Master-Arbeit

Entwicklung eines visuellen Metadaten-Browsers für die Mediothek Konstanz

Adaption der VisMeB-Architektur an eine konkrete Anwendungsdomäne

Christian Grün

1. Gutachter: Prof. Dr. Harald Reiterer

2. Gutachter: Prof. Dr. Marc Scholl

Arbeitsgruppe Mensch-Computer-Interaktion
FB Informatik und Informationswissenschaft
Universität Konstanz

November 2004

Zusammenfassung

Diese Master-Arbeit beschreibt die Konzeption, Entwicklung und Implementierung eines Browsers für bibliothekarische Metadaten. Der Browser, genannt MEDIOVIS, ermöglicht zum einen die Suche im Bibliothekskatalog und eröffnet zum anderen unterschiedliche Sichtweisen auf die angezeigten Daten mit dem Ziel, dem Nutzer einen möglichst schnellen, einfachen und dennoch vielseitigen Zugang zu relevanter Information zu bieten. MEDIOVIS wird mittlerweile erfolgreich in der Mediothek Konstanz eingesetzt, welche eine Abteilung der gesamten Bibliothek darstellt und neben Büchern und Zeitschriften auch zahlreiche audiovisuelle Medien wie CDs, Videos oder DVDs anbietet. Die vorhandenen Bibliotheksdaten wurden durch weitere externe Daten angereichert, um dem Nutzer auch auf inhaltlicher Ebene Mehrwerte zu bieten. MEDIOVIS wurde von Anfang an generisch konzipiert, dass es jederzeit an andere Bibliotheksdatenbanken und andere Anwendungsdomänen angepasst werden kann.

Abstract

This Master Thesis illustrates the conception, development and implementation of a browser for Meta Data in libraries. The browser, which is called MEDIOVIS, provides on the one hand search facilities in the library data base and offers on the other hand different views on the displayed data, aiming to give the user a quick, simple and still versatile access to relevant information. MEDIOVIS was successfully established in the Mediothek Konstanz, a section of the library which also offers numerous audiovisual media such as CDs, videos and DVDs beside books and papers. The existing library data was enriched by external data in order to give the system additional value on content base. MEDIOVIS was designed generically by the beginning so that it can easily be adapted to other library data bases and application domains.

Inhaltsverzeichnis

Kapitel 1: Einleitung

Bei der Entwicklung visueller Wissenswerkzeuge steht man oft vor dem Problem, aussagekräftige Datenbestände zu ermitteln, die den Nutzen innovativer Visualisierungen bekräftigen. Die Daten, die in digitalen Bibliothekskatalogen gespeichert werden, bieten aufgrund ihres Umfangs und ihrer Heterogenität eine interessante Spielwiese für die Entwicklung von Recherche- und Visualisierungswerkzeugen. Die große Anzahl an Bibliotheksnutzern erleichtert zudem die Durchführung von Evaluationen der Suchsysteme. Die Benutzoberfläche für das bibliothekarische Recherchesystem sollte einen besonders hohen Grad an Benutzbarkeit aufweisen, da die Nutzer gewöhnlich über sehr unterschiedliche Vorkenntnisse verfügen und nur selten die Logik von Bibliothekskatalogen kennen.

Es ist daher erstaunlich und fast alarmierend, dass die meisten Suchoberflächen der Bibliothekskataloge, die sich heute im Einsatz befinden, einen eher verstaubten und unflexiblen Eindruck machen, da sie zum einen noch stark an der internen Struktur und Logik der Bibliothekskataloge verhaftet sind und zum anderen ästhetisch nicht sehr ansprechend sind. Da viele Kataloge noch vor wenigen Jahren nur über textbasierte DOS-Oberflächen abgefragt werden konnten und heute schon bibliotheksübergreifend über das Internet erreichbar sind – vorbildhaft ist in diesem Zusammenhang der Karlsruher Virtuelle Katalog, eine textuelle, listenbasierte Meta-Suchmaschine für Bibliotheks-Kataloge[1] – kann man damit rechnen, dass sich das Angebot über die nächsten Jahre deutlich verbessern wird. Die Applikation MEDIOVIS, welche im Rahmen dieser Master-Arbeit für die Konstanzer Bibliothek entwickelt wurde und aufgrund ihres generischen Ansatzes leicht an die Datenbestände anderer Bibliotheken angepasst werden kann, soll nicht zuletzt einen weiteren, innovativen Beitrag zur benutzerfreundlichen Gestaltung von Bibliothekskatalogen darstellen.

1.1 Aufbau

Im Kapitel »Theoretische Grundlagen« wird die Geschichte des Bibliothekswesens und der Katalogisierung bis in die Antike zurückverfolgt. Die geschichtliche Entwicklung soll veranschaulichen, warum es auch heute noch deutliche Divergenzen zwischen der Qualität web-

[1] KVK Karlsruher Virtueller Katalog: http://www.ubka.uni-karlsruhe.de/kvk.html

basierter Suchmaschinen und den Rechercheoberflächen von Bibliothekskatalogen gibt. Die weiteren Unterkapitel sind der Definition einiger notwendiger Begriffe gewidmet, die in dieser Arbeit von zentraler Bedeutung sind.

Das Kapitel »Projekte« geht auf vergangene Arbeiten der Arbeitsgruppe MCI ein, die einen entscheidenden Beitrag zur Entwicklung von MEDIOVIS geleistet haben. Einen besonderen Schwerpunkt sollen dabei die Erfahrungen einnehmen, die im Rahmen dieser Projekte gemacht worden sind und MEDIOVIS beeinflusst haben.

Die Konzeption und Umsetzung von MEDIOVIS wird im nächsten Kapitel beleuchtet. Als erstes werden die Ergebnisse der Meetings und Brainstormings vorgestellt, dann werden die einzelnen Schritte beschrieben, die von ständigen Evaluationen begleitet waren, mit dem Zweck, das Interesse der Nutzer nicht aus den Augen zu verlieren.

Das Kapitel »Evaluation« geht näher auf die Testverfahren ein, die MEDIOVIS beeinflusst haben und die Effizienz des Systems gewährleisten sollten. Neben der intellektuellen Auswertung einzelner Tests wird auch die DROID-Technik vorgestellt, die speziell für MEDIOVIS entwickelt wurde und eine automatische Auswertung der Programmnutzung auf Datenbank-Basis gestattet.

Das letzte Kapitel fasst die Erfahrungen zusammen, die im Rahmen des Projekts gemacht worden sind, und gibt einen Ausblick auf das Potenzial, das in der Weiterentwicklung von MEDIOVIS steckt.

1.2 Abgrenzung

Da die Bibliothekslandschaft allein in Deutschland sehr breit gefächert ist, wird die Arbeit keinen Einblick in andere Projekte liefern können, die ebenfalls die Visualisierung von Bibliotheksdaten zum Schwerpunkt haben. Die Projekte der Arbeitsgruppe MCI, die MEDIOVIS vorausgegangen sind, werden ebenfalls nur knapp umrissen, da bereits zahlreiche Arbeiten verfasst worden sind, die diese Projekte schon umfassend beschreiben.[2]

[2] Publikationen des HCI –Lehrstuhls, Universität Konstanz:
http://hci.uni-konstanz.de/index.php?a=publications

Kapitel 2: Theoretische Grundlagen

Dieses Kapitel soll einen kleinen Überblick in die Begriffswelt der Mensch Computer Interaction geben. Konkretere wissenschaftliche Ansätze, die sich beispielsweise bestimmten Interaktionstechniken widmen, werden erst an späterer Stelle vorgestellt.

2.1 Bibliothekswesen

Die Geschichte des Bibliothekswesens lässt sich bis in das Altertum zurückverfolgen. Die Mesopotamier gelten als das Volk mit der für damalige Verhältnisse komplexesten Bürokratie. Zudem spricht man ihnen heute einen ausgeprägten Tradierungswillen nach, der für die Entstehung der ersten Archive ca. 2000 Jahre v. Chr. verantwortlich war. Diese Archive enthielten Tontafeln, die mit Keilschrift beschrieben waren, und einige der Tontafeln – dies erscheint heute besonders überraschend – wurden schon genutzt, um den Archivbestand ansatzweise zu katalogisieren. Assurbanipal, der letzte große Herrscher Assyriens, schuf die erste systematisch zusammengestellte Bibliothek des Altertums, die schon ca. 1500 Einzeltitel umfasste[3]. 37 v. Chr. gründete der Römer Gaius Asinius Pollio aus eigenen Mitteln die erste öffentliche Bibliothek, um das damalige Wissen einer größeren Volksschicht zugänglich zu machen, und im vierten Jahrhundert nach Christus soll es bereits 29 öffentliche Bibliotheken in Rom gegeben haben, die von der intellektuelleren Schicht der Stadt besucht wurden.[4] Während Bibliotheken in den nächsten Jahrhunderten vornehmlich in Klöstern zu finden waren, kam es zur nächsten Revolution mit der Buchdruckkunst.

Die ersten Formen der Katalogisierung wiesen noch keine Systematik auf und stellten lediglich eine beliebige Auflistung aller vorhandenen Papyrusrollen, Tontafeln und sonstiger Schriftwerke da. Die erste eindeutige Dokumentbezeichnung geht auf ein Werk 43 v. Chr. zurück: die Sammleredition von Caesars Gallischen Werken mit Vorwort von Marc Anthony erhielt die eindeutige Nummer »IXIVVIIXVIIIVIIIVIVII«.[5] Heute besitzt jedes Buch die ISBN als eindeutige Kennzeichnung, und neben der ISBN gibt es weitere Standardisierungen wie den Einheitssachtitel, der den Titel eines Buches von anderen unterscheidbar macht. Diese und weitere An-

[3] Imperium Romanum: http://imperiumromanum.com/kultur/bildung/bibliothek_index.htm
[4] 5000 Jahre Bibliotheken: http://biblio.unibe.ch/stub/vorl96
[5] Great Moments in the History of Technical Services: http://sun3.lib.uci.edu/~murrizol/ts_history/tshist.htm

gaben über eine Veröffentlichung, wie das Erscheinungsjahr, den Verlag oder die Seitenzahl, bezeichnet man allgemein als *Metadaten*. Interessant in diesem Zusammenhang ist, dass der Titel eines Dokuments im Umgangssprachlichen oft mit dem Dokument gleichgesetzt wird, während der Titel eigentlich nur eine Dokument-Eigenschaft darstellt. Die Trennung zwischen Metadaten und dem tatsächlichen Dokument, welches durch sie beschrieben wird, ist bei genauerer Betrachtung ebenfalls nicht unproblematisch: der Volltext eines Dokuments wird gewöhnlich nicht zu den Metadaten gezählt, obwohl er ebenfalls nicht mehr als eine Eigenschaft eines Dokuments darstellt. Noch anschaulicher lässt sich die Problematik beim Betrachten von Bildarchiven zeigen: Das Photo eines Gemäldes ist nicht mit dem Gemälde austauschbar, es gibt lediglich ein grobes Abbild des betrachteten Gemäldes wider. Eine verbreitete Definition von Metadaten der Bibliothek Göttingen umgeht diese Problematik elegant: »Unter Metadaten (»Daten über Daten«) versteht man strukturierte Daten, mit deren Hilfe eine Informationsressource beschrieben und dadurch besser auffindbar gemacht wird.«[6] Die Definition legt den Schwerpunkt auf den Suchprozess von Informationsressourcen; somit kann selbst ein Volltext als Metadatum bezeichnet werden, da er dazu dienen kann, das Dokument schneller aufzufinden, indem beispielsweise nach Zitaten gesucht wird, die im Text vorkommen.

Noch in den 80er Jahren wurden die meisten Bibliothekskataloge auf Basis von Karteikarten organisiert, die der Bibliotheksbesucher manuell durchsuchen konnte. Neben der alphabetischen Sortierung wurden oft auch systematische und chronologische Sortierungen angeboten. Dieses Vorgehen verlangte eine strikte Systematisierung, um eine bibliotheksübergreifende Konsistenz der Kataloge zu gewährleisten. In englischsprachigen Ländern wurden die Anglo American Cataloguing Rules, kurz *AACR*, eingeführt, in Deutschland nutzte man die *RAK* (Regeln für die alphabetische Katalogisierung) als Standard und *MAB*/*MAB2* (Maschinelles Austauschformat für Bibliotheken) für elektronische Daten. Die Einführung dieser Standards konnte zwar nicht verhindern, dass dieselben Medien zum Teil weiterhin unterschiedlich katalogisiert wurden – dies ist vor allem auf den großen Umfang der Regelwerke und daraus entstehende Inkonsistenzen zurückzuführen – der eindeutige Vorteil dieses Standardisierungswillens bestätigte sich jedoch, sobald der Computer schließlich die Möglichkeit bot, Metadaten zu ver-

[6] Einführung in Metadaten: http://www2.sub.uni-goettingen.de/intrometa.html

walten und suchbar zu machen: die stark strukturierte Katalogisierung der erfassten Medien ließ sich sehr leicht in digitale Datenbestände überführen.

Die Bibliothek der Universität Konstanz begann schon seit ihrer Gründung 1966 mit der maschinenlesbaren Erfassung der Katalogdaten[7]. Zunächst wurde das von der Bibliothek selbst entwickelte System KOBAS verwendet. 1987 wurden die damals ca. 1 Million Titel in den bibliotheksübergreifenden SWB-Katalog (Südwestdeutscher Katalogverbund) übernommen. Das lokale Suchsystem KOALA (Konstanzer Ausleih- und Anfragesystem) ist ebenfalls eine Eigenentwicklung der Universität, welches 2000 modernisiert wurde und seitdem über das Internet zugänglich ist.[8] Da die Bibliothek mittlerweile über 2 Millionen Titel verfügt und die Grenzen von KOALA sichtbar werden, soll das Katalog-System ab ca. 2005 durch ein neues System namens LIBERO WEBOPAC ersetzt werden, das von einer australischen Firma entwickelt wurde und ironischerweise ebenfalls den Koala-Bären als Logo benutzt.[9] Das neue Katalog-System zeichnet sich vor allem durch seine Geschwindigkeit aus und bietet auch sonst einige Verbesserungen zu KOALA, ist aber aus Usability-Sicht auf den ersten Blick eher enttäuschend. In Kapitel 4.1 werde ich näher auf die Funktionalität der zwei Suchsysteme eingehen, um zu zeigen, welche Konsequenzen sich daraus für MEDIOVIS ergeben haben.

Der große Anteil der heutigen Suchoberflächen für Bibliothekskataloge hält sich immer noch stark an die interne Katalogstruktur und macht eine strikte Trennung zwischen den erfassten Metadaten. Man muss sich beispielsweise vor der Suche entscheiden, ob man den Titel oder Autor sucht. Diese Trennung mag auf den ersten Eindruck hin sinnvoll erscheinen, wenn man jedoch zugleich Dokumente von einem und über einen Autor finden will, scheitert diese Suchstrategie. Die ersten Tests mit MEDIOVIS-Prototypen unterstreichen diese Problematik: es war auffallend, dass die Nutzer mittlerweile eine andere Suchstrategie zu verfolgen scheinen als dies früher der Fall war: neben der expliziten Suche nach Titeln oder Autoren wurden vornehmlich Freitextbegriffe eingegeben, denen konventionelle Metadaten-Kategorien nicht mehr gerecht werden können. Die Erfolge der webbasierten Suchmaschinen, die meist auf der Indexierung kompletter Internet-Dokumente basieren und nur selten eine Trennung zwischen Da-

---

[7] Bibliothek der Universität Konstanz – Bibliotheksprofil: http://www.ub.uni-konstanz.de/bibprofil-2004.pdf

[8] Lokaler Katalog (Koala): http://www.ub.uni-konstanz.de/koala/

[9] Libero Head Office: http://www.libero.com.au/, Lib-IT: http://www.lib-it.de/

ten und Metadaten vornehmen, haben u. a. auch dazu geführt, dass die Nutzer weniger in Kategorien denken und vielmehr konkrete Fragestellungen beantwortet bekommen wollen. Vorträge zur Nutzung von Bibliothekskatalogen, die auf einer Fortbildungsveranstaltung des VdB (Verein Deutscher Bibliothekare e.V.) am 5. Juli 2004 gehalten wurden[10], schienen diese Annahme zu bestätigen.

2.2 Mensch-Computer-Interaktion

Die Interaktion zwischen Mensch und Computer ist ein junges Forschungsgebiet, welches sich zum Ziel gesetzt hat, anhand interdisziplinärer Erkenntnisse aus der Informatik, Psychologie, Soziologie und Design Systeme zu entwickeln, die sich an der menschlichen Wahrnehmung und Auffassungsgabe orientieren. *Mensch-Computer-Interaktion* (MCI) kann somit definiert werden als »a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.« [ACM92] Während die Interaktion mit den ersten Computer-Systemen vornehmlich durch die technische Machbarkeit bestimmt war und nur durch Experten, die die Logik des Computers kannten, nachvollzogen werden konnte, ermöglicht der technische Fortschritt es uns heute, benutzerfreundliche Interfaces zu entwickeln, die sich an der menschlichen Wahrnehmungsfähigkeit orientieren.

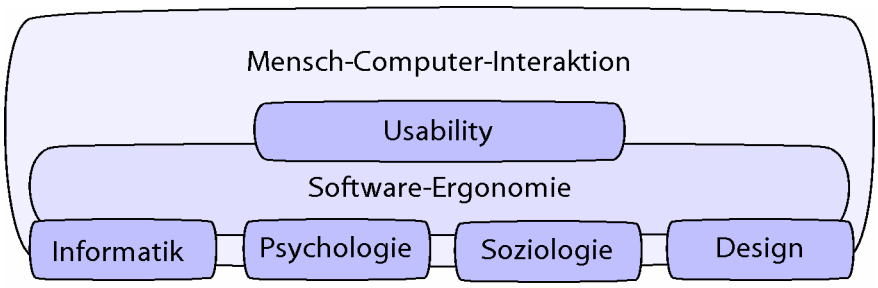


Abb. 1: Eingliederung der Begrifflichkeiten im MCI-Umfeld

Eine klare Eingliederung der Begrifflichkeiten im MCI-Umfeld fällt schwer, da viele Begriffe gleichbedeutend angewandt werden und gerade durch die Interdisziplinität in verschiedenen Kontexten zu finden sind. Im Rahmen dieser Arbeit soll MCI als Überbegriff für weitere wis-

[10] Der OPAC der Zukunft - neue Wege der Erschließung: http://www.vdb-online.org/landesverbaende/sw/berichte/2004-fortbildung-stuttgart.php

senschaftliche Instanzen betrachtet werden. Die *Software-Ergonomie* beschäftigt sich demnach explizit mit der Frage, wie die Erkenntnisse anderer Forschungsgebiete genutzt werden können, um benutzerzentrierte Software zu entwickeln. Die *Usability* stellt die zentrale Schnittstelle zwischen der Mensch-Computer-Interaktion und der Software-Ergonomie dar.

2.2.1 Usability

Als Usability bezeichnet man im Allgemeinen die Qualität der Benutzbarkeit beliebiger Produkte im Sinne ihrer Effektivität, Effizienz und Zufriedenstellung [ISO 9241-11]. Das Ziel einer usability-orientierten Software-Entwicklung sollte also vereinfacht ausgedrückt sein, die Benutzung eines Programms so unkompliziert und angenehm wie möglich zu gestalten. Je ausgereifter die Usability eines Systems ist, desto weniger fallen dem Nutzer Systemschwächen überhaupt auf, da er weniger Zeit damit verbringen muss, sich mit der internen Logik seines Werkzeugs auseinanderzusetzen. Auch der Unterhaltungsfaktor spielt in der Usability eine immer größere Rolle, da ein Produkt, mit dem man gerne arbeitet, verständlicherweise zugleich die intrinsische Motivation erhöht, sich damit auseinanderzusetzen.

Da MEDIOVIS von einem breiten Publikum mit sehr unterschiedlichen Computer-Vorkenntnissen genutzt werden sollte, hatte die dahinter stehende Usability eine hohe Priorität. Die VISMEB-Oberfläche, auf der MEDIOVIS aufbaut, wurde in der Konzeptionsphase immer weiter auf elementare Funktionen reduziert, da die ersten Tests der System-Prototypen die Vermutung bestätigt haben, dass der gewöhnliche Nutzer durch zu viele Interaktionsangebote eher abgeschreckt als motiviert wird. Nach dieser »Entschlackung« wurden nach und nach neue Funktionen ergänzt, ohne dass ein einfacher Zugang zur Standard-Oberfläche verhindert wurde. Unterstrichen werden soll an dieser Stelle, dass den Nutzern von MEDIOVIS nicht die Kompetenz aberkannt werden sollte, auch mit komplizierteren Visualisierungen umgehen zu können. Es stand vielmehr die Beobachtung im Vordergrund, dass Bibliothekskataloge nur sporadisch abgefragt werden und schnelle Antworten liefern sollen. Daher bestand die Kunst bei der Konzeption des Systems darin, konkrete Anfragen schnell zu beantworten und die Benutzer dennoch spielerisch zur gründlicheren Exploration des Bibliotheksbestandes zu verleiten.

Die einfache Benutzbarkeit des Systems sollte durch kontinuierliche Tests mit Nutzern, die zuvor noch nicht mit dem System gearbeitet hatten, sichergestellt werden. Die große Anzahl an Universitäts-Studenten, die mit der Problematik des Suchens von Bibliotheksmedien vertraut sind, bot dafür eine optimale Basis.

2.3 Informationsvisualisierung

Da die *Informationsvisualisierung (IV)* immer noch als neue wissenschaftliche Disziplin bezeichnet werden kann, wird sie auch sehr unterschiedlich definiert. Im Allgemeinen bezeichnet sie die »Sichtbarmachung«, also die visuelle Darstellung von Informationen, die dem Nutzer einen besseren Zugang zu abstrakten Daten ermöglichen soll. Da die Erstellung von Visualisierungen heute zumeist durch den Computer geschieht und die Visualisierungen somit durch den Nutzer dynamisch beeinflusst werden können, definiert Shneiderman Informationsvisualisierung als »The use of computer-supported, interactive, visual representations of abstract data to amplify cognition.« [CMS99] Die Definition stellt neben der bloßen Sichtbarmachung von Informationen heraus, dass Visualisierungen prinzipiell kognitive Mehrwerte bieten und dem Benutzer Zusammenhänge begreiflich machen können, die durch das Betrachten abstrakter Daten kaum oder gar nicht erfasst werden können. Informationsvisualisierung geht also über die reine Wiedergabe abstrakter Daten in visueller Form hinaus. Sie soll durch geeignete Darstellungsformen neue Einblicke in Daten geben können. Bezogen auf MEDIOVIS entstand daraus zwei Ansätze: zum einen sollten die recherchierten Daten in einer ansprechenden und schnell erfassbaren Art und Weise dargestellt werden, zum anderen sollten auch neue Repräsentationsformen geschaffen werden, die es den Benutzern ermöglicht, informationelle Mehrwerte und Zusammenhänge zwischen den Daten zu erkennen.

2.3.1 Visualisierungsmodell

Um Daten in einem visuellen Wissenswerkzeug darstellen zu können, müssen sie erst in brauchbare interne Strukturen überführt und richtig interpretiert werden. Ein oft zitiertes und praktisch direkt umsetzbares Modell zur Veranschaulichung der internen Datenverarbeitung von den ursprünglichen Daten bis zu ihrer Visualisierung ist in Abb. 2 dargestellt.

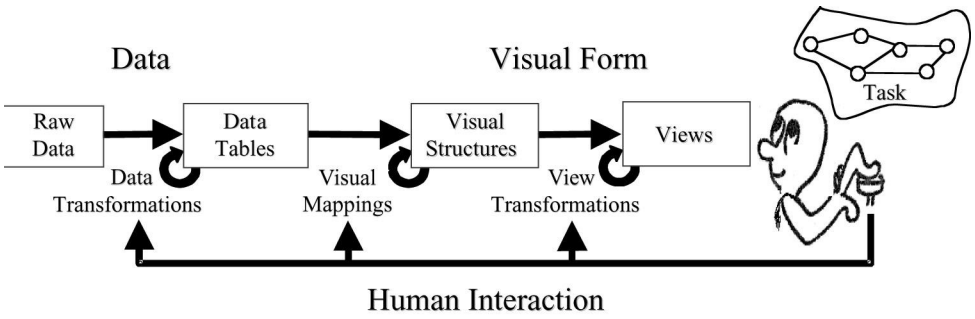


Abb. 2: Referenz-Visualisierungsmodell [CMS99]

Es definiert drei notwendige Schritte, um die Ursprungsdaten in Visualisierungen abbilden zu können und dem Nutzer gleichzeitig eine flexible Interaktion zu ermöglichen:

- Die Rohdaten werden als erstes in leichter handhabbare Datentabellen gebracht. Diesen Prozess bezeichnet man als *Data Transformations*. Er wird nach dem Start einer neuen Suche oder der Filterung eingelesener Daten in Gang gesetzt. Je nach Struktur der Rohdaten ist der Transformationsprozess in Datentabellen unterschiedlich aufwändig. Wenn Rohdaten schon in tabellenähnlichen Strukturen vorliegen, können sie relativ einfach in interne Datentabellen übernommen werden; falls sie jedoch völlig unstrukturiert oder multidimensional sind und sich der perzeptionellen Vorstellungskraft entziehen, fällt der Transformationsprozess dementsprechend komplexer aus.
- Der nächste Schritt, der die Datentabellen in V*isual Mappings* überführt, ist gewöhnlich der aufwändigste, da jede Visualisierung die Daten in unterschiedlichen Strukturen abbildet. Eine tabellarische Visualisierung der Daten weist Parallelen zur internen Datentabelle auf, ein Scatterplot stellt Datensätze hingegen als Punkte in einem Koordinatensystem dar. Zur Darstellung der Datenpunkte muss die interne Datenstruktur um weitere Informationen erweitert werden: die Position eines Datenpunktes innerhalb des Koordinatensystems, das Aussehen eines Datenpunkts, Angaben, ob sich mehrere Datenpunkte überdecken usw. Neue Visual Mappings werden jedes Mal erzeugt, wenn der Benutzer andere Daten in Visualisierungen sehen will und zum Beispiel neue Achsenbelegungen in einem Scatterplot auswählt.
- Die endgültige Repräsentation der visuellen Strukturen innerhalb so genannter *Views* nennt man die *View Transformation*. Ein View ist die konkrete Ansicht einer Visualisierung und besitzt neue Eigenschaften wie Position und Größe am Bildschirm. Die View Transformation wird durch jede Benutzer-Interaktion innerhalb der Views in Gang gesetzt. Wenn der Benutzer die Größe eines Views verändert, einen Datensatz auswählt oder einen Volltext scrollt, findet eine View Transformation statt, und ein View muss neu gezeichnet werden.

Viele wissenschaftliche Experten-Werkzeuge verarbeiten Daten, die zu komplex sind, um in ihrer Rohform interpretiert werden zu können. Aufwändige Datentransformationen sind somit unumgänglich, um die Bedeutung hinter den Daten sichtbar zu machen. Bibliotheksdaten stellen hingegen einfach verständliche Daten dar. Der Großteil dieser Daten, wie Titel oder Autor eines Dokuments, ist textbasiert, einige Informationen wie das Erscheinungsjahr oder die Anzahl der Ausleihen eines Mediums sind numerischer Natur. Alle Daten lassen sich zudem in zweidimensionalen Tabellen darstellen, verfügen also über keine Hierarchien oder gegenseitige

Verknüpfungen. Komplexe Verfahren zur Vereinfachung und Aggregation der Daten sind somit unnötig, und der *Data Transformation*-Prozess besteht vorrangig aus einem Auswahlprozess der relevanten Daten. Das *Visual Mapping* der Daten stellt den anspruchsvollsten Prozess im Visualisierungsmodell dar, da ein ausgefeiltes Abbildungsmodell die Flexibilität einer Visualisierung erhöht und dem Nutzer mehr Interaktionsmöglichkeiten bietet. Die *View Transformations* sind für die Darstellung der Visualisierungen zuständig und müssen möglichst performant programmiert sein, so dass der Nutzer im Idealfall keine Verzögerungen bei der Arbeit wahrnimmt.

2.4 Information Retrieval

Neben der Visualisierung der relevanten Daten muss ein System auch Recherchemöglichkeiten bieten, um den vorhandenen Datenbestand nach relevanten Treffern durchsuchen zu können. Derartige Suchsysteme fasst man unter dem Begriff *Information Retrieval Systeme (IR-Systeme)* zusammen. Die Forschungsbereiche IV und IR wachsen immer mehr zusammen, da ein Visualisierungswerkzeug ohne relevante Daten keinen besonderen Mehrwert bietet und IR-Systeme im Gegenzug Visualisierungen benötigen, um die Daten darstellen zu können.

Die am weitesten verbreiteten IRS sind die web-basierten Suchmaschinen. Die Qualität eines IR-Systems ist von verschiedenen Kriterien abhängig:

- Die Eigenschaften der vorliegenden Datenbasis beeinflussen den weiteren Aufbau und die Funktionsweise eines Retrieval-Systems. Internet-Texte verfügen beispielsweise über keine einheitliche, inhaltliche Struktur, daher stehen sie konventionellen Volltexten am nächsten, und ein Retrieval-System sollte über entsprechende Algorithmen verfügen, die Volltexte effizient abfragen können. Adressdatenbanken weisen wiederum eine sehr festgelegte Struktur auf, und da der Nutzer sich gewöhnlich schon bei der Anfrage entscheidet, ob er nach Namen, Straßen oder Telefonnummern suchen will, sollte auch das Retrieval-System kategorienbasiert arbeiten. In wissenschaftlichen Anwendungsbereichen ist es hingegen oft von Interesse, aus den komplexen Datenbeständen erst nach dem Start einer Suche neue Daten zu generieren, die dem Nutzer konkrete Fragestellungen beantworten können.
- Das wichtigste Kriterium ist das konkrete Informationsbedürfnis des Nutzers. Das Potenzial eines IRS darf durch die Logik der Datenbasis nicht so weit eingeschränkt werden, dass der Nutzer gezwungen ist, den Aufbau des Datenbestandes zu verstehen, um erfolgreiche Recherchen durchzuführen. Im Optimalfall sollte jede Recherche des Nutzers die relevantesten

Treffer aus dem Datenbestand erbringen. Dieses Ziel ist selbstverständlich in den meisten Fällen nur annähernd zu erreichen, da sonst ein Recherche-System in der Lage sein müsste, die Nutzeranfrage nach menschlichen, intelligenten Kriterien zu interpretieren.

- Aufgrund der fehlenden Intelligenz des Computers muss ein Kompromiss gefunden werden. Umfangreichere Eingabemasken können helfen, die Trefferqualität zu erhöhen. Je mehr der Nutzer bei seiner Anfrage gesteuert wird, desto kleiner ist die Gefahr, dass die Nutzeranfragen fehlinterpretiert werden. Gleichzeitig sinkt jedoch auch die Wahrscheinlichkeit, dass der Nutzer wirklich die Treffer erhält, die ihn tatsächlich interessieren. Der Nutzer sollte grundsätzlich nicht durch zu komplexe Anfragemöglichkeiten überfordert werden.
- Das Potenzial eines IR-Systems zeigt sich spätestens bei der Ausgabe der gefundenen Treffer. Abhängig von der Komplexität und Eindeutigkeit einer Suche muss ein IRS über mehr oder weniger ausgefeilte Verfahren verfügen, um brauchbare Treffer zurückliefern zu können. Um dieses Ziel zu erreichen, werden dem Programmierer neben der technischen Kompetenz auch Kenntnisse über das Suchverhalten der Nutzer abverlangt.

Die Recherche-Funktionen von MEDIOVIS können sicher nicht mit dem Potenzial professioneller Retrieval-Systeme mithalten, dennoch konnten sie individueller gestaltet werden als das Rechercheangebot, das konventionelle Bibliothekssysteme anbieten. Um den Benutzeranfragen möglichst gerecht zu werden, wurde eine zweistufige Anfrage-Architektur entwickelt: im ersten Schritt holt sich das System eine grobe Treffermenge von der Datenbank mittels einer Standard-SQL-Anfrage. Im zweiten Schritt werden die angeforderten Treffer weiter reduziert und verfeinert. Dieses Vorgehen hat einige Vorteile gegenüber einer konventionellen Datenbankanfrage:

- Fast alle Datenbanken basieren auf dem SQL-Standard. Der Portierungsaufwand beim Übertragen von MEDIOVIS auf andere Bibliotheksdatenbanken wird minimal gehalten.
- Die benutzten SQL-Anfragen sind verhältnismäßig einfach aufgebaut und liefern schnell Treffermengen zurück.
- Die Verfeinerung der Treffermenge ist programmspezifisch und erlaubt genauere Anfragen seitens der Nutzer, die mittels SQL-Anfragen nur beschränkt zu bewältigen sind.

Die programminterne Verfeinerung der Treffermenge geht sehr zügig vonstatten, da alle Treffer direkt im Hauptspeicher verarbeitet werden, und es ergeben sich bei nicht übergroßen Treffer-

mengen keine wahrnehmbaren Geschwindigkeitseinbußen. Weitere Details zur zweistufigen Suchstrategie finden sich in Kapitel 6.2.1.

2.5 Datenbank

Da in dieser Arbeit zahlreiche Begriffe verwendet werden, die dem Umfeld von Datenbanken angehören, werden in diesem Kapitel einige Begrifflichkeiten geklärt und ihr Bezug zu den Bibliotheksdaten veranschaulicht.

Datenbanken werden genutzt, um Daten elektronisch zu speichern und leicht verfügbar zu machen. Die Daten einer Datenbank können durch *Datenbanksysteme* abgefragt werden. Bekannte Datenbanksysteme sind Oracle, Microsoft Access oder PostgreSQL. Die Abkürzung *SQL* steht für »Standard Query Language« und bezeichnet eine standardisierte Anfragesprache, die zahlreiche Datenbanksysteme zur Kommunikation nutzen.

Eine Datenbank kann aus mehreren *Tabellen* bestehen. Jede Tabelle verfügt über Zeilen und Spalten; in den Zeilen werden die *Datensätze* abgelegt. Jeder Datensatz besteht wiederum aus mehreren *Datenfeldern*, welche bestimmten *Attributen*, bzw. *Kategorien* zugeordnet sind. Die Attribute existieren für alle Datensätze und sind über die Spalten einer Tabelle erreichbar. Um Datensätze in einer Tabelle unterscheiden zu können, erhält jeder Datensatz meist eine eindeutige *ID* als Erkennungsmerkmal.

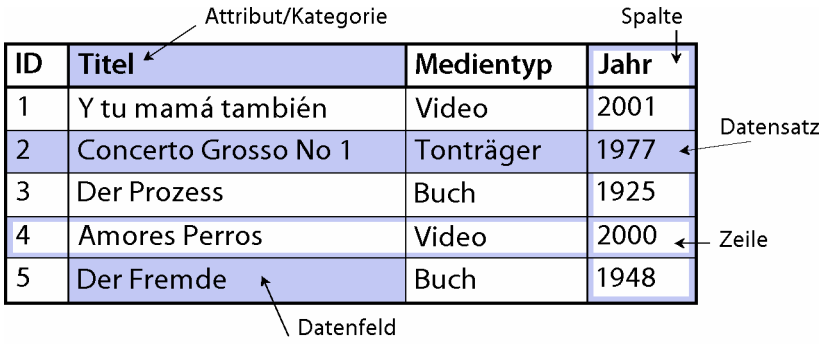


Abb. 3: exemplarischer Aufbau einer Datenbank-Tabelle

Abb. 3 veranschaulicht den Aufbau einer Tabelle. Wenn man sich eine Tabelle als Karteikasten mit Bibliotheksdaten vorstellt, entspricht ein Datensatz einer einzelnen Karteikarte mit allen Angaben zu einem Medium. Die Datenfelder sind die Kategorien eines Bibliothekskatalogs, die z. B. Auskunft über den Titel, den Medientyp oder das Erscheinungsjahr eines Mediums geben.

Im Bibliotheksumfeld wird eine leicht abgewandelte Terminologie benutzt. Besonders irreführend ist der Begriff *Titel*, der sich zum einen auf den Namen von Medien bezieht (Buchtitel, Filmtitel usw.), zum anderen aber auch für ganze Datensätze steht. Vereinzelt findet man für Datensätze auch die Bezeichnung Titelaufnahme. Um Missverständnissen vorzubeugen, werden *Datensätze* in dieser Arbeit entweder direkt als Datensätze, als Bibliothekstitel oder (an passender Stelle) vereinfacht als Medien bezeichnet. Wenn die Bezeichnung *Titel* verwendet wird, bezieht er sich hier explizit auf den Titel eines bibliothekarischen Datensatzes.

Für *Attribute* wird im Bibliotheksjargon gewöhnlich die Bezeichnung *Kategorie* verwendet. Die Datenbank einer Bibliothek wird gerne mit dem *Bibliothekskatalog* gleichgesetzt, obwohl der Katalog streng genommen nur in der Datenbank abgelegt ist.

2.6 Interaktionstechniken

Dieses Kapitel soll lediglich als kurze Einleitung zur Thematik verschiedener Interaktionstechniken dienen; detailliertere Informationen zu den eingesetzten Techniken finden sich erst später in Kapitel 5.4, da die Kenntnis des Programmaufbaus und der eingesetzten Visualisierungen von MEDIOVIS zum besseren Verständnis der Interaktionen hilfreich ist.

2.6.1 Direct Manipulation

Shneiderman führte 1983 den zentralen Begriff der *direct manipulation* ein, worunter er drei Anforderungen an die Umsetzung von Benutzeraktionen stellt [BC83]:

- die interessierenden Objekte und Aktionen sollten immer direkt sichtbar sein
- Aktionen sollten schnell, umkehrbar und stufenweise durchgeführt werden können
- komplexe Kommandos sollten durch eine direkte, visuell sichtbare Aktionen ersetzt werden

Durch die Entwicklung graphischer Benutzeroberflächen und die Maus als neues Eingabegerät wurde eine intuitive Möglichkeit geschaffen, visuell mit dem Computer zu interagieren. Ein weiterer großer Schritt waren Textverarbeitungssysteme auf WYSIWYG-Basis[11]. Dennoch wurden Usability-Fragen vorerst relativ wenig berücksichtigt, weshalb Shneiderman 1987 weitere

[11] WYSIWYG ist ein Akronym für »What you see is what you get« und beschreibt Textverarbeitungen, die den Text am Bildschirm genauso darstellen, wie er später gedruckt wird. Die Abkürzung wurde von John Seybold, der die erste graphische Textverarbeitung für Xerox entwickelte, eingeführt.

wünschenswerte Eigenschaften der direkten Manipulation zusammenfasste [BS87]. Besonders hervorgehoben sei hier seine Forderung, dass auf Interaktionen immer sofortige Reaktionen folgen sollten, und die Feststellung, dass Fehlermeldungen bei einer direkt-manipulativen Benutzereingabe selten benötigt werden.

Die direkte Manipulation erwartet somit eine möglichst freie Exploration von Benutzeroberflächen, die den Benutzer nicht steuert, sondern ihm die Wahl gibt, seine Aktionen selbst zu bestimmen und sofortige Rückmeldungen zu erhalten. Unabhängig von Shneiderman und der Forschungsdisziplin der Mensch-Computer-Interaktion stellt beispielsweise auch der Konstruktivismus fest, dass jeder Mensch eigene, subjektive Vorgehensweisen nutzt, um seine individuellen Ziele zu erreichen [GE92], und mit dem Konstruktivismus verbundene Lerntheorien fordern, dass dem Menschen keine starren Konzepte vorgegeben werden sollen, da diese den Menschen nur daran hindern, eigene, individuelle Vorgehensweisen zu verfolgen [BP94]. Die Entwicklung eines explorativen Computer-Systems müsste also von Interaktionsmustern Abstand nehmen, die den Benutzer in bestimmte Bahnen lenken, und es müssten beispielsweise auch dialogorientierte Interaktions-Konzepte in Frage gestellt werden, die dem Benutzer mehrere Schritte anbieten, um eine Aktion zu vollenden, da ein der Computer nicht auf alle individuellen Bedürfnisse des Benutzers reagieren kann. Ein »Dialog« mit dem Computer sieht in den meisten Fällen auch tatsächlich so aus, dass der Computer bestimmte Benutzereingaben erwartet, die in mehrere Einzelschritte aufgeteilt werden. Kommt der Benutzer mit dem Dialog nicht klar, kann er im besten Fall Hilfeinformationen aufrufen oder den Dialog abbrechen. Um dialogorientierte Interaktionsmuster nicht pauschal zu diskreditieren, soll jedoch nicht verschwiegen werden, dass alle Benutzer ein unterschiedlich stark ausgeprägtes Bedürfnis zur freien Exploration mitbringen, was auch den Absolutheitsanspruch der in Form der Lerntheorien konkretisierten Ideologie der konstruktivistischen Weltanschauung in Frage stellt.

In MEDIOVIS wurde auf dialogorientierte Interaktions-Konzepte verzichtet. Alle verfügbaren Visualisierungen sollten eine möglichst hohe Interaktivität aufweisen und Aktionen, die mehrere Schritte verlangten, vermeiden. Falls zukünftige Versionen von MEDIOVIS und Varianten des Systems jedoch wieder komplexere Wege beschreiten und mächtige Visualisierungen aufnehmen, die erst einen gewissen Lernaufwand benötigen, um mit ihnen effektiv arbeiten zu können, sollten komplexere Interaktionsmöglichkeiten wieder neu diskutiert werden.

2.6.2 Multiple Coordinated Views

Einen zentralen Gedanken bei der Umsetzung von MEDIOVIS spielte der Einsatz so genannter *Multiple Coordinated Views (MCV)*, die North als »a set of visualizations and a set of coordinations between the visualizations« definiert [NO00]. Alle Visualisierungen sind untereinander dynamisch verbunden, Aktionen in einer Visualisierung wie beispielsweise die Fokussierung oder Selektion von Titeln führen auch zu visuellen Veränderungen in den anderen Ansichten. Durch die Verwendung von MCVs kann der Benutzer besser nachvollziehen, wie die präsentierten Visualisierungen zusammenhängen und welche Auswirkungen seine Aktionen auf den gesamten Datenbestand haben. MCVs fördern zudem eine konsistente View-Architektur, weil die logischen Beziehungen unter den Visualisierungen klar definiert sein müssen. North unterscheidet zwischen zwei grundlegenden Interaktionsmustern:

- *Selektion:* Daten, die für den Benutzer von Interesse erscheinen, können in einer Visualisierung hervorgehoben und selektiert werden, um womöglich weitere Aktionen mit Ihnen durchzuführen
- *Navigation:* der Benutzer kann in Ansichten navigieren, um verdeckte Daten sichtbar zu machen oder die Ansicht auf Daten zu verändern, z. B. durch Zooming, Panning, Scrolling oder Rotation. Die Navigation ist als Aktionskonzept immer dann angebracht, wenn eine Visualisierung nicht alle Daten komplett im sichtbaren Bereich unterbringen kann, wie dies z. B. in längeren Tabellen der Fall ist.

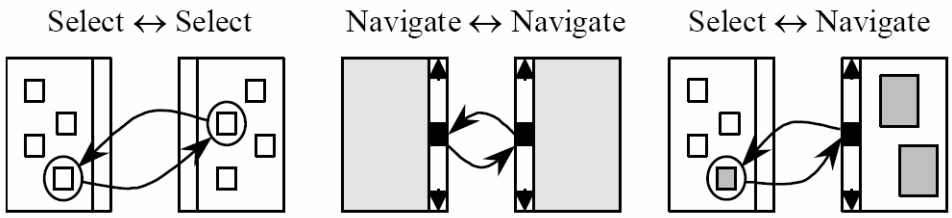


Abb. 4: Taxonomie möglicher MCV-Verknüpfungen[NO97 / NO00]

In Beziehung gebracht ergeben sich drei mögliche Verknüpfungen zwischen den Aktionen (Select ↔ Navigate und Navigate ↔ Select sind zusammengefasst), zu sehen in Abb. 4:

- Select ↔ Select: die Selektion von Daten führt zur Selektion der Daten in der anderen Ansicht.

- Navigate ↔ Navigate: die Navigation in einer Ansicht führt zur Navigation der anderen Ansicht.
- Select ↔ Navigate: die Selektion von Daten führt zur Navigation der anderen Ansicht und umgekehrt

In MEDIOVIS wird eine leicht unterschiedliche Terminologie verwendet, um Aktionen zwischen den Visualisierungen zu definieren (weitere Informationen zur Umsetzung des MCV-Konzepts finden sich in Kapitel 6.1):

- *Fokus:* wenn Daten fokussiert werden, werden sie in den Visualisierungen besonders hervorgehoben. Ein Fokuswechsel hebt die Hervorhebung wieder auf.
- *Selektion:* selektierte Daten werden ständig in den Visualisierungen hervorgehoben und müssen erst explizit wieder deselektiert werden.

Beide Aktionen lassen sich der Select-Aktion zuordnen. Die Navigate-Aktion findet auch in einigen Visualisierungen Anwendung in Form von Scrolling und Zooming, führt jedoch zu keinen Interaktionen mit den anderen Visualisierungen. North stellt in seiner Taxonomie jeder Aktion eine einzige andere gegenüber, in der Tat kann eine Aktion jedoch zu mehreren Aktionen in anderen Visualisierungen führen:

- Select ↔ Select + Navigate: die Selektion von Daten führt zur Selektion der Daten und Navigation in der anderen Ansicht.
- Navigate ↔ Navigate + Select: die Navigation in einer Ansicht führt zur Navigation und gleichzeitigen Selektion der Daten in der anderen Ansicht.

Es ist beispielsweise sinnvoll, Daten, die in einer Ansicht selektiert werden, in den Ansichten zu selektieren und gleichzeitig mittels Navigation in den sichtbaren Bereich zu schieben.

Weitere in MEDIOVIS eingesetzte Interaktionstechniken wie *Linking & Brushing* oder *Focus & Context* werden in Kapitel 5 näher erläutert.

Kapitel 3: Projekte

Der Entwicklung von MEDIOVIS gehen langjährige Forschungsarbeiten in der Arbeitsgruppe MCI voraus. Im September 1998 wurde das EU-Projekt INSYDER lanciert und Ende 2001 durch ein weiteres EU-Projekt INVISIP fortgesetzt, aus welchem der eigenständige Metadaten-Browser VISMEB entstand, auf dessen Architektur MEDIOVIS basiert. Die einzelnen Projekte werden hier mit Schwerpunkt auf die implementierten Visualisierungen kurz vorgestellt. Des Weiteren werden die Programm-Features, die besonderen Einfluss auf MEDIOVIS ausgeübt haben, besonders hervorgehoben.

3.1 Insyder

INSYDER entstand als Teilprojekt des EU-Programms ESPRIT[12]. INSYDER sollte kleinen und mittelständischen Unternehmen die Suche, Visualisierung und Analyse von wirtschaftlich relevanten Daten im Internet erleichtern. Ein Thesaurus und ein semantischer Agent, welche die Suchanfrage des Benutzers interpretieren, übernehmen die Formulierung der Suchanfrage, die an mehrere Suchmaschinen weitergeleitet wird. Nach einer anschließenden Aggregation der Treffermenge und der Filterung der Treffer nach Kriterien, die u. a. durch das Benutzerprofil festgelegt sind, werden die relevanten Internet-Seiten mittels verschiedener Visualisierungen dargestellt. Der Mehrwert zu konventionellen Suchmaschinen im Netz besteht also zum einen im Einsatz linguistischer Recherche-Verfahren zur Unterstützung der Suche und zum anderen in den vielfältigen Darstellungsarten der gefundenen Treffer, die im Vergleich zur listenorientierte Darstellung einige Mehrwerte bieten [MRM00], [MT01].

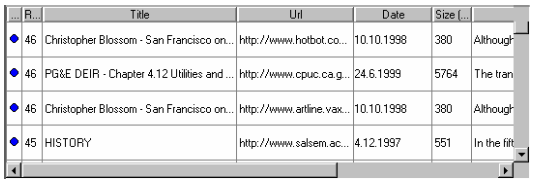
Ein weiteres Feature von INSYDER ist die Speicherung von Benutzerprofilen und Suchvorgängen. Bei einem späteren Aufruf des Systems findet der Benutzer wieder seine so genannten *Spheres of Interest (SOI)* vor, die ähnlich zum Windows Explorer als Baumstruktur visualisiert sind. Neue Suchen können mit alten Suchbegriffen durchgeführt werden, um festzustellen, ob

[12] INVISIP – Internet Système de Rechèrche
ESPRIT Software Technologies, Projektnr. #29232,
s. auch: http://hci.uni-konstanz.de/index.php?a=research&b=projects&c=1810650

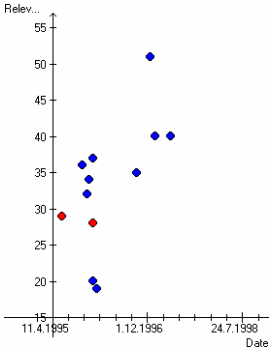
sich das Informationsangebot eines SOI verändert hat. Die Ergebnisse der Suche können außerdem abgespeichert und in andere Formate exportiert werden.

Vier unterschiedliche Visualisierungen wurden in das System eingebettet:

- die *Result Table* stellt die Metadaten in der verhältnismäßig vertrauten tabellarischen Ansicht dar. Daten können in der Result Table einfach sortiert werden; die Dokumente können beispielsweise nach ihrer Größe oder ihrem Alter angeordnet werden. Ein weiterer Vorteil der tabellarischen Darstellung ist, dass die meisten Nutzer schon mit Tabellen gearbeitet haben und somit wenig Einarbeitungszeit benötigen, um mit der Visualisierung schnell die gewünschten Ziele zu erreichen. Dies konnte in den Evaluationen bestätigt werden, die nach dem Abschluss des Projekts durchgeführt wurden [MT01].

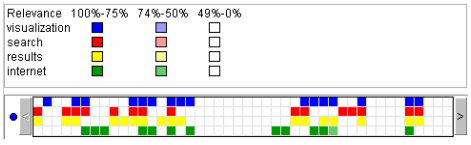


- der *Scatterplot* stellt alle Dokumente zweidimensional in einem Koordinatensystem dar; der X- und Y-Achse ist jeweils ein Metadaten-Attribut zugeordnet, in diesem Beispiel die Relevanz und das Alter eines Dokuments. Neue und relevante Dokumente sind in dieser Ansicht rechts oben zu finden. Die Scatterplot-Darstellungsform eignet sich besonders, um Zusammenhänge zwischen Dokumenten und Cluster-Bildungen erkennen zu können. Die Achsenbelegungen können zudem frei ausgewählt werden; eine Achsenauswahl der Dokumentsprache veranschaulicht beispielsweise auf einen Blick, in welcher Sprache die meisten Dokumente zu finden sind.



- die *Bar Chart*-Visualisierung stellt alle Dokumente ebenfalls tabellarisch dar. Die Zellen der Tabelle enthalten keine textuellen Informationen, sondern graphische Interpretationen der zu visualisierenden Daten, wie in diesem Beispiel die Relevanz eines Dokuments (Gesamtrelevanz und Relevanz einzelner Suchbegriffe). Bar Charts haben Überblickscharakter, da sie große Datenmengen gleichzeitig am Bildschirm darstellen können.

- der *Segment View* zeigt die Verteilung der Suchbegriffe innerhalb eines Dokuments an. Alle Dokumente werden in Segmente eingeteilt, die in



der Visualisierung von links nach rechts abgetragen werden. Die verschiedenen Farben verweisen auf die Suchbegriffe, die der Benutzer eingegeben hat. Besonders in großen Textdokumenten können relevante Textstellen mit dieser Visualisierung schneller aufgefunden werden.

Die Oberflächenprogrammierung von INSYDER wurde mit JAVA durchgeführt, um eine zügige Entwicklung zu ermöglichen, die Such-Agenten wiederum basieren auf C++, um den Geschwindigkeitsvorteil der systemnahen Sprache zu nutzen. Die Visualisierungen wurden alle in INVISIP/VISMEB übernommen und zum Teil in abgewandelter Form implementiert, und die *Result Table* und *Scatterplot*-Visualisierungen finden sich auch in MEDIOVIS wieder.

3.2 Invisip

Während INSYDER auf die Suche von Internet-Dokumenten spezialisiert war, konzentrierte sich ein zweites EU-Forschungsprojekt namens INVISIP, das Ende 2001 ins Leben gerufen wurde, auf die Recherche und Visualisierung geo-räumlicher Metadaten[13]. Zum einen sollte INVISIP als standardisierte, technische Plattform dienen, um den reibungslosen Austausch von Metadaten zu ermöglichen, zum anderen sollten Entscheidungs- und Planungsprozesse, die sich über mehrere Parteien erstreckten, über intuitive und intelligente Visualisierungen vereinfacht und optimiert werden.

INVISIP wurde gänzlich mit JAVA als Programmiersprache umgesetzt und sollte sowohl als Applikation als auch als Applet aufgerufen werden können. Da zum Beginn des Projekts noch keine zu visualisierenden Daten existierten, wurden vorerst exemplarische, im XML-Format gespeicherte INSYDER-Daten als Visualisierungs-Input übernommen. Sobald die ersten richtigen Daten verfügbar waren, wurden die Daten per SQL aus einer Datenbank abgerufen. Als Information Retrieval-Interface wird dem Nutzer anfangs ein Formular präsentiert, in welchem Suchbegriffe eingegeben werden und die Treffer zusätzlich numerisch (z. B. über ein Jahresintervall) und kategorisch (Begrenzung auf bestimmte Sprachen, Domains etc.) eingeschränkt werden können. Später wurde die *Query Preview* als zusätzlich visuelles Suchkonzept eingeführt, welche auch als neue Visualisierung in das System integriert wurde (s. Abb. 6d).

---

[13] INVISIP – Information Visualization in Site Planning
IST – Information Visualization in Site Planning, IST-2000-29640
s. auch: http://hci.uni-konstanz.de/index.php?a=research&b=projects&c=1810649

Die technischen und konzeptuellen Erkenntnisse, die beim INSYDER-Projekt gewonnen wurden, waren hilfreich bei der Konzeption des neuen Projekts. Die Programmlogik wurde konzeptuell übernommen, und die Visualisierungen konnten im Zuge der Ergebnisse einer umfangreichen Evaluation verbessert werden [MT01]. Mehrere Visualisierungen sollten zur gleichen Zeit sichtbar sein, und die Result Table wurde durch die so genannte *SuperTable* ersetzt, die neben textuellen Metadaten auch *Bar Charts* und *Segment Views*, auch *Relevance Curves* genannt, darstellen konnte (Abb. 5). Später wurde die SuperTable durch zwei unterschiedliche Tabellentypen erweitert:

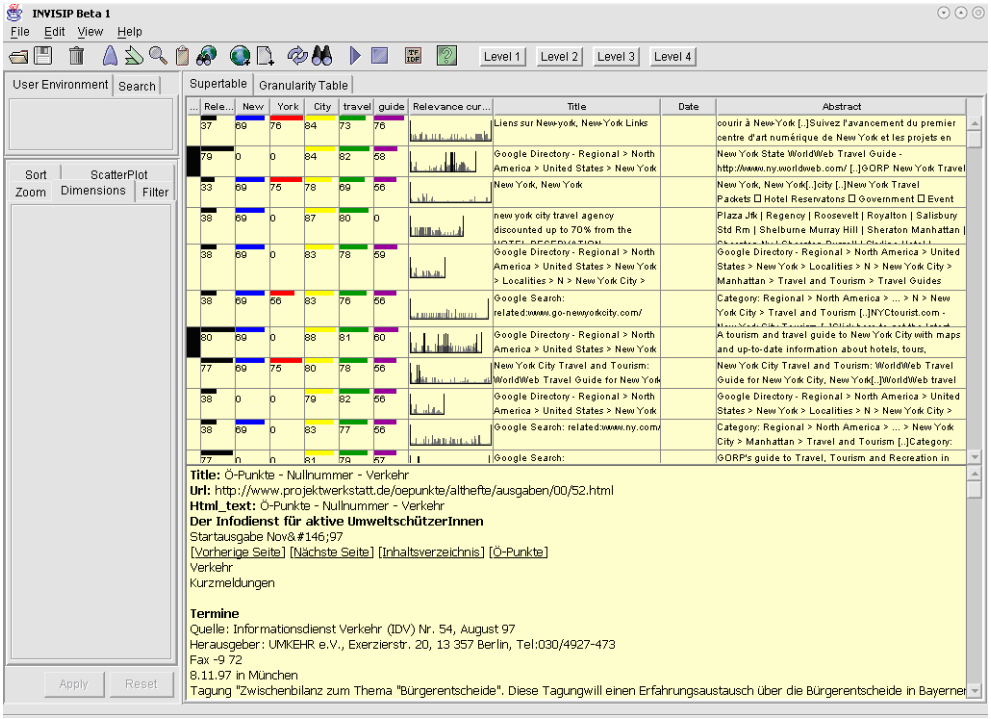


Abb. 5: Screenshot des INVISIP-Projekts; Darstellung der *Level Table* mit integrierten *Bar Charts* und *Relevance Curves* und darunter liegende Volltext-Ansicht

- In der *Level Table* sind die Metadaten auf vier Tabellen verteilt, in denen die Metadaten textuell und graphisch visualisiert werden. Die Levels können abwechselnd aufgerufen und angezeigt werden. Ein höherer Level bietet zugleich einen höheren Detailgrad. Während der erste Level eine graphische Gesamtansicht möglichst vieler Datensätze präsentiert, konzentriert sich der höchste Level auf die Darstellung einzelner Volltexte. In Abb. 5 ist das dritte der vier Levels aktiviert, und neben der Gesamtrelevanz und den Einzelrelevanzen, dem Titel und dem Datum der Datensätze wird auch die Relevance Curve und ein Ausschnitt des Abstracts in jeder Zeile eingeblendet.

- die *GranularityTable* besitzt insgesamt sechs Detailstufen (so genannte Granularitätsstufen), die ebenfalls eine schrittweise Darstellung von einer vorrangig graphisch orientierten Gesamtansicht zur einzelnen Volltextansicht ermöglichen. In der GranularityTable kann die Detailstufe einzelner Zeilen verändert werden, während der Wechsel auf ein anderes Level in der Level Table Auswirkungen auf alle Zeilen hat.

Wie zuvor erwähnt, wurde auch der *Scatterplot* neu in INVISIP integriert [GF02]. Seine Funktionalität wurde erweitert, zum Beispiel enthält er eine Zoom-Funktion, um eng beieinander liegende Datenpunkte besser unterscheiden zu können. Des Weiteren wurden *Movable Filters* implementiert [BSP93] [FS95]. Darunter versteht man kleine, transparente Fenster, hinter denen Daten mit bestimmten Kriterien zeitweise ausgeblendet werden (in Abb. 6a bleiben beispielsweise größere Dokumente unter dem Filter sichtbar).

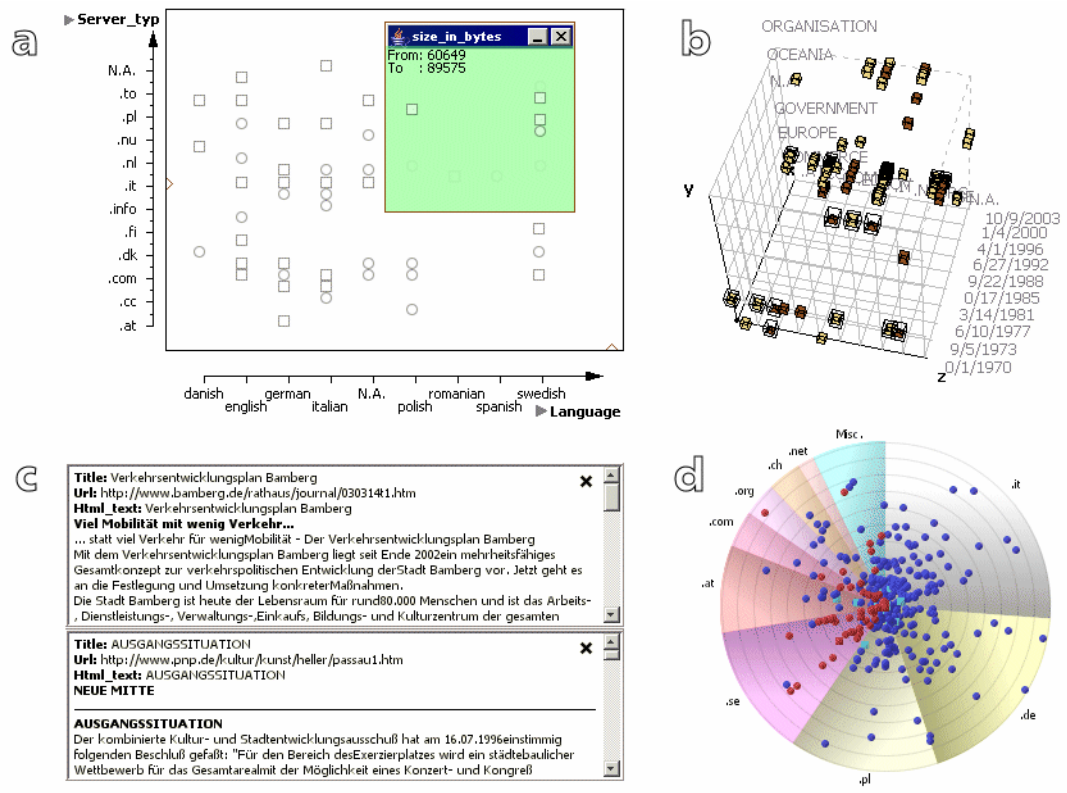


Abb. 6: Visualisierungen in INVISIP/VISMEB:
a) 2D-Scatterplot mit *Movable Filter*, b) *3D-Scatterplot*, c) *Text Browser*, d) *Circle Segment View*

Da ein gewöhnlicher Scatterplot nur zwei Dimensionen darstellen kann, wurde zusätzlich ein *3D-Scatterplot* entwickelt, der beliebig gezoomt und rotiert werden kann (Abb. 6b) [KW03]. Die geo-räumlichen Daten enthalten auch Texte mit HTML-Tags, der JAVA-interne Browser ist jedoch viel zu langsam, um längere Texte schnell zu rendern. Deshalb wurde auch ein neuer

Text Browser entwickelt, der beliebig viele HTML-Dokumente gleichzeitig darstellen kann und interaktive Textmarkierungen erlaubt (Abb. 6c). Eine weitere Visualisierung, die an Tortendiagramme erinnert, jedoch deutlich mehr Funktionalitäten zur Verfügung stellt, ist der so genannte *Circle Segment View* (Abb. 6d) [KP04] Alle Datensätze werden dort abhängig von zwei frei wählbaren Dimensionen – ähnlich wie beim Scatterplot – auf dem Kreis abgetragen. Eine Zoom-Funktion wird über Slider angeboten, mit deren Hilfe der angezeigte Datenbereich eingeschränkt werden kann und dichte Punktwolken auseinander gezogen werden können.

Da INVISIP und VISMEB in der Dissertation von Mann umfassend dokumentiert ist [MT01], wurden die Visualisierungen von INVISIP/VISMEB hier nur kurz umrissen. Interessant für die Weiterentwicklung sind vor allem die *Level Table*, der *Scatterplot* und der *Text Browser*, die alle in modifizierter Version oder neu implementiert wieder in MEDIOVIS auftauchen.

3.3 VisMeB

Das Ziel von VISMEB war, INVISIP zukünftig für beliebige Datenbanken und Anwendungskontexte einsetzen zu können. Da INVISIP speziell für den Abruf und die Visualisierung von Geo-Daten entwickelt wurde, mussten einige Programmteile systematisch überarbeitet und neu konzipiert werden, um generische Aufgaben erfüllen zu können.

Die Hauptforderung der schon Anfang der 80er Jahre entwickelte *Model-View-Controller* Architektur [GA83] ist die strikte Trennung der Daten im Programm von ihrer visuellen Repräsentation. Das Datenmodell verwaltet die Daten des Programms autonom in möglichst effizienten Datenstrukturen. Die Views (Visualisierungen) erreichen die benötigten Daten über klar definierte Schnittstellen des Datenmodells. Der Hauptvorteil einer solchen Architektur ist, dass Veränderungen im Programm nur an bestimmten Stellen durchgeführt werden müssen. Das Datenmodell und die Views können unabhängig voneinander entwickelt werden, und ein Programmierer muss nicht das ganze Projekt kennen, um sich zurechtzufinden. Diese Trennung zwischen Datenmodell und Visualisierungen war anfangs auch im INVISIP-Projekt vorgesehen, wurde jedoch bald nicht mehr konsequent durchgehalten, was zur Folge hatte, dass kleine Änderungen am Projekt an zahlreichen Stellen des Programms durchgeführt werden mussten. Dies war zeitlich sehr aufwändig und führte immer wieder zu Konflikten zwischen parallel entwickelten Programmteilen, was auch damit zusammen hing, dass zahlreiche Programmierer

im Rahmen von studentischen Projekten an der Entwicklung von INVISIP beteiligt waren und es keine einheitlichen Dokumentationen und Richtlinien zum Programm-Code gab.

Im Rahmen eines umfangreichen Redesigns wurden viele Programm-Routinen schließlich neu strukturiert, vereinfacht und optimiert, wodurch eine generische Architektur entstand, die die effiziente Weiterentwicklung des Projekts ermöglichte. INVISIP und VISMEB wurden simultan weiterentwickelt, daher stellen beide Systeme dieselben Anfragemöglichkeiten, Visualisierungen und Interaktionstechniken bereit. Die größten Veränderungen betrafen Modifikationen im Datenmodell und waren somit für den Nutzer nicht sichtbar. Es wurden neue Datenbank-Klassen geschrieben, die Daten aus unterschiedlichen Datenbanken einlesen konnten. Derzeit existieren Schnittstellen für Oracle-, PostgreSQL- und Offline-Datenbanken, durch die Ausnutzung der Möglichkeiten, die die Objektorientierung bietet, können weitere Datenbank-Schnittstellen mit minimalem Aufwand ergänzt werden. In Anlehnung an Tanin und Shneiderman [TS02] wurde ein so genannter *Assignment Editor* entwickelt – auch als *Visual Configurator* bezeichnet – mit dem dynamisch festgelegt werden konnte, welche Metadaten in welchen Visualisierungen dargestellt werden sollen. Dies war ein weiterer wichtiger Schritt, um Daten aus unterschiedlichen Datenbanken überhaupt visualisieren zu können. Während eine Datenbank mit Internet-Seiten beispielsweise Metadaten wie den Titel, die Größe oder die Sprache eines Dokuments bereithält, enthalten typische geo-räumliche Metadaten wie Koordinatenangaben, Städte oder Regionen. Beim Aufruf einer anderen Datenbank muss also zugleich auch festgelegt sein, wo die eingelesenen Daten in den Visualisierungen zu sehen sein sollen, und durch den Einsatz eines Assignment Editors kann der Benutzer selbst die zu visualisierenden Metadaten festlegen.

Kapitel 4: Planungsphase

Für die VISMEB-Architektur wurde nach Abschluss des INVISIP-Projekts Ende 2003 nach neuen Einsatzgebieten gesucht. Der Katalog der Bibliothek Konstanz kristallisierte sich aufgrund den großen und intellektuell gepflegten Datenmengen und der intensiven Nutzung durch Bibliotheksbesucher als interessantes Anwendungsfeld an.

Es erschien von Anfang an sinnvoll, die Entwicklung von MEDIOVIS aus zwei Perspektiven zu betrachten: zum einen sollte mit dem System eine innovative und intuitive Alternative zum existierenden Suchkatalog geschaffen werden, zum anderen sollten bibliotheksspezifische Arbeitsfelder analysiert und Visualisierungen entwickelt werden, die die Erfassung und Pflege der Bibliotheksdaten erleichtern. Es wurden also zwei unterschiedliche Benutzergruppen betrachtet: die Besucher und die Mitarbeiter der Bibliothek. Da in den ersten Meetings mit den Bibliotheksmitarbeitern jedoch noch kein konkreter Bedarf nach neuen Werkzeugen zu ermitteln war, wurde der Schwerpunkt vorerst auf die Entwicklung der Suchoberfläche gelegt.

Da INVISIP und VISMEB auch als Applet über den Web-Browser aufgerufen werden konnten, bot es sich an, MEDIOVIS ebenfalls als Applikation und Applet zu entwickeln, so dass das System auch außerhalb der Bibliothek aufgerufen werden konnte.

4.1 Ist-Stand KOALA / LIBERO WEBOPAC

Der erste Schritt bei der Umgestaltung der VISMEB-Oberfläche bestand in der Analyse des vorhandenen Suchsystems der Universitätsbibliothek Konstanz. Wie in Kapitel 2.1 beschrieben, wird das lokale KOALA-System seit 2000 online über die Homepage angeboten. Neben der Suche nach Bibliothekstiteln stehen auch Servicefunktionen zur Vormerkung von Medien zur Verfügung. Die Einstiegsseite des Such-Katalogs (s. Abb. 7) bietet ein gewohntes Web-Suchformular, in welchem nach Personen, Titelstichworten, Institutionen und Signaturen gesucht werden kann. Die Suche kann außerdem nach Jahren, Fachgebieten und so genannten Sonderstandorten eingeschränkt werden. Das Kürzel »med« bezeichnet hierbei den für diese Arbeit relevanten Sonderstandort Mediothek. Als Suchmodi werden die exakte und trunkierte Suche und eine logische UND-/ODER-Verknüpfung angeboten. Die erweiterte Suche, die ebenfalls auf der Homepage verfügbar ist, ermöglicht die Suche nach mehreren Personen und die direkte Suche über den Index aller Personen, Titel und Institutionen. Es gibt unseres Wissens keine

statistischen Auswertungen, die Aussagen über die Nutzung der angebotenen Suchoptionen machen, die folgenden Ausführungen sollen jedoch einige Probleme aufzeigen, die mit den angebotenen Suchoptionen auftreten können.

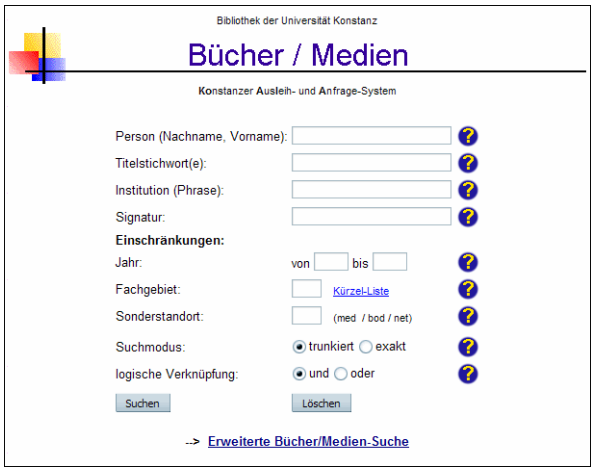


Abb. 7: Web-Suchformular des KOALA-Katalogs der Universitätsbibliothek Konstanz

Die erste Kritik, die auch in Kapitel 2.1 schon angeschnitten wurde, richtet sich gegen die starre Kategorisierung der Suchfelder. Kategorienübergreifende Suchen sind weder in der einfachen noch in der erweiterten Suche möglich. Die Suche ist zudem auf die angebotenen Suchfelder begrenzt. Der Versuch, alle Bibliothekstitel zu finden, die beispielsweise einen Bezug zu Regensburg haben, lässt sich lediglich über das Titel-Suchfeld durchführen. Dabei werden jedoch alle Bibliothekstitel ignoriert, die in Regensburg erschienen sind oder deren Beschreibung »Regensburg« enthält. Wissenswertes über den Autor Kafka muss ebenfalls über mehrere Suchen (Person, Titel, Institution) ermittelt werden. Daraus ergab sich die Forderung an die MEDIOVIS-Oberfläche, analog zu Web-Suchmaschinen ein einziges Suchfeld anzubieten, das die Suche in den wichtigsten Kategorien gleichzeitig ermöglicht. Zusätzlich sollte eine erweiterte Suche integriert werden, um Suchanfragen genauer spezifizieren zu können. Die Suchfelder des KOALA-Systems sollten größtenteils übernommen werden, da sie die wichtigsten Kategorien abdecken und zudem den KOALA-Nutzern schon vertraut sind.

Die nächsten Überlegungen betrafen die Begrifflichkeit der angebotenen Suchoptionen. Unter »exakter Suche« versteht man die 1:1-Übereinstimmung von Suchbegriffen, während eine »trunkierte Suche« auch unvollständige Worte erlaubt (die Suche nach »Psycho« findet auch Einträge wie »Psychologie« oder »Psychothriller«). Die benutzten Begriffe erscheinen uns zu

technisch, weshalb sie durch ausführlichere Beschreibungen ersetzt wurden. Die exakte und trunkierte Suche sollte umbenannt werden in »Suche ganze Wörter« und »Suche Wortteile«.

Im KOALA-System werden Wörter zudem nur am Ende der Suchbegriffe trunkiert (die Suche nach »begriff« findet »Begrifflichkeit«, jedoch nicht »Suchbegriff«).[14] In MEDIOVIS konnte dieses Problem aufgrund der zweistufigen Suchstrategie, die in Kapitel 2.4 schon erwähnt wurde und in Kapitel 6.2.1 genauer vorgestellt wird, umgangen werden.

Die logische UND-Verknüpfung verlangt, dass alle eingegebenen Begriffe in einem Datensatz vorkommen müssen, bevor er als Treffer zurückgegeben wird, während eine ODER-Verknüpfung mindestens ein Wort der Suchanfrage beinhalten muss. Die logische Verknüpfung entstammt der booleschen Algebra[15] und führt bei uneingeweihten Benutzern gerne zu Widersprüchen, da sie sich von der umgangssprachlichen Verwendung der und-/oder-Konjunktionen unterscheidet. Wenn der Benutzer beispielsweise UND-verknüpft nach den Büchern »Der Idiot« und »Der Spieler« sucht, erhält er keine Treffer, da das Retrieval-System nach Bibliothekstiteln sucht, in denen beide Begriffe gleichzeitig vorkommen. Eine Umbenennung erschien sinnvoll, und die Entscheidung fiel auf die Bezeichnungen »Finde alle Suchbegriffe« und »Finde mindestens einen Suchbegriff«.

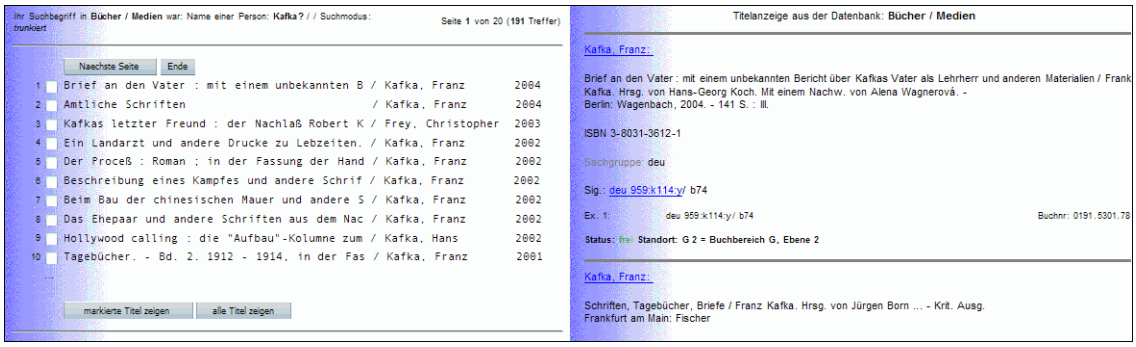


Abb. 8: Listenanzeige der ersten Treffer und Volltextanzeige des KOALA-Katalogs

Als nächstes wurde die Treffer-Ausgabe von KOALA analysiert (s. Abb. 8). Es werden die Titel, Autoren und Erscheinungsjahre der ersten zehn gefundenen Datensätze aufgelistet, die nach

[14] Dies lässt sich auf technische Beschränkungen zurückführen, da alle vorkommenden Wörter in Datenbanken in alphabetischen Indizes verwaltet werden, um die Suchgeschwindigkeit mit Hilfe spezieller Algorithmen wie binären Suchbäumen zu optimieren.

[15] Benannt nach George Boole (1815-1864)

der nächsten Interaktion im Volltext angezeigt werden. Alternativ können einzelne Bibliothekstitel über Checkboxen ausgewählt und die Volltext-Anzeige auf die Auswahl beschränkt werden.

Die Beschränkung auf 10 Treffer stellt eine deutliche Einschränkung dar, da es abgesehen von der Jahreszahl kein Ranking gibt, nach welchen die Treffer angeordnet werden. Eine weitere Kritik richtet sich gegen die Zeit, die für eine einzelne Suche aufgebracht werden muss. Von der Suche bis zur Volltextanzeige müssen jedes Mal drei Internet-Seiten aufgerufen werden. Dieses Prozess sollte in MEDIOVIS optimiert werden. Das Suchformular, die Listen- und Volltextanzeige sollten komplett auf einem Bildschirm untergebracht werden, dass der Suchprozess immer auf den ersten Blick ersichtlich bleibt.

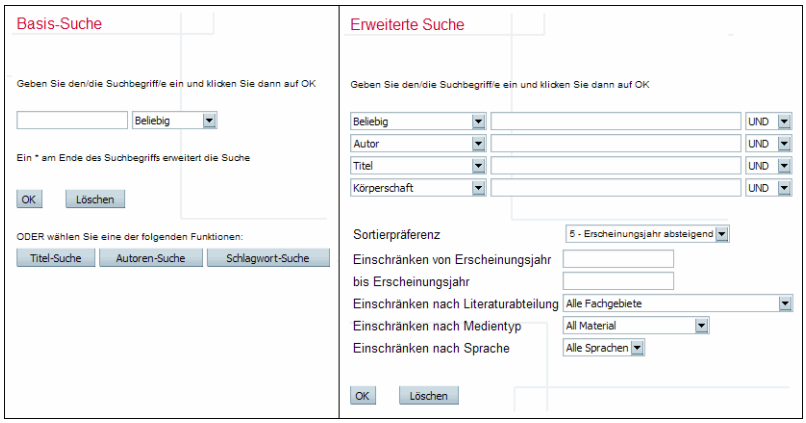


Abb. 9: Suchformular des LIBERO WEBOPAC-Systems

Das neue Suchsystem LIBERO WEBOPAC bietet die kategorienübergreifende Suche erfreulicherweise schon in der einfachen Suche an (s. Abb. 9). Die erweiterte Suche ähnelt dem Aufbau des alten KOALA-Systems, die UND-/ODER-Verknüpfung hat jedoch einen noch formaleren Charakter als im alten System.[16] Positiv ist die Möglichkeit, die Treffer nach unterschiedlichen Kriterien sortieren zu können. Nach dem Start einer Suche erscheint eine Listendarstellung der ersten 20 gefundenen Datensätze, die neben dem Autor, Titel und Jahr auch den Medientyp mit einschließt. Die Titel können direkt angeklickt werden, wodurch die Volltextanzeige aufgerufen

[16] Die mehrfache Auswahlmöglichkeit der UND-/ODER-Verknüpfung führt zudem zu einem weiteren logischen Problem, da die Prioritäten der Verknüpfungen nicht genau definiert sind. Dies soll an einem Beispiel veranschaulicht werden: die Suche »Autor=Kafka UND (Titel=Amerika ODER Titel=Schloss)« führt zu anderen Ergebnissen als die Suche »(Autor=Kafka UND Titel=Amerika) ODER Titel=Schloss«

wird. Durch die schnelle LIBERO-Technologie fällt das Aufrufen neuer Internet-Seiten weniger ins Gewicht als bei KOALA, dennoch sind ebenfalls drei Schritte notwendig, um die Volltextanzeige zu erreichen.

4.2 Ist-Stand VISMEB

VISMEB wurde als generischer Metadaten-Browser entwickelt, um eine Vielzahl unterschiedlicher Domänen abzudecken. Die heterogenen Visualisierungen erlauben neben dem bloßen Betrachten der Treffermenge eine tiefer gehende Analyse der präsentierten Daten. MEDIOVIS sollte jedoch zunächst eine einfache, intuitive Suchoberfläche für den gewöhnlichen Bibliotheksnutzer bieten.

Nachdem ein erster Auszug der Mediotheks-Katalogdaten verfügbar war, konnten diese Daten relativ zügig in das VISMEB-Format konvertiert und angezeigt werden. Die Mediothek verfügt über ca. 25.000 Medien, 2000 davon wurden mit VISMEB visualisiert.

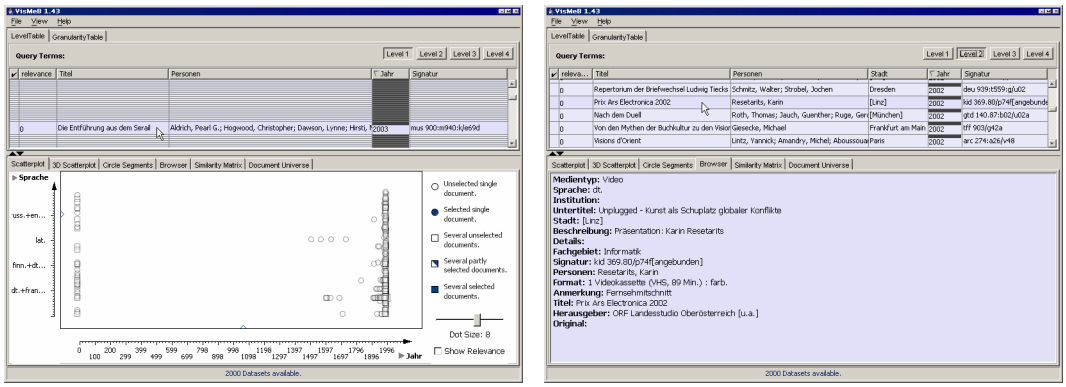


Abb. 10: Anzeige von 2000 Mediotheks-Datensätzen mit VISMEB

In Abb. 10 sind zwei Screenshots von VISMEB zu sehen, in denen 2000 Mediotheks-Datensätze wiedergegeben werden. Auf der linken Seite sieht man oben die tabellarische Ansicht mit allen angezeigten Datensätzen. Die Zeile unter dem Cursor wird vergrößert dargestellt; diese Technik wird als *Table Lens* bezeichnet [RC94]. Im unteren Bildschirmbereich wird der *Scatterplot* angezeigt. Der rechte Screenshot präsentiert erneut die *Level Table* im zweiten Level, in welchem alle Tabellenzeilen komplett lesbar sind. Unter der Tabelle zeigt der *Browser View* den aktuellen Datensatz im Volltext an.

Beim Explorieren in den Datenbeständen wurde bald offensichtlich, dass VISMEB nicht allen Anforderungen gerecht werden konnte, die von einem Bibliotheksbrowser zu erwarten waren:

- die Visualisierungen von VISMEB waren zu generisch konstruiert, um konkrete Daten in allen Ausprägungen ansprechend darzustellen: der in Abb. 10 dargestellte *Scatterplot* ordnet beispielsweise die Datensätze nach ihren Jahresangaben auf der X-Achse an. Da zahlreiche Titel keine Jahresangaben haben, verläuft die Jahres-Skala von 0 bis 2004, und der Großteil des verfügbaren Platzes wird nicht genutzt. Die tabellarische Ansicht im ersten Level hat Übersichtscharakter, interessant für den Nutzer wird hier aber eher der zweite Level, der sofort textuellen Einblick in die Daten gibt. Die Anzeige der Datensätze im *Browser View* ist zwar vollständig, aber nicht sehr übersichtlich und attraktiv. Deshalb galt es, einige Visualisierungen grundlegend neu zu gestalten und Mehrwerte einzubringen, die sich an den verfügbaren Daten orientieren. Außerdem sollten auch neue Visualisierungen diskutiert werden, die im Bibliotheks-Kontext sinnvoll erscheinen.
- die Funktionsvielfalt von VISMEB würde den gewöhnlichen Katalognutzer überfordern. Die zwei angebotenen Tabellen mit insgesamt 10 unterschiedlichen Detailstufen und sechs weitere Visualisierungen stellen einen großen Overhead dar, den der Nutzer erst begreifen muss. Des Weiteren sind nicht alle Visualisierungen von VISMEB selbsterklärend.

VISMEB war das Ergebnis wissenschaftlicher Projekte, in denen die Zielgruppen vorwiegend Experten und keine Gelegenheitsnutzer waren. Da MEDIOVIS grundsätzlich später auch Bibliotheksmitarbeitern bei der Analyse der Datenbestände Mehrwerte bieten sollte, wurde die Möglichkeit nicht ausgeschlossen, komplexere Visualisierungen wieder ins System zu integrieren oder nur einem kleineren Benutzerkreis zur Verfügung zu stellen. Der prioritäre Anspruch bei der Konzeption war jedoch, ein intuitives, selbsterklärendes System zu schaffen, das im Idealfall für alle Nutzer verständlich ist.

4.3 Ist-Stand Bibliotheksdaten

Die bisherigen Überlegungen bezogen sich vorwiegend auf die Recherche und Visualisierung der bibliothekarischen Daten. Die Daten selbst wurden von der Bibliothek im MAB2-Format geliefert, ein standardisiertes Format für Bibliotheksdaten. Das MAB2-Format ermöglicht die Erfassung beliebiger Bibliotheks-Medien, unterscheidet jedoch nicht grundlegend zwischen verschiedenen Medientypen. Bis zu 25 Personen können beispielsweise pro Datensatz gespeichert werden, die Funktion der Person kann jedoch nicht näher bestimmt werden. Eine erfasste Person kann somit Autor, Regisseur oder Komponist sein, dies ist jedoch nicht aus den Daten ablesbar. Es gibt weitere Felder im MAB2-Format, die Volltextbeschreibungen zu einem Medium erlauben. In diesen Volltexten findet man gewöhnlich Aufzählungen der Mitwirkenden

und deren Funktion. Da das Format dieser Texte nicht standardisiert ist, können die Daten wiederum nicht automatisch geparst werden. Somit können in MEDIOVIS leider keine spezifischen Suchen nach Regisseuren oder Darstellern eines Films angeboten werden.

Die vorhandenen Daten sollten durch weitere aussagekräftige Inhalte angereichert werden. Die zahlreichen Internet-Shops und -Datenbanken haben große, neue Datenbestände eröffnet. Covers und Textausschnitte von Büchern lassen sich problemlos online betrachten, bevor man das Buch in die Hand bekommt, und man findet zahlreiche Rezensionen zu CDs oder Poster und Trailer zu Filmen.

Die Titelerfassung ist einerseits eine traditionelle Aufgabe der Bibliotheken, und der Arbeitsaufwand zur Datenpflege und Datenanreicherung kann bei der großen Anzahl an Bibliothekstiteln große Dimensionen annehmen. Da die Bibliotheksmitarbeiter in ersten Gesprächen deutliches Interesse an der Idee zeigten, die vorhandenen Daten durch Fremddaten anzureichern, wurden im Rahmen des Projekts Vorgehensweisen gesucht und gefunden, wie die Bibliotheksdaten halbautomatisch angereichert werden können und der dafür notwendige Zeitaufwand gleichzeitig minimal gehalten werden kann. Besonderes Interesse galt dabei den Filmbeständen auf Video und DVD, die ungefähr ein Drittel des Gesamtbestandes der Mediothek ausmachen.

4.4 Usability Engineering Lifecycle

Die Überlegungen, die die Konzeption von MEDIOVIS begleitet haben, sollen anhand des verbreiteten »Usability Engineering Lifecycle« von Mayhew veranschaulicht werden [MD99]. Dieser Lifecycle lehnt sich an das Software-Entwicklungsmodell OOSE an und gliedert den benutzerzentrierten Entwicklungsprozess von Software grob in die drei Phasen »Requirements Analysis«, »Design / Testing / Development« und »Installation«, von denen die Einzelschritte der ersten Phase detaillierter ausgeführt werden.

4.4.1 Requirements Analysis

4.4.1.1 User Profiles

Benutzerprofile werden entworfen, um festzustellen, für welche Zielgruppe ein neues System entwickelt wird, welche Kenntnisse die Benutzer mitbringen und welche Motivation sie haben, das System zu nutzen.

Bei der Erhebung der Suchkatalog-Nutzer war eine 2002 erschienene statistische Evaluierung des WWW-Angebots der Bibliothek hilfreich [EF02] Den größten Prozentsatz der Nutzer stellen dort Studenten. Ein nicht unerheblicher Teil der Benutzer (ca. 20%) besteht aus weiteren Katalognutzern wie Angestellten der Bibliothek, Doktoranden, Professoren und externen Besuchern, und die Bibliotheksmitarbeiter machen ca. 10% der Nutzer aus. Diese Zahl ist real gesehen noch niedriger, da in der Evaluierung die Besuche der Bibliotheks-Homepage ausgewertet wurden und diese Seite auf vielen Bibliotheks-Rechnern als Startseite eingetragen ist.

Die Studenten verfügen zumeist mindestens über Grundkenntnisse mit dem Umgang von Computern und werden MEDIOVIS eher gelegentlich bei Bibliotheksbesuchen benutzen. Da MEDIOVIS vorerst als optionales Suchsystem angeboten werden sollte, ging man bei den Nutzern, die MedioVis aufrufen würden, von einem grundsätzlichen Interesse aus, sich mit neuen, unbekannten Systemen auseinanderzusetzen. Die Computer-Kenntnisse der zweiten Nutzergruppe sind je nach Fachbereich oder beruflicher/sonstiger Ausrichtung recht unterschiedlich und schwer zu kategorisieren. Leichter fiel die Auswertung der Bibliotheksmitarbeiter, die gewöhnlich über gute bis sehr gute Kenntnisse ihres Bibliotheks-Katalogs und KOALA als Recherche-Werkzeug verfügen.

Aufgrund des breit gefächerten Benutzerkreises war ein Ergebnis der Benutzeranalyse, dass das zukünftige System eine möglichst universelle Oberfläche bieten sollte, um für alle Nutzer leicht bedienbar zu sein. Weitere, komplexere Visualisierungen wie der angedachte *Scatterplot* sollten dennoch ins System Eingang finden, um versiertere Nutzer nicht unnötig einzuschränken.

4.4.1.2 Contextual Task Analysis

In der kontextuellen Aufgabenanalyse werden Modelle erstellt, wie die Aufgaben, die ein neues System bieten soll, momentan von den Benutzern gemeistert werden. Anhand dieser Modelle können funktionale und qualitative Anforderungen an das geplante System gestellt werden. Da die meisten Studenten der Universität schon nach kurzer Zeit Erfahrungen mit KOALA und dem Bibliotheks-Dienstangebot sammeln, konnten verhältnismäßig einfach Kritikpunkte am momentanen System in Form von Kurz-Interviews gesammelt werden. Die wichtigsten Probleme

bei der Bibliotheks-Nutzung, die für die Konzeption von MEDIOVIS interessant erschienen, seien nachfolgend kurz aufgelistet:[17]

- Viele Medien, die eigentlich in der Bibliothek verfügbar sein müssten, werden über KOALA nicht gefunden.
- Es ist mühsam, Medien in der Bibliothek zu finden, da KOALA keine Angaben über die Standorte der Medien macht.
- KOALA reagiert zu langsam und ist zeitaufwändig zu bedienen, da die Anfrageergebnisse auf mehr als eine Bildschirmseite verteilt werden.

Im Rahmen der Aufgabenanalyse werden gewöhnlich *Task Scenarios* (Aufgabenszenarien) erstellt, die den Ablauf einer Aufgabenerfüllung dokumentieren. Vier Szenarien solcher Bibliotheksbesuche seien hier exemplarisch wiedergegeben:

- Eine Studentin kommt in die Bibliothek und will sich den Film »Golem« von Paul Wegener ausleihen und zu Hause anschauen, weil sie die Vorführung im Rahmen ihrer Kunst- und Medien-Vorlesung verpasst hat. Sie ruft die einfache Suche des KOALA-Katalogs auf, gibt »Golem« als Suchbegriff ein und erhält die ersten 10 der 42 Treffer chronologisch sortiert zurück. Weil sie sich erinnert, dass der Film 1920 gedreht worden ist, klickt sie sich ein paar Seiten weiter, muss aber feststellen, dass unter dieser Jahreszahl kein Film eingetragen ist. Deshalb schaut sie sich alle Treffer genauer an und entdeckt, dass »Wegener, Paul« als mitwirkende Person aufgezählt ist. Diesen Eintrag wählt sie aus und sieht, dass der Film gerade verfügbar ist. Sie notiert sich die Signatur und sucht den Film im Regal.
- Ein Physik-Student hat gehört, dass die Mediothek zahlreiche, ausleihbare DVDs anbietet. Er gibt in der einfachen KOALA-Suche DVD als Titelstichwort ein, erhält jedoch keine Treffer. Weil er keine Möglichkeit findet, den Medientyp genauer zu spezifizieren, begibt er sich direkt ans Regal, um dort nach DVDs zu suchen.
- Eine Angestellte des Sprachlabors besucht ab und zu die Mediothek, um zu sehen, ob in letzter Zeit neue Italienisch-Sprachkurse angeschafft wurden. Sie gibt in das Fachgebiets-

[17] kritische Feststellungen stehen in dieser Aufzählung im Vordergrund; viele Bibliotheks-Nutzer haben auch recht positive Erfahrungen mit dem Konstanzer Bibliotheks-System gemacht, v. a. dann, wenn sie schon Erfahrungen mit Bibliotheks-Systemen anderer Institutionen gemacht hatten.

Suchfeld das Kürzel namens FSI ein und startet die Suche. Da keine Treffer zurückgeliefert werden, nutzt sie die Hilfe-Funktionen von KOALA und erfährt, dass sie für die Suche nach Sprachkursen das Kürzel FSA benötigt, welches alle Sprachen umfasst. Die Suche scheitert wieder, deshalb ruft sie erneut das Suchformular auf und gibt zusätzlich zum Fachgebiet ihr Interessengebiet »Italienisch« als Titelstichwort ein. Nun erhält sie 212 Treffer, die allerdings nicht komplett sind, wie sie später am Regal feststellt.

- Ein Kunst- und Medien-Student im 4. Semester, der sich öfter in der Mediothek aufhält, schreibt eine Hausarbeit über Luis Buñuel und sucht nach Medien, die sich mit dem Regisseur auseinandersetzen. Er ruft KOALA auf, gibt den Regisseur im Personen-Suchfeld ein und spezifiziert die Mediothek als Sonderstandort für seine Suche. Er erhält 26 Treffer als Antwort, es handelt sich aber bei allen Bibliothekstiteln um Filme, die Buñuel selbst gedreht hat. Er geht auf die Hauptseite zurück und wählt die thematische Suche, gibt Buñuel als Suchbegriff ein und schränkt die Suche auf die Mediothek ein. Da er wieder keine Treffer erhält, weitet er seine Suche wieder auf die ganze Bibliothek aus und erhält nun 67 Treffer, von denen einige zu seiner Überraschung doch in der Mediothek stehen.

Aus Szenarien dieser Art konnten wichtige Anforderungen an das neues System MEDIOVIS entwickelt werden, vor allem bezüglich der Retrieval-Komponenten:

- die angebotenen Suchformulare dürfen nicht zu restriktiv sein, wenn unterschiedlichste Benutzeranfragen bewältigt werden sollen.
- die Anfrageergebnisse müssen unterschiedlich sortiert werden können, weil das Jahr nicht immer ein optimales Treffer-Ranking darstellt.
- MEDIOVIS sollte die direkte Suche nach Medientypen, Sprachen und bestimmten Fachgebieten gestatten.

4.4.1.3 Usability Goal Setting

Usability-Ziele werden aus den Erkenntnissen des Benutzerprofils und der Aufgabenanalyse erstellt. Sie sind maßgebend für den Entwurf eines Software-Projekts und lassen sich in qualitative und quantitative Ziele aufteilen. Während man unter quantitativen Zielen messbare System-Anforderungen versteht wie zum Beispiel »der Nutzer erhält in spätestens zwei Sekunden Antwort auf seine Anfrage.«, werden in dieser Arbeit ausschließlich qualitative Anforderungen gestellt, da sie allgemeinere Aussagen treffen und vom wissenschaftlichen Standpunkt aus interessanter sind. Quantitative Ziele waren dennoch ein wichtiges Kriterium, da die Standard

SWING-Komponenten von JAVA relativ langsam sind; dies betrifft jedoch vorrangig die Implementierung, die in Kapitel 6 näher erläutert wird. Auf Basis der Erkenntnisse, die am aktuellen Ist-Stand der zwei konventionellen Suchsysteme und der VISMEB-Software gewonnen wurden und den Erkenntnissen, die durch Interviews und Userszenarios zustande kamen, ergaben sich folgende Anforderungen für MEDIOVIS:

- Die Oberfläche von MEDIOVIS soll die drei Schritte eines Suchprozesses auf einer Bildschirmseite vereinen: Suche nach Bibliothekstiteln, Anzeige der Titel als Gesamtübersicht und im Volltext.
- Die einfache Suche soll sich auf ein Suchfeld beschränken, das kategorienübergreifende Suchen gestattet (gleichzeitige Suche nach Titel, Personen, Institutionen usw.). Gleichzeitig sollen erweiterte Suchfelder eingeblendet werden können, mit denen genauer spezifizierte Suchen durchgeführt werden können und die Suche nach Jahresangaben, Medientyp, Sprache und Fachgebiet eingeschränkt werden kann, wie es in KOALA nur teilweise oder gar nicht möglich ist.
- Es sollen nur die Visualisierungen angeboten werden, die zum schnellen Auffinden der Medien notwendig sind. Dafür bietet sich die *Level Table* als Visualisierung an, die den meisten Nutzern vertraut erscheint und mehr Interaktion als eine normale, listenbasierte Darstellung bietet. Um Volltexte zu visualisieren, soll eine überarbeitete Version des *Browser View* eingesetzt werden. Als Alternative zur *Level Table* soll weiterhin an zweiter Stelle ein *Scatterplot* im System angeboten werden, um erfahreneren Nutzern weitere Recherchemöglichkeiten nicht vorzuenthalten.
- Die Möglichkeit, im KOALA-System einzelne Datensätze in der Listenansicht auswählen zu können, soll aufgegriffen werden. Es soll eine Listenansicht der ausgewählten Bibliothekstitel angeboten werden, die weitere Funktionen wie das Ausdrucken der Liste, das Abspeichern oder das Versenden per e-Mail ermöglicht.
- Eine Standort-Ansicht soll in das System integriert werden, die sofort anzeigt, wo sich das gerade fokussierte Medium in der Bibliothek befindet.
- Die verfügbaren Bibliotheksdaten sollen durch zusätzliche Daten angereichert werden, um den Nutzern auch inhaltliche Mehrwerte gegenüber dem konventionellen System bieten zu können und den vorwiegend textuellen Datenbestand durch multimediale Inhalte abwechs-

lungsreicher und informativer zu gestalten. Das System soll daher prinzipiell auch Graphiken anzeigen und Videos abspielen können.

- Weitere Visualisierungen mit innovativem Charakter wären wünschenswert, um den Nutzern unabhängig von den konventionellen Interaktionskomponenten neue Möglichkeiten zu bieten, effektiv und effizient nach relevanten Medien zu suchen.
- Die Bibliotheks-Systeme KOALA und LIBERO geben Auskunft, ob die angezeigten Bibliothekstitel gerade verfügbar oder ausgeliehen sind. Zudem lassen sich ausgeliehene Medien online vormerken. Diese Funktionalität verlangt eine direkte Kommunikation mit der ständig aktualisierten Bibliotheks-Datenbank. MEDIOVIS sollte vorerst auf Basis der gespiegelten Offline-Daten der Mediothek entwickelt werden. Eine direkte Abfrage der Bibliotheks-Datenbank wäre jedoch wünschenswert, um MEDIOVIS als Bibliotheks-System mit voller Funktionalität nutzen zu können.

4.4.1.4 Platform Capabilites and Constraints

Unter Berücksichtigung der vorhandenen Computer-Ausstattung muss beachtet werden, welche Software-Lösungen überhaupt realistisch eingesetzt werden können. Neben der Prozessor-Geschwindigkeit der Rechner spielen auch der vorhandene RAM-Speicher, die Bildschirmauflösung, das eingesetzte Betriebssystem, Internet-Anbindung und die Eingabegeräte eine Rolle.

Da MEDIOVIS in JAVA entwickelt werden sollte, durften die eingesetzten Rechner nicht zu veraltet sein. Die meisten Computer in der Mediothek waren zwar zum Zeitpunkt der Erhebung schon einige Jahre im Einsatz, erfüllen jedoch die Voraussetzungen für ein multitasking-fähiges Fenster-System. Sowohl die Bildschirmauflösung (1024 x 768 Pixel), die Prozessor-Geschwindigkeit (300 MHz – 1 GHz) und der RAM-Speicher (64 MB – 256 MB) waren ausreichen, um mit dem zukünftigen MEDIOVIS flüssig arbeiten zu können. Sämtliche Rechner waren zudem mit dem Windows-Betriebssystem ausgestattet, mit dem Internet verbunden und hatten Tastatur und Maus als Standard-Eingabegeräte. Zudem konnte man über alle Rechner die Drucker in der Bibliothek nutzen.

4.5 Prototyp

In den ersten zwei Januar-Wochen wurde ein MEDIOVIS-Prototyp angefertigt, um den Bibliotheksmitarbeitern ein besseres Bild vom geplanten Katalog-Browser verschaffen zu können. Der Prototyp war von der Architektur und Ästhetik noch stark an die VISMEB-Architektur angelehnt und stellte hauptsächlich eine Reduzierung der VISMEB-Funktionalität dar.

Mitte Januar fand ein Brainstorming statt, bei dem das Potenzial von MEDIOVIS für den Einsatz neuer Visualisierungen analysiert werden sollte. Es entstanden damals einige Ideen, die parallel zum normalen Endnutzer-System auf Forschungsbasis verfolgt wurden. Dazu zählen das *Media Grid* und ein *Table Filter*, zwei Visualisierungen, auf die ich in Kap. 5.3.6 und 5.3.7 noch näher eingehen werde.

Neben den innovativen Ansätzen wurden den Usability Goals unterschiedliche Prioritäten verliehen und ein Zeitplan erstellt. MEDIOVIS sollte auf Basis kontinuierlicher Evaluationen bis Ende Mai implementiert und anschließend in der Mediothek eingerichtet werden. Die Entwicklung von MEDIOVIS verschob sich schließlich in den Juli hinein, da weitere Konzepte in das System integriert wurden, beispielsweise auch die DROID-Technik, die eine automatische Auswertung der Benutzer-Interaktionen auf Datenbank-Basis ermöglicht und in Kapitel 7.2 detaillierter beschrieben wird.

Kapitel 5: MEDIOVIS

In diesem Kapitel werden die einzelnen Features und Komponenten von MEDIOVIS vorgestellt. Der Aufbau des Kapitels orientiert sich an der Abfolge, wie der Benutzer die unterschiedlichen Features und Visualisierungen zu Gesicht bekommt. Neben den in der Mediothek eingesetzten Visualisierungen werden die innovativen Erweiterungen des Systems vorgestellt, die sich noch im Forschungsstadium befinden.

5.1 Programmstart

Nach dem Start von MEDIOVIS wird ein Startmenü eingeblendet, in dem sich der Nutzer für die Programmsprachen deutsch oder englisch entscheiden kann. Da die Universität Konstanz einen ausgesprochen hohen Anteil an ausländischen Studenten besitzt, erschien ein mehrsprachiges System als sinnvoll, und die ersten Evaluationen über DROID (s. auch Kapitel 7.2) haben bestätigt, dass sich immerhin 10% der Benutzer für die englische Programmversion entscheiden.

Abb. 11: MEDIOVIS-Startmenü

5.2 Suchanfragen

Nach der Sprachauswahl wird das Hauptfenster geöffnet und das System initialisiert. Das Hauptfenster ist bewusst sehr einfach gehalten (Abb. 12). Der Hauptbereich des Fensters dient der Begrüßung und aktuellen Informationen der Mediothek, und im oberen Bildschirmbereich befindet sich ein Textfeld, in das der Benutzer seine Suchbegriffe eingeben kann. In der anschließenden Datenbankanfrage werden alle relevanten Datenfelder auf die Suchbegriffe hin überprüft. Unter die relevanten Felder fallen Titel, Originaltitel, Untertitel, mitwirkende Personen, Erscheinungsjahr, Titelbeschreibung, Anmerkungen, Medientyp, Sprache, Verlag, Stadt, Signatur und weitere Detailangaben. Der Benutzer kann somit auch die Daten durchsuchen, die im KOALA-Katalog nur für die Trefferausgabe gespeichert wurden.

Um die Suche weiter einzuschränken, kann der Benutzer erweiterte Suchemöglichkeiten aufrufen. Der »Erweitert«-Button vergrößert das Suchformular und blendet weitere Textfelder und Comboboxen ein, mit denen die Suche mittels festgelegter Kriterien genauer spezifiziert werden kann. Neben der Eingabe von Personen und Titelstichworten lässt sich die Suche auf Erscheinungsjahre, den Medientyp, die Sprache und das Fachgebiet festlegen.

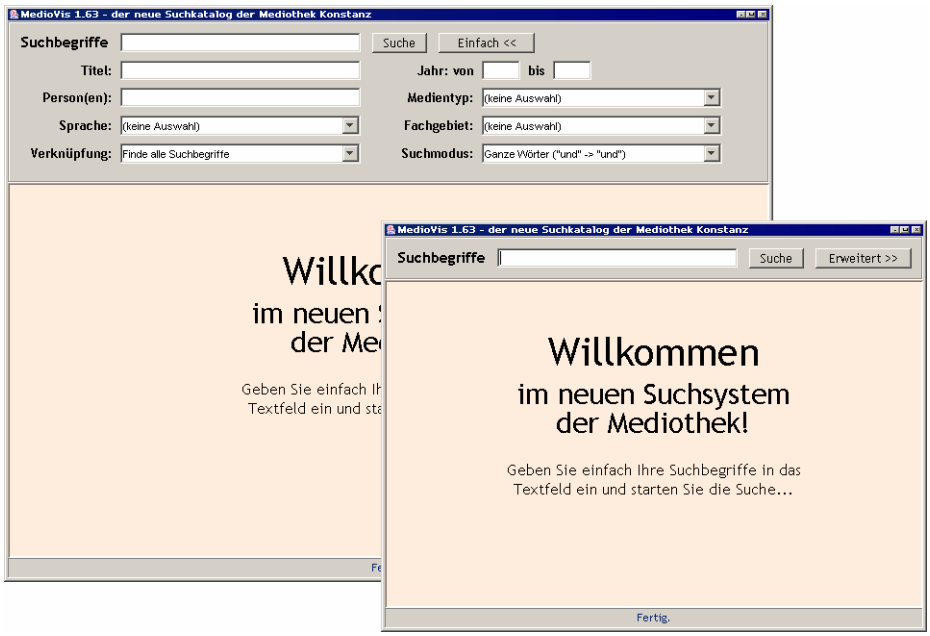


Abb. 12: MEDIOVIS-Suchformulare: Fenster mit einfacher und erweiterter Suche

Die Retrieval-Komponente von MEDIOVIS unterstützt UND-/ODER-Verknüpfungen, die über eine Combobox ausgewählt werden können. In einer UND-verknüpften Suche werden nur die Datensätze zurückgeliefert, in denen alle eingegebenen Suchbegriffe auftauchen, während die ODER-Verknüpfung alle Datensätze akzeptiert, die wenigstens einen der Suchbegriffe beinhalten. Wie in Kapitel 4.1 angedeutet, ist die technische Formulierung der UND-/ODER-Verknüpfung für viele Benutzer nicht nachvollziehbar, daher wurden Umschreibungen für die Verknüpfungsarten gefunden. »Finde alle Suchbegriffe« entspricht der UND-Verknüpfung, »Finde mindestens einen Suchbegriff« der ODER-Verknüpfung.

Suchbegriffe lassen sich über das zweite Combobox-Feld automatisch trunkieren. Das trunkierte Suchverfahren sucht nicht nur nach ganzen Wörtern, sondern überprüft, ob die Suchbegriffe als Zeichenfolgen innerhalb von Wörtern auftreten. Die exakte und trunkierte Suche wird in MEDIOVIS mit den Formulierungen »Ganze Wörter (und -> und)« und »Teilbegriffe (und -> hundert)« umschrieben. Wörter lassen sich zudem auch direkt im Suchfeld über die Eingabe von Sternchen trunkieren. Eine Suche nach »*haus*« liefert somit auch »Hausbesitzer« und »Geisterhaus« zurück.

5.3 Visualisierungen

Nach dem Start der Suche werden die relevanten Datensätze eingelesen und das Hauptfenster aufgebaut. Wie in Abb. 13 zu sehen ist, gliedert es sich in drei Bereiche: die obere Hälfte des Fensters wird genutzt, um alle gefundenen Datensätze in tabellarischer Form darzustellen. Neben der tabellarischen Ansicht lässt sich auch die so genannte »Graphische Ansicht« aufrufen,

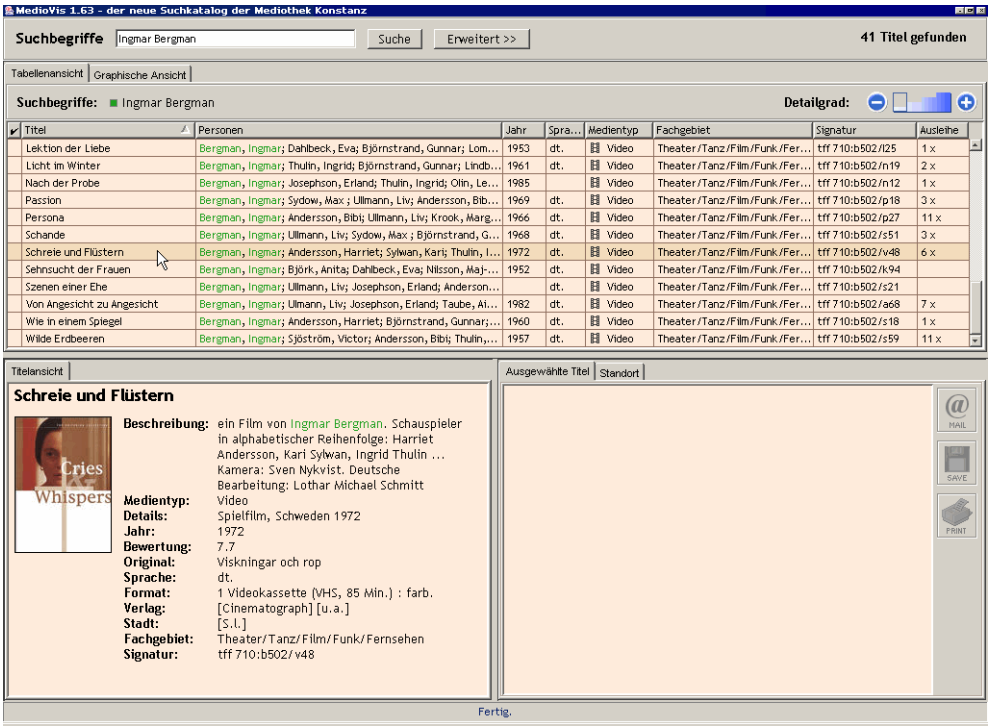


Abb. 13: MEDIOVIS-Hauptfenster mit Standard-Visualisierungen;
Auflösung 1024 x 768 Pixel

die die Daten in einem Scatterplot visualisiert. Der untere, linke Bereich zeigt den in der Tabelle aktuell fokussierten Datensatz im Volltext an. Rechts unten befindet sich eine Auflistung der Datensätze, die per Mausklick in der Tabelle selektiert werden. Als Alternative zur Darstellung der ausgewählten Datensätze lässt sich auch die Standort-Visualisierung anzeigen, die Auskunft über den Standort eines Bibliothekstitels gibt. Alle Visualisierungen sind nach dem *Multiple Coordinated View*-Konzept (MCV) miteinander verknüpft, welches in Kapitel 2.6.2 vorgestellt wurde.

Die folgenden Unterkapitel gehen detailliert auf die einzelnen Visualisierungen ein. In Kapitel 5.3.6 und 5.3.7 werden zwei weitere mächtige, voll funktionsfähige Visualisierungen vorgestellt, die ebenfalls in das System eingebettet wurden, sich jedoch noch in der Forschungsphase befinden.

5.3.1 Tabellenansicht

Die Tabellenansicht besteht in der Standard-Ansicht aus neun Spalten, die einen schnellen Zugriff auf die wichtigsten Datenfelder eines Bibliothekstitels ermöglichen. Die Medientypen werden sowohl textuell (Video, Buch, Tonträger etc.) als auch visuell durch kleine Symbole repräsentiert. Der Tabelleneintrag unter dem Maus-Cursor wird *fokussiert*[18], sichtbar durch die farbige Hervorhebung des Eintrags. Der aktuell fokussierte Eintrag wird in der darunter liegenden Titelansicht im Volltext dargestellt.

Die Spalten der Tabelle können durch Klicks auf die jeweiligen Tabellenköpfe auf- und absteigend sortiert werden. Die Tabelle merkt sich die letzten Sortierungen und sortiert rekursiv. Dies heißt im Klartext, dass Zeilen mit demselben Inhalt nach der zuvor durchgeführten Sortierung nachsortiert werden. Die Sortierung nach Jahr und Medientyp hat beispielsweise zur Folge, dass alle Bücher, Videos usw. zusätzlich nach ihrer Jahresangabe sortiert präsentiert werden. Durch die Sortierung können Hunderte von Datensätzen in die Tabelle gepackt werden, ohne dass der Benutzer die Übersicht über die Daten verliert. Dies ist bei einer listenbasierten Darstellung nicht möglich.

Die Tabelle verarbeitet auch Tastatureingaben. So kann man in der Tabelle mit den Cursor-Tasten navigieren und Datensätze mit der Leertaste auswählen. Analog zum Windows-Explorer und anderen listenorientierten Visualisierungen kann man Wörter über die Tastatur eingeben, und der Tabellenfokus springt direkt an die Position, an der der eingegebene Titel gefunden wird.

Über der Tabelle werden die eingegebenen Suchbegriffe erneut aufgelistet. Trunkierte Suchbegriffe werden mit Sternchen flankiert. Die Suchbegriffe werden zusätzlich farbig in der Tabelle hervorgehoben, wodurch man auf den ersten Blick erkennen kann, ob die Wörter im Titel, in der Personenliste oder in einem anderen Feld auftreten. Im Beispiel in Abb. 13 wurde beispielsweise der Regisseur Ingmar Bergman gesucht, und bei drei der angezeigten Treffer handelt es sich offensichtlich um Werke, die nicht vom Regisseur selbst geschaffen wurden, sondern um Biografien und Dokumentarfilme.

---

[18] Als *fokussierte* Einträge werden die Datensätze in MEDIOVIS bezeichnet, die gerade im Interesse des Benutzers stehen. Sie werden in den Visualisierungen besonders hervorgehoben.

In der rechten oberen Ecke der Tabelle lässt sich der Detailgrad der Tabelle modifizieren. Dieses Feature ist ein Pendant zum Level-Prinzip der *Level Table* aus VISMEB. Standardmäßig wird der einfachste Detailgrad angezeigt. Der Wechsel auf einen höheren Detailgrad vergrößert die Zeilenhöhen der Tabelle und fügt weitere Spalten hinzu. So lassen sich die angezeigten Datensätze nach zusätzlichen Kriterien sortieren.

5.3.2 Titelansicht

Die Titelansicht zeigt ausführliche Informationen zum aktuell fokussierten Bibliothekstitel an. Alle für den Nutzer interessanten Angaben werden in dieser Ansicht aufgeführt. Die präsentierten Informationen werden aus ca. 20 verschiedenen Datenfeldern ausgewählt; leere Datenfelder werden ignoriert. Dies macht Sinn, da die Datensätze der Bibliothek recht unterschiedlich gefüllt sind, wie in Abb. 14 zu sehen ist. Während Bücher über eine ISBN verfügen, besitzen Videos zusätzliche Angaben wie den Originaltitel oder die Sprache. Andere Datensätze wie der rechts dargestellte Tonträger sind nur sehr spärlich mit Informationen gefüllt.

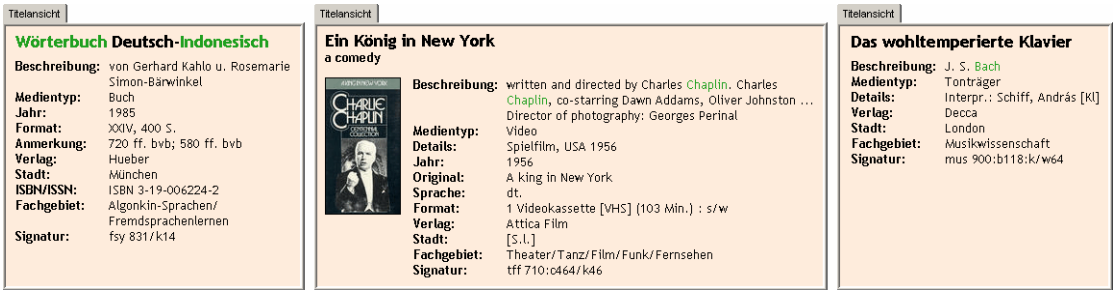


Abb. 14: Drei Titelansichten unterschiedlicher Suchanfragen:
»Wörterbuch Indonesisch«, »Charlie Chaplin« und »Bach«

Die Bibliotheksdaten wurden durch ca. 2000 Film-Plakate angereichert. Wenn ein Plakat zu einem Film existiert, wird es unter dem Titel platziert und die restlichen Angaben entsprechend eingerückt. Es wurde ebenfalls ein Feature implementiert, um Film-Trailer über MEDIOVIS per Mausklick auf einen Link starten zu können, aufgrund der großen Datenmengen, die Filmdaten beanspruchen, wegen lizenzrechtlichen Fragen und aufgrund der beschränkten Multimedia-Fähigkeiten, die die Mediotheks-Rechner zur Verfügung stellen, wurde das Abspielen von Filmen nur im Rahmen der Forschungsversion weiter verfolgt und spielt in der *Media Grid*-Visualisierung wieder eine verstärkte Rolle (Kapitel 5.3.7).

5.3.3 Ausgewählte Titel

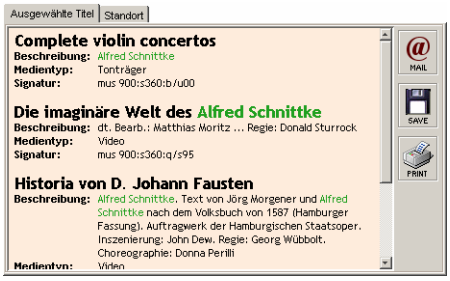


Abb. 15: Darstellung ausgewählter Titel

Wenn Einträge in der Tabelle per Mausklick *selektiert* werden, werden sie der Auswahlansicht, die sich im rechten unteren Bildschirmbereich befindet, hinzugefügt. Die Visualisierung beschränkt sich auf die wichtigsten Angaben der Datensätze wie den Titel, die Titelbeschreibung, den Medientyp und die Signatur. Die selektierten Datensätze können über die rechts angeordneten Button per e-Mail versandt, als Text-Datei abgespeichert oder ausgedruckt werden. Aufgrund der Sicherheitsbeschränkungen von JAVA erlaubt die Applet-Version von MEDIOVIS lediglich den Ausdruck der Bibliothekstitel.

Wenn die Bibliothekstitel in der Auswahl-Box mit der Maus überfahren werden, werden sie wiederum in den anderen Visualisierungen fokussiert und können somit schneller aufgefunden werden.

5.3.4 Graphische Ansicht

Eine sehr mächtige Visualisierung wurde bisher nur am Rande erwähnt. Es handelt sich um den *Scatterplot*, der in MEDIOVIS vereinfacht als graphische Ansicht bezeichnet wird und alternativ zur Tabellenansicht angezeigt werden kann. Die besondere Herausforderung beim Scatterplot war, die grundsätzlich eher technische Visualisierung so ansprechend ins System zu integrieren, dass die Benutzer gerne von sich aus damit arbeiten und selbst die Mehrwerte erkennen, die ihnen die visuell orientierte Exploration bieten kann.

Ein Scatterplot ist ein Punktdiagramm, das alle Datensätze graphisch in einem Koordinatensystem darstellt. Der MEDIOVIS-Scatterplot ist in Abb. 16 abgebildet. Die graphische Darstellung der Datensätze wurde über Icons realisiert, die den Medientyp des jeweiligen Datensatzes visualisieren. Den Icons wurden zur besseren Unterscheidung unterschiedliche Farben verliehen. Die X- und Y-Achsen sind mit Kategorien belegt, und die Position der Icons wird durch die festgelegten Achsenwerte bestimmt. Beim ersten Aufruf der Visualisierung werden die Datensätze, wie in der Abbildung zu sehen, auf der X-Achse nach ihrem Erscheinungsjahr angeordnet, während die Y-Achse für die Anzahl der Ausleihen genutzt wird. Ein Datensatz, der oben rechts zu finden ist, ist demnach erst vor kurzem erschienen und wurde schon oft ausgeliehen.

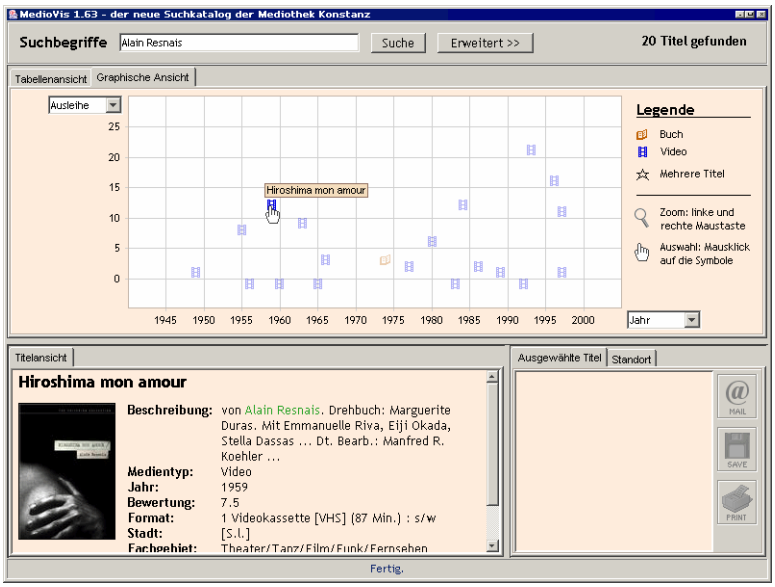


Abb. 16: MEDIOVIS-Hauptfenster mit Scatterplot-Darstellung;
Auflösung 800 x 600 Pixel

Die Legende auf der rechten Seite der Visualisierung veranschaulicht die Bedeutung der verwendeten Icons und informiert über den momentan aktiven Interaktionsmodus. Standardmäßig hat der Cursor das Aussehen einer Lupe, die darauf hinweist, dass man in den Scatterplot hinein- und hinauszoomen kann.

Die Zoom-Funktionalität wurde integriert, um Datensätze, die sehr nah beieinander liegen, durch die Verzerrung besser unterscheiden zu können. Vor der Implementierung wurden verschiedene Ansätze für die Funktionsweise des Zooms diskutiert:

- Eine weit verbreitete Methode, um Darstellungen jeglicher Art zu zoomen, findet man in Graphikprogrammen und einigen Visualisierungswerkzeugen wie auch VISMEB: mit der Maus lässt sich ein Rechteck aufziehen, welches den Bereich definiert, der gezoomt werden soll. Der Vorteil dieser Auswahlmöglichkeit ist zum einen die Exaktheit, mit der ein interessanter Bereich ausgewählt werden kann, und zum anderen die Möglichkeit, die Darstellung in X- oder Y-Richtung unterschiedlich zu vergrößern. Ein hohes, schmales Auswahlrechteck vergrößert die Darstellung beispielsweise fast nur horizontal. Eine Rechteck-Auswahl beeinträchtigt jedoch die Vorstellungskraft, welcher Datenbereich nach dem Zooming tatsächlich zu sehen ist, und sie stellt eine konzeptuelle Sackgasse dar, weil man über die Rechteck-Auswahl immer weiter in die Ansicht hineinzoomt und nicht hinauszoomen kann. Es werden somit auf jeden Fall zusätzliche Interaktionstechniken vonnöten, um ein intuitives und vollständiges Zoomen zu ermöglichen. Sinnvoll wäre – entsprechend dem Detail-Overview

Konzept – die Einblendung eines kleinen Übersichts-Fensters des gesamten Scatterplots, das den gerade gezoomten Bereich hervorhebt, um die räumliche Position innerhalb des Datenraums zu verdeutlichen. Außerdem muss eine Möglichkeit existieren, die Vergrößerungen wieder schrittweise rückgängig zu machen. Bei VISMEB wurde das Hinein- und Herauszoomen über ein Popup-Menü realisiert, welches die Optionen »Zoom In«, »Zoom Out« und »Zoom Out Full« anbietet (Abb. 17 links).

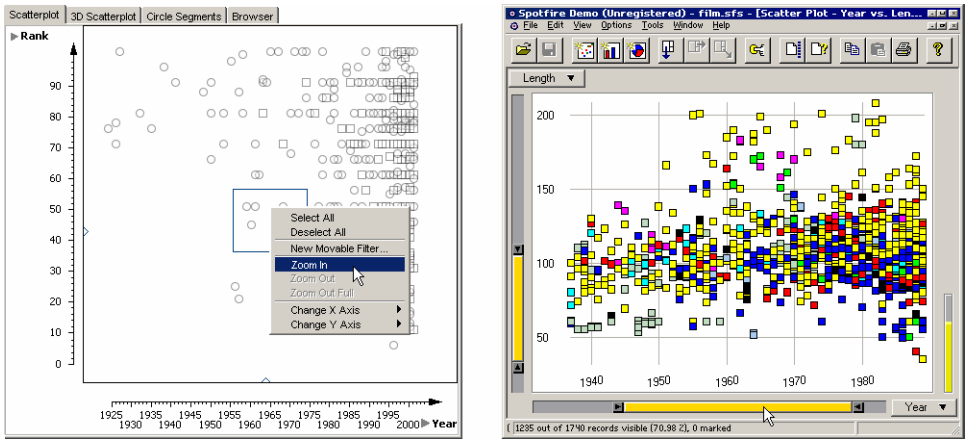


Abb. 17: Zoom-Alternativen in Scatterplots: Rechteck-Auswahl in VISMEB, *Range Sliders* in SPOTFIRE

- Eine elegante Methode, in Scatterplots hineinzuzoomen, kann durch so genannte *Range Slider* realisiert werden, die sich seitlich der Visualisierung befinden. Sie wurden zum ersten Mal in einem Projekt namens HOME FINDER eingesetzt [AWS92] und fanden später u. a. in das kommerzielle Produkt SPOTFIRE Eingang (Abb. 17 rechts). Range Slider lassen sich an beiden Enden modifizieren und innerhalb des Slider-Bereichs beliebig verschieben. Der Einsatz von Range Slidern hat den Vorteil, dass die Ansicht interaktiv gezoomt werden kann, während die Slider-Werte verändert werden. Die Slider visualisieren zudem anschaulich, welcher Bereich des Scatterplots gerade sichtbar ist. Der technische Charakter dieser Interaktionstechnik wurde bei der Konzeption des Scatterplots in MEDIOVIS jedoch als tendenziell unvorteilhaft bewertet.

- In MEDIOVIS wurde eine dritte Alternative gewählt, die aufwändiger zu implementieren war, obwohl (oder gerade weil) sie sich aller zusätzlichen Interaktionskomponenten entledigt. Da der Scatterplot im Gegensatz zu den anderen Visualisierungen für den unerfahrenen Anwender eher als Herausforderung betrachtet werden muss, sollte der naturgemäß technische Charakter des Scatterplots in den Hintergrund gerückt werden und beim ersten Anblick keinen abschreckenden Eindruck hinterlassen. Deshalb wurde das Zoomen durch

einfache Mausklicks realisiert. Beim Druck auf die linke Maustaste wird automatisch der Bereich um den Cursor vergrößert, während ein Rechtsklick die Ansicht wieder verkleinert. Der Zoom-Vorgang wird animiert dargestellt und beendet, sobald die Maustaste wieder losgelassen wird. Dieses Prinzip ist trotz seiner Schlichtheit eine gute Methode, um sich schnell an interessante Daten heranzutasten. Es besitzt nicht die Universalität einer umfangreicheren Interaktionstechnik, ist jedoch schnell und intuitiv zu bedienen.

Wenn man mit der Maus ein Icon überfährt, verwandelt sich der Cursor in eine Hand mit Zeigefinger. Das fokussierte Icon erhält eine intensivere Farbe, und der Titel des Datensatzes wird über dem Icon eingeblendet. Analog zur Tabellenansicht und den anderen Visualisierungen wird der Datensatz in der Titelansicht dargestellt, und ein Mausklick führt zur Selektion des Datensatzes.

Die Achsenbelegungen des Scatterplots können jederzeit über zwei Comboboxen verändert werden. Wenn numerische Kategorien ausgewählt werden, werden geeignete Achsenbeschriftungen erstellt, um alle Werte zwischen Minimum und Maximum darstellen zu können. Durch Veränderungen der Scatterplot-Größe werden dynamisch neue Achsenbeschriftungen erzeugt. Die Ausprägungen textueller Kategorien werden direkt als Achsenbeschriftung übernommen und alphabetisch sortiert visualisiert. Falls es zu viele Achsenbeschriftungen gibt, wird ihre Anzahl auf den zur Verfügung stehenden Platz begrenzt. Die Abb. 18 präsentiert eine numerische und vier textuelle Belegungen der Y-Achse.

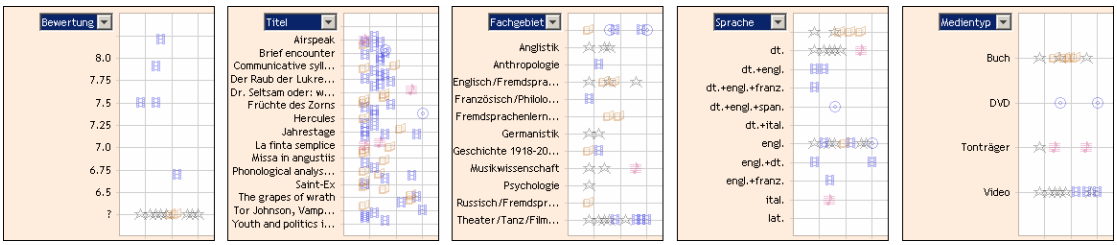


Abb. 18: verschiedene Y-Achsen-Belegungen im Scatterplot

Ein Problem stellen Datenpunkte im Scatterplot dar, die aufgrund gleicher Ausprägungen an derselben Stelle gezeichnet werden müssten. Cleveland schlägt für dieses Problem fünf unterschiedliche Lösungsansätze vor [CW94]:

- Sich überdeckende Datenpunkte können durch zusätzlich eingeblendete Linien um den Mittelpunkt des Datenpunkts visualisiert werden. Diese als *Sunflowers* bezeichnete Methode

gibt Auskunft über die Anzahl der Datensätze eines Datenpunkts, ist aber schlecht erkennbar, sobald sich andere Datenpunkte in der Nähe befinden..

- Aufeinander liegende Datenpunkte können um einen festen Wert (*Moving*) oder über Zufallswerte (*Jittering*) leicht verschoben werden. Dadurch verlieren sie jedoch ihre exakte Position, und bei großen Datenpunktanhäufungen führt eine Verschiebung zur Überdeckung anderer Punkte.
- Eine *logarithmische Darstellung* kann helfen, teilweise überlappende Icons besser zu verteilen. Falls Punkte aber genau an derselben Stelle liegen, hilft eine Verzerrung nicht weiter.
- Die *Open Circles*-Methode stellt ungefüllte Kreise als Datenpunkte dar. Dadurch wird jedoch nicht die Problematik genau übereinander liegender Datenpunkte gelöst, und in MEDIOVIS werden ohnehin Icons als Symbole dargestellt.

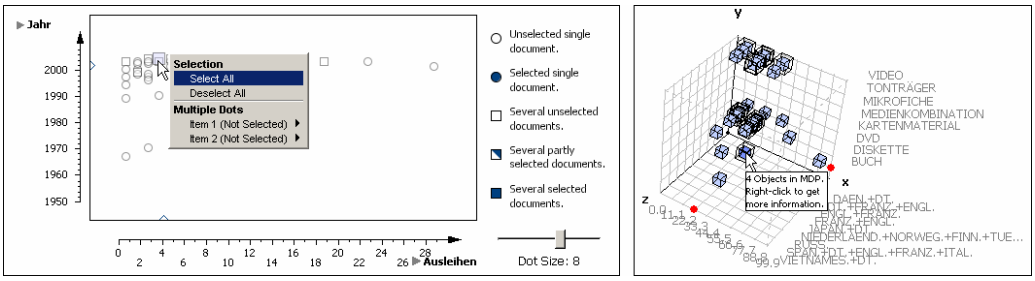


Abb. 19: 2D- und 3D-Scatterplot in VISMEB

Im 2D-Scatterplot von VISMEB (Abb. 19 links) werden alle Datenpunkte als Kreise und übereinander liegende Punkte, hier *Multiple Data Points* (MDP) genannt, als Rechtecke visualisiert. Über ein Popup-Menü erhält man Zugriff auf die einzelnen Datensätze. Der 3D-Scatterplot (Abb. 19 rechts) visualisiert alle Datenpunkte als Würfel. MDPs erhalten einen größeren, transparenten Würfel als Kennzeichnung, und durch einen Klick auf einen MDP wird auf eine weitere Darstellung gewechselt, in der die entsprechenden Datensätze ähnlich wie Karteikarten betrachtet werden können.

Da der MEDIOVIS-Scatterplot ohnehin schon verschiedene Icons nutzt, um die Medientypen zu visualisieren, wurde ein Stern als zusätzliches Icon eingeführt, um auf übereinander liegende Punkte aufmerksam zu machen. Ein Klick auf einen solchen Stern öffnet ein kleines Fenster, in dem alle Datensätze, die unter dem Stern zusammengefasst sind, erneut als Icons präsentiert werden. Die Icons können wiederum, wie in Abb. 20 dargestellt, einzeln fokussiert werden, wodurch ihr Titel eingeblendet und der Datensatz in der Titelansicht angezeigt wird.

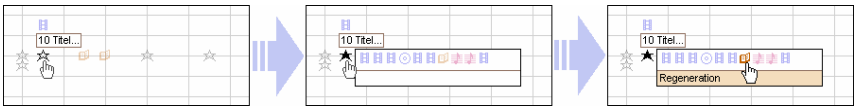


Abb. 20: Auswahl des Stern-Symbols, welches mehrere Datensätze symbolisiert

Ein weiteres, oft ungelöstes Problem in Scatterplots bereiten schlecht gefüllte Datensätze. Alle Datensatz müssen in einem Scatterplot dargestellt werden, unabhängig davon, ob seine Datenfelder für die ausgewählten Achsenbelegungen gefüllt sind oder nicht. In den Mediotheks-Daten fehlt beispielsweise bei ca. jedem sechsten Datensatz die Jahresangabe.[19] Fehlende Zahlen werden intern als »0« und fehlende Texte als Leerstring »« abgebildet. In Abb. 21 links ist der VISMEB-Scatterplot zu sehen. Fehlende Werte werden auf die 0-Position gemappt. In MEDIOVIS wurde eine zusätzliche Achse in den Scatterplot integriert, die mit dem Fragezeichen »?« beschriftet ist. Sie wird eingeblendet, wenn die zu visualisierenden Datensätze für die ausgewählte Achsenbelegung nicht komplett gefüllt sind. Der restliche Platz wird für die gefüllten Achsenwerte genutzt. Dadurch lässt sich der vorhandene Platz besser ausnutzen, ohne dass schlecht gefüllte Datensätze aus der Visualisierung verschwinden.

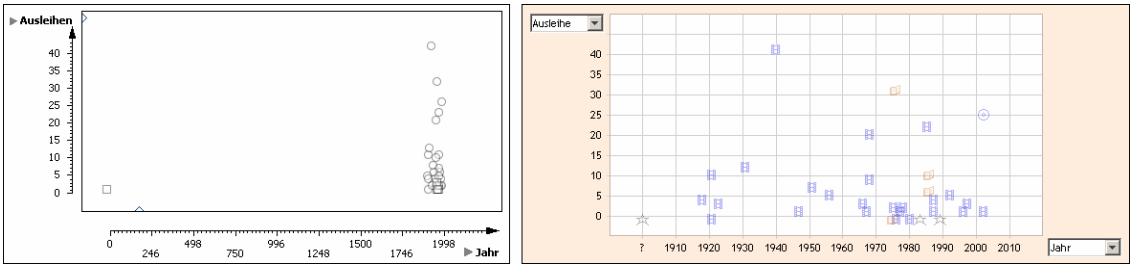


Abb. 21: Darstellung von 40 Bibliothekstiteln mit dem VISMEB- und MEDIOVIS-Scatterplot

Der große Vorteil von Scatterplots ist im Allgemeinen, dass die Daten über zwei Dimensionen dargestellt werden. Somit können mehr Aussagen über die Daten gemacht werden als über listen- und tabellenbasierte Visualisierungen. Allein die visuelle Darstellung der Datensätze gibt oft schon einen schnelleren Einblick in den Datenbestand als eine textuelle Darstellung. Es sollen einige Anwendungsgebiete aufgezählt werden, in denen sich der Scatterplot in MEDIOVIS als dienlich erwiesen hat:

[19] 4147 der 26268 Datensätze verfügen über keine Jahresangabe, bei ca. 24% fehlt der Spracheintrag

- Die Darstellung der Daten nach der Anzahl der Ausleihen ermöglicht eine schnelle Feststellung, welche Bibliothekstitel beliebt zu sein scheinen.
- Die Visualisierung nach Jahresangabe und Anzahl der Ausleihen ist beispielsweise bei Filmen hilfreich, weil ein Blick genügt, um festzustellen, wann ein Regisseur Filme gedreht und wann seine wichtigsten Werke erschienen sind.
- Die Visualisierung der Datensätze nach ihrem Fachgebiet zeigt schnell, in welchem Bereich der Bibliothek sich die meisten Medien befinden, die von allgemeinem Interesse sein könnten.
- Die Icons repräsentieren die Medientypen eines Datensatzes, bieten also eine dritte Dimension, um die Ausprägungen der Datensätze zu erfassen. Bei der Suche nach Komponisten wird man z. B. unabhängig von den Achsenbelegungen auf Tonträger aufmerksam gemacht und sieht dennoch auf einen Blick, welche sonstigen Medien von der gesuchten Person existieren bzw. erschienen sind.

Viele der Erkenntnisse, die sich aus dem Icon-basierten Scatterplot gewinnen lassen, können auch aus einer Liste oder Tabelle erschlossen werden, die visuelle Darstellung ermöglicht jedoch auch spontane Einsichten in die Daten, die sich durch eine textuelle Darstellungsform nicht von alleine erschließen. In den Benutzertests, die in Kapitel 7.1 detaillierter ausgeführt werden, konnte beobachtet werden, dass die Benutzer mit der Scatterplot-Visualisierung spielerisch gearbeitet haben und durch die direkte Interaktion motiviert wurden, den Datenbestand freier zu explorieren, als dies in der Tabelle der Fall war. Für die gewöhnliche Suche wurde jedoch weiterhin die tabellarische Darstellung bevorzugt, was durch die vertrautere Sichtweise auf die Daten erklärt werden kann. Des Weiteren bietet die Visualisierung in der Tat keinen großen Mehrwert für Nutzer, die genau wissen, was sie suchen, und nur möglichst schnell Informationen zu einem bestimmten Mediums herausfinden wollen.

Im Rahmen dieser Arbeit konnten noch keine Evaluationen über längere Zeit durchgeführt werden, daher bleibt vorerst abzuwarten, ob die Benutzer den Scatterplot nach öfterer Nutzung als echte Alternative zur Tabelle akzeptieren oder sogar bevorzugen.

5.3.5 Standort-Ansicht

Zum Auffinden von Büchern in einer Bibliothek sind mehrere Einzelschritte notwendig: als erstes wird das Buch im lokalen Katalogsystem gesucht, und anhand des ihm zugeordneten Fachgebiets und den in den Bibliotheken aufgehängten groben Übersichtskarten kann man den

entsprechenden Bibliotheksbereich ansteuern und sich dort anhand der eventuell vorhandenen näheren Beschriftungen an das relevante Regal vorarbeiten, nicht immer ohne anschließend einen Rechner aufzusuchen, um die nächsten Signaturen nachzuschlagen, die man wieder vergessen hat. Dieser Vorgang ist vermutlich jedem Bibliotheksnutzer vertraut , und er beansprucht viel Zeit, weshalb eine Visualisierung der Medien-Standorte innerhalb der Bibliothek von Anfang an ein wichtiger Bestandteil der MEDIOVIS-Projektidee war und von vielen befragten Benutzern als wünschenswerte Erweiterung des vorhandenen Bibliothekskatalogs gesehen wurde. Dabei existierte meist keine konkrete Vorstellung, ob und wie solch eine Idee realisierbar sei.

In der Tat mussten erst einige konzeptionelle Fragen geklärt werden, bevor die Standort-Visualisierung umgesetzt werden konnte. Das erste Problem stellte der riesige Medienbestand einer Bibliothek dar. Die Mediothek Konstanz umfasst ca. 25.000, die gesamte Bibliothek über zwei Millionen Medien. Es war natürlich von Anfang an ausgeschlossen, den Standort jedes einzelnen Mediums in MEDIOVIS zu erfassen – allein schon deshalb, weil die Bücher regelmäßig umgestellt werden. Die Bibliothekstitel mussten in größere Einheiten zusammengefasst werden. Bei der Analyse der Medienaufstellung wurde schnell ersichtlich, dass die Signatur eine zentrale Rolle bei der Standortvergabe spielt. Der hierarchische Aufbau von Signaturen erleichterte dabei die flexible Definition der Standorte. Eine Signatur an der Universität Konstanz ist folgendermaßen aufgebaut:

Kürzel	Beschreibung	Beispiel
6	Sonderstandort	Mediothek
tff	Fachgebiet	Theater/Tanz/Film/Funk/Fernsehen
710	engerer thematischer Bereich	Einzelne Filmregisseure
t826	genauere Titelbestimmung	Lars von Trier
d16	Individualsignatur eines Exemplars	Dancer in the Dark, 1. Exemplar

Abb. 22: Aufbau der Signatur »6 tff 710:t826/d16«

Die erste Ziffer benennt einen möglichen Sonderstandort des Mediums. Alle Mediothekstitel haben die »6« als Präfix. Die nächsten drei Buchstaben verweisen auf das Fachgebiet (auch Sachgruppe genannt), das dem Medium zugeordnet ist. Der engere thematische Bereich grenzt das Fachgebiet weiter ein. Die genauere Titelbestimmung beginnt mit dem Anfangsbuchstaben des Autors, Regisseurs oder Komponisten eines Werks. Die Individualsignatur schafft eine ein-

deutige Beziehung zu einem Bibliothekstitel und beinhaltet zusätzlich die Exemplarangabe, falls mehrere Exemplare von einem Medium existieren.

Die Räumlichkeiten der Bibliothek lagen zum Zeitpunkt der Umsetzung schon als eingescannte Graphiken vor, selbst die Regale waren eingezeichnet. Deshalb konnte die Karte der Mediothek aus dem gesamten Bibliotheksplan extrahiert und nach einigen visuellen Korrekturen in MEDIOVIS übernommen werden.

Mit Hilfe der Signaturen und der Kenntnis ihres Aufbaus musste nun nicht mehr allen Medien einzeln ihre Position in der Bibliothek zugewiesen werden, sondern es wurde der entgegen gesetzte Weg verfolgt: Teile der Signaturen wurden den Räumlichkeiten der Bibliothek zugewiesen. Der Genauigkeitsgrad der Zuordnung konnte flexibel bestimmt werden. Im einfachsten Fall hätten alle Mediothekstitel, die durch das Präfix »6« gekennzeichnet waren, der ganzen Mediotheks-Karte zugewiesen werden können. Stattdessen wurde die Zuweisung zumeist auf Regalebene durchgeführt (s. Abb. 23). Etwa 300 unterschiedliche Fachgebiete und thematische Bereiche wurden in einer Begehung der Mediothek identifiziert und anschließend dem Kartenmaterial zugeordnet. Bei über 25.000 Bibliothekstiteln kommen so im Schnitt ca. 80 Medien auf eine zugeordnete Einheit, was eine gute Abdeckung darstellt, da in einem Regal oft schon hundert Bücher untergebracht sind. Die dynamische Signatur-Zuordnung ließ hier einen weiteren Vorteil erkennen. Da jedes Fachgebiet mit einer sehr unterschiedliche Medienzahl in der Bibliothek vertreten ist, können sehr gut abgedeckte Fachgebiete über eine detailliertere Signaturenzuordnung genauer zugewiesen werden, während spärlich besetzte Fachgebiete durch die ersten drei Stellen einer Signatur genau genug lokalisiert werden können.

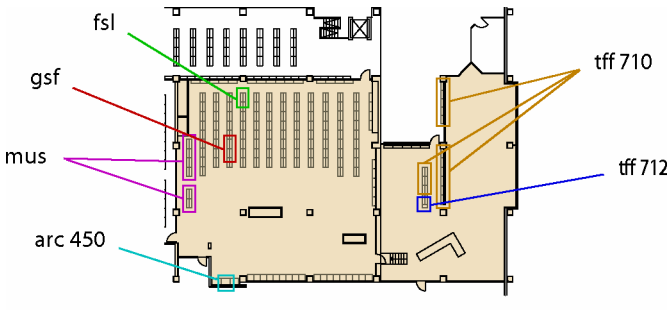


Abb. 23: exemplarische Zuweisung der Fachgebiete und thematischen Bereiche auf das Kartenmaterial der Bibliothek

Der Lageplan der Mediothek wurde in MEDIOVIS als weitere Visualisierung integriert. Durch das Fokussieren eines Bibliothekstitels in einer Visualisierung wird das entsprechende Regal

auf der Karte farbig hervorgehoben. Durch das Überfahren der Bibliothekstitel in der Tabelle oder im Scatterplot kann somit auf einen Blick festgestellt werden, wo die Fachgebiete, die für eine bestimmte Fragestellung von Interesse sind, in der Mediothek untergebracht sind. Die Größe des Lageplans wird beim Vergrößern und Verkleinern des Anwendungsfensters automatisch angepasst, um den vorhandenen Platz optimal auszunutzen (s. Abb. 24).

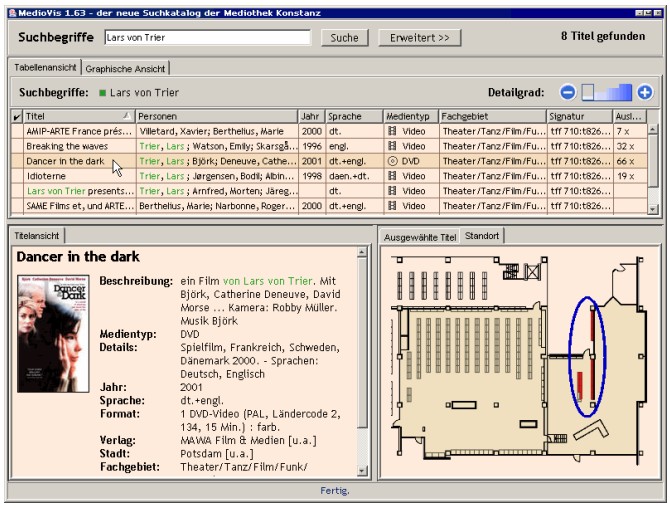


Abb. 24: MEDIOVIS-Hauptfenster mit Standort-Visualisierung und gesondert in blau hervorgehobener Standort-Markierung

Der Standort-Visualisierung wurde in den Benutzertests große Beachtung geschenkt. Aufgrund der übersichtlichen Größe der Mediothek konnten sich die Benutzer die tatsächliche Position der Medien schnell einprägen und die gesuchten Medien auf Anhieb in den Regalen ausfindig machen.

Der Lageplan für die Mediothek konnte aufgrund der beschränkten Räumlichkeiten effizient umgesetzt werden. Eine Standort-Visualisierung für eine größere Bibliothek wirft jedoch neue Probleme auf:

- der Lageplan der Mediothek wurde auf die komplette Darstellung am Bildschirm optimiert. Da größere Räumlichkeiten nicht mehr komplett sichtbar dargestellt werden können, werden neue Interaktionen nötig, um Lagepläne vergrößern und verkleinern zu können. Dem Benutzer muss neben den Medien-Standorten zugleich verdeutlicht werden, wo er sich gerade selbst befindet.
- Räumlichkeiten, die sich über mehrere Stockwerke erstrecken, verlangen dem Benutzer eine bessere räumliche Vorstellungskraft ab, da der Mensch zwar imstande ist, dreidimensionale Strukturen zu erkennen, diese jedoch anhand einer zweidimensionalen Wahrnehmung re-

konstruiert. Der Bildschirm eines Computers kann wiederum beliebige Strukturen nur zweidimensional abbilden, wodurch selbst dreidimensionale Computer-Modelle nur beschränkt visualisiert werden können. Die Orientierung in den Räumlichkeiten muss somit durch eine ausgefeilte Interaktion und zusätzliche Visualisierungskonzepte unterstützt werden.

- Im konkreten Beispiel der Universitätsbibliothek Konstanz repräsentiert die Mediothek nur ungefähr ein Achtzigstel des gesamten Umfangs. Die Erfassung der Standorte für alle Bibliothekstitel nähme somit deutlich mehr Zeit in Anspruch als die Begehung der Mediothek. Die komplette Übernahme der Standort-Daten der Mediothek konnte zwar in ca. einer Stunde abgeschlossen werden, die konsistente Aufstellung der Mediothekstitel nach Signaturen war jedoch die Grundlage für eine effektive Erfassung. Die gesamte Bibliothek verfügt über zahlreiche Sonderstandorte, weshalb sich Bibliothekstitel mit ähnlichen Signaturen an recht unterschiedlichen Stellen befinden können; eine pauschale Standort-Vergabe über Teile der Signatur ist somit nicht in allen Fällen gewährleistet.

Ein weiteres Problem stellt die Pflege einmal eingegebener Standort-Daten dar. Der Medienbestand von Bibliotheken wandelt sich kontinuierlich, weshalb Bücher umsortiert, Regale umgestellt oder neue Räumlichkeiten geschaffen werden. Für MEDIOVIS wurde ein prototypischer *Standort-Editor* entworfen, der zur Eingabe der verfügbaren Standort-Daten genutzt wurde (Abb. 25). Solch ein Editor könnte mit vertretbarem Aufwand weiter entwickelt werden, um beliebigen Anwendern die Pflege von Standort-Daten für beliebige Lagepläne zu ermöglichen. Solange der Prozess der Standort-Vergabe jedoch in absehbarer Zeit nicht völlig automatisiert werden kann, entsteht durch digitale Lagepläne ein unumgänglicher zukünftiger Mehraufwand für die Mitarbeiter der Bibliothek.

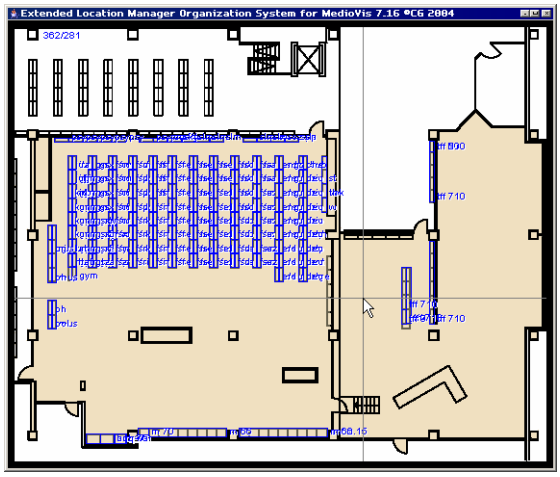


Abb. 25: Prototyp des MEDIOVIS Standort-Editors

Das Konzept der Standort-Visualisierung wird jedoch wieder sehr viel versprechend, wenn mobile Geräte eingesetzt werden, mit denen man sich mit MEDIOVIS an beliebigen Orten Informationen über den Bibliotheksbestand abrufen kann.

5.3.6 Table Filter

Mit den gegebenen Visualisierungen sind die grundlegenden Anforderungen abgedeckt, um Bibliothekstitel effizient aufzufinden. Darstellungen wie der Scatterplot erlauben zudem weitere Aussagen über die Daten, die mit dem konventionellen Katalogsystem nicht erfassbar sind. Ein übliches Feature professioneller Visualisierungs-Systeme stellt jedoch die Filterung angezeigter Datensätze dar. Auch VISMEB verfügt über unterschiedliche Methoden zur Filterung der Daten, die in Form von Dialogen oder als zusätzliche Interaktionskomponenten in den Visualisierungen integriert wurden. Um die Datensätze in MEDIOVIS einzuschränken, muss eine neue Suche gestartet werden, in der explizit die Filterkriterien über das erweiterte Suchformular festgelegt werden. Einen weiteren Ansatz stellt der *Table Filter* dar, der schon voll funktionsfähig in MEDIOVIS implementiert wurde, in Abb. 26 abgebildet ist und in seinen Grundzügen vorgestellt werden soll.

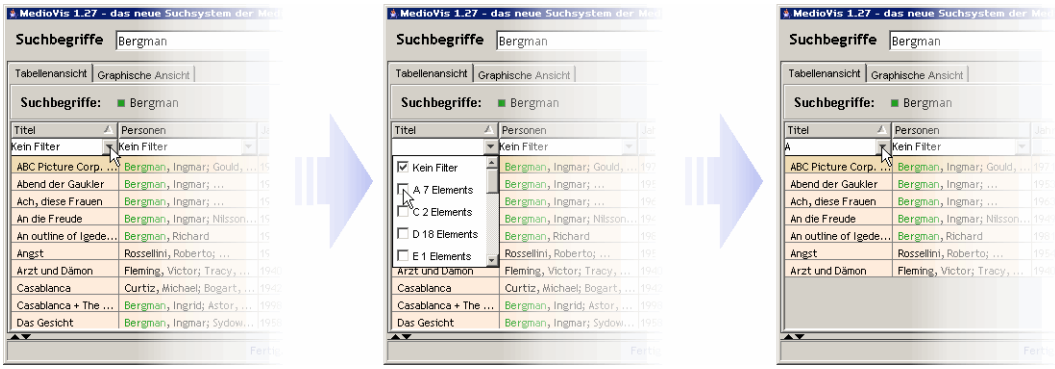


Abb. 26: Filterung über den *Table Filter*; Auswahl aller Titel mit »A« als Anfangsbuchstaben

Unter den Spaltenköpfen der Tabelle werden weitere Interaktionskomponenten eingeblendet, die für jede der angezeigten Kategorien eine Filterung der Daten erlauben. Im abgebildeten Beispiel wurde der Titel-Filter aufgerufen. Er listet die Anfangsbuchstaben der vorhandenen Titel und die Anzahl der Treffer pro Buchstabe in einer Combobox auf. Jeder Eintrag der Combobox ist mit einer Checkbox versehen, die angeklickt werden kann, um die dargestellten Treffer auf die ausgewählten Anfangsbuchstaben zu beschränken. Wie in der rechten Abbildung zu sehen ist, werden nach der Filterung nur noch die Datensätze angezeigt, die mit dem Buchstaben »A« beginnen. Die Beschriftung »kein Filter« im Combobox-Feld wurde durch die entsprechende Filterauswahl ausgewechselt. Über die Checkboxen können weitere Buchstaben gefiltert werden. Das Textfeld der Combobox kann aber auch direkt genutzt werden, um die gewünschte Filterdefinition einzutippen; die Eingabe von »Bergman« führt beispielsweise zur Filterung

nach allen Datensätzen, die unabhängig von der Groß- und Kleinschreibung mit »Bergman« beginnen.

Abhängig von den Datentypen einer Kategorie wird einer von drei Filtertypen in der Filterzelle eingeblendet:

- *Nominale Daten* sind textuelle Daten, wie beispielsweise die Titel der Datensätze. Sie werden nach ihren Anfangsbuchstaben in Kategorien zusammengefasst.
- *Ordinale Daten* sind ebenfalls textuelle Daten, die über kategorische Ausprägungen verfügen. Hierunter fällt z. B. der Medientyp oder das Fachgebiet. Die vorhandenen Kategorien werden direkt als Filterkategorien übernommen.
- *Numerische Daten* wie das Jahr können direkt mit Hilfe eines *Range Sliders* gefiltert werden (ein solcher Slider wurde bereits auf Seite 49 in Abb. 17 vorgestellt).

Wenn mehrere Filter eingesetzt werden, werden nur noch die Datensätze angezeigt, die die Einschränkungen aller Filter erfüllen, d. h. die Filter werden untereinander UND-verknüpft. Ein Filter selbst verknüpft seine Auswahl jedoch über die ODER-Verknüpfung. An einem Beispiel konkretisiert heißt das, dass die Auswahl der Buchstaben »A« und »B« im Titel-Filter und die Auswahl von Videos im Medientyp-Filter nur noch Datensätze anzeigt, deren Titel mit einem »A« ODER »B« beginnen UND die als »Video« gekennzeichnet sind.

Besonders das Feature, die gewünschte Filterung direkt in das Textfeld eingeben zu können, ermöglicht ein schnelles Auffinden einzelner Titel, ohne dass der Datenbestand mittels einer weiteren Suchanfrage erneut spezifiziert werden muss. Die Implementierung des jetzigen *Table Filters* kann jedoch trotz seiner hilfreichen Funktionen zu einigen Verständnisproblemen führen:

- die UND-Verknüpfung über mehrere Filter ist auf den ersten Blick verwirrend, da durch wenige Mausklicks alle Datensätze aus der Tabelle verschwinden können.
- durch die Filterung der Daten werden weniger Daten angezeigt als davor. Dementsprechend müssten sich die Einträge der anderen Filter dynamisch anpassen: wenn keine Datensätze mit Medientyp »Video« angezeigt werden, macht eine Filteroption für Videos eigentlich keinen Sinn mehr.
- Vor der Filterung sind alle Checkboxen der Filter deaktiviert, und es werden alle Daten dargestellt. Nach der Aktivierung einer Checkbox sind plötzlich nur noch die gefilterten Buch-

staben zu sehen. Die standardmäßige Aktivierung aller Checkboxen würde jedoch wiederum die Filterung nach einzelnen Buchstaben mühsam gestalten, da alle anderen Checkboxen zuvor deaktiviert werden müssten.

Es gibt eine Reihe von Ansätzen zur Vereinheitlichung des *Table Filters*, die alle in sich konsistent, aber zu sehr an der mathematischen Logik der Filterfunktionalität verhaftet sind und nicht intuitiv zugänglich sind. Ein Filter muss selbstverständlich korrekt mathematisch arbeiten, zugleich muss seine Funktionsweise aber für den Benutzer nachvollziehbar sein. Daher wird zukünftig vermutlich eine vereinfachte Version des *Table Filters* in das System Eingang finden, in der die Comboboxen durch einfache Textfelder ersetzt und die Eingaben komplett als Filterbegriffe interpretiert werden, die an beliebigen Stellen in den Zelltexten vorkommen können. Die direkte Reaktion auf Benutzereingaben bleibt dadurch erhalten, und des weiteren müssen die Filterbegriffe nicht mehr unbedingt am Anfang der Zelltexte auftreten, was sich bei Einträgen mit Artikeln (»Der goldene Schnitt«, »The Game« etc.) positiv auswirkt..

5.3.7 Media Grid

Die *Media Grid* ist ein neues, innovatives Konzept, welches bisher nur in die Forschungsversion von MEDIOVIS integriert wurde, jedoch einige andere Visualisierungen in sich aufnehmen und diese als selbständige Visualisierungen überflüssig machen könnte. Die Grundlagen der Media Grid, einer zoombaren Tabellenansicht (auch *Drill-Down Table* oder *Media Lens* genannt), wurden bereits 1981 geschaffen. Die Veröffentlichung von Furnas über *Fisheye Views* zog zahlreiche weitere Publikationen im Bereich *Zoomable User Interfaces* (*ZUI*) nach sich [FG81]. Ein Fisheye View stellt einen fokussierten Bildschirmbereich vergrößert dar und verkleinert die restlichen Bildinformationen, ohne dass sie vom Bildschirm verschwinden (s. Abb. 27). Durch diese Art der Darstellung bleibt der Kontext zwischen Detail- und Gesamtansicht gewahrt, und der Fokus kann ohne zusätzliche Interaktionskonzepte schnell verändert werden.

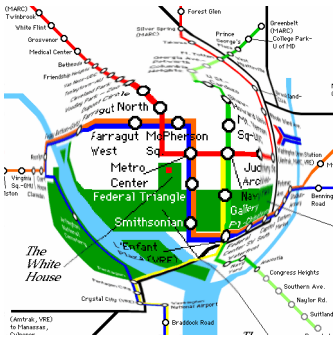


Abb. 27: *Fisheye View* der Karte von Washington D.C. [FD04]

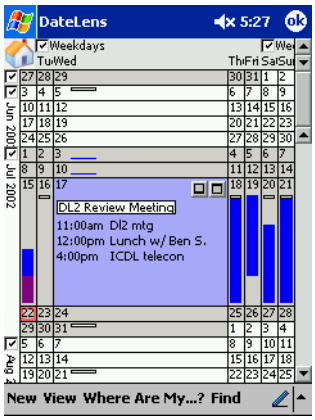


Abb. 28: Pocket PC Version der *Date Lens* [BC02]

Auf Basis des Fisheye Views entstand u. a. die *Table Lens* [RC94], die auch in VISMEB integriert wurde, und die *Date Lens*, die speziell für mobile Geräte entwickelt wurde und bei der Konzeption der *Media Grid* als wichtige Inspirationsquelle diente (Abb. 28) [BC02]. Da Table Lens, Date Lens und Media Grid vorwiegend textuelle Inhalte visualisieren, werden dargestellten Zeilen und Spalten im Gegensatz zum Fisheye View nicht assymetrisch verzerrt, sondern nur verschoben und so eine gute Lesbarkeit der Texte beibehalten.

Jede Zelle der Media Grid kann in unterschiedlichen *Granularitätsstufen* dargestellt werden[20]. Die Stufe bestimmt dabei die Informationstiefe einer Zelle: in einer tieferen Stufe werden mehr Informationen visualisiert, jedoch auch mehr Bildschirmplatz benötigt. Die Zellen verfügen in der momentanen Implementierung über maximal vier Granularitätsstufen, und alle Zellen werden anfänglich in der niedrigsten Granularitätsstufe visualisiert. Die Visualisierung ähnelt somit anfangs der tabellarischen Datenansicht. Die wichtigsten Kategorien des Bibliothekskatalogs wurden im Rahmen eines Brainstormings und weiterer Überlegungen den sieben Spalten zugeordnet und lassen sich aus der ersten Granularitätsstufe der Abb. 29 ablesen. Die Zeilen repräsentieren wie gewohnt die Ausprägungen der Datensätze. Sobald eine einzelne Zelle nun angeklickt wird, wechselt diese in die nächsttiefere Granularitätsstufe und ergänzt die jeweilige Hauptkategorie durch zusätzliche Daten. Im Gegensatz zu

	Kat. 1	Kat. 2	Kat. 3	Kat. 4	Kat. 5	Kat. 6	Kat. 7
I	Jahr	Sprache	Status	Titel	Fachgebiet	Medientyp	Ausleihen
II		Stadt Verlag	Leihfrist	Untertitel Originaltitel	Signatur	Format ISBN / ISSN	
III				Plakat Beschreibung Details			
IV				Trailer			

Abb. 29: Darstellung der vier Granularitätsstufen der *Media Grid*

[20] Die Idee der Benutzung von Granularitätsstufen fand sich bereits in der Granularity Table und selbst in der Level Table von VISMEB

konventionellen Tabellendarstellungen erscheinen die Daten also direkt am Ort des Geschehens, ohne dass ein visueller Kontextwechsel stattfindet. Da zur Jahresangabe und den Ausleihen eines Datensatzes keine weiteren Angaben existieren, ermöglichen die links und rechts platzierten Kategorien keine tiefere Granularitätsstufe. Der Titel wurde als zentrale Datensatz-Kategorie in der horizontalen Bildschirmmitte angesiedelt, da er die meisten zusätzlichen Detailinformationen bietet und vier Granularitätsstufen bietet.

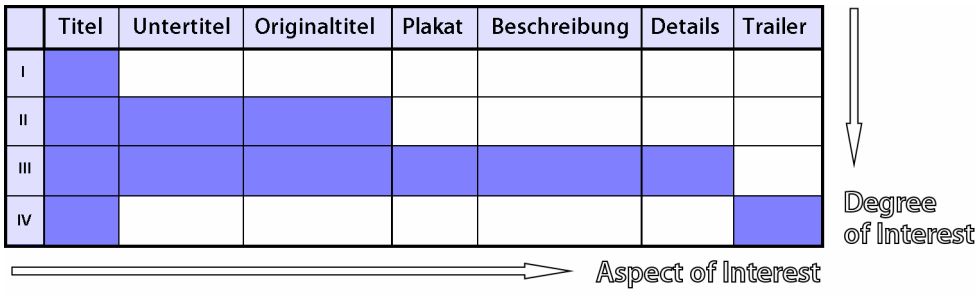


Abb. 30: Darstellung der Präsentationsmatrix für die Titel-Spalte der *Media Grid*

Ein weiteres Konzept zum Verteilen der Datenfelder auf die Zellen basiert auf der Erstellung von *Präsentationsmatrizen* (Abb. 30) [RH98]. Der Detaillierungsgrad (Degree of Interest, kurz DOI) der Matrix entspricht den Granularitätsstufen der Media Grid, die Kategorien (Aspect of Interest, AOI) werden hingegen horizontal abgetragen. Die Darstellung über die Präsentationsmatrix hat den Vorteil, dass hinzugefügte Kategorien in einem höheren Detaillierungsgrad wieder entfernt werden können; der Platzverbrauch ist jedoch größer, da sich eine Präsentationsmatrix nur auf die Modellierung einer einzelnen Spalte der Media Grid beschränkt.

Das Zoomen einer Zelle hat zur Folge, dass die restlichen Zellen am Bildschirm ebenfalls eine Größenveränderung erfahren. Da die Tabellenstruktur erhalten und eine assymetrische Verzerrung vermieden werden soll, werden neben der fokussierten Zelle alle Zellen derselben Spalte und Zeile ebenfalls vergrößert. Dies mag auf den ersten Eindruck hin problematisch erscheinen. Da die Zellen einer Zeile jedoch denselben Datensatz repräsentieren und somit die Informationen der gezoomten Zelle sinnvoll ergänzen können, bietet es sich an, dort ebenfalls in die nächsttiefere Granularitätsstufe zu wechseln, selbst wenn die Zellen nicht breit genug sein werden, um alle vorhandenen Informationen komplett darstellen zu können.

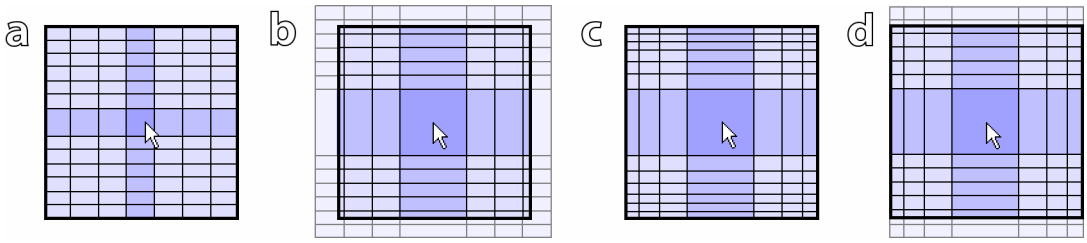


Abb. 31: Zoom-Varianten in einer Zellenmatrix: a) ungezoomte Darstellung, b) Zoom ohne Zellen-Verzerrung, c) Zoom mit Verzerrung, d) Zoom in Media Grid

Es gibt grundsätzlich zwei Varianten, wie die restlichen Zellen am Bildschirm angeordnet werden können:

- In Abb. 31 b) behalten die Zellen um die gezoomte Zelle herum ihre ursprüngliche Größe, indem sie einfach verschoben werden und aus dem sichtbaren Bereich verschwinden. Der Vorteil dieser Vorgehensweise ist, dass die Texte der noch sichtbaren Zellen komplett lesbar sind. Der eindeutige Nachteil der Verschiebung ist, dass die Zellen nicht mehr komplett am Bildschirm zu sehen sind.
- In Abb. 31 c) sind alle Zellen weiterhin sichtbar, jedoch deutlich zusammengestaucht. Dadurch leidet jedoch die Lesbarkeit der am Rand angeordneten Zellen.

In der Media Grid wurde ein Kompromiss zwischen beiden Alternativen gewählt, wie er in Abb. 31 d) zu sehen ist. Die Spalten werden gestaucht, so dass die anderen Kategorien des fokussierten Datensätze immer komplett sichtbar bleiben. Die Zeilen werden jedoch lediglich verschoben und sind wie in der Tabelle über eine Scrollbar erreichbar, da ihre Anzahl ohnehin meist zu groß ist, als dass sie komplett und gut lesbar im sichtbaren Bereich angeordnet werden könnten. Da die Zeilen nicht verzerrt werden, können mehrere Datensätze zugleich bis zur tiefsten verfügbaren Granularitätsstufe aufgeklappt und miteinander verglichen werden, ohne dass sie durch das Aufklappen anderer Zeilen wieder gestaucht werden.

Abb. 32 zeigt eine Zeile aus der Media Grid in den vier verfügbaren Granularitätsstufen. Um mehr Informationen über einen Bibliothekstitel zu erhalten, kann eine Zelle durch einen Mausklick aufgezoomt werden, wodurch die Zellen nebenan horizontal verkleinert werden. Der Zoom-Vorgang wird in Echtzeit animiert, um den Zusammenhang zwischen den verschiedenen Granularitäten zu veranschaulichen. Neben dem Untertitel, der unter dem Haupttitel erscheint, werden auch die benachbarten Zellen mit weiteren Informationen gefüllt. Ein weiterer Klick zeigt, sofern vorhanden, das Filmplakat eines Films und gibt zusätzliche Informationen zum Titel wieder. Die vierte Granularitätsstufe blendet schließlich einen Film-Trailer in die Zelle ein,

der nach dem Ladevorgang automatisch abläuft und mit einem weiteren Mausklick auf dem ganzen Bildschirm im Fullscreen-Modus angezeigt werden kann. Sobald andere Zellen derselben Zeile angeklickt werden oder die Zelle mit der rechten Maustaste wieder verkleinert wird, wechselt sie wieder in eine niedrigere Granularitätsstufe.

Abb. 32: einzelne Zeile der *Media Grid*, visualisiert in den vier Granularitätsstufen

Besonderer Wert wurde auf die Implementierung eines Algorithmus gelegt, der die Größe der Zellen auf die Darstellung möglichst vieler Informationen hin optimiert. Die gesamte Media Grid kann somit auch stark vergrößert und verkleinert werden, wodurch eine gute Lesbarkeit der Texte gewährleistet bleibt. Da jeder Datensatz der Bibliothek unterschiedlich viele Daten enthält, entstehen (wie in Abb. 33 links zu sehen) immer bestimmte Freiräume, die nie komplett mit Informationen gefüllt werden. Die Freiräume treten hauptsächlich dann auf, wenn Zellen in tiefere Granularitätsstufen aufgeklappt werden, da Filmplakate und Trailer einen bestimmten Platz in die Vertikale beanspruchen, der schlecht minimiert werden kann. Der interaktive Wechsel der Granularitäten und die Möglichkeit, beliebig viele Zellen in unterschiedlichen Granularitätsstufen darzustellen, tragen jedoch dazu bei, dass auftretende Freiräume kaum ins Gewicht fallen.

Bei einer konsequenten Weiterentwicklung der *Media Grid* werden die restlichen Visualisierungen immer überflüssiger. Die Titelansicht enthält schon jetzt nicht mehr viele zusätzliche Informationen, da die meisten Daten der Bibliothekstitel schon über die unterschiedlichen Granu-

Abb. 33: MEDIOVIS-Hauptfenster mit *Media Grid* (weitere Visualisierungen wurden ausgeblendet); Auflösung 1024 x 768 Pixel / 640 x 480 Pixel

laritätsstufen verteilt wurden. Die Standort-Ansicht wurde ebenfalls schon als Zusatzinformation zum Fachgebiet und zur Signatur in die Media Grid implementiert. Auch die Anzeige der ausgewählten Titel ist mit der Visualisierung möglich, da die Zeilen mittels der Combobox, die rechts oben im Fenster platziert ist, auf die Anzeige der selektierten Titel beschränkt werden kann. Da der Mausklick in der Media Grid für das Aufklappen der Zellen genutzt wird, wurde links vor den Titeln eine kleine Checkbox eingeblendet, mit denen Datensätze weiterhin selektiert werden können. Die Buttons zum Versenden, Speichern und Drucken der Titel können problemlos in die Media Grid integriert werden und unabhängig vom ausgewählten View (alle Titel / selektierte Titel) die aktuell angezeigten Titel verarbeiten.

Zusammenfassend soll noch einmal als großer Vorteil der *Media Grid* hervorgehoben werden, dass die angeforderten Informationen immer direkt dort eingeblendet werden, wo der Benutzer seine Interaktionen ausführt, ohne dass die Gesamtübersicht über die Daten verloren geht. Die Einfachheit der Interaktion über Mausklicks macht weitere Interaktionskomponenten überflüssig, weshalb die Visualisierung ohne großen Lernaufwand bedient werden kann. Obwohl zoombare Tabellendarstellungen bis heute kaum in Standard-Softwareanwendungen zu finden sind, haben die ersten Tests gezeigt, dass sich die Benutzer sehr schnell an das Zoomen der

Zellen gewöhnt haben und immer überrascht waren, wenn die gewöhnliche Tabellenansicht auf Mausklicks plötzlich statisch blieb.

5.4 Interaktionen

In Kapitel 2.6 wurden Überlegungen angestellt, wie die Benutzerinteraktion in Computer-Systemen aussehen sollte. Die ersten konzeptionellen Überlegungen bezüglich MEDIOVIS setzten sich ganz grundlegend mit den vorhandenen Eingabegeräten auseinander, über die der Benutzer mit dem Computer kommunizieren kann. Ein gewöhnliches Computer-System ermöglicht die Interaktion per Maus und Tastatur. Die Maus erlaubt Bewegungen des Maus-Cursors und bietet zwei Knöpfe an, mit denen Aktionen ausgelöst werden können. Über die Kombination von Bewegung und Mausklicks lassen sich zudem auch Drag'n'Drop-Operationen durchführen, mit denen Objekte am Bildschirm verschoben werden können. Die Tastatur verfügt neben dem Standard-Zeichensatz über Navigationstasten (Cursor-Tasten, Bild-Auf/Ab usw.), Aktionstasten (Eingabe-Taste, Escape, Funktionstasten F1-F12 u. a.) und Kombinationstasten (Shift, Strg, Alt).

Demgegenüber bieten graphische Oberflächen Interaktionskomponenten an, die auf die Interaktionen des Benutzers reagieren. Die Interaktionskomponenten sind mehr oder weniger standardisiert. Visualisierungen wie Textfelder, Buttons, Combo-Boxen oder Scrollbars werden im ganzen Betriebssystem und an zahlreichen Stellen in Anwendungsprogrammen eingesetzt und werden auch als Widgets[21] bezeichnet. Sie sind den Nutzern durch ihr wiederholtes Auftreten relativ bald vertraut. Komplexere Visualisierungen wie Tabellen und graphische Darstellungen können Widgets aufnehmen und eigene Interaktionen anbieten.

5.4.1 Mausbewegungen

Die einfachste und direkteste Interaktion mit graphischen Oberflächen lässt sich durch die Bewegung der Maus erreichen, wodurch der Maus-Cursor seine Position am Bildschirm verändert. Neben der Cursor-Bewegung lassen sich jedoch noch zahlreiche weitere Aktionen auslösen, die für den Benutzer hilfreich sein können:

[21] »Widget« ist eine Kombination aus »Window« und »Gadget« und bezeichnet standardisierte Interaktionskomponenten

- das Aussehen des Cursor kann verändert werden, um auf Programmzustände und mögliche Aktionen hinzuweisen (s. Abb. 34). Während der Mauspfeil in allen Betriebssystemen und den meisten Anwendungen als Standard eingesetzt wird und die Sanduhr Wartezeiten visualisiert, verweist die Hand mit ausgestrecktem Zeigefinger auf eine mögliche Aktion. Die Hand wurde speziell über die Internet-Browser zum Synonym für den Verweis, dass sich die Maus über etwas Klickbarem befindet. Sie wird in MEDIOVIS u. a. im *Scatterplot* eingeblendet, wenn sich der Cursor über einem Icon befindet. Die Lupe ist aus Graphikprogrammen bekannt und verweist im Scatterplot auf die Zoom-Funktionalität, die ebenfalls durch Klicks ausgelöst wird. Die vier Symbole rechts in der Abbildung werden in der *Media Grid* genutzt. Sie visualisieren, ob die Zelle unter dem Cursor im momentanen Zustand weiter auf- oder zugeklappt werden kann.

Abb. 34: In MEDIOVIS verwendete Cursor-Formen

- neben dem Cursor selbst können auch Interaktionskomponenten eines Programms auf Mausbewegungen reagieren. Wenn der Benutzer Textfelder und Combo-Boxen mit der Maus in MEDIOVIS überfährt, werden diese automatisch aktiviert, ohne dass sie angeklickt werden müssen. Zum einen stellt die automatische Fokussierung einen Aufforderungscharakter dar, die Komponenten zu nutzen, zum anderen wird dadurch die schnelle Eingabe von Texten erleichtert.

- als besonders interessant hat sich die Möglichkeit herausgestellt, die Visualisierungen selbst auf Mausbewegungen reagieren zu lassen. Ein Großteil der Informationen kann nun somit in MEDIOVIS abgerufen werden, ohne dass man explizite Aktionen durchführt. In der Tabellenansicht und dem Scatterplot werden die Datensätze, die sich unter dem Cursor befinden, automatisch *fokussiert*. Durch die Fokussierung erfahren auch alle anderen Visualisierungen, welcher Datensatz gerade im Mittelpunkt des Interesses steht. Diese Technik bezeichnet man allgemein als *Linking & Brushing*[22]. Da die Visualisierungen von MEDIOVIS verschiedene Funktionen erfüllen, reagiert jede Visualisierung unterschiedlich auf den Fokus. Die Tabellenansicht und der Scatterplot heben den fokussierten Datensatz visuell hervor, um ihn von

[22] Unter *Linking* (verknüpfen) versteht man die Verknüpfung mehrerer Visualisierungen untereinander, während die direkt-manipulative Interaktion mit dem Dargestellten als *Brushing* (streifen) bezeichnet wird.

den anderen dargestellten Datensätzen zu unterscheiden. Die Standort-Ansicht markiert die entsprechenden Regale, an denen sich das fokussierte Medium befindet. Die Titelansicht zeigt hingegen den fokussierten Datensatz in Vollansicht dar. Die Technik, fokussierte Datensätze einzeln und alle Datensätze in der Übersicht darzustellen, wird als *Focus & Context* oder auch *Detail & Overview* bezeichnet.

Zu viel Interaktivität kann sich jedoch auch negativ auf die intuitive Bedienung eines Programms auswirken. In der Media Grid wurde beispielsweise ein Testlauf gestartet, indem die dargestellten Zellen allein durch Mausbewegungen aufgeklappt wurden. Die Visualisierung befand sich somit in ständiger Bewegung, und durch kleine Verschiebungen des Maus-Cursors wurden ständig neue Zellen gezoomt. Die hohe Interaktivität erschien für eine professionelle Nutzung der Visualisierung halbwegs sinnvoll, gewöhnliche Nutzer wären aber vermutlich durch den ständigen Fokus-Wechsel völlig überfordert.

5.4.2 Mausklicks

Die zwei Maustasten werden standardgemäß genutzt, um explizite Aktionen im Programm auszulösen. In MEDIOVIS erfüllen Mausklicks grundsätzlich zwei Aufgaben:

- Widgets wie Buttons oder Combo-Boxen verhalten sich konform zum Betriebssystem. Durch Mausklicks lassen sich beispielsweise Kategorien in den Combo-Boxen auswählen oder über den Suche-Button die Suche starten.
- Interessanter sind die Interaktionen mit den Visualisierungen. Mit der linken Maustaste können Datensätze in der Tabelle *selektiert* (ausgewählt) werden, wodurch sie der Ansicht der ausgewählten Titel hinzugefügt werden. Im Scatterplot ist die Klick-Aktion vom Aussehen des Maus-Cursors abhängig. Wenn sich der Cursor über einem Icon befindet, werden die Datensätze selektiert, wenn er das Aussehen der Lupe hat, wird in den Scatterplot hineingezoomt.

Da keine Popup-Menüs in MEDIOVIS verwendet werden, hat die rechte Maustaste eine untergeordnete Bedeutung. Sie erfüllt zumeist die gegenteilige Wirkung des linken Mausklicks und wird im Scatterplot genutzt, um den gezoomten Bereich wieder zu verkleinern. In der Media Grid können Zellen mit der rechten Maustaste wieder zugeklappt werden.

5.4.3 Drag'n'Drop

Die Visualisierungen sind nach dem Start von MEDIOVIS immer an derselben Position zu finden. Sie können jedoch in den vier verfügbaren *Panels*[23] beliebig angeordnet werden. In Abb. 35 wird links oben die Media Grid angezeigt, rechts gefolgt von der Titelansicht. Der Scatterplot und die Standort-Ansicht wurden im unteren Fensterbereich angeordnet.

Die individuelle Anordnung ermöglicht neue Zusammenspiele unter den Visualisierungen. Interessant ist beispielsweise die gleichzeitige Ansicht der Tabelle und des Scatterplots, wie dies in VISMEB standardmäßig der Fall war.

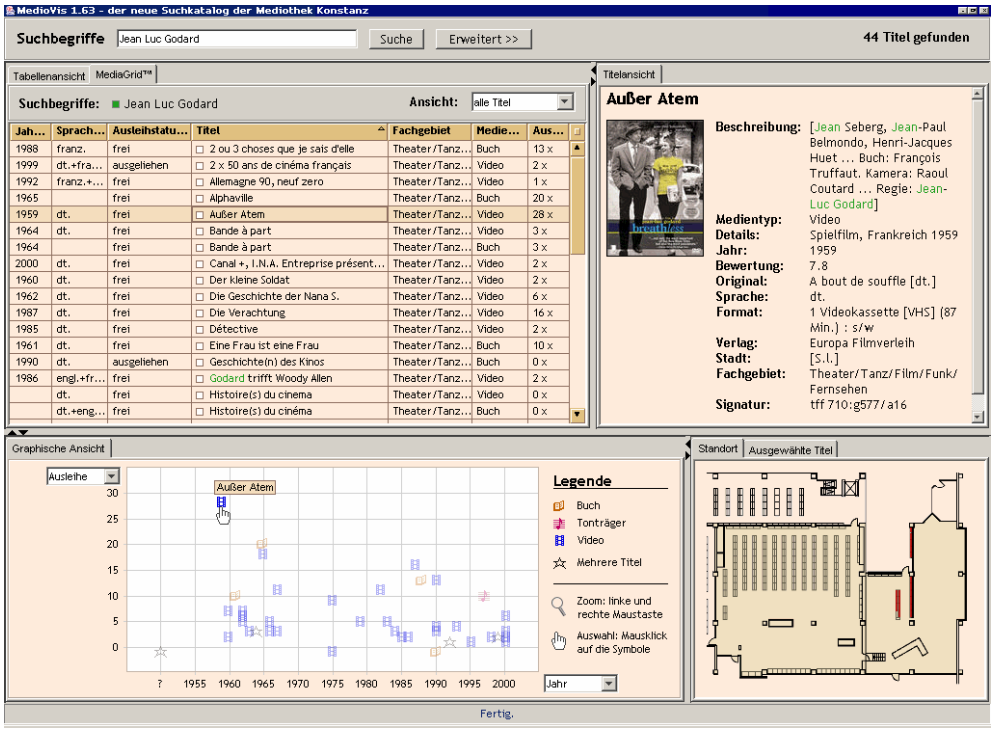


Abb. 35: MEDIOVIS-Hauptfenster; Individuelle Anordnung der Visualisierungen

Die Neuanordnung der Visualisierungen wurde über Drag'n'Drop realisiert. Um eine Ansicht zu verschieben, muss lediglich die Beschriftung der Visualisierung angeklickt und mit gedrückter Maustaste in ein anderes Panel gezogen werden, wie dies in Abb. 36 veranschaulicht ist. Die einzelnen Panels sind zudem durch Balken voneinander getrennt. Diese Balken können eben-

[23] Als Panel bezeichnet man in JAVA eine graphische Fläche, die mit beliebigen Inhalten gefüllt werden kann. In MEDIOVIS werden vier Panels angeboten, von denen drei standardmäßig mit Visualisierungen belegt sind.

falls mit gedrückter Maustaste beliebig verschoben werden, um einzelne Visualisierungen zu vergrößern oder zu verkleinern. Das vierte Panel ist am rechten oberen Rand über den Balken erkennbar und lässt sich aufklappen, um Platz für eine weitere, gleichzeitig sichtbare Visualisierung zu schaffen.

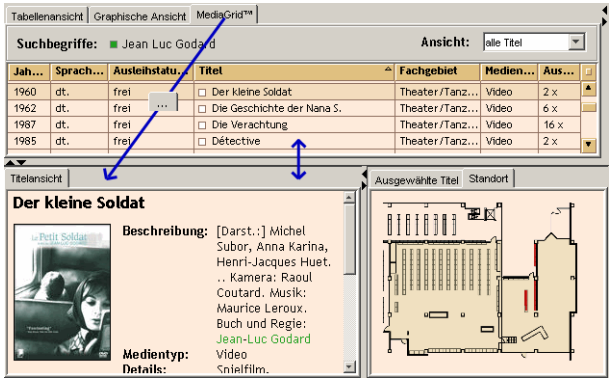


Abb. 36: Verschieben und Vergrößern/Verkleinern der Visualisierungen per Drag'n'Drop

Die Drag'n'Drop-Features wurden nicht weiter in MEDIOVIS dokumentiert, da sie eher von fortgeschrittenen Benutzern genützt würden und das System vorrangig für den gewöhnlichen Bibliotheksbenutzer implementiert wurde.

5.4.4 Tastatureingaben

Die meisten Benutzer bevorzugen die Maus auf graphischen Oberflächen als Navigationswerkzeug. Da die Tastatur jedoch in vielen Fällen die schnelle, zielgerichtete Navigation erleichtern kann, wurden auch erwartungskonforme Tastatursteuerungen in das System implementiert.

Zur Eingabe von Suchbegriffen ist die Tastatur als Eingabeinstrument natürlich unerlässlich, sie findet aber auch in den Visualisierungen Anwendung. Die Fokussierung von Datensätzen lässt sich in der Tabellenansicht neben der Maus auch durch die Cursor-Tasten, die Bild-Auf/Ab- und Anfang-/Ende-Tasten der Tastatur durchführen. Datensätze lassen sich zudem mit der Leertaste selektieren. Neben der schrittweisen Navigation lassen sich Datensätze auch direkt durch die Eingabe des Titels anspringen, ein Feature, das konzeptuell vom Windows-Explorer übernommen wurde. Der eingegebene Titel wird automatisch in der Tabelle fokussiert, und nach einer Pause von ca. einer Sekunde wird der Zeichen-Puffer der Eingabe geleert, wodurch eine neue Titeleingabe ermöglicht wird. Dieselbe Funktionalität findet sich in der Media Grid, und das Aufklappen von Zellen in der Media Grid wurde dort durch die Cursor-Tasten links / rechts realisiert.

Kapitel 6: Implementierung

Der MEDIOVIS-Architektur liegen einige klassische Konzepte zugrunde, die zum Teil eng mit Erkenntnissen aus der Informationsvisualisierung zusammenhängen. Das Kapitel stellt diese Konzepte einleitend vor und erläutert anschließend exemplarisch projektspezifische Implementierungen, die für eine effiziente Interaktion mit den Visualisierungen nötig waren.

6.1 Model-View-Controller Konzept

Das schon in Kapitel 3.3 angesprochene *Model-View-Controller Konzept* (*MVC*) [GA83] wurde in MEDIOVIS konsequent durchgehalten. Das MVC-Konzept teilt die Programmarchitektur in drei Bereiche auf: das Modell (auch Datenmodell genannt) ist für die effiziente Verwaltung der Programmdaten zuständig, der View visualisiert die Daten, und der Controller reagiert auf Benutzereingaben. Da Visualisierung und Interaktion im Sinne der direkten Manipulation eng miteinander zusammenhängen, wurde deren Trennung später aufgelöst.

Grundlegend ist jedoch die strikte Separation des Datenmodells von ihrer visuellen Repräsentation. Das Datenmodell ist auf eine effiziente Datenverwaltung hin optimiert und verfügt über keine Referenzen nach außen, sondern stellt lediglich klar definierte Anfragemethoden für die Views zur Verfügung. Visualisierungsspezifische Daten (Schrift, Icon-Größe etc.) oder das Datenmodell und die Views können somit unabhängig voneinander betrachtet werden, da Veränderungen der Datenstrukturen keinen Einfluss auf die Visualisierungen haben und umgekehrt. Ein Programmierer muss durch die klare Strukturierung zudem nicht das ganze Projekt kennen, um das System weiterzuentwickeln.

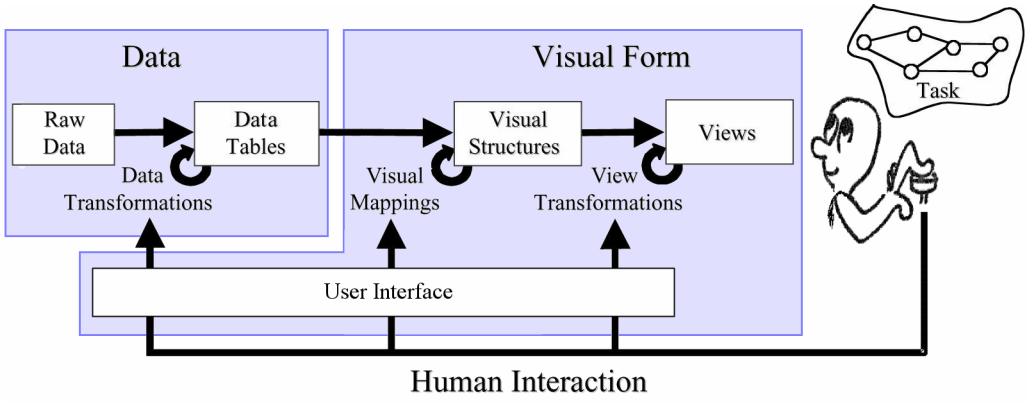


Abb. 37: modifiziertes Referenz-Visualisierungsmodell

Das MVC-Konzept lässt sich mit dem schon vorgestellten Referenz-Visualisierungsmodell in Einklang bringen, welches hier in leicht abgewandelter Form zu sehen ist (Abb. 37). Das Datenmodell organisiert die Rohdaten und Datentabellen, während die Views sich um die Darstellung kümmern und die visualisierungsspezifischen Strukturen enthalten. Der Benutzer kann in allen Programmphasen mit dem Datenmodell und den Views über die Benutzeroberfläche interagieren. Die Interaktion zwischen dem Benutzer und den Views ist eng aneinander gekoppelt, und das Einlesen neuer Daten geschieht über das Datenmodell.

6.2 Datenmodell

Die weiteren Ausführungen basieren auf dem Datenmodell-Schema von Abb. 38:

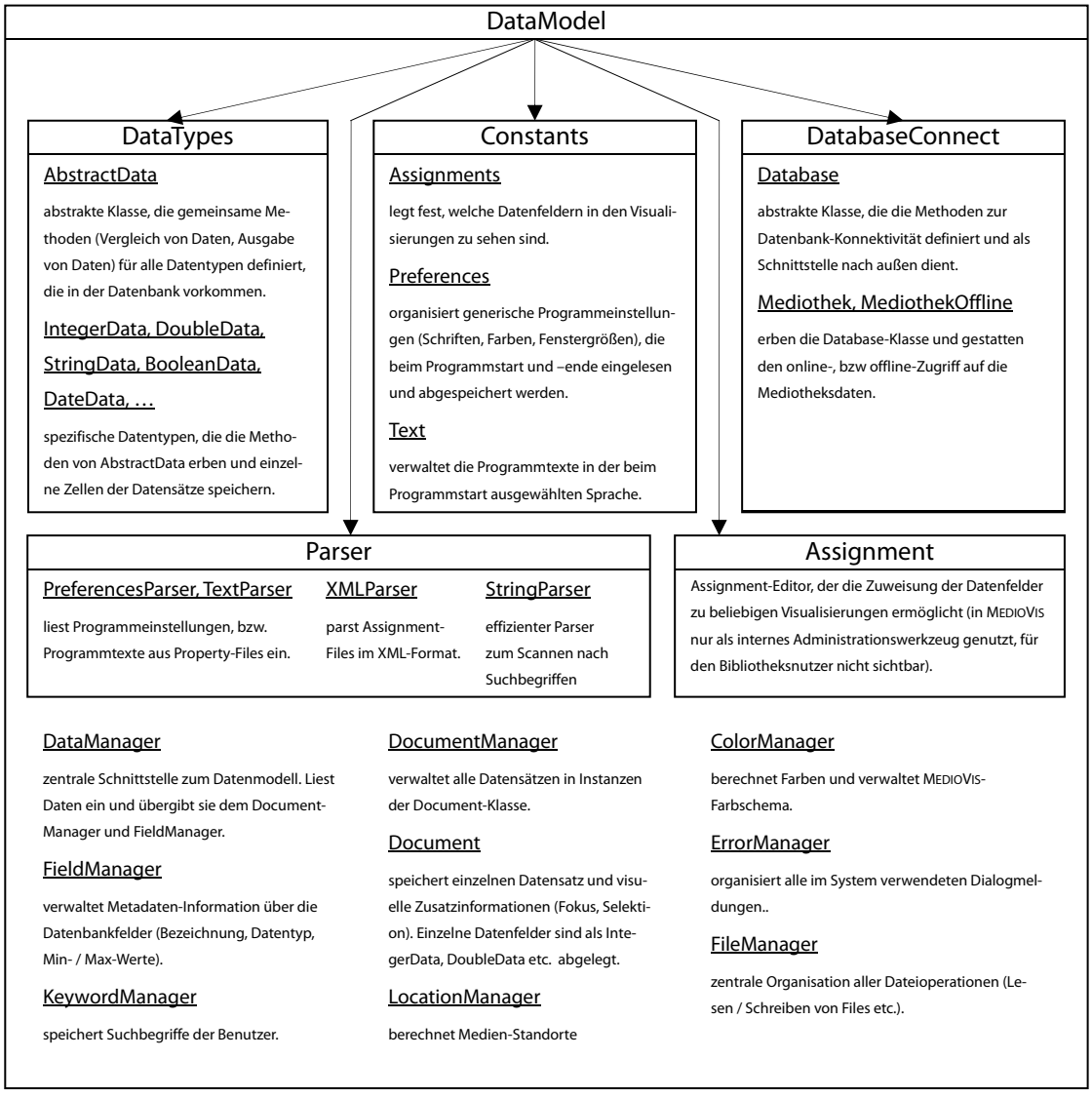


Abb. 38: Schematischer Aufbau des MEDIOVIS-Datenmodells

Nach dem Start des Programms und der Auswahl einer Sprache werden der `PreferencesParser` und der `TextParser` aufgefordert, die Programmeinstellungen und die Programmtexte einzulesen. Anschließend wird die zentrale `DataManager`-Klasse aufgerufen, um die Datenbankverbindung herzustellen und die Assignments[24] und Datenfelder einzulesen. Dazu wird eine Instanz der `Database`-Klasse im `DatabaseConnect`-Package erstellt. Je nachdem, ob die Daten online über eine PostgreSQL-Datenbank oder offline über ein Textfile eingelesen werden, wird die Klasse `Mediothek`, bzw. `Mediothek0ffline` instanziiert. Anschließend werden die im XML-Format abgelegten Assignments eingelesen und der `Assignments`-Klasse zugewiesen. Die Datenfelder, die in den Visualisierungen verwendet werden, werden dem `FieldManager` bekannt gegeben. Nachdem die Initialisierung abgeschlossen ist, erscheint das Hauptfenster von MEDIOVIS. Wenn der Benutzer eine Suche startet, wird wieder der `DataManager` aufgerufen, der die Suchbegriffe im `KeywordManager` speichert und die Daten über die Datenbankklasse einliest. Die eingelesenen Datensätze werden an den `DocumentManager` weitergereicht, der alle Datensätze verwaltet und für jeden Datensatz ein `Document` erstellt. Die `Document`-Klasse erstellt wiederum für jedes Datenfeld der Zelle eine passende Klasse aus dem `DataTypes`-Package. Abb. 39 veranschaulicht die Speicherung der Daten in Anlehnung an die Datenbanktabelle aus Abb. 3 (Seite 17).

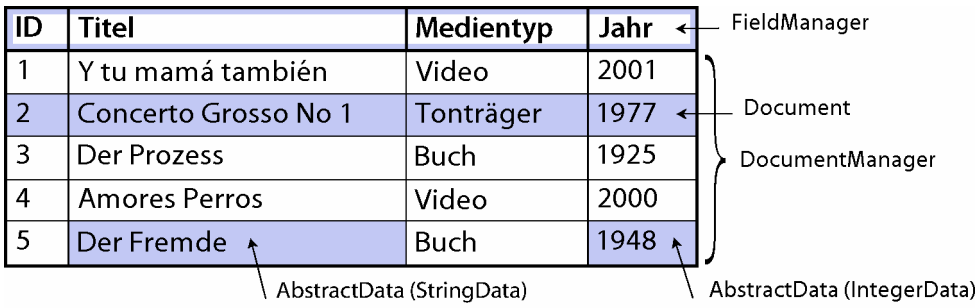


Abb. 39: Datenbanktabelle im Datenformat von MEDIOVIS

6.2.1 Zweistufige Suche

Wie in Kapitel 2.4 schon angedeutet wurde, besteht die Suche nach den relevanten Datensätzen aus zwei Schritt. Im ersten Schritt wird bei der Benutzung der Online-Datenbank eine einfache,

[24] Als Assignments werden in dieser Arbeit die Verknüpfungen zwischen den Datenfeldern und Visualisierungen bezeichnet. Sie legen fest, welche Daten in den jeweiligen Visualisierungen dargestellt werden.

schnelle SQL-Anfrage an die Datenbank gesendet, die alle Treffer zurückliefert, in denen die gewünschten Suchbegriffe an einer beliebigen Stelle vorkommen. Alle Suchbegriffe werden standardmäßig mit der SQL-Trunkierung ausgestattet. Eine mögliche Suche über das einfache Suchformular kann wie folgt aussehen:

```
Select TITLE, PERSONS [, ...] FROM DATABASE WHERE
   (TITLE ILIKE '%Bergman%' OR PERSONS ILIKE '%Bergman%' [OR ...]) AND
   (TITLE ILIKE '%Casablanca%' OR PERSONS ILIKE '%Casablanca%' [OR ...])
```

Alle relevanten Datenbankfelder (Titel, Personen etc.) werden in der einfachen Suche nach den eingegebenen Wörtern durchsucht. Durch die Verwendung des `ILIKE`-Operators wird Groß- und Kleinschreibung ignoriert, und die Prozentzeichen garantieren, dass der Suchbegriff an einer beliebigen Stelle im Datenfeld als ganzes oder Teilwort auftreten kann.

Eine Suche wie die obige liefert jedoch neben Treffern wie »Bergman« auch »Bergmann« zurück. Daher werden alle Datensätze im zweiten Schritt unter Berücksichtigung erweiterter Suchparameter vom `StringParser` untersucht und entweder verworfen oder als relevant erkannt. Die Berechnungszeit wird vom Benutzer kaum wahrgenommen, da der `StringParser` zum schnellen Parsen von Suchanfragen optimiert wurde.

In der Offline-Variante von MEDIOVIS wird von der Annahme ausgegangen, dass alle Daten relevant sind. Alle Datensätze werden daher komplett sequentiell eingelesen und vom `StringParser` auf ihre Relevanz hin untersucht. Dies führt selbstverständlich zu einer längeren Wartezeit für den Benutzer[25]. Da die Online-Variante als Standard benutzt werden sollte, wurden keine weiteren Optimierungen vorgenommen.

Alternativ zur zweistufigen Suchstrategie haben sich zwei weitere Vorgehensweisen zur Umsetzung angeboten, die jedoch für den konkreten Anwendungsbereich mehr Nachteile als Vorteile aufwiesen:

- es lassen sich komplexere SQL-Befehle konstruieren, die mit Hilfe regulärer Ausdrücke gleich das gewünschte Resultat zurückliefern. Trotz zahlreicher Optimierungen, die Postgre-

[25] das Parsen der 26.000 Datensätzen dauert – unabhängig von der Treffermenge – auf einem Rechner mit 1 GHz Prozessorleistung ca. fünf Sekunden und ist somit noch eine gute Alternative zur Online-Variante, wenn eine langsame Internet-Verbindung (Modem / ISDN) besteht.

SQL als Datenbank der Wahl benutzt, werden solche Anfragen jedoch deutlich langsamer abgearbeitet als einfache trunkierte Anfragen.

- die Erstellung von Indizes auf der Datenbank ist der übliche Weg, einfache Anfragen deutlich zu optimieren[26]. Da Indizes gewöhnlich nur ganze Wörter speichern, können sie nicht oder nur eingeschränkt genutzt werden, um nach Wortteilen zu suchen.

6.3 Views

Abb. 40 visualisiert den schematischen Aufbau der MEDIOVIS-Views:

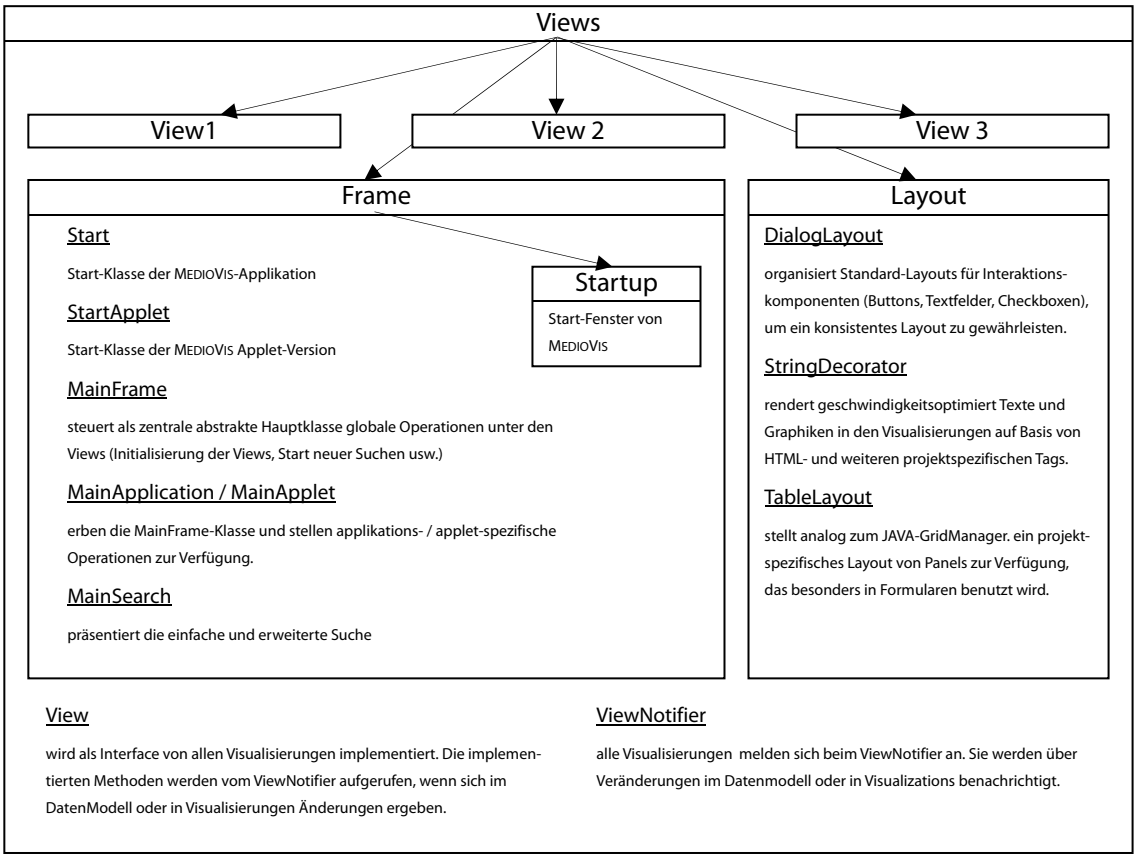


Abb. 40: Schematischer Aufbau der MEDIOVIS-Views

Die Views stellen neben dem `DataModel` den zweiten großen Programmblock dar. Neben dem `MainFrame`-Hauptfenster sind alle Visualisierungen in das `Views`-Package eingebettet. Über

[26] Datenbankindizes sind Wortlisten der Datenbankeinträge, die aufgrund spezieller Datenstrukturen viel schneller durchsucht werden können als mehrdimensionale, unsortierte Daten. Alle Index-Einträge verweisen wieder zurück auf die eigentlichen Datensätze.

die Klasse `Start/StartApplet` wird die Applikation oder Applet-Version von MEDIOVIS gestartet. Nach dem Einblenden des Startfensters über das `StartupDialog`-Package und der Initialisierung des Datenmodells über den zuvor vorgestellten `DataManager` wird `MainFrame` als Hauptfenster aufgerufen. Diese Klasse legt den Fensteraufbau fest und koordiniert alle globalen Programm-Interaktionen. Beim ersten Aufruf wird die `MainSearch`-Klasse eingebettet, die das Suchformular anzeigt. Nach der erfolgreichen Durchführung einer Suche werden alle Visualisierungen initialisiert und auf die drei Panels verteilt.

Alle Visualisierungen verfügen über eine Hauptklasse, die die einzige Schnittstelle nach außen hin darstellt. Zur besseren Übersicht wurde jede Visualisierung in einem eigenen Package eingebunden.

6.3.1 Singleton Pattern

Da in der jetzigen Implementierung von MEDIOVIS alle Visualisierungen nur einmalig vorkommen, nutzen sie das *Singleton-Pattern*, um von beliebigen Stellen im Programm angesprochen werden zu können.

Das Singleton-Pattern ist ein Konzept, das bewusst das Potenzial der Objektorientierung, mehrere Objekte einer Klasse erstellen zu können, umgeht. Ein Scatterplot, der in einer Klasse gespeichert ist, kann gewöhnlich in Form von Objekten beliebig oft erzeugt werden. Der Zugriff auf die einzelnen Objekte erfolgt über Referenzen. Wird eine Klasse jedoch nur einmalig verwendet, kann das Singleton-Pattern angewandt und die Referenz direkt in der Klasse gespeichert und über eine `get`-Methode abgefragt werden. Somit umgeht man das Problem, dass die Referenz auf das Objekt im restlichen Programm-Code weitergeleitet werden muss:

```
// Visualisierung "Scatterplot"
class Scatterplot {
  // Referenz auf einzige Klasseninstanz
  private static Scatterplot instance;

  // Konstruktur der Klasse, über die ein neues Objekt der Klasse erzeugt wird
  private Scatterplot() {}

  // Methode, die die einzige Klasseninstanz zurückliefert
  public static Scatterplot get() {
    if(instance == null) instance = new Scatterplot();
    return instance;
  }
}

// Abruf des Scatterplot-Objekts:
Scatterplot.get();
```

6.3.2 Observer Pattern

Um die in Kapitel 2.6.2 vorgestellten *Multiple Coordinated Views* umzusetzen, wurde das so genannte Observer Pattern angewandt. Alle Views implementieren das `View`-Interface, welches die Aktionen zusammenfasst, die zu Veränderungen in den anderen Views führen:

```
interface View {
  // view wird initialisiert
  public abstract void initView();
  // view wird neu gezeichnet
  public abstract void repaintView();
  // datensatz mit angegebener id wurde fokussiert
  public abstract void repaintFocus(int id);
  // datensatz mit angegebener id wurde selektiert
  public abstract void repaintSelection(int id);
}
```

Alle Views müssen sich zudem beim `ViewNotifier` über eine `attach`-Methode anmelden, um über Aktionen in anderen Views benachrichtigt zu werden:

```
// Visualisierung "Scatterplot" implementiert View-Interface
class Scatterplot extends JPanel implements View {
  // Referenz auf einzige Klasseninstanz
  private static Scatterplot instance;

  // Konstruktur der Klasse, über die ein neues Objekt der Klasse erzeugt wird
  private Scatterplot() {
    ViewNotifier.attach(this);
    // weitere Konstruktoraufrufe...
  }

  // Methode, die die einzige Klasseninstanz zurückliefert
  public static Scatterplot get() {
    if(instance == null) instance = new Scatterplot();
    return instance;
  }

  // diese Methoden werden vom ViewNotifier bei Veränderungen aufgerufen:
  // Initialisierung des Views
  public void initView() {}
  // Neuzeichnen des Views
  public void repaintView() {}
  // Focus eines Datensatzes
  public void repaintFocus(int id) {}
  // Selektion eines Datensatzes
  public void repaintSelection(int id) {}
}
```

Die `ViewNotifier`-Klasse verfügt über Methoden (`notifyInit`, `notifyFocus` etc.), die aufgerufen werden können, um Benachrichtigungen auszulösen. Diese Methoden leiten die Benachrichtigungen über die Methoden `initView`, `updateView` etc. an alle angemeldeten Views weiter.

```
class ViewNotifier {
  // View vector
  private static Vector views = new Vector();

  // neuer View wird angehängt
  public static void attach(View view) {
    views.add(view);
  }

  // Benachrichtigung für Initialisierung
  public static void notifyInit(){
    for(int i=0; i<views.size(); i++) views.get(i).initView();
  }
  // Benachrichtigung für ein Neuzeichnen des Views
  public static void notifyRepaint() {
    for(int i=0; i<views.size(); i++) views.get(i).repaintView();
  }
  // Benachrichtigung für Datensatz-Fokus
  public static void notifyFocus(int id) {
    for(int i=0; i<views.size(); i++) views.get(i).repaintFocus(id);
  }
  // Benachrichtigung für Datensatz-Selektion
  public static void notifySelection(int id) {
    for(int i=0; i<views.size(); i++) views.get(i).repaintSelection(id);
  }
}
```

Das Observer-Pattern ist ein konsistentes Konzept, um beliebige weitere Benachrichtigungen ins System einzubinden. Beispielsweise könnte eine Methode `filterView` integriert werden, die auf die Filterung von Daten hinweist.

6.3.3 Text-Rendering

In den vergangenen Projekten INVISIP / VISMEB traten große Performance-Probleme in den Visualisierungen auf. Zur Umsetzung wurden die JAVA SWING-Komponenten eingesetzt, die mächtige Funktionen bieten, um Daten anwendungsspezifisch zu visualisieren. Aufgrund ihrer Universalität arbeiten sie jedoch zum Teil sehr langsam, was besonders negativ bei der Text-Darstellung auffiel. Deshalb wurden einige Komponenten durch einfache Panels ersetzt und eine eigene Routine namens `StringDecorator` geschrieben, die die in MEDIOVIS benötigten Visualisierungsmerkmale effizient unterstützt. Neben den reinen Textdarstellungen sollte der Text-Renderer über weitere Funktionen verfügen:

- farbliche Hervorhebung von Suchbegriffen
- unterschiedliche Schriftgrößen und -formen
- Links / Verweise auf beliebige Dateien
- Anzeige von Graphiken

- flexible Anordnung von Texten durch Formatierungsangaben

Um die erweiterten Eigenschaften zur Textausgabe dynamisch festlegen zu können, bot sich die Anwendung von Tags in Anlehnung an das HTML-Format an. Die folgenden Tags werden vom `StringDecorator` unterstützt:

TAG	Bedeutung
<A>	Links (Verweise)
<B>	Fettschrift
<H1>	Überschriften
<U>	Unterstreichung von Texten
<ID>	ID eines dargestellten Datensatzes für farbliche Hervorhebungen
<KEY>	Angabe des dargestellten Datenfelds
<IND>	Einrückung von Texten
<T>	fester/dynamischer Tabulator
<IMG>	Darstellung von Bildern

Abb. 41: Auflistung der vom Text-Renderer interpretierten Tags

Der `StringDecorator` ist im Gegensatz zu den SWING-Klassen keine graphische Komponente. Er wird von den `paint-`Methoden der Panels aufgerufen und erhält bei der Initialisierung den darzustellenden Text und weitere Angaben über die Position und Größe des zu beschreibenden Bereichs mitgeliefert. Sobald die `write-`Methode vom Renderer angestoßen wird, wird der Text über den `StringParser` des Datenmodells wortweise abgearbeitet und auf den spezifizierten Bildschirmbereich gemappt. Wenn es sich bei einem Wort um einen Suchbegriff handelt, wird dieses automatisch eingefärbt. Tags werden ebenfalls erkannt und getrennt im Programm behandelt. Die Grundzüge des Programm-Codes sind nachfolgend wiedergegeben:

```
class StringDecorator {
  // Variablendefinitionen...

  // Initialisierung des Texts und der graphischen Informationen
  public void init(String text, Graphics g, int x, int y, int weight, int height) {
    // Initialisierung des StringParser
    sp = new StringParser(text);
    // Referenz auf Schrifteigenschaften
    fm = g.getFontMetrics();
    // weitere Variablenzuweisungen...
  }

  // Rendert den ganzen Text
  public boolean write() {
```

```
    while(more()) write();
  }

  // Testet, ob weiter gerendert werden soll
  public boolean more() {
    // testet, ob Text zu Ende ist; wenn ja, wird Parsing beendet
    if(! sp.more()) return false;

    // testet, ob Zeilenende erreicht ist; wenn ja, Zeilenwechsel
    if(x - startX > width) {
      x = startX;
      y += g.getFontMetrics().getHeight();
    }
    // testet, ob vertikales Ende erreicht ist; wenn ja, wird Parsing beendet
    if(y - startY > height) return false;
    // Parsing wird fortgesetzt
    return true;
  }

  // Rendert nächstes Wort
  public void next() {
    // holt nächstes Wort
    word = sp.next();

    // Sonderbehandlung von Tags
    if(sp.isTag()) {
      checkTag();
    } else {
      // gibt Wort aus
      g.drawString(word, x, y);

      // erhöht x-Position um die Wortbreite
      x += fm.stringWidth(word);
    }
  }

  // ändert Textausgabe entsprechend der Tags
  public void checkTag() {
    // alle Tags werden überprüft...
  }
}
```

Der Text-Renderer wird in allen Views eingesetzt, die textuelle Daten visualisieren. Er arbeitet deutlich effizienter als die Text-Komponenten von SWING, weil nur der Bereich des Texts gezeichnet wird, der gerade sichtbar ist, und Formatierungsangaben ebenfalls in Echtzeit bestimmt werden und somit nicht in Form weiterer Datenstrukturen mitgespeichert werden müssen. Außerdem wird Zeit und Speicher gespart, da zur Textdarstellung keine neuen, zusätzlichen Klassen angelegt werden. Stattdessen wird die schnelle Graphik-Engine von JAVA direkt angesprochen, um den Text zu rendern. Dadurch erhöht sich die Geschwindigkeit der Darstellung abhängig vom darzustellenden Text um das 20-100fache im Vergleich zur HTML-Renderer-Komponente von JAVA, wodurch beispielsweise die Echtzeit-Animation der Zellen in der *Media Grid* überhaupt erst ermöglicht wird.

6.3.4 Location View

Die Konzeption der Verknüpfung zwischen Signaturen und Medienstandorten wurde in Kapitel 5.3.5 schon ausführlich vorgestellt. Bei jeder Aktualisierung der Standort-Ansicht wird die Signatur des fokussierten Mediums abgefragt und mit den abgespeicherten Standorten verglichen. Die entsprechende Methode durchsucht dazu die Einträge der Standort-Liste und liefert eine projektspezifische Klasse `Location` zurück, die die Standortangaben zu der Signatur bereitstellt. Das einfachste Verfahren, nach dem Standort zu suchen, lässt sich über Schleifen realisieren:

```
// Standorte werden in projektspezifischer Location-Klasse gespeichert
Location[] locations;

// alle Standorte werden auf die Signatur hin verglichen
public Location getLocation(String signature) {
  // durchläuft alle Locations
  for(int len = signature.length(); len > 0; len--) {
    // verkürzt Signatur auf <len> Zeichen
    signature = signature.substring(0, len);
    // durchläuft alle Locations
    for(int i = 0; i < locations.length; i++) {
      // wenn Ergebnis 'true', dann wurde ein Standort gefunden
      if(signature.equals(locations[i].signature)) {
        return locations[i];
      }
    }
  }
  // keine Signatur gefunden
  return null;
}
```

Die innere Schleife der Methode durchläuft alle gespeicherten Standorte. Durch die äußere Schleife wird zuerst ein kompletter Signaturenvergleich durchgeführt und die Signatur anschließend um jeweils ein Zeichen verkürzt. Dadurch werden als erstes exakte und in den weiteren Durchläufen gröbere Standortzuweisungen gefunden. Die Routine verfügt bei genauerer Betrachtung noch über eine dritte Schleife, da die `equals`-Anweisung ebenfalls zwei Strings zeichenweise vergleicht.

Die drei Schleifen wären jedoch bei 25.000 Bibliothekstiteln zu langsam, um zügig mit der Standort-Visualisierung arbeiten zu können. Somit wurde ein alternativer Weg beschritten und die innere Schleife durch eine Hash-Funktion ersetzt, die lediglich etwas mehr Speicher benötigt, den Algorithmus aufgrund ihrer konstanten Laufzeit jedoch um ein Vielfaches beschleunigt und zwei Schleifen unnötig macht:

```
// Standorte werden in projektspezifischer Location-klasse gespeichert
Hashtable locations;

// alle Standorte werden auf die Signatur hin verglichen
public Location getLocation(String signature) {
  for(int len = signature.length(); len > 0; len--) {
    // holt sich Location-Klasse für angegebenen String
    Object loc = locations.get(signature.substring(0, len));
    // wenn Ergebnis nicht null, dann wurde ein Standort gefunden
    if(loc != null) {
      return (Location) loc;
    }
  }
  // keine Signatur gefunden
  return null;
}
```

Da die Hash-Funktion direkt auf abgespeicherte Signaturen zugreifen kann, ohne String-Vergleiche durchführen zu müssen, fällt auch der Zeichenvergleich über die `equals`-Anweisung weg. Die Suche mittels der Hash-Funktion ist zudem eleganter, weil sie sich in der Umsetzung auf wenige Zeilen beschränkt.

6.3.5 Scatterplot View

Bei der Umsetzung der Scatterplot-Visualisierung halfen ebenfalls die Erfahrungen aus den vergangenen Projekten des Lehrstuhls. Die folgenden Aktionen der Visualisierung hatten am meisten Einfluss auf die Geschwindigkeit:

- das Zeichnen aller Icons
- die Überprüfung, ob und welche Icons sich auf dem sichtbaren Bildschirmbereich überdecken (*Multiple Data Points*, s. Kapitel 5.3.4)
- die Reaktion auf Mausbewegungen (Fokussierung der richtigen Datenpunkte)
- das Zoomen der Visualisierung (Neuberechnung der Icon-Positionen etc.)

6.3.5.1 Zeichnen der Icons

Der VISMEB-Scatterplot basierte ebenso wie die textuellen Visualisierungen auf SWING-Komponenten. Alle Punkte wurden als eigene Klassen definiert (`JComponent`) und konnten somit relativ einfach verwaltet und mit Mausaktionen versehen werden. SWING erwies sich jedoch bei der Darstellung größerer Komponentenmengen als recht schwerfällig. Die Änderung der Achsenbelegung oder eine Größenänderung der Ansicht hatte zudem zur Folge, dass für alle Punkte wieder neue große Komponenten erzeugt werden mussten. Daher wurde im MEDIOVIS-Scatterplot auf die Verwendung graphischer Komponenten verzichtet. Die Datenstrukturen der

Icons werden in einer Klasse `PlotDots` verwaltet. Die Positionen der Icons werden in Arrays gespeichert. Zum Zeichnen der Icons müssen lediglich die Arrays durchlaufen und die Icons auf den sichtbaren Bereich gemappt werden. Weitere Informationen wie das Aussehen und der Zustand der Icons (Normalzustand, Fokus, Selektion) werden ebenfalls in Echtzeit abgefragt.

6.3.5.2 Multiple Data Points

Um Icons zu identifizieren, die sich überdecken, werden die Positionen der Icons zusätzlich in einer Hashtabelle abgelegt. Die Positionen der Icons in der Tabelle verweisen auf einen Vektor, der die IDs der Icons an der entsprechenden Position enthält. Beim Zeichnen der Icons genügt somit ein einfacher Blick in die Hash-Tabelle, um festzustellen, ob und welche Icons sich an einer Position befinden. Falls sich mehrere Icons überdecken, werden diese als Multiple Data Point in Form von Sternchen gezeichnet.

6.3.5.3 Mausbewegungen

Mausbewegungen konnten im VisMeB-Scatterplot sehr einfach durch die Maus-Ereignisse (Mouse Events) von JAVA abgefragt werden. Jeder Datenpunkt besaß Methoden, die auf Mausbewegungen reagierten. Sobald ein Punkt mit der Maus überfahren wurde, wurde die `mouseEntered`-Methode des Punkts aufgerufen, die die anderen Visualisierungen über die Neu-Fokussierung informierte. Die Benachrichtigung über JAVA hatte jedoch zum einen den Nachteil, dass sie relativ langsam auf Mausbewegungen reagierte, zum anderen konnten Punkte, die von anderen Punkten überdeckt wurden, nur schlecht oder gar nicht fokussiert werden.

In der MEDIOVIS-Implementierung werden bei jeder Mausbewegung alle Icon-Positionen, die in der Hash-Tabelle gespeichert sind, durchlaufen und mit der aktuellen Mausposition verglichen. Wenn sich ein Icon in der Nähe des Maus-Cursors befindet, werden der entsprechende Datenpunkt und sein Abstand zum Cursor zwischengespeichert. Der Abstand wird vereinfacht durch die Multiplikation der X- und Y-Koordinaten berechnet. Wird beim Parsen der Positionsdaten ein Icon gefunden, das sich näher am Cursor befindet, werden der neue Punkt und der Minimalabstand zwischengespeichert. Nachdem alle Icon-Positionen überprüft worden sind, bleibt der nächstliegende Punkt als Ergebnis übrig. Dadurch wird gewährleistet, dass jeweils der Datenpunkt fokussiert wird, der dem Maus-Cursor tatsächlich am nächsten ist.

Das Parsen aller Icon-Positionen könnte beispielsweise durch eine sortierte Speicherung der X- und Y-Positionen noch optimiert werden, in den Tests hat sich jedoch herausgestellt, dass die lineare Suche durch alle Icon-Positionen immer noch um ein Vielfaches schneller ist als das

Zeichnen der Punkte. Der folgende Code-Abschnitt veranschaulicht die Suche nach dem nächsten Icon:

```
// x/y: aktuelle Mausposition

// anfänglicher Entfernungswert = Maximum
int dist = Integer.MAX_VALUE;
// nächstliegender Datenpunkt
Vector dots = null;

// Datenpunkte als Hashtable
Hashtable hash = dots.getDots();
// Durchlauf aller Datenpunkte
Enumeration en = hash.keys();
while(en.hasMoreElements()) {
  // Nächstes Element, Abstand zum Cursor
  Pos pos = (Pos) en.nextElement();
  int distX = Math.abs(pos.x - x);
  int distY = Math.abs(pos.y - y);
  // Nah genug und näher als vorheriger Punkt?
  if(distX < 7 && distY < 7 && dist > distX * distY) {
    // Speichere Punkt und neuen Minimalabstand
    dots = (Vector) hash.get(pos);
    dist = distX * distY;
  }
}
```

6.3.5.4 Zooming

Beim Zoomen des Scatterplots bleiben die intern gespeicherten Koordinaten der Icons unangetastet. Der gezoomte Bereich stellt lediglich eine Ansicht auf die Gesamtsicht dar; durch einen Zoom-Schritt wird nur die Position und Größe des jeweils sichtbaren Ausschnitts neu berechnet. Jeder zu zeichnende Punkt wird nach der absoluten Positionsbestimmung mit den Koordinaten des Zoom-Bereichs verrechnet.

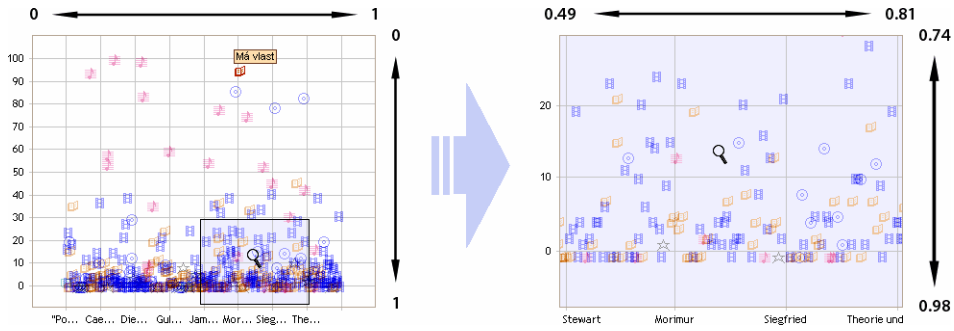


Abb. 42: Standardansicht und gezoomte Ansicht des Scatterplots mit normalisierten Achsenwerten

Die horizontalen und vertikalen Start- und Endpositionen des Sichtfensters (des sichtbaren Bereichs) sind in der Normalansicht auf 0 und 1 normalisiert (s. Abb. 42). Sobald der Zoom-Vorgang angestoßen wird, werden die Werte des Sichtfensters neu berechnet. Je weiter in das

Sichtfeld hineingezoomt wird, desto langsamer nähern sich Start- und Endpositionen an, wodurch sich der Eindruck einer gleichmäßigen Zoom-Geschwindigkeit ergibt.

```
// dir: zoom-richtung (1/-1: zoom in/out)

// normalisierte Position des Maus-Cursors
double posX = e.getX() / (double) getWidth();
double posY = e.getY() / (double) getHeight();
// modifizierte Position, um Zooming im Randbereich zu verbessern
posX = Math.max(0, Math.min(1, (Math.pow(posX * 2, 1.5) / 2)));
posY = Math.max(0, Math.min(1, (Math.pow(posY * 2, 1.5) / 2)));
// Start- und Endpositionen des Sichtfensters
double startX = xaxis.getStartPos();
double endX   = xaxis.getEndPos();
double startY = yaxis.getStartPos();
double endY   = yaxis.getEndPos();

// neue Zoomgeschwindigkeit (abhängig von Sichtfensterbreite)
zoomStep = dir * (endX - startX) * 0.02;
// Berechnung der neuen Start- und Endposition
if(endX - startX > zoomStep * 2 && endY - startY > zoomStep * 2) {
  // Berechnung der Sichtfensterbreite, begrenzt auf 0 bis 1
  double step1 = zoomStep * posX;
  double step2 = zoomStep * (1 - posX);
  startX = Math.max(0, startX + step1 + (-step2 > 1 - endX ? step2 : 0));
  endX   = Math.min(1, endX   - step2 - (-step1 > startX   ? step1 : 0));
  xaxis.setStartEndPos(startX, endX);

  // Berechnung der Sichtfensterhöhe (analoge Berechnung)
  // ...
}
```

Wie im Programm-Code zu sehen ist, ist das Zoomen ist abhängig von der Position des Maus-Cursors. Wenn sich der Cursor in der Mitte des Sichtfelds befindet, werden alle umliegenden Icons gleichmäßig in alle Richtungen gezoomt. Ein Cursor am Bildschirmrand führt dazu, dass alle Icons in die entgegengesetzte Richtung verschwinden. Dazu wird die normalisierte Mausposition durch eine leichte Potenzierung mit dem Wert 1.5 etwas verzerrt, wodurch das Zoomen von Icons in den Eckbereichen erleichtert wird.

6.3.6 MediaGrid View

Die umfassende Beschreibung der Implementierung der *MediaGrid* würde den Rahmen dieser Arbeit bei weitem sprengen, deshalb soll hier besonders auf die Features der Implementierung eingegangen werden, die zur ruckellosen Darstellung der Visualisierung notwendig waren.

Der Vorteil des Verzichts auf die SWING-Komponenten wurde in den vergangenen Kapiteln schon ausführlich erörtert. Als Basis für die MediaGrid hätte die Tabelle dienen können, die in der tabellarischen Ansicht integriert wurde. Da die `JTable`-Klasse von SWING jedoch ohnehin nicht die Flexibilität bot, um alle Anforderungen an die Visualisierung umzusetzen, wurde die

Tabelle komplett von Grund auf neu implementiert. Die Zeichenroutine der MediaGrid zeichnet sich durch die folgenden Merkmale aus:

- der SWING-Tabelle liegt eine komplexe Datenstruktur zugrunde, in der die darzustellenden Inhalte schon vor dem Zeichnen definiert werden. Für jede Zelle werden graphische Komponenten erzeugt, die während dem Zeichenvorgang auf dem Bildschirm platziert werden. Der Speicherverbrauch steigt entsprechend an, wenn große Datenmengen in einer solchen Tabelle untergebracht werden. In der MediaGrid werden die Zellinhalte in Echtzeit bestimmt und dynamisch auf den sichtbaren Bereich gemappt.
- da sich die Zellgrößen durch die direkte Interaktion ständig verändern, müssen durch das direkte Zeichnen der Zellen keine neuen Komponenten berechnet werden. Dadurch wird ein ruckelfreies Zoomen der Zellen gewährleistet.
- die Zellrechtecke werden in der Standard-Implementierung der Tabelle für jede Zelle extra gezeichnet. In der MediaGrid wird das Zellraster erst nach dem Rendern aller Daten über die Tabelle gelegt. Dadurch beschränkt sich das Zeichnen des Rasters auf ein paar wenige Zeichenbefehle.

Eine weitere wichtige Optimierung stellte die Implementierung eines eigenen Scrollpanes dar. Scrollpanes werden dann eingesetzt, wenn ein darzustellendes Panel nicht komplett auf dem Bildschirm untergebracht werden kann. Durch das Bewegen der Scroll-Leiste kann der Benutzer den Bereich des Panels, der am Bildschirm dargestellt werden soll, auswählen. Bevor das Scrollpane genutzt werden kann, muss jedoch erst das komplette Panel erzeugt werden, was sich bei großen Datenmengen als problematisch herausstellt: bei 25.000 Datensätzen, einer Zeilenhöhe von 20 Pixeln und einer Bildschirmbreite von ca. 1000 Pixeln verbraucht ein solches Panel mit 50 Millionen Pixeln einen immensen Speicherplatz.

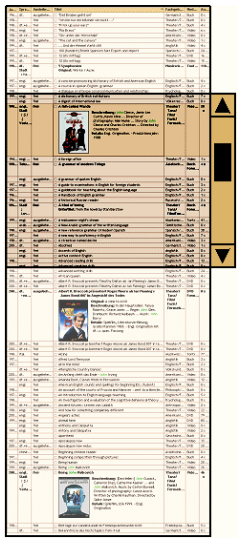


Abb. 43: Aufbau einer Scrollpane

Bei der Implementierung des visualisierungsspezifischen Scrollpanes lag es nahe, nur den Bereich des Panels zu rendern, der gerade sichtbar ist. Die Scrollpane speichert lediglich die absolute obere Position des sichtbaren Bereichs. Beim Zeichnen der Visualisierung werden die Zeilenhöhen addiert, und es wird erst mit dem Zeichnen begonnen, wenn der sichtbare Bereich erreicht ist. Die nachfolgenden Zellen werden auf

den Bildschirm gemappt, und das Rendern der Zellinhalte wird von dem in Kapitel 6.3.3 vorgestellten Text-Renderer übernommen.

Abb. 44: Zoom-Varianten der MediaGrid: a) sofortiges Aufklappen, b) lineares Aufklappen, c) Aufklappen mit Beschleunigung

Ein letztes Implementierungsdetail soll zuletzt verdeutlichen, dass kleine Veränderungen in Visualisierungen, die programmtechnisch nur wenig zusätzlichen Aufwand bedeuten, oft große Auswirkung auf den Joy of Use eines Programms haben können. Zur Umsetzung des Zoom-Algorithmus, der die Zellen der MediaGrid aufklappt, standen drei Alternativen zur Auswahl (siehe Abb. 44):

a) im einfachsten Fall kann beim Anklicken einer Zelle sofort die neue Zellgröße gesetzt und ein repaint durchgeführt werden. Diese Variante stellt die schnellste Interaktionsform dar, da die nächste Granularitätsstufe fast zeitgleich mit dem Mausklick erreicht wird. Da sich durch die Vergrößerung einer Zelle jedoch auch die anderen Zellen am Bildschirm verschieben, kann der plötzliche Wechsel auf die neue Ansicht jedoch verwirrend wirken.

b) um den Prozess des Aufklappens zu veranschaulichen, kann durch das Anklicken einer Zelle ein Animationsvorgang gestartet werden, der die Zelle schrittweise von der alten in die neue Zellgröße überführt. Der Prozess des Zoomens macht die Auswirkung der eigenen Interaktion greifbarer.

c) die schrittweise, lineare Animation hat einen verhältnismäßig technischen Charakter, da die Bewegung von Objekten aus physikalischer Sicht immer durch einen Beschleunigkeitsvorgang in Gang gesetzt wird. Daher können die linearen Animationsschritte sowohl beschleunigt als auch wieder verlangsamt werden, wodurch die Visualisierung einen viel dynamischeren Charakter erhält.

Kapitel 7: Evaluation

Im Rahmen von MEDIOVIS lassen sich die Evaluationsbestrebungen in zwei Typen aufteilen:

- Zur Konzeption des Systems wurden persönliche Benutzerbefragungen durchgeführt und Benutzerprofile erstellt, die in Kapitel 4.4.1 schon ausgeführt wurden. Die Implementierung des Systems war zudem von Benutzertests begleitet, die eine intuitive und einfache Benutzung des Systems gewährleisten sollten. Ergebnisse dieser Tests werden in Kapitel 7.1 vorgestellt.
- Neben den persönlichen Benutzerbefragungen ermöglicht die in MEDIOVIS implementierte DROID-Technologie, die die Interaktionen der Anwender automatisch in einer Datenbank mitloggt, automatische Auswertungen des Benutzerverhaltens.

Die Ausführungen dieses Kapitels fallen relativ kurz aus, da bis zum jetzigen Zeitpunkt keine umfangreiche summativen Evaluationen des Systems durchgeführt worden ist. Aus den kontinuierlichen Tests konnten viele hilfreiche Erkenntnisse gewonnen werden, die jedoch keine abschließenden Aussagen über die Qualität von MEDIOVIS ermöglichen.

7.1 Benutzertests

Zur Durchführung der insgesamt 24 Benutzertests wurden jeweils 20-30 Minuten anberaumt. Die Tests wurden durchgeführt, um Benutzungsprobleme mit dem System zu diagnostizieren und weitere Anregungen, wie das System erweitert werden könnte, aufzunehmen. Einige der Ideen wurden auch unmittelbar umgesetzt. Da die Tests über den ganzen Zeitraum des Projekts durchgeführt wurden und die Benutzer das System in unterschiedlichen Entwicklungsstadien kennen lernten, wurden sinnvollerweise nicht alle Test-Ergebnisse quantitativ ausgewertet.

Die Probanden hatten alle gemeinsam, dass sie MEDIOVIS zuvor noch nie benutzt hatten; in den meisten Fällen hatten sie jedoch schon Erfahrungen mit dem KOALA-Katalog gesammelt. Die Computer-Kenntnisse der Benutzer waren heterogen, beschränkten sich jedoch in den meisten Fällen auf die Grundlagen zur Nutzung des Internets und Textverarbeitungen. Da die meisten Erkenntnisse über die Beobachtung der Testnutzer zu erwarten waren, erhielten diese während der Tests keine Hilfestellungen, wurden jedoch gebeten, ihre Gedanken laut auszusprechen.

Um das Szenario realer Bibliotheksbesuche zu simulieren und die Ergebnisse der Tests nicht unnötig einzuschränken, erhielten die Probanden keine Aufgaben, sondern wurden lediglich

gebeten, nach Medien zu recherchieren, die sie in der Mediothek vorzufinden hofften. Zum Teil lieferten schon wenige Tests wichtige Erkenntnisse, die im weiteren Testverlauf bestätigt wurden:

- die Probanden stellten wider Erwarten sehr heterogene Suchanfragen. Die nachfolgend aufgelisteten Ergebnisse treffen keine Unterscheidung zwischen den Teilnehmern der Tests, die zum Teil mehrere Suchen durchführten, sondern beziehen sich auf die Gesamtheit der durchgeführten Suchen. Zu ca. 72% waren Filme das Ziel der Suche, zu 18% wurden Tonträger gesucht, und die restlichen 10% fielen auf die Suche nach Büchern. Während zu 35% explizit nach Filmtiteln und zu 31% nach Namen von Schauspielern, Regisseuren, Komponisten oder Musikern gesucht wurde, wurden zu 22% Schlagwörter wie »DVD«, »Tierfilme« oder »Amerika« eingegeben. Die restlichen 12% bezogen sich auf die Nutzung der erweiterten Suche zur Auswahl bestimmter Kategorien, z. B. »Video« als Medientyp und »russisch« als Sprache. Ungefähr der Hälfte der Probanden stand nur die einfache Suche zur Verfügung, weil die erweiterte Suche noch nicht implementiert war; einige der Probanden wünschten sich explizit eine erweiterte Suche. Da die Durchführung der Suchen in MEDIOVIS auf denselben Interaktionsprinzipien basiert wie die Suche im Internet und im KOALA-Katalog, traten bei keinem der Probanden Probleme auf.
- mit der Darstellung der Treffer in der Tabellenansicht und der einzelnen Titelansicht kamen erfreulicherweise alle Probanden gut zurecht. Einzig die sofortige Fokussierung der Datensätze durch das bloße Überfahren der Titel erschien zwei der Probanden als zu interaktiv, während die überwiegende Mehrzahl der Nutzer dieses Feature besonders begrüßte. Die Navigation in der Tabelle über die Tastatur war ursprünglich nicht geplant, wurde jedoch doch noch implementiert, nachdem fünf der Nutzer dieses Feature als wünschenswert bezeichneten.
- die Anzeige der ausgewählten Titel wurde von den Probanden ebenfalls positiv aufgenommen. Da die ersten Programmversionen jedoch außer der Darstellung der Titel keine weiteren Interaktionen zuließen, wurden möglichst bald die schon geplanten Features zum Versenden, Abspeichern und Ausdrucken der Titelliste integriert. Einer der Probanden äußerte die sinnvolle Anregung, dass ausgewählte Titel durch eine neue Suche nicht gelöscht werden sollten. Dies ist auch in der momentanen Programmversion nicht anders gelöst, weil bisher keine konzeptuell konsistente Lösung gefunden werden konnte, schon ausgewählte Titel weiterhin in neuen Suchen beizubehalten.

- die Standort-Ansicht fand besonderen Zuspruch und konnte mit Hilfe der Tests sowohl aus ästhetischer wie auch aus funktioneller Sicht weiter verbessert werden. Besonders die visuelle Hervorhebung der Standorte wurde einigen Testphasen unterzogen, bis sie als selbsterklärend und attraktiv empfunden wurde.
- die graphische Ansicht rief sehr unterschiedliche Reaktionen hervor. Wenige Benutzer verstanden sofort, wie ihnen der Scatterplot hilfreich sein könnte; die Visualisierung wurde oft als zu komplizierte Darstellung oder Spielerei abgetan. Erwartungsgemäß kamen Studenten in höheren Semestern und Studenten mit naturwissenschaftlichen Fächern besser mit der Darstellung zurecht. Wirkliches Interesse an der Visualisierung zeigten nur fünf der Probanden, und drei der Testnutzer betrachteten den Scatterplot als tatsächliche Alternative zur tabellarischen Darstellung. Um die Akzeptanz des Scatterplots zu testen, müsste somit über längere Zeit beobachtet werden, ob die Benutzer der Visualisierung Mehrwerte abgewinnen können. Positiv hervorzuheben ist die Tatsache, dass die meisten Teilnehmer die Interaktion mit dem Scatterplot und dessen Darstellung als ansprechend empfanden. Die Tests brachten zudem einige Ideen zu Tage, die bei der Weiterentwicklung des Scatterplots berücksichtigt wurden; die Kreise im Scatterplot wurden beispielsweise durch farbige Icons abgelöst. Die Legende wurde integriert, um die benutzten Icons zu erklären und die Funktion des Cursors (Hand / Lupe) zu veranschaulichen usw.
- einige Probanden vermissten die Anzeige des Ausleihstatus der Medien und die Möglichkeit, selektierte Titel vormerken zu können. Da MEDIOVIS jedoch vorerst noch einen Abzug der Originaldaten nutzt und keinen Direktzugriff auf die Daten der Bibliotheks-Datenbank hat, konnten diese Angaben, die zweifellos für einen Bibliotheks-Browser mit vollem Funktionsumfang notwendig wären, bisher nicht integriert werden.

Weitere Tests mit Probanden, die das System zum wiederholten Mal benutzten, sollen an dieser Stelle nicht weiter aufgeführt werden, da die daraus gewonnenen Erkenntnisse nicht schriftlich festgehalten wurden. Als Resumée der Benutzertests bleibt festzuhalten, dass die kontinuierliche Evaluation des Systems – sowohl im offiziellen Rahmen als auch durch informelle Tests – der Entwicklung eines benutzerzentrierten Systems sehr dienlich war und auch einen großen Motivationsfaktor zur Weiterentwicklung von MEDIOVIS dargestellt hat.

7.2 DROID

Die Technologie von DROID ermöglicht zahlreiche Aussagen über die Interaktionen der Benutzer und kann somit zum Beispiel ersichtlich machen, welche Visualisierungen am liebsten benutzt werden, oder auch eventuelle Probleme aufdecken, die beim Arbeiten mit dem System auftreten [JH03]. Während der Benutzer mit MEDIOVIS arbeitet, werden Interaktionen wie die Eingabe von Suchbegriffen, das Drücken von Buttons, die Auswahl von Visualisierungen und Fokuswechsel automatisch im Hintergrund in einer Datenbank gespeichert, ohne dass der Benutzer in seiner Arbeit mit dem System eingeschränkt wird. Die Daten werden anonym erfasst, da im System ohnehin keine Benutzerauthentifizierung existiert.

Beim Beenden von MEDIOVIS wird der Benutzer dazu aufgerufen, einen Fragebogen auszufüllen. Der Fragebogen besteht aus zehn Multiple-Choice Fragen und einem Textfeld, in dem persönliche Kommentare zum System gemacht werden können.

Eine Analyse der umfangreichen Daten, die innerhalb der letzten zwei Monate mit DROID gesammelt werden konnten, würde den Rahmen dieser Arbeit bei weitem sprengen, deshalb soll nur kurz auf einige beispielhafte Aspekte eingegangen werden, unter denen eine Analyse der Daten lohnenswert erscheint:

- Suchen, die keine Treffer zurückliefern, liefern interessante Feststellungen, die genutzt werden sollten, um die Retrieval-Komponenten des Systems zu verbessern. Durch die Analyse fehlgeschlagener Suchen lassen sich unterschiedliche Problem-Kategorien bilden. Der gewöhnliche Grund für eine fehlgeschlagene Suche ist, dass keine Bibliothekstitel existieren, die im Interesse des Benutzers stehen. Es gibt jedoch noch eine Reihe andere Gründe, warum eine Suche keine Ergebnisse liefert: Suchbegriffe können falsch geschrieben sein, sie können andere als die in der Datenbank benutzten Schreibweisen enthalten, oder sie können zu genau spezifiziert sein.
- ein weiterer interessanter Aspekt betrifft die Nutzung der angebotenen Visualisierungen. Werden bestimmte Datenansichten kaum verwendet, wird entweder der Mehrwert einer Visualisierung vom Benutzer als zu gering eingeschätzt, oder die Visualisierung wird schlichtweg übersehen und muss auffälliger im System präsentiert werden. Der Zeitpunkt, wie lange Visualisierungen genutzt werden und wann sie im Ablauf der Programmnutzung aufgerufen werden, hilft, diese zwei Fälle unterscheiden zu können.

- dieselbe Beobachtung kann auf beliebige Interaktionskomponenten übertragen werden. Wenn bestimmte Komponenten selten genutzt werden, könnten sie Schwächen aufweisen, die im Rahmen eigener Evaluationen genauer untersucht werden sollten.
- die Analyse der geloggten Daten lässt Rückschlüsse auf die Benutzer zu. So nutzen beispielsweise die meisten Anwender die Maus zur Navigation, diejenigen Anwender, die die Tastatur als Eingabegerät bevorzugen, arbeiten auch eher mit dem Scatterplot.

Im Rahmen weiterer Publikationen unserer Arbeitsgruppe werden demnächst detailliertere Analysen der DROID-Daten folgen.

Kapitel 8: Resumée

Diese Arbeit wurde geschrieben, um einen umfassenden Einblick in die Konzeption und Implementierung von MEDIOVIS zu geben. Neben theoretischen Überlegungen und den praktischen Erfahrungen der abgeschlossenen Projekte der Arbeitsgruppe MCI spielten die kontinuierlichen Benutzertests bei der Umsetzung eine große Rolle. Im MEDIOVIS-Kapitel wurden sowohl die Visualisierungen den Mediothek-Systems präsentiert als auch die innovativen Ansätze, die in der Forschungsversion Eingang gefunden haben. Im Implementierungs-Kapitel wurde die System-Architektur zusammengefasst und einige projektspezifische Algorithmen und Programmroutinen vorgestellt, die die Flexibilität der JAVA-Sprache ausnützen und entscheidend für die Umsetzung direkt-interaktiver Visualisierungen waren. Das Evaluations-Kapitel sollte einen Einblick in die projektbegleitenden, formativen Evaluationen verschaffen, die zur Gewährleistung der Benutzerfreundlichkeit des Systems durchgeführt wurden. Sie können jedoch abschließende summative Evaluationen nicht ersetzen.

Während der Awendungskontext von MEDIOVIS stark an der Bibliotheksdomäne ausgerichtet war, wurde beim Projektverlauf deutlich, dass die eingesetzten Visualisierungen ein großes Potenzial für beliebige andere Einsatzgebiete bieten. Besonders die *MediaGrid* eröffnet ganz neue Möglichkeiten, Daten zweidimensionaler und hierarchischer Art dynamisch zu visualisieren, weshalb sie von uns in naher Zukunft für weitere Anwendungsbereiche genutzt werden wird. Durch die flexible Veränderung der Zellgrößen können innerhalb der *Media Grid* neben beliebigen textuellen und multimedialen Inhalten komplette Visualisierungen eingebettet werden. Neue Systeme könnten auf die dokumentzentrierte Media Grid und den übersichtszentrierten *Scatterplot* beschränkt werden und dennoch alle Inhalte am Ort des Geschehens visualisieren, die in anderen Systemen über zahlreiche Fenster und Ansichten verteilt sind.

Ein weiteres breites Forschungsfeld eröffnet sich durch den mobilen Einsatz von MEDIOVIS. Für portable Handgeräte kann mittlerweile durch die Standardisierung der Entwicklungsumgebungen problemlos eigene Software entwickelt werden. Speziell das Auffinden von Medien könnte in einer Bibliothek durch ein mobiles Navigationssystem deutlich vereinfacht werden.

Es ist erfreulich zu sehen, dass MEDIOVIS einerseits seit seinem Einsatz Juli 2004 in der Bibliothek echte Mehrwerte für die Benutzer bietet und sich ansteigender Beliebtheit erfreut und andererseits aus wissenschaftlicher Sicht neue Perspektiven eröffnet, um in zahlreichen neuen Domänen eingesetzt zu werden. Die Entwicklung von MEDIOVIS hat erst begonnen!

Anhang

Literatur- & Quellenverzeichnis

[ACM92] ACM 1992
ACM SIGCHI: Curricula for Human Computer Interaction
URL: http://www.acm.org/sigchi/cdg/cdg1.html

[AWS92] AHLBERG, CHRISTOPHER / WILLIAMSON, CHRISTOPHER / SHNEIDERMAN, BEN 1992
Dynamic queries for information exploration: An implementation and evaluation, Proc. ACM CHI'92: Human Factors in Computing Systems, 619-626

[BC02] BEDERSON, BENJAMIN B. / CZERWINSKI, MARY P. / ROBERTSON, GEORGE R. 2002
A Fisheye Calendar Interface for PDAs: Providing Overviews for Small Displays. HCIL Tech Report #HCIL-2002-09

[BP94] BAUMGARTNER, PETER / PAYER, SABINE 1994
Lernen mit Software. Österreichischer Studien Verlag, Innsbruck

[BSP93] BIER, ERIC A. / STONE, MAUREEN C. / PIER, KEN ET AL. 1993
Toolglass and Magic Lenses: The See-Through Interface; Proceedings of 1993 ACM SIGGRAPH Conference, 73-80 ML.

[CMS99] CUARD, STUART / MACKINLAY, JOCK / SHNEIDERMAN, BEN 1999
Readings in Information Visualization. Morgan Kaufmann Publishers, San Fracisco

[CW94] CLEVELAND, WILLIAM S. 1994
The Elements of Graphing Data (Revised ed.). Summit, New Jersey, USA: Hobart Press.

[DR99] DÄßLER, ROLF 1999
Informationsvisualisierung: Stand,Kritik,Perspektiven. Projektgruppe InfoVis, FH Potsdam
URL: http://fabdp.fh-potsdam.de/daessler/paper/InfoVis99.pdf

[EF02] ENGSTER, FLORIAN 2002
Evaluierung des WWW-Angebotes der Bibliothek der Universität Konstanz
URL: http://www.ub.uni-konstanz.de/kops/volltexte/2002/754/

[FS95] FISHKIN, KEN / STONE, MAUREEN C. 1995
Enhanced Dynamic Queries via Movable Filters. CHI'95 Conference Proceedings, 415-420. New York, NY: ACM Press. ML

[FD04] FRIENDLY, MICHAEL / DENIS, DANIEL J. 2004
Milestones in the History of Thematic Cartography, Statistical Graphics, and Data Visualization.
URL: http://www.math.yorku.ca/SCS/Gallery/milestone/milestone.pdf

[FG81] FURNAS, GEORGE W. 1981
The Fisheye View: A New Look at Structured Files . Tech. Rep. Technical Memorandum 81-11221-9, Bell Labs.

[GA83] GOLDBERG, ADELE 1983
Smalltalk-80: The Interactive Programming Environment. Addison-Wesley Publishers, Menio Park

[GE92] GLASERSFELD, ERNST VON 1992
Konstruktion der Wirklichkeit und des Begriffs der Objektivität. Carl Friedrich von Siemens Stiftung, München

[GF02] GUNDELSWEILER, FREDRIK 2002
INVISIP – Implementation eines Scatterplots zur Visualisierung von geo-räumlichen Meta-Daten. Bachelor-Arbeit, Universität Konstanz, Fachbereich Informatik und Informationswissenschaft
URL: http://www.ub.uni-konstanz.de/kops/volltexte/2002/909/

[ISO 9241-11] ISO/DIS 9241-11.2 1997
Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability.

[JH03] JETTER, HANS-CHRISTIAN 2003
Usability-Evaluation im Rahmen von INVISIP. Bachelor-Arbeit, Universität Konstanz, Fachbereich Informatik und Informationswissenschaft

[KK98] KELLY, KEVIN 1998
New Rules for the New Economy: 10 Radical Strategies for a Connected World. NY: Viking Penguin

[KW03] KÖNIG, WERNER 2003
Konzeption und Implementation eines 3D-Scatterplots zur Visualisierung von Metadaten. Bachelor-Arbeit, Universität Konstanz, Fachbereich Informatik und Informationswissenschaft

[MD99] MAYHEW, D. J. 1999
The Usability Engineering Lifecycle. A Practicioner's Handbook for User Interface Design. Morgan Kaufmann, San Francisco, CA

[MRM00] MUßLER, GABRIELA; REITERER, HARALD; MANN, THOMAS; HANDSCHUH, S. 2000
INSYDER – Information Retrieval Aspects of a Business Intelligence System. In: Proceedings of the 7. Internationales Symposium für Informationswissenschaft, UKV Universitätsverlag Konstanz

[MT01] MANN, M. THOMAS 2001
Visualization of Search Results from the World Wide Web. Dissertation, Universität Konstanz, Fachbereich Informatik und Informationswissenschaft
URL: http://www.ub.uni-konstanz.de/kops/volltexte/2002/751/

[MT02] MEMMEL, THOMAS 2001
INVISIP – Implementation einer tabellenbasierten Visualisierung für geo-räumliche Metadaten. Bachelor-Arbeit, Universität Konstanz, Fachbereich Informatik und Informationswissenschaft
URL: http://www.ub.uni-konstanz.de/kops/volltexte/2002/892/

[NO97] NORTH, CHRISTOPHER L. / SHNEIDERMAN, B. 1997
A Taxonomy of Multiple-Window Coordination. University of Maryland Computer Science Dept. Technical Report #CS-TR-3854

[NO00] NORTH, CHRISTOPHER L. 2000
A User Interface for coordinating visualizations based on a relational schemata: Snap-Together Visualization. Dissertation, University of Maryland

[RC94] RAO, RAMANA / CARD, STUART K 1994
The Table Lens. Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In: Adelson, B.; Dumais, S.; Olson, J. S. (Eds.): CHI 1994: Conference Proceedings Human Factors in Computing Systems. Conference: Boston, MA

[RH98] RÜGER, H. MICHAEL 1998
Zoom-Techniken zur Benutzerunterstützung. Dissertation, Otto-von-Guericke-Universität Magdeburg

[SB83] SHNEIDERMAN, BEN 1983
Direct manipulation: A step beyond programming languages. IEEE Computer, August, pp. 57-69

[SB87] SHNEIDERMAN, BEN 1987
Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley Publishing Company, 1987.

[TS02] TANIN, E.; SHNEIDERMAN, B. 2002
Browsing Large Online Data Tables Using Generalized Query Previews. Online-Draft.
URL: ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/2001-13html/2001-13.pdf

Alle URLs wurden zuletzt am 25. Oktober 2004 auf ihre Gültigkeit hin überprüft.

Abbildungsverzeichnis

Sachindex