglichkeit, mit den verfügbaren Beziehungstypen (Vererbung, Assoziation, Aggregation) aussagekräftige Modelle zu erstellen und zum anderen existieren für die Erstellung solcher Diagramme leistungsfähige Werkzeuge.
Das System soll objektorientiert realisiert werden. Ein großer Vorteil ist deshalb, daß viele Konstrukte im ER-Modell einen engen Bezug zu den Klassen besitzen, welche für die objektorientierte Realisierung benötigt werden. Aus dem entworfenen ER-Modell können deshalb teilweise automatisch die erforderlichen Klassen erzeugt werden.
Ein wesentlicher Unterschied zwischen Entity-Relationship-Modellierung (ERM) und der objektorientierten Analyse besteht darin, daß ERM-Objekte lediglich statische Aspekte wie Attribute und Beziehungen darstellen können, während Klassen auch Methoden enthalten können. Dadurch können letztere auch hinsichtlich ihres Verhaltens betrachtet werden. ERM ersetzt also keinesfalls die objektorientierte Analyse. Vielmehr kann durch ERM die Analyse unterstützt werden und durch Verwendung der geeigneten Notationen und Werkzeuge kann der Implementierungsaufwand reduziert werden.

2.10 Datenbanken

In Lehr-/Lernsystem vorhandene Daten und vorhandenes Wissen muß permanent gespeichert werden. In Frage kommen hierfür momentan vor allem relationale oder alternativ hierzu objektorientierte bzw. objektrelationale Datenbanksysteme. Sie werden in diesem Kapitel beschrieben.

2.10.1 Relationales Datenmodell

Nachfolgend soll das Relationenmodell kurz charakterisiert werden [vgl. GERNETH 92, MANN 93]. Für eine detailliertere Beschreibung wird auf die in großen Umfang verfügbare Literatur verwiesen [ULLMAN 88; SCHLAGETER, STUCKY 83; GARDARIN, VALDURIEZ 89].

Attribut

Eine Variable *A* vom Typ Attribut sei definiert durch

$$A\text{: } \mathbf{ATT}(\text{DTYP}|\underline{\mathbf{sib}}^0)$$

Dabei bezeichnet DTYP ein beliebiger Datentyp (`real`, `string`, ...), dessen Wertebereich durch **WB**(DTYP) bezeichnet wird. **WB**(DTYP) beinhaltet grundsätzlich auch den fehlenden Wert ω.

Der Wertebereich von A wird eingeschränkt durch die Menge semantischer Integritätsbedingungen $\underline{\mathbf{sib}}^0$:

$$\mathbf{WB}(A) := \{x \in \mathbf{WB}(\text{DTYP}) \mid \forall\ \mathbf{sib} \in \underline{\mathbf{sib}}^0 : \mathbf{sib}(x) = \text{'wahr'}\}$$

mit

$$\underline{\mathbf{sib}}^0 :\subseteq \{\mathbf{sib} \mid \mathbf{sib}: \mathbf{WB}(\text{DTYP}) \rightarrow \{\text{'wahr','falsch'}\}\}$$

Relationsschema

Als Relationsschema wird die Zusammenfassung einer endlichen Menge von Attributen zur Attributmenge

$$\underline{A} := \{A_1, A_2, ..., A_n\}$$

bezeichnet.

Eine Abbildung $\mathbf{a}$: $\underline{A} \rightarrow \bigcup_{i=1}^{n} \mathbf{WB}(A_i)$ mit $\mathbf{a}(A_i) \in \mathbf{WB}(A_i)$, n ≥ 1 wird als n-Tupel über $\underline{A}$ bezeichnet. Tupel sind Abbildungen, die jedem Attribut des Relationsschemas $\underline{A}$ einen aktuellen Wert zuweisen.

Der Wertebereich des Relationsschemas $\underline{A}$ ist definiert durch

$$\mathbf{WB}(\underline{A}) := \{x \mid x \text{ ist Tupel über } \underline{A}\}$$

Relation

Eine Variable R vom Typ Relation sei definiert als

$$R: \mathrm{REL}(\underline{A} \mid \underline{\mathbf{sib}}^1)$$

Der Wertebereich von R wird eingeschränkt durch die Menge semantischer Integritätsbedingungen $\underline{\mathbf{sib}}^1$ über dem Relationsschema $\underline{A}$.

$$\mathbf{WB}(R) := \{\underline{x} \subseteq \mathbf{WB}(\underline{A}) \mid \forall\ \mathbf{sib} \in \underline{\mathbf{sib}}^1 : \mathbf{sib}(\underline{x}) = \text{'wahr'}\} \cup \{\varnothing\}$$

mit

$$\underline{\mathbf{sib}}^1 :\subseteq \{\mathbf{sib} \mid \mathbf{sib}: \mathbf{PM}(\mathbf{WB}(\underline{A})) \rightarrow \{\text{'wahr', 'falsch'}\}\}$$

PM($\underline{X}$) sei dabei die Potenzmenge einer Menge $\underline{X}$.

DB-Schema

Als DB-Schema wird die Zusammenfassung einer endlichen Menge von Relationen zu einer Menge

$$\underline{R} := \{R_1, R_2, ..., R_m\}$$

$$R_i: \mathrm{REL}(\underline{A}_i \mid \underline{\mathbf{sib}}_i^1)\ ,\ i = 1, ..., m;\ m \geq 1$$

bezeichnet.

Datenbank

Eine Variable *DB* vom Typ Datenbank sei definiert durch

$$DB: \mathbb{DB}(\underline{R} \mid \underline{\mathbf{sib}}^2)$$

Der Wertebereich von *DB* wird eingeschränkt durch die Menge semantischer Integritätsbedingungen $\underline{\mathbf{sib}}^2$ über dem Datenbankschema $\underline{R}$

$$\mathbf{WB}(DB) := \{x \subseteq \mathbf{WB}(\underline{R}) \mid \forall\, \mathbf{sib} \in \underline{\mathbf{sib}}^2 : \mathbf{sib}(x) = \text{'wahr'}\}$$

mit

$$\underline{\mathbf{sib}}^2 :\subseteq \{\mathbf{sib} \mid \mathbf{sib}: \mathbf{WB}(\underline{R}) \rightarrow \{\text{'wahr', 'falsch'}\}\}$$

Projektion

Gegeben sei eine Attributmenge $\underline{A} := \{ A_1,A_2,...,A_n\}$, $\mathbf{a} \in \mathbf{WB}(\underline{A})$ sowie $\underline{C} :\subseteq \underline{A}$ mit $\underline{C} := \{C_1 ,...,C_I\}$, $1 \leq I \leq n$

Die Projektion des Tupels **a** über der Attributmenge $\underline{A}$ auf die Teilmenge $\underline{C}$ ist definiert als

$$\mathbf{a}|_{\underline{C}}: \underline{C} \rightarrow \bigcup_{i=1}^{I} \mathbf{WB}(C_i), \text{ mit } \mathbf{a}|_{\underline{C}}(C_i) := \mathbf{a}(C_i),\ i = 1,...,I$$

Sie entspricht einer Einschränkung der Abbildung **a** auf $\underline{C}$.

Semantische Integritätsbedingungen

Zur vereinfachten Darstellung semantischer Integritätsbedingungen der Ebene 1 ($\underline{\mathbf{sib}}^1$) werden einige Vereinbarungen getroffen [GERNETH 92]:

Gegeben seien die Relationsvariablen *R*: $\mathbb{REL}(\underline{A} \mid \underline{\mathbf{sib}}^1{}_R)$ und *R'*: $\mathbb{REL}(\underline{A}' \mid \underline{\mathbf{sib}}^1{}_{R'})$ mit *R, R'* $\in \underline{R}$ sowie die Datenbankvariable *DB*: $\mathbb{DB}(\underline{R} \mid \underline{\mathbf{sib}}^2)$

(1) Die Primärschlüssel von *R* können mit der boole'schen Funktion $\mathbf{PS}(P,\underline{x})$, mit $P \in \underline{A}$, $\underline{x} \subseteq \mathbf{WB}(\underline{A})$ kontrolliert werden:

$$\mathbf{PS}(P, \underline{x}) :\Leftrightarrow (\forall\, \boldsymbol{a,b} \in \underline{x}: (\boldsymbol{a}|_{\{P\}} = \boldsymbol{b}|_{\{P\}} \Rightarrow (\boldsymbol{a}=\boldsymbol{b})) \wedge (\forall\, \boldsymbol{a} \in \underline{x}: \boldsymbol{a}(P) \neq \omega)).$$

In den semantischen Integritätsbedingungen wird mittels der boole'schen Funktion $\mathbf{PS}(P,\underline{x})$ die Eindeutigkeit des Primärschlüssels (Schlüsselintegrität) gewährleistet. Als Primärschlüssel sollen nur einstellige Schlüsselattribute (Surrogatschlüssel) verwendet werden. Sie tragen keine Bedeutung und dienen lediglich der zeitinvarianten und eindeutigen Identifikation von Tupeln einer Relation bzw. von entsprechenden Objekten eines Objekttyps.

(2) Die boole'sche Funktion **nf**($\underline{B}$, $\underline{x}$) (**nf** steht für **n**icht **f**ehlend) stellt sicher, daß alle Attribute einer Attributmenge $\underline{B}$ in jedem Tupel einen Wert besitzen. Sei $\underline{B} \subseteq \underline{A}$, $|\underline{B}| \geq 1$ und $\underline{x} \subseteq$ **WB**($\underline{A}$), dann ist **nf**($\underline{B}$, $\underline{x}$) wie folgt definiert:

$$\mathbf{nf}(\underline{B}, \underline{x}) :\Leftrightarrow \forall\, \boldsymbol{a} \in \underline{x}: (\forall\, B \in \underline{B}: (\boldsymbol{a}(B) \neq \omega)).$$

(3) Die boole'sche Funktion **nf1**($\underline{B}$, $\underline{x}$) stellt sicher, daß mindestens ein Attribute einer Attributmenge $\underline{B}$ in jedem Tupel einen Wert besitzen. Sei $\underline{B} \subseteq \underline{A}$, $|\underline{B}| \geq 1$ und $\underline{x} \subseteq$ **WB**($\underline{A}$), dann ist **nf1**($\underline{B}$, $\underline{x}$) wie folgt definiert:

$$\mathbf{nf1}(\underline{B}, \underline{x}) :\Leftrightarrow \forall\, \boldsymbol{a} \in \underline{x}: (\exists\, B \in \underline{B}: (\boldsymbol{a}(B) \neq \omega)).$$

Für die vereinfachte Spezifizierung semantischer Integritätsbedingungen auf Ebene 2 ($\underline{\mathbf{sib}}^2$) wird die boole'sche Funktion **FS**((*R,P*),(*R',P'*), $\boldsymbol{v}$) definiert (**FS** steht für Fremdschlüssel). Dabei sei $P \in \underline{A}$, $P' \in \underline{A}'$ und $\boldsymbol{v} \in$ **WB**($\underline{R}$). Bei Fremdschlüsseln handelt es sich um Attribute, welche in der Datenbank in einer beliebigen Relation als Primärschlüsselattribut auftreten. Die angegebene boole'sche Funktion kontrolliert, ob ein Fremdschlüssel *P* der Relation *R* auch als Primärschlüssel *P'* in der angegebenen Relation *R'* von *DB* vorhanden ist.

$$\mathbf{FS}((R,P), (R',P'), \boldsymbol{v}) :\Leftrightarrow \forall\, \boldsymbol{b} \in \boldsymbol{v}(R), \boldsymbol{b}(P) \neq \omega: (\exists\, \boldsymbol{b}' \in \boldsymbol{v}(R'): (\boldsymbol{b}(P) = \boldsymbol{b}'(P'))).$$

Um die Integrität in der Datenbank zu gewährleisten, dürfen keine Tupel gelöscht werden, die von Fremdschlüsseln referenziert werden.

Diskussion des Relationenmodells

Das Relationenmodell [CODD 70] stößt recht schnell an seine Grenzen, wenn man komplexere Sachverhalte (z.B. CAD-Graphiken) angemessen modellieren möchte [vgl. LAUSEN, VOSSEN 96]. Problematisch ist beim Relationenmodell auch, daß lediglich zwei Konstruktoren zur Darstellung von Informationen existieren: Tupel und Mengen. Der Mengenoperator darf zur Bildung einer Relation nur einmal angewendet werden. Die fehlende Möglichkeit, komplexere Objekte als Ganzes zu referenzieren, führt zu aufwendigen Suchanfragen.
Allerdings existieren für das Relationenmodell eine Vielzahl ausgereifter, kommerziell erhältlicher und in großem Umfang eingesetzter Datenbankmanagementsysteme und es existieren Standards. Die Verwendung eines solchen Systems bietet deshalb den großen Vorteil, daß jederzeit ohne große Schwierigkeiten auf ein Softwareprodukt eines anderen Herstellers gewechselt werden kann.

2.10.2 Objektorientierte und objektrelationale Datenbanken

Objektorientierte Datenbanken [HEUER 92] stellen eine ernsthafte Alternative in Bereichen dar, in denen sich relationale Datenbanken als unzulänglich erwiesen haben. Beispielsweise verlangen Multimedia-Systeme, daß komplexe Objekte gespeichert und einfach zugegriffen werden können. Außerdem sind objekt- bzw. typspezifische Operationen in vielen Fällen wünschenswert.
Kommerziell sind eine ganze Reihe objektorientierter Datenbanksysteme erhältlich, z.B. GemStone, ObjectStore, Illustra und O_2. GemStone erweitert Smalltalk so, daß ein Datenbanksystem entsteht. ObjectStore erweitert C++ in ähnlicher Weise. Illustra hat seine Wurzeln in Postgres und Ingres und ist damit die Erweiterung eines der ersten relationalen Datenbanksysteme (Ingres) um Objektorientierung. O_2 ist ein Datenbanksystem im klassischen Sinne mit eigener DDL (data definition language) und DML (data manipulation language). Es gibt also zwei verschiedene Entwicklungsrichtungen bei objektorientierten Datenbanksystemen. Zum einen die Hinzunahme von Datenbankfunktionalität zu einer höheren, objektorientierten Programmiersprache, zum anderen ein mit einer geeigneten Sprache und geeignetem Objektmodell ausgestattetes Datenbanksystem. Während sich objektorientierte DBMSe am Markt bisher kaum durchsetzen konnten, wird dies bei objektrelationalen Systemen [STONEBRAKER, MOORE 98] vermutlich der Fall sein. Denn sie ermöglichen die Nutzung alter Datenbestände in herkömmlicher Weise, gleichzeitig ermöglichen aber auch die Definition einer Objektebene über den vorhandenen Relationen, so daß objektorientiert auf die Datenbank zugegriffen werden kann. Bei neuen CBT-Projekten wird sicherlich in Zukunft verstärkt auf objektrelationale DBMSe zurückgegriffen werden, wenn eine Datenbank eingesetzt werden soll. Bei Verwendung von objektorientierten Programmiersprachen zur Realisierung fällt der Umsetzungsschritt von Relationen zu (komplexeren) Objekten der Programmiersprache weg.
Einer der Gründe für die bisher eher geringe Verbreitung von objektorientierten Datenbanksystemen ist neben der Tatsache, daß alte Datenbestände erst aufwendig portiert werden müssen, das Fehlen allgemein anerkannten Standards. Standardisierungsversuche sind SQL3 (Structured Query Language) und ODMG-93 (Objekt Database Management Group), wobei jedoch noch keiner der beiden Standards verabschiedet ist. Bei SQL3 handelt es sich um eine Erweiterung von SQL, dem relationalen Sprachenstandard. Die Vorschläge der ODMG werden von einer Gruppe von Herstellern objektorientierter Datenbanksysteme entworfen. Der ODMG-93 Vorschlag umfaßt ein Objektmodell, eine Objektdefinitionssprache, eine Objektabfragesprache und die Anbindung an objektorientierte Programmiersprachen wie Smalltalk und C++ [LAUSEN, VOSSEN 96 S. 178; BALZERT 96 S. 740]

3 Entwurf und Modellierung

3.1 Anforderungen an WBT-Systeme

Die im ersten Kapitel erwähnten Schwächen herkömmlicher CBT-Systeme erfordern den Entwurf neuartiger Lehr-/Lernsysteme. Die zu erfüllenden Anforderungen aus Autoren- und Entwicklersicht sollen nachfolgend beschrieben werden [HAAG, MAYLEIN et al. 98]. Zahlreiche weitere Kriterien für die Qualität elektronischer Publikationen aus Nutzerperspektive beinhaltet der *Kriterienkatalog für elektronische Publikationen in der Medizin* [SCHULZ, AUHUBER et al.]. Allerdings ist der Begriff *Elektronische Publikation in der Medizin* allgemeiner definiert als *WBT-System*, so daß nicht alle definierten Anforderungen auf WBT-Systeme angewandt werden können.
Qualitativ hochwertige WBT-Systeme erfüllen neben inhaltlichen auch architektonische Anforderungen und basieren auf Basistechniken, die hohen Ansprüchen genügen. Auf Grundlage der nachfolgend aufgeführten Anforderungen können einerseits bereits existierende WBT-Systeme beurteilt und andererseits neue WBT-Systeme entwickelt werden.

3.1.1 Inhalt und Didaktik

Inhaltliche Anforderungen beschränken sich nicht auf WBT-Systeme. Diese müssen auch von konventionellen CBT-Systemen und anderen Lehr-/Lernmedien wie Büchern erfüllt werden:

- Die Zielgruppe bzw. Zielgruppen, für die ein System entwickelt wurde, müssen klar definiert sein, und dem Anwender zur Kenntnis gebracht werden, bevor er mit der Bearbeitung beginnt. Dadurch wird der Unzufriedenheit von Nutzern eines Systems vorgebeugt, die daraus entsteht, daß die vermittelten Lerninhalte nicht dem aktuellen Wissensstand der Lernenden adäquat sind.
- Die Lernziele, welche mit einem WBT-System erreicht werden sollen, müssen klar definiert sein und dem Anwender bekannt gemacht werden. Dadurch kann ein Anwender selbst vor der Bearbeitung eines Systems entscheiden, ob er durch die Bearbeitung des Systems seinen persönlichen Lernzielen näher kommt.
- Die Breite und Tiefe des dargestellten Lehrstoffs muß für die Zielgruppe hinreichend sein.
- Das im System vorhandene Wissen muß fachlich korrekt und aktuell sein. Dies wird am besten durch einen Review-Prozess des fertigen Systems durch medizinische Fachgebietsexperten erreicht.
- Der didaktische Aufbau des Systems sollte der Zielgruppe und den Lernzielen angemessen sein.
- Das System sollte adäquates Feedback geben, um den Lernfortschritt zu fördern.
- Das System sollte intuitiv bedienbar sein. Die Verwendung von Metaphern kann dazu beitragen.

3.1.2 Architektur

Die Architektur eines WBT-Systems sollte möglichst viele der nachfolgend aufgeführten Anforderungen erfüllen:

- Hohe Performance: WBT-Systeme sollten eine gute Performance aufweisen, um von den Anwendern akzeptiert zu werden. Die Performance ist sowohl von der verfügbaren Netzbandbreite als auch von der Systemarchitektur abhängig. Idealerweise sollten WBT-Systeme auch bei geringer Netzbandbreite, wie sie bei der Verwendung schneller Modems (z.B. 33,6 Kbit/s Übertragungsrate) zur Verfügung stehen, noch akzeptable Warte- und Antwortzeiten bieten.
- Geringe Netzbelastung: Ein WBT-System sollte mit den verfügbaren Netzressourcen möglichst schonend umgehen. Dadurch ergibt sich in der Regel auch eine bessere Performance des Gesamtsystems. Bei der Netzbelastung spielt die Größe eines WBT-Systems eine wichtige Rolle. Entwickler sollten deshalb auf überflüssige Features wie z.B. Musik beim Programmstart verzichten.
- Gute Wiederverwendbarkeit: Einzelne Komponenten eines WBT-Systems müssen mit möglichst geringem Aufwand für andere Projekte wiederverwendbar sein. Inhalte müssen im System so repräsentiert sein, daß deren Export und die Wiederverwendung in anderen Systemen leicht möglich ist.
- Einfache Erweiterbarkeit: Die Architektur des WBT-Systems sollte es ermöglichen, die Systemfunktionalität mit möglichst geringem Aufwand zu erweitern, ohne daß die Lehr-/Lerninhalte beeinflußt werden.
  Eine Aktualisierung bzw. Ergänzung der Lehr-/Lerninhalte sollte einfach möglich sein. Idealerweise sollte hierfür eine Autorenkomponente zur Verfügung stehen, so daß medizinische Fachgebietsexperten Aktualisierungen und Ergänzungen schnell, komfortabel und ohne fremde Hilfe vornehmen können.
- Hinreichende Adaptivität und Adaptierbarkeit: Das System sollte sich zur Förderung des Lernfortschrittes an die persönlichen Vorlieben und Bedürfnisse eines Nutzers selbständig adaptieren. Es sollte aber auch von den Nutzern selbst adaptierbar sein.
- Automatisches Update: Änderungen am System müssen den Anwendern „automatisch“ zur Verfügung stehen.
- Geringe Hardwarevoraussetzungen: Die Architektur des Systems sollte so gewählt werden, daß auf Client-Seite keine High-End-Computer erforderlich sind.
- Multimediale Komponenten: Medizinstudenten geben sich heute kaum mehr mit Lehr-/Lernsytemen zufrieden, die nur eine reine Textoberfläche besitzen. Die Verwendung verschiedener Medien (Text, Bild, Animation, Sound...) macht die Systeme für Anwender attraktiver und die Lerneffekte sollen sich nach der Summierungsthese in der Mediendidaktik verstärken, wenn mehrere Kanäle zur Informationsaufnahme benutzt werden [WEIDENMANN 91 S. 13]. Die Architektur eines WBT-Systems muß die Verwendung von multimedialen Daten unterstützen.
- Ergonomische Benutzeroberfläche: Die Benutzeroberfläche sollte sich an gängigen GUI-Standards [MICROSOFT 95, APPLE 92] orientieren, so einfach wie möglich gehalten sein und eine intuitive Bedienung des Systems ermöglichen.
- Einfache Implementierbarkeit: Die Architektur sollte mit vertretbarem Aufwand realisierbar sein.

- Gute Integrierbarkeit: WBT-Systeme müssen mit wenig Aufwand in klinische Arbeitsplatzsysteme integrierbar sein. Durch Verfügbarkeit von Systemen am Arztarbeitsplatz kann ein wichtiger Beitrag zur Weiterbildung von im Beruf stehenden Ärzten geleistet werden.
- Integrierte Kommunikationskomponente: Die Kommunikation zwischen den Lernenden untereinander sowie zwischen Lernenden und Lehrenden sollte direkt vom System aus möglich sein, so daß die Nutzer zum Zwecke der Kommunikation keine zusätzlichen Programme starten müssen. Auftretende Fragen, die nicht alleine beantwortet werden können, können so diskutiert und beantwortet werden.

3.1.3 Basistechniken

Die Wahl geeigneter Werkzeuge und Programmiersprachen ist eine wichtige Voraussetzung für die Erstellung hochwertiger WBT-Systeme. In Tabelle 2 sind die wichtigsten Basistechniken (siehe Kapitel 2.6.2) dargestellt und den Architekturtypen (siehe Kapitel 2.5.1) zugeordnet, für deren Erstellung sie besonders geeignet sind.

Basistechnik(en)	Architektur(en)
HTML	Alle
JavaScript	Client-based
Java	Client-based, Remote Data & Knowledge, Distributed Teaching
RMI, CORBA (JAVA)	Distributed Teaching
ActiveX	Client-based
Plug-Ins	Client-based
CGI & beliebige Programmiersprache	Server-based
VRML	Client-based

Tabelle 2: Basistechniken zur Erstellung von WBT-Systemen

An die Basistechniken sind folgende Anforderungen zu stellen:

- Weitgehende Plattformunabhängigkeit: Die eingesetzten Basistechniken sollten die Erstellung von WBT-Systemen ermöglichen, die auf möglichst vielen Plattformen und unter möglichst vielen Betriebssystemen einwandfrei funktionieren. Hierdurch wird die Zahl potentieller Nutzer des Systems so groß wie möglich gehalten. Plattformunabhängigkeit ist vor allem auf Client-Seite von Bedeutung. Dagegen spielt die serverseitige Plattformunabhängigkeit nur eine untergeordnete Rolle.
- Hohe Sicherheit: Diesem Aspekt muß große Aufmerksamkeit geschenkt werden, da sich viele der in der medizinischen Aus- und Weiterbildung eingesetzten Computer in Kliniknetzen befinden, in denen mit hoch sensiblen Patientendaten gearbeitet wird. Es muß also ausgeschlossen sein, daß durch „bösartige“ WBT-Systeme Daten ausspioniert bzw. Schäden an den Computern verursacht werden können, auf denen mit Lehr-/Lernsystemen gearbeitet wird. Für Entwickler bedeutet dies, daß sie keine Basistechniken einsetzen dürfen, die keine ausreichenden Schutzmechanismen gegen Mißbrauch aufweisen.

- Hohe Performance: Die eingesetzten Basistechniken sollten eine hohe Performance aufweisen.
- Unterstützung von Netzwerkfunktionen: Für die Realisierung von WBT-Systemen mit Distributed Teaching-Architektur ist es wichtig, daß die Basistechniken Netzwerkfunktionen wie Remote Procedure Call und dergleichen unterstützen. Außerdem sollte der Zugriff auf Internet-Dienste wie Newsgruppen leicht möglich sein.
- Unterstützung von Multimedia: Die Verwendung von multimedialen Komponenten muß gut unterstützt werden.
- Installationsfreiheit: Die eingesetzten Basistechniken sollten die Erstellung von WBT-Systemen ermöglichen, bei denen keine Installation erforderlich ist. Installationen können Instabilitäten auf einem Rechner hervorrufen und dafür verantwortlich sein, daß bereits installierte Programme nicht mehr funktionsfähig sind. Außerdem ist die Hemmschwelle zur Nutzung eines Systems höher, wenn eine Installation erforderlich ist.
- Hohe Produktivität: Um die Entwicklungskosten in Grenzen zu halten, sollten die eingesetzten Basistechniken eine hohe Produktivität ermöglichen.

3.1.4 Architekturentscheidung

Als Architekturtyp wurde *Distributed Teaching* gewählt. Dieser Architekturtyp bietet bei geringen Netzbandbreiten Performance-Vorteile gegenüber den anderen in Kapitel 2.5.1 vorgestellten Architekturtypen. Die aufwendigere Implementierung im Vergleich zu den anderen Architekturen wird dabei in Kauf genommen. Unter anderem deshalb, weil durch das in Kapitel 3.3 vorgestellte Lehr-/Lernsystemshell-Konzept eine exzellente Wiederverwendbarkeit gegeben ist.
Tabelle 3 bewertet eine Distributed Teaching-Architektur hinsichtlich der in Kapitel 2 spezifizierten Anforderungen [vgl. HAAG, LEVEN 97]. Da unterschiedliche Basistechniken für die Implementierung eingesetzt werden können, ist in der Tabelle die Beurteilung der Architektur unter Einsatz der in Frage kommenden Basistechniken enthalten. HTML dient dabei immer als Grundlage für die Einbettung der weiteren Basistechniken. Die Beurteilungen beziehen sich dabei auf ideale Implementierungen, d.h. die Beurteilung von realen Systemen kann unter Umständen schlechter ausfallen als in der Tabelle angegeben.
Während die besonderen Stärken der Distributed Teaching-Architektur (hohe Performance, geringe Netzbelastung, Möglichkeit zur Lastverteilung) unabhängig von den verwendeten Basistechniken sind, gibt es Unterschiede bei der zu erreichenden Softwareproduktivität und bei der Komplexität der Implementierung. Während es sehr mühselig und zeitaufwendig ist, Kommunikationsmechanismen mit den Socket-Klassen von Java zu programmieren oder Interface-Beschreibungen für die Kommunikation mittels IDL bei CORBA zu definieren, ist die Produktivität bei Verwendung von Java-RMI deutlich höher. Die Implementierung ist mit HTML & Java am schwierigsten, weil nicht wie bei RMI transparent auf die entfernten Objekte zugegriffen werden kann, als wären es lokale Objekte. Der Entwickler muß sich hier ausführlich Gedanken darüber machen, welche Daten und Objekte in welcher Form zwischen Client und Server des Lehr-/Lernsystems ausgetauscht werden sollen. Auch bei Verwendung von Java und CORBA gestaltet sich die Programmierung etwas schwieriger als bei Java & RMI, bedingt durch das sprachunabhängige Objektmodell von CORBA.

Anforderungen	Basistechniken		
	HTML & Java	HTML & Java & RMI	HTML & Java & CORBA
Weitgehende Plattformunabhängigkeit	+	+	+
Hinreichende Sicherheit	+	+	+
Hohe Performance	++	++	++
Geringe Netzbelastung	++	++	++
Keine Installation	++	++	++
Gute Wiederverwendbarkeit	+	+	+
Einfache Erweiterbarkeit	+	+	++
Automatisches Update	++	++	++
Geringe Kosten / Hohe Produktivität	-	o	-
Multimediale Komponenten	+	+	+
Ergonomische Benutzeroberfläche	+	+	+
Einfache Implementierbarkeit	--	o	-

Tabelle 3: Beurteilung einer idealen Distributed Teaching-Architektur

Legende:

&	und	o	befriedigend
++	sehr gut	-	schlecht
+	gut	--	sehr schlecht

3.2 Phasenmodell zur Entwicklung von WBT-Systemen

Bei der Entwicklung von CBT-Systemen [KERRES 98] ist ebenso wie bei der Erstellung anderer Softwaretypen eine systematische, strukturierte Vorgehensweise erforderlich und von großem Vorteil. Nachfolgend wird deshalb eine Vorgehensweise beschrieben, welche die Erstellung von Lehr-/Lernsystemen unterstützen soll. Im vorgestellten Phasenmodell werden keine Methoden des Software-Engineering definiert. Vielmehr kann hier auf die in diesem Bereich verbreiteten und etablierten Methoden zurückgegriffen werden [BURKHARDT 97; BOOCH 94; RUMBAUGH, BLAHA et al. 91; JACOBSON, CHRISTERSON et. al. 92]. Ein Entwickler kann also jederzeit das Phasenmodell mit den ihm vertrauten Analyse- und Designmethoden des Software-Engineerings anwenden, ohne sich in neue Methoden einarbeiten zu müssen.
Die im vorhergehenden Kapitel genannten Anforderungen „gute Wiederverwendbarkeit“ und „einfache Erweiterbarkeit“ machen eine möglichst vollständige Trennung von Lehrwissen (Lehrstrategien usw.) und Domänendaten bzw. Domänenwissen erforderlich. Von anderen Phasenmodellen zur Erstellung von Software bzw. CBT-Systemen (z.B. [BODENDORF 90 S. 76]) unterscheidet sich das vorgestellte Phasenmodell deshalb u.a. in der strikten Trennung und expliziten Aufführung der Phasen *Domänenmodellierung* und *Lehrfunktionsmodellierung*. Für die Implementierung wird eine objektorientierte Programmiersprache vorgeschlagen, um der Forderung nach guter Wiederverwendbarkeit nachzukommen. Die Verwendung einer solchen Sprache ist aber nicht unbedingt Voraussetzung für die Anwendbarkeit des Phasenmodells.
Aufbauend auf diesem Phasenmodell kann ein Autorensystem entwickelt werden, das eine effiziente Erstellung von CBT-Systemen ermöglicht.

3.2.1 Didaktische Analyse

Bei der Entwicklung von Lehr-/Lernsystemen muß in einer ersten Phase das didaktische Konzept erstellt werden. Der Autor (Fachgebietsexperte) muß hierfür die Lehr- und Lernziele ebenso definieren wie die zu vermittelnden Lehr-/Lerninhalte und die angepeilte(n) Zielgruppe(n). Die Lehr-/Lerninhalte sollten dabei so strukturiert werden, wie sie später auch den Nutzern des CBT-Systems präsentiert werden sollen. Für die Definition der Lehr-/Lernziele sollte eine Lernzieltaxonomie verwendet werden (z.B. [BLOOM 72]). Zum Erreichen dieser Ziele stehen bei medizinischen CBT-Systemen fünf unterschiedliche Interaktionsformen mit spezifischen Stärken und Schwächen (siehe Kapitel 2.3.1) zur Verfügung. Auf Basis der gewählten Ziele und Lerninhalte wählt der Autor die seiner Meinung nach geeignetste(n) Interaktionsform(en).

3.2.2 Anforderungsanalyse

Bei der *Anforderungsanalyse* erstellen Autor und Entwickler gemeinsam ein Pflichtenheft, in dem möglichst detailliert die Anforderungen an die Funktionalität des zu erstellenden Lehr-/Lernsystems aufgelistet werden. Es muß genau festgelegt werden, wie das CBT-System auf Nutzereingaben und -aktivitäten reagiert. Außerdem muß spezifiziert werden, ob und in welcher Weise sich das System an die Nutzer adaptieren soll. Ebenfalls definiert werden muß, welche Systemfunktionen über ein Menü bzw. über Buttons auf Bildschirmseiten (z.B. Hilfefunktion) verfügbar gemacht werden sollen.

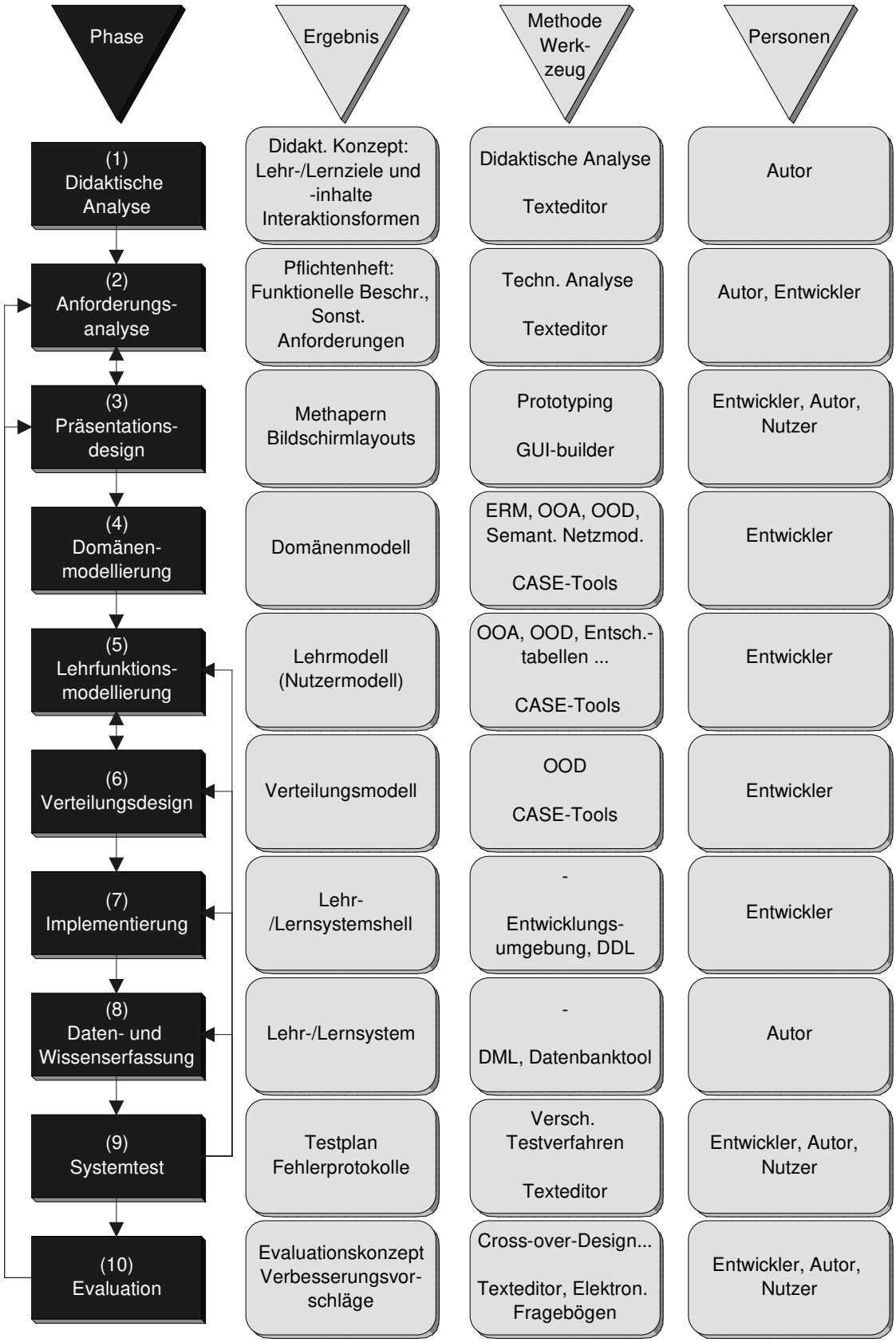

Abbildung 10: Phasenmodell zur Erstellung von WBT-Systemen

Die Spezifikation wird dadurch erleichtert, daß im vorhergehenden Schritt bereits die Interaktionsformen festgelegt wurden. Soll das System beispielsweise die Interaktionsform Browsing aufweisen, wünscht der Autor in der Regel eine Suchfunktion oder einen Index. Der Entwickler ist dadurch besser in der Lage, den Autoren auf seiner Meinung nach fehlende Systemfunktionen hinzuweisen. Nachdem alle Anforderungen aufgelistet worden sind, wählt der Entwickler eine geeignete Programmiersprache und Programmierumgebung für die Implementierung aus. Anschließend kann er die personellen, organisatorischen und hardwaretechnischen Anforderungen bestimmen und einen Zeitplan erstellen.

3.2.3 Präsentationsdesign

Als erstes muß in dieser Phase entschieden werden, ob Metaphern verwendet werden können und sollen. Die Darstellung eines Arztzimmers mit verschiedenen Untersuchungswerkzeugen, einer Bücherwand usw. bei dem die Programmbedienung durch Anwahl von Objekten mit der Maus möglich ist, ist ein Beispiel für eine Metapher. Metaphern ermöglichen es auch computerunerfahrenen Nutzern ein CBT-System intuitiv zu bedienen. Sie sollten also verwendet werden, wenn dies sinnvoll möglich ist.
Anschließend kann mit der Erstellung der Bildschirmlayouts begonnen werden. Während der Autor seine Vorstellungen vom Aussehen der Layouts einbringt, achtet der Entwickler auf Konsistenz und Einhaltung der User Interface Guidelines. Darüber hinaus muß der Entwickler darauf achten, daß die im Pflichtenheft vorhandenen und vom Benutzer direkt aufrufbaren Systemfunktionen in den Bildschirmlayouts in Erscheinung treten. Die erstellten Bildschirmlayouts können mit Personen aus dem Kreis der zukünftigen Nutzer besprochen und gleich in diesem frühen Stadium an die Vorstellungen der Zielgruppe angepaßt werden (Prototyping). Durch die erstellten Bildschirmlayouts wird ein guter Eindruck vom zukünftigen Lehr-/Lernsystem vermittelt. Dadurch ist eine gewisse Vollständigkeitskontrolle der *Anforderungsanalyse* möglich. Bei Bedarf kann nochmals dorthin zurückgekehrt werden. Um die Bildschirmlayouts schnell und ohne Programmieraufwand erstellen zu können, wird vorgeschlagen, einen GUI-Builder (Graphical User Interface) der gewählten Programmierumgebung zu verwenden, falls ein solcher verfügbar ist. Bei der Implementierung können die erstellten Layouts dann ohne zusätzlichen Aufwand verwendet werden.

3.2.4 Domänenmodellierung

Die strikte Trennung zwischen Domänendaten und Domänenwissen auf der einen Seite und Lehrwissen auf der anderen Seite macht es erforderlich, diese auch strikt voneinander getrennt zu modellieren. Es wird empfohlen, mehrere Modelle zu entwerfen, um die Komplexität zu reduzieren und das Domänenmodell insgesamt übersichtlicher und verständlicher zu machen. Werkzeuge, welche einen Model-View-Controller (MVC) besitzen, sind hierfür besonders geeignet. Bei teilweise überlappenden Modellen werden Änderungen an gleichzeitig in mehreren Objekten vorhandenen Objekten automatisch auch in den anderen Modellen wirksam. Das Domänenwissen kann in semantischen Netzen dargestellt werden. Semantische Netze stellen Wissen in Form von Objekten und semantischen Beziehungen zwischen diesen dar. Sie weisen eine gewisse Ähnlichkeit zu Frames [MINSKY 75] auf, semantische Netze sind jedoch das allgemeinere Konzept. Bei der Implementierung werden häufig Ähnlichkeiten mit dem Entity-Relationship-Modell sichtbar [BODENDORF 90 S. 125], welches sich für die Modellierung der Domänendaten eignet. Editoren, mit denen es möglich ist, den Code für die modellierten Klassen sowie Befehle für die Erzeugung von Datenbankrelationen bzw. -objekten zu generieren, verbessern die Produktivität der Entwickler.

3.2.5 Lehrfunktionsmodellierung

Bei der *Lehrfunktionsmodellierung* geht es darum, die in der *Anforderungsanalyse* von Autor und Entwickler gemeinsam erarbeitete und definierte Lehrfunktionalität zu modellieren. Es ist festzulegen, wie das System auf Benutzeraktionen bzw. das Ausbleiben von Benutzeraktionen reagiert, ob es eine aktive oder passive Rolle gegenüber dem Nutzer einnimmt und wann es dem Nutzer welche Teile der im System enthaltenen Domänendaten und des Domänenwissens präsentiert. Es muß also aktives, prozedurales Wissen modelliert werden. Die Beschreibung kann z.B. mittels Entscheidungstabellen, Regeln oder Pseudocode erfolgen. Bei Verwendung einer objektorientierten Programmiersprache ist festzulegen, welche Objekte welche Methoden aufweisen, für welche Aufgaben zuständig sind und welche Nachrichten sie untereinander austauschen. Für die Darstellung sind die verfügbaren OOA/OOD-Notationen (objektorientierte Analyse, objektorientiertes Design), z.B. [BURKHARDT 97; COAD, YOURDON 92; BOOCH 94; RUMBAUGH, BLAHA et al. 91; JACOBSON, CHRISTERSON 92] gut geeignet. Idealerweise werden Editoren für die Modellierung eingesetzt, die in der Lage sind, aus dem erstellten Modell automatisch alle Klassen und deren Beziehungen zueinander in der gewählten Programmiersprache zu generieren. Soll ein adaptives Lehr-/Lernsystem erstellt werden, muß zusätzlich zum Lehrmodell auch ein Nutzermodell erstellt werden. Beim Entwurf des Nutzermodells muß bestimmt werden, welche Informationen über den Nutzer dieses enthalten muß, wie diese Informationen gewonnen werden und wie sie im Nutzermodell repräsentiert werden sollen.

3.2.6 Verteilungsdesign

CBT-Systeme können in die drei Schichten Präsentationsschicht, Lehrsystemlogikschicht sowie Domänendaten- & Wissensschicht eingeteilt werden [HAAG, MAYLEIN et al. 98]. Beim *Verteilungsdesign* muß entschieden werden, auf welchem Rechner (Client oder Server) diese Schichten angesiedelt sind. Bei der Distributed Teaching-Architektur (siehe Kapitel 2.5.1.3) muß zusätzlich für alle Lehrobjekte (siehe Phase *Lehrfunktionsmodellierung*) bestimmt werden, wo sie angesiedelt sind. Sollte sich dabei herausstellen, daß sich durch eine Umstrukturierung der Klassen bzw. der Klassenzuständigkeiten Vorteile für das verteilte CBT-System ergeben, kann nochmals zur *Lehrfunktionsmodellierung* zurückgekehrt werden. Der Entwurf einer Kommunikationsschnittstelle zwischen Client und Server gehört ebenso zu den Aufgaben dieser Phase wie die Definition der Daten- und Wissensbankschnittstellen (falls Daten- und Wissensbanken verwendet werden sollen). Auch die Definition einer Schnittstelle zu externen Ressourcen ist Aufgabe dieser Phase, falls eine solche Schnittstelle erforderlich ist.
Die Erstellung des *Verteilungsdesigns* ist alleinige Aufgabe des Entwicklers. Bei Verwendung einer objektorientierten Sprache kann OOD unter Verwendung bekannter Notationen eingesetzt werden.

3.2.7 Implementierung

Mit Hilfe der bereits gewählten Entwicklungsumgebung wird das „leere“ Lehr-/Lernsystem implementiert (nur Oberfläche und Lehrsystemlogik, Struktur der Domänendaten- & Wissensschicht). Dabei können die schon erstellten Bildschirmlayouts sowie die in den beiden vorangegangenen Schritten evtl. generierten Klassen verwendet werden. Bei der Implementierung muß das bei der *Lehrfunktionsmodellierung* entworfene aktive, prozedurale Wissen in der Zielsprache implementiert, die Datenbank erstellt und der Zugriff auf diese mittels geeigneter Treiber und Klassen implementiert werden.

3.2.8 Daten- und Wissenserfassung

Nachdem die Lehr-/Lernsystemshell fertiggestellt ist, muß sie mit Inhalten (Domänendaten und Domänenwissen) gefüllt werden. Dies ist Aufgabe des Autors. Er muß nun die bei der *didaktischen Analyse* definierten Lehr-/Lerninhalte in das System bringen. Um diese Aufgabe zu erleichtern, sollte dem Autoren hierfür ein geeignetes Tool zur Verfügung gestellt werden. Bei Verwendung eines DBMS muß ansonsten auf Befehle der DML (data manipulation language) eines DBMS zurückgegriffen werden (INSERT-Anweisungen bei relationalen DBMS). Nach Abschluß der Eingabe liegt ein vollständiges Lehr-/Lernsystem vor, das nun ausgetestet werden kann.

3.2.9 Systemtest

Am Test des fertiggestellten CBT-Systems sollten sowohl Entwickler, Autor als auch Mitglieder der geplanten Zielgruppe teilnehmen. Vor Beginn des Tests muß der Entwickler einen Testplan aufstellen, um sicherzustellen, daß auch das gesamte Lehr-/Lernsystem und nicht nur Teile davon ausgetestet werden. Die durchgeführten Tests und aufgetretene Fehler müssen schriftlich festgehalten werden. Der Entwickler kann dann auf Basis der Fehlerprotokolle das System überarbeiten. Test und Fehlerkorrektur werden so lange durchgeführt, bis keine Fehler mehr gefunden werden. Das System kann nun in der Praxis eingesetzt werden.

3.2.10 Evaluation

Um feststellen zu können, ob mit dem erstellten Lehr-/Lernsystem die in der *didaktischen Analyse* definierten Lehr-/Lernziele erreicht werden, muß eine *Evaluation* durchgeführt werden. Vorher ist allerdings ein Evaluationskonzept zu erarbeiten, in dem die Fragestellungen aufgeführt und geeignete Methoden für deren Beantwortung spezifiziert sind. Die Benutzerakzeptanz eines Systems oder der Grad der Benutzerzufriedenheit läßt sich dabei relativ einfach durch von den Anwendern auszufüllende Fragebögen feststellen. Eine Aussage darüber, ob die definierten Lehr-/Lernziele erreicht werden, kann nur mit aufwendigen Studien gemacht werden.

3.3 Lehr-/Lernsystemshell-Konzept

Betrachtet man verfügbare Lehr- und Lernsysteme im medizinischen Bereich, so findet man bei diesen praktisch immer die in Kapitel 2.3.1 beschriebenen Interaktionsformen bzw. Kombinationen von diesen. Auch wenn sich CBT-Systeme mit gleicher Interaktionsform in ihrer Benutzungsschnittstelle stark unterscheiden mögen, so ist die bereitgestellte Funktionalität in der Regel sehr ähnlich. Bei Browsingsystemen beispielsweise ist fast immer ein Inhaltsverzeichnis und/oder ein Index vorhanden, über die die Nutzer gezielt nach Informationen suchen können. Fallsimulationen sind fast immer gleich aufgebaut, beginnen mit Anamnese und klinischer Untersuchung und enden mit der Wahl von Arbeitsdiagnosen und Therapieprinzipien. Während im Bereich der Künstlichen Intelligenz sogenannte *Expertensystemshells* wie z.B. D3 existieren, welche den Aufbau von Wissensbasen unterstützen und gleichzeitig auch verschiedene Problemlösungsmethoden bereitstellen, existieren solche umfassenden Shell-Ansätze für den Bereich der computerunterstützten Ausbildung nicht. Die angesprochene Expertensystemshell D3 (siehe Kapitel 2.8.3) kann zwar auch als Basis für Lehr-/Lernsysteme dienen [PUPPE, REINHARDT 95; REINHARDT, PUPPE 97], allerdings ist D3 nicht speziell für den CBT-Bereich entwickelt worden. Das Erstellen von Wissensbasen, die auch für Ausbildungszwecke gut geeignet sind, ist deshalb nicht ganz einfach und erfordert einen hohen Zeitaufwand. Für jedes Fachgebiet, für welches ein Intelligentes Tutorielles System erstellt werden soll, muß eine Wissensbasis vorliegen, in der das zugrundeliegende Domänenwissen in hinreichender Tiefe verfügbar ist. Es ist jedoch sehr schwierig, für die Erstellung von Wissensbasen Fachgebietsexperten zu gewinnen, wodurch dieser Ansatz in der praktischen Umsetzung auf Schwierigkeiten stößt. Das in dieser Arbeit vorgestellte Konzept für plattformunabhängige und adaptive Lehr-/Lernsysteme wurde deshalb bewußt so konzipiert, daß es ohne eine umfangreiche Wissensbasis auskommt. Nicht zuletzt deshalb, weil für ITSe bzw. tutoriell nutzbare Expertensysteme der tutorielle Nutzen bisher noch gar nicht nachgewiesen werden konnte (siehe auch Kapitel 2.8.2) und deshalb auch nicht geklärt ist, ob sich der wesentlich höhere Aufwand für die Erstellung solcher Systeme überhaupt lohnt.
Normalerweise ist man also bei der Entwicklung von CBT-Systemen dazu gezwungen, das gesamte System von Grund auf selbst zu entwickeln, z.B. auf Basis des in Kapitel 3.2 vorgestellten Phasenmodells. Nur in Teilbereichen, beispielsweise für die Erstellung von Präsentations- und Browsingsysteme existieren kommerziell erhältliche Autorensysteme (siehe Kapitel 2.3.2), welche die benötigte Funktionalität wenigstens teilweise auf relativ hohem Abstraktionsniveau zur Verfügung stellen, jedoch auch Nachteile besitzen.
Zur wesentlichen Vereinfachung der Entwicklung von Lehr-/Lernsystemen wird deshalb in Analogie zu den Expertensystemshells der Entwurf und die Implementierung einer Lehr-/Lernsystemshell vorgeschlagen, welche es Fachgebietsexperten erlauben soll, ohne bzw. nur mit geringer Unterstützung von Informatikern bei vergleichsweise geringem Zeitaufwand qualitativ hochwertige Lehr-/Lernsysteme erstellen zu können. Eine solche Shell sollte alle fünf in Kapitel 2.3.1 vorgestellten Interaktionsformen unterstützen, um den Autoren maximale Flexibilität bei der Erstellung ihrer Lehr-/Lernsystems zu geben. Einen wesentlichen Punkt einer solchen Shell stellt eine leicht zu bedienende Autorenkomponente dar. Die leichte Bedienbarkeit kann z.B. dadurch erreicht werden, daß sich die Autorenkomponente an die persönlichen Vorlieben eines Autors anpassen läßt und nicht benötigte Funktionalität automatisch ausgeblendet wird. Voraussetzung für eine funktionsfähige Autorenkomponente ist, daß sie auf einem Phasenmodell basiert, wie es z.B. in Kapitel 3.2 vorgestellt wurde, um unerfahrene Nutzer bei der schrittweisen, systematischen Erstellung von CBT-Systemen zu unterstützen. Bei Nutzung einer Lehr-/Lernsystemshell

müßten jedoch nur die Phasen 1 bis 2 und Phase 8 bis 10 berücksichtigt werden. Die Phasen 3 bis 7 sind in diesem Fall bereits von den Entwicklern der Shell durchgeführt worden. Die breite Verfügbarkeit der Shell kann dadurch erreicht werden, daß neben dem zu erstellenden Lehr-/Lernsystem auch die Autorenkomponente plattformunabhängig und im Internet verfügbar ist. Ein weiterer wichtiger Vorteil einer Lehr-/Lernsystemshell besteht darin, daß eine Wiederverwendbarkeit der Inhalte einzelner Lehr-/Lernsysteme hervorragend möglich ist.

3.4 CAMPUS-Konzept

Für die Umsetzung der im Rahmen dieser Arbeit entwickelten Ideen, Konzepte und Modelle wurde das CAMPUS-Projekt (Computerunterstützte Aus- und Weiterbildung in der Medizin durch plattformunabhängige Software) gestartet. Mit diesem Projekt ist das Heidelberger *Labor für computerunterstützte Ausbildung in der Medizin* auch am Projekt VIROR (Virtuelle Hochschule Oberrhein) [VIROR] des Landes Baden-Württemberg beteiligt.
Das CAMPUS-Konzept soll möglichst viele der in Kapitel 3.1 definierten Anforderungen an WBT-Systeme erfüllen. Darüber hinaus soll die Idee einer Lehr-/Lernsystemshell (siehe Kapitel 3.3) berücksichtigt werden, da sich daraus einige wichtige Vorteile insbesondere in Bezug auf die Wiederverwendbarkeit der Inhalte und die schnelle Erstellung von Lehr-/Lernsystemen ergeben. Das zu erstellende Autorensystem soll auf dem in Kapitel 3.2 vorgestellten Phasenmodell basieren.
Da die Konzeption und Implementierung einer umfassenden Shell den Rahmen dieser Arbeit bei weitem sprengen würde, konzentrieren sich die nachfolgenden Ausführungen auf die Interaktionsformen *Simulation* (Fallsimulation), *Browsing* (systematisches Lehrbuchwissen) und *Drill & Practice* (Fragen). Das vorgestellte Konzept kann jedoch problemlos um die noch fehlenden Interaktionsformen und zusätzliche Funktionalität erweitert werden. Insbesondere wird darauf hingewiesen, daß bei der Konzepterstellung davon ausgegangen wurde, daß das System im Rahmen von Lehrveranstaltungen eingesetzt wird. Das Ziel des CAMPUS-Konzeptes besteht nicht darin, für die Studierenden der Medizin ein Werkzeug zu entwickeln, mit dem diese sich selbst mittels des Lesens von elektronischen Lehrbüchern und der Bearbeitung von Lehr-/Lernfällen im Selbststudium Medizin aneignen können. Vielmehr sollen die Medizindozenten ein einfach zu nutzendes Werkzeug an die Hand bekommen, mit dem sie vertiefendes und ergänzendes Unterrichtsmaterial für die Studierenden erstellen können, sowie theoretisch vermittelte, systematische Vorgehensweisen anhand von Fallsimulationen einüben lassen können. Durch das CAMPUS-System werden weder traditionelle Lehrveranstaltungen noch die sehr wichtige Ausbildung am realen Patienten überflüssig. Es ist also als Ergänzung traditioneller Lehr-/Lernformen zu sehen.

3.4.1 CAMPUS Lehr-/Lernsystemshell

Die CAMPUS Lehr-/Lernsystemshell besteht aus vier Subsystemen (siehe Abbildung 11):

1. Autorensubsystem
2. Lernsubsystem
3. Lehrsubsystem
4. Administrationssubsystem

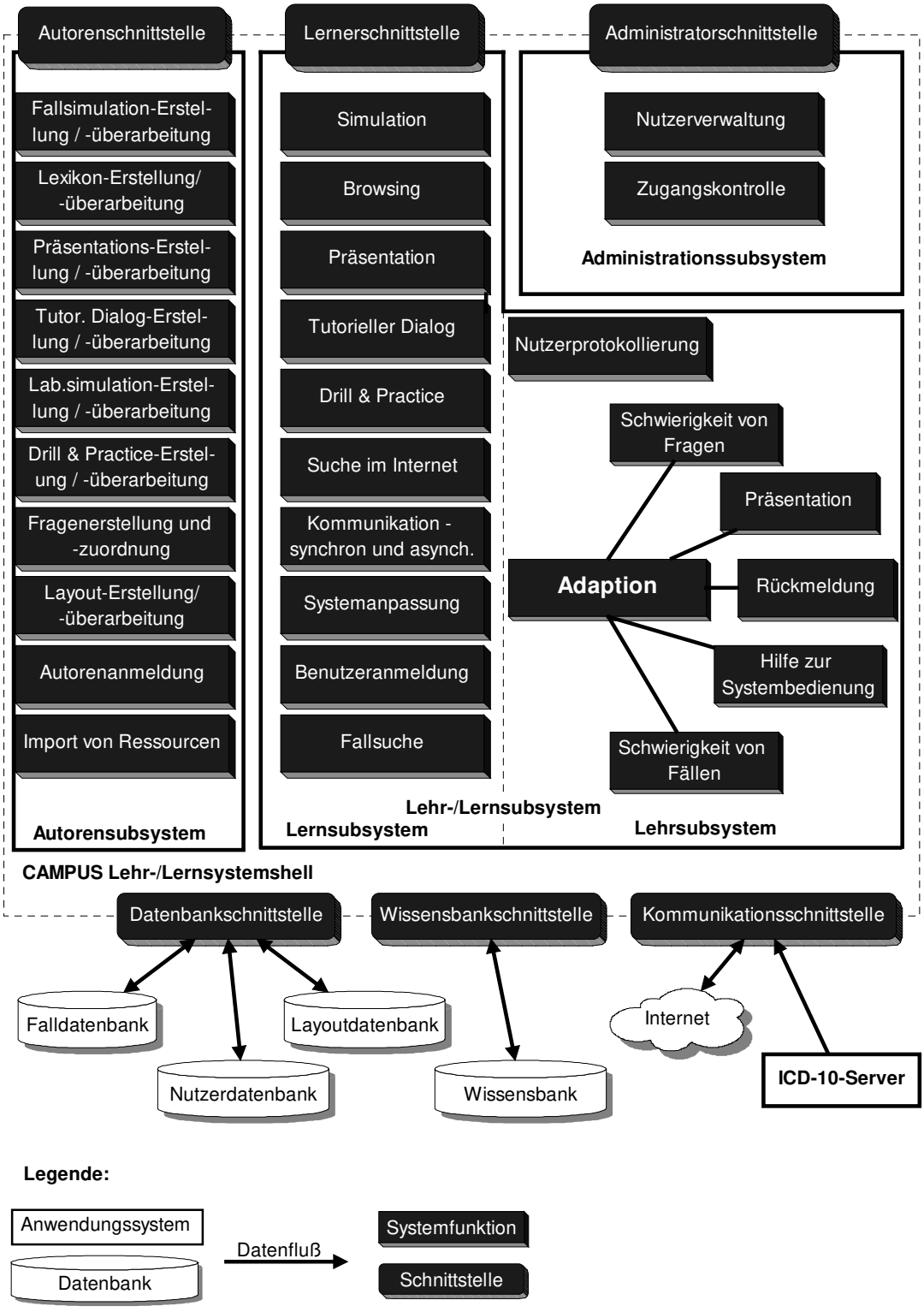

Abbildung 11: CAMPUS Lehr-/Lernsystemshell

Mit dem Autorensubsystem [SINGER, HAAG et al. 98] können Fachgebietsexperten CAMPUS Lehr-/Lernsysteme (mit verschiedenen Interaktionsformen) erstellen. Dabei benötigen sie keinerlei Informatikkenntnisse. Die Autoren können bzw. sollen sich ganz auf die fachlichen Inhalte konzentrieren. Das Autorensubsystem ist plattformunabhängig und auf gängigen Computern mit Internetanbindung ablauffähig. Eine Nutzung am Arbeitsplatz

eines Dozenten ist dadurch meist möglich. Das Autorensubsystem unterstützt auch die Nutzung bereits vorhandener Ressourcen im WWW. So ist es möglich, existierende HTML-Dokumente in die Wissensbank aufzunehmen [PELZER 98]. Autoren können solche Dokumente an geeigneten Stellen in das zentrale Lexikon des CAMPUS-Systems integrieren bzw. einzelne Elemente in eigenen Dokumenten wiederverwenden. Das Lexikon enthält u.a. Wissen, welches Nutzer für die Bearbeitung von Lehr-/Lernfällen benötigen. Darüber hinaus kann jedoch auch umfangreiches Lehrbuchwissen enthalten sein, das durch die Interaktionsform Browsing den Nutzern des CAMPUS-Systems zugänglich ist. Mit der Systemfunktion Fragenerstellung und -zuordnung können Autoren Fragen mit unterschiedlichen Antworttypen erstellen und diese Fragen beliebigen Knoten des Lexikons bzw. beliebigen Zeitpunkten in der Fallbearbeitung zuordnen.
Lernsubsystem und Lehrsubsystem bilden zusammen mit den von einem Autor mit Hilfe des Autorensubsystems in die Daten- und Wissensbank eingetragenen Inhalten ein CAMPUS Lehr-/Lernsystem.
Das Lernsubsystem erlaubt es den Lernenden, sich einzuloggen und mit CAMPUS Lehr-/Lernsystemen zu arbeiten. Es unterstützt alle fünf in Kapitel 2.3.1 beschriebenen Interaktionsformen. Das Einloggen ist erforderlich, damit das CAMPUS-System beispielsweise feststellen kann, welche Lehr-/Lernfälle bereits erfolgreich bearbeitet wurden, in welchem Semester sich ein Student befindet usw. Das Lernsubsystem präsentiert Lehr-/Lernfälle, zeigt Fragen an und ermöglicht den komfortablen Zugriff auf externe Ressourcen im Internet und die Kommunikation mit Dozenten und Mitstudenten per E-Mail und IRC (Internet Relay Chat). Die Lernenden können auch selbst Lehr-/Lernfälle auswählen (nach verschiedenen Kriterien), die sie bearbeiten möchten. Die Funktion *Systemanpassung* ermöglicht es den Lernenden, das System selbst zu adaptieren. Durch seine Plattformunabhängigkeit kann das Lernsubsystem auf den meisten Computern mit Internetanbindung genutzt werden.
Das Lehrsubsystem protokolliert die wesentlichen Lerneraktionen mit und legt sie in der Nutzerdatenbank ab. Das Lehrsystem ist durch ständige Auswertung dieser Daten in der Lage, sich an die Bedürfnisse des Lernenden, was beispielsweise die Art der Fallpräsentation bzw. Interaktionsform betrifft, anzupassen. Das Lehrsubsystem ist also für die Anpassung des Systems an die Lernenden (siehe Kapitel 3.4.3) sowie die Generierung geeigneter Rückmeldungen auf Aktionen der Lernenden verantwortlich.
Das Administrationssubsystem wird für die Nutzerverwaltung und die Zugangskontrolle eingesetzt. Es können Autoren- und Lerneraccounts eingerichtet und entsprechend parametrisiert werden. Während die Lernenden nur einen lesenden Zugriff auf das System haben, können Autoren auch schreibend zugreifen. Sie können neue Lehr-/Lernsysteme entwickeln oder bereits im System befindliche Systeme überarbeiten.
Falldaten, Nutzerdaten und Layouts werden in einer Datenbank gehalten und über eine Datenbankschnittstelle zugegriffen. Fallorientiertes Domänenwissen und Lehrbuchwissen könnte in einer Wissensbank abgelegt werden. Da zur Zeit allerdings keine sehr umfangreiche Wissensbasis vorhanden ist, soll das Domänenwissen ebenfalls in einem relationalen Datenbanksystem gespeichert werden. Eine detaillierte Beschreibung des konzeptuellen Datenbankschemas erfolgt in Kapitel 3.5.

3.4.2 Schichtenmodell

Client/Server-Applikationen, bei denen für die Datenspeicherung ein Datenbankmanagementsystem verwendet wird, basieren üblicherweise auf einem Zwei- bzw. Dreischichtenmodell (Two Tier Model bzw. Three Tier Model oder Multi Tier Model) [KLUTE 97]. Beim Zweischichtenmodell enthält die Clientschicht die gesamte Programmlogik einschließlich der Treiber für den Zugriff auf das DBMS in der Serverschicht. Man spricht in diesem Zusammenhang von sogenannten *Fat Clients*. Beim Dreischichtenmodell kommt eine zusätzliche Schicht (üblicherweise *Middleware* genannt) zum Einsatz. Dadurch wird die Clientschicht vollkommen unabhängig vom verwendeten DBMS. In der Clientschicht werden keinerlei Treiber für den Zugriff auf das DBMS mehr benötigt, wodurch diese gegenüber dem Zweischichtenmodell an Umfang verliert. die Datenbank kann sogar ersetzt werden, ohne daß Änderungen in der Clientschicht notwendig werden. Alle drei Schichten können auf unterschiedlichen Rechnern laufen.
Für ein Internet-basiertes Lehr-/Lernsystem kommt ein Zweischichtenmodell aus Performance-Gründen nicht in Frage (vergleichsweise umfangreicher Client, welcher auf den Client-Rechner übertragen werden muß). Das Dreischichtenmodell bietet hier unübersehbare Vorteile, auch wenn die Implementierung aufwendiger als beim Zweischichtenmodell ist. Um den Erfordernissen von Lehr-/Lernsystemen gerecht zu werden, die auch bei geringen Netzbandbreiten akzeptable Lade- und Antwortzeiten bieten sollen, wird sowohl die Client- als auch die Serverschicht des Dreischichtenmodells jeweils in drei Schichten aufgeteilt. Diese Unterteilung führt zu einer 7-Schichten-Architektur (siehe Abbildung 12).

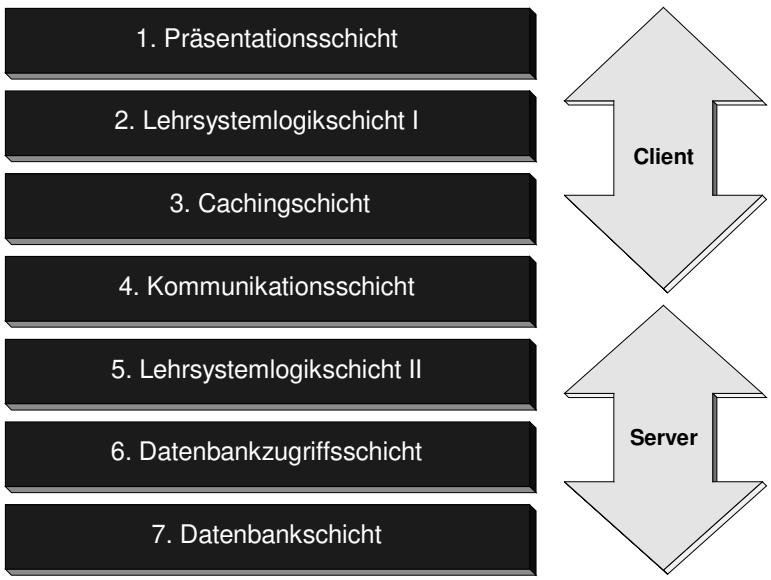

Abbildung 12: CAMPUS Schichtenmodell

Verbesserte Antwortzeiten des Systems werden dadurch erreicht, daß eine *Cachingschicht* (Schicht 3) eingeführt wird. Diese ermöglicht es, den Datenbankzugriff von den Nutzeraktivitäten abzukoppeln. Es können prospektiv Daten vom DBMS angefordert und in der *Cachingschicht* gespeichert werden. Werden die Daten jetzt gebraucht, liegen sie bereits auf dem Client vor und müssen nicht erst angefordert werden.
Auch die Aufteilung der *Lehrsystemlogikschicht* in zwei Teile (Schicht 2, Schicht 5) bringt Performanceverbesserungen. Der eine Teil wird auf dem Server angesiedelt, der andere

Teil auf dem Client. Dadurch kann eine Minimierung der Zahl der notwendigen Kommunikationsvorgänge sowie der Menge der zu übertragenden Daten zwischen Lehr-/Lernsystemclient und Lehr-/Lernsystemserver erreicht werden. Auf dem Server ist es sinnvoll Funktionen anzusiedeln, bei denen auf Basis von Datenbankabfrageergebnissen weitere Datenbankabfragen erstellt und durchgeführt werden. Die Auswahl geeigneter Lehr-/Lernfälle für einen bestimmten Nutzer aus den Fällen in der Datenbank ist ein Beispiel hierfür. Dabei muß u.a. berücksichtigt werden, wie viele und welche Lehr-/Lernfälle bereits erfolgreich bearbeitet wurden, welche nicht erfolgreich gelöst wurden und natürlich auch, in welchem Semester sich der Nutzer befindet.
Die *Präsentationsschicht* (Schicht 1) stellt die Schnittstelle zwischen Nutzer und System dar. Die *Kommunikationsschicht* (Schicht 4) ist für die Kommunikation zwischen Client und Server verantwortlich. Für deren Implementierung stehen verschiedene Basistechniken zur Verfügung. Da sie lediglich „Transportfunktionen" wahrnimmt, kann sie jederzeit durch eine andere Implementierung ersetzt werden, falls dies z.B. aus Performancegründen erforderlich ist.
Die *Datenbankzugriffsschicht* (Schicht 6) ist für den transparenten Zugriff der *Lehrsystemlogikschicht II* (Schicht 5) auf die Datenbank zuständig. Das verwendete Datenbankmanagementsystem kann dadurch nachträglich ohne Probleme ausgetauscht werden. In der *Datenbankschicht* (Schicht 7) ist schließlich das DBMS angesiedelt, in dem Domänendaten und Domänenwissen gespeichert sind.

3.4.3 Lehrmodell

Das Lehrmodell beschreibt, wie sich das System an seine Nutzer anpaßt und dadurch den Lernfortschritt optimal zu fördern versucht. Die einzelnen Parameter des Lehrmodells können von den Autoren auf Wunsch jederzeit geändert werden.

Im System sind die folgenden Adaptivitätsarten vorgesehen:

1. „Automatisches" Einblenden von Hilfetexten zur Systembedienung:
   ja ↔ nein
2. Allgemeine Fragen zur Fallbearbeitung:
   ja (Schwierigkeitsstufen: leicht, mittel, schwer) ↔ nein
3. Konkrete Fragen zu einem speziellen Lehr-/Lernfall:
   ja (Schwierigkeitsstufen: leicht, mittel, schwer) ↔ nein
4. Präsentations-/Interaktionsform
   Totale ↔ Vollständig ↔ Entscheidung ↔ Kompakt
5. Schwierigkeitsgrad von Lehr-/Lernfällen
   leicht ↔ mittel ↔ schwer
6. Rückmeldezeitpunkt
   sofort ↔ am Ende der Fallsimulation

Beim Öffnen eines Fensters wird einem Nutzer automatisch die zugehörige Systemhilfe eingeblendet, wenn dieses neu für ihn ist oder er dieses erst ein einziges mal gesehen hat. Die Hilfe enthält sowohl Anleitungen zur Programmbedienung als auch zur Fallbearbeitung. Defaultmäßig wird zu jedem Fenster zweimal die zugehörige Hilfeseite eingeblendet. Dieser Wert kann vom Autor jedoch geändert werden. Selbstverständlich können die Nutzer jederzeit durch Anklicken von Hilfe-Buttons kontextbezogene Hilfeseiten anzeigen lassen.

Im CAMPUS-System sind zwei disjunkte Mengen von Fragen enthalten. Zum einen gibt es allgemeine Fragen zur Fallbearbeitung, beispielsweise zu Labortests oder zu technischen Untersuchungsverfahren. Zum anderen sind spezielle Fragen zu den konkreten Lehr-/Lernfällen vorhanden. Diese Fragen dienen u.a. dazu, um die Nutzer bei der Fallbearbeitung zu unterstützen.

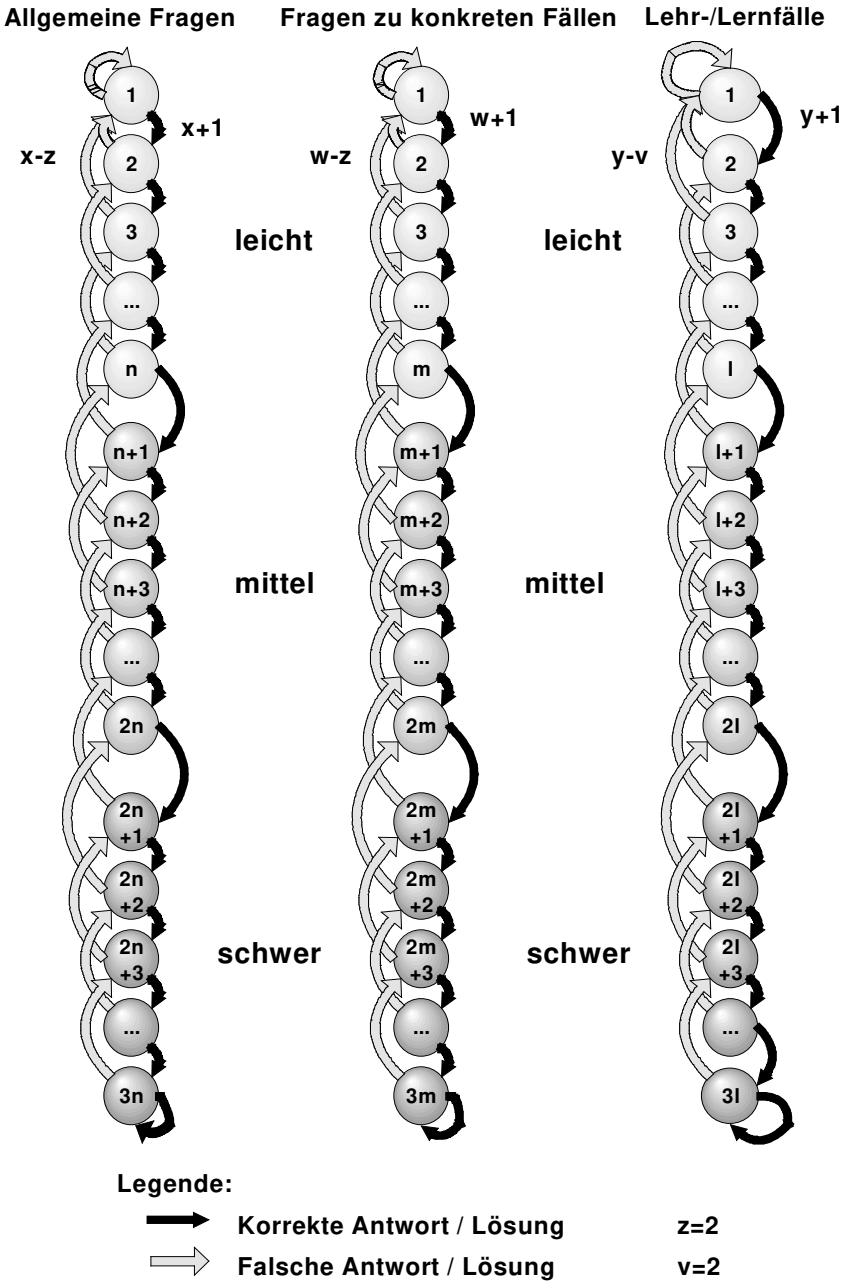

Abbildung 13: Adaption der Schwierigkeitsstufe von Fragen und Lehr-/Lernfällen

Alle Fragen im System sind einer von drei Schwierigkeitsstufen (leicht, mittel, schwer) zugeordnet. Die in einer konkreten Situation geeignete Schwierigkeitsstufe wird wie folgt ermittelt:

Auf Basis der Semesterzahl eines neuen Nutzers wird ein bestimmter „Ausgangszustand“ x := (Aktuelle Semesterzahl) für allgemeine Fragen und w := (Aktuelle Semesterzahl) für Fragen zu konkreten Lehr-/Lernfällen festgelegt. In den Zuständen $1 \leq x \leq$ n sowie $1 \leq w \leq$ m (bei Fragen zu konkreten Lehr-/Lernfällen) werden leichte Fragen, in den Zuständen $(n + 1) \leq x \leq 2n$ bzw. $(m+1) \leq w \leq 2m$ werden mittelschwere Fragen und in den Zuständen $(2n + 1)$ $x \leq 3n$ sowie $(2m + 1) \leq w \leq 3m$ werden schwere Fragen ausgewählt und angezeigt (siehe Abbildung 13). Die Parameter n, m und deren ganzzahlige Vielfache stehen für die Zustandsnummern, bei der zur nächsthöheren bzw. nächstniedrigeren Schwierigkeitsstufe bei allgemeinen Fragen bzw. bei Fragen zum konkreten Lehr-/Lernfall geschaltet wird. Diese Parameter können ebenfalls von den Autoren frei gewählt werden (Default-Werte: n = 10, m=10).
Wird eine gestellte Frage in der zur Verfügung stehenden Zeit korrekt beantwortet, so kommt der Nutzer in den nächsthöheren Zustand ($x := x + 1$ bzw. $w := w + 1$). Beantwortet er sie falsch, dann wird er um z Zustände (Default-Wert: z = 2) zurückgestuft ($x := x$ - z bzw. $w := w$ -z). Allgemeine Fragen und Fragen zu den einzelnen Lehr-/Lernfällen werden dabei grundsätzlich getrennt behandelt. Es kann also vorkommen, daß ein Nutzer *schwere* allgemeine Fragen gestellt bekommt, jedoch gleichzeitig *leichte* Fragen zum konkreten Lehr-/Lernfall. Allgemeine Fragen werden so lange gestellt, bis eine festgelegte Zahl an Fragen (Default-Wert: 5) mit höchstem Schwierigkeitsgrad nacheinander im höchstmöglichen Zustand (x = 3n) erfolgreich beantwortet worden sind. Dasselbe gilt für Fragen zu den konkreten Lehr-/Lernfällen. Auch hier werden die Fragen solange gestellt, bis im höchstmöglichen Zustand (w = 3m) eine festgelegte Anzahl von Fragen (Default-Wert: 10) korrekt beantwortet wurden.
Für die Präsentation von Lehr-/Lernfällen sind vier unterschiedliche Präsentations-/Interaktionsformen vorgesehen. Die Präsentations-/Interaktionsform *Totale* zeigt einen Lehr-/Lernfall in tabellarischer Form. Neben allen Untersuchungsergebnissen sind alle zu wählenden Verdachts- und Arbeitsdiagnosen sowie die anzuwendenden Therapieprinzipien enthalten. Die Präsentations-/Interaktionsform *Totale* ermöglicht es den Nutzern, den Aufbau der medizinischen Lehr-/Lernfälle in CAMPUS sowie die „Musterlösung“ des Autors kennenzulernen. Nachdem ein Student mehrere Lehr-/Lernfälle kennenlernen konnte (Default-Wert: 2), wechselt das System zur Präsentations-/Interaktionsform *Vollständig*. Hier muß der Nutzer jeden Schritt bei der Fallbearbeitung explizit wählen (siehe Abbildung 14) und erhält alle durchgeführten Untersuchungen (Anamnese, klinische Untersuchung, technische Untersuchungen, Laboruntersuchungen) sowie die dazugehörigen Ergebnisse in tabellarischer Form angezeigt. Auf Grundlage der Befundergebnisse muß er dann Verdachts- und Arbeitsdiagnosen stellen sowie sich entscheiden, welche Art der Betreuung jeweils notwendig ist (ambulant, stationär oder Intensivstation). Bei dieser Präsentations-/Interaktionsform müssen keine Entscheidungen über den Informationsbedarf getroffen werden. Durch die Präsentation der durchgeführten Untersuchungen nebst Ergebnissen in tabellarischer Form lernen die Studenten, welche Untersuchungen angefordert werden müssen, um bestimmte Diagnosen stellen zu können. Nachdem genügend Lehr-/Lernfälle hinreichend erfolgreich in der Präsentations-/Interaktionsform *Vollständig* bearbeitet worden sind, wechselt das CAMPUS-System zur Präsentations-/Interaktionsform *Entscheidung*. Bei dieser Präsentations-/Interaktionsform müssen die Nutzer zusätzlich zur Wahl von Verdachts- und Arbeitsdiagnosen auch darüber entscheiden, welche Informationen sie benötigen. Sie müssen also die Anamnese und klinische Untersuchung selbst durchführen und die notwendigen technischen Untersuchungen und Laboruntersuchungen anfordern. Wenn eine hinreichende Anzahl an Lehr-/Lernfällen in dieser Präsentations-/Interaktionsform erfolgreich bearbeitet wurde, dann wechselt das CAMPUS-System zur Präsentations-/Interaktionsform *Kompakt*. Die Untersuchungsergebnisse werden wie bei

der Präsentations-/Interaktionsform *Vollständig* in Listenform präsentiert. Jetzt werden allerdings nur noch die pathologischen Befunde angezeigt. Neben der Wahl von Verdachts- und Arbeitsdiagnosen und dem Treffen einer Entscheidung für eine bestimmte Betreuungsart müssen nun zusätzlich geeignete Therapieprinzipien gewählt werden. Der Zeitpunkt, wann das System die Präsentations-/Interaktionsform wechselt, hängt von verschiedenen Faktoren ab. Es werden eine fest vorgegebene Anzahl an Lehr-/Lernfällen in der Präsentations-/Interaktionsform *Totale* gezeigt (Default-Wert: 2). Der Wechsel zwischen den anderen Präsentations-/Interaktionsformen erfolgt, wenn bestimmte „Schwellwerte" hinreichend oft überschritten werden (s.u.).
Der Bearbeitungserfolg bei der Fallbearbeitung wird wie folgt ermittelt. Für alle vom Nutzer getroffene Entscheidungen (Diagnosenwahl, Anforderung von Untersuchungen usw.) werden die in Tabelle 4 aufgelisteten positiven bzw. negativen Punktwerte ermittelt und aufaddiert. Die aufaddierten Punktwerte werden durch den maximal erreichbaren Punktwert (abhängig von der Präsentations-/Interaktionsform) dividiert. Bei vollständig korrekter Fallbearbeitung ist der Fallbearbeitungserfolg also gleich 1.

Entscheidung	**Punktwert**
Korrekte Verdachtsdiagnose / Arbeitsdiagnose	+1
Fehlende Verdachtsdiagnose / Arbeitsdiagnose	-1
Vollkommen falsche Verdachtsdiagnose / Arbeitsdiagnose	-1
Zu spezielle Verdachtsdiagnose / Arbeitsdiagnose	-0,25
Zu allgemeine Verdachtsdiagnose / Arbeitsdiagnose	-0,25
Entscheidung über Akuttherapie (korrekt / falsch)	+1 / -1
Entscheidung über Betreuungsart (korrekt / falsch)	+1 / -1
Korrektes Therapieprinzip	+1
Fehlendes Therapieprinzip	-1
Falsches Therapieprinzip	-1
Kontraindiziertes Therapieprinzip	-2
Korrekte Anamnesefrage	+0,25
Unnötige Anamnesefrage	-0,25
Korrekte klinische Untersuchung	+0,25
Unnötige klinische Untersuchung	-0,25
Korrekte Anforderung einer techn. Untersuchung	+1
Unnötige Anforderung einer techn. Untersuchung	-1
Anforderung einer kontraindizierten techn. Untersuchung	-2
Korrekte Anforderung eines Labortests	+0,25
Unnötige Anforderung eines Labortests	-0,25
Anforderung eines kontraindizierten Labortests	-0,5
Entscheidung über Fallende (korrekt / falsch)	+1 / -1

Tabelle 4: Bewertungspunkte für Entscheidungen

In der Präsentations-/Interaktionsform *Vollständig* muß eine Mindestpunktzahl bei der Summe aller bisherigen Fallbearbeitungen von a Punkten (Default-Wert: a = 5) erreicht werden. Danach werden die Lehr-/Lernfälle in der Präsentations-/Interaktionsform *Entscheidung* präsentiert. Die Weiterschaltung zur Präsentations-/Interaktionsform Kompakt erfolgt, wenn ein Nutzer 2a Punkte erhalten hat.
Die Schwierigkeitsstufe von Lehr-/Lernfällen wird auf entsprechende Art festgestellt, wie die Schwierigkeitsstufe von Fragen (siehe Abbildung 13). Aus der Semesterzahl eines

neuen Nutzers wird dessen Anfangseinstufung errechnet (y := Aktuelle Semesterzahl). Für jeden hinreichend korrekt bearbeiteten Lehr-/Lernfall (Default-Schwellwert: s = 0,7) steigt er einen Zustand ($y := y + 1$), für jeden nicht korrekt gelösten Lehr-/Lernfall wird er v Zuständen (Default-Wert: v = 2) niedriger eingestuft ($y := y - v$).
Das System bietet zwei mögliche Zeitpunkte für Rückmeldungen. Anfänger erhalten stets eine sofortige Rückmeldung, wenn sie bei der Anforderungen bzw. Durchführung von Untersuchungen (Anamnese, klinische Untersuchung, technische Untersuchungen, Laboruntersuchungen) einen Fehler gemacht haben, entweder indem sie eine Untersuchung unnötig angefordert haben bzw. eine Untersuchung vergessen haben anzufordern. Fortgeschrittene Nutzer erhalten die Rückmeldung erst bei der Fallzusammenfassung und werden so bei der Fallbearbeitung nicht ständig unterbrochen. Die Umschaltung erfolgt automatisch, wenn der Nutzer schwere Lehr-/Lernfälle erreicht hat. Falsch gewählte Diagnosen oder Therapieprinzipien werden immer sofort angezeigt. Dadurch wird vermieden, daß ein Fallverlauf nicht mehr realistisch ist.
Die Zuordnung einzelner Fragen zu bestimmten Situationen beim Fallablauf oder zu bestimmten Knoten und Kapiteln im Lexikon erfolgt durch die Autoren. Die Zuordnungen werden in den Attributen *ObjektNr* und *Objektname* in den Objekttypen KNOTENZUORDNUNG und FRAGEZUORDNUNG abgelegt. Das boole'sche Attribut *inDiskus* des Objekttyps FRAGEZUORDNUNG zeigt an, ob die zugeordnete Frage in der Fallabschlußdiskussion am Ende einer Fallbearbeitung angezeigt werden soll oder nicht.
Die nachfolgende formale Beschreibung des Schemas ZUORDNUNG erfolgt mit der in Abbildung 8 vorgestellten Syntax zur ER-Schema-Deklaration.

```
schema ZUORDNUNG;

entityset KNOTENZUORDNUNG
(attributes:        knotenzuordnungNr        long,
                    objektNr                 long  not null,
                    objektname               char(50) not null,
identifier:         knotenzuordnungNr);

entityset FRAGEZUORDNUNG
(attributes:        fragezuordnungNr         long,
                    objektNr                 long  not null,
                    objektname               char(50) not null,
                    inDiskus                 boolean  not null,
identifier:         fragezuordnungNr);

relationship KNOTENZUORDNUNG_KNOTEN
(participants:      (KNOTENZUORDNUNG, (1,1)),
                    (KNOTEN, (0,n)));

relationship FRAGEZUORDNUNG_FRAGE
(participants:      (FRAGEZUORDNUNG, (1,1)),
                    (FRAGE, (0,n)));
```

3.4.4 Fallablaufmodell

Nachdem sich ein Nutzer beim System angemeldet und für die Bearbeitung eines Lehr-/Lernfalles entschieden hat, bekommt er einen Patienten mit dessen administrativen Daten und Vorgeschichte vorgestellt. Angezeigt werden Name, Geschlecht, Alter, Größe und Gewicht des Patienten sowie evtl. ein Bild, mit dem sich der Nutzer einen ersten optischen Eindruck vom Patienten verschaffen kann. Der weitere Ablauf hängt dann davon ab, in welche Adaptionskategorien der Nutzer vom System eingruppiert worden ist bzw. sich selbst eingruppiert hat. Prinzipiell muß der Nutzer zuerst eine gründliche Anamnese und eine gründliche klinische Untersuchung durchführen. Auf Basis von Anamnese und klinischer Untersuchung muß er sich danach für mindestens eine Verdachtsdiagnose entscheiden. Die Diagnosen können als Freitext eingegeben werden. Das System liefert daraufhin alle in Frage kommenden Diagnoseschlüssel nach ICD-10 [RIEDEL, HAAG et al. 98]. Daraus muß er die gewünschte(n) Diagnose(n) selektieren. Nach Wahl der Verdachtsdiagnose(n) muß er eine Entscheidung treffen, ob eine Akuttherapie erforderlich ist oder nicht. Falls ja, muß ein Therapieplan aufgestellt werden. Dem Nutzer wird eine Liste mit Therapieprinzipien eingeblendet, aus der er beliebig auswählen kann. Falls keine Akuttherapie notwendig ist kann er gleich die Entscheidung darüber treffen, ob eine ambulante, eine stationäre oder aber eine Behandlung auf der Intensivstation erforderlich ist. Nachdem diese Entscheidung getroffen wurde, müssen gezielt technische Untersuchungen und Laboruntersuchungen angefordert werden, um die Verdachtsdiagnose(n) zu bestätigen bzw. zu widerlegen. Technische Untersuchungen werden allerdings nur dann durchgeführt, wenn sie sinnvoll sind. Dies entspricht dem gängigen Procedere am Klinikum der Universität Heidelberg. Ein Arzt prüft dort, ob die Begründung für die Notwendigkeit einer Untersuchung ausreichend ist. Bei der Anforderung von Laboruntersuchungen findet eine solche Prüfung zur Zeit noch nicht statt. Nach Anforderung aller Untersuchungen werden die Untersuchungsergebnisse vom System präsentiert. Anschließend muß der Nutzer mindestens eine Arbeitsdiagnose stellen. Das Procedere ist das Gleiche, wie bei der Wahl der Verdachtsdiagnosen. Auf Basis der Arbeitsdiagnosen muß er erneut Therapieprinzipien auswählen. Vor der Anforderung von Untersuchungen zur Kontrolle des Therapieverlaufs muß er wiederum bzgl. der Art der Betreuung (ambulant, stationär, Intensivstation) eine Entscheidung treffen. Nach Einsicht in die Ergebnisse der angeforderten Untersuchungen und der Präsentation des klinischen Verlaufs durch das System ist eine Entscheidung zu treffen, ob der Fall abgeschlossen werden kann oder nicht. Im ersten Fall muß der Nutzer eine Prognose abgeben und er erhält eine kurze Zusammenfassung des Lehr-/Lernfalles angezeigt, im anderen Fall kann er seine Arbeitsdiagnose(n) bzw. den Therapieplan abändern oder aber neue technische Untersuchungen bzw. Laboruntersuchungen anfordern. Der Lehr-/Lernfall wird dadurch fortgesetzt (siehe Abbildung 14).
Das Fallablaufmodell wurde in enger Kooperation mit der Universitätskinderklinik Heidelberg entwickelt. Seine Eignung für die Infektiologie konnte in einem gemeinsamen Projekt mit dem Hygiene-Institut der Ruprecht-Karls-Universität Heidelberg festgestellt werden.

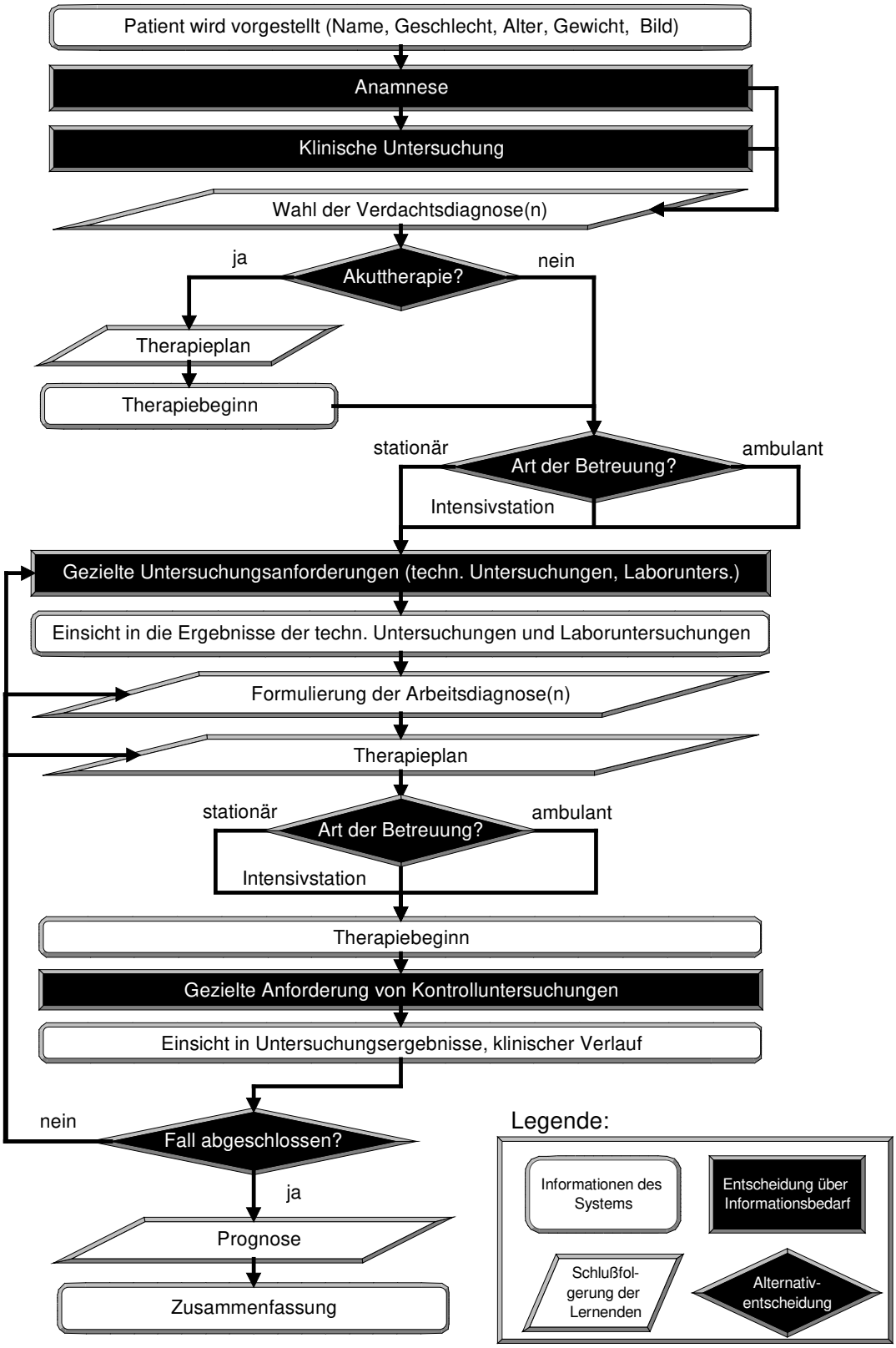

Abbildung 14: Fallablaufmodell

3.5 Konzeptuelle Modellierung

In Kapitel 2.9 wurde ein Drei-Ebenen-Modell der Datenmodellierung vorgestellt. Das Modell sieht vor, daß zuerst ein von Implementierungsaspekten vollkommen unabhängiges *konzeptuelles Modell* entworfen wird. Das konzeptuelle Modell des CAMPUS-Systems besteht aus mehreren Teilmodellen, welche in den nachfolgenden Unterkapiteln vorgestellt werden. In der Verbalbeschreibung der einzelnen Modelle werden nur diejenigen Attribute erläutert, die möglicherweise nicht intuitiv verständlich sind. Objekttypen, welche in mehreren Modellen vorkommen, sind in den grafischen Darstellungen nur einmal vollständig inklusive aller Attribute enthalten und werden nur einmal formal beschrieben. Die formale Beschreibung erfolgt mit der in Abbildung 8 beschriebenen Syntax zur Deklaration von ER-Schemata. Integritätsbedingungen sind mit dem in Abbildung 9 gezeigten Entity-Relationship Calculus spezifiziert.

3.5.1 Normalbefundemodell

Normalbefunde eignen sich hervorragend für die Wiederverwendung in anderen Lehr-/Lernsystemen. Mit dem Entwurf eines Normalbefundemodells (siehe Abbildung 15) wird diesem Aspekt Rechnung getragen. Durch das Modell ist es möglich, Normalbefunde ohne zusätzlichen Aufwand beliebig wiederzuverwenden. So können Autoren bei der Lehr-/Lernfallerstellung jederzeit auf vorhandene Normalbefunde zurückgreifen und müssen diese nicht für jedes Lehr-/Lernsystem neu erstellen. Das Lehrsubsystem besitzt durch die Verfügbarkeit von Normalbefunden die Möglichkeit unauffällige Befunde automatisch zu generieren, wenn ein Befundergebnis zu einer angeforderten Untersuchung nicht vorliegt. Die Nutzer können jederzeit die im System enthaltenen Normalbereiche und Normalbefunde einsehen, wenn sie sich hinsichtlich der Interpretation eines Untersuchungsergebnisses nicht sicher sind.
Unauffällige Antworten (Objekttyp ANAMNESENORMALANTWORT) auf Anamnesefragen (Objekttyp ANAMNESEFRAGE) sind abhängig vom Geschlecht (GESCHLECHT) und vom Alter (ALTERSSTUFE) eines Patienten. Unauffällige Befunde (KLIN-NORMALBEFUND) bei Durchführung der klinischen Untersuchungsarten (KLIN-UNTERSUCHUNGSART) Inspektion, Palpation, Auskultation und Perkussion sind ebenfalls vom Geschlecht und Alter des Patienten abhängig. Darüber hinaus aber auch von der Körperregion (KOERPERREGION), an der die Untersuchung durchgeführt worden ist. Dasselbe gilt für Normalbefunde (TECHNNORMALBEFUND) beim Einsatz technischer Untersuchungsverfahren (TUNTERSVERFAHREN). Normalbefunde (LABORNORMALBEFUND) bei Labortests (LABORTEST) schließlich sind vom Geschlecht und Alter des Patienten sowie von der verwendeten Probenart (PROBENART) abhängig. Fachgebiete (FACHGEBIET) und Körperregionen sind hierarchisch gegliedert. Boole'sche Attribute im Objekttyp KOERPERREGION (*inspektion*, *palpation*, *auskultation*, *perkussion*, *tUntersuchung*) geben an, bei welchen klinischen Untersuchungsarten bzw. technischen Untersuchungsverfahren eine Körperregion prinzipiell als Untersuchungsgebiet dienen kann. Anamnesefragen ist jeweils mindestens ein Fachgebiet zugeordnet. Es können also fachgebietsspezifische Anamnesefragen angegeben werden. Zu jedem Anamnesetyp (ANAMNESETYP) existiert mindestens eine Anamnesefrage.

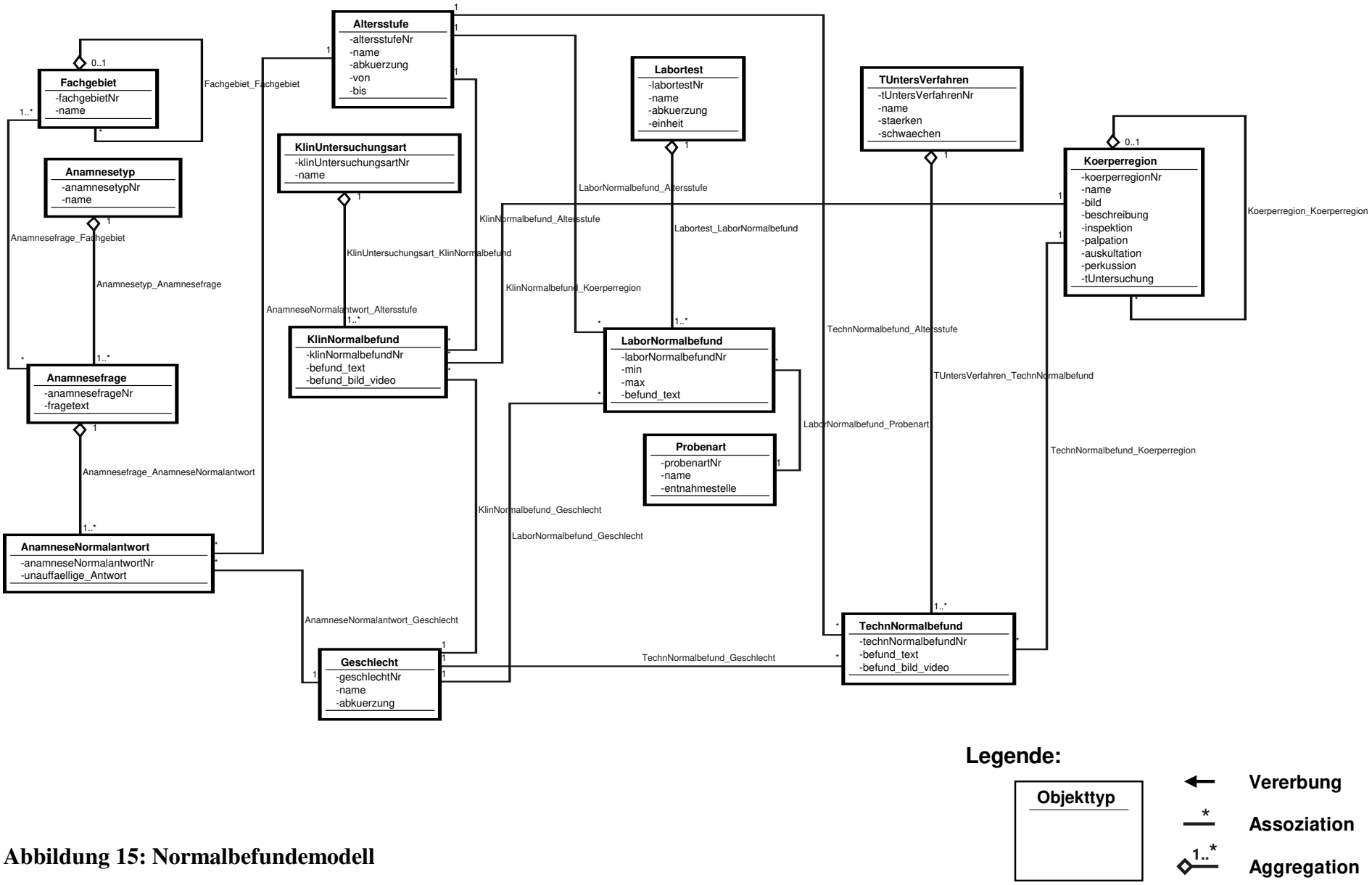

Abbildung 15: Normalbefundemodell

schema NORMALBEFUNDEMODELL;

entityset ANAMNESETYP
(attributes: *anamnesetypNr* l o n g ,
name c h a r (50) not null,
identifier: *anamnesetypNr);*

entityset ANAMNESEFRAGE
(attributes: *anamnesefrageNr* l o n g ,
frage_text c h a r (2000) not null,
identifier: *anamnesefrageNr);*

entityset ANAMNESENORMALANTWORT
(attributes: *anamneseNormalantwortNr* l o n g ,
unauffaellige_Antwort c h a r (2000) not null,
identifier: *anamneseNormalantwortNr);*

entityset KLINUNTERSUCHUNGSART
(attributes: *klinUntersuchungsartNr* l o n g ,
name c h a r (50) not null,
identifier: *klinUntersuchungsartNr);*

entityset KLINNORMALBEFUND
(attributes: *klinNormalbefundNr* l o n g ,
befund_text c h a r (2000) not null,
befund_bild_video c h a r (50),
identifier: *klinNormalbefundNr);*

entityset TUNTERSVERFAHREN
(attributes: *tUntersVerfahrenNr* l o n g ,
name c h a r (50) not null,
staerken c h a r (2000),
schwaechen c h a r (2000),
identifier: *tUntersVerfahrenNr);*

entityset TECHNNORMALBEFUND
(attributes: *technNormalbefundNr* l o n g ,
befund_text c h a r (2000) not null,
befund_bild_video c h a r (50),
identifier: *technNormalbefundNr);*

entityset LABORTEST
(attributes: *labortestNr* l o n g ,
name c h a r (50) not null,
abkuerzung c h a r (20),
einheit c h a r (10),
identifier: *labortestNr);*

entityset LABORNORMALBEFUND
(attributes: *laborNormalbefundNr* long,
befund_text char(2000),
min real,
max real,
identifier: *laborNormalbefundNr);*

$(\forall \boldsymbol{l})(\text{LABORNORMALBEFUND}(\boldsymbol{l}) \rightarrow (\boldsymbol{l}[min] == \text{null} \vee \boldsymbol{l}[max] == \text{null}) \vee (\boldsymbol{l}[min] \leq \boldsymbol{l}[max]))$
$(\forall \boldsymbol{l})(\text{LABORNORMALBEFUND}(\boldsymbol{l}) \rightarrow \neg(\boldsymbol{l}[min] == \text{null} \wedge \boldsymbol{l}[max] == \text{null} \wedge \boldsymbol{l}[befund\_text] == \text{null}))$
$(\forall \boldsymbol{l})(\text{LABORNORMALBEFUND}(\boldsymbol{l}) \rightarrow \neg(\neg(\boldsymbol{l}[min] == \text{null}) \wedge \neg(\boldsymbol{l}[max] == \text{null}) \wedge \neg(\boldsymbol{l}[befund\_text] == \text{null}))$

entityset ALTERSSTUFE
(attributes: *altersstufeNr* long,
name char(20) not null,
abkuerzung char(5),
von integer not null,
bis integer not null,
identifier: *altersstufeNr);*

entityset GESCHLECHT
(attributes: *geschlechtNr* long,
name char(10) not null,
abkuerzung char(5) not null,
identifier: *geschlechtNr);*

entityset PROBENART
(attributes: *probenartNr* long,
name char(50) not null,
entnahmestelle char(50) not null,
identifier: *probenartNr);*

entityset KOERPERREGION
(attributes: *koerperregionNr* long,
name char(50) not null,
bild char(50) not null,
beschreibung char(2000),
inspektion boolean not null,
palpation boolean not null,
auskultation boolean not null,
perkussion boolean not null,
tUntersuchung boolean not null,
identifier: *koerperregionNr);*

```
entityset FACHGEBIET
(attributes:        fachgebietNr                long,
                    name                        char(50) not null,
identifier:         fachgebietNr);

relationship ANAMNESENORMALANTWORT_ALTERSSTUFE
(participants:      (ANAMNESENORMALANTWORT, (1,1)),
                    (ALTERSSTUFE, (0,n)));

relationship KLINNORMALBEFUND_ALTERSSTUFE
(participants:      (KLINNORMALBEFUND, (1,1)),
                    (ALTERSSTUFE, (0,n)));

relationship TECHNNORMALBEFUND_ALTERSSTUFE
(participants:      (TECHNNORMALBEFUND, (1,1)),
                    (ALTERSSTUFE, (0,n)));

relationship LABORNORMALBEFUND_ALTERSSTUFE
(participants:      (LABORNORMALBEFUND, (1,1)),
                    (ALTERSSTUFE, (0,n)));

relationship ANAMNESENORMALANTWORT_GESCHLECHT
(participants:      (ANAMNESENORMALANTWORT, (1,1)),
                    (GESCHLECHT, (0,n)));

relationship KLINNORMALBEFUND_GESCHLECHT
(participants:      (KLINNORMALBEFUND, (1,1)),
                    (GESCHLECHT, (0,n)));

relationship TECHNNORMALBEFUND_GESCHLECHT
(participants:      (TECHNNORMALBEFUND, (1,1)),
                    (GESCHLECHT, (0,n)));

relationship LABORNORMALBEFUND_GESCHLECHT
(participants:      (LABORNORMALBEFUND, (1,1)),
                    (GESCHLECHT, (0,n)));

relationship KLINNORMALBEFUND_KOERPERREGION
(participants:      (KLINNORMALBEFUND, (1,1)),
                    (KOERPERREGION, (0,n)));

relationship TECHNNORMALBEFUND_ KOERPERREGION
(participants:      (TECHNNORMALBEFUND, (1,1)),
                    (KOERPERREGION, (0,n)));

relationship LABORNORMALBEFUND_PROBENART
(participants:      (LABORNORMALBEFUND, (1,1)),
                    (PROBENART, (0,n)));

relationship ANAMNESEFRAGE _FACHGEBIET
```

(participants: (ANAMNESEFRAGE, (1,n)),
(FACHGEBIET, (0,n)));

wholepartassociation ANAMNESEFRAGE_ANAMNESENORMALANTWORT
(participants: (ANAMNESEFRAGE, (1,n)),
(ANAMNESENORMALANTWORT, (1,1)));

wholepartassociation ANAMNESETYP_ANAMNESEFRAGE
(participants: (ANAMNESETYP, (1,n)),
(ANAMNESEFRAGE, (1,1)));

wholepartassociation KLINUNTERSUCHUNGSART_KLINNORMALBEFUND
(participants: (KLINUNTERSUCHUNGSART, (1,n)),
(KLINNORMALBEFUND, (1,1)));

wholepartassociation LABORTEST_LABORNORMALBEFUND
(participants: (LABORTEST, (1,n)),
(LABORNORMALBEFUND, (1,1)));

wholepartassociation TECHNUNTERSVERFAHREN_TECHNNORMALBEFUND
(participants: (TECHNUNTERSVERFAHREN, (1,n)),
(TECHNNORMALBEFUND, (1,1)));

wholepartassociation KOERPERREGION_KOERPERREGION
(participants: (KOERPERREGION, UEBERGEORDNE-
TE_KOERPERREGION, (0,n)),
(KOERPERREGION, UNTERGEORDNE-
TE_KOERPERREGION, (0,1)));

wholepartassociation FACHGEBIET_FACHGEBIET
(participants: (FACHGEBIET, UEBERGEORDNETES_FACHGEBIET,
(0,n)),
(FACHGEBIET, UNTERGEORDNETES_FACHGEBIET,
(0,1)));

3.5.2 Fallmodell

Das Fallmodell (siehe Abbildung 16) beschreibt die Struktur eines medizinischen Lehr-/Lernfalles. Zuerst war angedacht, aus den bereits existierenden Ansätzen für die elektronische Krankenakte [VAN BEMMEL, MCCRAY (eds.) 95; SAFRAN, RIND et al. 96] einen Geeigneten auszuwählen und für die Speicherung von Lehr-/Lernfällen zu verwenden. Leider gibt es bei den elektronischen Krankenakten bisher noch keine allgemein akzeptierte Lösung und alle betrachteten Ansätze stellten sich als nicht geeignet heraus. Daraufhin wurde ein eigenes Fallmodell entwickelt. Durch das vorgestellte Fallmodell wird es möglich, Lehr-/Lernfälle mit unterschiedlichen Präsentations-/Interaktionsformen (siehe Kapitel 3.4.3) anzeigen zu lassen und bearbeiten zu können. Dies stellt einen wichtigen Unterschied zu dem in Kapitel 2.3.3 vorgestellten CASUS-System dar.
Ein Lehr-/Lernfall (Objekttyp FALL) besteht aus den bei einem Patienten durchgeführten Untersuchungen samt Untersuchungsergebnissen. Jeder Lehr-/Lernfall besitzt genau eine Anamnese (Objekttyp ANAMNESE) [DAHMER 94], genau eine klinische Untersuchung (KLINUNTERSUCHUNG) [MÜLLER, SEIFERT 89], eine beliebige Anzahl Laboruntersuchungen (LABORUNTERSUCHUNG) sowie eine beliebige Anzahl technischer Untersuchungen (TECHNUNTERSUCHUNG). Die Anamnese besteht aus mindestens einer Anamneseantwort (ANAMNESEANTWORT), die klinische Untersuchung aus mindestens einem Untersuchungsergebnis (KUNTERSERGEBNIS). Zu einer Technischen Untersuchung gehört genau ein Untersuchungsbefund (TUNTERSERGEBNIS). Zu einer Laboruntersuchung gehört mindestens ein Ergebnis (LABORTESTERGEBNIS) eines Labortests (LABORTEST). Ein Lehr-/Lernfall beinhaltet außerdem mindestens eine Diagnosestellung (DIAGNOSESTELLUNG) und mindestens eine Therapie (THERAPIE). Er besitzt mindestens einen klinischen Verlauf (KLINVERLAUF) und genau eine Prognose (PROGNOSE). Ihm ist mindestens ein Fachgebiet (FACHGEBIET) zugeordnet. Einer Diagnosestellung ist mindestens eine Diagnose (DIAGNOSE) zugeordnet, einer Therapie mindestens ein Therapieprinzip (THERAPIEPRINZIP). Therapieprinzipien sind allgemeine Therapiegrundsätze (z.B. Schmerztherapie) ohne Beschreibung der genauen Durchführung.
Der Objekttyp FALL besitzt u.a. die Attribute *einleitungstext, zusammenfassung* und *beschreibung*. Im Attribut *einleitungstext* ist kurz die Ausgangssituation zu Beginn der Fallsimulation beschrieben. Das Attribut *zusammenfassung* enthält eine Fallbesprechung, die nach Abschluß der Fallsimulation angezeigt wird und alle wesentlichen Aspekte des Lehr-/Lernfalles zusammenfaßt und erläutert. Im Attribut *beschreibung* ist der Lehr-/Lernfall skizziert. Anwender können damit auf Wunsch selbst die sie interessierenden Lehr-/Lernfälle aus der Falldatenbank auswählen. Das Attribut *eingangssymptom* bei Anamneseantwort und dem Ergebnis der klinischen Untersuchung (Objekttyp KUNTERSERGEBNIS) zeigt an, ob der Patient ohne Nachfrage des Arztes das entsprechende Symptom geschildert hat. Das Attribut *kontrolluntersuchung* dient bei den Objekten vom Typ LABORUNTERSUCHUNG und TECHUNTERSUCHUNG dazu, um zu vermerken, ob eine Untersuchung zur Diagnosestellung oder zur Kontrolle des Therapieverlaufs durchgeführt wurde. Das Attribut *diagnoseText* im Objekttyp DIAGNOSE ist zusätzlich zu den Drei- bzw. Vierstellertiteln der ICD-10 enthalten, weil diese nicht immer dem lokal üblichen Sprachgebrauch der Mediziner entsprechen. Das Attribut *therapie_text* im Objekttyp THERAPIE enthält die detaillierte Therapiebeschreibung, die im angegebenen Zeitraum (Attribute *von*, *bis*) beim Patienten durchgeführt worden ist. Das Attribut *betreuungsart* gibt Auskunft darüber, ob im angegebenen Zeitraum der Patient ambulant, stationär oder auf der Intensivstation betreut worden ist. Das boole'sche Attribut *akuttherapie* zeigt an,

ob es sich bei der durchgeführten Therapie um eine Akuttherapie handelt. Das Attribut *therapieart* im Objekttyp THERAPIE_THERAPIEPRINZIP enthält die Information, ob es sich um eine primäre, eine supportive oder aber eine kontraindizierte Therapie handelt. Kontraindizierte Therapien dürfen auf keinen Fall durchgeführt werden.

schema FALLMODELL;

entityset FALL
(attributes: *patientId* long,
geschlecht char(9) not null,
vorname char(20) not null,
nachname char(30) not null,
lebensalter integer not null,
gewicht integer not null,
groesse integer not null,
bild char(50),
beschreibung char(2000) not null,
einleitungstext char(2000) not null,
zusammenfassung char(2000) not null,
erstelldatum date not null,
schwierigkeitsstufe char(50) not null,
identifier: *patientId);*

entityset ANAMNESE
(attributes: *anamneseNr* long,
datum date not null,
zusammenfassung char(2000) not null,
identifier: *anamneseNr);*

entityset KLINUNTERSUCHUNG
(attributes: *klinUntersuchungNr* long,
datum date not null,
zusammenfassung char(2000) not null,
identifier: *klinUntersuchungNr);*

entityset TECHNUNTERSUCHUNG
(attributes: *techn UntersuchungNr* long,
anforderung_datum date not null,
befund_datum date not null,
kontrolluntersuchung boolean not null,
identifier: *techn UntersuchungNr);*

$(\forall \boldsymbol{t})(\text{TECHNUNTERSUCHUNG}(\boldsymbol{t}) \rightarrow \boldsymbol{t}[\textit{anforderung\_datum}] \leq \boldsymbol{t}[\textit{befund\_datum}])$

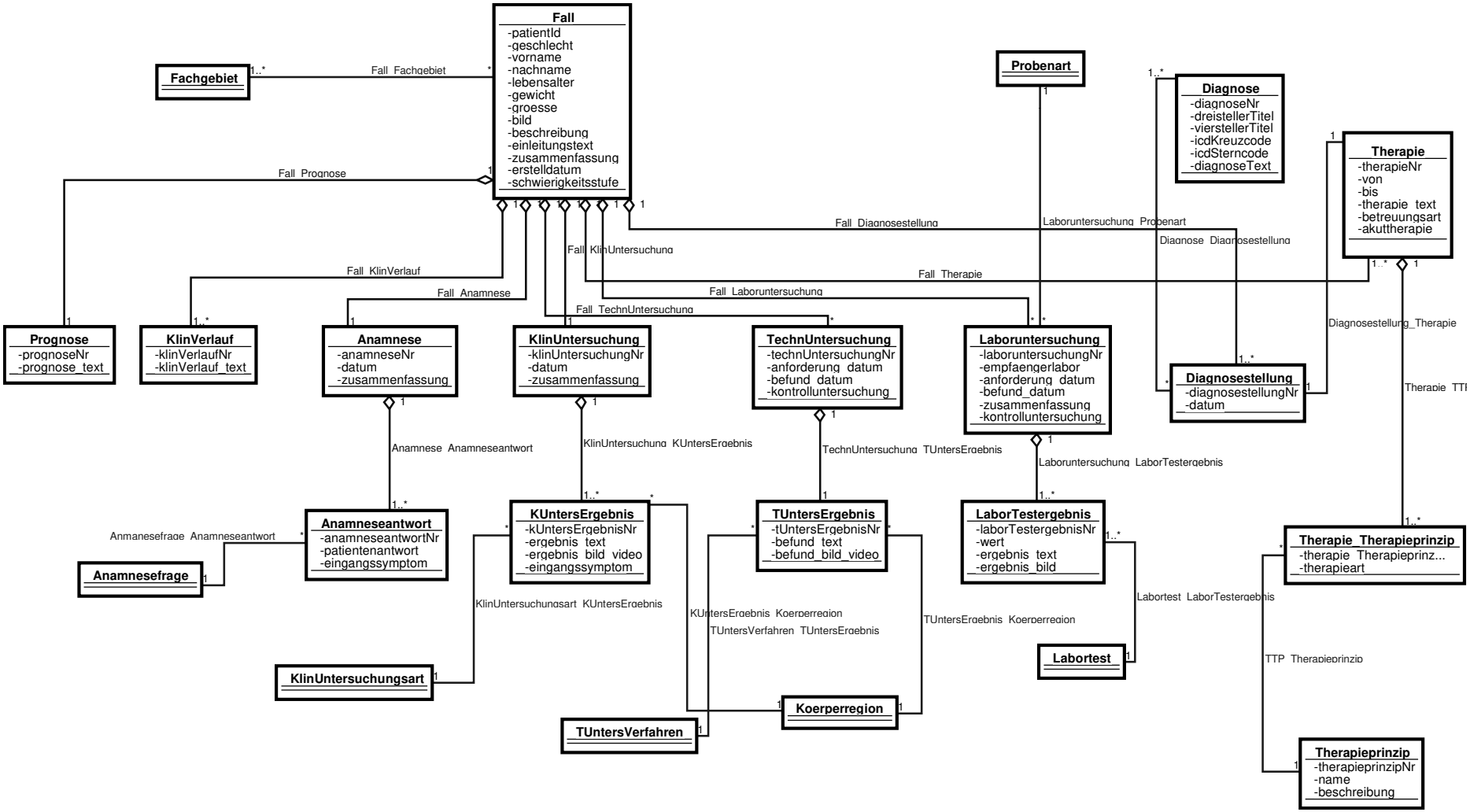

Abbildung 16: Fallmodell

entityset LABORUNTERSUCHUNG
(attributes: *laboruntersuchungNr* long,
empfaengerlabor char(50) not null,
anforderung_datum date not null,
befund_datum date not null,
zusammenfassung char(2000),
kontrolluntersuchung boolean not null,
identifier: *laboruntersuchungNr);*

$(\forall \boldsymbol{l})$(LABORUNTERSUCHUNG($\boldsymbol{l}$) $\rightarrow$ $\boldsymbol{l}$[*anforderung_datum*] $\leq$ $\boldsymbol{l}$[*befund_datum*])

entityset DIAGNOSESTELLUNG
(attributes: *diagnosestellungNr* long,
datum date not null,
identifier: *diagnosestellungNr);*

entityset THERAPIE
(attributes: *therapieNr* long,
von date not null,
bis date not null,
therapie_text char(2000) not null,
betreuungsart char(40) not null,
akuttherapie boolean,
identifier: *therapieNr);*

$(\forall \boldsymbol{t})$(THERAPIE($\boldsymbol{t}$) $\rightarrow$ $\boldsymbol{t}$[*von*] $\leq$ $\boldsymbol{t}$[*bis*]
$(\forall \boldsymbol{t})$(THERAPIE($\boldsymbol{t}$) $\rightarrow$ $\boldsymbol{t}$[*betreuungsart*] = “ambulant“ $\vee$ $\boldsymbol{t}$[*betreuungsart*] = “stationaer“ $\vee$ $\boldsymbol{t}$[*betreuungsart*] = “Intensivstation“)

entityset THERAPIE_THERAPIEPRINZIP
(attributes: *therapie_therapieprinzipNr* long,
therapieart char(50) not null,
identifier: *therapie_therapieprinzipNr);*

$(\forall \boldsymbol{t})$(THERAPIE_THERAPIEPRINZIP($\boldsymbol{t}$) $\rightarrow$ $\boldsymbol{t}$[*therapieart*] = “primär“ $\vee$ $\boldsymbol{t}$[*therapieart*] = “supportiv“ $\vee$ $\boldsymbol{t}$[*therapieart*] = “kontraindiziert“)

entityset THERAPIEPRINZIP
(attributes: *therapieprinzipNr* long,
name char(150) not null,
beschreibung char(2000),
identifier: *therapieprinzipNr);*

entityset PROGNOSE
(attributes: *prognoseNr* long,
prognose_text char(2000) not null,
identifier: *prognoseNr);*

entityset KLINVERLAUF
(attributes: *klinVerlaufNr* long,
klinVerlauf_text char(2000) not null,
identifier: *klinVerlaufNr);*

entityset ANAMNESEANTWORT
(attributes: *ananmneseantwortNr* long,
patientenantwort char(2000) not null,
eingangssymptom boolean not null,
identifier: *anamneseantwortNr);*

entityset KUNTERSERGEBNIS
(attributes: *kUntersErgebnisNr* long,
ergebnis_text char(2000) not null,
ergebnis_bild_video char(50),
eingangssymptom boolean not null,
identifier: *kUntersErgebnisNr);*

entityset TUNTERSERGEBNIS
(attributes: *tUntersErgebnisNr* long,
befund_text char(2000) not null,
befund_bild_video char(50),
identifier: *tUntersErgebnisNr);*

entityset LABORTESTERGEBNIS
(attributes: *laborTestergebnisNr* long,
wert double,
ergebnis_text char(2000),
ergebnis_bild char(50),
identifier: *laborTestergebnisNr);*

$(\forall \boldsymbol{l})(\text{LABORTESTERGEBNIS}(\boldsymbol{l}) \rightarrow \neg(\boldsymbol{l}[\textit{ergebnis\_text}] == \text{null} \wedge \boldsymbol{l}[\textit{wert}] == \text{null}) \wedge \neg(\neg(\boldsymbol{l}[\textit{ergebnis\_text}] == \text{null}) \wedge \neg(\boldsymbol{l}[\textit{wert}] == \text{null})))$

entityset DIAGNOSE
(attributes: *diagnoseNr* long,
dreistellerTitel char(255) ,
vierstellerTitel char(255),
icdKreuzcode char(5) not null,
icdSterncode char(5),
diagnoseText char(255) not null,
identifier: *diagnoseNr);*

wholepartassociation FALL_PROGNOSE
(participants: (FALL, (1,1)),
(PROGNOSE, (1,1)));

```
wholepartassociation FALL_KLINVERLAUF
(participants:          (FALL, (1,n)),
                        (KLINVERLAUF, (1,1)));

wholepartassociation FALL_ANAMNESE
(participants:          (FALL, (1,1)),
                        (ANAMNESE, (1,1)));

wholepartassociation FALL_KLINUNTERSUCHUNG
(participants:          (FALL, (1,1)),
                        (KLINUNTERSUCHUNG, (1,1)));

wholepartassociation FALL_TECHNUNTERSUCHUNG
(participants:          (FALL, (0,n)),
                        (TECHNUNTERSUCHUNG, (1,1)));

wholepartassociation FALL_LABORUNTERSUCHUNG
(participants:          (FALL, (0,n)),
                        (LABORUNTERSUCHUNG, (1,1)));

wholepartassociation FALL_DIAGNOSESTELLUNG
(participants:          (FALL, (1,n)),
                        (DIAGNOSESTELLUNG, (1,1)));

wholepartassociation FALL_THERAPIE
(participants:          (FALL, (1,n)),
                        (THERAPIE, (1,1)));

wholepartassociation ANAMNESE_ANAMNESEANTWORT
(participants:          (ANAMNESE, (1,n)),
                        (ANAMNESEANTWORT, (1,1)));

wholepartassociation KLINUNTERSUCHUNG_KUERGEBNIS
(participants:          (KLINUNTERSUCHUNG, (1,n)),
                        (KUNTERSERGEBNIS, (1,1)));

wholepartassociation TECHNUNTERSUCHUNG_TUNTERSERGEBNIS
(participants:          (TECHNUNTERSUCHUNG, (1,1)),
                        (TUNTERSERGEBNIS, (1,1)));

wholepartassociation LABORUNTERSUCHUNG_LABORTESTERGEBNIS
(participants:          (LABORUNTERSUCHUNG, (1,n)),
                        (LABORTESTERGEBNIS, (1,1)));

wholepartassociation THERAPIE_TTP
(participants:          (THERAPIE, (1,n)),
                        (THERAPIE_THERAPIEPRINZIP, (1,1)));
```

```
relationship FALL_FACHGEBIET
(participants:          (FALL, (1,n)),
                        (FACHGEBIET, (0,n)));

relationship ANAMNESEFRAGE_ANAMNESEANTWORT
(participants:          (ANAMNESEFRAGE, (0,n)),
                        (ANAMNESEANTWORT, (1,1)));

relationship KLINUNTERSUCHUNGSART_KUNTERSERGEBNIS
(participants:          (KLINUNTERSUCHUNGSART, (0,n)),
                        (KUNTERSERGEBNIS, (1,1)));

relationship TUNTERSVERFAHREN_TUNTERSERGEBNIS
(participants:          (TUNTERSVERFAHREN, (0,n)),
                        (TUNTERSERGEBNIS, (1,1)));

relationship LABORTEST_LABORTESTERGEBNIS
(participants:          (LABORTEST, (0,n)),
                        (LABORTESTERGEBNIS, (1,1)));

relationship DIAGNOSE_DIAGNOSESTELLUNG
(participants:          (DIAGNOSE, (0,n)),
                        (DIAGNOSESTELLUNG, (1,n)));

relationship DIAGNOSESTELLUNG_THERAPIE
(participants:          (DIAGNOSESTELLUNG, (1,1)),
                        (THERAPIE, (1,1)));

relationship TTP_THERAPIEPRINZIP
(participants:          (THERAPIE_THERAPIEPRINZIP, (1,1)),
                        (THERAPIEPRINZIP, (0,n)));

relationship LABORUNTERSUCHUNG_PROBENART
(participants:          (LABORUNTERSUCHUNG, (1,1)),
                        (PROBENART, (0,n)));

relationship KUNTERSERGEBNIS_KOERPERREGION
(participants:          (KUNTERSERGEBNIS, (1,1)),
                        (KOERPERREGION, (0,n)));

relationship TUNTERSERGEBNIS_KOERPERREGION
(participants:          (TUNTERSERGEBNIS, (1,1)),
                        (KOERPERREGION, (0,n)));
```

3.5.3 Fallwissensmodell

Das Fallwissensmodell (siehe Abbildung 17) beschreibt die Struktur des im CAMPUS-System vorhandenen fallbezogenen Wissens. Autoren müssen für alle Untersuchungsergebnisse angeben, ob das Ergebnis unauffällig ist, ob es sich beim Untersuchungsergebnis um ein Leitsymptom handelt und ob die zugehörige Durchführung der Untersuchung überhaupt notwendig war (boole'sche Attribute *normal*, *leitsymptom* und *notwendig* in den Objekttypen AANTWORTBEURTEILUNG, KUERGEBNISBEURTEILUNG, TUERGEBNISBEURTEILUNG und LUERGEBNISBEURTEILUNG).
Alle weiteren Angaben sind optional. Das Fallwissensmodell ermöglicht es den Autoren, auf Wunsch auch differentialdiagnostisches (heuristisches) Wissen angeben zu können. Die Funktionsfähigkeit des Lehr-/Lernsystems ist aber, im Gegensatz zu Intelligenten Tutoriellen Systemen, nicht vom Vorhandensein diesen Wissens abhängig. Falls ein Autor durch Eingabe differentialdiagnostischen Wissens zusätzliche Arbeit in die Erstellung eines Lehr-/Lernfalles investiert, können Nutzer bei der Fallbearbeitung dieses Wissen abrufen bzw. das System kann bei Wahl von falschen oder korrekten Verdachts- und Arbeitsdiagnosen detailliertere Rückmeldungen geben.
Ein Objekt vom Typ DIAGNOSE (Verdachts- oder Arbeitsdiagnose) kann mit den unterschiedlichen Untersuchungsergebnissen der Anamnese (Objekttyp ANAMNESEANTWORT), klinischen Untersuchung (KUNTERSERGEBNIS), technischen Untersuchungen (TUNTERSERGEBNIS) und Laboruntersuchungen (LABORTESTERGEBNIS) in Beziehung stehen (über die Objekttypen DIAGANAMNESEANTWORT, DIAGKUNTERSERGEBNIS, DIAGTUNTERSERGEBNIS und DIAGLABORTESTERGEBNIS). Es können positive bzw. negative Evidenzen (Attribut *evidenz*) angeben werden. Die möglichen Evidenzkategorien sind in Kapitel 4.2 aufgelistet. Neben den Evidenzen können Autoren Freitextkommentare (Attribut *kommentar*) angeben. Auch Beziehungen zwischen Diagnosen (DIAGNOSE) und Therapieprinzipien (THERAPIEPRINZIP) können von den Autoren abgelegt werden (über den Objekttyp DIAGTHERAPIEPRINZIP). Es kann angegeben werden, welcher Verlauf und welche Prognose der Erkrankung bei Anwendung eines bestimmten Therapieprinzips zu erwarten sind (Attribute *verlauf*, *prognose*).
Falls bei einem Lehr-/Lernfall bestimmte technische Untersuchungsverfahren (TUNTERSVERFAHREN) und Labortests (LABORTEST) nicht angewandt bzw. durchgeführt werden dürfen oder sollten, kann hierfür eine Begründung angegeben werden (Attribut *ablehnungsgrund* in den Objekttypen FALL_TUNTERSVERFAHREN und FALL_LABORTEST). Der Objekttyp DIAGNOSESTELLUNGSBEURTEILUNG enthält u.a. das boole'sche Attribut *verdachtsdiagnose*, welches angibt, ob es sich um eine Verdachtsdiagnose handelt. Durch das Attribut *kategorie* wird festgelegt, ob es sich um eine Hauptdiagnose handelt, ob ein Zusammenhang der Diagnose mit der Hauptdiagnose besteht oder ob kein bzw. kein relevanter Zusammenhang mit der Hauptdiagnose besteht. Das boole'sche Attribut *vergangen* gibt an, ob es sich um eine Diagnose handelt, die bereits in der Vergangenheit (vor Beginn der Fallsimulation) gestellt worden ist.

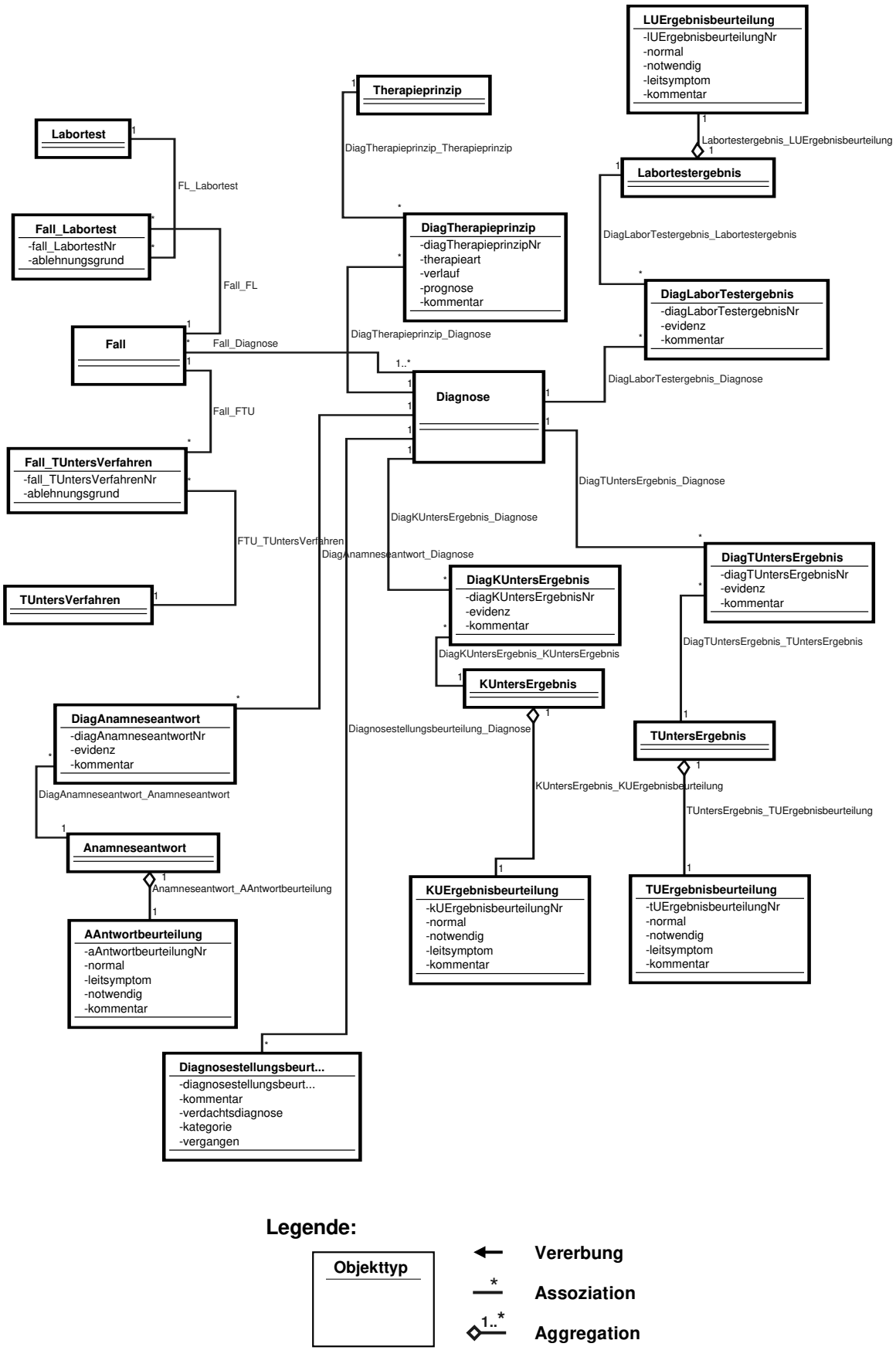

Abbildung 17: Fallwissensmodell

schema FALLWISSENSMODELL;

entityset DIAGLABORTESTERGEBNIS
(attributes: *diagLaborTestergebnisNr* l o n g ,
evidenz c h a r (3) not null,
kommentar c h a r (2000),
identifier: *diagLaborTestergebnisNr);*

(∀***dlt***)(DIAGLABORTESTERGEBNIS(***dlt***) → ***dlt***[*evidenz*] = "p3" ∨ ***dlt***[*evidenz*] = "p2" ∨ ***dlt***[*evidenz*] = "p1" ∨ ***dlt***[*evidenz*] = "n3" ∨ ***dlt***[*evidenz*] = "n2" ∨ ***dlt***[*evidenz*] = "n1" ∨ ***dlt***[*evidenz*] = "pn")

entityset DIAGTUNTERSERGEBNIS
(attributes: *diagTUntersErgebnisNr* l o n g ,
evidenz c h a r (3) not null,
kommentar c h a r (2000),
identifier: *diagTUntersErgebnisNr);*

(∀***dtu***)(DIAGTUNTERSERGEBNIS(***dtu***) → ***dtu***[*evidenz*] = "p3" ∨ ***dtu***[*evidenz*] = "p2" ∨ ***dtu***[*evidenz*] = "p1" ∨ ***dtu***[*evidenz*] = "n3" ∨ ***dtu***[*evidenz*] = "n2" ∨ ***dtu***[*evidenz*] = "n1" ∨ ***dtu***[*evidenz*] = "pn")

entityset DIAGKUNTERSERGEBNIS
(attributes: *diagnKUntersErgebnisNr* l o n g ,
evidenz c h a r (3) not null,
kommentar c h a r (2000),
identifier: *diagnKUntersErgebnisNr);*

(∀***dku***)(DIAGKUNTERSERGEBNIS(***dku***) → ***dku***[*evidenz*] = "p3" ∨ ***dku***[*evidenz*] = "p2" ∨ ***dku***[*evidenz*] = "p1" ∨ ***dku***[*evidenz*] = "n3" ∨ ***dku***[*evidenz*] = "n2" ∨ ***dku***[*evidenz*] = "n1" ∨ ***dku***[*evidenz*] = "pn")

entityset DIAGANAMNESEANTWORT
(attributes: *diagnAnamneseantwortNr* l o n g ,
evidenz c h a r (3) not null,
kommentar c h a r (2000),
identifier: *diagnAnamneseantwortNr);*

(∀***daa***)(DIAGANAMNESEANTWORT(***daa***) → ***daa***[*evidenz*] = "p3" ∨ ***daa***[*evidenz*] = "p2" ∨ ***daa***[*evidenz*] = "p1" ∨ ***daa***[*evidenz*] = "n3" ∨ ***daa***[*evidenz*] = "n2" ∨ ***daa***[*evidenz*] = "n1" ∨ ***daa***[*evidenz*] = "pn")

entityset DIAGTHERAPIEPRINZIP
(attributes: *diagnTherapieprinzipNr* long,
therapieart char(30),
verlauf char(2000),
prognose char(2000),
kommentar char(2000),
identifier: *diagnTherapieprinzipNr);*

($\forall$***dt***)(DIAGTHERAPIEPRINZIP(***dt***) $\rightarrow$ $\neg$(***dt***[*therapieart*] == null $\wedge$ ***dt***[v*erlauf*] == null $\wedge$ ***dt***[*prognose*] == null $\wedge$ ***dt***[*kommentar*] == null))

entityset LUERGEBNISBEURTEILUNG
(attributes: *lUErgebnisbeurteilungNr* long,
normal boolean not null,
notwendig boolean not null,
leitsymptom boolean not null,
kommentar char(2000),
identifier: *lUErgebnisbeurteilungNr);*

entityset TUERGEBNISBEURTEILUNG
(attributes: *tUErgebnisbeurteilungNr* long,
normal boolean not null,
notwendig boolean not null,
leitsymptom boolean not null,
kommentar char(2000),
identifier: *tUErgebnisbeurteilungNr);*

entityset KUERGEBNISBEURTEILUNG
(attributes: *kUErgebnisbeurteilungNr* long,
normal boolean not null,
notwendig boolean not null,
leitsymptom boolean not null,
kommentar char(2000),
identifier: *kUErgebnisbeurteilungNr);*

entityset AANTWORTBEURTEILUNG
(attributes: *aAntwortbeurteilungNr* long,
normal boolean not null,
notwendig boolean not null,
leitsymptom boolean not null,
kommentar char(2000),
identifier: *aAntwortbeurteilungNr);*

entityset DIAGNOSESTELLUNGSBEURTEILUNG
(attributes: *diagnosestellungsbeurteilungsNr* long,
kommentar char(2000),
verdachtsdiagnose boolean not null,
kategorie integer,
vergangen boolean not null,
identifier: *diagnosestellungsbeurteilungsNr);*

$(\forall \boldsymbol{d})(\text{DIAGNOSE}(\boldsymbol{d}) \rightarrow \boldsymbol{d}[kategorie] > 0 \wedge \boldsymbol{d}[kategorie] < 4)$

entityset FALL_TUNTERSVERFAHREN
(attributes: *fall_TUntersVerfahrenNr* long,
ablehnungsgrund char(2000) not null,
identifier: *fall_TUntersVerfahrenNr);*

entityset FALL_LABORTEST
(attributes: *fall_LabortestNr* long,
ablehnungsgrund char(2000) not null,
identifier: *fall_LabortestNr);*

relationship DIAGTHERAPIEPRINZIP_DIAGNOSE
(participants: (DIAGTHERAPIEPRINZIP, (1,1)),
(DIAGNOSE, (0,n)));

relationship DIAGANAMNESEANTWORT_DIAGNOSE
(participants: (DIAGANAMNESEANTWORT, (1,1)),
(DIAGNOSE, (0,n)));

relationship DIAGKUNTERSERGEBNIS_DIAGNOSE
(participants: (DIAGKUNTERSERGEBNIS, (1,1)),
(DIAGNOSE, (0,n)));

relationship DIAGTUNTERSERGEBNIS_DIAGNOSE
(participants: (DIAGTUNTERSERGEBNIS, (1,1)),
(DIAGNOSE, (0,n)));

relationship DIAGLABORTESTERGEBNIS_LABORTESTERGEBNIS
(participants: (DIAGLABORTESTERGEBNIS, (1,1)),
(LABORTESTERGEBNIS, (0,n)));

relationship DIAGTUNTERSERGEBNIS_TUNTERSERGEBNIS
(participants: (DIAGTUNTERSERGEBNIS, (1,1)),
(TUNTERSERGEBNIS, (0,n)));

relationship DIAGKUNTERSERGEBNIS_KUNTERSERGEBNIS
(participants: (DIAGKUNTERSERGEBNIS, (1,1)),
(KUNTERSERGEBNIS, (0,n)));

```
relationship DIAGANAMNESEANTWORT_ANAMNESEANTWORT
(participants:          (DIAGANAMNESEANTWORT, (1,1)),
                        (ANAMNESEANTWORT, (0,n)));

relationship DIAGTHERAPIEPRINZIP_THERAPIEPRINZIP
(participants:          (DIAGTHERAPIEPRINZIP, (1,1)),
                        (THERAPIEPRINZIP, (0,n)));

relationship DIAGNOSESTELLUNGSBEURTEILUNG_DIAGNOSE
(participants:          (DIAGNOSESTELLUNGSBEURTEILUNG, (1,1)),
                        (DIAGNOSE, (0,n)));

relationship FALL_DIAGNOSE
(participants:          (FALL, (1,n)),
                        (DIAGNOSE, (0,n)));

relationship FALL_FTU
(participants:          (FALL, (0,n)),
                        (FALL_TUNTERSVERFAHREN, (1,1)));

relationship FALL_FL
(participants:          (FALL, (0,n)),
                        (FALL_LABORTEST, (1,1)));

relationship FTU_TUNTERSVERFAHREN
(participants:          (FALL_TUNTERSVERFAHREN, (1,1)),
                        (TUNTERSVERFAHREN, (0,n)));

relationship FL_LABORTEST
(participants:          (FALL_LABORTEST, (1,1)),
                        (LABORTEST, (0,n)));


wholepartassociation LABORTESTERGEBNIS_LUERGEBNISBEURTEILUNG
(participants:          (LABORTESTERGEBNIS, (1,1)),
                        (LUERGEBNISBEURTEILUNG, (1,1)));

wholepartassociation TUNTERSERGEBNIS_TUERGEBNISBEURTEILUNG
(participants:          (TUNTERSERGEBNIS, (1,1)),
                        (TUERGEBNISBEURTEILUNG, (1,1)));

wholepartassociation KUNTERSERGEBNIS_KUERGEBNISBEURTEILUNG
(participants:          (KUNTERSERGEBNIS, (1,1)),
                        (KUERGEBNISBEURTEILUNG, (1,1)));

wholepartassociation ANAMNESEANTWORT_AANTWORTBEURTEILUNG
(participants:          (ANAMNESEANTWORT, (1,1)),
                        (AANTWORTBEURTEILUNG, (1,1)));
```

3.5.4 Lexikonmodell

Im CAMPUS-System existieren verschiedene Arten von Domänenwissen. Das enthaltene informelle Lehrbuchwissen wird den Nutzern in Form eines Hypertextes [KUHLEN 91] zugänglich gemacht. Das Lexikonmodell (siehe Abbildung 18) beschreibt die Struktur dieses Hypertextes. Ein Generator [PELZER 98] ermöglicht es, aus den Datenbankinhalten jederzeit einen mit gängigen WWW-Browsern zu lesenden Hypertext im HTML-Format zu erstellen. Mit Hilfe eines Parsers [PELZER 98] können bereits existierende HTML-Dokumente in Einzelteile zerlegt und in die Datenbank des CAMPUS-Systems aufgenommen werden.
Das erzeugte Lexikon ist für alle CAMPUS Lehr-/Lernsysteme einheitlich. In den einzelnen Systemen sind jedoch spezifische Verweise auf einzelne Seiten oder Kapitel des Lexikons vorhanden. Auch das Lexikonmodell stellt dadurch einen wichtigen Beitrag zur leichten Wiederverwendbarkeit dar. Die inhaltliche Gliederung des Lexikons wird in Kapitel 4.2 beschrieben.
Ein Knoten (Objekttyp KNOTEN) setzt sich jeweils aus mindestens einem Informationselement (Objekttyp INFOELEMENT) und genau einer Navigationsleiste (NAVIGATIONSLEISTE) zusammen. Informationselemente können beispielsweise Texte, Bilder, Töne oder Videos sein. Jeder Knoten ist Bestandteil genau eines Kapitels (KAPITEL). Jedem Knoten ist genau ein Objekt vom Typ KNOTENTYP zugeordnet. Ein Knoten kann beliebig viele Anker (ANKER) enthalten, welche Ausgangspunkt für Verweise (VERWEIS) auf Knoten sind. Einem Verweis ist genau ein Verweistyp (VERWEISTYP) zugeordnet. Navigationsleisten bestehen aus mindestens einem Navigationselement (NAVIGATIONSELEMENT). Deren Position in einer Navigationsleiste wird im Objekttyp POSITION (Attribut *reihenfolgeposition*) festgelegt. Anker können entweder Navigationselemente oder Informationselemente bzw. Teile von diesen sein.
Das Attribut *pfad* im Objekttyp KNOTEN gibt an, wo sich der „RestHTML-Code“ eines Knotens befindet. Dieser Code bildet das „Grundgerüst“ (im HTML-Format), in das die Informationselemente sowie die Navigationsleiste eines Knotens eingebunden werden, wenn das gesamte Lexikon bzw. ein einzelner Knoten den Nutzern per WWW-Browser zugänglich gemacht wird. Der Vorteil dieser Lösung besteht darin, daß unabhängig vom verwendeten HTML-Editor und der genutzten HTML-Version immer gewährleistet wird, daß ein importiertes HTML-Dokument in seinem ursprünglichen Layout wiederhergestellt werden kann [PELZER 98]. Das Attribut *elementtyp* im Objekttyp ANKER gibt an, ob der Anker an einem Informationselement oder an einem Navigationselement verankert ist. Die Attribute *zielanker*, *rel*, *rev* und *title* sind optionale Attribute des <A>-Tags von HTML. Das Attribut *ausrichtung* im Objekttyp NAVIGATIONSLEISTE beinhaltet die Information darüber, ob die Navigationsleiste zentriert, links- oder rechtsbündig angezeigt wird. Das Attribut *ausrichtungsebene* entscheidet darüber, ob die Navigationsleiste horizontal oder vertikal angezeigt wird. Durch das Attribut *url* im Objekttyp VERWEIS wird es ermöglicht, auf externe HTML-Dokumente zu verweisen. Systemeigene Knoten können über das Attribut *knotenNr* als Verweisziel angegeben werden.

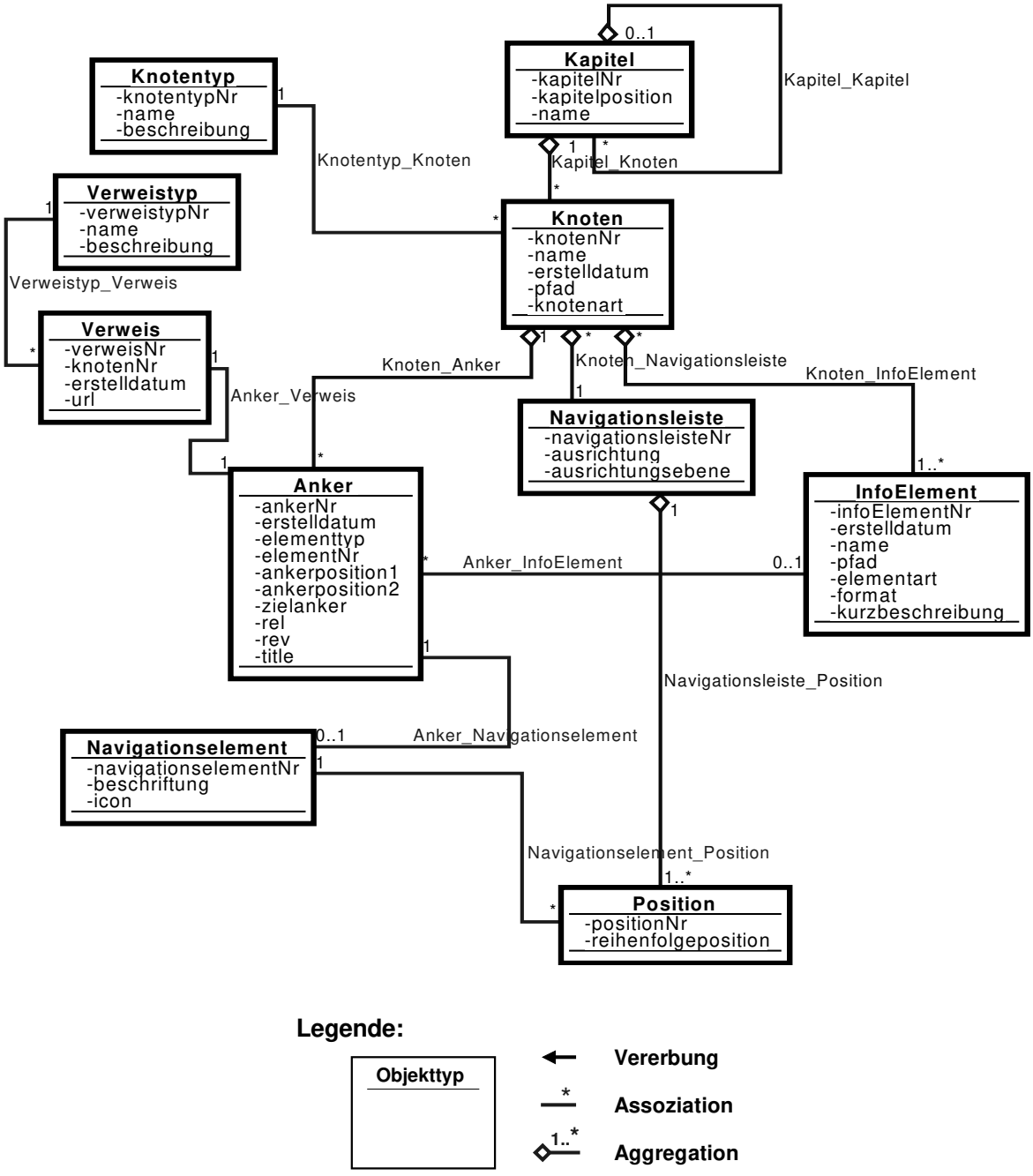

Abbildung 18: Lexikonmodell

schema LEXIKONMODELL;

entityset KNOTEN
(attributes: *knotenNr* long,
name char(255),
erstelldatum date not null,
pfad char(50) not null,
knotenart char(50) not null,
identifier: *knotenNr);*

$(\forall \boldsymbol{k})(\text{KNOTEN}(\boldsymbol{k}) \rightarrow \boldsymbol{k}[knotenart] = \text{“RestHTML“} \vee \boldsymbol{k}[knotenart] = \text{“Sonstiges“})$

entityset KNOTENTYP
(attributes: *knotentypNr* long,
name char(50) not null,
beschreibung char(2000) not null,
identifier: *knotentypNr);*

entityset ANKER
(attributes: *ankerNr* long,
erstelldatum date not null,
elementtyp char(1) not null,
elementNr long not null,
ankerposition1 integer not null,
ankerposition2 integer not null,
zielanker char(50) not null,
rel char(50),
rev char(50),
title char(50),
identifier: *ankerNr);*

entityset VERWEIS
(attributes: *verweisNr* long,
knotenNr long not null,
erstelldatum date not null,
url char(80) not null,
identifier: *verweisNr);*

entityset VERWEISTYP
(attributes: *verweistypNr* long,
name char(50) not null,
beschreibung char(2000) not null,
identifier: *verweistypNr);*

entityset KAPITEL
(attributes: *kapitelNr* long,
kapitelposition integer,
name char(80) not null,
identifier: *kapitelNr);*

$(\forall \boldsymbol{k_1}, \boldsymbol{k_2})$(KAPITEL($\boldsymbol{k_1}$) $\wedge$ KAPITEL($\boldsymbol{k_2}$) $\wedge$ $\boldsymbol{k_1}$[kapitelposition] = $\boldsymbol{k_2}$[kapitelposition] $\rightarrow \boldsymbol{k_1} == \boldsymbol{k_2}$)

entityset NAVIGATIONSLEISTE
(attributes: *navigationsleisteNr* long,
ausrichtung char(1) not null,
ausrichtungsebene char(1) not null,
identifier: *navigationsleisteNr);*

$(\forall \boldsymbol{n})$(NAVIGATIONSLEISTE($\boldsymbol{n}$) $\rightarrow$ $\boldsymbol{n}$[*ausrichtung*] = "l" $\vee$ $\boldsymbol{n}$[*ausrichtung*] = "r" $\vee$ $\boldsymbol{n}$[*ausrichtung*] = "c")

$(\forall \boldsymbol{n})$(NAVIGATIONSLEISTE($\boldsymbol{n}$) $\rightarrow$ $\boldsymbol{n}$[*ausrichtungsebene*]= "h" $\vee$ $\boldsymbol{n}$[*ausrichtungsebene*]= "v")

entityset POSITION
(attributes: *positionNr* long,
reihenfolgeposition integer not null,
identifier: *positionNr;*

$(\forall \boldsymbol{p_1}, \boldsymbol{p_2})$(POSITION($\boldsymbol{p_1}$) $\wedge$ POSITION($\boldsymbol{p_2}$) $\wedge$ $\boldsymbol{p_1}$[reihenfolgeposition] = $\boldsymbol{p_2}$[reihenfolgeposition] $\rightarrow \boldsymbol{p_1} == \boldsymbol{p_2}$)

entityset NAVIGATIONSELEMENT
(attributes: *navigationselementNr* long,
beschriftung char(10),
icon char(50),
identifier: *NavigationselementNr);*

$(\forall \boldsymbol{n})$(NAVIGATIONSELEMENT($\boldsymbol{n}$) $\rightarrow \neg(\boldsymbol{n}$[*beschriftung*] == null $\wedge$ $\boldsymbol{n}$[icon] == null) $\wedge$ $\neg(\neg(\boldsymbol{n}$[beschriftung] == null) $\wedge \neg(\boldsymbol{n}$[*icon*] == null)))

entityset INFOELEMENT
(attributes: *infoElementNr* long,
erstelldatum char(10) not null,
name char(30),
pfad char(50) not null,
elementart char(15) not null,
format char(15) not null,
kurzbeschreibung char(2000),
identifier: *infoElementNr);*

```
wholepartassociation KAPITEL_KNOTEN
(participants:          (KAPITEL, (0,n)),
                        (KNOTEN, (1,1)));

wholepartassociation KAPITEL_KAPITEL
(participants:          (KAPITEL, (0,n)),
                        (KAPITEL, (0,1)));

wholepartassociation KNOTEN_ANKER
(participants:          (KNOTEN, (0,n)),
                        (ANKER, (1,1)));

wholepartassociation KNOTEN_NAVIGATIONSLEISTE
(participants:          (KNOTEN, (1,1)),
                        (NAVIGATIONSLEISTE, (0,n)));

wholepartassociation KNOTEN_INFOELEMENT
(participants:          (KNOTEN, (1,n)),
                        (INFOELEMENT, (0,n)));

wholepartassociation NAVIGATIONSLEISTE_POSITION
(participants:          (NAVIGATIONSLEISTE, (1,n)),
                        (POSITION, (0,n)));

relationship ANKER_NAVIGATIONSELEMENT
(participants:          (ANKER, (0,1)),
                        (NAVIGATIONSELEMENT, (1,1)));


relationship NAVIGATIONSELEMENT_POSITION
(participants:          (NAVIGATIONSELEMENT, (0,n)),
                        (POSITION, (1,1)));

relationship KNOTENTYP_KNOTEN
(participants:          (KNOTENTYP, (0,n)),
                        (KNOTEN, (1,1)));

relationship VERWEISTYP_VERWEIS
(participants:          (VERWEISTYP, (0,n)),
                        (VERWEIS, (1,1)));

relationship ANKER_INFOELEMENT
(participants:          (ANKER, (0,1)),
                        (INFOELEMENT, (0,n)));

relationship ANKER_VERWEIS
(participants:          (ANKER, (1,1)),
                        (VERWEIS, (1,1)));
```

3.5.5 Layoutmodell

Autoren können für Laborformulare (Objekttyp LABORFORMULAR) und für technische Formulare (Objekttyp TECHNFORMULAR) eigene Layouts definieren. Laborformulare können dabei aus mehreren Teilformularen (TEILFORMULAR) zusammengesetzt sein. Die Position eines Teilformulars in einem Laborformular wird durch Objekte des Typs FORMULARPOSITION beschrieben. Dieser Objekttyp besitzt die Attribute *spaltenposition* und *reihenfolgeposition*. Das Attribut *spaltenposition* gibt an, in welcher Spalte des Laborformulars das Teilformular angesiedelt ist. Das Attribut *reihenfolgeposition* bestimmt, an welcher Stelle in der Spalte das Teilformular positioniert ist. Weiterhin ist es möglich, den einzelnen CAMPUS Lehr-/Lernsystemen Layouts (LAYOUT) mit den Attributen *hintergrundfarbe*, *kopfzeile* und *fusszeile* zuzuordnen. Diese wirken sich vor allem auf die Anzeige des Lexikons (Hypertext) aus.

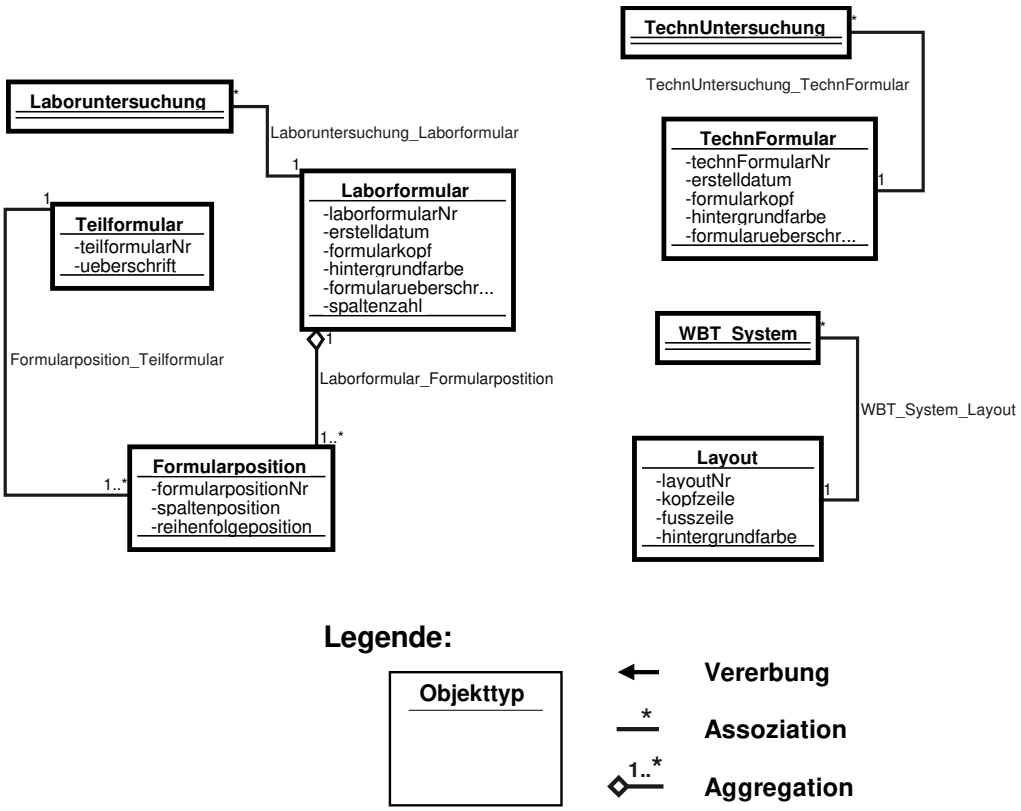

Abbildung 19: Layoutmodell

Autoren haben damit in gewissem Umfang die Möglichkeit, das Aussehen der erstellten Systeme zu beeinflussen und vor allem Formulare so zu gestalten, wie sie im „eigenen" Krankenhaus benutzt werden. Dadurch wird eine noch größere Realitätsnähe der Fallsimulationen erreicht. Das Aussehen der Lehr-/Lernsystemfenster (z.B. Anordnung der Buttons, Farbgestaltung, Aufteilung usw.) kann jedoch nicht beeinflußt werden. Dies hat zwei Gründe: Zum einen wurden bei der Gestaltung die einschlägigen User-Interface-Guidelines berücksichtigt. Die meisten Medizindozenten haben üblicherweise darüber keine detaillierten Kenntnisse und wären aus diesem Grund auch ohne Schulung nicht in der Lage, entsprechende Bildschirmlayouts zu definieren. Zum anderen ergibt sich für die Nutzer von CAMPUS ein großer Vorteil: Für sie entfällt viel Einarbeitungsaufwand, weil

die Systeme in der Bedienung immer gleich sind. Darüber hinaus stehen ja z.B. für die Fallbearbeitung sowieso verschiedene Präsentations-/Interaktionsformen zur Verfügung (siehe Kapitel 3.4.3), so daß die Nutzer jederzeit in der Lage sind, die Präsentations-/Interaktionsform zu wählen, welche ihren momentanen Bedürfnissen am besten entspricht.

schema LAYOUTMODELL;

entityset LABORFORMULAR
(attributes: *laborformularNr* long,
erstelldatum char(2000) not null,
formularkopf char(50),
hintergrundfarbe char(15),
formularueberschrift char(80) not null,
spaltenzahl integer not null,
identifier: *laborformularNr);*

entityset TECHNFORMULAR
(attributes: *technFormularNr* long,
erstelldatum char(2000) not null,
formularkopf char(50),
hintergrundfarbe char(15),
formularueberschrift char(80) not null,
identifier: *technFormularNr);*

entityset FORMULARPOSITION
(attributes: *formularpositionNr* long,
spaltenposition integer not null,
reihenfolgeposition integer not null,
identifier: *formularpositionNr);*

entityset TEILFORMULAR
(attributes: *teilformularNr* long,
ueberschrift char (50) not null,
identifier: *teilformularNr);*

entityset LAYOUT
(attributes: *layoutNr* long,
kopfzeile char (50),
fusszeile char (50),
hintergrundfarbe char (15),
identifier: *layoutNr);*

$(\forall \boldsymbol{l})(\text{LAYOUT}(\boldsymbol{l}) \rightarrow \neg(\boldsymbol{l}[\textit{kopfzeile}] == \text{null} \wedge \boldsymbol{l}[\textit{fusszeile}] == \text{null} \wedge \boldsymbol{l}[\textit{hintergrundfarbe}] == \text{null}))$

```
relationship TECHNUNTERSUCHUNG_TECHNFORMULAR
(participants:          (TECHNUNTERSUCHUNG, (1,1)),
                        (TECHNFORMULAR, (0,n)));

relationship WBT_SYSTEM_LAYOUT
(participants:          (WBT_SYSTEM, (1,1)),
                        (LAYOUT, (0,n)));

relationship FORMULARPOSITION_TEILFORMULAR
(participants:          (FORMULARPOSITION, (1,1)),
                        (TEILFORMULAR, (1,n)));

relationship LABORUNTERSUCHUNG_LABORFORMULAR
(participants:          (LABORUNTERSUCHUNG, (1,1)),
                        (LABORFORMULAR, (0,n)));

wholepartassociation LABORFORMULAR_FORMULARPOSITION
(participants:          (LABORFORMULAR, (1,n)),
                        (FORMULARPOSITION, (1,1)));
```

3.5.6 Nutzermodell

Das Nutzermodell (siehe Abbildung 20) beschreibt die Vorlieben von Nutzern und deren bisherige Nutzung des CAMPUS-Systems. Es ist Voraussetzung für die Adaption des Systems an seine Nutzer.
Ein Objekt vom Typ NUTZER besteht aus einer beliebigen Anzahl von Fallbearbeitungen (Objekttyp FALLBEARBEITUNG), Fragebearbeitungen (FRAGEBEARBEITUNG), Hilfeprotokollen (HILFEPROTOKOLL) und Knotenbesuchen (KNOTENBESUCH). Jedem Objekt vom Typ HILFEPROTOKOLL ist genau ein Objekt vom Typ HILFETEXT zugeordnet. Das Attribut *ereignis* im Objekttyp HILFETEXT bestimmt das Ereignis, bei dessen Eintritt der Hilfetext-String (Attribut *hilfetext*) angezeigt wird. Jeder Nutzer besitzt eine eindeutige Nutzerkennung (Attribut *kennung*) und ein Paßwort (Attribut *passwort*). Das Datum, an dem der Nutzer angelegt wurde, wird im Attribut *anmeldedatum* abgelegt. Aus der Semesterzahl (Attribut *semesterzahl*) beim Anlegen des Nutzers und dem Anmeldedatum kann jederzeit das aktuelle Semester eines Studenten errechnet werden. Im Attribut *loginDatum* ist das Datum der letzten Nutzung des CAMPUS-Systems abgelegt. Dadurch ist es möglich, Nutzer zu erkennen, die das System offensichtlich nicht mehr nutzen. Jeder Benutzer kann selbst entscheiden, ob sich das System selbst adaptieren soll, oder nicht (Attribut *adaptationAktiviert*). Falls er die automatische Adaption nicht aktiviert hat, dann kann er das System selbst adaptieren. Er kann auch entscheiden, ob das System prospektiv Daten vom Server laden soll oder nicht (Attribut *prospektivesLaden*). Ein Nutzer kann weiterhin selbst bestimmen, ob er von anderen Nutzern für gegenseitige Diskussionen angetalkt werden darf oder nicht (Attribut *talk*). Das Attribut *totaleNurPatho* enthält die Information darüber, ob der Nutzer bei der Präsentations-/Interaktionsform *Totale* nur die auffälligen Befunde angezeigt bekommt.
Beim Nutzermodell handelt es sich um ein einfaches Zuordnungsmodell [BODENDORF 90 S. 132]. Abhängig von der Zahl korrekt bearbeiteter Lehr-/Lernfälle, korrekt beantworteter Fragen und der aktuellen Semesterzahl wird der Nutzer in verschiedene Kategorien eingeteilt (Attribute *bedienungsanleitung, praesentationsform, fragen, rueckmeldezeitpunkt*).

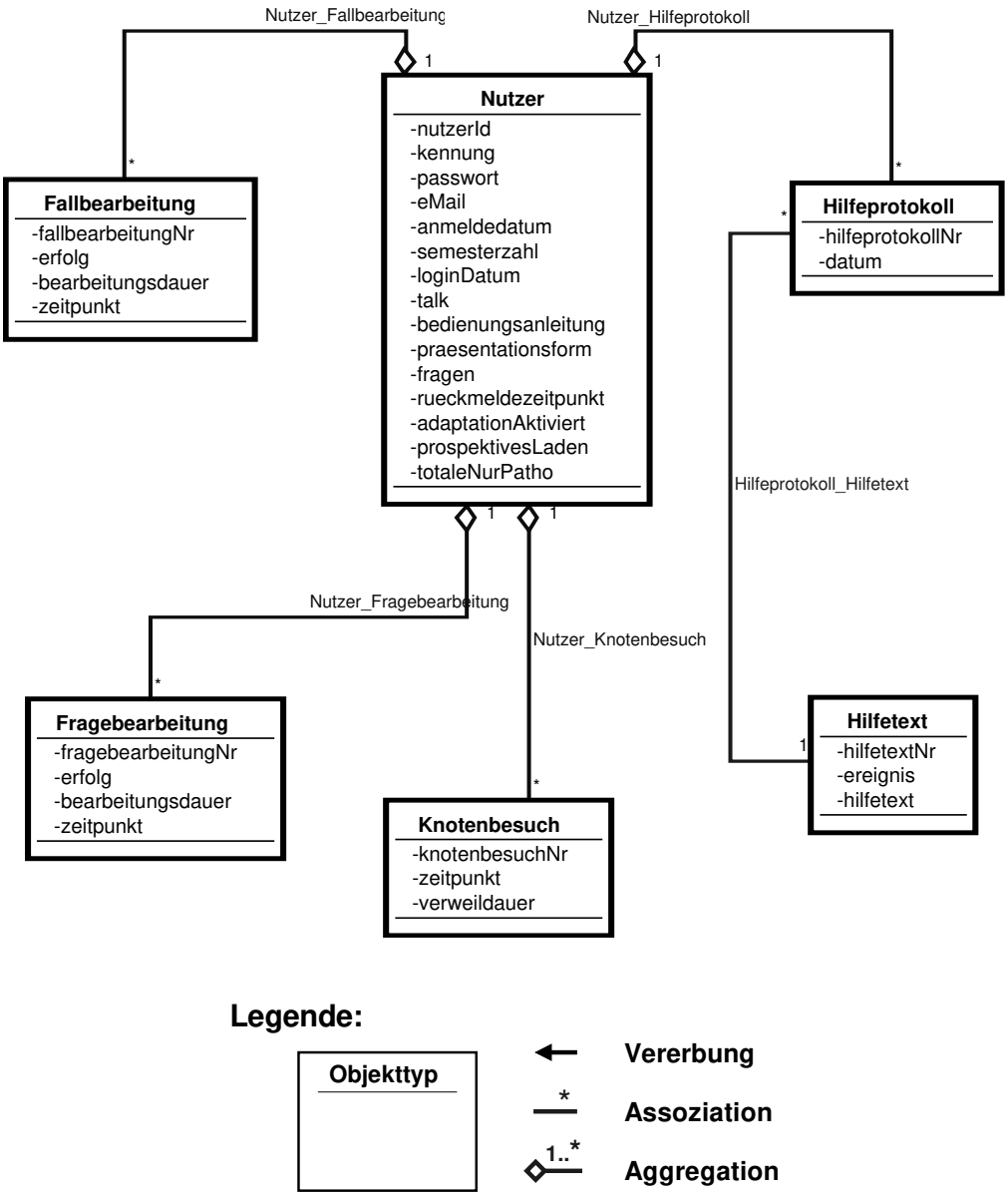

Abbildung 20: Nutzermodell

Bei jeder Fallbearbeitung und Fragebearbeitung werden Erfolg, Bearbeitungsdauer und Zeitpunkt (Attribute *erfolg*, *bearbeitungsdauer*, *zeitpunkt*) ermittelt. Beim Knotenbesuch werden Zeitpunkt und Verweildauer (Attribute *zeitpunkt*, *verweildauer*) bestimmt. Beim Einblenden von Hilfeseiten wird das Datum (Attribut *datum*) mitprotokolliert. Auf Basis dieser ganzen Informationen kann das Lehrsubsystem geeignete Lehr-/Lernfälle und Fragen auswählen sowie bei Bedarf geeignete Hilfen anbieten.

schema NUTZERMODELL;

entityset NUTZER

(attributes:	*nutzerId*	long,
	kennung	char(20) not null,
	passwort	char(20) not null,
	eMail	char(50),
	anmeldedatum	date not null,
	semesterzahl	integer not null,
	loginDatum	date not null,
	talk	boolean not null,
	bedienungsanleitung	integer not null,
	praesentationsform	integer not null,
	fragen	integer not null,
	rueckmeldezeitpunkt	integer not null,
	adaptationAktiviert	boolean not null,
	prospektivesLaden	boolean not null,
	totaleNurPatho	boolean not null,
identifier:	*nutzerId);*	

Semantische Integritätsbedingungen

$(\forall \boldsymbol{n})(\text{NUTZER}(\boldsymbol{n}) \rightarrow \boldsymbol{n}[bedienungsanleitung] = 0 \vee \boldsymbol{n}[bedienungsanleitung] = 1)$
$(\forall \boldsymbol{n})(\text{NUTZER}(\boldsymbol{n}) \rightarrow \boldsymbol{n}[praesentationsform] > -1 \wedge \boldsymbol{n}[praesentationsform] < 4)$
$(\forall \boldsymbol{n})(\text{NUTZER}(\boldsymbol{n}) \rightarrow \boldsymbol{n}[fragen] > -1 \wedge \boldsymbol{n}[fragen] < 3)$
$(\forall \boldsymbol{n})(\text{NUTZER}(\boldsymbol{n}) \rightarrow \boldsymbol{n}[rueckmeldezeitpunkt] = 0 \vee \boldsymbol{n}[rueckmeldezeitpunkt] = 1)$
$(\forall \boldsymbol{n})(\text{NUTZER}(\boldsymbol{n}) \rightarrow \boldsymbol{n}[anmeldedatum] \leq \boldsymbol{n}[logindatum])$

entityset FALLBEARBEITUNG

(attributes:	*fallbearbeitungNr*	long,
	erfolg	real not null,
	bearbeitungsdauer	integer not null,
	zeitpunkt	date not null,
identifier:	*fallbearbeitungNr);*	

entityset FRAGEBEARBEITUNG

(attributes:	*fragebearbeitungNr*	long,
	erfolg	boolean not null,
	bearbeitungsdauer	integer not null,
	zeitpunkt	date not null,
identifier:	*fragebearbeitungNr);*	

entityset KNOTENBESUCH

(attributes:	*knotenbesuchNr*	long,
	zeitpunkt	date not null,
	verweildauer	integer not null,
identifier:	*knotenbesuchNr);*	

```
entityset HILFEPROTOKOLL
(attributes:        hilfeprotokollNr        long,
                    datum                   date not null,
identifier:         hilfeprotokollNr);

entityset HILFETEXT
(attributes:        hilfetextNr             long,
                    ereignis                char(50) not null,
                    hilfetext               char(2000) not null,
identifier:         hilfetextNr);

relationship HILFEPROTOKOLL_HILFETEXT
(participants:      (HILFEPROTOKOLL, (1,1)),
                    (HILFETEXT, (0,n)));

wholepartassociation NUTZER_FALLBEARBEITUNG
(participants:      (NUTZER, (0,n)),
                    (FALLBEARBEITUNG, (1,1)));

wholepartassociation NUTZER_FRAGEBEARBEITUNG
(participants:      (NUTZER, (0,n)),
                    (FRAGEBEARBEITUNG, (1,1)));

wholepartassociation NUTZER_KNOTENBESUCH
(participants:      (NUTZER, (0,n)),
                    (KNOTENBESUCH, (1,1)));

wholepartassociation NUTZER_HILFEPROTOKOLL
(participants:      (NUTZER, (0,n)),
                    (HILFEPROTOKOLL, (1,1)));
```

3.5.7 Fragemodell

Mit dem Fragemodell (siehe Abbildung 21) werden Fragen und die zugehörigen Antworttypen beschrieben.

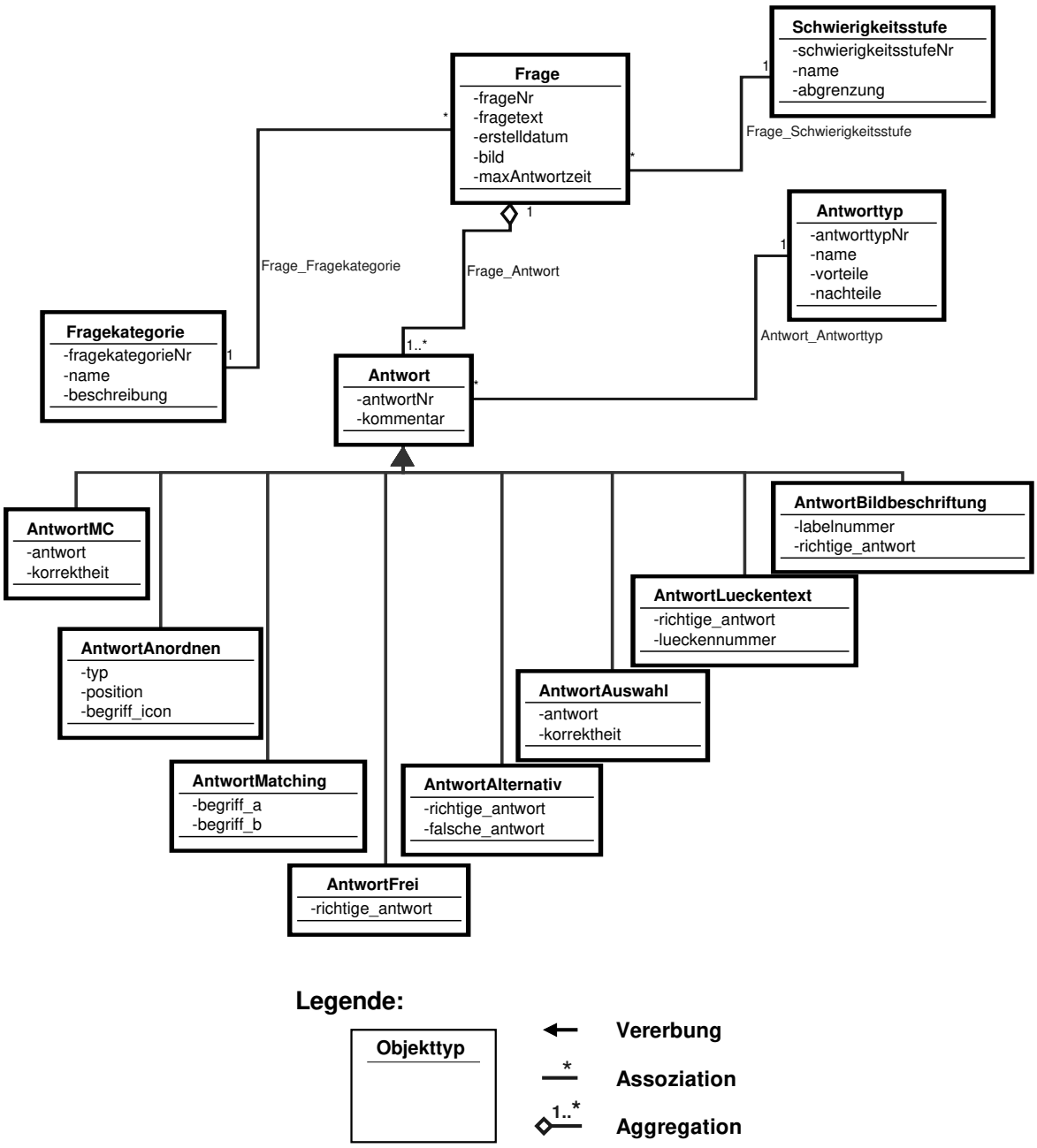

Abbildung 21: Fragemodell

Ein Objekt vom Typ FRAGE besitzt die Attribute *erstelldatum*, *fragetext*, *bild* (Verweis auf ein evtl. der Frage zugeordnetes Bild) und *maxAntwortzeit* (maximale Antwortzeit). Zu jeder Frage gehören mindestens eine Antwort (Objekttyp ANTWORT) genau eines der folgenden Typen von Antworten (siehe auch Kapitel 4.2):

- Freie Eingabe (ANTWORTFREI)
- Alternativantworten (ANTWORTALTERNATIV)
- Auswahlmenü (ANTWORTAUSWAHL)
- Multiple-Choice (ANTWORTMC)
- Matching (ANTWORTMATCHING)
- Bildbeschriftung (ANTWORTBILDBESCHRIFTUNG)
- Anordnen von Begriffen/Icons (ANTWORTANORDNEN)
- Lückentext (ANTWORTLUECKENTEXT)

Jeder Frage ist außerdem genau eine Schwierigkeitsstufe (SCHWIERIGKEITSSTUFE), genau eine Fragekategorie (FRAGEKATEGORIE) und genau ein Antworttyp (ANTWORTTYP) zugeordnet.
Durch die vom System unterstützten 8 unterschiedlichen Antworttypen haben Autoren die Möglichkeit, genau auf ihre speziellen Anforderungen passende Wissenskontrollfragen und Fragen zur Unterstützung der Fallbearbeitung zu erstellen.

```
schema FRAGEMODELL;

entityset FRAGE
(attributes:        frageNr                 long,
                    fragetext               char(2000) not null,
                    erstelldatum            date  not null,
                    bild                    char(50),
                    maxAntwortzeit          integer,
identifier:         frageNr);

entityset ANTWORTTYP
(attributes:        antworttypNr            long,
                    name                    char(30) not null,
                    vorteile                char(2000) not null,
                    nachteile               char(2000) not null,
identifier:         antworttypNr);

entityset SCHWIERIGKEITSSTUFE
(attributes:        schwierigkeitsstufeNr   long,
                    name                    char(30)  not null,
                    abgrenzung              char(2000)  not null,
identifier:         schwierigkeitsstufeNr);
```

entityset FRAGEKATEGORIE
(attributes: *fragekategorieNr* long,
name char(30) not null,
beschreibung char(2000) not null,
identifier: *fragekategorieNr);*

entityset ANTWORT
(attributes: *antwortNr* long,
kommentar char(2000) not null,
identifier: *antwortNr);*

entityset ANTWORTBILDBESCHRIFTUNG
(subset of ANTWORT,
attributes: *labelnummer* integer not null,
richtige_antwort char(200) not null);

entityset ANTWORTLUECKENTEXT
(subset of ANTWORT,
attributes: *lueckennummer* integer not null,
richtige_antwort char(200) not null);

entityset ANTWORTAUSWAHL
(subset of ANTWORT,
attributes: *antwort* char(200) not null,
korrektheit boolean not null);

entityset ANTWORTMC
(subset of ANTWORT,
attributes: *antwort* char(200) not null,
korrektheit boolean not null);

entityset ANTWORTFREI
(subset of ANTWORT,
attributes: *richtige_antwort* char(200) not null);

entityset ANTWORTMATCHING
(subset of ANTWORT,
attributes: *begriff_a* char(200) not null,
begriff_b char(200) not null);

entityset ANTWORTANORDNEN
(subset of ANTWORT,
attributes: *typ* char(1) not null,
position integer not null),
begriff_icon char(50) not null);

$(\forall \boldsymbol{aan})(\text{NUTZER}(\boldsymbol{aan}) \rightarrow \boldsymbol{aan}[typ] = \text{“b“} \vee \boldsymbol{aan}[typ] = \text{“i“})$

entityset ANTWORTALTERNATIV
(subset of ANTWORT,
attributes: *richtige_antwort* c h a r (200) not null
falsche_antwort c h a r (200) not null) ;

relationship ANTWORT_ANWORTTYP
(participants: (ANTWORT, (1,1)),
(ANTWORTTYP, (0,n)));

relationship FRAGE_SCHWIERIGKEITSSTUFE
(participants: (FRAGE, (1,1)),
(SCHWIERIGKEITSSTUFE, (0,n)));

relationship FRAGE_FRAGEKATEGORIE
(participants: (FRAGE, (1,1)),
(FRAGEKATEGORIE, (0,n)));

wholepartassociation FRAGE_ANTWORT
(participants: (FRAGE, (1,n)),
(ANTWORT, (1,1)));

Nur ein Typ von Antworten ist pro Frage zulässig:
$(\forall \boldsymbol{f},\boldsymbol{a1},\boldsymbol{a2})(\text{FRAGE}(\boldsymbol{f}) \wedge \boldsymbol{f}[\text{ANTWORT}(\boldsymbol{a1})] \wedge \boldsymbol{f}[\text{ANTWORT}(\boldsymbol{a2})] \wedge \neg(\boldsymbol{a1}==\boldsymbol{a2})$
$\rightarrow (\exists\, \boldsymbol{aat1}, \boldsymbol{at1})\ (\text{ANTWORT\_ANTWORTTYP}(\boldsymbol{aat1}) \wedge \text{ANTWORTTYP}(\boldsymbol{at1}) \wedge$
$\boldsymbol{aat1}:\text{ANTWORT}==\boldsymbol{a1} \wedge \boldsymbol{aat1}:\text{ANTWORT\_ANTWORTTYP} == \boldsymbol{at1}$
$\wedge\neg(\exists\, \boldsymbol{aat2}, \boldsymbol{at2})\ (\text{ANTWORT\_ANTWORTTYP}(\boldsymbol{aat2}) \wedge \text{ANTWORTTYP}(\boldsymbol{at2}) \wedge$
$\boldsymbol{aat2}:\text{ANTWORT}==\boldsymbol{a2} \wedge \boldsymbol{aat2}:\text{ANTWORT\_ANTWORTTYP} == \boldsymbol{at2} \wedge \neg(\boldsymbol{at1}==\boldsymbol{at2})))$

3.5.8 WBT-Systemmodell

Das WBT-Systemmodell (siehe Abbildung 22) beschreibt, aus welchen Komponenten ein CAMPUS Lehr-/Lernsystem besteht bzw. mit welchen es in Beziehung steht. Dabei wurden nur die Interaktionsformen *Simulation* (Fallsimulation), *Browsing* (systematisches Lehrbuchwissen) und *Drill & Practice* (Fragen) berücksichtigt. Eine Erweiterung des Modells durch die Integration der im Rahmen dieser Arbeit nicht näher betrachteten Interaktionsformen (*Präsentation*, *Tutorieller Dialog*, *Grundlagensimulation*) ist jederzeit möglich.
Einem WBT-System (Objekttyp WBT_SYSTEM) ist genau ein Autor (Objekttyp AUTOR) und genau ein Layout (LAYOUT) zugeordnet. Dagegen kann ein Autor eine beliebige Anzahl von WBT-Systemen erstellen bzw. bearbeiten und ein Layout kann für eine beliebige Anzahl von WBT-Systemen verwendet werden. Ein WBT-System kann eine beliebige Anzahl von Lehr-/Lernfällen (FALL) und Fragen (FRAGE) enthalten, außerdem muß ihm mindestens ein Knoten (KNOTEN) zugeordnet sein. Lehr-/Lernfälle und Fragen sind immer in mindestens einem WBT-System enthalten und ein Knoten ist immer mindestens einem WBT-System zugeordnet. Jedes WBT-System kann von einer beliebigen Anzahl von Nutzern (NUTZER) genutzt werden, ein Nutzer ist jedoch immer mindestens einem WBT-System zugeteilt. Ein WBT-System ist mindestens einem Fachgebiet (FACHGEBIET) zugeordnet. Zu einem Fachgebiet können eine beliebige Anzahl von WBT-Systemen existieren.

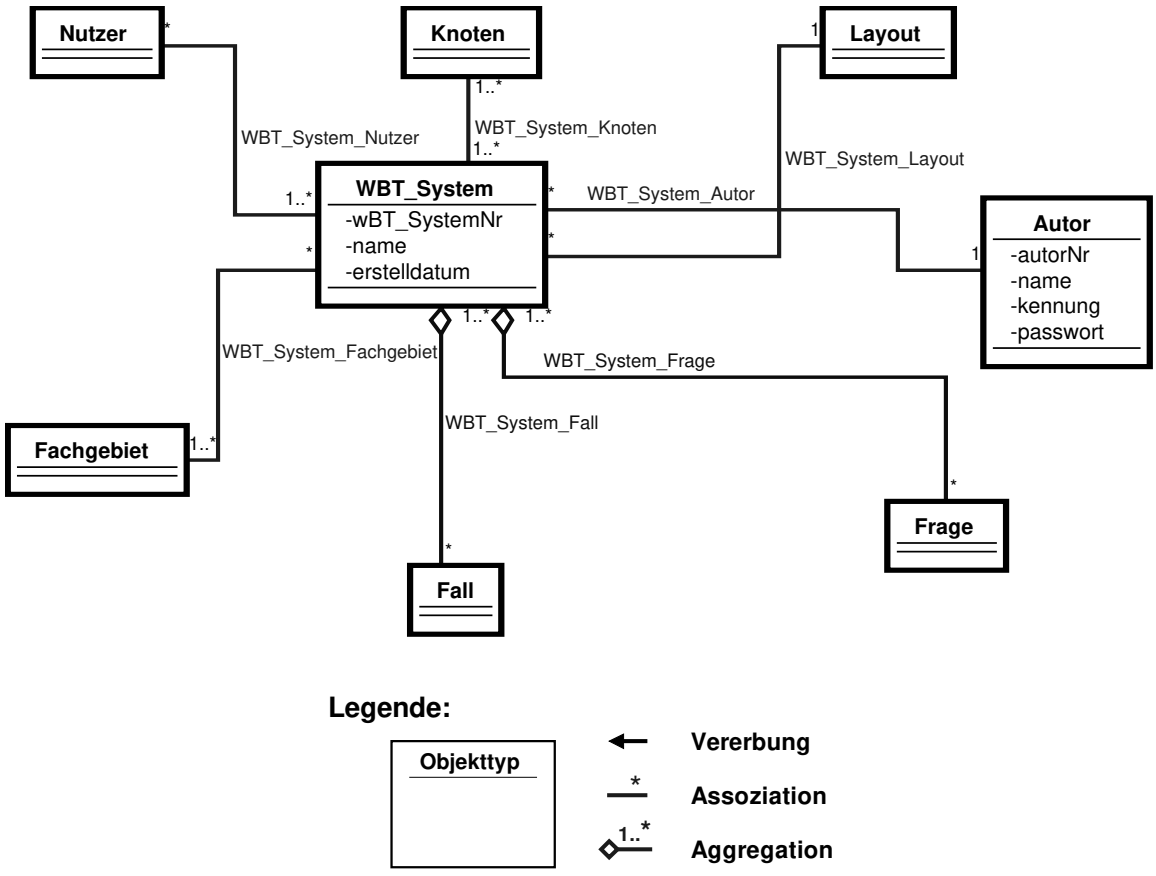

Abbildung 22: WBT-Systemmodell

```
schema WBT_SYSTEMMODELL;

entityset WBT_SYSTEM
(attributes:        wBT_SystemNr          long,
                    name                  char(80) not null,
                    erstelldatum          date not null,
identifier:         wBT_SystemNr);

entityset AUTOR
(attributes:        autorNr               long,
                    name                  char(50) not null,
                    kennung               char(15) not null,
                    passwort              char(15) not null,
identifier:         autorNr);

relationship WBT_SYSTEM_AUTOR
(participants:      (WBT_SYSTEM, (1,1)),
                    (AUTOR, (0,n)));

relationship WBT_SYSTEM_KNOTEN
(participants:      (WBT_SYSTEM, (1,n)),
                    (KNOTEN, (1,n)));

relationship WBT_SYSTEM_FACHGEBIET
(participants:      (WBT_SYSTEM, (1,n)),
                    (FACHGEBIET, (0,n)));

relationship WBT_SYSTEM_NUTZER
(participants:      (WBT_SYSTEM, (0,n)),
                    (NUTZER, (1,n)));

relationship WBT_SYSTEM_LAYOUT
(participants:      (WBT_SYSTEM, (1,1)),
                    (LAYOUT, (0,n)));

wholepartassociation WBT_SYSTEM_FRAGE
(participants:      (WBT_SYSTEM, (0,n)),
                    (FRAGE, (0,n)));

wholepartassociation WBT_SYSTEM_FALL
(participants:      (WBT_SYSTEM, (0,n)),
                    (FALL, (0,n)));
```

4 Realisierung

4.1 Werkzeuge und Basistechniken

Bei der Konzeption des Systems wurde auf eine möglichst weitgehende Verwendung von Standards geachtet. Dadurch ergeben sich verschiedene Vorteile. U.a. können einzelne Bestandteile des Systems besser wiederverwendet werden und ein Wechsel von Tools und Programmierumgebungen ist bei Bedarf leicht möglich. Als Programmiersprache wird die objektorientierte Programmiersprache Java [GOSLING, YELLIN et al. 97a,b] verwendet. Als Entwicklungsumgebung kommen *Symantec Café* und *Symantec Visual Café* [SYMANTEC] sowie das *JDK* (*Java Development Kit*) von Sun zum Einsatz.
Bei der Implementierung des Lehrsubsystems und des Lernsubsystems wird ein Sprachumfang genutzt, der voll im JDK 1.1 enthalten ist. Das Autorensubsystem wird auf Wunsch des dafür zuständigen Entwicklers mit den Swing-Klassen implementiert, die erst ab JDK 1.2 verfügbar sind. Durch die Beschränkung der Lehr- und Lernsubsysteme auf das JDK 1.1 erweitert sich die Zahl der Rechner stark, von denen aus die CAMPUS-Systeme genutzt werden können. Der CAMPUS Lehr-/Lernsystemclient läuft u.a. in der virtuellen Maschine von Standard-WWW-Browsern und wurde speziell für diese Umgebung konzipiert. Mit den weitverbreiteten WWW-Browsern von Netscape (Navigator 3.x, Communicator 4.x) unter Windows funktioniert der CAMPUS Lehr-/Lernsystemclient problemlos. Andere Plattformen und andere Browsertypen sind bisher noch nicht ausreichend getestet worden. Prinzipiell sollte jeder Browser, welcher die volle JDK 1.1 - Unterstützung bietet, als Client-Plattform geeignet sein.

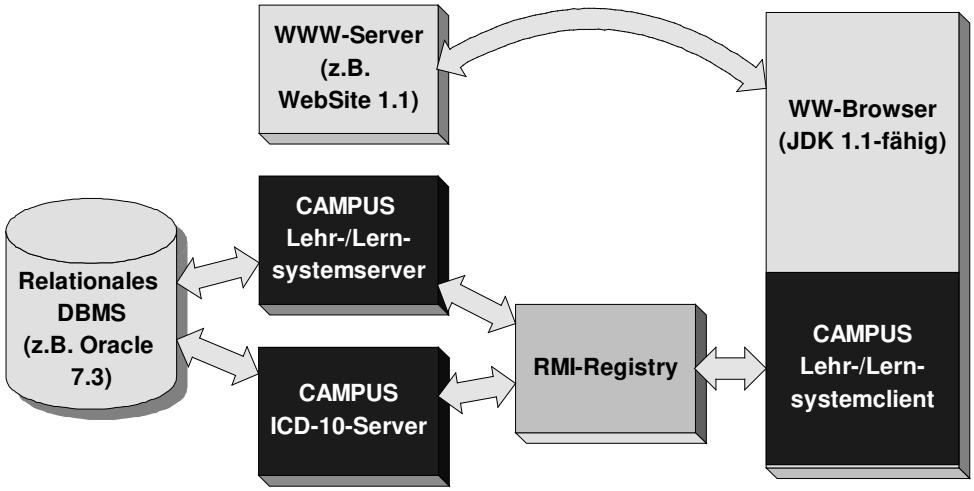

Abbildung 23: Einbettung der CAMPUS Lehr- und Lernsubsysteme in eine Standardsoftwareumgebung

Daten und Wissen werden im relationalen Datenbankmanagementsystem ORACLE 7.3 [HERRMAN, LENZ et al. 97] gespeichert. Die Entscheidung für ORACLE fiel, weil es zuverlässig sowie sehr leistungsfähig ist. Außerdem ist ORACLE für eine Vielzahl unterschiedlicher Hardwareplattformen erhältlich. Darüber hinaus steht mittlerweile mit ORACLE 8 ein objektrelationaler Nachfolger zur Verfügung. ORACLE 8 bietet die Möglichkeit, auf bestehende Relationen eine objektorientierte Sicht aufzusetzen. Das DBMS kann dann auf Anfragen komplexe Objekte zurückgeben. Der Zugriff von objektorientier-

ten Programmiersprachen wie Java auf die Datenbank wird dadurch vereinfacht. Weiterhin unterstützt ORACLE 8 abstrakte Datentypen. Bilder, Videos und Sounds können so in Zukunft wesentlich besser als bisher direkt in der Datenbank abgelegt werden.
Die Kommunikation zwischen CAMPUS Lehr-/Lernsystemserver und CAMPUS ICD-10-Server [RIEDEL 98] sowie CAMPUS Lehr-/Lernsystemclient wird über die mit dem JDK 1.1 mitgelieferte RMI-Registry abgewickelt (siehe Abbildung 23). Das CAMPUS-Lexikon ist auf einem kommerziellen WWW-Server (WebSite 1.1 von O'Reilly unter Windows NT) abgelegt und kann mittels praktisch aller verfügbaren WWW-Browser zugegriffen und gelesen werden. Die WWW-Serversoftware könnte bei Bedarf durch eine beliebige andere WWW-Serversoftware auf einer beliebigen Hard- und Softwareplattform ersetzt werden. WWW-Server, CAMPUS Lehr-/Lernsystemserver und CAMPUS ICD-10-Server können auf verschiedenen Rechnern laufen.

4.2 Autorensubsystem

Das Autorensubsystem ermöglicht es Fachgebietsexperten, auf komfortable Art und Weise Lehr-/Lernsysteme zu erstellen. Sie müssen dabei keinerlei Informatikkenntnisse aufweisen und können sich voll und ganz auf inhaltliche und didaktische Aspekte konzentrieren. Die Implementierung wurde in einem Softwarepraktikum des Studiengangs Medizinische Informatik begonnen und im Rahmen zweier Diplomarbeiten [SINGER 98; PELZER 98] weitergeführt. Daraus resultierte ein Prototyp, welcher allerdings noch der Weiterentwicklung bedarf. In der ersten Ausbauphase des Autorensubsystems werden diejenigen Systemteile realisiert, welche für die Erstellung von Fallsimulationen, für die Erstellung des CAMPUS-Lexikons und für die Erstellung von Fragen benötigt werden. Ziel ist es, den Fachgebietsautoren ein leicht zu bedienendes und robustes Autorensystem zur Verfügung zu stellen.

Für die Angabe differentialdiagnostischen Wissens stehen im Autorensubsystem die in Tabelle 5 aufgeführten Evidenzkategorien zur Verfügung. Wie bereits in Kapitel 3.5.3 angesprochen, ist die Angabe differentialdiagnostischen Wissens keine zwingende Voraussetzung für die Funktionsfähigkeit eines CAMPUS Lehr-/Lernsystems.

p3 = sichert	n3 = schließt aus
p2 = spricht stark für	n2 = spricht stark gegen
p1 = spricht für	n1 = spricht gegen
pn = neutral	

Tabelle 5: Evidenzkategorien im CAMPUS-System

Das CAMPUS-Lexikon besitzt die folgende (grobe) Gliederung (jeweils mit Unterpunkten):

- Krankheitsbeschreibungen [KNAUP 94]: Definition, Epidemiologie, Allgemeine Information, Pathogenese, Ätiologie, Anamnese, Symptome, Apparative Diagnostik, Labor, Histologische Befunde, Differentialdiagnosen, Therapie, Verlauf, Prognose, Prophylaxe
- Therapien (Beschreibung von Therapien)
- Anamnese (Darstellung verschiedener Anamnesetechniken und Anamnesefragen)
- klinische Untersuchung (Erläuterungen zur klinischen Untersuchung)
- technische Untersuchung (Beschreibung technischer Untersuchungsverfahren)
- Laboruntersuchung (Erläuterungen zu verschiedenen Labortests)

Basierend auf dieser Grobgliederung können die Fachgebietsautoren Lehrbuchwissen zur Verfügung stellen. Selbstverständlich steht es den Autoren frei, die oben vorgeschlagene Grobgliederung weiter zu verfeinern und zu ergänzen. Die Übernahme bereits verfügbarer HTML-Seiten ist dabei möglich [PELZER 98].

Autoren können mit dem Autorensubsystem komfortabel Wissenskontrollfragen und Fragen zur Unterstützung der Bearbeitung von Lehr-/Lernfällen erstellen. Dabei stehen insgesamt acht unterschiedliche Antworttypen zur Verfügung. Jeder dieser Antworttypen besitzt charakteristische Stärken und Schwächen (siehe Tabelle 6). Bzgl. der korrekten Anzeige und Auswertung der Fragen brauchen sich die Autoren keine Gedanken zu machen.

Antworttyp	Vorteile	Nachteile
Freie Eingabe	Lernende werden nicht durch Antwortvorgaben in ihren Antwortmöglichkeiten begrenzt.	Antwortanalyse problematisch, da häufig verschiedene richtige Antwortmöglichkeiten bestehen. Diese müssen vom Autor möglichst alle berücksichtigt werden.
Alternativ-antworten	Optimaler Antworttyp für Fragen, auf die es nur zwei mögliche Antworten gibt.	Nur für eine sehr eingeschränkte Zahl von Fragen geeignet. Nur kurze Entweder-Oder-Antwortvorgaben sind möglich.
Auswahlmenü	Gut geeignet für Fragen, auf die stichwortartig geantwortet werden kann.	Durch Antwortvorgaben wird die Beantwortung erleichtert und es können nur kurze Antwortmöglichkeiten angegeben werden.
Multiple-Choice	Antwortvorgaben können auch längere Texte umfassen.	Durch vorgegebene Antworten wird die Beantwortung erleichtert.
Matching	Gut geeignet, um die gegenseitige Zuordnung von Begriffen üben zu lassen.	Nur für eine sehr eingeschränkte Zahl von Fragen geeignet.
Bildbeschriftung	Gut geeignet, um bildbezogene Fragen zu stellen (z.B. in der Anatomie).	Frage ist aufwendig in der Erstellung, da ein Bild (muß evtl. erst noch eingescannt werden) vom Autor mit Hilfe eines Graphiktools mit Labels versehen werden muß.
Anordnen von Begriffen/Icons	Für die Abfrage von Abläufen, Arbeitsschritten usw. gut geeignet.	Nur für eine sehr eingeschränkte Zahl von Fragen geeignet.
Lückentext	Zur Abfrage von Merksätzen gut geeignet.	Nur für eine sehr eingeschränkte Zahl von Fragen geeignet.

Tabelle 6: Frage/Antworttypen im CAMPUS-System

Alle Fragen müssen von den Autoren einer von drei Schwierigkeitsstufen zugeordnet werden. Die Schwierigkeitsstufe *leicht* umfaßt grundlegende Sachverhalte, die jeder Medizinstudent ab einem bestimmten Semester beantworten können sollte und die kein tieferes Verständnis eines Fachgebietes voraussetzen. In der Menge der *schweren* Fragen befinden sich alle Fragen, die nur nach gründlicher Beschäftigung mit einem Fachgebiet beantwortet werden können, bzw. die ein tieferes Verständnis für ein Fachgebiet voraussetzen. Alle Fragen, die nicht in diese beiden Kategorien eingeteilt werden können, besitzen die Schwierigkeitsstufe *mittel*.
Zum derzeitigen Entwicklungsstand des CAMPUS-Systems kann der Autor Fragen einer Vielzahl verschiedener Situationen und Objekte zuweisen. Für jede Frage muß explizit angegeben werden, ob sie sich auf ein Kapitel, Unterkapitel oder Knoten im Lexikon, auf einen konkreten Lehr-/Lernfall bezieht oder aber fallübergreifend verwendet werden kann.

4.3 Lernsubsystem

Das Lernsubsystem stellt die Benutzungsschnittstelle des CAMPUS-Systems für die Lernenden bereit und ist vollständig auf dem Client-Rechner angesiedelt. Vor der Arbeit mit CAMPUS muß sich ein Nutzer beim System anmelden und eines der verfügbaren CAMPUS Lehr-/Lernsysteme auswählen. Anwender können danach zum derzeitigen Stand des Projektes entscheiden, ob sie Fallsimulationen bearbeiten, Lehr-/Lernfälle suchen, Fragen beantworten oder im CAMPUS-Lexikon lesen möchten. Darüber hinaus können Sie das CAMPUS-System adaptieren (siehe Abbildung 24) oder die Systemhilfe aufrufen.

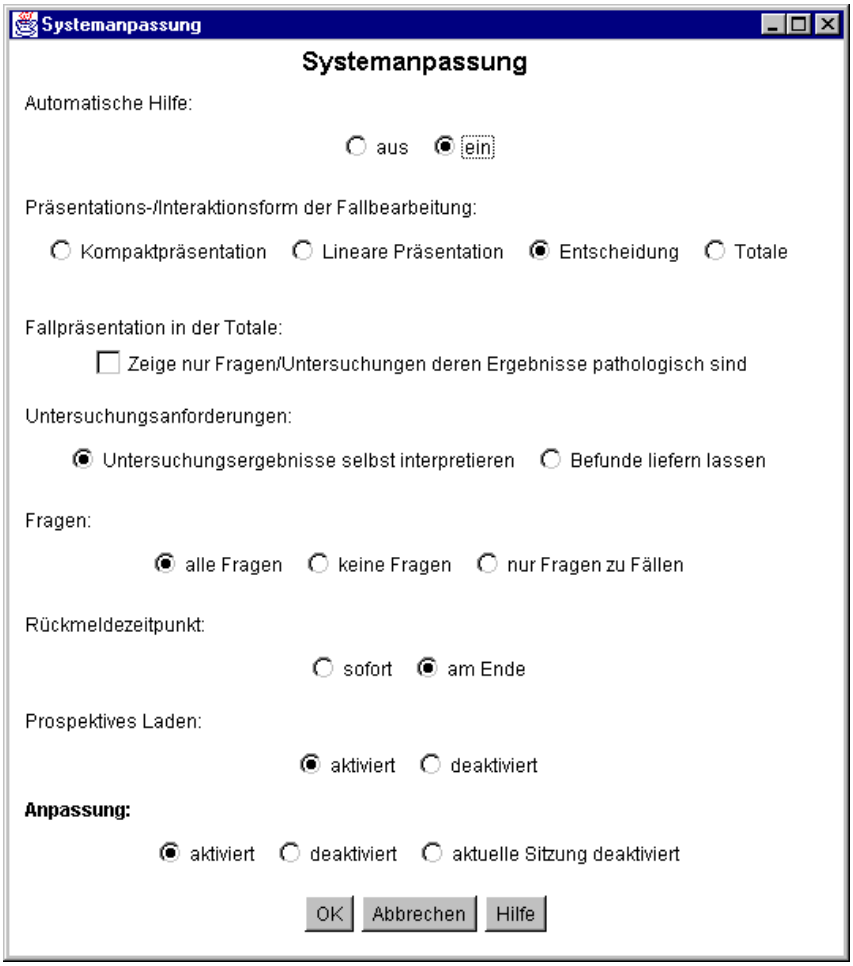

Abbildung 24: Adaptierung des CAMPUS-Systems

Das Lexikon stellt für die Lernenden ein strukturiertes (Hypertext)-Lehrbuch dar. In der Regel wird es dazu verwendet werden, um bei der Bearbeitung von Fallsimulationen oder bei der Beantwortung von Fragen schnell einzelne Details nachlesen zu können (z.B. über Symptom-Diagnose-Zusammenhänge, die Häufigkeit bestimmter Diagnosen oder die Vor- und Nachteile bestimmter Untersuchungsverfahren). Die Nutzer haben so jederzeit Zugriff auf systematisches Wissen. Autoren können an beliebigen Stellen des Fallablaufs Verweise auf bestimmte Stellen im Lexikon angeben. Dadurch können sie die Nutzer gezielt zu bestimmten Informationen führen und so die systematische Bearbeitung von Fällen unterstützen.

Um der Problematik der Freitexteingaben von Diagnosen zu begegnen, wurde in einer Diplomarbeit [RIEDEL 98] ein ICD-10-Server implementiert. Dieser Server ermöglicht es, Diagnosen als Freitext einzugeben und dann aus den gefundenen ICD-10-Drei- und Vierstellern die gewünschte Diagnose herauszusuchen. Die Beurteilung der von den Nutzern gewählten Diagnosen ist dadurch wesentlich besser möglich, als dies bei einer reinen Freitexteingabe möglich wäre. Es kann eine Rückmeldung gegeben werden, ob eine gewählte Verdachts- bzw. Arbeitsdiagnose korrekt, falsch, zu allgemein oder zu speziell ist. Der ICD-10-Server steht natürlich nicht nur im Lernsubsystem zur Verfügung sondern auch im Autorensubsystem. Die Autoren können damit komfortabel die zu einem Lehr-/Lernfall gehörigen Diagnosen nach ICD-10 verschlüsseln. Der Verschlüsselungsdialog ist in Abbildung 25 dargestellt.

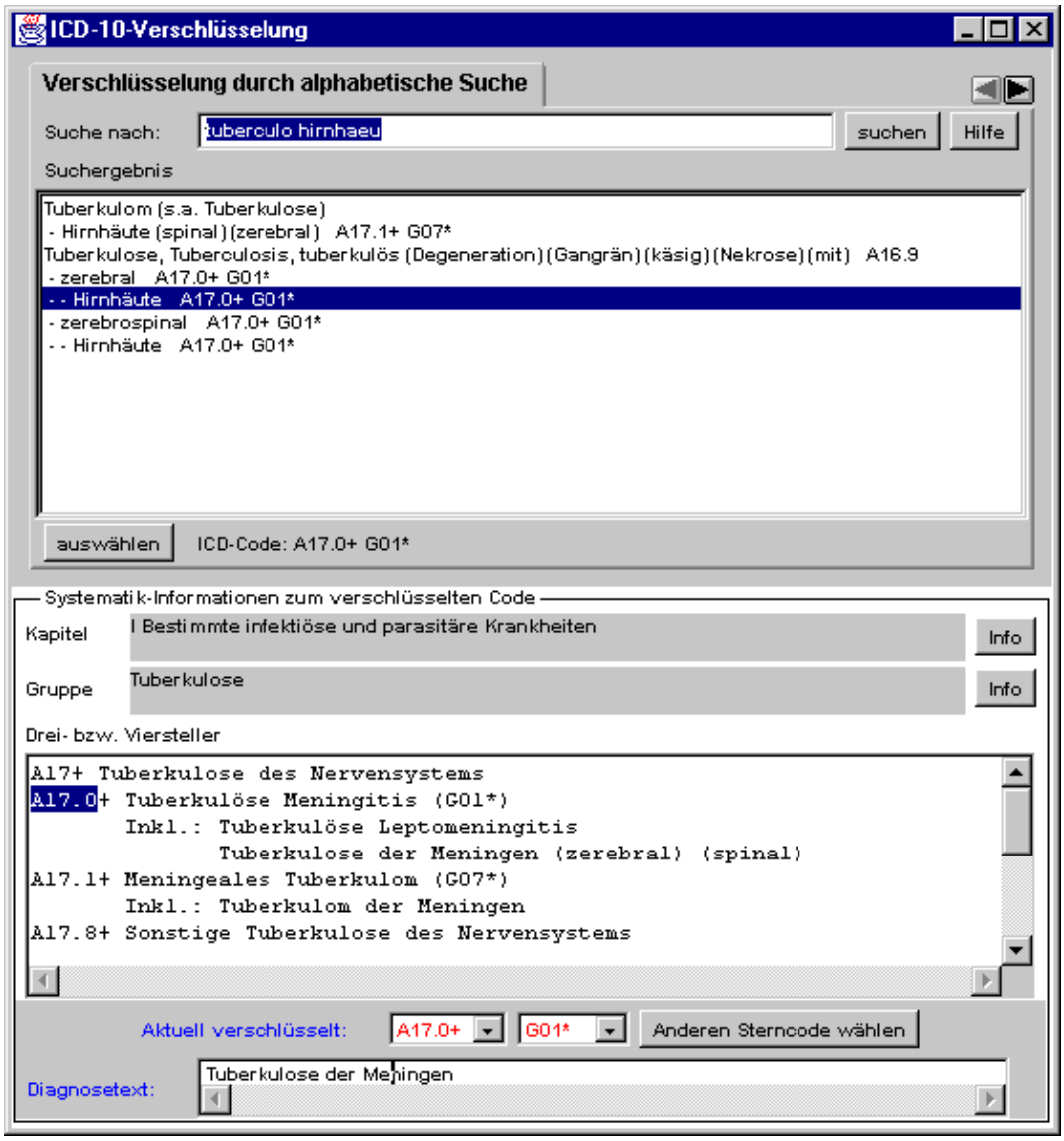

Abbildung 25: Dialog zum Verschlüsseln von Diagnosen nach ICD-10

4.4 Lehrsubsystem

Das Lehrsubsystem generiert geeignete Rückmeldungen auf Nutzeraktionen und ist für die Adaption des CAMPUS-Systems an seine Nutzer verantwortlich. Außerdem sorgt es für den „korrekten“ Fallablauf auf Basis des in Kapitel 3.4.4 vorgestellten Fallablaufmodells. Eine Instanz der Klasse LEHRER ist das zentrale Objekt des Lehrsubsystems von CAMPUS. Ein Lehrer (Klasse LEHRER) besteht aus verschiedenen Objekten, die jeweils für spezifische Aufgaben verantwortlich sind und teilweise auf dem CAMPUS Lehr-/Lernsystemserver und teilweise auf dem CAMPUS Lehr-/Lernsystemclient angesiedelt sind.

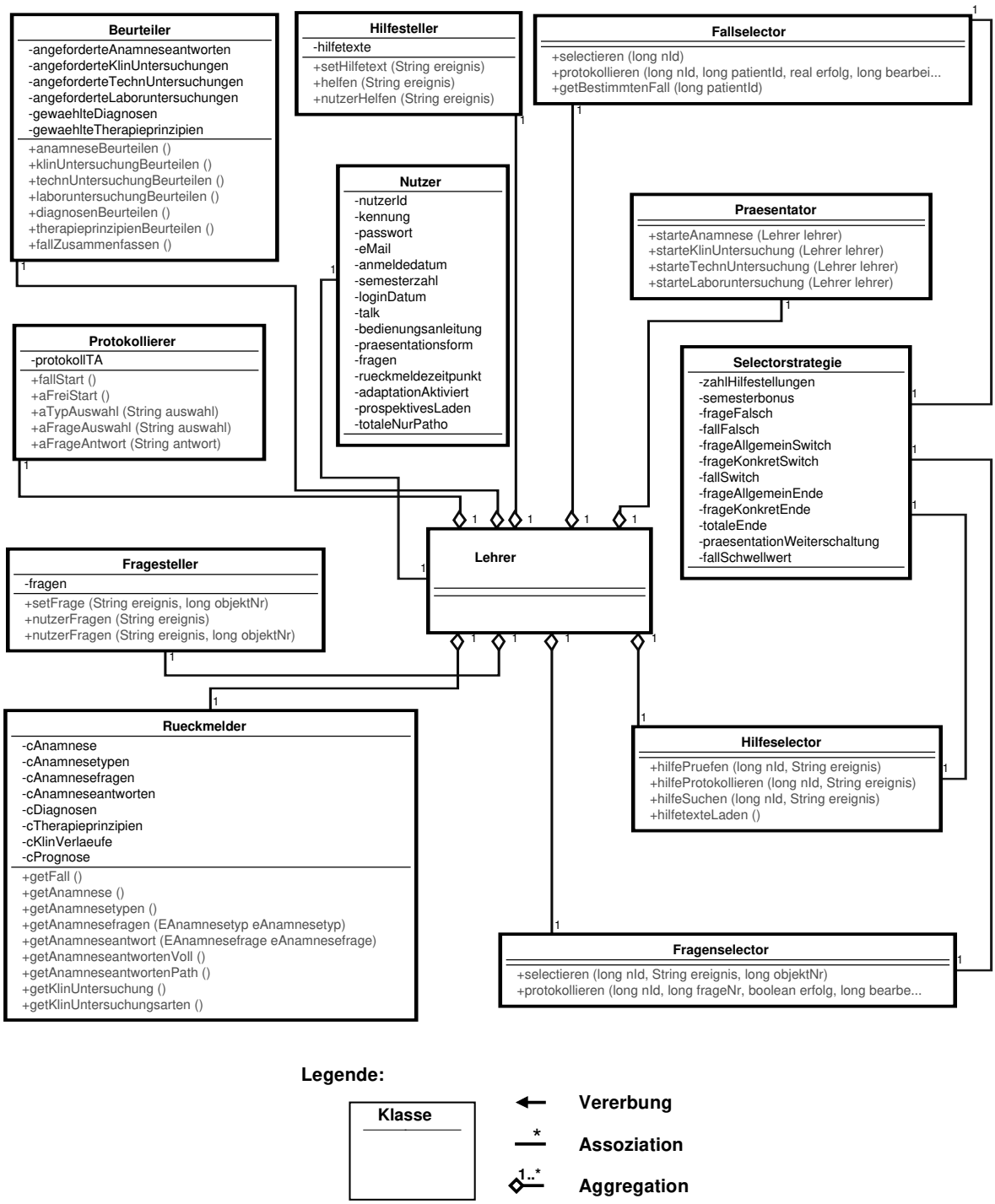

Abbildung 26: Modell des Lehrsubsystems

Frageselektor, Fallselektor und Hilfeselektor sind Instanzen der Klassen FRAGESELECTOR, FALLSELECTOR und HILFESELECTOR. Sie sind auf dem Lehr-/Lernsystemserver angesiedelt. Die Aufgabe des Frageselektors besteht darin, geeignete Fragen auszuwählen und den Erfolg bei der Beantwortung der Fragen samt Zeitbedarf für die Beantwortung in der Datenbank zu protokollieren. Eine vergleichbare Aufgabe besitzt der Fallselektor. Er wählt aus den verfügbaren Lehr-/Lernfällen den für den einzelnen Nutzer am besten geeigneten Lehr-/Lernfall aus und protokolliert den Erfolg bzw. Mißerfolg bei der Bearbeitung samt Bearbeitungsdauer. Der Hilfeselektor wählt zu bestimmten Ereignissen die zugeordneten Hilfetexte aus und übermitteln sie an den Client. Darüber hinaus protokolliert er jedes Anzeigen eines Hilfetextes. Die Kriterien, nach denen die angesprochenen „Selektoren“ ihre Auswahl treffen, richtet sich nach den in einem Objekt vom Typ SELECTORSTRATEGIE enthaltenen Merkmalen (siehe hierzu auch Kapitel 3.4.3). Diese stammen aus einer parametrisierbaren Textdatei, welche beim Start des CAMPUS Lehr-/Lernsystemservers eingelesen wird und von den Autoren verändert werden kann.
Alle weiteren Objekte des Lehrsubsystemmodells (siehe Abbildung 26) sind auf dem Client angesiedelt. Der Beurteiler (Klasse BEURTEILER) beurteilt alle Aktionen des Nutzers und errechnet den Fallerfolg (Berechnung siehe Kapitel 3.4.3). Er prüft u.a., welche Untersuchungen vergessen wurden bzw. unnötigerweise gestellt wurden oder welche Verdachts- und Arbeitsdiagnosen korrekt, falsch, zu allgemein oder zu speziell gewählt wurden. Der Hilfesteller (Klasse HILFESTELLER) wird aktiv, wenn ein Nutzer Hilfe anfordert, bzw. wenn die Lernsituation die automatische Hilfestellung des Systems erforderlich macht. Dann blendet er automatisch den zugeordneten Hilfetext ein. Falls der Hilfetext in der Cachingschicht des Lehr-/Lernsubsystems noch nicht verfügbar ist, fordert er ihn vom Frageselektor auf dem Lehr-/Lernsystemserver an. Der Fragesteller (FRAGESTELLER) wird während der Fallbearbeitung aktiv, wenn aufgrund der Eingruppierung des Nutzers Fragen gestellt werden sollen. Falls eine benötigte Frage noch nicht prospektiv auf den Lehr-/Lernsystemclient geladen worden ist, fordert der Fragesteller eine geeignete Frage beim Frageselektor auf dem Server an. Der Präsentator (PRAESENTATOR) ist für die Präsentation eines Lehr-/Lernfalles entsprechend der eingestellten Präsentaions-/Interaktionsform zuständig. Der Rückmelder (RUECKMELDER) ist dafür verantwortlich, auf Nutzeraktionen eine sofortige Rückmeldung zu geben, falls die aktuelle Ausprägung des Nutzermodells (Merkmal *rueckmeldezeitpunkt* in Klasse NUTZER) dies verlangt. Darüber hinaus enthält der Rückmelder den Großteil der Objekte der Cachingschicht. Die Namen aller Caching-Objekte beginnen mit „c“. Bei den in der Klasse RUECKMELDER aufgeführten Merkmalen (siehe Abbildung 26) handelt es sich also ausschließlich um Objekte der Caching-Ebene des Lehr-/Lernsystemclient. Bei aktiviertem prospektivem Laden (Merkmal *prospektivesLaden* in der Klasse NUTZER) füllt ein Loader-Objekt im Hintergrund sukzessive diese Cachingschicht, wodurch die Reaktionszeiten des Systems bei langsamen Netzverbindungen stark beschleunigt werden. Und dies, ohne daß der Nutzer bei der Arbeit mit dem System beeinträchtigt wird. Die Protokollierung aller Nutzeraktivitäten wird vom einem Objekt des Typs PROTOKOLLIERER übernommen. Er zeigt in seinem Fenster alle vom Nutzer durchgeführten Aktionen an. Die Nutzer haben dadurch einen besseren Überblick über den Verlauf einer Fallsimulation.
Abbildung 26 zeigt aus Gründen der Übersichtlichkeit bei weitem nicht alle in den einzelnen Klassen vorhandenen Merkmale und Methoden an. Für eine detailliertere Beschreibung wird auf die Softwareentwicklungsdokumentation zu CAMPUS verwiesen.

4.5 CAMPUS/Pädiatrie und CAMPUS/Infektiologie

Um die im Rahmen dieser Arbeit entworfenen Konzepte zu erproben, werden Lehr-/Lernsysteme für die Pädiatrie (CAMPUS/Pädiatrie) und die Infektiologie (CAMPUS/Infektiologie) entwickelt.
Der prototypische CAMPUS Lehr-/Lernsystemserver umfaßt zum jetzigen Zeitpunkt etwa 5000 Codezeilen und wird aus Instanzen von 20 unterschiedlichen Klassen gebildet, die ca. 120 Methoden besitzen. Der CAMPUS ICD-10-Server besteht aus ca. 1400 Zeilen Code (10 Klassen). Der Code des CAMPUS Lehr-/Lernsystemclient hat einen Umfang von ca. 14000 Zeilen. Der Client wird aus Instanzen von ca. 160 Klassen (davon ca. 70 Klassen für die Bildung der Cachingschicht) gebildet. Die Client-Klassen besitzen ca. 250 Methoden zuzüglich ca. 220 Methoden der Cachingschicht.
Für den Fall, daß man bei der Implementierung statt einer *Distributed Teaching*-Architektur eine *Remote Data & Knowledge*-Architektur zugrunde legt, vergrößert sich der Umfang des Client von 14000 auf ca. 20400 Zeilen Code, d.h. in diesem Fall müßte ca. 45% mehr Code über das Netz übertragen werden. Dadurch würde sich die Wartezeit beim Laden des Systems auch um 45% verlängern. Diese Abschätzung geht davon aus, daß sich der Umfang der implementierten Kommunikationsmechanismen zwischen Server und Client bei beiden Architekturtypen nicht wesentlich unterscheidet. Da bei der *Remote Data & Knowledge*-Architektur Client und Server häufiger miteinander kommunizieren müssen (die gesamte Lehrsystemlogik ist auf dem Client angesiedelt), ergeben sich zusätzlich verlängerte Wartezeiten. Wie stark diese verlängerten Wartezeiten letztlich ins Gewicht fallen, ist von der verfügbaren Netzbandbreite abhängig. Bei langsamen Netzverbindungen ergeben sich durch die *Distributed Teaching*-Architektur große Performancevorteile.
CAMPUS/Pädiatrie entsteht in Kooperation mit der Universitätskinderklinik Heidelberg (OA PD Dr. B. Tönshoff) und der Universitätskinderklinik in Freiburg (OA PD Dr. Zimmerhackl). Ziel des Projektes ist die Bereitstellung pädiatrischer Lehr-/Lernfälle, Lernfragen und von Lehrbuchwissen. Der Einsatz des Systems ist in Heidelberg im Rahmen des Praktikums der Kinderheilkunde Teil 1 und Teil 2 vorgesehen.
CAMPUS/Infektiologie entsteht in Zusammenarbeit mit dem Hygiene-Institut der Universität Heidelberg (Prof. Dr. H.-K. Geiss). Auch hier sollen Lernfälle, Lernfragen und Lehrbuchwissen bereitgestellt werden. Im Unterschied zu CAMPUS/Pädiatrie wurde als zusätzliche Anforderung eine Suchmöglichkeit nach Lehr-/Lernfällen (siehe Abbildung 27) definiert [RIEDEL 98]. CAMPUS/Infektiologie soll im Rahmen von Lehrveranstaltungen eingesetzt werden, aber auch von niedergelassenen Ärzten als Informationssystem genutzt werden können. Durch das Shellkonzept von CAMPUS steht die Fallsuche natürlich auch in CAMPUS/Pädiatrie zur Verfügung.

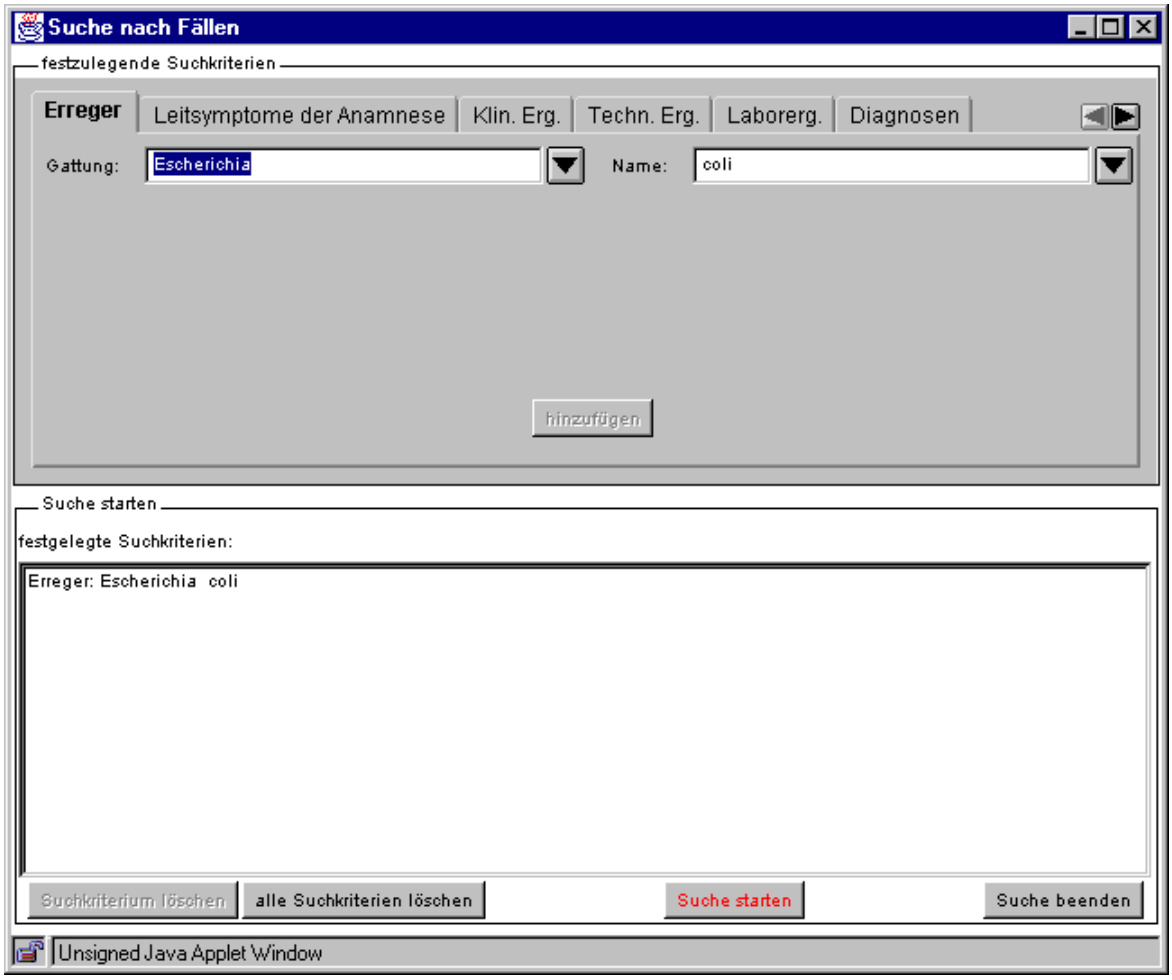

Abbildung 27: Suche nach Lehr-/Lernfällen in CAMPUS

5 Zusammenfassung und Diskussion

Der Forschungsschwerpunkt im Bereich des Computer-based Training in der Medizin hat sich in den letzten Jahren in Richtung internetbasierter Lehr-/Lernsysteme verlagert. Im Zuge der Forschungsarbeiten sind eine ganze Reihe sehr interessanter und innovativer Anwendungen entstanden (z.B. [BITTORF, BAUER et al. 97; HAYES, LEHMANN 96; THE WHOLE BRAIN ATLAS]). Allerdings werden die Möglichkeiten des Internets und der darauf aufbauenden Anwendungen und Standards bisher bei weitem noch nicht ausgeschöpft [FRIEDMAN 96]. Teilweise werden lediglich bereits existierende Lehr-/Lerneinheiten praktisch unverändert ins Internet gebracht. Internet und WWW bieten jedoch vielfältige Möglichkeiten, um Lehr-/Lernsysteme mit große Vorteilen gegenüber konventionellen CBT-Systemen aber auch gegenüber anderen Lehr-/Lernformen [CHODOROW 96] zu konzipieren und zu implementieren.

Ziel der vorliegenden Arbeit war die „Erstellung eines innovativen Konzeptes für die Realisierung von in der medizinischen Aus- und Weiterbildung eingesetzten Lehr-/Lernsystemen“ sowie dessen prototypische Umsetzung. Die im Zusammenhang mit diesem Ziel gestellten Fragen (siehe Kapitel 1.3) sollen nachfolgend beantwortet werden:

Zu Frage 1.1: „Welche konzeptuellen Merkmale muß ein Lehr-/Lernsystem aufweisen, um die in Kapitel 1.2 beschriebenen Probleme konventioneller CBT-Systeme zu lösen?“

Die „klassischen“ Probleme konventioneller CBT-Systeme (Plattformabhängigkeit, Notwendigkeit einer Installation, aufwendiges Update) werden durch die Wahl eines internetbasierten Ansatzes unter Einsatz von WWW-Standards und der Programmiersprache Java gelöst. Auf Anwenderseite ist als Voraussetzung lediglich ein Internet-Zugang sowie ein installierter WWW-Browser mit JDK 1.1-Unterstützung notwendig. Auch ein weiteres in Kapitel 1.2 angesprochenes Problem, daß Studierende beim Selbststudium mit konventionellen CBT-Systemen auf sich alleine gestellt bleiben, wenn Sie Fragen mit dem in einem Lehr-/Lernsystem vorhandenen Wissen nicht lösen können, wird durch einen WWW-basierten Ansatz beseitigt. Mit geringem (Implementierungs-)Aufwand können den Lernenden symmetrische und asymmetrische Kommunikationsmöglichkeiten (Electronic Mail, Newsgruppen, Diskussionslisten, IRC usw.) für die Diskussion auftretender Fragen mit ihren Dozenten und Mitstudenten geboten werden. Darüber hinaus ist bei WBT-Systemen eine Integration externer Ressourcen möglich. Entwickler von Lehr-/Lernsystemen können dadurch ohne großen Aufwand Möglichkeiten für die Lernenden schaffen, das erworbene Wissen individuell zu vertiefen. Die zwangsläufig vorhandenen Grenzen konventioneller Systeme können so stark erweitert werden. Die bei vielen existierenden Lehr-/Lernsystemen fehlende Adaption des Systems an den Nutzer bzw. die Adaptierbarkeit des Systems durch den Nutzer wurde im Konzept ebenfalls berücksichtigt. Eine Adaptierung der Präsentations-/Interaktionsformen bei Fallsimulationen, der Schwierigkeitsstufe von Fragen und Lehr-/Lernfällen, des Rückmeldezeitpunkts auf Nutzeraktionen und der automatischen Systemhilfe ist vorhanden. Die einzelnen Parameter können jederzeit von den Nutzern auch selbst adaptiert werden. Unzufriedenheit der Nutzer mit einer evtl. unzureichenden Adaptionsleistung des Systems kann dadurch vorgebeugt werden.
Dem großen Problem der mangelnden Wiederverwendbarkeit von Lehr-/Lerninhalten und Lehr-/Lernsystemfunktionalität in vielen existierenden CBT-Systemen wurde besondere

Aufmerksamkeit geschenkt. So wird auf eine strikte Trennung von Inhalten und Systemlogik geachtet. Nur dadurch ist ein einfacher Export von Lehr-/Lerninhalten und die Wiederverwendung einzelner Systembestandteile möglich. Die Wiederverwendbarkeit von Programmcode wird durch die Verwendung der objektorientierten Programmiersprache Java erleichtert. Die Struktur der Lehr-/Lerninhalte wurde ausführlich modelliert und dient als Basis für die vorgeschlagene Lehr-/Lernsystemshelllösung. Dadurch wird einerseits der Forderung nach Wiederverwendbarkeit Rechnung getragen, andererseits ermöglicht eine solche Shell die schnelle Erstellung von Lehr-/Lernsystemen durch Medizindozenten. Von besonderer Bedeutung ist in diesem Zusammenhang das Vorhandensein eines intuitiv zu bedienenden Autorensystems. Durch den Shellansatz kann auch das letzte in Kapitel 1.2 angesprochene Problem, die kritische Haltung vieler Dozenten gegenüber nicht unter ihrer Mitarbeit entwickelten CBT-Systemen, angegangen werden. Dozenten können Systeme erstellen, die genau ihren persönlichen Anforderungen entsprechen und dabei auf bereits verfügbare Inhalte und die Funktionalität der Shell zurückgreifen.

Zu Frage 1.2: „Wie kann eine häufigere Nutzung von Lehr-/Lernsystemen durch die Studierenden erreicht werden?“

Damit die Studierenden CBT-Systeme einsetzen, müssen zwei Voraussetzungen erfüllt sein. Sie müssen davon überzeugt sein, daß sie durch die Nutzung solcher Systeme in den Prüfungen profitieren und es muß für sie ohne große organisatorische Schwierigkeiten möglich sein, solche Systeme zu nutzen. Idealerweise sollte die Nutzung von CBT-Systemen im Curriculum verankert sein und die Studierenden sollten auch auf ihrem eigenen Rechner zu Hause solche Systeme nutzen können. Darüber hinaus müssen aber auch an der Hochschule eine hinreichende Anzahl von Rechnern für die computerunterstützte Ausbildung bereitstehen. Und dies auch außerhalb der Kernvorlesungszeiten, denn bei weitem nicht alle Medizin-Studierenden besitzen einen eigenen Computer. Durch die vorgestellte Architektur des CAMPUS-Systems ergibt sich eine sehr große Zahl von Rechnern, auf denen prinzipiell CAMPUS Lehr-/Lernsysteme ohne Notwendigkeit einer Installation genutzt werden können.

Zu Frage 1.3: „Wie kann die Akzeptanz von Lehr-/Lernsystemen bei den Medizindozenten verbessert werden?“

Selbstentwickelte CBT-Systeme bzw. Systeme, bei denen Dozenten aktiv bei der Erstellung beteiligt waren, besitzen gegenüber kommerziellen Produkten eine deutlich höhere Akzeptanz. Leider ist die Entwicklung kompletter Lehr-/Lernsysteme sehr aufwendig und teuer. Der vorgeschlagene Shellansatz ermöglicht die schnelle Erstellung von „maßgeschneiderten“ CBT-Systemen und steigert dadurch die Akzeptanz von Lehr-/Lernsystemen bei den Medizindozenten.

Zu Frage 1.4: „Wie können Dozenten dazu gebracht werden, verstärkt in ihren Lehrveranstaltungen CBT-Systeme einzusetzen und ihren Studenten die Arbeit mit CBT-Systemen zu empfehlen?“

Eine Änderung der derzeitigen Approbationsordnung könnte dafür sorgen, daß den Studenten mehr Zeit für das Selbststudium bliebe. Dozenten könnten dann ihren Studierenden in den Lehrveranstaltungen verstärkt Empfehlungen für das Selbststudium, auch mit Lehr-/Lernsystemen, geben. Allerdings ist mit einer solchen Änderung zur Zeit nicht zu rechnen, obwohl sich alle Beteiligten darüber einig sind, daß das Medizinstudium dringend reformbedürftig ist.
Auf jeden Fall müssen Lehr-/Lernsysteme wesentlich besser verfügbar sein, als dies bisher der Fall ist. Durch den in dieser Arbeit vorgestellten Ansatz sind Lehr-/Lernsysteme auf sehr vielen vernetzten Computern, sogar bei den Studierenden und Dozenten zu Hause, nutzbar. Die vorgeschlagene Lehr-/Lernsystemshell besitzt ein internetbasiertes, plattformunabhängiges Autorensubsystem. Dadurch können die Dozenten eigene Lehr-/Lernsysteme bequem an ihrem eigenen Arbeitsplatz entwickeln.

Zu Frage 1.5: „Inwieweit kann das Konzept fachgebietsunabhängig gehalten werden?“

Das Konzept konnte weitgehend fachgebietsunabhängig gehalten werden. Das CAMPUS-Lexikon ermöglicht es den einzelnen Dozenten, Domänenwissen nach ihren eigenen Wünschen und entsprechend den speziellen Anforderungen eines Fachgebietes darzustellen. Das entwickelte Fragenmodell ermöglicht die Erstellung von Fragen für beliebige Fachgebiete. Im Normalbefundemodell wurden die Anforderungen verschiedener Fachgebiete dadurch berücksichtigt, indem Anamnesefragen einzelnen Fachgebieten zugeordnet werden können. Falls erforderlich können auch einzelne Labortests oder technische Untersuchungsverfahren Fachgebieten zugeordnet werden. Das Normalbefundemodell ist damit weitgehend fachgebietsunabhängig. Das Fallmodell ist in seiner Grundstruktur (eine Lehr-/Lernfall besteht aus Anamnese, klinische Untersuchung, technischen Untersuchungen...) für verschiedenste Fachgebiete einsetzbar. Es muß allerdings bei Bedarf um fachgebietsspezifische Aspekte erweitert werden. Für die Infektiologie wurde beispielsweise der Objekttyp ERREGER hinzugefügt. Dies war allerdings die einzige Erweiterung, die am Fallmodell vorgenommen werden mußte, damit dieses nicht nur für die Speicherung pädiatrischer sondern auch für infektiologische Lehr-/Lernfälle geeignet war. Die weiteren in Kapitel 3.5 beschriebenen Modelle sind vom Fachgebiet unabhängig.

Das CAMPUS-System befindet sich zur Zeit im Prototypenstadium. Eine Demoversion des Lehr-/ und Lernsubsystems steht unter der URL http://www.hyg.uni-heidelberg.de/campus zur Verfügung. Durch eine Beteiligung am Landesforschungsprojekt VIROR ist die Weiterentwicklung für drei Jahre gesichert und es kann eine gründliche Evaluation vorbereitet und durchgeführt werden. Ein abschließende Beurteilung der Konzepte kann erst erfolgen, nachdem das System durch systematische Evaluation mit Unterstützung von Medizindozenten und Medizinstudenten evaluiert worden ist.
Besonderes Augenmerk bei der Weiterentwicklung des CAMPUS-Systems benötigt das Autorensubsystem. Schließlich wird das CAMPUS-System nur dann von den Medizindozenten akzeptiert werden, wenn es Ihnen möglich ist, ohne lange Einarbeitungszeit und mit vergleichsweise geringem Zeitaufwand Lehr-/Lernsysteme zu erstellen. Dies stellt hohe Anforderungen an das Autorensubsystem von CAMPUS.
Schwerpunkt bei der Weiterentwicklung des Lernsubsystems muß die Benutzungsschnittstelle sein. Nur durch ein attraktives und intuitiv zu bedienendes User-Interface ist eine hohe Akzeptanz der erstellten Systeme durch die Nutzer zu erreichen.

6 Literatur

AAMC (Association of American Medical Colleges) (1984): *Physicians for Twentyfirst Century: The GPEP-Report*. Report of the Panel on the General Professional Education of the Physician and College Preperation for Medicin. Washington DC.

ÄAppO (1989): Approbationsordnung für Ärzte vom 28.10.1970 in der Fassung vom 21.12 1989 (BGBl I S. 2549). In: THÜRK: *Recht im Gesundheitswesen, Lfg. 52, April 90*

ADLER, M.; DIETRICH, J.W.; HOLZER, M.F.; FISCHER, M.R. (Hrsg.) (1998): *Computer Based Training in der Medizin: Technik - Evaluation - Implementation*. Aachen: Shaker.

ALBANESE, M.A.; MITCHELL, S. (1993): Problem-based learning. A Review of Literature on Its Outcomes and Implementation Issues. *Academic Medicine* **68**(1), 52-81.

APPLE (1992). *Macintosh Human Interface Guidelines*. New York: Addison-Wesley.

APPLE (1993): *HyperCard Reference Manual*. Cupertino: Apple.

ARBEITSKREIS MEDIZINERAUSBILDUNG D. ROBERT BOSCH STIFTUNG (1995): *Das Arztbild der Zukunft: Analysen zukünftiger Anforderungen an den Arzt; Konsequenzen für die Ausbildung und Wege zu ihrer Reform*. 3., vollst. Überarb. Aufl., Gerlingen: Bleicher.

AUHUBER, T.C. (1998): *Entwicklung und Evaluation eines computergestützten Lernsystems in der Medizin. MicroPat – ein interaktiver Atlas der Histopathologie mit adaptierbarem Tutor*. Europäische Hochschulschriften VII/D/31. Frankfurt am Main; Berlin: Lang.

BALZERT, H. (1996): *Lehrbuch der Software-Technik: Software-Entwicklung*. Heidelberg; Berlin; Oxford: Spektrum.

BAUMGARTNER, P.; PAYR, S. (1994): *Lernen mit Software*. Innsbruck: Österreichischer Studien-Verlag.

BAUR, M.P.; MICHAELIS, J. (Hrsg.) (1990): *Computer in der Ärzteausbildung*. München; Wien: Oldenbourg.

BERNES-LEE, T.; CAILLEAU, R.; GROFF, J.F.; POLLERMANN, B. (1992): World Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy*, Spring **1**(2).

BEUN, R.-J.; BAKER, M.; REINER, M. (eds.) (1995): *Dialogue and Instruction*. Berlin; Heidelberg: Springer.

BIERMAN, D.J.; KAMSTEEG, P.A.; SANDBERG, J.A.C. (1992): Student Models, Scratch-Pads, and Simulation. In: [COSTA (ed.) 92], 135-145.

BITTORF, A.; BAUER, J.; SIMON, M.; DIEPGEN, T.L. (1997): Web-based Training Modules in Dermatology. *MD Comput*. **14**(5), 371-376.

BLOOM, B.S. (1972): *Taxonomie von Lehrzielen im kognitiven Bereich*. Weinheim: Belz.

BOB, A.; ERDINGER, L. (1994): *Original-Prüfungsfragen mit Kommentar GK III: Allgemeinmedizin und ökologisches Stoffgebiet*. London; New York; Weinheim: Chapman&Hall.

BODENDORF F. (1990): *Computer in der fachlichen und universitären Ausbildung*. München; Wien: Oldenbourg.

BOOCH, G. (1994): *Objektorientierte Analyse und Design*. Bonn; Paris: Addison-Wesley.

BUNDESMINISTERIUM FÜR GESUNDHEIT (Hrsg.) (1993): *Diskussionsentwurf eines Gesetzes zur Änderung der Bundesärzteordnung und zur Änderung der Approbationsordnung für Ärzte*. Bonn: Typoskript.

BURKHARDT, R. (1997): *UML - Unified Modeling Language: Objektorientierte Modellierung für die Praxis*. Bonn; Paris: Addison-Wesley.

CHEIKES, B.A.; RAGNEMALM, E.L. (1995): Simulator-Based Training-Support Tools for Process-Control Operators. In: [BEUN, BAKER, REINER (eds.) 95], 85-101.

CHEN, P.P.S. (1976): The Entity-Relationship Model - Towards a Unified View of Data. *ACM Transactions on Database Systems* **1**(1), 9-36.

CHIZZALI-BONFADIN, C.; ADLASSNIG, K.P.; KREIHSL, M.; HATVAN, A.; HORAK, W. (1997): A WWW-Accessible Knowledge Base for the Interpretation of Hepatitis Serologic Tests. *International Journal of Medical Informatics* **47**(1-2), 57-60.

CHODOROW, S. (1996): Educators Must Take the Electronic Revolution Seriously. *Academic Medicine* **71**(3), 221-226.

CIMINO, J.J.; SOCRATOUS, S.A.; CLAYTON, P.D. (1995): Internet as Clinical Information System: Application Development Using the World Wide Web. *JAMIA* **2**(5), 273-284.

COAD, P.; YOURDON, E. (1991): *Object-Oriented Analysis*. Englewood Cliffs: Yourdon Press, Prentice Hall.

CODD, E.F. (1970): A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM* **13**, 377-387.

CONKLIN, J. (1987) : Hypertext - An Introduction and a Survery. *IEEE Comput.* **20**(9), 17-41.

CONRADI, H.; KREUTZ, R.; SPITZER, K. (Hrsg.) (1997): *CBT in der Medizin - Methoden, Techniken, Anwendungen* -. Aachen: Verlag der Augustinus Buchhandlung.

COSTA, E. (ed.) (1992): *New Directions for Intelligent Tutoring Systems*. Berlin; Heidelberg: Springer.

COVELL, D.G.; UMAN, G.C.; MANNING, P.R. (1985): Information Needs in Office Practice: Are They Being Met? *Annals of Internal Medicine* **103**(4), 596-599.

DAHMER, J. (1994): *Anamnese und Befund: Die ärztliche Untersuchung als Grundlage klinischer Diagnostik*. 7. Aufl., Stuttgart; New York: Thieme.

DALITZ, W., HEYER G. (1995): *Hyper-G: das Internet-Informationssystem der 2. Generation*. Heidelberg: dpunkt.

DETMER, W.M.; SHORTLIFE, E.H. (1997): Using the Internet to Improve Knowledge Diffusion in Medicine. *Communications of the ACM* **8**, 101-108.

EFFELSBERG, W. (1995): Das Projekt Teleteaching der Universitäten Heidelberg und Mannheim. *PIK 18(4)*. München: Saur-Verlag.

EITEL, F (1993a): Die Studienreform ist tot, es lebe die Studienreform. *Medizinische Ausbildung* **10**(2), 114-122.

EITEL, F. (1993b): Die Ausbildungsmisere. In: SCHWEIBERER, L.; IZBICKI, J.R.: *Akademische Chirurgie: Aus-, Weiter- und Fortbildung; Analysen und Perspektiven*. Berlin; Heidelberg: Springer, 123-132.

ENGLER, C.; FÜHRER, A.; HEMPEL, P.; BUSCHER, H.-P. (1995): HEPA-CADS: Entwicklung eines tutoriell nutzbaren Expertensystems zur Diagnostik von Lebererkrankungen. *Informatik, Biometrie und Epidemiologie in Medizin und Biologie* **26**(3), 275-285.

FISCHER, M.; GRÄSEL, C.; SCHERBAUM, W.; GÄRTNER, R.; MANDL, H.; SCRIBA, P. (1995): Problemorientiertes Lernen in der Medizin mit dem computerunterstützten Autorensystem "CASUS". In: [ARBEITSKREIS MEDIZINERAUSBILDUNG 95], 350-354.

FLANAGAN, D.; KUHNERT, R. (1997): *JavaScript: das umfassende Referenzwerk.* Cambridge: O'Reilly.

FLANAGAN, D. (1998): *Java in a Nutshell*. Bonn: O'Reilly, Internat.Thomson-Verl.

FONTAINE, D.; LE BEUX, P.; RIOU, C.; JACQUELINET, C. (1994), An Intelligent Computer-Assisted Instruction System for Clinical Case Teaching. *Meth. Inf. Med.* **33**(4), 433-445.

FRASSON, C.; GAUTHIER G. (eds.) (1990): *Intelligent Tutoring Systems: At the Crossroad of Artificial Intelligence and Education*. Norwood, NJ: Ablex.

FRIEDMAN, R.B. (1996): Top Ten Reasons the World Wide Web May Fail to Change Medical Education. *Academic Medicine* **71**(9), 979-981.

GARDARIN, G.; VALDURIEZ, P. (1989): *Relational Databases and Knowledge Bases.* Bonn; Paris: Addison-Wesley.

GERNETH, F. (1992): *Konzeption und Realisierung eines dedizierten Immunologischen Datenhaltungs- und Auswertungssystems zur Integration klinischer und immunologischer Daten unter Verwendung eines semantischen Datenmodells.* Universität Heidelberg, Abt. Medizinische Informatik: Dissertation.

GÖBEL, E.; REMSTEDT, S. (Hrsg.) (1995): *Leitfaden zur Studienreform in Human- und Zahnmedizin: mit einem Überblick über Studienreformprojekte und Studienreformvorschläge*. Frankfurt am Main: Mabuse.

GOSLING, J.; YELLIN, F. ; JAVA TEAM (1997a): *Java™ API. Band 1: Die Basispakete*. Bonn; Paris: Addison-Wesley.

GOSLING, J.; YELLIN, F.; JAVA TEAM (1997b): *Java™ API. Band 2: Das Window Toolkit und Applets*. Bonn; Paris: Addison-Wesley.

HAAG, M. (1995): *Gesamtkonzept für die Entwicklung und den Einsatz von computerunterstützten Lehr-/Lernsystemen in der Medizinerausbildung an der Universität Heidelberg*. Heidelberg; Heilbronn: Diplomarbeit im Studiengang Medizinische Informatik.

HAAG, M.; LEVEN, F.J. (1997): WWW-basierte Lehr-/Lernsysteme für die medizinische Ausbildung: Stand und zukünftige Entwicklungen. In: [CONRADI, KREUTZ, SPITZER (Hrsg.) 97], 105-116.

HAAG, M.; MAYLEIN, L.; LEVEN, F.J.; TÖNSHOFF, B.; HAUX, R. (1998): Web-Based Training: A New Paradigm in Computer-Assisted Instruction in Medicine. *International Journal of Medical Informatics.* Zur Veröffentlichung angenommen.

HALASZ, F.G. (1988): Reflection on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM* **31**, 836-52.

HALLER, R.; BURGER, W.; SCHEFFNER D. (1995): Der Reformstudiengang Medizin am Klinikum Rudolf Virchow der Freien Universität Berlin. In: [ARBEITSKREIS MEDIZINERAUSBILDUNG 95], 288-296.

HASMAN, A. (1989): The Role of Medical Informatics in a Problem-oriented Educational Environment. In: [SALAMON, PROTTI et. al. (eds.) 89], 97-102.

HAYES, K.A.; LEHMANN, C.U. (1996): The Interactive Patient: A Multimedia Interactive Educational Tool on the World Wide Web. *M.D. Computing* **13**(4), 330-334.

HERRMANN, U.; LENZ, D.; UNBESCHEID, G. (1997): *Oracle 7.3: verwalten, optimieren, vernetzen*. Bonn; Paris: Addison-Wesley.

HERSH, W.R.; BROWN, K.E.; DONOHOE, L.C.; CAMPELL, E.M.; HORACEK, A.E. (1996): CliniWeb: Managing Clinical Information on the World Wide Web. *JAMIA* **3**(4), 273-280.

HEUER, A. (1992): *Objektorientierte Datenbanken: Konzepte, Modelle, Systeme*. Bonn; Paris: Addison-Wesley.

HIRSCH, M. C.; BRAUN, H.; HUBER M.; RIEDER, R.; VOIGT, K.H.; FISCHER, M.R. (1993): SimNerv - Ein virtuelles Physiologielabor zu Summenaktionspotentialen am Nervus ischiadicus des Frosches. *Arzt und Computer*, 341-344.

HOOPER, J.; O'CONNOR, J.; CHEESMAR, R.; PRICE, C.P. (1995): Tutorial Software for Clinical Chemistry Incorporating Interactive Multimedia Clinical Cases. *ClinChem.* **41**(9), 1345-1348.

IBRAHIM, B.; FRANKLIN, S.D. (1995): Advanced Educational Uses of the World Wide Web. *Computer Networks and ISDN Systems* **27**(6), 871-879.

JACOBSON, I.; CHRISTERSON, M.; JONSSON, P.; ÖVERGAARD, G. (1992): *Object-Oriented Software Engineering - A Use Case Driven Approach*. Wokingham: Addison-Wesley.

KALKBRENNER, G. (1996): *Computerunterstütztes Lernen und Teledienste.* Wiesbaden: Dt. Univ.-Verlag; Gabler.

KALLINOWSKI, F.; MEHRABI, A.; GLÜCKSTEIN, CH.; BENNER, A.; LINDINGER, M.; HASHEMI, B.; LEVEN, F.J.; HERFARTH, CH. (1997): Computer-basiertes Training - Ein neuer Weg der chirurgischen Aus- und Weiterbildung. *Chirurg* **68**(4), 433-438.

KAPPE, F.; MAURER H. (1993): Hyper-G: Ein großes universelles Hypermediasystem und einige Spin-offs. In: *Informationstechnik und Technische Informatik* **35**(2), 39-47.

KASS, R. (1989): Student Modeling in Intelligent Tutoring Systems - Implications for User Modeling. In: [KOBSA, WAHLSTER (eds.) 89], 386-410.

KERRES, M. (1998): *Multimediale und telemediale Lernumgebungen: Konzepte und Entwicklung*. München; Wien: Oldenbourg.

KLAR, R.; BAYER, U. (1990): Computer-Assisted Teaching and Learning in Medicine. *Int. J. Biomed. Comput.* **26**(1-2), 7-27.

KLUTE, R. (1997): Dateneintopf. Zusammenspiel: ODBC, JDBC und Treibersoftware. *iX* 6/97, 126-132.

KNAUP, P. (1994): *Rechnerunterstützte Erstellung medizinischer Lehrbücher unter Verwendung formal repräsentierten Wissens*. Universität Heidelberg, Abt. Medizinische Informatik: Dissertation.

KOEBKE, J.; NEUGEBAUER, E.; LEFERING, R. (Hrsg.) (1996): Die Qualität der Lehre in der Medizin. München; Wien; Baltimore: Urban und Schwarzenberg.

KROGSÆTER, M.; THOMAS, CH.G. (1994): Adaptivity: System-initiated Individualization. In: [OPPERMANN (ed.) 94], 67-96.

KUHLEN, R. (1991): *Hypertext: ein nichtlineares Medium zwischen Buch und Wissensbank*. Berlin; Heidelberg: Springer.

LARKIN, J.H.; CHABAY, R.W. (eds.) (1992): *Computer-Assisted Intruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches*. Hillsdale: Lawrence Erlbaum.

LAUSEN, G.; VOSSEN, G. (1996): Objekt-orientierte Datenbanken: Modelle und Sprachen. München; Wien: Oldenbourg.

LEUTNER, D (1992): *Adaptive Lehrsysteme; Instruktionspsychologische Grundlagen und experimentelle Analysen*. Weinheim: Psychologie-Verl.-Union.

LEVEN, F.J., SCHULZ, S., ALLE, W., KLAR, R. (1995). Rechnergestützte Lehr- und Lernsysteme in den Klinika: Stand und zukünftige Entwicklungen. In: BUCHHOLZ,W.; HAUX, R. (Hrsg.) (1995): *Informationsverarbeitung in den Universitätsklinika Baden-Württembergs*. Heidelberg. 187-193.

LEVEN, F.J.; ALLE, W.; HAAG, M.; VIELHAUER, A. (1996): Labor "Computerunterstützte Ausbildung in der Medizin" am Klinikum der Universität Heidelberg. In: [KOEBKE, NEUGEBAUER, LEFERING (Hrsg.) 96], 251-257.

LINDBERG, D.A.; HUMPHREYS, B.L.; MCCRAY, A.T. (1993): The Unified Medical Language System. *Meth. Inf. Med*. **32**(4), 281-291.

LLAURADO, J.G. (1997): Commentary. *International Journal of Medical Informatics* **46**, 1-5.

LOWE, H.J.; LOMAX, E.C.; POLONKEY, S.E. (1996): The World Wide Web: A Review of an Emerging Internet-based Technology for the Distribution of Biomedical Information. *JAMIA* **3**, 1-14.

LUSTI, M. (1992): *Intelligente tutorielle Systeme*. München; Wien: Oldenbourg.

MACROMEDIA (1994): *Director-Benutzerhandbuch*. San Francisco: Macromedia.

MAIER, G.; WILDERGER, A. (1995): *In 8 Sekunden um die Welt: Kommunikation über das Internet*. Bonn; Paris: Addison-Wesley.

MANN, G. (1993): *Integration wissensbasierter Systeme in der Medizin am Beispiel eines Daten- und Wissensbanksystems in der Neurologie*. Universität Heidelberg, Abt. Medizinische Informatik: Dissertation.

MAYLEIN, L.; HAAG, M. (1998): *Integration vorhandener Ressourcen in WBT-Systeme*. In Vorbereitung.

MCCALLA, G.I. (1992): The Central Importance of Student Modelling to Intelligent Tutoring. In: [COSTA (ed.) 92], 107-131.

MFT-PRÄSIDIALKOMMISION (1996): *Empfehlungen zur Neufassung der Approbationsordnung für Ärzte*.

MICROSOFT (1995): The Windows Interface Guidelines for Software Design. Redmond: Microsoft Press.

MIDDENDORF, S.; SINGER, R.; STROBEL, S. (1996): *Java: Programmierhandbuch und Referenz*. Heidelberg: dpunkt.

MINSKY, M. (1975): A Framework for Representing Knowledge. In: WINSTON, P. (ed.): *The Psychology of Computer Vision*. McGraw-Hill, 211-277.

MÜLLER, F.; SEIFERT O. (1989): *Taschenbuch der medizinisch-klinischen Diagnostik*. Berlin; Heidelberg: Springer.

NIELSEN, J. (1995): *Multimedia & Hypertext: The Internet and Beyond*. London: Academic Press.

NIEMANN, K.D. (1996): *Client/Server-Architektur*. Braunschweig; Wiesbaden: Vieweg.

OLIVER, R.E.; WINGERT, K.; REID, J.C.; LOCATIS, C. (1996): The Virtual Health Care Team™: an Example of Distributed Virtual Education. *AMIA Symposium Proceedings, JAMIA supplement*, 887.

OPPERMAN, R.; SIMM, H. (1994): Adaptability: User-Initiated Individualization. In: [OPPERMANN (ed.) 94], 14-66.

OPPERMANN, R. (ed.) (1994): *Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software*. Hillsdale: Lawrence Erlbaum.

OWEN, S.G.; HALL, R.; ANDERSON, J.; SMART, G.A. (1965): Programmed Learning in Medical Education. An Experimental Comparison of Programmed Instruction by Teaching Machine with Conventional Lecturing in the Teaching of Electrocardiography to Final Year Medical Students. *Postgrad. Med. J.* **41**(474), 201-205.

PAETAU, M. (1990): *Mensch-Maschine-Kommunikation: Software, Gestaltungspotentiale, Sozialverträglichkeit*. Frankfurt; New York: Campus.

PASTERKAMP, H. (1991): Computer Assisted Learning of Chest Auscultation: The Respiration Acoustics Laboratory Environment. In: VAN BEMMEL; ZVÁRONOVÁ (eds.): *Knowledge, Information and Medical Education*. North Holland: Elsevier, 244-251.

PELZER, G. (1998): *Konzept und Implementierung eines Systems zur rechnerunterstützten Generierung von hypermedialen Informationseinheiten auf HTML-Basis aus einer relationalen Datenbank*. Heidelberg; Heilbronn: Diplomarbeit im Studiengang Medizinische Informatik.

PUPPE, F. (1993): *Systematic Introduction to Expert Systems*. Berlin; Heidelberg: Springer.

PUPPE, F; REINHARDT, B (1995): Generating Case-Oriented Training From Diagnostic Expert Systems. *Machine-Mediated Learning* **3&4**, 199-219.

PUPPE, F.; GAPPA, U.; POECK, K.; BAMBERGER, S. (1996): *Wissensbasierte Diagnose- und Informationssysteme: mit Anwendungen des Expertensystem-Shell-Baukastens D3*. Berlin; Heidelberg: Springer.

QUADE, G.; PUSCHEL, N.; FAR, F. (1996): CancerNet Redistribution via WWW. *Proc-AMIA-Annu-Fall-Symp*, 403-407.

RAGGETT, D. (1997): *HTML 3.2 : neue Möglichkeiten für das Web-Publishing*. Bonn [u.a.]: Addison-Wesley.

RAMM, F. (1996): *Recherchieren und Publizieren im World Wide Web*. Braunschweig; Wiesbaden: Vieweg.

RAO, R.; PEDERSEN, J.O.; HEARST, M.A.; MACKINLAY, J.D.; CARD, S.K.; MASINTER, L.; HALVORSEN, P.K.; ROBERTSON, G.G. (1995): Rich Interaction in the Digital Library. *Communications of the ACM* **38**(4), 29-39.

RAUH, O; STICKEL, E. (1997): *Konzeptuelle Datenmodellierung*. Stuttgart; Leipzig: Teubner.

REDLICH, J.-P. (1996): *Corba 2.0: Praktische Einführung für C++ und Java.* Bonn; Reading; Mass. [u.a.]: Addison-Wesley.

REINHARDT, B.; SCHEWE, S. (1995): A Shell for Intelligent Tutoring Systems. *Proc. Conference on Artificial Intelligence in Education (AIED 95)*, 83-90.

REINHARDT, B.; PUPPE, F. (1997): Didaktische Aspekte in fallorientierten intelligenten Trainingsystemen. In: [CONRADI, KREUTZ et. al. (Hrsg.) 97], 157-168.

RICH, E. (1989): Stereotypes and User Modeling. In: [KOBSA, WAHLSTER (eds.) 89], 35-51.

RIEDEL, J. (1998): *Konzeption und Realisierung eines Web-Based Training Systems für infektiologische Fälle*. Heidelberg; Heilbronn: Diplomarbeit im Studiengang Medizinische Informatik.

RIEDEL, J.; HAAG, M.; LEVEN, F.J. (1998): WBT-Infekt: Ein Web-Based Training und Auskunftssystem für infektiologische Fälle. In: [ADLER, DIETRICH et al.98 (Hrsg.)], 123-129.

RODGERS, R.P.C. (1996): Java and Its Future in Biomedical Computing. *JAMIA* **3**(5), 303-308.

RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSON, W. (1991): *Object Oriented Modeling and Design*. Englewood Cliffs: Prentice Hall.

SAFRAN, C.; RIND, D.M.; SANDS, D.Z.; DAVIS, R.B.; WALD, J.; SLACK, W.V. (1996): Development of a Knowledge-Based Electronic Patient Record. *M.D. Computing* **13**(1), 46-54.

SALAMON, R.; PROTTI, D.; MOEHR, J. (eds.) (1989): *Proceedings of the 1989 International Symposium of Medical Informatics and Education*. School of Health Information Science, University of Victoria, British Columbia, Canada.

SAYEGH, M. (1997): *CORBA: Standard, Spezifikationen, Entwicklung*. Köln: O'Reilly.

SCHLAGETER, G.; STUCKY, W. (1983): *Datenbanksysteme: Konzepte und Modelle*. Stuttgart: Teubner.

SCHMITT, H.-J. (1996): Bewegliche Ziele. ActiveX: Microsofts Antwort auf Java. *c't* 12/96, 258-264.

SCHNEIDER-HUFSCHMIDT, M.; KÜHME, T.; MALINOWSKI, U. (eds.) (1993): *Adaptive User Interfaces: Principles and Practice*. North Holland: Elsevier.

SCHULMEISTER, R. (1996): G*rundlagen hypermedialer Lernsysteme: Theorie - Didaktik - Design*. Bonn; Paris: Addison-Wesley.

SCHULZ, S.; SCHRADER, U.; KLAR, R. (1997): Computer Based Training and Electronic Publishing in the Health Sector: Tools and Trends. *Meth. Inform. Med.* **36**(2), 149-153.

SCRIBA, P.C.; MANDL, H., SCHERBAUM, W. (Hrsg.) (1997): *CASUS Autoren-Handbuch*. München: CASUS-Eigenverlag.

SELF, J. A. (1990): Bypassing the Intractable Problem of Student Modeling. In: [FRASSON, GAUTHIER (eds.) 90], 107-123.

SILBERG, W.M.; LUNDBERG, G.D.; MUSACCHIO, R.A. (1997): Assessing, Controlling and Assuring the Quality of Medical Information on the Internet (Editorial). *JAMA*, **227**(15), 1244-1245.

SINGER, R. (1998): *Konzeption und Realisierung einer Autorenkomponente für Web-based Training-Systeme im Rahmen des CAMPUS-Projektes*. Heidelberg; Heilbronn: Diplomarbeit im Studiengang Medizinische Informatik.

SINGER, R.; HAAG, M.; LEVEN, F.J. (1998): CaSiMo - Ein Modellierungstool für fallbasierte WBT-Systeme. In: [ADLER, DIETRICH et al. (Hrsg.) 98], 141-150.

SPENSLEY, F.; ELSOM-COOK, M.; BYERLEY, P.; BROOKS, P.; FEDERICI, M.; SCARONI, C. (1990): Using Multiple Teaching Strategies in an ITS. In: [FRASSON, GAUTHIER (eds.) 90], 188-205.

SPITZER, K.; BÜRSNER, S. (1994): Wissensbasierte Systeme in der Medizin. In: *Informationstechnik und Technische Informatik* **36**(6), 53-59.

STOLL, C. (1996): *Die Wüste Internet. Geisterfahrten auf der Datenautobahn*. Frankfurt am Main: S. Fischer.

STONEBRAKER, M.; MOORE, D. (1998): *Objektrelationale Datenbanken*. München: Hanser.

STRASSER, A. (1992): *Generierung domänenspezifischer Wissensrepräsentationssysteme und Transformation von Wissensbasen mit einer Anwendung in Rechtsinformatik.* Sankt Augustin: Infix.

ULLMAN, J.D. (1988): *Principles of Database and Knowledge-Base Systems. Volume I: Classical Database Systems.* Rockville: Computer Science Press.

VAN BEMMEL, J.H.; MCCRAY A.T. (eds.) (1995): *Yearbook of Medical Informatics 1995: The Computer-based Patient Record*. Stuttgart: Schattauer.

WEED L.L. (1989): New Premises and New Tools for Medical Care and Medical Education. In: [SALAMON et. al. 89], 23-30.

WEICHELT, U.; SCHMIDT, H.; ADLER, M.; BAEHRING, T.; FISCHER, M.R. (1998): Fallorientierte medizinische Aus- und Weiterbildung im WWW: Komplexe Interaktionsmöglichkeiten durch eine Java-basierte Client-Server-Lösung. In: [ADLER, DIETRICH et al. (Hrsg.) 98], 159-164.

WEIDENMANN, B. (1991): *Lernen mit Bildmedien. Psychologische und didaktische Grundlagen*. Weinheim; Basel: Beltz.

WENGER, E. (1987): *Artificial Intelligence and Tutoring Systems*. Los Altos: Morgan Kaufmann.

WINKELS, R.; BREUKER, J. (1992): What's in an ITS? A Functional Decomposition. In: [COSTA (ed.) 92], 57-68.

WISSENSCHAFTSRAT (1992): *Leitlinien zur Reform der medizinischen Ausbildung*. Köln: Eigenverlag.

WISSENSCHAFTSRAT (1998): *Empfehlungen zur Hochschulentwicklung durch Multimedia in Studium und Lehre*. Mainz: Eigenverlag.

WOOLF, B. (1990): 20 Years in the Trenches: What Have We Learned? In: [FRASSON, GAUTHIER (eds.) 90], 234-250.

YETIM, F. (1994): *Erklärungen im Kontext der Mensch-Computer-Interaktion: Ein Konzept zur Integration der Methoden von Hypertext und künstlicher Intelligenz*. Universität Konstanz: Dissertation.

Elektronische Publikationen

Bei Verweisen auf elektronische Publikationen wurde auf die Angabe eines „Publikationsdatums“ verzichtet, da sich dieses häufig nicht feststellen läßt und jederzeit mit Änderungen an den Publikationen gerechnet werden muß.
Die in der Arbeit vorhandenen Referenzen beziehen sich auf elektronische Publikationen in der Fassung, in der sie am 18. September 1998 über Internet weltweit abrufbar waren.

CBT-Server: CBT-Server Medizin der Universität Heidelberg. http://www.hyg.uni-heidelberg.de.
DAETWYLER, CH.: The Interactive Exploration of the Fundus Diabeticus. http://www.aum.iawf.unibe.ch/vlz/bwl/eye_www.htm.
HON (Health On the Net) Code Principles. http://www.hon.ch/Conduct.html.
HSTAT (NLM). http://text.nlm.nih.gov.
LINKS CHECKERS. http://WWW.stars.com/Authoring/HTML/Validation/Links.html.
MACROMEDIA AUTHORWARE. http://www.macromedia.com/software/authorware/.
PITTSBURGH CASE INDEX. http://path.upmc.edu:80/cases/.
PUBMED-FREE MEDLINE: http://www.nlm.nih.gov/databases/freemedl.html.
SCHULZ, S.; AUHUBER, T.; SCHRADER, U.; KLAR, R.: Qualitätskriterien für Elektronische Publikationen in der Medizin. http://www.imbi.uni-freiburg.de/medinf/cbt_qk.htm.
STANFORD UNIVERSITY DIGITAL LIBRARIES PROJECT: http://walrus.stanford.edu/diglib/pub/.
SUN: *Remote Method Invocation Specification*. http://www.javasoft.com/products/jdk/1.1/docs/guide/rmi/spec/rmiTOC.doc.html.
SYMANTEC Visual Cafe for Java 2.5 - Database Development Edition. http://www.symantec.com/vcafedde/vcafedde25.html.
THE WHOLE BRAIN ATLAS. http://www.med.harvard.edu/AANLIB/cases/java/case.html.
UC BERKELEY DIGITAL LIBRARY PROJECT. http://elib.cs.berkeley.edu/.
UNIVERSTITY OF IOWA FAMILY PRACTICE HANDBOOK. http://vh.radiology.uiowa.edu/Providers/ClinRef/FPHandbook/FPContents.html.
UPMC (University of Pittsburgh School of Medicine) Health System 98. http://www.upmc.edu/ccehs/default.htm.
VENTER, H.: EBNF. http://www.cs.upe.ac.za/slim/ebnf.html.
VIROR. Verbundprojekt VIROR: Virtuelle Hochschule Oberrhein. http://ad.informatik.uni-freiburg.de/viror.
WEBDOCTOR MDFORUM. http://www.gretmar.com/webdoctor/MDForum.html.

Anhang

I Symbole und Notation

In der Arbeit werden die folgenden Schrifttypen (teilweise in Kombination) verwendet:

Konzept	**Notation**	**Beispiel**
Abbildungen	**fett**	**f**(x)
Datentypen	g e s p e r r t	d a t e
Mengen	unterstrichen	M
Objekttypen	GROSSBUCHSTABEN	FRAGE
Typen	KONTUR	ATT
Variablen	*kursiv*	*erstelldatum*

Die folgenden Operatoren werden verwendet:

Operator	**Bedeutung**
$\wedge$	logisches UND
$\vee$	logisches ODER
$\neg$	logisches NICHT
$\Rightarrow$	Implikation
$\Leftrightarrow$	Äquivalenz
$\exists$	Existenzquantor
$\forall$	Allquantor
$\in$	... ist Element von ...
$\subseteq$	Teilmenge
$\cup$	Vereinigungsmenge
$\underline{A} \rightarrow \underline{B}$	Abbildung von A nach B
ω	fehlender Wert
:=	Zuweisung
::=	besteht aus

II Logisches Modell

Das in Kapitel 2.9 vorgestellte 3-Ebenen-Modell zur Datenmodellierung sieht nach der Erstellung des konzeptuellen Modells (siehe Kapitel 3.5) die Erstellung des logischen Modells vor. Dieses berücksichtigt, daß ein relationales DBMS verwendet wird und führt aus Gründen eines schnelleren und effizienteren Zugriffes sowie der einfacheren Implementierbarkeit und Wartbarkeit die konzeptuellen Modelle zu einem einzigen logischen Modell zusammen.
Die bei der konzeptuellen Modellierung definierten Kardinalitätsangaben müssen selbstverständlich auch im logischen Modell erfüllt sein. Da die Einhaltung dieser Integritätsbedingungen (Ebene 2) vom Autorensubsystem überwacht wird und keine Spezifizierung im DBMS erfolgt, wird auf eine Spezifizierung der Kardinalitäten im logischen Modell verzichtet.
Nachfolgend wird das logische Modell vorgestellt. Für das einfachere Verständnis sind zu allen Attributen Kommentare angeben. Die formale Beschreibung erfolgt mit dem in Kapitel 2.10.1 vorgestellten Formalismus.

FACHGEBIET:

	REL({*Fachgebiet#, UebergeordFachgebiet#, Fachgebiet_Name*} | {**sib**$^1_{fg,1}$, **sib**$^1_{fg,2}$, **sib**$^2_{fg,1}$}), mit	
Fachgebiet#:	ATT(long |∅)	Primärschlüssel
UebergeordFachgebiet#:	ATT(long |∅)	Fremdschlüssel
Fachgebiet_Name:	ATT(char(50) |∅)	Name des Fachgebietes

sib$^1_{fg,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Fachgebiet#*, $\underline{x}$),
sib$^1_{fg,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Fachgebiet_Name*}, $\underline{x}$),
sib$^2_{fg,1}(v)$ = 'wahr' :⇔ **FS**((*FACHGEBIET*, *UebergeordFachgebiet#*), (*FACHGEBIET*, *Fachgebiet#*), *v*).

AUTOR:

	REL({*Autor#, Autor_Name, Kennung, Passwort*} | {**sib**$^1_{aut,1}$, **sib**$^1_{aut,2}$}), mit	
Autor#:	ATT(long |∅)	Primärschlüssel
Autor_Name:	ATT(char(50) |∅)	Kompletter Name des Autors
Kennung:	ATT(char(15) |∅)	Nutzerkennung des Autors
Passwort:	ATT(char(15) |∅)	Paßwort des Autors

sib$^1_{aut,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Autor#*, $\underline{x}$),
sib$^1_{aut,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Autor_Name, Kennung, Passwort*}, $\underline{x}$).

LAYOUT:

	REL({*Layout#, Kopfzeile, Fusszeile, Hintergrundfarbe*} | {**sib**$^1_{lay,1}$, **sib**$^1_{lay,2}$}), mit	
Layout#:	ATT(long |∅)	Primärschlüssel
Kopfzeile:	ATT(char(50) |∅)	Kopfzeile bei HTML-Seiten
Fusszeile:	ATT(char(50) |∅)	Fußzeile bei HTML-Seiten
Hintergrundfarbe:	ATT(char(15) |∅)	Hintergrundfarbe bei HTML-Seiten

sib$^1_{lay,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Layout#*, $\underline{x}$),
sib$^1_{lay,2}$($\underline{x}$) = 'wahr' :⇔ **nf1**({*Kopfzeile, Fusszeile, Hintergrundfarbe*}, $\underline{x}$).

WBT_SYSTEM:

	REL({*WBT_System#, Layout#, Autor#, WBT_System_Name, Erstelldatum*} | {**sib**$^1_{ws,1}$, **sib**$^1_{ws,2}$, **sib**$^2_{ws,1}$, **sib**$^2_{ws,2}$}), mit	
WBT_System#:	ATT(long |∅)	Primärschlüssel
Layout#:	ATT(long |∅)	Fremdschlüssel
Autor#:	ATT(long |∅)	Fremdschlüssel
WBT_System_Name:	ATT(char(80) |∅)	Name des WBT-Systems
Erstelldatum:	ATT(date |∅)	Erstelldatum des Systems

sib$^1_{ws,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*WBT_System#*, $\underline{x}$),
sib$^1_{ws,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Layout#, Autor#, WBT_System_Name, Erstelldatum*}, $\underline{x}$),
sib$^2_{ws,1}$(v) = 'wahr' :⇔ **FS**((*WBT_SYSTEM*, *Layout#*), (*LAYOUT, Layout#*) , v),
sib$^2_{ws,2}$(v) = 'wahr' :⇔ **FS**((*WBT_SYSTEM*, *Autor#*), (*AUTOR, Autor#*), v).

WBT_SYSTEM_FACHGEBIET:

	REL({WBT_System_*Fachgebiet#, WBT_System#, Fachgebiet#*} | {**sib**$^1_{wsfg,1}$, **sib**$^1_{wsfg,2}$, **sib**$^2_{wsfg,1}$, **sib**$^2_{wsfg,2}$}), mit	
WBT_System_Fachgebiet#:	ATT(long |∅)	Primärschlüssel
WBT_System#:	ATT(long |∅)	Fremdschlüssel
Fachgebiet#:	ATT(long |∅)	Fremdschlüssel

sib$^1_{wsfg,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*WBT_System_Fachgebiet#*, $\underline{x}$),
sib$^1_{wsfg,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*WBT_System#*, *Fachgebiet#*}, $\underline{x}$),
sib$^2_{wsfg,1}$(v) = 'wahr' :⇔ **FS**((*WBT_SYSTEM_FACHGEBIET*, *WBT_System#*), (*WBT_SYSTEM*, *WBT_System#*), $\underline{x}$),
sib$^2_{wsfg,2}$(v)= 'wahr' :⇔ **FS**((*WBT_SYSTEM_FACHGEBIET*, *Fachgebiet#*), (*FACHGEBIET*, *Fachgebiet#*), v).

FALL:

REL({*Patient#, Geschlecht, Vorname, Nachname, Lebensalter, Gewicht, Groesse, Bild, Beschreibung, Schwierigkeitsstufe, Einleitungstext, Zusammenfassung, Erstelldatum*} | {$\mathbf{sib}^1_{f,1}$, $\mathbf{sib}^1_{f,2}$}), mit

Patient#:	ATT(long |∅)	Primärschlüssel
Geschlecht:	ATT(char(9) |∅)	Geschlecht des Patienten
Vorname:	ATT(char(20) |∅)	Vorname des Patienten
Nachname:	ATT(char(30) |∅)	Nachname des Patienten
Lebensalter:	ATT(integer |∅)	Alter des Patienten
Gewicht:	ATT(integer |∅)	Gewicht des Patienten
Groesse:	ATT(integer |∅)	Größe des Patienten
Bild:	ATT(char(50) |∅)	Verweis auf Bild des Patienten
Beschreibung:	ATT(char(2000) |∅)	Beschreibung des Falles (benötigt für Fallauswahl durch Nutzer)
Schwierigkeitsstufe:	ATT(char(50)|∅)	Schwierigkeitsgrad des Falles
Einleitungstext:	ATT(char(2000) |∅)	Einleitungstext zum Fall
Zusammenfassung:	ATT(char(2000) |∅)	Zusammenfassung eines Falles (eingeblendet nach Beendigung der Fallbearbeitung)
Erstelldatum:	ATT(date |∅)	Erstelldatum des Falles

$\mathbf{sib}^1_{f,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Patient#*, $\underline{x}$),

$\mathbf{sib}^1_{f,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Geschlecht, Vorname, Nachname, Lebensalter, Gewicht, Groesse, Beschreibung, Einleitungstext, Schwierigkeitsstufe, Zusammenfassung, Erstelldatum*}, $\underline{x}$).

FALL_FACHGEBIET:

REL({*Fall_Fachgebiet#, Patient#, Fachgebiet#* } | {$\mathbf{sib}^1_{fafa,1}$, $\mathbf{sib}^1_{fafa,2}$, $\mathbf{sib}^2_{fafa,1}$, $\mathbf{sib}^2_{fafa,2}$}), mit

Fall_Fachgebiet#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
Fachgebiet#:	ATT(long |∅)	Fremdschlüssel

$\mathbf{sib}^1_{fafa,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Anamnese#*, $\underline{x}$),

$\mathbf{sib}^1_{fafa,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Patient#, Fachgebiet#*}, $\underline{x}$),

$\mathbf{sib}^2_{fafa,1}(v)$ = 'wahr' :⇔ **FS**((*FALL_FACHGEBIET*, *Patient#*), (*FALL*, Patient#), *v*),

$\mathbf{sib}^2_{fafa,2}(v)$ = 'wahr' :⇔ **FS**((*FALL_FACHGEBIET*, *Fachgebiet#*), (*FACHGEBIET*, *Fachgebiet#*), *v*).

WBT_SYSTEM_FALL:

	REL({*WBT_System_Fall#, WBT_System#, Patient#*} | {**sib**$^1_{wsf,1}$, **sib**$^1_{wsf,2}$, **sib**$^2_{wsf,1}$, **sib**$^2_{wsf,2}$}), mit	
WBT_System_Fall#:	ATT(long | ∅)	Primärschlüssel
WBT_System#:	ATT(long | ∅)	Fremdschlüssel
Patient#:	ATT(long | ∅)	Fremdschlüssel

sib$^1_{wsf,1}$($\underline{x}$) = ‘wahr’ :⇔ **PS**(*WBT_System_Fall#*, $\underline{x}$),
sib$^1_{wsf,2}$($\underline{x}$) = ‘wahr’ :⇔ **nf**({*WBT_System#, Patient#*}, $\underline{x}$),
sib$^2_{wsf,1}$(v) = ‘wahr’ :⇔ **FS**((*WBT_SYSTEM_FALL*, *WBT_System#*), (*WBT_SYSTEM*, *WBT_System#*), v),
sib$^2_{wsf,2}$(v) = ‘wahr’ :⇔ **FS**((*WBT_SYSTEM_FALL, Patient#*), (*FALL, Patient#*), v).

ANAMNESE:

	REL({*Anamnese#, Patient#, Datum, Zusammenfassung*} | {**sib**$^1_{a,1}$, **sib**$^1_{a,2}$, **sib**$^2_{a,1}$}), mit	
Anamnese#:	ATT(long | ∅)	Primärschlüssel
Patient#:	ATT(long | ∅)	Fremdschlüssel
Datum:	ATT(date | ∅)	Datum der Anamnesedurchführung
Zusammenfassung:	ATT(char(2000) | ∅)	Zusammenfassung der gesamten Anamnese

sib$^1_{a,1}$($\underline{x}$) = ‘wahr’ :⇔ **PS**(*Anamnese#*, $\underline{x}$),
sib$^1_{a,2}$($\underline{x}$) = ‘wahr’ :⇔ **nf**({*Patient#, Datum*}, $\underline{x}$),
sib$^2_{a,1}$(v) = ‘wahr’ :⇔ **FS**((*ANAMNESE, Patient#*), (*FALL, Patient#*), v).

KLINUNTERSUCHUNG:

	REL({*KlinUntersuchung#, Patient#, Datum, Zusammenfassung*} | {**sib**$^1_{ku,1}$, **sib**$^1_{ku,2}$, **sib**$^2_{ku,1}$}), mit	
KlinUntersuchung#:	ATT(long | ∅)	Primärschlüssel
Patient#:	ATT(long | ∅)	Fremdschlüssel
Datum:	ATT(date | ∅)	Datum der Anamnesedurchführung
Zusammenfassung:	ATT(char(2000) | ∅)	Zusammenfassung der gesamten Anamnese

sib$^1_{ku,1}$($\underline{x}$) = ‘wahr’ :⇔ **PS**(*KlinUntersuchung#*, $\underline{x}$),
sib$^1_{ku,2}$($\underline{x}$) = ‘wahr’ :⇔ **nf**({*Patient#, Datum*}, $\underline{x}$),
sib$^2_{ku,1}$(v) = ‘wahr’ :⇔ **FS**((*KLINUNTERSUCHUNG*, *Patient#*), (*FALL, Patient#*), v).

TECHNUNTERSUCHUNG:

REL({*TechnUntersuchung#, Patient#, TechnFormular#, TechnUntersuchungsverfahren#, Koerperregion#, Befund_Text, Befund_Bild_Video, Anforderung_Datum, Befund_Datum, Normal, Notwendig, Kommentar, Leitsymptom*} | {$\mathbf{sib}^1_{tu,1}$, $\mathbf{sib}^1_{tu,2}$, $\mathbf{sib}^1_{tu,3}$, $\mathbf{sib}^2_{tu,1}$, $\mathbf{sib}^2_{tu,2}$, $\mathbf{sib}^2_{tu,3}$, $\mathbf{sib}^2_{tu,4}$}), mit

TechnUntersuchung#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
TechnFormular#:	ATT(long |∅)	Fremdschlüssel
TechnUntersuchungsverfahren#:	ATT(long |∅)	Fremdschlüssel
Koerperregion#:	ATT(long |∅)	Fremdschlüssel
Befund_Text:	ATT(char(2000) |∅)	Text des Befundes
Befund_Bild_Video:	ATT(char(50) |∅)	Pfad zu Bild oder Video
Anforderung_Datum:	ATT(date |∅)	Anforderungsdatum der techn. Untersuchung
Befund_Datum:	ATT(date |∅)	Datum, an dem das Ergebnis geliefert wird
Normal:	ATT(boolean |∅)	Untersuchungbefund normal?
Notwendig:	ATT(boolean |∅)	Untersuchungsanforderung notwendig?
Kommentar:	ATT(char(2000) |∅)	Kommentar zum Untersuchungsbefund
Leitsymptom:	ATT(boolean |∅)	Ist Untersuchungsbefund Leitsymptom?

$\mathbf{sib}^1_{tu,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*TechnUntersuchung#*, $\underline{x}$),

$\mathbf{sib}^1_{tu,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Patient#, TechnFormular#, TechnUntersuchungsverfahren#, Koerperregion#, Befund_Text, Anforderung_Datum, Befund_Datum, Normal, Notwendig, Leitsymptom*}, $\underline{x}$),

$\mathbf{sib}^1_{tu,3}(\underline{x})$ = 'wahr' :⇔ $\forall\, \boldsymbol{a} \in \underline{x}: \boldsymbol{a}|_{\{\text{Anforderung\_Datum}\}} \leq \boldsymbol{a}|_{\{\text{Befund\_Datum}\}}$,

$\mathbf{sib}^2_{tu,1}(v)$ = 'wahr' :⇔ **FS**((*TECHNUNTERSUCHUNG, Patient#*), (*FALL, Patient#*) , *v*),

$\mathbf{sib}^2_{tu,2}(v)$ = 'wahr' :⇔ **FS**((*TECHNUNTERSUCHUNG, TechnFormular#*), (*TECHNFORMULAR,TechnFormular#*), *v*),

$\mathbf{sib}^2_{tu,3}(v)$ = 'wahr' :⇔ **FS**((*TECHNUNTERSUCHUNG, TechnUntersuchungsverfahren#*), (*TECHNUNTERSUCHUNGSVERFAHREN, TechnUntersuchungsverfahren#*), *v*),

$\mathbf{sib}^2_{tu,4}(v)$ = 'wahr' :⇔ **FS**((*TECHNUNTERSUCHUNG, Koerperregion#*), (*KOERPERREGION, Koerperregion#*), *v*).

LABORUNTERSUCHUNG:

REL({*Laboruntersuchung#, Patient#, Probenart#, Empfaengerlabor, Anforderung_Datum, Befund_Datum*} | {$\mathbf{sib}^1_{lu,1}$, $\mathbf{sib}^1_{lu,2}$, $\mathbf{sib}^1_{lu,3}$, $\mathbf{sib}^2_{lu,1}$, $\mathbf{sib}^2_{lu,2}$}), mit

Laboruntersuchung#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
Probenart#:	ATT(long(50) |∅)	Fremdschlüssel
Empfaengerlabor:	ATT(char(50) |∅)	Name des Empfängerlabors
Anforderung_Datum:	ATT(date |∅)	Anforderungsdatum der Laboruntersuchung
Befund_Datum:	ATT(date |∅)	Datum, an dem das Ergebnis geliefert wird
Zusammenfassung:	ATT(char(2000) |∅)	Zusammenfassung der Laboruntersuchung

$\mathbf{sib}^1_{lu,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Laboruntersuchung#*, $\underline{x}$),
$\mathbf{sib}^1_{lu,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Patient#, Probenart#, Anforderung_Datum, Befund_Datum*}, $\underline{x}$),
$\mathbf{sib}^1_{lu,3}(\underline{x})$ = 'wahr' :⇔ $\forall \boldsymbol{a} \in \underline{x}: \boldsymbol{a}|_{\{\text{Anforderung\_Datum}\}} \leq \boldsymbol{a}|_{\{\text{Befund\_Datum}\}}$,
$\mathbf{sib}^2_{lu,1}(v)$ = 'wahr' :⇔ **FS**((*LABORUNTERSUCHUNG, Patient#*), (*FALL, Patient#*), *v*),
$\mathbf{sib}^2_{lu,2}(v)$ = 'wahr' :⇔ **FS**((*LABORUNTERSUCHUNG, Probenart#*), (*PROBENART, Probenart#*), *v*).

LABORTESTERGEBNIS:

REL({*LaborTestergebnis#, Laboruntersuchung#, Labortest#, Ergebnis_Text, Ergebnis_Bild_Video, Wert, Normal, Notwendig, Kommentar, Leitsymptom*} | {$\mathbf{sib}^1_{lte,1}$, $\mathbf{sib}^1_{lte,2}$, $\mathbf{sib}^1_{lte,3}$, $\mathbf{sib}^2_{lte,1}$, $\mathbf{sib}^2_{lte,2}$}), mit

LaborTestergebnis#:	ATT(long |∅)	Primärschlüssel
Laboruntersuchung#:	ATT(long |∅)	Fremdschlüssel
Labortest#:	ATT(long |∅)	Fremdschlüssel
Ergebnis_Text:	ATT(char(2000) |∅)	Befund_Text
Ergebnis_Bild_Video:	ATT(char(50) |∅)	Pfad zu Bild oder Video
Wert:	ATT(real |∅)	Befundergebnis
Normal:	ATT(boolean |∅)	Laborbefund normal?
Notwendig:	ATT(boolean |∅)	Testanforderung notwendig?
Kommentar:	ATT(char(2000) |∅)	Kommentar zum Testergebnis
Leitsymptom:	ATT(boolean |∅)	Ist Testergebnis Leitsymptom?

$\mathbf{sib}^1_{lte,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*LaborTestergebnis#*, $\underline{x}$),
$\mathbf{sib}^1_{lte,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Laboruntersuchung#, Labortest#, Normal, Notwendig, Leitsymptom*}, $\underline{x}$),
$\mathbf{sib}^1_{lte,3}(\underline{x})$ = 'wahr' :⇔ **nf1**({*Ergebnis_Text, Wert*}, $\underline{x}$),
$\mathbf{sib}^2_{lte,1}(v)$ = 'wahr' :⇔ **FS**((*LABORTESTERGEBNIS*, *Laboruntersuchung#*), (*LABORUNTERSUCHUNG*, *Laboruntersuchung#*), *v*),

$\mathbf{sib}^2_{lte,2}(v)$ = 'wahr' :⇔ **FS**((*LABORTESTERGEBNIS, Labortest#*), (*LABORTEST, Labortest#*), *v*).

KLINUNTERSUCHUNGERGEBNIS:

REL({*KlinUntersuchungErgebnis#, KlinUntersuchung#, KlinUntersuchungsart#, Koerperregion#, Ergebnis_Text, Ergebnis_Bild_Video, Normal, Eingangssymptom, Notwendig, Kommentar, Leitsymptom* } | {$\mathbf{sib}^1_{kue,1}$, $\mathbf{sib}^1_{kue,2}$, $\mathbf{sib}^2_{kue,1}$, $\mathbf{sib}^2_{kue,2}$, $\mathbf{sib}^2_{kue,3}$}), mit

KlinUntersuchung-Ergebnis#:	ATT(long |∅)	Primärschlüssel
KlinUntersuchung#:	ATT(long |∅)	Fremdschlüssel
KlinUntersuchungsart#:	ATT(long |∅)	Fremdschlüssel
Koerperregion#:	ATT(long |∅)	Fremdschlüssel
Ergebnis_Text:	ATT(char(2000) |∅)	Befund_Text
Ergebnis_Bild_Video:	ATT(char(50) |∅)	Pfad zu evtl. mitgeliefertem Bild oder Video
Normal:	ATT(boolean |∅)	Befund normal?
Eingangssymptom:	ATT(boolean |∅)	Wurde Symptom vom Patienten ohne Untersuchung geschildert?
Notwendig:	ATT(boolean |∅)	War Untersuchung notwendig?
Kommentar:	ATT(char(2000) |∅)	Kommentar zum Untersuchungsbefund
Leitsymptom:	ATT(boolean |∅)	Ist Untersuchung Leitsymptom?

$\mathbf{sib}^1_{kue,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*KlinUntersuchungErgebnis#*, $\underline{x}$),

$\mathbf{sib}^1_{kue,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*KlinUntersuchung#, KlinUntersuchungsart#, Koerperregion#, Ergebnis_Text, Normal, Eingangssymptom, Notwendig, Leitsymptom*}, $\underline{x}$),

$\mathbf{sib}^2_{kue,1}(v)$ = 'wahr' :⇔ **FS**((*KLINUNTERSUCHUNGERGEBNIS, KlinUntersuchung#*), (*KLINUNTERSUCHUNG*, *KlinUntersuchung#*), *v*),

$\mathbf{sib}^2_{kue,2}(v)$ = 'wahr' :⇔ **FS**((*KLINUNTERSUCHUNGERGEBNIS, KlinUntersuchungsart#*), (*KLINUNTERSUCHUNGSART, KlinUntersuchungsart#*), *v*),

$\mathbf{sib}^2_{kue,3}(v)$ = 'wahr' :⇔ **FS**((*KLINUNTERSUCHUNGSERGEBNIS, Koerperregion#*), (*KOERPERREGION, Koerperregion#*), *v*).

ANAMNESEANTWORT:

REL({*Anamneseantwort#, Anamnese#, Anamnesefrage#, Patientenantwort, Normal, Eingangssymptom, Notwendig, Kommentar, Leitsymptom*} | {$\mathbf{sib}^1_{aa,1}$, $\mathbf{sib}^1_{aa,2}$, $\mathbf{sib}^2_{aa,1}$, $\mathbf{sib}^2_{aa,2}$}), mit

Anamneseantwort#:	ATT(long |∅)	Primärschlüssel
Anamnese#:	ATT(long |∅)	Fremdschlüssel
Anamnesefrage#:	ATT(long |∅)	Fremdschlüssel
Patientenantwort:	ATT(char(2000) |∅)	Antwort des Patienten
Normal:	ATT(boolean |∅)	Patientenantwort unauffällig?

Eingangssymptom:	ATT(boolean |∅)	Wurde Symptom vom Patienten ohne Untersuchung geschildert?
Notwendig:	ATT(boolean |∅)	Anamnesefrage notwendig?
Kommentar:	ATT(char(2000) |∅)	Kommentar zur Patientenantwort
Leitsymptom:	ATT(boolean |∅)	Patientenantwort Leitsymptom?

sib$^1_{aa,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Anamneseantwort#*, $\underline{x}$),
sib$^1_{aa,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Anamnese#, Anamnesefrage#, Patientenantwort, Normal, Eingangssymptom, Notwendig, Leitsymptom*}, $\underline{x}$),
sib$^2_{aa,1}(v)$ = 'wahr' :⇔ **FS**((*ANAMNESEANTWORT*, *Anamnese#*), (*ANAMNESE, Anamnese#)*, *v*),
sib$^2_{aa,2}(v)$ = 'wahr' :⇔ **FS**((*ANAMNESEANTWORT*, *Anamnesefrage#*), (*ANAMNESEFRAGE*, *Anamnesefrage#*), *v*).

ANAMNESEFRAGE:

REL({*Anamnesefrage#, Anamnesetyp#, Fragetext*} | {**sib**$^1_{af,1}$, **sib**$^1_{af,2}$, **sib**$^2_{af,1}$}), mit

Anamnesefrage#:	ATT(long |∅)	Primärschlüssel
Anamnesetyp#:	ATT(long |∅)	Fremdschlüssel
Fragetext:	ATT(char(2000) |∅)	Text der Frage

sib$^1_{af,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Anamnesefrage#*, $\underline{x}$),
sib$^1_{af,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Anamnesetyp#, Fragetext*}, $\underline{x}$),
sib$^2_{af,1}(v)$ = 'wahr' :⇔ **FS**((*ANAMNESEFRAGE*, *Anamnesetyp#*), (*ANAMNESETYP*, *Anamnesetyp#*), *v*).

ANAMNESEFRAGE_FACHGEBIET:

REL({*Anamnesefrage_Fachgebiet#, Anamnesefrage#, Fachgebiet#*} | {**sib**$^1_{aff,1}$, **sib**$^1_{aff,2}$, **sib**$^2_{aff,1}$, **sib**$^2_{aff,2}$}), mit

Anamnesefrage_Fachgebiet#:	ATT(long |∅)	Primärschlüssel
Anamnesefrage#:	ATT(long |∅)	Fremdschlüssel
Fachgebiet#:	ATT(long |∅)	Fremdschlüssel

sib$^1_{aff,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Laboruntersuchung#*, $\underline{x}$),
sib$^1_{aff,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Anamnesefrage#, Fachgebiet#*}, $\underline{x}$),
sib$^2_{aff,1}(v)$ = 'wahr' :⇔ **FS**((*ANAMNESEFRAGE_FACHGEBIET, Anamnesefrage#*), (*ANAMNESEFRAGE*, *Anamnesefrage#*), *v*),
sib$^2_{aff,2}(v)$ = 'wahr' :⇔ **FS**((*ANAMNESEFRAGE_FACHGEBIET*, *Fachgebiet#*), (*FACHGEBIET*, *Fachgebiet#*), *v*).

ANAMNESETYP:

	REL({*Anamnesetyp#, Anamnesetyp_Name*} | {**sib**$^1_{at,1}$, **sib**$^1_{at,2}$}), mit	
Anamnesetyp#:	ATT(long |∅)	Primärschlüssel
Anamnesetyp_Name:	ATT(char(30) |∅)	Name des Anamnesetyps

sib$^1_{at,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Laboruntersuchung#*, $\underline{x}$),
sib$^1_{at,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Anamnesetyp_Name*}, $\underline{x}$).

KLINUNTERSUCHUNGSART:

	REL({*KlinUntersuchungsart#, Klinuntersuchungsart_Name*} | {**sib**$^1_{kua,1}$, **sib**$^1_{kua,2}$}), mit	
KlinUntersuchungsart#:	ATT(long |∅)	Primärschlüssel
KlinUntersuchungsart_Name:	ATT(char(30) |∅)	Name der klinischen Untersuchungsart

sib$^1_{kua,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*KlinUntersuchungsart#*, $\underline{x}$),
sib$^1_{kua,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*KlinUntersuchungsart_Name*}, $\underline{x}$).

TECHNUNTERSUCHUNGSVERFAHREN:

	REL({*TechnUntersuchungsverfahren#, TechnUntersuchungsverfahren _Name, Staerken, Schwaechen*} | {**sib**$^1_{tuv,1}$, **sib**$^1_{tuv,2}$}), mit	
TechnUntersuchungsverfahren#:	ATT(long |∅)	Primärschlüssel
Tuntersuchungsverfahren_Name:	ATT(char(30) |∅)	Name des techn. Untersuchungsverfahrens
Staerken:	ATT(char(2000) |∅)	Stärken eines techn. Untersuchungsverfahrens
Schwaechen:	ATT(char(2000) |∅)	Schwächen eines techn. Untersuchungsverfahrens

sib$^1_{tuv,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*TechnUntersuchungsverfahren#*, $\underline{x}$),
sib$^1_{tuv,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*TUntersuchungsverfahren_Name*}, $\underline{x}$).

LABORTEST:

	REL({*Labortest#, Labortest_Name, Labortest_Abkuerzung, Einheit*} | {**sib**$^1_{lt,1}$, **sib**$^1_{lt,2}$}), mit	
Labortest#:	ATT(long |∅)	Primärschlüssel
Labortest_Name:	ATT(char(50) |∅)	Name des Labortests
Labortest_Abkuerzung:	ATT(char(20) |∅)	Abkürzung für Labortest
Einheit:	ATT(char(10) |∅)	Einheit des Testergebnisses

sib$^1_{lt,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Labortest#*, $\underline{x}$),
sib$^1_{lt,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Labortest_Name, Einheit*}, $\underline{x}$).

LABORFORMULAR:

REL({*Laborformular#, Erstelldatum, Formularkopf, Hintergrundfarbe, Formularueberschrift, Spaltenzahl*} | {**sib**$^1_{lf,1}$, **sib**$^1_{lf,2}$}), mit

Laborformular#:	ATT(long |∅)	Primärschlüssel
Erstelldatum:	ATT(date |∅)	Erstelldatum des Laborformulars
Formularkopf:	ATT(char(50) |∅)	Formularkopf
Hintergrundfarbe:	ATT(char(15) |∅)	Hintergrundfarbe des Formulars
Formularueberschrift:	ATT(char(80) |∅)	Überschrift des Formulars
Spaltenzahl:	ATT(integer |∅)	Spaltenzahl des Formulars

sib$^1_{lf,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Laborformular#*, $\underline{x}$),
sib$^1_{lf,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Erstelldatum, Formularueberschrift, Spaltenzahl*}, $\underline{x}$).

TECHNFORMULAR:

REL({*TechnFormular#, Erstelldatum, Formularkopf, Hintergrundfarbe, Formularueberschrift* } | {**sib**$^1_{tf,1}$, **sib**$^1_{tf,2}$}), mit

TechnFormular#:	ATT(long |∅)	Primärschlüssel
Erstelldatum:	ATT(date |∅)	Erstelldatum des techn. Formulars
Formularkopf:	ATT(char(50) |∅)	Formularkopf
Hintergrundfarbe:	ATT(char(15) |∅)	Hintergrundfarbe des Formulars
Formularueberschrift:	ATT(char(80) |∅)	Überschrift des Formulars

sib$^1_{tf,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*TechnFormular#*, $\underline{x}$),
sib$^1_{tf,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Erstelldatum, Formularueberschrift*}, $\underline{x}$).

LABORF_TEILFORM:

REL({*Laborf_Teilform#, Laborformular#, Teilformular#, Spaltenposition, Reihenfolgeposition*} | {**sib**$^1_{lftf,1}$, **sib**$^1_{lftf,2}$, **sib**$^2_{lftf,1}$, **sib**$^2_{lftf,2}$}), mit

Laborf_Teilform#:	ATT(long |∅)	Primärschlüssel
Laborformular#:	ATT(long |∅)	Fremdschlüssel
Teilformular#:	ATT(long |∅)	Fremdschlüssel
Spaltenposition:	ATT(integer |∅)	Nummer der Spalte
Reihenfolgeposition:	ATT(integer |∅)	Reihenfolgenummer in Spalte

sib$^1_{lftf,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Laborf_Teilform#*, $\underline{x}$),
sib$^1_{lftf,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Laborformular#, Teilformular#, Spaltenposition, Reihenfolgeposition* }, $\underline{x}$),
sib$^2_{lftf,1}$(v) = 'wahr' :⇔ **FS**((*LABORF_TEILFORM*, *Laborformular#*), (*LABORFORMULAR*, *Laborformular#*), v),
sib$^2_{lftf,2}$(v) = 'wahr' :⇔ **FS**((*LABORF_TEILFORM*, *Teilformular#*), (*TEILFORMULAR*, *Teilformular#*), v).

TEILFORMULAR:

REL({*Teilformular#, Ueberschrift*} | {**sib**$^1_{tef,1}$, **sib**$^1_{tef,2}$}), mit

Teilformular#:	ATT(long | ∅)	Primärschlüssel
Ueberschrift:	ATT(char(50) | ∅)	Überschrift des Teilformulars

sib$^1_{tef,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Teilformular#*, $\underline{x}$),
sib$^1_{tef,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Ueberschrift*}, $\underline{x}$).

TEILFORMULAR_LABORTEST:

REL({*Teilformular_Labortest#, Teilformular#, Labortest#*} | {**sib**$^1_{tflt,1}$, **sib**$^1_{tflt,2}$, **sib**$^2_{tflt,1}$, **sib**$^2_{tflt,2}$}), mit

Teilformular_Labortest#:	ATT(long | ∅)	Primärschlüssel
Teilformular#:	ATT(long | ∅)	Fremdschlüssel
Labortest#:	ATT(long | ∅)	Fremdschlüssel

sib$^1_{tflt,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Teilformular_Labortest#*, $\underline{x}$),
sib$^1_{tflt,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Teilformular#, Labortest#*}, $\underline{x}$),
sib$^2_{tflt,1}$(v) = 'wahr' :⇔ **FS**((*TEILFORMULAR_LABORTEST, Teilformular#*), (*TEILFORMULAR, Teilformular#*), v),
sib$^2_{tflt,2}$(v) = 'wahr' :⇔ **FS**((*TEILFORMULAR_LABORTEST, Labortest#*), (*LABORTEST, Labortest#*), v).

PROBENART:

REL({*Probenart#, Probenart_Name, Entnahmestelle*} | {**sib**$^1_{pa,1}$, **sib**$^1_{pa,2}$}), mit

Probenart#:	ATT(long | ∅)	Primärschlüssel
Probenart_Name:	ATT(char(50) | ∅)	Name der Probenart
Entnahmestelle:	ATT(char(50) | ∅)	Entnahmestelle der Probe

sib$^1_{pa,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Probenart#*, $\underline{x}$),
sib$^1_{pa,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Probenart_Name*}, $\underline{x}$).

KOERPERREGION:

REL({*Koerperregion#, Koerperregion_Name, Beschreibung, Bild, Inspektion, Palpation, Auskultation, Perkussion, Untersuchung*} | {**sib**$^1_{kr,1}$, **sib**$^1_{kr,2}$}), mit

Koerperregion#:	ATT(long | ∅)	Primärschlüssel
Koerperregion_Name:	ATT(char(50) | ∅)	Name der Probenart
Beschreibung:	ATT(char(50) | ∅)	Beschreibung
Bild:	ATT(char(50) | ∅)	Verweis auf Bild der Körperregion
Inspektion:	ATT(boolean | ∅)	Kann Körperregion inspiziert werden?
Palpation:	ATT(boolean | ∅)	Kann Körperregion palpiert werden?

Auskultation:	ATT(boolean |∅)	Kann Körperregion auskultiert werden?
Perkussion:	ATT(boolean |∅)	Kann Körperregion perkutiert werden?
TUntersuchung:	ATT(boolean |∅)	Kann an Körperregion eine techn. Untersuchung durchgeführt werden?

$\mathbf{sib}^1_{kr,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Koerperregion#*, $\underline{x}$),
$\mathbf{sib}^1_{kr,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Koerperregion_Name, Inspektion, Palpation, Auskultation, Perkussion, TUntersuchung*}, $\underline{x}$).

ALTERSSTUFE:

REL({*Altersstufe#, Altersstufe _Name, Alterstufe_Abkuerzung*} | {$\mathbf{sib}^1_{as,1}$, $\mathbf{sib}^1_{as,2}$, $\mathbf{sib}^1_{as,3}$}), mit

Altersstufe#:	ATT(long |∅)	Primärschlüssel
Altersstufe_Name:	ATT(char(20) |∅)	Name der Altersstufe
Altersstufe_Abkuerzung:	ATT(char(5) |∅)	Abkürzung
Von:	ATT(long |∅)	Von (Angabe in Tagen)
Bis:	ATT(long |∅)	Bis (Angabe in Tagen)

$\mathbf{sib}^1_{as,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Altersstufe#*, $\underline{x}$),
$\mathbf{sib}^1_{as,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Altersstufe_Name, Von, Bis*}, $\underline{x}$),
$\mathbf{sib}^1_{as,3}(\underline{x})$ = 'wahr' :⇔ $\forall \boldsymbol{a} \in \underline{x}: \boldsymbol{a}|_{\{Von\}} < \boldsymbol{a}|_{\{Bis\}}$.

LABORNORMALBEFUND:

REL({*LaborNormalbefund#, Labortest#, Probenart#, Geschlecht, Befund_Text, Min, Max*} | {$\mathbf{sib}^1_{lnb,1}$, $\mathbf{sib}^1_{lnb,2}$, $\mathbf{sib}^1_{lnb,3}$, $\mathbf{sib}^1_{lnb,4}$, $\mathbf{sib}^2_{lnb,1}$, $\mathbf{sib}^2_{lnb,2}$}), mit

LaborNormalbefund#:	ATT(long |∅)	Primärschlüssel
Labortest#:	ATT(long |∅)	Fremdschlüssel
Probenart#:	ATT(long |∅)	Fremdschlüssel
Geschlecht:	ATT(char(9) |∅)	Geschlecht
Befund_Text:	ATT(char(2000) |∅)	Befundtext, z.B. bei mikrobiologischen Untersuchungen
Min:	ATT(real |∅)	Untere Grenze des Normbereichs
Max:	ATT(real |∅)	Obere Grenze des Normbereichs

$\mathbf{sib}^1_{lnb,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Labornormalbefund#*, $\underline{x}$),
$\mathbf{sib}^1_{lnb,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Labortest#, Probenart#, Geschlecht*}, $\underline{x}$),
$\mathbf{sib}^1_{lnb,3}(\underline{x})$ = 'wahr' :⇔ **nf1**({*Befund_Text, Min, Max*}, $\underline{x}$),
$\mathbf{sib}^1_{lnb,4}(\underline{x})$ = 'wahr' :⇔ $\forall \boldsymbol{a} \in \underline{x}: \boldsymbol{a}|_{\{Min\}} < \boldsymbol{a}|_{\{Max\}}$,
$\mathbf{sib}^2_{lnb,1}(v)$ = 'wahr' :⇔ **FS**((*LABORNORMALBEFUND, Labortest#*),(*LABORTEST*, *Labortest#*), *v*),

$\mathbf{sib}^2_{lnb,2}(v)$ = 'wahr' :⇔ **FS**((*LABORNORMALBEFUND, Probenart#*), (*PROBENART*, *Probenart#*), *v*),

TECHNNORMALBEFUND:

REL({*TechnNormalbefund#, TechnUntersuchungsverfahren#, Koerperregion#, Altersstufe#, Geschlecht, Befund_Text, Befund_Bild_Video*} | {$\mathbf{sib}^1_{tnb,1}$, $\mathbf{sib}^1_{tnb,2}$, $\mathbf{sib}^2_{tnb,1}$, $\mathbf{sib}^2_{tnb,2}$, $\mathbf{sib}^2_{tnb,3}$}), mit

TechnNormalbefund#:	ATT(long |∅)	Primärschlüssel
TechnUntersuchungsverfahren#:	ATT(long |∅)	Fremdschlüssel
Koerperregion#:	ATT(long |∅)	Fremdschlüssel
Altersstufe#:	ATT(long |∅)	Fremdschlüssel
Geschlecht:	ATT(char(9) |∅)	Geschlecht
Befund_Text:	ATT(char(2000) |∅)	Befundtext
Befund_Bild_Video:	ATT(char(50) |∅)	Pfad zu Bild oder Video

$\mathbf{sib}^1_{tnb,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*TechnNormalbefund#*, $\underline{x}$),
$\mathbf{sib}^1_{tnb,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*TechnUntersuchungsverfahren#, Koerperregion#, Altersstufe#, Geschlecht, Befund_Text*}, $\underline{x}$),
$\mathbf{sib}^2_{tnb,1}(v)$ = 'wahr' :⇔ **FS**((*TECHNNORMALBEFUND*, *TechnUntersuchungsverfahren#*), (*TECHNUNTERSUCHUNGSVERFAHREN*, *TechnUntersuchungsverfahren#*), *v*),
$\mathbf{sib}^2_{tnb,2}(v)$ = 'wahr' :⇔ **FS**((*TECHNNORMALBEFUND*, *Koerperregion#*), (*KOERPERREGION*, *Koerperregion#*), *v*),
$\mathbf{sib}^2_{tnb,3}(v)$ = 'wahr' :⇔ **FS**((*TECHNNORMALBEFUND*, *Altersstufe#*), (*ALTERSSTUFE*, *Altersstufe#*), *v*).

KLINNORMALBEFUND:

REL({*KlinNormalbefund#, KlinUntersuchungsart#, Koerperregion#, Altersstufe#, Geschlecht, Ergebnis_Text, Ergebnis_Bild_Video*} | {$\mathbf{sib}^1_{knb,1}$, $\mathbf{sib}^1_{knb,2}$, $\mathbf{sib}^2_{knb,1}$, $\mathbf{sib}^2_{knb,2}$, $\mathbf{sib}^2_{knb,3}$}), mit

KlinNormalbefund#:	ATT(long |∅)	Primärschlüssel
KlinUntersuchungsart#:	ATT(long |∅)	Fremdschlüssel
Koerperregion#:	ATT(long |∅)	Fremdschlüssel
Altersstufe#:	ATT(long |∅)	Fremdschlüssel
Geschlecht:	ATT(char(9) |∅)	Geschlecht
Ergebnis_Text:	ATT(char(2000) |∅)	Befundtext
Ergebnis_Bild_Video:	ATT(char(50) |∅)	Pfad zu Bild oder Video

$\mathbf{sib}^1_{knb,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*KlinNormalbefund#*, $\underline{x}$),
$\mathbf{sib}^1_{knb,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*KlinUntersuchungsart#, Koerperregion#, Altersstufe#, Geschlecht, Ergebnis_Text*}, $\underline{x}$),
$\mathbf{sib}^2_{knb,1}(v)$ = 'wahr' :⇔ **FS**((*KLINNORMALBEFUND*, *KlinUntersuchungsart#*), (*KLINUNTERSUCHUNGSART*, *KlinUntersuchungsart#*), *v*),

$\mathbf{sib}^2_{knb,2}(v)$ = 'wahr' :⇔ **FS**((*KLINNORMALBEFUND*, *Koerperregion#*) ,*KOERPERREGION*, *Koerperregion#*), *v*),
$\mathbf{sib}^2_{knb,3}(v)$ = 'wahr' :⇔ **FS**((*KLINNORMALBEFUND*, *Altersstufe#*), (ALTERSSTUFE, *Altersstufe#*), *v*).

ANAMNESENORMALANTWORT:

REL({*AnamneseNormalantwort#, Anamnesefrage#, Koerperregion#, Altersstufe#, Geschlecht, Unauffaellige_Antwort*} | {$\mathbf{sib}^1_{anno,1}$, $\mathbf{sib}^1_{anno,2}$, $\mathbf{sib}^2_{anno,1}$, $\mathbf{sib}^2_{anno,2}$, $\mathbf{sib}^2_{anno,3}$}), mit

AnamneseNormalantwort#:	ATT(long |∅)	Primärschlüssel
Anamnesefrage#:	ATT(long |∅)	Fremdschlüssel
Koerperregion#:	ATT(long |∅)	Fremdschlüssel
Altersstufe#:	ATT(long |∅)	Fremdschlüssel
Geschlecht:	ATT(char(9) |∅)	Geschlecht
Unauffaellige_Antwort:	ATT(char(2000) |∅)	Antworttext

$\mathbf{sib}^1_{anno,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*AnamneseNormalantwort#*, $\underline{x}$),
$\mathbf{sib}^1_{anno,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Anamnesefrage#, Koerperregion#, Altersstufe#, Geschlecht, Unauffaellige_Antwort*}, $\underline{x}$),
$\mathbf{sib}^2_{anno,1}(v)$ = 'wahr' :⇔ **FS**((*ANAMNESENORMALANTWORT*, *Anamnesefrage#*), (*ANAMNESEFRAGE*, *Anamnesefrage#*), *v*),
$\mathbf{sib}^2_{anno,2}(v)$ = 'wahr' :⇔ **FS**((*ANAMNESENORMALANTWORT*, *Koerperregion#*), (*KOERPERREGION*, *Koerperregion#*), *v*),
$\mathbf{sib}^2_{anno,3}(v)$ = 'wahr' :⇔ **FS**((*ANAMNESENORMALANTWORT*, *Altersstufe#*), (*ALTERSSTUFE*, *Altersstufe#*), *v*).

KLINVERLAUF:

REL({*KlinVerlauf#, Patient#, Reihenfolgeposition, KlinVerlauf_Text* } | {$\mathbf{sib}^1_{kv,1}$, $\mathbf{sib}^1_{kv,2}$, $\mathbf{sib}^2_{kv,1}$}), mit

KlinVerlauf#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
Reihenfolgeposition:	ATT(integer |∅)	Reihenfolge (bei mehreren klinischen Verläufen eines Falles)
KlinVerlauf_Text:	ATT(char(2000) |∅)	Beschreibung des klinischen Verlaufes

$\mathbf{sib}^1_{kv,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*KlinVerlauf#*, $\underline{x}$),
$\mathbf{sib}^1_{kv,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Patient#, Reihenfolgeposition, KlinVerlauf_Text*}, $\underline{x}$),
$\mathbf{sib}^2_{kv,1}(v)$ = 'wahr' :⇔ **FS**((*KLINVERLAUF*, *Patient#*), (*FALL*, *Patient#*), *v*).

PROGNOSE:

	REL({*Prognose#, Patient#, Prognose_Text* } | {**sib**$^1_{p,1}$, **sib**$^1_{p,2}$, **sib**$^2_{p,1}$}), mit	
Prognose#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
Prognose_Text:	ATT(char(2000) |∅)	Prognose des Patienten

sib$^1_{p,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Prognose#*, $\underline{x}$),
sib$^1_{p,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Patient#, Prognose_Text*}, $\underline{x}$),
sib$^2_{p,1}(v)$ = 'wahr' :⇔ **FS**((*PROGNOSE*, *Patient#*), (*FALL*, *Patient#*), v).

THERAPIEKETTE:

	REL({*Therapiekette#, Patient#, Von, Bis, Reihenfolgeposition, Therapie_Text, Neue_Diagnose, Neue_Therapie, Betreuungsart, Akuttherapie, Kommentar* } | {**sib**$^0_{tk,1}$, **sib**$^1_{tk,1}$, **sib**$^1_{tk,2}$, **sib**$^2_{tk,1}$}), mit	
Therapiekette#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
Von:	ATT(date |∅)	Beginn der Therapie
Bis:	ATT(date |∅)	Ende der Therapie
Reihenfolgeposition:	ATT(integer |∅)	Reihenfolge des Therapiekettenelementes
Therapie_Text:	ATT(char(2000) |∅)	Beschreibung der durchgeführten Therapie
Neue_Diagnose:	ATT(boolean |∅)	Sind aufgrund von Untersuchungsergebnissen bzw. des klin. Verlaufes neue Diagnosen zu stellen?
Neue_Therapie:	ATT(boolean |∅)	Ist aufgrund von Untersuchungsergebnissen bzw. des klin. Verlaufes eine neue Therapie erforderlich?
Betreuungsart:	ATT(char(40) |∅)	ambulant, stationär oder Intensivstation
sib$^0_{tk,3}(w)$ = 'wahr' :⇔	$w \in$ {'ambulant', 'stationaer', 'Intensivstation'}	
Akuttherapie:	ATT(boolean |∅)	Beim Element mit Reihenfolgeposition 1: Ist Akuttherapie notwendig?
Kommentar:	ATT(char(2000) |∅)	Kommentar des Dozenten zur durchgeführten Therapie

sib$^1_{tk,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Therapiekette#*, $\underline{x}$),
sib$^1_{tk,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Patient#, Von, Bis, Reihenfolgeposition, Therapietext, Neue_Diagnose, Neue_Therapie, Betreuungsart, Akuttherapie*}, $\underline{x}$),
sib$^2_{tk,1}(v)$ = 'wahr' :⇔ **FS**((*THERAPIEKETTE*, *Patient#*), (*FALL, Patient#*), v).

THERAPKETTE_DIAGNOSE:

REL({*TherapKette_Diagnose#, Therapiekette#, Diagnose#*} | {**sib**$^1_{tkd,1}$, **sib**$^1_{tkd,2}$, **sib**$^2_{tkd,1}$, **sib**$^2_{tkd,2}$}), mit

TherapKette_Diagnose#:	ATT(long | ∅)	Primärschlüssel
Therapiekette#:	ATT(long | ∅)	Fremdschlüssel
Diagnose#:	ATT(long | ∅)	Fremdschlüssel

sib$^1_{tkd,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*TherapKette_Diagnose#*, $\underline{x}$),
sib$^1_{tkd,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Therapiette#, Diagnose#*}, $\underline{x}$),
sib$^2_{tkd,1}(v)$ = 'wahr' :⇔ **FS**((*THERAPKETTE_DIAGNOSE*, *Therapiette#*), (*THERAPIEKETTE*, *Therapiette#*), *v*),
sib$^2_{tkd,2}(v)$ = 'wahr' :⇔ **FS**((*THERAPKETTE_DIAGNOSE*, *Diagnose#*), (*DIAGNOSE*, *Diagnose#*), *v*).

THERAPKETTE_LUNTERSUCHUNG:

REL({*TherapKette_Luntersuchung#, Therapiekette#, Laboruntersuchung#, Kontrolluntersuchung* } | {**sib**$^1_{tlu,1}$, **sib**$^1_{tlu,2}$, **sib**$^2_{tlu,1}$, **sib**$^2_{tlu,2}$}), mit

TherapKette_Luntersuchung#:	ATT(long | ∅)	Primärschlüssel
Therapiekette#:	ATT(long | ∅)	Fremdschlüssel
Laboruntersuchung#:	ATT(long | ∅)	Fremdschlüssel
Kontrolluntersuchung:	ATT(boolean | ∅)	Handelt es sich um eine Kontrolluntersuchung?

sib$^1_{tlu,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*TherapKette_ LUntersuchung#*, $\underline{x}$),
sib$^1_{tlu,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Therapiekette#, Laboruntersuchung#, Kontrolluntersuchung*}, $\underline{x}$),
sib$^2_{tlu,1}(v)$ = 'wahr' :⇔ **FS**((*THERAPKETTE_LUNTERSUCHUNG*, *Therapiekette#*), (*THERAPIEKETTE*, *Therapiekette#*), *v*),
sib$^2_{tlu,2}(v)$ = 'wahr' :⇔ **FS**((*THERAPKETTE_LUNTERSUCHUNG*, *Laboruntersuchung#*), (*LABORUNTERSUCHUNG*, *Laboruntersuchung#*), *v*).

THERAPKETTE_TUNTERSUCHUNG:

REL({*TherapKette_TUntersuchung#, Therapiekette#, TechnUntersuchung#, Kontrolluntersuchung* } | {**sib**$^1_{ttu,1}$, **sib**$^1_{ttu,2}$, **sib**$^2_{ttu,1}$, **sib**$^2_{ttu,2}$}), mit

TherapKette_Tuntersuchung#:	ATT(long | ∅)	Primärschlüssel
Therapiekette#:	ATT(long | ∅)	Fremdschlüssel
TechnUntersuchung#:	ATT(long | ∅)	Fremdschlüssel
Kontrolluntersuchung:	ATT(boolean | ∅)	Handelt es sich um eine Kontrolluntersuchung?

sib$^1_{ttu,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*TherapKette_ TUntersuchung#*, $\underline{x}$),
sib$^1_{ttu,2,}(\underline{x})$ = 'wahr' :⇔ **nf**({*Therapiekette#, TechnUntersuchung#, Kontrolluntersuchung*}, $\underline{x}$),

$\mathbf{sib}^2_{ttu,1}(v)$ = 'wahr' :⇔ **FS**((*THERAPKETTE_TUNTERSUCHUNG*, *Therapiekette#*),(*THERAPIEKETTE*, *Therapiekette#*), *v*),
$\mathbf{sib}^2_{ttu,2}(v)$ = 'wahr' :⇔ **FS**((*THERAPKETTE_TUNTERSUCHUNG*, *TechnUntersuchung#*), (*TECHNUNTERSUCHUNG*, *TechnUntersuchung#*), *v*).

THERAPKETTE_THERAPPRINZIP:

REL({*TherapKette_TherapPrinzip#, Therapiekette#, Therapieprinzip#, Therapieart* } | {$\mathbf{sib}^0_{tktp,1}$, $\mathbf{sib}^1_{tktp,1}$, $\mathbf{sib}^1_{tktp,2}$, $\mathbf{sib}^2_{tktp,1}$, $\mathbf{sib}^2_{tktp,2}$}), mit

TherapKette_TherapPrinzip#:	**ATT**(long |∅)	Primärschlüssel
Therapiekette#:	**ATT**(long |∅)	Fremdschlüssel
Therapieprinzip#:	**ATT**(long |∅)	Fremdschlüssel
Therapieart:	**ATT**(char(50) |∅)	primär, supportiv, kontraindiziert

$\mathbf{sib}^0_{tktp,1}(w)$ = 'wahr' :⇔ $w \in$ {'primaer', 'supportiv', 'kontraindiziert'}

$\mathbf{sib}^1_{tktp,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*TherapKette_TherapPrinzip#*, $\underline{x}$),
$\mathbf{sib}^1_{tktp,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Therapiekette#, Therapieprinzip#, Therapieart*}, $\underline{x}$),
$\mathbf{sib}^2_{tktp,1}(v)$ = 'wahr' :⇔ **FS**((*THERAPKETTE_THERAPPRINZIP*, *Therapiekette#*), (*THERAPIEKETTE, Therapiekette#*), *v*),
$\mathbf{sib}^2_{tktp,2}(v)$ = 'wahr' :⇔ **FS**((*THERAPKETTE_THERAPPRINZIP*, *Therapieprinzip#*), (*THERAPIEPRINZIP, Therapieprinzip#*), *v*),

THERAPIEPRINZIP:

REL({*Therapieprinzip#, Therapieprinzip_Name* } | {$\mathbf{sib}^1_{tp,1}$, $\mathbf{sib}^1_{tp,2}$}), mit

Therapieprinzip#:	**ATT**(long |∅)	Primärschlüssel
Therapieprinzip_Name:	**ATT**(char(150) |∅)	Name des Therapieprinzips

$\mathbf{sib}^1_{tp,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Therapieprinzip#*, $\underline{x}$),
$\mathbf{sib}^1_{tp,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Therapieprinzip_Name*}, $\underline{x}$).

DIAGNOSE:

REL({*Diagnose#, DreistellerTitel, VierstellerTitel, IcdKreuzcode, IcdSterncode, DiagnoseTex*t} | {$\mathbf{sib}^1_{d,1}$, $\mathbf{sib}^1_{d,2}$}), mit

Diagnose#:	**ATT**(long |∅)	Primärschlüssel
DreistellerTitel:	**ATT**(char(255) |∅)	ICD-10
VierstellerTitel:	**ATT**(char(255) |∅)	ICD-10
IcdKreuzcode:	**ATT**(char(5) |∅)	ICD-10
IcdSterncode:	**ATT**(char(5) |∅)	ICD-10
DiagnoseText:	**ATT**(char(255) |∅)	Diagnosetext

$\mathbf{sib}^1_{d,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Diagnose#*, $\underline{x}$),
$\mathbf{sib}^1_{d,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*IcdKreuzcode, DiagnoseText*}, $\underline{x}$).

FALL_DIAGNOSE:

	REL({Fall_*Diagnose#, Patient#, Diagnose#, Datum, Verdachtsdiagnose, Kommentar, Kategorie, Vergangen*} | {$\mathbf{sib}^0_{fd,1}$, $\mathbf{sib}^1_{fd,1}$, $\mathbf{sib}^1_{fd,2}$, $\mathbf{sib}^2_{fd,1}$, $\mathbf{sib}^2_{fd,2}$}), mit	
Fall_Diagnose#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
Diagnose#:	ATT(char(255) |∅)	Fremdschlüssel
Datum:	ATT(date |∅)	Datum der Diagnosestellung
Verdachtsdiagnose:	ATT(boolean |∅)	Handelt es sich um Verdachtsdiagnose?
Kommentar:	ATT(char(2000) |∅)	Kommentar
Kategorie:	ATT(integer |∅)	1=Hauptdiagnose, 2=Zshg. zur
$\mathbf{sib}^0_{fd,1}(w)$ = 'wahr' :⇔	$w \in \{1, 2, 3\}$	Hauptdiagn. gegeben , 3=kein Zshg. Zur Haupdiagn. bzw. unwichtig?
Vergangen:	ATT(boolean |∅)	Diagnose aus der Vergangenheit (vor Fallbeginn)?

$\mathbf{sib}^1_{fd,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Diagnose#*, $\underline{x}$),
$\mathbf{sib}^1_{fd,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Patient#, Diagnose#, Datum, Verdachtsdiagnose, Kategorie, Vergangen*}, $\underline{x}$),
$\mathbf{sib}^2_{fd,1}(v)$ = 'wahr' :⇔ **FS**((*FALL_DIAGNOSE*, *Patient#*), (*FALL, Patient#*), v),
$\mathbf{sib}^2_{fd,2}(v)$ = 'wahr' :⇔ **FS**((*FALL_DIAGNOSE, Diagnose#*), (*DIAGNOSE*, *Diagnose#*), v).

DIAGNOSE_THERAPIEPRINZIP:

	REL({*Diagnose_Therapieprinzip#, Diagnose#, Therapieprinzip#, Therapieart, Verlauf, Prognose, Kommentar*} | {$\mathbf{sib}^0_{dtp,1}$, $\mathbf{sib}^1_{dtp,1}$, $\mathbf{sib}^1_{dtp,2}$, $\mathbf{sib}^1_{dtp,3}$, $\mathbf{sib}^2_{dtp,1}$, $\mathbf{sib}^2_{dtp,2}$}), mit	
Diagnose_Therapieprinzip#:	ATT(long |∅)	Primärschlüssel
Diagnose#:	ATT(long |∅)	Fremdschlüssel
Therapieprinzip#:	ATT(long |∅)	Fremdschlüssel
Therapieart:	ATT(char(30) |∅)	primär, supportiv
$\mathbf{sib}^0_{dtp,1}(w)$ = 'wahr' :⇔	$w \in$ {'primaer', 'supportiv'},	
Verlauf:	ATT(char(2000) |∅)	Typischer Verlauf der Erkrankung unter best. Therapie
Prognose:	ATT(char(2000) |∅)	Prognose bei Anwendung Therapie
Kommentar:	ATT(char(2000) |∅)	Kommentar

$\mathbf{sib}^1_{dtp,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Diagnose_Therapieprinzip#*, $\underline{x}$),
$\mathbf{sib}^1_{dtp,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Diagnose#, Therapieprinzip#*}, $\underline{x}$),
$\mathbf{sib}^1_{dtp,3}(\underline{x})$ = 'wahr' :⇔ **nf1**({*Therapieart, Verlauf, Prognose, Kommentar*}, $\underline{x}$),
$\mathbf{sib}^2_{dtp,1}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_THERAPIEPRINZIP*, *Diagnose#*), (*DIAGNOSE, Diagnose#*), v),

$\mathbf{sib}^2_{dtp,2}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_THERAPIEPRINZIP*, *Therapieprinzip#*), (*THERAPIEPRINZIP, Therapieprinzip#*), *v*).

DIAGNOSE_ANAMNESEANTWORT:

REL({*Diagnose_Anamneseantwort#, Diagnose#, Anamneseantwort#, Evidenz, Kommentar*} | {$\mathbf{sib}^1_{daa,1}$, $\mathbf{sib}^1_{daa,2}$, $\mathbf{sib}^2_{daa,1}$, $\mathbf{sib}^2_{daa,2}$}), mit

Diagnose_Anamneseantwort#:	ATT(long |∅)	Primärschlüssel
Diagnose#:	ATT(long |∅)	Fremdschlüssel
Anamneseantwort#:	ATT(long |∅)	Fremdschlüssel
Evidenz:	ATT(char(10) |∅)	Evidenz für Diagnose bei Anamneseantwort
Kommentar:	ATT(char(2000) |∅)	Kommentar zum Zusammenhang Diagnose - Anamneseantwort

$\mathbf{sib}^1_{daa,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Diagnose_Anamneseantwort#*, $\underline{x}$),
$\mathbf{sib}^1_{daa,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Diagnose#, Anamneseantwort#, Evidenz*}, $\underline{x}$),
$\mathbf{sib}^2_{daa,1}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_ANAMNESEANTWORT*, *Diagnose#*), (*DIAGNOSE*, *Diagnose#*), *v*),
$\mathbf{sib}^2_{daa,2}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_ANAMNESEANTWORT*, *Anamneseantwort#*), (*ANAMNESEANTWORT*, *Anamneseantwort#*), *v*).

DIAGNOSE_KLINUNTERSUCHERG:

REL({*Diagnose_KlinUntersuchErg#, Diagnose#, KlinUntersuchErg#, Evidenz, Kommentar*} | {$\mathbf{sib}^1_{dkue,1}$, $\mathbf{sib}^1_{dkue,2}$, $\mathbf{sib}^2_{dkue,1}$, $\mathbf{sib}^2_{dkue,2}$}), mit

Diagnose_KlinUntersuchErg#:	ATT(long |∅)	Primärschlüssel
Diagnose#:	ATT(long |∅)	Fremdschlüssel
KlinUntersuchErg#:	ATT(long |∅)	Fremdschlüssel
Evidenz:	ATT(char(10) |∅)	Evidenz für Diagnose bei Vorliegen des Ergebnisses einer klin. Untersuchung
Kommentar:	ATT(char(2000) |∅)	Kommentar zum Zusammenhang Diagnose - Ergebnis klinische Untersuchung

$\mathbf{sib}^1_{dkue,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Diagnose_KlinUntersuchErg#*, $\underline{x}$),
$\mathbf{sib}^1_{dkue,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Diagnose#, KlinUntersuchErg#, Evidenz*}, $\underline{x}$),
$\mathbf{sib}^2_{dkue,1}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_KLINUNTERSUCHERG*, *Diagnose#*), (*DIAGNOSE, Diagnose#*), *v*),
$\mathbf{sib}^2_{dkue,2}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_KLINUNTERSUCHERG*, *KlinUntersuchErg#*), (*KLINUNTERSUCHERG, KlinUntersuchErg#*), *v*).

DIAGNOSE_TECHNUNTERSUCHUNG:

REL({*Diagnose_TechnUntersuchung#, Diagnose#, TechnUntersuchung#, Evidenz, Kommentar*} | {$\mathbf{sib}^1_{dtu,1}$, $\mathbf{sib}^1_{dtu,2}$, $\mathbf{sib}^2_{dtu,1}$, $\mathbf{sib}^2_{dtu,2}$}), mit

Diagnose_TechnUntersuchung#:	ATT(long | ∅)	Primärschlüssel
Diagnose#:	ATT(long | ∅)	Fremdschlüssel
TechnUntersuchung#:	ATT(long | ∅)	Fremdschlüssel
Evidenz:	ATT(char(10) | ∅)	Evidenz für Diagnose bei Vorliegen des Ergebnisses einer techn. Untersuchung
Kommentar:	ATT(char(2000) | ∅)	Kommentar zum Zusammenhang Diagnose - Ergebnis techn. Untersuchung

$\mathbf{sib}^1_{dtu,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Diagnose_TechnUntersuchung#*, $\underline{x}$),
$\mathbf{sib}^1_{dtu,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Diagnose#, TechnUntersuchung#, Evidenz*}, $\underline{x}$),
$\mathbf{sib}^2_{dtu,1}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_TECHNUNTERSUCHUNG*, *Diagnose#*), (*DIAGNOSE*, *Diagnose#*), *v*),
$\mathbf{sib}^2_{dtu,2}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_TECHNUNTERSUCHUNG*, *TechnUntersuchung#*), (*TECHNUNTERSUCHUNG*, *TechnUntersuchung#*), *v*).

DIAGNOSE_LABORTESTERGEBNIS:

REL({*Diagnose_LaborTestergebnis#, Diagnose#, LaborTestergebnis#, Evidenz, Kommentar*} | {$\mathbf{sib}^1_{dlte,1}$, $\mathbf{sib}^1_{dlte,2}$, $\mathbf{sib}^2_{dlte,1}$, $\mathbf{sib}^2_{dlte,2}$}), mit

Diagnose_LaborTestergebnis#:	ATT(long | ∅)	Primärschlüssel
Diagnose#:	ATT(long | ∅)	Fremdschlüssel
LaborTestergebnis#:	ATT(long | ∅)	Fremdschlüssel
Evidenz:	ATT(char(10) | ∅)	Evidenz für Diagnose bei Vorliegen des Labortestergebnisses
Kommentar:	ATT(char(2000) | ∅)	Kommentar zum Zusammenhang Diagnose - Ergebnis Labortest

$\mathbf{sib}^1_{dlte,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Diagnose_LaborTestergebnis#*, $\underline{x}$),
$\mathbf{sib}^1_{dlte,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Diagnose#, LaborTestergebnis#, Evidenz*}, $\underline{x}$),
$\mathbf{sib}^2_{dlte,1}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_LABORTESTERGEBNIS*, *Diagnose#*), (*DIAGNOSE, Diagnose#*), *v*),
$\mathbf{sib}^2_{dlte,2}(v)$ = 'wahr' :⇔ **FS**((*DIAGNOSE_LABORTESTERGEBNIS*, *LaborTestergebnis#*), (*LABORTESTERGEBNIS, LaborTestergebnis#*), *v*).

ERREGER:

	REL({*Erreger#, Erreger_Name, Erreger_Gattung, Gattung_Kuerzel*} | {**sib**$^1_{err,1}$, **sib**$^1_{err,2}$}), mit	
Erreger#:	ATT(long |∅)	Primärschlüssel
Erreger_Name:	ATT(char(200) |∅)	Name des Erregers
Erreger_Gattung:	ATT(char(200) |∅)	Gattung, welcher der Erreger angehört
Gattung_Kuerzel:	ATT(char(5) |∅)	Kürzel für Erregergattung

sib$^1_{err,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Erreger#*, $\underline{x}$),
sib$^1_{err,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Erreger_Name*}, $\underline{x}$),

FALL_ERREGER:

	REL({*Fall_Erreger#, Patient#, Erreger#*} | {**sib**$^1_{fe,1}$, **sib**$^1_{fe,2}$, **sib**$^2_{fe,1}$, **sib**$^2_{fe,2}$}), mit	
Fall_Erreger#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
Erreger#:	ATT(long |∅)	Fremdschlüssel

sib$^1_{fe,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Fall_Erreger#*, $\underline{x}$),
sib$^1_{fe,1}(\underline{x})$ = 'wahr' :⇔ **nf**({*Patient#, Erreger#*}, $\underline{x}$),
sib$^2_{fe,1}(v)$ = 'wahr' :⇔ **FS**((*FALL_ERREGER*, *Patient#*), (*FALL*, *Patient#*), *v*),
sib$^2_{fe,2}(v)$ = 'wahr' :⇔ **FS**((*FALL_ERREGER*, *Erreger#*), (*ERREGER*, *Erreger#*)*v*).

FALL_LABORTEST:

	REL({*Fall_Labortest#, Patient#, Labortest#, Ablehnungsgrund*} | {**sib**$^1_{fla,1}$, **sib**$^1_{fla,2}$, **sib**$^2_{fla,1}$, **sib**$^2_{fla,2}$}), mit	
Fall_Labortest#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
Labortest#:	ATT(long |∅)	Fremdschlüssel
Ablehnungsgrund:	ATT(char(2000) |∅)	Weshalb darf Labortest nicht durchgeführt werden?

sib$^1_{fla,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Fall_Labortest#*, $\underline{x}$),
sib$^1_{fla,1}(\underline{x})$ = 'wahr' :⇔ **nf**({*Patient#, Labortest#, Ablehungsgrund*}, $\underline{x}$),
sib$^2_{fla,1}(v)$ = 'wahr' :⇔ **FS**((*FALL_LABORTEST*, *Patient#*), (*FALL*, *Patient#*), *v*),
sib$^2_{fla,2}(v)$ = 'wahr' :⇔ **FS**((*FALL_LABORTEST*, *Labortest#*), (*LABORTEST*, *Labortest#*), *v*).

FALL_TUVERFAHREN:

REL({*Fall_TUVerfahren#, Patient#, TechnUntersuchungsverfahren#, Ablehnungsgrund*} | {**sib**$^1_{ftu,1}$, **sib**$^1_{ftu,2}$, **sib**$^2_{ftu,1}$, **sib**$^2_{ftu,2}$}), mit

Fall_TUVerfahren#:	ATT(long |∅)	Primärschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
TechnUntersuchungsverfahren#:	ATT(long |∅)	Fremdschlüssel
Ablehnungsgrund:	ATT(char(2000) |∅)	Weshalb darf techn. Untersuchungsverfahren nicht angewandt werden?

sib$^1_{ftu,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Fall_TUVerfahren#*, $\underline{x}$),
sib$^1_{ftu,1}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Patient#, TechnUntersuchungsverfahren#, Ablehungsgrund*}, $\underline{x}$),
sib$^2_{ftu,1}$(v) = 'wahr' :⇔ **FS**((*FALL_TUVERFAHREN*, *Patient#*), (*FALL*, *Patient#*), v),
sib$^2_{ftu,2}$(v) = 'wahr' :⇔ **FS**((*FALL_TUVERFAHREN*, *TechnUntersuchungsverfahren#*), (*TECHNUNTERSUCHUNGSVERFAHREN*, *TechnUntersuchungsverfahren#*), v).

ANTWORTTYP:

REL({*Antworttyp#, Antworttyp_Name, Vorteile, Nachteile*} | {**sib**$^1_{at,1}$, **sib**$^1_{at,2}$}), mit

Antworttyp#:	ATT(long |∅)	Primärschlüssel
Antworttyp_Name#:	ATT(char(30) |∅)	Name des Antworttyps
Vorteile:	ATT(char(2000) |∅)	Vorteile des Antworttyps
Nachteile:	ATT(char(2000) |∅)	Nachteile des Antworttyps

sib$^1_{at,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Antworttyp#*, $\underline{x}$),
sib$^1_{at,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Antworttyp_Name*}, $\underline{x}$).

SCHWIERIGKEITSSTUFE:

REL({*Schwierigkeitsstufe#, Schwierigkeitsstufe_Name, Abgrenzung*} | {**sib**$^1_{sst,1}$, **sib**$^1_{sst,2}$}), mit

Schwierigkeitsstufe#:	ATT(long |∅)	Primärschlüssel
Schwierigkeitsstufe_Name#:	ATT(char(50) |∅)	Bezeichnung der Schwierigkeitsstufe
Abgrenzung:	ATT(char(2000) |∅)	Abgrenzung zu anderen Schwierigkeitsstufen

sib$^1_{sst,1}$($\underline{x}$) = 'wahr' :⇔ **PS**(*Schwierigkeitsstufe#*, $\underline{x}$),
sib$^1_{sst,2}$($\underline{x}$) = 'wahr' :⇔ **nf**({*Schwierigkeitsstufe_Name, Abgrenzung*}, $\underline{x}$).

FRAGEKATEGORIE:

	REL({*Fragekategorie#, Fragekategorie_Name, Beschreibung*} | {$\mathbf{sib}^1_{fk,1}$, $\mathbf{sib}^1_{fk,2}$}), mit	
Fragekategorie#:	ATT(long |∅)	Primärschlüssel
Fragekategorie_Name#:	ATT(char(50) |∅)	Name der Fragekategorie
Beschreibung:	ATT(char(2000) |∅)	Beschreibung der Fragekategorie

$\mathbf{sib}^1_{fk,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Fragekategorie#*, $\underline{x}$),
$\mathbf{sib}^1_{fk,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Fragekategorie_Name*}, $\underline{x}$).

FRAGE:

	REL({*Frage#, Autor#, Antworttyp#, Schwierigkeitsstufe#, Fragekategorie#, Erstelldatum, Fragetext, Bild, MaxAntwortzeit*} | {$\mathbf{sib}^1_{fra,1}$, $\mathbf{sib}^1_{fra,2}$, $\mathbf{sib}^2_{fra,1}$, $\mathbf{sib}^2_{fra,2}$, $\mathbf{sib}^2_{fra,3}$, $\mathbf{sib}^2_{fra,4}$}), mit	
Frage#:	ATT(long |∅)	Primärschlüssel
Autor#:	ATT(long |∅)	Fremdschlüssel
Antworttyp#:	ATT(long |∅)	Fremdschlüssel
Schwierigkeitsstufe#:	ATT(long |∅)	Fremdschlüssel
Fragekategorie#:	ATT(long |∅)	Fremdschlüssel
Erstelldatum:	ATT(date |∅)	Erstelldatum der Frage
Fragetext:	ATT(char(2000) |∅)	Fragetext
Bild:	ATT(char(50) |∅)	Verweis auf Bild zur Frage
MaxAntwortzeit:	ATT(integer |∅)	Maximale Antwortzeit in Sekunden

$\mathbf{sib}^1_{fra,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Frage#*, $\underline{x}$),
$\mathbf{sib}^1_{fra,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Autor#, Antworttyp#, Schwierigkeitsstufe#, Fragekategorie#, Erstelldatum, Fragetext*}, $\underline{x}$),
$\mathbf{sib}^2_{fra,1}(v)$ = 'wahr' :⇔ **FS**((*FRAGE*, *Autor#*), (*AUTOR, Autor#*), *v*),
$\mathbf{sib}^2_{fra,2}(v)$ = 'wahr' :⇔ **FS**((*FRAGE*, *Antworttyp#*), (*ANTWORTTYP, Antworttyp#*), *v*),
$\mathbf{sib}^2_{fra,3}(v)$ = 'wahr' :⇔ **FS**((*FRAGE*, *Schwierigkeitsstufe#*), (*SCHWIERIGKEITSSTUFE, Schwierigkeitsstufe#*), *v*),
$\mathbf{sib}^2_{fra,4}(v)$ = 'wahr' :⇔ **FS**((*FRAGE*, *Fragekategorie#*), (*FRAGEKATEGORIE, Fragekategorie#*), *v*).

ANTWORTBILDBESCHRIFTUNG:

REL({*AntwortBildbeschriftung#, Frage#, Labelnummer, Richtige_Antwort, Kommentar*} | {**sib**$^1_{abb,1}$, **sib**$^1_{abb,2}$, **sib**$^2_{abb,1}$}), mit

AntwortBildbeschriftung#:	ATT(long |∅)	Primärschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel
Labelnummer:	ATT(integer |∅)	Nummer des Labels
Richtige_Antwort:	ATT(char(200) |∅)	Text der richtigen Antwort
Kommentar:	ATT(char(2000) |∅)	Kommentar

sib$^1_{aab,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*AntwortBildbeschriftung#*, $\underline{x}$),
sib$^1_{aab,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Frage#, Labelnummer, Richtige_Antwort*}, $\underline{x}$),
sib$^2_{abb,1}(v)$ = 'wahr' :⇔ **FS**((*ANTWORTBILDBESCHRIFTUNG*, *Frage#*), (*FRAGE*, *Frage#*), *v*).

ANTWORTLUECKENTEXT:

REL({*AntwortLueckentext#, Frage#, Lueckennummer, Richtige_Antwort, Kommentar*} | {**sib**$^1_{alt,1}$, **sib**$^1_{alt,2}$, **sib**$^2_{alt,1}$}), mit

AntwortLueckentext#:	ATT(long |∅)	Primärschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel
Lueckennummer:	ATT(integer |∅)	Nummer der Lücke
Richtige_Antwort:	ATT(char(200) |∅)	Text der richtigen Antwort
Kommentar:	ATT(char(2000) |∅)	Kommentar

sib$^1_{alt,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*AntwortLueckentext#*, $\underline{x}$),
sib$^1_{alt,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Frage#, Lueckennummer, Richtige_Antwort*}, $\underline{x}$),
sib$^2_{alt,1}(v)$ = 'wahr' :⇔ **FS**((*ANTWORTLUECKENTEXT*, *Frage#*), (*FRAGE*, *Frage#*), *v*).

ANTWORTAUSWAHLMC:

REL({*AntwortAuswahlMC#, Frage#, Typ, Antwort, Korrektheit, Kommentar*} | { **sib**$^0_{aam,1}$, **sib**$^1_{aam,1}$, **sib**$^1_{aam,2}$, **sib**$^2_{aam,1}$}), mit

AntwortAuswahlMC#:	ATT(long |∅)	Primärschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel
Typ:	ATT(char(1) |∅)	Auswahl- oder MC-Frage?
sib$^0_{aam,1}(w)$ = 'wahr' :⇔	$w \in$ {'a', 'm'},	
Antwort:	ATT(char(200) |∅)	Text der Antwort
Korrektheit:	ATT(boolean |∅)	Antwort korrekt?
Kommentar:	ATT(char(2000) |∅)	Kommentar

sib$^1_{aam,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*AntwortAuswahlMC#*, $\underline{x}$),
sib$^1_{aam,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Frage#, Typ, Antwort, Korrektheit*}, $\underline{x}$),
sib$^2_{aam,1}(v)$ = 'wahr' :⇔ **FS**((*ANTWORTAUSWAHLMC*, *Frage#*), (*FRAGE*, *Frage#*), *v*).

ANTWORTALTERNATIV:

REL({*AntwortAlternativ#, Frage#, Richtige_Antwort, Falsche_Antwort, Kommentar*} | {**sib**$^1_{aal,1}$, **sib**$^1_{aal,2}$, **sib**$^2_{aal,1}$}), mit

AntwortAlternativ#:	ATT(long |∅)	Primärschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel
Richtige_Antwort:	ATT(char(200) |∅)	Text einer korrekten Antwort
Falsche_Antwort:	ATT(char(200) |∅)	Text einer falschen Antwort
Kommentar:	ATT(char(2000) |∅)	Kommentar

sib$^1_{aal,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*AntwortAlternativ*, $\underline{x}$),
sib$^1_{aal,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Frage#, Richtige_Antwort, Falsche_Antwort*}, $\underline{x}$),
sib$^2_{aal,1}(v)$ = 'wahr' :⇔ **FS**((*ANTWORTALTERNATIV*, *Frage#*), (*FRAGE*, *Frage#*), v).

ANTWORTFREI:

REL({*AntwortFrei#, Frage#, Richtige_Antwort, Kommentar*} | {**sib**$^1_{aaf1}$, **sib**$^1_{aaf2}$, **sib**$^2_{aaf1}$}), mit

AntwortFrei#:	ATT(long |∅)	Primärschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel
Richtige_Antwort:	ATT(char(200) |∅)	Text einer korrekten Antwort
Kommentar:	ATT(char(2000) |∅)	Kommentar

sib$^1_{aaf,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*AntwortFrei#*, $\underline{x}$),
sib$^1_{aaf,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Frage#, Richtige_Antwort*}, $\underline{x}$),
sib$^2_{aaf,1}(v)$ = 'wahr' :⇔ **FS**((*ANTWORTFREI*, *Frage#*), (*FRAGE*, *Frage#*), v).

ANTWORTMATCHING:

REL({*AntwortMatching#, Frage#, Begriff_A, Begriff_B, Kommentar*} | {**sib**$^1_{am,1}$, **sib**$^1_{am,2}$, **sib**$^2_{am,1}$}), mit

AntwortMatching#:	ATT(long |∅)	Primärschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel
Begriff_A:	ATT(char(200) |∅)	Dieser Begriff ist jenem
Begriff_B:	ATT(char(200) |∅)	Begriff zugeordnet
Kommentar:	ATT(char(2000) |∅)	Kommentar

sib$^1_{am,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*AntwortMatching#*, $\underline{x}$),
sib$^1_{am,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Frage#, Begriff_A, Begriff_B*}, $\underline{x}$),
sib$^2_{am,1}(v)$ = 'wahr' :⇔ **FS**((*ANTWORTMATCHING*, *Frage#*), (*FRAGE*, *Frage#*), v).

ANTWORTANORDNEN:

REL({*AntwortAnordnen#, Frage#, Typ, Begriff_Icon, Position, Kommentar*} | { **sib**$^0_{anan,1}$, **sib**$^1_{anan,1}$, **sib**$^1_{anan,2}$, **sib**$^2_{anan,1}$}), mit

AntwortAnordnen#:	ATT(long |∅)	Primärschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel
Typ:	ATT(char(1) |∅)	Begriffe='b', oder Icons = 'i'

$\mathbf{sib}^0_{anan,1}(w)$ = 'wahr' :⇔	$w \in$ {'b', 'i'}	
Begriff_Icon:	ATT(char(50) |∅)	Begriff bzw. Pfad auf Icon
Position:	ATT(integer |∅)	Position in der Anordnungsreihenfolge
Kommentar:	ATT(char(2000) |∅)	Kommentar

$\mathbf{sib}^1_{anan,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*AntwortAnordnen#*, $\underline{x}$),
$\mathbf{sib}^1_{anan,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Frage#, Typ, Begriff_Icon, Position*}, $\underline{x}$),
$\mathbf{sib}^2_{anan,1}(v)$ = 'wahr' :⇔ **FS**((*ANTWORTANORDNEN*, *Frage#*), (*FRAGE*, *Frage#*), v).

WBT_SYSTEM_FRAGE:

REL({*WBT_System_Frage#, WBT_System#, Frage#*} | {$\mathbf{sib}^1_{wsf,1}$, $\mathbf{sib}^1_{wsf,2}$, $\mathbf{sib}^2_{wsf,1}$, $\mathbf{sib}^2_{wsf,2}$}), mit

WBT_System_Nutzer#:	ATT(long |∅)	Primärschlüssel
WBT_System#:	ATT(long |∅)	Fremdschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel

$\mathbf{sib}^1_{wsf,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*WBT_System_Knoten#*, $\underline{x}$),
$\mathbf{sib}^1_{wsf,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*WBT_System#, Frage#*}, $\underline{x}$),
$\mathbf{sib}^2_{wsf,1}(v)$ = 'wahr' :⇔ **FS**((*WBT_SYSTEM_FRAGE*, *WBT_System#*), (*WBT_SYSTEM*, *WBT_System#*), v),
$\mathbf{sib}^2_{wsf,2}(v)$ = 'wahr' :⇔ **FS**((*WBT_SYSTEM_FRAGE*, *Frage#*), (*FRAGE*, *Frage#*), v).

NUTZER:

REL({*Nutzer#, Kennung, Passwort, EMail, Talk, LoginDatum, Bedienungsanleitung, Praesentationsform, Fragen, Rueckmeldezeitpunkt, Anmeldedatum, Semesterzahl, AdaptationAktiviert, ProspektivesLaden, TotaleNurPatho*} | {$\mathbf{sib}^1_{nu,1}$, $\mathbf{sib}^1_{nu,2}$}), mit

Nutzer#:	ATT(long |∅)	Primärschlüssel
Kennung:	ATT(char(20) |∅)	Kennung für Nutzerlogin
Passwort:	ATT(char(20) |∅)	Paßwort für Nutzerlogin
Email:	ATT(char(50|∅)	E-Mail Adresse des Nutzers
Talk:	ATT(boolean |∅)	Darf Nutzer angetalkt werden?
LoginDatum:	ATT(date |∅)	Datum der letzten Systemnutzung
Bedienungsanleitung:	ATT(integer |∅)	Automatische Bedienungsanleitung ja/nein
Praesentationsform:	ATT(integer |∅)	Wie zeigt sich ein Fall dem Nutzer?
Fragen:	ATT(integer |∅)	Welche Fragekategorien werden eingeblendet?
Rueckmeldezeitpunkt:	ATT(integer |∅)	Wann erfolgen Systemrückmeldungen?
Anmeldedatum:	ATT(date |∅)	Wann hat sich der Nutzer das erste mal beim System angemeldet?

Semesterzahl:	ATT(integer |∅)	In welchem Semester befand sich der Nutzer zum Zeitpunkt der ersten Anmeldung?
AdaptationAktiviert:	ATT(boolean |∅)	Ist die automatische Systemanpassung aktiviert?
ProspektivesLaden:	ATT(boolean |∅)	Ist das prospektive Laden von Falldaten und Domänenwissen aktiviert?
TotaleNurPatho:	ATT(boolean |∅)	Sollen bei der Präsentations-/Interaktionsform *Totale* nur pathologische Befunde angezeigt werden?

$\mathbf{sib}^1_{nu,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Nutzer#*, $\underline{x}$),
$\mathbf{sib}^1_{nu,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Kennung, Passwort, EMail, Talk, Bedienungsanleitung, Praesentationsform, Fragen, Rueckmeldezeitpunkt, Anmeldedatum, Semesterzahl, AdaptationAktiviert, ProspektivesLaden, TotaleNurPatho*}, $\underline{x}$).

WBT_SYSTEM_NUTZER:

REL({*WBT_System_Nutzer#, WBT_System#, Nutzer#*} | {$\mathbf{sib}^1_{wsn,1}$, $\mathbf{sib}^1_{wsn,2}$, $\mathbf{sib}^2_{wsn,1}$, $\mathbf{sib}^2_{wsn,2}$}), mit

WBT_System_Nutzer#:	ATT(long |∅)	Primärschlüssel
WBT_System#:	ATT(long |∅)	Fremdschlüssel
Nutzer#:	ATT(long |∅)	Fremdschlüssel

$\mathbf{sib}^1_{wsn,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*WBT_System_Nutzer#*, $\underline{x}$),
$\mathbf{sib}^1_{wsn,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*WBT_System#, Nutzer#* }, $\underline{x}$),
$\mathbf{sib}^2_{wsn,1}(v)$ = 'wahr' :⇔ **FS**((*WBT_SYSTEM_NUTZER*, *WBT_System#*), (*WBT_SYSTEM, WBT_System#*), *v*),
$\mathbf{sib}^2_{wsn,2}(v)$ = 'wahr' :⇔ **FS**((*WBT_SYSTEM_NUTZER*, *Nutzer#*), *(NUTZER, Nutzer#*), *v*).

FALLBEARBEITUNG:

REL({*Fallbearbeitung#, Nutzer#, Patient#, Erfolg, Bearbeitungsdauer, Zeitpunkt*} | {$\mathbf{sib}^1_{fab,1}$, $\mathbf{sib}^1_{fab,2}$, $\mathbf{sib}^2_{fab,1}$, $\mathbf{sib}^2_{fab,2}$}), mit

Fallbearbeitung#:	ATT(long |∅)	Primärschlüssel
Nutzer#:	ATT(long |∅)	Fremdschlüssel
Patient#:	ATT(long |∅)	Fremdschlüssel
Erfolg:	ATT(real |∅)	Erfolg der Fallbearbeitung
Bearbeitungsdauer:	ATT(long |∅)	Bearbeitungsdauer in Sekunden
Zeitpunkt:	ATT(date |∅)	Zeitpunkt der Bearbeitung

$\mathbf{sib}^1_{fab,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Fallbearbeitung#*, $\underline{x}$),
$\mathbf{sib}^1_{fab,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Nutzer#, Patient#, Erfolg, Bearbeitungsdauer, Zeitpunkt*}, $\underline{x}$),
$\mathbf{sib}^2_{fab,1}(v)$ = 'wahr' :⇔ **FS**((*FALLBEARBEITUNG*, *Nutzer#*), (*NUTZER, Nutzer#*), *v*),

$\mathbf{sib}^2_{fab,2}(v)$ = 'wahr' :⇔ **FS**((*FALLBEARBEITUNG*, *Patient#*), (*FALL, Patient#*), *v*).

FRAGEBEARBEITUNG:

	REL({*Fragebearbeitung#, Nutzer#, Patient#, Erfolg, Bearbeitungsdauer, Zeitpunkt*} | {$\mathbf{sib}^1_{frb,1}$, $\mathbf{sib}^1_{frb,2}$, $\mathbf{sib}^2_{frb,1}$, $\mathbf{sib}^2_{frb,2}$}), mit	
Fragebearbeitung#:	ATT(long |∅)	Primärschlüssel
Nutzer#:	ATT(long |∅)	Fremdschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel
Erfolg:	ATT(boolean|∅)	Erfolg der Fragebearbeitung
Bearbeitungsdauer:	ATT(long |∅)	Bearbeitungsdauer in Sekunden
Zeitpunkt:	ATT(date |∅)	Zeitpunkt der Bearbeitung

$\mathbf{sib}^1_{frb,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Fragebearbeitung#*, $\underline{x}$),
$\mathbf{sib}^1_{frb,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Nutzer#, Frage#, Erfolg, Bearbeitungsdauer, Zeitpunkt*}, $\underline{x}$),
$\mathbf{sib}^2_{frb,1}(v)$ = 'wahr' :⇔ **FS**((*FRAGEBEARBEITUNG*, *Nutzer#*), (*NUTZER, Nutzer#*), *v*),
$\mathbf{sib}^2_{frb,2}(v)$ = 'wahr' :⇔ **FS**((*FRAGEBEARBEITUNG*, *Frage#*), (*FRAGE*, *Frage#*), *v*).

KNOTENBESUCH:

	REL({*Knotenbesuch#, Nutzer#, Knoten#, Verweildauer, Zeitpunkt*} | {$\mathbf{sib}^1_{knb,1}$, $\mathbf{sib}^1_{knb,2}$, $\mathbf{sib}^2_{knb,1}$, $\mathbf{sib}^2_{knb,2}$}), mit	
Knotenbesuch#:	ATT(long |∅)	Primärschlüssel
Nutzer#:	ATT(long |∅)	Fremdschlüssel
Knoten#:	ATT(long |∅)	Fremdschlüssel
Verweildauer:	ATT(long |∅)	Verweildauer in Sekunden
Zeitpunkt:	ATT(date |∅)	Zeitpunkt des Knotenbesuchs

$\mathbf{sib}^1_{knb,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Knotenbesuch#*, $\underline{x}$),
$\mathbf{sib}^1_{knb,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Nutzer#, Knoten#, Verweildauer, Zeitpunkt*}, $\underline{x}$),
$\mathbf{sib}^2_{knb,1}(v)$ = 'wahr' :⇔ **FS**((*KNOTENBESUCH*, *Nutzer#*), (*NUTZER, Nutzer#*), *v*),
$\mathbf{sib}^2_{knb,2}(v)$ = 'wahr' :⇔ **FS**((*KNOTENBESUCH*, *Knoten#*), (*KNOTEN, Knoten#*), *v*).

HILFETEXT:

	REL({*Hilfetext#, Ereignis , Hilfetext*} | {$\mathbf{sib}^1_{hit,1}$, $\mathbf{sib}^1_{hit,2}$}), mit	
Hilfetext#:	ATT(long |∅)	Primärschlüssel
Ereignis:	ATT(char(50) |∅)	Ereignis, bei dem Hilfetext angezeigt werden soll
Hilfetext:	ATT(char(2000) |∅)	Hilfetext

$\mathbf{sib}^1_{hit,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Hilfetext#*, $\underline{x}$),
$\mathbf{sib}^1_{hit,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Ergeignis, Hilfetext*}, $\underline{x}$).

HILFEPROTOKOLL:

REL(Hilfeprotokoll*#, Nutzer#, Hilfetext#, Datum*} | {**sib**$^1_{hp,1}$, **sib**$^1_{hip,2}$, **sib**$^2_{hip,1}$, **sib**$^2_{hip,2}$}), mit

Hilfeprotokoll#:	ATT(long |∅)	Primärschlüssel
Nutzer#:	ATT(long |∅)	Fremdschlüssel
Hilfetext#:	ATT(long |∅)	Fremdschlüssel
Datum:	ATT(date |∅)	Datum der Hilfestellung

sib$^1_{hip,1}$($\underline{x}$) = ‘wahr’ :⇔ **PS**(*Hilfeprotokoll#*, $\underline{x}$),
sib$^1_{hip,2}$($\underline{x}$) = ‘wahr’ :⇔ **nf**({*Nutzer#, Hilfetext#, Datum*}, $\underline{x}$),
sib$^2_{hip,1}$(v) = ‘wahr’ :⇔ **FS**((*HILFEPROTOKOLL*, *Nutzer#*), (*NUTZER, Nutzer#*), v),
sib$^2_{hip,2}$(v) = ‘wahr’ :⇔ **FS**((*HILFEPROTOKOLL*, *Hilfetext#*), (*HILFETEXT, Hilfetext#*), v).

FRAGEZUORDNUNG:

REL(Fragezuordnung*#, Frage#, Objek#, Objektname*} | {**sib**$^1_{fzu,1}$, **sib**$^1_{fzu,2}$, **sib**$^2_{fzu,1}$, **sib**$^2_{fzu,2}$}), mit

Fragezuordnung#:	ATT(long |∅)	Primärschlüssel
Frage#:	ATT(long |∅)	Fremdschlüssel
Objekt#:	ATT(long |∅)	Fremdschlüssel
Objektname:	ATT(char(50) |∅)	Name des referenzierten Objektes
InDiskus:	ATT(boolean |∅)	Soll Frage in Falldiskussion aufgeführt werden?

sib$^1_{fzu,1}$($\underline{x}$) = ‘wahr’ :⇔ **PS**(*Fragezuodnung#*, $\underline{x}$),
sib$^1_{fzu,2}$($\underline{x}$) = ‘wahr’ :⇔ **nf**({*Frage#, Objekt#, Objektname*}, $\underline{x}$),
sib$^2_{fzu,1}$(v) = ‘wahr’ :⇔ **FS**((*FRAGEZUORDNUNG, Frage#*), (*FRAGE, Frage#*), v),
sib$^2_{fzu,2}$(v) = ‘wahr’ :⇔ **FS**((*FRAGEZUORDNUNG*, *Objekt#*), (*OBJEKTNAME, Objekt#*), v).

KNOTEN:

REL({*Knoten#, Kapitel#, Knotentyp#, Navigationsleiste#, Erstelldatum, Knoten_Name, Pfad, Knotenart*} | { **sib**$^0_{kn,1}$, **sib**$^1_{kn,1}$, **sib**$^1_{kn,2}$, **sib**$^2_{kn,1}$, **sib**$^2_{kn,2}$, **sib**$^2_{kn,3}$}), mit

Knoten#:	ATT(long |∅)	Primärschlüssel
Kapitel#:	ATT(long |∅)	Fremdschlüssel
Knotentyp#:	ATT(long |∅)	Fremdschlüssel
Navigationsleiste#:	ATT(long |∅)	Fremdschlüssel
Knoten_Name:	ATT(char(50) |∅)	Name des Knotens
Erstelldatum:	ATT(date |∅)	Erstelldatum
Pfad:	ATT(char(50) |∅)	Pfad zu Restcode
Knotenart:	ATT(char(20) |∅)	Art des Knotens
sib$^0_{kn,1}$(w) = ‘wahr’ :⇔	$w \in$ {‘RestHTML’, ‘Sonstiges’}	

$\mathbf{sib}^1_{kn,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Knoten#*, $\underline{x}$),
$\mathbf{sib}^1_{kn,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Kapitel#, Knotentyp#, Navigationsleiste#, Erstelldatum, Pfad, Knotenart*}, $\underline{x}$),
$\mathbf{sib}^2_{kn,1}(v)$ = 'wahr' :⇔ **FS**((*KNOTEN*, *Kapitel#*), (*KAPITEL*, *Kapitel#*), *v*),
$\mathbf{sib}^2_{kn,2}(v)$ = 'wahr' :⇔ **FS**((*KNOTEN*, *Knotentyp#*), (*KNOTENTYP*, *Knotentyp#*), *v*),
$\mathbf{sib}^2_{kn,3}(v)$ = 'wahr' :⇔ **FS**((*KNOTEN*, *Navigationsleiste#*), (*NAVIGATIONSLEISTE*, *Navigationsleiste#*), *v*).

KNOTENTYP:

REL({*Knotentyp#, Knotentyp_Name, Beschreibung*} | {$\mathbf{sib}^1_{knt,1}$, $\mathbf{sib}^1_{knt,2}$}), mit

Knotentyp#:	ATT(long |∅)	Primärschlüssel
Knotentyp_Name:	ATT(char(50) |∅)	Name des Knotentyps
Beschreibung:	ATT(char(2000) |∅)	Beschreibung / Abgrenzung

$\mathbf{sib}^1_{knt,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Knotentyp#*, $\underline{x}$),
$\mathbf{sib}^1_{knt,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Knotentyp_Name, Beschreibung*}, $\underline{x}$).

KNOTENZUORDNUNG:

REL({*Knotenzuordnung#, Objekt#, Knoten#, Objektname*} | {$\mathbf{sib}^1_{kzu,1}$, $\mathbf{sib}^1_{kzu,2}$, $\mathbf{sib}^2_{kzu,1}$, $\mathbf{sib}^2_{kzu,2}$}), mit

Knotenzuordnung#:	ATT(long |∅)	Primärschlüssel
Knoten#:	ATT(long |∅)	Fremdschlüssel
Objekt#:	ATT(long |∅)	Fremdschlüssel
Objektname:	ATT(char(50) |∅)	Name des referenzierten Objekttypnamens

$\mathbf{sib}^1_{kzu,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Knotenzuordnung#*, $\underline{x}$),
$\mathbf{sib}^1_{kzu,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Knoten#, Objekt#, Objektname*}, $\underline{x}$),
$\mathbf{sib}^2_{kzu,1}(v)$ = 'wahr' :⇔ **FS**((*KNOTENZUORDNUNG*, *Knoten#*), (*KNOTEN*, *Knoten#*), *v*),
$\mathbf{sib}^2_{kzu,2}(v)$ = 'wahr' :⇔ **FS**((*KNOTENZUORDNUNG*, *Objekt#*), (*OBJEKTNAME*, *Objekt#*), *v*).

ANKER:

REL({*Anker#, Knoten#, Element#, Erstelldatum, Elementtyp, Ankerposition1, Ankerposition2, Zielanker, REL, REV, TITLE*} | {$\mathbf{sib}^0_{ank,1}$, $\mathbf{sib}^1_{ank,1}$, $\mathbf{sib}^1_{ank,2}$, $\mathbf{sib}^2_{ank,1}$, $\mathbf{sib}^2_{ank,2}$}), mit

Anker#:	ATT(long |∅)	Primärschlüssel
Knoten#:	ATT(long |∅)	Fremdschlüssel
Element#:	ATT(long |∅)	Fremdschlüssel
Elementty **:** $\mathbf{sib}^0_{ank,1}(w)$ = 'wahr' :⇔	ATT(char(1) |∅) $w \in$ {'i', 'n'}	InfoElement ('i') oder Navigationselement ('n')?
Erstelldatum:	ATT(date |∅)	Erstelldatum
Ankerposition1:	ATT(integer |∅)	Position des Ankerbeginns
Ankerposition2:	ATT(integer |∅)	Position des Ankerendes

Zielanker:	ATT(char(50) |∅)	Zielanker in lokalem Dokument. Optionales Attribut des Tags <A>
REL:	ATT(char(50) |∅)	optionales Attribut des Tags <A>
REV:	ATT(char(50) |∅)	optionales Attribut des Tags <A>
TITLE:	ATT(char(50) |∅)	optionales Attribut des Tags <A>

$\mathbf{sib}^1_{ank,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Anker#*, $\underline{x}$),
$\mathbf{sib}^1_{ank,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Knoten#, Element#, Erstelldatum, Elementtyp, Ankerposition1, Ankerposition2*}, $\underline{x}$),
$\mathbf{sib}^2_{ank,1}(v)$ = 'wahr' :⇔ **FS**((*ANKER*, *Knoten#*), (*KNOTEN*, *Knoten#*), v),
$\mathbf{sib}^2_{ank,2}(v)$ = 'wahr' :⇔ **FS**((*ANKER*, *Element#*), (*ELEMENT*, *Element#*), v).

VERWEIS:

REL({*Verweis#, Verweistyp#, Anker#, Knoten#, Erstelldatum, URL*} | {$\mathbf{sib}^1_{ver,1}$, $\mathbf{sib}^1_{ver,2}$, $\mathbf{sib}^2_{ver,1}$, $\mathbf{sib}^2_{ver,2}$, $\mathbf{sib}^2_{ver,3}$}), mit

Verweis#:	ATT(long |∅)	Primärschlüssel
Verweistyp#:	ATT(long |∅)	Fremdschlüssel
Anker#:	ATT(long |∅)	Fremdschlüssel
Knoten#:	ATT(long |∅)	Fremdschlüssel (Ziel des Verweises)
Erstelldatum:	ATT(date |∅)	Erstelldatum
URL:	ATT(char(80) |∅)	Uniform Ressource Locator

$\mathbf{sib}^1_{ver,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Verweis#*, $\underline{x}$),
$\mathbf{sib}^1_{ver,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Verweistyp#, Anker#, Knoten#, Erstelldatum, URL* }, $\underline{x}$),
$\mathbf{sib}^2_{ver,1}(v)$ = 'wahr' :⇔ **FS**((*VERWEIS*, *Verweistyp#*), (*VERWEISTYP*, *Verweistyp#*), v),
$\mathbf{sib}^2_{ver,2}(v)$ = 'wahr' :⇔ **FS**((*VERWEIS*, *Anker#*), (*ANKER*, *Anker#*), v),
$\mathbf{sib}^2_{ver,3}(v)$ = 'wahr' :⇔ **FS**((*VERWEIS*, *Knoten#*), (*KNOTEN*, *Knoten#*), v).

VERWEISTYP:

REL({*Verweistyp#, Verweistyp_Name, Beschreibung*} | {$\mathbf{sib}^1_{vert,1}$, $\mathbf{sib}^1_{vert,2}$}), mit

Verweistyp#:	ATT(long |∅)	Primärschlüssel
Verweistyp_Name:	ATT(char(50) |∅)	Name des Verweistyps
Beschreibung:	ATT(char(2000) |∅)	Beschreibung / Abgrenzung

$\mathbf{sib}^1_{vert,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Verweistyp#*, $\underline{x}$),
$\mathbf{sib}^1_{vert,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Verweistyp_Name, Beschreibung*}, $\underline{x}$).

KAPITEL:

REL({*Kapitel#, UebergeordKapitel#, Kapitelposition, Kapitel_Name*} | {$\mathbf{sib}^1_{kap,1}$, $\mathbf{sib}^1_{kap,2}$, $\mathbf{sib}^2_{kap,1}$}), mit

Kapitel#:	ATT(long |∅)	Primärschlüssel
UebergeordKapitel#:	ATT(long |∅)	Fremdschlüssel
Kapitelposition:	ATT(integer |∅)	Reihenfolge im übergeordneten Kapitel
Kapitel_Name:	ATT(char(50) |∅)	Kapitelname

$\mathbf{sib}^1_{kap,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Kapitel#*, $\underline{x}$),
$\mathbf{sib}^1_{kap,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Kapitel_Name*}, $\underline{x}$),
$\mathbf{sib}^2_{kap,1}(v)$ = 'wahr' :⇔ **FS**((*KAPITEL*, *UebergeordKapitel#*), (*KAPITEL, Kapitel#*), *v*).

NAVIGATIONSLEISTE:

REL({*Navigationsleiste#, Ausrichtung, Ausrichtungsebene*} | {$\mathbf{sib}^0_{nale,1}$, $\mathbf{sib}^0_{nale,2}$, $\mathbf{sib}^1_{nale,1}$, $\mathbf{sib}^1_{nale,2}$}), mit

Navigationsleiste#:	ATT(long |∅)	Primärschlüssel
Ausrichtung:	ATT(char(1) |∅)	links ('l'), rechts ('r') oder zentriert ('c')
$\mathbf{sib}^0_{nale,1}(w)$ = 'wahr' :⇔	$w \in$ {'l', 'r', 'c'}	
Ausrichtungsebene:	ATT(char(1) |∅)	horizontal ('h')oder vertikal ('v')
$\mathbf{sib}^0_{nale,2}(w)$ = 'wahr' :⇔	$w \in$ {'h', 'v'}	

$\mathbf{sib}^1_{nale,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Navigationsleiste#*, $\underline{x}$),
$\mathbf{sib}^1_{nale,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Ausrichtung, Ausrichtungsebene*}, $\underline{x}$).

NAVILEISTE_NAVIELEMENT:

REL({*NaviLeiste_NaviElement#, Navigationsleiste#, Navigationselement#, Reihenfolgeposition*} | {$\mathbf{sib}^1_{nana,1}$, $\mathbf{sib}^1_{nana,2}$, $\mathbf{sib}^2_{nana,1}$, $\mathbf{sib}^2_{nana,2}$}), mit

NaviLeiste_NaviElement#:	ATT(long |∅)	Primärschlüssel
Navigationsleiste#:	ATT(long |∅)	Fremdschlüssel
Navigationselement#:	ATT(long |∅)	Fremdschlüssel
Reihenfolgeposition:	ATT(integer |∅)	Position eines Elementes in der Leiste

$\mathbf{sib}^1_{nana,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*NaviLeiste_NaviElement#*, $\underline{x}$),
$\mathbf{sib}^1_{nana,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Navigationsleiste#, Navigationselement#, Reihenfolgeposition*}, $\underline{x}$),
$\mathbf{sib}^2_{nana,1}(v)$ = 'wahr' :⇔ **FS**((*NAVILEISTE_NAVIELEMENT*, *Navigationsleiste#*), (*NAVIGATIONSLEISTE*, *Navigationsleiste#*), *v*),
$\mathbf{sib}^2_{nana,2}(v)$ = 'wahr' :⇔ **FS**((*NAVILEISTE_NAVIELEMENT*, *Navigationselement#*), *NAVIGATIONSELEMENT*, *Navigationselement#*), *v*).

NAVIGATIONSELEMENT:

REL({*Navigationselement#, Beschriftung, Icon*} | {**sib**$^1_{nael,1}$, **sib**$^1_{nael,2}$}), mit

Navigationselement#:	ATT(long |∅)	Primärschlüssel
Beschriftung:	ATT(char(10) |∅)	Beschriftung des Icons
Icon:	ATT(char(50) |∅)	Pfad zu Icon

sib$^1_{nael,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Navigationselement#*, $\underline{x}$),
sib$^1_{nael,2}(\underline{x})$ = 'wahr' :⇔ **nf1**({*Beschriftung, Icon*}, $\underline{x}$).

INFOELEMENT:

REL({*InfoElement#, Erstelldatum, InfoElement_Name, Pfad, Elementart, Format, Kurzbeschreibung*} | {**sib**$^1_{inel,1}$, **sib**$^1_{inel,2}$}), mit

InfoElement#:	ATT(long |∅)	Primärschlüssel
Erstelldatum:	ATT(date |∅)	Datum der Aufnahme in DB
InfoElement_Name:	ATT(char(30) |∅)	Evtl. Name des Elements
Pfad:	ATT(char(50) |∅)	Zugriffspfad zu InfoElement
Elementart:	ATT(char(15) |∅)	Text, Bild, Video, Sound...
Format:	ATT(char(15) |∅)	Speicherformat
Kurzbeschreibung:	ATT(char(2000) |∅)	Textuelle Beschreibung des Elements

sib$^1_{inel,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*InfoElement#*, $\underline{x}$),
sib$^1_{inel,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Erstelldatum, Pfad, Elementart, Format*}, $\underline{x}$).

KNOTEN_INFOELEMENT:

REL({*Knoten_InfoElement#, Knoten#, InfoElement#*} | {**sib**$^1_{knie,1}$, **sib**$^1_{knie,2}$, **sib**$^2_{knie,1}$, **sib**$^2_{knie,2}$}), mit

Knoten_InfoElement#:	ATT(long |∅)	Primärschlüssel
Knoten#:	ATT(long |∅)	Fremdschlüssel
InfoElement#:	ATT(long |∅)	Fremdschlüssel

sib$^1_{knie,1}(\underline{x})$ = 'wahr' :⇔ **PS**(*Knoten_InfoElement#*, $\underline{x}$),
sib$^1_{knie,2}(\underline{x})$ = 'wahr' :⇔ **nf**({*Knoten#, InfoElement#*}, $\underline{x}$),
sib$^2_{knie,1}(v)$ = 'wahr' :⇔ **FS**((*KNOTEN_INFOELEMENT*, *Knoten#*), (*KNOTEN, Knoten#*), *v*),
sib$^2_{knie,2}(v)$ = 'wahr' :⇔ **FS**((*KNOTEN_INFOELEMENT*, *InfoElement#*), (*INFOELEMENT*, *InfoElement#*), *v*).

WBT_SYSTEM_KNOTEN:

	REL({*WBT_System_Knoten#, WBT_System#, Knoten#*} | {**sib**$^1_{wsk,1}$, **sib**$^1_{wsk,2}$, **sib**$^2_{wsk,1}$, **sib**$^2_{wsk,2}$}), mit	
WBT_System_Nutzer#:	ATT(long | ∅)	Primärschlüssel
WBT_System#:	ATT(long | ∅)	Fremdschlüssel
Knoten#:	ATT(long | ∅)	Fremdschlüssel

sib$^1_{wsk,1}$($\underline{x}$) = ‘wahr’ :⇔ **PS**(*WBT_System_Knoten#*, $\underline{x}$),
sib$^1_{wsk,2}$($\underline{x}$) = ‘wahr’ :⇔ **nf**({*WBT_System#, Knoten#*}, $\underline{x}$),
sib$^2_{wsk,1}$(v) = ‘wahr’ :⇔ **FS**((*WBT_SYSTEM_KNOTEN*, *WBT_System#*), (*WBT_SYSTEM, WBT_System#*), v),
sib$^2_{wsk,2}$(v) = ‘wahr’ :⇔ **FS**((*WBT_SYSTEM_KNOTEN, Knoten#*), (*KNOTEN, Knoten#*), v).

III Abkürzungsverzeichnis

ÄAppO	Approbationsordnung für Ärzte
ANSI	American National Standards Institute
ATI	Aptitude-Treatment-Interactions
CAD	Computer-Aided Design
CAMPUS	Computerunterstützte Aus- und Weiterbildung in der Medizin durch plattformunabhängige Software
CASE	Computer-Aided Software Engineering
CBT	Computer-Based Training
CGI	Common Gateway Interface
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
DBMS	Datenbankmanagementsystem
DDL	Data Definition Language
DML	Data Manipulation Language
DOMINIE	Domain Independent Instructional Environment
EBNF	Eweiterte Backus-Naur-Form
ER	Entity-Relationship
ERM	Entity-Relationship-Modellierung
ERC	Entity-Relationship Calculus
et al.	et altera
FS	Fremdschlüssel
GMDS	Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie
GUI	Graphical User Interface
HAM	Hypertext Abstract Machine
HTML	Hypertext Markup Language
HTTP	Hypertext Transmission Protocol
ICD	International Classification of Diseases
IDL	Interface Definition Language
IRC	Internet Relay Chat
ITS	Intelligentes Tutorielles System
JDK	Java Development Kit
MC	Multiple-Choice
MVC	Model-View-Controller
nf	nicht fehlend
NLM	National Library of Medicine
ODMG	Objekt Database Management Group
OOA	Objektorientierte Analyse
OOD	Objektorientiertes Design
ORB	Object Request Broker
PS	Primärschlüssel
RMI	Remote Method Invocation
RPC	Remote Procedure Call
sib	semantische Integritätsbedingung
SPARC	Standards Planing and Requirements Committee
SQL	Structured Query Language

s.u.	siehe unten
TCP	Transmission Control Protocol
u.a.	unter anderem
UML	Unified Modeling Language
UMLS	Unified Medical Language System
URL	Uniform Ressource Locator
vgl.	vergleiche
VIROR	Virtuelle Hochschule Oberrhein
WBT	Web-Based Training
WWW	World Wide Web
z.B.	zum Beispiel

IV Abbildungsverzeichnis

V Tabellenverzeichnis

VI Index

Lebenslauf

Angaben zur Person:

Name	Martin Ernst Haag
Straße	Finkenweg 25
Wohnort	69214 Eppelheim
Geburtstag	04. Dezember 1969
Geburtsort	74585 Rot am See - Brettheim
Staatsangehörigkeit	deutsch
Familienstand	ledig
Religion	evangelisch

Schulausbildung:

Grundschule	1976 - 1980 Grundschule Brettheim
Gymnasium	1980 - 1989 Gymnasium Gerabronn
Datum des Abschlusses	10. Mai 1989
Art des Abschlusses	Allgemeine Hochschulreife

Wehrdienst: Juni 1989 - August 1990

Studium:

Studienfach und Studiendauer	Medizinische Informatik an der Universität Heidelberg / Fachhochschule Heilbronn vom WS 1990/91 - SS 1995
Diplomprüfung (Diplom-Informatiker der Medizin)	28. Juni 1995

Berufstätigkeit:

Wissenschaftliche Hilfskraft	von November 1993 bis Juni 1995 im *Labor Computerunterstützte Ausbildung in der Medizin* der Universität Heidelberg
Wissenschaftlicher Angestellter	seit Oktober 1995 im *Labor Computerunterstützte Ausbildung in der Medizin* und am Institut für Medizinische Biometrie und Informatik, Abteilung Medizinische Informatik der Universität Heidelberg

Heidelberg, September 1998

Danke!

An dieser Stelle möchte ich mich bei allen Personen bedanken, die zum erfolgreichen Abschluß meiner Promotion beigetragen haben.

Prof. Dr. R. Haux danke ich sehr herzlich für die Übernahme der Betreuung dieser Promotion.

Den Leitern des Labors *Computerunterstützte Ausbildung in der Medizin*, Prof. Dr. Dr. h.c. H.-G. Sonntag und Prof. F.J. Leven danke ich dafür, daß ich die gesamte Infrastruktur des Labors nutzen konnte.
Prof. Leven verdient darüber hinaus meinen besonderen Dank für die intensive inhaltliche Mitbetreuung dieser Arbeit. Ich verdanke ihm viele gute Anregungen.

Die Herren Dipl.-Inform. Med. G. Pelzer, Dipl.-Inform. Med. J. Riedel und Dipl.-Inform. Med. R. Singer haben im Rahmen des CAMPUS-Projektes ihre Diplomarbeiten mit Erfolg angefertigt und mir dadurch viel Arbeit abgenommen. Dafür und für die gute Zusammenarbeit ein herzliches Dankeschön!

Den Herren PD Dr. B. Tönshoff und AiP T. Ullinski von der Universitätskinderklinik sowie Herrn Prof. Dr. H.-K. Geiss und der AiP Frau E. Müller vom Hygiene-Institut danke ich für die fachliche Unterstützung und die Bereitstellung von Lehr-/Lernfällen.

Die Herren Dipl.-Inform. Med. B. Chevreux und Dipl.-Inform. Med. T. Pfisterer haben sich freundlicherweise dazu bereiterklärt, die vorliegende Arbeit Korrektur zu lesen. Für ihre überaus gründliche Arbeit gebührt ihnen mein besonderer Dank!

Herrn Dipl.-Inform. Med. W. Alle danke ich für die jahrelange gute Zusammenarbeit bei Aufbau und Betrieb des Labors *Computerunterstützte Ausbildung in der Medizin*.

Allen Freunden, Bekannten und meinen Eltern danke ich für die „mentale“ Unterstützung während der vergangenen drei Jahre.