Effiziente Strategien und Werkzeuge zur Generierung und Verwaltung von e-Learning-Systemen

Dissertation

der Fakultät für Informatik
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von

Dipl. - Inform. **Mechthild Uesbeck**
aus Tübingen

Tübingen
2001

Tag der mündlichen Qualifikation	14. Februar 2001
Dekan	Prof. Dr. Andreas Zell
1. Berichterstatter	Prof. Dr. Wolfgang Straßer
2. Berichterstatter	Prof. Dr. Wolfgang Küchlin

Zusammenfassung

Aus Sicht der digitalen Informations- und Kommunikationstechnologie beginnt das neue Jahrtausend mit Innovationen, die noch vor wenigen Jahren unvorstellbar schienen und die Gesellschaft nachhaltig beeinflussen werden. Die Frequenz üblicher PC-Prozessoren übersteigt die Gigahertz-Marke, Personal Information Manager (PIM) im Scheckkartenformat verdrängen Taschenkalender, Mobilfunk macht den rechnergestützten Zugriff auf Informationen nahezu unabhängig von Ort und Zeit. Zu keinem Zeitpunkt hat sich die Informationstechnologie so rasant weiterentwickelt wie in den letzten Jahren.

Auch die Wissenslandschaft bleibt von diesem Trend nicht unberührt, da ein hoher Wissensstand einerseits die Grundlage jeglichen Fortschritts ist, und Wissen andererseits als Folge dieser technischen Innovationen mehr und mehr die Rolle eines sich schnell wandelnden, kostbaren Wirtschaftsguts einnimmt. Damit wird es in allen Bildungsbereichen zunehmend wichtiger, sich auf der Basis dieser neuen Informationsstrukturen Wissen permanent und bedarfsgerecht anzueignen: das „lebenslange Lernen“.

Die Verfügbarkeit entsprechender technologischer Möglichkeiten allein verändert hier allerdings noch nichts. Deren sinnvoller Einsatz beispielsweise im Rahmen von Computergestützten Lernangeboten setzt vor allem erhebliche Anstrengungen in Forschung und Entwicklung voraus und ist an die Erfüllung komplizierter Randbedingungen geknüpft. So existieren anspruchsvolle Anforderungen bezüglich der Erhöhung der Lerneffizienz, einer besseren Nutzung der eingesetzten finanziellen und personellen Ressourcen und vor allem im Hinblick auf eine verstärkte Individualisierung des Lernens.

Diese Dissertation beschreibt die Konzeption und Implementierung des sog. e-Learning-System-Baukastens (**eLS-Baukasten**), der effiziente Strategien und Werkzeuge zur automatisierten Generierung Webbasierter Lernprogramme (e-Learning-Systeme) für unterschiedliche Anwendungsbereiche bereitstellt. Die generierten Systeme zeichnen sich durch eine besonders modulare Architektur aus und basieren auf einem mächtigen Wissens- und Benutzermodell, das die Abbildung von Lernstrategien und -interaktionsformen erlaubt, welche aktuellen, kognitionspsychologischen Gesichtspunkten entsprechen.

Der eLS-Baukasten gewährt im Hinblick auf Generierung und Wartung von e-Learning-Systemen weitestgehende Transparenz aller technischen Vorgänge. Jedes generierte System kann dabei durch die Kombination entsprechender Werkzeuge und Verfahren Defizite und Schwächen existierender e-Learning-Systeme von Anfang an vermeiden. Weiterer Bestandteil der mit dem Baukasten generierbaren Systeme ist eine Komponente zur verteilten, plattformunabhängigen und vollständig graphisch unterstützten Erstellung und Bearbeitung der multimedialen Inhalte. Für diese Komponente wurde eine schmale, internetbasierte Schnittstelle zur Datenbank-Komponente geschaffen.

Auf Präsentationsseite wurden Verfahren entwickelt, die innerhalb der generierten Systeme eine individuelle Benutzerunterstützung für jeden Lernenden zur Verfügung stellen. Ein aus der Literatur in dieser Form bisher nicht bekannter Ansatz erweitert dabei diese Präsentationsmöglichkeiten um eine modulare Komponente zur inhaltsbasierten Suche in den komplexen, multimedialen Wissensinhalten. So werden bewährte Eigenschaften heutiger Datenbanksysteme mit den Vorteilen webbasierter Informationssysteme und mit aktuellen Verfahren des inhaltsbasierten Information Retrievals zu einem mächtigen, für unterschiedlichste Anwendungsbereiche geeigneten Gesamtsystem verknüpft.

Der im Rahmen dieser Dissertation realisierte Ansatz stellt einen wesentlichen Beitrag zur Entschärfung derzeitiger Probleme im Bereich des e-Learnings dar, der in der Möglichkeit zur Nutzung neuer Ansätze der Wissensvermittlung mündet und dabei finanzielle und personelle Ressourcen optimal zu nutzen weiß.

Danksagung

An dieser Stelle möchte ich mich bei all denen bedanken, die das Entstehen der vorliegenden Arbeit gefördert haben:

- Herrn Prof. Dr.-Ing. Dr.-Ing. E.h. W. Straßer danke ich für die Übernahme des Erstgutachtens, die Betreuung der Arbeit und den mir großzügig gewährten Freiraum bei deren Durchführung.
- Für die Übernahme des Zweitgutachtens gilt mein Dank Herrn Prof. Dr. sc. techn. W. Küchlin.
- Herrn Dr. med. M. Skalej von der Radiologischen Universitätsklinik Tübingen, Abteilung für Neuroradiologie (Leitung: Prof. Dr. med. K. Voigt), danke ich für die Möglichkeit zur praktischen Durchführung der Arbeit und zahlreiche Anregungen.
- Allen meinen Kolleginnen und Kollegen in der Abteilung für Neuroradiologie der Radiologischen Universitätsklinik und am Lehrstuhl für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts für Informatik danke ich für die gute Zusammenarbeit in den vergangenen Jahren, ihre stete Hilfsbereitschaft und die angenehme Arbeitsatmosphäre. Mein besonderer Dank gilt den Kolleginnen und Kollegen meiner Arbeitsgruppe, die in der Endphase dieser Arbeit viele meiner Aufgaben übernommen haben. Das sehr gute und nette Arbeitsklima in der Gruppe während all dieser Jahre hat ganz sicher sehr zum Gelingen dieser Arbeit beigetragen.
- Mein Dank gilt außerdem den Studenten, die im Rahmen von Diplom- bzw. Studienarbeiten an der Realisierung des Gesamtsystems mitgewirkt haben.
- Meiner Familie und meinen Freunden, die das Entstehen der vorliegenden Arbeit in unterschiedlicher Weise begleitet haben, danke ich für ihre große Unterstützung. Besonderen Dank schulde ich gleichermaßen Michael Hartmeier, Karin Sobottka und Volker Fischer für die Durchsicht des Manuskripts und hilfreiche Anmerkungen. Mein Dank gilt außerdem Sieglinde Kull und Till Karsten Hauser für die Unterstützung bei Korrektur und Layoutgestaltung dieser Arbeit.
- Schließlich bedanke ich mich beim Land Baden-Württemberg, dem Interdisziplinären Zentrum für Klinische Forschung (IZKF) in Tübingen und der Deutschen Telekom AG für die Förderung der Arbeit im Rahmen des Landesforschungsschwerpunktes „NeuroAssistant“ und der Gemeinschaftsinitiative „Multimedia-gestützte Studiengänge an Hochschulen“.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

1.1 Motivation: Von der Zukunft des Lernens

Die digitale Informations- und Kommunikationstechnologie (IuK) hat besonders auch auf das weltweit verfügbare Wissen einen entscheidenden Einfluß. Experten gehen davon aus, dass als Folge rasch wechselnder Produkttechnologien und Innovationen in diesem Bereich die Halbwertszeit des Wissens inzwischen nur noch drei Jahren beträgt [Ste00]. Folge dieser Entwicklung ist die Notwendigkeit der permanenten Wissensaufnahme und -aktualisierung innerhalb aller Bildungsbereiche und Institutionen: das „lebenslange Lernen“ [Bmb98, Bmb99, Bmb00].

Seit einiger Zeit befassen sich daher vor allem technische und kognitionswissenschaftliche Einrichtungen mit der Frage, wie unter Einsatz heutiger und zukünftiger Technologien die Wissensvermittlung qualitativ und quantitativ verbessert und an neue Einsatzmöglichkeiten angepaßt werden kann. Im Umfeld von Hochschulen erfolgt dies besonders unter dem Druck aktueller, kontrovers geführter Diskussionen um teilweise hohe Studentenzahlen und dem damit einhergehenden Qualitätsverlust der praktischen Ausbildung. Der Trend hin zu einer Internationalisierung des Bildungsmarktes erhöht die Notwendigkeit, unter Einsatz neuer Informations- und Kommunikationstechnologien innovative Lehrmethoden zu entwickeln, um die führende Stellung deutscher Aus- und Weiterbildungseinrichtungen im globalen Wettbewerb zu sichern. Genauso wächst auch in Wirtschaftsunternehmen die Erkenntnis, dass die Generierung von Wissen, sowie dessen Verteilung und Bereitstellung der ausschlaggebende Wettbewerbsfaktor der kommenden Jahre sein wird. Digitale Wissenspools und computergestützte betriebliche Qualifikationsmaßnahmen, vernetzte Unternehmensbereiche und die elektronische Zusammenarbeit innerhalb von Mitarbeiterteams können durchgehende Wertschöpfungsketten bilden und eine hervorragende Grundlage für unternehmenseinheitliche Lern-Infrastrukturen schaffen, die in übergreifende Weiterbildungsmodelle integriert werden könnten. Auch ist Wissen ein auf den internationalen Märkten inzwischen hoch gehandeltes Gut. Die Möglichkeiten des virtuellen Lernens tragen somit entscheidend dazu bei, sich als Unternehmen auf diesem Markt zu behaupten.

Neuesten Prognosen zufolge werden der IT-basierten Aus- und Weiterbildung besonders gute Wachstumsraten vorhergesagt[1]. Die aktuelle Herausforderung innerhalb des gesamten Bildungssektors besteht darin, neue Lehr- und Lernformen zu etablieren, die unter Einsatz derzeitiger und zukünftiger Informations- und Kommunikationstechnologien den oben beschriebenen Entwicklungen Rechnung tragen. Sie sollen zu eigenverantwortlichen, selbstgesteuerten und gleichzeitig kooperativen Lernprozessen beitragen und dadurch den Prozeß des lebenslangen Lernens effektiv unterstützen.

[1] Nach [Bru00] können ein zukünftiger Anteil am Weiterbildungsvolumen von 15 - 50 % und Kosteneinsparungen zwischen 20 und 25 % erwartet werden. In den USA wird nach [Ste00] der aktuelle Markt für „Virtual Higher Education“ mit über 10 Milliarden Dollar pro Jahr geschätzt, das „Corporate E-Learning“, d.h. die virtuelle Aus- und Weiterbildung im Berufsalltag, mit 11 Milliarden.

1.2 e-Learning-Systeme

e-Learning-Systeme, oftmals auch als **WBT**-Systeme (WBT- Web Based Training) bezeichnet, zählen zu den neuen Lehr- und Lernformen, die unter Einsatz von IuK-Technologien und nach verschiedenen kognitionspsychologischen Methoden Lerninhalte vermitteln. Sie lassen sich in nahezu allen Fach- und Anwendungsbereichen einsetzen und sind bezeichnenderweise im Internet allgemein verfügbar.

Unterschieden werden zwei Systemvarianten: Zum einen existieren **Informationssysteme**, die zur Vermittlung **systematischen Wissens** eingesetzt werden. Derartige Systeme umfassen i.R. multimediale Lerninhalte, die durch Hyperlinks miteinander vernetzt sind. Als Medientypen finden vornehmlich Texte und Bilder, Audio- und Videosequenzen Verwendung [Klu00]. Die zweite Systemvariante sind **Trainingssysteme**. Sie verwenden ebenfalls die genannten Medientypen, der Schwerpunkt liegt hier jedoch auf der Vermittlung von **problemorientierten Inhalten**. Eingesetzt werden dazu vor allem interaktive, graphische Komponenten, die das *aktive* Lernen, das für das Verständnis eines Sachverhaltes unabdingbar ist, unterstützen sollen [Han98]. Vergleicht man e-Learning-Systeme mit traditionellen Lehrmethoden wie Büchern, Vorlesungen, Seminaren oder Praktika, so finden sich eine Reihe von Vorzügen.

- e-Learning-Systeme sind zunehmend zeit- und ortsunabhängig verfügbar, so dass die Lernorganisation flexibilisiert und stärker am persönlichen Interesse und Lerntempo ausgerichtet werden kann.
- Verglichen mit schriftlichem Material bieten die Systeme einen höheren Grad der Interaktivität.
- Im Gegensatz zum gedruckten Lehrbuch oder einer Vorlesung sind die Inhalte eines e-Learning-Systems einfach und zentral zu aktualisieren.
- Sie bieten neuartige didaktische Möglichkeiten, da netzgestützt kooperative und kommunikative Lerntechniken bzw. Anwendungsszenarien bereitgestellt werden können.

Damit erscheinen e-Learning-Systeme als attraktive Ergänzung zu traditionellen Lehrmethoden. Angesichts der oben geschilderten Notwendigkeit des lebenslangen Lernens kann eine zukünftig schnell wachsende Akzeptanz solcher Systeme erwartet werden.

1.3 Lernparadigmen

Der sinnvolle Einsatz von e-Learning-Systemen hängt von der Quantität und der Qualität der verwendeten Technik und der präsentierten Inhalten ab; entscheidend jedoch ist, welcher Lernprozeß angesprochen bzw. simuliert werden soll.

„Lernen“ ist nach [Gol98] die zusammenfassende Bezeichnung für alle Prozesse zur Aneignung verschiedener Wissensarten. Lernen umfaßt die Befähigung eines Individuums, sein Verhalten zu ändern, sich Kenntnisse und Werthaltungen anzueignen und Einsichten und Fertigkeiten zu erwerben. Von jeher beschäftigten sich verschiedene Fachdisziplinen mit der Frage, wie Lernen optimal unterstützt werden kann. In der Lern- und Kognitionspsychologie haben sich hierzu vier grundlegende Ansätze der Wissensvermittlung etabliert [Iss97]:

- **Behaviorismus**: Lernen durch Verstärkung,
- **Kognitivismus**: Lernen durch Einsicht,
- **Konstruktivismus**: Lernen durch Erleben und Interpretieren,
- **Situierte Kognition**: Lernen durch der Realität nachempfundener Situationen.

Im Paradigma des **Behaviorismus** bedeutet Lernen ein Trainieren mit dem Ziel einer Verhaltensänderung im Sinne eines „richtigen“ Verhaltens. Das Verstehen und Anwenden komplexer Zusammenhänge bleibt dabei nahezu unberücksichtigt. Im Hinblick auf computerbasierte Lernsysteme haben behavioristisch geprägte Programme den Charakter einer „starren Paukmaschine“ [Thi97]. Der Stoff ist übersichtlich strukturiert und die Lernziele eindeutig festgelegt. Positive bzw. negative Verstärkung wird durch direktes Feedback des Systems auf die Aktionen des Nutzers erreicht. Typische Beispiele behavioristisch geprägter Lernprogramme sind Vokabel-Trainer oder einfache Mathematiklernprogramme.

Der **Kognitivismus** versteht sich als Gegenbewegung zum Behaviorismus. Kernpunkt ist, Erkenntnisse über die kognitiven Prozesse der menschlichen Informationsverarbeitung in die Lehr-/Lerntheorien einfließen zu lassen. Eine wichtige Rolle spielen dabei Problemlösungsstrategien, Entscheidungsprozesse und das Nachvollziehen komplexer Zusammenhänge, weniger wichtig ist das Trainieren richtiger Antworten und Handlungen. Typische Beispiele kognitivistischer Lernprogramme sind Systeme zum Erlernen spezieller Fähigkeiten, beispielsweise der Bedienung eines technischen Gerätes oder Software-Programms.

Entscheidende Impulse für die Gestaltung neuer Bildungsmedien liefert seit Ende der 80er-Jahre der **Konstruktivismus**, der Erkenntnisse verschiedener wissenschaftlicher Disziplinen wie Neurobiologie, Kognitionspsychologie, Linguistik und Informatik verbindet. Hier besteht Lernen aus einem aktiven, selbstgesteuerten Prozess, in dem Lernende bewusst mit komplexen Situationen konfrontiert werden, in denen die zu lösenden Probleme eigenständig als solche erkannt werden müssen [Man95]. Konstruktivistische Lernprogramme zeichnen sich dadurch aus, dass sie den Lernenden wenig anleiten und ihm eher knappes Lernmaterial präsentieren, sondern eine Lernumgebung zur Verfügung stellen, die es ermöglicht, selbst auszuprobieren, sich mit Themen zu beschäftigen, auch falsche Handlungsalternativen zu wählen und Inhalte und Zusammenhänge eigenständig zu entdecken.

Schließlich entwickelte sich die **situierte Kognition** als Spezialisierung des Konstruktivismus. Sie geht davon aus, dass Wissen nicht einfach von einer Person zur anderen weitergereicht werden kann, sondern beim Lernenden dazu ein aktiver Konstruktionsprozeß erforderlich ist. Die zu lernenden Inhalte können dabei nicht vom Vorgang des Lernens und ebenso wenig von der Situation getrennt werden, in der gelernt wird.

Für die Vermittlung deklarativen Wissens sind die behavioristischen Lernmodelle gut geeignet. Nach diesem Modell konzipierte Lernsysteme sind die bereits in Abschnitt 1.2 erwähnten *Informationssysteme zur Vermittlung systematischen Wissens*. Zur Anwendung kommt hier das sog. **Instruktionsparadigma**, bei dem der Lernende Inhalte rezipiert, die zuvor von Experten in didaktisch sinnvolle, aufeinander aufbauende Lerneinheiten unterteilt und entsprechend der vorhandenen Vorkenntnisse aufbereitet wurden.

Zur Vermittlung prozeduralen, adaptiven und strategischen Wissens müssen dagegen andere Strategien gewählt werden. Hier bieten sich die erwähnten *Trainingssysteme zur Vermittlung problemorientierten Wissens* an, die sowohl kognitivistischen wie auch konstruktivistischen Charakter haben sollten. Derartige Systeme sind nach dem **Problemlösungsparadigma** konzipiert, bei dem das Lernen als aktiver, dynamischer Prozeß verstanden wird. Der Lernprozeß besteht aus der eigenverantwortlichen Zusammenstellung von Inhalten oder aus bereits aufbereiteten Inhalten, die zur eigenständigen Wissenskonstruktion herausfordern. In der Regel stellt ein konkretes Problem die Ausgangssituation dar, zu dem sich die Lernenden nach und nach das nötige Hintergrundwissen aneignen, um dann einen individuellen Problemlösungsprozeß zu durchlaufen. Definiert man den Wissenstransfer von der Lern- in die Anwendungssituation als Ziel von Lernhandlungen, so ist der Ansatz des situierten Lernens optimal für die Gestaltung von Lernprogrammen geeignet.

1.4 Problemfelder und aufgezeigte Lösungsansätze

Trotz der vielversprechenden Möglichkeiten von e-Learning-Systemen werden diese besonders im deutschen Bildungssektor bislang nur punktuell eingesetzt. Als Ursache dafür lassen sich Probleme in unterschiedlichen Bereichen von Gesellschaft und Technik ausmachen.

Gesellschaftliche Problemfelder werden in dieser Arbeit nicht angesprochen, sind jedoch Teil des umfassenden Bildes und sollen hier kurz erwähnt werden. Zu nennen sind hier in erster Linie die noch immer geringe Erfahrung sowohl der Lehrenden als auch der Lernenden im Umgang mit neuen Medien und Informationstechnologien [Bmb99]. Zudem sorgt die oftmals mangelnde Einbindung digitaler Lernsysteme in das jeweilige Aus- bzw. Fortbildungsprogramm für eine nur geringe Akzeptanz. Auch unter finanziellen Gesichtspunkten sind e-Learning-Systeme bislang wenig attraktiv. Typischerweise stehen hohe initiale Produktionskosten einer kurzen Nutzungsdauer und einem kleinen Anwenderkreis gegenüber [Ste00].

Die folgenden Abschnitte beschreiben technische und konzeptionelle Probleme derzeit existierender e-Learning-Systeme, die einem breiteren Einsatz entgegenstehen, und skizzieren Lösungskonzepte, die im Rahmen dieser Arbeit entwickelt wurden.

1.4.1 Generierung von e-Learning-Systemen

Viele derzeit existierende „e-Learning-Systeme“ bieten lediglich eine digitale Aufbereitung vorhandenen Lehrmaterials, besitzen jedoch kein didaktisches Konzept, das die Möglichkeiten der neuen Medien wirklich nutzt. Unbestritten existieren sehr gut konzipierte Beispiele, jedoch erfordert die Realisierung derartiger Systeme oft einen extrem hohen Aufwand auf technischer, didaktischer und inhaltlicher Seite. Zur Vermeidung vor allem des hohen technischen Aufwands werden deshalb eher schnell zu realisierende „Insel“-Lösungen mit begrenzter Funktionalität und nur statischen Inhalten erstellt, bei denen eine Wiederverwendbarkeit des Programmcodes bzw. die inhaltliche Erweiterung der Systeme nur schwer möglich ist.

Abhilfe verspricht hier die Entwicklung eines Baukastensystems, mit dem vollständige, initiale e-Learning-Systeme für möglichst breite Einsatzbereiche generiert werden können. Jedes derart aufgesetzte System umfaßt Komponenten zur automatischen Datenverwaltung und -bearbeitung, sowie spezielle Subsysteme für Administratoren, Anwender und Inhalteerzeuger. Das Konzept stellt eine durchgängige Trennung von Systemlogik und Inhalten sicher und ermöglicht es Experten und Autoren, sich auf die Gestaltung der Lerninhalte zu konzentrieren.

Das im Folgenden als **eLS-Baukasten** bezeichnete Gesamtsystem wird somit ein einziges Mal realisiert und kann als Komplettpaket an Institutionen weitergegeben werden, die mit seiner Hilfe konkrete e-Learning-Systeme aufsetzen können. Durch die jeweils integrierten Inhalte, die dynamische Vernetzbarkeit der multimedialen Informationseinheiten und die Abbildung eigener Lehrstrategien durch die jeweiligen Autoren erhält dabei jedes System seinen individuellen, fachbereichsabhängigen und didaktisch angemessenen Charakter. Der Schwerpunkt dieser Dissertation liegt auf der Konzeption, Umsetzung und Erprobung dieses eLS-Baukastens.

1.4.2 Benutzerunterstützung in e-Learning-Systemen

Für den Lernenden spielt neben der didaktisch angemessenen Präsentation der Inhalte vor allem auch die individuelle Unterstützung während der Interaktionen mit dem komplexen System eine wichtige Rolle. Eine effektive Unterstützung umfaßt beispielsweise die Möglichkeit, jederzeit über den eigenen, aktuellen Standpunkt innerhalb des Systems Bescheid zu wissen, zum zuletzt besuchten Inhalt zurückkehren zu können, bereits bearbeitete Inhalte angezeigt zu bekommen, die persistente Erstellung persönlicher Annotationen zu einzelnen Informationseinheiten oder die Ablage persönlicher Lesezeichen zuzulassen. Der

Großteil derzeit existierender Systeme bietet hierzu nur wenige Möglichkeiten.

Ausgehend von dieser Situation wird in der vorliegenden Arbeit eine Lösung entwickelt, die, basierend auf einem umfassenden Benutzermonitoring, dem Anwender gezielt Methoden und Hilfswerkzeuge zur Arbeit mit dem System zur Verfügung stellt. Die protokollierten Daten können gleichzeitig für rechnergestützte quantitative Evaluierungsvorgänge genutzt werden.

1.4.3 Erzeugung multimedialer Inhalte für e-Learning-Systeme

Ein bisher nur unzureichend behandeltes Problem ist die Erstellung und Bearbeitung multimedialer Inhalte, die innerhalb eines e-Learning-Systems präsentiert werden sollen. Die Konstruktion und Umsetzung des didaktischen Gesamtaufbaus von Inhalten, die Erstellung von Verknüpfungsstrukturen, interaktiven Komponenten und Animationen, sowie die Überarbeitung und Ergänzung der Inhalte stellt für Dozenten und Autoren einen immensen Arbeitsaufwand dar oder ist aufgrund fehlender Medienkompetenz nicht zu realisieren.

Für eine größere Akzeptanz von e-Learning-Systemen auf Seiten der Autoren ist es unverzichtbar, eine intuitiv verständliche, graphische Schnittstelle anzubieten, die es jedem Autor erlaubt, Inhalte selbsttätig zu bearbeiten und einzustellen. Die Nutzung sollte nur wenig technisches Know-How voraussetzen und das plattformunabhängige, direkte Einspielen erzeugter Inhalte in das e-Learning-System via Internet ermöglichen. Ein Autorensystem, das all diesen Anforderungen genügt und zahlreiche Einsatzmöglichkeiten erlaubt, wurde im Rahmen des Projekts MURMEL[2] realisiert. Innerhalb dieser Arbeit wurden alle dazu notwendigen Verfahren zur Inhaltsablage, -bearbeitung und -erfragung implementiert und eine schmale Schnittstelle zwischen clientseitigem Autorensystem und Server realisiert.

1.4.4 Multimediale Informationssuche in e-Learning-Systemen

Ein generelles Defizit von elektronischen Informationssystemen besteht in dem bisher nicht zufriedenstellend gelösten Problem der effektiven inhaltsbasierten Informationssuche in multimedialen Daten. Neben flexiblen textbasierten Suchanfragen fehlen insbesondere Möglichkeiten zur Suche in großen Beständen von Bilddaten. Hierzu stellt diese Dissertation einen neuen Ansatz zur kombinierten Suche über verschiedene Typen multimedialer Daten vor, der in den entwickelten eLS-Baukasten integriert wurde. Der Ansatz beruht auf einer text- und bildbasierten Indizierung aller Bilddaten, die während der Einstellung von Inhalten in einem mit Hilfe des Baukastensystems generierten e-Learning-System transparent durchgeführt wird und stets konsistent gehalten wird.

1.5 Zielsetzung

Die in Abschnitt 1.4 skizzierten Lösungsansätze werden im Rahmen des sog. **eLS-Baukasten** durch folgende Teilschritte realisiert:

1. Die Implementierung eines komponentenbasierten Systems zur Generierung und Wartung von e-Learning-Systemen. Diese „Basis“ umfaßt die Konzeption und Entwicklung:
   - eines didaktisch-methodischen Grundkonzepts zur zielgruppengerechten Aufbereitung und Präsentation multimedialen Lehrmaterials,
   - eines Datenmodells zur Ablage und Verwaltung verschiedener Inhaltstypen unter Einsatz eines geeigneten Datenbanksystems,
   - paralleler serverseitiger Datenbankapplikationen zur Bearbeitung lesender und schreibender Systemanfragen im laufenden Betrieb,

[2] MURMEL - Multimediales Ausbildungssystem für die Medizinische Lehre.

- verschiedener clientseitiger Benutzerschnittstellen: eines Administrationssystems für Aufsetzung und Wartung von e-Learning-Systemen, eines Autorensystems zur Inhaltsbearbeitung und einer Schnittstelle für Lernende, die aus dem kompletten Set zu Laufzeiten dynamisch aufgebauter WWW-Templates besteht.

2. Die Konzeption und Entwicklung von Werkzeugen zur individuellen Benutzerunterstützung.
3. Die Anbindung eines plattformunabhängigen Autorensystems.
4. Die Konzeption und Entwicklung von Werkzeugen zur Unterstützung der multimedialen Informationssuche innerhalb von e-Learning-Systemen.

Insgesamt besteht das Ziel dieser Dissertationsarbeit somit in der *Algorithmisierung der Lehr-Algorithmen-Entwicklung* [Fra66]. Gemeint ist die Entwicklung eines „Kompakt"- oder Baukastensystems unter dem Paradigma der komponentenbasierten Programmierung für Generierung und Unterhalt von e-Learning-Systemen samt aller hierzu notwendigen Komponenten (Verwaltung, Inhalteerstellung bzw. -präsentation, Navigation, Informationssuche, etc). Die Anwendung dieses Baukastensystems resultiert in plattformunabhängigen Systemen, die zeit- und kostensparend zentral aufgesetzt und durch verteiltes Multi-Authoring mit multimedialen Inhalten gefüllt und stets aktualisiert werden können. Sie beinhalten wiederverwendbare Module in Form von Lehrinhalten und Lernfunktionalitäten und sorgen außerdem für verschiedene digitale Kommunikationsformen, welche die sozialen Aspekte des gemeinsamen Lernens unterstützen.
Bei der Realisierung aller algorithmischen Konzepte wurde Wert auf eine anwendungsunabhängige Umsetzung gelegt. Als Nachweis des Nutzens und Mehrgewinns derart generierfähiger Systeme werden zwei bereits realisierte e-Learning-Systeme aus dem Schnittstellenbereich zwischen Medizin und Informatik, *NeuroAssistant* und *MURMEL*, vorgestellt, die in den letzten Jahren im Rahmen zweier Kooperationen zwischen dem WSI/GRIS[3] der Universität Tübingen, der Abteilung für Neuroradiologie des Universitätsklinikums Tübingen und weiteren Kooperationspartnern entstanden sind. Diese Dissertation richtet sich daher insbesondere an potenzielle Anwender, die ohne großen zeitlichen und finanziellen Aufwand ein e-Learning-System in die Praxis einführen wollen.

1.6 Aufbau der Arbeit

Abbildung 1.1 gibt einen Überblick über den Aufbau der Arbeit, die in vier Abschnitte untergliedert ist. Im ersten Teil der Arbeit wird der theoretische Hintergrund aller relevanten Teilaspekte erarbeitet und Ansätze für die Behebung existierender Schwächen entwickelt. Dazu führt Kapitel 2 zunächst in das Gebiet der e-Learning-Systeme ein, anschließend werden gängige Werkzeuge zur Protokollierung des individuellen Nutzerverhaltens und zur Unterstützung von Nutzeraktionen vorgestellt. Kapitel 3 vergleicht Ansätze zur Datenhaltung und untersucht die Web-Anbindung von Datenbanken. Kapitel 4 beschreibt schließlich theoretische Grundlagen der multimedialen Informationssuche.

Der zweite Teil der Arbeit widmet sich dem Stand der Technik von bestehenden Systemen in Forschung und Praxis. Dazu stellt Kapitel 5 im ersten Abschnitt beispielhaft einige existierende e-Learning-Systeme aus dem Anwendungsbereich Medizin vor. Dies geschieht stets unter Berücksichtigung der im ersten Teil eingeführten Grundlagen von Datenmanagement, Benutzermodellierung, Evaluierungstechniken und Inhalteerstellung. Der zweite Abschnitt von Kapitel 5 geht speziell auf Informationssysteme ein, die verschiedene Ansätze für eine multimediale Suche realisieren. Der Schwerpunkt liegt hierbei auf Suchmöglichkeiten für Bildmaterial.

Der dritte Teil stellt die im Rahmen dieser Dissertation entwickelten neuen Lösungen vor. Dazu wird in Kapitel 6 zunächst der eLS-Baukasten vorgestellt, der eine automatisierte Generierung von e-Learning-Systemen ermöglicht. Der Schwerpunkt des Kapitels liegt hierbei auf der Beschreibung aller serverseitig

[3] WSI/GRIS - Wilhelm-Schickard-Institut für Informatik / Graphisch-Interaktive Systeme.

eingesetzten Techniken und Konzepte wie grundlegende Architektur, Datenmanagement, Wissensmodellierung und Durchführung von Generierungen. Kapitel 7 erläutert anschließend die Clientseite derart generierter Systeme. Hier werden auch die entwickelten Werkzeuge zur Erstellung von Inhalten, Benutzerunterstützung und Evaluierung dargestellt. Um die Beschreibung des neuen Ansatzes zur multimedialen Informationssuche geht es schließlich in Kapitel 8.

Der letzte Teil der Arbeit umfaßt die Bewertung der entwickelten Konzepte und Werkzeuge und zeigt bisher erzielte Ergebnisse. Kapitel 9 enthält eine kurze Beschreibung zweier Beispiele medizinischer e-Learning-Systeme, die mit dem eLS-Baukasten realisiert wurden, und erste Evaluierungsergebnisse. Kapitel 10 beinhaltet schließlich eine Zusammenfassung und Diskussion, sowie einen Ausblick auf weitere Arbeiten.

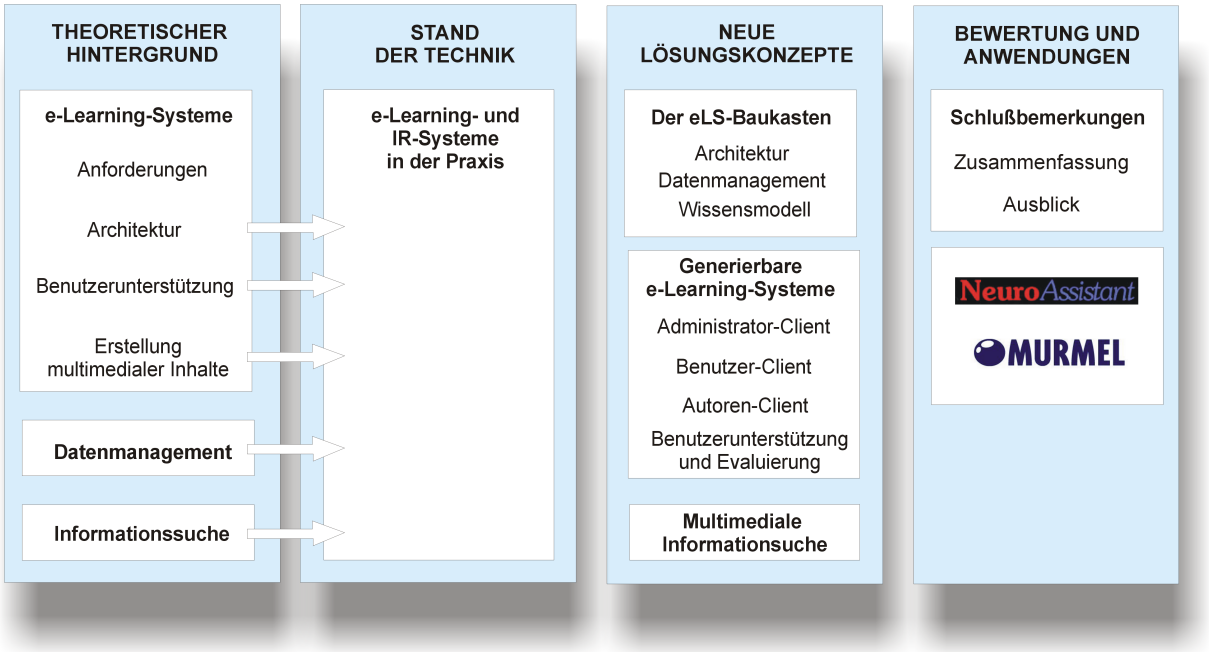

Abbildung 1.1: Inhaltlicher Aufbau der Arbeit

Teil I

Theoretischer Hintergrund

Kapitel 2

e-Learning-Systeme: Aufbau, Benutzerunterstützung und Inhalteerstellung

Um computerbasierte Lernsysteme zu entwickeln, die dem Nutzer einen erfassbaren Mehrgewinn im direkten Vergleich zu konventionellen Lernmethoden bieten, spielen neben deren inhaltlicher Qualität vor allem didaktische und technische Faktoren eine wichtige Rolle. Darüber hinaus sind Werkzeuge erforderlich, die auf der einen Seite Benutzer bei der Arbeit mit dem System unterstützen, und auf der anderen Seite eine komfortable Erzeugung von Inhalten für diese Systeme ermöglichen. In den folgenden Abschnitten werden diese Einzelaspekte näher untersucht und bestehende Defizite bzw. Ansätze für Verbesserungen erläutert.

2.1 Entwicklungsgeschichte

Traditionelle Lehr- und Lernmethoden wie Vorlesungen und Seminare im Stil des Frontalunterrichts oder aber Praktika und Übungsgruppen sind bewährte und dominierende Methoden der Wissensvermittlung. Problematisch ist die dabei fehlende zeitliche und örtliche Flexibilität bei Lehrenden und Lernenden. Ebenso sind die häufig großen Gruppen und das einheitliche Lerntempo in den letzten Jahren zunehmend in die Kritik geraten, weil dadurch die Möglichkeit des selbstorganisierten Lernens eingeschränkt wird. Der heute allgemein anerkannten Forderung nach kontinuierlicher Weiterbildung stehen besonders auch der zeitliche, organisatorische und finanzielle Aufwand traditioneller Methoden im Wege.

Um die beschriebenen Defizite auszugleichen, entstanden zu Beginn der 80er Jahre die ersten computergestützten Lern- und Lehrsysteme, sog. **CBT-Systeme (CBT - Computer Based Training)**, die den Lernenden größere Flexibilität bei der Organisation des Lernens und neue, multimediale Lernformen zur Vermittlung komplexer Sachverhalte anboten. Diese Programme wurden entweder auf CD-ROM vertrieben oder als plattformabhängige Programme auf Standalone Rechnern einzelnen Benutzergruppen zur Verfügung gestellt.

Konzeptionelle Schwächen ließen eine Etablierung dieser Lernsysteme im Aus- und Weiterbildungsbetrieb anfangs nicht zu. So beschränkte beispielsweise die Abhängigkeit der einzelnen Systeme von einer speziellen technischen Konfiguration oder einem speziellen Betriebssystem deren Akzeptanz und Einsatzmöglichkeiten. Der Aufwand für Installation, Wartung und Aktualisierung von Lerninhalten, sowie die fehlende Wiederverwendbarkeit erarbeiteter Lehrinhalte bzw. Systemfunktionalitäten und meist proprietäre Datenformate erschwerten die effiziente Nutzung dieser Systeme. Da in der Regel keine Internet-Anbindung existierte, war die Kommunikation der Lernenden untereinander, sowie zwischen Lernenden und Lehrenden und jegliche Form der Benutzer- und Systemevaluierung umständlich und aufwändig.

Die gemachten Erfahrungen führten zur Entwicklung der ersten Generation Web-basierter **e-Learning-Systeme**, die die erkannten Schwächen auszugleichen versuchten. Unbestrittene Vorteile der zentralen Verwaltung auf Servern sind die zeit- und ortsunabhängige Verfügbarkeit auf allen gängigen Betriebssystemen und die schnelle Aktualisierbarkeit. Der Einsatz aktueller Internet-Technologien bietet zudem neuartige didaktische Möglichkeiten in Form netzgestützter kooperativer und kommunikativer Lerntechniken bzw. Anwendungsszenarien. Beim Versuch, diese Systeme in der Aus- und Weiterbildung zu etablieren, wurden jedoch konzeptionelle und technische Schwächen deutlich, die ein erneutes Umdenken hinsichtlich didaktischer und technischer Aufbereitung erforderlich machten. Die momentan entstehende Systemgeneration soll die erkannten Defizite beheben und so zur höheren Akzeptanz sowohl bei Lehrenden als auch Lernenden beitragen.

2.2 Anforderungen an e-Learning-Systeme

Die Anforderungen, die insgesamt an e-Learning-Systeme gestellt werden, betreffen die eingesetzten Techniken zur Implementierung, Inhalteerzeugung und Präsentation.

Bei der Wahl einzusetzender technischer Instrumente ist zu berücksichtigen, dass die zu entwickelnde Anwendung für die bei der Zielgruppe vorhandene Systemumgebung konzipiert ist und möglichst keine Systemmodifikationen auf Anwenderseite erfordert. Etablierte Standards schaffen die Offenheit hin zu anderen Systemen und Technologien. Insgesamt muss ein ausgewogenes Verhältnis zwischen Hardware-Anforderungen, Laufzeitverhalten, gegebenen Transferraten und gewählten Präsentationswerkzeugen gefunden werden.

Von administrativer Seite ist der Aufwand für die Systemverwaltung und -wartung minimal zu halten, beispielsweise durch eine zentrale Installation des Systems und die orts-, zeit- und plattformunabhängige Administration.

Von inhaltlicher Seite entsprechen die Anforderungen denen, die auch an gedrucktes Lehrmaterial gestellt werden, z.B. sachliche Richtigkeit, Ausbildungsrelevanz und Aktualität. Durch das neue Präsentationsmedium werden diese durch Bedingungen wie die zielgruppengerechte Aufbereitung des Lehrmaterial, die angemessene Nutzung der multimedialen und hypertextuellen Präsentations- und Interaktionsmöglichkeiten, sowie die Integration von Nachschlagemöglichkeiten und weiterführender Literatur ergänzt.

An die didaktische Aufbereitung und Präsentation des Lehrmaterials werden die weitaus größten Anforderungen an e-Learning-Systeme gestellt, die der Übersichtlichkeit halber in der folgenden Auflistung zusammengefaßt sind.

- Konzeption der Lerninhalte nach dem Instruktions- und dem Problemlösungsparadigma: Dies umfaßt an individuelle Interessen und Vorkenntnisse der Nutzer adaptierbare Lernziele, das Angebot verschiedener Lernwege und Schwierigkeitsgrade, die aktive und handlungsorientierte Vermittlung von Wissen, das eigenständige Erarbeiten von Lerninhalten und die Darbietung multipler Perspektiven und Kontexte für Übertragbarkeit von Wissensinhalten.
- Einsatz motivationssteigernder multimedialer und interaktiver Module.
- Kontrollmöglichkeiten für das erarbeitete Wissen.
- Integration verschiedener Kommunikationsdienste.
- Einheitliche, intuitiv bedienbare, ansprechende Benutzeroberfläche.
- Plattformunabhängige Nutzung und schneller Zugriff.

Wird ein System diesen Anforderungen gerecht, können die Vorteile des digitalen Lernens gegenüber herkömmlichen Methoden in vollem Umfang genutzt werden.

2.3 Aufbau von e-Learning-Systemen

Eine Unterteilung von e-Learning-Systemen in didaktisch und technisch motivierte Kategorien kann anhand verschiedener Systemtypen bzw. -architekturen vorgenommen werden.

2.3.1 Didaktische Sicht: Unterstützte Interaktionsformen

Bei einer Kategorisierung von e-Learning-Systemen gemäß der Wissensarten und des eingesetzten Lernmodell können nach [Haa97] derzeit fünf Typen unterschieden werden. Die zur Unterscheidung verwendeten Kriterien sind dabei *lerntheoretischer* Natur und betreffen im Wesentlichen den Grad der Interaktionsfreiheit, über die der Lernende im System verfügt [Sch97]:

- **Präsentationssysteme** liefern Informationen in festgelegter, sequenzieller Reihenfolge, die vom Nutzer nur eingeschränkt beeinflußbar ist.
- **Browsing-Systeme** bieten, über ein Inhalts- oder Stichwortverzeichnis organisiert, einzelne Inhaltsseiten oder Kapitel zu verschiedenen Sachverhalten.
- **Drill & Practice-Systeme** haben eine starre Ablaufstruktur und dienen der reinen Abfrage des aktuellen Wissensstands. Dazu werden in Anlehnung an reale Prüfungssituationen eine oder mehrere Fragen in Folge gestellt, die vom Lernenden beantwortet werden müssen. Das System reagiert mit einem Feedback in Form von „richtig“ oder „falsch“ und einer abschließenden Gesamtbewertung der Befragung.
- **Tutorielle Systeme** präsentieren neuen Lehrstoff, stellen inhaltliche Fragen und machen den weiteren Kursablauf von den Antworten des Nutzers abhängig. Neben einem Feedback in Form von „richtig“ oder „falsch“ werden auch Hinweise zur richtigen Lösungsfindung gegeben. Ziel ist die Vermittlung und Einübung neuen Wissens.
- **Simulationssysteme** basieren auf Modellen, die einen Ausschnitt der Realität möglichst genau nachzubilden versuchen. Für Lernende besteht die aktive Aufgabe darin, in mehreren Verfahrensschritten den richtigen Zielzustand des Systems zu erreichen. Das Trainieren komplexer Handlungsabläufe wird durch die starke Dynamik und den hohen Interaktionsgrad, mit dem der Lernende in den Problemlösungsprozess integriert wird, unterstützt.
- **Intelligente tutorielle Systeme** entstammen dem Gebiet der Künstlichen Intelligenz und bieten eine zeitlich korrelierte Anpassung des Systems an den aktuellen Nutzer mit dem Ziel einer optimalen Unterstützung des individuellen Lernfortschritts [Lus92]. Parallel werden unter anderem das Lernverhalten, die Vorkenntnisse, oder auch Vorlieben des einzelnen Nutzers protokolliert und kontinuierlich ausgewertet.

In der Praxis findet man bereits einige Systeme, die einem oder einer Kombination aus mehreren der vorgestellten Systemtypen entsprechen. Einige Beispiele werden in Kapitel 5.1 vorgestellt.

Eine weitere Differenzierungsmöglichkeit orientiert sich nach [Haa98a] an verschiedenen logischen Schichten innerhalb des Systems. Hier wird unterschieden zwischen:

- der **Präsentationsschicht** als Schnittstelle zwischen Benutzer und Lehrsystemlogik,
- der **Lehrsystemlogikschicht**, deren Aufgabe die Protokollierung von Benutzeraktivitäten, das Auslösen einer Reaktion, und deren Weitergabe an die Präsentationsschicht ist[1], sowie

[1] Die Verarbeitung basiert auf Lehrwissen, das in der Lehrsystemlogikschicht enthalten ist.

- der **Domänendaten- und Wissensschicht**, in der das in einem Lernsystem verfügbare und zu vermittelnde Domänenwissen sowie multimediale Informationseinheiten etwa in Form einer Datenbank, als Textdateien oder ähnlichem abgelegt sind.

Diese Schichten spielen auch bei der anschließenden Beschreibung möglicher e-Learning-System-Architekturen eine Rolle.

2.3.2 Technische Sicht: Architektur

Was die technische Organisation von e-Learning-Systemen betrifft, so kann eine Kategorisierung in Bezug auf die zugrunde liegende Systemarchitektur vorgenommen werden. Abbildung 2.1 zeigt dazu eine Übersicht der drei gängigsten Architekturtypen unter Berücksichtigung der im vorangegangen Abschnitt beschriebenen logischen Schichten.

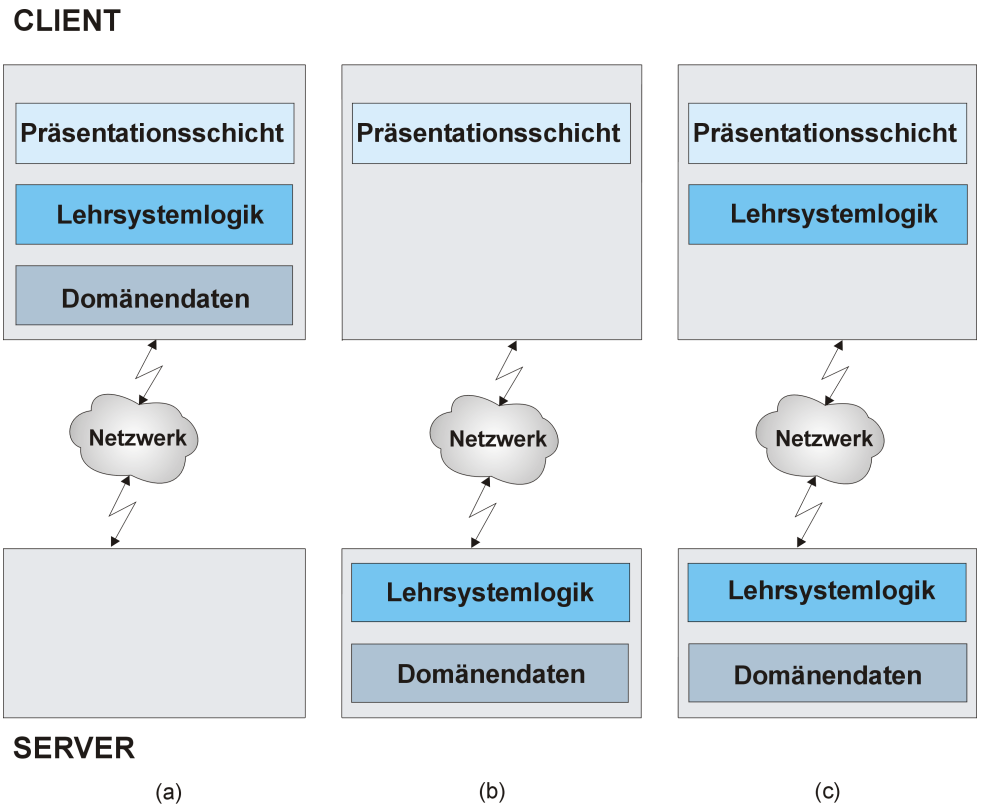

Abbildung 2.1: Clientzentrierte (a), serverzentrierte (b) und verteilte (c) Architektur für e-Learning-Systeme nach [Haa98a]

In einer **clientzentrierten Architektur** dient der Server lediglich als File-System, von dem das komplette Lernprogramm auf den Client heruntergeladen wird (Abbildung 2.1 (a)). Der Hauptnachteil dieses Ansatzes liegt im notwendigen Download. Wegen des fehlenden Rückkanals können keine automatisch aktualisierten Inhalte angeboten werden, eine statistische Erhebung über individuelles Nutzerverhalten bzw. die Kommunikation zwischen einzelnen Lernenden und Lehrenden ist nicht möglich.

Im Falle einer **serverzentrierten Architektur** befindet sich die gesamte Lehrsystemfunktionalität auf dem Server (Abbildung 2.1 (b)). Der Client wird ausschließlich zur Anzeige der vom Server erzeugten Daten genutzt, d.h. er baut bei sämtlichen Benutzereingaben eine Verbindung zum Server auf und sendet

diesem die Benutzereingaben. Nachdem der Server über eine CGI-Schnittstelle die Benutzerdaten an ein spezielles externes Programm weitergegeben hat, werden die entsprechenden Ergebnisse über den WWW-Server zum Client zurückübertragen. Diese Architektur bietet weitgehende Plattform-Unabhängigkeit, breite Verfügbarkeit und Aktualität, und erfordert clientseitig keinerlei Installation. Als nachteilig erweist sich jedoch die einseitige Lastverteilung mit einhergehender hoher Netzbelastung.

In einer **verteilten Architektur** ist die Lehrsystemlogikschicht auf Client und Server aufgeteilt (Abbildung 2.1 (c)). Anfragen an Domänendaten- und Wissensschicht können in Abhängigkeit von den vorhergehenden Nutzeraktionen auf dem Server erzeugt und die Ergebnisse der Abfragen dort analysiert werden. Lediglich die endgültigen Ergebnisse solcher Anfragen werden anschließend auf den Client übertragen. e-Learning-Systeme, die auf dieser Architektur basieren, bieten einerseits durch die Art der Lastverteilung gute Performance und geringe Netzbelastung, sind andererseits aber aufwendig zu implementieren, da eine Client- und eine Serverapplikation erstellt werden müssen, die über Rechnernetze miteinander kommunizieren. Zur Erläuterung der hierzu notwendigen Basistechniken sei auf Abschnitt 3.7 verwiesen.

2.4 Benutzerunterstützung in e-Learning-Systemen

Ein weiterer Grund für die bisher eher geringe Akzeptanz von e-Learning-Systemen ist der Mangel an individueller Benutzerunterstützung auf Clientseite. In den vergangenen Jahren wurden viele Versuche unternommen, Benutzermodelle und Anpassungsfunktionen zu entwickeln, deren Zweck eine verbesserte Zusammenarbeit zwischen den Benutzern und dem Lernsystem, sowie eine Steigerung der Lerneffizienz ist. Zur weiteren Erörterung dieser Thematik sind zunächst einige Begriffsdefinitionen notwendig, die der folgende Unterabschnitt vornimmt.

2.4.1 Terminologie

Unter dem Begriff **Benutzerunterstützung** werden alle technischen, inhaltlichen und didaktischen Komponenten zusammengefaßt, die einem Nutzer eine *individuelle* Arbeitsunterstützung bieten.

In diesem Zusammenhang nimmt die **Benutzermodellierung** einen wichtigen Stellenwert ein. Das Benutzermodell protokolliert den „persönlichen Zustand“ des einzelnen Benutzers und speichert Informationen wie z.B. Interessensschwerpunkte, den Ausbildungsstand, bisher bearbeitete Lerninhalte oder erreichte Ergebnisse bei Quiz-Modulen. Diese Daten bilden die Basis für die individuelle Anpassung des Systems an den Benutzer, wobei **adaptierbare** und **adaptive** Systeme unterschieden werden. *Adaptierbarkeit* liegt vor, wenn ein e-Learning-System vom Benutzer oder Administrator so verändert wird, dass es dem speziellen Unterstützungsbedarf des Lernenden gerecht wird. Nach [Leu97] zeigt sich die *Adaptivität* eines Systems dagegen in dessen Fähigkeit, den Unterstützungsbedarf jeden einzelnen Benutzers zu analysieren und individuelle Anpassungen des Lehrangebots vorzunehmen. Derartige Methoden werden als *adaptive Werkzeuge* bezeichnet.

Das Zusammenspiel von Benutzermodellen und adaptiven Werkzeugen kann beispielsweise durch “intelligente Agenten“ oder spezieller durch „Interface Agenten“ ermöglicht werden [Woo95, Pea96, Maes94]. Diese haben die Funktion, Benutzer anzuleiten, sie bei der Ausführung von Aufgaben zu begleiten, mit dem Benutzer und dem System zu „kommunizieren“ und ein Gedächtnis bereitzustellen (vgl. Abbildung 2.2). Ein Beispiel eines solchen Agenten ist der Assistent, den Microsoft in der aktuellen Version seines Office 2000-Pakets dem Benutzer zur Seite stellt.

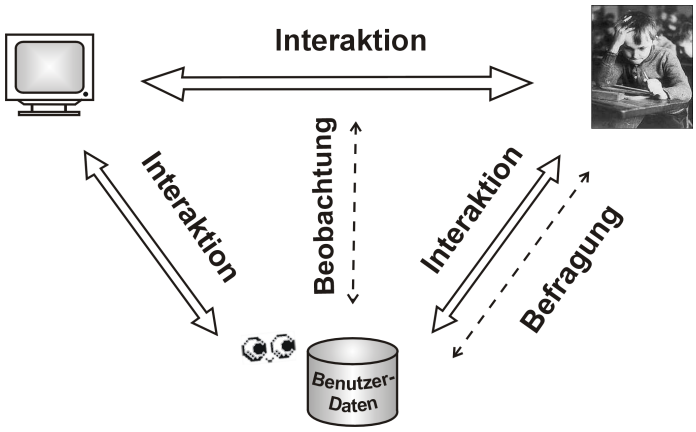

Abbildung 2.2: Interface Agent nach [Maes94]

2.4.2 Benutzermodellierung

So wie ein erfahrener Lehrer im klassischen Unterricht individuelles Vorwissen, Lernstil und Interessensschwerpunkte eines Schülers erfassen und im Ablauf der Lehre berücksichtigen kann, sollte auch ein e-Learning-System die Charakteristika eines Lernenden ermitteln, modellieren und in die Präsentation von Lerninhalten mit einfließen lassen können.

Hier sind *Benutzermodelle* hilfreich, die Informationen über den einzelnen Nutzer sammeln und persistent ablegen. Typische Daten sind zu Ausbildungsstand und Fachkenntnissen, persönliche Lernziele oder Resultate bereits absolvierter Übungsaufgaben, wobei die Daten automatisch und nutzertransparent oder im Dialog mit dem Nutzer ermittelt werden. Der Aufbau eines entsprechenden Benutzermodells könnte in diesem Fall folgendermaßen aussehen:

- **Angaben zur Person**: Alter, Geschlecht, Ausbildungsstand, Vorwissen, Interessensgebiete, etc.
- **Angaben bzgl. der inhaltlichen Nutzung**: Welche Inhalte wurden selten/oft besucht, welche Ergebnisse wurden bei Wissensabfrage-Modulen erzielt, welche Darstellungsformen für welche Inhaltetypen wurden bevorzugt, etc.
- **Angaben bzgl. der Funktionsweise des Systems**: Wie oft und wann wurden Online-Hilfe, Kommunikationstools, Volltextsuche, GuidedTour, Sitemap, etc. benutzt.

Die Akquisition derartiger Daten erfolgt nach [Jör98] durch unterschiedliche Vorgehensweisen:

- **Explizite Befragung des Benutzers**: dies geschieht durch Ausfüllen eines Fragebogens, bei dem der Benutzer Angaben zur Person, seinen Fachkenntnissen, Interessen, Lernzielen, etc. macht,
- **Explizite Vorschläge an den Benutzer**: hier werden zu Beginn einer Sitzung eine Reihe von Inhalten vorgeschlagen und der Benutzer um eine Bewertung gebeten,
- **Automatische Protokollierung des Benutzerverhaltens**: hierbei wird von Systemseite ermittelt, wie lange sich ein Benutzer im System aufhält, wie er im System navigiert, welche Inhalte er aufruft oder welche Punktzahlen er bei der Durchführung von Quizmodulen erzielt. Ein Beispiel für eine einfache Protokollierung von Nutzerdaten sind die sog. *Cookies*, die als Textdatei von Webservern zur Ablage kurzer Informationen auf dem Clientrechner abgelegt werden.

Die ersten beiden Ansätze haben den Vorteil, aussagekräftige und zutreffende Daten über jeden einzelnen Benutzer akquirieren zu können. Ihr Nachteil liegt jedoch in der fehlenden Akzeptanz auf Seiten der Benutzer, die den notwendigen Aufwand oder die Preisgabe persönlicher Informationen scheuen. Der letzte Ansatz vermeidet diesen Nachteil, dafür sind aufwendigere Implementierungen zur Protokollierung und Auswertung der Daten notwendig. Mögliche Modelle, in denen auf eine oder mehrere dieser drei Arten Daten abgelegt werden, werden nach [Stoe97] in statische und dynamische Modelle eingeteilt. Während **statische Modelle** die unveränderlichen, einmal erfaßten Daten über einen Nutzer enthalten, erfolgt bei **dynamischen Modellen** eine Aktualisierung der Daten während der Interaktion mit dem Anwendungsprogramm.

Die innerhalb von Benutzermodellen gesammelten Daten können für unterschiedliche Zwecke verwendet werden. Sie können Grundlage sein für die Adaptivität von Lernangeboten an individuelle Nutzer (vgl. Abschnitt 2.4.3) oder die notwendige Voraussetzung für quantitative Evaluierungsmaßnahmen eines e-Learning-Systems bilden.

2.4.3 Adaptivität und Adaptierbarkeit

Die individuelle Anpassung von Softwaresystemen an den jeweiligen Benutzer ist in den letzten Jahren auf großes Interesse sowohl in der grundlagen- als auch der anwendungsorientierten Forschung gestoßen [Bru00, Abi93]. Seit geraumer Zeit wird erforscht, auf welche Art den Benutzern hier individuelle und aktive Unterstützung angeboten werden kann. Im Hinblick auf e-Learning-Systeme existieren bisher erst wenige Ansätze, die sich speziell mit der Anpassung eines Systems an das aktuelle Lernverhalten eines Benutzers befassen.

Mit den bereits in Abschnitt 2.3.1 erwähnten **intelligenten tutoriellen Systemen (ITS)** wird von Seiten der Künstlichen Intelligenz der Versuch unternommen, der Anforderung gerecht zu werden, Benutzern eine individuelle und aktive Unterstützung beim computerbasierten Lernen anzubieten. Auf der Basis eines vorhandenen Benutzermodells werden dazu der spezielle Kenntnisstand eines Nutzers ermittelt und ihm an diesen angepasste Aufgaben und individuelle Hilfestellungen angeboten.

Vom Einsatz intelligenter tutorieller Systeme verspricht man sich insgesamt vor allem folgende positive Einwirkungen auf den Lernprozess:

- eine Korrektur von fehlerhaftem Wissen des Lernenden,
- die Aufdeckung von Lernfehlern durch die Rückverfolgung des individuellen Problemlöseprozesses,
- das Aufdecken von Wissens- und Lernlücken,
- eine Verschiebung des Lernangebot-Niveaus auf der Basis einer Analyse des individuellen Wissensstands, und
- die Aufzeichnung eines jeden Lernvorgangs.

In der Praxis konnten sich die angestrebten Ziele allerdings nur teilweise realisieren lassen und die Lernwirksamkeit dieser oftmals sehr komplexen Programme ist nach mehreren Untersuchungen überraschend gering [Sch97]. Auch scheint der Aufwand für die Realisierung eines intelligenten Tutors innerhalb von e-Learning-Systemen unverhältnismäßig hoch im Vergleich zu seinem praktischen Nutzen, da sich der Lernende im Vergleich zu anderen Software-Applikationen eher kurz mit einem speziellen Lernsystem beschäftigt und noch bevor das System ein intelligentes Modell zu diesem Benutzer entwickeln bzw. füllen kann, hat dieser die Arbeit mit dem System bereits beendet. Erfolgreiche Anwendungen müssen daher einen geeigneten Kompromiss zwischen intelligenten Werkzeugen, dem damit verbundenen Entwicklungs- und Integrationsaufwand und dem tatsächlichen praktischen Nutzen finden [Enc97].
Für die Erarbeitung eines entsprechenden Lösungsansatzes sind die folgenden Begriffe hilfreich [Leu97]:

- **Adaptationsrate**: Dies ist der zeitliche Abstand, in dem ein einmal hergestellter Adaptationszustand neu adaptiert wird. Unter *Mikroadaptation* werden dabei alle Maßnahmen verstanden, die (dynamisch) innerhalb einer laufenden Lernsitzung getroffen und überprüft werden, unter *Makroadaptation* solche, die nur einmal vor Beginn einer Arbeitssitzung festgelegt werden. Beide Formen können im Rahmen von e-Learning-Systemen sinnvoll angewendet werden.

- **Art der Adaptationsmaßnahme**: Hierbei werden Lehrziel, Lehrmethode und Lehrzeit an den einzelnen Lernenden angepasst, wobei für e-Learning-Systeme nur die ersten beiden Maßnahmen relevant sind. Das Lehrziel kann bei der Bearbeitung von Inhalten oder Abfragemodulen dynamisch an den aktuellen Wissensstand des Lernenden angepasst werden, indem beispielsweise bei fehlerhafter Bearbeitung automatisch ein Inhalt einer nächstniedrigeren Schwierigkeitsstufe angeboten wird.

- **Adaptationszwecke**: Möglicher Zweck ist die Beseitigung individueller Wissensdefizite, beispielsweise durch das Angebot an den Lernenden, weitere Wissenseinheiten oder Lernmodule zu bearbeiten oder gezielte Lernhilfen in Anspruch zu nehmen.

Zur Unterscheidung der Adaptation, bei der eine automatische Anpassung von Systemseite vorgenommen wird, vom Begriff der **Adaptierbarkeit** lassen sich nach [Sch97] zwei Sichtweisen formulieren:

- **Adaptierbarkeit der Funktionalität**: Diese bezieht sich auf die innerhalb eines Systems zur Verfügung stehender Werkzeuge zur speziellen Ablage benutzerspezifischer Daten. Beispiele sind das Setzen elektronischer Lesezeichen für schnelles Wiederauffinden interessanter Inhalte (“Bookmarks“), die Angaben, an welchen Themenstellungen ein Benutzer speziell interessiert ist oder die Voreinstellung individualisierter Suchmöglichkeiten[2].

- **Adaptierbarkeit der Benutzerschnittstelle**: Einfache Beispiele dieser Kategorie sind die Angabe, welche Werkzeuge innerhalb des Systems verfügbar sein sollen, benutzerdefinierte Kommandos, individuelle Einstellungen beim Dialogverhalten mit dem System und Layoutfestlegungen.

Die Anpassung eines e-Learning-Systems an seine Benutzer kann sich somit sowohl auf die Bedienungsfunktionalität, als auch auf die Darbietung vorhandener Inhalte beziehen und dabei automatisch oder durch benutzerseitige Aktionen erfolgen. Die persistente Ablage derartiger benutzerspezifischer Informationen kann dabei auf der Basis eines entsprechenden Benutzermodells erfolgen.

2.5 Erstellung von Inhalten für e-Learning-Systeme

Nachdem der vorangegangene Abschnitt Möglichkeiten zur Unterstützung der Lernenden eines e-Learning-Systems beschrieben hat, soll in diesem Abschnitt untersucht werden, wie auch die Arbeit von Autoren, die für die Erstellung von Inhalten verantwortlich sind, erleichtert werden kann. Mit Hilfe spezieller **Autorensystemen** wird hier versucht, den Autor bei der Bearbeitung und Integration von Inhalten zu unterstützen.

Sowohl kommerzielle Produkte als auch individuelle Lösungen sind inzwischen verfügbar. Im Wesentlichen lassen sich Entwicklungen für die Erzeugung von Systemen für CD-ROM und für das Internet unterscheiden, wobei der Trend eindeutig in Richtung Internettechnologien geht. Nachfolgend werden allgemeine Eigenschaften von Autorensystemen und aktuelle Anforderungen im Hinblick auf ein Konzept für einen breiten Einsatz und die Akzeptanz bei den Inhalteerzeugern beschrieben.

[2] Die Benutzerunterstützung bei der Informationssuche in e-Learning-Systemen wird ausführlich in Kapitel 4 behandelt.

2.5.1 Terminologie

Generell versteht man unter dem Begriff **Autorensystem** graphisch gestaltete Programme zur Erstellung, Pflege und Wartung digitaler Dokumente. Einfache Beispiele sind Textverarbeitungsprogramme wie *Word* zur Bearbeitung von Texten, Bildverarbeitungsprogramme wie *Adobe Photoshop* zur Bearbeitung von Bildern oder Präsentations-Tools wie *MS Power Point* zur Bearbeitung von Präsentationen.

Im Zusammenhang mit e-Learning spricht man von Autorensystem, wenn ein Programm multimediale Lerninhalte erstellen, pflegen und nach verschiedenen didaktischen Gesichtspunkten arrangieren hilft.

Eine Differenzierung des Begriffs „Autorensystem“ ist im Hinblick auf dessen Zweck möglich: Es existieren zum einen Systeme, mit deren Hilfe *komplette* Lernprogramme zu einem abgeschlossenen, meist überschaubaren Themengebiet realisiert werden können. Bei diesen Systemen, die auch als *Courseware* bezeichnet werden [Sch97], ist der Autor Administrator, Didaktiker, Designer und Lieferant der Inhalte in einer Person und das Autorensystem fungiert als „Programmgenerator“. Zum anderen existieren Programme, die Autoren bei der Erzeugung der eigentlichen *Inhalte* für ein bereits aufgesetztes Lernsystem-Framework unterstützen. In dieser Konstellation besteht eine Trennung zwischen didaktischer und technischer Realisierung und der eigentlichen Inhalteerzeugung. In diesem Fall spielen Faktoren wie Modularisierung, Wiederverwendbarkeit und langfristiger, breiter Einsatz eine große Rolle. Das Autorensystem ist nur eine von mehreren wichtigen Komponenten eines Gesamtkonzepts.

2.5.2 Anforderungen an Autorensysteme

Verschiedene Forschungsgruppen haben sich mit der Evaluierung existierender Autorensysteme und der Erarbeitung allgemeiner Anforderungen an diese beschäftigt und folgendene Empfehlungen formuliert [W3C00, Rod89] :

- **Zugang zum System**: Orts-, zeit- und plattformunabhängiger Autorenzugriff auf das angebundene Lehrsystem ermöglicht flexible Arbeitszeiten und -orte.
- **Verteiltes Arbeiten**: Werden die Inhalte eines Lehrsystems von verschiedenen Autoren beigesteuert, sollte ein zeitgleiches, paralleles Arbeiten am selben Lehrsystem möglich sein. Hierzu gehört beispielsweise auch die Einbindung von Werkzeugen zum kooperativen Arbeiten.
- **Unterstützung aktueller Medientechnologien**: Die neuen Informations- und Kommunikationstechnologien müssen auch in Lehrsystemen Berücksichtigung finden. Dies setzt voraus, dass das Autorensystem mit diesen Medien und Technologien umzugehen weiß.
- **Wiederverwendbarkeit erzeugter Inhalte**: Die von Autoren erzeugten, grundlegenden Inhalte sollten ohne Mehraufwand beliebig oft in Lernmodulen eingesetzt werden können. Das System muss dabei die konsistente Datenhaltung gewährleisten (Versionsverwaltung).
- **Wiederverwendbarkeit didaktischer Konzepte**: Die wiederholte Konstruktion typischer didaktischer Wissensmodelle sollte automatisiert werden können.
- **Standards der erzeugten Inhalte**: Die Verantwortung für die Einhaltung etablierter Standards bezüglich inhaltlicher Gestaltung, Layout und eingesetzter Visualisierungstechniken unterliegt dem Autorensystem.
- **Graphische Benutzerschnittstelle**: Die notwendige Einarbeitungszeit in die Handhabung eines Autorensystems muss kurz sein. Dies setzt eine intuitive, komfortable Gestaltung der Benutzerschnittstelle voraus, die nicht durch Überfrachtung mit Funktionen überfordert, sondern graphische Unterstützung an entsprechender Stelle bietet.

- **Sicherheit**: Das Autorensystem sollte Zugriffsrechte für Inhalte flexibel vergeben können. Der Autor sollte sicher sein, dass kein anderer Autor seine Inhalte unerlaubterweise modifiziert, und Kenntnis darüber haben, welche anderen Autoren seine Inhalte referenzieren.

In der Praxis wird je nach Problemstellung eine Gewichtung einzelner Anforderungen in Abhängigkeit vom jeweiligen Kontext erfolgen müssen.

2.5.3 Klassifikation und Einschränkungen der Anwendbarkeit

Autorensysteme lassen sich hinsichtlich ihrer Möglichkeiten in drei unterschiedliche Kategorien einteilen [Haa98a]:

Zeitbasierte Autorensysteme eignen sich besonders für die Erzeugung festverdrahteter Präsentationen, bei denen multimediale Inhalte in zeitlicher Abfolge miteinander verknüpft werden. Die Systeme sind einfach zu bedienen, bieten aber nur sehr begrenzte Möglichkeiten, die Abfolge von angezeigten Inhalten flexibel zu gestalten. Beispiel eines zeitbasierten Systems ist das Programm *Macromedia Shockwave*.

Bei **iconbasierten** Autorensystemen erstellen Autoren Inhalte, indem in einer graphischen Benutzeroberfläche Icons positioniert und zueinander in Beziehung gesetzt werden können. Die Icons stehen stellvertretend für Medien und Inhalte oder auch für Konstrukte wie „Schleife“, „if“, oder „Verschieben nach“. Diese komfortable Arbeitsweise erlaubt den Aufbau komplexerer Abläufe, die jedoch auf die durch das System vorgegebenen Möglichkeiten beschränkt sind. Ein Beispiel für ein iconbasiertes Autorensystem ist das Programm *Macromedia Authorware*.

Einen flexibleren Ansatz bieten sog. **skript-** bzw. **kartenbasierte Systeme**. Hier müssen vom Autor Skripte erstellt werden, die auf Aktionen des Lernenden hin, z.B. die Auswahl eines bestimmten Buttons oder die Eingabe einer Antwort, ausgeführt werden. Skriptsprachen sind meist sehr leistungsfähig, so dass auch komplexere Lernmodule erzeugt werden können. Allerdings müssen Autoren zunächst die Skriptsprache erlernen. Beispiele dieser Kategorie sind Tools wie *Asymmetrix Toolbook* oder *Hypercard* für Macintosh-Rechner.

Zusätzliche Systemeigenschaften lassen nach [Rei99] auch folgende Unterteilung zu:

- **Anwendungsbreite**: Man unterscheidet anwendungsunabhängige Programme von speziell entwickelten Systemen für Einzellösungen.

- **Didaktisches Konzept**: Ein Autorensystem ist umso komplexer, je problemorientierter das didaktische Konzept des dazugehörigen Lernprogramms ist. Autorensysteme für die in Abschnitt 2.3.1 eingeführten Präsentations- und Browsingsysteme müssen z.B. nur einfache Funktionalitäten zur Erzeugung, Ablage und Verknüpfung multimedialer Inhalte bieten. Bei (intelligenten) tutoriellen oder Simulationssystemen muss dagegen auch das ausführbare, prozedurale und problemorientierte Wissen entsprechend organisiert und mitabgelegt werden. Diese Systeme stellen somit weitaus komplexere Anforderungen an ein Autorensystem.

- **Grad der Autorenunterstützung bei der Inhalteerzeugung**: Der notwendige Unterstützungsgrad hängt von der Anzahl der unterstützten Wissensarten ab. Autorensysteme, über die nur wenige, einfache Wissensarten wie systematisches Wissen bereitgestellt werden sollen, können eine komfortable, graphisch unterstützte Wissenseingabe anbieten. Bei Systemen, die nicht auf einer vorgegebenen Wissensart festgelegt sind, ist eine Unterstützung von Seiten des Autorensystems dagegen komplizierter und weniger komfortabel, wie beispielsweise bei den oben erwähnten skriptbasierten Systemen.

- **Entwicklungsdauer, Wiederverwendbarkeit und Wartung von Lerninhalten**: Durch Autorensysteme wird die Entwicklungsdauer von Lernprogrammen erheblich reduziert. Die Wiederver-

wendbarkeit der erzeugten Inhalte hängt dabei von der Komplexität des zugrunde liegenden didaktischen Konzepts ab. Ist keine Trennung zwischen den Domäneninhalten und der eigentlichen Lehrsystemfunktionalität möglich (vgl. Abschnitt 2.3.2), wird die Wiederverwendbarkeit von Inhalten und entwickelten didaktischen Konzepten erheblich erschwert. Entsprechend aufwendig ist auch die Wartung bereits erzeugter Wissensinhalte.

Die erwähnten Einschränkungen hinsichtlich der Verwendbarkeit der jeweiligen Autorensystemtypen werden auch in [Ker98] bestätigt. Dort wird besonders herausgestellt, dass oftmals keine angemessene Unterstützung bei der Realisierung komplexer Interaktionsräume geboten wird. Dadurch tendieren Autoren dazu, eher lineare Wissensstrukturen aufzubauen, was die Orientierung an didaktischen Ansätzen einschränkt. Darüber hinaus existiert in einfachen Systemen keine angemessene Unterstützung bei der Konstruktion von Wissensmodellen und es fehlen Hinweise von Systemseite bei logischen und systematischen Fehlern der Autoren. Ein teilweise unangemessen hoher Aufwand zur Anpassung von erzeugten Lerninhalten an verschiedene Lernergruppen, Lehrgegenstände und -bedingungen erschwert dabei den Einsatz von Autorensystemen.

Die Forderung nach effizienter und flexibler Unterstützung bei der Inhalteerstellung für Lernprogrammen wird von derzeit verfügbaren Autorensystemen nur teilweise erfüllt. Aspekte hinsichtlich des gebotenen Bedienungskomfort stehen vermeintlich notwendigen Einschränkungen entgegen, die entstehen, wenn das System auch die Erzeugung flexibler und problemorientierter Lerninhalte erlauben soll.

Unter Berücksichtigung etablierter Programmierwerkzeuge und der Vorteile neuer Informations- und Kommunikationstechnologien ist jedoch zwischenzeitlich die technische Basis geschaffen worden, den genannten Anforderungen weitestgehend nachkommen zu können. Die Umsetzung eines entsprechenden Autorensystems wird in Abschnitt 7.3 vorgestellt.

2.6 Zusammenfassung

Zur Einführung in das Gebiet der e-Learning-Systeme wurden in diesem Kapitel deren technische, inhaltliche und didaktische Grundlagen vorgestellt, sowie Anforderungen an diese definiert.

Desweiteren wurde untersucht, inwiefern einerseits individuelle Benutzer eines e-Learning-Systems, andererseits Autoren, die für die Erstellung von Inhalten verantwortlich sind, in ihrer Arbeit unterstützt werden können.

Schwächen derzeitiger Verfahren wurden diskutiert und herausgearbeitet, welche Bedingungen erfüllt sein müssen, um sowohl Lernende, als auch Lehrende in angemessener Weise zu unterstützen.

Auf dieser Basis behandeln die folgenden Kapitel zwei weitere Aspekte, die im Hinblick auf die Akzeptanz und Nutzung von e-Learning-Systemen eine Rolle spielen.
Folgende Fragestellungen werden dazu beantwortet:

- Welche Möglichkeiten des effizienten Datenmanagements innerhalb von e-Learning-Systemen existieren?
- Wie können Benutzer effektiv bei der Informationssuche in den multimedialen Daten unterstützt werden?

Kapitel 3

Datenmanagement in e-Learning-Systemen

Ein zentraler Bestandteil von e-Learning-Systemen ist eine umfangreiche Datensammlung des zu vermittelnden Wissens. Zu dessen Verwaltung bietet sich der Einsatz eines Datenbanksystems an, das ein effizientes Hilfsmittel zur rechnergestützten Organisation und transaktionsbasierten Verwaltung umfangreicher Datensammlungen darstellt. Die für e-Learning-Systeme spezifischen Anforderungen an eine Komponente zur Datenhaltung umfassen hierbei vor allem die Modellierung komplexer und vernetzter Datentypen, die Verwaltung multimedialer Objekte und den Zugriff über das WWW.

Der Schwerpunkt dieses Kapitels liegt auf der Beschreibung der theoretischen und methodischen Grundlagen von Datenbanksystemen, die für die Realisierung einer Datenhaltungskomponente innerhalb von e-Learning-Systemen relevant sind. Die Auswahl eines Datenbanksystems für die Integration in den im Rahmen dieser Arbeit entwickelten eLS-Baukasten soll dabei durch die Gegenüberstellung relationaler und objektorientierter Konzepte motiviert werden. Die letzten Unterabschnitte dieses Kapitels behandeln dabei die in diesem Zusammenhang wichtige Frage, mit welchen speziellen Techniken der WWW-basierte Zugriff auf multimediale Daten realisiert werden kann.

3.1 Terminologie

Nach [Klu00] bezeichnet der Begriff **Datenbank** ein Programmsystem zur rechnergestützten zentralen Verwaltung umfangreicher Datenmengen. Es besteht aus den persistent abgespeicherten Daten (der **Datenbasis**) und allen Anwendungsprogrammen (den **hörenden Prozessen**), die auf die Daten zugreifen und **Transaktionen** wie z.B. Lesen, Schreiben oder Löschen von Daten durchführen. Eine Transaktion ist hier eine Folge von Datenbankaufrufen, die logisch zusammengehören und eine Datenbasis von einem konsistenten Zustand in einen anderen überführen.

Basis einer Datenbank ist das **Datenbank- (DBS)** oder **Datenbank-Management-System (DBMS)**, das als Schnittstelle zwischen Benutzern und der Datenbasis fungiert und über den hörenden Prozess mit beiden kommuniziert. Abbildung 3.1 zeigt den vereinfachten Aufbau einer Datenbank.

In klassischen Anwendungen orientiert sich die Entwicklung einer Datenbank am **3-Schichten-Modell** nach ANSI/SPARC, das die Organisation von Daten festlegt und jeder Ebene ein eigenes Schema zuordnet [Ans75]:

1. **Konzeptionelle Ebene**: Sie legt fest, welche Objekte, Vorgänge und Beziehungen zwischen Objekten modelliert werden müssen.

2. **Interne Ebene**: Sie beinhaltet die physische Speicherung und Organisation von Daten.

3. **Externe Ebene**: Sie umfaßt alle individuellen Sichten einzelner Benutzer oder Anwendungsprogramme auf eine Datenbank.

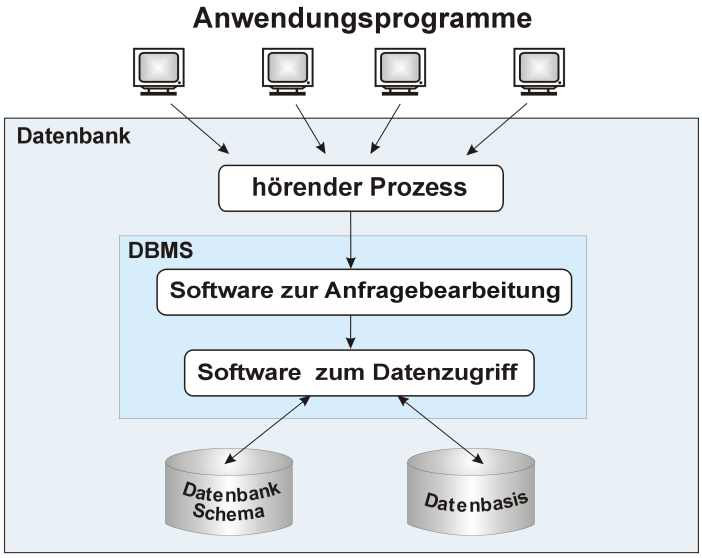

Abbildung 3.1: Aufbau einer Datenbank nach [Elm94]

Eine strikte Trennung dieser drei Schichten bezweckt, dass beispielsweise Änderungen in der internen Ebene vorgenommen werden können, ohne dass die konzeptionelle Ebene davon berührt wird. Desgleichen können bestimmte Änderungen an der konzeptionellen Ebene vorgenommen werden, ohne dass bereits existierende Benutzersichten davon berührt werden.

Das **Datenmodell** ist das zentrale Hilfsmittel zur Erstellung einer Abstraktion über einen gegebenen „Weltausschnitt“ und der gleichzeitigen Abstraktion von den Einzelheiten der physischen Speicherung. Es umfaßt in der Regel eine Menge von Konzepten, mit welchen sich die Struktur einer Datenbank beschreiben lässt, und von Operationen zur Beschreibung lesender und schreibender Anfragen. Beispiele für Datenmodelle sind das **hierarchische Modell**, das **Netzwerkmodell**, das **relationale** und das **objektorientierte Modell**.

Während das hierarchische Modell nur baumförmige Beziehungen zwischen Objekten kennt, die bei der Erstellung der Datenbank festgelegt werden müssen, erlaubt das Netzwerkmodell zusätzlich netzwerkartige Verbindungsstrukturen. Das relationale Modell modelliert Beziehungen zwischen Objekten mit Hilfe von *Relationen*, während das objektorientierte Modell Beziehungen sowohl zwischen beliebigen Objekten als auch deren Einzelbestandteilen herstellen kann. Die beiden letztgenannten Ansätze werden in den Abschnitten 3.3 und 3.5 näher betrachtet, um die Entscheidung für ein objektorientiertes Modell als Grundlage des realisierten e-Learning-Systems zu fördern.

Mit Hilfe von **Datenbank-Anfragesprachen** wird lesend und schreibend auf die Datenbasis zugegriffen: Bei *satzorientierten* Sprachen werden Daten satzweise aus der Datenbank gelesen und zurückgeschrieben, wobei Zugriffsreihenfolge und -pfade von der Struktur des verarbeitenden Programms definiert werden. *Mengenorientierte* oder *deskriptive* Sprachen, die in der Regel in relationalen Systemen zu finden sind, erfragen dagegen nicht einen konkreten Datensatz, sondern erlauben die Formulierung von Anfragen über Datensätze, die bestimmte vorgegebene Kriterien erfüllen. Wichtigster Vertreter dieses Typs ist die *Structured Query Language SQL* [Ach00]. Neben diesen datenflussorientierten Anfragesprachen existieren die sog. *in Host-Sprachen eingebetteten Sprachen*, die meist in Verbindung mit einer höheren Programmiersprache verwendet werden. Dabei werden letztere um Funktions- oder Objektbibliotheken ergänzt, die den Umgang mit Datenbanken in Anwendungsprogrammen erlauben.

3.2 Aufbau und Eigenschaften eines Datenbanksystems

Die allgemeine Architektur eines Datenbanksystems zeigt Abbildung 3.2.

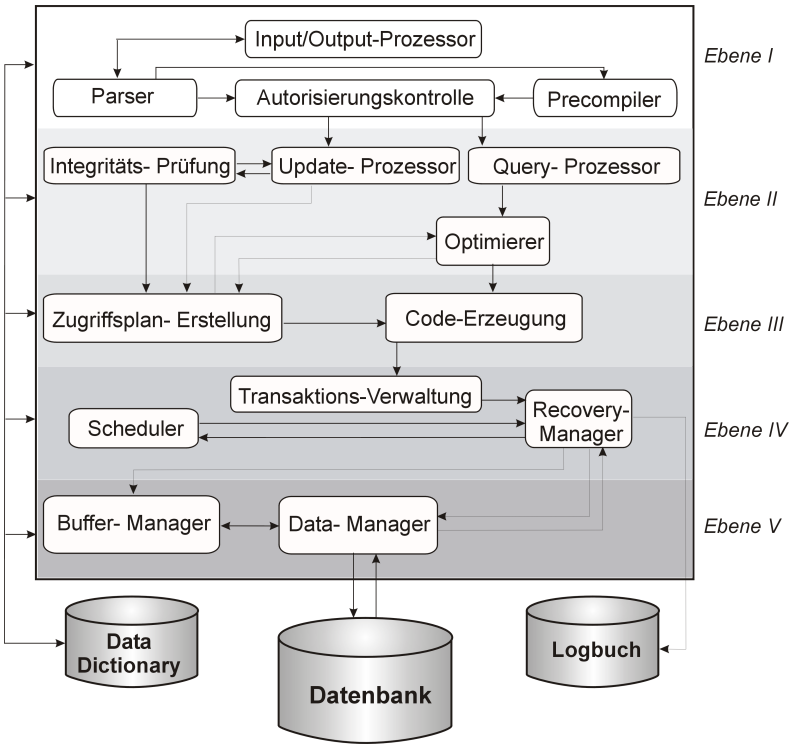

Abbildung 3.2: Komponenten eines Datenbanksystems nach [Vos99]

Hier kann grob zwischen dem eigentlichen Datenbanksystem (Ebenen I-V), das als ausführbares Programm im Hauptspeicher des betreffenden Rechners angesiedelt ist, und den drei im unteren Teil der Graphik dargestellten Datenbeständen unterschieden werden, auf denen das Datenbanksystem arbeitet: der Datenbasis selbst, dem *Data Dictionary*, einer Art Zusammenfassung des internen, konzeptionellen und des externen Schemas, und dem internen *Logbuch* der Datenbank, welches zum Zwecke des Wiederanlaufs nach einem Systemfehler geführt wird. Dem Benutzer einer Datenbank unmittelbar zugeordnet ist dagegen der obere Teil der Architektur (**Ebene I**): ein *Input/Output-Prozessor*, der Kommandos entgegennimmt, ein *Parser*, der deren syntaktische Analyse durchführt, ein *Precompiler* zur Interpretation eingebetteter Kommandos, und eine *Autorisierungskontrolle*, die feststellt, ob ein Benutzer auf die angesprochenen Daten zugreifen darf. Als Resultat dieser ersten Phase liegt die Benutzeranfrage bereits in einer „internen“ Form vor. In **Ebene II** wird im Falle eines Datenbank-Updates der *Update-Prozessor* aktiviert, der die semantische Korrektheit der Anfrage überprüft (*Integritätsprüfung*). Im Falle einer einfachen Datenbankanfrage wird dagegen der *Query-Prozessor* und — zur möglichen Anfragen-Vereinfachung — der *Optimierer* angestoßen. **Ebene III** der Auftragsbearbeitung besteht in der *Zugriffsplan-Erstellung* und Ausführung dieses Plans und mündet in der *Code-Generierung* für den Benutzerauftrag. Dies überführt die Anfrage in eine Folge echter Lese- bzw. Schreibbefehle. Aufgrund des Mehrbenutzerbetriebs einer Datenbank existiert in **Ebene IV** zur Bearbeitung paralleler Aufträge eine *Transaktionsverwaltung*, die durch einen *Scheduler* unterstützt wird. Tritt während einer Transaktion ein Fehler auf, so kümmert sich der *Recovery-Manager* um die Wiederherstellung des konsistenten Datenbestandes. Um eventuell bereits ausgeführte Veränderungen wieder rückgängig machen zu können, bedient er sich dazu des anfangs erwähnten *Logbuch*es. **Ebene V** umfaßt die eigentliche Speicherverwaltung des Systems. Hier werden bei

einem Datenbank-Update der *Buffer*- und der *Data-Manager* aktiv, wobei der erste die Verwaltung eines im Hauptspeicher befindlichen Puffers vornimmt, und der letzte die Vergabe der Hardware-Betriebsmittel und die physischen Zugriffe auf die Datenbank kontrolliert.

Auch wenn die interne Arbeitsweise eines Datenbanksystems aufwendig erscheinen mag, bietet sich dessen Einsatz im Gegensatz zur Verwendung eines einfachen Dateiensystems besonders für die Verwaltung großer, komplexer, zueinander in Beziehung stehender und oftmals zu verändernder Datenmengen an. Es erleichtert die persistente und redundanzfreie Datenablage und zeichnet sich darüber hinaus durch folgende Eigenschaften aus:

- **Datenunabhängigkeit:** Durch die Herstellung einer *physischen* und *logischen Datenunabhängigkeit* bei Einhaltung des ANSI/SPARC-3-Schichten-Modells ist die konzeptionelle Sicht auf einen Datenbestand unabhängig von dessen Speicherung. Zudem ist die Datenbasis unabhängig von Änderungen und Erweiterungen der Anwendungsschnittstellen.

- **Datenschutz und Datensicherheit**: Durch den Ausschluss unautorisierter Zugriffe auf gespeicherte Daten und die Bereitstellung von Methoden zur Verhinderung von Datenverlusten können gesetzliche Richtlinien zum Datenschutz eingehalten und betriebswirtschaftliche Aspekte der Datensicherheit berücksichtigt werden.

- **Transaktionskonzept**: Die besonderen Eigenschaften von Transaktionen (Atomarität und Persistenz von Operationen, Einhaltung von Integritätsbedingungen und die isolierte Behandlung von Transaktionen) ermöglichen den störungsfreien parallelen Zugriff auf eine Datenbasis durch verschiedene Anwendungsprogramme bzw. Benutzer, ermöglicht die Synchronisation konkurrierender Transaktionen mehrerer Benutzer, und gewährleistet die Korrektheit von Datenbankinhalten (Integritätssicherung). Sie ermöglicht insbesondere den konsistenz-erhaltenden Wiederanlauf nach einem Systemabsturz (Recover-Fähigkeit).

- **Datenbanksprachen**: Die Existenz mächtiger Datenbanksprachen erlaubt die Definition von Datenbankobjekten, eine Rekonstruktion von Anwendungsobjekten und die Formulierung von Datenbankanweisungen. Zusammen mit **benutzergerechten Anfragesprachen** ermöglichen sie eine komfortable Anfrageformulierung ohne die Notwendigkeit der Kenntnis der internen Realisierung einer Datenbank. Somit ist eine Realisierung unterschiedlicher **Benutzersichten** auf den Datenbestand für unterschiedliche Anwendungen möglich.

Die genannten Eigenschaften werden von derzeit zur Auswahl stehenden Datenbanksystemen unterschiedlich gut erfüllt. In den folgenden Abschnitten werden mit relationalen und objektorientierten Systemen die beiden wichtigsten Typen von Datenbanksystemen gegenübergestellt und bezüglich der hier beschriebenen Kriterien verglichen.

3.3 Relationale Datenbanksysteme

Relationale Datenbanksysteme sind in zahlreichen Publikationen ([Här99, Sau92, Wed81]) und Handbüchern (z.B. [Fro00, Sch00, Maz00]) ausführlich beschrieben. Sie basieren auf einem einfachen konzeptionellen Modell der Relationen bzw. Tabellen, wobei an Tabellenpositionen nur „atomare“ Attributwerte stehen können. Dies entspricht der *ersten Normalform (1NF)* für Relationen. Da Datensätze im relationalen Modell keine eindeutige Identität besitzen, benötigt man spezielle Attribute, die die Eindeutigkeit sicherstellen und gleichzeitig auch der Modellierung von Beziehungen zwischen verschiedenen Relationen dienen. Zur Formulierung von Anfragen, Eingaben, Änderungen und Löschungen von Datensätzen in Datenbanken stehen dem Benutzer mächtige Anfragesprachen zur Verfügung. Hier hat sich mit *SQL*

eine interaktive Anfragesprache durchgesetzt, die direkt vom Terminal aus genutzt und auch in Programmiersprachen eingebettet werden kann.

Nach [Hug92] zeichnen sich relationale Datenbanksysteme vor allem durch einen hohen Grad an physischer Datenunabhängigkeit aus. Sie eignen sich nach [Heu97] insbesondere für Einsatzbereiche mit folgenden Kennzeichen:

- **Einfach strukturierbare Daten** in festem Format.
- **Einfache Datentypen** mit vorwiegend (alpha)numerischen Attributwerten.
- **Kurze, wiederkehrende Transaktionen** für oft ausgeführte Anfragen und Updates.
- **Hohe Transaktionsraten** bei kommerzieller Ausrichtung der Anwendung.
- **Häufige, einfache „in place“-Updates**, bei denen der Wert eines Attributes unwiederbringlich durch einen neuen ersetzt wird.

Im praktischen Einsatz entsprechen insbesondere Anwendungen im administrativ-betriebswirtschaftlichen Bereich wie z.B. Reservierungssysteme, Bibliotheksverwaltungen, Bank- und Auskunftswesen oder Personalverwaltung diesen Kennzeichen. Dementsprechend nehmen relationale Datenbanksysteme hier eine zentrale Marktposition ein und sind heute für praktisch alle derzeit existierenden Hardware-Plattformen erhältlich. Zur Datenhaltung in den oben erwähnten Anwendungsbereichen haben sich relationale Datenbanksysteme zwar bestens bewährt, es lassen sich jedoch bereits Aspekte identifizieren, die durch relationale Systeme nicht optimal abgedeckt werden können. Um diese soll es in den folgenden Abschnitten gehen.

3.4 Objektrelationale Datenbanksysteme

Auch wenn relationale Datenbanksysteme die derzeit dominierende Technologie bei Unternehmensdatenbanken darstellen, wird die Anwendungsentwicklung in verstärktem Maße von objektorientierten Systemen und Komponenten beherrscht [Boo95]. Beispiele für diesen Trend sind der Erfolg der objektorientierten Programmiersprache Java [Arn00] oder objektorientierter Komponentenstandards wie CORBA [Say99]. Ausgehend von dieser Situation wurde der objektrelationale Ansatz entwickelt, der objektorientierte Anwendungsentwicklungen auf Basis relationaler Systeme ermöglichen soll. Objektrelationale Datenbankmanagementsysteme basieren im Wesentlichen auf einem relationalen Datenmodell, das alle Merkmale relationaler Datenbankmanagementsysteme unterstützt, diese jedoch um objektorientierte Methoden erweitert. Derartige Erweiterungen ermöglichen beispielsweise die Definition neuer Typen aus gegebenen Basistypen im Kontext von SQL-Deklarationen und die Bindung von Operationen an benutzerdefinierte Typen. Eine Unterstützung komplexer Objekte im SQL-Kontext wird dadurch erreicht, dass der Attributwert einer Relation nun auch ein Tupel einer beliebigen anderen benutzerdefinierten Relation sein kann. Außerdem kann eine Relation als Spezialisierung einer anderen definiert werden, wodurch Attribute und Operationen vererbbar sind. Obwohl fast alle Hersteller relationaler Datenbanksysteme derzeit an objektorientierten Erweiterungen arbeiten, existiert bisher kein allgemeingültiger Standard.

Beispiele für objektrelationale Datenbanksysteme sind *Oracle 8* ([Hoh98]), *PostgreSQL* ([Vos99], [postgr]) oder *Informix* ([Pet97], [Lum99]). Ausführliche Informationen zu objektrelationalen Datenbanksystemen finden sich auch in [Sto99], [Mei00] oder [Saa97].

3.5 Objektorientierte Datenbanksysteme

Objektorientierte Datenbanksysteme sind die konsequente Weiterentwicklung relationaler Systeme. Sie sind besonders geeignet für den Einsatz innerhalb neuerer Anwendungsbereiche wie z.B. in Multimedia-

Anwendungen, im Computer-Aided Software-Engineering, in der Medizin oder in ingenieurwissenschaftlichen Applikationen [Mei00, Sto99, Heu97, Saa97]. Bedingt durch die Komplexität dieser Anwendungstypen sind hier Modellierungsmöglichkeiten erforderlich, die das relationale Modell um Konzepte zur besseren Darstellung der Struktur von Anwendungsobjekten und deren objektspezifischen Operationen erweitern. Das „OODBMS Manifesto“ beinhaltet eine erste Definition von Mindestanforderungen an ein solches Modell [Atk89]. Neben bewährten Schlüsseleigenschaften aus dem Bereich der relationalen Datenbanksysteme ist hier auch das Vorhandensein folgender Eigenschaften aus dem Bereich der objektorientierten Programmiersprachen (*Paradigma der Objektorientierung*) vorgeschrieben:

- **Typen, Klassen und komplexe Objekte**: Für Eigenschaften von Objekten sind nicht nur Standard-Datentypen erlaubt, sondern auch die wiederholte Anwendung von Typkonstruktoren (*Strukturteil*).
- **Kapselung**: Neben den Eigenschaften von Objekttypen können auch die mit ihnen durchführbaren Methoden in die Objekttypdefinition eingekapselt werden (*Operationenteil*).
- **Vererbung**: Objekttypen können in einer Vererbungshierarchie angeordnet werden.
- **Objektidentität**: Objekte existieren unabhängig von den Werten ihrer Eigenschaften.
- **Erweiterbarkeit**: Auf der konzeptuellen Ebene ist Erweiterbarkeit durch neue Datentypen und Funktionen gegeben.
- **Berechnungsvollständigkeit**: Durch Einbettung von Datenbankoperationen in eine Programmiersprache wird der sog. „impedance mismatch“ vermieden (vgl. Abschnitt 3.6).

In Bezug auf eine allgemein anerkannte Begriffsdefiniton für objektorientierte Datenbanksysteme hat die *Object Database Management Group (ODMG)*, ein Zusammenschluß von Herstellern objektorientierter Datenbanksysteme, im Jahre 1993 ein Referenzmodell entwickelt, das allgemein als *ODMG-93 Referenzmodell* bezeichnet wird [Cat97]. Gegenstand ist eine Spezifikation objektorientierter Basiskonstrukte, eine *Object Definition Language (ODL)* und eine spezielle Abfragesprache für objektorientierte Systeme mit dem Namen *Object Query Language (OQL)*. Der Standard umfaßt außerdem Vorschriften für die Einbettung von ODL in Smalltalk und C++.
Bei objektorientierten Datenbanksystemen dominieren Systeme, bei denen es sich um Erweiterungen objektorientierter Programmiersprachen handelt, beispielsweise *GemStone* ([Gem96]), *ObjectStore* ([Obj98a, Obj98b]), *Versant* ([Ver92]) und *POET* ([Poe96]). Ganz neuentwickelt wurden dagegen z.B. *O2* ([$O_2$93]) oder *ITASCA* ([Ita91]). In [Ah92] und [Hoh96] finden sich Vergleiche dieser und weiterer objektorientierter Datenbanksysteme, auf das System *ObjectStore* wird in Abschnitt 6.3 noch detailliert eingegangen.

3.6 Abwägung der Eigenschaften verschiedener Systemtypen

Im Hinblick auf die spezifischen Anforderungen von e-Learning-Systemen (Modellierung komplexer und vernetzter Datentypen, Verwaltung multimedialer Objekte und Zugriff über das WWW) wurden relationale und objektorientierte Datenbanksysteme charakterisiert und verglichen. Hierbei zeigten sich vor allem die im Folgenden zusammengefaßten Eigenschaften relationaler Systeme als ausschlaggebend für den Einsatz eines objektorientierten Systems.

- Je nach Anwendungsfall kann es relationalen Systemen an der notwendigen Ausdrucksmächtigkeit fehlen, um komplexe Objekte und hochdimensionale Informationsräume differenziert und strukturiert abzubilden [Ass98].

- In Bereichen wie dem Computer-Aided Design, Wissenschaft und Medizin und generell in multimedialen Anwendungen existieren nur beschränkte Modellierungsmöglichkeiten. Komplexe Objekte müssen aufgrund der notwendigen Abbildung durch normalisierte Relationen in einfache Objekte unterteilt werden, was zum Verlust der Objektidentität führen kann [Wed81].

- Die verfügbaren Datentypen für Attribute beschränken sich in der Regel auf einfache Typen wie Zahlen und Text sowie Fremdschlüssel, was insbesondere die Speicherung multimedialer Daten erschwert.

- Mit SQL verfügen relational organisierte Datenbestände über eine mächtige Abfragesprache, die jedoch nicht berechnungsvollständig ist. Zur Anwendungsentwicklung wird deshalb die Einbettung der Abfragesprache in eine berechnungsvollständige Programmiersprache erforderlich [Kemp96]. Durch eine solche Einbettung entstehen verschiedene Probleme, die man unter dem Begriff **impedance mismatch** zusammenfaßt, z.B. die unterschiedlichen Typsysteme und Sprachstile von Abfragesprachen und Programmiersprachen, von denen erstere mengenorientiert, letztere jedoch satzorientiert arbeiten.

- Schwächen existieren auch bei „aktiven“ Merkmalen, z.B. für konsistenzerhaltene Operationen oder zur automatischen Bereitstellung abgeleiteter Daten. In objektorientierten Systemen ist die Datenbank in eine Programmiersprache eingebettet, wodurch Daten und die darauf agierenden Operationen sind nicht mehr getrennt sind.

Objektorientierte Datenbanken und allgemeine Eigenschaften der Objektorientierung erlauben dagegen die Umsetzung komplexer Systemkonzepte in handhabbare Softwaremodule, die Einbindung standardisierter Komponenten in Anwendungsframeworks und somit insgesamt die Entwicklung von in sich abgerundeten, vor allem auch internetkompatiblen Anwendungen. Mit den Möglichkeiten der Objektorientierung lässt sich außerdem der Mehraufwand bei der Verwaltung vielfältig verknüpfter Daten reduzieren, da so weitgehend beliebig komplex strukturierte Informationen dargestellt und an Strukturen mit spezifischem Verhalten gebunden werden können.

In der endgültigen Wahl eines Systems speziell im Zusammenhang mit e-Learning-Systemen spielen neben den hier diskutierten Eigenschaften in neuerer Zeit zwei weitere Faktoren eine wichtige Rolle: Die Möglichkeiten der Datenbankanbindung an das WWW und die zusätzliche Verwaltung multimedialer Objekte. Die folgenden Abschnitte beschäftigen sich mit dieser Problematik.

3.7 Datenbanken und Internet

Ein wesentlicher Unterschied zwischen e-Learning-Systemen und klassischen Datenbankanwendungen (beispielsweise einer Personaldatenverwaltung) liegt in der Rolle des Endbenutzers, für den sie entwickelt werden: Die Zielgruppe bei e-Learning-Systemen ist ein möglichst breiter Benutzerkreis. Auch können weder geschulte Sachbearbeiter noch spezielle Rechner- und Netzwerkumgebungen vorausgesetzt werden. Daraus ergeben sich zwei Anforderungen, die Datenbanken bei einem Einsatz innerhalb eines e-Learning-Systems erfüllen müssen: Allgemeine Verfügbarkeit und eine einfache Benutzerschnittfläche.

Der Begriff der *allgemeinen Verfügbarkeit* bezieht sich sowohl auf den Ort als auch auf die erforderliche Rechnerumgebung: im Idealfall lässt sich ein e-Learning-System an beliebigen Orten auf beliebigen Plattformen verwenden. Durch eine *einfache Benutzerschnittfläche* soll auch Personen mit durchschnittlichen Computerkenntnissen der Umgang mit einem e-Learning-System möglich sein. Ein Ansatz, um allgemeine Verfügbarkeit zu erreichen, ist die Verwendung eines **HTTP-basierten Ansatzes**. Hierbei ist die Clientsoftware in einen gewöhnlichen Webbrowser eingebettet und kommuniziert über das Hypertext Tranfer Protocol (HTTP) mit einem HTTP-Server. Der Vorteil gegenüber anderen Ansätzen ist die

sehr einfache Benutzeroberfläche. Ein Benutzer muß keine besondere Clientsoftware installieren und die Fähigkeit zur Bedienung eines Webbrowsers kann bei den Benutzern vorausgesetzt werden.

Zur technischen Realisierung von HTTP-basierten Ansätzen gibt es verschiedene Möglichkeiten. Die wichtigsten Techniken zeigen die Abbildungen 3.3 bis 3.7, die im Folgenden kurz erläutert werden [Vos99]:

1.) Common Gateway Interface (Abbildung 3.3): Traditionell gibt es im Bereich des WWWs Gateway-Standards wie das *Common Gateway Interface (CGI)*, das für alle Arten von Übergängen zwischen den einzelnen Diensten des Internets und damit auch für die Anbindung von Datenbanken genutzt werden kann. Bei Anbindungen der ersten Generation waren in den Skripts die möglichen Datenbankanfragen fest codiert, die der zweiten Generation können dagegen beliebige Datenbankanfragen bearbeiten.

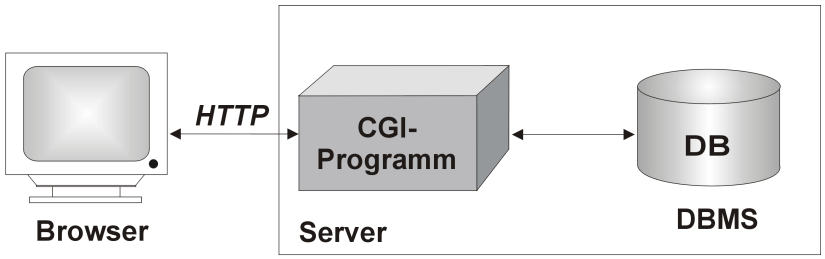

Abbildung 3.3: Datenbanken und Common Gateway Interface

2.) Server-Erweiterung durch APIs (Abbildung 3.4): Einige Internetserver bieten spezielle Server-Application-Programming-Interfaces (APIs) an, mit deren Hilfe die Funktionalität eines Servers anwendungs- oder aufgabenspezifisch erweitert werden kann. So legt etwa ODBC, als Standardschnittstelle für den Zugriff auf beliebige Datenquellen ([Gei95]), eine (API-)Schicht über die Datenbankschicht und zeigt sich Anwendungsprogrammen als einheitliches, homogenes Interface, über das serverseitige Datenbanken angesprochen werden können. Servererweiterungen werden meistens als dynamische Funktions- oder Objektbibliotheken realisiert, die vom Server bei Bedarf geladen werden können.

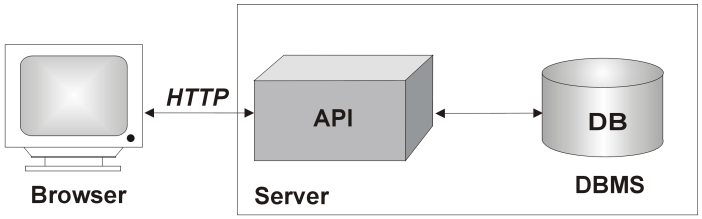

Abbildung 3.4: Datenbanken und Server-Erweiterung durch APIs

3.) Client Side Includes (Abbildung 3.5): Hier lädt der Client, transparent für den Benutzer, eine spezielle Software, die nicht über HTTP, sondern beispielsweise als Java Applet via *Java Database Connectivity (JDBC)*, mit dem Datenbankserver kommuniziert. Somit handelt es sich nicht um eine HTTP-basierte Lösung im eigentlichen Sinne.

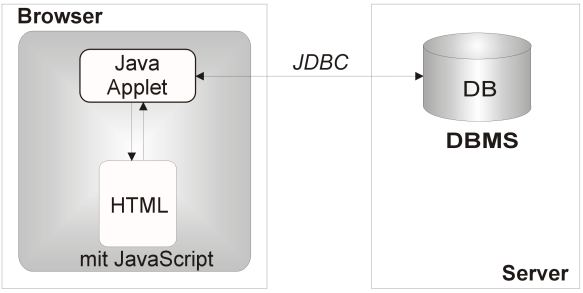

Abbildung 3.5: Datenbanken und Client Side Includes

4.) Server Side Includes (Abbildung 3.6): Hier kann der Server selbst mit Datenbankkommandos umgehen, wenn diese z.B. innerhalb von HTML-Dokumenten auftreten. Anfragen und Änderungen werden direkt in HTML ausgedrückt und über dem Server zur Verfügung stehende Zugriffsfunktionen ausgeführt.

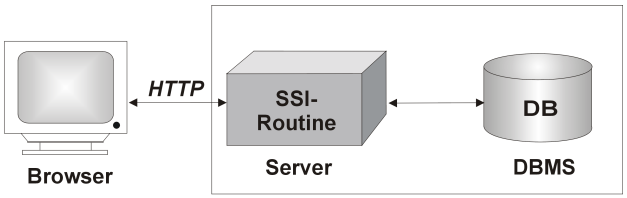

Abbildung 3.6: Datenbanken und Server Side Includes

5.) Datenbanksystem mit integriertem Webserver (Abbildung 3.7): Bei diesem Ansatz werden bestehende Datenbanksysteme um die Funktionalität eines HTTP-Servers erweitert. Die Problematik der Konfiguration von Gateways, von Transport und von gegebenenfalls notwendiger Transformation von Daten zwischen Datenbank- und HTTP-Server entfällt somit vollständig.

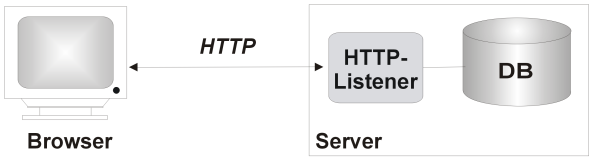

Abbildung 3.7: Datenbanken und integrierte Webserver

Alle vorgestellten Ansätze bieten ausreichende Performanz und Stabilität. Besonders die HTTP-basierten Konzepte scheint sich dabei für einen Einsatz im Rahmen von e-Learning-Systemen zu eignen, da hier eine einfache Anwenderschnittstelle für den Datenzugriff existiert, die keinerlei clientseitige Installation erfordert.

3.8 Datenbanken und Multimedia

Neben der weltweit ermöglichten Sicht auf Datenbanken über das WWW ist eine weitere Folge der neuen IuK-Technologien der vermehrte Einsatz multimedialer Dokumente. Die Verwaltung dieser Daten ge-

schieht bisher in vielen Fällen durch Speicherung in einfachen Dateisystemen. Gründe hierfür sind die besonderen Eigenschaften multimedialer Daten, die deren automatische Verwaltung erschweren: Sie sind meist binärer Natur, umfangreich und stellen Echtzeitanforderungen an die sie verwaltenden Systeme. Zur Einhaltung von Echtzeitbedingungen sind neue Speicherungsstrukturen und spezielle Zugriffsalgorithmen notwendig, der große Platzbedarf kann beispielsweise durch eine Ablage der Daten außerhalb der Datenbank gedeckt werden.

Viele klassische Datenbanksysteme bieten lediglich einige wenige Standarddatentypen für deren automatische Verwaltung und keine Möglichkeiten der Datenmanipulation. Abhilfe schaffen sog. **Multimedia-Datenbanksysteme**, die um Vorrichtungen für effiziente Speichervorgänge und Zugriffsmöglichkeiten multimedialer Daten erweitert wurden und darüber hinaus Funktionen zur Manipulation der Medien umfassen [Sub96, Sub98]. Die Kombination einer Multimedia-Datenbank mit WWW-basiertem Zugang für Verwaltung und Manipulation der multimedialen Daten wird in jüngster Zeit auch als *Web Content Management (WCM)* bezeichnet [Loe00]. Derartige WCM-Systeme verbinden somit durch die Kombination von Orts- und Plattformunabhängigkeit sowie der automatischen Wartung und Verteilung multimedialer Information innovative Ansätze, die auch im Hinblick auf e-Learning-Systeme sinnvoll sind.

3.9 Zusammenfassung

In diesem Kapitel wurden Vor- und Nachteile verschiedener Datenbanksysteme im Hinblick auf eine Verwendung innerhalb von e-Learning-Systemen verglichen und der Einsatz eines objektorientierten Systems motiviert. Die Kopplung von Datenbank und WWW wird inzwischen von allen großen Datenbank-Herstellern unterstützt und kann auf unterschiedliche, ähnlich performante Art und Weise durch HTTP-basierte Verfahren realisiert werden. Im Zusammenhang mit e-Learning-Systemen bietet sich eine Lösung an, für die eine einfache clientseitige Schnittstelle zur Verfügung gestellt werden kann. Der HTTP-basierte Ansatz verfügt über eine solche Schnittstelle und bietet dabei die für e-Learning-Systeme erforderliche Flexibilität zur Bearbeitung beliebiger Datenbankanfragen.

Kapitel 4

Informationssuche in e-Learning-Systemen

Die nach wie vor rasant fortschreitende Leistungssteigerung von Computern und deren zunehmende Vernetzung, die Entwicklung von neuen Trägersystemen und Protokollen sowie die kommerzielle Nutzung des Internets führt gegenwärtig dazu, dass weltweit immer mehr Informationen auf Computern verfügbar sind. Die effiziente Nutzung dieser Informationen setzt das Vorhandensein adäquater Möglichkeiten zur Suche in komplexen und multiplen Daten- und Medientypen voraus, jedoch zeigen sich klassische, auf textorientierten Techniken basierende *Retrieval*-Verfahren zur Indizierung komplexer Medien als nur bedingt geeignet [Kla99, Smi96c, Pen94].

Im Rahmen von e-Learning-Systemen stellt die Möglichkeit zur aktiven Erfragung von Informationen durch den Anwender eine zusätzliche Möglichkeit zur Vermittlung von Wissen dar. Die in diesem Fall zu durchsuchenden Daten sind in der Regel auf vielfältige Weise vernetzt, und Informationen werden unter didaktischen Gesichtspunkten auf unterschiedlichste Weise im System präsentiert. Daher gewinnen Techniken zur *inhaltsbasierten* Suche in verschiedenen Medientypen zunehmend an Bedeutung [Cro95, Kla99, Rod95, Smi99].

Vor diesem Hintergrund vermittelt das vorliegende Kapitel die Grundlagen der Informationssuche in e-Learning-Systemen. Dazu werden zunächst die erforderlichen Techniken zur Suche in verschiedenen Medientypen, insbesondere in Text- und Bilddaten, vorgestellt. Anschließend werden Aspekte der WWW-basierten Suche und der Suche in Datenbanken gegenübergestellt, die ähnliche Ausgangsbedingungen im Hinblick auf die zu indizierenden Dokumente aufweisen. Anhand von WWW-Suchmaschinen werden dazu zunächst Probleme der Erschliessung von komplex vernetzten, multimedialen und großen Datenmengen für eine effektive Informationssuche erörtert; daran anschließend wird gezeigt, dass ein Teil der aufgedeckten Schwächen durch den Einsatz eines Datenbanksystems vermieden bzw. reduziert werden kann.

4.1 Terminologie

Unter dem Begriff **Information Retrieval** werden EDV-gestützte Methoden zur gezielten Suche nach Informationen in einem — potenziell sehr großen — Datenbestand zusammengefaßt. Ein Kernproblem hierbei ist die **Indizierung**, unter der die inhaltliche Erschließung der Informationen mit dem Ziel des effizienten Wiederauffindens verstanden wird. Beim **textbasierten Indizieren** werden relevante Begriffe aus allen Dokumenten mit Angabe des Dokumentennamens in der **Indextabelle** oder dem **Index** abgelegt. Dies beschleunigt den Zugriff auf die gewünschte Information, da ein gesuchter Begriff nicht mehr in allen Dokumenten, sondern nur im Index gefunden werden muß. Die Verwendung spezieller Datenstrukturen zur Implementierung von Indextabellen und effizienter Suchalgorithmen wirkt hierbei unterstützend.

Unter dem Begriff **Information-Retrieval-System (IR-System)** wurde in der Vergangenheit eine Software zur Verwaltung und Suche überwiegend unformatierter Textdaten variabler Länge verstanden.

Bedingt durch den zunehmenden Einsatz anderer Datentypen wird inzwischen in diesem Zusammenhang anstelle von „Textdaten“ von „multimedialen Daten“ gesprochen und es entstanden Begriffe wie *Text Retrieval*, *Image Retrieval*, *Video Retrieval* oder auch *Audio Retrieval*, die alle Gegenstand aktueller Forschung sind [Als98, Leh00, You97].

WWW-Suchmaschinen, auch als *Search-Engine* oder *Information Broker* bezeichnet, sind spezielle IR-Systeme, die für den internetbasierten Einsatz entwickelt wurden und mit automatisierten Methoden das WWW durchsuchen. Sie bestehen in der Regel aus einem Indizierungs– und einem Retrievalprogramm und werden durch eine Benutzeroberfläche ergänzt. Neben einfachen Suchmaschinen existieren *Meta-Suchmaschinen*, die keinen eigenen Index besitzen, sondern den Index anderer Suchmaschinen nutzen, *Spezialsuchmaschinen*, die nur einen speziellen Bereich der im Internet verfügbaren Informationen indizieren, beispielsweise nur Dokumente einer bestimmten Sprache oder einer Domäne vorgegebenen Namens, sowie *Kataloge*, bei denen es sich um hierarchisch gegliederte Sammlungen von Texten eines bestimmten Wissensgebiets handelt [Cro95].

4.2 Bewertung von Suchergebnissen

Ein Problem der Suche in großen lokalen Datenbeständen und insbesondere im WWW ist die zumeist große Zahl an Resultaten, die als Ergebnis einer Suchanfrage geliefert wird [Kla99]. Sie macht eine Sortierung der Ergebnisdokumente nach ihrer Relevanz erforderlich, die ein Maß für die Güte der Suche ist. Hierzu haben sich verschiedene Verfahren zur Sortierung der Fundstellen (*Relevance Ranking*) etabliert. Beim *Coordination Level Match* ist die Relevanz beispielsweise die absolute Häufigkeit des Suchbegriffs in dem Ergebnisdokument, während das *Best Match*-Verfahren die Ähnlichkeit eines Vergleichsdokuments mit dem Ergebnisdokument zur Berechnung von dessen Relevanz verwendet. Die Definition der Ähnlichkeit erfolgt hierbei in Abhängigkeit vom gesuchten Medientyp. Wird der Anwender in die Bewertung des Suchprozesses einbezogen, beipielsweise zur Trennung guter von weniger guten Suchergebnissen, so wird auch von **Relevance Feedback** gesprochen [Wil93].

Die am häufigsten verwendeten Gütemaße des Retrievals sind **Recall** und **Precision** [Poe99]. Der Recall

$$R = \frac{A}{B} \quad ; \quad (0 \leq R \leq 1) \tag{4.1}$$

stellt ein Maß für die Vollständigkeit des Retrieval-Ergebnisses dar, die Precision

$$P = \frac{A}{C} \quad ; \quad (0 \leq P \leq 1) \tag{4.2}$$

ist ein Indikator für die Fähigkeit des Systems, nicht relevante Dokumente auszuschließen [Sal87]. Hierbei bezeichnet A die Anzahl der *nachgewiesenen* relevanten Dokumente, B die Anzahl aller relevanten Dokumente, und C die Anzahl aller nachgewiesen Dokumente. Der Recall beschreibt somit das Verhältnis der korrekterweise gefundenen relevanten Dokumente zu den insgesamt im System vorhandenen relevanten Dokumenten. Die Precision gibt dagegen das Verhältnis der korrekterweise gefundenen relevanten Dokumente zu den insgesamt gefundenen an. Die beiden Maße sind in gewisser Weise gegenläufig. Zur Erläuterung zeigt Abbildung 4.1 beispielhaft einen Recall-Precision-Graph für Suchanfragen an ein IR-System.´Für jede Anfrage wurden hierbei jeweils der Recall und die Precision berechnet und diese Tupel im Diagramm eingezeichnet. Zur Verdeutlichung der Abhängigkeit zwischen Recall und Precision ist die Betrachtung der beiden „Extremfälle“ hilfreich: Wenn alle existierenden relevanten Dokumente auf eine Suchanfrage hin zurückgeliefert werden, so ist der Recall gleich 1. Wenn zwar alle relevanten Dokumente zurückgeliefert werden, aber darüber hinaus noch viele andere, d.h. C ist größer als B, ist die Precision niedrig. Wird umgekehrt nur ein einziges Dokument aus einer Menge B relevanter Dokumente gefunden,

so ist der Recall sehr schlecht, die Precision kann aber hoch sein, sofern nur dieses einzige Dokument zurückgeliefert wurde. Allgemein gilt, dass sich bei einer Verkleinerung der Antwortmenge durch eine spezifischere Anfrage eine bessere Precision, aber ein schlechterer Recall ergibt. Dagegen wird bei einer Vergrößerung der Antwortmenge durch eine allgemeinere Anfrage ein größerer Recall, gleichzeitig aber eine kleinere Precision erzielt.

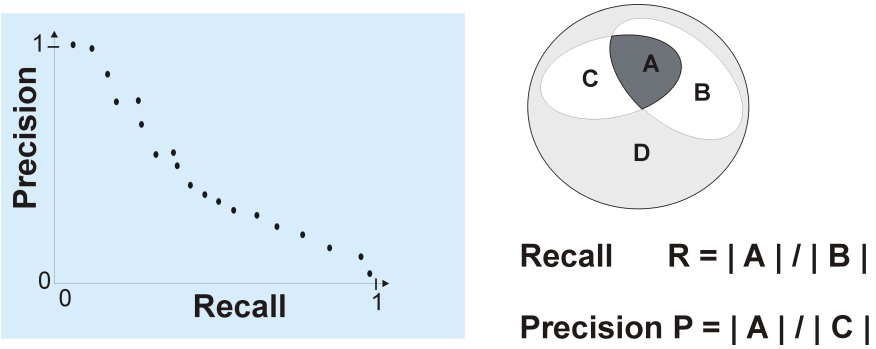

Abbildung 4.1: Recall-Precision-Graph zur Gütemessung von IR-Systemen

4.3 Multimediale Suche

Verfahren der **multimedialen Suche** befassen sich mit dem Auffinden von Daten eines bestimmten Typs in multimedialen Informationsmengen [Fuh00, Mel00, Rod95]. Ein Schlüsselproblem in diesem Zusammenhang, für das bislang keine generell etablierten Lösungsansätze bestehen, ist die Indizierung multimedialer Daten. Es lassen sich jedoch einige grundlegende, gemeinsame Konzepte erkennen, die im Folgenden anhand von text- und bildbasierten Verfahren vorgestellt werden. Zur Entwicklung eines allgemeinen Ansatzes zur multimedialen Informationssuche, der insbesondere für e-Learning-Systeme geeignet ist, werden abschliessend Möglichkeiten der Informationssuche in anderen Medientypen skizziert.

4.3.1 Textbasierte Suche

Abbildung 4.2 (a) zeigt die prinzipielle Vorgehensweise beim **Indizieren eines Textes** in klassischen IR-Systemen, Abbildung 4.2 (b) veranschaulicht analog die Textindizierung in WWW-Suchmaschinen. Die Angaben (1) bis (5) bzw. (0) bis (7) bezeichnen einzelne Arbeitsschritte.

Aus einem Textdokument wird dabei durch einen Text-Parser die Liste der enthaltenen Wörter ermittelt (1). Anschließend werden vorgegebene Zeichen (Signs) und sog. Stoppwörter wie Artikel, Konjunktionen oder Präpositionen mit Hilfe spezialisierter Filter aus dieser Liste entfernt (2,3). Für jedes verbleibende Wort wird die Anzahl seines Vorkommens im Text festgehalten (4) und anschließend werden beide Angaben in den Index übertragen (5).

Der Index von WWW-Suchmaschinen wird rekursiv von einem sog. *Robot* aufgebaut. Hierbei handelt es sich um eine Software, die aufgrund einer vorgegebenen URL[1] aktiv das zu indizierende HTML-Dokument ermittelt (0) und dessen Text an den Text-Parser weiterleitet (1). Während die eigentliche Indizierung die in klassischen IR-Systemen eingesetzten Methoden verwendet (Schritte 2-5), extrahiert der Parser hier zusätzlich alle Verweise auf weitere HTML-Dokumente (6). Die entsprechenden URLs werden dem Robot mitgeteilt (7), der wiederum die zugehörigen HTML-Dokumente ermittelt (0).

[1] URL - Uniform Resource Locator.

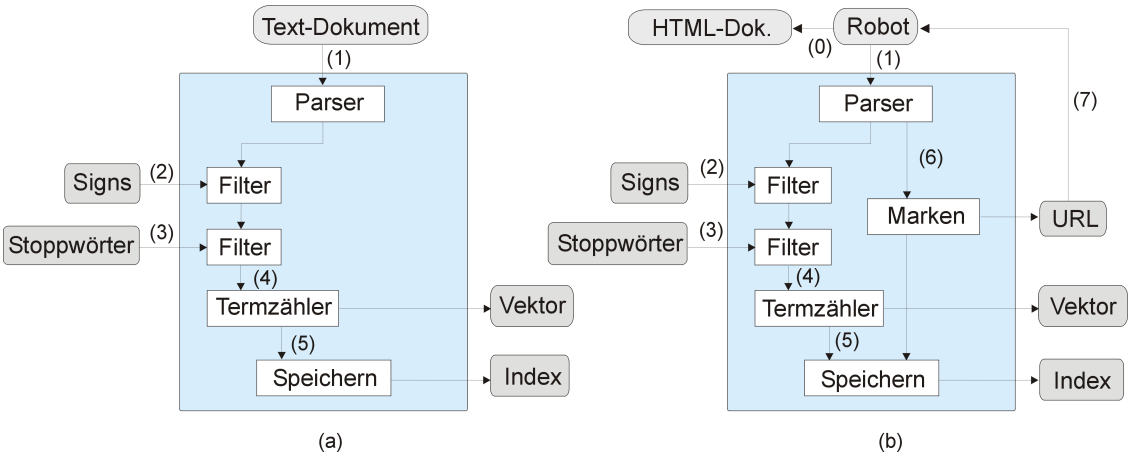

Abbildung 4.2: Textindizierung in IR-Systemen (a) und WWW-Suchmaschinen (b)

Beim **Text-Retrieval** wird der vorgegebene Suchbegriff im Index gesucht und die zugehörige Information zurückgeliefert. Im einfachsten Fall ist dies eine Liste der Dokumentennamen bzw. der URLs, die den gesuchten Begriff enthalten. Dies führt aber nicht in allen Fällen zu zufriedenstellenden Retrieval-Ergebnissen. Gründe hierfür liegen insbesondere in der Vielschichtigkeit und Komplexität der natürlichen Sprache, die beispielsweise die Verwendung von Homonymen und Synonymen zuläßt, und aus deren Morphologie sich häufig zahlreiche Variationsmöglichkeiten von Worten und Suchbegriffen ergeben. Diesen Umständen wird durch erweiterte Suchmöglichkeiten nachgekommen, beispielsweise durch die Verknüpfung verschiedener Suchbegriffe mittels Bool'scher Operatoren, oder durch die Angabe sog. *Wildcards*, die es ermöglichen, nur einen Teil des Suchbegriffs fest vorzugeben.

4.3.2 Bildbasierte Suche

Durch die zunehmende Bedeutung von Bildern zur Darstellung von Information und Vermittlung von Inhalten („*Ein Bild sagt mehr als 1000 Worte*“) kommt der bildbasierten Suche innerhalb des Gebiets des Information Retrievals eine wichtige Bedeutung zu. Aufgrund der gänzlich anderen Datenstruktur sind die Indizierung und das Retrieval von Bildern mit den eben skizzierten Verfahren nicht möglich. Zahlreiche IR-Systeme bieten allerdings die Möglichkeit, Bilder manuell durch das Hinzufügen von Schlagworten oder Annotationen zu indizieren, um so zumindest eine textbasierte Suche über diese Zusatzinformation vornehmen zu können. Dieses Vorgehen ist im allgemeinen jedoch mit hohem manuellen Aufwand verbunden und weder vollständig noch eindeutig [Dah00].

Abhilfe versprechen sog. **Image Retrieval Systeme**, die eine automatische Indizierung durch den Einsatz von Verfahren aus dem angrenzenden Forschungsgebiet der Bildanalyse versuchen [Bre99, Car00, How98, Lip97]. Die Bildanalyse befaßt sich generell mit der Extraktion von Bildmerkmalen mit dem Ziel der Objekterkennung [Nie90, Hab95, Jäh97]. Das Problem des Retrievals in Bildern läßt sich damit auf die Frage zurückführen, wie signifikante Merkmale aus Bildern extrahiert und in einem Index abgelegt werden können. Abbildung 4.3 skizziert den allgemeinen Aufbau eines **bildbasierten Information Retrieval Systems**.

Im ersten Schritt werden aus jedem abzuspeichernden Bild signifikante Merkmale extrahiert und in Form eines Merkmalvektors s in der Indextabelle des Systems abgelegt. Suchanfragen erfolgen in der Regel über die Vorgabe eines Suchbildes, für das ebenfalls der Merkmalsvektor v berechnet und mit den im Index abgelegten Vektoren verglichen wird. Ein solches Vorgehen wird als *Query by Example* bezeichnet. Die Bestimmung der Ähnlichkeit zu allen im Index abgelegten Merkmalsvektoren erfolgt unter

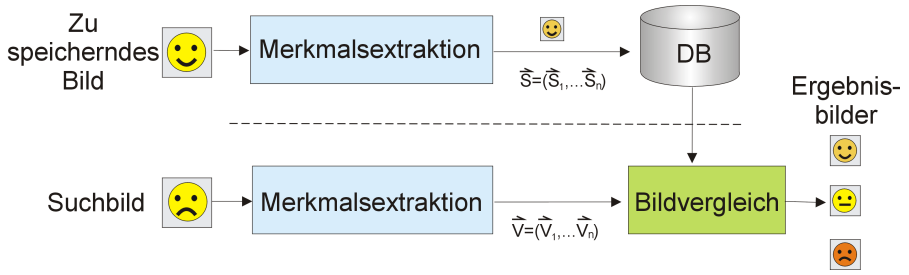

Abbildung 4.3: Aufbau eines Image Retrieval Systems

Verwendung geeigneter *Abstandsmaße*, deren Wert um so geringer ist, je „ähnlicher“ sich zwei Bilder sind [Hab95]. Image Retrieval Systeme verwenden häufig Merkmale, die auch bei der visuellen Wahrnehmung des Menschen eine bevorzugte Stellung einnehmen. Hierzu gehören zum Beispiel *Farb-*, *Textur-*, *Form-*, sowie *Kantenmerkmale* [Jai96, Rub97]). Merkmale können dabei in unterschiedlichen Repräsentationen oder *Signaturen* [Rub98] zur Indizierung eines Bildes beitragen. Abbildung 4.4 veranschaulicht dieses. Beispielsweise können im Falle von Texturen die ersten n Koeffizienten der Wavelet- oder der Fourier-

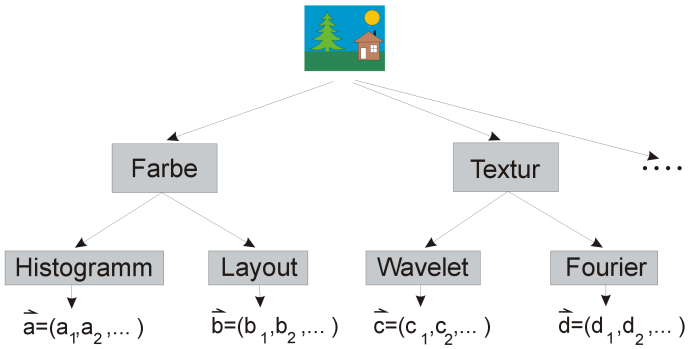

Abbildung 4.4: Berechnung von Bildsignaturen nach [Mit97]

Transformation als Repräsentationen der Textur gespeichert werden [Mit97]. Für das Merkmal ’Farbe’ ist die Wahl eines geeigneten Farbraumes besonders wichtig, da nach [Dah00] nur in perzeptiven Farbräumen eine Entsprechung von visuell wahrnehmbaren Farbunterschieden und geometrischen Farbdistanzen vorliegt.

Hinsichtlich der Merkmalsextraktion werden **globale** und **lokale** Merkmalsrepräsentationen unterschieden [Nie90]. Im ersten Fall wird die Signatur basierend auf sämtlichen Pixeln eines Bildes berechnet, im zweiten Fall erfolgt die Berechnung auf der Basis nur eines signifikanten Bildausschnittes.

Die Berechnung lokaler Repräsentationen eignet sich nach [Sch96] besonders für eine Anwendung in der Nähe signifikanter Bildinformation innerhalb kleinerer Bildregionen und liefert gute Retrieval-Resultate. Eine globale Vorgehensweise eignet sich dagegen für Anwendungen, die durch das Vorliegen einfach aufgebauter, wenig strukturierter Bilder mit homogen Bildbereichen gekennzeichnet sind.

Zur Berechnung der Ähnlichkeit zweier Signaturen $\mathbf{x}, \mathbf{y} \epsilon R^n$ werden **Abstandsmaße** eingesetzt. In der Literatur existieren hierzu zwar zahlreiche Vorschläge (siehe z.B. [Nie90]), die Frage nach dem für eine bestimmte Anwendung optimalen Abstandsmaß entzieht sich jedoch oft einem systematischen Zugang und wird daher im allgemeinen heuristisch beantwortet [Mit97]. Weit verbreitete Maße zur allgemeinen

Bestimmung des Abstandes $d(x, y)$ zweier Punkte $\mathbf{x}, \mathbf{y} \epsilon \mathbf{R}$ entstammen der Familie der durch

$$d_{L_p}(\mathbf{x}, \mathbf{y}) \quad := \quad (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{P}} \tag{4.3}$$

definierten $\mathbf{L_p Metriken}$. Am häufigsten werden die als *City-Block-Metrik* bekannte L_1-Metrik

$$d_{L_1}(\mathbf{x}, \mathbf{y}) \quad := \quad \sum_{i=1}^{n} |x_i - y_i| \tag{4.4}$$

und der Euklidsche Abstand

$$d_{L_2}(\mathbf{x}, \mathbf{y}) \quad := \quad \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4.5}$$

eingesetzt [Nie83, Swa91]. Weitere Vertreter geometrischer Abstandsmaße sind die χ^2-Metrik

$$d_{\chi^2}(\mathbf{x}, \mathbf{y}) \quad := \quad \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i} \tag{4.6}$$

und die Cosinus-Metrik

$$\alpha(\mathbf{x}, \mathbf{y}) \quad := \quad \arccos \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||}, \tag{4.7}$$

welche den Winkel zwischen den Merkmalsvektoren $\mathbf{x}$ und $\mathbf{y}$ berechnet. Dabei ist $||\mathbf{x}||$ die L_2-Norm des Vektors $\mathbf{x}$.
Für Bildsignaturen, die statistische Verteilungen repräsentieren, können auch statistische Abstandsmaße wie die **Bhattacharyya Metrik**

$$d_{Bhattacharyya}(\mathbf{x}, \mathbf{y}) \quad := \quad -ln \sum_{i=1}^{n} \sqrt{\frac{x_i y_i}{|\mathbf{x}| \cdot |\mathbf{y}|}} \tag{4.8}$$

oder die **Hellinger Metrik**

$$d_{Hellinger}(\mathbf{x}, \mathbf{y}) \quad := \quad 1 - \sum_{i=1}^{n} \sqrt{\frac{x_i y_i}{|\mathbf{x}||\mathbf{y}|}}, \tag{4.9}$$

verwendet werden, mit $|x|$ als L_1-Norm des Vektors $\mathbf{x}$:

$$||\mathbf{x}|| = \sum_{i=1}^{n} |x_i|. \tag{4.10}$$

Im Falle zweier Signaturen $\mathbf{x}, \mathbf{y} \epsilon R^n$ kann auf diese Art also der Abstand zwischen diesen beiden Vektoren bestimmt werden. Ein kleiner Wert bedeutet hier eine große Ähnlichkeit der entsprechenden Bilder hinsichtlich dieser bestimmten Merkmalsrepräsentation. Für den gesamten Vergleich zweier Bilder hinsichtlich verschiedener Merkmale werden z.B. gewichtete Summen über alle berechneten Distanzmaße gebildet.

Der Anwendungskontext, in dem Image Retrival durchgeführt wird, spielt für die Auswahl der Verfahren für Signaturberechnung und Abstandsberechnung eine wesentliche Rolle. So wird beispielsweise

in [Bac96] belegt, dass ein allein auf Texturmerkmalen basierendes Ähnlichkeitsmaß im Kontext natürlicher Bilder gute Resultate erzielt, wohingegen es für Anwendungen in der medizinischen Bildgebung weniger geeignet ist. In [Mit97] wird argumentiert, dass sich eine kleine Menge spezieller Merkmale für eine weitaus größere Menge möglicher Anwendungsbereiche eignet als ein einziges Merkmal.

Die Effizienz von Image Retrieval Systemen wird zumeist anhand der auch beim Text Retrieval verwendeten Größen Recall und Precision bewertet (vgl. Gl. 4.2 und 4.2). Einige Systeme verwenden auch die aus der medizinischen Statistik stammenden Größen von *Sensitivität SE* und *Spezifität SP* [Dah00, Leh97]. Während die Sensitivität exakt dem Recall entspricht, wird die Precision hier als *Vorhersagewert PV* bezeichnet und die Spezifität durch

$$SP \quad = \quad \frac{D}{D+E}, (0 \leq SP \leq 1) \tag{4.11}$$

als neue Größe definiert. Hierbei bezeichnet D die Anzahl der korrekterweise nicht zurückgelieferten Bilder, und E die Anzahl aller fälschlicherweise zurückgelieferten Bilder. Die Spezifität berücksichtigt somit im Gegensatz zur Precision auch die absolute Größe der Ausgangsmenge, was in bestimmten Fällen durchaus sinnvoll kann.

4.3.3 Suche in weiteren Medien

Die wachsende Verfügbarkeit multimedialer Informationen im Internet läßt auch die Notwendigkeit für Werkzeuge zur Suche in anderen Medien, wie Video- oder beispielsweise Audiodaten, entstehen. Da der Schwerpunkt der vorliegenden Arbeit jedoch auf der Suche in Text– und (statischen) Bilddaten liegt, soll an dieser Stelle nur ein kurzer Überblick über Verfahren des Video- und Audioretrieval gegeben werden. Weiterführende Informationen zur videobasierten Suche finden sich in [Smi99, Lev00, Ard97], audiobasierte Verfahren sind unter anderem in [Mel00, You97, Ng98] beschrieben.

Die bisherige Vorgehensweise zur Informationssuche in klassischen Datenbanksystemen, die Video- und Audiodaten beinhalten, besteht auch bei diesen Medien in der Regel in einer rein textuellen Indizierung und Suche. Gegenwärtig werden jedoch unterschiedliche Alternativen im Bereich des **Videoretrievals** untersucht, so zum Beispiel die textuelle Indizierung von Filmmaterial anhand vorhandener Untertitel, wie sie etwa in Nachrichtensendungen existieren oder in klinischen Radiologien [Lin98, Leh00]. Dazu wird aus den einzelnen Video-Frames, die nach festen Zeitintervallen aufgeteilt sind, enthaltener Text segmentiert, interpretiert und anschließend nach konventionellen textuellen Verfahren indiziert. In diesem Fall reduziert sich die Aufgabe auf ein Image und Textretrieval Problem, das ein rein textbasiertes Retrieval ermöglicht.

Die gleichzeitige Verarbeitung von Bild- und Audiodaten eines Videofilms kann beispielsweise wie im System „MoCA“ der Universität Mannheim erfolgen [Lin98, Fis95]: Hier wird die Bildsequenz zunächst in zeitlich aufeinanderfolgende Frames unterteilt, wobei jeder Frame aus einem Bild und dem zugehörigen Audioframe besteht. In einer syntaktischen Analyse werden Merkmale für jeden Bild- und Audioframe berechnet. Als Grundlage zur Berechnung von Filmschnitten (Wechsel zwischen Szenen) werden für jedes Bild Farbhistogramme berechnet und die zeitlich aufeinanderfolgender Frames verglichen. Die Standardabweichung der Farbwerte Rot, Grün, Blau dient zur Erkennung monochromer Bilder. Zur Berechnung von Bewegung innerhalb eines Videos werden die Bildframes in Regionen unterteilt und die Farbstatistiken von Regionen aufeinanderfolgender Frames verglichen. Die Summe der Differenzen aller Segmente dient dann als Indikator für den „Grad“ der Bewegung. Auch Kamera- und Objektbewegungen können so unterschieden werden. Die verwendeten Merkmale zur Indizierung der Audio-Daten sind Amplituden- und Frequenzstatistiken.

Eine semantische Analyse interpretiert die gewonnenen Merkmale. Die erkannten Filmschnitte dienen beispielsweise zur Unterteilung eines Videos in Szenen, die Bewegungsstatistiken werden zur Bestimmung von Kameraschwenks verwendet, und anhand der Farbhistogramme können wiederkehrende

Objekte, beispielsweise Logos, erkannt werden. Die während der syntaktischen Analyse berechneten Amplituden- und Frequenzstatistiken der Audio-Frames dienen zur Unterscheidung von Sprache, Musik, Rauschen und Stille. Durch eine automatische **Klassifikation** wird ein Video im System „MoCa“ schließlich einer von fünf möglichen Klassen (*Nachrichten, Autorennen, Tennis, Werbung und Zeichentrickfilm*) zugeordnet. Diese sind durch die manuelle Profilanalyse einer entsprechend großen Stichprobenmenge ermittelt worden. Die verwendeten Klassifikationsalgorithmen sind aufgrund ihrer Komplexität bisher nicht echtzeitfähig.

Ein Beispiel eines reinen **Audio Retrieval** Systems im WWW ist „SMILE“, das von der IMS Research Group der Universität Padua entwickelt wurde [Mel00, smile]. Die Benutzerschnittstelle besteht aus einem in Java implementierten virtuellen Keyboard, das es Anwendern ermöglicht, eine gesuchte Tonfolge direkt in das System „einzuspielen“.

Ein Vorteil von Retrieval Systemen für gesprochene Sprache ist die hohe Robustheit gegen Wortfehler. Nach [Wit97] wurde in einer umfangreichen Studie nachgewiesen, dass Sprachdokumente mit einer Wortfehlerrate von 30% den Recall eines Retrievalsystems um 4% reduzieren, bei einer Fehlerrate von 50% erhöht sich dieser Wert nur auf 10%.

4.4 WWW-basierte Suche

In den vorangegangenen Abschnitten wurden Techniken zur Suche in bestimmten Medientypen vorgestellt. In diesem und dem folgenden Abschnitt wird nun auf vom Medientyp unabhängige Gesichtspunkte der Informationssuche eingegangen. Dazu wird innerhalb dieses Abschnitts zunächst erarbeitet, welche Schwierigkeiten bei der Informationssuche über WWW-Suchmaschinen auftreten können. Aus der offenen und verteilten Struktur hypertextbasierter Dokumente im Internet ergeben sich für Aufgaben der Informationssuche eine Reihe grundlegender Probleme:

- **Aktualität und Erreichbarkeit der Suchergebnisse**: Web-Seiten werden ständig verändert, hinzugefügt oder auch entfernt. Dies macht eine kontinuierliche Aktualisierung des Index erforderlich, was jedoch aufgrund der Größe des WWWs sehr zeitaufwendig ist. Aus diesem Grund ist der Index von Suchmaschinen niemals ganz aktuell und Veränderungen an Dokumenten können nur mit einer gewissen Verzögerung berücksichtigt werden. Darüberhinaus verfügt kaum eine Maschine über eine Verwaltung früherer Versionen von Dokumenten bzw. der Historie indizierter Dokumente [Met94]. Durch Netzüberlastung, die sich aufgrund der hohen Anzahl anfragender Nutzer zu bestimmten Zeiten ergibt, sind WWW-Suchmaschinen außerdem nicht immer erreichbar.

- **Vollständigkeit**: Die Anzahl der im WWW vorhandenen Dokumente wächst zu schnell, um vollständig von Suchmaschinen erfasst werden zu können[2]. Auch existiert neben statischen Dokumenten eine große Zahl dynamischer Seiten, die auch als *Deep Web* bezeichnet werden. Beim Deep Web handelt es sich um Seiten, die typischerweise aufgrund einer Benutzeranfrage dynamisch aus Datenbanken generiert werden. Nach einer Studie des US-amerikanischen Startup-Unternehmen „BrightPlanet“ ist die Anzahl der Seiten im Deep Web um den Faktor 500 größer als die Zahl der statisch verfügbaren Seiten [bright].

- **Vielfalt von Daten- und Inhaltstypen**: Derzeitige Suchverfahren können die Indizierung nicht auf bestimmte Dokumententypen oder -quellen beschränken [Her95].

- **Suchoptionen und Relevanz von Ergebnissen**: Die meisten Suchmaschinen bieten zur Verfeinerung von Suchanfragen nur eingeschränkte Möglichkeiten. Es fehlen differenziertere Methoden zur Suche nach Metadaten, die Berücksichtigung zeitlicher Beschränkungen oder eine nutzergesteuerte

[2] Die in dieser Hinsicht beste Maschine „Google“ indiziert beispielsweise 30% der gesamten statischen Webseiten [google].

Sortierung der Suchergebnisse. Als Folge ungenauer Indizierung und Abfragemöglichkeiten ist der Precision-Wert beim Retrieval oftmals sehr gering.

- **Nutzerprofil**: Klassische WWW-Suchmaschinen wissen nichts über das Profil ihrer Anwender. Dadurch können weder individuelle Ansprüche verschiedener Nutzergruppen berücksichtigt noch eine individuelle Unterstützung bei der Formulierung von Suchanfragen geboten werden.
- **Browsing-Verzeichnisse**: Es gibt keine wirklichen Inhaltverzeichnisse, in die alle Webseiten eingetragen werden können. Zwar existieren Kataloge, doch muß deren Inhalt manuell gepflegt werden und ist deshalb unvollständiger als der Index klassischer Suchmaschinen.

Suchprogramme der sog. „2. Generation“ versuchen hier zumindest teilweise Abhilfe zu schaffen [Coh00]. So ermitteln sie beispielsweise die Relevanz gefundener Dokumente durch Berücksichtigung weiterer Kriterien wie Kontext, Popularität oder Anzahl der Seiten, die auf das gefundene Dokument verweisen, und erzielen so einen bedeutend höheren Recall als herkömmliche Verfahren. Doch gerade die Probleme, die mit der offenen, dezentralen Struktur des Internets und der exponentiell anwachsenden Informationsmenge zusammenhängen, sind weiterhin ungelöst.

4.5 Datenbankbasierte Suche

Datenbanken zeichnen sich im Gegensatz zu WWW-Suchmaschinen (und auch im Gegensatz zu klassischen IR-Systemen) durch viele Eigenschaften aus, die eine Informationssuche sinnvoll unterstützen und dabei einige der im vorangegangenen Abschnitt erläuterten Probleme gezielt vermeiden können (vgl. Kapitel 3). Im Folgenden werden diese im direkten Vergleich zu klassischen WWW-Suchmaschinen bewertet:

- **Aktualität und Erreichbarkeit**: Ein DBMS kontrolliert alle Änderungen am Datenbestand und aktualisiert einen Index unmittelbar in Folge einer Änderung am Datenbestand. Dadurch existiert das Problem der fehlenden Aktualität und Erreichbarkeit indizierter Dokumente nicht.
- **Vollständigkeit**: In Datenbanken beträgt der Anteil indizierter Dokumente 100% bzw. liegt in der Verantwortlichkeit des Entwicklers.
- **Vielfalt von Daten- und Inhaltetypen**: In Datenbanken besteht über das vorhandene Datenmodell die Möglichkeit der gezielten, differenzierten Indizierung und Suche, auch untergliedert nach bestimmten Dokumententypen oder beliebigen anderen Eigenschaften, die innerhalb der Datenbank modelliert sind.
- **Suchoptionen und Relevanz der Ergebnisse**: Durch die Datenbankstruktur bzw. die einzelnen ansprechbaren Attributfelder verschiedener Datentypen ist deren Retrieval, auch durch Boolesche Verknüpfungen, direkt durch klassische Datenbankabfragen lückenlos möglich. In herkömmlichen WWW-Dokumenten existieren dagegen keine zusätzlichen Informationen in Form von „Attributen“ oder nur in geringem Umfang als „Zustandsinformation“ im Dokumenten-Header [Coh00].
- **Nutzerprofil**: Bei Vorhandensein eines Benutzermodells (vgl. Abschnitt 2.4.2) existieren zahlreiche Möglichkeiten, das Retrieval zu individualisieren und so die Güte von Suchergebnissen zu verbessern.
- **Browsing-Verzeichnisse**: Durch eine zugrunde liegende Datenbank innerhalb eines Informationssystems existieren automatisch auch ein „Inhaltsverzeichnis“, sowie ggfs. weitere Sichtweisen auf den gesamten Datenbestand. Damit steht ein vollständiger Katalog aller existierenden Dokumente zur Verfügung. Im Gegensatz zu WWW-Katalogen wird dieser bei jeder Änderungen durch die Autoren automatisch mitgepflegt, ist somit also stets konsistent und vollständig.

Bezüglich der Datenverwaltung besitzen datenbankbasierte e-Learning-Systeme charakteristische Eigenschaften des WWWs, basieren aber im Gegensatz zu diesem auf einer strukturierten, kontrollierbaren Datenmenge. Es bietet sich daher an, die Mächtigkeit des WWW als Präsentationsmedium und Konzepte der datenbankbasierten Suche zu einer effizienten Informationssuche in e-Learning-Systemen zu verknüpfen.

4.6 Zusammenfassung

Zur Unterstützung der Lernenden, die sich in komplexen, tiefverzweigten und multimedialen Inhalten eines e-Learning-Systems orientieren müssen, ist eine Komponente zur Informationssuche hilfreich. Diese Komponente muß komplexe Suchmöglichkeiten bieten, die sich über unterschiedliche Medientypen erstrecken, schnell und einfach im Gebrauch sind, qualitativ vollständige und präzise Ergebnisse liefern und Suchergebnisse aufschlußreich darstellen. In den vorangegangenen Unterabschnitten wurden einzelne Techniken zur Realisierung einer solchen Komponente vorgestellt. Dabei wurde zunächst auf spezifische Ansätze zur Suche nach bestimmten Medientypen eingegangen. Unabhängig vom Medientyp wurden danach generelle Aspekte der Informationssuche behandelt, die es bei der Realisierung einer solchen Komponente zu berücksichtigen gilt.

Basierend auf diesen erarbeiteten Aspekten hat sich in diesem Kapitel gezeigt, dass e-Learning-Systeme durch ihre multimedialen, vernetzten Inhalte, das WWW als Präsentationsmedium und eine zugrunde liegende Datenbank sehr gute Voraussetzungen für eine effiziente Informationssuche bieten.

Teil II

Stand der Technik

Kapitel 5

e-Learning- und IR-Systeme in der Praxis

Im ersten Teil dieser Arbeit wurden die theoretischen Grundlagen von e-Learning- und IR-Systemen entwickelt. Aufbauend auf den dort erarbeiteten Konzepten schließt sich im zweiten Teil der Arbeit eine Beschreibung konkreter Systeme an. Dazu werden in den folgenden Unterabschnitten verschiedene Beispiele von e-Learning-Systemen vorgestellt, die momentan im praktischen Einsatz sind. Da nach derzeitigem Kenntnisstand bisher kein e-Learning-System existiert, das eine Komponente zur multimedialen, speziell bildbasierten Informationssuche beinhaltet, werden im zweiten Abschnitt zusätzlich einige allgemeine Informationssysteme beschrieben, die sich mit dieser speziellen Problematik befassen.

5.1 e-Learning-Systeme in der Praxis

Im Hinblick auf die folgenden Beispiele von e-Learning-Systemen handelt es sich um Systeme speziell aus der medizinischen Lehre, in deren Umfeld auch die hier vorliegende Arbeit entstanden ist. Die beschriebenen Systemkonzepte lassen sich genauso aber auch in Systemen anderer Anwendungsbereiche [Han00, Jah00, Fer00] einsetzen. Aus der Vielzahl interessanter medizinischer Systeme wurden folgende Beispiele ausgewählt: **Docs 'n Drugs** der Ulmer Poliklinik, **CASUS/ProMediWeb** der Münchener und Düsseldorfer Universitätskooperation, **CAMPUS** des Heidelberger Klinikums und schließlich der Würzburger **D3 Trainer**, der auf Methoden der Künstlichen Intelligenz basiert. In Anlehnung an die Abschnitte 2.3 bis 2.5 werden folgende Kriterien untersucht:

- Aufbau des Systems,
- Inhalte, Benutzerunterstützung und Didaktik,
- Inhalteerstellung und
- Systemvoraussetzungen.

5.1.1 Docs 'n Drugs

Das Projekt "Docs 'n Drugs - Die virtuelle Poliklinik“ ist Teil des Förderprogrammes „Virtuelle Hochschule“ des Landes Baden-Württemberg [Mar99]. Das gemeinsam von Universität, Fachhochschule und Universitätsklinikum Ulm entwickelte System wird derzeit im Medizinstudium und in medizinbezogenen Studiengängen eingesetzt und soll langfristig auch die medizinische Fort- und Weiterbildung unterstützen.

Aufbau des Systems
Den technischen Aufbau des serverzentrierten Systems zeigt Abbildung 5.1.1. Die Hauptkomponente

besteht aus dem System zur Ablage und Verwaltung aller Inhalte. Sie umfaßt das sog. Lehrprozeßmodell und das Falldatenmodell. Im Falldatenmodell werden alle Inhalte gespeichert. Diese bestehen aus dem systematischen Basiswissen, das unabhängig von Fällen abgelegt ist, das speziell einem Fall zugeordnete Fallwissen und den Informationen über Beziehungen zwischen diesen beiden Wissensarten. Das Lehrprozeßmodell wird vom Autor eingesetzt, um die Struktur eines Falles festzulegen, diesem Inhalte zuzuordnen und Relationen zum systematischen Wissen zu erzeugen. Alle Daten werden in einer relationalen Datenbank abgelegt. Über JDBC kommunizieren insgesamt drei Java-basierte Anwendungen für Administrator, Autoren und Anwender mit dem Basissystem.

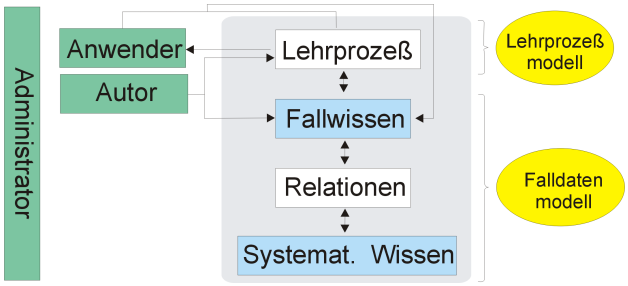

Abbildung 5.1: Systemarchitektur des Docs ’n Drugs-Systems nach [Sei99]

Inhalte, Benutzerunterstützung und Didaktik
Das System präsentiert graphisch aufbereitete Fälle aus der medizinischen Praxis, mit denen die ärztliche Entscheidungsfindung und das Erkennen von Handlungserfordernissen trainiert werden soll. Die inhaltliche Basis bildet dabei das im Falldatenmodell strukturierte und formalisiert abgelegte medizinische Wissen. Bei der Bearbeitung eines Falles bewegt sich der Anwender durch einen Baum sog. „Situationen“, die jeweils mit „Lehrtasks“, beispielsweise in Form einer zu beantwortenden Frage, abgeschlossen werden. Das System protokolliert während einer Fallbearbeitung die Daten bereits bearbeiteter Situationen in einer „Befundmappe“ und stellt sie dem Anwender zum Nachschlagen zur Verfügung. An didaktischen Konzepten werden interaktive und multimediale Elemente zur Veranschaulichung komplexer medizinischer Sachverhalte, Abfragemodule und verschiedene Schwierigkeitsgrade bei der Fallbearbeitung geboten. Fälle können geführt, halbgeführt und ungeführt durchlaufen werden. Das Lehrsystem kann sowohl im Lernmodus als auch im Prüfungsmodus eingesetzt werden. Im Lernmodus werden Kommentare, Hyperlinks und Hilfen angeboten, im Prüfungsmodus werden diese Mittel ausgeblendet. Docs ’n Drugs besteht insgesamt somit aus einem Simulationssystem mit Komponenten aus den Bereichen Präsentation und Drill & Practice.

Inhalteerstellung
Für die Aufbereitung eines Falles durch einen Fachautor existieren einzelne, lokal aufzurufende Programme. Die Erfassung des Fallwissens erfolgt über eine Java-Applikation, die Drehbucherfassung über eine HTML- und Javascript-basierte Oberfläche. Ein Editor zur Festlegung eines Fallablaufs liegt bisher nur prototypisch als Java Applikation vor. Durch den Zugriff auf das Falldatenmodell werden Verweise auf systematische Inhalte generiert. Hinsichtlich der didaktischen Konstruktion von Fällen ist es den Autoren selbst überlassen, wie komplex und mit welchen tutoriellen Kommentaren und Hilfestellungen sie diese versehen.

Systemvoraussetzungen
Das e-Learning-System wurde für Windows-Betriebssysteme entwickelt und setzt clientseitig einen Pentium-Klasse-Prozessor mit mindestens 200 MHz, mindestens 64 MB RAM für die Benutzer- und 96 MB für die Autorenschnittstelle, mindestens 80 MB Festplattenspeicher und einen schnellen Internetzugang voraus. Die Benutzerschnittstelle erfordert den Netscape Navigator, außerdem müssen Java und Quick-

time Plugin installiert sein. Zur Benutzung der im Lehrsystem integrierten Applikationen, die während einer Arbeitssitzung über das Netz auf den Clientrechner übertragen werden, sind darüber hinaus sog. „Policies“ einzurichten, die den Applikationen den Zugriff auf den Server erlauben.

5.1.2 CASUS/ProMediWeb

Das in der Münchener AG INSTRUCT in Zusammenarbeit mit weiteren Universitäten entwickelte System CASUS ist ein „fallorientiertes multimediales Lern- und Autorensystem für die Ausbildung von Medizinstudenten und Ärzten zur Verbesserung der Ausbildung in der Medizin durch die Realisierung problemorientierten Lernens“ [Fis98]. Dozenten der Medizin bringen mittels CASUS Fälle ihrer medizinischen Praxis, die ihnen für die Lehre relevant erscheinen, didaktisch strukturiert auf den Computer.

Aufbau des Systems
Abbildung 5.2 zeigt des Aufbau des CASUS/ProMediWeb-Systems. Das serverzentrierte System setzt sich aus dem CASUS Autorensystem für Macintosh, dem Java-basierten Player zum Abspielen von Fällen und einer Oracle-Datenbank zusammen, darüber hinaus existiert eine Schnittstelle zur Anbindung weiterer Datenbanken.

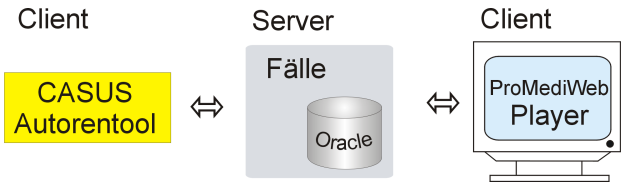

Abbildung 5.2: Systemarchitektur des CASUS/ProMediWeb-Systems

Inhalte, Benutzerunterstützung und Didaktik
Casus/ProMediWeb orientiert sich am problemorientierten Lernansatz. Dem Nutzer werden derzeit etwa 40 Lernfälle aus verschiedenen medizinischen Fachgebieten präsentiert, denen kein systematisches Wissen zugrunde liegt. Zur Benutzerunterstützung wurden eine Hilfefunktion und ein Benutzerhandbuch implementiert. CASUS/ProMediWeb kann als Kombination aus einem Simulationssystem mit Drill- & Practice - Komponente angesehen werden.

Inhalteerstellung
Das auf dem Produkt Toolbook basierende Autorensystem CASUS ermöglicht es dem Autor, ohne vorausgesetzte Programmierkenntnisse Lernkarten mit Text, Bildern und Videos zu gestalten und in ein vorgegebenes, didaktisches Format zu bringen. Generiert werden können zudem Fragen an den Nutzer mit 6 verschiedenen Antworttypen, Expertenkommentare und Antwortkommentare.

Systemvoraussetzungen
Für das Autorensystem wird momentan ein Macintosh-Rechner vorausgesetzt, das Java-basierte Abspielmodul für Fälle erfordert auf Anwenderseite einen Webbrowser mit Java, Quicktime und JavaScript-Fähigkeit.

5.1.3 CAMPUS

Ziel des Heidelberger Projekts „CAMPUS - Computerunterstützte Aus- und Weiterbildung in der Medizin durch plattformunabhängige Software“ war die Entwicklung einer Lehr- und Lernsystemshell, die mit einem herkömmlichen WWW-Browser zur Aus- und Weiterbildung in der Medizin genutzt werden kann [Sin99].

Aufbau des Systems
Die Grundkomponenten des CAMPUS-Systems und deren Funktionen sind in Abbildung 5.3 dargestellt.

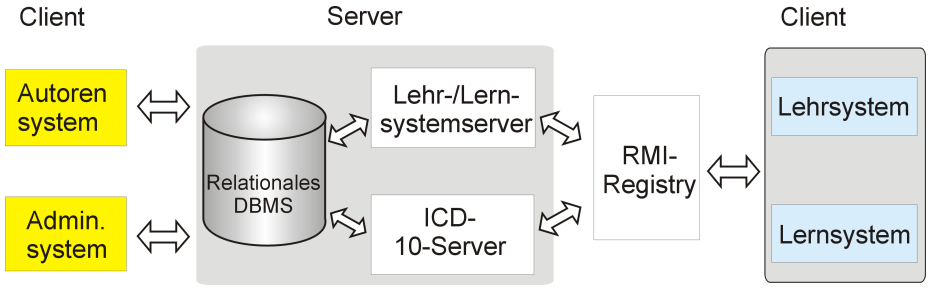

Abbildung 5.3: Systemarchitektur des CAMPUS-Systems

Das serverzentrierte System umfaßt ein Autorensystem, ein Administrationssystem, ein Lehr- und ein Lernsystem und eine zentrale Wissensbasis. Das Lehrsystem ist für eine Anpassung des Systems an die Nutzer gedacht, das Lernsystem stellt die graphische Benutzerschnittstelle zur Fallpräsentation und Fallsuche dar. Mittels des ICD-10[1]- Servers kann auf Verschlüsselungen für Erkrankungen zugegriffen werden. Alle Komponenten wurden in Java unter Einsatz einer relationalen Datenbank realisiert.

Inhalte, Benutzerunterstützung und Didaktik
In CAMPUS werden sowohl systematisches Wissen als auch medizinische Fallsimulationen abgelegt. Der Nutzer soll eine Fallsimulation durchspielen, die dem ärztlichen Handeln im klinischen Alltag entspricht. Dabei kann ein Fall erst abgeschlossen werden, wenn vorgegebene Arbeitsschleifen durchlaufen wurden. Das Lehrsystem ist für die Adaptivität an den Nutzer verantwortlich, das heißt, es werden sämtliche Aktivitäten des Nutzers protokolliert und mit Hilfe eines Punkteschemas bewertet. Je nach Punktestand wird der Benutzer auf unterschiedliche Niveaus gehoben, die bestimmten Schwierigkeitsstufen der Fall- und Fragensammlung entsprechen. Das Lernsystem zur Fallbearbeitung ist geführt, halbgeführt oder linear benutzbar. CAMPUS vereint somit sowohl Komponenten aus dem Bereich der Simulationsprogramme, der Browsing-, Präsentations- und tutoriellen Systeme.

Inhalteerstellung
Das Autorensystem ermöglicht das Einbringen medizinischer Falldaten in das System und die Verwaltung des systematischen Wissens. Die Diagnosen der präsentierten Fälle werden ICD-10 - codiert abgelegt. Bereits vorhandene inhaltliche Bausteine werden über Listen zur Verfügung gestellt und können mehrfach verwendet werden. Einzelnen Komponenten können Expertenkommentare zugefügt werden, die während der Fallsimulation angezeigt werden. Systematisches Wissen wird mit Hilfe des kommerziell verfügbaren Softwareprodukts „Framemaker“ erzeugt und in XML gespeichert.

Systemvoraussetzungen
Clientseitig wird ein Internetzugang und ein Java-fähiger WWW-Browser vorausgesetzt.

5.1.4 D3 / D3 Trainer

Der mit Hilfe des am Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik der Universität Würzburg entwickelte D3/D3Trainer ist ein System, mit dem sich wissensbasierte Trainingssysteme als CD-ROM- oder WWW-basierte Variante generieren lassen, die das Trainieren diagnostischer Problemlösungstrategien unterstützen sollen. Erste medizinische Anwendungen wurden bereits realisiert und werden kommerziell vertrieben [Rei99].

[1] ICD-10: International Classification of Diseases.

Aufbau des Systems
Die Gesamtarchitektur zeigt Abbildung 5.4. Danach basiert das Gesamtsystem auf dem Experten-system-shell-baukasten D3, mit dem allgemeine Problemlöseumgebungen für diagnostische Probleme konfiguriert werden können. D3 beinhaltet die Komponente „Wissenserwerb“, mit der die Wissensbasis erzeugt und als Lisp-File abgelegt wird. Diese wird von der Komponente „Wissensnutzung“ verwendet, um die Falldaten, innerhalb eines Filesystems abgelegt, zu erstellen. Der D3Trainer ist der „Dialog“ bzw. die Schnittstelle für die Lernenden zum Zugriff auf die Falldaten. Um auf Benutzeraktionen reagieren zu können, umfaßt D3 schließlich noch das Expertensystem „XPS“, welches die hierzu gesammelten Benutzermerkmale in einer einfachen Datei verwaltet. Die Wissensbasis und die WWW-basierte Variante arbeiten serverzentriert.

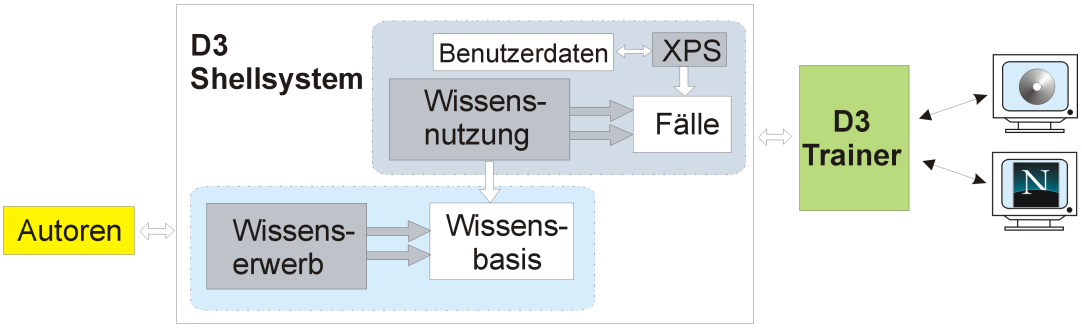

Abbildung 5.4: Systemarchitektur des D3 Trainers

Inhalte, Benutzerunterstützung und Didaktik
Jedes mit Hilfe des D3Trainers generierte System richtet sich in seinem inhaltlichen Aufbau nach den in Abschnitt 2.3.1 vorgestellten Intelligenten Tutoriellen Systemen. Das Wissensmodell, welches den mit dem D3Trainer erzeugten Systemen zugrunde liegt, umfaßt systematisches und problemorientiertes Wissen und unterstützt unter anderem Wissensarten wie Fallwissen, strukturelles und strategisches Wissen und Hintergrundwissen. Dieses wird unter Einsatz von Text und Bildern aufbereitet und ist durch Navigation über Schaltflächen und seltener Menü-Optionen erreichbar. Das Studentenmodell sieht die Berücksichtigung von Lernerpräferenzen vor und ermöglicht die explizite Adaptation der Systemoberfläche und teilweise auch der didaktischen Strategie an den Lernenden. Das didaktische Modell ist im Prinzip lernergesteuert, bietet aber einige didaktische Varianten an, bei denen das System den Benutzer stärker führt (geführte/ungeführte Fallbearbeitung). Dem Lernenden werden graphisch Fälle präsentiert, die er durch Erkennen von Symptomen in Bildern, durch Auswahl von Untersuchungen, Tests, Diagnosen und Therapien lösen muß. Dabei werden die verschiedenen Aktionen des Studenten mit den Schlussfolgerungen des Expertensystem verglichen, wodurch alle Teilgebiete der diagnostischen Problemlösung (Symptome erkennen, Nachfragen, Schlussfolgern) abgedeckt werden. Während jeder Aktion ist eine kontextbezogene Entscheidunghilfe abrufbar. Für die Bewertung einer Aktion durch das System stehen verschiedene Bewertungskategorien zur Verfügung, die durch differenzierte Rückmeldungen erklärt werden. Die mit dem D3Trainer erzeugten Systeme umfassen im Prinzip alle in Abschnitt 2.3 vorgestellten Systemtypen. Der Schwerpunkt liegt hierbei auf den Intelligenten Tutoriellen Systemen und den Simulationssystemen.

Inhalteerstellung
Die Entwicklung eines Trainingssystems besteht aus insgesamt drei Schritten: Zunächst muß ein Experte sein Wissen mit Hilfe des Shellbaukastens D3 formalisieren. Der zweite Schritt besteht im Aufbau einer Fallsammlung. Dazu gibt der Experte die Symptomatik des Falls an, bereitet den Fall durch zusätzliche Informationen auf, erfaßt die zur Symptomerkennung nötigen Bilder und verknüpft diese Bilder mit den Falldaten. Anhand des so formalisierten Wissens kann nun mit Hilfe des D3Trainers ein intelligentes

tutorielles System aufgesetzt werden.

Eine besondere Komponente von D3 ist das sog. Bildsystem. Ausgehend von einem Startbild kann ein Autor ein Netzwerk von Bildern und Texten aufbauen, indem er in einem Bild Regionen markieren und diese mit weiteren Bildern oder mit einem Textfeld verknüpfen. Genauso können auch in einem Textfeld Textbereiche markiert werden, die wiederum mit Bildern verknüpft sind. Die Erstellung eines Bildsystem erfordert nach [Rei99] allerdings sehr viel Zeit und Sorgfalt.

Systemvoraussetzungen
Die Hardwarevoraussetzungen für die Benutzung von D3 unter Microsoft Windows 9x/NT und Apple Macintosh MacOSx liegen bei 64 MB, der D3Trainer erfordert für die WWW-basierte Version einen Internetzugang und Webbrowser, für die CD-ROM Version sind keine zusätzlichen Voraussetzungen notwendig.

5.1.5 Zusammenfassung

Ein jedes der in den vorangegangenen Unterabschnitten beschriebenen Systeme liefert eine auf jeweils eine spezielle Fragestellung zugeschnittene Systemlösung. Je nach Schwerpunkt werden dabei eine oder mehrere der in Abschnitt 2.3 eingeführten Interaktionsformen zum Lernen angeboten, die Möglichkeit zur Ablage systematischen Wissens als zugrundeliegende Wissensbasis ist nicht in allen Systemen vorgesehen. Bezüglich der Erstellung und Bearbeitung multimedialer Lerneinheiten durch Autoren werden zum Teil kommerzielle Tools eingesetzt, die keine konkrete Unterstützung im Hinblick auf eine didaktische Aufbereitung und Verknüpfung verschiedener Inhalte bieten. Im Hinblick auf eine Wiederverwendung der entwickelten Systemsoftware ist nur das System D3 explizit darauf ausgerichtet, eine Generierung von Lernsoftware im Allgemeinen zu ermöglichen. Zwar bieten einige der beschriebenen Systeme Möglichkeiten der textbasierten Informationssuche, es existiert jedoch in keinem Fall, weder für Autoren noch für Benutzer, eine Unterstützung speziell im Hinblick auf eine in multimedialen Informationssystemen sinnvolle bildbasierte Suche.

5.2 Bildbasierte Informationssuche in der Praxis

Nach der Darstellung von Methoden zur bildbasierten Informationssuche in Abschnitt 4.3.2 werden in den folgenden Unterabschnitten zunächst zwei Systeme beschrieben, anhand derer mögliche Vorgehensweisen beim Image Retrieval erläutert werden sollen. Das erste System, „IRIS“, Ergebnis einer Kooperation zwischen der Universität Bremen und IBM, verwendet Farb-, Textur- und Konturmerkmale und kombiniert diese zur Objekterkennung mit einer regelbasierten Wissensbank. Das im System „IRMA“ der Universität Aachen entwickelte Vorgehen besteht dagegen aus einer Kaskade aufeinander aufbauender Verarbeitungsschritte ohne Festlegung konkreter Merkmale.
Abschließend werden im letzten Unterabschnitt einige weitere IR-Systeme skizziert.

5.2.1 IRIS

Nach [Als96] und [Dah00] ist das System „IRIS - Image Retrieval for Information Systems“ das einzige bislang verfügbare IR-System, das auf semantischer Ebene arbeitet und durch den Einsatz einer regelbasierten Wissensbank Bilder natürlicher Szenen automatisch textuell beschreibt. Dazu werden Methoden und Techniken aus den Bereichen der Bildverarbeitung und Künstlichen Intelligenz kombiniert.

Das generelle Vorgehen in IRIS ist in Abbildung 5.5 veranschaulicht und läßt sich wie folgt zusammenfassen: Zunächst werden durch drei unabhängig voneinander arbeitende Module die Merkmale Farbe, Textur und Kontur für verschiedene Bildsegmente des Bildes ermittelt. Dazu unterteilt jedes Modul das Originalbild zunächst in regelmäßige Segmente, die jeweils zu größeren Segmenten kombiniert werden,

sofern benachbarte Segmente einen ähnlichen Wert des jeweiligen Merkmals aufweisen. Diese Variante des „Split-and-Merge-Algorithmus“ [Nie90] resultiert in einer jeweils unterschiedlichen Segmentierung des Originalbildes, in dem jedes der enthaltenen Segmente textuell durch den Wert seines Merkmals beschrieben wird. Ein spezielles Objekterkennungsmodul, das auf einer regelbasierten Wissensbank agiert, kombiniert die textuellen Informationen derjenigen Segmente, die in bestimmten Nachbarschaftsbeziehungen zueinander stehen (z.B. die sich überlappen, sich berühren, sich umschließen oder enthalten) und deren verschiedene Merkmalswerte semantisch gesehen sinnvoll sind. So entsteht eine textuelle Beschreibung von komplexeren Objekten innerhalb des Bildes. Anschließend werden, wieder unter Anwendung semantischer Regeln, die textuellen Beschreibungen dieser Objekte zueinander in Beziehung gesetzt, was in der textuellen Gesamtbeschreibung des Bildes resultiert. Diese kann mit normalen textbasierten Verfahren indiziert werden.

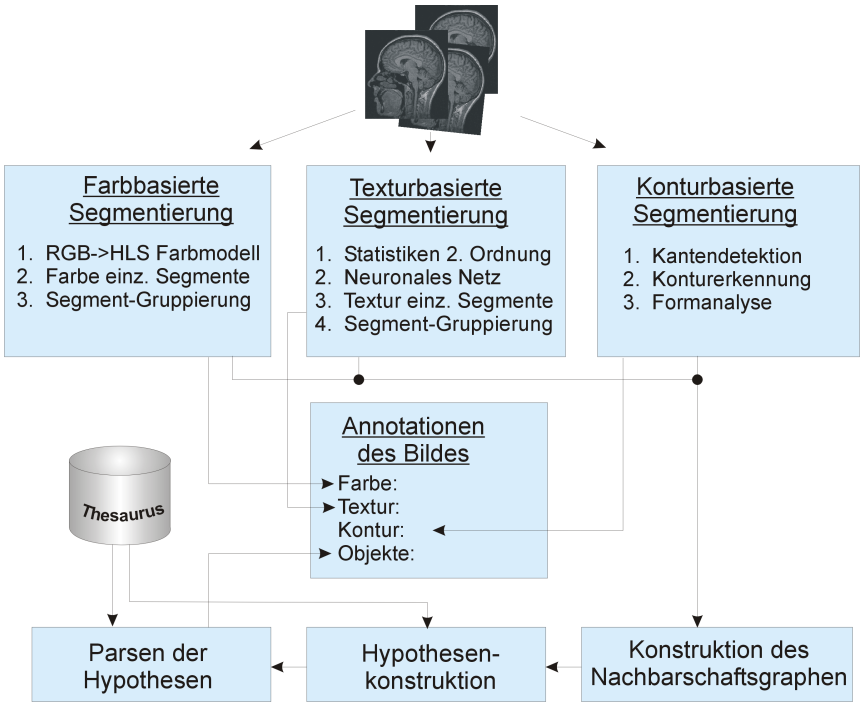

Abbildung 5.5: Image Retrieval mit IRIS nach [Her95]

Zur Farbmerkmalsbestimmung wird das Bild von RGB in das HLS Farbmodell transformiert (vgl. Abschnitt 4.3.2). Für jedes regelmäßige Segment des Bildes wird nun das Farbhistogramm berechnet und die dominante Farbe mittels eines einfachen Schwellwertverfahrens bestimmt. Segmente mit identischer dominanter Farbe werden anschließend gruppiert und das sie umgebende Rechteck berechnet. Zu jedem so entstandenen Segment werden Position, relative Größe und klassifizierte Farbe abgelegt.

Zur Bestimmung der Bildtextur werden für alle initialen Segmente des Bildes nach dem Haralick-Verfahren die Repräsentanten *Kontrast*, *Korrelation*, *Varianz* und *Entropy* in je vier Orientierungen bestimmt und anschließend, um unabhängig von der Orientierung zu sein, je Repräsentant der Mittelwert berechnet [Her95]. Alle Mittelwerte werden anschließend durch ein vorab trainiertes neuronales Netz in eine von acht möglichen Klassen klassifiziert. Diese sind *Himmel*, *Wolken*, *Sand*, *Wald*, *Gras*, *Stein*, *Schnee* und *Eis*. Segmente mit ähnlichen Texturen werden analog zum Vorgehen im erstgenannten Modul zu größeren Segmenten gruppiert, zu denen jeweils die Größe, die Dichte (das Verhältnis der jeweiligen Segmentgröße zur Summe aller anderen Segmentgrößen mit gleichem Klassifikationsergebnis) und die ermittelte Textur abgelegt wird.

Zur Konturbestimmung werden durch Faltung des Bildes mit der ersten Ableitung des Gauss-Filters zunächst relevante Bildkanten bestimmt und diese über ein Konturschlussverfahren geschlossen [Can86, Zha95]. Aus den so entstandenen primitiven Objekten werden durch einfache Formanalyse Repräsentanten wie Schwerpunkt, Fläche, u.ä. bestimmt.

Im Anschluss ermittelt ein eigens entwickelter Graph-Parser, der sich an das in [Lut89] beschriebene Vorgehen anlehnt, die globale Beschreibung des Gesamtbildes. Ausgangsbasis hierfür sind die von den Modulen zur Merkmalsextraktion gebildeten Segmente mit ihren individuellen Merkmalswerten. Die Bildbeschreibung wird in zwei Schritten ermittelt: Im ersten Schritt werden zunächst Hypothesen über die von der Konturbestimmung ermittelten primitiven Objekte gebildet. Diese Hypothesenbildung berücksichtigt für jedes primitive Objekt alle Segmente, die in einer bestimmten Nachbarschaftsbeziehung zu diesem Objekt stehen. Unter Einsatz der anfangs erwähnten regelbasierten Wissensbank erhält nun jedes primitive Objekt eine textuelle Beschreibung, die sich aus den Beschreibungen der beteiligten Segmente ergibt. Diese Beschreibung ist vorerst eine Hypothese.

Im zweiten Schritt werden nun die ermittelten textuellen Beschreibungen der primitiven Objekte zueinander in Beziehung gesetzt und es ergeben sich Beschreibungen für komplexere Objekte. In deren Verlauf werden einige der Hypothesen über primitive Objekte verworfen, andere werden bestätigt, und bilden so mit bestätigten Hypothesen über andere primitive Objekte die Gesamtbildbeschreibung. Die abschließende Indizierung ist textbasiert und ermöglicht wieder textbasierte Suchanfragen. Im Falle einer Query-by-Example-Anfrage durchläuft das Suchbild ebenfalls die beschriebene Verarbeitungskette, und die so gewonnene textuelle Beschreibung wird textbasiert mit den Bildbeschreibungen im Index verglichen.

5.2.2 IRMA

Ziel des Projekts „IRMA - Image Retrieval in Medical Applications“ ist die Entwicklung von Methoden des Image Retrievals, die semantische und formale Anfragen an ein medizinisches Bildarchiv ermöglichen. Unter semantischen Anfragen werden Suchvorgaben nach intra- und interpersonalen, krankheits- und variationsorientierten Informationen verstanden, formale Anfragen sind beispielsweise die nach der Untersuchungstechnik und den Aufnahmemodalitäten.

Medizinisches Bildmaterial besteht im Gegensatz zu Bildern anderer Domänen aus eher homogenen, kontrastschwachen Röntgen-, Ultraschall-, oder Magnetresonanzbildern, die somit keine Farbwerte und nur wenig Textur oder klar abgrenzbare Objekte enthalten. Aus diesem Grund sind bisherige Konzepte, deren Schwerpunkt auf einer Klassifikation der Merkmale Farbe, Textur, Form und Objektlage liegt, im Zusammenhang mit dem Image Retrieval medizinischer Daten nur bedingt anwendbar und erzielen nur unzufriedenstellende Werte für Recall and Precision [Dah00].

Der innerhalb des IRMA-Systems verfolgte Ansatz zur Realisierung eines medizinischen Image Retrieval Systems ist in Abbildung 5.6 verdeutlicht. Hier wird eine strikte Trennung und eindeutige Formalisierung von insgesamt sieben sequentiellen Schritte durchgeführt, die im Folgenden beschrieben werden [Leh00, Bre99].

- **Globale Bildbeschreibung**: Durch Auswertung globaler Bildmerkmale oder DICOM[2]-Informationen findet zunächst eine Kategorisierung des Bildmaterials nach Bildmodalität, Anatomie (Strukturen) und Aufnahmebedingungen statt. Die eingesetzten Verfahren zur Merkmalsberechnung sind dabei nicht festgelegt, was eine der wesentlichen Eigenschaften des Systems ist. Alle ermittelten Kategorien werden mit der jeweiligen Wahrscheinlichkeit der Zugehörigkeit gespeichert.

- **Bildregistrierung**: Zur Ermittlung der Aufnahmegeometrie eines Bildes werden die Registrierungsparameter in Bezug auf ein Referenzbild berechnet und die Transformationsmatrix ebenfalls als Bildinformation abgelegt.

[2]DICOM- Digital Imaging and Communications in Medicine.

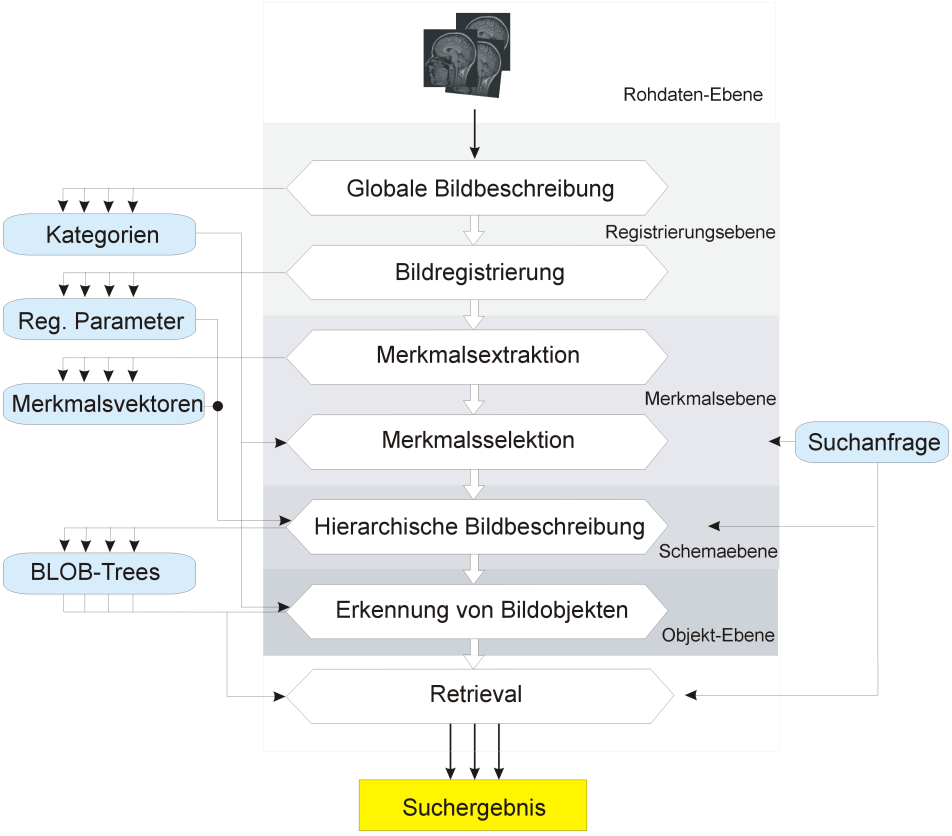

Abbildung 5.6: Image Retrieval mit IRMA nach [Leh00]

- **Merkmalsextraktion**: Für jedes Bild erfolgt die Berechnung verschiedener lokaler Merkmale pro Bildpunkt. Die Entscheidung, welche lokalen Merkmale berechnet werden, hängt dabei von der vorab ermittelten Bildmodalität ab. Pro Merkmal entsteht somit ein Bild, in dem jedes Pixel den lokalen Wert dieses Merkmals beschreibt. Die Merkmalsbilder werden ebenfalls zum Bild gespeichert.

- **Merkmalsselektion**: Basierend auf der anfänglichen Kategorisierung und — falls vorhanden — dem Kontext einer Suchanfrage werden die nicht relevanten Merkmalsbilder ausgeblendet. Beispielsweise spielen in Röntgenaufnahmen farbbeschreibende Merkmalskomponenten keine Rolle.

- **Hierarchische Bildbeschreibung**: Zur abstrakten Beschreibung eines Bildes werden aus den zugehörigen gebildet und hierarchisch angeordnet [Car00]. Dieser Vorgang erfolgt allein auf der Basis der vorhandenen Merkmale einzelner Bildpunkte. An dieser Stelle besteht jeder Blob aus einer Liste, die angibt, an welchen Pixelpositionen im Bild sich anatomische Strukturen mit welchen lokalen Merkmalen befinden.

- **Erkennung von Bildobjekten**: Die inhaltliche Zuordnung von Blobs zu benennbaren Strukturen (beispielsweise Organen) erfolgt durch die Berücksichtigung des in den vorherigen Schritten ermittelten a-priori-Wissens und des Vergleichs mit einem Atlas bereits kategorisierter Bilder. Der Atlas, der momentan noch in der Entwicklung ist, wird für jede Kategorie eine Liste passender Blobs samt der zugehörigen Merkmale enthalten. Mit Hilfe eines statistischen Klassifikators soll dann die Zuordnung durchgeführt werden. An dieser Stelle können auch Fehlklassifikationen erkannt werden.

Damit ist die Indizierung eines Bildes abgeschlossen, denn es wurde eine Klassifikation des Bildes in Form von Benennungen enthaltener Strukturen vorgenommen.

- **Retrieval**: Die Suche nach Bildern, die einem vorgegebenen möglichst ähnlich sehen, erfolgt durch einen Vergleich der hierarchisch organisierten Blobs. Dazu muß zunächst festgelegt werden, auf welcher Abstraktions- oder Auflösungsebene ein Vergleich stattfinden soll. Die anfangs durchgeführte Bildregistrierung erlaubt an dieser Stelle auch die benutzerseitige Markierung einer „Region of Interest“ (ROI) im registrierten Bild, auf der Anfragen bearbeitet werden sollen.

Der Vorteil des IRMA-Verfahrens besteht vor allem in der Möglichkeit, medizinische Suchanfragen auf Bildmaterial unterschiedlicher Modalität adaptiv und in verschiedenen Kontexten durchführen zu können. Dies läßt einen besonders guten Recall erwarten; eine diesbezügliche experimentelle Untersuchung steht jedoch noch aus.

5.2.3 Weitere IR-Systeme

Die Mehrzahl momentan verfügbarer IR-Systeme verwendet Farb-, Textur- und Konturmerkmale, sowie gegebenfalls noch deren strukturelle Anordung, die global für das Gesamtbild oder für ausgewählte Regions of Interests (ROI) ermittelt werden. Beispiele sind hier das **QBIC** (Query By Image Content System von IBM ([Nib93]) oder das System **MARS** (Multimedia Analysis and Retrieval System) der University of Illinois ([Meh97]). Bei letzterem liegt der Schwerpunkt auf Anfragen, die auf der gewichteten Kombination mehrerer Merkmale basieren.

Das System **Photobook** des Massachusetts Institute of Technology verwendet ebenfalls alle vier Merkmale [Pen94]. Dieses System umfaßt zahlreiche Abstandsmaße, die einzeln oder entsprechend kombiniert zum Einsatz kommen. Neu ist der Anfrageunterstützungsagent *FourEyes*, der den Benutzer bei der Auswahl von für ein Problem besonders geeigneten Merkmalen graphisch unterstützt.

Das an der Universität von Columbia entwickelte IR-System **VisualSeek** ([Smi96b]) beschränkt sich auf die Analyse von Farbe und deren räumliche Anordnung innerhalb des Bildes, wobei eine automatische ROI-Extraktion durchgeführt wird. **WebSeek** ist die WWW-basierte Erweiterung des Systems. Das **BlobWorld**-Projekt der University of California at Berkeley ist auf natürliche Bilder spezialisiert und arbeitet mit den Merkmalen Farbe und Textur, wobei besonders deren räumliche Zusammenhänge berücksichtigt werden. Die Extraktion der durch Ellipsen beschriebenen ROIs geschieht automatisch [Car00].

Das Fraunhofer Institut für graphische Datenverarbeitung (IGD) in Darmstadt arbeitet im Rahmen eines ESPRIT-Projekts derzeit an einer Image Retrieval Software namens **COBWEB** [Vol99], mit deren Hilfe eine inhaltsbasierte Suche in Bildarchiven unterstützt werden soll. Die zu archivierenden Bilder werden dazu zunächst auf ein einheitliches Format verkleinert und per diskreter Wavelet-Transformation relevante Bildregionen ermittelt (vgl. Abschnitt 4.3.2). Die nachfolgende Berechnung von Luminanz- und Chrominanz-Werten erfolgt nur auf diesen Bildregionen und führt zu einem „digitalen Fingerabdruck“. Beim Retrieval kann eine Skizze oder ein Bild vorgegeben werden, für das ebenfalls der Fingerabdruck berechnet wird. Anschließend erfolgt mittels geeigneter Distanzmaße der Vergleich dieses Abdrucks mit jedem einzelnen der im Bildarchiv gespeicherten Bilddaten. Textindizes sind mit diesem Verfahren nur noch für Daten wie Datum, Ort, Name des Photographen, etc. notwendig, auf eine textuelle Bildbeschreibung wird vollkommen verzichtet.

Im System **Surfimage** des Institut National de Recherche en Informatique et en Automatique in Frankreich wurde ein inhaltsbasiertes Image Retrieval auf Basis einer flexiblen Kombination verschiedener Repräsentationen aller vier Merkmale realisiert [Mit97]. Auf diese sind jeweils verschiedene Distanzmaße anwendbar, deren Resultate zu einem globalen Abstandsmaß kombiniert werden. Eine Besonderheit ist das integrierte Relevance Feedback, mit dem Anwender durch Einteilung der Suchergebnisse in positive und negative eine weitere, verfeinerte Suche starten können.

Ein ähnliches, WWW-basiertes graphisches Verfahren zum Relevance Feedback innerhalb der Ergebnismenge, das *Multidimensional-Scaling (MDS)* wird auch innerhalb des Systems **IRS** eingesetzt [Rub97, Rub98]. Dieses System verwendet zur Merkmalsextraktion nur verschiedene globale Repräsentationen des Merkmals Farbe, läßt sich prinzipiell aber auch auf andere Merkmalstypen übertragen. Die Ähnlichkeit zwischen Bildern wird mittels des eigens entwickelten Abstandsmaßes *Earth Mover's Distance (EMD)* berechnet.

Teil III

Neue Lösungskonzepte und ihre Realisierung

Kapitel 6

Der eLS-Baukasten zur Generierung und Verwaltung von e-Learning-Systemen

Im ersten Teil der vorliegenden Arbeit wurden konzeptuelle Grundlagen und aktuelle Techniken von e-Learning-Systemen vorgestellt, und es wurde aufgezeigt n, in welchen Bereichen Defizite und Ansätze für Verbesserungen existieren. Des weiteren wurden aus dem Bereich des multimedialen Information Retrievals Grundlagen und derzeitige Ansätze beschrieben. Zur Illustration wurden anschließend im zweiten Teil der Arbeit einige konkrete e-Learning- und IR-Systeme vorgestellt.

Im dritten Teil der Arbeit sollen nun die gewonnenen Erkenntnisse beider Bereiche in neue Lösungsansätze umgesetzt werden. Die Zielsetzung besteht dabei in der Implementierung einer Software zur Erstellung von e-Learning-Systemen, die Schwächen und Defizite derzeitiger Systeme vermeiden. Als Schlüssel dazu wird sich die Entwicklung eines umfassenden Konzepts zur automatischen und komfortablen Generierung und Wartung von e-Learning Systemen durch den **eLS-Baukasten** erweisen. Das vorliegende Kapitel beschreibt dazu dessen komponentenbasierte Gesamtarchitektur und der durch ihn generierbaren e-Learning-Systeme, das zugrundeliegende Datenmanagement und Wissensmodell, und erläutert die für die Generierung auszuführenden Schritte.

6.1 Architektur des eLS-Baukastens

Mit dem eLS-Baukasten ist es möglich, ohne technisches Spezialwissen auf effiziente Weise neue e-Learning-Systeme zu generieren. Abbildung 6.1 zeigt dessen Aufbau.

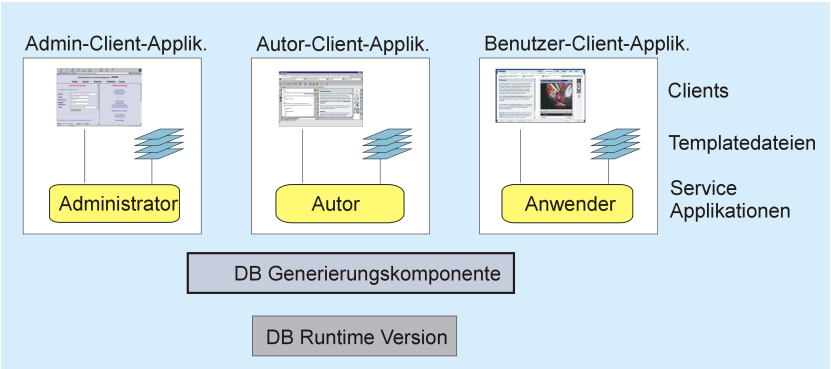

Abbildung 6.1: Architektur des eLS-Baukastens

Der eLS-Baukasten besteht aus einer 3-schichtigen Architektur. Die erste Schicht umfaßt die Runtime Version des Datenbanksystems ObjectStore [Obj97, Obj98a]. Die zweite Schicht besteht aus der Generierungskomponente für die Datenbasis, die für das Aufsetzen einer initialen ObjectStore-Datenbank zuständig ist. Diese Komponente bildet die Basis eines jeden neu generierten e-Learning-Systems und kommt genau einmal zu Beginn eines neuen Projekts zum Einsatz. Die dritte Schicht besteht aus Client-Applikationen, die auf die Datenbank zugreifen. Jede Client-Applikation setzt sich aus drei Komponenten zusammen: Die Service Applikation ist — hier zunächst vereinfacht ausgedrückt – für die Kommunikation zwischen Datenbank und Webserver zuständig und bedient sich dazu einer Menge von Templates, die als Übertragungsmedium zwischen Client- und Service Applikation genutzt werden. Die dritte Komponente ist der eigentliche Client, d.h. die Anwenderschnittstelle für Administratoren, Autoren und Benutzer.

Zur Generierung eines neuen e-Learning-Systems müssen der eLS-Baukasten auf einen Rechner aufgespielt, die Generierungskomponente zur Erzeugung einer initialen Datenbank ausgeführt und anschließend die drei Service Applikationen gestartet werden. Einzelheiten hierzu werden in Abschnitt 6.6 erläutert.

6.2 Architektur generierter e-Learning-Systeme

Der eLS-Baukasten generiert e-Learning-Systeme, die eine Client-Server-Architektur aufweisen. Abbildung 6.2 zeigt den typischen Aufbau dieser Systeme.

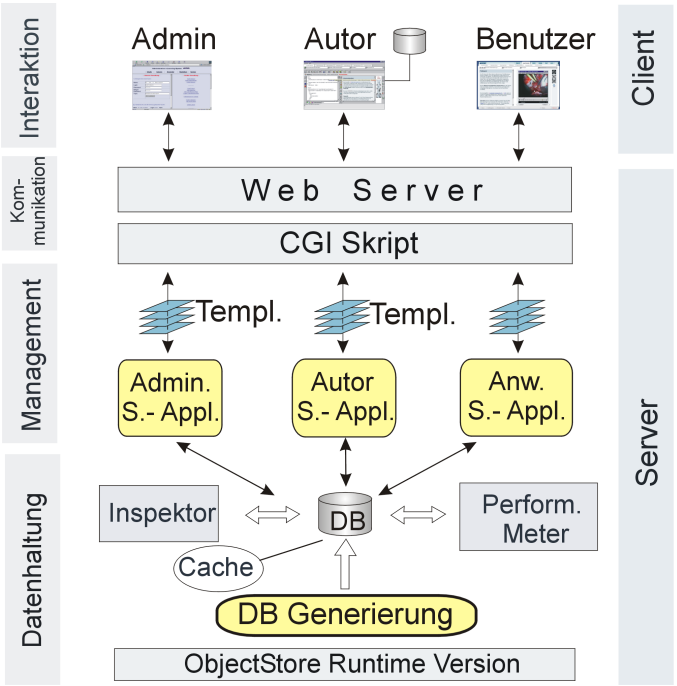

Abbildung 6.2: Architektur generierter e-Learning-Systeme

Der „Rumpf“ eines jeden e-Learning-Systems ist dabei in Form mehrerer Server-Applikationen auf einem (oder auch mehreren) Servern installiert und von Clients über das Netz erreichbar. Die insbesondere zur Datenhaltung erforderliche Rechenleistung wird hauptsächlich von den Server- Applikationen benötigt, die deshalb auf leistungsfähigen Maschinen laufen sollten. Die Client-Applikationen stellen dagegen keine nennenswerten Anforderungen an die Rechenleistung ihrer Hostrechner. Diese bewährte Architektur

gestattet den parallelen Zugang vieler Benutzer bei gleichzeitiger zentraler Administration der Server.

Bei den generierten e-Learning-Systemen werden drei Typen von Clients unterschieden, die den drei im oberen Teil von Abbildung 6.2 angedeuteten Komponenten für Administratoren, Autoren und Benutzer entsprechen. Für jede dieser Gruppen steht eine speziell auf deren Aufgabenbereich zugeschnittene Client-Applikation zur Verfügung. Der Administrator-Client dient zur Pflege des e-Learning-Systems und der Verwaltung von Benutzerrechten. Für Autoren steht ein Autoren-Client zur Verfügung, über den Lerninhalte eingegeben und modifiziert werden können. Zusätzlich wird auf dem Client eine Logdatei gepflegt, die jegliche Aktivitäten eines Autors protokolliert. Der Benutzer-Client ist der Zugang für Lernende und zuständig für die Präsentation multimedialer Lerninhalte und die Protokollierung benutzerabhängiger Aktionen.

Die Client-Applikationen für Administrator und Benutzer können direkt über einen Webbrowser angesprochen werden. Der Administrator-Client basiert auf HTML-Formularen, über die Verwaltungsinformationen eingegeben und editiert werden können. Die Schnittstelle des Benutzer-Clients ist unter Verwendung von HTML, Java- und Javascript realisiert. Beim Autor-Client handelt es sich dagegen um eine pure Java-Applikation. Somit sind alle Client-Applikationen weitgehend plattformunabhängig einsetzbar. Genauere Einzelheiten zu allen Clients folgen in Kapitel 7.

Serverseitig ist jeder der drei Client-Applikationen eine Server-Applikation zugeordnet, welche die Anfragen der Clients entgegennimmt und entsprechende Aktionen auf der Datenbank initiiert. Jede Server-Applikation arbeitet als eigener Prozess, konkurrierende Datenbankzugriffe werden durch das Transaktionsmanagement des Datenbankservers koordiniert. Sofern die Server-Applikationen auf demselben Rechner installiert sind, wird zur Verbesserung der Zugriffsgeschwindigkeit ein Datenbank-Cache für alle Applikationen verwendet. Befinden sich die Server-Applikationen aus Gründen der Lastverteilung auf verschiedenen Rechnern, erhöht sich die Anzahl der verwendeten Caches entsprechend (vgl. Abschnitt 6.3.2). Die Kommunikation zwischen Client und dem zugeordneten Server wird über einen Webserver abgewickelt. Dies wird anhand eines konkreten Beispiels in Abschnitt 6.4.1 beschrieben.

Die grundlegende Komponente eines jeden generierten e-Learning-Systems ist die ObjectStore-Datenbank (vgl. Abschnitt 6.3). Ihre primäre Aufgabe ist die Speicherung der Lerninhalte. Auch werden administrative Daten über Autoren und Benutzer gespeichert, die zur Verwaltung und zur Benutzermodellierung des e-Learning-Systems benötigt werden. Praktisch alle Aktivitäten der Server-Applikationen führen zu Aktionen auf dieser Datenbank. Zusätzlich sind zwei Werkzeuge, der Inspektor und der Performance Meter, zur Überwachung der Datenbank und der Kommunikation mit den Service Applikationen in jedem generierten e-Learning-System verfügbar.

Die modulare Architektur des eLS-Baukastens ermöglicht die Generierung flexibler e-Learning-Systeme, die auf unterschiedlichen Lernparadigmen basieren können. Die Festlegung eines Lernparadigmas des in Abschnitt 2.3 eingeführten *didaktischen* sowie *technischen* Lernsystemtyps wird auf diesem Abstraktionsniveau des eLS-Baukastens nicht fest vorgegeben, sondern bestimmt sich erst durch die Komponenten „DB-Generierungskomponente“, der Client-Applikationen für die Autoren und Benutzer, und insbesondere durch die von den Autoren vorgegebene Struktur und Organisation des Wissens. Ein zukünftig notwendiger Austausch bzw. eine Anpassung der Applikationen an veränderte Anforderungen ist aufgrund des komponentenbasierten Charakters des eLS-Baukastens ohne technischen Aufwand möglich und erfordert lediglich die Überprüfung und Anpassung der direkten Schnittstellen zu anderen Komponenten. Darüberhinaus können einzelne Komponenten aus dem System gelöst und in anderen Systemen wiederverwendet werden.

Insgesamt liegt mit jedem e-Learning-System ein Anwendungs-Framework vor, das sich an das sog. *Komponentenparadigma* anlehnt [All98, Gri98]. Ein nach diesem Paradigma konzipiertes System bietet unter Erfüllung der Forderung nach Autonomie und Wiederverwendbarkeit eine performante und leicht wartbare Kombination implementierter sowie standardisierter Komponenten und ist dadurch offen auch

für Systemerweiterungen oder -veränderungen.

6.3 Das Datenbanksystem ObjectStore

In den eLS-Baukasten ist das Datenbanksystem ObjectStore eingebettet, dessen im Hinblick auf die besonderen Anforderungen von e-Learning-Systemen wichtigsten Eigenschaften in diesem Abschnitt beschrieben werden.

6.3.1 Motivation

In Kapitel 3 waren anhand der Gegenüberstellung relationaler und objektorientierter Konzepte die Vorteile eines objektorientierten Ansatzes hervorgehoben worden. Zur Verwendung in dem zu realisierenden eLS-Baukasten wäre prinzipiell jedes der in Kapitel 3.5 erwähnten objektorientierten Systemen geeignet. Gründe, die für einen Einsatz der Datenbank-Programmiersprache ObjectStore der Firma Object Design Inc. sprechen, sind die Kompatibilität zu allen gängigen Plattformen, die Verfügbarkeit von C++ -, Java- und ActiveX-Schnittstellen und eine Multi-Client- /Multi-Server-Architektur, die sich durch einen leistungsfähigen Datenbankserver, eine schlanke WWW-Schnittstelle, sowie eine spezielle Bibliothek zum Multimedia-Management auszeichnet. Die Vorteile von ObjectStore werden im Folgenden anhand einer Beschreibung von Architektur und Arbeitsweise genauer herausgearbeitet. Einen weiterführenden Überblick liefern [Obj97, Obj98a, Vos99] oder die unter [ostore] verfügbaren *White Paper*.

6.3.2 Technische Konzepte

Multi-Client- /Multi-Server-Architektur

ObjectStore ist eine komponentenbasierte Anwendungsplattform, die eine zeit- und ortsunabhängige sowie skalierbare Informationsspeicherung und -verteilung ermöglicht. Die Architektur setzt sich aus den drei Komponenten Client-Prozess, Applikations- und Datenbankserver zusammen, die in Abbildung 6.3 dargestellt sind.

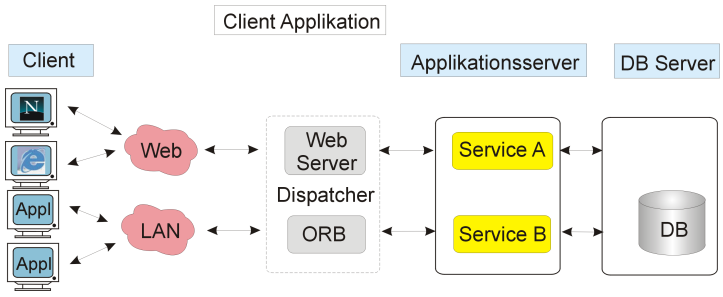

Abbildung 6.3: Anwendungsplattform ObjectStore

Da Client-Prozess und Applikationsserver eng verzahnt sind, werden diese oftmals auch unter dem Begriff *Client Applikation* zusammengefasst. Der **Client-Prozess** kann aus WWW-Anwendungen und/oder anderen Frontends bestehen und richtet alle Datenbank-Anfragen über das Web oder beispielsweise ein LAN[1] an einen *Dispatcher*. Im Fall der Kommunikation via Internet handelt es sich bei letzterem um einen Webserver, im anderen Fall z.B. um einen Object Request Broker (ORB). Der Dispatcher leitet alle ihn erreichenden Anfragen an den **Applikationsserver** weiter, der aus einer oder mehreren Routinen

[1] LAN - Local Area Network.

besteht, die kontinuierlich auf derartige Anfragen warten und sie an den zuständigen **Datenbankserver** weiterleiten. Der Datenbankserver befragt die Datenbank und liefert die angefragten Daten an den Applikationsserver zurück, von wo aus sie über den Dispatcher an den Client-Prozess gesendet werden. Abbildung 6.4 zeigt, wie Datenbankserver (DB-Server) und Client-Applikationen vernetzt werden können.

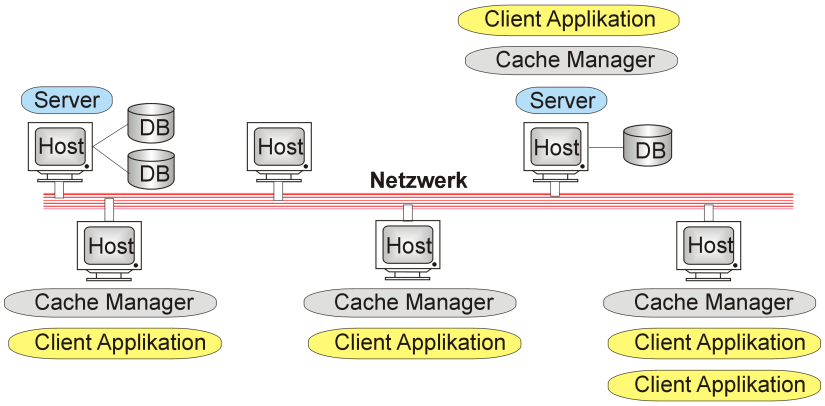

Abbildung 6.4: ObjectStore-Server und Client-Applikationen nach [Obj97]

Bei dem in Abbildung 6.4 dargestellten Aufbau handelt es sich um eine Multi-Client- / Multi-Server-Architektur, die sich durch folgende Eigenschaften auszeichnet:

- Auf mehreren Hosts können mehrere DB-Server und Client-Applikationen laufen.
- Jeder DB-Server kann mehrere Datenbanken und Client-Applikationen umfassen.
- Jeder Host, auf dem mindestens eine Client-Applikation läuft, besitzt genau einen Cache-Manager.

Besondere Eigenschaften

ObjectStore zeichnet sich im Vergleich zu anderen objektorientierten Datenbanksystemen durch die folgenden Besonderheiten aus:

Virtual Memory Mapping (VMM):
Angefragte ObjectStore-Objekte werden direkt in den virtuellen Adressraum der jeweiligen Applikation geladen und verbleiben dort, bis der Platz anderweitig benötigt wird. Falls sich ein persistentes Objekt bereits im Hauptspeicher befindet, kann jeder weitere Zugriff folglich ebenso effizient erfolgen wie bei transientem Zugriff. Performance-Steigerungen werden außerdem dadurch erreicht, dass ObjectStore-Objekte auf der Festplatte und im Hauptspeicher dieselbe Speicherstruktur besitzen. So werden zeitaufwändige Transformationen vermieden.

Multithreaded Server:
Die innerhalb von ObjectStore für die Multi-Client-Kommunikation eingesetzte Thread-Architektur verwendet das Thread-Management des jeweiligen Betriebssystems. Dadurch können die Vorteile der asynchronen I/O-Unterstützung und des symmetrischen Multiprozessings genutzt werden. Erstere erlaubt einem einzelnen Prozess, mehr als einen I/O-Request gleichzeitig zu initiieren. Der ObjectStore-Server verwendet pro Client einen eigenen, von anderen Clients nicht blockierbaren Thread. Symmetrische Multiprozessor-Architekturen unterstützen die Verteilung mehrerer Threads eines einzelnen Prozesses auf verschiedene Prozessoren. Auch hier profitiert der ObjectStore-Server von den resultierenden Leistungsvorteilen.

Verteilte Cache-Forward-Architektur:
ObjectStore ermöglicht das clientseitige Zwischenspeichern von Objekten. Der Cache-Manager, ein eigenständiger clientseitiger Prozess, kommuniziert dazu sowohl mit der Client-Applikation als auch mit dem Datenbankserver. Ein vom Client angefordertes Objekt wird bei diesem Ansatz auch nach Transaktionsende im zugehörigen Cache behalten, solange genügend Speicherplatz vorhanden ist. Bei erneuter Anforderung wird es zunächst im Cache gesucht, bevor der Client den Server mit der Beschaffung beauftragt. Konsistenz und Aktualität wird bei diesem Vorgehen dadurch sichergestellt, dass der Server ein Objekt „zurückfordert“, sofern es von einer anderen Client-Applikation zwischenzeitlich verändert wurde (“Callback Locking“). Sowohl dieses Vorgehen als auch die Lastverteilung zwischen Server und Client resultieren in einem deutlichen Leistungsgewinn im Vergleich zu anderen Architekturen.

Seiten-Server-Konzept
Zur Reduzierung des Kommunikationsaufwandes werden im Gegensatz zu anderen Datenbanksystemen bei Datenbankanfragen keine einzelnen Objekte, sondern stets Gruppen von Objekten als Menge von Seiten an den virtuellen Adressraum zurückgeliefert. ObjectStore benutzt dazu intern den Seitenverwaltungsmechanismus des jeweiligen Betriebssystems.

Object Clustering/Segmentierung:
Eine weitere Leistungssteigerung kann durch das kontrollierte Gruppieren persistenter Objekte im Datenbankspeicher erreicht werden. Das Konzept beruht darauf, logisch zusammengehörige Objekte im selben physikalischen Segment, welches aus einer festzulegenden Zahl von Seiten besteht, abzulegen. Wird auf ein Objekt zugegriffen, so ist die Wahrscheinlichkeit des anschließenden Zugriffs auf ein (logisch) benachbartes Objekt groß. Befindet sich dieses Nachbarobjekt auf derselben Seite, so wird es direkt bei Anforderung des ersten Objektes mit in den Hauptspeicher geladen und ist für einen eventuellen Zugriff schnell verfügbar.

C++-Interface:
Auch das C++-Interface von ObjectStore weist im Vergleich zu anderen objektorientierten Systemen einige Besonderheiten auf, die nachstehend kurz beschrieben werden.

Bezüglich des *Strukturteil*s stimmen Typen, Typkonstruktoren, Objektidentitäten, Klassen und Klassenhierarchie mit C++ überein. Zusätzlich existieren jedoch generische Klassen wie etwa *Collection, Set, List* und binäre *Relationships* zwischen Klassen desselben oder unterschiedlichen Typs der Kardinalität *1:1*, *1:n* und *n:m*. Deren automatische Pflege und Konsistenzhaltung erspart dem Entwickler entsprechenden Programmier- und Wartungsaufwand. Bezüglich des *Operationenteil*s erfolgt der Zugriff auf Datenbankobjekte entweder navigierend über definierbare *Datenbank-Einstiegspunkte*, von wo aus andere persistente Objekte erreichbar sind, oder deskriptiv über Selektionsausdrücke, die an sämtliche Variablen einer *Collection* gestellt werden können.

Im Hinblick auf die **Persistenz** von Daten wird erst während der Entstehung eines Objektes entschieden, ob dieses persistent oder transient angelegt werden soll. Eine Klasse kann somit gleichzeitig persistente und transiente Objekte definieren.

An **Zugriffspfaden** stehen Hash-Techniken für ungeordnete und B-Bäume für geordnete Wertebereiche zur Verfügung. Durch die explizite Angabe von Pfadindizes kann die Zugriffsgeschwindigkeit hierbei noch einmal erhöht werden.

Es werden geschachtelte und lange **Transaktionen** angeboten, die als *read-only* oder *update* deklariert werden können. Statische Transaktionsgrenzen werden im Programmtext festgelegt, dynamische dagegen zur Laufzeit als *begin()*- oder *commit()*-Botschaft an die spezielle Klasse *Transaction* geschickt. Lange Transaktionen werden durch einen Check-Out/Check-In-Mechanismus unterstützt.

In engem Zusammenhang zum Transaktionsmanagement steht die **Versionenverwaltung**. Das sog. *Multiversion Concurrency Control (MVCC)* ist ein zusätzliches Transaktionsmodell, das lesenden Transaktionen eine konsistente Sicht auf Objekte erlaubt, auf welche gleichzeitig innerhalb einer update-Trans-

aktion schreibend zugegriffen wird. Dazu werden zwei parallele Applikationen gestartet, von der die eine nur lesende Zugriffe erlaubt, die zweite dagegen lesende und schreibende Zugriffe. Die erste Applikation verwendet eine Kopie der aktuellen Datenbasis und garantiert so, dass read-only-Transaktionen unberührt von update-Transaktionen an der Original-Datenbank bleiben. Die Kopie wird in festgelegten Zeitabständen durch die aktuelle Version ersetzt.

Die **Migration bestehender C- oder C++ -Programme**, die bisher Datenpersistenz mittels Dateienverwaltung realisieren, auf ObjectStore-Anwendungen ist möglich. Dazu sind Anpassungen notwendig, die das Öffnen der Datenbank, das Einrichten von Transaktionsgrenzen, oder etwa die Allokation von Hauptspeicher (*malloc*) betreffen.

Eine **Anbindung relationaler Datenbanken** wird durch eine spezielle Webserver-Anwendung realisiert, die eine objektorientierte Sicht auf eine Datenbank mit relationalem Datenmodell besitzt. Dazu wird eine Schicht zur Verfügung gestellt, die den Zugriff auf der Basis von Objekten und nicht Relationen erlaubt.

Für notwendige Änderungen an Klassen, zu denen bereits Objekte in der Datenbank existieren, beinhaltet ObjectStore Routinen für einfache Anpassungen wie etwa das Hinzufügen oder Entfernen von Attributen. Die Implementierung spezieller **Schemaevolutions**-Applikationen für komplexe Änderungen wird durch eine spezielle Programmierschnittstelle unterstützt. In diesem Fall erfolgt eine Redefinition der entsprechenden Klassen, eine Modifikation des Datenbankschemas und eine Anpassung der Repräsentationen aller betroffenen Objekte, d.h. aller existierenden Inhalte.

Die besonders im Hinblick auf e-Learning-Systeme wichtigen Aspekte der multimedialen Objektverwaltung und der WWW-Anbindung der Datenbank werden in den folgenden Abschnitten erläutert.

6.4 Effizientes Datenmanagement

Eine Aufgabe, für die sich Datenbanksysteme zwar aufgrund ihrer Eigenschaften prinzipiell gut eignen, mit der sie bisher aber eher selten beauftragt sind, ist die Verwaltung und Bearbeitung von multimedialen, vor allem im WWW bereitgestellten Inhalten. ObjectStore umfaßt eine Bibliothek sog. **ObjectManager**, auf deren Basis ein Ausbau des Systems zu einer Variante von Multimedia-Datenbanken möglich wird (vgl. Abschnitt 3.8). Durch Einsatz spezieller Bibliotheken wird dabei die Implementierung einer text- und bildbasierter Indizierung unterstützt. Einzelheiten zu diesen Bibliotheken und ihrer Verwendung innerhalb des Wissensmodells erläutert Abschnitt 6.5. Im Folgenden werden nur einige ihrer grundlegenden Eigenschaften beschrieben.

Die abstrakte Basisklasse *osmmType* stellt die Schnittstelle zur Erzeugung und Manipulation multimedialer Objekttypen zur Verfügung. Die Subklassen *osmmText*, *osmmHTML*, *osmmImage*, *osmmVideo*, *osmmAudio* oder *osmmJava* werden zur Beschreibung der jeweiligen Metainformation eines Objekts verwendet, die eigentlichen Rohdaten werden separat davon als eine Instanz der Klasse osmmStorage abgelegt.

Die Klasse *osmmIndex* definiert ein allgemeines Protokoll, auf dessen Basis eine inhaltsbasierte Indizierung von Text- und Bilddaten implementiert werden kann. In direktem Zusammenhang mit osmmIndex steht die Klasse *osmmSearch*, durch die eine Unterstützung bei der Informationssuche in indizierten Objekten zur Verfügung gestellt wird. Aufgaben wie die Aktualisierung und Konsistenzhaltung der Indices oder das Abgleichen mit den Objekten der eigentlichen Datenbank bei Einfügen, Ändern oder Löschen müssen dagegen vollständig implementiert werden.

6.4.1 WWW-Schnittstelle zur Datenbank

Für einen WWW-basierten Zugriff auf eine ObjectStore-Datenbank, wie er im Rahmen des eLS-Baukastens möglich sein soll, wurde der in Abbildung 6.5 dargestellte Ansatz umgesetzt. Der Vorteil dieses

HTTP-basierten Ansatzes liegt in der einfachen Benutzerschnittstelle für den Datenzugriff, der keinerlei clientseitige Installationen erfordert. Auch gehört das eingesetzte CGI-Skript zur 2. Generation seiner Art, mit der keine Datenbankanfragen fest codiert sind, sondern beliebige Anfragen bearbeitet werden können (vgl. Abschnitt 3.7).

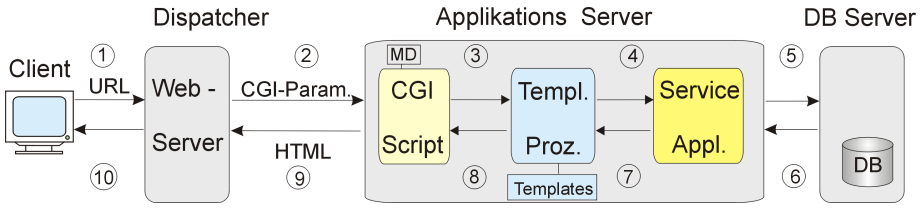

Abbildung 6.5: Anbindung einer ObjectStore-Datenbank an das WWW

In Anlehnung an Abbildung 6.3 gestaltet sich dieser Ansatz folgendermaßen: Der Client ist der Webbrowser, der in Form einer URL eine Datenbankanfrage an den Dispatcher, in diesem Fall einen Webserver, absetzt. Dieser kontaktiert den Applikationsserver, der im Fall von ObjectStore aus einer Kombination eines speziellen CGI-Skripts samt Mappingdatei ("MD"), einem Templateprozessor und einer **Service Applikation** besteht und mit dem Datenbankserver kommuniziert. Der Templateprozessor greift auf Templatedateien zu, die gemeinsam die graphische Benutzeroberfläche bilden. Diese bestehen aus HTML-Code, JavaScript, Java-Anweisungen und zusätzlichen Datenbank-Anfragen, die vom Templateprozessor herausgefiltert und an die Service Applikation weitergereicht werden.

Der exakte Vorgang einer clientseitigen Anfrage bis zur Darstellung des Anfrageergebnisses entspricht den folgenden Schritten, die in Abbildung 6.5 mit 1 - 10 bezeichnet sind [Mil97].

1. Der Client setzt, beispielsweise über ein Formular, eine URL ab, z.B.:
   http://www.host.de/cgi-bin/osform.exe/run_user?osform_template=login.oft&LOGIN=meier&PWD=hans,
   Über den ersten Teil *http://www.host.de* der URL kann der Zielrechner angesteuert werden.

2. Dieser interpretiert den verbleibenden Teil der URL:
   /cgi-bin/osform.exe/run_user?osform_template=login.oft&LOGIN=meier&PWD=hans
   und weiß anhand der lokalen Pfadangabe */cgi-bin/*, dass er das im dortigen Verzeichnis befindliche Skript *osform.exe* aufrufen und diesem den Rest der URL übergeben muss.

3. Das CGI Skript interpretiert den an ihn übergebenen Inputstring:
   run_user?osform_template=login.oft&LOGIN=meier&PWD=hans.
   Dazu muß das Skript aus der ihm zugeordneten Mapping Datei ("MD") ermitteln, auf welcher Portnummer die Service Applikation *run_user* „hört". Die Mapping-Datei ist beispielsweise wie in Abbildung 6.6 dargestellt aufgebaut:

run_admin	host A	3276
run_user	host A	3277
check_num	host B	3278

Abbildung 6.6: Beispiel einer Mapping Datei

In diesem Fall laufen auf *host A* die zwei Applikationen *run_admin* und *run_user*, auf *host B* die Applikation *check_num*. Über eine spezielle Kommunikationsbibliothek schickt das Skript den Rest der URL über den dort angegebenen Port *3277* an den Templateprozessor.

4. Der Templateprozessor erhält den Input String:
   osform_template=login.oft&LOGIN=meier&PWD=hans,
   und sucht die hinter *osform_template* angegebene Templatedatei *login.oft*. Diese ist beispielsweise wie in Abbildung 6.7 gezeigt aufgebaut.

```
<%comment Filename: login.oft%>
<HTML>
<%query name="Login" osfunction="cbLogin"%>
<%if Login.Status eq 0%>
    Willkommen, User <%LOGIN%>! Dies ist Ihr <%NUMBER_VISITS%>. Besuch.
<%else%>
    Sie sind nicht registriert.
<%endif%>
</BODY>
</HTML>
```

Abbildung 6.7: Beispiel einer Templatedatei

   Der Templateprozessor extrahiert die durch „<%query...%>“ gekennzeichnete Datenbank-Anweisung „cbLogin“ und sendet diesen Namen zusammen mit dem Rest der URL *&LOGIN=meier&PWD=hans* an die Service Applikation.

5. Die Service Applikation besteht aus einer Liste von Unterprogrammen, sog. Callback-Funktionen, die darauf warten, dass Anfragen an sie gerichtet werden. Die Callback-Funktion „cbLogin“ könnte z.B. wie in Abbildung 6.8 dargestellt implementiert sein. Sie liest die übergebenen Parameter *LOGIN=meier* und *PWD=hans* ein und schickt diese Werte über den Aufruf „create_pick“ an die ObjectStore-Datenbank.

```
int cbLogin(osFormApiType apiObj, const char * apiOperation,osFormStatus status )
{
  OS_BEGIN_TXN(T1,0,os_transaction::read_only)
  // Input
     const char *inLg = osFormGetStringVariable( apiObj,"LOGIN");
     const char *inPwd = osFormGetStringVariable( apiObj,"PWD");
  // Output
     osFormAttr attrNumberVis = osFormCreateAttribute(apiObj,"NUMBER_VISITS",...);
  ... *query = &os_coll_query::create_pick("User*","!strcmp(m_Login,(const char* )ARG1)"
   && "!strcmp(m_Password,(const char* )ARG2)",db);
  ... boundQuery(*Query,(os_keyword_arg("ARG1",inLg), os_keyword_arg("ARG2",inPwd)));
  User* user = (User*)db_ptr->query_pick(boundQuery);
  if (!user)
      return ERR;
  else
     osFormAddNewValue(attrNumberVis,(void *) user->getNumberVisits, 1);
}
```

Abbildung 6.8: Beispiel einer Callback-Funktion in ObjectStore

6. Konnte aufgrund der angefragten Parameterwerte die Anfrage nach dem User erfüllt werden, liefert die Datenbank als Antwort an die Service Applikation die entsprechende Instanz dieser Klasse zurück.

7. Der Service „verpackt“ den von ihm aus der Instanz ausgelesenen Wert für „NumberVisits“ in den Rückgabeparameter *attrNumberVis* und sendet diesen an den Templateprozessor.

8. Dieser ersetzt die in *login.oft* enthaltenen Parameter *<%LOGIN%>* und *<%NUMBER_VISITS%>* durch die entsprechenden Werte und sendet die nun interpretierte HTML-Datei an das CGI-Skript zurück.
9. Das CGI-Skript leitet die so erzeugte HTML-Datei *login.oft* an den Webserver weiter.
10. Der Webserver sendet sie an den Client, wo sie in dessen Browser folgendermaßen dargestellt wird:

Willkommen, User Meier! Dies ist Ihr 17. Besuch.

Nach diesem Prinzip lassen sich sämtliche lesenden und schreibenden Datenbankanfragen durchführen.

6.4.2 Strategien für effiziente Datenzugriffe

Im Folgenden werden einige Strategien erläutert, die auf der Basis von ObjectStore implementiert wurden, und einerseits zu einer Effizienzsteigerung hinsichtlich Netzauslastung und Frequentierung des Datenbankservers beitragen, andererseits eine Unabhängigkeit vom Hostrechner ermöglichen und so zu einem verringerten Implementierungsaufwand beitragen.

Ablage von Medien auf dem Server

Prinzipiell sind zwei verschiedene Formen der Verwaltung multimedialer Objekte innerhalb einer Datenbank denkbar: Bei der ersten erfolgt die Ablage des Mediums in der Datenbank selbst, d.h. eine Mediendatei wird als entsprechender Byte-Strom im Speicherbereich der Datenbank abgelegt und kann über eine zu implementierende Methode gelesen und geschrieben werden. Die zweite Variante speichert dagegen lediglich eine Referenz auf die zugehörige Datei ab, die selbst im Dateisystem des Hostrechners abgelegt ist. Abbildung 6.9 veranschaulicht dieses Prinzip.

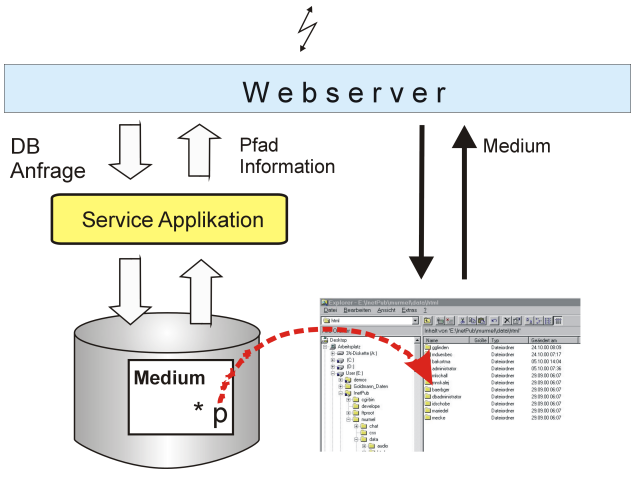

Abbildung 6.9: Kombination von Datenbank- und Filesystem

Neben der Tatsache, dass die Medienobjekte auch bei einem Ausfall der Datenbank erreicht werden können, bietet dieses Vorgehen einen weiteren entscheidenen Vorteil: Der Datenbankserver liefert nur die textuelle Information über den Pfad eines Mediums in Form einer URL und die Transaktion wird nach Senden der Pfadinformation an den Webserver beendet. Der Webserver ist anschließend gänzlich unabhängig von der Datenbank dafür verantwortlich, das Medium aus dem Filesystem an den Client zu transferieren. Dieses Vorgehen soll im Folgenden als „pointer-to-file-Strategie“ bezeichnet werden.

Im Hinblick auf schreibende Zugriffe durch die Autoren besitzt diese Strategie außerdem den Vorteil, dass für alle Medientypen ein einheitliches Vorgehen im Hinblick auf deren Verwaltung realisiert werden kann. Auch existieren bei der Übertragung von Informationen über einen HTTP-Request von Seiten des Webservers Limitierungen bezüglich der Länge einer URL. Diese kann beispielsweise bei 2048 oder 4096 Zeichen liegen. Die meisten Medien umfassen jedoch weitaus größere Datenmengen, so dass es hier zu Problemen bei der Übertragung kommen könnte. Auch werden auf diese Weise Daten im clientseitigen Webbrowser Cache gehalten und können bei Bedarf schnell direkt von dort geladen werden. Schließlich kann durch dieses Prinzip der Medienverwaltung eine einfache Variante der Versionenverwaltung von Seiten der Datenbank erfolgen: Bei einer autorenseitigen Aktualisierung eines Medienobjekts wird nach einer eindeutigen Systematik die alte Datei umbenannt und an die neue ein nach festgelegten Regeln gebildeter Name vergeben. So kann von Datenbankseite leicht auf alle jemals zu einem Medienobjekt zugeordneten Dateien zugegriffen werden. Wird die Instanz eines Medienobjekts gelöscht, so werden automatisch alle verwalteten früheren Versionen ebenfalls aus dem Filesystem entfernt.

Verwendung von Templates

Die Verwendung von Templates entlastet neben dem Datenbankserver auch die Service Applikation und führt gleichzeitig zu einer vollständigen Unabhängigkeit des e-Learning-Systems vom Hostrechner. Neben der Extraktion von Datenbankanfragen aus den Templatedateien (vgl. Abschnitt 6.4.1), ist der Templateprozessor auch für die anschließende Einbettung von Anfrageergebnissen in die Templatedatei zuständig. Oftmals sind hierzu Informationen über den Hostrechner erforderlich, beispielsweise die IP-Adresse bzw. der Domainname des Hostrechners, Laufwerksbezeichnungen oder relative Pfade, hinter denen sich Mediendaten befinden. Diese Information ist innerhalb des eLS-Baukastens nicht fest codiert, sondern wird beim Hochfahren einer Service Applikation aktuell und online berechnet bzw. eingelesen und in einer „Headerdatei“ im Templateverzeichnis abgelegt; Abbildung 6.10 zeigt beispielhaft einen Ausschnitt aus einer solchen Datei.

<%HTTP_HOST	=„http://www.murmel.uni-tuebingen.de/“%>
<%HTTP_HOST_DATA	= <%HTTP_HOST%>data/"%>
<%HTTP_HOST_BM	=<%HTTP_HOST%>MyMurmel/"%>
<%HTTP_SERV_TEMPL_ADMIN	= „/cgi-bin/osform.exe/run_admin?...“%>
...	...
<%ADMIN_STATUS	= „DWRX“%>
<%DEFAULT_LOGO	= „logo.jpg“%>

Abbildung 6.10: Beispielausschnitt einer Headerdatei

Der Name der Headerdatei wird durch einen einfachen Befehl in jedes Template eingebunden und ermöglicht so dem Templateprozessor, dort enthaltene Parameter zur Laufzeit aufzulösen. Die Vorteile dieses Vorgehens sind offensichtlich: Service Applikation und Datenbankserver werden nicht damit belastet, häufig angefragte Daten wie Rechnernamen, absolute Pfadangaben, Laufwerksbezeichnungen, etc. zu transferieren, da der Templateprozessor diese Information direkt aus der aktuellen Headerdatei beziehen kann. Gleichzeitig werden so Abhängigkeiten der Datenbank, der Service Applikation und der Templatedateien von einem bestimmten Host vermieden. Eine Migration des gesamten e-Learning-Systems auf einen anderen Rechner wird durch dieses Prinzip ganz erheblich vereinfacht.

Übertragung minimaler Datenmengen an den Client

Zur Reduktion des Datentransfers über das Netz wird zunächst nur eine geringe Datenmenge zur Präsentation von Inhalten des e-Learning-Systems durch den Client-Browser übertragen. Auf der Basis der so gewonnenen Information kann dann die Anforderung weiterer, detaillierterer Inhalte erfolgen. Fordert der Anwender beispielsweise eine Liste von Objekten an, so werden zunächst ein Thumbnail und/oder eine kurze textuelle Beschreibung an den Client zurückgeliefert. Diese Information vermittelt dem Anwen-

der einen ersten Eindruck vom eigentlichen Medium. Das datenintensive Medienobjekt selbst wird erst auf explizite Anfrage nach dem oben beschriebenen Vorgehen an den Client gesendet. Auf die dadurch mögliche Einsparung von Übertragungszeit und Serverkapazität wird im Rahmen der Beschreibung von Navigationsmöglichkeiten für den Benutzer eines e-Learning-Systems in Abschnitt 7.2 näher eingegangen.

Der Hilfsassistent DBInterface

Die in Abschnitt 6.4.1 beschriebene Funktionsweise von Service Applikationen wird durch den Hilfsassistenten „DBInterface“, der die Schnittstelle zwischen Datenbankserver und Service Applikation realisiert, erweitert. Hierbei handelt es sich um eine mächtige, transient angelegte Klasse, die jeden Datenbankeinstiegspunkt, sowie sämtliche Listen und Verknüpfungen kennt, und über Methoden zum Lesen, Schreiben und Manipulieren datenbanknaher Objekte verfügt. Jede der drei in Abbildung 6.11 dargestellten Service Applikationen verwendet eine eigene Instanz dieser Klasse für den Zugriff auf die Datenbank (Abbildung 6.11).

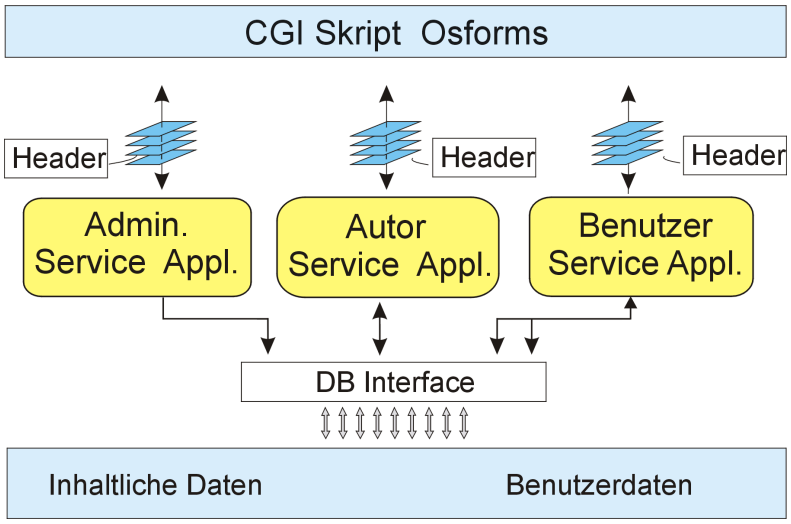

Abbildung 6.11: Der Hilfsassistent *DBInterface* als neue DB-Schnittstelle

Die einmalige Implementierung des Assistenten ermöglicht die schnelle Entwicklung von Applikationen und die zentrale Wartung der Datenbankschnittstelle. Zugleich benötigen die Applikationen keinerlei Kenntnis mehr über die für einen Datenbankzugriff notwendige Syntax.

6.4.3 Zusammenfassung

Durch den Einsatz von ObjectStore im eLS-Baukasten werden alle Vorteile heutiger Datenbanksysteme wie Persistenz, Transaktionsmanagement, Query-Unterstützung, Skalierbarkeit, Multiuser-Fähigkeit und Sicherheit für e-Learning-Systeme verfügbar. Durch die Erweiterung um eine WWW-Schnittstelle können darüber hinaus Anforderungen nach orts-, zeit- und plattformunabhängigem Zugriff auf die Daten erfüllt werden. Auch die Verwaltung multimedialer Objekte läßt sich unter Einsatz einiger grundlegender Datentypen geeignet modellieren. Auf der Basis von ObjectStore realisierte e-Learning-Systeme lassen sich somit im weitesten Sinne als „Web Content Management“-Systeme bezeichnen (vgl. Abschnitt 3.7).

6.5 Konzepte zur Wissensmodellierung

Aufbauend auf den im Abschnitt 2.3.1 erläuterten verschiedenen Formen der Wissensvermittlung wurde im Rahmen dieser Dissertation ein Klassenmodell entwickelt, mit dem Lerninhalte in generierten e-Learning-Systemen abgelegt werden können. Neben Datentypen, die als Container zur Aufnahme elementarer Medienobjekte wie Text, Bild oder Videos dienen, sind dazu auch komplexere Datentypen erforderlich, die im Sinne des in Abschnitt 1.3 eingeführten Instruktions- und des Problemlösungsparadigmas verschiedene didaktische Lehrmodelle abbilden können. Der Aufbau und die Funktionsweise einzelner Klassen berücksichtigen dabei zum einen, auf welche Weise Wissen gespeichert und vernetzt werden soll, andererseits aber auch dessen WWW-basierte Präsentation. Abbildung 6.12 gibt einen ersten Überblick über die Klassen, die der eLS-Baukasten zur Modellierung des Wissens bereitstellt.

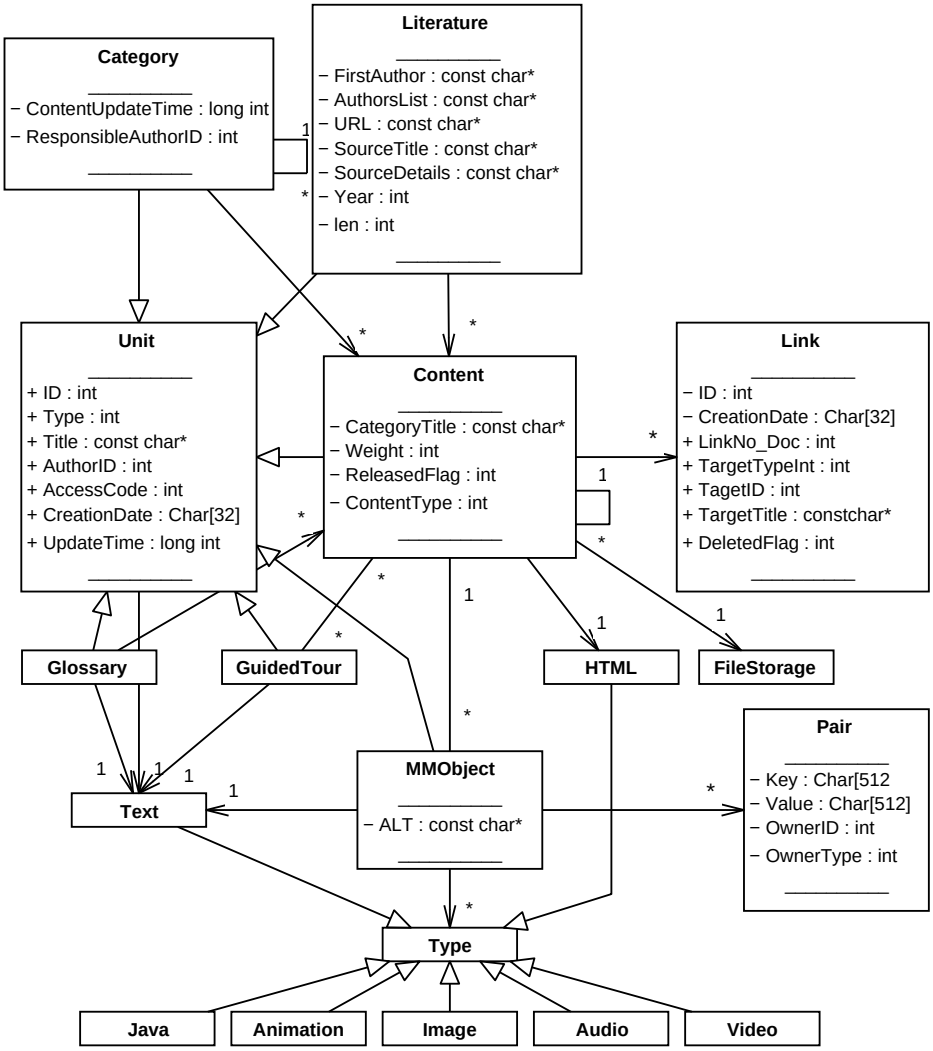

Abbildung 6.12: Wissensmodell generierter e-Learning-Systeme: Überblick

Wissen wird modelliert, indem elementare Bauteile wie Texte, Bilder oder andere Medienobjekte zu komplexen Objekten zusammengesetzt werden. Im Folgenden werden zunächst die wichtigsten elementaren Klassen vorgestellt. Anschließend wird erläutert, wie mit ihrer Hilfe unterschiedliche Wissensarten bzw. Interaktionsformen modelliert werden können (vgl. Abschnitt 2.3.1).

6.5.1 Elementare Klassen zur Wissensmodellierung

Der eLS-Baukasten unterstützt neben einfachen textuellen Attributen die Medientypen Text, Bild, Video, Audio, Java und Animation. Die bereits in Abschnitt 6.4 erwähnten Klassen *osmmType*, *osmmText*, *osmmHTML*, *osmmImage*, *osmmVideo* und *osmmAudio* repräsentieren diese Medienobjekte innerhalb des Klassenmodells (vgl. Abbildung 6.12); alle verschiedenen Lerninhalte sind aus diesen Medienobjekten zusammengesetzt. Auf dieser Basis wurden die atomaren Klassen des Modells entwickelt. Der Großteil dieser Klassen erbt von der Klasse **Unit**, deren wesentliche Attribute Abbildung 6.13 zeigt.

```
class Unit
{
public:
   int m_ID;
   int m_Type;
   const char* m_Title;
   int m_AuthorID;
   int m_AccessCode;
   Char[32] m_CreationDate;
   osmmText* m_Keywords;
}
```

Abbildung 6.13: Wissensmodell: Die Klasse Unit

Die Unit-Klasse umfaßt im Wesentlichen Metadaten, die zu jedem Objekt festgehalten werden sollen. Dies sind zunächst eine eindeutige Identifizierung, ein Typ und ein Titel. Desweiteren werden Schlüsselworte, Angaben über den Autor und damit Erzeuger einer Unit-Instanz sowie das Datum der Erstellung gespeichert.

Die Klasse **MMObject** ist ein Container für eine Liste von Medienobjekten. In der Liste werden Medienobjekte mit unterschiedlichen Varianten gleichen Inhalts zusammengefasst, beispielsweise Bilder in unterschiedlichen Auflösungen, oder ein Video und einige Standbilder. Darüber hinaus erbt MMObject von Unit, so dass zu den enthaltenen Medienobjekten Metainformationen gespeichert werden können.

Das wichtigste Attribut der Klasse MMObject ist *m_MediaList* (Abbildung 6.14 (a)). Hier können beliebige der oben erwähnten Medientypen gespeichert werden. Ein Beispiel soll dies verdeutlichen: Instanzen von MMObject können beispielsweise verwendet werden, um eine WWW-gerechte Präsentation von Bildern zu speichern. Für größere Bilder ist es sinnvoll, eine verkleinerte und eine oder mehrere detaillierte Varianten des Bildes zu speichern. Dazu werden die einzelnen Varianten in einem MMObject zusammengefaßt. Wird ein MMObject dargestellt, sieht der Benutzer zunächst die iconisierte Variante des Bildes als eine Art Voransicht. Detaillierte Varianten werden erst geladen, wenn der Benutzer sie tatsächlich anfordert.

Das Attribut *m_ROIList* steht im engen Zusammenhang mit der bildbasierten Informationssuche. Es wird zur Ablage von Bildregionen genutzt, die vom Autor als besonders relevant innerhalb eines größeren Bildes angesehen werden; hierauf wird in Kapitel 8 näher eingegangen. Schließlich wird über das Attribut *m_Content* eine Beziehung zwischen der Klasse MMObject und der Klasse **Content** hergestellt (Abbildung 6.14 (b)). Sie repräsentiert eine elementare „Wissenseinheit“ eines e-Learning-Systems, die einen — zumeist kleinen — in sich geschlossenen, didaktisch sinnvollen Themenkomplex bilden, der nicht weiter unterteilt wird. Diese Klasse nimmt damit eine zentrale Stellung innerhalb des Wissensmodells ein. Content-Objekte setzen sich im Wesentlichen aus einem HTML Text (im Attribut *m_HTML*) und einer MMObject Liste (im Attribut *m_MMObjectList*) zusammen.

Über die genannten Inhalte hinaus speichern Content-Objekte eine von den Autoren vergebene Gewichtung, welche die Relevanz des Contents unter dem Gesichtspunkt der Wichtigkeit für ein Thema ausdrückt. Das Attribute *m_ReleasedFlag* kennzeichnet, ob eine Instanz bereits freigegeben und damit

innerhalb der Benutzeroberfläche zugänglich ist. Die übrigen Attribute sind mehr „technischer Natur“. So speichert ein Content-Objekt in zwei Listen, auf welche anderen Objekte es verweist und von welchen anderen Objekten auf dieses Content Objekt verwiesen wird. Darüber hinaus wird ein eindeutiger Bezug zu einer Category und dem Autor, der sie angelegt hat, verwaltet. Die Category Klasse wird im nächsten Unterabschnitt vorgestellt, für sie sind auch die Attribute *up* und *down* der Klasse Content reserviert.

Die Content Klasse bildet den Grundbaustein für die nachfolgend beschriebenen komplexen Klassen des Wissensmodells. Diesbezügliche Einsatzmöglichkeiten sind charakteristisch für das Konzept des eLS-Baukastens: Didaktische (Grund-) Bausteine werden als Content Objekte modelliert und sind damit in allen generierten e-Learning-Systemen verfügbar. Autoren können aus diesen Grundbausteinen beliebige Lehrmodelle zusammensetzen und über das Autorensystem entsprechende Lehrinhalte erstellen. Dadurch ist der eLS-Baukasten nicht auf einen vorgegebenen Satz von Lehrmodellen festgelegt.

```
class MMObject: public Unit
{
  os_List<osmmType*>* m_MediaList;
  Content* m_Content;
  os_List<Pair*>* m_ROIList;
  const char* m_ALT;
  osmmText* m_Legend;
}
```

```
class Content: public Unit
{
  int m_ContentType;
  osmmHTML* m_HTML;
  osmmFileStorage* m_HTML_FileStorage;
  int m_Weight;
  int m_ReleasedFlag;
  int m_Weight;
  const char* m_CategoryTitle;
  os_List<MMObject*>* m_MMObjectList;
  os_List<Link*>* m_LinkList;
  os_List<Content*>* m_LinkSourceList;
public:
  Content::os_rel_Content_down down;
  Content::os_rel_Content_up up;
  Content::os_rel_Content_auth auth;
  Content::os_rel_Content_gts gts;
}
```

Abbildung 6.14: Wissensmodell: Die Klassen MMObject (a) und Content (b)

Das Wissensmodell bietet weitere elementare Klassen wie **Glossar** oder **Literatur**, die zur Ablage von eher einfach strukturiertem Wissen verwendet werden. (vgl. Abbildung 6.12). Schließlich existiert die Klasse **Link**, die beliebige Contents mittels bidirektionaler Verknüpfungen verbindet. Abbildung 6.15 zeigt die wichtigsten ihrer Attribute.

```
class Link
{
public:
  int m_ID;
  int m_Type;
  int m_LinkNo_Doc;
  int m_TargetTypeInt;
  int m_TargetID;
  const char* m_TargetTitle;
  const char* m_DeletedFlag;
  Char[32] m_CreationDate;
}
```

Abbildung 6.15: Wissensmodell: Die Klasse Link

Die Klasse Link speichert Typinformationen des Objekts, auf das ein Content verweist: die absolute

Nummer innerhalb des Dokuments, in das der Link eingetragen wurde, Typ, eindeutige ID und Titel, die Information, ob das Target zwischenzeitlich gelöscht wurde (*m_DeletedFlag*) und das Datum seines Anlegens. Eine besondere Rolle nimmt diese Klasse zum einen für Autoren ein, die so in der Pflege der von ihnen generierten Links unterstützt werden (vgl. Abschnitt 7.3), zum anderen ist sie auch im Zusammenhang mit der innerhalb der generierten e-Learning-Systeme angebotenen Volltextsuche (s. Kapitel 8) von Bedeutung.

6.5.2 Tutorieller Dialog, Drill & Practice und Guided Tour

In Abschnitt 2.3.1 waren unterschiedliche Lernstrategien und Interaktionsformen vorgestellt worden, deren Modellierung Gegenstand dieses und der folgenden Abschnitte ist. In diesem Unterabschnitt wird die Modellierung von Drill & Practice, Tutoriellem Dialog und Guided Tour erläutert.

Die Interaktionsform **Drill & Practice**, bei der gelerntes Faktenwissen vertieft werden soll, wird primär mit der Klasse Content modelliert und im Rahmen des eLS-Baukastens als „Quiz“ bezeichnet. Einzelne Content-Instanzen enthalten in diesem Fall HTML-Inhalte, die Fragen mit den zugehörigen Antwortmöglichkeiten repräsentieren. Für richtige Antworten wird dem Benutzer eine festgelegte Anzahl von Punkten gutgeschrieben und innerhalb der Klasse **User** abgelegt.

Bei der Interaktionsform **Tutorieller Dialog** werden dem Benutzer zunächst Wissensinhalte präsentiert und diese anschließend durch Kontrollfragen überprüft. Im Falle einer falschen Antwort können Hinweise auf weiterführende Informationen, die durch die Klasse Link realisiert werden, gegeben werden. Abbildung 6.16 zeigt die Grundstruktur eines solchen Tutorials.

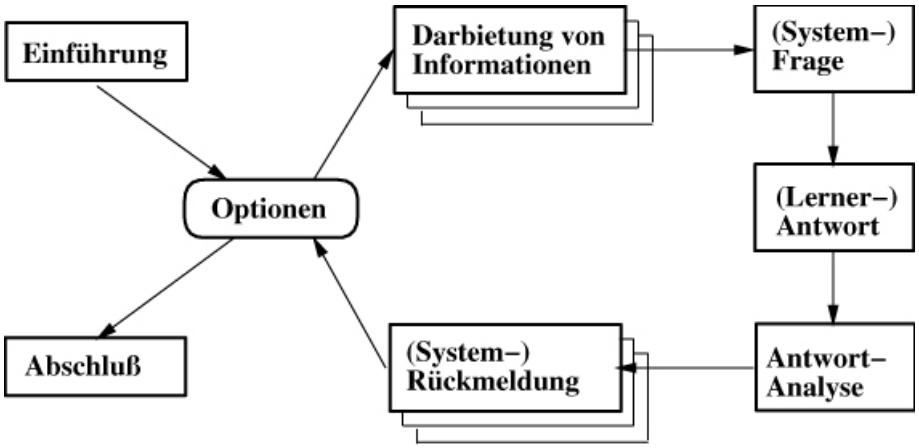

Abbildung 6.16: Grundstruktur eines tutoriellen Dialogs nach [Wur98]

Die Modellierung des Tutoriellen Dialogs entspricht der Realisierung von Drill & Practice, ist aber um die Möglichkeit weiterführender Hinweise ergänzt. Inhaltlich wird durch tutorielle Dialoge außerdem eher prozedurales als reines Faktenwissen vermittelt. In der Praxis wird deshalb ein Tutorial in der Regel mehrere Contents umfassen, während bei Drill & Practice -Dialogen meist ein sequenzielles Präsentieren und Abfragen erfolgt.

Zur Realisierung der Interaktionsform **Guided Tour** dient die Klasse gleichen Namens. Sie modelliert eine einfache Präsentation, bei der dem Benutzer Lerninhalte in einer festen linearen Anordnung präsentiert werden. Die Klasse umfaßt neben den von Unit ererbten Attributen lediglich ein Textfeld zur Ablage einer Beschreibung und eine Liste mit Referenzen auf Content-Instanzen des e-Learning-Systems, die beliebigen Kategorien entstammen können. Durch die Angabe einer festen Reihenfolge gibt der Autor für die Guided Tour eine aus seinem Blickwinkel sinnvolle Abfolge der Inhalte vor.

6.5.3 Simulieren und Trainieren

Als weitere Interaktionsform unterstützt der eLS-Baukasten problemorientiertes Lernen in Form von **Simulieren und Trainieren** typischer Handlungsabläufe. Eine grundlegende Form stellen dabei die sog. *Fallbeispiele* dar, die allgemeine Vorgehensweisen zum Erreichen eines bestimmten Ziels beschreiben. Autoren können zur Erzeugung von Fallbeispielen dazu entweder auf bereits vorhandene Contents zurückgreifen, die zu einer fixen Folge verknüpfen werden, oder gänzlich neue Contents erzeugen. Ein komplexeres Vorgehen erfordert die Erzeugung eines simulierten Handlungsablaufs, bei dem der Weg zum Ziel nicht eindeutig vorgegeben ist. In Abbildung 6.17 ist ein abstraktes Beispiel für einen solchen Handlungsablauf angegeben.

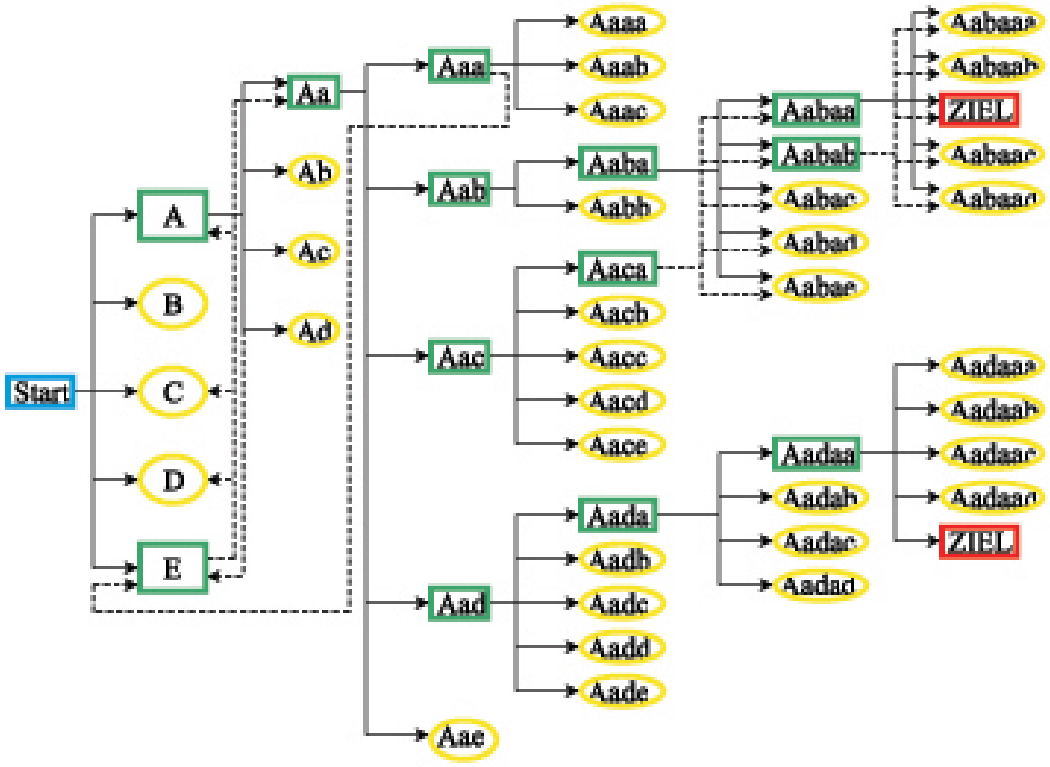

Abbildung 6.17: Schema eines Handlungsablaufs nach [Wur98]

Abbildung 6.17 zeigt eine Menge von Content-Objekten, die durch die Attribute *up* und *down* und durch generierte Verweise zu einer baumartigen Struktur verknüpft sind. Der mit „Start“ bezeichnete Content beinhaltet die Beschreibung der Ausgangsituation, die eckig umrandeten Kästchen stehen für „interne Schritte“, die eine Zwischenstufe entweder auf dem Weg hin zum richtigen Zielzustand darstellen oder in einer „Sackgasse“ (innerhalb von Abbildung 6.17 durch eine ovale Kontur gekennzeichnet) enden. Das „Ziel“ selbst und die Wege dorthin sind nicht notwendigerweise eindeutig. Die verschiedenen Pfade durch diesen komplexen Baum repräsentieren verschiedene Handlungsabläufe, denen ein Benutzer folgen kann. Dieser kennt die Gesamtübersicht über den Baum nicht, sondern sieht stets nur den Unterbaum eines von ihm gewählten Handlungsschrittes, d.h. Contents.

Die einzelnen Content-Knoten beinhalten wie gewohnt Text und weitere Medienobjekte sowie Links zu anderen Contents. Im Unterschied zu den bisher gezeigten Verwendungsmöglichkeiten der Content-Klasse besitzen Content-Instanzen hier eine größere Abhängigkeit von den übrigen Contents der Simulation, d.h. ein einzelner Content beschreibt in diesem Fall keinen in sich abgeschlossenen Inhalt eines

Fachgebietes. Vielmehr sind gezielt bestimmte Medienobjekte, Fragen oder absichtlich irreführende Inhalte, Expertenkommentare und ähnliches integriert. Besonders bei Simulationen liegt somit ein großer Teil des didaktischen Konzepts in den Händen der Autoren, da ihnen bei der Gestaltung keine Einschränkungen auferlegt werden.

6.5.4 Browsing

Zur Realisierung der Interaktionsformen **Präsentation** und **Browsing** dient die Klasse **Category**, im Folgenden als „Kategorie“ bezeichnet, deren Aufbau Abbildung 6.18 zeigt.

```
class Category: public Unit
{
   Content* m_RootContent;
   osmmText* m_Description;
   os_List<osmmVirageImage*>* m_ImageList;
   int m_ResponsibleAuthorID;
   long int m_ContentUpdateTime;
   os_List<Content*>* m_AlphaContentList;
public:
   class Category::os_rel_Category_down: public os_rel_m_1{}
   class Category::os_rel_Category_up: public os_rel_1_m{}
   Category::os_rel_Category_down down;
   Category::os_rel_Category_up up;
}
```

Abbildung 6.18: Wissensmodell: Die Klasse Category

Eine Kategorie besteht aus einen Baum, dessen Knoten durch Instanzen der Klasse Content repräsentiert werden. Das Attribut *m_RootContent* zeigt auf die Wurzel dieses Baumes, und die bereits im letzten Abschnitt erwähnten Attribute *up* und *down* der Content-Klasse verknüpfen die einzelnen Kapitel oder Unterkapitel. Neben dem Attribut *m_RootContent* existieren eine Reihe weiterer Attribute. So gestattet das Attribut *m_Description* der Kategorie eine kurze Beschreibung zuzuordnen. Das Attribut *m_ImageList* definiert Bilder, mit denen die Kategorie in der Benutzeroberfläche dargestellt wird, beispielsweise Icons, die der Benutzer auswählen kann, um den der Kategorie zugeordneten Content-Baum anzuzeigen. Weiterhin wird mit *m_ResponsibleAuthorID* ein Autor ausgezeichnet, der die Berechtigung besitzt, alle Contents innerhalb der Kategorie zu ändern. Änderungen sind ansonsten nur den Autoren des jeweiligen Contents gestattet. Das Attribut *m_ContentUpdateTime* schließlich speichert des Datum der letzte Änderung an einem Content.

Das Attribut *m_AlphaContentList* beinhaltet eine alphabetisch sortierte Liste aller enthaltenen Content-Instanzen. Auf diese Weise steht ein Index zur Verfügung, anhand dessen gezielt zu einer bestimmten Content-Instanzen navigiert werden kann.

Das in Abbildung 6.19 dargestellte Beispiel demonstriert die Verwendung von Kategorien. In einem medizinischen e-Learning-System könnte es die Kategorien „Krankheitsbilder“, „Symptomatologie“ und „Neurotopologie“ geben. Unterhalb der Kategorie „Symptomatologie“ existieren Instanzen von Contents, z.B. auf der ersten Ebene „Sensorische Symptome“, auf der zweiten Ebene „Hörstörungen“, „Sensible Störungen“, usw. Die Attribute up und down der Klasse Content werden zum Aufbau des Baumes herangezogen, jeder Content hat dabei genau einen „Vater“ und eine Liste von „Kindern“.

Insgesamt gestattet eine Kategorie die Navigation durch Wissensinhalte, bei der ein Benutzer einem vorgegebenem Aufbau (hier der Baumstruktur) folgt. Gleichzeitig kann das Browsen der Inhalte aber ermöglicht werden, indem Instanzen der Klasse Content über Querverweise (Instanzen der Klasse Link) miteinander verknüpft werden. Diese sind über Anker im Text einer Content-Instanz erreichbar.

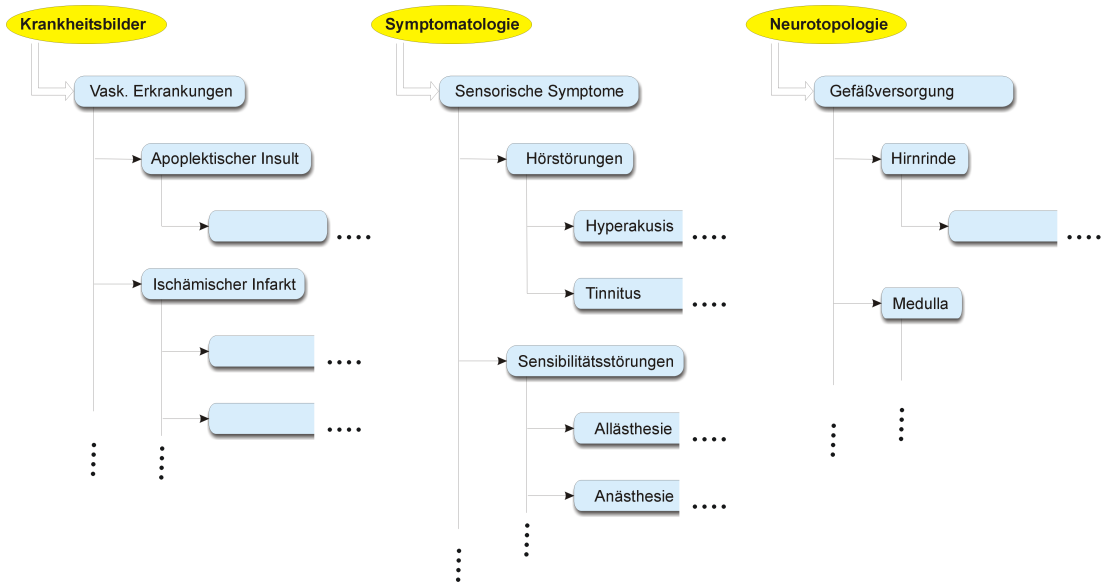

Abbildung 6.19: Beispiel von Contentbäumen der Klasse Category

Alle Klassen des Wissensmodells wurden so entworfen, dass Autoren im Hinblick auf die Abbildung verschiedener Lernstrategien ein gewisses Maß an Gestaltungsspielraum haben. Einerseits können Instanzen zu einer der beschriebenen Interaktionsformen zusammengesetzt werden, genausogut können aber auch Verknüpfungen generiert werden, durch die komplexere Lernmodule, beispielsweise ein längerer, verzweigter Handlungsablauf, repräsentiert werden. Auf diese Weise gestattet der eLS-Baukasten die Generierung von e-Learning-Systemen unterschiedlichen Lernsystemtyps.

6.6 Prozess zur Generierung von e-Learning-Systemen

Eine wichtige Zielsetzung bei der Entwicklung des eLS-Baukastens war die Entlastung von IT-Spezialisten von der wiederkehrenden, komplexen Aufgabe der Implementierung neuer e-Learning-Systeme und deren teilweise sehr aufwendigen Wartung und Pflege. Die Generierung eines neuen e-Learning-Systems durch den eLS-Baukasten ist einfach und schnell durchführbar und erfordert vom Administrator lediglich die Ausführung der in Abbildung 6.20 dargestellten fünf Arbeitsschritte. Dazu muß zunächst die Runtime Version des Datenbanksystems installiert werden (1). Anschließend werden im öffentlich zugänglichen Webserver-Verzeichnis die Start-Sites für die Administratoren- und Benutzeroberfläche, das CGI-Skript und die angepaßte Mappingdatei abgelegt (vgl. Abschnitt 6.4.1)(2). In ein vom Administrator wählbares Verzeichnis werden die Service Applikationen und ihre zugehörigen Templates gespeichert (3). Nach dem Ausführen der DB-Generierungs-Routine (4) werden diese im letzten Schritt gestartet (5). Jede Service Applikation besteht dabei aus unterschiedlich vielen Callback-Funktionen: Die Administrator-Schnittstelle umfaßt beispielsweise ca. 120 solcher Funktionen, für den Autor wurden ca. 50 Routinen implementiert und auf Benutzerseite existieren knapp 100 verschiedene Varianten für den lesenden und schreibenden Zugriff auf die Datenbank.

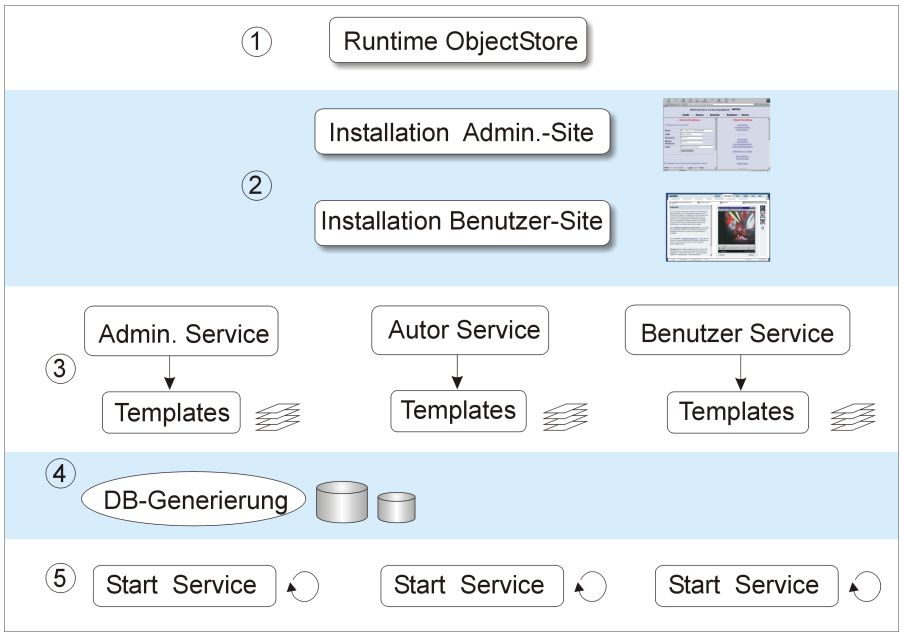

Abbildung 6.20: Ablauf zur Generierung von e-Learning-Systemen

Die **Ausführung der DB-Generierungs-Routine** (in Abbildung 6.20 als “DB-Generierung“ bezeichnet) setzt sich aus den in Abbildung 6.21 angegebenen sechs Teilschritten zusammen.

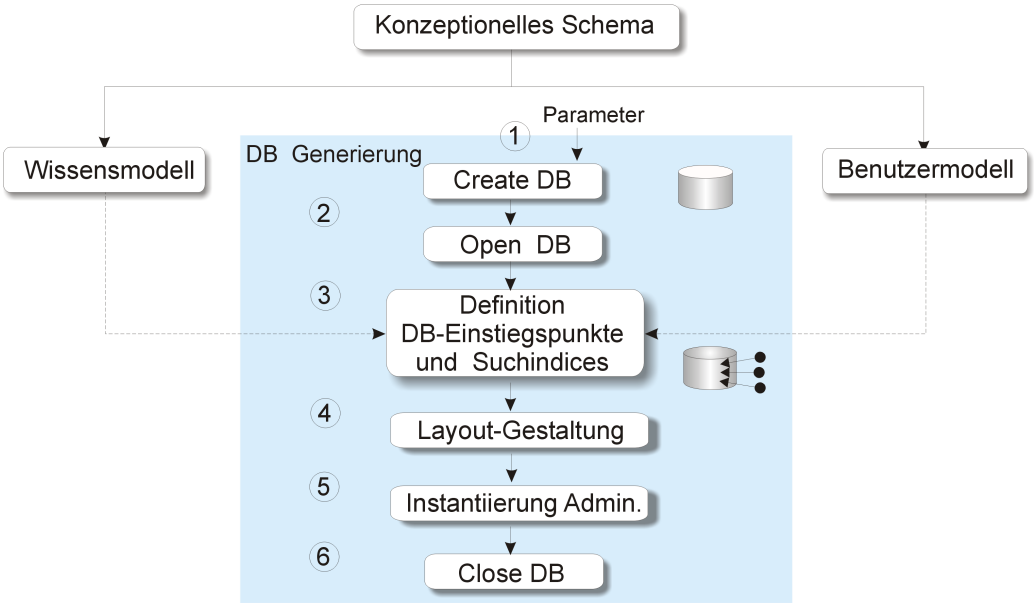

Abbildung 6.21: Ablauf zur Datenbank-Generierung für e-Learning-Systeme

Zunächst wird ein Name für die zu erzeugende Datenbank als Eingabeparameter erwartet (1). Außerdem können zwei Laufwerksnamen angegeben werden, von denen der eine den Ort des Webserver-Verzeichnisses, der andere den für die Ablage der zu erzeugenden Datenbank angibt. Anschließend erfolgt die Angabe eines Verzeichnisses, das für die Ablage aller Mediendateien genutzt wird. Wie die Dateien

dort abgelegt werden, wird im Zusammenhang mit der Beschreibung des Autoren-Clients in Abschnitt 7.3 erläutert.

Wie in Abschnitt 6.3.2 beschrieben, sind Inhalte einer Datenbank nur über physikalisch definierte Einstiegspunkte zugänglich. Nach Anlegen und Öffnen der Datenbank (2) werden die vom eLS-Baukasten initial vorgesehenen Einstiegspunkte für alle persistent abzulegenden Inhaltstypen angelegt (3). Bei Bedarf können zu jedem späteren Zeitpunkt aber mit Hilfe der Administrator-Service Applikation weitere Einstiegspunkte definiert werden. Ein Mechanismus, der ebenfalls in Abschnitt 6.3.2 beschrieben worden war, ist das Prinzip des „Object Clustering“ bzw. der „Segmentierung“, bei dem Daten eines bestimmten Typs so gespeichert werden, dass ein schneller Zugriff auf diese sichergestellt ist. Die Festlegung der physikalischen Segmente für die Instanzen eines Typs erfolgt zusammen mit der Definition der Einstiegspunkte. Im nächsten Schritt wird der Administrator bei der Angabe spezifischer Projekt- bzw. Kursangaben unterstützt (4), die innerhalb des e-Learning-Systems abrufbar sein sollen. Beispielsweise können Logos, Projekt- bzw. Kursbeschreibungen oder allgemeine Hilfemenüs in das System eingefügt werden. Als letztes wird in der Datenbank eine Administrator-Instanz der Klasse Autor angelegt (5) und die Datenbank wieder geschlossen (6). Mit Hilfe der Adminstrator-Instanz kann ein Administrator sich über die Client-Applikation ortsunabhängig am e-Learning-System anmelden und lesend und schreibend auf alle Daten zugreifen.

Beim **Starten der drei Service Applikationen** als letzter Schritt der e-Learning-System- Generierung in Abbildung 6.20 wird jeweils ein kurzer, für alle Applikationen identischer Initialisierungsprozess durchgeführt. Dieser erfolgt jeweils nach dem in Abbildung 6.22 dargestellten Schema.

/ Service Applikation für ...*/*	
Öffnen einer (weiteren) Sicht auf die Datenbank	
IF	Parameter über akt. Laufwerk, etc. vorgegeben
THEN	Parameter verarbeiten
ELSE	Rechnerabhängige Angaben automatisch ermitteln
Template-Verzeichnis bestimmen	
Spezielle Headerdatei erzeugen	
Instanz des Hilfsassistenten DBInterface anlegen.	
Datenbank-Einstiegspunkte anfordern	
Starten des Listeners	

Abbildung 6.22: eLS-Baukasten: Initialisierung einer Service Applikation

Parallel zum Start einer Service Applikation wird zu eventuell bereits laufenden Service-Applikationen anderer Clients ein weiterer Prozess gestartet, der als Kanal für die Kommunikation zwischen Applikation und Datenbank verwendet wird. Optional können hierbei Parameter über Webserver- und Datenbank-Laufwerk, etc. angegeben werden, sofern diese sich seit Aufsetzen des e-Learning-Systems geändert haben. Die Service Applikation überprüft die Existenz des ihr zugeordneten Templateverzeichnisses und erzeugt die für diese Applikation vorgesehene Headerdatei; pro Applikation wird außerdem eine transiente Instanz des Hilfsassistenten DBInterface angelegt. Anschließend begibt sich die Service Applikation in eine Schleife und wartet auf Anfragen von Clients.

Innerhalb kürzester Zeit kann auf diese Weise ein neues e-Learning-Framework aufgesetzt werden, auf welches über die drei Service-Applikationen lesend und schreibend zugegriffen werden kann und in das über die spezielle Autoren-Applikation Inhalte hinzugefügt und bearbeitet werden können.

6.7 Zusammenfassung

In diesem Kapitel wurde demonstriert, wie e-Learning-Systeme mit Hilfe des eLS-Baukastens generiert und auf effiziente Weise gewartet werden können. Es wurde deutlich, dass sich der Baukasten besonders durch seine leichte Anwendbarkeit und die weitestgehende Transparenz aller technischen Vorgänge auszeichnet.

Architektur und Funktionsweise des eLS-Baukastens wurden beschrieben und nach Einführung des verwendeten Datenbanksystems einige spezielle Implementierungsdetails gegeben, durch die eine weitaus höhere Systemleistung erreicht werden kann. Anschließend wurde das realisierte Wissensmodell zur Abbildung verschiedener Lernformen erörtert und schließlich die notwendigen Schritte zur Generierung eines so konzipierten e-Learning-Systems dargestellt.

Mit dem nach dem Komponentenparadigma konzipierten eLS-Baukasten wurde ein Werkzeug geschaffen, das e-Learning-Systeme generiert, die sich durch hohe Flexibilität und performante Zugriffsmöglichkeiten speziell über das Netz auszeichnen. Die in diesem Kapitel beschriebenen Strategien sind dabei für Personen, die ein e-Learning-System aufsetzen möchten, völlig transparent und konfrontieren in keiner Weise mit technischen Details oder durchzuführenden Systemkonfigurationen.

Eine Kosten- und Zeitersparnis für Institutionen, die computerbasiertes Lernen über das WWW anbieten wollen, wird insbesondere durch die Möglichkeit der problemlosen Vervielfältigung der verwendeten Modelle und Techniken gewährleistet. Ebenso trägt eine Trennung von Lehrfunktion und Lehrinhalten dazu bei, die Wiederverwendung von didaktischen Modellen und integriertem Wissen sicherzustellen. Der komponentenbasierte Aufbau stellt darüber hinaus sicher, dass Systeme an veränderte Anforderungen, weiterentwickelte Techniken und zukünftige Medientypen adaptierbar sind.

Das innerhalb eines jeden generierten e-Learning-Systems enthaltene Wissensmodell zeichnet sich durch einen strukturierten, modularen Aufbau aus, der einerseits standardmäßig verschiedene, vorimplementierte Lernstrategien zur Nutzung durch Autoren anbietet, andererseits durch den hohen Grad an modularen Strukturen diesen auch die Möglichkeit bietet, zusätzliche, individuelle Lehrkonzepte abzubilden.

Durch die in Folge der Modularisierung ermöglichte strikte Trennung von systemseitiger Datenablage und clientseitiger Präsentation der Daten ist es dem Administrator dabei zu jedem Zeitpunkt möglich, Anpassungen des e-Learning-Systems an individuelle Gegebenheiten durchzuführen.

Kapitel 7

Realisierte Client-Applikationen und Funktionalitäten

Ein mit Hilfe des eLS-Baukastens generiertes e-Learning-System bietet Client-Applikationen für Administratoren, Autoren und für die Benutzer des Systems. Das vorliegende Kapitel beschreibt die Anwendung und Funktionsweise dieser verschiedenen Clients, erläutert Implementierungsdetails und erörtert speziell entwickelte Werkzeuge zur individuellen Benutzerunterstützung und Evaluierung eines im Einsatz befindlichen e-Learning-Systems.

7.1 Administrator-Client

Abbildung 7.1 zeigt exemplarisch die Benutzeroberfläche des Administrator-Clients. Im oberen Frame sind die verschiedenen zu bearbeitenden Bereiche angezeigt: Inhalte, Autoren, Benutzer, Statistiken und System. Im unteren, unterteilten Frame werden alle Funktionalitäten eingeblendet, die für einen bestimmten Bereich durchführbar sind. Als Beispiel wird in Abbildung 7.1 ein neuer Autor für ein e-Learning-System zugelassen. Dazu legt der Administrator für diesen über ein entsprechendes Formular ein Autorenprofil an. Dieses Profil wird in der Datenbank gespeichert und zusätzlich ein Dateiverzeichnis auf den Namen des registrierten Autors angelegt, das die vom Autor in das e-Learning-System eingefügten Dateien verwalten wird (vgl. Abschnitt 6.4.2).

Der Administrator-Client erlaubt den passwortgeschützten Zugang zu folgenden Funktionalitäten, die innerhalb der Administrator Service Applikation implementiert wurden:

1. Inhalteverwaltung:
   - Anlegen, Bearbeiten und Löschen von Kategorien und allen Inhalten.
   - Ausführen von Änderungen, die alle Inhalte eines Datentyps betreffen.
   - Durchführung sämtlicher Aktivitäten, die Autoren über das Autorensystem ausführen können.
2. Autorenverwaltung: Anlegen und Löschen von Autorenprofilen, Vergabe von Zugriffsrechten.
3. Benutzerverwaltung: Anlegen und Verwalten registrierter Benutzerprofile, Vergabe von Zugriffsrechten für Benutzer, Durchführung aller Benutzeraktionen.
4. Statistische Auswertungen und Analysen von Benutzeraktionen und Datenbankinhalten.
5. System: Definition globaler Einstellungen für das graphische Layout der Benutzerschnittstelle, z.B. Einbinden von Logos, Anlegen spezieller Seiten zur Projektbeschreibung und Hilfemenüs.

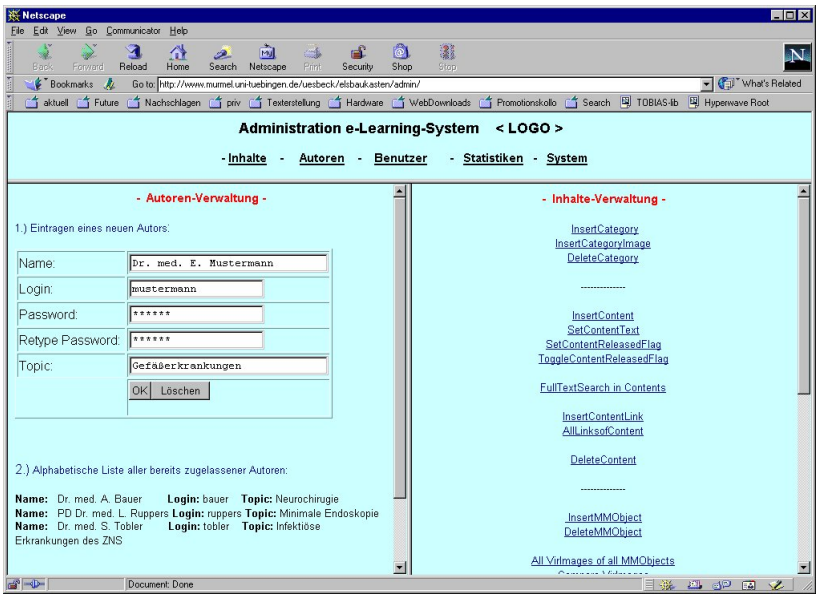

Abbildung 7.1: Administrator-Client

Diese Funktionalitäten werden mit Hilfe von knapp 120 implementierten Callbackfunktionen realisiert.

Die gesamte Administrationsoberfläche ist HTML-basiert implementiert, da diese Lösung einen stabilen schnellen Remote-Zugang zu einem e-Learning-System gestattet und für Administratoren aufgrund ihrer Vorkenntnisse und Intention keine aufwendige graphische Anwenderschnittstelle erforderlich ist.

Anders verhält es sich bei der Benutzerschnittstelle für die Lernenden, die nachfolgend vorgestellt wird.

7.2 Benutzer-Client

Dieser Abschnitt beschreibt zunächst den generellen Aufbau der Benutzeroberfläche, demonstriert die Arbeit mit einem generierten e-Learning-System aus Sicht der Benutzer und beschreibt schließlich einige Implementierungsdetails.

7.2.1 Genereller Aufbau der Benutzeroberfläche

Die benutzerseitige Service Applikation ist auf den lesenden, inhaltlichen Datenzugriff ausgerichtet und umfaßt derzeit ca. 100 modulare Funktionen, die unterschiedliche Datenbankzugriffe realisieren. Schreibender Zugriff erfolgt hierbei nur im Zusammenhang mit der Erzeugung und Verwaltung individueller Benutzermodelle, die in Abschnitt 7.4 beschrieben werden. Den generellen Aufbau der framebasierten Benutzeroberfläche zeigt Abbildung 7.2.

Die obere und untere Menüleiste sind während der Arbeit mit dem e-Learning-System stets erreichbar und erlauben das Ansteuern aller verfügbaren Bereiche des Systems. Der mittlere Bereich der Oberfläche dient zur Darstellung von Inhalten.

Über die oberen Schaltflächen sind verschiedene Inhalte erreichbar: Informationen über das Projekt oder der vom e-Learning-System angebotenen Kurse sind hinter dem Button „Info zu finden, der Einstieg in den inhaltlichen Bereich des Systems ist über den Button „Hauptmenü“ möglich. „Suche“ bietet Zugang zu verschiedenen Suchfunktionen, die fester Bestandteil aller mit Hilfe des eLS-Baukastens generierten e-Learning-Systeme sind. Hier finden sich Möglichkeiten zur attributbasierten, keyword-, text-

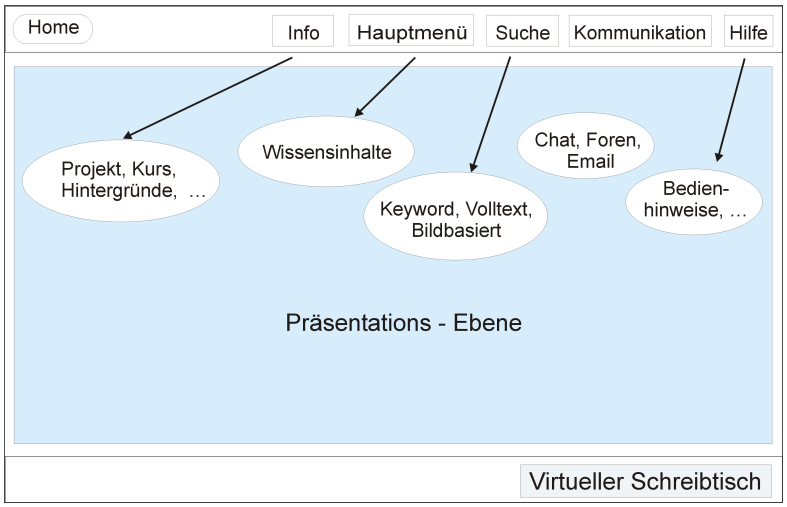

Abbildung 7.2: Ebenen der Benutzeroberfläche

und bildbasierten Suche in den multimedialen Wissensinhalten. Unter dem Menüpunkt „Kommunikation“ befinden sich verschiedene Werkzeuge wie Chat, Foren und Email-Vorrichtungen, über die Lernende und Lehrende miteinander in Kontakt treten können. Hier sind dem momentanen Stand der Technik entsprechende kommerzielle Produkte eingebunden, die aufgrund der komponentenbasierten Struktur des Systems leicht austauschbar sind und auch deaktiviert werden können. Unter „Hilfe“ verbirgt sich schließlich ein Menü, das die Funktionsweise des aktuellen e-Learning-Systems beschreibt.

Die untere Navigationsleiste präsentiert zu Beginn einer Arbeitssitzung zunächst nur den Zugang zum „Virtuellen Schreibtisch“. Dahinter verbirgt sich ein spezieller Bereich zur individualisierten Benutzerunterstützung, der jedem Benutzer verschiedene Werkzeuge zur Verfügung stellt, die die Arbeit mit dem e-Learning-System vereinfachen.

Bei Auswahl des Menüpunktes „Hauptmenü“ unterteilt sich der mittlere Fensterbereich in weitere Frames. Abbildung 7.3 zeigt diesen generellen Aufbau.

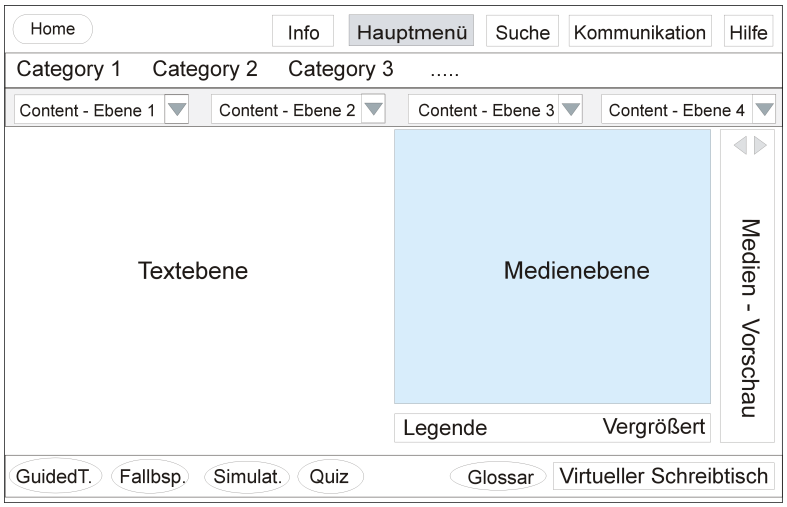

Abbildung 7.3: Wissenspräsentation innerhalb von e-Learning-Systemen

Im oberen Bereich der Oberfläche werden alle verfügbaren Kategorien dargestellt. Nach Auswahl einer Kategorie erscheinen direkt unterhalb vier sog. Dropdown-Menüs, über die auf die einzelnen Ebenen des zugehörigen Content-Baumes zugegriffen werden kann. Die Dropdown-Menüs zeigen dabei stets von links nach rechts einen Ausschnitt dieses Baumes. Nach Auswahl eines Inhalts wird dieser im Bereich unterhalb der Dropdown-Menüs dargestellt. Links wird der zugehörige Text eingeblendet, ganz rechts, in der „Medien-Voransicht“, alle zu diesem Inhalt verfügbaren Medien. Zwischen Text und Voransicht (“Medienebene“) wird die Normalansicht eines ausgewählten Mediums dargestellt, die Legende zu einem Medium bzw. die vergrößerte Ansicht können über die darunter befindlichen Schaltflächen abgerufen werden. Der Vorteil dieser zeitlich aufeinanderfolgenden Informationspräsentation wurde in Abschnitt 6.4.2 erläutert.

Im untersten Frame der Oberfläche befinden sich die Schaltflächen für das problemorientierte Wissen. Hier werden Inhalte der in Abschnitt 6.5 beschriebenen Interaktionsformen GuidedTour, Fallbeispiele, Simulationen und Quiz-Module angeboten. Bei Auswahl einer Interaktionsform werden die entsprechenden Inhalte identisch zum systematischen Wissen innerhalb der Präsentationsebene dargestellt.

Der beschriebene Aufbau ist unabhängig von den Inhalten, die im Rahmen verschiedener generierter e-Learning-Systeme präsentiert werden. Er berücksichtigt Anforderungen, die generell an graphische Benutzeroberflächen zur Präsentation systematischen und problemorientierten Wissens gestellt werden und ist deshalb nach Generierung eines neuen e-Learning-Systems zunächst für alle Systeme identisch. Wie später noch in Kapitel 9 bei der Vorstellung konkreter Systeme gezeigt wird, lassen sich jedoch ohne technischen Aufwand Änderungen des Layouts durchführen.

7.2.2 Ein e-Learning-System aus Benutzersicht

Für die Beschreibung der verschiedenen Navigationsmöglichkeiten durch die Wissensinhalte eines e-Learning-Systems werden im Folgenden einige Beispiele aus dem in Abschnitt 9.2 vorgestellten medizinischen e-Learning-System MURMEL gegeben. Abbildung 7.4 zeigt dazu als erstes die Einstiegsseite in dieses e-Learning-System.

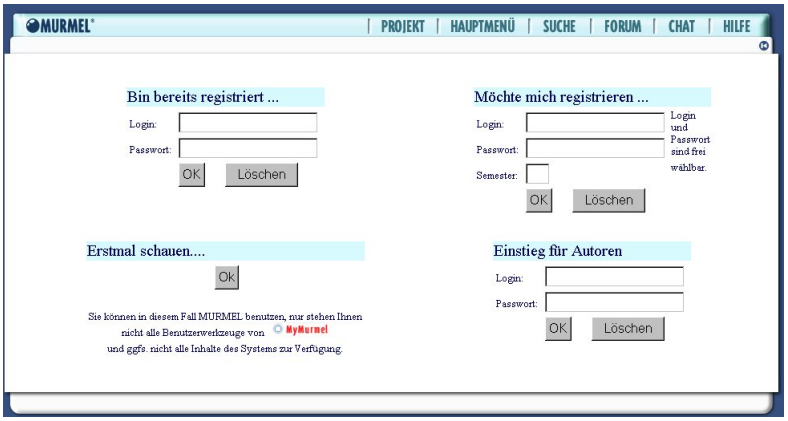

Abbildung 7.4: Die Startseite eines e-Learning-Systems

Im Hauptfenster befinden sich verschiedene Einstiegsformulare: Neben der Login- und Passwortabfrage für Autoren unten rechts existieren für Benutzer des Systems drei verschiedene Möglichkeiten der Einwahl. So ist es möglich, sich selbständig ein Login und Passwort einzurichten und damit als registrierter Benutzer den vollen Zugriff auf alle Funktionalitäten zu erhalten (in Abbildung 7.4 oben rechts). In diesem Fall werden alle abgerufenen oder bearbeiteten Informationen protokolliert und stehen dem Lernenden bei seinem nächsten Besuch wieder zur Verfügung. Das linke Formular wird von bereits regi-

strierten Benutzern für die Wiederanmeldung benutzt, der Button darunter ist für eine anonyme Anmeldung. Nach der Anmeldung werden Benutzer direkt zum Hauptmenü des e-Learning-Systems geleitet. Abbildung 7.5 zeigt als Beispiel einen Eintrag des systematischen Wissens, also der Interaktionsform **Browsing**.

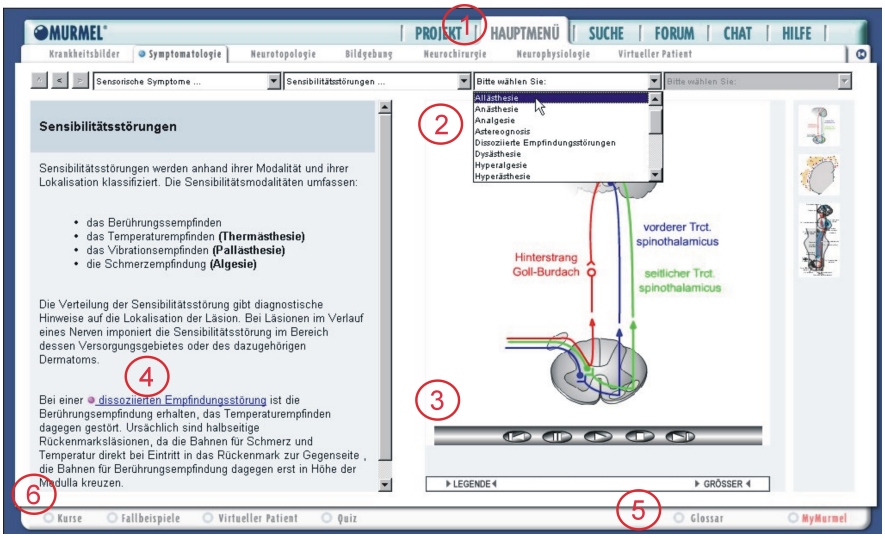

Abbildung 7.5: Darstellung eines Contents in der Benutzeroberfläche

Hier wurde bereits die Kategorie „Symptomatologie“ und aus dem zugehörigen Baum die Wissenseinheit „Sensibilitätsstörungen“ ausgewählt und das dritte Dropdown-Menü ist geöffnet (Punkt 2). Im Medienfenster ist bereits eines der zugehörigen Medien geladen, in diesem Fall ein Shockwave-Video, das über die untere Schaltfläche ähnlich der Funktionalität eines Videorecorders bedienbar ist (Punkt 3).

Im Text des linken Frames zeigt der Anker „dissoziierte Empfindungsstörung“ auf weiterführende Information zur aktuellen Wissenseinheit (Punkt 4). Hierbei handelt es sich um einen sog. typisierten Link, da das vorangestellte, farbliche Icon bereits anzeigt, um welche Art von Verweis es sich hierbei handelt. Möglich sind Verweise auf beliebige andere Contents des systematischen und des problemorientierten Wissens, auf Medien, Glossar und Literatur. Die insgesamt in einem e-Learning-System enthaltenen Literaturinhalte sind außerdem auch unter dem Auswahlpunkt *Suche* der oberen Leiste zu finden. Eine Liste aller verfügbaren, alphabetisch sortierten Glossareinträge kann über den entsprechenden Button im unteren Frame aufgerufen werden (Punkt 5). In Abbildung 7.5 durch Punkt 6 gekennzeichnet, sind die Bereiche für verschiedene problemorientierte Lernstrategien zu finden. Im Fall des hier betrachteten medizinischen e-Learning-Systems wurden im Sinne der Interaktionsform **Tutoring** sog. *Kurse* (im Wissensmodell sind dies die GuidedTours) in das System integriert. *Fallbeispiele*, als zweiter Auswahlpunkt zu sehen, können ebenfalls dem **Tutoring** zugeordnet werden; sie sind zugleich auch Elemente des Bereiches **Simulation**, dem auch die sog. *Virtuellen Patienten* angehören. *Quiz*-Module sind Vertreter der Lernstrategie **Drill & Practice**. In Kapitel 9 werden Beispiele dieser verschiedenen Interaktionsformen gezeigt.

7.2.3 Die Benutzeroberfläche aus technischer Sicht

Für Benutzer vollständig transparent erfolgt bei der Navigation durch ein e-Learning-System die Kommunikation zwischen Client, Server, Service Applikation und Datenbankserver. Der diesbezügliche Ablauf wird im Folgenden kurz skizziert. Unter „Hauptmenü“ bietet sich dem Lernenden dazu zunächst die Liste aller verfügbaren Kategorien. Abbildung 7.6 zeigt den vereinfachten Ausschnitt aus der zugehörigen

Templatedatei.

```
<%comment Filename: cat_list.oft%>
<HTML>
<%query name=“CatList“ osfunction=“cbCatList“%>
<%begindetail name=““CatList“%>
<A HREF=“#“ onClick=“activate(“<%HTTP_CLICK%>“); return true;“>
<IMG ...’ SRC=“<%HTTP%>“></A>
<%enddetail%>
</HTML>
```

Abbildung 7.6: Templatedatei zur Ermittlung der Category-Liste

Durch diese Anfrage wird nach dem in Abschnitt 6.4.1 beschriebenen Verfahren beim Datenbankserver die Liste aller Kategorien angefordert. Die ausgewertete Templateseite zeigt Abbildung 7.7.

```
<%comment Filename: cat_list.oft%>
<HTML>
<A HREF=“#“ onClick=“activate(“kb_click.gif“); ><IMG ... SRC=“/kb.gif“ ></A>
<A HREF=“#“ onClick=“activate(“sympto_click.gif“); ><IMG ... SRC=“/sympto.gif“ ></A>
<A HREF=“#“ onClick=“activate(“nt_click.gif“);><IMG ... SRC=“/nt.gif“ ></A>
...
```

Abbildung 7.7: Ausgewertetes Template zur Ermittlung der Category-Liste

Hinter der Funktion „activate“ verbirgt sich eine JavaScript-Funktion, die bei Wahl einer Kategorie deren zweites abgelegte Bild “<% HTTP_CLICK %>“ anzeigt. Abbildung 7.8 zeigt das graphische Resultat dieser Datenbankanfrage.

Abbildung 7.8: Oberflächenausschnitt: Category-Leiste

Die Auswahl einer Kategorie bewirkt die Ausführung einer weiteren Templatedatei, die in Abbildung 7.9 dargestellt ist.

```
<APPLET NAME=“NavApplet“ ID=“NavApplet“ CODE=“Nav.class“ ARCHIVE=“Nav.jar“>
<PARAM NAME=“jsOnNewTreeAndResize“ VALUE=“showClicked“>
<PARAM NAME=“historySize“ VALUE=“40“>
<PARAM NAME=“treeurl“ VALUE=“<%TEMPLATE%cont_list_applet.oft&CAT_ID=“>
<PARAM NAME=“initCatID“ VALUE=«%CAT_ID%>“>
...
</APPLET>
```

Abbildung 7.9: Category: Zugehörige Content-Liste (1)

Dieses Template erzeugt den Aufruf des Java-Applet *NavApplet*, über das mit Hilfe einer singulären Datenbankanfrage der zur Kategorie zugehörige Content-Baum gelesen und dargestellt wird. Bei der Initialisierung des Applets mit der vom Benutzer gewählten Kategorie werden dazu durch Anstoßen des

Templates „cont_list_applet.oft“ über einen normalen HTTP-Request die in Abbildung 7.10 dargestellten Parameter bei der Datenbank erfragt.

```
<%query name=“CatSingle_ID“ osfunction=cbCatSingle_ID“%>
<%begindetail name=“CatSingle_ID“%>
  category.id=<%CAT_ID%>
  category.url=<%TEMPLATE%>cont_tree.oft
                &CAT_ID=<%CAT_ID%>&CONT_ID=<%ROOTCONT_ID%>
  category.changed=<%CHANGED%>
  content.url=<%TEMPLATE%>content_frames.oft&CONT_ID=
  root.title=<%CAT_TITLE%>
  root.id=<%ROOTCONT_ID%>
  root.noc=<%NUMBER_CHILDREN%>
  <%query name=“ContListofCat_ID“ osfunction=“cbContListofCat_ID“>
  <%begindetail name=“ContListofCat_ID“%>
    <%PARENT_ID%>.<%COUNTER%>.id=<%CONT_ID%>
    <%PARENT_ID%>.<%COUNTER%>.type=<%CONT_TYPE%>
    <%PARENT_ID%>.<%COUNTER%>.title=<%CONT_TITLE%>
    <%PARENT_ID%>.<%COUNTER%>.noc=<%NUMBER_CHILDREN%>
  <%enddetail%>
<%enddetail%>
...
```

Abbildung 7.10: Category: Zugehörige Content-Liste (2)

Hierbei handelt es sich um eine geschachtelte Transaktion, die eine Beschreibung der gewählten Kategorie in einer speziellen Darstellung liefert. Im Gegensatz zu den bisher gezeigten Templates erzeugt dieses Template keinen HTML-Code, sondern lediglich eine Liste von <Key>=<Value> Paaren. Das Template erzeugt zunächst Angaben über den Wurzelknoten, anschließend wird dessen Content-Baum zurückgeliefert. Für jeden Content werden die ID, der spezielle Typ, sowie Titel und Anzahl der „Kinder“ festgehalten. <%PARENT_ID%> ist jeweils die ID des Vaters. Das Applet liest die so formatierte Antwort und erzeugt die Visualisierung des navigierbaren Inhaltsbaumes im Browser. Abbildung 7.11 zeigt das Resultat am Beispiel eines Ausschnitts von Abbildung 7.5.

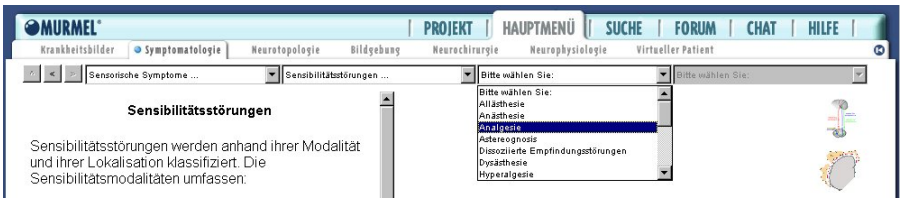

Abbildung 7.11: Oberflächenausschnitt: Navigation durch Wissensbäume

Bis zu diesem Zeitpunkt wurden zu allen Contents einer Kategorie genau die Attribute an den Client gesendet, die für eine erste Übersicht ausreichend sind. Da es sich nur um kleine Datenmengen handelt, erfolgt die Umsetzung schnell und verursacht wenig Netzbelastung.

Das für die Navigation innerhalb von Kategorie-Bäumen eingesetzte Java-Applet besitzt sowohl ein „Gedächtnis“, als auch „Intelligenz": Es speichert die zu einer Kategorie gelesenen Content-Daten für die gesamte Lebensdauer des Applets und fordert einen Content-Baum nur dann erneut von der Datenbank

an, wenn ein vorhandener Zeitstempelvergleich ergibt, dass der in der Datenbank gespeicherte Content-Baum aktualisiert wurde. „Intelligent“ ist es insofern, dass es intern eine History- Liste der Contents verwaltet, die vom Benutzer bereits im Laufe der Arbeitssitzung mit dem e-Learning-System ausgewählt worden waren, und diese eigenständig rekonstruieren kann.

Die Präsentation eines Contents erfolgt durch Anzeigen seines Textes, der als Datei im Filesystem des Hostrechners abgelegt und in der Datenbank nur referenziert wird (vgl. Abschnitt 6.4.2) und durch die Liste der zugehörigen MMObjects. Zu jedem Medium wird zunächst die verkleinerte Fassung mit zugehörigem ALT-Text dargestellt, welcher eine kurze im Zusammenhang mit dem Icon als „Tooltip“ sichtbare Bilderklärung enthält. Dies erleichtert die Entscheidung des Lernenden, welches MMObject letztendlich im mittleren großen Frame dargestellt werden soll. Zusätzlich werden der Legendentext und die Pfadangabe für die restlichen Medien vom Datenbankserver geladen. Erst bei Auswahl eines Icons wird eine zweite Anfrage an den Webserver des Hostrechners gestellt, um für die Normalansicht des Mediums die entsprechende Datei zu laden. Wie oben beschrieben wird dies ohne Zutun des Datenbankservers ausgeführt.

Nach dem in diesem Abschnitt vorgestellten Prinzip werden sämtliche Datentypen vom Server auf den Client übertragen. Alle technischen Vorgänge laufen dabei völlig transparent für den Anwender ab.

7.2.4 Zusammenfassung

Im Hinblick auf die Akzeptanz eines e-Learning-Systems durch seine Benutzer sind eine einfach und intuitiv nutzbare Bedienungsoberfläche und gute Performanz beim Datenzugriff wichtige Faktoren. Die Entwicklung des eLS-Baukastens muss darüber hinaus den Anforderungen Rechnung tragen, die daraus resultieren, dass e-Learning-Systeme für unterschiedlichste Anwendungsbereiche generiert werden sollen. In diesem Abschnitt wurde gezeigt, wie dies auf strukturierte und effiziente Weise realisiert werden kann. Dazu wird mit Hilfe von Templates und einer geschickt implementierten Service Applikation ein grundlegendes Konzept für die Bedienoberfläche und die Art des Datenbankzugriffs zur Verfügung gestellt. Diese prinzipielle Organisation kann dann je nach Anwendungsbereich an spezielle Wünsche angepasst werden.

7.3 Autoren-Client

Der Autoren-Client, für den im Rahmen dieser Arbeit eine Schnittstelle zur Datenbank realisiert wurde, ist eine Java- Applikation zur graphischen Eingabe und Bearbeitung von Lerninhalten, die über dieselben Kommunikationsprotokolle mit dem Datenbankserver kommuniziert wie der Administrator- und der Benutzerclient. Dieser Abschnitt beschreibt die Funktionalität und spezifische Aspekte der Implementierung des Autorensystems.

7.3.1 Aufbau der Autorenoberfläche und Funktionalitäten

Aufbau, Struktur und Gestaltung der Oberfläche des Autorensystems orientieren sich an den in „Techniques for Authoring Tool Accessibility; Guidelines 1.0 „ der W3C WAI Authoring Tools Working Group zertifizierten Richtlinien [W3C00].

Nach dem Start der Javaapplikation muss sich der Autor zunächst bei der Datenbank anmelden. Als Resultat dieser Anmeldung liefert die Datenbank einen sog. „Access Control Key“ zurück, der bei allen weiteren Aufrufen als Parameter an die Datenbank mitübergegeben wird. Der Schlüssel dient der Datenbank zur Identifizierung des Autors und zum Schutz vor unberechtigten Zugriffen. Nach erfolgreicher

Anmeldung präsentiert sich dem Autor das System wie in Abbildung 7.12 dargestellt. Die eingefügten Zahlen von 1 bis 10 dienen der nachfolgenden Beschreibung der Oberfläche.

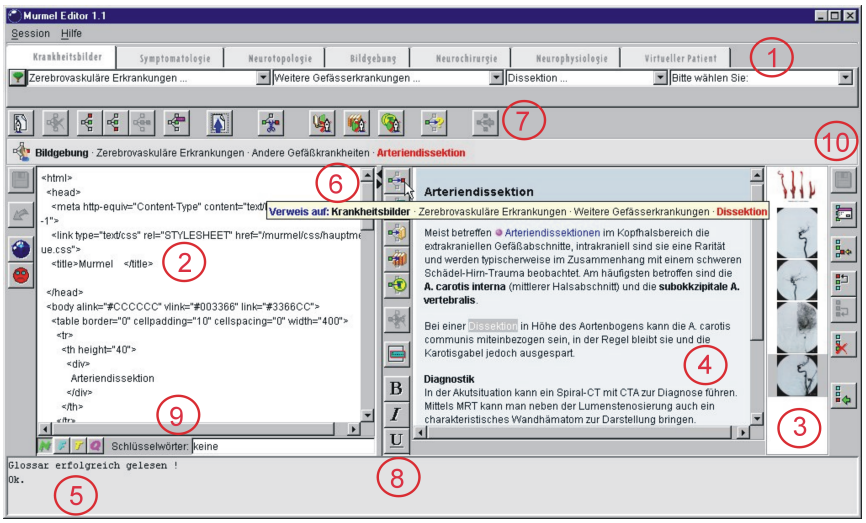

Abbildung 7.12: Autorensystem: Die graphische Benutzerschnittstelle

Die Oberfläche im Überblick
Im oberen Bereich des Autorensystems befindet sich die Navigationsleiste, in der Kategorien und Contents nach dem in Abschnitt 7.2 erläuterten Prinzip der Dropdown-Menüs dargestellt werden (1). Ein ausgewählter Content erscheint analog zu seiner Darstellung innerhalb der Benutzeroberfläche, d.h. im linken Frame wird der zugehörige Text angezeigt (2), ganz rechts die Medien-Voransicht eingeblendet (3). Zwischen diesen beiden Frames erscheint die jeweils aktuelle Version des editierten HTML-Textes (“WYSIWYG“) (4). Der untere Bereich der Oberfläche dient der Anzeige von Statusmeldungen an den Autor als Reaktion auf dessen Aktionen (5).

Die Schaltflächen
Umgeben sind die Editierfenster von zahlreichen Schaltflächen, mit deren Hilfe Text und Medien bearbeitet werden können. Alle Bedienknöpfe werden automatisch vom Programm je nach Zustand des Systems aktiviert bzw. deaktiviert, so dass mit ihnen stets nur sinnvolle Aktionen verknüpft sind (6). Die Knöpfe unterhalb der Navigationleiste sind mit Aktionen zum aktuellen Content oder zu Glossar- und Literatureinträgen belegt (7). Über diese lassen sich Inhalte anlegen, editieren und löschen. Während Glossar- und Literatureinträge grundsätzlich alphabetisch einsortiert werden, liegt die Einordnung eines Eintrags im Falle von Contents allein im Ermessen des Autors. Alternativ läßt sich auch ein bereits existierender Content-Teil-Baum durch Auswahl eines Knoten-Contents und anschliessende Anwahl des neuen Vater- bzw. Bruderknotens an eine beliebige andere Stelle innerhalb des Systems verschieben. Die angeschlossene Service Applikation prüft dabei die logische Korrektheit der neuen Content-Position. Beispielsweise wäre es nicht möglich, einen Teil einer GuidedTour in einen simulierten Handlungsablauf eines anderen Autors zu verschieben.

Bearbeitung von Text
Die inhaltliche Textbearbeitung erfolgt im linken großen Textfenster, während der interpretierte Text gleichzeitig im rechten Fenster dargestellt wird. Bei der Textgenerierung werden vorhandene HTML-Syntaxfehler direkt im Status-Fenster angezeigt. Über die mittlere Buttonleiste stehen Funktionalitäten zur automatischen Text-Layout-Gestaltung in Form verschiedener Formatierungsangaben zur Verfügung (8). Auf dieser Leiste befinden sich außerdem die Funktionalität zur Erzeugung von Verweisen, die als

typisierte Hyperlinks in den Text integriert werden. Derzeit stehen fünf Verweistypen zur Verfügung: auf einen anderen Content (des systematisches Wissens, eines Fallbeispiels, Quiz oder einer Simulation bzw. eines typischen Handlungsablaufs), auf Medien, Glossar- oder Literatureinträge und auf GuidedTours. Bei der Generierung von Verweisen wird von Seiten der Service Applikation die Existenz des Ziels überprüft und gegebenenfalls eine entsprechende Fehlermeldung ausgegeben. Sofern Link-Ziele zu einem späteren Zeitpunkt von einem Autor gelöscht werden, erhält der Autor, der auf einen solchen Eintrag verweist, eine entsprechende Warnung präsentiert, sobald er das Autorensystem erneut startet.

Unterhalb des Text-Editier-Fensters befinden sich schließlich Auswahlmöglichkeiten zur Klassifizierung von Contents als 'Normal' (systematisches Wissen), 'Fallbeispiel', 'Quiz' und 'Training' (Simulationen/typische Handlungsabläufe) und für die Eingabe von Schlüsselwörtern (9). Die Typisierung von Contents ermöglicht die differenzierte Behandlung unterschiedlicher Inhalte durch die zugehörige Service Applikation.

Bearbeitung von Medienobjekten
Eine Bearbeitung von Medienobjekten ist im rechten Teil der Oberfläche möglich (10): Hier existieren Knöpfe zum Anlegen, Editieren, Verschieben und Löschen von Medien, zur Angabe von verkleinerten, normalen und vergrößerten Ansichten, sowie zum Anlegen und Ändern von Keywords, Legenden und ALT-Texten. Abbildung 7.13 zeigt das Popup-Menü zum Editieren eines Medienobjekts.

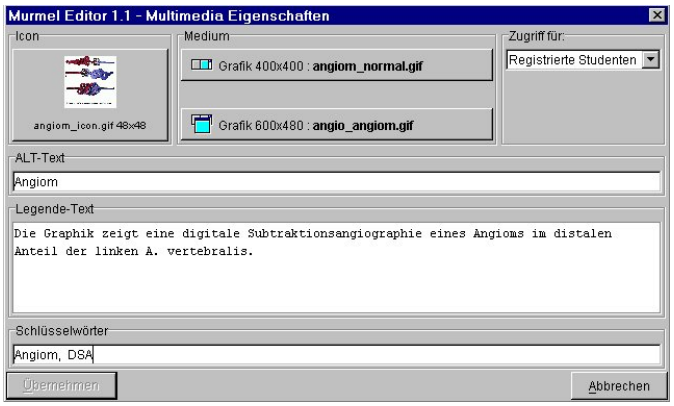

Abbildung 7.13: Das Autorensystem: Editieren eines MMObjects

Festlegen von Zielgruppen
Das Autorensystem gestattet die Definition von Zielgruppen, durch die der Zugang zu Inhalten beschränkt werden kann. Das heißt, es können Bereiche innerhalb des e-Learning-Systems definiert werden, die unabhängig von einer Registrierung allen Nutzern offen stehen. Ohne zusätzlichen Aufwand lassen sich genauso aber auch Bereiche exklusiv für einzelne Zielgruppen einrichten. Der Zugang zu bestimmten Informationen von Seiten der Nutzer wird dabei über Kenn- und Passwort geregelt.

Freigabe von Contents
Um zu verhindern, dass über die Benutzeroberfläche auf veraltete oder zu bearbeitende Inhalte zugegriffen wird, besitzt jeder Autor das Recht, jede einzelne seiner eigenen Seiten zu einem von ihm gewählten Zeitpunkt freizugeben bzw. zu sperren, beispielsweise während der Zeit der Aktualisierung eines Eintrags. Durch Absenden der editierten bzw. neu erstellten Inhalte werden diese für die jeweils definierte Zielgruppe freigeschaltet. Die neue Version ersetzt die bisherige Onlineversion, die alte Version wird separat gesichert.

Speicherung und Updates
Editierte Inhalte können jederzeit per Knopfdruck als Bytestrom an den Server geschickt werden, wo sie

von der zugehörigen Service Applikation an die Datenbank weitergereicht werden. Alle Modifikationen erfolgen grundsätzlich ad-hoc und an der Stelle, an der die veränderten Inhalte nach der Veröffentlichung auch in der Benutzeroberfläche zu sehen sind. Die Service Applikation überprüft dabei alle gesendeten Daten auf logische und syntaktische Korrektheit, bindet die Inhalte in die bisherige Systemstruktur ein und stößt die Schlüsselwort-, Text- und Bildindizierung an (vgl. Kapitel 8).

Für Medien wie Text, Bilder, Videos, Shockwave-Animationen oder Audiodateien, die im Rahmen eines neuen oder editierten Datenbankobjekts in die Datenbank übertragen werden sollen, wird bei Anstoßen des Speichervorgangs auf Clientseite im Hintergrund ein ftp-Prozess gestartet, der die zugehörige Datei vom Client auf den Server überträgt. Dort wird sie in einem speziell für Autoren vorgesehenen Verzeichnis zwischengespeichert. Das Autorensystem kontrolliert das Vorhandensein der zu übertragenden Dateien, der Adminstrator eines e-Learning-Systems ist für das Vorhandensein des entsprechenden Autorenverzeichnisses verantwortlich. Anschließend werden nach dem üblichen Vorgehen alle zur Speicherung des Datenbankobjekts erforderlichen Parameter an die Datenbank gesendet, u.a. auch der Dateiname des zugehörigen Medienobjekts. Während der Transaktion zur Abspeicherung des Datenbankobjekts kopiert die Applikation anhand des ihr bekannten Dateinamens die entsprechende Datei vom ftp-Verzeichnis in ein von außen nicht zugängliches, vollständig von der Datenbank kontrolliertes Verzeichnis, das nur die Dateien des aktuellen Autors enthält. Bei Beendigung einer Autorensystem- Sitzung werden alle vom Autor übertragenen Dateien aus dem ftp-Verzeichnis entfernt. Der gesamte Vorgang läuft transparent für den Autor im Hintergrund ab.

Modellierung didaktischer Konzepte
Über die Schaltflächen unterhalb des Textfensters (9) kann mit dem Ziel der Festlegung einer bestimmten didaktischen Lernstrategie einem Content ein spezieller Typ zugeordnet werden. Das Autorensystem kontrolliert die logische Richtigkeit von Typisierungen, d.h., ähnlich wie beim Verschieben von Teilbäumen wird beispielsweise beim Aufbau eines simulierten Handlungsablaufs geprüft, ob alle verknüpften Contents vom selben Typ sind. Zusätzlich erwünschte Lernstrategien können von den Autoren durch die individuelle Gestaltung von Contents und Medienobjekten, deren Kombination und die flexible Art der Verknüpfung verschiedener Inhaltstypen realisiert werden.

Kooperatives Arbeiten
Neben der Funktionalität zum Editieren von Inhalten unterstützt und koordiniert das Autorensystem die Teamarbeit mehrerer Autoren, die parallel auf das System zugreifen. Es wird insbesondere sichergestellt, dass jeder Autor stets eine aktuelle Sicht auf sämtliche Inhalte des Systems hat; Modifikationen sind nur an den eigenen Inhalten möglich. Für bestimmte Bereiche existiert darüber hinaus jeweils ein verantwortlicher Autor, der die Rechte für alle Contents dieses Bereichs besitzt. Schließlich existiert eine umfangreiche Report-Funktion, die für jeden Autor Abfolge und Art seiner Aktivitäten protokolliert. So lassen sich jederzeit über jeden Autor wichtige Informationen aller vorgenommenen Transaktionen nachvollziehen.

7.3.2 Implementierungsdetails

Im Gegensatz zum Administrator- und Benutzer-Client basiert die Implementierung des Autoren-Clients nicht auf HTML, JavaScript und Java, sondern ist eine pure Java-Applikation. Ein Autor, der multimediale Inhalte eingibt, verändert und zu komplexen Konstrukten kombiniert, benötigt eine Reihe von Funktionalitäten, die allein mit HTML/JavaScript nur schwer oder gar nicht zur Verfügung gestellt werden können. Java wurde eingesetzt, da die Sprache zwei wichtige Anforderungen erfüllt: sie ist plattformunabhängig und die Programmierung verteilter Anwendungen wird unterstützt [Arn00, Jia00, Ste01]. Die Implementierung des Autorensystems ist somit unabhängig von Hardware und Betriebssystem.

Die Plattformunabhängigkeit von Java wird durch die Verwendung von sog. Bytecode erreicht, d.h. eine Java-Applikation wird nicht in Maschinencode für einen realen Prozessor übersetzt, sondern in By-

tecode für eine fiktive Maschine. Dieser Code wird von der *Java Virtual Machine* ausgeführt.

Java ist in der Lage, nicht nur Daten, sondern auch echte Funktionalität über das Netz zu übertragen. Dadurch wird die Möglichkeit einer Lastverteilung zwischen dem Server, auf dem Datenbank und Service Applikationen installiert sind, und den Clientrechnern, die von den Autoren benutzt werden, geschaffen.

Java unterstützt die Anwendungsentwicklung durch umfangreiche Standardbibliotheken. Diese bieten neben gebräuchlichen Datentypen vor allem Klassen, die zur Programmierung verteilter Anwendungen und graphischer Benutzerschnittstellen eingesetzt werden können. Dies umfaßt insbesondere Klassen für Uniform Resource Locators (URLs) und die Kommunikation über HTTP, so dass das Autorensystem auf einfache Weise über HTTP-Requests mit den entsprechenden serverseitigen Templatedateien und der zugehörigen Service Applikation kommunizieren kann. Dadurch kann die Kommunikation zwischen Autoren-Client und Service Applikation auf die gleiche Art und Weise über Templates realisiert werden, wie die Kommunikation zwischen Administrator- bzw. Benutzer-Client und Server.

Im Falle von Adminstrator- und Benutzer-Client bestimmt der Inhalt der Templatedateien das graphische Layout der Oberfläche, d.h. der entsprechende Webbrowser interpretiert die in der Templatedatei enthaltenen HTML-, JavaScript- und Java-Anweisungen und stellte das Resultat graphisch aufbereitet dar. Im Falle des Autorensystems werden die Templatedateien dagegen von der Java-Applikation interpretiert. Dazu wurde ein Protokoll definiert, das den generellen Aufbau für generierte Antworten von Serverseite festgelegt:

1. Zeile: eine Kennung, um den Inhalt als Antwort der Datenbank zu identifizieren. Dies ist zur Erkennung von Verbindungsfehlern notwendig, wie sie beispielsweise durch Proxyserver entstehen können.

2. Zeile: ein Return-Code, der anzeigt, ob ein Datenbank-Request erfolgreich durchgeführt wurde.

3. Zeile: eine Meldung, deren Aufbau vom vorangegangenen Return-Code abhängt.

4. und folgende Zeilen: die angefragten Daten.

Dieses Prinzip der Kommunikation soll durch das bereits aus Abschnitt 6.5.4 bekannte einfache Beispiel zum Lesen eines Content-Baumes verdeutlicht werden: Die Antwort einer Datenbankanfrage nach beispielsweise der Liste aller aktuellen Kategorien innerhalb des medizinischen e-Learning-Systems MURMEL zeigt Abbildung 7.14.

```
#!MURMEL
0
Ok.
2
1*Krankheitsbilder*http://.../kb.gif*
2*Symptomatologie*http://.../sympto.gif*
```

Abbildung 7.14: Beispiel einer Datenbank-Anfrage nach Kategorien im Autorensystem

Der Ausdruck „#!MURMEL“ stellt die Kennung dar, die zweite und dritte Zeile zeigen an, dass die Datenbankanfrage erfolgreich bearbeitet werden konnte, ab der vierten Zeile folgen die Daten. Diese Datenzeilen werden von der Java-Applikation geparst und zur Erzeugung entsprechender Java-Objekte benutzt. Im vorliegenden Beispiel enthalten die Zeilen Kategorienummer, Name und Pfad des zugehörigen Kategorie-Icons. Zur Anzeige beispielsweise aller Inhalte der zurückgelieferten Kategorie „Symptomatologie“ besteht die von Serverseite zurückgelieferte Templatedatei aus der in Abbildung 7.15 (a) gezeigten Abfolge, Abbildung 7.15 (b) zeigt den zugehörigen Baumausschnitt.

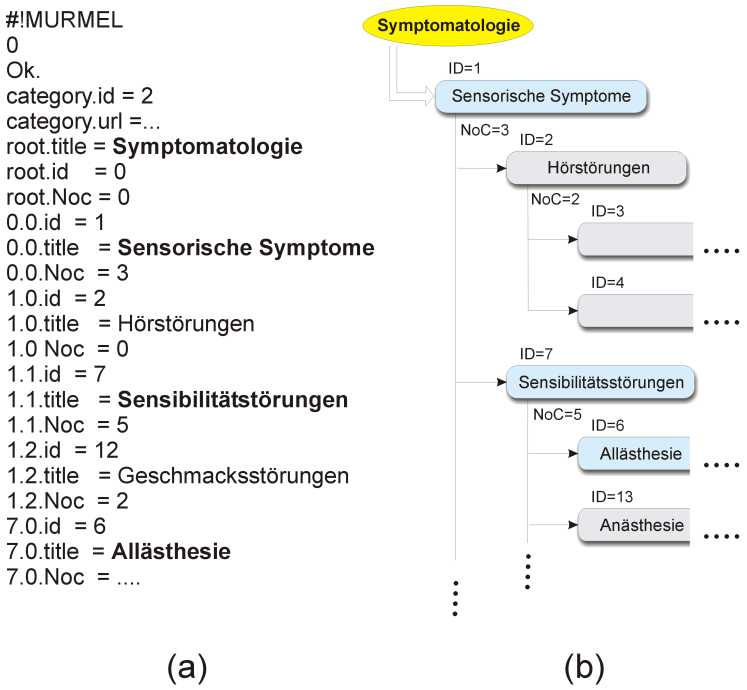

Abbildung 7.15: Autorensystem: Symptomatologie-Baum

Die interpretierte Templatedatei beinhaltet einige Daten zur aktuellen Kategorie und die URL, unter der alle Inhalte abrufbar sind, dann folgen die Daten der zugehörigen Content-Knoten des Baums. Zu jedem Content werden Titel, ID und Anzahl der Unterknoten (“Noc“ = number of children) angegeben. Hierbei setzt sich der jeweils eindeutige Schlüssel aus *<Nummer des Elternknotens>.<Laufende Nummer des Unterknotens>.<Item Name>* zusammen. Aus diesen Daten wird vom Autorensystem ein Baum von Java-Objekten erzeugt, der zur Navigation in den Inhalten dient.

Analog zu dem in Abschnitt 6.4.2 beschriebenen Zugriff des Templateprozessors auf eine Headerdatei, aus der aktuelle Daten wie Hostname, Pfadnamen, etc. gelesen werden, werden zur Kommunikation zwischen clientseitigem Autorensystem und Server drei Konfigurationsdateien verwendet. Die aus den drei Dateien ausgelesenen Daten beinhalten unter anderem:

1. Die Konfiguration der Internetverbindung des Autorensystems zur Datenbank. Dies ist die Hostadresse der Datenbank, sowie die Daten gegebenenfalls zu benutzender HTTP- bzw. FTP-Proxies.

2. Die Konfiguration der DB-Kommunikation. Hier sind im Wesentlichen die Aufrufe der Schnittstellen mit ihren Parametern definiert.

3. Die Konfiguration der graphischen Benutzeroberfläche des Autorensystems. Hier werden benutzerspezifische Einstellungen gespeichert, die beim nächsten Aufruf des Autorensystems wieder zur Verfügung stehen.

Grundsätzliche Philosophie des Autorensystems ist der modulare Aufbau aller Komponenten mit einer relativ schmalen Schnittstelle zum Server. Dies wird durch die Verwendung von HTTP-Requests und des Datenbank-CGI-Skripts sichergestellt, wodurch eine flexible Anbindung des Autorensystems auch an andere Systeme ebenso wie die individuelle Erweiterung der Module ermöglicht wird. Abbildung 7.16 zeigt abschließend zu diesem Unterabschnitt den Klassenaufbau des Autorensystems.

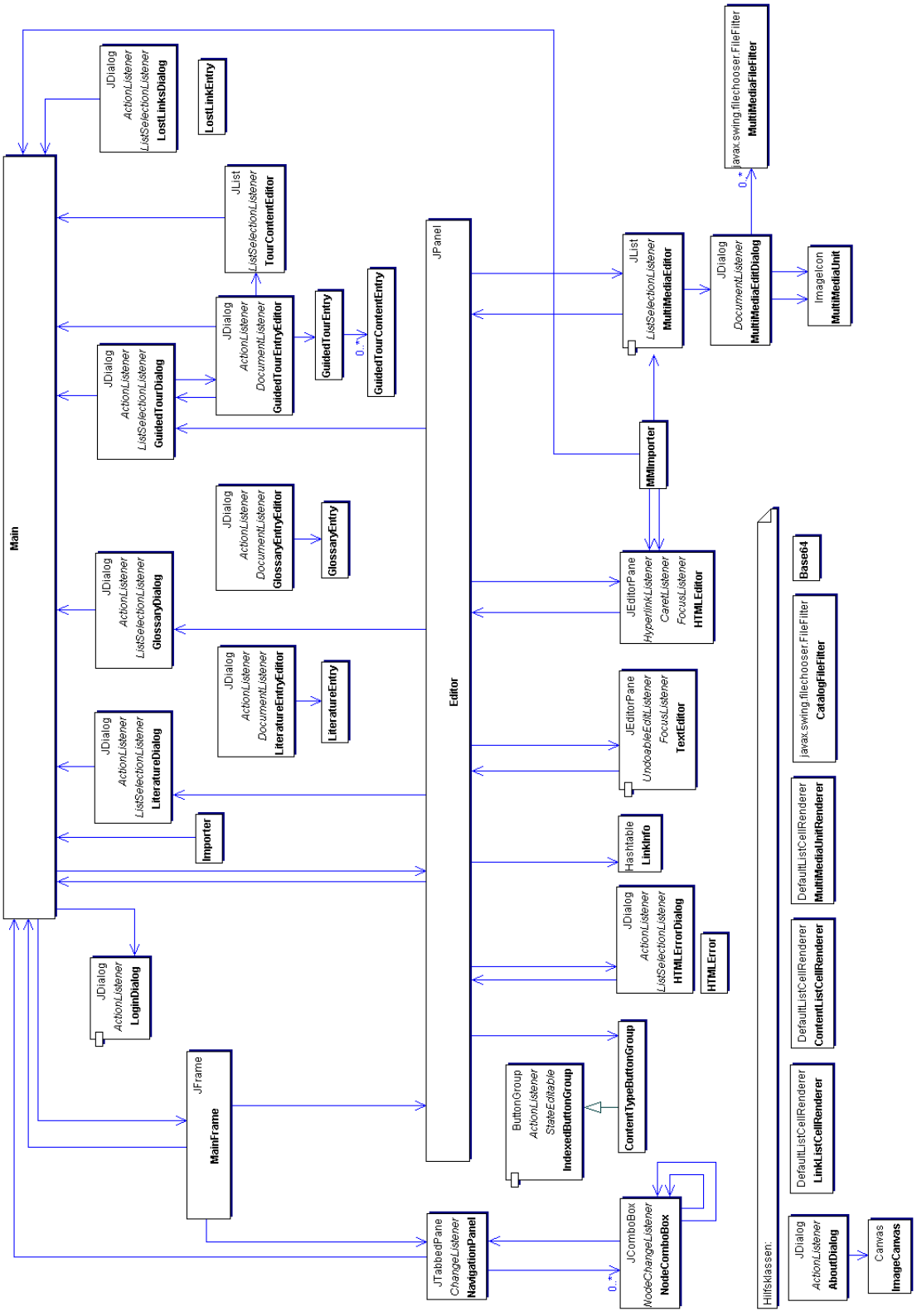

Abbildung 7.16: Autorensystem: Das Klassenmodell

7.3.3 Das Autorenmodell

Für jeden Autor, der vom Administrator registriert wird, werden zahlreiche personen- und inhaltsbezogenen Informationen gespeichert. Abbildung 7.17 zeigt das diesbezügliche Klassenmodell.

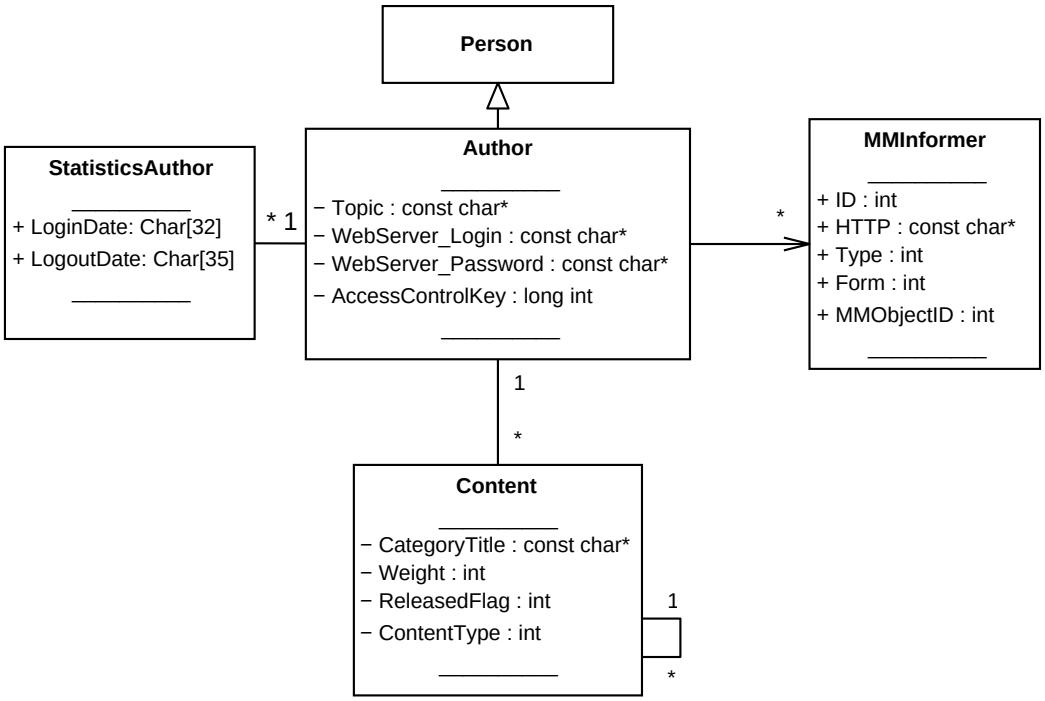

Abbildung 7.17: Das Autorenmodell

Nicht nur auf Client-, sondern auch auf Serverseite werden so Informationen zum Autor wie etwa Name, Login, Passwort, Email, oder Status protokolliert. Dies wird durch die Klasse **Person** ermöglicht, (s. Abbildung 7.18), von der die Klasse **Author** erbt.

```
class Person
{
public:
   int m_ID;
   int m_Name;
   int m_Login;
   int m_Password;
   int m_Email;
   int m_Type;
   int m_Status;
   Char[32] m_CreationDate;
}
```

Abbildung 7.18: Autorenmodell: Die Klasse Person

Neben diesen Attributen beinhaltet die Klasse Author Attribute zur Beschreibung des Arbeitsgebiets eines Autors (*Topic*), außerdem kann ein zusätzliches Login und Passwort für den Fall gespeichert werden, dass dieses für den Server, auf dem das generierte e-Learning-System installiert ist, erforderlich ist. Zusätzlich werden für jeden Autor Details des Zugriffs über das Autorensystem auf die Datenbank protokolliert. Dies wird durch die Klasse **StatisticsAuthor** bewerkstelligt. Zur Protokollierung verschiedener Informationen

zu Inhalten, die ein Autor erstellt hat, führt die Klasse **MMInformer** Buch, beispielsweise welche Medien der Autor an welchen Stellen verwendet hat, auf welche anderen Einträge er verweist, etc. Beispielsweise im Fall des Löschens eines Datenbankinhalts kann so effektiv kontrolliert werden, ob ein Datei auch physikalisch aus dem Dateisystem gelöscht werden kann.

7.3.4 Zusammenfassung

In diesem Abschnitt wurde das Autorensystem beschrieben, das standardmäßig mit in den eLS-Baukasten integriert ist. Es ermöglicht eine besonders komfortable und endbenutzertaugliche Erfassung und Bearbeitung sowie redundanzfreie und plattformunabhängige Organisation aller multimedialen Inhalte, die in einem generierten e-Learning-System bearbeitet werden sollen. Die Oberfläche ist intuitiv benutzbar und ermöglicht die Generierung unterschiedlichster Lernmodule. Technische Transaktionen laufen transparent im Hintergrund ab, so dass keinerlei besonderen Kenntnisse im Umgang mit dem Rechner und dem Programm erforderlich sind.

7.4 Gezielte Benutzerunterstützung

Die mit Hilfe des Autorensystems erzeugten multimedialen Lerninhalte eines e-Learning-Systems sind didaktisch so gestaltet, dass sie den individuellen Lernprozess eines Benutzers optimal unterstützen. Daneben sollte jedes e-Learning-System aber auch über spezielle Werkzeuge zur Unterstützung in der Handhabung eines solchen Systems verfügen. Unter dem Begriff „Benutzerunterstützung“ wurden in Abschnitt 2.4 alle technischen, inhaltlichen und didaktischen Softwarekomponenten zusammengefaßt, die jedem Benutzer eines e-Learning-Systems eine individuelle Arbeitsunterstützung bieten. Die im Rahmen des eLS-Baukastens implementierten Werkzeuge setzen dabei die Existenz eines geeigneten Benutzermodells voraus, welches im Folgenden beschrieben wird. Auf der Basis dieses Modells konnten Methoden realisiert werden, die sowohl adaptive als auch adaptierbare Eigenschaften aufweisen. Grundlegend bei deren Entwicklung war die Zielsetzung, sowohl die Benutzerunterstützung als auch die im nachfolgenden Abschnitt beschriebene Evaluierung von e-Learning-Systemen auf denselben Modellen und Komponenten basieren zu lassen.

7.4.1 Benutzermodellierung

Abbildung 7.19 zeigt das innerhalb des eLS-Baukastens implementierte Benutzermodell. Die Klasse **Person** ist Basis nicht nur der Klasse Author (vgl. Abschnitt 7.3.3), sondern ebenfalls der Klasse **User**. Sie umfaßt alle Attribute, die zu jedem Anwender des Systems protokolliert werden sollen: Die *ID* ist eine eindeutige Kennung für jeden Anwender, die zusätzlich im Rahmen von Evaluierungen genutzt wird, um Daten zu anonymisieren. *Name*, *FirstName* und *Gender* dienen statistischen Zwecken, *Login* und *Passwort* sind optionale Angaben, die für die Organisation der Zugangsbeschränkung benötigt werden. Die Rechte eines Anwenders werden durch das Attribut *Type* und *Status* festgelegt. Das Attribut *Email* dient schließlich als Kontaktadresse, *CreationTime* protokolliert, wann die User-Instanz eines Anwenders im System angelegt wurde.

Die Klasse **User** ergänzt die Person-Klasse um Attribute, die für Benutzer spezifisch sind. Dazu gehört beispielsweise das Attribut *Level*, welches den Ausbildungsstand eines Benutzer kennzeichnet, eine Kennung, welche angibt, ob der Nutzer per Email über Neuigkeiten im Zusammenhang mit dem e-Learning-System informiert werden möchte (Attribut *News*), und das Attribut *Notes* für aufsummierte Punktzahlen bei Wissensabfragemodulen und bearbeiteten Simulationen.

Zu jedem Benutzer, der sich mit Login und Passwort bei einem e-Learning-System registrieren läßt, wird eine Instanz der Klasse User persistent in der Datenbank angelegt. Dies geschieht auch für anonym

angemeldete Nutzer, deren Daten werden jedoch nicht langfristig gespeichert, sondern nach einer eventuellen internen statistischen Auswertung gelöscht.

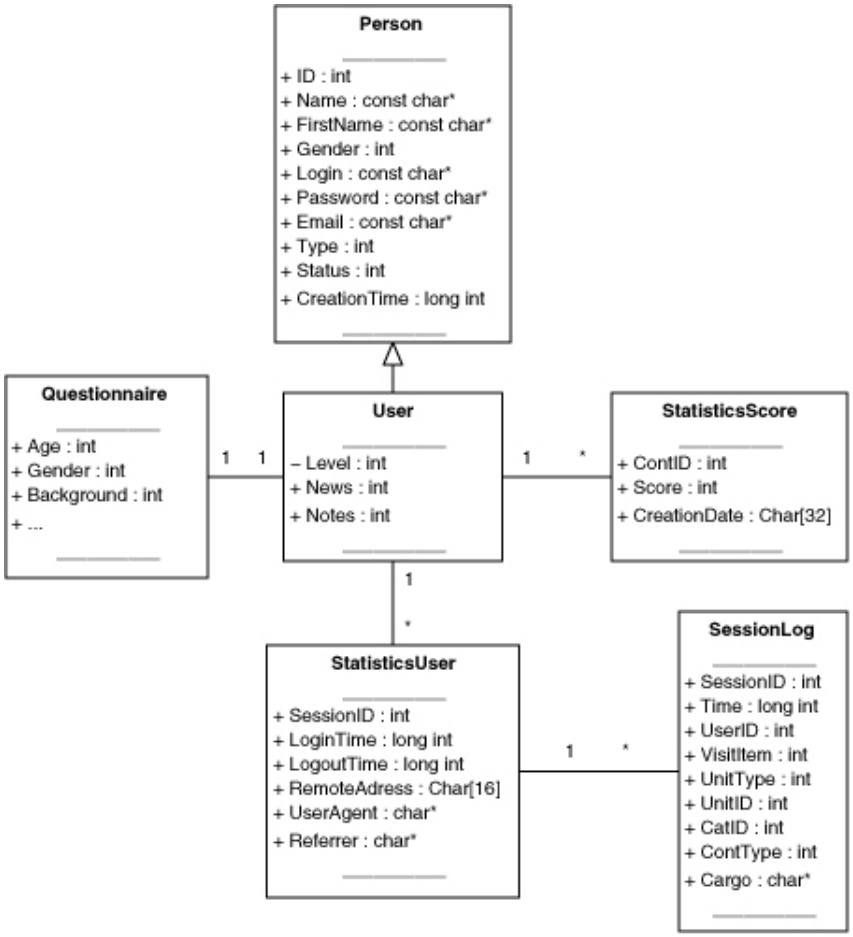

Abbildung 7.19: Benutzermodell für e-Learning-Systeme nach [Mor01]

Zur Ermittlung benutzerspezifischer Informationen ist die Klasse User über eine 1:1 Relation mit der Klasse **Questionnaire** verbunden. Dies erfolgt im Hinblick auf einen elektronische Fragebogen, den Benutzer jederzeit und über mehrere Sitzungen verteilt ausfüllen können (vgl. Abschnitt 7.5). Questionnaire speichert die Antworten zu insgesamt 38 Fragen des weiteren enthält die Klasse das Erzeugungs- und Aktualisierungsdatum des Fragebogens, sowie Attribute über den aktuellen Ausbildungsstand des Benutzers. Zur globalen Auswertung im Rahmen von Evaluierungen wird der Questionnaire auch dann gespeichert, wenn ein Nutzer anonym im System angemeldet war.

Instanzen der Klasse **StatisticsScore** werden angelegt, wenn ein Nutzer Wissensüberprüfungsmodule nutzt. Dazu werden das aktuelle Datum, die erreichte Punktzahl und die eindeutige ID des entsprechenden Contents gespeichert. Die StatisticsScore-Instanzen aller Benutzer können wiederum einer globalen Auswertung zugeführt werden.

Bei jeder neuen Arbeitssitzung mit einem e-Learning-System wird eine neue Instanz der Klasse **StatisticsUser** für den Benutzer angelegt. Darin werden Informationen wie Datum des Einwählens, Remote Adresse, UserAgent (verwendeter Webbrowser, Sprache, Betriebssystem) und Referrer abgespeichert. Über eine 1:n Relation ist StatisticsUser mit der Klasse **SessionLog** verknüpft, die während einer Sitzung des Benutzers verschiedene relevante Größen wie etwa die Nutzungsdauer, aber auch die besuchten

Content- bzw. Lernstrategietypen, etc. erfaßt. Die Kombination der Attribute *VisitItem*, *UnitType* und *UnitID* beschreibt dabei eindeutig, welcher Inhalt besucht wurde und welchen Typ der Inhalt besitzt. *Cargo* dient als Container für verschiedene Zusatzinformationen. Beispielsweise können hier im Falle einer protokollierten Suche die Suchbegriffe abgelegt werden (vgl. Abschnitt 8.4). Auch hier werden alle Instanzen aller Benutzer für globale Analysezwecke verwendet.

Das Benutzermodell kann gleichzeitig der Gruppe der statischen wie auch der dynamischen Modelle (vgl. Abschnitt 2.4) zugeordnet werden. Es erinnert darüber hinaus an den in Abbildung 2.2 dargestellten „Interface“-Agenten, da ähnliche Funktionen auch durch das hier entworfene Benutzermodell unterstützt werden. Die im Folgenden beschriebenen Werkzeuge beruhen dabei jedoch nicht auf Verfahren der Künstlichen Intelligenz, sondern sind partiell heuristisch motiviert.

7.4.2 Werkzeuge zur Benutzerunterstützung

Im Hinblick auf Methoden, die den einzelnen Benutzer während der Arbeit mit einem e-Learning-System unterstützen sollen, wurden sowohl adaptive als auch adaptierbare Werkzeuge realisiert und stehen in allen mit Hilfe des eLS-Baukastens generierten Systemen zur Verfügung. Abbildung 7.20, als Beispiel aus dem medizinischen e-Learning-System MURMEL, gibt anhand eines Screenshots des sog. **Virtuellen Schreibtisches**, der in einem separaten Fenster dargestellt ist, zunächst einen Überblick über die vorhandenen adaptierbaren Werkzeuge.

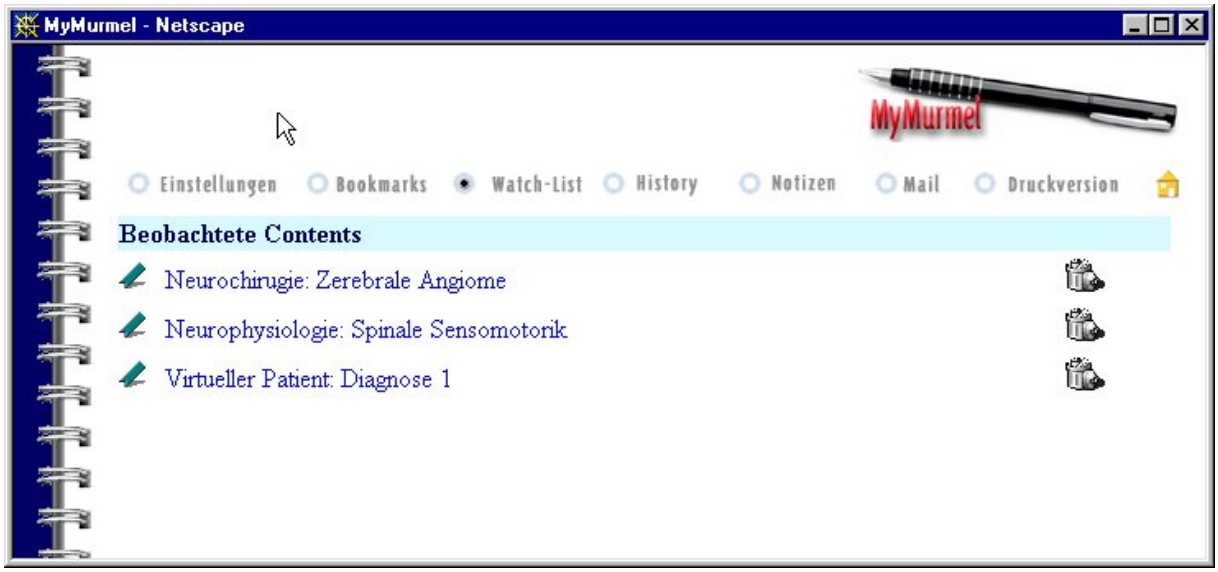

Abbildung 7.20: Der Virtuelle Schreibtisch

Im oberen Teil des Virtuellen Schreibtisches stehen alle verfügbaren Werkzeuge zur Auswahl, unterhalb davon wird das jeweils ausgewählte Instrument dargestellt. In Abbildung 7.20 ist momentan das Werkzeug zur Anzeige der Watchlist eines Benutzers aktiviert.
Insgesamt stehen, von links nach rechts betrachtet, folgende Funktionalitäten zur Verfügung:

- **Ändern der persönlichen Einstellungen**: Hier können verschiedene Angaben zum Benutzer, die in der Instanz der Klasse User abgelegt sind, geändert werden: beispielsweise der Name, die Emailadresse oder der gewählte Schwierigkeitsgrad für Inhalte und Quizmodule.

- Hinter dem Begriff **Bookmarks** verbirgt sich die Verwaltung einer persönlichen Lesezeichen-Liste für Contents des e-Learning-Systems. Die Verwaltung und Anzeige dieser Bookmarks wurde analog zu den Methoden, die in derzeitigen Webbrowsern verfügbar sind, implementiert. Bei Anwahl eines Eintrags wird der Zielinhalt in der e-Learning-System-Oberfläche dargestellt.

- Die Funktion **Watchlist** ermöglicht es einem Benutzer, Inhalte zu „beobachten“, d.h. über Änderungen und Neuheiten zu von ihm festgelegten Themenstellungen informiert zu werden.

- **History** offeriert einem Benutzer den Einblick in bereits abgeschlossene Aktionen, besuchte Inhalte und den Zeitpunkt des letzten Aufrufs.

- Die **Notizfunktion** macht die persistente Ablage von Annotationen zu beliebigen Contents möglich. Diese sind editierbar und ausdruckbar.

- Das Werkzeug **Mail** führt in den Kommunikationsbereich des e-Learning-Systems. Neben den bereits erwähnten kooperativen Funktionalitäten wie Chat und Foren werden hier beispielsweise automatisch Nachrichten für den individuellen Benutzer generiert, sofern von Autoren Einträge bestimmter Kategorien modifiziert wurden.

- Hinter dem Menüpunkt **Druckversion** verbirgt sich die Funktionalität festzulegen, welche Teilkomponenten eines Gesamtcontents für eine Druckversion kombiniert und aufbereitet werden sollen.

Im Folgenden wird aufgelistet, inwiefern bei den durch den eLS-Baukasten generierten Systemen auch Formen der Adaptivität, d.h. der indirekten Anpassung des Systems an den Benutzer, verwirklicht sind. Das erarbeitete Konzept geht dabei davon aus, dass Lernen und die Lernergebnisse nicht vollständig von außen geplant und vorhergesagt werden sollen, sondern einen selbstgesteuerten und selbstverantworteten Prozess darstellt. Die Rolle des e-Learning-Systems wird in diesem Kontext nicht als direktes, sondern vielmehr als indirektes Einwirken auf den Lernprozess verstanden. Dies wird durch folgende Konzepte unterstützt:

- **Makroadaptation**: Durch die Kenntnis über den aktuellen Ausbildungsstand eines Benutzers zusammen mit der vom Autor festzulegenden Gewichtung für einen Content existiert in jedem generierten e-Learning-System die Möglichkeit der automatischen Anpassung des Systems an den Ausbildungsstand individueller Benutzer. Dies ist gleichermaßen für das systematische und das problemorientierte Wissen möglich.

- Die unterschiedlichen **Varianten des Lernzugriffs** unterstützen sowohl eigenständiges wie auch geführtes Lernen mit einem e-Learning-System. Dies nimmt Bezug auf individuelle Interessen und Bedürfnisse der Benutzer.

- **Wissensüberprüfung**: Vom System aufgedeckte Wissenslücken können durch die Bearbeitung vorgeschlagener Inhalte im System geschlossen werden. Hierzu werden dynamische Verweise generiert, die vom Lernenden direkt weiterverfolgt werden können (Mikroadaptation).

Mit Ausnahme des Kommunikationstools sind alle Werkzeuge direkt über die benutzerseitige Service Applikation an die Datenbank gekoppelt. Von dort können die Daten auch für statistische Auswertungen oder Evaluierungen gelesen werden.

7.4.3 Zusammenfassung

Benutzermodellierung, Adaptivität und Adaptierbarkeit sind wirksame Hilfsmittel, um den Lernerfolg einzelner Lernender zu verbessern und allgemein den Umgang mit e-Learning-Systemen zu erleichtern.

Aus diesem Grund wurden in die Client-Applikationen des eLS-Baukastens die in diesem Abschnitt beschriebenen Werkzeuge integriert. Sie stehen in jedem generierten e-Learning-System zur Verfügung und schaffen eine praktisch anwendbare und effektive Möglichkeit zur individuellen Benutzerunterstützung.

7.5 Entwickelte Werkzeuge zur Systemevaluierung

Um Aktualität und Attraktivität der generierten e-Learning-Systeme sicherzustellen und zu überprüfen, werden verschiedene Formen der Evaluierung unterstützt. Das Gesamtkonzept beruht auf einer modularen Kombination von studienunabhängigen und studienabhängigen Verfahren. Die studienunabhängigen Evaluierungsmethoden kamen bereits während der Entwicklung des eLS-Baukastens selbst zum Einsatz, studienabhängige Verfahren sind nur mit generierten e-Learning-Systemen möglich, die bereits über Inhalte verfügen. Alle Instrumente eignen sich dabei gleichermaßen für die **formative** und die **summative** Evaluierung. Erstere dient dazu, während des Aufbaus eines e-Learning-Systems Evaluierungsdaten zu erheben und diese direkt in den Prozess der Systementwicklung einfließen zu lassen. Die summative Evaluierung dient der Qualitätskontrolle eines fertiggestellten e-Learning-Systems im realen Einsatz. Für generierte e-Learning-Systeme stehen insgesamt die in Tabelle 7.1 dargestellten Methoden zur Verfügung.

Studienunabhängige Verfahren
Expertenbefragung
Kriterienkatalog
Studienabhängige Verfahren
Verhaltensrecording
elektronischer Fragebogen
Papierfragebogen
Interviews
Verhaltensbeobachtung
Testergebnisse von Quizmodulen

Tabelle 7.1: Evaluierung im Rahmen des eLS-Baukastens

Bei der Expertenbefragung wird ein menschlicher Experte gebeten, die von Administrator und Autoren entworfenen Inhalte und Funktionalitäten zu testen und unter verschiedenen Aspekten zu bewerten. Geeignet ist diese Methode besonders während der formativen Evaluierung, da das Feedback der Experten direkt in den weiteren Verlauf des Wissensaufbaus einfließen kann.

Die schon bei der Entwicklung des eLS-Baukastens berücksichtigten Qualitätskriterien zum Entwurf und der Gestaltung von e-Learning-Systemen [W3C00, Sch99b] nehmen auch bei generierten e-Learning-Systemen eine wichtige Rolle ein. Hervorzuheben sind hier vor allem Kriterien, die sich auf die inhaltlichen Bereiche beziehen, die weitestgehend unkontrolliert vom System gestaltet werden.

Die im vorhergehenden Abschnitt beschriebene Benutzermodellierung und -protokollierung ermöglicht die Auswertung eines umfassenden Verhaltensrecordings, das im Hinblick auf eine Systemevaluierung wertvolle Hinweise auf Benutzerakzeptanz und Schwächen des Systems bietet. Hierbei werden alle durchgeführten Aktionen der einzelnen Nutzer in Form differenzierter Daten automatisch erfaßt. Das implementierte Verfahren gibt beispielsweise Aufschluß über besuchte Inhalte, durchlaufende Lernpfade, durchgeführte Interaktionen oder die verwendeten Werkzeuge.
Ein elektronischer Fragebogen ergänzt das Benutzermodell um Daten, die nicht automatisch erfaßt werden können. Neben personenbezogenen Daten werden hier vor allem Fragen zur Feststellung von Lernerfolg, Lerntransfer und Lernmotivation erfaßt. Abbildung 7.21 zeigt einen Ausschnitt des Fragebogens. Im Gegensatz zu dem im Hintergrund ablaufenden Verhaltensrecording ist der Erfolg hier jedoch von der Motivation der Lernenden abhängig, den Fragebogen auch ausfüllen zu wollen.

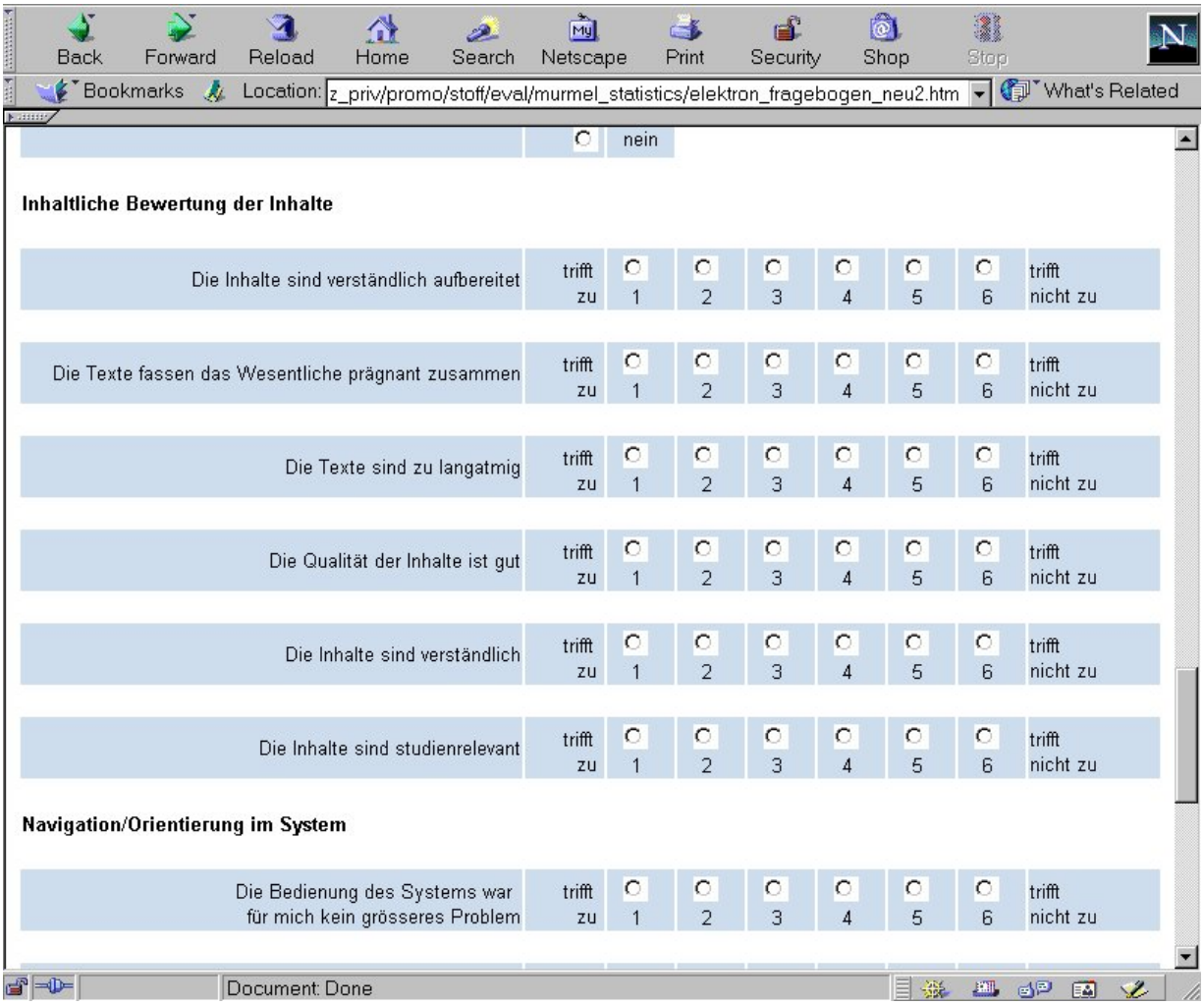

Abbildung 7.21: Ausschnitt aus dem elektronischen Fragebogen

Mit Hilfe des Fragebogens werden Benutzer strukturiert zu Motivation, Akzeptanz und erwartetem Lernerfolg befragt. Im Vergleich zu konventionellen Evaluierungstechniken kann so aufgrund der Einfachheit und Integration in den „Virtuellen Schreibtisch“ eines jeden Benutzers die Ausfüllbereitschaft erhöht werden. Daneben ermöglicht der Fragebogen eine komplett automatische Erfassung, Ablage, Präsentation und Analyse aller Daten.

Neben dem elektronischen Fragebogen wurde außerdem ein Papierfragebogen entworfen, der schwerpunktmäßig Fragestellungen behandelt, die im generellen Zusammenhang mit computerbasiertem Lernen stehen. Tabelle 7.2 zeigt einige dieser Fragen.

Papierfragebogen zur Evaluierung	
Alter	
Geschlecht	w/m
Ausbildungsstand	
Eigener Computer	J/N
Zugang zum Internet	J/N
Nutzung des Internets	
Welche Dienste werden genutzt	
Bedeutung von Computern und IuK-Technologien	J/N
Bedeutung des Internets für die Ausbildung	
Erwartungen an computergestütztes Lernen	
...	

Tabelle 7.2: Fragebogen zur Evaluierung computerbasierten Lernens

Um den Umgang der Lernenden mit einem aufgesetzten e-Learning-System zu überprüfen, wurden im Rahmen des eLS-Baukastens desweiteren Interviews und strategische Vorgehensweisen für Verhaltensbeobachtungen entwickelt. Im Rahmen von Interviews werden Informationen und Einschätzungen der Lernenden zu technischen, inhaltlichen und didaktischen Themenbereichen erhoben. Bei Verhaltensbeobachtungen werden dagegen wahrnehmbare Verhaltensweisen und Reaktionen von Lernenden während einer Sitzung mit einem e-Learning-System erfaßt. Diese Formen der Evaluierung spielen trotz der nur qualitativ möglichen Ergebnisse eine ähnlich wichtige Rolle wie quantitative Evaluierungswerkzeuge, da hier der Interviewer direkt beobachten kann, wo in Bezug auf die Systembedienung Schwierigkeiten und somit Ansätze für Verbesserungen existieren. In Kapitel 9 werden konkrete Beispiele zu den hier vorgestellten Evaluierungswerkzeugen gegeben.

7.6 Zusammenfassung

Gegenstand dieses Kapitels war die Beschreibung der Client-Applikationen, mit denen verschiedene Anwender auf e-Learning-Systeme zugreifen. Diese Applikationen bilden die Schnittstelle zum Anwender und haben dadurch wesentlichen Einfluß auf die Akzeptanz generierter e-Learning-Systeme. Neben einer kurzen Beschreibung des Administrator-Clients lagen weitere Schwerpunkte des Kapitels auf der Erläuterung des Benutzer-Clients und des Autorensystems.

Wie in diesem Kapitel gezeigt werden konnte, gestattet die realisierte Client-Server-Architektur die Realisierung leicht bedienbarer Client-Applikationen bei gleichzeitig effizienter Datenhaltung auf Serverseite.

Zur individuellen Unterstützung der Lernenden wurden ein Benutzermodell und verschiedene Werkzeuge realisiert, die jeden Benutzer sinnvoll in seiner Arbeit mit einem generierten e-Learning-System unterstützen. Unter anderem auf der Basis dieses Benutzermodells erfolgte die Entwicklung von Evaluierungswerkzeugen, die vor allem den Autoren eines e-Learning-Systems umfassenden Aufschluss darüber geben können, wie das System von den Benutzern angenommen wird.

Kapitel 8

Strategien zur Informationssuche in generierten e-Learning-Systemen

In Abschnitt 7.4 wurden Werkzeuge vorgestellt, um Benutzer von e-Learning-Systemen beim Umgang mit komplexen, multimedial aufbereiteten Wissensinhalten zu unterstützen. Die Suche in umfangreichen, vernetzten und multimedialen Inhalten stellt hier eine weitere Möglichkeit dar, die Lernenden bei ihrer Arbeit mit einem e-Learning-System sinnvoll zu unterstützen.

Dieses Kapitel beschreibt die Retrievalkomponente, mit der der eLS-Baukausten jedes neu generierte e-Learning-System ausstattet. Hierbei handelt es sich um ein eigenständiges, komponentenbasiertes Modul, das eine „hybride Informationssuche“ realisiert, bei der unterschiedliche Retrievaltechniken zur Suche in unterschiedlichen Medien genutzt werden können. Die Benutzeroberfläche besteht dabei aus einer graphischen Suchkomponente, die ohne technische Kenntnisse von Lernenden eingesetzt werden kann und über leistungsfähige Suchmöglichkeiten verfügt.

In der Retrievalkomponente des eLS-Baukastens kommen unterschiedliche Techniken zur inhaltsbasierten Suche zum Einsatz. Im Einzelnen sind dies Text-Retrieval-Funktionen, Algorithmen zur bildbasierten Suche und attributbasierte Techniken des Datenbanksystems. Zusätzlich zur bildbasierten Informationssuche wird ein weiteres Verfahren zum Indizieren und Retrieval von Bildern vorgestellt, das den Autoren eine semi-automatische Namenszuordnung für Objekte in Bildern ermöglicht.

Alle Inhalte eines e-Learning-Systems sind so strukturiert, dass ein gezielter Zugriff auf diese möglich ist. Dazu verwaltet das System ein Verzeichnis aller Kategorien, über das der Benutzer einen nach Sachgebieten geordneten Zugang zu den Wissensinhalten erhält. Darüber hinaus wird zu jeder Kategorie ein Index gepflegt, der eine alphabetische Liste der wichtigen Begriffe in einer Kategorie bietet. Diese Suchfunktionalität im weiteren Sinne wurde als Teil des Wissensmodells beschrieben (vgl. Abschnitt 6.5). Schließlich bietet auch die alphabetisch organisierte Glossarliste einen systematischen Zugriff auf Wissen.

In den nachfolgenden Abschnitten werden zusätzliche Retrievalmöglichkeiten beschrieben, die diese Suchverfahren durch weitere, innovative Konzepte ergänzen. Dazu folgt zunächst ein konzeptioneller Überblick über die Retrievalkomponente des eLS-Baukastens, anschließend wird die text- und die bildbasierte Suche beschrieben und schließlich sollen Möglichkeiten aufgezeigt werden, die verschiedenen Verfahren zu einer umfassenden Suchkomponente zu verbinden.

8.1 Konzept

Den Lerninhalten eines e-Learning-Systems liegt das in Abschnitt 6.5 vorgestellte Wissensmodell zugrunde, d.h. die Datenbank speichert Instanzen der Klassen dieses Modells. Basierend auf diesen Instanzen

werden Inhalte indiziert und können von entsprechenden Algorithmen durchsucht werden. Aus Sicht des Information Retrievals lassen sich hierbei folgende Arten von Daten unterscheiden:

1. Klassenattribute wie Titel, Autor, Datum, etc.
2. Keywords, die von Autoren zu allen Datentypen angegeben werden können.
3. Metainformation zu Medienobjekten, z.B. Format und Auflösung eines Bildes oder der Typ einer Animation.
4. Texte, die beispielsweise in Instanzen der Klasse Content oder MMObject verwendet werden.
5. Bilddokumente eines MMObjects.

Die verschiedenen Datenarten erfordern unterschiedliche Retrievaltechniken (vgl. Kapitel 4): Bei Titel, Autorenname, Metainformationen und weiteren Datenfeldern aller Klassen handelt es sich um Attribute, die mit normalen, attributbasierten Mechanismen der verwendeten Datenbank durchsucht werden können. Die Suche in Textdokumenten erfordert die vorherige Indizierung und setzt zur Suche Techniken des Textretrievals ein. Die Suche nach Bilddokumenten erfordert die Extraktion von Merkmalen aus den Bildern und die Anwendung geeigneter Abstandsmaße. Abbildung 8.1 gibt einen Überblick über alle Arten von Daten, sowie deren unterschiedlichen Retrievaltechniken.

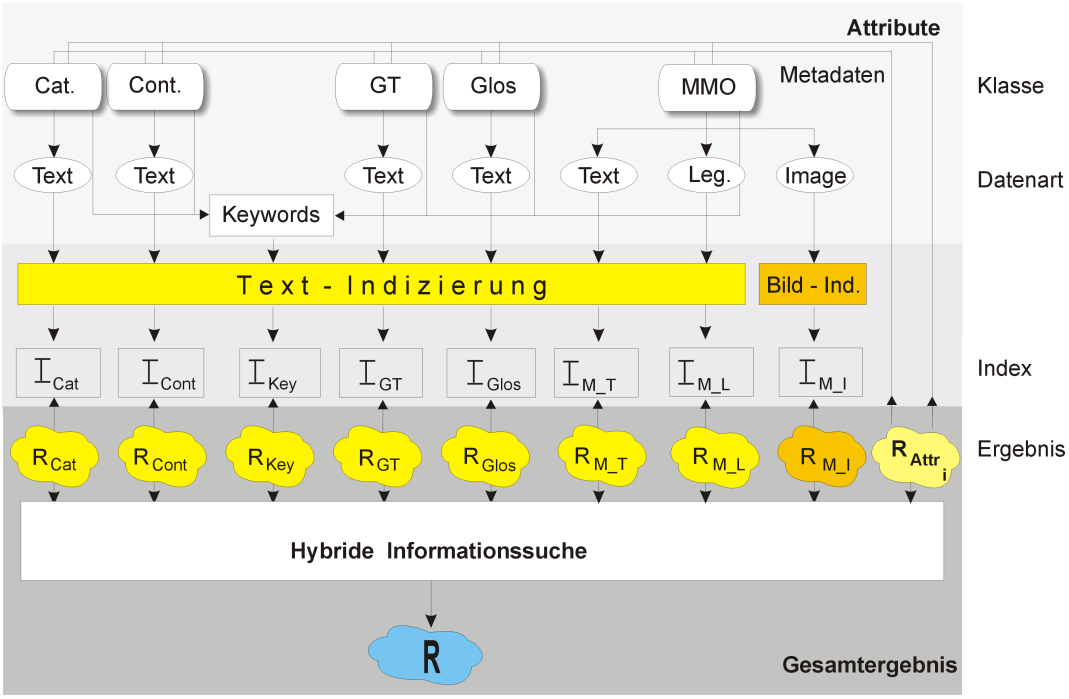

Abbildung 8.1: Informationssuche in e-Learning-Systemen

Im oberen Teil von Abbildung 8.1 sind die unterschiedlichen Arten von Daten dargestellt, die Gegenstand der Informationssuche sind. Abhängig von der Datenart wird ein jeweils spezifischer Index erstellt: Für Instanzen der Klassen **Category** (Cat), **Content** (Cont), **GuidedTour** (GT) und **Glossary** (Glos) sind dies ein Volltextindex für den enthaltenen Text, sowie ein Index der jeweils enthaltenen Keywords. Für Instanzen der Klasse **MMObject**, die Container für beliebige Medien darstellen, wird ein Volltextindex (für enthaltenen Text und Legenden) und eine Variante eines Index für extrahierte Bild-Merkmale (für Bilddokumente) angelegt. Die resultierenden Indices werden in Abbildung 8.1 durch die Indextabellen I_{Cat},

I_{Cont}, I_{Key}, I_{GT}, I_{Glos}, I_{MMO_T}, I_{MMO_L} und I_{MMO_I} bezeichnet. Im rechte oberen Teil der Abbildung ist außerdem die Suche nach Datenbankattributen angedeutet. Hierfür ist keine Indizierung erforderlich, da die vorhandenen Retrievalmethoden der Datenbankapplikation verwendet werden können.

Die im unteren Teil dargestellten „Wolken“ stellen die spezifischen Retrievaltechniken pro Index dar. Inhalte verschiedener Typen werden dabei separat durchsucht und führen zu den in Abbildung 8.1 mit R_{Cat}, R_{Cont}, R_{Key}, R_{GT} und R_{Glos}, R_{M_T}, R_{M_L}, R_{M_I} bezeichneten Ergebnismengen. R_{Attr_i} bezeichnet die Ergebnismenge einer attributbasierten Suche, wobei i für die verschiedenen Attribute wie Titel, Autorname, Datum, etc. steht.

Die einzelnen Retrievalmengen lassen sich je nach Fragestellung einzeln verwenden oder im Zuge der sog. „hybriden Informationssuche“ auf verschiedene Weise kombinieren und münden in einer Ergebnismenge, die in Abbildung 8.1 mit R bezeichnet ist. Diese Retrievalkomponente wird über die graphische Schnittstelle des Benutzer-Clients angesprochen und bietet dem Benutzer einen einheitlichen Zugang zu den darunterliegenden, unterschiedlichen Suchtechniken. Alle Retrievalalgorithmen werden serverseitig ausgeführt, so dass effizient auf die Datenbank zugegriffen werden kann. Erst nach Abschluß werden die Ergebnisse dem Benutzer-Client übermittelt.

Das Neue an diesem Ansatz ist die Verknüpfung der Stärken einzelner Recherchetechniken: Durch die Kombination von Anfragen konventioneller Datenbanken, textbasierter Suchmöglichkeiten, wie sie von WWW-Suchmaschinen bekannt sind, und Verfahren zur bildbasierten Informationssuche werden alle oben genannten Arten von Daten erschlossen. Bisher existiert innerhalb derzeitiger e-Learning-Systeme kein Ansatz, der eine derartige Retrievalkomponente umfaßt.

8.2 Textbasierte Informationssuche

Abbildung 8.2 zeigt die Benutzeroberfläche für die Volltextsuche des MURMEL-Systems.

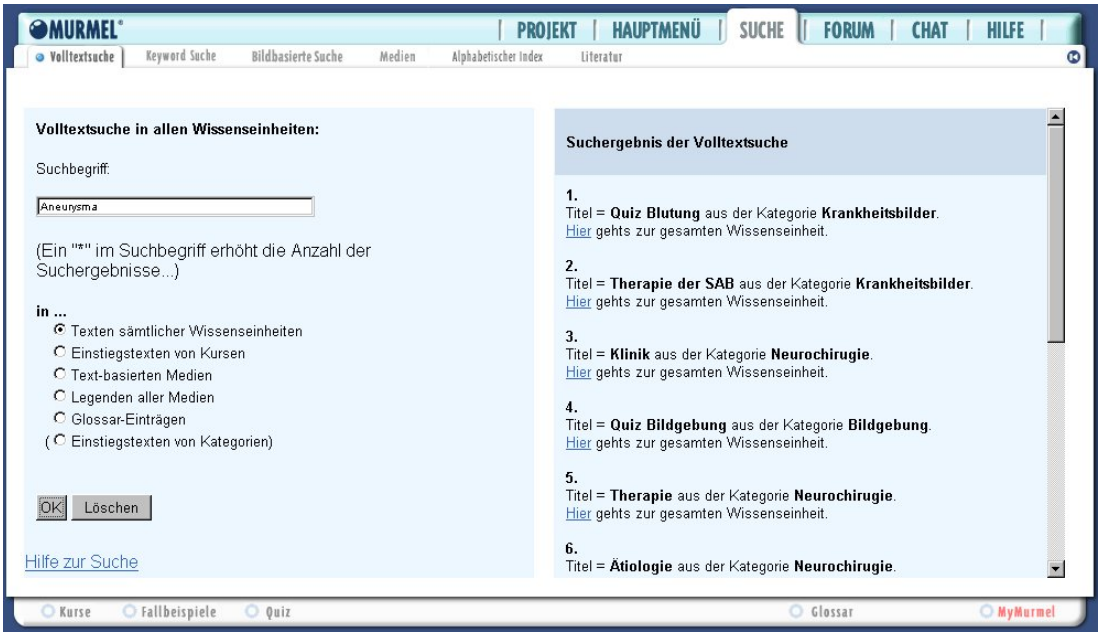

Abbildung 8.2: Volltextsuche in generierten e-Learning-Systemen

Der erste Reiter „Volltextsuche“ ist angewählt, daneben befinden sich Auswahlknöpfe für die Keyword-

suche, die Einstiegspunkte zur bildbasierten und hybriden Suche, die in den zwei folgenden Abschnitten vorgestellt werden, ein Button für die Anzeige aller Medien einer bestimmten Kategorie, ein weiterer für den alphabetischen Index verschiedener Kategorien und ein Button für die Literatursuche, bei der in den einzelnen Attributfeldern aller eingetragener Literaturhinweise gesucht werden kann. Im linken Frame befindet sich das Formular für die textbasierte Suche, die auf verschiedene Inhaltstypen und Attribute beschränkt werden kann. Der rechte Frame ist für ermittelte Suchergebnisse vorgesehen. Die im Beispiel aufgrund des vorgegebenen Suchbegriffs „Aneurysma“ ermittelten Ergebnisdokumente sind rechts angezeigt und können direkt angewählt werden.

Voraussetzung für die textbasierte Suche ist eine Indizierung aller im e-Learning-System enthaltenen Textdokumente. Diese vollzieht sich wie in Abbildung 8.3 dargestellt.

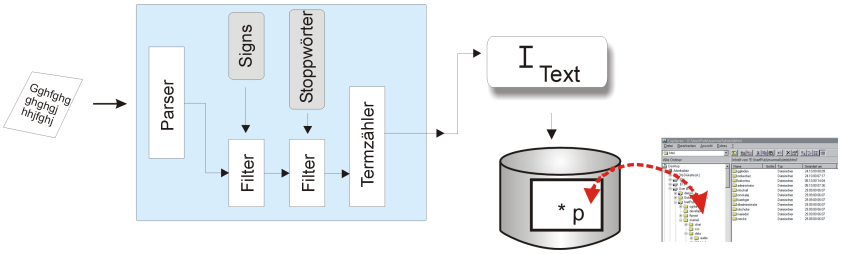

Abbildung 8.3: Textbasierte Indizierung in e-Learning-Systemen

Der Text wird wie in Abschnitt 4.3.1 beschrieben geparst, alle Stoppworte und in diesem Fall die HTML-Syntax aus den generierten Listen entfernt, und das endgültige Resultat in den zugehörigen Index übertragen. Dieser wird in Form einer kleinen Verzeichnisstruktur außerhalb der Datenbank im Dateisystem des Hostrechners abgelegt. Für jedes e-Learning-System existiert ein separater Index für die wichtigsten Datentypen des Wissensmodells wie z.B. *Content*, *GuidedTour*, *Keywords*, etc.

Zur Indizierung wird für jede erzeugte Indextabelle eine Instanz der Klasse *osmmTextIndex* und *osmmTextIndexHandle* angelegt, über die die Verwaltung der Tabelle gesteuert wird.

Die Datenbank stellt dabei zwar die persistente Ablage von Instanzen dieser Klassen zur Verfügung, steht darüber hinaus aber in keiner Beziehung zur implementierten Volltextindizierung. Die Klassen kümmern sich lediglich um die Indizierung und Suche von Textdokumenten. Das heißt, die Service Applikation übernimmt die Kontrolle, Speicherung und Aktualisierung von Indextabellen und stellt die Schnittstelle zwischen Anfragen und TextHandler dar. Beispielsweise muß beim Abbruch einer Transaktion, in deren Verlauf die Datenbank alle *datenbankabhängigen* Aktionen selbsttätig revidiert, eine vorgenommene Änderung am *Index* explizit wieder rückgängig gemacht werden. Dies erfordert das Eingreifen der Service Applikation und das Revidieren aller bereits vorgenommener Änderungen an einem Index. Abbildung 8.4 veranschaulicht dieses Verfahren.

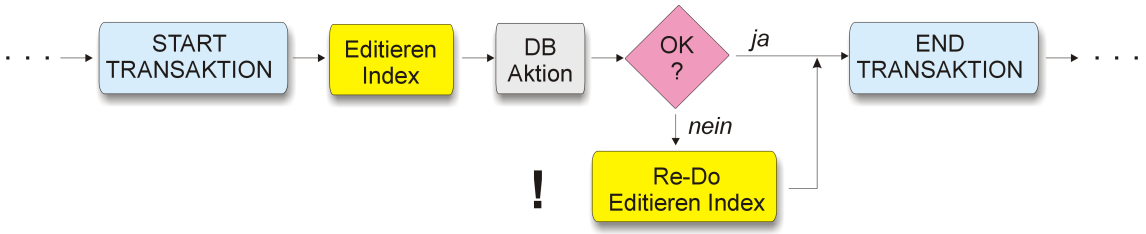

Abbildung 8.4: Kontrolle von Indexmanipulationen bei Transaktionen

Bei der Informations*suche* wird analog zu dem in Abschnitt 4.3.1 beschrieben Vorgehen der Suchbe-

griff mit den Einträgen einer Indextabelle verglichen und die entsprechenden Dokumente zurückgeliefert. Hierzu wird eine Instanz der Klasse *osmmTextSearch* verwendet. Sie umfaßt Methoden zur Festlegung des Schwellwerts, der angibt, ab wann ein Suchergebnis als „relevant“ eingestuft werden soll und zur Durchführung des Indexvergleichs, Platzhalter für die Anzahl zurückzuliefernder Suchergebnisse und eine Liste zur Aufnahme und Präsentation von Suchergebnissen. Dem Benutzer kann aufgrund der mitabgespeicherten Metadaten direkt mitgeteilt werden, welchem Kontext ein gefundenes Textdokument entstammt. Unter anderem wird beispielsweise bei einem zurückgelieferten Text-Medienobjekt auch dessen zugehöriger Content und der Titel der Kategorie mitgeliefert, der dem Benutzer bereits an dieser Stelle ein erstes Maß für die Relevanz des Dokuments bietet.

Die textbasierte Suche innerhalb der e-Learning-Systeme umfaßt eine Vielzahl Boole'scher und weiterer Operatoren, deren genaue Bedeutung und Anwendung über eine integrierte Hilfefunktion abgefragt werden kann. Tabelle 8.1 zeigt einige Beispiele und ihre Bedeutung im Hinblick auf die in Abschnitt 4.2 eingeführten Bewertungsmaße Recall und Precision.

Retrievalmethode	Recall	Precision
Bool'sche Operatoren AND, NOT		+
Bool'sche Operatoren OR	+	
Näheoperatoren		+
Nesting (Klammerung v. Suchausdrücken)		+
Trunkierung (Wildcards, Asterix, ...)	+	
Beschränkung auf best. Datentypen		+
Separate Suche in Metadaten	+	+

Tabelle 8.1: Retrievaloperatoren mit Recall & Precision

Anhand der grundlegenden Operatoren *AND*, *NOT* und *OR* soll die Tabelle erläutert werden: Die Operatoren *AND* und *NOT* sorgen für eine Erhöhung des Precision-Wertes, da durch diese Operatoren die Anzahl zurückgelieferter Dokumente verringert wird. *OR* erhöht dagegen den Recall, da die Anzahl der nicht-relevanten, aber zurückgelieferten Dokumente nicht mitberücksichtigt wird.

Zum „Relevance Ranking“ (vgl. Abschnitt 4.2) kann ein Benutzer aktiv vor Absenden einer Suchanfrage bestimmen, welche Dokumente bevorzugt angezeigt werden sollen. Zur Auswahl stehen dazu alle Attributfelder wie Titel, Datum, Autor, Gewichtung, etc. Darüber hinaus lassen sich aufgrund der vielfältigen und zahlreichen Informationen zu jedem Objekt eines e-Learning-Systems viele weitere Möglichkeiten zur Sucheinschränkung realisieren. So ließe sich beispielsweise auch der Kontext, in dem ein Dokument vorkommt, gesondert berücksichtigen, indem einem Dokument ein höherer Relevanzwert zugewiesen wird, wenn der Suchbegriff auch in den mit diesem in Zusammenhang stehenden, z.B. über Verweise verknüpften Objekten vorkommt. Ein solches „PageRank“-Verfahren ähnelt dem Vorgehen neuerer Suchmaschinen. Der Nachteil besteht allerdings in dem zeitlich größeren Aufwand zur Bearbeitung einer solchen Suchanfrage.

Die hier beschriebene Volltextindizierung ist auf gleiche Weise sowohl für das systematische als auch das problemorientierte Wissen möglich.

8.3 Bildbasierte Informationssuche

Neben der textbasierten Informationssuche stellen Verfahren zur inhaltsbasierten Suche in Bilddokumenten eine sinnvolle Ergänzung für e-Learning-Systemen mit multimedialen Inhalten dar. Für die durch den eLS-Baukasten erzeugten Systeme wurde deshalb ein Werkzeug implementiert, das die Indizierung und Suche beliebiger Bilddokumente über das Web ermöglicht. Eine Bildindizierung ist dabei einerseits automatisch und transparent über das Autorensystem möglich, andererseits können mit Hilfe eines in die

Benutzeroberfläche integrierten Verfahrens einzelne Bereiche eines Bildes zusätzlich indiziert werden.

Voraussetzung für die bildbasierte Informationssuche ist ein Datenbankeinstiegspunkt, über den der Bildindex angesprochen werden kann. Dazu wird während der Generierung eines neuen e-Learning-Systems nach dem in Abschnitt 6.6 beschriebenen Verfahren eine persistente Instanz der Klasse *osmmVirageIndex* angelegt. Abbildung 8.5 zeigt die wichtigsten Attribute dieser Klasse.

```
class osmmVirageIndex
{
protected:
   os_Collection<osmmImage*> &     _image_coll;
public:
   virtual osmmIndexHandle &       openIndex(int searchOnly = 0);
   os_Collection<osmmImage*> &     getIndexCollection() {return _image_coll;}
   static osmmFeatureVector*       analyzeImage(osmmRGBBuffer & rgb_buf);
   static int                      compareLessThan(...);
   static int                      compare(...);
}
```

Abbildung 8.5: Die Klasse osmmVirageIndex

Die Klasse *osmmVirageIndex* besitzt eine Methode zum Öffnen des Index, zum Zurückliefern der Liste aller bereits indizierten Bilder, die als Instanzen der Klasse *osmmImage* gespeichert vorliegen (vgl. Abschnitt 6.4), die Methode *analyzeImage()* zur Berechnung des Merkmalsvektors und zwei Methoden, mit deren Hilfe ein Vergleich von Merkmalsvektoren durchgeführt werden kann. Die Methode *compareLessThan()* verwendet zum Bildvergleich einen Schwellwert, der zum Abbruch des Vergleichs führt, sobald das berechnete Abstandsmaß diesen Wert überschreitet. Als Alternative zur Methode *compare()* erlaubt dieses Verfahren eine schnellere Bildanalyse, die in einer kleineren Ergebnismenge mit höheren Recall-Wert resultiert. Durch das Öffnen des Index wird eine transiente Instanz des Handlers *osmmVirageIndexHandle* angelegt, der, analog zum textbasierten Handler, für das Einfügen, Entfernen und Aktualisieren indizierter Bilder aus dem Index verantwortlich ist.

Die Bildindizierung bzw. deren Aktualisierung wird durchgeführt, wenn ein Autor über das Autorensystem zu einem Content ein neues Medienobjekt erzeugt bzw. editiert hat und per Knopfdruck die datenbankseitige Speicherung veranlaßt. Abbildung 8.6 beschreibt diesen Ablauf.

/* *Callback Funktion zur Bildindizierung* */	
Anstoßen der Speicherung eines Medienobjekts	
ftp-Transfer der zugehörigen Datei vom Client zum Server	
Anstoßen der Service Applikation mit Parameterübertragung	
...	
IF	Bilddatei nicht im ftp-Verzeichnis vorhanden
THEN	return *ERR_FILE_NOT_FOUND*
Anlegen einer neuen MMObject-Instanz	
IF	Medium EQ image
THEN	Lesen von osmmVirageIndex *virindex*
	Instantiieren des transienten osmmVirageIndexHandle
	Einlesen der Bilddaten durch den Handler
	Weiterleiten der Bilddaten an *virindex*
	Berechnen des Merkmalvektors durch *virindex*
	Ablegen des Merkmalvektors zur Bildinstanz
	Eintragen der Bildinstanz in die Image-Liste
	Schließen von *virindex*

Abbildung 8.6: Ablauf der Bildindizierung innerhalb der Autoren- Service Applikation

Nachdem die Mediendatei per ftp auf den Server übertragen wurde, wird die Service Applikation angestoßen, die zunächst die Existenz und Korrektheit eingelesener Parameter überprüft. Dann wird die zugehörige Content-Instanz aus der Datenbank geholt und eine neue MMObject-Instanz instanziiert. Anschließend wird im Fall eines Textmediums die Textindizierung angestoßen, bei einem Bild die bildbasierte Indizierung. In diesem Fall muß der Index explizit zum Schreiben geöffnet werden. Der anschließend erzeugte Handler überträgt das zu indizierende Medium an den Index und speichert nach der von diesem durchgeführten Analyse das Medium samt zugehörigem Merkmalsvektor in der Liste des Index, der schließlich wieder geschlossen wird. Zum Abschluß werden ggfs. vorhandene Keywords und die Legende volltextindiziert.

Das im Rahmen der e-Learning-Systeme integrierte Verfahren zur Bestimmung des Merkmalsvektors basiert auf einer globalen Berechnung der in Abbildung 8.7 dargestellten Merkmale.

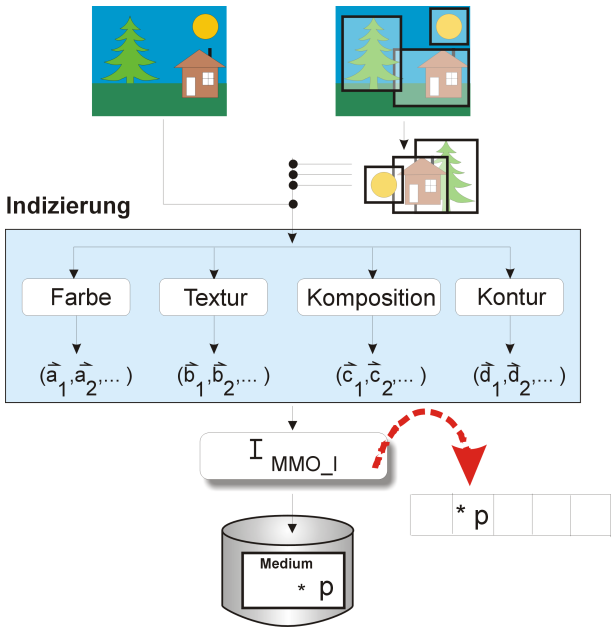

Abbildung 8.7: Bildbasiertes Indizieren und Retrieval in e-Learning-Systemen

Das zu indizierende Bild wird dabei separat nach den vier Merkmalen *Farbe, Textur, Komposition* und *Kontur* analysiert und verschiedene Repräsentationen dazu im Merkmalsvektor abgelegt. Beispielsweise werden beim Merkmal *Farbe* sowohl globale, als auch lokale Repräsentationen berechnet. Die Berechnung der ersten Ausprägung bezieht die dominierende Farbe und die Farbverteilung über das Gesamtbild mit in den Parameter ein. Hier werden beispielsweise die Fourierkoeffizienten der Farbverlaufsfrequenz zur Charakterisierung benutzt. Das lokale Farbmerkmal analysiert dagegen Farbe in Beziehung zu dessen räumlichen Lage im Bild. Im Hinblick auf das Merkmal *Textur* werden beispielsweise periodisch wiederkehrende Muster des Gesamtbildes und lokale, fein granulierte Bildbereiche analysiert, deren Eigenschaften wie Fläche, Länge oder Höhe quantifiziert werden. Beim Merkmal *Komposition* werden Eigenschaften umschlossener, großer Flächen im Hinblick auf Form, Fläche etc. ermittelt und zu anderen Objekten des Bildes in Beziehung gesetzt. Zur Bestimmung des Attributes *Kontur* wird das globale Kantenstärkebild berechnet.

Insgesamt beschreibt somit jedes Merkmal das Gesamtbild im Hinblick auf verschiedene Ausprägungen dieses bestimmten Merkmals (vgl. Abschnitt 4.3.2). Der zugehörige Merkmalvektor fasst schließlich alle Repräsentationen zusammen.

8.3.1 Indizierung mittels Java Applet

Im oberen Teil von Abbildung 8.7 sind zwei verschiedene Möglichkeiten dargestellt, Bildmaterial in e-Learning-Systemen abzulegen und durch die Bildindizierung zu erfassen. Neben der bereits beschriebenen Vorgehensweise über das Autorensystem besteht die zweite Möglichkeit in einem in die Benutzeroberfläche integrierten Java Applet. Dies zeigt Abbildung 8.8.

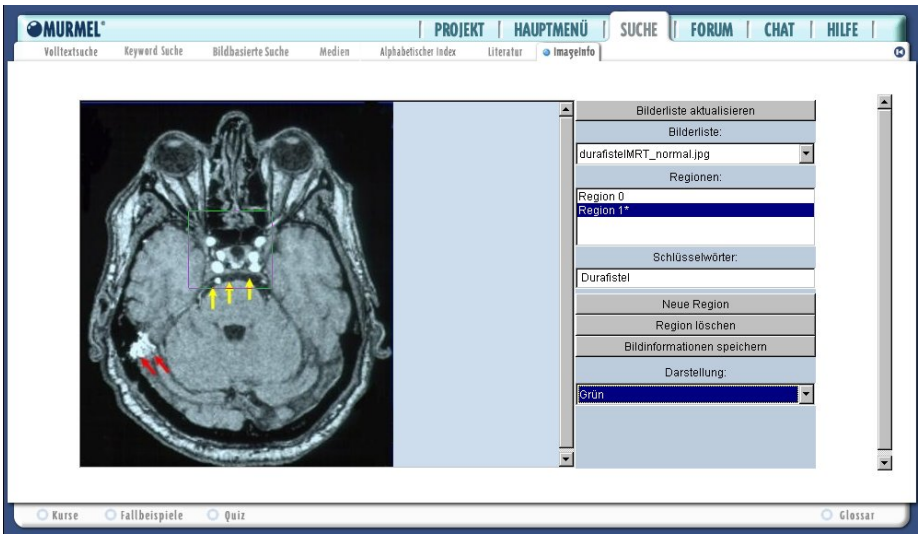

Abbildung 8.8: Regionenbasiertes Indizieren von Bildinhalten

Im linken Teil der Oberfläche ist das Bilddokument angezeigt, in dem ein Autor per Maus Regionen markieren kann. Im rechten Bereich hat er anschließend die Möglichkeit, zur eingezeichneten Region ein Keyword zu vergeben und Koordinaten samt Keyword an die Service Applikation zu senden. Dort wird die entsprechende Region indiziert und mit dem Keyword in der Datenbank abgelegt. Die extrahierten Subbilder werden wie jedes andere Bild mit in die bildbasierte Suche aufgenommen.

Alternativ können Autoren auch nur die selektierte Region an die Datenbank schicken. Zu dieser wird wiederum der zugehörige Merkmalsvektor bestimmt, anschließend die ähnlichsten Bilder des e-Learning-Systems ermittelt und deren Keywords zurückgeliefert. Der Autor kann eines dieser Keywords für seine selektierte Region auswählen oder letztendlich doch ein eigenes vergeben. Je mehr Bilddaten im System verfügbar sind, desto geeignetere Keyword-Vorschläge können dem Autor präsentiert werden. Der Vorteil dieses Vorgehens liegt in der Möglichkeit, langfristig Keywords auch automatisch vergeben zu können. Dies würde in einer Art Beschreibungskomponente für Bilddokumente resultieren, die den Autoren Routinearbeit bei der Kennzeichnung ihrer Bilder abnimmt.

Das implementierte Verfahren weist Ähnlichkeiten zu dem in Abschnitt 5.4 beschriebenen Bildsystem auf. Dort können allerdings nur Regionen in Bildern markiert und mit anderen Dokumenten des Systems verknüpft werden.

8.3.2 Informationssuche in Bildern

Abbildung 8.9 zeigt die Benutzeroberfläche für die Informationssuche in Bilddokumenten.
Ein Benutzer, der ein e-Learning-System nach bestimmtem Bildmaterial durchsuchen möchte, gibt dazu ein Suchbild vor und kann festlegen, mit welcher Gewichtung jedes der beschriebenen vier Merkmale in die Suche einfließen soll. Werte zwischen 0 (nicht relevant) und 1 (relevant) geben hier die Wichtigkeit des Merkmals an. Außerdem kann ein Schwellenwert gesetzt werden, der festlegt, wann eine bildbasierte

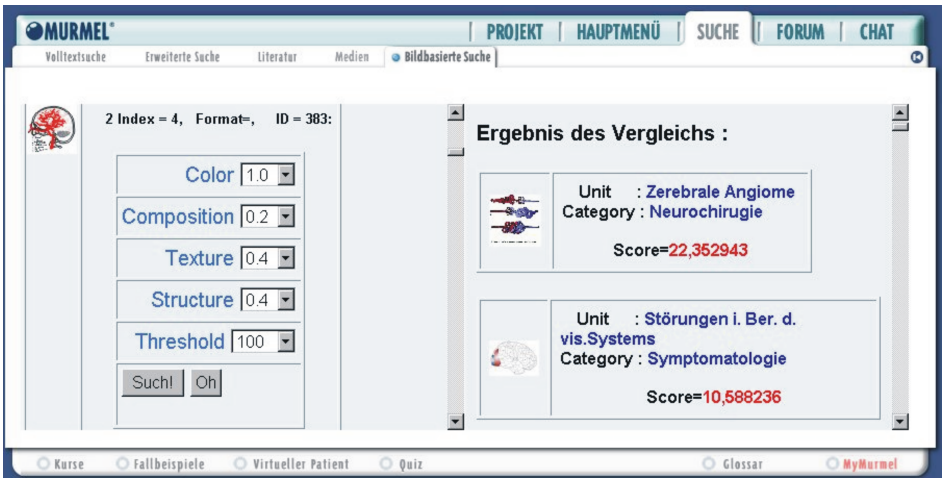

Abbildung 8.9: Ergebnis einer bildbasierten Suche innerhalb der Benutzeroberfläche

Suche abgebrochen werden soll. Auf Knopfdruck wird die Suchanfrage an den Server geschickt und nach dem serverseitigen Bildvergleich die Suchergebnisse im rechten Teil der Benutzeroberfläche angezeigt. Ein besonders niedriger Relevanzwert weist hierbei auf besonders hohe Ähnlichkeit hin. Abbildung 8.10 zeigt die auf Serverseite erforderlichen Arbeitsschritte zur Suchbearbeitung.

/* *Callback Funktion zur Bildbasierten Suche* */
...
Eingabebild *j* , Gewichte und Schwellwert einlesen
Merkmalsvektor zum Eingabebild *j* berechnen
Index öffnen und Bildliste einlesen
Vektor für Ergebnisbilder anlegen
für alle Bilder *i* der Liste
compareLessThan(i,j, Schwellwert, Gewichte)
Resultat in Ergebnisliste eintragen
Zurückliefern der Bildliste
...

Abbildung 8.10: Ablauf einer bildbasierten QbE-Anfrage innerhalb der Autoren Service Applikation

Das Verfahren zur Berechnung des Abstandsmaßes zwischen zwei Merkmalsvektoren arbeitet in zwei Stufen. Zunächst werden für alle Repräsentationen eines Merkmals die Abstände zu den entsprechenden Repräsentationen des Vergleichsbildes bestimmt. Der Mittelwert aller Abstände ergibt dann den endgültigen Wert für die Ähnlichkeit. Die zurückgelieferten Ergebnisse werden zunächst in Form von Thumbnails und einiger weiterer Metainformationen dargestellt. Erst bei Auswahl eines Icons wird das tatsächliche Medium an den Client gesendet und in der Benutzeroberfläche dargestellt.

Im Fall der bildbasierten Suche wird der Index nicht wie im textbasierten Fall in Form einer Verzeichnisstruktur im Dateisystem abgelegt, sondern besteht lediglich aus der von *osmmVirageIndex* verwalteten Bilderliste und den beschriebenen Methoden zum Zugriff auf diese. Der Merkmalsvektor selbst wird als Attribut direkt zum jeweiligen Bild abgelegt. Im Vergleich zum Textindex ist dies hier sinnvoll, da ein Merkmalsvektor nicht wie ein Wort einfach an eine feste Stelle innerhalb einer Indextabelle abgelegt werden kann. Für Merkmalsvektoren wäre dies nur im Fall einer zusätzlichen Klassifikation sinnvoll. Eine solche Klassifikation wird hier nicht vorgenommen, da die Image Retrievalkomponente für e-Learning-Systeme unterschiedlicher Anwendungsbereiche konzipiert ist und keine allgemeinen Verfahren existie-

ren, die für jeden Fall eine gute Klassifikation zur Verfügung stellen können (vgl. Unterabschnitt 4.3.2).

8.4 Hybride Informationssuche

Attribut-, text- und bildbasierte Suchmethoden können innerhalb eines e-Learning-Systems zu einer großen Retrieval-Komponente vereint werden. Eine solche Komponente muß genügend Ausdrucksmöglichkeiten bieten, um möglichst vielen Benutzerwünschen und -vorkenntnissen im Umgang mit Suchmöglichkeiten von Seiten der Lernenden gerecht zu werden. Dabei ist zu berücksichtigen, dass das hier vorgestellte Verfahren in e-Learning-Systemen unterschiedlichster Anwendungsbereiche zum Einsatz kommen wird und sich nicht an festen Anforderungen orientieren kann. Auf der anderen Seite muss die Komponente einfach zu bedienen sein, da es sich bei den Benutzern in der Regel um Personen ohne spezielle technische Kenntnisse handelt. Abbildung 8.11 gibt einen Überblick über die hybride Informationssuche.

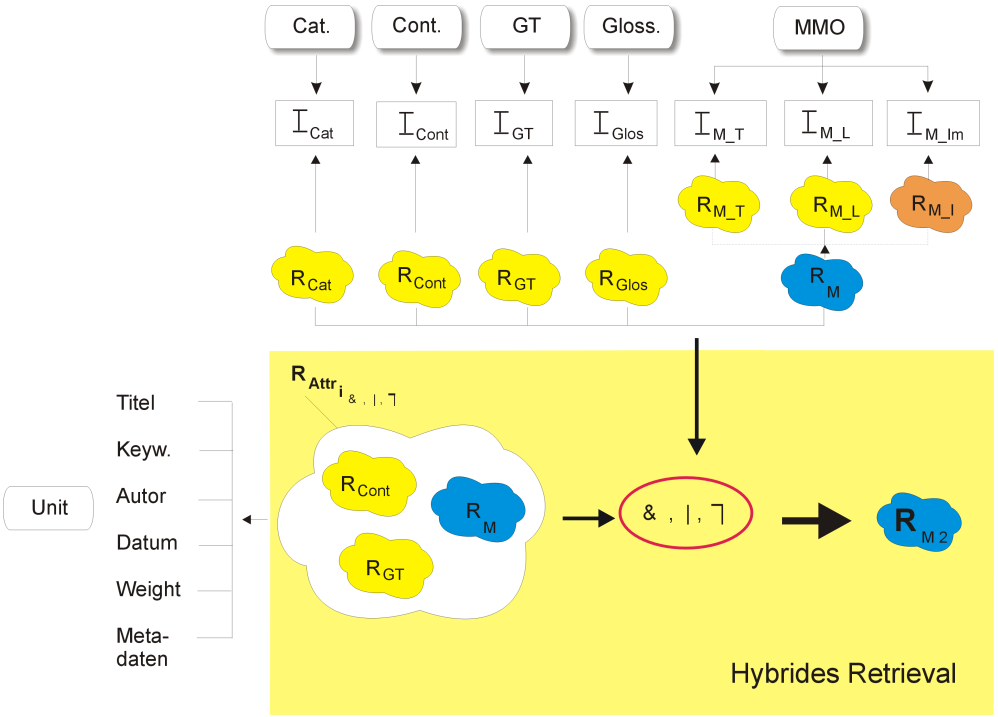

Abbildung 8.11: Hybride Informationssuche

Gegenstand der Suche sind die bereits bekannten fünf Inhaltstypen Category, Content, GuidedTour, Glossary und multimediale Objekte der Klasse MMObject. In diesem Datenbestand kann mit drei verschiedenen Methoden gesucht werden: Anhand von Textindizes, anhand von Merkmalen für Bilder und anhand von Attributen einzelner Klassen. Die Suche in Textindizes oder Bildmerkmalen liefert jeweils Instanzen einer bestimmten Klasse, die Suche über Attribute dagegen Instanzen unterschiedlicher Klassen.

Jede elementare Retrievalmethode liefert eine Ergebnismenge, die in Abbildung 8.11 als R_j mit *j = Cat, Cont, GT, Gl* und *MMO* bezeichnet wird, d.h. diese Ergebnismenge enthält nur Elemente vom Typ Kategorie oder Content oder .... Die attributbasierte Suche liefert dagegegen eine Ergebnismenge R_{Attr_i} mit *i = Titel, Keyw, Autor, ...* Das Ziel der hybriden Retrievalkomponente ist eine Suchmöglichkeit, die einerseits die jeweils notwendige Suchmethode für den Benutzer transparent macht und dadurch leicht bedienbar ist, und die andererseits vielfältige Kombinationsmöglichkeiten einzelner Ergebnismengen gestattet.

Die Retrievalkomponente ist, in zwei Frames unterteilt, in die graphische Oberfläche des Benutzerclients integriert (s. Abbildung 8.12).

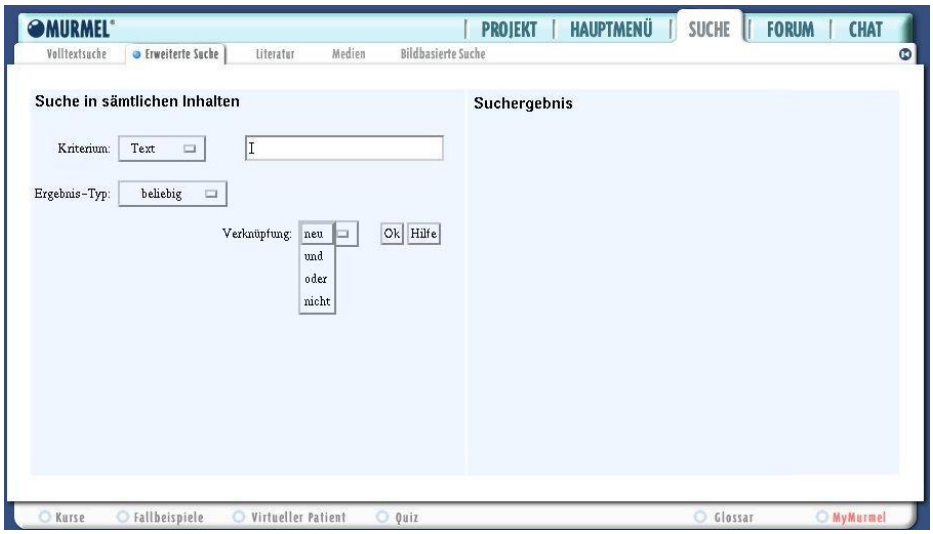

Abbildung 8.12: Suchmaske zur hybriden Suche

In der ersten Zeile bestimmt der Benutzer die Art des Suchkriteriums, d.h. er legt fest, welche Inhaltsarten durchsucht werden sollen. Möglich sind die Arten Text, Bild, Titel, Keyword, Autor, Datum, Gewichtung und Metadaten. In der zweiten Zeile kann der Benutzer den Typ des gewünschten Suchergebnisses bestimmen. Dies sind die zuvor erwähnten Typen Kategorie, Content, Glossary, GuidedTour, Medien oder Literatur. Die Einstellung 'beliebig' ist zu verwenden, wenn keine Einschränkung des Typs gewünscht ist. Im Formularfeld wird der Suchbegriff angegeben. Hier wird entweder Text oder beispielsweise ein Bildname eingetragen. Die Suchmaske wurde in erster Linie mit dem Ziel einfacher Bedienbarkeit gestaltet. Aus diesem Grund werden in den DropDown-Menüs „Text“ als Art des Suchkriteriums und „beliebig“ als Ergebnistyp voreingestellt. Dadurch erhält ein Benutzer in den meisten Fällen bereits sinnvolle Ergebnisse, auch wenn lediglich ein Suchbegriff vorgegeben wird. Die Suchmaske besitzt einen einheitlichen Aufbau, unabhängig davon, welche Suchmethoden im Hintergrund letztendlich eingesetzt werden.
Die folgenden Tabellen zeigen einige Beispiele für die Suchmöglichkeiten, die bereits durch diese einfache Suchmaske ermöglicht werden.

Art des Kriteriums	**Kriterium**	**Typ des Ergebnisses**	**Ergebnis**
Text	Gehirn	beliebig	*Informationen über das Gehirn*
Bild	sample.bmp	beliebig	*Bilder ähnlich zu sample.bmp*
Text	Gehirn	Medien	*Bilder, deren Legende das* *Wort* Gehirn *enthält*

Tabelle 8.2: Beispiele für einfache Suchanfragen

Durch die Verknüpfung mehrerer Suchergebnisse werden die Möglichkeiten für den Benutzer erheblich erweitert. Denkbar ist beispielsweise folgende, in Tabelle 8.3 dargestellte Liste von Anfragen zur Ermittlung aller Bilder, die dem Bild 'sample.bmp' ähneln, deren Legende das Wort 'Gehirn' enthält und dessen Autor nicht den Namen 'Huber' hat.

Schritt	Verknüpfung	Art des Kriteriums	Kriterium	Typ des Ergebnis
1.		Bild	sample.bmp	beliebig
2.	und	Text	Gehirn	Medien
3.	nicht	Author	Huber	beliebig

Tabelle 8.3: Beispiele für drei verknüpfte Suchanfragen

8.5 Zusammenfassung

Inhalt dieses Kapitels war die Beschreibung eines in dieser Form bisher nicht realisierten eigenständigen Systems zur multimedialen Informationssuche, das im Rahmen des eLS-Baukastens für unterschiedlichste Anwendungsbereiche einsetzbar ist.

Der vorgestellte Ansatz integriert bewährte attributbasierte Indizierungs- und Anfrage-Techniken mit textbasiertem Volltext- bzw. Schlüsselwortretrieval, innovativen Suchalgorithmen auf bildbasiertem Datenmaterial und neuartigen Suchmethoden, die über ein in die Oberfläche integriertes Java Applet möglich sind. Auf diese Weise steht eine mächtige Plattform zur Indizierung und zum vollständigen Retrieval von Text- und Bildmaterial zur Verfügung. Das implementierte Verfahren schließt dabei die Suche nach beliebigen weiteren multimedialen Komponenten ein, indem über bestimmte Kriterien wie z.B. Titel, Datum, Autor oder Metadaten gesucht wird.

Teil IV

Bewertung und Anwendungen

Kapitel 9

Realisierte e-Learning-Systeme

Zum Abschluss werden im Folgenden zwei Anwendungen vorgestellt, die mit Hilfe des eLS-Baukastens generiert wurden: Das medizinische e-Learning-System *NeuroAssistant* und das bereits mehrmals innerhalb dieser Arbeit erwähnte System *MURMEL*. Beide Systeme basieren auf den Komponenten des eLS-Baukastens, zeichnen sich aber durch einige Unterschiede aus, auf die an entsprechender Stelle innerhalb der folgenden Abschnitten hingewiesen wird. Der NeuroAssistant wurde mit dem ersten Release des eLS-Baukastens generiert, das System MURMEL mit der endgültigen, derzeit aktuellen Version.

9.1 Das e-Learning-System NeuroAssistant

Der NeuroAssistant beinhaltet und präsentiert medizinisches Wissen aus dem Fachbereich der Neurowissenschaften. Die Zielgruppen, die auf Wissen unterschiedlichen Niveaus zugreifen können, sind Ärzte, Medizinstudenten und Patienten. Abbildung 9.1 zeigt den inhaltlichen Aufbau des NeuroAssistant.

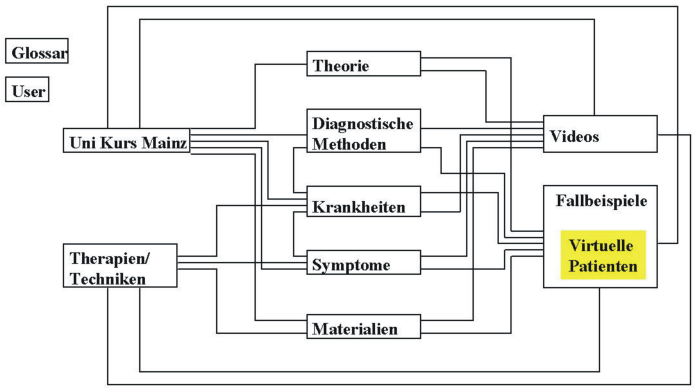

Abbildung 9.1: Inhaltlicher Aufbau des NeuroAssistant

Abbildung 9.1 zeigt eine Übersicht der wichtigsten Typen von Wissensinhalten. Hierbei handelt es sich um systematisches (*Theorie, Diagnostische Methoden, ...*) und problemorientiertes Wissen (*Fallbeispiele*, *Kurse* und *Virtuelle Patienten*). Hinter dem Begriff Virtuelle Patienten verbergen sich die in Abschnitt 6.5.3 eingeführten Simulationen. In diesem Fall handelt es sich um interaktiv gestaltete komplexe Simulationen von Patientenfällen. Stellvertretend für die Gesamtinhalte des NeuroAssistant wird im folgenden Unterabschnitt ein Beispiel eines solchen Virtuellen Patienten gegeben.

9.1.1 Virtuelle Patienten

Die Virtuellen Patienten sind insbesondere auf die Ausbildung von Medizinstudenten ausgerichtet. Sie beschreiben Handlungsabläufe, die im Umgang mit Patienten typisch sind. Abbildung 9.2 zeigt die Eingangsseite zur Auswahl eines Patienten.

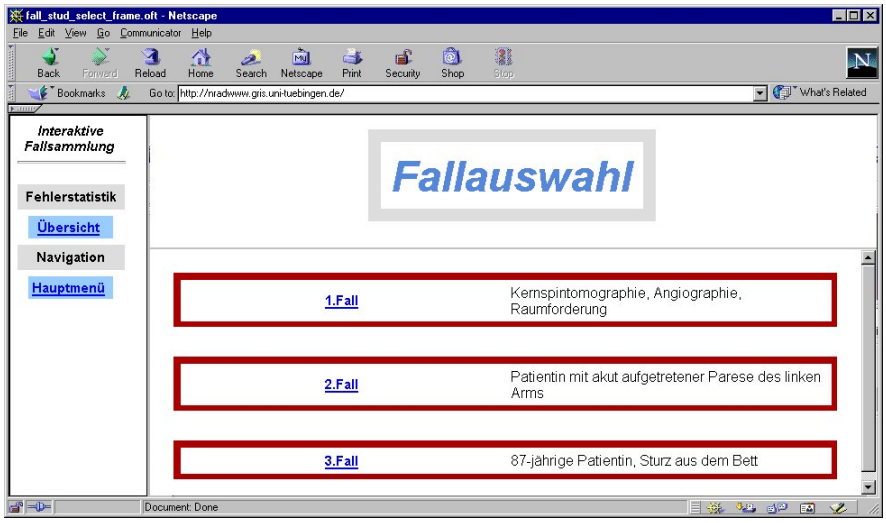

Abbildung 9.2: Einstiegsseite der Virtuellen Patienten im NeuroAssistant

Nach Auswahl eines Falles erscheint in der Regel zunächst die Anamnese eines Patienten. Abbildung 9.3 zeigt einen typischen Bildschirmaufbau für diesen Schritt.

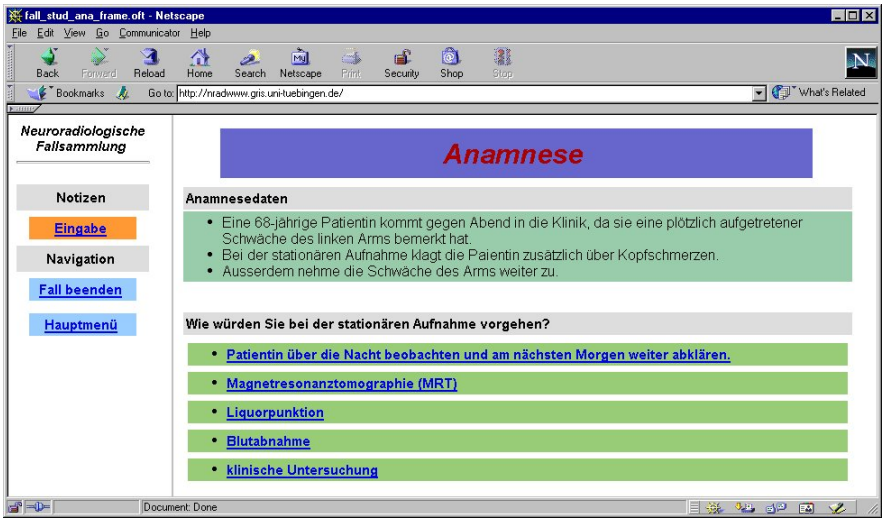

Abbildung 9.3: Bearbeitung eines Virtuellen Patienten im NeuroAssistant (1)

Hier wird die Krankengeschichte des Patienten vorgestellt und der Lernende aufgefordert, erste Untersuchungsschritte einzuleiten. Bei einer falschen Auswahl erscheint ein entsprechender Expertenkommentar, der den Fehler erläutert. Im Fall der richtigen Entscheidung werden die Ergebnisse der angeordneten Untersuchung präsentiert. Abbildung 9.4 zeigt als Beispiel die Präsentation von Untersuchungsergebnissen einer Computertomographie.

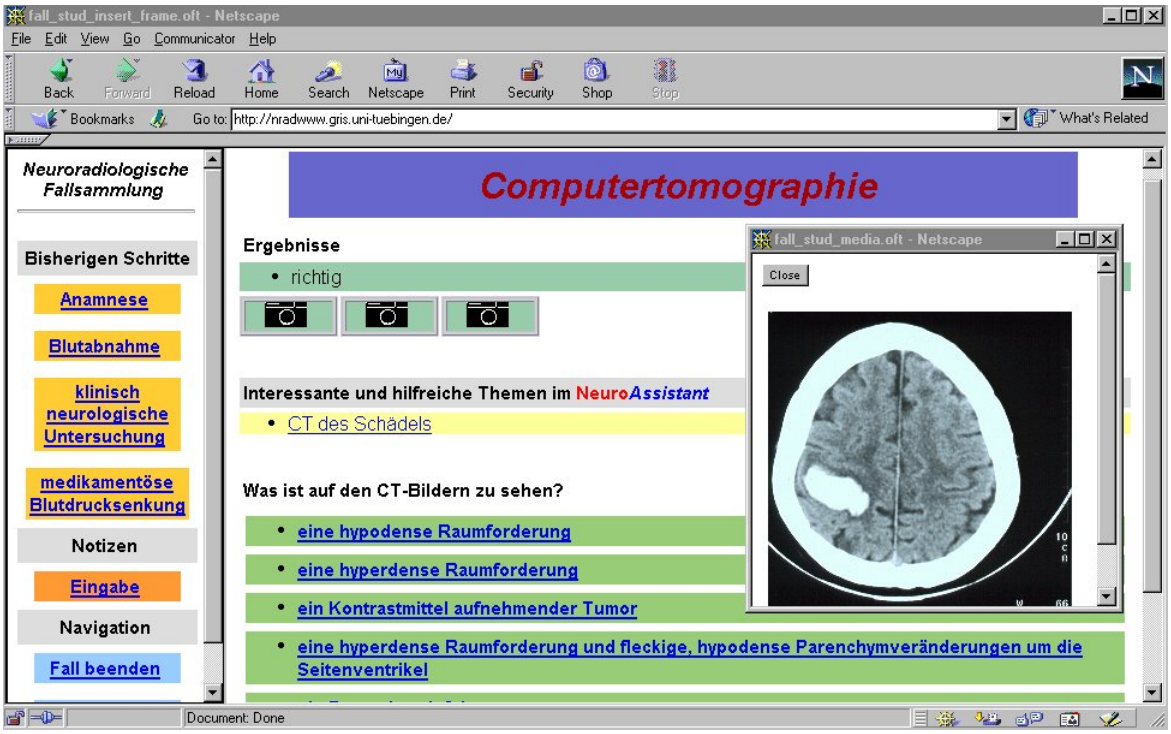

Abbildung 9.4: Bearbeitung eines Virtuellen Patienten im NeuroAssistant (2)

Die linke Navigationsleiste beinhaltet alle bereits durchgeführten Schritte des Handlungsablaufs, die Notizblockfunktion zur Ablage von Notizen und den Schaltknopf für das Beenden einer Fallbearbeitung. Im rechten Frame sind die Ergebnisse der durchgeführten Untersuchung, weiterführende Informationen und die im Anschluss zu beantwortenden Fragen dargestellt. Eines der durch Kamerasymbole angedeuteten Bildergebnisse ist zur Ansicht geöffnet. Durch Anwahl eines Antwort-Ankers wird der nächste Schritt des Virtuellen Patienten angestoßen. Ein Benutzer navigiert somit halbgeführt durch den gesamten Wissensbaum, der einen Virtuellen Patienten beschreibt, und versucht aus den jeweils gebotenen Vorgehensweisen dem „richtigen“ Weg bis zur korrekten Diagnose zu folgen. Im speziellen Fall des NeuroAssistant erfolgt die Erstellung von Virtuellen Patienten durch ein HTML-basiertes Autorensystem. Abbildung 9.5 zeigt einen Ausschnitt aus einer Fallbearbeitung durch einen Autor. Das Autorensystem enthält dazu im oberen Fensterbereich stets den gesamten Wissensbaum, der zu einem Virtuellen Patienten gehört. Dieser kann sukzessive vom Autor erzeugt und jederzeit verändert werden. Unterschiedliche Farben symbolisieren, ob es sich bei einem Schritt momentan noch um eine „Sackgasse“ (gelb), einen „internen Schritt“ mit nachfolgenden Schritten (grün) oder um Start oder Ziel (blau) handelt. Ein Autor kann einen Schritt anwählen bzw. neu erzeugen und ihn im unteren Fensterbereich editieren. Im Fall von Abbildung 9.5 wird momentan der Schritt „Magnetresonanztomographie“ bearbeitet und ein einführender Ergebnistext eingegeben. Über die Schaltfläche „Zur Eingabe der Bilder“ können dem Schritt Ergebnisbilder hinzugefügt werden.

Der NeuroAssistant besitzt im Gegensatz zum standardisierten Aussehen der durch den eLS-Baukasten

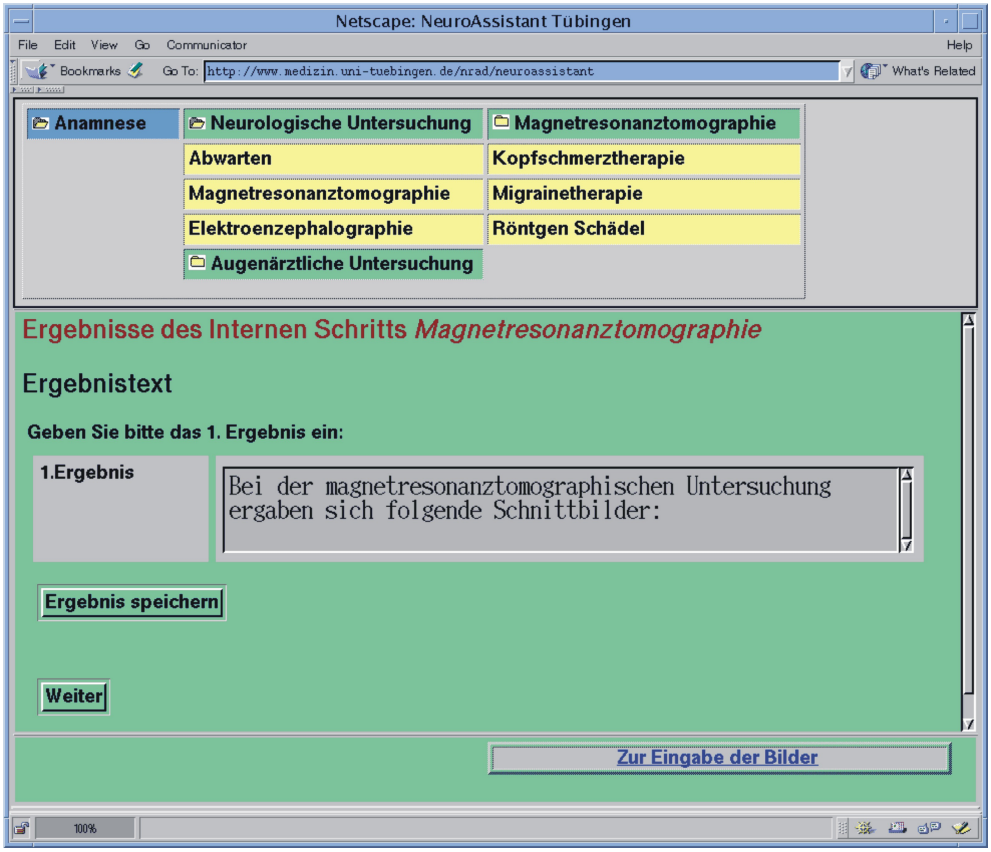

Abbildung 9.5: Generierung eines Virtuellen Patienten durch einen Autor

generierten e-Learning-Systeme eine anders aufgebaute Benutzeroberfläche. Hieran zeigt sich erneut ein Vorteil der komponentenbasierten Struktur aller durch den eLS-Baukasten erzeugten e-Learning-Systeme: Nahezu unabhängig von der serverseitigen Implementierung der Service Applikation kann das Layout und die Funktionsweise der Clientseite an verschiedene Bedürfnisse angepaßt und jederzeit verändert werden. Die zugrunde liegenden Techniken bleiben davon unberührt.

9.1.2 Statistische Auswertungen

In diesem Abschnitt werden einige statistische Auswertungen des NeuroAssistant im Hinblick auf das Nutzerverhalten innerhalb des beispielhaft betrachteten 1-jährigen Erfassungszeitraumes von Februar 1998 bis Januar 1999 demonstriert. Diese ergeben sich durch Auswertung der sog. *Webserver Logdatei*, die in mehr oder weniger umfangreichem Maße Daten über die Besucher eines Webservers, in diesem Fall den Netscape Enterprise Server, erfaßt. Für alle im Folgenden gelieferten Auswertungen sind die verwendeten Begriffserklärungen in Abschnitt 9.3 zusammengefaßt.

Abbildung 9.6 und Tabelle 9.1 protokollieren den Zugriff auf den NeuroAssistant über den gesamten Beobachtungszeitraum (Anzahl Sitzungen pro Monat).

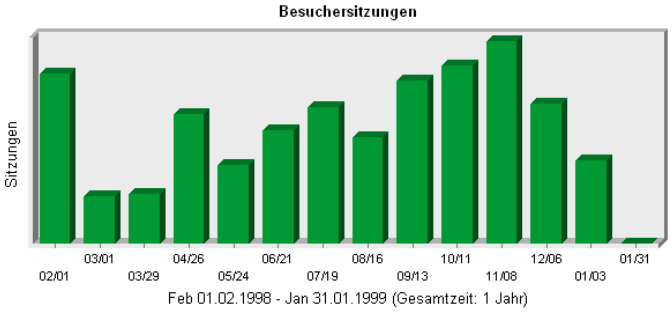

Abbildung 9.6: Allgemeine Zugriffstatistik des NeuroAssistent

Anwender	Anzahl der eindeutigen Besucher	870
	Anzahl der Besucher, die nur einmal kamen	635
	Anzahl der Besucher, die mehrmals kamen	235
Hits	Anzahl der erfolgreichen Hits auf die gesamte Site	192,028
	Mittlere Anzahl an Hits pro Tag	526
	Anzahl der Hits auf Homepage	290
Seitenimpressionen	Anzahl der Seitenimpressionen	152,750
	Mittlere Anzahl an Seitenimpressionen pro Tag	418
Sitzungen	Anzahl der Anwendersitzungen	3,197
	Mittlere Anzahl Anwendersitzungen pro Tag	8
	Mittlere Länge einer Anwendersitzung	00:15:38
	Internationale Anwendersitzungen	34.53%
	Anwendersitzungen mit unbekanntem Herkunftsland	54.67%
	Anwendersitzungen aus USA	10.79%

Tabelle 9.1: Allgemeine Zugriffsstatistik des NeuroAssistant

Demnach arbeitete im Februar, April, Juli und im Zeitraum von September bis November der weitaus größte Anteil an Nutzern mit dem NeuroAssistant. Diese Zeiträume stimmen einerseits grob mit den Semesterzeiträumen überein, was darauf schließen läßt, dass der NeuroAssistant besonders von Medizinstudenten beansprucht wurde. Auf der anderen Seite wird der NeuroAssistant auch von bereits tätigen Medizinern genutzt, deren Arbeitszeiten in der Regel nicht an Semesterzeiträume gebunden sind. Die in Tabelle 9.1 angegebenen Zahlen erfassen nur Clients, deren Ursprung eindeutig ermittelt werden konnten. Von diesen waren ca. 2/3 einmalige Besucher, 1/3 benutzte den NeuroAssistant mehrmals, dabei insgesamt im Schnitt 16 Minuten pro Anwender.

Abbildung 9.7 zeigt die aktivsten und inaktivsten Stunden pro Tag während der Berichtszeitspanne. In der zugehörigen Tabelle 9.2 sind aus Gründen der Übersichtlichkeit die Angaben nur für den aktivsten Zeitraum von 7 h bis 20h dargestellt und diese Aktivitäten zusätzlich aufgeteilt, um die mittlere Aktivität während der einzelnen Stunden aufzulisten.

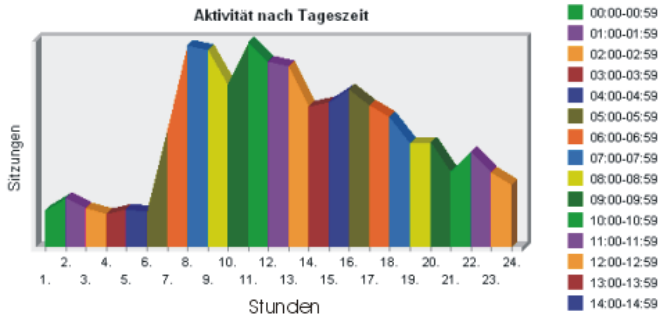

Abbildung 9.7: Aktivität nach Stunden beim NeuroAssistant

Stunde	Hits	% von Gesamt%	Sitzungen
07:00 - 07:59	3,858	2.00%	237
08:00 - 08:59	16,071	8.36%	231
09:00 - 09:59	23,999	12.49%	191
10:00 - 10:59	29,305	15.26%	240
11:00 - 11:59	28,523	14.85%	218
12:00 - 12:59	15,754	8.20%	212
13:00 - 13:59	9,068	4.72%	165
14:00 - 14:59	9,677	5.03%	171
15:00 - 15:59	9,297	4.84%	186
16:00 - 16:59	9,745	5.07%	167
17:00 - 17:59	6,175	3.21%	153
18:00 - 18:59	5,796	3.01%	122
19:00 - 19:59	6,434	3.35%	124

Tabelle 9.2: Aktivität nach Stunden in NeuroAssistant

Erwartungsgemäß arbeiteten die meisten Nutzer in den Morgen- und frühen Nachmittagsstunden mit dem NeuroAssistant.

Abbildung 9.8 gruppiert schließlich Besuchersitzungen nach ihrer Besuchsdauer. Für jede an der x-Achse dargestellte Gruppe wird die Gesamtanzahl der Besucher und die Gesamtanzahl der angezeigten Seiten berechnet. In der zugehörigen Tabelle 9.3 werden die Angaben der ersten zehn Besuchsminuten protokolliert.

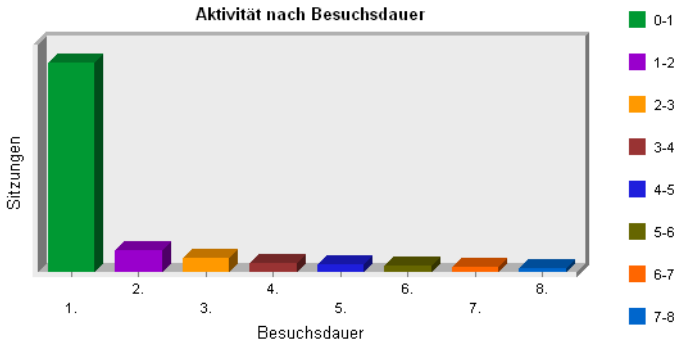

Abbildung 9.8: Besuchsdauer von Anwendern im NeuroAssistent

Besuchsdauer (Min.)	Besuche	Seitenansichten	% aller Besuche	% aller Ansichten
0-1	1,845	7,058	57.71%	4.62%
1-2	200	4,145	6.25%	2.71%
2-3	127	3,463	3.97%	2.26%
3-4	84	2,628	2.62%	1.72%
4-5	79	2,613	2.47%	1.71%
5-6	61	1,986	1.90%	1.30%
6-7	53	2,375	1.65%	1.55%
7-8	42	2,183	1.31%	1.42%
8-9	29	903	0.90%	0.59%
9-10	40	1,301	1.25%	0.85%

Tabelle 9.3: Besuchsdauer von Anwendern im NeuroAssistent

Besonders Abbildung 9.8 macht deutlich, dass die Protokollierungsfähigkeit des Webservers relativ schnell an ihre Grenzen stößt. Mit Ansprechen des serverseitigen CGI-Skripts endet die Protokollierung jeglicher Nutzeraktivität. Im konkreten Fall erreicht ein Nutzer den NeuroAssistant zwar über eine statische Einstiegs-HTML-Seite, ab dieser Seite erfolgt die Kommunikation mit dem Server jedoch stets über das Datenbank-CGI-Skript (vgl. Abschnitt 6.4.1), d.h. an dieser Stelle endet die Protokollierung. So ist auch die weitaus häufigste Besuchsdauer von 0-1 min zu erklären.

9.1.3 Zusammenfassung

Der NeuroAssistant als erste praktische, prototypische Anwendung des eLS-Baukastens weist bereits wesentliche Eigenschaften auf, die im Theorieteil dieser Dissertation als notwendige Voraussetzungen für den erfolgreichen Einsatz eines e-Learning-Systems erarbeitet worden waren. Die rein HTML-basierte Lösung ist performant und stabil und für den größten Teil zugreifender Benutzer akzeptabel. Die Erstellung von Inhalten für das System über das rein HTML-basierte Autorensystem setzte bei den Autoren jedoch ein bestimmten Maß an Grundlagenwissen bezüglich der Übertragung von Dateien auf den Server

und generell des Betriebssystems voraus. Die Organisation des Arbeitsablaufes beispielsweise zur Erstellung eines Virtuellen Patienten allein durch HTML und JavaScript erweist sich gerade im Hinblick auf die Darstellung von Baumstrukturen oder generellen Verknüpfungen zwischen Wissenseinheiten als nicht immer mächtig genug.

Bezüglich der im vorangegangenen Unterabschnitt präsentierten Statistiken und Nutzerzugriffe wird deutlich, dass innerhalb des Datenmodells eines jeden e-Learning-Systems Strukturen für die Erfassung und Ablage benutzerabhängiger Daten unabdingbar sind. Dies hängt unmittelbar mit dem „Deep Web“ zusammen, d.h. der Webserver ist, genauso wie die in Abschnitt 4.4 erwähnten Web-Robots, nicht in der Lage Dateien zu erfassen, die über ein CGI-Skript angesprochen werden. Sobald also Benutzer das eigentliche e-Learning-System über das CGI-Skript „betreten“, sind sie für den Webserver nicht mehr erfassbar. Diese Problematik konnte durch die Erweiterung des Datenmodells innerhalb des eLS-Baukastens um eine entsprechende Benutzermodellierung behoben werden. Das im folgenden Abschnitt beschriebene e-Learning-System setzt dieses Benutzermodell bereits ein.

9.2 Das e-Learning-System MURMEL

Das System MURMEL, das im Rahmen der Gemeinschaftinitiative „Multimedia-gestützte Studiengänge an Hochschulen“ vom Land Baden-Württemberg und der Dt. Telekom AG entstanden ist, basiert auf der neuesten Version des eLS-Baukastens. Der folgende Unterabschnitt präsentiert Beispiele der in Abschnitt 6.5 beschriebenen Lernformen. Im Anschluss daran folgen einige statistische Betrachtungen hinsichtlich der Nutzer- und Autorenaktivitäten und es werden erste Resultate durchgeführter Evaluierungsmaßnahmen vorgestellt.

9.2.1 Beispielseiten

Die folgenden Abbildungen beschreiben einige spezielle Interaktionsformen und Lerninhalte des MURMEL-Systems. Beispielsweise zeigt Abbildung 9.9 einen Eintrag des systematischen Wissens, bei dem unter anderem ein Java Applet als Medienobjekt eingebunden wurde. Dieses kann von den Benutzern des e-Learning-Systems interaktiv und als einfache Form der Wissensüberprüfung verwendet werden.

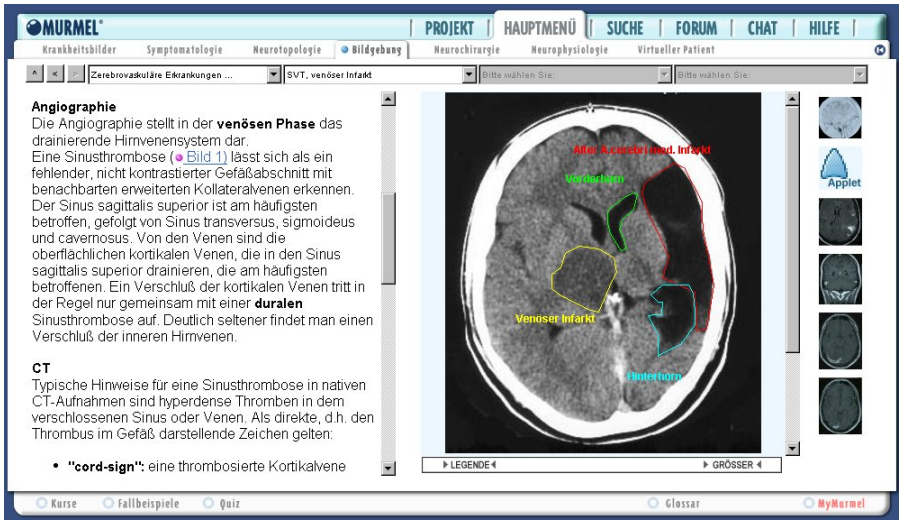

Abbildung 9.9: MURMEL: Beispiel eines Wissensinhalts mit Java Applet

Die mächtigeren Visualisierungsmöglichkeiten innerhalb von e-Learning-Systemen im Vergleich zu Printmedien werden auch durch den in Abbildung 9.10 gezeigten Wissensinhalt deutlich: Als Medienobjekt wurde hier ein Video eingebunden, das einen „virtuellen Flug durch das Ventrikelsystem“ erlaubt [Bar99].

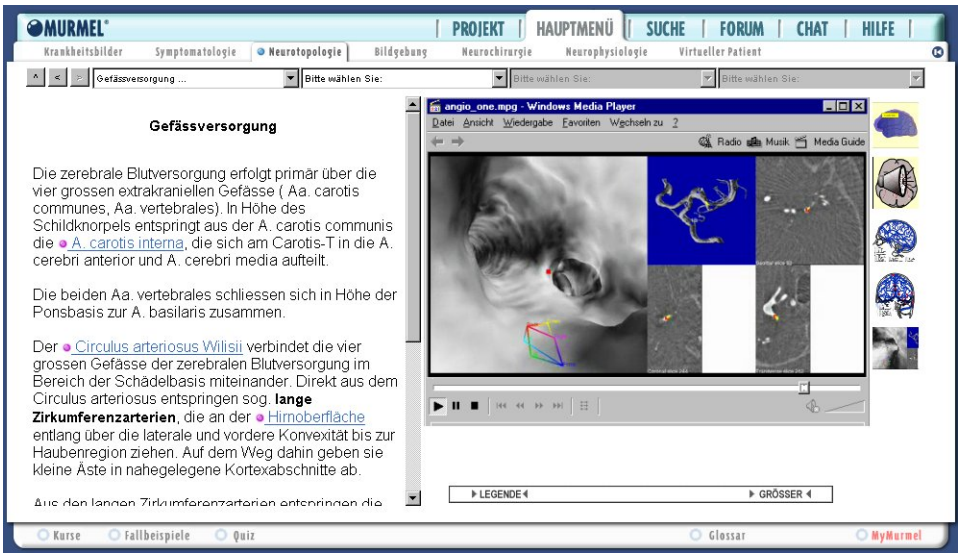

Abbildung 9.10: MURMEL: Beispiel eines Wissensinhalts mit virtuellem Ventrikelflug

Einen ähnlich großen Informationsgehalt haben Wissensinhalte, die beispielsweise Videos von realen Patienteneingriffen oder -untersuchungen beinhalten. Gerade für Medizinstudenten, die noch nicht in der praktischen Routine tätig sind, sind derartige Inhalte von größtem Interesse. Abbildung 9.11 zeigt als Beispiel einen Ausschnitt eines neurochirurgischen Videos [Duf00].

Abbildung 9.11: MURMEL: Beispiel eines Wissensinhalts mit neurochirurgischem Video

Zur Bearbeitung eines Fallbeispiels navigiert der Lernende ähnlich wie beim Durchlaufen einer Guided Tour sequenziell durch eine Serie von Wissensinhalten. Abbildung 9.12 zeigt ein solches Beispiel.

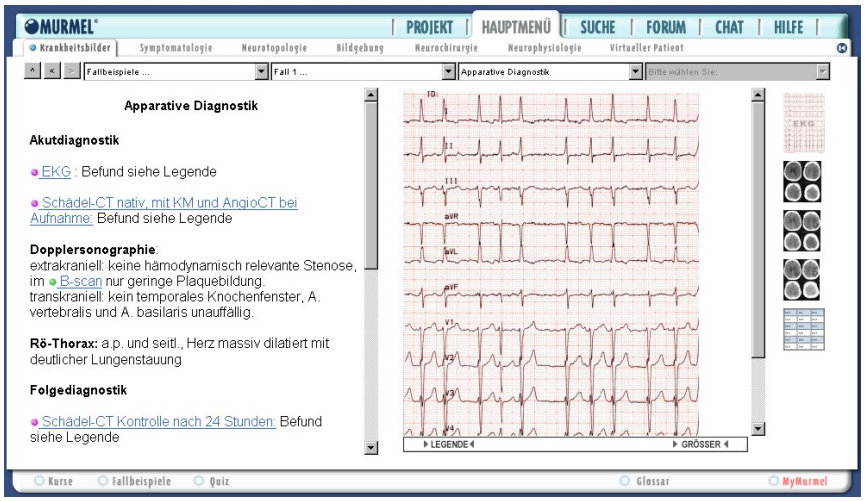

Abbildung 9.12: MURMEL: Untersuchungsergebnisse eines Fallbeispiels

Eine spezielle Variante eines Fallbeispiels ist beispielsweise die Angabe der Musterlösung, die im Zusammenhang mit der Darstellung eines typischen Handlungsablaufs in das e-Learning-System eingefügt wurde. Im Fall des MURMEL-Systems sind diese typischen Handlungsabläufe als „Virtuelle Patienten“ realisiert. Abbildung 9.13 zeigt den üblicherweise ersten Inhalt eines solchen Handlungsablaufs, die Anamnese.

Abbildung 9.13: Simulationen am Beispiel eines Virtuellen Patienten

Im Beispiel sind diesem Schritt keine Medien zugeordnet, sondern nur ein Expertenkommentar, der ganz rechts (die Person mit dem erhobenen Zeigefinger) angedeutet ist. Am Ende der Bearbeitung folgt eine Zusammenstellung der einzelnen Expertenkommentare, eine Auswertung der richtigen und falschen Anworten, die der Lernende im Verlaufe der Bearbeitung gegeben hat, sowie die Präsentation einer „Musterlösung“.

9.2.2 Fragebogenaktionen

Nach Generierung des MURMEL-Systems durch den eLS-Baukasten, der Gestaltung der Benutzeroberfläche und der Erstellung einer ausreichenden Zahl an Inhalten wurden erste Evaluierungsmaßnahmen durchgeführt. In insgesamt zwei Runden wurden Studierende per Fragebogen sowohl zu allgemeinen Einstellungen, als auch speziell dem MURMEL-System befragt.

In der ersten Runde wurde an insgesamt 103 Studierende, davon 56 (54%) weibliche und 47 (46%) männliche, des ersten und zweiten klinischen Studienabschnitts ein Fragebogen verteilt, der u.a. die generelle Einstellung zu computergestütztem Lernen ermitteln sollte. Die wesentlichen Ergebnisse sollen im Folgenden diskutiert werden.

Auffallend ist, dass bei eigentlich allen Variablen keine signifikanten Unterschiede zwischen weiblichen und männlichen Studierenden festgestellt werden konnten. Die oft gehörte These, dass Frauen weniger mit dem Internet zu tun haben und Computern skeptischer gegenüberstehen als Männer, kann durch die Fragebogenergebnisse nicht belegt werden.

Immerhin 82% der Studierenden besitzen einen eigenen Computer. Zugang zum Internet innerhalb der Universität hat hingegen nur die Hälfte aller Studierenden, von zuhause können sogar nur 40% ins Internet, 30% der Studierenden haben momentan weder innerhalb der Uni noch von zuhause aus einen Netzzugang.

Die zunehmende Bedeutung von Computern wird von einer großen Mehrheit der Studierenden als positiv empfunden (68%), lediglich 8% bewerten diese als negativ, das restliche Viertel steht dem gleichgültig gegenüber.

Über die Hälfte aller Befragten nutzt das Internet sehr selten oder nie, 38% der Studierenden zumindestens ab und zu (mehrmals in der Woche bzw. im Monat) und lediglich 6% gaben an, fast jeden Tag im Netz zu sein (vgl. Abbildung 9.14).

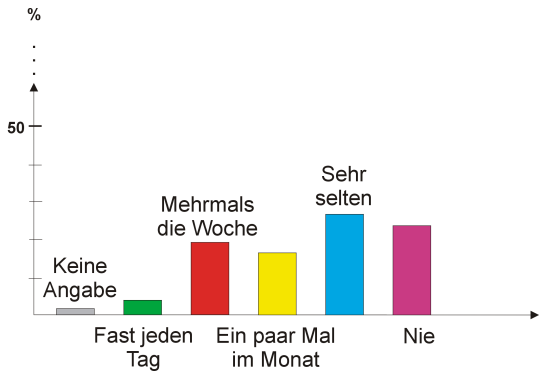

Abbildung 9.14: Internetnutzung unter Medizinstudenten 2000

Die einzelnen Internetdienste werden unterschiedlich stark eingesetzt. Im Mittel wird das E-Mailen am häufigsten genutzt, gefolgt vom World Wide Web. Das Usenet bzw. das Chatten hat hingegen nur eine untergeordnete Bedeutung.

Knapp die Hälfte der Studierenden gaben an, dass das Internet für ihr Studium sinnvoll nutzbar sei. Mittlerweile unentbehrlich wird das Netz nur von einer Person erachtet. Die andere Hälfte meint, dass das Internet ab und zu ganz nützlich sei, es aber auch ohne ginge bzw. vollkommen überflüssig sei.

Abbildung 9.15 zeigt die Antworten auf die Frage, in welchem Rahmen Studierende das Internet für ihr Studium nutzen.

Knapp zwei Drittel gaben an, im Internet nach Informationen zu suchen, 58% nutzen das Netz zur Literaturrecherche.

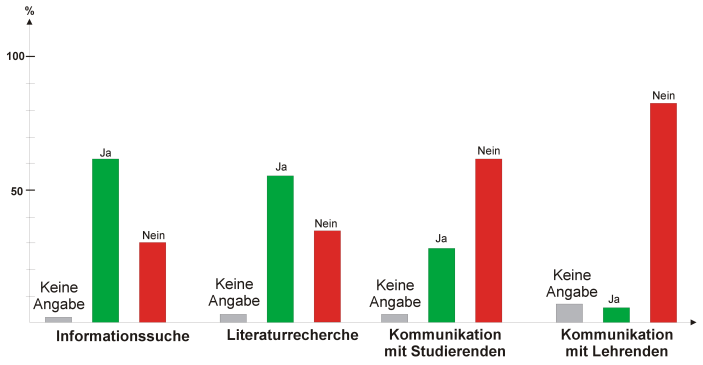

Abbildung 9.15: Internetnutzung für das Medizinstudium 2000

Die Möglichkeit zur Kommunikation mit Studierenden (im Rahmen des Studiums) wird immerhin noch von einem Drittel wahrgenommen, Kontakt zu Lehrenden über das Internet haben von den Befragten hingegen lediglich knapp 7%. Erfahrungen aus anderen Projekten zeigen, dass gerade die computervermittelten Kommunikationsformen nicht von selbst in Gang kommen, sondern fester in den Studien- bzw. Lehralltag integriert werden müssen. Besonders von den Lehrenden wird hierzu ein Umdenken erforderlich sein.

Eine überwältigende Mehrheit (90%) glaubt, dass das Internet für ihr Studium künftig zunehmend wichtiger werden wird. Immerhin knapp die Hälfte (47%) der Befragten haben dabei schon einmal computergestütze Lernsysteme für ihr Studium eingesetzt. Davon wurde am häufigsten die sog. „Gelbe Reihe“ o.ä. genannt, also Systeme, die sich vorwiegend mit dem Stellen von Aufgaben bzw. Multiple-Choice-Abfragen beschäftigen und weniger Lernsysteme im engeren Sinne darstellen.

Knapp zwei Drittel der Befragten erhoffen sich mehr Flexibilität (zeit- und ortsunabhängiger) von computergestütztem Lernen (vgl. Abbildung 9.16). Noch mehr (72%) erwarten mehr Abwechslung beim Lernen und 60% denken, dass das Lernen effektiver werden könnte.

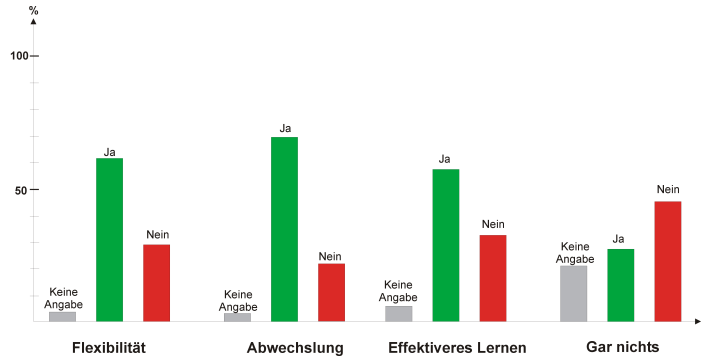

Abbildung 9.16: Erwartungen an Computergestütztes Lernen unter Medizinstudenten 2000

Diese Ergebnisse lassen sich für das System MURMEL so interpretieren, dass mit einer hinreichend großen Gruppe von ‚interneterfahrenen’ Studierenden gerechnet werden kann. Immerhin 20% gaben an, das Internet mehrmals pro Woche zu nutzen (6% sogar fast jeden Tag). Einer Evaluierung bezüglich der Qualität und Akzeptanz des Produktes steht somit in dieser Hinsicht zunächst einmal nichts im Wege.

Für das angestrebte Ziel einer ‚nachhaltigen Verbesserung der Lehre' muss aber ebenso berücksichtigt werden, dass es eine relativ große Gruppe von Studierenden gibt, die momentan keinen „bequemen" Weg ins Internet finden und folglich auch keine große Interneterfahrung aufweisen können.

Der Grund für die geringere Bedeutung des Internets bei dieser Gruppe scheint aber weniger an einer generellen ‚Computerfeindlichkeit' oder einer übergroßen Skepsis zu liegen, zumal eine große Mehrheit die zunehmende Bedeutung von Computern als positiv bewertet und 90% der Studierenden davon ausgehen, dass das Internet für ihr Studium künftig wichtiger werden wird. Es scheint vielmehr so, dass die Studierenden, sofern der Zugang und die Nutzung des Internets erleichtert und integraler Bestandteil des Studiums wird, durchaus bereit sind, die neuen Möglichkeiten des Internets (auch zum Lernen) zu nutzen. Hierzu müssen sinnvolle Nutzungsmöglichkeiten erfahrbar werden, d.h. es müssen auf die jeweiligen Studienabschnitte abgestimmte Inhalte in ansprechender Form angeboten werden, und der „herkömmliche" Lehrbetrieb muss Bezug auf diese Angebote nehmen.

Eine **zweite Fragebogenaktion** wurde zu Beginn des Wintersemesters 2000/2001 mit 42 Medizinstudenten des überwiegend vierten Semesters im Rahmen eines Neurophysiologie-Blockpraktikums durchgeführt. Die befragte Gruppe wies einen Altersbereich von 21 bis 28 Jahren auf. Im Folgenden werden schwerpunktmäßig die Evaluierungsergebnisse diskutiert, die signifikant unterschiedlich zur ersten Fragebogenaktion ausgefallen sind bzw. werden vor allem die konkreten Erfahrungen der Studierenden im Umgang mit dem MURMEL-System beschrieben.

Sämtliche Befragte stuften die zukünftige Bedeutung des Internets für das Lernen als besonders hoch ein. Dies zeigt auch die überwältigende Mehrheit von 92 %, die bereits mit computerbasierten Lernprogrammen gearbeitet hat. Bezüglich der Erwartungen an diese neue Lernform gaben 61 % der Befragten an, sich mehr Flexibilität beim Lernen zu erhoffen. Eine weitaus höhere Wertigkeit nimmt die Abwechslung und der Spass beim Lernen ein. 84 % bzw. 92 % stufen diese beiden Faktoren als besonders wichtig ein. Dagegen glauben nur knapp 50 % der Studierenden, dass durch computerbasiertes Lernen eine höhere Lerneffektivität erzielt werden kann.

Abbildung 9.17 zeigt eine Übersicht über die Bewertung der medizinischen Inhalte des MURMEL-Systems.

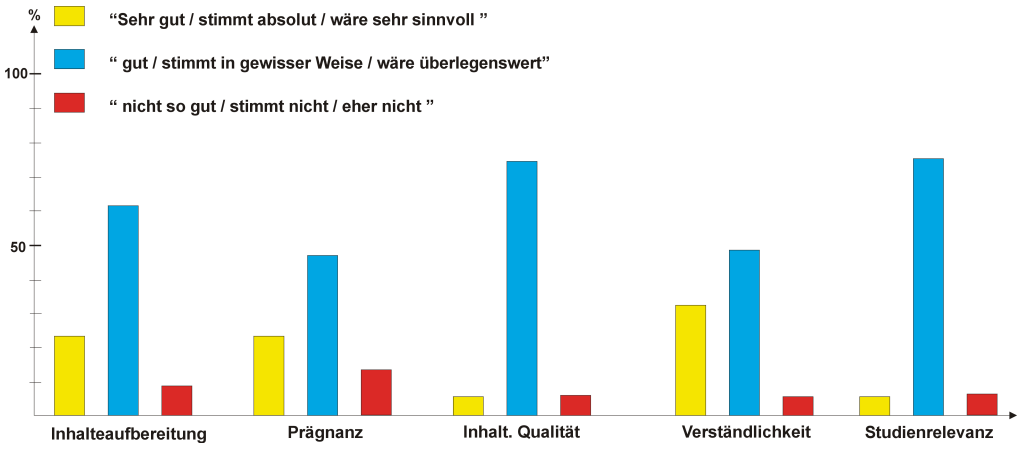

Abbildung 9.17: Bewertung der Inhalte des MURMEL-Systems unter Medizinstudenten

Über 90 % der Befragten bewerteten die aufbereiteten Inhalte als sehr bzw. ausreichend verständlich (23 % bzw. 69 %). Dass die Texte das Wesentliche prägnant zusammenfassen, entsprach der Meinung von 23 % bzw. 53 % der Befragten, die inhaltliche Qualität wurde von der überwiegenden Mehrheit als gut bezeichnet (84 %). Auch die Verständlichkeit der medizinischen Inhalte bewerteten lediglich knapp 8 %

als nicht ausreichend. Über 84 % der Studierenden stuften schließlich die Inhalte des MURMEL-Systems als studienrelevant ein.
Auch die Navigation und Orientierung innerhalb des e-Learning-Systems wurde von den Studierenden bewertet. Die diesbezüglichen Ergebnisse zeigt Abbildung 9.18.

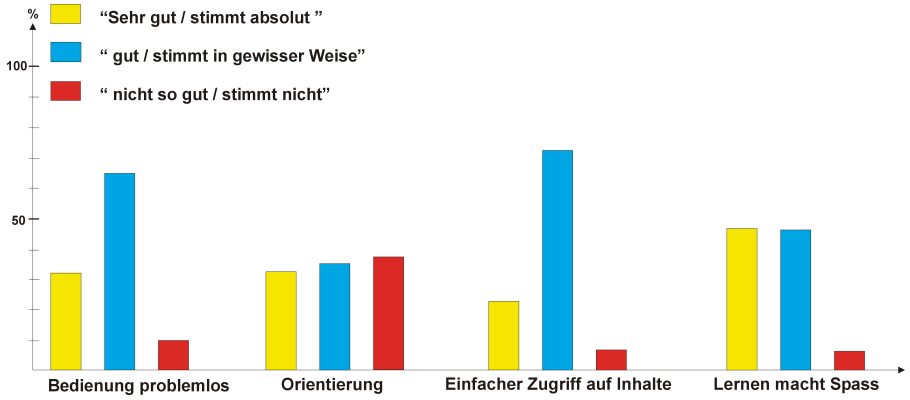

Abbildung 9.18: Bewertung der Orientierung innerhalb des MURMEL-Systems

Hier sind besonders die Antworten auf die Frage nach der Orientierung innerhalb von MURMEL interessant: Im Gegensatz zu den insgesamt eher positiven Beurteilungen gab es hier auch kritische Stimmen (38 %), die durch die ungeführten Möglichkeiten der Inhaltebenutzung verwirrt wurden. Dies kann mit der Tatsache zusammenhängen, dass sich das Medizinstudium bisher stark an festgelegten Lehrplänen und zu erlernenden Inhalten orientiert hat. Es scheint, dass sich die Medizinstudenten an die neue Form dieser ungeführteren Wissensaneignung zunächst gewöhnen müssen. Trotzdem bewerteten insgesamt 76 % der Befragten den Zugriff auf die eigentlichen Inhalte als problemlos und einfach. Positiv ist auch die Bewertung der Studierenden im Hinblick auf den Spass beim Lernen: Nur etwas über 7 % der Befragten machte die Arbeit mit dem MURMEL-System keinen Spass.

Schließlich wurden die Studierenden nach dem Bezug des e-Learning-Systems zum Medizinstudium befragt. Die entsprechenden Ergebnisse zeigt Abbildung 9.19.

Abbildung 9.19: Bewertung des Bezugs des MURMEL-Systems zum Medizinstudium

Hier erfolgten nahezu identische Aussagen darüber, dass sich das System gut in den Studienalltag integrieren ließe und es die Vorlesung zum Teil ersetzen könnte. Insgesamt positiv beantwortet wurde schließlich die Frage, ob das MURMEL-System auch in Zukunft genutzt werden würde.

9.2.3 Interview/Beobachtung

Im Rahmen der formativen Evaluierung wurden im Verlauf der Inhalteerstellung Studenten gebeten, das System zu testen. Dabei wurden sie von einem Pädagogen beobachtet und anschließend befragt.

Die Anmerkungen der Studierenden ergaben dabei keine besonders überraschenden Erkenntnisse:

- Das Design fand allgemeine Zustimmung und wurde als ansprechend bezeichnet.
- Die Navigation bereitete nur wenige Schwierigkeiten, die existierenden Schwierigkeiten verschwanden, nachdem wenige Minuten mit dem System gearbeitet worden war.
- Einige Studierende betonten, dass vorrangig die Qualität und die Vollständigkeit der dargebotenen medizinischen Inhalte von Bedeutung wäre, Design und Benutzerführung seien dagegen eher nebensächlich.
- Einige Studierende betonten die Nutzbarkeit der zur Verfügung stehenden Werkzeuge zur Benutzerunterstützung wie beispielsweise die Notizblockfunktion und die Bookmarkverwaltung.
- Auch wurde die Frage, ob computergestütztes Lernen generell Sinn macht, diskutiert. Insgesamt begrüßten die Studierenden den hohen Grad an Interaktivität und enthaltener Animationen.
- Es wurde bemerkt, dass eher wenige Studierende freiwillig ein Lernsystem wie MURMEL nutzen würden, sofern dieses nicht auch fester Bestandteil des Studienplans werden würde.
- Die Möglichkeiten der computervermittelten Kommunikation wurden als sehr positiv betrachtet, vorausgesetzt es findet sich eine hinreichend große Gruppe aktiv beteiligter Personen.
- Einige kritische Stimmen äußerten, es wäre eventuell sinnvoller, das Geld für die Entwicklung computergestützter Lernprogramme in den Stellenausbau 'herkömmlicher' Lehrkräfte und Lernmittel zu investieren. Es scheint, dass die Studierenden sich darüber im klaren sind, dass nicht alle Mängel im Studium durch mehr Multimedia gelöst werden können. Ebenso wird erkannt, dass mehr Multimedia auch Rationalisierung bedeuten kann.

Die Mehrzahl der angeführten Argumente spricht für die Sinnhaftigkeit eines breiteren Einsatzes des eLS-Baukastens auch in anderen Fachbereichen. Daneben existiert weiterhin die Notwendigkeit, die Akzeptanz von e-Learning-Systemen bei den Lernenden zu erhöhen. Dies kann nur durch aktive Unterstützung der Lehrenden und durch die Integration des MURMEL-Systems in das studentische Curriculum erreicht werden.

9.2.4 Statistische Auswertungen

Abschließend folgen an dieser Stelle einige Analysen der aktuellen Webserver Logdatei des MURMEL-Systems für den exemplarischen Zeitraum vom 06.11.2000 bis 05.12.2000. Bei dieser Logdatei des Apache Webservers werden weitaus vielfältigere Angaben protokolliert, als dies mit dem im NeuroAssistant verwendeten Netscape Enterprise Server möglich war. Das in Abschnitt 9.1.2 generell diskutierte Problem des „Deep Web“ existiert zwar weiterhin, wird aber besonders durch das implementierte Benutzermodell kompensiert. Es ermöglicht die nahezu lückenlose Protokollierung unterschiedlichster Aktionen und Verhaltensmuster, die zur Anpassung des Systems an einzelne Nutzer, vor allem aber auch im Hinblick auf zukünftig regelmäßig durchzuführende Evaluierungsmaßnahmen eingesetzt werden können. Ausführliche Untersuchungen und konkrete Angaben sind in [Mor01] zu finden.

Abbildung 9.20 und Tabelle 9.4 beschreiben den Zugriff über den gesamten Beobachtungszeitraum auf das System MURMEL. Erklärungen zu verwendeten Begriffen sind wiederum in Abschnitt 9.3 zu finden.

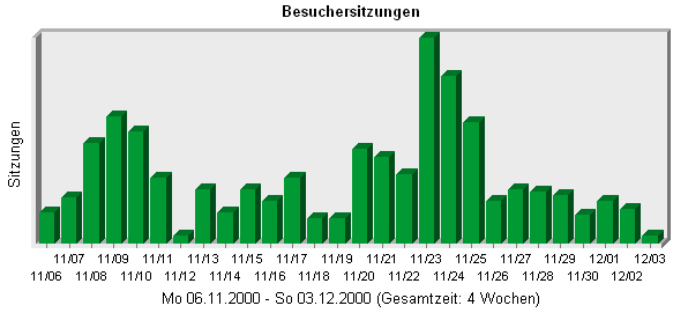

Abbildung 9.20: Allgemeine Zugriffstatistik des MURMEL Systems

Statistik		
Hits	Anzahl der erfolgreichen Hits auf die gesamte Site	69,693
	Mittlere Anzahl an Hits pro Tag	34,846
Seitenimpressionen	Anzahl der Seitenimpressionen	40,858
	Mittlere Anzahl an Seitenimpressionen pro Tag	20,429
Sitzungen	Anzahl der Anwendersitzungen	643
	Mittlere Anzahl an Anwendersitzungen pro Tag	321
	Mittlere Länge einer Anwendersitzung	00:18:30
	Internationale Anwendersitzungen	33.9%
	Anzahl der Besucher, die nur einmal kamen	220
	Anzahl der Besucher, die mehrmals kamen	55

Tabelle 9.4: Allgemeine Zugriffsstatistik des MURMEL-Systems

Die Graphik zeigt einen deutlichen Anstieg des Zugriffs auf den MURMEL-Server zu Beginn einer jeden Woche. Gegen Ende lassen die Aktivitäten dann wieder nach. Abbildung 9.21 zeigt, aus welchen geographischen Regionen und Ländern die Nutzer des MURMEL-Servers im Erfassungszeitraum kamen.

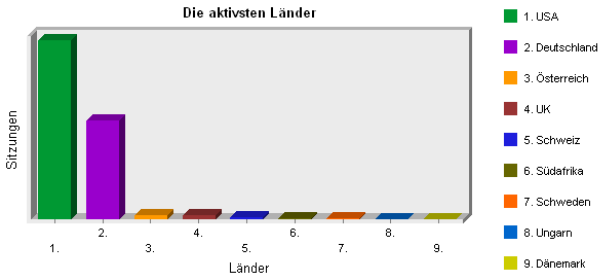

Abbildung 9.21: Aktivste Länder beim Zugriff auf das MURMEL System

Abbildung 9.22 und Tabelle 9.5 zeigen die wichtigsten aufgerufenen Seiten von MURMEL.

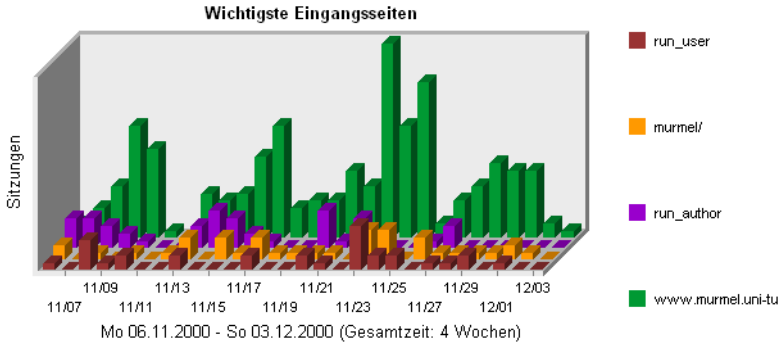

Abbildung 9.22: Wichtigste Eingangsseiten des MURMEL Systems

Wichtigste Eingangsseiten			
	Seiten	**% von Gesamt**	**Sitzungen**
1	Murmel: Ein multimediales Ausbildungs-system für die medizinische Lehre http://www.murmel.uni-tuebingen.de/	35.81%	221
2	http://www.murmel.uni-tuebingen.de/ cgi-bin/osform.exe/ run_author	6.96%	43
3	Murmel: Ein multimediales Ausbildungs-system für die medizinische Lehre http://www.murmel.uni-tuebingen.de/murmel/	5.99%	37
4	http://www.murmel.uni-tuebingen.de/ cgi-bin/osform.exe/ run_user	4.86%	30
5	Murmel http://www.murmel.uni-tuebingen.de/ frame_home.htm	4.86%	30
...			

Tabelle 9.5: Wichtigste Eingangsseiten des MURMEL Systems

Auf dem MURMEL-Server befindet sich neben der datenbankbasierten Version des Systems eine kleine Demoversion ohne Datenbankfunktionalität. Diese wurden laut Abbildung 9.22 am häufigsten aufgerufen. Die Service Applikation für Autoren, „run_author“, wurde am zweithäufigsten genutzt. Dies wird durch die weiterhin aktuelle Phase der Inhalteerstellung für das MURMEL-System bewirkt. Anschließend folgen die Applikation („run_user“) und die Einstiegsseite für die Benutzer, auf die von Nutzerseite zugegriffen wird. Im Hinblick auf die Autoren- und Benutzeranwendung endet die Protokollierung jeweils am CGI-Skript. Auch hier wird deutlich, dass folglich allein mit der Logdatei des Webservers keine wirklich detaillierte Benutzermodellierung realisierbar ist.

Abbildung 9.23 und Tabelle 9.6 zeigen die verschiedenen Verzeichnisse auf dem Webserver, die von den zugreifenden Nutzern angefordert wurden.

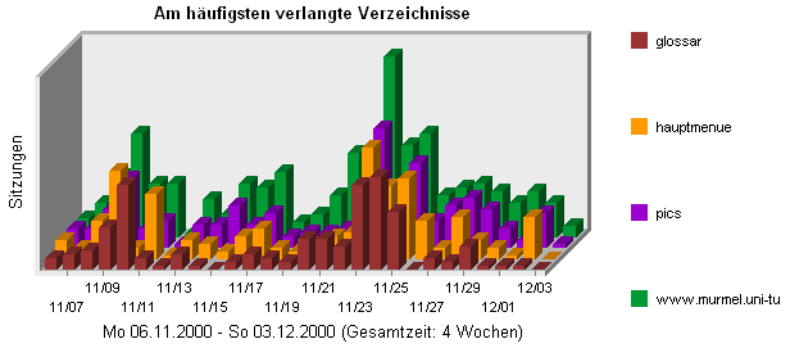

Abbildung 9.23: Die häufigsten verlangten Verzeichnisse des MURMEL Systems

Am häufigsten verlangte Verzeichnisse					
	Verzeichnispfad	**Hits**	**% von Gesamt**	**Nicht-zwischen gespeichert**	**Sitzung**
1	http://www.murmel.uni-tuebingen.de/	4,299	6.16%	80.94%	389
2	http://www.murmel.uni-tuebingen.de/pics	2,526	3.62%	84.56%	246
3	http://www.murmel.uni-tuebingen.de/hauptmenue	10,592	15.19%	87.68%	222
4	http://www.murmel.uni-tuebingen.de/glossar	684	0.98%	92.83%	162
...					

Tabelle 9.6: Die am häufigsten verlangten Verzeichnisse von MURMEL

Demzufolge wurden am häufigsten auf die „festverdrahtete“ Demoversion des MURMEL-Systems zugegriffen: Auf das Hauptverzeichnis, in dem sich die eigentlichen Inhalte befinden, auf die unterhalb von „pics“ befindlichen Bilddaten und an dritter Stelle auf das Glossar. Da die Inhalte der datenbankbasierten Version grundsätzlich über das CGI-Skript an die Clients gesendet werden, kann dies nicht statistisch erfaßt werden. „Nicht-zwischengespeichert“ bezieht sich hierbei auf den Prozentteil der Hits, die sich noch nicht im Zwischenspeicher des clientseitigen Browsers befanden.

Schließlich soll betrachtet werden, über welche Browser und Plattformen die Benutzer des MURMEL-Systems am häufigsten zugriffen. Abbildung 9.24 und 9.25 zeigen die ermittelten Daten.

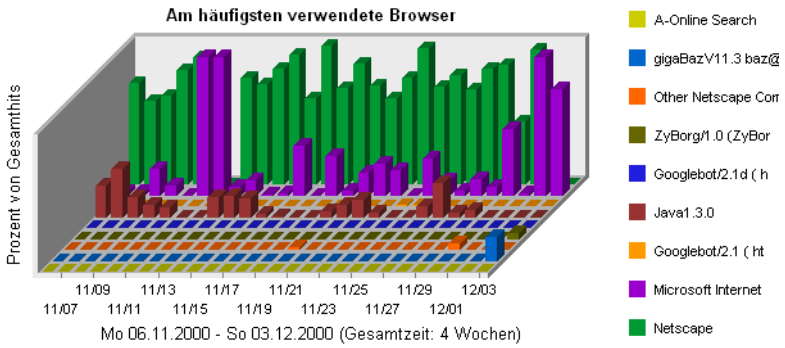

Abbildung 9.24: Am häufigsten verwendete Browser des MURMEL Systems

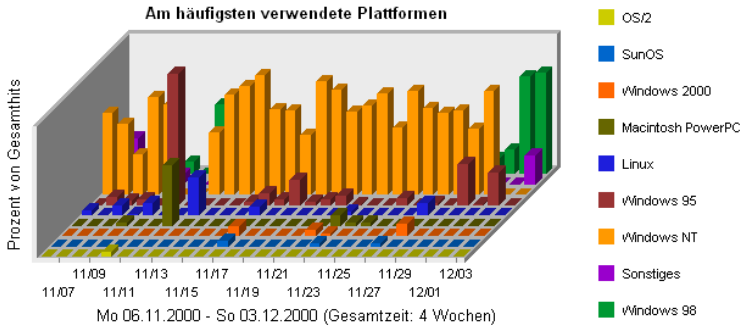

Abbildung 9.25: Am häufigsten verwendete Plattformen des MURMEL Systems

Demnach sind der Netscape Browser und der Internet Explorer die am häufigsten benutzen Browser, die meisten Benutzer verfügen darüber hinaus über einen Windows-basierten Rechner.

Die Auswertungen der Webserver Logdatei zeigen, dass sich mit Hilfe verschiedener Webserver-Einstellungen bereits aus der zugehörigen Logdatei wichtige Informationen ermitteln lassen, die zur Systemevaluierung herangezogen werden können. Eine Unterscheidung einzelner zugreifender Nutzer ist jedoch auch hier nur bedingt möglich. Das im Rahmen des eLS-Baukastens implementierte Benutzermodell löst dieses Problem in Zukunft durch die genauere Protokollierung detaillierter Daten zu einzelnen Benutzern.

Es wird sich zeigen, wie sich der eLS-Baukasten und die mit ihm generierten Systeme in der Praxis bewähren. Innerhalb des Projekts MURMEL werden dazu ab Wintersemesters 2000/2001 weiterführende Evaluierungen durchgeführt. Besonders die automatische Protokollierung und Auswertung des Benutzerverhaltens von Systemseite verspricht hier hilfreiche Informationen und Hinweise, die sowohl die inhaltliche, als auch die didaktische und technische Seite des e-Learning-Systems betreffen. Ausführliche Daten dazu sind in [Mor01] zu finden.

9.3 Legende zur Auswertung der Webserver Logdateien

Die folgende Liste enthält die Legende zu den in den Unterabschnitten 9.1.2 und 9.2.4 angegebenen Webserverauswertungen.

- **Hits**
  Gesamtzahl der Hits als Anzahl aller erfolgreichen Hits, einschließlich von HTML-Seiten, Bildern, Formularen, Skripts und heruntergeladenen Dateien, **Anzahl der Hits auf der gesamten Site** (erfolgreich), **mittlere Anzahl der Hits pro Tag** als Anzahl der erfolgreichen Hits dividiert durch die Gesamtanzahl der Tage in der Logdatei und die **Anzahl der Hits auf der Homepage**, die angibt, wie häufig die Homepage besucht wurde.

- **Seitenansichten/impressionen**
  Gesamtzahl der Seitenansichten als Anzahl der Hits auf Seiten, die als „Dokumente“ oder „Formulare“ definiert sind, die **mittlere Anzahl der Seitenansichten pro Tag** als Anzahl der Seitenansichten dividiert durch die Gesamtanzahl der Tage in der Logdatei und **Dokumentansichten** als Anzahl der Hits auf Seiten, die gemäß Definition als Dokumente erkannt werden. Dateien, die als Formulare definiert wurden, werden nicht erfaßt.

- **Sitzungen**
  Anzahl der Besuchersitzungen auf der Site, die **Dauer** einer Besuchersitzung, die **mittlere Anzahl der Besuchersitzungen pro Tag** als Anzahl der Besuchersitzungen dividiert durch die Gesamtanzahl der Tage in der Logdatei, die **mittlere Länge** (ohne Länge Null) der Besuchersitzungen, Besuchersitzungen „International“, „Unbekannt“ und „USA“.

- **Besucher**
  Anzahl der eindeutigen IPs innerhalb der Berichtszeitspanne, unabhängig davon, ob die Echtheit über Domänenname oder Cookie bestätigt wurde. Eine Besuchersitzung ist standardmäßig 30 Minuten.

Kapitel 10

Schlußbemerkungen

10.1 Zusammenfassung

Der Forschungsschwerpunkt im Bereich computerbasierten Lernens und Lehrens hat sich in jüngster Zeit in Richtung internetbasierter Systeme verschoben. Trotz vielversprechender Möglichkeiten der so zur Verfügung stehenden IuK-Techniken werden diese in der Aus- und Weiterbildung bislang jedoch nur punktuell eingesetzt.

Die vorliegende Dissertation hat die Gründe für diese zögerliche Entwicklung aufgezeigt und Ansätze zu deren Lösung entwickelt. Einleitend wurde erörtert, welche Herausforderungen an das Thema Lernen und Wissensvermittlung aufgrund der neuen IuK-Technologien entstanden sind und welche derzeitigen Probleme im Zusammenhang mit computerbasierten Lernsystemen existieren. Unter Berücksichtigung der dabei erarbeiteten Problemstellungen wurden anschließend die theoretischen Grundlagen von e-Learning-Systemen näher beleuchtet, Aufbau und Funktionsweise aus technischer und didaktischer Sicht betrachtet, ein kritischer Blick auf die derzeitigen Möglichkeiten der Benutzerunterstützung in diesen Systemen geworfen und Möglichkeiten der Inhalteerstellung für e-Learning-Systeme diskutiert.

Zur Beantwortung der Frage, auf welche Weise eine effiziente Datenverwaltung innerhalb dieser Systeme ermöglicht werden kann, wurden Argumente für den Einsatz eines Datenbanksystems zur Informationsverwaltung erarbeitet. Anhand der Gegenüberstellung relationaler und objektorientierter Konzepte wurden dabei die Vorteile eines objektorientierten Ansatzes hervorgehoben.

Eine besondere Rolle innerhalb von e-Learning-Systemen nimmt die Informationssuche in den stark vernetzten, multimedialen Inhalten ein. Die notwendigen Bedingungen für eine echte Unterstützung der Benutzer wurden anhand der getrennten Untersuchung von text- und bildbasierter Informationssuche evaluiert und resultierten in der Erkenntnis, dass e-Learning-Systeme durch ihre multimedialen, vernetzten Inhalte, das WWW als Präsentationsmedium und eine zugrunde liegende Datenbank sehr gute Voraussetzungen für eine effiziente, umfassende Informationssuche bieten.

Nachdem zur Verankerung dieser theoretischen Konzepte einige schon existierende Systeme zum e-Learning und zur Informationssuche vorgestellt wurden, folgte schließlich die Erläuterung des im Rahmen dieser Dissertation entwickelten eLS-Baukastens. Das nach dem Komponentenparadigma konzipierte System vermeidet die im ersten Teil der Arbeit erörterten Problemstellungen und erzeugt e-Learning-Systeme, die sich durch hohe Flexibilität und performante Zugriffsmöglichkeiten speziell über das Netz auszeichnen. Sie verfügen über eine effiziente, objektorientierte Komponente zur Datenverwaltung und basieren auf einem ausgefeilten Datenmodell zur Ablage verschiedener Wissensarten und Lernstrategien. Sie werden initial einheitlich und fachbereichsunabhängig aufgesetzt und erhalten über die angekoppelten Administrations- und Autorensysteme ihren individuellen Charakter. Der eLS-Baukasten eignet sich daher für einen Einsatz innerhalb unterschiedlicher Kontexte und Anwendungsgebiete. Jedes System ermöglicht multiple Perspektiven auf die präsentierten Inhalte. Durch deren hohen Interaktivitätsgrad

wird das selbstgesteuerte Lernen bei den Benutzern gefördert. Durch die Verwendung des WWWs als Distributions- und Nutzungsplattform ist dabei außerdem ein einfacher und kostengünstiger Zugang sowohl für Autoren als auch Lernende gegeben.

Ausführlich wurden die verschiedenen Anwenderschnittstellen für Administratoren, Benutzer und Autoren beschrieben, die in jedem generierten e-Learning-System enthalten sind. Speziell für die Benutzer der e-Learning-Systeme wurden neuartige Verfahren entwickelt, die sie bei der Arbeit mit dem System unterstützen. Transparente Werkzeuge zur Benutzerprotokollierung stellen dabei eine fundierte Basis für flexible Formen der Nutzer- und Systemevaluierung dar. Erste Evaluationen wurden bereits durchgeführt und werden in Kapitel 9 beschrieben. Das entwickelte Benutzermodell sorgt hierbei für eine nahezu lückenlose Protokollierung des individuellen Benutzerverhaltens.

Die in jedes e-Learning-System eingebettete Komponente zur multimedialen Informationssuche stellt ein in dieser Form erstmalig realisiertes, anwendungsunabhängiges Verfahren dar. Dies wird durch eine Kombination von attribut-, text- und bildbasierten Methoden bei gleichzeitiger Berücksichtigung unterschiedlicher Kontextinformationen möglich. Unter der einschränkenden Bedingung, e-Learning-Systeme für verschiedene Anwendungsbereiche anbieten zu wollen, ist eine derartig konzipierte Retrievalkomponente ein möglichee Weg, praxistaugliche Suchwerkzeuge anzubieten. Bisher ist das vorgestellte Verfahren einer der wenigen Ansätze, der neben verschiedenen Indizierungs- und Retrievaltechniken an eine komplexe Datenbankanwendung gekoppelt ist. Mit dieser Architektur unterscheidet sich jedes generierte e-Learning-System grundlegend von existierenden Informationssystemen und herkömmlichen WWW-Suchmaschinen. Gerade im Hinblick auf das Ziel eines e-Learning-Systems, nämlich der Unterstützung von Lernenden, kann durch eine solche Informationssuche die Vermittlung von Inhalten um eine weitere, neue Sicht auf komplexes Wissen ergänzt werden.

Der eLS-Baukasten stellt ein Softwaresystem dar, das sich in unterschiedlichsten Anwendungsbereichen einsetzen lässt. Es erlaubt die einfache Generierung modularer, komponentenbasierter e-Learning-Systeme, die sich durch eine besonders flexible Struktur auszeichnen und über zahlreiche Komponenten zur multimedialen und interaktiven Wissenspräsentation, zur Benutzerunterstützung und zur Suche in den komplexen Inhalten verfügen.

10.2 Ausblick

Mit dem eLS-Baukasten wurde ein Referenzmodell und eine Methodik für die automatisierte Generierung von e-Learning-Systemen entwickelt und in die Praxis umgesetzt.

Offene Fragestellungen betreffen verschiedene, größtenteils technische Aspekte. So ist beispielsweise zu evaluieren, ob die generierten e-Learning-Systeme unter jeglichen Bedingungen bezüglich Anzahl parallel zugreifender Clients, Datenbankgröße oder Netzbelastung ihre Leistungsfähigkeit aufrecht erhalten können. Wie an verschiedenen Stellen innerhalb der Arbeit angeführt, wurden die verschiedenen Komponenten jedoch besonders unter Beachtung dieser Gesichtspunkte entwickelt.

Bezüglich der momentan eingesetzten Techniken ist es erforderlich, kontinuierlich die Entwicklung neuer Techniken zu beobachten und den eLS-Baukasten gegebenenfalls entsprechend weiterzuentwickeln. Eine wichtige Rolle kommt dabei der Berücksichtigung zukünftiger Standards im Bereich der webbasierten Informationssysteme zu. Seit einigen Jahren entwickeln verschiedene Organisationen, beispielsweise das *IMS Global Learning Consortium* [ims00] und das *IEEE Learning Technology Standards Committee* [iee00] Standards für die Entwicklung und Implementierung von e-Learning- Systemen. Unterstützt von Firmen wie IBM, Oracle, Sun oder Macromedia entstehen Spezifikationen und Softwareprodukte, die die webbasierte Erstellung, Verwaltung und Distribution von Lerninhalten standardisieren sollen. Diesbezügliche Aktivitäten und Entwicklungen gilt es zu beobachten bzw. aktiv mitzugestalten.

Trotz teilweise noch ausbaufähiger Akzeptanz computergestützter Lernsysteme bei Lehrenden und

Lernenden wird sich die derzeitige Entwicklung nicht aufhalten lassen. Auch ist zu erwarten, dass durch die Synergie virtueller Lehre und klassischer Präsenzveranstaltungen im Hinblick auf Qualitätssteigerung, Motivationsförderung, beschleunigtem Wissenstransfer und Kostenreduktion durch optimale Nutzung von Ressourcen ein deutlicher Mehrgewinn in Studium, Aus- und Weiterbildung erzielt werden kann.

Positives Beispiel dieser Entwicklung ist das im Mai 2001 anlaufende, länderübergreifende BMBF-geförderte Projekt „Prometheus“. Das Projekt basiert auf einer Kooperation des Universitätsklinikums Tübingen, dem Lehrstuhl für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts für Informatik und Projektpartnern der in Abbildung 10.1 dargestellten Universitäten. Verwaltet und gesteuert von einer zentralen Projektgruppe in Tübingen will das Projekt durch die Kooperation verschiedener medizinischer Fakultäten einen Beitrag zur Reform des Studiengangs Medizin liefern. Erklärtes Ziel und Schwerpunkt ist die Integration vorhandener Inhalte und Methoden in einer einheitlichen Plattform und die Schaffung von Planungs-, Kontroll-, Steuerungs- und Migrationsstrukturen, die der Qualitätssicherung und der permanenten intensiven Evaluierung der Studierenden, der Lehrenden und des Systems dienen. Neben Softwareressourcen anderer Kooperationspartner soll auch der eLS-Baukasten innerhalb der zu entwickelnden Plattform Einsatz finden und die Basis für zukünftige Weiterentwicklungen darstellen.

Abbildung 10.1: Übersicht der Kooperationspartner im Projekt PROMETHEUS

Literaturverzeichnis

[Abi93] 1. bis 8. GI-Workshop *Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*. Ges. für Informatik (GI) e.V. Fachgruppen 1.1.4 und 2.3.3, GMD - Forschungszentrum Informationstechnik GmbH.

[Ach00] A. Achilles: *SQL: Standardisierte Datenbanksprache vom PC bis zum Mainframe*. 7. Auflage, Oldenbourg Verlag; München, Wien, 2000.

[Ah92] S. Ahmed, A. Wong, D. Sriram, R. Logcher: *Object-oriented database management systems for engineering: A comparison*. Journal of Object-Oriented Programming, Bd. 5, Nr. 3, 1992.

[All98] P. Allen, S. Frost: *Component-Based Development for Enterprise Systems*. Cambridge University Press, 1998.

[Als96] P. Alshuth, T. Hermes, C. Klauck, J. Kreyß, M. Röper: *IRIS - Image retrieval for images and videos*. In: Center for Advanced Studies Conference, Toronto, November, 1996.

[Als98] P. Alshuth, Th. Hermes, L. Voigt and O. Herzog: *On Video Retrieval: Content Analysis by ImageMiner*. In: Proc. SPIE, Vol. 3312, San Jose, Jan. 1998.

[Ans75] ANSI/X3/SPARC Study Group on Database Management Systems, Interim Report, FDT ACM SIGMOD 7,2, 1975.

[Ard97] E. Ardizzone, M. Cascia: *Automatic Video Database Indexing and Retrieval*. Multimedia Tools and Applications, Vol. 4, 1997.

[Arn00] K. Arnold, J. Gosling, D. Holmes: *the Java Programming Language*. 3. Auflage, Addison-Wesley Longman, Amsterdam; Sun Microsystems, 2000.

[Ass98] R. Assfalg, U. Goebels, H. Welter: *Internet Datenbanken. Konzepte, Methoden, Werkzeuge*. Addison-Wesley, 1998.

[Atk89] Atkinson, Manish et al: *The Object-oriented Database System Manifesto*. In: Proc. of 1. Intern. Conf. On Deductive and Object-Oriented Databases (DOOD),1989.

[Atz98] P. Atzeni, G. Mecca, P. Merialdo: *Design and maintenance of data-intensive Web sites*. In: VI Int. conference on Extending Database Technology (EDBT98), Spanien, 1998.

[Bac96] J.R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, C. Shu: *The Virage image search engine: An open framework for image management*. In: SPIE Stoarge and Retrieval for Image and Video Databases, IV, San Jose, CA, USA; 1996.

[Bae99] T. Baehring, A. Becker: *Web-basiertes Lehren und Lernen mit authentischen Fällen in der Medizin.* In: Rechnergestützte Verfahren in Orthopädie und Unfallchirugie. Hrsg.: J. Jerosch, K. Nicol, K. Peikenkamp; Steinkopff Verlag, Darmstadt, 1999.

[Bar99] D. Bartz, M. Skalej, D. Welte, W. Straßer: *3D Interactive Virtual Angiography*. In : Lemke, H.U., Vaannier, M.W., Inamura, K. (Hrsg): Computer Assisted Radiology and Surgery (CARS99). Paris, 1999.

[Bau94] P. Baumgartner, S. Payr: *Lernen mit Software*. Reihe Digitales Lernen. Österreichischer Studienverlag. Innsbruck. 1994.

[Bau99] P. Baumgartner: *Evaluation mediengestützten Lernens. Theorie - Logik - Modelle*. In: M. Kindt (Hrsg.): Projektevaluation in der Lehre. Medien der Wissenschaft, Band 7, Waxmann Verlag, Münster, 1999.

[Blu98] A. Blumstengel: *Entwicklung hypermedialer Lernsysteme*. Wissenschaftlicher Verlag Berlin. 1998.

[Bmb98] Bundesministerium für Bildung und Forschung: *Zur Zukunft der Weiterbildung in Europa. Lebenslanges Lernen für Alle in veränderten Lernumwelten*. Referat Öffentlichkeitsarbeit, Bonn-Berlin, 1998.

[Bmb99] Bundesministerium für Bildung und Forschung: *Innovationen und Arbeitsplätze in der Informationsgesellschaft des 21. Jahrhunderts.* Referat Öffentlichkeitsarbeit, Bonn-Berlin, September, 1999.

[Bmb00] Bundesministerium für Bildung und Forschung: *Unternehmen Zukunft - Innovationsförderung. Hilfe für Forschung und Entwicklung*. Referat Öffentlichkeitsarbeit, Bonn-Berlin, Juli, 2000.

[Bre99] J. Bredno, S. Brandt, J. Dahmen, B. Wein, T. Lehmann : *Kategorisierung von Röntgenbildern mit aktiven Konturmodellen.* In: Bildverarbeitung für die Medizin 2000. München, 2000.

[Boo95] G. Booch: *Objektorientierte Analyse und Design.* Addison-Wesley, München, 1995.

[Bru00] B. Bruns, P. Gajewski: Multimediales Lernen im Netz - Leitfaden für Entscheider und Planer. 2. Auflage, Springer, 2000.

[Bru00] P. Brusilovsky, O. Stock, C. Strapparava: *Adaptive Hypermedia and Adaptive Web-Based Systems*. Int. Conf. on Adaptive Hypermedia and Adaptive Web-based Systems. Lecture Notes in Computer Science 1892. Springer, 2000.

[Can86] J. Canny: *A computational approach to edge detection.* In: IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI - 8(6), November, 1986.

[Car97] C. Carson, S. Belongie, H. Greenspan, J. Malik: *Region-based image querying.* In: Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries, Juni 1997.

[Car00] C. Carson, S. Belongie, H. Greenspan, J. Malik: *Color- and Texture-Based Image Segmentation using EM and its Application to Image Querying and Classification*. In: IEEE Pattern Analysis and Machine Intelligence, 2000.

[Cat97] R. Cattell: *Object database standard ODMG 2.0*. Morgan Kaufmann, 1997.

[Cle66] C.W. Cleverdon, E.M. Keen: *Factors Determining the Performance of Indexing Systems*. Cranfield, UK. Aslib Cranfile Research Project, 1966.

[Cod82] E.F. Codd: *Relational database: A practical foundation for productivity.* Communications of the ACM, Band 25, Nr. 2, S. 109-117, Februar, 1982.

[Cod90] E.F. Codd: *The Relational Model for Database Management: Version2*. Addison-Wesley, Reading, MA, 1990.

[Coh00] L. Cohen: *Second Generation Searching on the Web*. University at Albany Libraries. http://library.albany.edu/internet/second.html, Oktober, 2000.

[Cro95] W. Croft: *What do people want from Information Retrieval?* In: D-Lib Magazine, November, 1995.

[Dae99] C. Daetwyler: *WWW-Lernprogramme in der Medizin*. In: Biomedical Journal - Telemedizin, Heft 53, 1999.

[Dah00] J. Dahmen, T. Lehmann, K. Spitzer, H. Ney: *Image Retrieval für klinische Bilddatenbanken.* In: Bildverarbeitung für die Medizin 2000. München, 2000.

[Dic00] H. Dicken, G. Hipper, P. Müßig-Trapp: *Datenbanken unter Linux. Oracle 8i, MySQL, Adabas, Informix, Sybase, DB2, PostgreSQL, MiniSQL, Empress*. MITP-Verlag, 2000.

[Duf00] F. Duffner, Abtlg. für Neurochirugie: http://www.medizin.uni-tuebingen.de/kliniken/neurochir_kl/

[Dug99] M. Duga, S. Noachtar: *Computer-Based-Training in der Neurologie mit Intranet-Technik am Beispiel Epilepsie*. In: Biomedical Journal - Telemedizin, Heft 53, 1999.

[Eak96] J.P. Eakins: *Automatic Image Content Retrieval - Are we getting anywhere?* 3. Int. Conf. Electronic Library and Visual Information Research. De Montford University, Milton Keynes, UK, 1996.

[Elm94] R. Elmasri, S. B. Navathe: *Fundamentals of database systems*. 2. Auflage. The Benjamin/Cummings Publishing Company, Inc., Redwood City, 1994.

[Enc97] M. Encarnaçao: *Concept and realization of intelligent user support in interactive graphics applications*. Dissertation, WSI/GRIS, Universität Tübingen, 1997.

[Erb97] G. Erb, M. Skalej, M. Uesbeck: *NeuroAssistant ein computerunterstützter Arbeitsplatz für die Neuroradiologie*. In: Medic@l News, KODAK, Vol. 4, 1997.

[Fal94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner: *Efficient and Effective Querying by Image Content*. Journal of Intelligent Information Systems. Vol. 3, 1994.

[Fer00] O. K. Ferstl , K. Hahn, K. Schmitz, C. Ullrich, C.: *Funktionen und Architektur einer Internet-Lernumgebung für individuelles und kooperatives Lernen*. In: Uellner, S.,Wulf, V. (Hrsg.):Heidelberg 2000: Vernetztes Lernen mit digitalen Medien. Proc. der ersten Tagung Computergestütztes Kooperatives Lernen (D-CSCL 2000), Darmstadt, März, 2000.

[Fid83] R. Fidel, D. Soergel: *Factors affecting online bibliographic retrieval: a conceptual framework for research.* J Am Soc Info Sci, 34, S. 163-180, 1983.

[Fid91a] R. Fidel: *Searcher's selection of search keys. I. The selection routine.* J Am Soc Infor Sci, 42, S. 490-500, 1991.

[Fid91b] R. Fidel: *Searcher's selection of search keys. II. Controlled vocabulary or free-text searching.* J Am Soc Infor Sci, 42, S. 501-514, 1991.

[Fid91c] R. Fidel: *Searcher's selection of search keys. III. Searching styples.* J Am Soc Infor Sci, 42, S. 515-527, 1991.

[Fil97] T.J. Filler, E.T. Peuker: *Stellenwert des Internets in der medizinischen Lehre.* MMW 139:764-767; 1997.

[Fil99] T.J. Filler, E.T. Peuker, J. Jerosch, G. Wessendorf: *Techniken für eine interaktive Nutzung des Internets.* In: Rechnergestützte Verfahren in Orthopädie und Unfallchirugie. Hrsg.: J. Jerosch, K. Nicol, K. Peikenkamp; Steinkopff Verlag, Darmstadt, 1999.

[Fis98] M. Fischer, J. Konschak (1998): *Fall geknackt mit CASUS*. In: M. Abel, H.-D. Berdelsmann, B. Berendt (Hrsg.): Handbuch Hoschschullehre.

[Fis95] S. Fischer, R. Lienhart, W. Effelsberg: *Automatic Recogition of Film Genres*. Proc. ACM Multimedia 95, San Francisco, CA, Nov. 1995.

[Fra66] H. Frank: *Ansätze zum algorithmischen Lehralgorithmieren.* In: H. Frank (Hrsg.): Lehrmaschinen in kybernetischer und pädagogischer Sicht, Band 4, Ernst Klett Verlag, 1966.

[Fri00] R. Fricke: Qualitätsbeurteilung durch Kriterienkataloge. Auf der Suche nach validen Vorhersagemodellen. In P. Schenkel, S. Tergan, A. Lottmann (Hrsg.): Qualitätsbeurteilung multimedialer Lern- und Informationssysteme. Evaluationsmethoden auf dem Prüfstand. Verlag Bildung und Wissen, Nürnberg, 2000.

[Fro00] J. Froese, M. Moazzami, C. Rautenstrauch, H. Welter: *Efficient server programming with Oracle 7*. Addison Wesley, 2000.

[Fuh00] N. Fuhr: *Probabilistic Datalog: Implementing Logical Information Retrieval for Advanced Applications*. Journal of the American Society for Information Science 51(2), pages 95-110, 2000.

[Gei95] K. Geiger: *Inside ODBC (Open Database Connectivity)*. Microsoft Press, 1995.

[Gem96] GemStone Systems: *GemStone Programming Guide*. Release 5.0, 1996.

[Ger95] H. Gerdes: *Lernen mit Hypertext*.
[http://www.psychologie.uni-bonn.de/ gerdes_h/hyper/inhalt.htm]

[Ger99] T. Gericke: *Modellversuch: Einsatz und Evaluierung eines problemorientierten Lernprogrammes in der inneren Medizin*. In: Medizinische Klinik, 94, No. 2, Urban & Vogel, München, 1999.

[Gol98] Goldmann Lexikon, Bertelsmann Lexikographisches Institut, Bertelsmann Lexikon Verlag, Gütersloh, 1998.

[Gri98] F. Griffel: *Componentware - Konzepte und Techniken eines Softwareparadigmas*. dpunkt-Verlag, 1998.

[Gut00] D. Gutierrez: *Web-Datenbanken für Windows-Plattformen entwickeln. ADO, OLE DB, ODBC, JDBC und mehr*. Verlag Markt und Technik, 2000.

[Här99] T. Härder, E. Rahm: *Datenbanksysteme. Konzepte und Techniken der Implementierung*, Springer, 1999.

[Haa97] M. Haag, F.J. Leven: *WWW-basierte Lehr-/Lernsystem für die medizinische Ausbildung: Stand und zukünftige Entwicklungen*. In: H. Conradi, R. Kreutz, K. Spitzer (Hrsg.): CBT in der Medizin -Methoden, Techniken, Anwendungen-. Proc. zum Workshop in Aachen. Juni, 1997.

[Haa98a] M. Haag: *Plattformunabhängige, adaptive Lehr-/Lernsysteme für die medizinische Aus- und Weiterbildung*. Dissertation der Medizinischen Fakultät, Universität Heidelberg, 1998.

[Hab95] P. Haberäcker: *Praxis der digitalen Bildverarbeitung und Mustererkennung*. München : Hanser, 1995.

[Han98] F. Hanisch: *Hypermediales Lernen und Lehren von Computergraphik*. Diplomarbeit, WSI/GRIS, Universität Tübingen, 1998.

[Han99] F. Hanisch, R. Klein, W. Straßer: *Ein Web-basierter Computergraphik-Kurs im Baukastensystem*. In: Martin Engelien and Jens Homann (Hrsg.): Virtuelle Organisation und Neue Medien - Workshop GeNeMe99, Eul-Verlag, Lohmar, 1999.

[Han00] F. Hanisch: *Basic Requirements for Interactive Web-based Courseware*. In: M. Auer (Hrsg.): Interactive Computer aided Learning Applications and Experiences, Carintha Tech Institute – School of Electronics, 2000.

[Han88] M.J. Hannafin, K.L. Pecl: *The Design, Development, and Evaluation of INstructional Software*. MacmillanPublishing Company, New York, 1988.

[Har73] R.M. Haralick, K. Shanmugam, I. Dinstein: *Textural Features for Image Classification*. In: IEEE Transactions on Systems, Man, and Cybernetics, 1973.

[Har98] D. Harvey, S. Beitler: *The Developer's Guide to Oracle Web Application Server 3*. Addison-Wesley Longman, Amsterdam, 1998.

[Hau99] M. Hauff: *Neues Lehren und Lernen im Netz*. In: K. Lehmann (Hrsg.): Studieren 2000. Medien der Wissenschaft, Band 8, Waxmann Verlag, Münster, 1999.

[Her95] T. Hermes, C. Klauck, J. Kreyß, J. Zhang: *Image Retrieval for Information Systems*. In: Storage and Retrieval for image and video databases (SPIE), 1995.

[Heu97] A. Heuer: *Objektorientierte Datenbanken: Konzepte, Modelle, Standards und Systeme*. 2. Auflage, Addison-Wesley, 1997.

[Hof95] M. Hofmann, L. Simon: *Problemlösung Hypertext: Grundlagen, Entwicklung, Anwendung.* Carl Hanser Verlag. München, Wien. 1995.

[Hoh96] U. Hohenstein, R. Lauffer, K.D. Schmatz, P. Weikert: *Objektorientierte Datenbanksysteme*. Vieweg, Braunschweig, 1996.

[Hoh98] U. Hohenstein, V. Pleßer: *Oracle 8. Effiziente Anwendungsentwicklung mit objektrelationalen Konzepten*. dpunkt, 1998.

[Hor83] G.L. Horowitz, J.D. Jackson, H.L. Bleich: *PaperChase: self-service bibliographic retrieval*. JAMA, 328, S. 2495-2500, 1983.

[Hor99] A. Horsch, T. Balbach, M. Hogg, F. Sturm: *The Case-based Internet Textbook ODITEB For Multi-modal Diagnosis Of Tumors - Development, Features And First Experiences.* MIE99, Ljubljana, 1999.

[How98] N. Howe: *Percentile blobs for image similarity*. In: Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries, Juni 1998.

[Hua97] J. Huang, S. Kumar, M. Mitra, W. Zhu, R. Zabih: *Image Indexing using color correlograms*. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1997.

[Hug92] J.G. Hughes: *Objektorientierte Datenbanken*. Dt. Übersetzung von A.S. Kratzer. Hanser; London: Prentice-Hall, München; Wien. 1992.

[Ill99] T. Illmann, M. Weber, A. Martens, A. Seitz: *A Pattern-Oriented Design of a Web-Based and Case-Oriented Multimedia Training System in Medicine*. 4th World Conference on Integrated Design & Process Technology, Kusadasi, Türkei, 1999.

[Iss97] L.J. Issing : *Instruktionsdesign für Multimedia*. In: L.J. Issing, P. Klimsa: Information und Lernen mit Multimedia. Psychologie Verlags Union, Weinheim. 1997.

[Iss99] L.J. Issing: *Neue Medien - neue Organisationsformen des Lehrens und Lernens*. In: K. Lehmann (Hrsg.): Studieren 2000. Medien der Wissenschaft, Band 8, Waxmann Verlag, Münster, 1999.

[Ita91] Itasca Systems, Inc.: *ITASCA Technical Summary*. Release 2.0, 1991.

[Jäh97] B. Jähne: *Digitale Bildverarbeitung*. 4. Aufl. Berlin; Heidelberg: Springer, 1997.

[Jai96] A.K. Jain, A. Vailaya: *Image retrieval using color and shape*. Pattern Recognition, 29(8), 1996.

[Jah00] B. Jahnke, A. Altenburger, N. Högsdal: *Cockpit: Betriebswirtschaftliche Lernumgebungen*. In: H. Krahn, J. Wedekind (Hrsg): Medien in der Wissenschaft, Bd. 9: Virtueller Campus '99, Waxmann-Verlag, Münster, New York, 1999.

[Jer99] J. Jerosch, A. Voiculescu, K. Stewing: *Computergestützter Unterricht (CUU) in der Orthopädie.* In: Rechnergestützte Verfahren in Orthopädie und Unfallchirugie. Hrsg.: J. Jerosch, K. Nicol, K. Peikenkamp; Steinkopff Verlag, Darmstadt, 1999.

[Jia00] X. Jia: *Object oriented software development using Java : principles, patterns, and frameworks*. Reading, Mass.: Addison Wesley, 2000.

[Jör98] T. Jöring, S. Michel, M. Popella: *Intelligentes Marketing durch adaptive Produktpräsentationen im Web*. In: M. Engelien, K. Bender (Hrsg.): GeNeMe98 - Gemeinschaften in Neuen Medien, Josef Eul Verlag, 1998.

[Kei99] R. Keil-Slawik: *Evaluation als evolutionäre Systemgestaltung*. In: M. Kindt (Hrsg.): Projektevaluation in der Lehre. Medien der Wissenschaft, Band 7, Waxmann Verlag, Münster, 1999.

[Kemp94] A. Kemper, A. Eickler: *Datenbanksysteme: Eine Einführung*. Oldenbourg-Verlag. 1996.

[Kemp96] A. Kemper, G. Moerkotte: *Object-Oriented Database Management: Applications in Engineering and Computer Science*. Englewood Cliffs, NJ: Prentice-Hall. 1994.

[Ker98] M. Kerres: *Multimediale und telemediale Lernumgebungen. Konzeption und Entwicklung*. Oldenbourg Verlag, München, Wien, 1998.

[Kin99] M. Kindt (Hrsg.): *Projektevaluation in der Lehre Multmedia an Hochschulen zeigt Profil(e)*. Medien der Wissenschaft, Band 7, Waxmann Verlag, Münster, 1999.

[Kla98] P. Klau: Das Internet. 3. Auflage, Thomson Publishing GmbH, Bonn, 1998.

[Kla99] C. Klas: *Ein neuer, effektiver Ansatz zur Kategorisierung von Web Dokumenten*. In: Heuer, A. (ed.). Proceedings ADI'99 (Agenten - Datenbanken - Information Retrieval), 1999.

[Klu00] N. Klußmann: *Lexikon der Kommunikations- und Informationstechnik: Telekommunikation, Datenkommunikation, Multimedia, Internet*. 2., erw. und aktualisierte Aufl., Heidelberg: Hüthig, 2000.

[Kör99] H. Körndle, S. Narciss: *Studierplatz 2000*. In: K. Lehmann (Hrsg.): Studieren 2000. Medien der Wissenschaft, Band 8, Waxmann Verlag, Münster, 1999.

[Kol98] R. Kolb, M. Uesbeck, M. Skalej: *NeuroAssistant - ein computergestützter neuroradiologischer Arbeitsplatz*. TELEMED '98. Telematik im Gesundheitswesen. Berlin,1998.

[Kra00] J. Krause: *MS Active Server Pages*. Addison-Wesley, München, 2000.

[Lal00] M. Lalmas, T. Rölleke, F. Turra, N. Fuhr: *Concepts for a Graphical User Interface for Hypermedia Retrieval*. In: Proc. 4th International Conference on Flexible Query Answering Systems (FQAS) , Warschau, 2000.

[Lan93] F.W. Lancaster, A.J. Warner: *Information Retrieval Today.* Arlington, VA: Information Resources Press, 1993.

[Leh97] T. Lehmann, W. Oberschelp, E. Pelikan, R. Repges: *Bildverarbeitung für die Medizin*. Springer Verlag Berlin, 1997.

[Leh00] T. Lehmann, B. Wein, J. Dahmen, J. Bredno, F. Vogelsang, M. Kohnen: *Content-Based Image Retrieval in Medical Applications: A Novel Multi-Step Approach*. In: Electronic Imaging 2000. IS&T/SPIE' s 12th Annual International Symposium, San Jose, 2000.

[Leu97] D. Leutner: *Adaptivität und Adaptierbarkeit multimedialer Lehr- und Informationssysteme*. In: L.J. Issing, P. Klimsa (Hrsg.): Information und Lernen mit Multimedia. Weinheim: Pschychologische Verlags Union, 1997.

[Lev00] B. Levienaise-Obadia, W. Christmas, J. Kittler, K. Messer, Y. Yusoff: *OVID: Towards Object-based VIDeo retrieval*. In: Storage and Retrieval for Video and Image Databases VIII (part of the SPIE/IT&T Symposium: Electronic Imaging '2000), San Jose(USA), 2000.

[Lin97] R. Lienhart, S. Pfeiffer, W. Effelsberg: *Video Abstracting*. Communications of the ACM, Vol. 40, No. 12, pp.55-62, Dezember 1997.

[Lin98] R. Lienhart, W. Effelsberg: *Automatic Text Segmentation and Text Recognition for Video Indexing*. Technical Report TR-98-009, Praktische Informatik IV, Universität Mannheim, Mai 1998.

[Lip97] P. Lipson, E. Grimson, P. Sinha: *Configuration based scene classification and image indexing*. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1997.

[Loe00] H. Loeser: *Effektiver Informationsaustausch durch ORDBS-basiertes Web Content Management*. CAD2000, Berlin, März, 2000.

[Luf97a] A. Luft, M. Uesbeck, M. Skalej, R. Kolb, A.K. Wakhloo, U. Drews, K. Voigt: *NeuroAssistant: Different user groups access knowledge collected by multiple authors in the fields of theoretical and clinical neuroscience*. Society for Neuroscience. New Orleans, 1997.

[Luf97b] A. Luft, C. Keller, A.K. Wakhloo, M. Skalej, M. Uesbeck, L Nelson Hopkins, K. Voigt: *DIRectNET: An Internet-Based Forum for the Exchange of Knowledge in Neuroendovascular Surgery*. Tagung der American Association of Neurological Surgeons. Philadelphia, 1998.

[Lum99] J. Lumbley: *The Informix DBA Survival Guide*. 2. Auflage. Prentice Hall Int., 1999.

[Lus92] M. Lusti : *Intelligente tutorielle Systeme. Eine Einführung in wissensbasierte Lernsysteme*. Band 15.4 des Handbuchs der Informatik, Oldenbourg Verlag, München, 1992.

[Lut89] R. Lutz: *Chart Parsing of Flowgraphs*. In: Proc. of the 11th Int. Joint Conf. on AI (IJCAI), 1989.

[Maes94] P. Maes: *Agents that reduce work and information overload*. In: Communications of the ACM (37) 7. ACM-Press, New York, 1994.

[Man95] H. Mandl, H. Gruber, A. Renkl: *Situiertes Lernen in multimedialen Lernumgebungen*. In L.J. Issing, P. Klimsa (Hrsg.): Information und Lernen mit Multimedia. Ein Lehrbuch zur Multimedia-Didaktik und - Anwendung, Beltz Psychologie Verlags Union, 1995.

[Mar99] A. Martens, J. Bernauer, T. Illmann, C. Scheuerer, A. Seitz, M. Weber: *Docs 'n Drugs - ein webbasiertes, multimediales Lehrsystem für die Medizin*. 4. Workshop der AG CBT in der Medizin der GMDS, Heidelberg, April, 1999.

[Maz00] M. Mazlakowski, T. Butcher: *Sams Teach Yourself MySQL in 21 days*. SAMS Verlag, 2000.

[Mar97] M. D. Marsicoi, L. Cinque, S. Levialdi: *Indexing Pictorial Documents by their Content: A Survey of Current Techniques*. In: Image and Vision Computing 15, 1997.

[Meh97] S. Mehrotra, Y. Rui, M. Ortega-Binderberger, T.S. Huang: *Supporting Content-Based Queries over Images in MARS*. In: IEEE Int. Conf. on Multimedia Computing and Systems, Canada, Juni, 1997.

[Mei00] A. Meier, T. Wüst: *Objektorientierte und objektrelationale Datenbanken. Ein Kompaß für die Praxis*. 2. Auflage, dpunkt-Verlag, 2000.

[Mel00] M. Melucci, N. Orio: *SMILE: A System for Content-based Musical Information Retrieval Environments*. RIAO 2000: Computer-assisted information retrieval, France, 2000.

[Met94] E.S. Metcalfe, M. E. Frisse, S. W. Hassan, J. L. Schnase: *Academic Networks: Mosaic and the World Wide Web.* Academic Medicine, 69, S. 270-273, 1994.

[Mil97] M. Milas: *Ein verteiltes Hypermedia-Autorensystem mit WWW-Schnittstelle auf der Basis einer objektorientierten Datenbank.* Diplomarbeit, Lehrstuhl für Graphisch-Interaktive Systeme, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 1997.

[Mit97] M. Mitschke: *Flexible content-based image retrieval systems*. Diplomarbeit im Fach Informatik, Lehrstuhl für Mustererkennung, Universität Tübingen, 1997.

[Mor01] H. Morgenstern: *Ein Verfahren zum automatischen User-Monitoring bei Datenbank-basierten Websystemen zur Unterstützung individualisierter Lernvorgänge*. Diplomarbeit, WSI/GRIS, Universität Tübingen; erscheint im Januar 2001.

[Nas97] C. Nastar, B. Mghaddam, A. Pentland: *Flexible images: Matching and recognition using learned deformations*. Computer Vision and Image Understanding. 65(2), 1997.

[Nib93] W. Niblack, R. Barber, W. Equitz, M. Flickner: *The QBIC Project: Querying Images by Content using Color, Texture and Shape*. In: Proc. SPIE, Volume 1908, 1993.

[Nie83] H. Niemann: *Klassifikation von Mustern*. Springer, Heidelberg, 1983.

[Nie90] H. Niemann: *Pattern Analysis and Understanding*. Springer, Heidelberg, 1990.

[Ng98] K. Ng: *Towards Robust Methods for Spoken Document Retrieval*. Proc. of Int. Conf. on Spoken Language Processing, 1998.

[$O_2$93] $O2_2$ Technology: *The O_2 User Manual*. Release 4.4, 1993.

[Obj97] ObjectDesign Inc.: *ObjectStore Technical Overview*. 1997.

[Obj98a] ObjectDesign Inc.: *ObjectStore Management*. Release 5.1, 1998.

[Obj98b] ObjectDesign Inc.: *ObjectStore API User Guide*. Release 5.1, 1998.

[Op98] R. Oppermann: *Anpassungsfähigkeit und Anpassbarkeit vonLernsystemen*. In: Computer World, Schweizer Wochenzeitung, Juli Ausgabe, 1998.

[Pao89] M. L. Pao: *Concepts of Information Retrieval*. Englewood, CO: Libraries Unlimited, 1989.

[Pau97] A. Pauk: *Technologie und Anwendung Intelligenter Agenten als Mittler in Elektronischen Märkten*. Diplomarbeit in der Fachgruppe Informationswissenschaft der Fakultät Verwaltungswissenschaft, Universität Konstanz, 1997.

[Pea96] S. Pearson: *An Overview of Agent Technology*. HP Technical Paper HPL-96-40. Hewlett-Packard Company, Palo Alto, CA, USA, 1996.

[Pen94] A. Pentland, R. Picard, S. Scarloff: *Tools for Content-Based Manipulation of Image Databases*. In: Proc. SPIE Storage and Retrieval of Image & Video Databases II, Volume 2185, 1994.

[Pet97] D. Petkovic: *INFORMIX Universal Server. Das objekt-relationale Datenbanksystem*. Addison-Wesley Longman, München, 1997.

[Peu98] E.T. Peuker, T.J. Filler, J. Jerosch, W. Held: *Möglichkeiten des Multimedialen Online Teachings (MOT) in der ärztlichen Weiterbildung.* Zentralbibl. Gynäkologie 120: 471-473; 1998.

[Poe96] POET Software Corporaton: *POET - The Object Database*. C++ Programmers Guide, 1996.

[Poe99] E. Poetzsch: *Information Retrieval : Einführung in Grundlagen und Methoden*. 1. Aufl., Potsdam : Verl. für Berlin-Brandenburg, 1998.

[Pri99] M. Prinz, T. Lorang, M. Gengler, E. Schuster: *Ein verteiltes Bilddatenbank- und Bildverarbeitungssystem für medizinische Bilder.* Bildverarbeitung für die Medizin 2000. Heidelberg, 1999.

[Rei99] B. Reinhardt: *Didaktische Strategien in generierten Trainingssystemen zum diagnostischen Problemlösen*. Dissertation, Fakultät für Mathematik und Informatik, Universität Würzburg, 1999.

[Rod89] M. Rode, J. Poirot: *Authoring Systems - Are they used?* In: Journal of Research on Computing in Education. Vol. 21, S. 191-198, 1989.

[Rod95] R.P. Rodgers: *Automated retrieval from multiple disparate information sources: the World Wide Web and the NLM's Sourcerer project.* Journal Am Soc Info Sci, 1995.

[Rub97] Y. Rubner, L. Guibas, C. Tomasi: *The Earth Movers Distance, Multi-Dimension Scaling, and Color-Based Image Retrieval.* In: Proceedings of the ARPA Image Understanding Workshop, 1997.

[Rub98] Y. Rubner, C. Tomasi, L. Guibas: *A Metric for Distributions with Applications to Image Databases*. In: Proc. of the IEEE International Conference on Computer Vision, 1998.

[Saa97] G. Saake, I. Schmitt, C. Türker: *Objektdatenbanken. Konzepte, Sprachen, Architekturen*. MITP-Verlag, 1997.

[Sal87] G. Salton, M.J. McGill: *Information Retrieval; Grundlegendes Für Informationswissenschaftler*. Mc Graw-Hill, 1987.

[San95] W. Sander-Beuermann: *Schatzsucher*. In: „Suchen im Web.". c't 13/98.

[Sau92] H. Sauer: *Relationale Datenbanken*. Addison-Wesley, 1992.

[Say99] A. Sayegh: *CORBA : Standard, Spezifikation, Entwicklung*. 2. Auflage. O'Reilly, Köln. 1999.

[Sca94] B. Scassellati, S. Alexopoulos, M. Flickner: *Retrieving Images by 2-D Shape: A Comparison of Computation Methods with Human Perceptual Judgemetns*. Proc. SPIE, Vol. 2185, 1994.

[Sca97] S. Scarloff, L. Taycher, M. La Cascia: *Imagerover: A content-based image browser for the world wide web*. In: European Conference on Computer Vision, Vol. 1, April, 1996.

[Sch99a] M. Schaller, M. Uesbeck, W. Penkaitis, G. Lindenthal: *MURMEL - Multimediales Ausbildungssystem für die Medizinische Lehre*. 4. Tübinger Expertentreffen: Innovationen und Trends des Medizinstudiums im klininschen Teil. Tübingen, 1999.

[Sch96] C. Schmid, R. Mohr: *Combining greyvalue invariants with local constraints fo robject recognition*. In Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. USA, 1996.

[Sch00] M. Schuchmann, W. Sanns: *Datenmanagement mit MS Access. Für Access 7.0/97/2000. Managementwissen für Studium und Praxis. 2000*.Oldenbourg Verlag, 2000.

[Sch97] R. Schulmeister: *Grundlagen hypermedialer Lernsysteme. Theorie - Didaktik - Design*. 2. Auflage, Oldenbourg Verlag. 1997.

[Sch99] R. Schulmeister: *Medien und Hochschuldidaktik: Welchen Beitrag können neue Medien zur Studienreform leisten?* In: M. Hauff (Hrsg.): media@uni-multi.media. Medien der Wissenschaft, Band 6, Waxmann Verlag, Münster, 1998.

[Sch99b] S. Schulz, R. Klar, T. Auhuber, U. Schrader, A. Koop, R. Kreutz, R. Oppermann, H. Simm: *Quality criteria for electronic publications in medicine.* In: Stud. Health Technol. Inform. 51, 217-26, 1998.

[Sei99] A. Seitz, A. Martens, J. Bernauer, C. Scheuerer, J. Thomsen: *An architecture for intelligent support of authoring and tutoring in multimedia learning environments*. World Conf. on Educational Multimedia and Hypermedia (ED-MEDIA & ED-TELECOM 98), Seattle, 1999.

[Shy99] C. Shyu, C. Brodley, A. Kak, A. Kosaka, A. Aisen, L. Broderick: *ASSERT: A Physician-in-the-Loop Content-Based Retrieval System for HRCT Image Databases.* In: Computer Vision and Image Understanding 75(1/2), 111-132, 1999.

[Sim99] H. Simon: *Anforderungen an die Medienzentren als Kompetenzzentren für Multimedia-Produktion*. In: K. Lehmann (Hrsg.): Studieren 2000. Medien der Wissenschaft, Band 8, Waxmann Verlag, Münster, 1999.

[Sin99] R. Singer, J. Riedel, M. Haag, F.J. Leven: *CAMPUS: Ein WBT-System für die Ausbildung und Entscheidungsunterstützung in der Medizin*. Proc. zum 4. Workshop der AG CBT in der Medizin der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie. Heidelberg, 1999.

[Ska97] M. Skalej, A. Luft, M. Uesbeck, T. Grunert, B. Kortmann, M. Milas, G. Erb, S. Schmid, R. Kolb, D. Welte, C. Keller: *NeuroAssistant - ein computerunterstützter Arbeitsplatz für die Neuroradiologie*. Jahrestagung der Dt. Gesellschaft für Neuroradiologie. Berlin, 1997.

[Smi96a] J.R. Smith, S.F. Chang: *Tools and Techniques for Color Image Retrieval.* In: Proc. SPIE, Volume 2670, 1996.

[Smi96b] J.R. Smith, S.F. Chang: *VisualSeek: A Fully Automated Content-Based Image Query System*. In: ACM Multimedia 96, Boston, November, 1996. In: Proc. SPIE, Volume 2670, 1996.

[Smi96c] J.R. Smith, S.F. Chang: *Searching for Images and Videos on the World-Wide Web*. In: IEEE Multimedia Magazine, Summer, 1997.

[Smi98] J.R. Smith, R. Mohan, C.S. Li: *Content-based Transcoding of Images in the Internet.* In: IEEE Inter. Conf. On Image Proc. (ICIP-98), Oktober, 1998.

[Smi99] J.R. Smith: *VideoZoom spatial-temporal video browse*. IEEE Transactions on Multimedia, Vol. 1, No. 2, June, 1999.

[Sof97] Soffer A: *Image Categorization Using Texture Features.* 4th Int. Conf. on Document Analysis and Recognition, Ulm, 1997.

[Sta91] J. L. Staud: *Online-Datenbanken. Aufbau, Struktur, Abfragen.* 1. Aufl Addison-Wesley, 1991. Rand McNally & Company, Chicago, 1965.

[Ste00] J. Steurer: *Auf der richtigen Seite des Lernens*. In: Computer World, Schweizer Wochenzeitung, Oktober Ausgabe, 2000.

[Ste01] R. Steyer: *Java 2*. New Reference. Engl. Edition. Prentice Hall Int., 2001.

[Stoe97] S. Stoev: Aktionsbasierte Auswertung unsicheren Wissens zur Benutzer- und Aufgabenmodellierung in graphisch-interaktiven Anwendungssystemen. Diplomarbeit, WSI/GRIS, Universität Tübingen, 1997.

[Sto95] C. Stoll: *Slicon Snake Oil*. Doubleday, New York, 1995.

[Sto65] L. M. Stolurow: *Model the master teacher or master the teaching model*. In J. Drumboltz (Hrsg.): Learning and the educational process, 1965.

[Sto99] M. Stonebraker, D. Moore: *Objektrelationale Datenbanken. Die nächste große Welle*. Hanser-Verlag, 1999.

[Sub96] V.S. Subrahmanian: *Multimedia database systems : issues and research directions*. Springer; Berlin; Heidelberg, 1996.

[Sub98] V.S. Subrahmanian: *Principles of Multimedia Database Systems*. Morgan Kaufmann Publishers, 1998.

[Swa91] M.J. Swain, D.H. Ballard: *Color indexing*. Int. Journal of Computer Vision, 7 (1): 11-32, 1991.

[Thi97] F. Thissen: *Das Lernen neu erfinden. Grundlagen einer konstruktivistischen Multimedia-Didaktik*. In: Uwe Beck, Windfried Sommmer (Hrsg.): Tagungsband zur LearnTec97. Karlsruhe, 1997.

[Tho89] D. Thomé: *Kriterien zur Bewertung von Lernsoftware*. Huething Verlag, Heidelberg, 1989.

[Ues97a] M. Uesbeck, M. Milas, M. Skalej, R. Kolb, A. Luft, T. Grunert, A.K. Wakhloo, U. Drews: *Eine medizinische Datenbank mit bidirektionalem Zugriff übers WWW.* In : Lemke, H.U., Vaannier, M.W., Inamura, K. (Hrsg): Computer Assisted Radiology and Surgery (CAR97): S.158. Berlin, Juni 1997.

[Ues97b] M. Uesbeck, M. Milas, M. Skalej, R. Kolb, A. Luft, T. Grunert, A.K. Wakhloo, U. Drews: *Eine medizinische Datenbank mit bidirektionalem Zugriff übers WWW*. gmds Workshop „Medizin und Internet“. München, 1997.

[Ues98a] M. Uesbeck, J. Zimmermann, B. Kortmann, G. Erb, M. Skalej, R. Kolb, G. Schriek, C. Keller, C.: *NeuroAssistant - A Prototype of a Medical Information And Education System Via WWW*. Demosession Proc. of Int.Conference on Extending Database Technology (EDBT’98). Valencia, 1998.

[Ues98b] M. Uesbeck, G. Erb, B. Kortmann, M. Skalej, R. Kolb, C. Keller, G. Schriek: *NeuroAssistant - A Computer Assisted Medical Information And Education System Via WWW*. World Conf. on Educational Multimedia and Hypermedia (ED-MEDIA & ED-TELECOM 98). Freiburg, 1998.

[Ues99a] M. Uesbeck, B. Wursthorn, B. Kortmann, M. Strayle-Batra, M. Skalej: *„Virtual Patients“ on the Web: development, usage and future*. Proc. zum 4. Workshop der AG CBT in der Medizin der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie. Heidelberg, 1999.

[Ues99b] M. Uesbeck, B. Wursthorn, B. Kortmann, M. Strayle-Batra, M. Skalej: *A new approach to a computer based clinical training of medical students in Neuroradiology*. In : Lemke, H.U., Vaannier, M.W., Inamura, K. (Hrsg): Computer Assisted Radiology and Surgery (CARS99). Paris, 1999.

[Ues99c] M. Uesbeck, M. Skalej, B. Kortmann, M. Strayle-Batra, B. Wursthorn: *New approaches for a world-wide usage and preparation of Virtual Patients*. ISCB-GMDS 99. 20th Annual Conference of the Int. Society for Clinical Biostatitics, 44th Annual Conference of the German Society for Medical Informatics, Biometry and Epidemiology, Heidelberg, 1999.

[Ues99d] M. Uesbeck: *MURMEL - Multimediales Ausbildungssystem für die Medizinische Lehre*. In: H. Krahn, J. Wedekind (Hrsg): Medien in der Wissenschaft, Bd. 9: Virtueller Campus ’99, Waxmann-Verlag, Münster, New York, 1999.

[Ues00a] M. Uesbeck, G. Lindenthal, M. Skalej: *Ein hybrider Ansatz für Image Retrieval in medizinischen WBT-Systemen*. In: A. Horsch, T. Lehmann (Hrsg): Informatik aktuell: Bildverarbeitung für die Medizin 2000; München, 2000.

[Ues00b] M. Uesbeck, G. Lindenthal, M. Schaller, M. Riedel, G. Pfister, M. Skalej: *Ein Kompaktsystem zur Generierung Web-basierter Trainings-Systeme für die Aus- und Weiterbildung*. Proc. zum Trier Symposium „Virtuelle Hochschule“, Institut für Telematik, Trier, 2000.

[Ues00c] M. Uesbeck, B. Kortmann, I. Schober-Melms, B. Gerstein-Erb, T. Quenzer, M. Skalej, W. Straßer: *MURMEL - Multimediales Ausbildungssystem für die Medizinische Lehre*. In: Telemedizinführer Deutschland, Ausgabe 2001, Hrsg.: A. Jäckel, Deutsches Medizin Forum, ISBN: 3-0000-4589-9, 2000.

[Ver92] Versant Object Technology: *Versant Version 1.7*. Language Interfaces Manual, 1992.

[Vol99] S. Volmer: *Content Based Image Retrieval on the Web. An Innovative Solution for Image Database*. In: Information Technology, S.15, 1999.

[Vos99] G. Vossen: *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme.* 3. Auflage, München, Wien Oldenbourg Verlag, 1999.

[W3C00] W3C Consortium: *Techniques for Authoring Tool Accessibility Guidelines*. http://www.w3.org/WAI/AU/WD-ATAG10-TECHS-20000918/. In Progress. September, 2000.

[Wed81] H. Wedekind: *Datenbanksysteme I*. K.H. Böhling, U. Kulisch, H. Maurer (Hrsg.). Wissenschaftsverlag, Bibliographisches Institut, 1981.

[Wei96] F. Weiler, F. Vogelsang: *Automatische Erkennung von Bildinhalten bei Standardröntgenaufnahmen.* In: Aachener Workshop 1996: Bildverarbeitung für die Medizin. Aachen, 1996.

[Wei97] P. Weierich, H. Niemann, M. Uesbeck, R. Graumann: *Fully automated image registration of DSA images with combined small markers*. In : Lemke, H.U., Vaannier, M.W., Inamura, K. (Hrsg): Computer Assisted Radiology and Surgery (CAR97), Berlin, 1997.

[Wil93] P. Willett: *Best Match Retrieval*. Library & Information Briefings, Vol. 49, London. 1993.

[Wis98] Wissenschaftsrat: *Empfehlungen zur Hochschulentwicklung durch Multimedia in Studium und Lehre*, Drs. 3536/98, Eigenverlag, 1998.

[Wis00] Deutscher Wissenschaftsrat: *Thesen zur künftigen Entwicklung des Wissenschaftssystems in Deutschland*, Drs. 4594/00, Eigenverlag,Bonn, Juli 2000.

[Wit97] M.J. Witbrock, A.G. Hauptmann: *Speech Recognition and Information Retrieval: Experiments in Retrieving Spoken Documents*. In: Proc. of the DARPA Speech Recognition Workshop, 1997.

[Wur98] B. Wursthorn: *Didaktische Konzeption und Implementierung eines Web-basierten, interaktiven und dynamischen Lernprogramms zur fallorientierten Simulation klinischer Handlungsabläufe*. Diplomarbeit, WSI/GRIS, Universität Tübingen, 1998.

[Woe] Wörterbuch der Psychologie, dtv, 1987.

[Woo95] M. J. Woolridge, N. R. Jennings: *Intelligent Agents: Theory and Practice*. Knowledge Engineering Review 10 (2). Cambridge University Press, 1995.

[You97] S.J. Young, M.G. Brown, J.T. Foote, G.J.F. Jones, K. Spärck Jones: *Acoustic indexing for multimedia retrieval and browsing*. Proc. ICASSP-97, Vol. 1, 1997.

[Zha95] H.J. Zhang, Y. Gong, C.Y. Low, S.W. Smoliar: *Image Retrieval based on Color Features: An Evaluation Study*. In: Proc. SPIE, Vol. 3606, 1995.

[bright] http://www.completeplanet.com/Tutorials/DeepWeb/contents04.asp

[dbmvgl] http://www.postgresql.org/mhonarc/pgsql-general/1999-11/msg00227.html

[direct] http://www.directhit.com

[google] http://www.google.com

[iee00] http://www.itl.nist.gov/div897/ctg/graphics/ieeeltsc.htm

[ims00] http://imsproject.org/

[ostore] http://www.odi.com/htm/object_prod.htm

[oingo] http://www.oingo.com

[postgr] http://www.de.postgresql.org/

[robots] The Web Robots Pages: http://info.webcrawler.com/mak/projects/robots/active.html

[smile] http://livia.dei.unipd.it/smile/

[speech] http://speechbot.research.compaq.com/

Lebens- und Bildungsgang

22. Mai 1967	geboren in Ahaus/Westf.
1973 - 1977	Sebastian Grundschule, Rosendahl.
1977 - 1986	Heriburg Gymnasium, Coesfeld; Abschluß: Abitur.
1986 - 1989	Ausbildung zur Mathematisch-Technischen Assistentin; Bayer AG, Leverkusen.
1989 - 1995	Studium der Informatik mit Nebenfach Medizin an der Friedrich-Alexander-Universität Erlangen.
1990	Werkstudentin; Bayer AG, Leverkusen.
1993 - 1995	Wissenschaftliche Hilfskraft; Bayer. Forschungszentrum für Wissensbasierte Systeme (FORWISS), Erlangen.
1994	Werkstudentin; Siemens AG - Medizintechnik, Erlangen.
1995	Diplomarbeit; Siemens AG - Medizintechnik, Erlangen.
seit 1996	Wissenschaftliche Mitarbeiterin am Lehrstuhl für Graphisch Interaktive Systeme am Wilhelm-Schickard-Institut für Informatik der Eberhard-Karls-Universität Tübingen (Prof. Dr.-Ing. Dr.-Ing. E.h. W. Straßer) und in der Abteilung für Neuroradiologie des Universitätsklinikums Tübingen (Prof. Dr. med. K. Voigt).