T U M

I N S T I T U T F Ü R I N F O R M A T I K

MultiMAP/2: Netzzugang und Netzbetrieb für das multimediale Datenbanksystem MultiMAP

Günther Specht, Martin Zirkel

TUM-I9920
Juni 99

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-06-I9920-85/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

MultiMAP/2
Netzzugang und Netzbetrieb für das multimediale Datenbanksystem MultiMAP

DFN-Projekt

1. Dezember 1996 - 30. November 1998

Dr. Günther Specht und Martin Zirkel

Institut für Informatik
Technische Universität München
Orleansstr. 34,
D-81667 München
Email: {specht,zirkel}@in.tum.de

Inhaltsverzeichnis

1 Einleitung und Kurzfassung

MultiMAP ist ein seit 1993 an der TU München entwickeltes, interaktives, multimediales Datenbanksystem, in dem Texte, Bilder, Audios und Videos einschließlich beliebiger Links zwischen Teilobjekten dieser Objekte mittels eines wesentlich mächtigeren Linkkonzepts als im gegenwärtigen WWW (1:n-Links, bidirektionale Links, verschachtelte Links, referentielle Integrität der Links, etc.) gespeichert werden können.

Im Rahmen des DFN-Entwicklungsprogramms “Fortgeschrittene Anwendungen” wurde für den Zeitraum vom 1. Dezember 1996 bis 30. November 1998 das Projekt “Netzzugang und Netzbetrieb für das multimediale Datenbanksystem MultiMAP” (Arbeitskürzel: MultiMAP/2) an der TU München unter der Leitung von Dr. Günther Specht durchgeführt. Das Projekt war dem DFN-Arbeitsbereich “Multimedia-Teledienste” zugeordnet. Dieser Abschlußbericht faßt die Ergebnisse aller acht Projektquartale zusammen und reicht von der Analyse der verschiedenen Techniken zur Datenbankanbindung an das WWW über die Spezifikation und Implementation der MultiMAP-Architektur bis hin zur Evaluation und dem Tuning. Zusätzlich enthält er die Benutzermanuale der darauf entstandenen Anwendungssysteme MultiLIB und MuliBHT um einen ersten Einblick in die Systeme zu ermöglichen.

MultiMAP war vor Projektstart nur lokal aufrufbar. Ziel des Projekts war es, MultiMAP so zu erweitern, daß die volle Funktionalität auch über das WWW im Netz zur Verfügung steht, um so die Einsatzmöglichkeiten für das System zu erweitern und es bei zwei Referenzanwendergruppen zum Einsatz zu bringen und zu evaluieren. Dabei stellten sich aufgrund der hohen interaktiven Multimedia-Funktionalität des Systems eine Reihe von komplexen Problemstellungen sowohl an die Systemarchitektur (Client/Server-Ketten-Verteilung) als auch an den Netzzugang und die Netzübertragung. Die beiden Anwendungsprojekte waren:

MultiLIB:
Ein multimedialer Führer durch die Universitätsbibliothek der Ludwig-Maximilians-Universität München (LMU) einschließlich ihrer Außenstellen und Zweigbibliotheken. Ziel war es, den Studenten ein schnelles Auffinden der Standorte der Bücher, der Zugangsberechtigungen und Öffnungszeiten und eine Unterstützung bei den verschiedenen Katalogrecherchen zu ermöglichen. Hierzu mußte das System von in ganz München verteilten Standorten aufrufbar sein. Das System wurde in Zusammenarbeit mit der Universitätsbibliothek München aufgebaut.

MultiBHT:
Das zweite Anwendungsgebiet lag in der multimedialen Aufbereitung von Ergebnissen der Sprachanalyse bei Linguisten mit dem Ziel der Erstellung korrekter Grammatiken und der Erstellung textkritischer Editionen. Eine Anwendung für Althebräisch läuft im Institut für Assyriologie und Hethitologie der LMU München und wird von ganz

Deutschland aus genutzt, da es derzeit die weltweit einzige vollständige, satzbezogene Konkordanz für Althebräisch (und damit insbesondere für das Alte Testament) darstellt.

Die Datenbanken wurden von den Projektpartnern bereits während der Projektlaufzeit mit Anwendungsdaten gefüllt, die Schnittstellen und Oberflächen mehrmals evaluiert und umgeschrieben. Ein umfangreiches Tuning sowohl auf der Datenbankseite als auch am Java-Client sowie an den Netzübertragungsfunktionen wurden durchgeführt. Die Systeme stehen jetzt den Partnern in einem optimierten Zustand zur Verfügung und werden eingesetzt.

Die verschiedenen Versionen von MultiMAP, die während der Förderphase entstanden, dokumentieren den jeweils aktuellen Stand der Entwicklung von Datenbankanschlüssen ans Netz: Zuerst wurde ein einfacher Java-Client (in JDK 1.0) verwendet, später eine Kopplung über einen “middle tier” realisiert, der die Synchronisation und Transaktion wiederherstellte (in JDK 1.1), schließlich kamen JDBC-Kopplungen mit einer “three tier architecture” zum Einsatz. Auch verschiedene Datenbanksysteme (TransBase, DB/2, etc.) wurden eingebaut und bezüglich ihrer JDBC-Fähigkeit getestet. Dadurch konnte die in der Forschung jeweils aktuelle Anbindungstechnik praktisch erprobt und evaluiert werden. Zusätzlich entwickelten wir einen eigenen Anbindungsvorschlag.

Darüberhinaus entstanden interessante Erweiterungen in MultiMAP wie

- Personalisierung (durch das von uns entwickelte GRAS-Verfahren, das sich auf Gauß-Kurven zur Berechnung der Signifikanz und der Relevanz eines gefundenen Objekts für den jeweiligen Benutzter stützt)
- Annotationen (Simulation der persönlichen gelben Klebezettel)
- Volltextsuche
- Verschachtelte und überlappende Links (genauer: Quellanker) in Texten und Bildern.

Die erste und die letzte Entwicklung wurde auch auf wissenschaftlichen Tagungen (HIM’97 und INET’98) vorgestellt.

Neben den beiden Referenzanwendungen wurden auch eine Reihe weiterer Anwendungen wie MultiMAP und MultiMED entwickelt:

MultiMAP:
Eine multimediale Aufbereitung von Landkarten und Stadtplänen mit Fotos, Skizzen, Texten, Ton- und Sprachausgabe und rekursiv weiteren Detailkarten. Anwendungen liegen in Stadtinformationssystemen, Biotopkartierungen oder auch im administrativen Bereich für Raumordnungsverfahren. Dabei kann nicht nur volltext- und stichwortunterstützt gesucht werden, sondern insbesondere auch durch Anklicken beliebiger Objekte (Straßen, Gebäude, Grundstücke, Biotope, etc.) auf den Landkarten. Die Landkarten sind als Pixelbilder abgelegt und z.B. bezüglich der Straßenführung mit Vektordaten hinterlegt. Ein Stadtführer von München ist weitgehend fertiggestellt. Eine Anfrage zur

Unterstützung von Raumordnungsverfahren vom Lehrstuhl für Raumforschung, Raumordnung und Landesplanung der TU München liegt vor.

MultiMED:
Ein weiteres Anwendungsgebiet erstreckt sich im medizinischen Bereich auf multimedial aufbereitete Röntgenbilder und CT-Aufnahmen, einschließlich Detailbilder und verbaler oder textueller Befundung. Ein Anschluß an die Stationsdatenbank mit der Patientenstammdatenverwaltung ist integriert. Ein Prototyp wurde in Zusammenarbeit mit der Orthopädischen Abteilung des St. Bernward Krankenhauses in Hildesheim erstellt.

Daneben war das Projekt MultiMAP/2 auch wissenschaftlich äußerst erfolgreich:

Es entstanden im Förderzeitraum (1996-1998):

- 6 wissenschaftliche Veröffentlichungen auf internationalen Tagungen,
- 11 Diplomarbeiten und Studienarbeiten
- 1 Habilitation

Im Förderzeitraum wurden insgesamt

- 10 Vorträge über MultiMAP auf Kongressen und Symposien gehalten, wie z.B. auf der PAS’97 in Aizu, Japan, der HIM’97 in Dortmund, dem DFN-Symposium in Berlin, der ATS’97 in München und der KI’98 in Bremen, etc.
- Darüberhinaus wurden die Systeme auf zahlreichen Messen, Kongressen und Tagungen vorgeführt, wie z.B. auf der ATS’97 und ATS’98 im München, der INET ‘98 in Genf, dem Tag der Informatik’97 in München, etc.

Einschließlich Grundausstattung, bewilligter Stelle, Hiwis und Diplomanden arbeiteten jeweils durchschnittlich 7 Personen am Projekt. Neben dem Leiter Dr. Günther Specht und Hrn. Dipl.-Inform. Martin Zirkel (Mitarbeiterstelle) arbeiteten Fr. Yvonne Zimmer, Fr. Eva Beinhofer, Hr. Theodor Myntidis und Hr. Athanasios Sarakatsanis als studentische Hilfskräfte mit. Hr. Markus Steindl, Hr. Theodor Myntidis, Hr. Thomas Kahabka, Hr. Athanasios Sarakatsanis, Fr. Martina Lindner und Hr. Helmut Lutzenberger schrieben im Rahmen des Projekts ihrer Diplomarbeiten und Fr. Eva Beinhofer, Hr. Eshref Januzaj, Fr. Yvonne Zimmer, Hr. Christian Trennhaus und Hr. Jörg Lanzinger ihre Studienarbeiten (Fopra). Ihnen allen sowie Hr. Dipl.-Inform. Michael Bauer sei für ihr großes Engagement, ihre guten Ergebnisse und die gute Teamarbeit recht herzlich gedankt.

2 Das Vorhaben

2.1 Projektbeschreibung

MultiMAP/2 wurde im Nachfolgeprogramm der *Regionalen Test-Beds* des DFN, im DFN-Entwicklungsprogramm *Fortschrittliche Anwendungen* (1996 - 1998) angesiedelt und gefördert. Die Zielsetzung war einen WWW-Zugang über das Breitband-Wissenschaftsnetz (B-WiN) mit Anschlußbandbreiten bis zu 155 Mb/s zur bisher nur lokal aufrufbaren multimedialen Datenbank MultiMAP zu entwickeln, zu testen und bei zwei Projektpartnern einzusetzen.

Im Rahmen des Vorläuferprogramms *Regionales Test-Bed (RTB) Bayern* (1994-1996) erstellten wir bereits einen Verbund elektronischer Bibliotheken (**OMNIS**).[1] Die Anfragen, die dabei über das Netz gingen, waren einfache Queries oder Volltextrecherchen. Als Antwort darauf werden bibliographische Informationen und Bilder der gefundenen Dokumente übertragen und angezeigt.[2] Neue Anfragen konnten weitgehend unabhängig von vorhergehenden bearbeitet werden.

Ziel des Projekts MultiMAP/2 war dagegen interaktive Multimedia-Datenbanken übers Netz zur Verfügung zu stellen. Dabei werden Objekte in der Bild-Datenbank durch beliebig umrandete Bildbereiche identifiziert. Diese Objekte sind über komplexe Links beliebig miteinander verknüpft. Die Funktionalität des interaktiven Multimedia-Datenbanksystems **MultiMAP** umfaßt:

1. Vielfältige Suchmöglichkeiten: Suche über Linkverfolgung (navigierend), über Objektrecherche und über Volltextrecherche.
2. Ein wesentlich mächtigeres Linkkonzept als HTML (siehe unten)
3. Eine extrem einfache und schnelle Erstellung von Anwendungen.
4. Integrierte und optimierte Ablage und Verwaltung aller Daten, auch der Texte, Bilder, Audios, Videos und Links in der Datenbank, dadurch hohe Effizienz auch bei großen Datenmengen.
5. Multiuser-Betrieb auf derselben Datenbank (auch bei Updates).

Gegenüber dem RTB-Projekt stellten sich dabei wesentlich komplexere Anforderungen:

1. Anfragen sind kontextabhängig vom augenblicklichen Recherchestand und vom Fensterinhalt bzw. Bildinhalt des jeweiligen Benutzers. Transaktionen über mehrere Seiten hinweg und Synchronisationen müssen über das zustandslose WWW hinweg

1. OMNIS im RTB Bayern: http://www.lrz-muenchen.de/projekte/rtbbay/ sowie
2. OMNIS Hauptseite mit eigener Recherchemöglichkeit: http://omnis.in.tum.de/cgi-bin/omnis bzw. mit neuer Oberfläche: http://elektra.informatik.tu-muenchen.de/cgi-bin/dibwin/konto

erhalten bleiben.

2. Aufgrund dessen war ein wesentlich komplexeres Client/Server-Konzept für die Multimedia-Datenbank erforderlich als für das Bibliotheksrecherchesystem.
3. Aufgrund der Interaktivität durch maussensitive Bereiche, Blinken von Ergebnissen, Audioausgaben, etc. wurde auch die Client-Seite einschließlich eines interaktiven Previewers wesentlich komplexer.
4. Für diese interaktive Hypermedia-Datenbank sind größere Datenübertragungsraten notwendig als bei Bibliothekssystemen wie OMNIS. Insbesondere sind zeitkritische und synchrone Datenlieferungen aus der Datenbank notwendig, was im Multiuser-Betrieb besondere Anforderungen an den Dokumentenserver stellt.

Vorarbeiten

Das Projekt setzt auf dem, vom Antragsteller entwickelten, nur lokal aufrufbarem, multimedialem Datenbanksystem MultiMAP/1 auf. In ihm ist es möglich Bilder (Fotos, Skizzen, Landkarten etc.), beliebig lange Texte und Töne abzuspeichern und diese, aber auch beliebige Objekte innerhalb der Bilder, durch Links miteinander zu verknüpfen. Das dazu entwickelte Linkkonzept geht weit über die üblichen WWW-Links hinaus:

1. Unterstützung beliebiger n:m-Links
   Es werden nicht nur 1:1- sondern beliebige n:m-Links unterstützt. Das stellt eine Erweiterung klassischer Hypertextstrukturen dar, die nur n:1-Links verwalten können. Bei 1:n- Links ist eine zusätzliche Auswahlkomponente nötig.
2. Unterstützung bidirektionaler Links
3. Unterstützung beliebiger graphischer Linkanker:
   Linkausgänge und Linkziele können beliebig umrandete Objekte auf einem Bild sein (beispielsweise ein Flußverlauf oder ein beliebiges Grundstück auf einer Landkarte). Insbesondere können sich diese beliebig inkludieren oder überlappen.
4. Unterstützung von Volltext und Objektrecherche:
   Zusätzlich zu den Links kann auch durch Volltextrecherche auf allen Texten, Bild- und Objektnamen in der Datenbank gesucht werden. Die Volltextsuche ist in die Objektrecherche integriert und verhält sich wie zusätzliche dynamische Links.
5. Referentielle Integrität von Links:
   Alle Objekte und Links werden innerhalb der Datenbank verwaltet und genügen daher der referentiellen Integrität und unterstützen Multiuser-Betrieb.

MultiMAP realisiert die Multimedia-Schicht und stützt sich zur internen Datenhaltung auf das relationale Datenbanksystem TransBase™ als Backend ab. In seiner zu Beginn des Projekts eingebrachten Version war MultiMAP nur lokal im LAN (via rlogin, NFS oder Bildschirmumleitung) aufrufbar.

Anwendungen

Das MultMAP/2 System kommt in zwei stark unterschiedlichen Anwendungen zum Einsatz:

In der Anwendung MultiLIB wurde ein multimedialer Führer durch die Universitätsbibliothek und ihre Außenstellen erstellt. Von der Anwendung her handelt es sich in erster Linie um ein Recherchesystem mit klar verteilten Rollen von Ersteller und Nutzer. Wir erwarten sehr viele Nutzer, bei einer gut skalierbaren Größe der zu übertragenden multimedialen Knoten. D.h. es wird zu einer ausgewogenen Netzbelastung kommen.

In der Anwendung MultiBHT geht es um eine interaktive Aufbereitung von Sprachanalyseergebnissen althebräischer Texte, insbesondere der Originaltexte des Alten Testaments, für die Arbeit der Linguisten. Dabei werden ausgehend von der Biblica Hebraica Transcripta (kurz: BHT) auf dem dargestellten Text Links auf die Ergebnisse der morphologischen Wortanalyse und der syntaktischen Analyse der einzelnen Wortgruppen einschließlich aller Mehrdeutigkeiten abgelegt. Eine vollständige computergestützte Analyse der Morphologie und der Morphosyntax wurde am Lehrstuhl für Hebraistik und Ugaristik der LMU München, Prof. Dr. W. Richter, (Projektpartner) in den letzten 10 Jahren durchgeführt und abgeschlossen. Die Ergebnisse lagen bereits am Rechner vor. Zusätzlich sind Verweise auf entsprechende Wörterbuch-Einträge und die zur Analyse verwendeten Grammatikregeln interessant. Auch Links auf die entsprechenden Stellen der verschiedenen Originalschriften, wie z.B. dem Codex St. Petersburg als älteste Bibelhandschrift oder Faksimile der Qumran-Funde, die oft nur Papierschnipsel darstellen und bei denen eine interaktive Zuordnung zu möglichen Paßstellen besonders interessant ist, sowie zu einer Reihe wichtiger Übersetzungen können eingetragen werden.

Das Ziel war ein Grundsystem mit den bisherigen Sprachanalyseergebnissen zu erstellen, das nicht nur zur Recherche sondern allen beteiligten Forschern zur weiteren Ergänzung und Annotation, sowie zur gemeinsamen Weiterarbeit als interaktives Nachschlagewerk und als persönlicher, interaktiver, sehr mächtiger Zettelkasten bereitsteht. Damit wird die Arbeitsweise effektiver und die Ergebnisse werden leichter zugreifbar.

Während die Anwendung MultiLIB sich mehr an ein breites Publikum, insbesondere an Studenten wendet, ist MultiBHT eine Anwendung im Forschungsbereich. Wir haben es daher hier mit einem anderen Nutzerprofil zu tun: Wenige Nutzer mit intensiveren Sitzungen und sehr unterschiedlich großen, zu übertragenden Datenmengen, wobei jeder gleichzeitig Recherchierer und Ersteller ist. Also ist mit Lastspitzen in beiden Richtungen zu rechnen. Während man es im ersten Fall eher mit Gelegenheitsnutzer zu tun hat, sitzen im zweiten Fall Experten an der Anwendung.

Projektziel laut Antragstellung

Ziel ist die Fertigstellung einer WAN-fähigen multimedialen Datenbank, angepaßt an die speziellen Benutzerwünsche der Projektpartner. Aufgrund der geforderten Mächtigkeit der Benutzerschnittstelle müssen dazu geeignete Java-Clients und Datenbank-Anbindungen (DB-Clients) entwickelt werden. Die Datenbanken werden von den Projektpartnern bereits während der Projektlaufzeit mit Anwendungsdaten gefüllt und sollen den Projektpartnern nach Abschluß des Projekts für ihre Arbeiten zur Verfügung stehen. In enger Abstimmung mit den Projektpartnern sollen regelmäßige Reviews stattfinden, um deren Anforderungen und Wünsche frühzeitig berücksichtigen zu können. Eine Anwendungsevaluation und Netzübertragungsevaluation schließen das Projekt ab.
Bei den Antragstellern am Institut für Informatik der TU München, Orleanstr.34, werden lokal die Software-Entwicklung und über Netzzugriffe zu den Anwenderinstallationen Arbeiten zu Test und Wartung durchgeführt. Weiterhin sind folgende Installationen und folgender Netzverkehr geplant:

- Im Anwendungsgebiet MultiLIB wird ein MultiMAP-Server an der TU München installiert, auf den über Netz von der Universitätsbibliothek der LMU sowie aus allen anderen LMU-Standorten, einschließlich den Außenstellen Großhadern und Weihenstephan in Freising, aber auch von allen Interessenten außerhalb zugegriffen wird.
- Im Anwendungsgebiet MultiBHT ist geplant einen Server bei Hr. Dr. Ch. Riepl, Rechnergestützte Forschung, Fakultät für Altertumskunde und Kulturwissenschaften der LMU München, Geschwister-Scholl-Platz 1, zu installieren, auf den von den Universitäten Würzburg, Bamberg, Aachen und Tübingen über das B-WiN zugegriffen wird.

Dabei entsteht ein erheblicher Bedarf an Netzverkehr, sowohl durch die räumliche Verteilung der Benutzer (TU München, LMU München, Weihenstephan, Würzburg, Bamberg, Aachen und Tübingen etc.), als auch durch die zu übertragenden, großen Datenmengen, insbesondere bei zeitkritischen interaktiven Ausgaben innerhalb von Bildern und bei den Audio Ein- und Ausgaben.

Aufgabenkurzbeschreibung

Die wesentlichen Aufgaben dieses Projekts gliedern sich wie folgt:

1. Um das Grundsystem MultiMAP netzwerkfähig zu machen, muß es um eine geeignete Client/Server-Struktur erweitert werden. Die Wahl der Client/Server-Schnittstelle beeinflußt entscheidend die Effizienz und das Verhalten des Systems. Dazu ist die bestehende Architektur von MultiMAP neu zu überarbeiten und partiell neu zu implementieren.

2. Die Aufrufschnittstellen müssen so neu aufbereitet werden, daß sie gegenüber dem zustandslosen Übertragungsprotokoll von HTTP, die (zustandsorientierte) Transaktionsfunktionalität einer multimedialen Datenbank zur Verfügung stellen.
3. Eine Behandlung der Echtzeit- bzw. Synchronisationsproblematik bei Datenbanken für das gleichzeitige Abspielen von vielen Objekten ist notwendig. Insbesondere sind dazu Überlegungen zur Belastung des Datenbankservers nötig.
4. Volle Einbindung ins WWW als Multimedia-Server und Test mit den beiden Applikationen MultiLIB und MultiBHT.

Bandbreitenbedarf und Netzbelastung

Durch die Einbindung von Audio- und Video-Daten entsteht ein Echtzeitbedarf. Zusätzlich entstehen durch die interaktiven Anwendungen kurzzeitig hohe Netzbelastungen, beim Retrieval vor allem von der Server- zur Client-Seite, bei Updates und beim Aufbau neuer Informationsblöcke (Multimediaknoten) auch in der Gegenrichtung. Wir rechnen mit kurzzeitigen Auslastungen bis zu 1 Mbit/sec je Nutzer und zwischen 10 und 30 Nutzern gleichzeitig in der Anwendung MultiLIB. In der Anwendung MultiBHT werden es eher wenige parallele Anwender sein (drei bis vier) aber wesentlich größere, zu übertragene Datenmengen je Richtung, da hier neben der reinen Anfrage auch die Update- und Erfassungsfunktion für jeden Nutzer eine wesentliche Rolle spielt.

2.2 Projektgliederung und Meilensteine

Das Gesamtprojekt gliederte sich in 4 Phasen: Spezifikation, Prototyp mit noch eingeschränkter Funktionalität, Überarbeitung und Erstellung einer Version mit voller Funktionalität sowie eine Evaluationsphase. Parallel dazu wurde in allen Phasen eine intensive Betreuung und Rückkopplung mit den Anwendern gesucht. Meilensteine waren alle 3 Monate vorgesehen.

Phase 1:

1.1. (3 Monate): Analyse und Spezifikation der möglichen Client/Server-Schnittstellen in MultiMAP, insbesondere unter Beachtung von Effizienzkriterien.

1.2. (1 Monat): Studie: Programme die zur Laufzeit im WWW-Client ablaufen, ermöglichen eine hohe Interaktivität. Es soll untersucht werden, welche Sprache und in welcher Kombination (Java, Java-Script, Trusted Applets, JDBC, etc.) bei der Entwicklung der (Recherche/Viewer) Clients eingesetzt werden soll.

1.3. (2 Monate): Bereitstellung und Installation des bestehenden (lokalen) MultiMAP Systems bei den Anwendern (für MultiLIB an der Universitätsbibliothek München

und für MultiBHT an der Stelle für ‘Rechnergestützte Forschung’ der Fakultät 12 der LMU) und Unterstützung der Anwender bei der Erstellung von Applikationen. Dies soll dazu dienen, daß bei der Distribution der netzfähigen Version bereits entsprechende Inhaltsdatenbanken zum Testen bestehen.

1. Meilenstein (28.2.97): Abgabe von 1.1 und Zwischenbericht 1.3.

Phase 2:

2.1. (2 Monate): Entwicklung der Abbildung vom Adressierungsschema des WWW zu Datenbankabfragen.

2.2. (6 Monate): Spezifikation und Implementation der grundlegenden Funktionen als WWW-Prototyp. Der Client wird dabei mit Hilfe existierender Viewer realisiert.

2.3. (2 Monat): Installation des ersten WWW-Prototyps bei den Anwendern und erste Benutzerevaluation bezüglich Oberfläche und Funktionalitätswünsche.

2. Meilenstein (31.5.97): Abgabe 1.2, 1.3 und 2.1.
3. Meilenstein (31.8.97): Zwischenbericht zu 2.2.

Phase 3:

3.1. (3 Monate) Integration des erweiterten Linkkonzepts und der noch fehlenden Interaktionen in den Prototypen, einschließlich evtl. notwendiger Konvertierprogramme für die Anwender. Evtl. Übergang auf der Clientseite zu eigenen, Java-basierten Previewern.

3.2. (3 Monate): Behandlung der Echtzeit- bzw. Synchronisationsproblematik bei Datenbanken für das gleichzeitige Abspielen von vielen Objekten. Insbesondere sind dazu Überlegungen zur Datenbankserverbelastung nötig.

3.3. (2 Monat): Rückkopplung der Benutzerwünsche in die Spezifikation des Systems. Insbesondere Analyse der Wünsche, die nicht nur die Oberfläche des Clients sondern die Funktionalität der Server-Seite des Systems betreffen. Gleichzeitig Leistungsmessungen und Evaluation der Netzlast.

3.4. (6 Monate): Erstellung der endgültigen Fassung einer optimierten Server-Seite sowie Erstellung der endgültigen Fassung der Client-Seite einschließlich der überarbeiteten Komponente für die schnelle Erfassung von Daten auch über das Netz.

4. Meilenstein (30.11.97): Abgabe 2.2. und 2.3.
5. Meilenstein (28. 2.98): Abgabe 3.1. und 3.2.
6. Meilenstein (31. 5.98): Abgabe 3.3. und Spezifikation zu 3.4.

Phase 4:

4.1. (3 Monate): Volle Einbindung ins WWW als Multimedia-Server bei allen Projektpartnern und Test mit mindestens zwei Applikationen (MultiLIB, MultiBHT oder einer multimedial aufbereiteten Vorlesung) mit vielen gleichzeitig aber asynchron arbeitenden Anwendern.

4.2. (2 Monat): Aufgrund der Leistungsmessungen Tuning und evtl. Spezialanpassungen für einzelne Anwendungen.

4.3. (1 Monat): Dokumentation und Publikation der Ergebnisse gemeinsam mit den Projektpartnern.

7. Meilenstein (31. 8.98): Abgabe 3.4. und erster Teil 4.1.
8. Meilenstein (30.11.98): Abgabe 4.1., 4.2. und 4.3.

Das ergibt in der Summe 36 Personenmonate, von denen 24 beim DFN beantragt und 12 als Eigenleistung eingebracht wurden.

Projektübersicht als Balkendiagramm

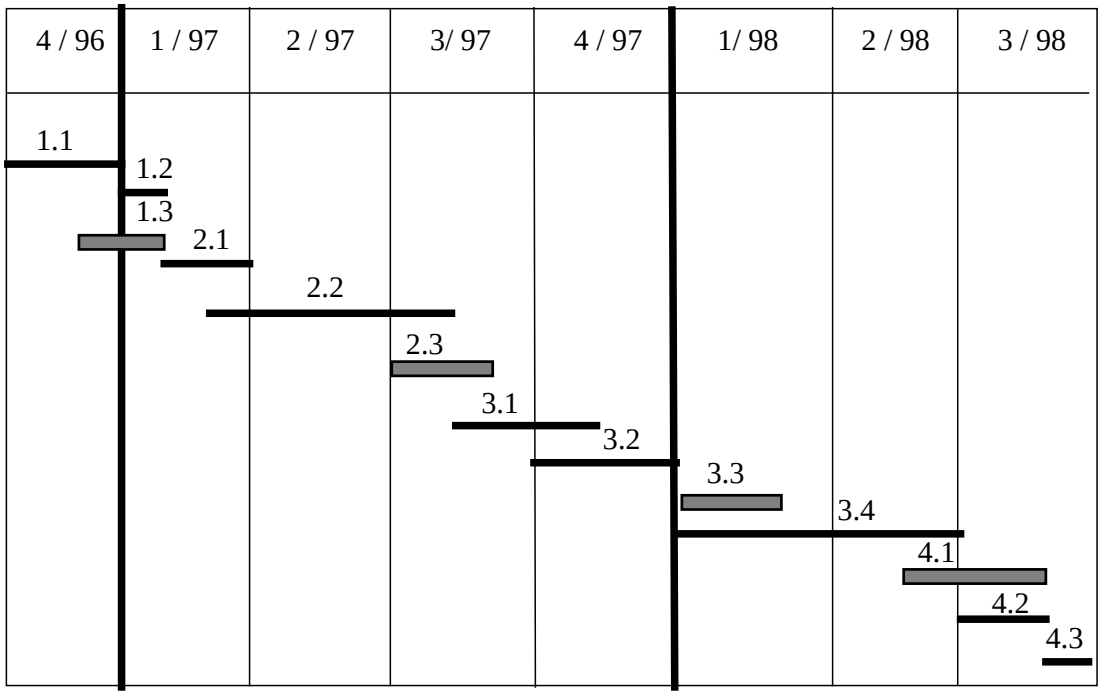

Abb. 1: Projektphasendiagramm

Die Phasen mit Softwaredistributionen an die Benutzer und Evaluationen der Benutzerwünsche sind besonders gekennzeichnet.

3 Entwicklung des Basissystems MultiMAP/2

Im folgenden wird die Entwicklung des Basissystems MultiMAP/2 beschrieben. Alle dabei gesetzten Meilensteine konnten in ihrer Zeit erreicht werden. Die Gliederung dieses Kapitels orientiert sich an der Phasengliederung des Projektes und beschreibt es von der ersten Analyse der verschiedenen Techniken zur Datenbankanbindung an das WWW über die Spezifikation und Implementation der MultiMAP-Architekturen bis hin zur Evaluation und dem Tuning.

3.1 Analyse der Techniken zu Datenbankanbindungen ans WWW

Zur Spezifikation einer geeigneten Architektur für MultiMAP/2 ist zuerst die Analyse, Bewertung und Auswahl der Technik der Datenbankanbindung notwendig. Allgemein können die Techniken zur Anbindung einer Datenbank an das WWW in zwei Kategorien eingeteilt werden:

- Anbindung auf der Client-Seite
- Anbindung auf der Server-Seite

3.1.1 DB-Anbindung auf Server-Seite

Zur Anbindung einer Datenbank auf der Server-Seite wird entweder die CGI-Schnittstelle (**C**ommon **G**ateway **I**nterface) oder die API (**A**pplication **P**rogramming **I**nterface) des HTTP-Servers verwendet. In Java gibt es darüberhinaus die Möglichkeit einer server-seitigen Anbindung mittels Servlets.

3.1.1.1 DB-Anbindung mit CGI

Eine häufig genutzte Anbindungsmethode zur Datenbankanbindung an das WWW ist die CGI-Schnittstelle. Hierbei wird ein CGI-Programm üblicherweise in das Verzeichnis cgi-bin des WWW-Servers abgelegt. Sobald dieses Programm vom WWW-Server aufgerufen wird, stellt es eine Verbindung mit der Datenbank her, führt bestimmte Transaktionen aus, erstellt aus dem Ergebnis eine neue HTML-Seite und gibt sie dem HTTP-Server schließlich als Ergebnis zurück.

Um dem CGI-Programm das für jeden Aufruf nötige Ein- und Ausloggen in die Datenbank zu ersparen, wird zwischen dem CGI-Programm und der Datenbank ein Datenbank-Client geschaltet, der ständig in die Datenbank eingeloggt ist. Dieser Datenbank-Client fungiert gegenüber dem CGI-Programm als Server, der auf Anfragen wartet, diese als SQL-Queries an die Datenbank weitergibt und die Ergebnisse wieder an das CGI-Programm zurücksendet. Die Umwandlung der Anfrageergebnisse in HTML-Code kann sowohl im Datenbank-Client als auch im CGI-Programm erfolgen.

Positiv ist, daß die CGI-Anbindung völlig unabhängig vom verwendeten WWW-Server ist, da CGI eine standardisierte HTTP-Server-Schnittstelle ist und am WWW-Client keine spezielle Software gestartet werden muß.

Nachteilig ist, daß Interaktivität auf dem Client nur sehr schwer zu realisieren ist, da die einzelnen Seiten, die zum Client geschickt werden, dort statisch sind. Ein weiterer Nachteil dieser Vorgehensweise liegt in der Zustands- und Verbindungslosigkeit von HTTP. Der HTTP-Server ist zustandslos, so daß jede Anfrage unabhängig von der vorhergehenden bearbeitet wird und somit Benutzersitzungen, die sich über mehrere Dokumente erstrecken, schwer zu realisieren sind (Transaktionsproblematik). Außerdem sind Authentifikation und Authentizität der Benutzer schwer realisierbar.

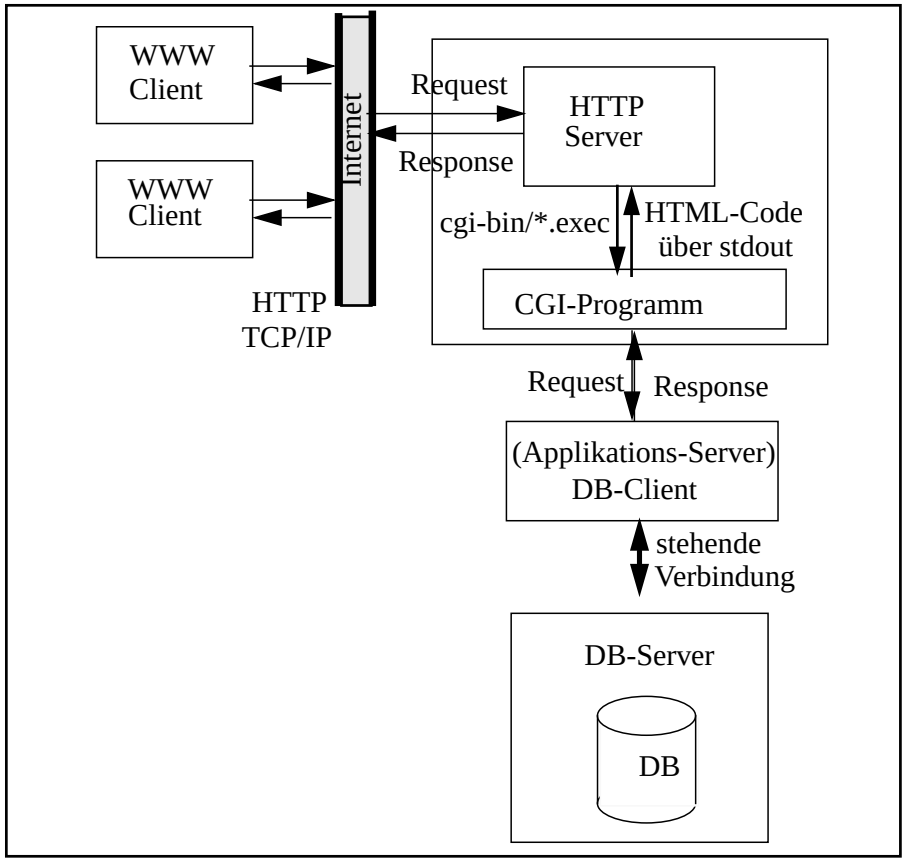

Abb. 2: Standardmodell einer CGI-Datenbankanbindung ans WWW

3.1.1.2 DB-Anbindung mit API

In diesem Fall besteht die Möglichkeit, die Datenbank an das WWW über eine spezielle API-Schnittstelle (Application Programming Interface) des WWW-Servers anzubinden. Hierbei wird eine Applikation geschrieben, die einerseits die API des WWW-Servers, andererseits die Schnittstelle zum Datenbanksystem benutzt. Die Applikation hat sodann die Möglichkeit durch (dynamisches) Binden mit dem Server auf die Datenbank zuzugreifen, ohne die CGI-Schnittstelle zu benutzen.

Vorteile:

- Da der WWW-Server als Datenbank-Client agiert, ist die Erzeugung eines neuen Prozesses für eine Datenbankanfrage nicht nötig.
- Es entfällt der Session-Aufbau. Da sich der WWW-Server beim Start in die Datenbank einloggt, entfällt ein Connect bei einer Anfrage des WWW-Clients. Dies führt zu einer Leistungssteigerung in der Antwortzeit und dem Datendurchsatz des Systems.

Nachteile:

- Durch die Benutzung der API des HTTP-Servers ist man an das Produkt gebunden. Die Benutzung eines anderen HTTP-Servers zu einem späteren Zeitpunkt führt zu erheblichen Modifikationen in der Applikation (Plattformabhängigkeit).
- Falls keine Server-Threads verwendet werden, kann der WWW-Server zum Flaschenhals für die Datenbankanbindung werden.
- Um Änderungen in die Applikation einzubringen, ist ein erneutes Kompilieren vonnöten.
- Zusätzlich gelten auch alle Nachteile, die bei der CGI-Anbindung genannt wurden.

3.1.1.3 Zusammenfassung

Die Verwendung der API-Schnittstelle scheidet für unser Projekt aus, da die Projektpartner unterschiedliche Softwareumgebungen betreiben. Die Technik, MultiMAP über die CGI-Schnittstelle des HTTP-Servers mit dem WWW zu verbinden, bietet vielfältige Möglichkeiten HTML-Seiten mit den Ergebnissen von Datenbankanfragen zu füllen. Allerdings verlangt MultiMAP, daß auch innerhalb einer HTML-Seite Aktionen ausgeführt werden können. Hier stoßen die statischen Möglichkeiten von HTML an ihre Grenzen.

Im Recherchemodus verursacht eine Beschränkung auf HTML folgende Schwierigkeiten:

- Bei der Recherche in Informationsblöcken soll der Benutzer in einem Informationsblock interaktiv arbeiten können. So sollen bei der Recherche in einem Bild durch einen Mausklick die Rechercheergebnisse angezeigt werden. Hierzu müßte für die Ergebnisse der Recherche jedesmal eine neue HTML-Seite aufgebaut werden.
- Zudem ist es in HTML nicht möglich, ein Objekt in einem Bild beispielsweise durch Blinken anzuzeigen. Statt dessen müßte für jedes Bildobjekt eine eigene Bilddatei erzeugt werden, in der dieses Objekt hervorgehoben ist. Das bedeutet jedoch einen großen Arbeits- und Speicherplatz-Aufwand und ist nicht praktikabel.

Weitere Probleme ergeben sich im Autorenmodus beim Erstellen neuer Objekte.

- Bei der Eingabe von Bildobjekten durch Markieren des Objekts mit der Maus können die bereits eingegebenen Punkte nicht im Bild angezeigt werden; es ist lediglich möglich, die Koordinaten der Punkte anzugeben.
- Textobjekte können nicht durch das Markieren des gewünschten Textbereichs definiert werden. Die Definition müßte durch die Angabe der Position im Text oder durch andere Lösungen erfolgen. Dadurch geht allerdings die Benutzerfreundlichkeit des Systems verloren.

Eine Lösung dieser Probleme bietet beispielsweise eine client-seitige Anbindung mittels der Programmiersprache Java. Durch den Einsatz von Java-Applets, Programme, die auf dem WWW-Client in einer HTML-Seite ablaufen, erreicht man ein großes Maß an Interaktivität und ist nicht mehr an die eingeschränkten Möglichkeiten von HTML gebunden.
In den Java-Applets können Datenbankoperationen aus einer HTML-Seite heraus ausgeführt werden, indem das Java-Applet direkt mit dem Datenbank-Client kommuniziert. Diese Technik des Datenbankzugriffs behandeln wir im nächsten Abschnitt.

3.1.2 Anbindung auf Client-Seite

Neben der Anbindung auf Server-Seite besteht zudem die Möglichkeit, den Client direkt mit dem Datenbank-Server zu verbinden. Folgende Methoden sind hierbei üblich:

- Anbindung mittels Java,
- Anbindung mit Hilfe von Plug-Ins.

3.1.2.1 Plug-Ins

Plug-Ins sind Erweiterungen für den Browser, die diesen mit einer zusätzlichen Funktionalität ausstatten. Es besteht nun die Möglichkeit, Plug-Ins für einen bestimmten WWW-Browser zu implementieren, der den Browser um eine Datenbankschnittstelle erweitert. Hierzu können die üblichen Programmiersprachen, z.B. C, eingesetzt werden. Diese Methode erscheint sehr effizient, sie führt jedoch zu einer Produktabhängigkeit, die für das Projekt MultiMAP nicht akzeptabel ist.

3.1.2.2 Java

Java ist eine objektorientierte Programmiersprache. Durch die Interpretation des aus Java-Quelltext für eine virtuelle Maschine erzeugten Byte-Codes wird Plattformunabhängigkeit erreicht. Dadurch ist es möglich, ein Programm (“Applet”) von einem Server in einen Java-fähigen WWW-Browser zu laden und es dort in der virtuellen

Maschine lokal auszuführen. Ein weiterer Vorteil liegt in der einfachen Integration der Applets in HTML-Dokumente. Die Applets können wie Bilder oder Texte in Hypertext-Dokumente integriert werden und erlauben so Animationen, Benutzerinteraktionen, Berechnungen sowie HTTP-unabhängige Netz- und damit auch DB-Verbindungen, die zustandsbehaftet sein können.

Die Verwendung von Java bringt jedoch auch einige Einschränkungen mit sich.

- Da mit einem Java-Applet ein fremdes Programm auf dem Benutzerrechner gestartet wird, besitzt Java einige Sicherheitseinschränkungen. So kann ein Applet eine Netzverbindung nur zu dem Rechner aufbauen, von dem das Applet selbst geladen wurde. Deshalb muß der Datenbank-Client, mit dem das Applet kommunizieren soll, auf demselben Rechner laufen wie der WWW-Server.
- Außerdem kann ein Applet nicht auf das lokale Filesystem des Benutzerrechners zugreifen, d.h. es kann dort weder Dateien lesen, noch schreiben. So ist es z.B. nicht möglich, in der Autorenkomponente digitalisierte Bilder vom WWW-Client aus mittels Java in die Datenbank einzubringen, da Bilder im Gegensatz zu Texteingaben immer zuerst auf der Platte liegen.

Diese Einschränkungen können nur mit zertifizierten Applets umgangen werden, die jedoch einer einfachen ad-hoc-Verwendung der Systeme entgegenstehen (geringe Akzeptanz bei Gelegenheitsbenutzern).

Bei der Verwendung von Java kann zusätzlich eine JDBC-Kopplung zwischen Applikation und Datenbank eingesetzt werden. Für Java-Kopplungen gibt es zwei Architekturen:

- zweistufige Architektur (two tier architecture)
- dreistufige Architektur (three tier architecture)

Um der Forderung, daß Java-Applets nur zusätzliche Verbindungen zu der IP-Adresse des Hosts aufbauen dürfen, von der sie geladen worden sind, gerecht zu werden, kann man im einfachsten Fall den Datenbankserver auf denselben Rechner wie den HTTP-Server legen.

Damit erhält man eine zweistufige Architektur. Sie entspricht einer typischen Client/Server-Architektur. Der WWW-Client fordert vom WWW-Server eine HTML-Seite mit dem eingebetteten Java-Applet zur DB-Anbindung an. Daraufhin wird vom WWW-Server zunächst die geforderte HTML-Seite übertragen und dann der vom Client nachgeforderte Java-Bytecode. Dieser wird in der JVM der Client-Maschine interpretiert. Er baut eine zusätzliche, stehende TCP/IP-Verbindung zum Datenbankserver auf. Die gesamte DB-Sitzung läuft nun am WWW-Server vorbei, in einer eigenen Verbindung.

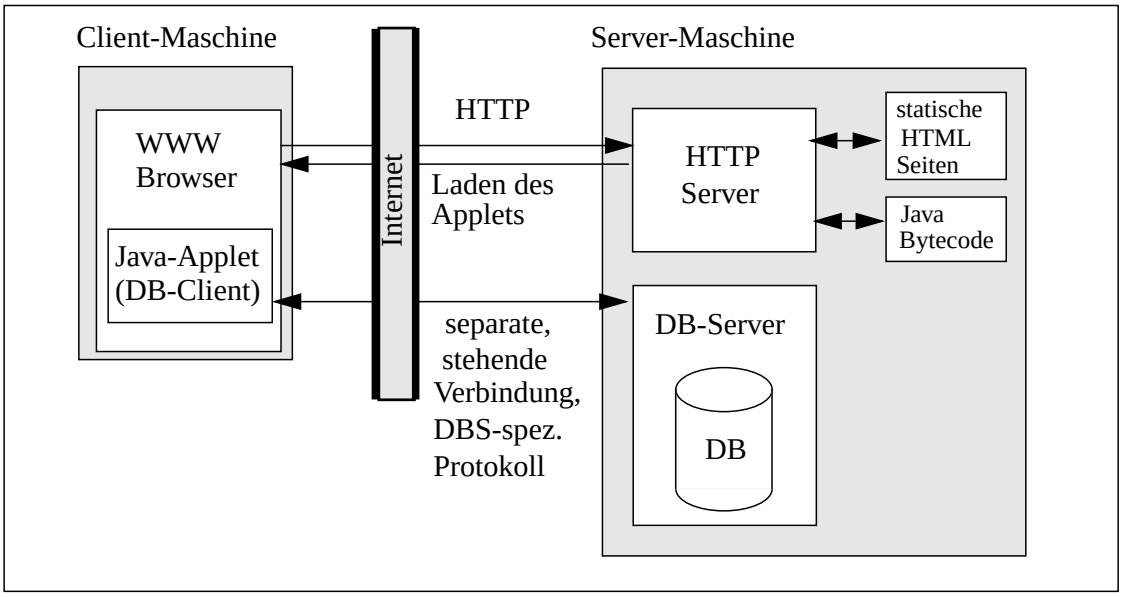

Abb. 3: Zweistufige Architektur

Bei der dreistufigen Architektur liegt der Datenbankserver auf einer eigenen Maschine. Da vom Java-Applet im WWW-Browser nur eine Verbindung bis zur Web-Server Maschine aufgebaut werden kann, wird dort ein Gateway als Vermittlungsstelle zwischengeschaltet. Dieses Gateway spielt nun gegenüber dem Datenbanksystem die Rolle eines DB-Clients, gegenüber dem Java-Applet aber die eines Applikation-Servers. Zwischen Gateway und DB-Server kann ein DB-spezifisches Protokoll gefahren werden, während das Protokoll zwischen Java-Applet und Gateway keine DBMS-Spezifika mehr zu enthalten braucht. Die Kommunikation mit dem Applet kann z.B. über RMI oder Sockets realisiert werden.

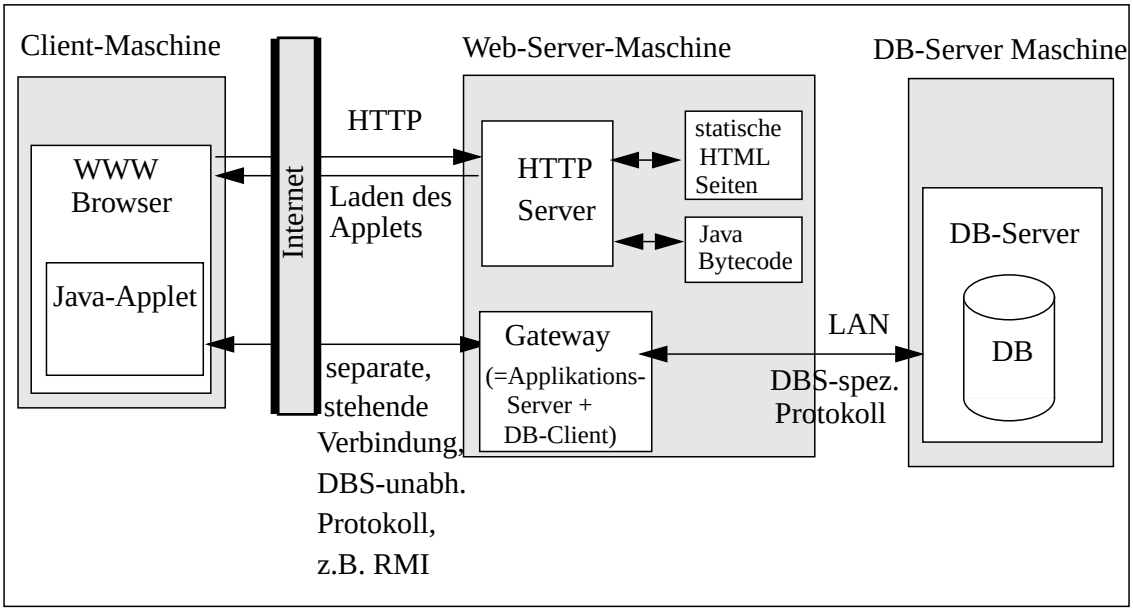

Abb. 4: Zwei-/Dreistufige Architektur

3.1.2.3 Zusammenfassung

Vorteile:

- *Informationsverarbeitung auf der Client-Seite:*

  Durch die DB-Anbindung auf der WWW-Client-Seite kann der Datentransfer verringert werden, da die Verarbeitung direkt auf der Client-Seite stattfindet. Dadurch besteht z.B. die Möglichkeit lokale Recherchen auf den einzelnen Informationsblökken (Anzeigen der sensitiven Bereiche im Bild durch Blinken) direkt, ohne DB-Nachfragen und damit ohne Netzbelastung zu realisieren.

- *Kein Connect pro Datenbankanfrage:*

  Es wird ausschließlich zum Sessionaufbau ein Connect zur Datenbank benötigt, weil der Client dann die Verbindung zur Datenbank ständig aufrechterhält. Diese Maßnahme führt sowohl zu einer CPU-Entlastung als auch zu einer kürzeren Antwortzeit der Datenbank.

- *Direkte Verbindung mit dem Datenbank-Server:*

  Der WWW-Client besitzt eine direkte Verbindung mit dem Datenbank-Server. Die Transaktionsverwaltung wird somit erheblich erleichtert. Somit sind Datenbankoperationen, die sich über mehrere Dokumente erstrecken, realisierbar.

- *Entlastung des WWW-Servers:*

  Da die Datenbankanfragen direkt an den Datenbank-Server geschickt werden, wird der WWW-Server stark entlastet.

- *Flexiblere GUI-Erstellung:*

  Bei der Erstellung der Benutzeroberfläche ist man nicht ausschließlich an die Möglichkeiten von HTML gebunden. Hier können z.B. beliebige interaktive graphische Elemente in die Benutzeroberfläche integriert werden.

Nachteil:

- *Ressourcenbedarf:*

  Ein Nachteil ist der größere Ressourcenbedarf im WWW-Browser, der durch das Starten der Java-Virtual-Machine (JVM) und das Hinzufügen der Datenbankanbindung verursacht wird.

3.1.3 Architekturentscheidung für MultiMAP/2

Wie bereits diskutiert, verursacht eine Beschränkung auf HTML bei der Darstellung auf dem WWW-Client eine Reihe von Schwierigkeiten:
Die HTML-Seite ist nach dem Verlassen des WWW-Servers statisch. Das bedeutet, daß beim WWW-Client keine interaktiven Aktionen stattfinden können.
Mit anderen Worten: Alle Aktionen auf einer HTML-Seite müssen dem WWW-Server mitgeteilt und von diesem behandelt werden.

Es können nur fertige Bilder dargestellt werden: Aktionen in Bildern, wie das Hervorheben von Objekten oder das Einzeichnen, Verschieben oder Löschen von Punkten eines Bildobjekts, sind nicht möglich.

Zusätzlich ergibt sich ein Problem bei der Behandlung von Bildobjekten, die ein Objekt repräsentieren und evtl. Anker von Links sind. In HTML gibt es für solche Zwecke die sogenannten Image-Maps, die bei einem Mausklick in das Bild alle Bereiche mit den Mauskoordinaten überprüfen. Dieser Test findet sogar lokal - ohne Kommunikation mit dem WWW-Server - auf dem WWW-Client statt. Überlappen sich jedoch einige dieser Bereiche, was bei MultiMAP sehr häufig vorkommt (z.B. eine Landkarte mit den Objekten Bayern und München oder bei den Objekten München und Isar), so bricht der Test nach dem ersten gefundenen Bereich ab, ohne daß alle weiteren Treffer für eine Auswahl durch den Benutzer aufgelistet werden. Man könnte dieses Problem umgehen, indem überlappende Bereiche aufgeteilt werden, so daß die Schnittmenge als ein eigener Bereich definiert wird. Das Eingefügen und das Löschen von Objekten würde demnach noch komplexer werden.

Aus diesen Gründen haben wir uns entschieden alle interaktiven und dynamischen Seiten, d.h. insbesondere die Bereiche Bilddarstellung, Recherche und die komplette Dateneingabe, als Java-Applets zu realisieren. Dadurch wird es möglich, Datenbankoperationen direkt aus dem Applet heraus aufzurufen, auf dem Rechner des Benutzers mehrere Fenster für eine übersichtlichere Darstellung zu öffnen und bei der Dateneingabe Objekte und Links einfacher zu erstellen und zu bearbeiten.

3.2 WWW-Adressierungsschema zur Datenbank MultiMAP

In Abbildung 2 wurde das Standardmodell einer Datenbank-Anbindung ans WWW mit CGI-Schnittstelle vorgestellt. Hier werden jedoch zwei Aspekte offen gelassen:

- Übermittlung einer DB-Query vom WWW-Browser zum DB-Client
- Umwandlung der Ergebnismenge in HTML

3.2.1 Dynamisches HTML

Wie bereits beschrieben, müssen in MultiMAP/2 dynamische HTML-Seiten generiert werden, die das Ergebnis der Datenbankanfrage beinhalten. Hier existieren vier mögliche Techniken:

- *Stored Procedures*: Datenbankanfragen werden als modulare Programmteile geschrieben und auf Server-Seite abgelegt, kompiliert und ausgeführt. Vom WWW-Browser wird hierbei ein Identifikator mit Parameter für die entsprechende Query mit Hilfe der URL übergeben.

- *Hard-Code-Gateway*: Anfragen an eine Datenbank werden fest in Form eines Programms oder Skripts codiert und im Dateisystem abgelegt. Das Seitenlayout der Ergebnisse der Anfrage wird entweder im Programm bereits spezifiziert oder in einer entsprechenden Konfigurationsdatei, HTML-Schablone oder Seitenbeschreibungssprache gespeichert. Nachteilig sind die schlechte Wartbarkeit und Skalierbarkeit.
- *URL-embedded Queries*: Die entsprechende Datenbankanfrage wird an die URL angehängt. Dies führt zum einem zu sehr langen URLs, zum anderen wird dem Benutzer das Datenbankschema offengelegt.
- *Server-Parsed-HTML*: Hierbei wird dem CGI-Programm ein Pfad übergeben, der auf eine HTML-Maske zeigt. Diese wiederum fungiert als Template. In ihr sind die Queries sowie die Formatierungsanweisungen für das Ergebnis enthalten. Die Syntax der Anfrage ist durch eine entsprechende Grammatik definiert, welche den gewünschten Befehlsumfang bietet. Die Parameter der Anfrage werden durch die POST-Operation (HTTP) übergeben und belegen die Variablen der Queries im Template. Das Gateway parst diese Maske, reicht die Anfragen an die Datenbank weiter und fügt die Ergebnistupel entsprechend der Formatierungsspezifikation in die HTML-Seite ein, die an den Browser als Ergebnis seiner Anfrage zurückgeschickt wird.

3.2.2 Einsatz in MultiMAP/2

Im Projekt MultiMAP/2 haben wir uns für alle statischen Ausgabeseiten für die Server-Parsed-HTML Methode entschieden. Durch eine URL wird hier eine bestimmte Maske adressiert, die von einem Programm (z.B. CGI-Programm) verarbeitet wird. Die Maske enthält die entsprechenden Befehle, die zur Kommunikation mit der Datenbank benötigt werden. Die einzelnen Masken stellen somit dynamische Dokumente dar. Die Syntax einer Maskensprache ist eine Erweiterung von HTML. Für DB-Anfragen wird der Kommentar-TAG <!-- --> erweitert. Die DB-Anfrage wird vom Programm ausgeführt und durch das Ergebnis ersetzt. Durch die Maske wird das Ergebnis entsprechend formatiert. Die so entstandene Seite wird an den Browser zurückgeschickt. In den folgenden Abschnitten werden die einzelnen Schritte genauer erläutert.

Die MultiMAP HTML-Maskensprache

Um die Maskenbefehle in eine HTML-Seite einbinden zu können, werden sie in HTML-Kommentare gekapselt. Ein Maskenbefehl hat folgende Form:

<!-- `MULTIMAP` *Befehl* -->

Alle im Folgenden erklärten Befehle müssen in diesem Format angegeben werden.

Behandlung von Variablen

Variablen können beliebig in die HTML-Masken eingebunden werden. Eine Variable hat die Form `&(Variablenname)`. Beim Abarbeiten der Maske wird jeder Ausdruck

`&(Variablenname)` durch den Wert der entsprechenden Variable ersetzt.

Setzen von Variablen
Mit dem Befehl `SET &(var) = expression` wird eine Variable belegt. `var` ist der Variablenname, `expression` ist ein Integer-Wert, eine andere Variable oder ein in doppelten Anführungszeichen stehender String. Mit `UNSET &(var)` wird die Variable wieder gelöscht.

Übergabe von Parametern an eine HTML-Maske
Der HTML-Maske können Parameter übergeben werden, indem sie in der Form /*param1*/param2/... der URL angefügt werden. In der Maske werden die Parameter mit dem Befehl `PATH var1 var2 ...` übernommen. Dann kann mit `&(var1)`, etc. auf die Variablen zugegriffen werden.

Umleiten des WWW-Clients
Mit dem Kommando `REDIRECT "<URL>"` wird der WWW-Client auf die neue URL `<URL>` umgeleitet.

SQL-Anweisungen
Mit dem Befehl `SQL_QUERY <sql_anweisung>` kann die Anweisung `sql_anweisung` an die Datenbank gestellt werden. Durch den Befehl `END_SQL` wird die Anweisung abgeschlossen. Handelt es sich dabei um eine SELECT-Anweisung, wird der Bereich zwischen dem `SQL_QUERY`- und dem `END_SQL`-Kommando für jedes Tupel des Ergebnisses, ähnlich einer Schleife, einmal durchlaufen. Dabei wird mit den Variablen `&(1)`, `&(2)`, etc. auf die Attributwerte des jeweiligen Schleifendurchlaufs entsprechend der Attribut-Reihenfolge in der Anfrage zugegriffen.

Beispiel: Bei der Anfrage

SELECT *value1*, *value2* FROM relation

bezieht sich im Programm `&(1)` auf den Wert von *value1* und `&(2)` auf den Wert von *value2*.

Bedingte Anweisungen
Mit dem Befehl

```
IF <bedingung> (If-Zweig) ELSE (Else-Zweig) ENDIF
```

werden bedingte Anweisungen ausgeführt. `<bedingung>` muß dabei einen boolschen Ausdruck enthalten, in dem zwei Ausdrücke expression mit den Operatoren "=", ">", "<", ">=", "<=" verglichen werden.

Werden bedingte Anweisungen und SQL-Anweisungen kombiniert, so ist bei der Verschachtelung auf die korrekte Reihenfolge der `ENDIF`- und der `END_SQL`-Anweisungen zu achten.

3.3 Die Grobspezifikation von MultiMAP/2 (Das Design-Modell)

Der folgende Abschnitt behandelt das Design-Modell von MuliMAP/2. Darin werden die grundlegenden Architekturmerkmale spezifiziert. Es bildet den Ausgangspunkt für die nachfolgende Feinspezifikation und die Implementation. Dieser Abschnitt gliedert sich in die Architektur des Kernsystems, das Datenbankschema sowie die Spezifikation der Client/Server-Schnittstelle.

3.3.1 Spezifikation der Architektur von MultiMAP/2

Die Architektur von MultiMAP/2 vereinigt die CGI-Anbindung auf der Server-Seite sowie die Java-Anbindung auf der Client-Seite. Durch diese Kombination erhält man ein extrem flexibles System, das sowohl die Funktionalität von HTML als auch die geforderte Interaktivität gewährleistet.

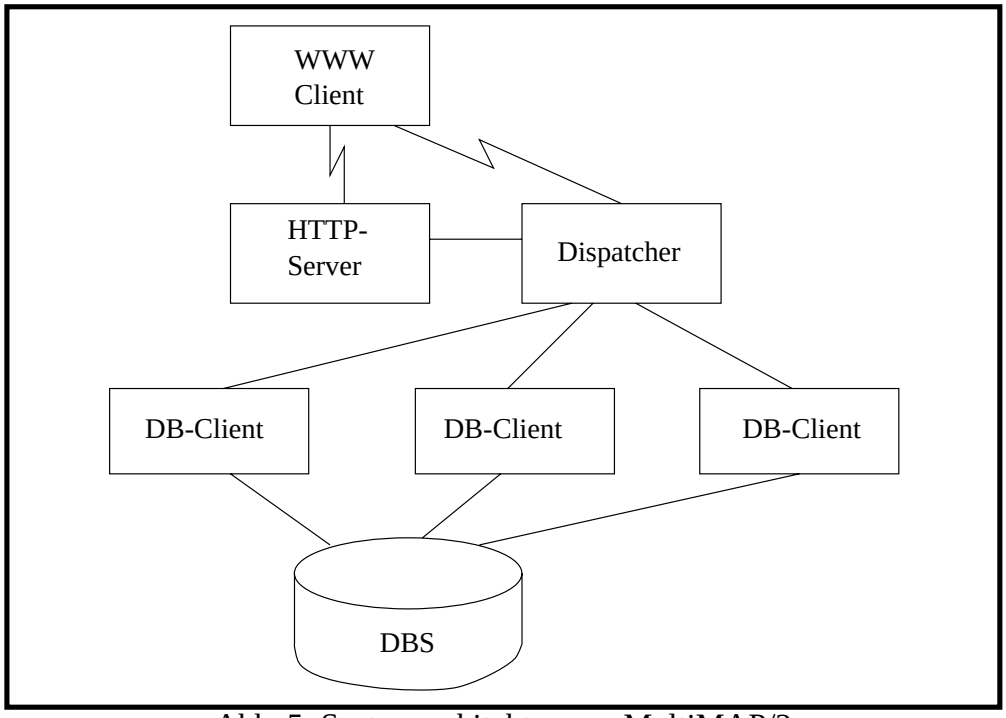

Abb. 5: Systemarchitektur von MultiMAP/2

Im Folgenden stellen wir die einzelnen Komponenten der Systemarchitektur kurz vor:

Das Front-End für MultiMAP wird durch einen WWW-Client realisiert. Bei dem WWW-Client handelt es sich um einen WWW-Browser, der Java-Applets ausführen kann. Der WWW-Client kommuniziert mit dem HTTP-Server. Daneben besteht eine direkte Verbindung zum Dispatcher, der die Datenbankanfragen an den entsprechenden DB-Client weiterleitet. Die Verbindung zwischen Dispatcher und DB-Client wird über Sockets realisiert. Für die Darstellung dynamischer Daten auf dem WWW-Client

werden Java-Applets eingesetzt, die mit dem DB-Client über den Dispatcher kommunizieren. Daneben gibt es eine Möglichkeit, die Informationen der Datenbank als Strings auszugeben, um ausschließlich textorientierte WWW-Browser, wie z.B. Lynx, zu unterstützen.

Der WWW-Server wird über die CGI-Schnittstelle mit dem DB-Client verbunden. Hierzu muß ein entsprechendes Gateway implementiert werden. Dieses Gateway basiert auf der Server-Parsed-HTML-Methode (s.o.). Dazu wird eine HTML-Erweiterung verwendet, mit der Datenbankanfragen und Kommandos, wie z.B. das Umleiten des WWW-Clients auf eine neue URL mit dem Kommando REDIRECT <URL>, in eine HTML-Seite eingefügt werden kann. Diese Technik zeichnet sich v.a. durch ihre einfache Modifizierbarkeit aus. Die Anfragen des HTTP-Servers werden an den Dispatcher weitergereicht.

Die Hauptaufgabe des Dispatchers ist die Verwaltung der einzelnen DB-Clients. Er leitet die Anfragen des WWW-Servers und des WWW-Browsers zu dem entsprechenden DB-Client (1:1-Beziehung). Diese Aufgabe ist deshalb wichtig, weil jedem WWW-Browser genau ein DB-Client zugeordnet ist. Daneben ist der Dispatcher sowohl für die Authentifikation als auch für die Authentizität verantwortlich.
Der DB-Client ist mit der Datenbank verbunden. Jedem Benutzer ist ein Datenbank-Client zugeordnet. Um die Datenübertragungen möglichst gering zu halten, werden alle SQL-Anfragen im DB-Client gekapselt. Ein weiterer Vorteil liegt darin, daß das Datenbankschema für den Anwender unsichtbar bleibt.
Als DB-Server kommt das relationale Datenbanksystem TransBase™ zum Einsatz.

3.3.2 Erweiterung des Datenbank-Schemas

Um die allgemeinen Anforderungen der generischen Darstellung von Graphikformaten zu verwirklichen, muß das Datenbank-Schema von MultiMAP erweitert werden. Abbildung 6 zeigt das Entity-Relationship (E/R) Diagramm zur Modellierung des Datenbankschemas von MultiMAP. Es stellt den Storage Layer des Dexter-Modells dar.

Eine zweite Erweiterung des Datenbankschemas der bisherigen, lokalen MultiMAP-Datenbank war aufgrund des Einsatzes von Java-Applets im WWW-Browser nötig: Da die Relation Bild keine Information über die Höhe und Breite des Bildobjekts enthält, kann der Platzbedarf erst ermittelt werden, wenn die gesamten Daten des Bildes auf den Client übertragen worden sind. Der Platzbedarf für das Applet muß jedoch zu Beginn der Übertragung festgelegt werden. Nach der Festlegung ist eine Änderung nicht mehr möglich. Diese Problematik führte noch zu einer Aufteilung der Recherchekomponente in den Informationsblock und den Bildbereich im ersten Demonstrator. Durch Hinzufügen der Attribute Breite, Höhe, Graphik-Format und Kompressionsart in die Relation Bild stellt sowohl die Implementierung eines generischen Graphikformats als auch die Integration des Bildes in das Browserfenster kein besonderes technisches Problem mehr dar.

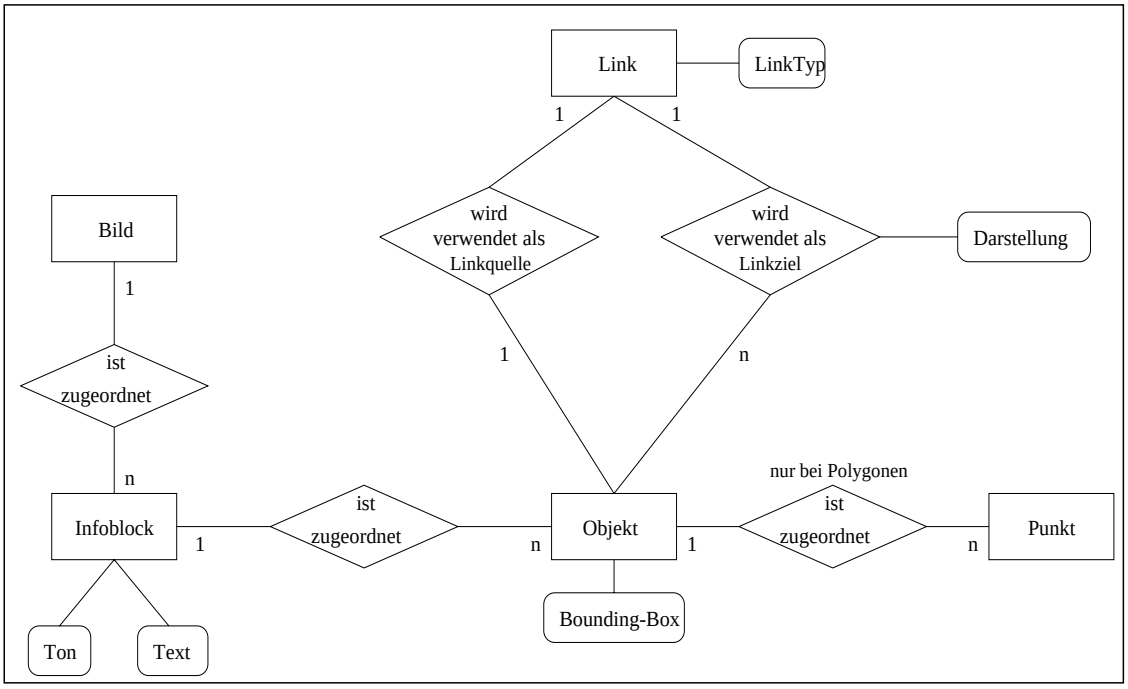

Abb. 6: E/R-Diagramm des DB-Schemas

3.3.3 Spezifikation der Client/Server-Schnittstelle

Im Gegensatz zum lokalen MultiMAP/1, bei dem das komplette Programm mit allen Modulen (Präsentation, Datenbankzugriffe, etc.) auf dem Rechner des Benutzers läuft, müssen diese Module im neuen System MultiMAP/2 auf die verschiedenen Komponenten (Datenbank-Client, Java-Applet und HTML-Seite) aufgeteilt werden.

Das Ziel der Spezifikation der Client/Server-Schnittstellen ist es, die Kommunikation zwischen den Java-Applets und dem Datenbank-Client effizient zu gestalten. Dazu werden Datenbankanfragen, die weitere Datenbankoperationen nach sich ziehen, zu einer komplexen Query zusammengefaßt. Ein Beispiel für eine solche Query ist: *“Hole die Daten zu allen Objekten eines Informationsblocks aus der Relation Objekt, hole zu jedem Objekt die vorhandenen Linkziele und, falls es sich um ein Bildobjekt handelt, die zugehörigen Punktkoordinaten”*[1].

Grundsätzlich werden alle SQL-Anfragen in Form von Funktionen auf dem DB-Client gekapselt. Auf die einzelnen Funktionen soll durch RMI (Remote Method Invocation) zugegriffen werden. Die Funktionen sind sowohl für Java als auch für die HTML-Masken verfügbar.

1. In SQL: SELECT * FROM Objekt WHERE InfoBlockID = <i>; für jedes dieser Tupel dann SELECT * FROM Punkt_xy WHERE ObjectID=<o> zum Laden der Punkte und SELECT i.Name, i.InfoBlockID, o.ObjektTyp, o.Name, o.ObjectID FROM Objekt o, InfoBlock i, Link l WHERE l.SourceObjID=<o> AND l.DestObjID=o.ObjektID AND o.InfoID=i.InfoBlockID zum Laden der Linkziele.

Neben den SQL-Anfragen, die direkt an die Datenbank weitergeleitet werden, werden folgende Operationen in den DB-Client integriert:

Objekt-Anfrage
Zu einem Informationsblock werden aus der Relation *Objekt* Name, ID, Typ, Autor und Erstellungsdatum aller Objekte ermittelt; falls es sich um ein Bildobjekt handelt, werden die Koordinaten der Punkte und der Bounding Box bestimmt. Schließlich wird über die Relation Link nach den Linkzielen zu jedem Objekt gesucht und für jedes Linkziel der Name und die ID des Informationsblocks sowie der Name, die ID und der Typ des Zielobjekts ermittelt.

Objekt-Eingabe
Diese Funkton realisiert die INSERT-Operation in die Relation Objekt

Objekt-Update
Hier werden in den Relationen Objekt und den Volltextrecherche-Relationen sowie, falls nötig, in der Relation Punkt_xy die eingegebenen Änderungen vorgenommen.

Löschen eines Objekts
Die zum Objekt gehörenden Daten werden aus den Relationen Objekt, Punkt_xy und den Volltextrecherche-Relationen gelöscht. Zusätzlich werden alle Links, bei denen das Objekt Linkquelle oder -ziel ist, aus der Relation Link entfernt.

Link-Eingabe
Die Daten eines Links werden in die Relation Link eingefügt, wobei für jedes Linkziel ein Eintrag erstellt werden muß. Handelt es sich um einen bidirektionalen Link, sind die Quelle-Ziel-Paare auch noch für die andere Richtung abzuspeichern.

Abspeichern eines Textes
Nach der Eingabe oder dem Löschen eines Textlinks muß der geänderte Text in der Relation InfoBlock aktualisiert werden.

Volltextrecherche
Aus dem durch “&”, “|” und “-” verknüpften Suchausdruck wird die zugehörige SQL-Anfrage erzeugt. Als Anfrageergebnis erhält man Name, ID und Art der Treffer.

Alle übrigen Funktionen von MultiMAP, wie z.B. die Aufbereitung der Anfrageergebnisse, die Organisation der Daten auf der Benutzer-Seite und ein Großteil der Präsentationen, werden von den Java-Applets übernommen.

3.3.4 Modularisierung

Diese Grobarchitektur läßt sich in folgende Module strukturieren:

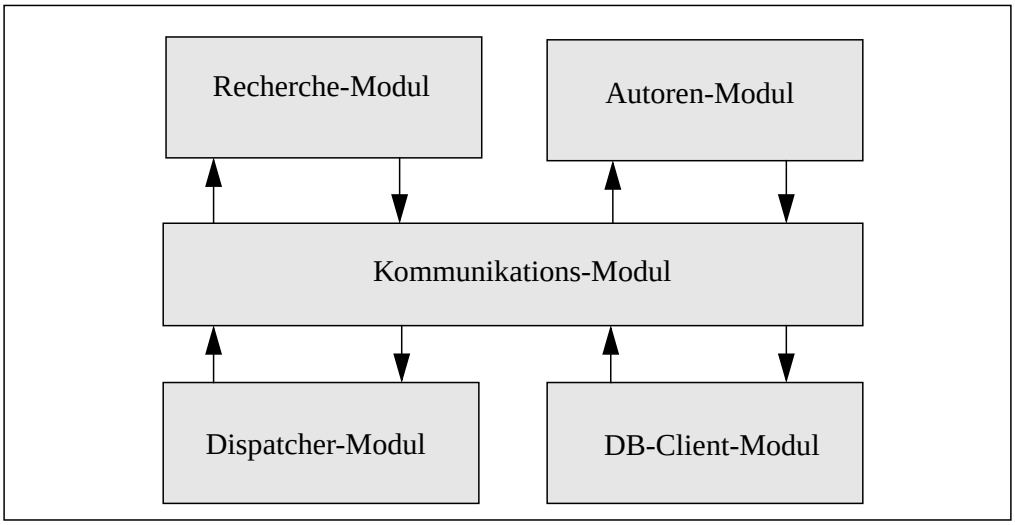

Abb. 7: Modularisierung von MultiMAP/2

Kernstück dieser Modularisierung stellt das Kommunikations-Modul dar. In ihm ist sowohl die Kommunikation über die CGI-Schnittstelle als auch die Kommunikation über RMI auf Java-Seite realisiert. Die einzelnen Module können nur über das Kommunikations-Modul kommunizieren.

3.4 Die Feinspezifikation von MultiMAP/2

Dieser Abschnitt behandelt die grundlegende Spezifikation von MultiMAP/2, in die insbesondere auch die Erfahrungen mit unserem vorläufigen WWW-Demonstrator, der zeitgleich entstand, einfließen.

3.4.1 Allgemeiner Ablauf

Ein Informationsblock in MultiMAP besteht aus einem Text, Bild und einer Ton-Datei, der innerhalb des WWW-Browsers dargestellt werden soll. Die einzelnen Bestandteile werden in einer Seite integriert. Um einen Informationsblock im WWW-Browser anzuzeigen, müssen folgende Schritte (siehe Abbildung 8) abgearbeitet werden:

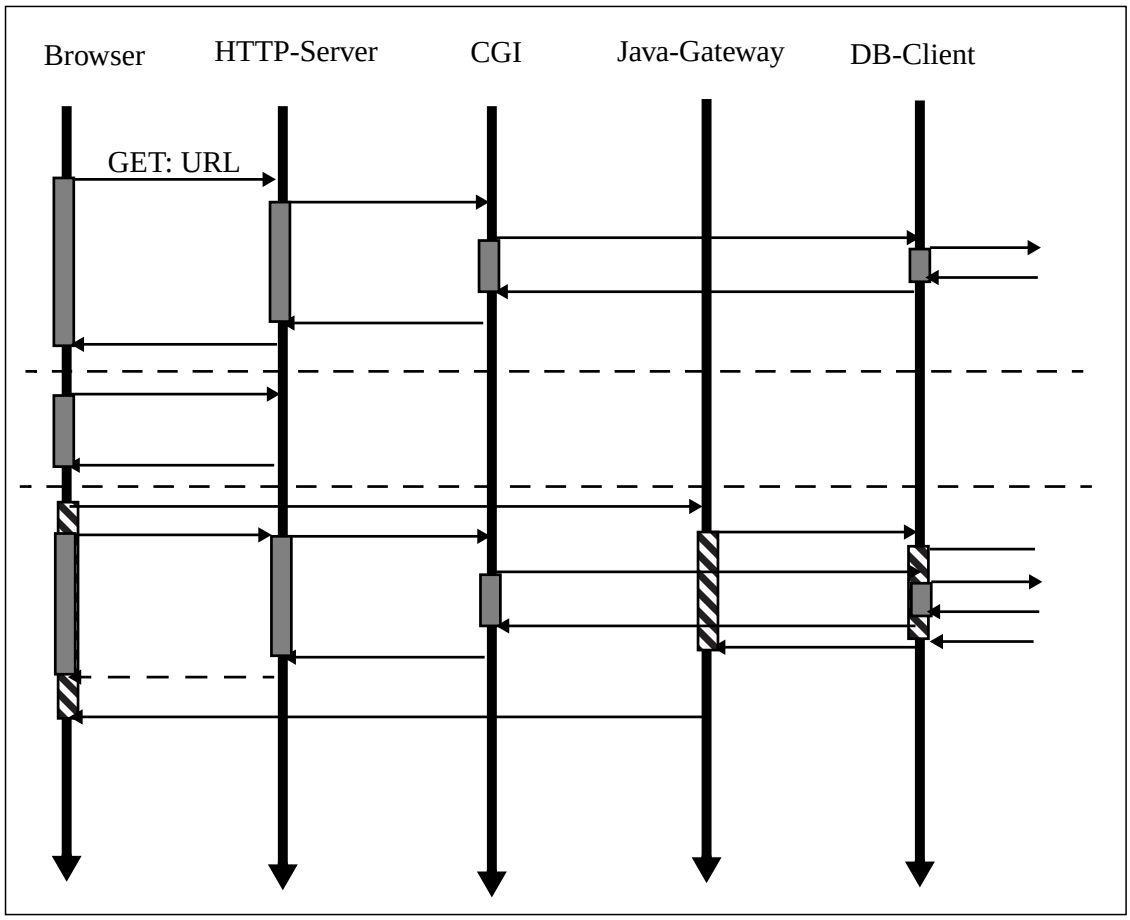

Abb. 8: Interaktionen der Module von MultiMAP/2

Der WWW-Browser fordert mit der Methode GET eine spezielle Seite, bei der es sich um eine HTML-Maske handelt, vom HTTP-Server an. Die HTML-Maske wird mit Hilfe einer URL adressiert und hat die Form

```
http://Server/CGI-Programm/Datenbank/USER-ID/HTML-Maske/Parameter
```

Der HTTP-Server startet das Gateway als CGI-Programm und übergibt ihm die URL. Das WWW-Gateway analysiert die URL und lädt die entsprechende HTML-Maske. In der HTML-Maske sind alle Anweisungen enthalten, um den Informationsblock zu erstellen. Dazu stellt das WWW-Gateway eine Verbindung (Connect) zum DB-Client her und schickt die einzelnen SQL-Statements, die in der HTML-Maske enthalten sind (Server-Parsed-HTML-Technik), zum DB-Client. Der DB-Client arbeitet die SQL-Statements ab und schickt die Ergebnisse an das WWW-Gateway zurück. Das WWW-Gateway reicht die erzeugte statische HTML-Seite, in der das Applet zur Darstellung des Bildes eingebettet ist, an den WWW-Server und dieser wiederum an den Browser weiter.

Im zweiten Schritt werden die einzelnen Java-Klassen vom WWW-Server geladen und im Browser ausgeführt.
Im dritten Schritt wird das Bild und die einzelnen Objekte, die im Bild enthalten sind, übertragen. Hierzu kommuniziert das Java-Applet direkt mit dem Java-Gateway. Das Java-Gateway kommuniziert wiederum mit dem DB-Client.

3.4.2 Architektur von MultiMAP/2

Die Architektur von MultiMAP/2 stellt eine Vereinigung von CGI-Anbindung auf der Server-Seite sowie eine Java-Anbindung auf Client-Seite dar. Aus dieser Architektur ergeben sich folgende Module:

1. WWW-Client
2. HTTP-Server
3. Dispatcher
4. DB-Client
5. Datenbanksystem

Dieses Modulkonzept wurde zu der in Abbildung 9 dargestellte Architektur verfeinert (zur Implementation siehe Abschnitt 3.5.1). Dieses Modell spiegelt die bekannte Drei-Schichten-Architektur für Datenbankanwendungen im Client/Server-Bereich wieder, die allgemein in der Literatur auch als “three tier achitecture” bezeichnet wird. Die drei Schichten teilen sich in einen Client (WWW-Browser), Middle Tier (DB-Client, CGI, httpd und Java Gateway) und Server (Datenbank) auf.

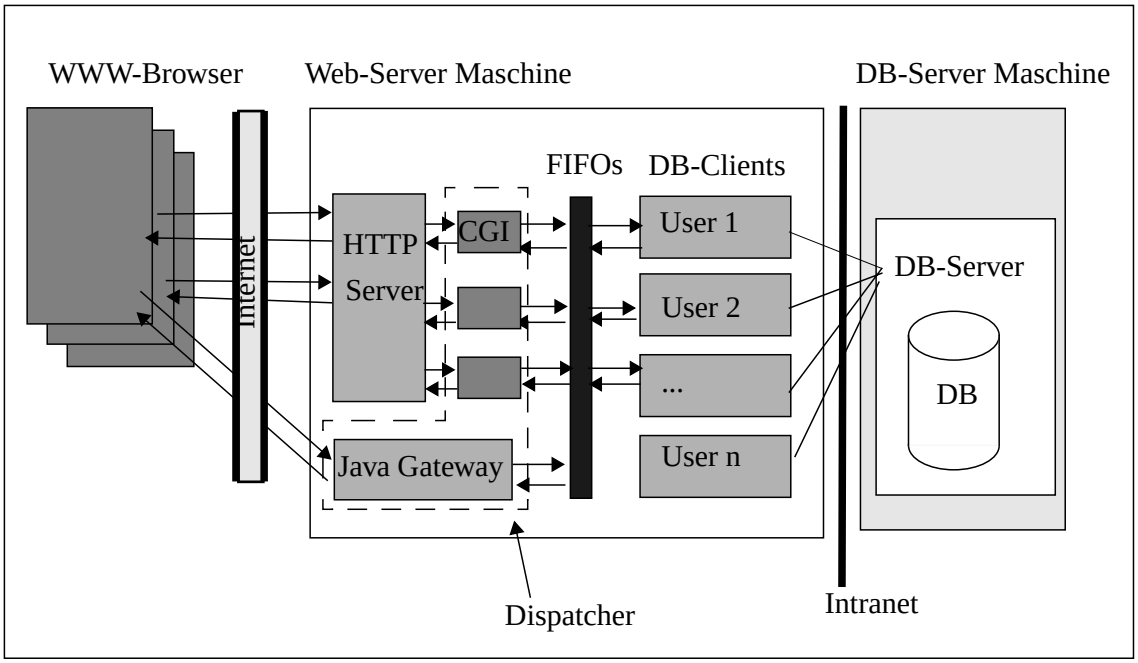

Abb. 9: Architektur von MultiMAP/2

Die Besonderheit dieser Architektur stellt das transaktionsorientierte Gateway (CGI-Programm und Java-Gateway) dar, auf das der HTTP-Server über die CGI-Schnittstelle (Common Gateway) zugreifen kann. Das Transaktionskonzept wird hierbei dadurch realisiert, daß jedem Anwender, der sich mit dem System verbindet, ein Applikationsserver (DB-Client) zugeordnet wird. Die Lebensdauer der einzelnen Applikationsserver ist auf die Dauer der jeweiligen Sitzung des Anwenders begrenzt. Hiermit ist es nun möglich, Transaktionen, die sich über mehrere HTML-Seiten erstrecken, zu verwalten.
Der WWW-Client, der durch einen Java-fähigen Browser realisiert wird, stellt das Front-End für MultiMAP dar, der mit dem WWW-Server (httpd) und dem Java-Gateway kommuniziert. Der WWW-Server startet hierzu ein CGI Programm. Das CGI-Programm und das Java-Gateway stellen zusammen das Dispatcher-Modul dar.
Der Dispatcher, der für die Verwaltung der einzelnen Applikationsserver verantwortlich ist, leitet die Anfragen an den entsprechenden Applikationsserver (DB-Client) weiter.
Die Kommunikation zwischen dem CGI-Programm, Java-Gateway und dem Applikationsserver wird durch “FIFOs” (named Pipes) realisiert. Hierbei handelt es sich um einen unidirektionale Kommunikationkanal, der über das File-System angesprochen werden kann. Der DB-Client öffnet zunächst eine FIFO mit einer normalen I/O-Operation, für die er eine Lese-Berechtigung hat. Das WWW-Gateway öffnet die selbe FIFO mit einer Schreib-Berechtigung und schickt seine Anfrage an den DB-Client. Die Antwort des DB-Client an das WWW-Gateway wird über eine separate FIFO übergeben, da FIFOs unidirektional sind. Da FIFOs durch ihren Dateinamen identifiziert werden, kann das WWW-Gateway die passende FIFO einfach feststellen. Hierzu wird die BenutzerID herangezogen. Der DB-Client realisiert dann die Kommunikation mit dem DB-System.

3.4.3 Spezifikation der Schnittstellen der einzelnen Module

In diesem Abschnitt werden die wichtigsten Schnittstellen der einzelnen Module, die in der Architektur dargestellt worden sind, kurz beschrieben.
Die Kommunikation zwischen der Mittelschicht und dem WWW-Browser unterteilt sich in zwei Kommunikationskanäle:

1. Kommunikation über HTTP mit dem WWW-Server (HTML)
2. Kommunikation mit dem Java-Gateway über Sockets (Java-Applet).

Kommunikation mit dem WWW-Server

Die Kommunikation des WWW-Browser mit dem WWW-Server wird mit Hilfe des HTTP-Protokolls verwirklicht. Hierbei wird die GET-Methode verwendet, um die entsprechende Maske vom WWW-Server zu holen.

```
GET /cgi-bin/database/p_start/user-id HTTP/1.0
```

Hierbei handelt es sich um einen Ausschnitt der Anfrage des WWW-Browsers an den WWW-Server. `/cgi-bin/database/p_start/user-id` bezeichnet dabei sie entsprechende URL.

Kommunikation mit dem Java-Gateway

Die Kommunikation mit dem Java-Gateway wird mit Hilfe von TCP/IP Sockets, die im *package java.net.Socket* zu Verfügung gestellt werden, verwirklicht. Dazu wird die Klasse *Connection* eingesetzt, in der die Kommunikation gekapselt ist. Sie stellt folgende Methoden bereit:

***Connection**(String **host**, int **port**)*
Erzeugt eine Verbindung zum Java-Gateway auf dem Rechner *host* mit der Portnummer *port*.

*public void **send**(String **msg**)*
Sendet über die Socketverbindung eine Zeichenkette.

*public void **sendBytes**(byte[] **bytearr**, int **size**)*
Sendet über die Socketverbindung eine Bytefolge bestimmter Länge.

*public String **readBytes**(int **len**)*
Liest *len* Bytes von der Socketverbindung.

*public String **receive**()*
Liest eine Zeichenkette bis zum Zeilenende.

*public void **stop**()*
Schließt die Verbindung zum Java-Gateway.

Auf die Klasse ***Connection***, mit der einzelne Befehle bereits an das Java-Gateway und dementsprechend an den DB-Client geschickt werden können, baut die Klasse ***SqlCon*** auf. Diese Klasse verwaltet SQL-Anfragen:

***SqlConn**(String **sql_statement**)*
Konstruktor-Methode: Führt eine SQL-Anfrage aus. Dabei wird der SQL-Ausdruck als kurze Transaktion behandelt.

***SqlConn**(int **amode**)*
Konstruktor-Methode: Führt einen Transaktionsstart durch. Hiermit können mehrere SQL-Anfragen mit *executeSql* innerhalb einer Transaktion durchgeführt werden.

*public Vector **executeSql**(String **sqls**)*
Führt eine SQL-Anfrage aus. Das Ergebnis wird in Form eines Vektors zurückgeliefert.

*public Vector **getoOneTupel**()*
Liefert ein Tupel der SQL-Anfrage zurück.

*public Vector **getAllTupels**()*
Liefert alle Tupel der SQL-Anfrage zurück.

3.5 Implementation von MultiMAP/2

3.5.1 Implementation der Mittelschicht

Dieser Abschnitt beschreibt die Implementation von MultiMAP/2. Da beim Design darauf geachtet worden ist, die Applikation in unabhängige Module aufzuteilen, die über festdefinierte Schnittstellen angesprochen werden, konnte die Implementierung der einzelnen Module parallel vorangetrieben werden. Begonnen wurde mit der Implementation des Datenbankschemas, sowie der Mittel-Schicht. Darauf folgte die Präsentationsschicht im WWW-Client. Hier konnten die Erfahrungen des Demonstrators einfließen.

Mittelschicht

Die Mittelschicht, die aus den Modulen DB-Client, CGI-Programm, Java-Gateway und dem WWW-Server besteht, stellt den Kern des Netzzugangs der multimedialen Datenbank MultiMAP dar. Wie in Abbildung 9 dargestellt, besteht der Dispatcher aus den beiden Modulen CGI-Programm und Java-Gateway. Der Dispatcher kommuniziert mit dem jeweiligen DB-Client über named Pipes.

Grundsätzlich (vgl. für das folgende Architekturabb. 9 und Interaktionsdiagramm Abb. 8) wird für jede Anfrage des WWW-Browsers (Anfrage nach einer Seite oder einem InfoBlock) ein CGI-Programm gestartet, dem die URL des Dokuments als Argument übergeben wird**.** Das WWW-DB-Gateway parst die URL und extrahiert die Benutzer-ID. Hiermit kann das DB-WWW-Gateway den verantwortlichen DB-Client bestimmen. Dazu wird in einem bestimmten Verzeichnis (z.B. tmp/fifo/) nach zwei named Pipes, die zur BenutzerID gehören, gesucht. Falls die named Pipes existieren, läuft ein entsprechender DB-Client und das DB-WWW-Gateway kann seine Anfrage an den DB-Client übertragen. Hierzu verlangt es einen exklusiven LOCK auf die Schreib-Fifo vom System. Diese Fifo stellt den Kanal zur Übertragung von Anfragen an den DB-Client dar. Sie ist nötig, da der DB-Client nur sequentiell Anfragen bearbeiten kann. Erhält das DB-WWW-Gateway den exklusiven LOCK, öffnet es die Lese-Fifo, womit das DB-WWW-Gateway mit dem DB-Client kommunizieren kann.

Stellt der Benutzer die erste Verbindung zum Transaktionsserver her, existiert jedoch noch kein persönlicher DB-Client, von dem alle Transaktionen für den Benutzter während einer Sitzung mit dem System verwaltet werden.
Zunächst erzeugt das DB-WWW-Gateway ein lock-file um sicherzustellen, daß nur ein DB-client pro Benutzer erzeugt wird. Dazu wird die Fifo mit der Option O_CREATE[2] und O_EXCL[3] angelegt, so daß nur der erste die Möglichkeit hat eine Named-Pipe

2. Erzeugt eine Datei, falls sie noch nicht besteht. Die Operationen benötigt ein weiteres Argument, indem die Zugriffsart spezifiziert wird.
3. Falls O_CREATE auch angegeben wird und die Datei bereits existiert, wird ein Fehler zurück gegeben. Dieser Test wird vom System als atomare Aktion ausgeführt.

anzulegen. Nun kann das WWW-Gateway den DB-Client starten, dem der Benutzername als Parameter übergeben wird. Der DB-Client erzeugt nun zwei passende Pipes, die als Lese- und als Schreibkanal zwischen DB-Client und WWW-Gateway/Java-Gateway fungieren. Das WWW-Gateway stellt eine Verbindung her, nachdem es die Schreib-Named-Pipe mit einer Sperre belegt hat. Danach wird der am Anfang erzeugte Lock wieder freigegeben.

HTTP-Server

Im Projekt wurde der Apache 1.2.1[4] HTTP-Server verwendet. Zur Kommunikation mit dem Applikationsserver (DB-Client) wird das Common Gateway Interface (CGI) 1.1[5], die vollständig vom Apache HTTP-Server unterstützt wird, benutzt.

3.5.2 Implementation der Benutzerschicht der Recherchekomponente

Das zentrale Applet der Benutzerschicht ist das Applet IBRecherche (Interaktive Bild Recherche). Es übernimmt in der Recherche-Komponente die Behandlung der interaktiven Bilder sowie die Funktionen zur orts- und namensbezogenen Recherche im aktuellen Informationsblock. Es wird über eine Referenz in der Informationsblock-Maske in diese eingebunden.

Für die Modellierung der Java-Applets in der Recherche- und der Autorenkomponente wurde die "*Object Modeling Technique*" (OMT) von J. Rumbaugh verwendet. Die OMT besteht aus dem Objektmodell, dem dynamischen Modell und dem funktionalen Modell.

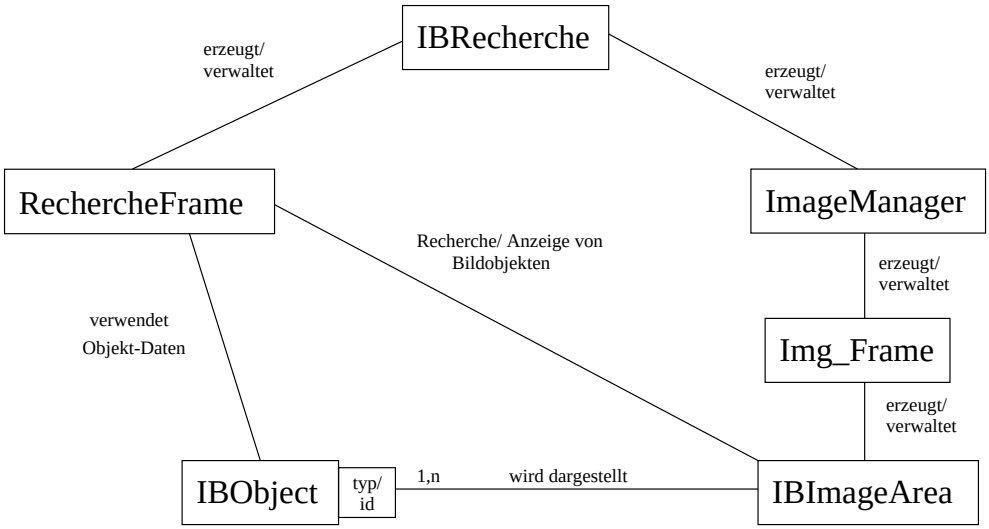

Abb. 10: Das Objektmodell von IBRecherche

4. http://www.apache.de/
5. http://www.w3.org/CGI/

Abbildung 10 enthält das Objektmodell des Applets IBRecherche. Die Klassen Recherche Frame und IBImageArea sind dadurch miteinander verknüpft, daß einerseits aus IBImageArea eine objektbezogene Recherche in RechercheFrame veranlaßt wird und andererseits RechercheFrame die Anzeige eines Bildobjekts in IBImageArea auslöst. RechercheFrame verwendet bei der Anzeige der Ergebnis-Objekte zu einer Recherche die Objektdaten aus der Menge der Objekte IBObject. Aus derselben Menge ermittelt IBImageArea die darzustellenden Bildobjekte durch eine Auswahl über ihren Typ bzw. ihre ID.
Das dynamische Modell des Applets in Abbildung 11 beschreibt seine komplette Struktur. Nachdem das Applet vollständig geladen und initialisiert wurde, wird das Zustandsdiagramm durchlaufen. Der Zusatz DB kennzeichnet die Zustände, in dem eine Datenbankoperation erforderlich ist.

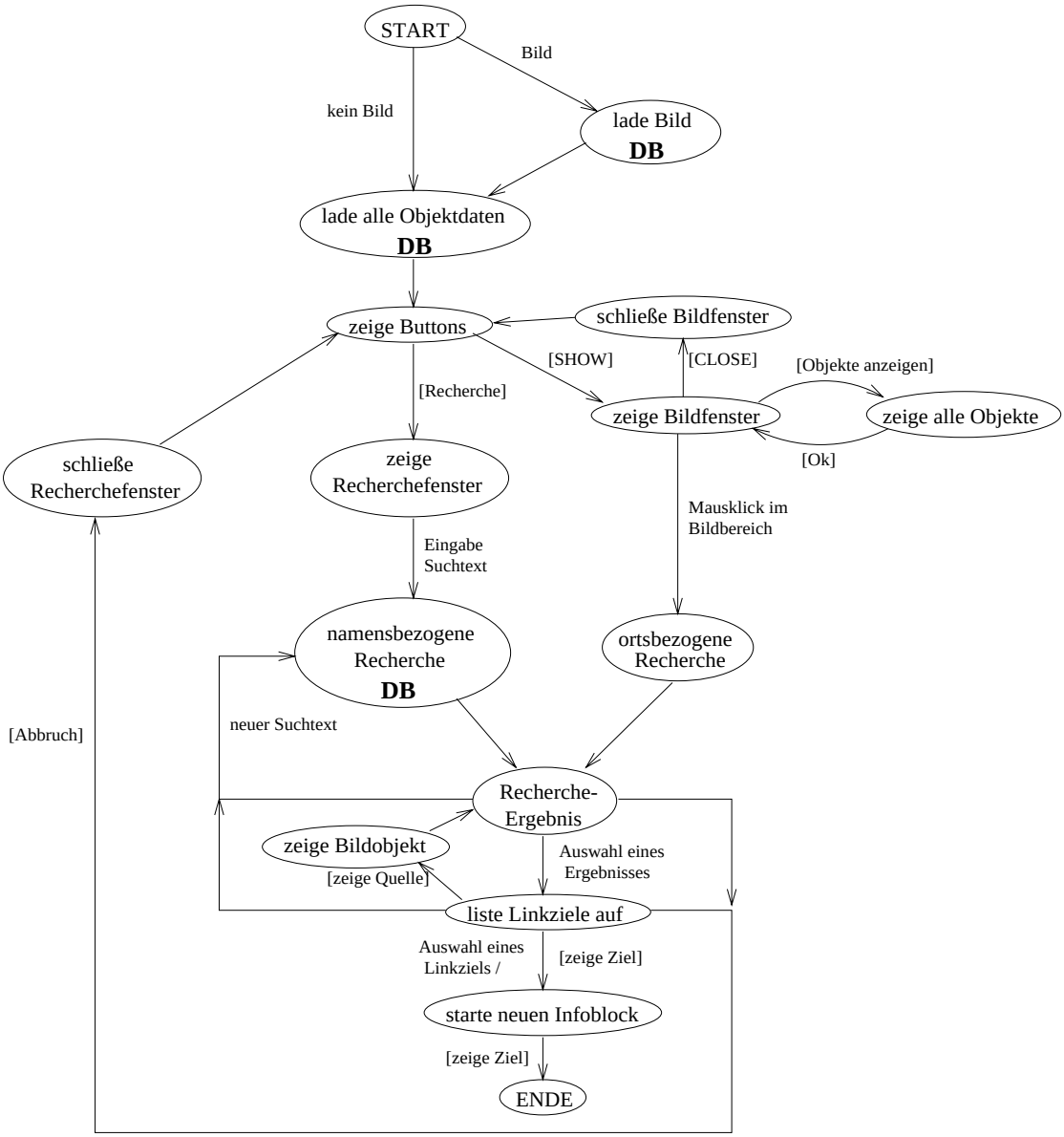

Abb. 11: Das dynamische Modell von IBRecherche

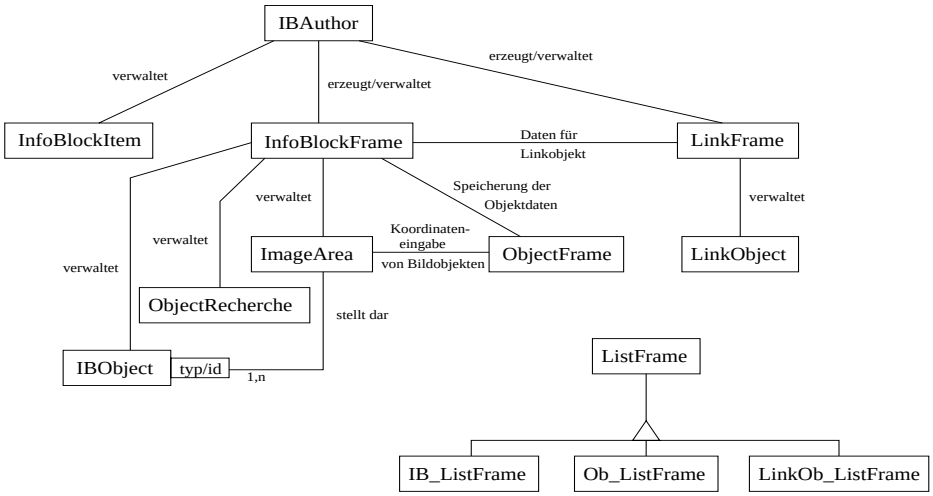

Abb. 12: Objektmodell von IBAuthor

Beim Verlassen der aktuellen Informationsblock-HTML-Seite wird das Applet beendet; wird jedoch ein Link verfolgt oder ein neuer Einstiegspunkt ausgewählt, wird das Diagramm von neuem durchlaufen, da jeder Informationsblock das gleiche Applet verwendet.

3.5.3 Implementation der Autorenkomponente

Die Autorenkomponente von MultiMAP/2 ist für die Dateneingabe zuständig. Hier werden Objekte eingegeben und Links zwischen Objekten erzeugt. Die Autorenkomponente besteht aus einer HTML-Seite, in der sich das Java-Applet IBAuthor befindet. Diese Seite besitzt keine weiteren HTML-Bestandteile, d.h. die Dateneingabe erfolgt ausschließlich in Java.

Abbildung 12 zeigt das Objektmodell des Applets IBAuthor. Die Hauptklasse IBAuthor ist sehr zentral im Modell, da jede Aktion, wie z.B. der Aufruf eines Informationsblocks zur Bearbeitung eines Objekts oder die Bearbeitung eines Links, von hier ausgeht. Nach seiner Instantiierung übernimmt InfoBlockFrame die Steuerung aller Vorgänge, die die Erstellung und Bearbeitung von Objekten betreffen. Wird in InfoBlockFrame ein Objekt als Linkelement ausgewählt, so ruft InfoBlockFrame das Linkdefinitionsfenster LinkFrame auf und übergibt diesem die Daten des Objekts.

Das dynamische Modell in Abbildung 13 veranschaulicht die umfangreiche Funktionalität des Applets IBAuthor. Dabei ist zu erwähnen, daß in diesem Modell die Operationen zur Bearbeitung eines Informationsblocks, d.h. Erstellen, Ändern und Löschen eines Informationsblocks, nicht enthalten sind.

Das Modell besitzt keinen Endzustand, da das Applet sich nicht selbst beenden kann. Es wird erst dann beendet, wenn der Benutzer die HTML-Seite der Autorenkomponente verläßt, indem er eine andere HTML-Seite aufruft bzw. den WWW-Browser schließt.

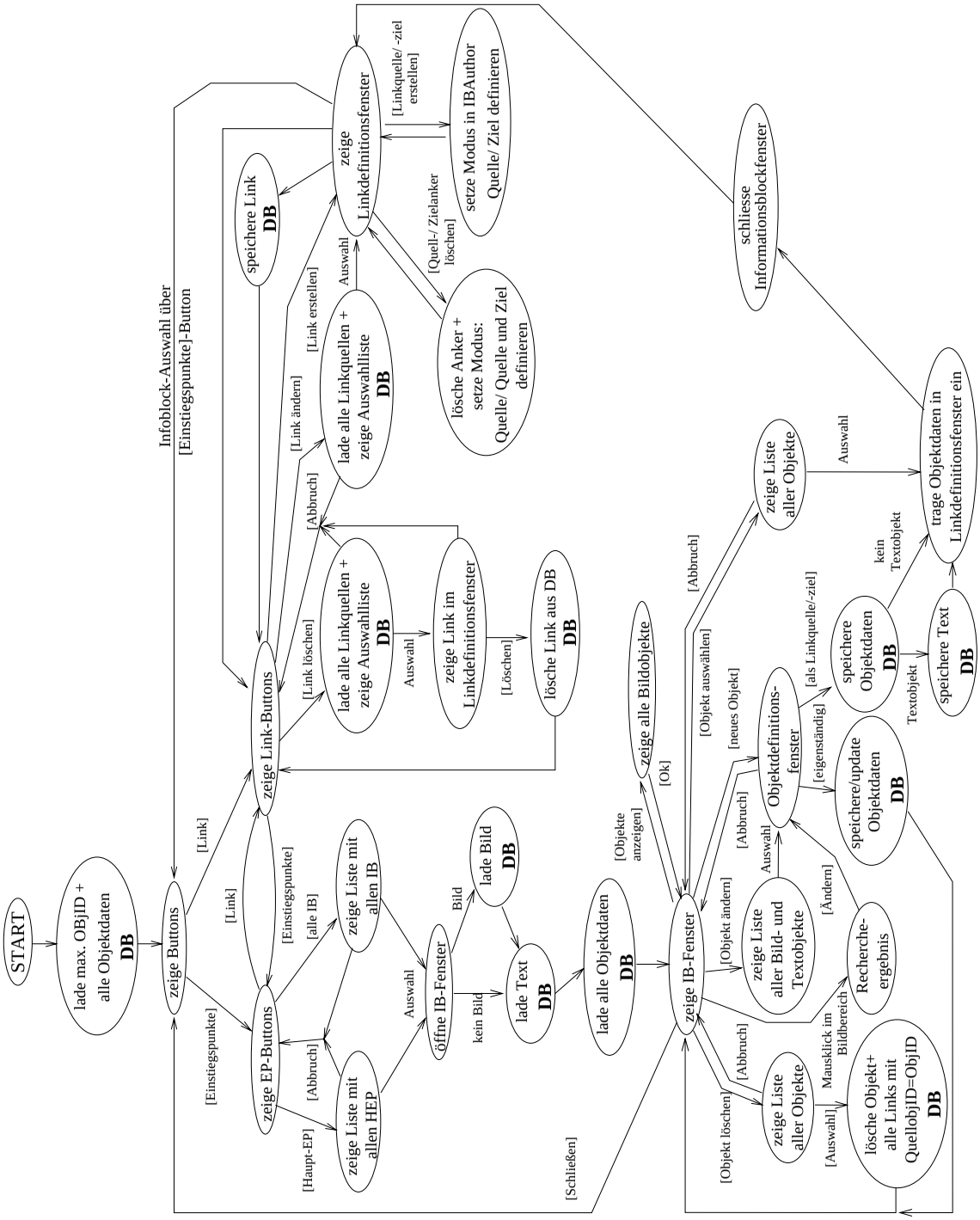

Abb. 13: Dynamisches Modell von IBAuthor

3.6 Installation und Benutzerevaluation

3.6.1 Installation des WWW-Prototypen

Um die Verteilung und Installation der Software bei den Anwendern zu erleichtern, wurde ein entsprechendes Installationsprogramm entwickelt.

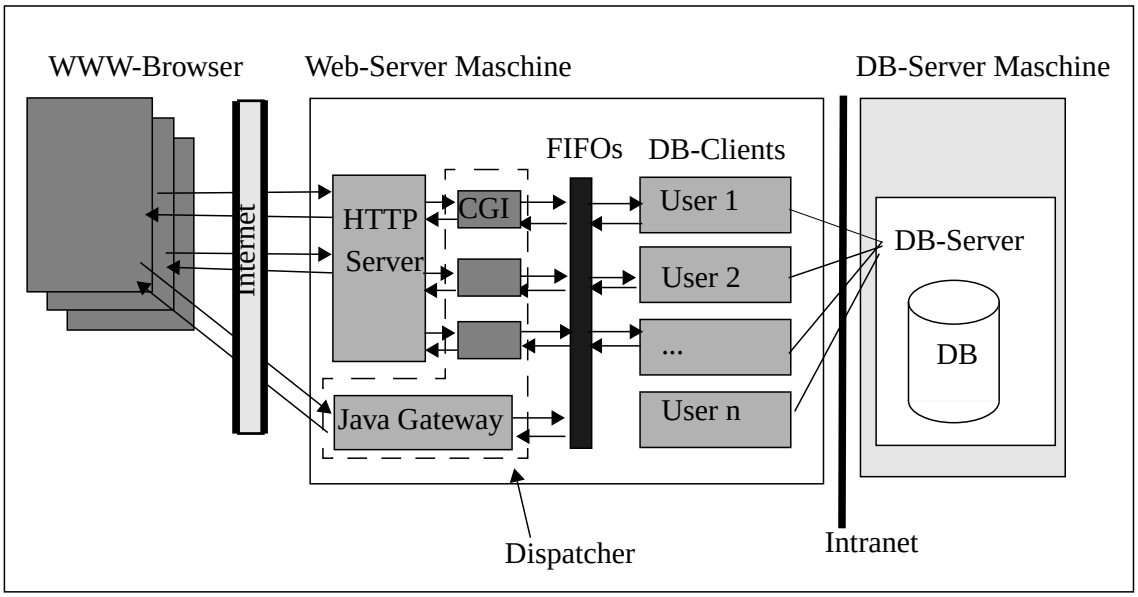

Abb. 14: Architektur von MultiMAP/2

Betrachtet man hierzu die Architektur von MultiMAP/2 (Abbildung 14) gliedert sich die Installation in zwei Teilbereiche: Die Installation des Datenbankservers einschließlich des Datenbankschemas für MultiMAP und die Installation des Middle Tier (dbclient, CGI-Gateway, Java-Gateway, http-Server). Die Installationen des Datenbanksystems TransBase sowie des HTTP-Servers sind Standard-Installationen und werden hier nicht weiter beschrieben.

Installationsprogramm Configure

Den Projektpartnern wird der WWW-Prototyp in Form einer Archivdatei zu Verfügung gestellt. Nach dem Entpacken des Archives zeigt das Root-Verzeichnis folgenden Inhalt:

conf: enthält das Installationsprogramm Configure,

database: enthält Datenbank-Utilities,

dbclient: enthält den Applikationsserver dbclient,

gateway: enthält sowohl das CGI-Gateway als auch das Java-Gateway,

html: enthält die einzelnen HTML-Masken,

java: enthält die Java-Klassen für die Java-Applikation, als auch der Java-Applets,

readme: enthält einzelne README-Dateien.

Um das WWW-MultiMAP-System auf den jeweiligen Hostrechner einzurichten, muß im Verzeichnis *conf* das Shell-Scrips ***configure*** angepaßt werden. Dazu wird das Script editiert und die folgenden Umgebungsvariablen konfiguriert.

P_ID: Diese Umgebungsvariable muß mit einem eindeutigen Bezeichner besetzt werden. Durch diesen Bezeichner werden die einzelnen Systeme auf dem Host unterschieden.

DATA_BASE: Enthält den Namen der Datenbank, die von dem Applikationserver angesprochen wird.

DB_HOST: Enthält den Hostnamen des Rechners, auf dem die Datenbank liegt.

MULTIMAPHOME: Enthält das Rootverzeichnis des MultiMAP/2-Systems.

WWWUSER: Enthält den Account, unter dem der WWW-Server gestartet wird.

WWW_SERVER_HOST: Enthält den Hostnamen des Rechners auf dem der WWW-Server installiert ist.

WWW_SERVER_PORT: Enthält die Portnummer, unter der der WWW-Server angesprochen werden kann. Die übliche Portnummer ist 80.

PUNKTGENAUIGKEIT: Diese Umgebungsvariable enthält einen Integer-Wert. Dieser Wert definiert die maximale Abweichung bei der Selektion eines Objektes mit der Maus.

DBPASSWORD: Enthält das Passwort, um sich in die Datenbank einzuloggen.

JAVA_PATH: Enthält ein eindeutiges Unterverzeichnis in *.html-data,* um verschiedene MultiMAP-Systeme auf demselben Host zu installieren.

CGI_BIN: Enthält den absoluten Pfad zum cgi-Verzeichnis des benutzten WWW-Servers.

MMCGI_CONF: Enthält den absoluten Pfad zu den Konfigurations-Dateien des WWW-Servers.

HTML_DATA: Enthält das Root-Verzeichnis der HTML-Verzeichnisse des verwendeten WWW-Servers.

SERVER_PORT: Enthält die Portnummer des Java-Gateways

Nach der Anpassung der einzelnen Umgebungsvariablen wird durch Ausführen des Shellscripts ein lauffähiges MultiMAP/2-System auf dem Hostrechner installiert.

3.6.2 Benutzerevaluation der GUI und Funktionalitätswünsche

3.6.2.1 MultiLIB

Durch die Bereitstellung eines Prototypen von MultiLIB auf unserem WWW-Server konnten bereits im vierten Projektquartal die ersten Benutzerevaluationen durch Herr Dr. Hilz (Projektpartner) vorgenommen werden. Dabei entstand der Wunsch einer Erweiterung der Autorenkomponente im Bereich der Link-Löschung, die anschließend entsprechend erweitert wurde.

3.6.2.2 MultiBHT

Der Prototyp Version 1.0 für MultiBHT entsprach dem von MultiLIB. Da es sich bei MultiBHT um ein erheblich komplexeres System handelt, war klar, daß das Grundsystem für diese Anwendung nicht ausreichen wird. Daher wurden zusammen mit den Projektpartnern weitere Anforderungen und Erweiterungen der Benutzerschnittstelle im Rahmen der ersten Benutzerevaluation gemeinsam festgelegt. Zusätzlich ergab sich dabei eine Restrukturierung der beim Anwender zugrundeliegenden Datenbank BHtDB. Der Name BHtDB steht für “Biblia Hebraica transcripta Datenbank” und enthält alle grammatikalischen Analysen aller Wörter, Wortgruppen und Sätze des vorhandenen Textkorpus der althebräischen Sprache und damit insbesondere aller Texte des Alten Testaments (einschließlich der aramäischen Teile).
Die Restrukturierung von einzelnen Analyseprogramm-bezogenen Datenbanken hin zu einer integrierten Datenbank ergab eine völlig überarbeitete BHt-2 Datenbank, die jetzt nur mehr aus 5 Relationen besteht.

3.6.3 Erweiterungen in MultiBHT aufgrund der Benutzerevaluation

In MultiBHT wurde neben der vollständigen Integration der Analyseergebnisse aus BHtDB eine Erweiterung der Recherche-Komponente gewünscht. Hier sollen Standardanfragen über Formulare sowie ad-hoc Anfragen zu Verfügung gestellt werden. Das Ergebnis der Systembewertung durch unseren Projektpartner hatte eine Umstrukturierung im Bereich des Clients, des Servers und der Datenbank zur Folge.

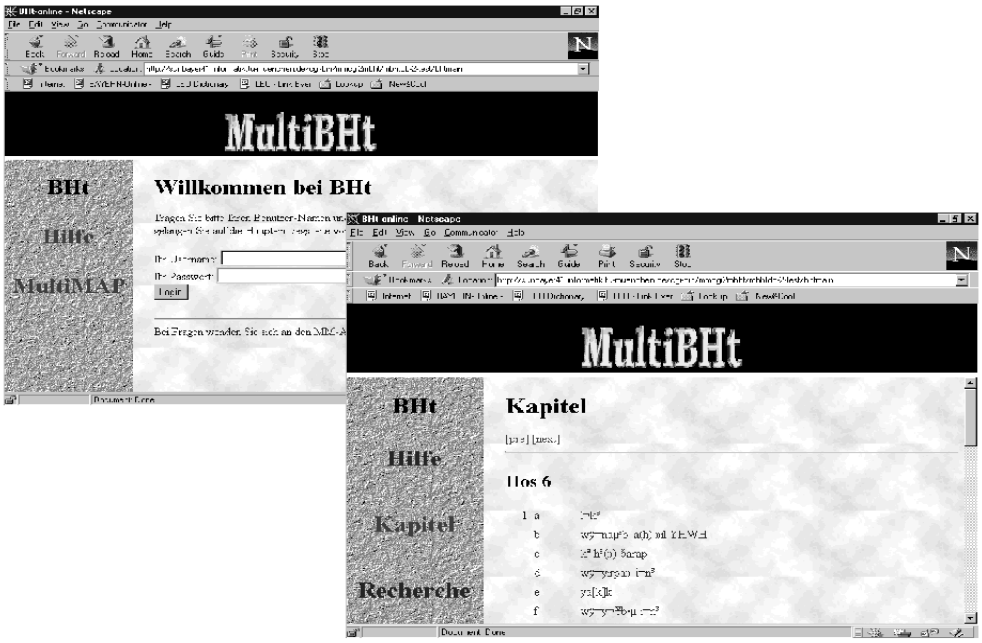

Abb. 15: MultiBHT Screendumps

Datenbank

Zur vollen Integration der BHtDB (Biblia Hebraica transcripta Datenbank), in der die sprachwissenschaftlichen Analyseergebnisse zum Alten Testament gespeichert sind, erweiterten wir das Schema von MultiBHt um das BHtDB-Schema. Es wurde in der Datenbank MultiBHtDB abgelegt. Außerdem wurde das Datenbankschema der BHtDB um die Relation *texte* erweitert. Sie enthält den jeweiligen Textkorpus eines Kapitels in HTML-Format in einem speziellen Zeichensatz (miryam). Dieser Zeichensatz wurde sowohl für windows-basierte als auch UNIX-basierte Systeme entwickelt. Für das Windows-System wurde ein True-Type Zeichensatz erstellt, womit die Skalierbarkeit auf beliebige Größen gewährleistet ist.

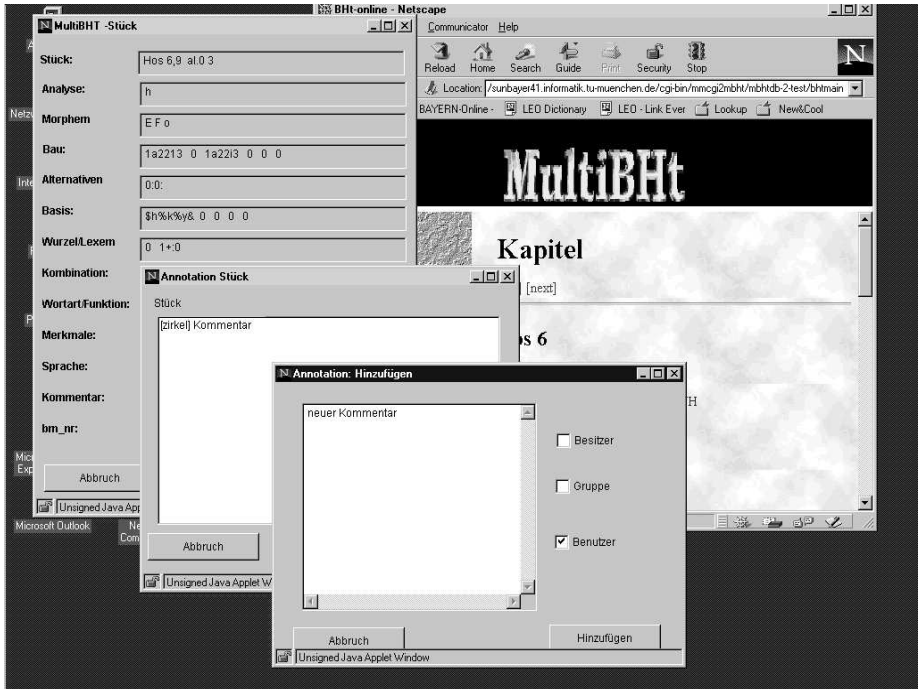

Abb. 16: MultiBHT Screendump

Client

Die Client-Seite wurde vollständig überarbeitet. Da die einzelnen althebräischen Texte in der neuen Version nicht mehr in Form von Bildern übertragen werden, sondern im Text-Format, wird das Bild-Applet durch eine HTML-Seite ersetzt. Allgemein werden jetzt zwei Frames und in den Frames CCS-Style-Sheets verwendet. Im oberen Frame ist das Applet enthalten, das zur Recherche, Annotation und Darstellung von AMOS-Bäumen (Satzstrukturen) dient.

Abbildung 15 zeigt die Start- und eine Textseite von MultiBHt. Nach der Eingabe des User-Account und Paßwort kann mit der Recherche begonnen werden. Hat der Benutzer ein Buch und Kapitel ausgewählt, bekommt er den entsprechenden Text auf der rechten Seite präsentiert. Innerhalb des Textes kann jedes Wort bzw. (Wort-) *Stück*, hierbei handelt es sich um die kleinste Informationseinheit aus dem ein Wort aufgebaut ist, ausgewählt werden. Selektiert der Benutzer ein entsprechendes Stück, wird dessen morphologische Analyse in einem Applet-Fenster dargestellt. Dies ist in Abbildung 16 links oben zu sehen. Zu jedem Stück können vom Benutzer Annotationen abgelegt werden (siehe Abbildung 16 mitte unten).

3.7 Leistungsmessungen und Tuning

3.7.1 Leistungsmessungen im Bereich des JAVA-Clients

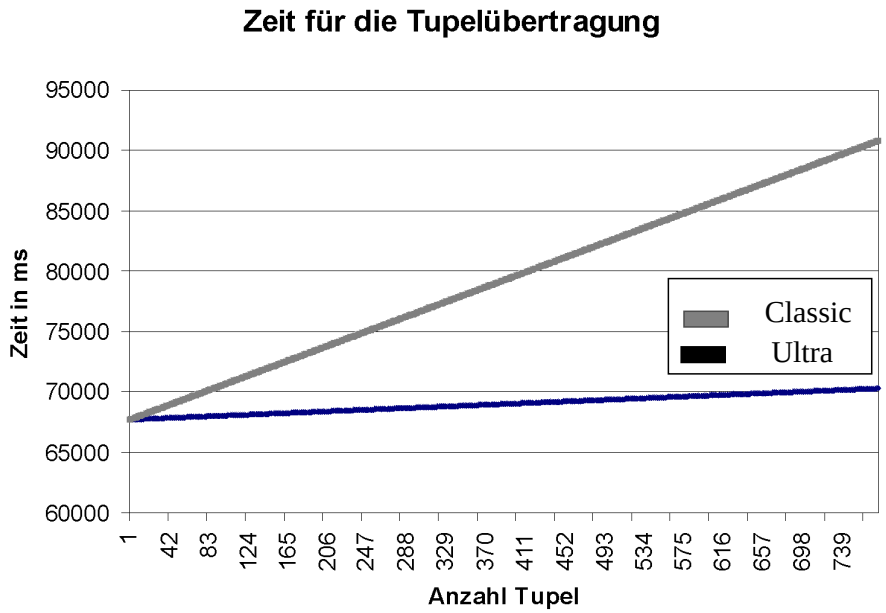

Abb. 17: Java-Netz

In Abbildung 17 sind die Ergebnisse einer Messung bei der Tupelübertragung vom DB-Client über das Java-Gateway in die virtuelle Maschine auf dem Client dargestellt. Hierbei wurde eine Sun SPARC 10 mit 33 MHZ und 96 Hauptspeicher als Datenbank-Server eingesetzt. Als Client kam eine Sun SPARC Classic mit 50 MHZ, 48 MB Hauptspeicher(sunbayer41) sowie eine Sun SPARC-Ultra-1 mit 143 MHZ und 128 Hauptspeicher(sunbayer51) zum Einsatz.

Messung:

Aus der Java-Applikation werden bei der Recherche oder Linkverfolgung mehrere SQL-Statements angestoßen. Für die gefundenen Seiten müssen all deren Bildobjekte und Linkanker berechnet werden. Für die Bildobjekte ist dazu insbesondere eine DB-Abfrage all ihrer Polygoneckpunkte nötig. Die Ergebnisse werden tupelweise in Strings verpackt und an die Java-Applikation geschickt. Die übertragung einer Ergebnismenge von ca. 800 Tupeln vom Datenbankserver über das lokale Netz ins Applet-Recherchefenster beträgt auf der Classic ca. 23 s und auf einer Ultra1 ca. 3 s. Das bedeutet, daß die Übertragung vom Netz in die virtuelle Maschine für ein Paket auf der Classic ca 28 ms und die Ultra ca 4 ms benötigt wird. Dies ist verhältnismäßig langsam, so daß wir uns entschieden haben, das Protokoll umzustellen. Die Antwort soll dann in Form eines einzigen Strings an die Java-Applikation übertragen werden.

3.7.2 Optimierung der Server-Seite

Aufgrund der Messungen wurde außerdem beschlossen die Architektur auf der Server-Seite zu modifizieren, um die Übertragungszeiten zu den Clients zu optimieren. Die Kommunikation wird von Pipes auf Sockets umgestellt. Die Aufgabe des Dispatchers wird durch den DB-Client übernommen. Die einzelnen Veränderungen gegenüber der bisherigen Architektur (vgl. Abbildung 9) sind im Abbildung 18 zusammengefaßt.

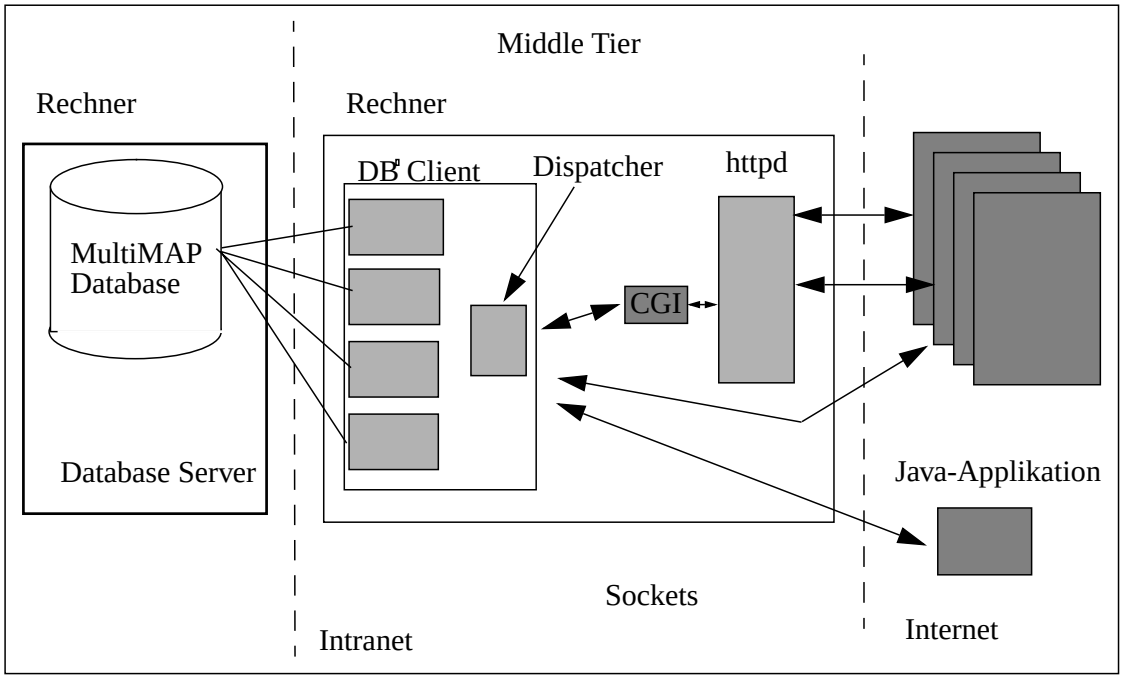

Abb. 18: Optimierte MultiMAP/2-Architektur

Die einzelnen Testläufe sowie der Einsatz beim Projektpartner in der Anwendung MultiBHT zeigten, daß das asynchrone Arbeiten durch die Umstellung auf Socket-Kommunikation die erwarteten Anforderungen erfüllt. Die Grenze der parallel arbeitenden DB-Clients - gleichzeitige Ausführung von Datenbankanfragen - wird zur Zeit durch die Datenbank-Lizenzen des Projektpartners MultiBHT auf 5 begrenzt. Durch das sofortige Freigeben der jeweiligen Lizenz nach Bearbeitung der Datenbankanfrage konnte die Anzahl asynchron arbeitender Clients mit MultiBHT-DB zusätzlich erhöht werden.

4 Entwickelte Anwendungssysteme für die Projektpartner

Dieses Kapitel beschreibt die MultiMAP-Anwendungssysteme *MultiLIB* und *MultiBHT*. Es dient gleichzeitig als Benutzermanual und enthält ausführliche Screendumps, anhand derer man einen guten Überblick über die Funktionalität der ausgelieferten Systeme erhält.

4.1 Benutzermanual MultiLIB

4.1.1 Einstieg ins System

Die Einstiegs-URL zum Bibliotheksinformationssystem MultiLIB lautet:

http://wwwmultimap.in.tum.de/MultiLIB.html

Über den Link ***MultiLIB-Start*** kommt man zu folgender Registrierungsseite (Abbildung 19). Der Einstieg ist frei (kein Passwort). Der Benutzername dient nur der Umgebungszuordnung.

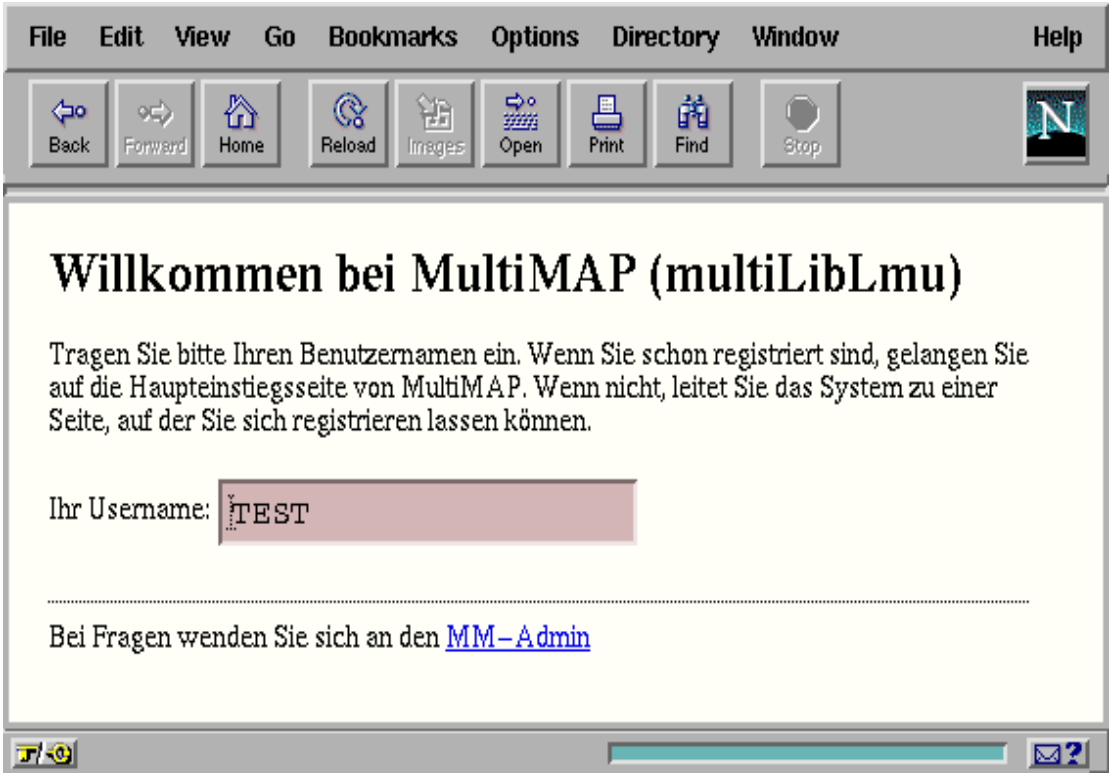

Abb. 19: Anmeldung

Liegt unter dem eingegeben Namen bereits eine Registrierung vor, so wird sofort zur MultiLIB-Startseite (Abbildung 24) weitergeleitet. Ansonsten folgen zuerst die Seiten zur Registrierung und Browsermächtigkeitseinstellung (Abbildung 20 - 23).

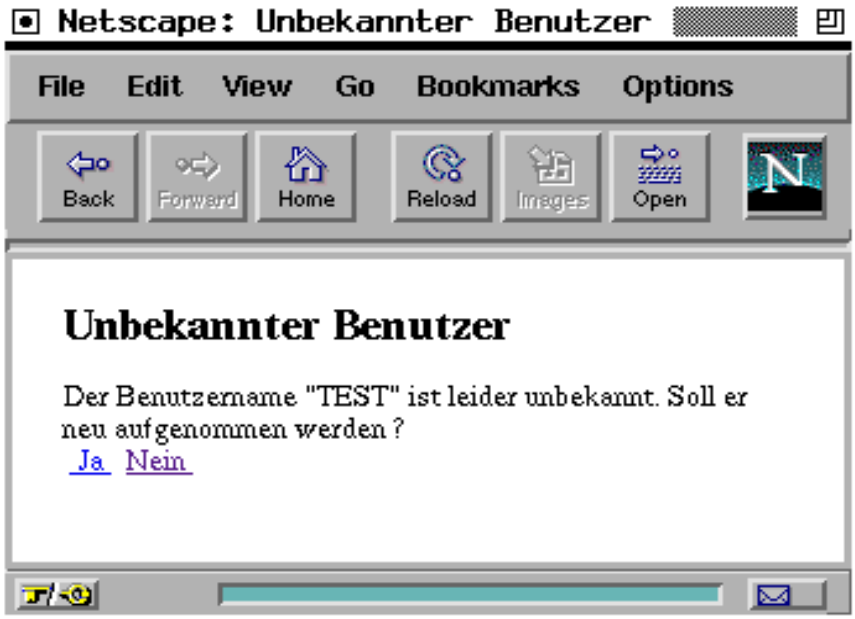

Abb. 20: Registrierung

Hier muß mit ***Ja*** bestätigt werden, damit der Benutzername neu in die Datenbank aufgenommen wird. Anschließend erhält man die Nachricht, daß der Name des Benutzers eingefügt worden ist (siehe Abbildung 21).

Abb. 21: Bestätigung der Registrierung

Mit einem Klick auf ***Weiter*** kann dann der vom Benutzer verwendete Browser ausgewählt werden, damit die Präsentation in einer optimalen Form erfolgt (siehe Abb. 22).

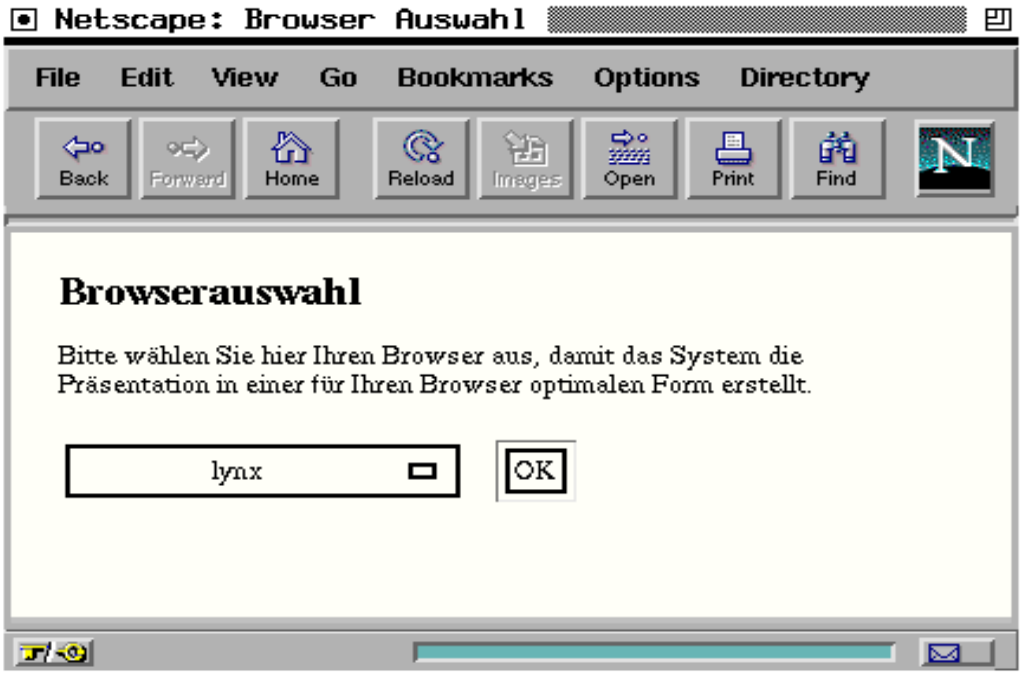

Abb. 22: Browserauswahl

Dazu öffnet man durch Klicken auf den größeren Button, der mit der Voreinstellung ***lynx*** versehen ist, ein Pulldownmenü. In diesem wählt man seinen Internetbrowser (z.B. Netscape 4.x) durch einen Mausklick aus. Dann erscheint der Name des eben selektierten Browsers auf dem Button und muß mit ***OK*** bestätigt werden.

Darauf erscheint nochmals der Name des Browsers und als zusätzliche Information die dabei vom System benutzbaren Anzeigemöglichkeiten (siehe Abbildung 23).

Die vom System zur Zeit unterstützten Browser sind in folgender Tabelle zusammengefaßt:

Name	Java	Java script	Tabellen	Bilder
Lynx	-	-	ok	-
Netscape Version 1.x	-	-	-	ok
Netscape Version 2.x	ok	-	ok	ok
Netscape Version 3.x	ok	ok	ok	ok
Netscape Communicator 4.x	ok	ok	ok	ok
Microsoft Internet Explorer 1.x -2.x	-	-	-	ok
Microsoft Internet Explorer 3.x	ok	-	ok	ok
Nokia Communicator 9000	-	-	-	ok

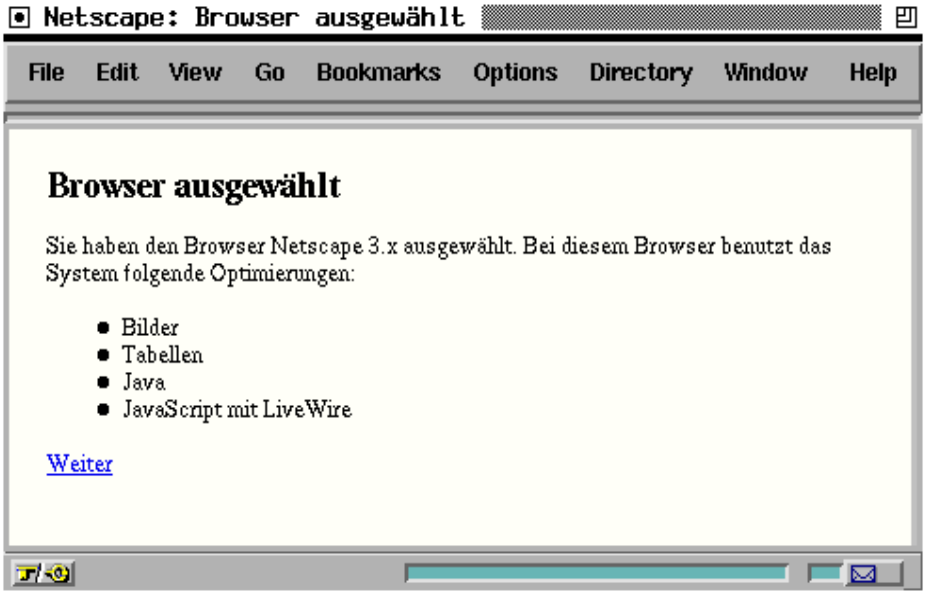

Abb. 23: Browsermächtigkeiten

Mit ***Weiter*** gelangt man zur MultiMAP-Hauptseite (siehe Abbildung 24).

Abb. 24: Startseite

Beim nächsten Einloggen wird der Benutzer sofort auf diese, dem Browser angepaßte Startseite, weitergeleitet. Hier trennen sich die Wege für den reinen Nutzer des Bibliotheksinformationssystems, der sich die Infoblöcke nur ansieht und Informationen daraus beschafft (***Haupteinstiegspunkte*** bzw. ***Alle Einstiegspunkte***), und für den Autor, der auch selbst Infoblöcke, Objekte in Infoblöcken oder Links erstellen, verändern oder löschen möchte (***Autorensystem***).
Mit dem Link ***Konfiguration*** hat man die Möglichkeit, jederzeit einen anderen Browser auszuwählen und somit zu der Seite in Abbildung 22 zurückzukehren.

4.1.2 Recherche-Komponente

Ab einer bestimmten Größe des Hypermedianetzes fällt es dem Benutzer schwer, die richtigen Einstiegspunkte aus der Fülle der Informationsblöcke herauszufinden. Diese kritische Größe ist bei der Anwendung MultiLIB mit mehr als 1000 Infoblöcken bereits erreicht. Es wird daher geraten, mit ***Haupteinstiegspunkte*** und nicht mit ***Alle Einstiegspunkte*** fortzufahren. ***Alle Einstiegspunkte*** liefert eine Auflistung aller in der Datenbank vorhandenen Informationsblöcke (siehe Abbildung 25). Im Gegensatz dazu bietet ***Haupteinstiegspunkte*** eine vom Autor getroffene Auswahl an Infoblöcken, die zur weiteren Navigation im Hypermedianetz besonders geeignet sind (siehe Abbildung 26). Durch einen Mausklick auf den gewünschten Infoblocktitel wird dieser angezeigt.

Abb. 25: Alle Einstiegspunkte

Abb. 26: Haupteinstiegspunkte

4.1.2.1 Informationsblock

Ein Informationsblock (kurz auch Infoblock genannt) besteht im wesentlichen aus einem Bild- und einem Textbereich. Als Überschrift erscheint der Titel des Infoblocks, gefolgt von Bild (falls vorhanden) und Text (siehe Abbildung 27). Im Textbereich können unterstrichen dargestellte Textlinks durch einen Mausklick verfolgt werden. Dabei kann es geschehen, daß es mehrere sich überschneidende Linkquellen gibt, denen wiederum mehrere Linkziele zugeordnet sein können. In diesem Fall erscheint ein Auswahlfenster (siehe Abbildung 28), in dem alle Zielobjekte geordnet nach Quellobjekten erscheinen und so das gewünschte Ziel durch Anklicken bestimmt werden kann.

Im Bildbereich befinden sich die drei Buttons ***Objekte anzeigen***, ***Lokale Recherche*** und ***Volltext-Recherche***. Mit ***Objekte anzeigen*** kann sich der Benutzer einen Überblick über alle maussensitiven, graphischen Objekte wie Linien, Rechtecke und Polygone verschaffen, indem ihre Umrisse und Eckpunkte dargestellt werden. Links im Bildbereich können durch Mausklicks verfolgt werden. Gibt es mehrere Linkziele, so erscheint auch hier eine Auswahlliste. Ansonsten wird direkt zum Zielobjekt verzweigt.

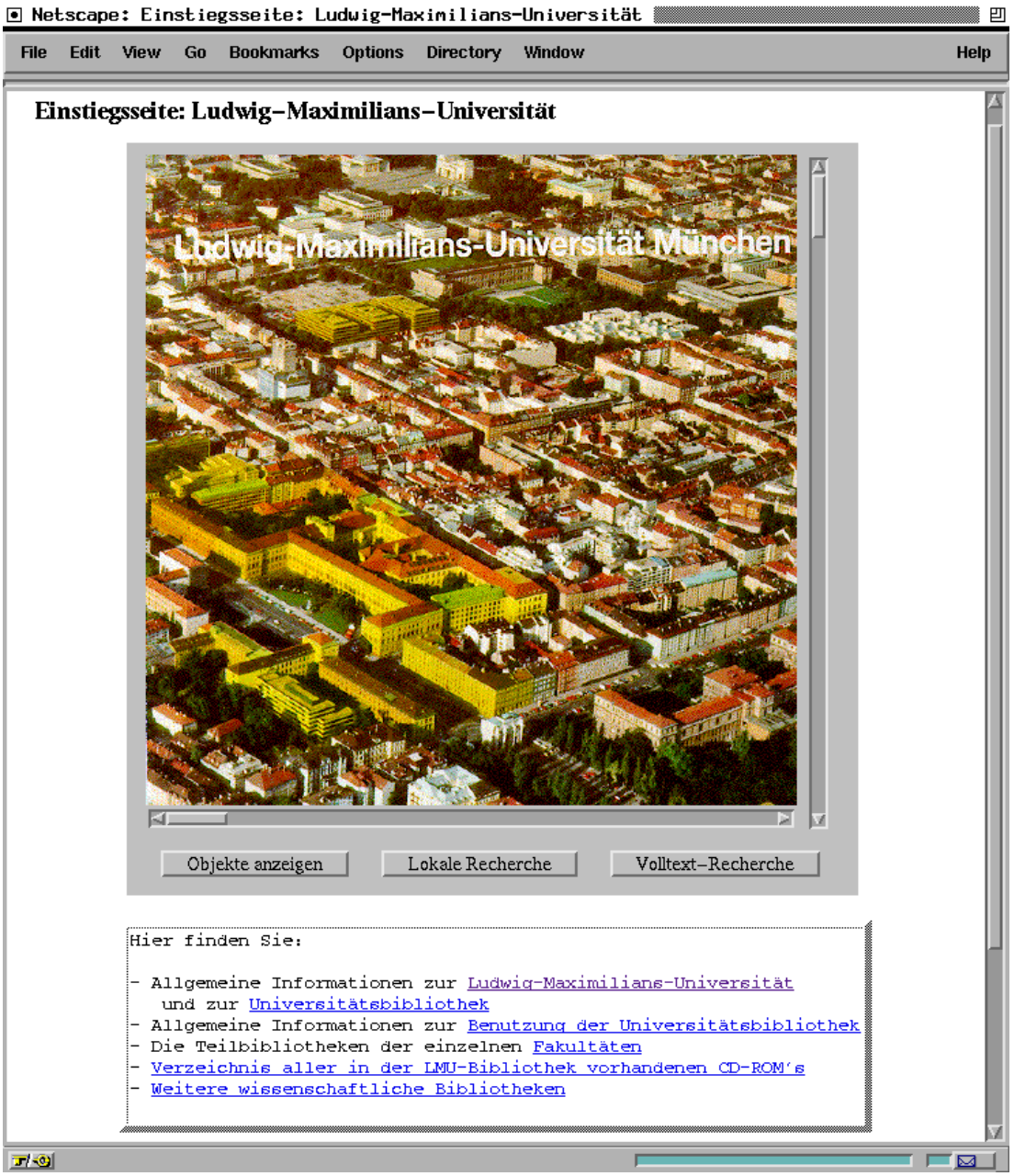

Abb. 27: Beispiel einer typischen Infoblock-Seite

Abb. 28: Auswahlfenster

4.1.2.2 Lokale Recherche

Der Pushbutton ***Lokale Recherche*** öffnet das Recherchefenster (siehe Abbildung 29). In diesem können bestimmte Bildquellobjekte gesucht und die dazugehörigen Zielobjekte angezeigt werden. Die Suche kann entweder durch Eingabe eines Suchausdrucks im Recherchefenster oder durch Mausklick im Bildbereich des Infoblocks erfolgen.

In Suchausdrücken dürfen Platzhalter und logische Verknüpfungen enthalten sein, wie sie in der Volltext-Recherche näher beschrieben werden. Eine Suche wird nach Eingabe des Suchtexts durch Drücken der Returntaste oder Anklicken von ***Recherche*** gestartet. Die Treffer werden in der *Ergebnisliste (Quellobjekte)* angezeigt. Der Benutzer kann ein selektiertes Objekt mit Hilfe von ***Quellobjekt zeigen*** sichtbar machen. Es blinkt darauf im Bildbereich des Infoblocks mehrmals kurz auf. Die Linkobjekte eines selektierten Quellobjekts erscheinen in der *Liste der Linkziele*. Mit Hilfe von ***Zielobjekt zeigen*** wird das ausgewählte Zielobjekt mehrmals blinkend im zugehörigen Informationsblock geladen.

Bei der Recherche mit der Maus werden Linkquellen in die *Ergebnisliste (Quellobjekte)* aufgenommen, sobald sie im Bildbereich des Infoblocks durch einen Mausklick erfaßt worden sind. Überschneiden sich hierbei mehrere Objekte, werden sie alle in der *Ergebnisliste (Quellobjekte)* angezeigt. ***Abbruch*** schließt das Recherchefenster.

Abb. 29: Objekt-Recherchefenster

4.1.2.3 Volltext-Recherche

Ein Klicken auf den Button ***Volltext-Recherche*** ermöglicht es dem Benutzer, sich nach Eingabe eines Suchtextes alle Informationsblöcke und Objekte, die diesen Suchtext enthalten, auflisten und anzeigen zu lassen. Die Suche erstreckt sich dabei sowohl auf Titel von Infoblöcken und Objekten als auch auf Textinformation innerhalb der Informationsblöcke (siehe Abbildung 30).

Im Eingabefeld ***Suchausdruck*** wird der Suchtext eingetragen, wobei auch Platzhalter wie “%” und “_”, “|’ logische Verknüpfungen wie “&”, und “-” und mehrere, durch ein Leerzeichen getrennte Wörter verwendet werden dürfen. Es wird hierbei jedoch nicht zwischen Groß- und Kleinschreibung unterschieden. Abstandssuche ist zur Zeit noch nicht möglich. Eine Suche wird nach Eingabe des Suchtextes durch Drücken der Returntaste oder Anklicken von ***Suche starten*** durchgeführt.

Das Prozentzeichen “%” steht für beliebig viele Zeichen und der Unterstrich “_” für genau ein Zeichen. Die beiden Platzhalter können mehrmals und an jeder Stelle im Suchbegriff verwendet werden.

Die Zeichen “&”, “|” und “-” stehen für die logischen Verknüpfungen UND, ODER und NICHT. Sie werden zwischen zwei Suchausdrücken eingefügt, wobei vor und nach einem solchen Sonderzeichen ein Leerzeichen stehen muß. Die Auswertung erfolgt dabei in der Reihenfolge UND, ODER und zuletzt NICHT.

Im Ergebnisfeld unter dem Suchausdruck wird nach einer erfolgreichen Suche die Menge der Treffer in der Form Dokument-Typ, Dokument-ID und Dokument-Name ange-

zeigt. Die Kürzel “I” und “O” stehen für die Dokumentarten “ganzer Infoblock” und “Objekt”. Ein Eintrag kann entweder durch Doppelklick oder Markieren und anschließendes Klicken auf ***Ergebnis zeigen*** ausgewählt werden. Objekte erscheinen im dazugehörigen Infoblock.

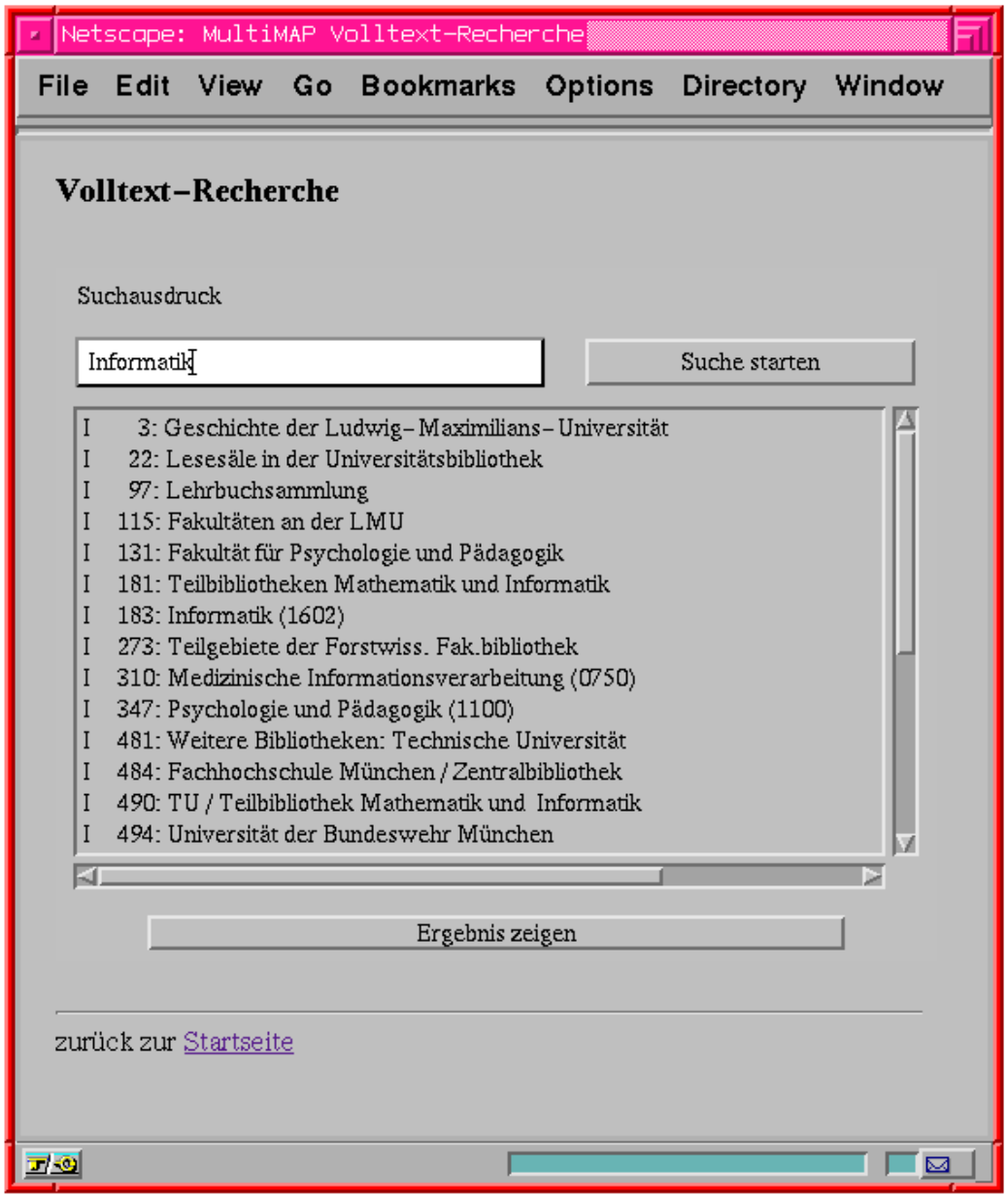

Abb. 30: Volltext-Recherchefenster

Am Ende jeder Infoblockseite befindet sich ein zusätzlicher Link zurück zur ***Startseite***. Durch ihn besteht die Möglichkeit, jederzeit zur Seite in Abbildung 24 zurückzukehren.

4.1.3 Autorenkomponente

Mit der Autorenkomponente hat der privilegierte Benutzer die Möglichkeit, Objekte in Informationsblöcken und Links neu zu definieren, zu ändern oder zu löschen. Um ganze Infoblöcke und Bilder erstellen, ändern oder löschen zu können, muß aufgrund von Java-Sicherheitsbeschränkungen auf den lokal verfügbaren Infoblock-Editor zurückgegriffen werden, der im Punkt 4.1.4 beschrieben wird.

Bei Aufruf des Autorensystems erscheinen die beiden Buttons ***Einstiegspunkte*** und ***Links***. Bei einem Klick auf ***Einstiegspunkte*** kann man ähnlich wie bei der Recherche-Komponente zwischen ***Haupteinstiegspunkte*** und ***alle Informationsblöcke*** wählen (siehe Abbildung 31).

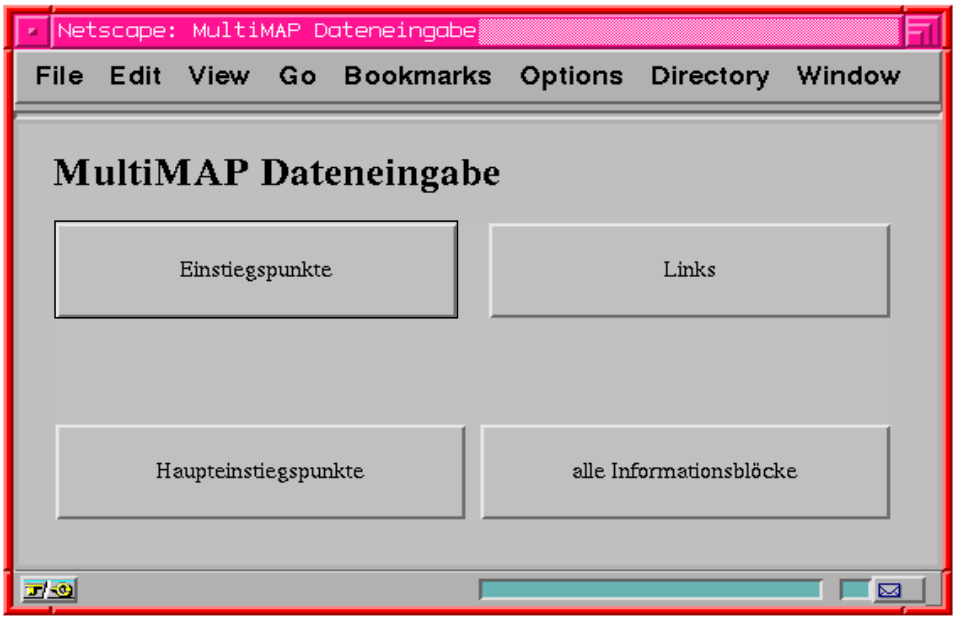

Abb. 31: Autorenkomponente: Infoblockbearbeitung

Es öffnet sich darauf ein Fenster mit allen Infoblöcken bzw. den Haupteinstiegspunkten, aus denen der gewünschte Infoblock durch Doppelklick oder Selektieren und **OK** aufgerufen wird (siehe Abbildung 32).

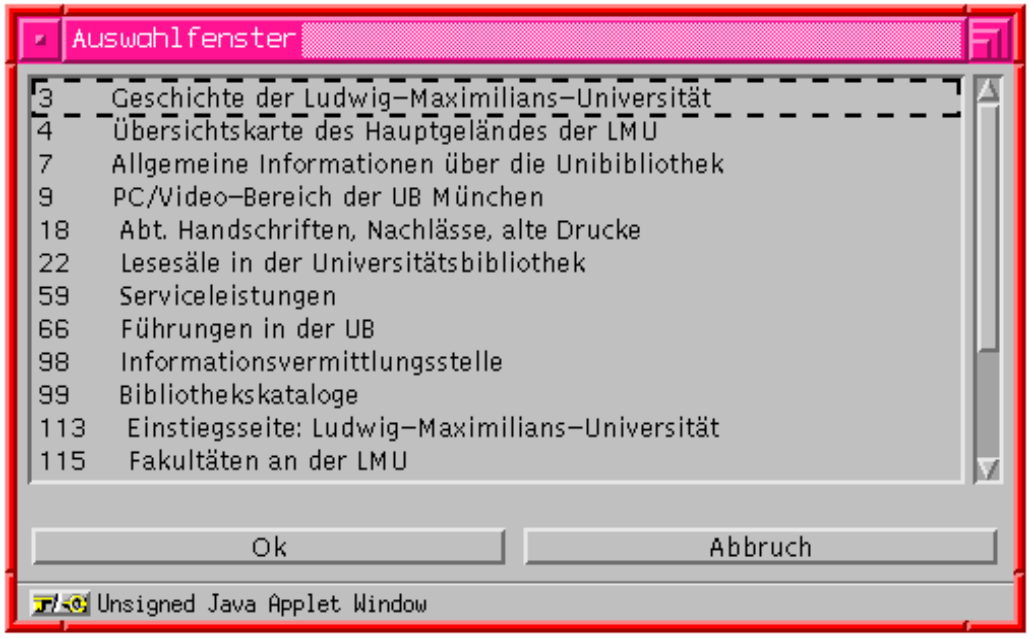

Abb. 32: Auswahl eines Infoblocks

4.1.3.1 Neues Objekt definieren

Im Autorensystem können Links in Infoblöcken nicht verfolgt werden. Textlinks sind hier in HTML-Quellcode dargestellt. Ein Objekt ist einem Infoblock zugeordnet und kann als Linkanker dienen. Um ein neues Objekt zu definieren, wählt man im Menü ***Objekte*** den Punkt ***Neues Objekt definieren*** und es erscheint das Fenster zur Objektdefinition (siehe Abbildung 33).

Die ObjektID wird automatisch festgelegt und kann nicht verändert werden. Der Objekttyp wird durch Auswahl eines der Buttons ***Rechteck***, ***Polygon***, ***Linie*** (diese Bildobjekte stehen allerdings nur zur Verfügung, falls ein Bildbereich im Infoblock existiert), ***Text*** und ***ganzer Infoblock*** bestimmt. Der Button ***ganzer Infoblock*** erscheint insensitiv, falls das Objekt bereits definiert worden ist. In diesem Fall kann der ganze Infoblock im Menüpunkt ***Objekt auswählen*** als Objekt ausgewählt und anschließend zur Linkdefinition weiterverwendet werden.

Textobjekte

Bei der Definition eines Textobjekts oder der Angabe eines ganzen Infoblocks wird als Objektname automatisch der vom Autor markierte Text bzw. der Titel des Infoblocks eingetragen. Bei der Definition eines Textobjekts wird der ausgewählte Text mit gedrückter linker Maustaste markiert und anschließend durch einen Klick auf ***Linkquelle definieren*** oder ***Linkziel definieren*** ein entsprechendes HTML-Konstrukt eingefügt. Möchte man einen solchen Vorgang rückgängig machen, so wird dies durch Markieren des Texts inklusive des HTML-Konstrukts und Klicken auf ***Selektion rückgängig machen*** erreicht.

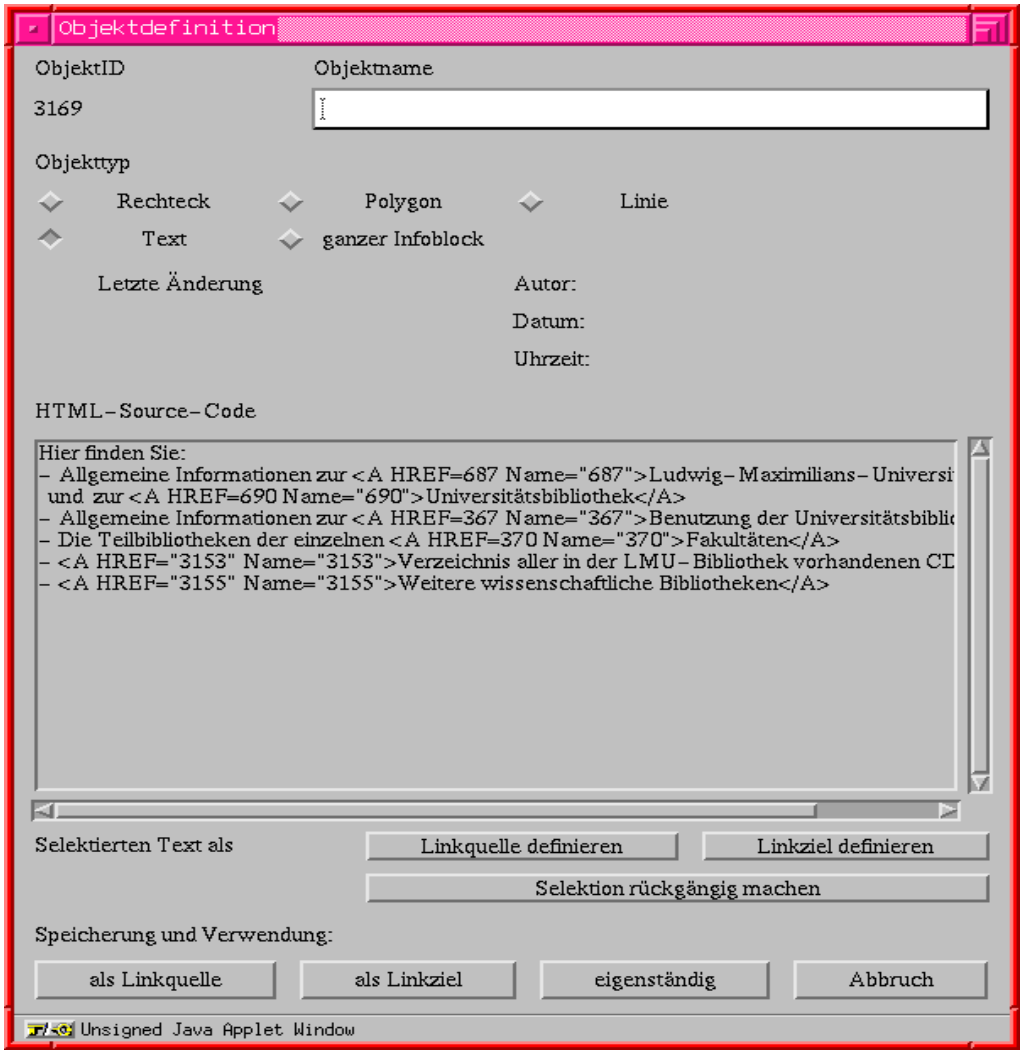

Abb. 33: Objektdefinition (Beispiel Textobjekt)

Bildobjekte

Soll ein graphisches Objekt in einem Bild (z.B. als Linkanker) definiert werden, so muß der Autor zunächst einen Objektnamen festlegen. Es erscheint ein Fenster zur Eingabe der Eckpunkte (siehe Abbildung 34). Ein Punkt kann entweder durch einen Mausklick im Bildbereich des Infoblocks oder durch Angabe seiner Koordinaten und den Button ***Eintragen*** bestimmt werden. Bei einem Rechteck müssen zwei sich diagonal gegenüberliegende Punkte angegeben werden. Ein Polygon oder eine Linie kann aus beliebig vielen Punkten bestehen, die automatisch miteinander verbunden werden. Eine solche Eingabe wird durch einen Klick auf den Button ***Stop*** beendet. Durch Anklicken eines Punkts im Bildbereich oder in der Punktliste wird ein Punkt aktiviert. Ein aktivierter Punkt kann durch **Entfernen** gelöscht werden. In einem Polygon oder einer Linie besteht die Möglichkeit, neue Punkte zum zuletzt definierten Punkt (unterster Punkt der Punktliste) hinzuzufügen. Der neue Punkt kann wiederum entweder durch direkte Koordinatenangabe und den Button **Eintragen** oder durch Klicken im Bildbereich bestimmt

werden. Die Koordinaten eines Punkts können durch Verschieben bei gedrückter linker Maustaste im Bildbereich oder durch Aktivieren des Punkts und neue Eingabe seiner Koordinaten mit anschließender Bestätigung durch ***Ändern*** verändert werden.

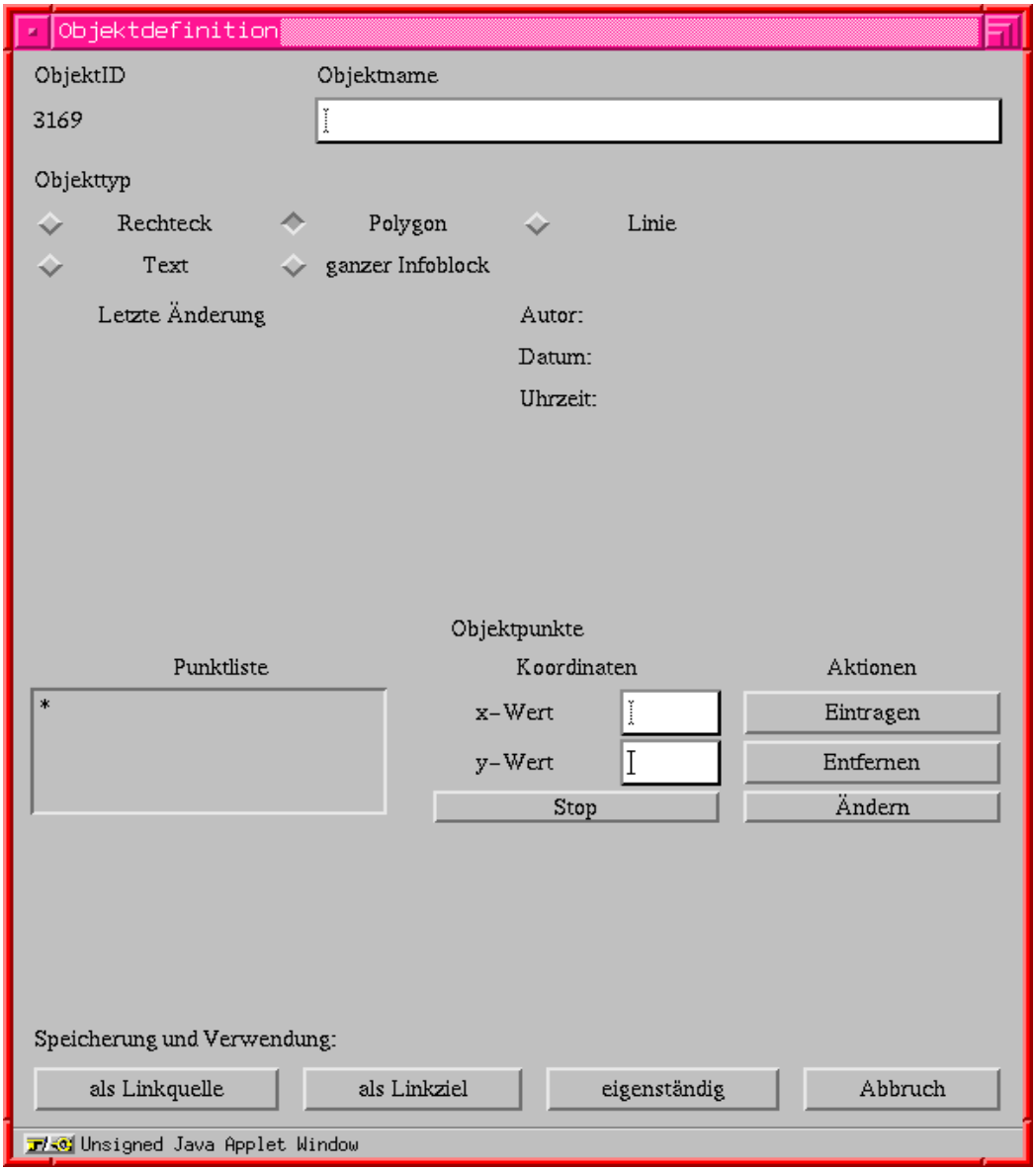

Abb. 34: Definition Bildobjekt

Objekt speichern

Ein Bildobjekt kann ***eigenständig***, ***als Linkquelle*** oder ***als Linkziel*** abgespeichert werden. Bei einem Textobjekt kommt es auf die Definition an. Ist es als Linkquelle definiert worden, so kann es auch nur als solche gespeichert werden. Dies wird durch Insensitivierung der anderen Buttons erreicht. Gleiches gilt für Linkziel. Ein eigenständiges Speichern ist bei einem Textobjekt nicht möglich.

Ein Klick auf ***als Linkquelle*** oder ***als Linkziel*** öffnet das Linkdefinitionsfenster (siehe Abbildung 39). Ist in diesem Fenster vom Autor schon vorher festgelegt worden, als welcher Linkbestandteil das neu definierte Objekt verwendet werden soll, sind nur die entsprechenden Buttons sensitiv.

4.1.3.2 Menü Objekte

Objekt ändern

Um ein bereits definiertes Objekt zu ändern, wählt man im Menü Objekte eines Infoblocks den Punkt ***Objekt ändern***. Es erscheint ein Fenster mit einer Liste aller Text- und Bildobjekte. Ein Eintrag wird durch Doppelklick oder Markieren und ***OK*** ausgewählt. Darauf öffnet sich das Fenster mit den Objektdaten (siehe Abbildung 33/34). Bei der Auswahl eines Bildobjekts wird dieses im Bildbereich des Infoblocks angezeigt. Das Ändern der Objektdaten erfolgt analog zum Definieren von Objekten, wie es im Punkt ***Neues Objekt definieren*** beschrieben worden ist. Mit dem Button ***eigenständig*** werden die Änderungen gespeichert.

Eine andere Möglichkeit zum Fenster zur Änderung der Objektdaten zu gelangen, besteht bei Bildobjekten darin, diese mit Hilfe von Mausklicks im Bildbereich des Infoblocks auszuwählen (siehe Menüpunkt ***Objekt-Recherche***).

Objekt auswählen

Dieser Menüpunkt dient zur Auswahl eines schon definierten Objekts, welches bei der Definition eines Links verwendet werden soll (siehe Abbildung 35). Ein Objekt kann als Linkquelle oder -ziel ausgewählt werden (falls dies nicht schon vorher festgelegt worden ist). Darauf öffnet sich das Fenster zur Linkdefinition mit dem an entsprechender Stelle eingetragenen Objekt. Handelt es sich um eine Linkquelle, werden außerdem alle dazugehörigen Linkziele aufgelistet.

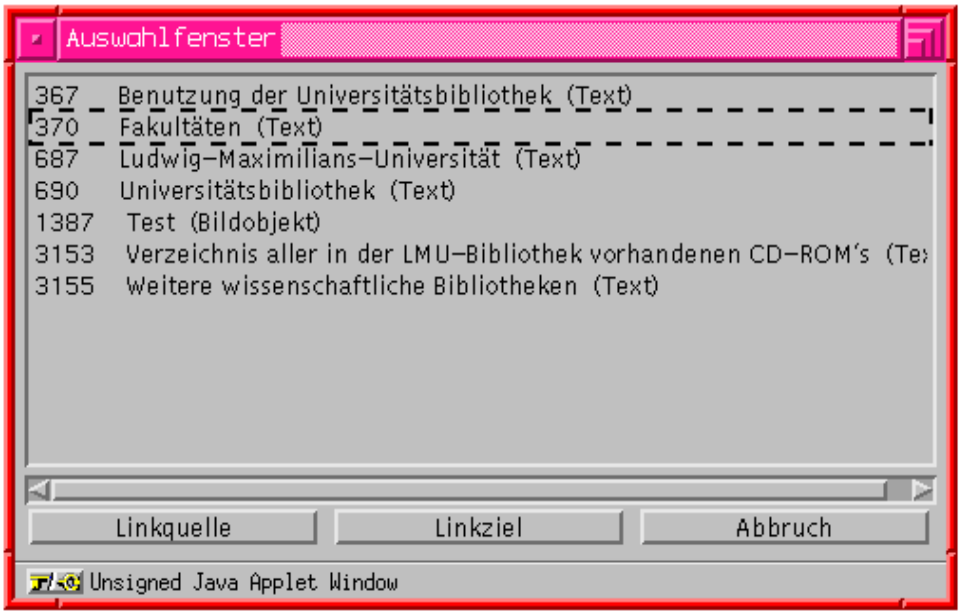

Abb. 35: Objektauswahl

(Bild-)Objekte anzeigen

Dieser Menüpunkt ermöglicht einen Überblick über bereits definierte Bildobjekte in diesem Infoblock. Sie werden im Bildbereich mit markierten Umrissen dargestellt.

Objekt löschen

Es erscheint ein Fenster, in dem das zu löschende Objekt ausgewählt wird (siehe Abbildung 36). Ein Objekt kann jedoch nur gelöscht werden, wenn keine Verweise mehr darauf existieren. Ist dies der Fall, so wird das Objekt zusammen mit allen von ihm ausgehenden Links entfernt. Andernfalls erscheint eine Fehlermeldung, daß noch Verweise auf das Objekt existieren, und es verbleibt weiterhin in der Datenbank.

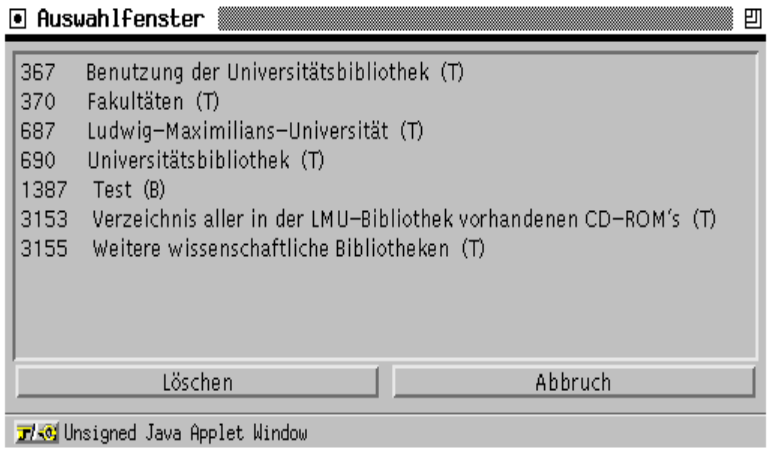

Abb. 36: Objekt löschen

(Objekt-)Recherche

Bei Aufruf dieses Menüpunkts öffnet sich ein Recherchefenster (siehe Abbildung 37). Ein Mausklick im Bildbereich des Infoblocks bestimmt einen Punkt, worauf im Fenster die Namen aller Bildobjekte erscheinen, die diese Punktkoordinaten enthalten. Die Objekte können anschließend, wie im Menüpunkt ***Objekt ändern*** bereits erklärt, ausgewählt und geändert werden.

Abb. 37: (Objekt-)Recherche

4.1.3.3 Link erstellen

Klickt man nach Aufruf des Autorensystems auf ***Links***, so erhält man die drei zusätzlichen Buttons ***Link erstellen***, ***Link ändern*** und ***Link löschen*** (siehe Abbildung 38).

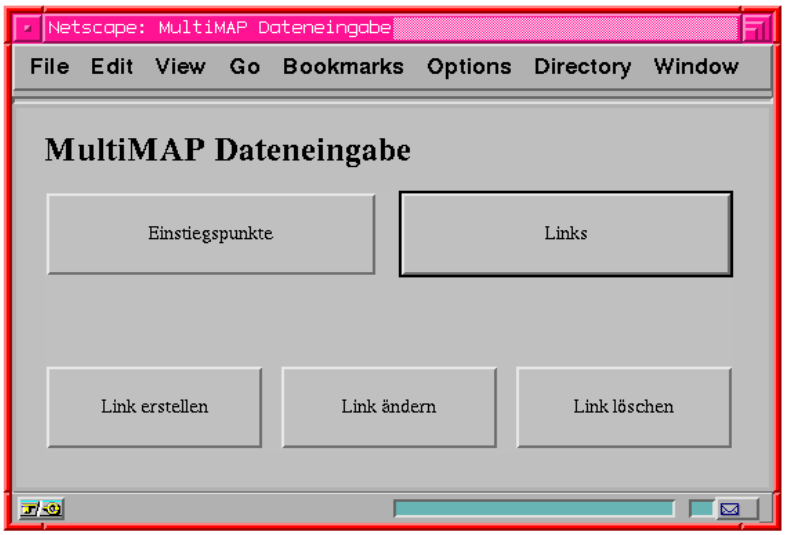

Abb. 38: Autorenkomponente: Linkbearbeitung

Mit Hilfe von ***Link erstellen*** können bisher unabhängig voneinander gespeicherte Infoblöcke miteinander verbunden werden. Es erscheint das Fenster zur Linkdefinition (siehe Abbildung 39).

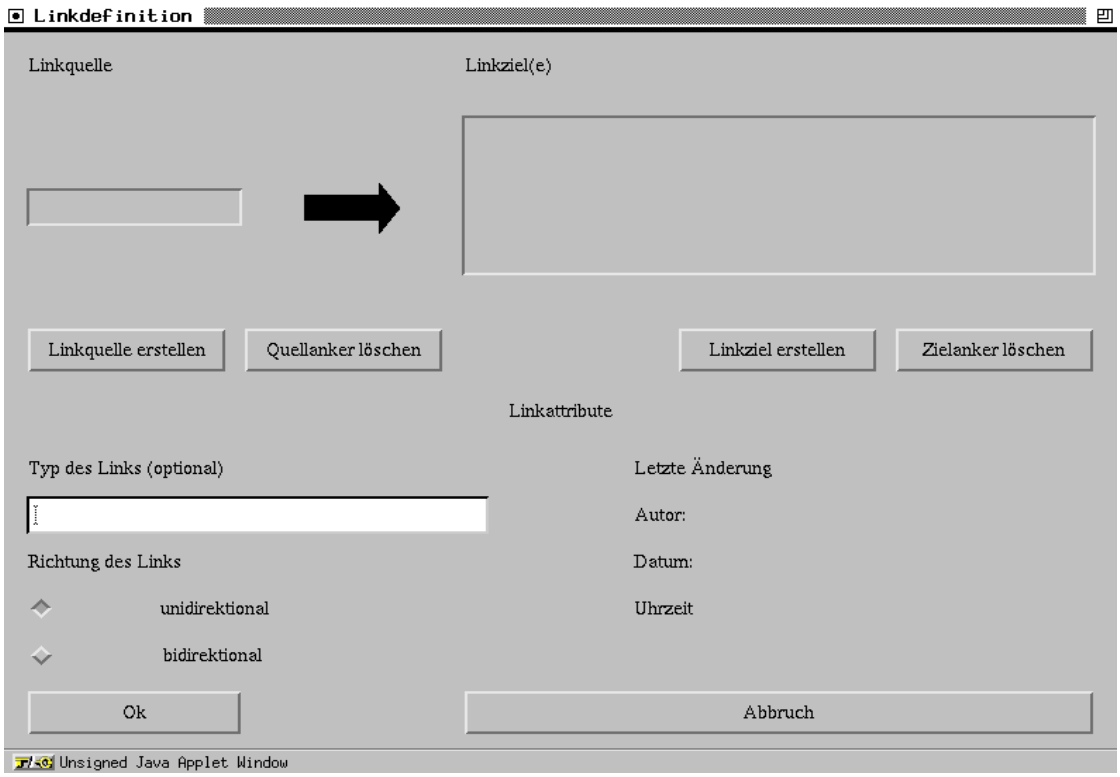

Abb. 39: Linkdefinition

Hier kann durch einen Klick auf ***Linkquelle erstellen*** bzw. ***Linkziel erstellen*** die Verwendung eines Objekts festgelegt werden. Bei der Neudefinition oder Auswahl des nächsten Objekts sind nur die zutreffenden Buttons sensitiv und das Objekt wird automatisch an die entsprechende Stelle im Linkdefinitionsfenster eingetragen. Handelt es sich um eine Linkquelle, werden außerdem alle dazugehörigen Linkziele aufgelistet. Der Verwendungszweck eines Objekts kann jedoch auch direkt bei der Definition (siehe ***Neues Objekt definieren***, Abbildung 33/34) oder der Auswahl eines Objekts (siehe ***Objekt auswählen***, Abbildung 35) bestimmt werden. Dies ist die weniger aufwendige Variante, da sich das Fenster zur Definition eines Links dann selbständig öffnet.

Unter ***Letzte Änderung*** sind der Zeitpunkt der Erstellung und der Autor des Links bezüglich des markierten Linkziels zu sehen.

Mit ***Quellanker löschen*** und ***Zielanker löschen*** kann eine Linkquelle bzw. ein in der Liste markiertes Linkziel entfernt werden. Dabei ist darauf zu achten, daß stets eine Linkquelle und mindestens ein Linkziel angegeben sind, da sich der Link sonst nicht abspeichern läßt.

Der Typ eines Links ist optional und stellt lediglich eine Hilfe für Autoren zur näheren Beschreibung des Links dar. Er wird momentan nicht weiter verwendet oder ausgewertet und ist für zukünftige Erweiterungen freigehalten.

Die Richtung eines Links kann entweder unidirektional oder bidirektional sein. Unidirektionale Links können nur in eine Richtung, von der Linkquelle zum Linkziel verfolgt

werden. Bei bidirektionalen Links besteht zusätzlich die Möglichkeit, den Link vom ursprünglichen Ziel zur Quelle zurück zu verfolgen.

Mit einem Klick auf ***OK*** wird ein Link in der Datenbank gespeichert.

4.1.3.4 Link ändern und löschen

Der Button ***Link ändern*** öffnet ein Fenster mit einer Liste aller schon erstellten Links. Ein Link wird durch seine Linkquelle zusammen mit dem Namen des zugehörigen Infoblocks repräsentiert. Ein Eintrag wird durch Doppelklick oder Markieren und einen Klick auf ***OK*** ausgewählt (siehe Abbildung 40). Darauf öffnet sich das Linkdefinitionsfenster (siehe Abbildung 39). Hier können nun, wie bereits im Punkt ***Link erstellen*** beschrieben, Linkquelle und -ziele gelöscht und neu hinzugefügt werden. Mit einem Klick auf ***OK*** werden die Änderungen in der Datenbank abgespeichert.

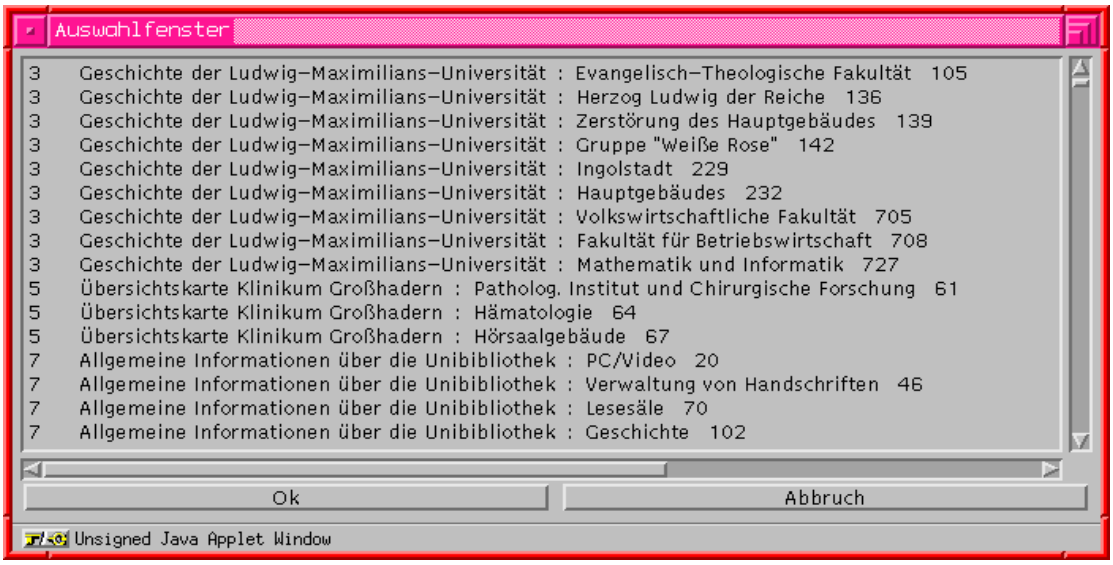

Abb. 40: Linkauswahl

Mit Hilfe von ***Link löschen*** erscheint eine Auswahl aller vorhandenen Links (siehe Abbildung 40). Ein Doppelklick oder Markieren und Drücken des Buttons ***OK*** öffnet ein Fenster, ähnlich zum Linkdefinitionsfenster (siehe Abbildung 39), mit den entsprechenden Eintragungen. Ein Klicken auf ***Löschen*** entfernt den Link aus der Datenbank.
Textlinks können allerdings nicht auf diese Weise gelöscht werden. Sie müssen zusammen mit Quellobjekt und Zielobjekten direkt im Infoblock entfernt werden, wie bereits in Punkt ***Objekt löschen*** beschrieben.

4.1.4 Infoblock-Editor

Aufgrund von Sicherheitsbeschränkungen von Java ist es bisher nicht möglich, mit der WWW-Version von MultiLIB vollständige Informationsblöcke vom Rechner eines Benutzers zum Datenbank-Client zu übertragen und sie dort in der Datenbank zu speichern. Daher muß zum Erstellen, Löschen und Ändern von Informationsblöcken auf den, auf der Server-Maschine lokal verfügbaren Infoblock-Editor zurückgegriffen werden. Dieser wird im Verzeichnis *.../java/classes/app* mit dem Aufruf *java InfoblockEditor -user <Benutzername>* gestartet (siehe Abbildung 41).

Abb. 41: Infoblock-Editor

4.1.4.1 Infoblock erstellen

Der Punkt ***Infoblock erstellen*** im Menü ***Infoblock*** öffnet das Infoblock-Definitionsfenster (siehe Abbildung 42). Hier muß zunächst der Titel des Infoblocks im dafür vorgesehenen Feld eingetragen werden. Soll der Informationsblock ein Bild enthalten, so kann dies entweder ein bereits in der Datenbank abgelegtes oder ein neues Bild aus einer Datei sein. Ein gespeichertes Bild wird durch Mausklick aus der Liste der in der Datenbank abgelegten Bilder ausgewählt. Der Button ***Durchsuchen*** ermöglicht mit Hilfe einer Dialogbox das Navigieren durch das Dateisystem. Der Pfad kann durch Doppelklick auf die Bilddatei, Markieren der Datei und einen Klick auf ***Open*** oder direkt per Hand eingetragen werden. Das Bild muß im TIFF-Format vorliegen. Das Medium Ton kommt beim Bibliotheksinformationssystem MultiLIB, auf Wunsch des Projektpartners Ludwig-Maximilian-Universität München, nicht zum Einsatz. Das Einlesen einer Tondatei ist analog zur Bilddatei. Text kann entweder direkt eingegeben oder ebenfalls durch Auswahl einer Textdatei mit Hilfe von ***Durchsuchen*** in den Infoblock eingebracht werden. Bleibt ein Feld des Definitionsfensters leer, so wird dieses Medium nicht verwendet. Aktivieren des Buttons ***Infoblock als Einstiegspunkt speichern*** führt zur Aufnahme des Informationsblocks in die Liste der Haupteinstiegspunkte (siehe Abbildung 26). Mit ***OK*** wird der Infoblock persistent in der Datenbank gespeichert.

Abb. 42: Infoblock-Definitionsfenster

4.1.4.2 Infoblock ändern

Nach einem Mausklick auf ***Infoblock ändern*** erscheint ein Auswahlfenster mit *IDs* und Titeln aller in der Datenbank gespeicherten Informationsblöcke (siehe Abbildung 43). Aus dieser Liste wird ein Infoblock mittels Doppelklick oder Markieren und den Button ***Ändern*** ausgewählt. Darauf erscheint erneut das Infoblock-Definitionsfenster (siehe Abbildung 44), samt seiner bisherigen Eintragungen. Diese Einträge können nun analog zur Erstellung eines Infoblocks verändert werden. Ein Mausklick auf ***OK*** speichert die Änderungen des Informationsblocks in der Datenbank.
Hierbei ist jedoch zu beachten, daß die *HREFs* nicht direkt im Text geändert oder gelöscht werden dürfen, da sie sie intern in Systemtabellen eingetragen sind und vom System automatisch verwaltet werden.

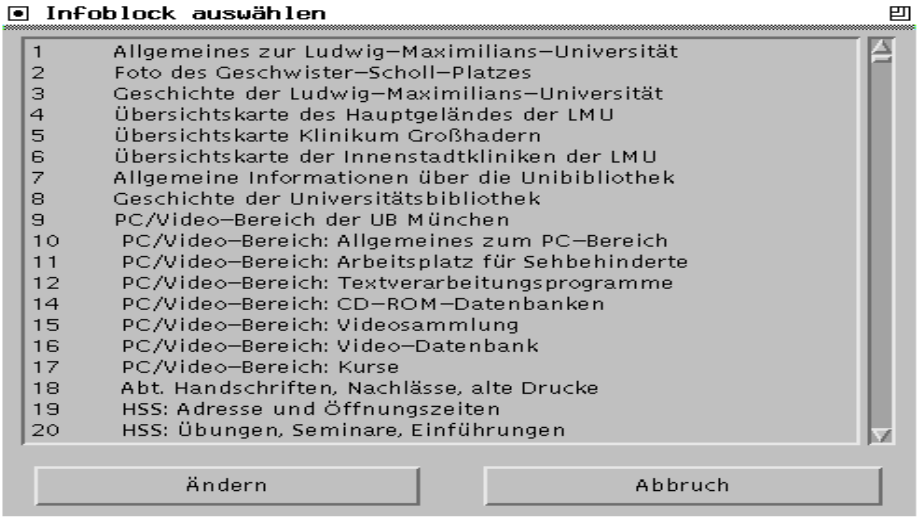

Abb. 43: Infoblockauswahl - Ändern

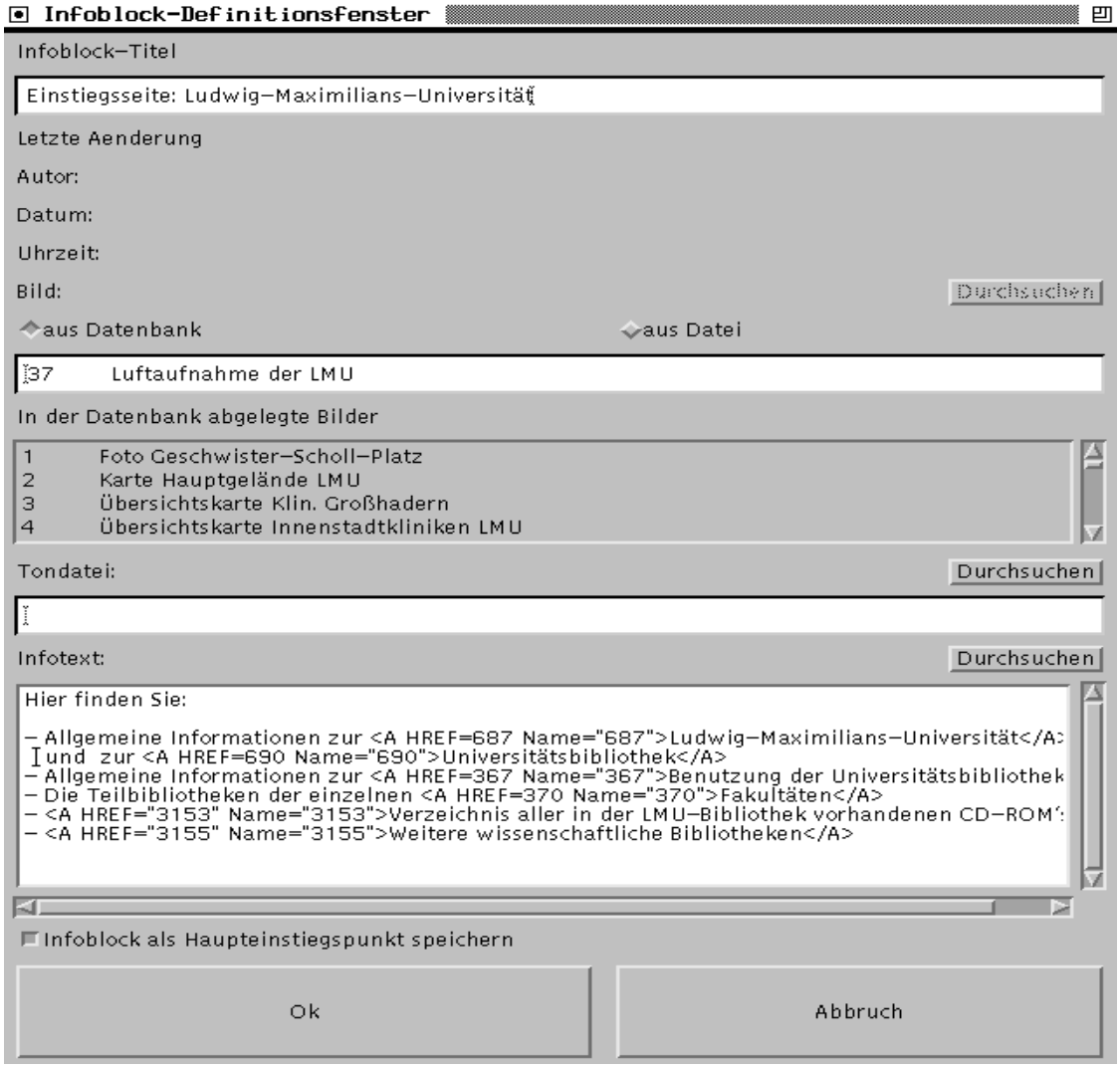

Abb. 44: Infoblock-Definitionsfenster

4.1.4.3 Infoblock löschen

Soll ein Infoblock aus der Datenbank entfernt werden, wird zunächst überprüft, ob eines oder mehrere seiner Objekte als Linkziele dienen. Dieser Test erfolgt automatisch im Menüpunkt ***Informationsblock löschen***, wenn aus dem Auswahlfenster (siehe Abbildung 45) ein Infoblock selektiert und der Button ***Löschen*** betätigt wird. Gibt es noch Verweise auf den zu löschenden Informationsblock, so erscheint eine entsprechende Fehlermeldung. Damit ein solcher Informationsblock entfernt werden kann, müssen zuerst die auf ihn verweisenden Links über den WWW-Browser (siehe ***Link löschen***, Abbildung 39) gelöscht werden. Existieren keine Verweise auf den Infoblock, so wird er zusammen mit seinen Objekten und daraus hervorgehenden Links aus der Datenbank entfernt.

Abb. 45: Infoblockauswahl - Löschen

4.1.4.4 Bild erstellen

Bilder können in der Datenbank abgespeichert und erst später in einen Informationsblock eingebettet werden. Ein einmal in der Datenbank abgelegtes Bild kann mehrmals in unterschiedlichen Infoblöcken Verwendung finden. Dies verringert den Speicherplatzbedarf erheblich. Im Bild-Definitionsfenster (siehe Abbildung 46) muß zunächst ein Bildtitel eingetragen werden. Darauf wird der Pfad der Bilddatei entweder direkt oder mit Hilfe von **Durchsuchen** (siehe Menüpunkt **Infoblock erstellen**) bestimmt. Das Bild muß im TIFF-Format vorliegen. Zusätzlich besteht die Möglichkeit, einen Text mit Informationen zum Bild einzugeben. Mit **OK** wird das Bild zusammen mit Titel und Informationstext gespeichert.

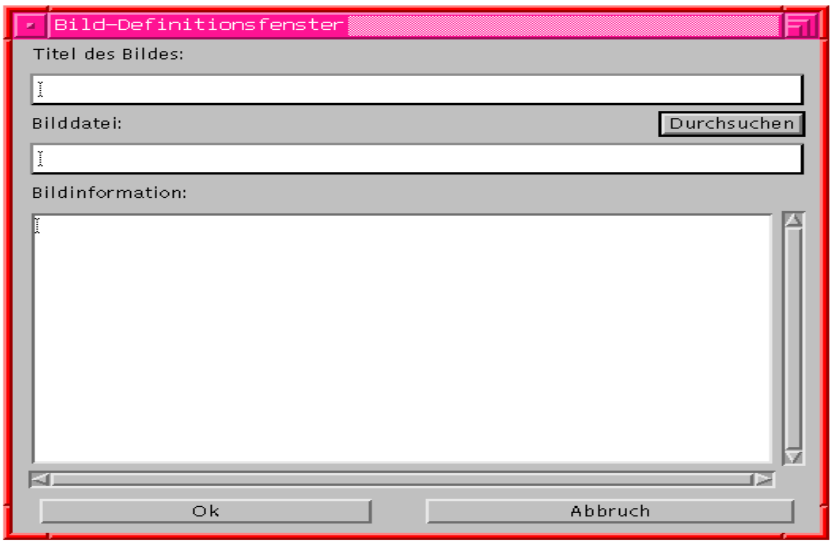

Abb. 46: Bild-Definitionsfenster

4.1.4.5 Bild löschen

Ein Bild kann erst aus der Datenbank entfernt werden, wenn es in keinem Informationsblock mehr verwendet wird. Bei der Löschung eines Infoblocks wird nur der interne Verwendungszähler des Bildes dekrementiert. Das Bild bleibt in der Datenbank gespeichert. Ein Bild kann gelöscht werden, indem aus dem Auswahlfenster (siehe Abbildung 47) ein Bildtitel selektiert und der Button ***Löschen*** betätigt wird. Hierbei wird automatisch überprüft, ob der interne Verwendungszähler Null ist. Wird das Bild noch in einem Infoblock benötigt, erscheint die entsprechende Fehlermeldung. Andernfalls wird das Bild mit Titel und Informationstext aus der Datenbank gelöscht.

Abb. 47: Bildauswahl - Löschen

Der Infoblock-Editor wird geschlossen, indem man den Menüpunkt ***Datei/Beenden*** wählt.

4.2 Benutzermanual MultiBHT

Die bei weitem wichtigste und erfolgreichste Anwendung von MultiMAP ist MultiBHT, ein System zur Unterstützung der sprach- und literaturwissenschaftlichen Arbeit an althebräischen Texten auf der Basis der *Biblia Hebraica transcripta* (BHt). MultiBHT ist bei unseren Projektpartnern Prof. Dr. Wolfgang Richter und Dr. Christian Riepl an der Philosophischen Fakultät für Altertumskunde und Kulturwissenschaften der Ludwig-Maximilians-Universität München installiert und seit nunmehr über einem Jahr in Betrieb. Der Zugriff erfolgt über

http://www.fak12.uni-muenchen.de/arf/bht/mbht.html

Die Grunddaten sind urheberrechtlich und verlagsrechtlich geschützt, daher ist ein Login und Passwort nötig. Diese können auf der obigen Seite oder bei Hr. Dr. Christian Riepl (email: Christian.Riepl@arf.fak12.uni-muenchen.de) angefordert werden.

Die einzelnen Java-Komponenten sind mit JDK 1.6 implementiert worden und laufen zur Zeit mit Netscape 4.5. Wir empfehlen als Client einen PC, da Netscape 4.5. unter UNIX noch nicht die volle Java-Mächtigkeit unterstützt (trotz Zusicherung). Einzelne Applets werden dort nicht angezeigt.

Zur korrekten Anzeige der althebräischen Texte ist ein eigener Zeichensatz (*Miryam*) nötig. Dieser wird auf obiger Einführungsseite bereitgestellt und kann von dort heruntergeladen werden. Die folgenden Abschnitte sind etwas ausführlicher und dienen gleichzeitig als Benutzermanual.

4.2.1 Anmeldung und Einstiegsseite

Über den Link ***Startseite MultiBHT*** auf obiger Einstiegsseite kommt man zur Registrierungsseite (Abb. 48). Am linken Rand sind Links zur ausführlichen Schemabeschreibung der gesamten darunterliegenden Inhaltsdatenbank BHtDB (Link: ***BHt***), zu einer Online-Hilfe mit Beispieldurchlauf und Screendumps (Link: ***Hilfe***), und zur MultiMAP-Projektseite (Link ***MultiMAP***) verfügbar.

Abb. 48: Anmeldung

Nach der Anmeldung kommt man auf die Startseite, auf der man zwischen einem Einstieg über die Volltexte des Alten Testaments (AT) (Link ***Bücher***) oder über gezielte Rechercheanfragen (Link ***Recherche***) wählen kann. Letztere bietet eine (trunkierbare) Volltextsuche über alle Worte oder Wortteile des ATs, Recherchen bezüglich der Morphologie, der Morphosyntax, der Satzkonkordanzen und eine vollkommen freie SQL-Recherchemöglichkeit auf der Grunddatenbank BHtDB. Klickt man auf ***Recherche***, so verzweigt man auf die Anzeige in Abbildung 55, beschrieben ab Seite 79 (Abschnitt 4.2.3). Der übliche Einstieg geht jedoch zunächst über die Volltexte der Bücher, vorgestellt im folgenden Abschnitt.

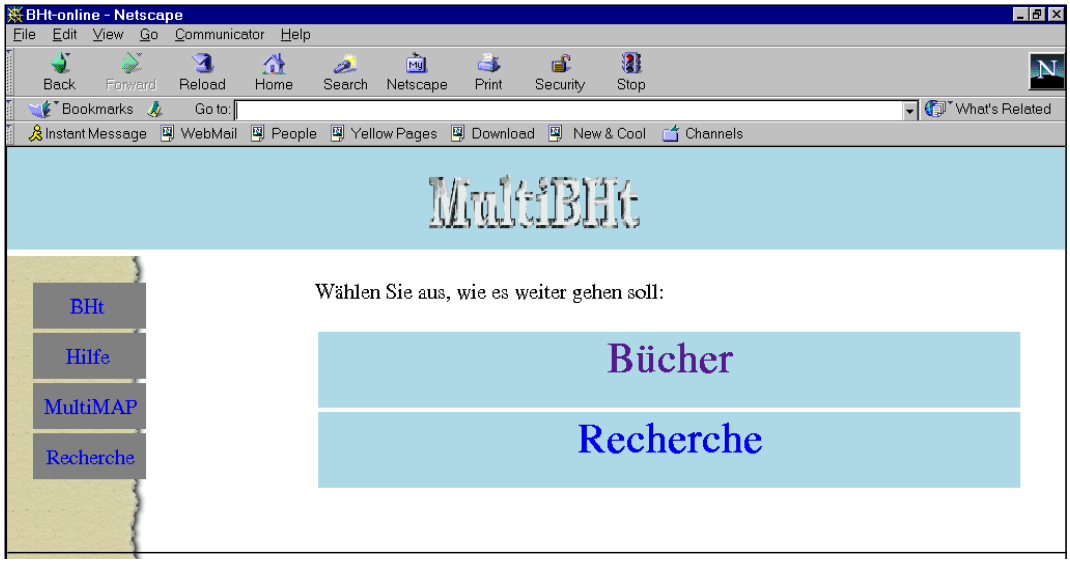

Abb. 49: Startseite

4.2.2 Einstieg über die Bücher der BHt (Volltexte)

Die einzelnen Kapitel der Bücher des Alten Testaments sind in transkribierter Form jeweils als eigene Seiten abgelegt. Über das Inhaltsverzeichnis, basierend auf dem Fisheye-Modell (siehe Abb. 50) kann der Benutzer durch Selektion des entsprechenden Buches und Kapitels zum Text gelangen. Hierbei wird nur der jeweilige Zugriffspfad im vollem Detaillierungsgrad angezeigt.

Abb. 50: Buchauswahl

4.2.2.1 BHt-Volltext-Anzeige

Der Text wird kapitelweise angezeigt; für jedes Kapitel eine Seite (siehe Abbildung 51). Zur korrekten Darstellung des transkribierten Textes wird ein spezieller Zeichensatz (*Miryam*) verwendet, der auf der jeweiligen Client-Maschine installiert sein muß. Zur Zeit existieren Zeichensätze für Unix, Linux und Microsoft Windows-Systeme, ladbar über die Einsteigsseite.

Um ein bequemes Blättern innerhalb eines Buches zu ermöglichen, sind die einzelnen Kapitel über bijektive Links ([***zurück***], [***vor***]) miteinander verknüpft. Daneben kann jedes Kapitel auch direkt angesprungen werden (über den Link [***Kapitel***]). Diese drei Strukturverweise werden am Kopf und Fuß jeder Textseite angezeigt.

Eine Textseite gliedert sich in die Überschrift, die aus dem jeweiligen Buchtitel und der Kapitelnummer besteht und dem dazugehörigen Text. Die einzelnen Zeilen des Textkorpus bestehend aus Vers- und Teilsatz-Identifikation, z.B. aR (eingeschobener Relativsatz), und dem entsprechenden Text in transkribierter Form.

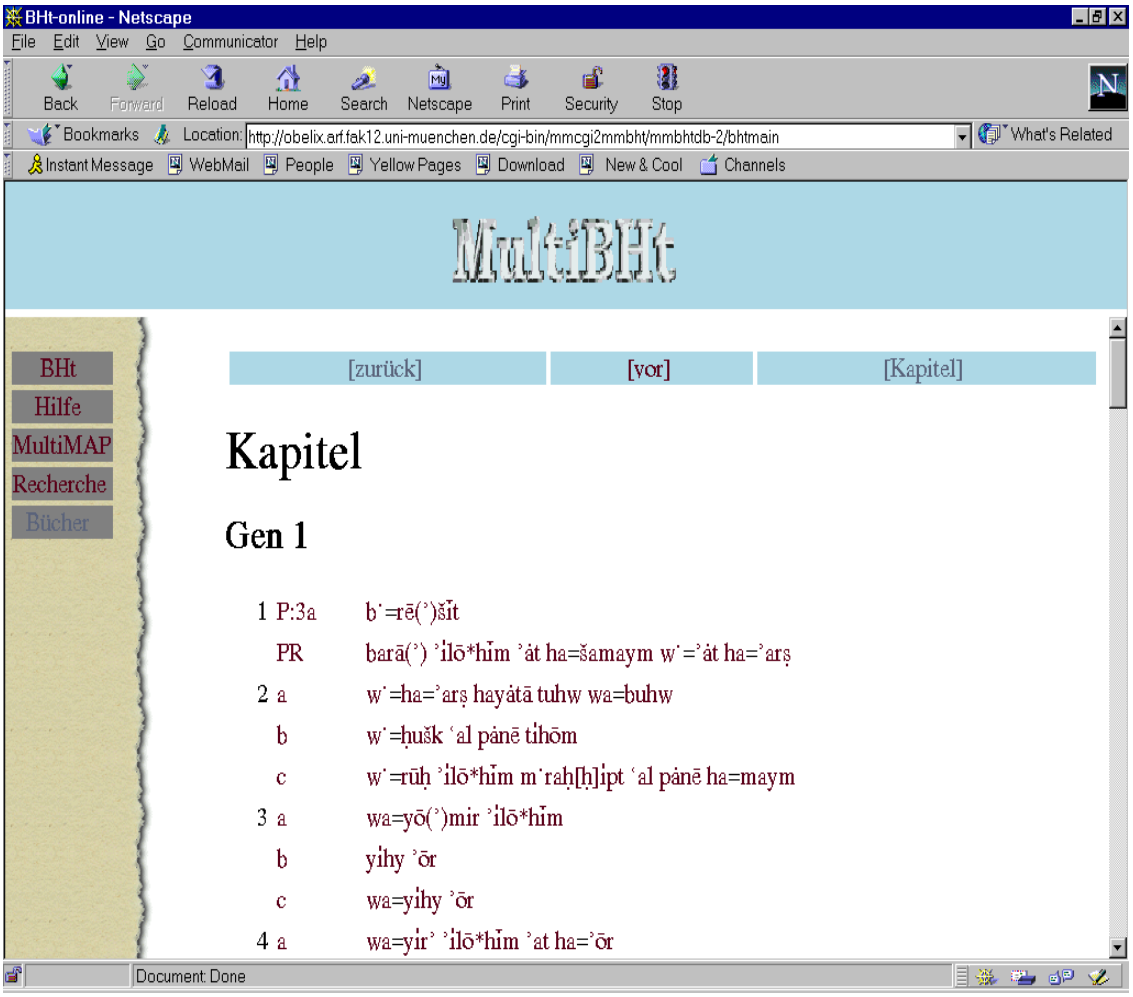

Abb. 51: BHt-Text

Alle Worte bzw. Wortstücke sind darin unabhängig klickbar. Das Hauptfenster bleibt bestehen. Es öffnen sich separate Appletfenster mit der jeweils zugehörigen morphologischen Analyse.

4.2.2.2 Morphologische Beleganalyse

Ausgangspunkt für die morphologische Analyse ist der Beleg. Er ist die kleinste Informationseinheit im System. Daraus werden rekursiv die sprachlichen Einheiten wie Wort und Satz aufgebaut. Durch Anklicken der einzelnen Belege (Wortstücke) im Text wird die zugehörige Beleganalyse in einem eigenen Applet angezeigt (siehe Abbildung 52). Die Grunddaten dazu sind den Relationen *beleg*, *wort* und *satz* aus der darunterliegenden Inhaltsdatenbank BHtDB entnommen. Folgende Informationen werden im BHt-Beleg angezeigt:

- Analyse:	Person/Status, Genus, Numerus, Stamm,
- Morpheme:	Endung, Erweiterung, Erweiterungsfunktionsklasse,
- Bau:	Bautyp, Baufunktionsklasse, Bauelement, Bauelementfunktionsklasse, Bauopposition, Bauvariante, Bauableitung,
- Alternativen:	Alternative,
- Basis:	Basis, Basiselement, Basiselementfunktion, Basisvariante, Basisableitung,
- Wurzel/Lexem:	Wurzel, Lexem,
- Kombination:	Kombination,
- Wortart/Funktionen:	Wortart, Funktionsebene, Wortartfunktion, Person-/ Statusfunktion, Genusfunktion, Numerusfunktion, semantische Funktion,
- Merkmale:	Kernsem, Merkmalsem, Semebene,
- Sprache:	Sprache,
- Kommentar:	Kommentar,

Eine genau Beschreibung ist unter dem Link ***BHt*** am linken Rand im MultiBHT-Browserfenster zu finden, bzw. unter

http://www.fak12.uni-muenchen.de/arf/bhtdb/schema6.html.

Durch die Appletfenster-Technik können mehrere morphologischen Beleganalysefenster gleichzeitig offen sein.

Abb. 52: BHt-Beleg

Vom Applet BHt-Beleg aus öffnet sich über den Knopf (Struktur-)***Baum*** ein weiteres Appletfenster mit dem umgebenden, morphosyntaktischen Analysebaum (Abb. 53), in den der Beleg an dieser Stelle eingebettet ist und über den Knopf ***Kommentar*** eine Kommentareingabemöglichkeit zu diesem Beleg (Abb. 54). Das Appletfenster BHt-Beleg bleibt jeweils bestehen. So kann man bequem die verschiedenen Analyseergebnisse und den Volltext dazu gleichzeitig am Bildschirm in je eigenen Fenstern sehen. Erst der Knopf ***Abbruch*** schließt das Appletfenster wieder (aber keine weiteren, auch nicht diejenigen, die von hier aus geöffnet wurden).

4.2.2.3 Morphosyntaktische Analyse

Wird im Appletfenster BHt-Beleg der Knopf (Struktur-)***Baum*** gedrückt, oder über den Einstieg BHt-Recherche die Morphosyntax aufgerufen und eine entsprechende Query gestellt, so wird, falls ein entsprechender Phrasenstrukturbaum (AMOS-Baum) existiert, dieser visualisiert (siehe Abbildung 53).
Die Buttons in der Fußzeile lassen zwischen der hier gezeigten ***Normal***-Anzeige (Default) und einer verkleinerten ***Überblick***s-Anzeige (sinnvoll bei sehr großen Strukturbäumen) hin und her schalten. ***Weiter*** schließt das Appletfenster.

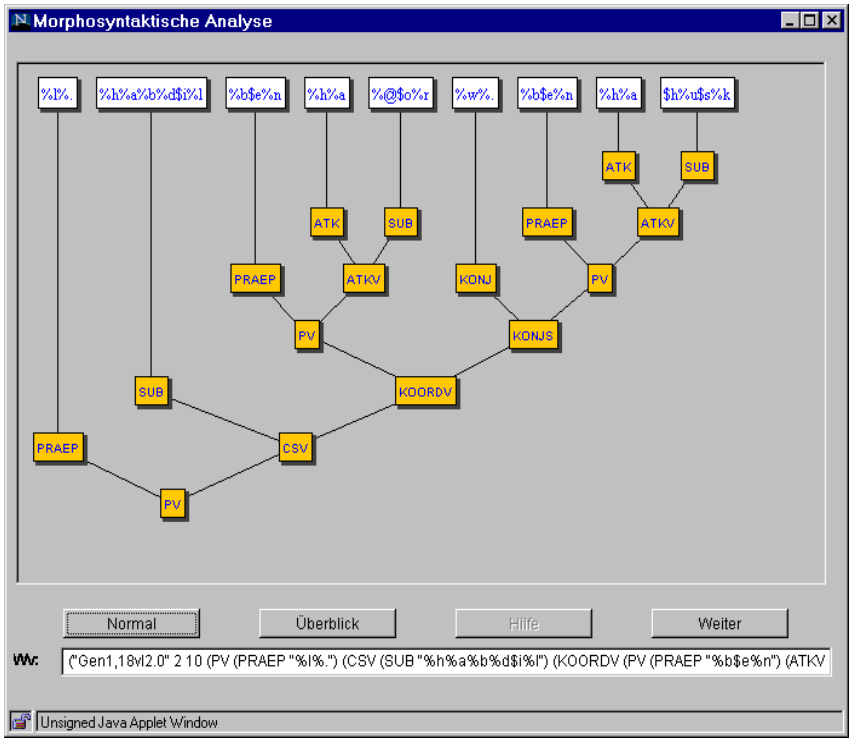

Abb. 53: Satzstrukturbäume (AMOS)

4.2.2.4 Annotationen

Das Annotationssystem enthält die Möglichkeit, Kommentare einzugeben (siehe Abbildung 54). Sie werden bei der Anzeige zur Zuordnung automatisch mit dem Loginnamen versehen.

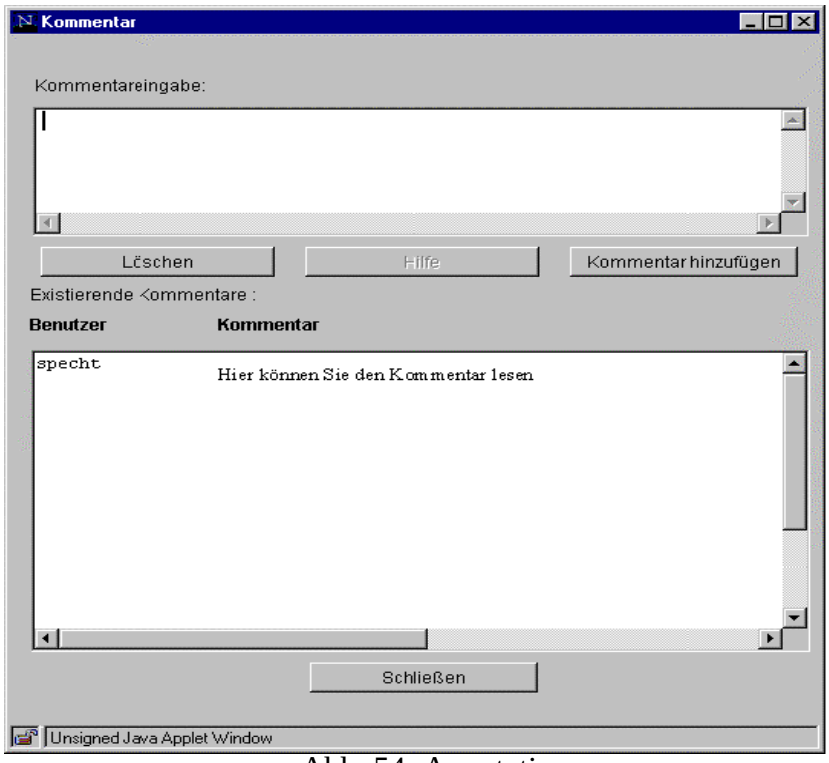

Abb. 54: Annotation

4.2.3 Einstieg über die BHt-Recherchekomponente

Neben dem Einstieg über die Bücher der BHt kann man von der Startseite (Abbildung 49) aus auch durch Auswahl der ***BHt-Recherche***-Komponente in MultiBHT einsteigen. Zwischen beiden Zweigen kann man auch jeweils direkt über die stets sichtbaren Auswahllinks am linken Rand des MultiBHT-Fensters wechseln.

Die BHt-Recherche (siehe Abbildung 55) bietet fünf verschiedene Such- und Analyse-Verfahren:

1. Volltextsuche:	Ausgabe aller Textstellen in einem Buch, an denen ein gesuchtes Wort oder Wortteil auftritt, sowie direkte Verzweigungsmöglichkeiten zu den gefundenen Texten.
2. Morphologie:	Ausgabe aller Datenbankeinträge zur morphologischen Analyse anhand der Wortartkategorie und der Basis.
3. Morphosyntax:	Ausgabe von Strukturbäumen zu Wortgruppen.
4. Satzkonkordanz:	Ausgabe einer Satzkonkordanz zu einer Wortartkategorie, einer Basis und einem Stamm.

5. SQL-Recherche: Voller SQL-Zugriff auf die darunterliegende Inhaltsdatenbank BHtDB.

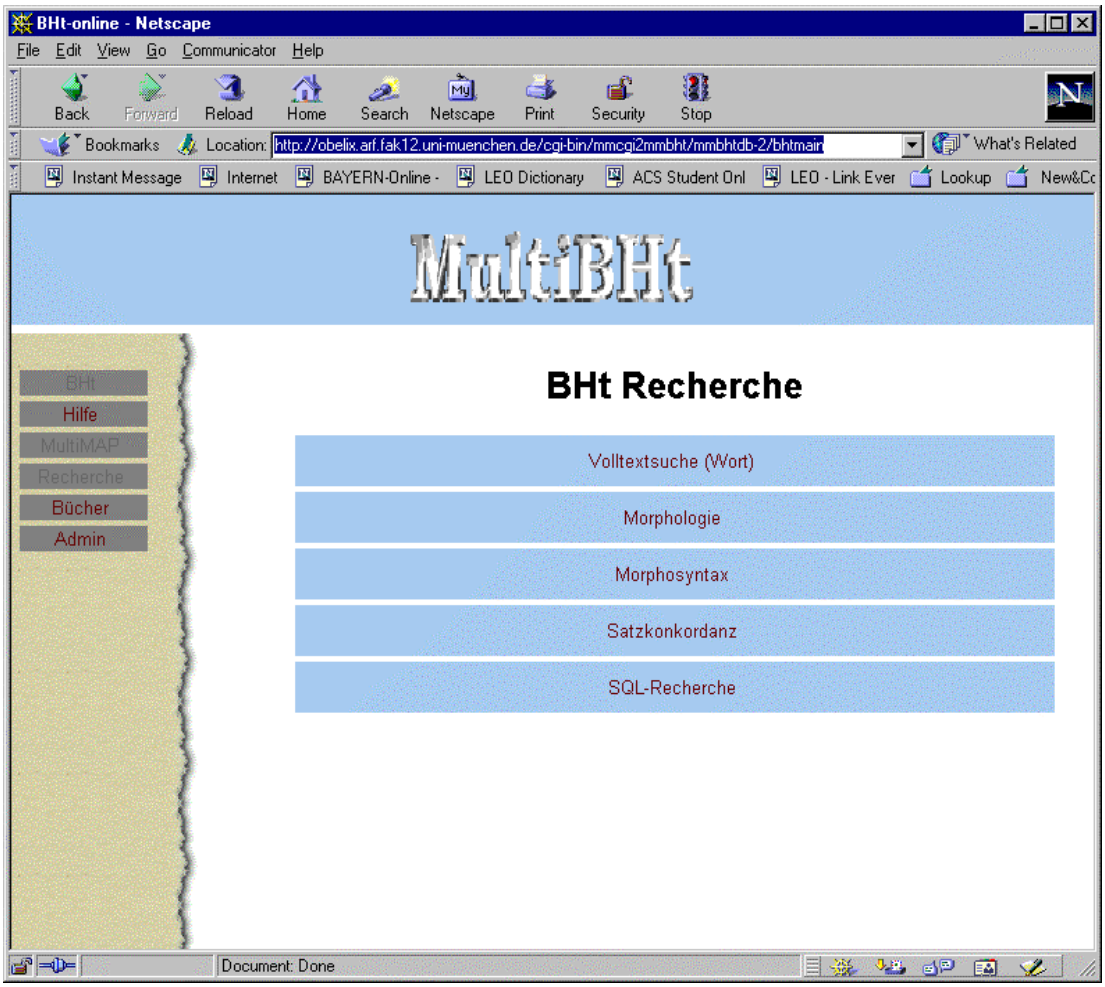

Abb. 55: BHt-Recherche

4.2.3.1 Volltextsuche

Die Volltextsuche ermöglicht den direkten Einstieg in eine oder mehrere Textstellen. Durch die Auswahl der Volltextsuche auf der Seite BHt-Recherche wird ein Java-Applet gestartet (siehe Abbildung 56). Um eine Volltextsuche abzuschicken, muß zunächst ein Buch aus der (z. Zt. noch alphabetisch sortierten) Auswahlliste bestimmt werden, z.B. “Gen” für die Genesis. Das Such-Wortfragment muß transkribiert in ASCII-Notation in die Eingabezeile eingegeben werden, z.B. “%r$e%(%”. Dabei steht das Zeichen % gleichzeitig für das Trunkierungszeichen (der Beispielstring ist damit vorne und hinten trunkiert), als auch für das Steuerzeichen % in der Text-Normal-Form (TNF). Die Eingabe wird durch das Drücken von ***Anfrage abschicken*** abgeschlossen und die Volltextsuche gestartet. Alle Ergebnisse werden in einer Tabelle mit den Spalten *Wort*, *Buch* und *Wortnummer* im Applet dargestellt. Zusätzlich wird das Wortfragment in die temporäre

Liste *Historie* zur Wiederverwendung und späteren Spezialisierung eingetragen. Ergebnisse weiterer Suchanfragen werden an die Ergebnisliste unten angehängt.

Mittels ***Ergebnis löschen*** kann die Ergebnisanzeige wieder gelöscht werden. Durch einen Doppel-Klick auf ein Ergebnis springt das System direkt auf die entsprechende Textstelle im Volltext der BHt, der im Browserfenster angezeigt wird (siehe Abbildung 51). Das Applet mit den Suchergebnissen bleibt weiterhin sichtbar. So kann bequem verglichen und zum nächsten Ergebnis gesprungen werden. Erst mit ***Weiter*** wird das Volltextsuch-Applet wieder geschlossen.

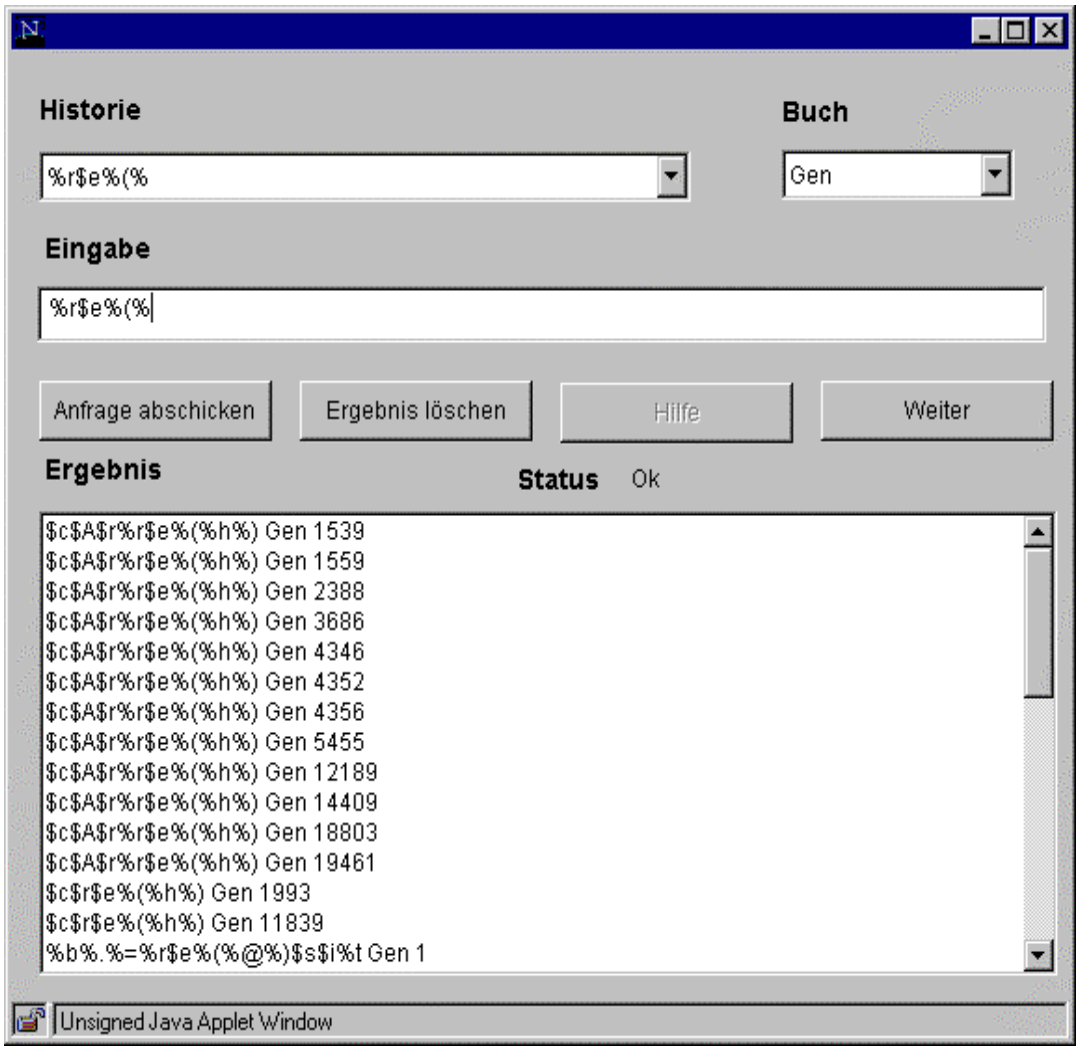

Abb. 56: Volltextsuche

4.2.3.2 Morphologische Analyse

Mit dieser Rechercheart können alle Datenbankeinträge zur morphologischen Analyse anhand der Wortartkategorie sowie der Basis (bzw. einem Stück) bestimmt werden.

Hierzu muß in der Maske “Morphologische Analyse” (siehe Abbildung 57) die entsprechende Wortartkategorie aus folgender Liste ausgewählt werden:

1. Verb
2. Verbalnomen
3. Nomen
4. Prowortart
5. Partikel
6. Eigennamen

Die Grundeinstellung ist die Hauptwortart “Verb”. Die Eingabe wird mit *Eingabe* abgeschickt.

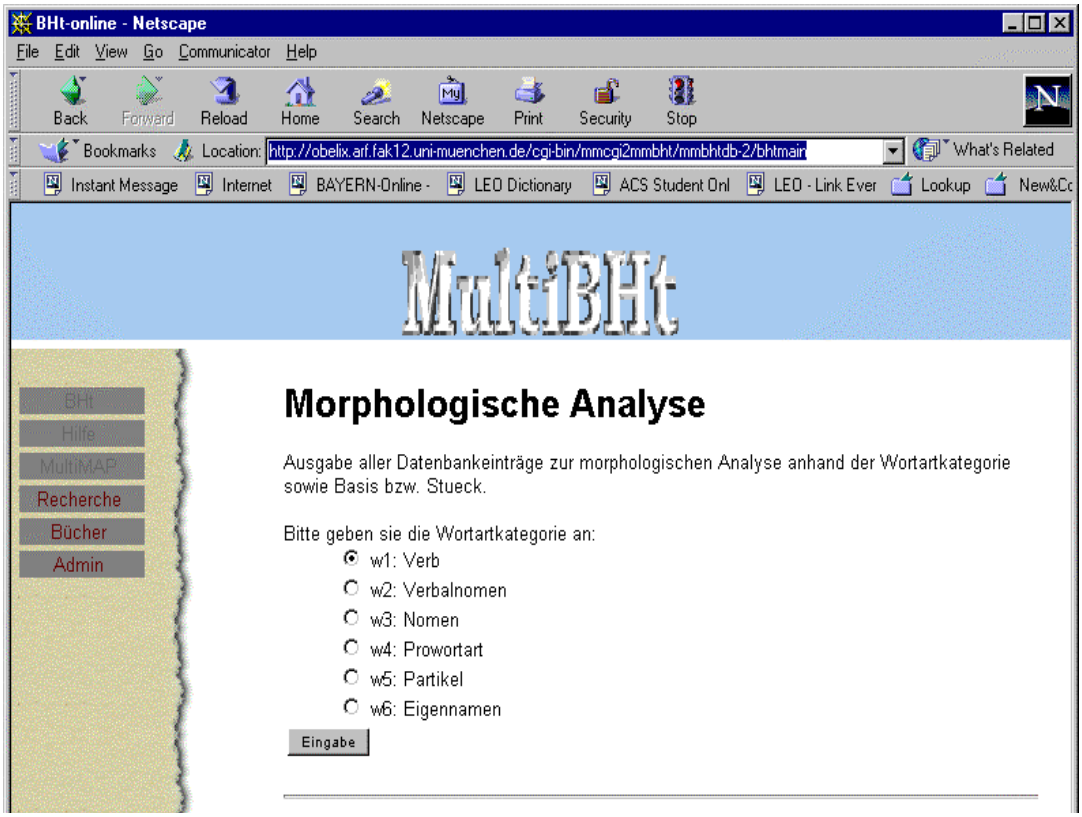

Abb. 57: Morphologische Analyse: Wortartkategorie

Anschließend muß die Basis eingegeben werden (siehe Abbildung 58). Die Zeichenketten werden in der sog. TNF-Struktur (Text-Normal-Form) eingegeben, jeder Buchstabe besteht dabei aus einem Steuerzeichen und einem Transkriptionszeichen (z.B.: $h%r%d). Ist zu einer Basis eine Differenzierung in Homonyme erforderlich, wird der Basis-String mit & abgeschlossen und eine arabische Ziffer angefügt: (z.B.: %m%l%k&1, %m%l%k&2). Auch diese Eingabe wird wieder mit *Eingabe* abgeschickt.

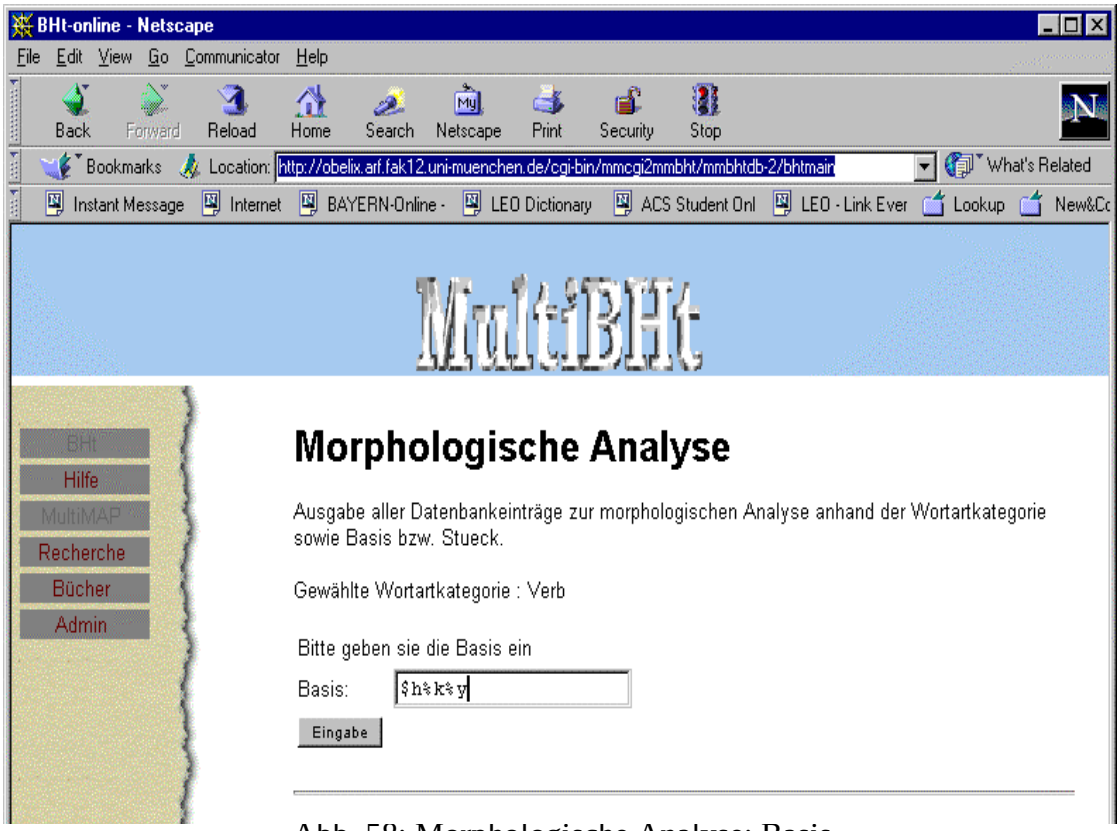

Abb. 58: Morphologische Analyse: Basis

Das Ergebnis der morphologischen Analyse wird in Form einer Tabelle im Browser angezeigt (siehe Abbildung 59). Die Tabelle hat folgende Spalten:

1. *b_nr* (Belegidentifikationsnummer): Die interne Belegidentifikationsnummer ist das zentrale Schlüsselattribut in der Anwendungsdatenbank BHtDB, sie ist Buch-bezogen, beginnt also je Buch wieder mit “1”. Die Attributkombination (buch, b_nr) identifiziert jedes Wortstück eindeutig in der gesamten BHt. Ihr ist genau ein Wert der Attributkombination (buch, kap, vers, stknr) zugeordnet.
2. *Stellenangabe*: Die Angabe von Buch, Kapitel und Vers.
3. *wa* (Wortart): Das Attribut enthält die genaue Wortart. Sie ist eine Unterkategorie der gewählten Wortartkategorie.
4. *basis* (Basis): Das Attribut enthält bei Hauptwörtern deren hebräische oder aramäische Basis, bei Partikel die merkmalhaften Konsonanten und bei Fremdwörter alle Konsonanten. Dabei unterscheiden arabische Ziffern homonyme Basen.
5. *Bautyp*,
6. *ps* (Person/Status): Das Attribut enthält bei Verben die grammatikalische Person in der das Verb steht, bei den nominalen Wortarten den Status.

7. *gen* (Genus): Der Genus eines Wortes kann maskulin (M), feminin (F) oder nicht vorhanden (0) sein.
8. *num* (Numerus): enthält den Numerus: Singular (S), Dual (D), Plural (P) oder nicht vorhanden (0).
9. *stamm* (Stamm): Das Attribut enthält den Verbalstamm.

Alle Einträge in der ersten Spalte sind klickbar. Durch Anklicken der Belegidentifikationsnummer kann zu der entsprechenden Stelle im Volltext der BHt (siehe Abbildung 51) gesprungen werden.

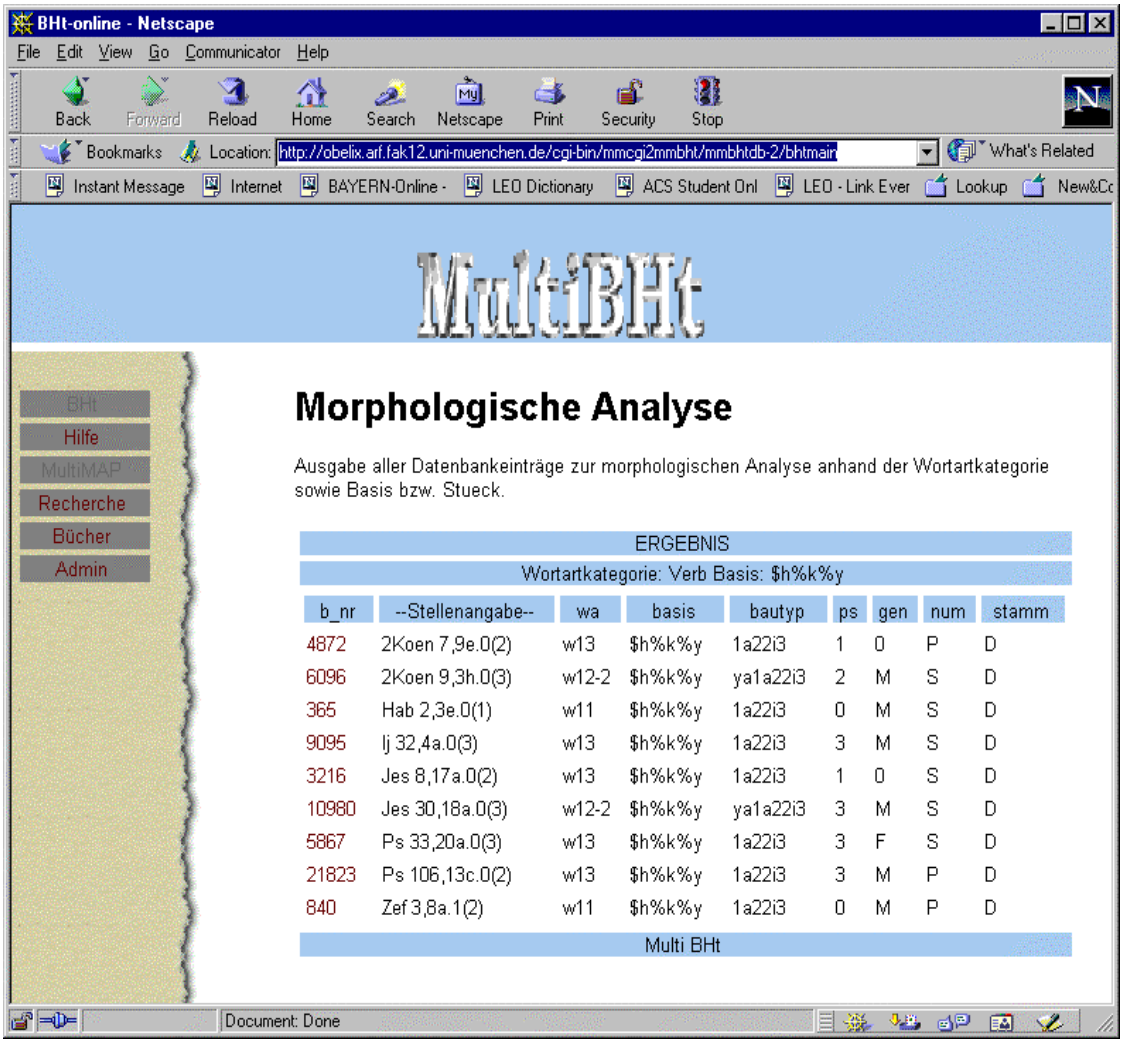

Abb. 59: Morphologische Analyse: Ergebnis

4.2.3.3 Morphosyntaktische Analyse

Wird in der Einstiegsseite BHt-Recherche (Abbildung 55) die Morphosyntax aufgerufen, so kann nach Wortverbindungen gesucht werden. Im Eingabefenster wird die Eingabe der Verbindungsart (Wortgruppe) in Großbuchstaben (z.B.: CSV, APPV, PV, ect.) und des Buches, in dem gesucht werden soll (z.B.: Gen), verlangt. Als Ergebnis erhält man alle gefundenen Stellen im Buch die diese Wortverbindungsart enthalten, dargestellt in einer Tabelle mit den Spalten *(interne) Baumnummer*, *Buch* und *Stelle*. Die Baumnummer ist wiederum klickbar und visualisiert die entsprechenden Strukturbäume (AMOS-Bäume) in je eigenen Applets (siehe Abbildung 53).

4.2.3.4 Satzkonkordanz

Da in der *Biblia Hebraica transcripta* Satzgrenzen eingetragen worden sind, ist gegenüber klassischen Konkordanzen mit Versangaben erstmalig die Extraktion einer Satzkonkordanz möglich. Erheblich erleichtert wird dadurch z.B. die für Valenzuntersuchungen notwendige Sammlung von Belegen, die bisher darin bestand, sämtliche Stellen eines Verbs anhand einer Konkordanz zu suchen, um sie dann, da die Belege in klassischen Konkordanzen ohne Satzbezug und mit manchmal eingeschränktem Kontext zitiert sind, durch Heranziehen des Textes zu vervollständigen und zu prüfen. Mit der Rechercheart Satzkonkordanz können erstmals alle Sätze, in denen eine gesuchte Basis mit einem bestimmten Stamm vorkommt, vollständig ausgegeben werden.
In der Eingabemaske “Satzkonkordanz” (siehe Abbildung 60) muß zunächst eine Wortartkategorie ausgewählt werden. Mögliche Wortartkategorien sind:

1. Verb	w1
2. Verbalnomen	w2
3. Nomen	w3
4. Prowortart	w4
5. Partikel	w5
6. Eigennamen	w6

Die Grundeinstellung ist die Hauptwortart Verb.

Abb. 60: Satzkonkordanz: Wortart

Anschließend muß eine entsprechende Basis in TNF-Struktur (http://www.fak12.uni-muenchen.de/arf/bhtdb/basis.html) und ein Stamm (http://www.fak12.uni-muenchen.de/arf/bhtdb/stamm.html) eingegeben werden (siehe Abbildung 61). Die Eingabe wird mit *Eingabe* abgeschickt.

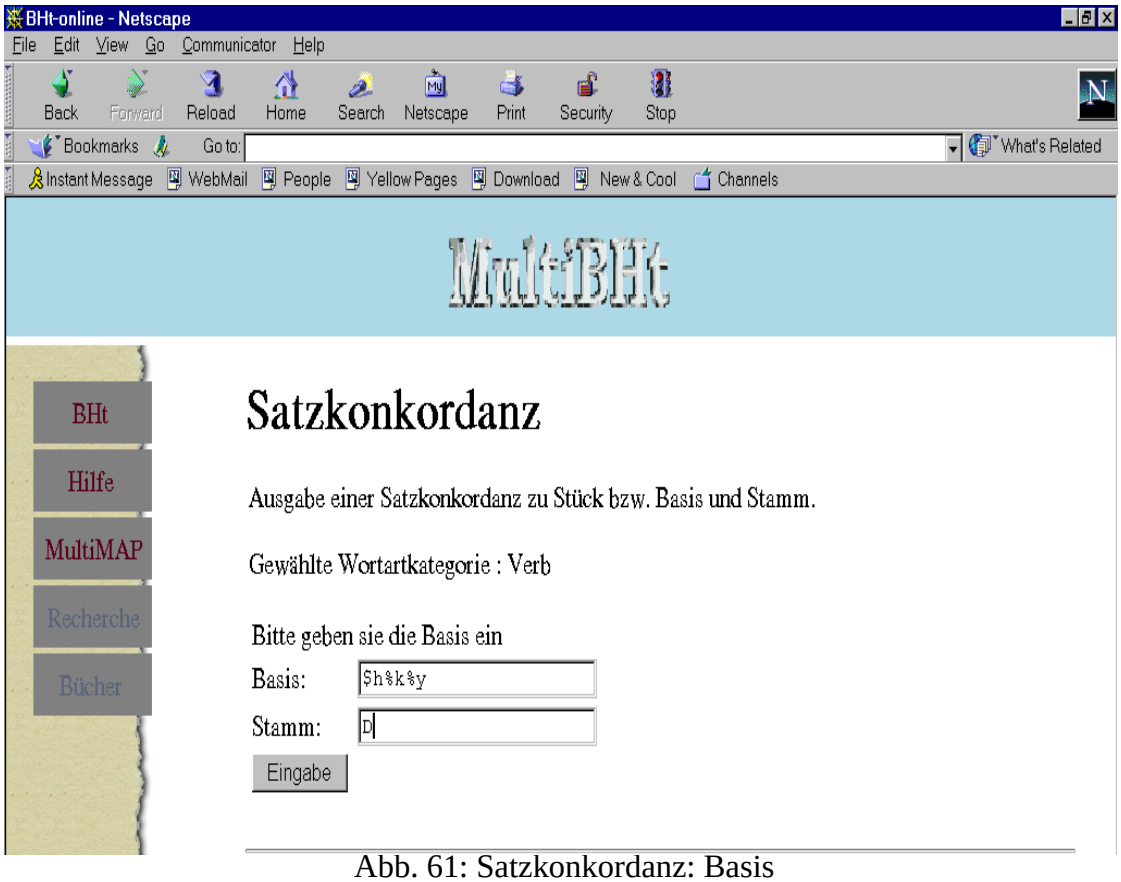

Abb. 61: Satzkonkordanz: Basis

Das Ergebnis der Satzkonkordanz wird wieder in Form einer Tabelle im Browser angezeigt (siehe Abbildung 62). Die Tabelle hat folgende Spalten:

1. *b_nr* (Belegidentifikationsnummer): Die interne Belegidentifikationsnummer ist das zentrale Schlüsselattribut in der Anwendungsdatenbank BHtDB, sie ist Buch-bezogen, beginnt also je Buch wieder mit “1”. Die Attributkombination (buch, b_nr) identifiziert jedes Wortstück eindeutig in der gesamten BHt.
2. *wa* (Wortart): Das Attribut enthält die genaue Wortart. Sie ist eine Unterkategorie der gewählten Wortartkategorie.
3. *Stellenangabe*: Die Angabe von Buch, Kapitel und Vers.
4. *s_nr*: (Satzidentifikationsnummer): Identifikation der Sätze innerhalb eines Buches.
5. *se_nr*: (Satzelementidentifikationsnummer): Identifikation der Satzelemente innerhalb eines Satzes. Identifiziert zusammenhängende Teile, die durch Relativsätze unterbrochen sind. Daraus ist erkennbar, ob ein Satz ein oder mehrere Elemente besitzt und der Kontext im BHt-Text betrachtet werden muß.
6. *Text*: transkribierter Satz.

Alle Einträge in der ersten Spalte sind klickbar. Durch Anklicken der Belegidentifikationsnummer kann wieder zu der entsprechenden Stelle im Volltext der BHt (siehe Abbildung 51) gesprungen werden.

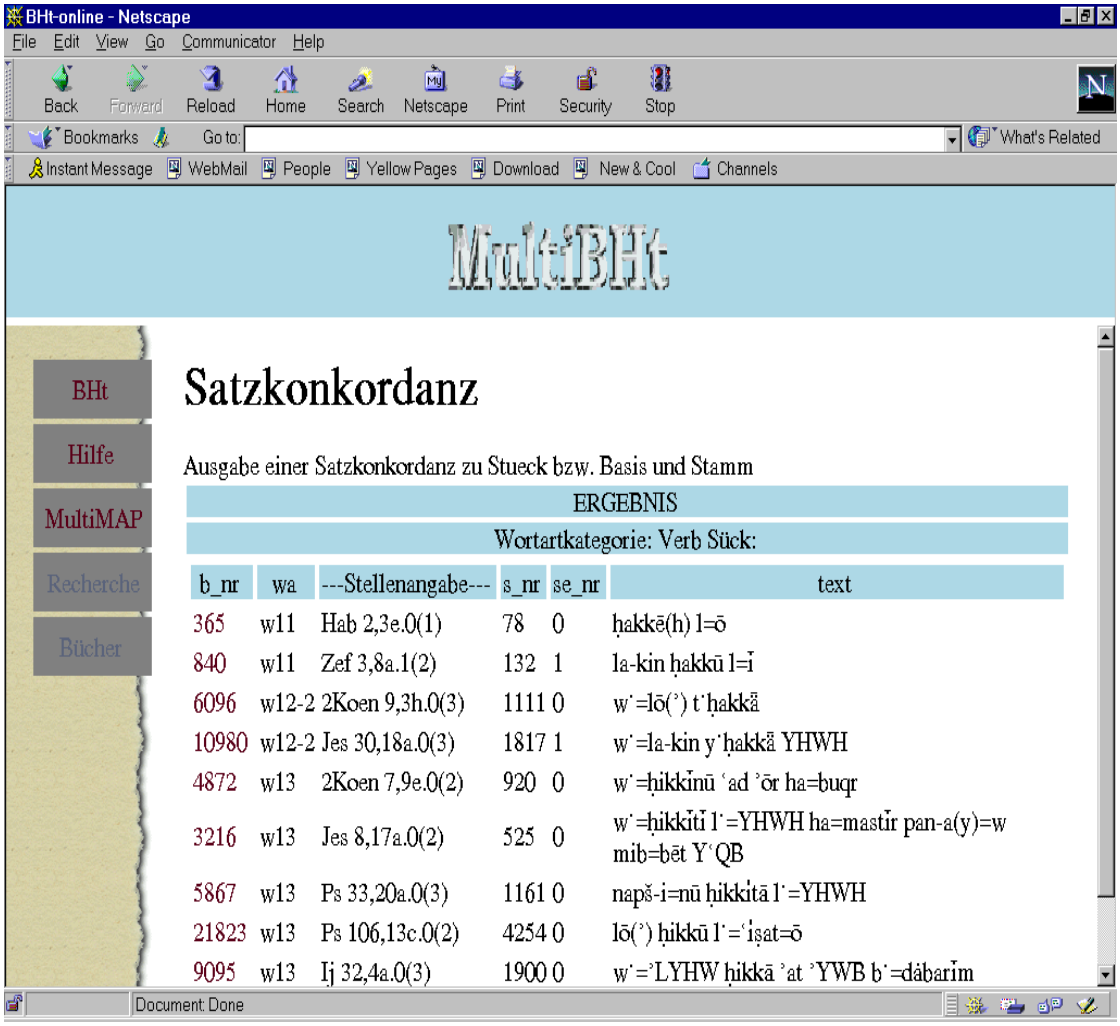

Abb. 62: Satzkonkordanz: Ergebnis

4.2.3.5 SQL-Recherche

Um Experten die volle Funktionalität der BHt-Inhaltsdatenbank bereitzustellen, wurde ein SQL-Interface-Applet entwickelt. Über dieses Interface können beliebige Datenbankanfragen abgeschickt werden. Die Ergebnisse werden in Form einer Liste ausgegeben. Wie das Applet Volltextsuche verfügt auch das SQL-Interface-Applet über eine temporäre Historien-Liste, die die letzten Anfragen enthält.

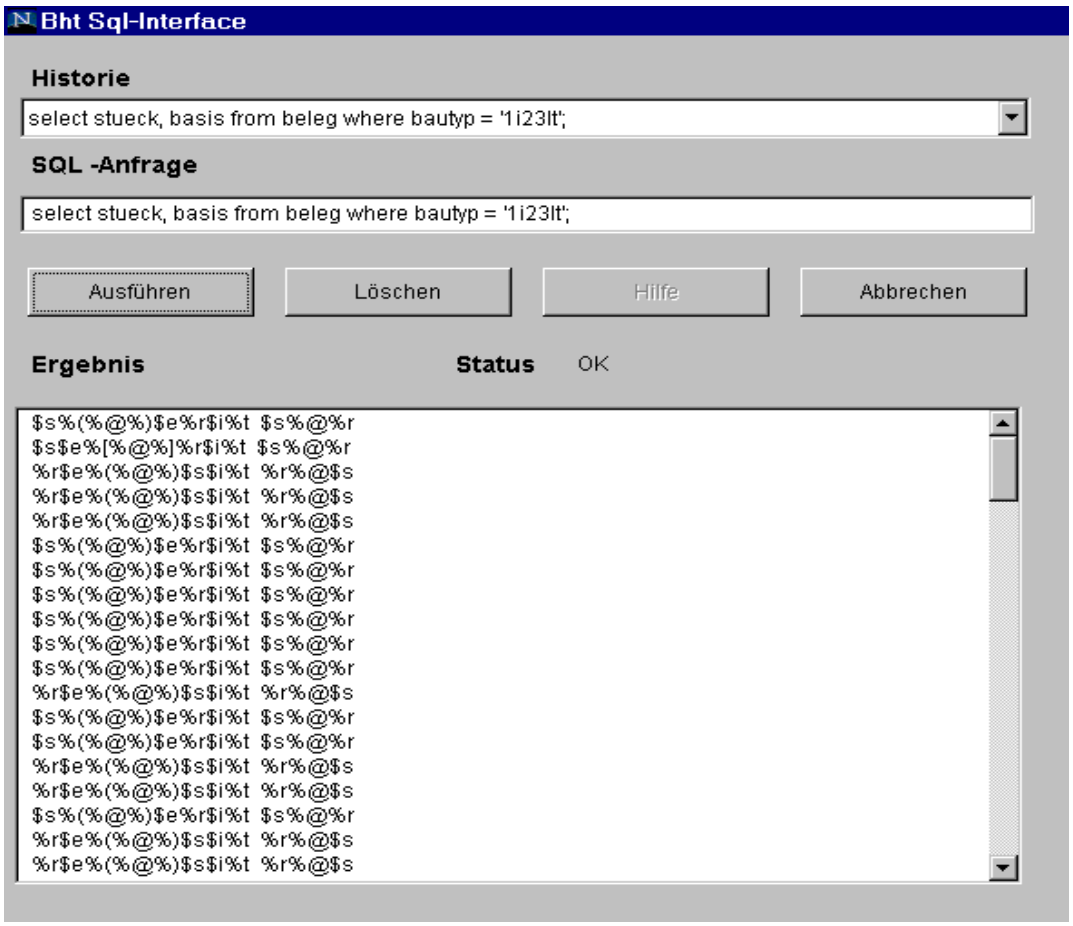

Abb. 63: SQL-Interface

5 Ausblick

Um den Source-Code zu warten, wurden alle komplexeren Teile mit Hilfe von DOC++ bzw. Javadoc dokumentiert. Der Java-Code für MultiBHT ist vollständig mit VisualAge for Java von IBM implementiert worden und wurde den Projektpartnern in Form eines Repository zur Verfügung gestellt.

Zusammenfassend läßt sich sagen, daß das Projekt MultiMAP/2 nicht nur in der Implementation und der Realisierung des Netzzugangs und Netzbetriebs für ein multimediales Datenbanksystem äußerst erfolgreich war, sondern auch wissenschaftlich. Es entstanden zahlreiche zusätzliche wissenschaftliche Erweiterungen und Ideen in seinem Umfeld wie z.B. Arbeiten zu

- Personalisierung,
- Überlappenden Links in Texten,
- Annotationen,
- Video-Integrationen,
- JDBC-Kopplungen, etc.

um nur einige zu nennen. Die vollständige Liste aller Arbeiten und Veröffentlichungen ist im Anhang abgedruckt.

Wir dürfen uns an dieser Stelle noch einmal ausdrücklich bei allen mitarbeitenden Studenten, unseren Projektpartnern, allen voran Hr. Dr. Chr. Riepl und Hr. Dr. H. Hilz, und insbesondere bei der DFN-Geschäftsstelle, vor allem bei Fr. G. Maiß, recht herzlich für die gute und konstruktive Zusammenarbeit bedanken. Den erstellten Systemen wünschen wir weiterhin einen erfolgreichen Einsatz und eine möglichst große Benutzerakzeptanz.

6 Teilnehmerkreis

Antragsteller / Projektleiter:

- Dr. habil. Günther Specht
  Institut für Informatik, Technische Universität München,
  Orleansstr. 34, 81667 München
  Tel: (089) 48095-178, Fax: (089) 48095-170
  Email: specht@informatik.tu-muenchen.de

- Prof. Rudolf Bayer. Ph.D.
  Institut für Informatik, Technische Universität München,
  Orleansstr. 34, 81667 München
  Tel: (089) 48095-171, Fax: (089) 48095-170
  Email: bayer@informatik.tu-muenchen.de

Projektmitarbeiter:

- Dipl.-Inform. Martin Zirkel
  Institut für Informatik, Technische Universität München,
  Orleansstr. 34, 81667 München
  Tel: (089) 48095-171, Fax: (089) 48095-170
  Email: zirkel@informatik.tu-muenchen.de

Studentische Mitarbeiter und Mitarbeiterinnen:

als studentische Hilfskräfte:

- Fr. Yvonne Zimmer,
- Fr. Eva Beinhofer,
- Hr. Theodor Myntidis
- Hr. Athanasios Sarakatsanis

im Rahmen ihrer Diplomarbeit:

- Hr. Markus Steindl,
- Hr. Theodor Myntidis,
- Hr. Thomas Kahabka,
- Hr. Athanasios Sarakatsanis,
- Fr. Martina Lindner
- Hr. Helmut Lutzenberger

im Rahmen ihrer Studienarbeit (Fopra)

- Fr. Eva Beinhofer,
- Hr. Eshref Januzaj,
- Fr. Yvonne Zimmer,
- Hr. Christian Trennhaus
- Hr. Jörg Lanzinger.

Projektpartner (Anwender):

in der Anwendung MultiLIB (Bibliotheksinformationssystem)

- Dr. Helmut Hilz, Leiter der Bibliothek für Wirtschaftswissenschaften, Universitätsbibliothek München, Ludwigstr. 28, 80539 München
- Dr. Fritz Junginger, Direktor der Universitätsbibliothek München, Geschwister-Scholl-
  Platz 1, 80536 München

in der Anwendung MultiBHT (Sprach- und Textanalysesystem):

- Dr. Christian Riepl, Rechnergestützte Forschung, Altertumskunde und Kulturwissenschaften, Universität München, Geschwister-Scholl-Platz 1, 80539 München
- Prof. Dr. Wolfgang Richter, Emeritus, Institut für Assyriologie und Hethitologie, Universität München, Geschwister-Scholl-Platz 1, 80539 München

weitere Nutzergruppen:

- Prof. Dr. Theodor Seidl, Lehrstuhl für Altes Testament und biblisch-orientalische Sprachen, Universität Würzburg, Sanderring 2, 97070 Würzburg
- Prof. Dr. Hubert Irsigler, Lehrstuhl für Altes Testament, Universität Bamberg,
  An der Universität 2, 96045 Bamberg
- Prof. Dr. Johannes Floß, Theologisches Institut, Rheinisch-Westfälische Technische Universität Aachen, Eilfschornstr. 7, 52056 Aachen
- Prof. Dr. Walter Groß, Lehrstuhl für Altes Testament, Universität Tübingen,
  Liebermeisterstr.12, 72076 Tübingen

7 Tagungen, Kongresse und Vorführungen

MultiMAP wurde bisher auf folgenden Tagungen und Kongressen und Ausstellungen vorgestellt:

- **DFN-Symposium, Berlin,** 29.1.97:
  Specht G.: *MultiMAP/2 - eine WWW Anbindung an eine multimediale Datenbank,* DFN-Symposium, Session: Multimedia-Teledienste, 28.-29.1.97 in Berlin

- **PAS ’97, Aizu, Japan**, 21.3.97
  Specht G.: *Introducing Parallelism in Multimedia Database Systems*, 2nd. Int. Symposium on Parallel Algorithms/Architectures Synthesis, 17.-21.3.1997 in Aizu-Wakamatsu, Japan

- **HIM ’97, Dortmund,** 2.10.97
  Kahabka T., Specht G.: *GRAS: An Adaptive Personalization Scheme for Hypermedia Databases*, 2nd. Conference on Hypertext - Information Retrieval - Multimedia (HIM ‘97), 29.9. - 2.10.1997 in Dortmund

- **ATS ’97, München,** 10.10.97
  Specht G., Zirkel M.: *Multimedialer Zugang zur Biblia Hebraica Transcripta über das WWW,* Vortrag mit Demo, 7. Konferenz “Arbeiten zu Text und Sprache im Alten Testament”, 9.10. -10.10.1997 in München

- **DFN-Betriebstagung, Berlin** 28.10.97
  Zirkel M., Specht G.: *MultiMAP/2 - Architektur und Implementation eines Hypermedia Datenbanksystems mit Java-Technik*, DFN-Betriebstagung, Session: Multimedia-Teledienste, 28.-29.10.97 in Berlin

- **Informatik-Kolloquium, TU München,** 7.5.98
  Specht G.: *Multimedia-Datenbanksysteme: Modelle, Architekturen und Retrieval*, Informatik-Kolloquium, TU München

- **INET ’89, Genf,** 21.-24.7.1998
  Specht G., Zirkel M.: *Introducing Nested and Overlapping Links in HTML-Texts*, Int. Conference INET ‘98, Internet Society, Postersession, 21.-24.7.1998, Genf

- **Frauenhofer Gesellschaft, Darmstadt**, 24.8.98
  Specht G.: *Architekturen interaktiver, multimedialer Hypermedia-Systeme*, Institut für Graphische Datenverarbeitung, Frauenhofer Gesellschaft, Darmstadt

- **KI ’98, Bremen,** 16.9.98
  Specht G.: *Architekturen von Multimedia-Datenbanksystemen zur Speicherung von Bildern und Videos*, 22. Jahrestagung für Künstliche Intelligenz (KI ‘98), Workshop: Inhaltsbezogene Suche von Bildern und Videosequenzen in digitalen multimedialen Archive, 15.-17.9.1998 in Bremen

- **ATS ’98, München,** 9.10.98
  Specht G., Zirkel M.: Demo zu MultiBHT, 8. Konferenz “Arbeiten zu Text und Sprache im Alten Testament”, 8.10. - 9.10.1998 in München

- **Informatik-Kolloquium, Uni Kiel,** 28.9.98
  Specht G.: *Das Dexter Referenzmodell und seine Implementation*, Informatik-Kolloquium, Universität Kiel

- **Workshop des Humanethologischen Filmarchivs der Max-Plank-Gesellschaft, Andechs**
  14.11.98
  Specht G.: *Interaktive digitalisierte Videoindizierung und Recherch*e, Workshop des Humanethologischen Filmarchivs der Max-Plan-Gesellschaft, Prof. Dr. Eibl-Eibesfeldt, 13.-14.11.1998 in Andechs

8 Vollständige Bibliographie zum MultiMAP-Projekt

8.1 Wissenschaftliche Veröffentlichungen zu MultiMAP

1. Specht G., Bauer M., 1995: *MultiMED - ein Multimedia-Datenbanksystem zur Aus- und Weiterbildung in der Röntgendiagnostik in der Orthopädie*, in: Schoop E., Witt R. Glowalla U. (Hg.): Hypermedia in der Aus- und Weiterbildung, Schriften zur Informationswissenschaft, Bd. 17, Universitätsverlag Konstanz (UVK), Konstanz 1995, pp. 209-210

2. Specht G., 1995: *MultiBHT - ein multimediales Datenbanksystem zur Sprachanalyse*, vorgestellt am 29. März 1995 an der LMU München vor der Fakultät für Alterstumskunde und Kulturwissenschaften. Sonderdruck, Fakultät für Alterstumskunde, LMU München, 9 S.

3. Specht G., Hofmann M., 1996: *Auswertung der Migration eines Multimedia-Informationssystems von einem relationalen auf ein objektorientiertes Datenbanksystem*, in: Proc. Jahrestagung der Gesellschaft für Informatik (GI), 25.-27.9.96 in Klagenfurt, Oldenbourg-Verlag, 1996, pp. 233-251 (best paper award). Ausgezeichnet mit dem Software-Engineering-Preis 1996 der Ernst-Denert-Stiftung.

4. Specht G., Zimmermann S., Clausnitzer A., 1997: *Introducing Parallelism in Multimedia Database Systems*, in: Proc of the 2nd. Int. Symposium on Parallel Algorithms/Architectures Synthesis, 17.-21.3.1997 in Aizu-Wakamatsu, Japan, IEEE Computer Society Press, 1997, pp. 348 - 355

5. Specht G., 1997: *Complexity Analysis of Link Navigation in Dexter Based Hypermedia Database Systems*, in: Informatica, (Journal), Vol. 8, No. 1, Vilnius, Lithuania 1997, pp. 23-42

6. Kahabka T., Korkea-aho M., Specht G., 1997: *GRAS: An Adaptive Personalization Scheme for Hypermedia Databases*, Proc. of the 2nd. Conference on Hypertext - Information Retrieval - Multimedia (HIM ‘97), 29.9. - 2.10.1997 in Dortmund, Schriften zur Informationswissenschaft, Bd. 30, Universitätsverlag Konstanz (UVK), Konstanz 1997, pp. 279 - 292

7. Specht G., Zirkel M. 1998: *Introducing Nested and Overlapping Links in HTML-Texts*, Int. Conference INET ‘98, Internet Society, Postersession, 21.-24.7.1998, Genf

8. Specht G., 1998: *Architekturen von Multimedia-Datenbanksystemen zur Speicherung von Bildern und Videos*, in: Luth N. (Hrsg.): Inhaltsbezogene Suche von Bildern und Videosequenzen in digitalen multimedialen Archiven, Beiträge eines Workshops der 22. Jahrestagung für Künstliche Intelligenz (KI ‘98), 15.-17.9.1998 in Bremen, Bremen 1998, pp. 7-25

9. Specht G., 1998: *Multimedia-Datenbanksysteme: Modellierung - Architektur - Retrieval*, Habilitationsschrift, Technische Universität München, 270 S.

8.2 Diplomarbeiten und Studienarbeiten im MultiMAP-Projekt

1. Kurzmann M., 1994: *Entwurf und Implementierung der multimedialen Datenbank MultiMAP für Stadtpläne und Landkarten*, Diplomarbeit, Technische Universität München

2. Myntidis T., 1995: *Implementation von MultiMED, einer multimedialen Datenbank für die Medizin*, Fortgeschrittenenpraktikum, Technische Universität München

3. Cornelius A., 1995: *Erweiterung der multimedialen Datenbank MultiMAP, insbesondere um eine Wegesuche-Komponente*, Diplomarbeit, Technische Universität München

4. Hofmann M., 1995: *Entwicklung einer Hypermedia-Datenbank auf dem objektorientierten Datenbanksystem O2*, Diplomarbeit, Technische Universität München

5. Mallow B., 1995: *Objektorientierte und wissensbasierte Orientierungsstrategien in Hypermedia-Datenbanken*, Diplomarbeit, Technische Universität München

6. Steindl M., 1996: *Erweiterung des multimedialen Datenbanksystems MultiMAP um Volltextsuche*, Fortgeschrittenenpraktikum, Technische Universität München

7. Myntidis T., 1996: *Weiterentwicklung der multimedialen Datenbank MultiMAP und Implementierung des Bibliotheksinformationssystems MultiLIB für die Universitätsbibliothek München*, Diplomarbeit, Technische Universität München

8. Steindl M., 1997: *Anbindung des multimedialen Datenbanksystems MultiMAP an das World Wide Web*, Diplomarbeit, Technische Universität München

9. Kahabka T., 1997: *Personalisation in MultiMAP*, (in Englisch) Diplomarbeit, Technische Universität München

10. Beinhofer E., 1997: *Erweiterung des multimedialen Informationssystems MultiLIB für die Universitätsbibliothek München,* Fortgeschrittenenpraktikum, Technische Universität Mün chen

11. Januzaj E., 1997: *Implementierung und Evaluierung eines verschachtelten Linkkonzeptes für MultiMAP,* Fortgeschrittenenpraktikum, Technische Universität München

12. Zimmer Y., 1998: *Using the New Event Model of JAVA 1.1 in MultiMAP/2,* (in Englisch) Fortgeschrittenenpraktikum, Technische Universität München

13. Sarakatsanis A., 1998: *Entwicklung und Implementierung einer auf JavaBeans basierten Oberfläche für MultiMAP sowie einer JDBC-Verbindung zu DB2,* Diplomarbeit, Technische Universität München

14. Trennhaus Ch., 1998: *Optimierung der Kommunikation zwischen Java-Applet und DB-Client,* Fortgeschrittenenpraktikum, Technische Universität München

15. Lindner M., 1998: *Entwurf und Implementierung einer Videokomponenten für die multimediale Datenbank MultiMAP/2,* Diplomarbeit, Technische Universität München

16. Lutzenberger H., 1998: *Evaluierung der Einsatzmöglichkeiten von XML im Projekt MultiMAP,* Diplomarbeit, Technische Universität München

17. Lanzinger J., 1998: *Visualisierung von AMOS-Bäumen mit JavaBeans im Projekt MultiMAP/2,* Systementwicklungsprojekt, Technische Universität München.