

**MedioVis Framework:  
Dokumentation und Redesign des Datenmodells zur  
Unterstützung von hierarchischen Strukturen  
komplexer Informationsräume**

Werner A. König

AG Mensch-Computer Interaktion  
Universität Konstanz

November 2005

# Inhaltsverzeichnis

<b>1</b>	<b>Motivation</b>	<b>3</b>
1.1	MedioVis . . . . .	3
1.2	Universitätsbibliothek Konstanz . . . . .	4
1.3	Problemstellung . . . . .	4
1.4	Aufbau der Arbeit . . . . .	5
<b>2</b>	<b>MedioVis Framework</b>	<b>6</b>
2.1	Historische Entwicklung . . . . .	6
2.2	Benutzerprofil . . . . .	8
2.3	Anwendungsszenario . . . . .	8
2.4	Datenquellen . . . . .	11
2.5	Visualisierungs- und Interaktionskonzepte . . . . .	12
2.5.1	TreeTable . . . . .	13
2.5.2	Detail-Ansicht . . . . .	16
2.5.3	Standort-Ansicht . . . . .	16
2.5.4	Merkliste . . . . .	17
<b>3</b>	<b>MedioVis Datenmodell</b>	<b>19</b>
3.1	Anforderungen . . . . .	19
3.2	Gesamtstruktur . . . . .	21
3.3	Data - abstrakter Datentyp . . . . .	23
3.4	Document - Dokumentenobjekt . . . . .	23
3.5	DocManager - Datenverwaltung . . . . .	24
3.6	Unabhängigkeit des Datenmodells . . . . .	25
3.7	Serialisierung des Datenmodells . . . . .	26
3.8	Hash Identifikation . . . . .	27
3.9	Hierarchisierung des Datenmodells . . . . .	27
3.10	Sortierung . . . . .	31
3.11	Filter- und Selektionsstatus . . . . .	32
3.12	Fokussierung . . . . .	33
<b>4</b>	<b>Zusammenfassung</b>	<b>34</b>

# 1 Motivation

Wiederverwendung ist eine Schlüsseltechnologie der Menschheit, ohne die Fortschritt auf jedem Gebiet nahezu unmöglich ist [BR91].

Die Automobilindustrie praktiziert diese Erkenntnis mittels mehrfach verwendbaren Modulen und Komponenten über verschiedene Autotypen hinweg schon seit Jahren. Auch die Softwareindustrie versucht aus der Mehrfachverwendung ökonomische und qualitätstechnische Vorteile zu gewinnen. Einerseits wird eine Vielzahl von Einzelmodulen entwickelt, welche für ganz spezifische Problemstellungen standardisierte oder bewährte Lösungsansätze bieten. Andererseits offerieren so genannte Frameworks zu meist domänenspezifische Rahmenlösungen oder Anwendungsarchitekturen, welche ein höheres Abstraktionslevel durch wohl definierte Schnittstellen ermöglichen.

## 1.1 MedioVis

Das Forschungsprojekt MedioVis der Arbeitsgruppe Mensch-Computer Interaktion<sup>1</sup> der Universität Konstanz hat nicht nur zum Ziel benutzerzentrierte, visuelle Suchsysteme der nächsten Generation zu entwickeln, sondern auch ein Framework zu formen, welches den Innovationsprozess durch größtmögliche Flexibilität, in Bezug auf die visuelle und interaktionstechnische Schnittstelle optimal, unterstützt. Verschiedenste Visualisierungen und Interaktionstechniken sollen mit minimalem Aufwand in das Rahmenprogramm integriert, getestet und angepasst werden können. Ebenso soll dieser gemeinsame und definierte Aufbau bei Evaluationen konstante Rahmenbedingungen und damit eine bessere Interpretierbarkeit gewährleisten.

In Bezug auf das Framework war für das Projekt MedioVis keine komplette Neuentwicklung von Nöten, sondern es konnte auf Teilmodule der Vorgängerprojekte INSYDER [RMMH00], INVISIP [GHRM02], VisMeB [KRML03] und auf die dort gesammelten Erfahrungen zurückgegriffen werden. Weitere Generalisierungen auf der Ebene der Datenschnittstelle bezüglich Anbindung und Organisation und auf der Ebene der Visua-

---

<sup>1</sup><http://hci.uni-konstanz.de>

lisierungsintegration waren bzw. sind Entwicklungsschwerpunkte für das Framework MedioVis.<sup>2</sup>

Immanent wichtig für das Projekt MedioVis ist der Live-Einsatz der verschiedenen Entwicklungsstufen als alternatives visuelles Suchsystem in der Universitätsbibliothek Konstanz. Hierbei können parallel zur Konzeption direkt Nutzungsdaten mit Hilfe von DROID<sup>3</sup>, eines Protokollierungssystems für Interaktionsvorgänge, gesammelt werden. Die Live-Daten und regelmäßige Evaluationen helfen neue Konzepte hinsichtlich ihrer Benutzbarkeit und Verständlichkeit zu überprüfen und aus den Daten gewonnene Erkenntnisse unmittelbar in das laufende Projekt mit einfließen zu lassen.

## 1.2 Universitätsbibliothek Konstanz

Die Bibliothek der Universität Konstanz offeriert einen Datenbestand von über zwei Millionen Titel verschiedenster Medientypen. Hier sind Bücher, Zeitschriften, Daten und Audio-CDs, Videos, DVDs und Mikrofiches zu finden. Des Weiteren bilden identische Exemplare oder auch mehrteilige Werke logische Hierarchien, welche theoretisch beliebig, aber praktisch nur bis zu einer Tiefe von drei Ebenen geschachtelt werden können. Dieser heterogene Informationsraum wird für den Bibliotheksanwender durch ein konventionelles Online-Suchsystem namens KOALA<sup>4</sup>, welches auf HTML-Formularen und Hyperlink-Strukturen basiert, zugänglich.

## 1.3 Problemstellung

Das Framework MedioVis enthält ein objektorientiertes Datenmodell mit definierten Schnittstellen, jeweils in Richtung der Datenquellen und Visualisierungen. Einfache Datenstrukturen und Relationen können konsistent abgebildet und performant verwaltet werden. Hierarchische Strukturen, welche in den Bibliotheksdaten ausgiebig Verwendung finden, wurden bisher jedoch noch nicht unterstützt.

Das Ziel dieser Arbeit war folglich, das Datenmodell zu Restrukturieren, um beliebi-

---

<sup>2</sup>Im Folgenden wird nicht nur das Projekt, sondern auch das Framework inklusive Visualisierungen als Gesamtgebilde mit dem Begriff „MedioVis“ belegt.

<sup>3</sup>DROID: Dynamic Remote Operation Incident Detection, AG Mensch-Computer Interaktion, Universität Konstanz, <http://hci.uni-konstanz.de/index.php?a=research&b=projects&c=198139>

<sup>4</sup>KOALA: Konstanzer Ausleih- und Anfrage-System, <http://www.ub.uni-konstanz.de/koala>, basiert seit Januar 2005 auf dem Library Management System LIBERO, <http://www.libero.com.au>

ge Hierarchien verwalten zu können und die Schnittstellen und Interaktionstechniken entsprechend anzupassen. Parallel hierzu wurde eine bereits bestehende tabellenartige Ansicht (siehe Kapitel 2.5.1) im Hinblick auf die neuen Gegebenheiten und den Einsatz in der Bibliothek überarbeitet. Je nach Bedarf sollen nun die logischen Strukturen auch visuell kommuniziert und Interaktionsmöglichkeiten, wie Sortierung, Filterung, Fokussierung und Selektion über alle Hierarchien hinweg angeboten werden.

Das Ergebnis der Restrukturierung des Datenmodells und der Umgestaltung der Visualisierungen findet derzeit als MedioVis „Jean-Luc“ Version 0.70m in der Universitätsbibliothek Konstanz für den Datenbestand der Mediothek reelle Anwendung. Die Mediothek ist ein physisch und thematisch abgegrenzter Teil der Gesamtbibliothek und beherbergt vornehmlich VHS, CDs, DVDs und weitere mediale Titel. Dieser Teilausschnitt ist auf Grund der extremen Heterogenität und dem ansteigenden Bedarf seitens der Studenten von besonderem Interesse und bietet darüber hinaus eine überschaubare Menge von Titeln. Technisch kann die eingesetzte Version auch auf die gesamte Bibliothek skaliert und somit als generelles Suchsystem verwendet werden.

## 1.4 Aufbau der Arbeit

In Kapitel 2 wird das MedioVis Framework und dessen Kontext vorgestellt. Es werden die für die Entwicklung relevanten Projekte, die Anwendungsdomäne und das Zielpublikum beschrieben. Außerdem werden Visualisierungs- und Interaktionskonzepte aufgeführt, welche im Bibliotheksszenario Anwendung finden.

Auf das umstrukturierte Datenmodell und dessen Grundkomponenten wird in Kapitel 3 näher eingegangen. Anhand von Code-Fragmenten werden die wichtigsten Elemente erklärt und die Zusammenhänge dargestellt. Des Weiteren wird der hierarchische Aufbau mittels Referenzierung und die Auswirkungen von Filter, Selektion und Fokussierung thematisiert.

Das Gesamtresumé in Kapitel 4 schließt die Arbeit ab.

## 2 MedioVis Framework

Das MedioVis Framework hat sich iterativ aus verschiedenen Vorgängerprojekten heraus entwickelt, wobei aufgrund der unterschiedlichen Applikationsdomänen und veränderten Anforderungen diverse Restrukturierungsmaßnahmen und Erweiterungen vollzogen wurden. Im Folgenden werden die relevanten Projekte und deren Ausrichtung kurz angesprochen. Ausführlichere Beschreibungen können im Rahmen dieser Arbeit nicht gegeben werden – nähere Informationen sind aber unter den aufgeführten Referenzen zu finden.

### 2.1 Historische Entwicklung

INSYDER<sup>1</sup>, Teilprojekt des EU-Programms ESPRIT, wurde 1998 mit dem Ziel initiiert kleinen und mittelständischen Unternehmen das Suchen, Visualisieren und Analysieren von wirtschaftlich relevanten Daten aus dem Internet durch eine intuitive, visuelle Umgebung und den Einsatz von linguistischen Recherche-Verfahren zu erleichtern [RMMH00]. Die Benutzeroberfläche basierte auf mehreren Ergebnisdarstellungen, Result Table, Scatterplot, Bar Chart und Segment View, welche wie auch das gemeinsame Framework in JAVA<sup>2</sup> umgesetzt wurden und als Ausgangsbasis für die weiteren Entwicklungen dienten.

Ende 2001 folgte ein weiteres EU-Projekt namens INVISIP<sup>3</sup>, welches Standortentscheidungen begleiten und nachfolgende Prozesse sowie beteiligte Parteien unterstützen sollte. Hierfür wurde ein Metadaten Browser zur Suche und Analyse von Geo-Metadaten konzipiert, entwickelt und evaluiert [GHRM02]. Die Diversität der veränderten Anwendungsumgebung – Einsatz als eingebundenes Applet oder alleinstehende Applikation, lokale XML-Dateien oder Online-Datenbanken als Datenquellen und zusätzliche Visualisierungen inklusive deren Synchronisation – stellte das von INSYDER übernommene Framework vor neue Herausforderungen und erzwang ein erstes Redesign.

Die Visualisierungs- und Interaktionskonzepte, welche im Rahmen von INVISIP entwickelt wurden, erschienen auch für die weitere Forschung von Interesse, waren aber

---

<sup>1</sup>INSYDER: INternet SYstème DE Recherche, Projekt-Nr. 29232, EU-Rahmenprogramm ESPRIT, <http://www.insyder.com>

<sup>2</sup>JAVA, Sun Microsystems, <http://java.sun.com>

<sup>3</sup>INVISIP: INformation VISualization in SIte Planning, IST-2000-39640, <http://www.invisip.de>

sehr auf die Anwendungsdomäne zugeschnitten und über die Monate hinweg implementierungstechnisch nicht optimal gewachsen. Daraus ergab sich zum Ende des Projektes der Wunsch nach einer Generalisierung der Konzepte sowie einer Abstraktion, wodurch sich eine erhöhte Flexibilität des Frameworks ergeben sollte. Dieses Vorhaben zog eine komplette Restrukturierung der einzelnen Teilkomponenten des bisherigen INVISIP Frameworks und eine Modularisierung der einzelnen Visualisierungen mit sich. Des Weiteren wurden ein so genannter Assignment Editor entwickelt [Grü04], welcher dem Anwender eine freie Konfiguration der Visualisierungen und der Datenressourcen erlaubt.

Aus diesem Vorhaben resultierte dann ein komplett restrukturiertes Framework mit verschiedenen kombinierbaren Visualisierungen, welches als Gesamtkonstrukt den Projektnamen VisMeB, Visueller Metadaten Browser, erhielt [KRML03].

Ein weiterer Meilenstein in der Entwicklung des heutigen Frameworks war die Identifizierung der Bibliothek Konstanz als interessante Einsatz- und Testdomäne. Die Menge an intellektuell gepflegten Daten, die beachtliche Medienvielfalt und die physische Nähe zu Bibliotheksmitarbeitern und -anwendern ließen eine optimale Forschungsumgebung erhoffen. Nach Gesprächen mit Bibliotheksverantwortlichen und Interviews mit realen Bibliothekskunden wurde VisMeB, welches im Rahmen von INVISIP ursprünglich als Experten-Tool konzipiert war, auf die Bedürfnisse von Gelegenheitsanwender und die in Gesprächen identifizierten typischen Suchaufgaben der Bibliotheksbenutzer zugeschnitten.

Mitte 2004 wurde dann eine erste Vorversion unter dem Projektnamen „MedioVis“ für den Testbetrieb im Mediotheksbereich der Universitätsbibliothek Konstanz freigegeben. MedioVis sollte alternativ zu dem Online-Suchsystem KOALA (siehe Kapitel 1.2) die Bibliothekskunden bei der Suche und Exploration des Mediotheksbestandes mittels geeigneter Visualisierungs- und Interaktionskonzepte unterstützen und durch angereicherte Metadaten, wie Bildillustrationen, Beschreibungen und Kennwerte, zusätzlichen Mehrwert bieten.

Eine Weiterentwicklung von MedioVis wurde im Oktober 2005 auf über 150 PC-Arbeitsplätzen<sup>4</sup> innerhalb der Universitätsbibliothek für den praktischen Einsatz zugänglich gemacht, wobei der explorierbare Datenraum, analog zu der Vorversion, auf den Titelbestand der Mediothek eingegrenzt wurde. In der neuen Version wurde nun erstmals die Problematik von mehreren identischen Exemplaren und mehrteiligen Werken – also inhärente Hierarchien, welche vor allem im Bezug auf eindeutige Signaturen und den jeweiligen Verfügbarkeitsstatus unabdingbar sind – behandelt. Die erforderlichen Maßnahmen zur Unterstützung von hierarchischen Strukturen im Framework MedioVis und die hierfür nötigen Designentscheidungen werden in dieser Arbeit näher erläutert.

---

<sup>4</sup>Bis zum Zeitpunkt 01. November 2005 wurden 169 unterschiedliche Clients durch DROID identifiziert, welche sich in der Bibliothek befinden und bis dato zumindest einmal MedioVis ausführten.

## 2.2 Benutzerprofil

Der Anwendungskontext von MedioVis in der Universitätsbibliothek Konstanz soll im Folgenden durch eine Profilbeschreibung der typischen Benutzer des Systems und im nächsten Kapitel durch ein charakteristisches Anwendungsszenario näher verdeutlicht werden.

Aufgrund der durch DROID erhobenen Nutzungsstatistiken aus über 3800 protokollierten Sessions seit Juli 2004 kann geschlossen werden, dass nur ca. 5% der Anwender von MedioVis Mitarbeiter<sup>5</sup> der Bibliothek sind, welche man sicherlich durch ihre speziellen Anforderungen von dem Rest der Benutzer trennen muss. Auch eine Benutzerstatistik des gesamten WWW-Angebots der Universitätsbibliothek Konstanz von 2002 [Eng02] kommt mit ca. 11% Mitarbeiteranteil auf vergleichbare Ergebnisse. Der Großteil der tatsächlichen Anwender sind demnach Studenten, Doktoranden, Professoren und Externe, welche Zugang zu den über 150 in der Bibliothek verteilten PC-Arbeitsplätzen haben.

Im Allgemeinen ist es schwierig für eine so breit gefächerte Benutzergruppe gültige Aussagen zu treffen. Man kann jedoch aufgrund der Hochschulanforderung einen gewissen Bildungsstand und zumindest grundlegende Computer-Kenntnisse voraussetzen. Auch kann von einem grundsätzlichen Interesse für MedioVis ausgegangen werden, da dieses alternativ zu dem konventionellen KOALA System zur Verfügung steht. Des Weiteren scheint die überwiegende Mehrheit der Anwender eher in die Gruppe der Erst- oder Gelegenheitsnutzer, als in die Expertengruppe einordbar zu sein.

MedioVis setzt sich folglich die Erst- und Gelegenheitsnutzer mit durchschnittlichen Computerkenntnissen und gutem Allgemeinverständnis als Hauptzielgruppe, wobei die Gruppe der Experten aber nicht vernachlässigt werden soll.

## 2.3 Anwendungsszenario

Der Verwendung des Systems und die einzelnen Komponenten sollen nun exemplarisch anhand eines charakteristischen Beispielszenarios eingeführt werden.

Evelyn Schmidt<sup>6</sup> studiert im zweiten Semester Politikwissenschaften an der Universität Konstanz. Ihr steht ihr erstes Referat mit dem Thema „Deutschland im Wandel des Zwanzigsten Jahrhunderts“ bevor und geht ausgestattet mit ein paar Ideen und Notizen

---

<sup>5</sup>Mitarbeiter der Bibliothek wurden anhand ihrer eindeutigen IP-Adressen bzw. internen Subnetzmaske identifiziert.

<sup>6</sup>Fiktiver aber für MedioVis-Benutzer exemplarischer Charakter

zu relevanter Literatur in die Bibliothek. Hier startet sie MedioVis an einem der öffentlich zugänglichen PC-Arbeitsplätze.

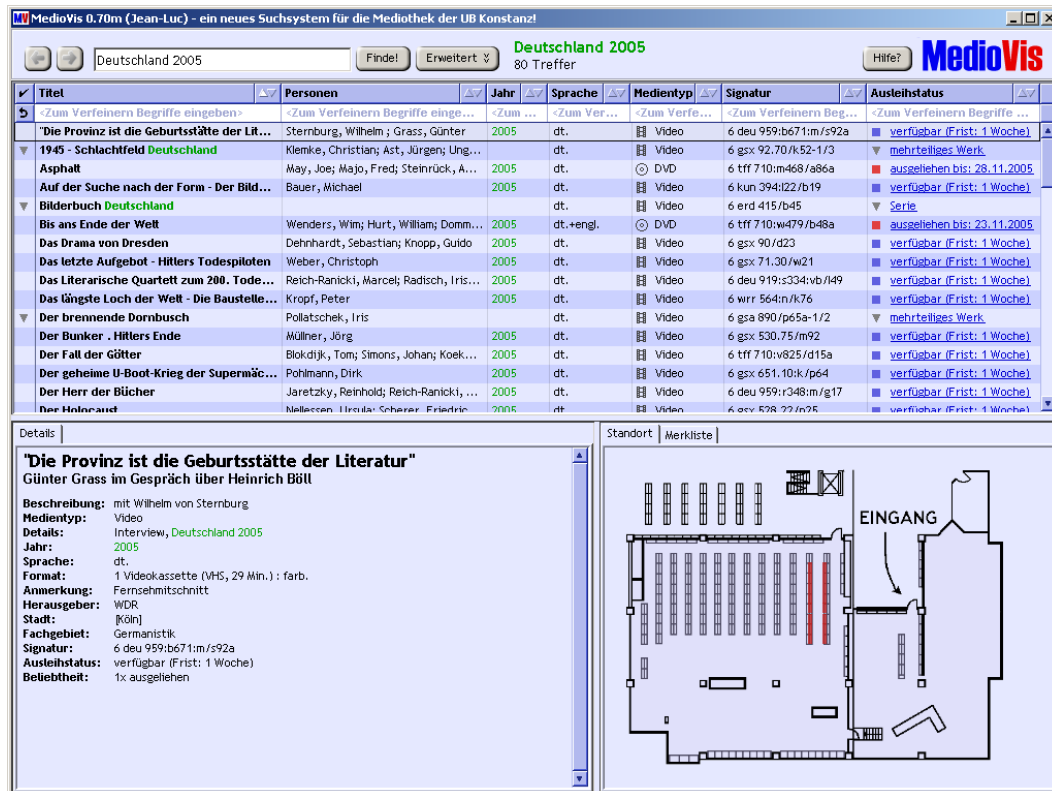


Abbildung 2.1: MedioVis Oberfläche mit TreeTable, Detail- und Standort-Ansicht. Bezüglich „Deutschland 2005“ wurden 80 Suchtreffer gefunden.

Zu Beginn ihres Vortrages möchte sie einen kleinen Filmausschnitt als Motivation zeigen. Hierfür wurde ihr eine erst vor kurzem erschienene mehrteilige Dokumentation zu dem Thema empfohlen, welche sich als DVD in der Mediothek befinden soll. Da sie keine genaueren Angaben zur gewünschten Dokumentation vorliegen hat, startet sie erstmals eine Suche mit dem Stichwort „Deutschland 2005“. Daraufhin bekommt sie einen Wimpernschlag später 80 Suchtreffer (siehe Abbildung 2.1) in der so genannten TreeTable, einer tabellenartigen Visualisierung (siehe Kapitel 2.5.1), präsentiert. Sie möchte die Suchresultate weiter einschränken und gibt beim integrierten Tabellenfilter in der Spalte Medientyp das Stichwort „DVD“ ein, wodurch die ursprünglich 80 Resultate auf nur noch überschaubare zehn Titel reduziert werden.

Durch die farbliche Hervorhebung der gefundenen Stichwörter fällt ihr nun der Titel „Hundert Jahre Deutschland“ auf und erkennt anhand des grauen Pfeils in der ersten Spalte und des Textes in der Spalte Ausleihstatus, dass es sich dabei um ein mehrteiliges Werk handelt. Nach Mausklick auf die Zeile des mehrteiligen Werkes entfalten sich in einer Animation die einzelnen Titel des Werkes und deren mehrfach vorkommenden iden-

tischen Exemplare. Evelyn jauchzt auf und freut sich über ihren Volltreffer. Mit einem kurzen Blick über die Spalte Ausleihstatus stellt sie fest, dass die für sie interessanten Titel noch verfügbar sind und für eine Woche ausgeliehen werden können.

Gemäß dem Brushing & Linking [BC87] Konzept werden ihr beim Überfahren der einzelnen Titel in der TreeTable zusätzliche Meta-Informationen, wie zugeordnete Fachgebiete, Herausgeber und zum Teil Abbildungen der entsprechenden Bücherfronten in der Detail-Ansicht (siehe Kapitel 2.5.2) präsentiert. Des Weiteren bietet die Standort-Ansicht (siehe Kapitel 2.5.3) für den jeweilig fokussierten Titel eine Standortmarkierung in einer schematischen Umrisskarte der Mediothek als Hilfe zur besseren Lokalisation.

The screenshot shows the MedioVis 0.70m search interface. The search results table is filtered to show 10 DVD titles from 2005. The selected title, 'Hundert Jahre Deutschland: Der Holocaust', is expanded to show its details in the 'Details' pane. The 'Standort' pane shows a list of related titles from the 'Hundert Jahre Deutschland' series.

Titel	Personen	Jahr	Sprache	Medientyp	Signatur	Ausleihstatus
Bis ans Ende der Welt	Wenders, Wim; Hurt, William; Domm...	2005	dt.+engl.	DVD	6 tff 710:ww479/b-48a	ausgeliehen bis: 23.11.2005
Die Messtheorie von Rasch in Psycholog...	Rost, Jürgen	2005	dt.	DVD	6 psy 64/r68	verfügbar (Frist: 1 Woche)
Die Reise nach Kafiristan	Dubini, Fosco; Dubini, Donatello; H...	2005	dt.	DVD	6 tff 710:d814/r24	ausgeliehen bis: 30.11.2005
Hundert Jahre Deutschland	Aust, Stefan	2005	dt.	DVD	6 gsx 521/a98	mehrteiliges Werk
Aufbruch aus Ruinen				DVD	6 gsx 521/a98-7...	mehrere Exemplare
Das Ende des sowjetischen Imperiums		2005	dt.	DVD	6 gsx 521/a98-7	ausgeliehen bis: 24.11.2005
Der Führerstaat		2005	dt.	DVD	6 gsx 521/a98-10...	mehrere Exemplare
Der Holocaust		2005	dt.	DVD	6 gsx 521/a98-10	ausgeliehen bis: 24.11.2005
Der Kalte Krieg		2005	dt.	DVD	6 gsx 521/a98-2...	mehrere Exemplare
Der Untergang des Dritten Reiches		2005	dt.	DVD	6 gsx 521/a98-2	verfügbar (Frist: 1 Woche)

**Details: Hundert Jahre Deutschland : Der Holocaust**

Medientyp: DVD  
 Details: Enth. Interviews mit Brigitte Hamann, Ian Kershaw, Helmut Schmidt, Richard von Weizsäcker, Roman Herzog, Hans-Jochen Vogel, Egon Bahr, Hans-Dietrich Genscher, Angela Merkel, Joachim Fest.  
 Jahr: 2005  
 Sprache: dt.  
 Format: 1 DVD-Video (144 Min.) : teilw. s/w  
 ISBN/ISSN: ISBN 3-937901-12-4  
 Fachgebiet: Geschichte 1918-2000  
 Signatur: 6 gsx 521/a98-4  
 Ausleihstatus: verfügbar (Frist: 1 Woche)  
 Beliebtheit: 2x ausgeliehen

**Standort Merkliste**

- Hundert Jahre Deutschland : Der Weg ins 21. Jahrhundert  
Ausleihstatus: ausgeliehen bis: 24.11.2005  
Medientyp: DVD  
Signatur: 6 gsx 521/a98-12
- Hundert Jahre Deutschland : Die deutsche Wiedervereinigung  
Ausleihstatus: ausgeliehen bis: 24.11.2005  
Medientyp: DVD  
Signatur: 6 gsx 521/a98-11
- Hundert Jahre Deutschland : Der Holocaust  
Ausleihstatus: verfügbar (Frist: 1 Woche)  
Medientyp: DVD  
Signatur: 6 gsx 521/a98-4
- Hundert Jahre Deutschland : Der Führerstaat

Abbildung 2.2: Die Suchtreffer wurden mittels Tabellenfilter auf 10 DVDs eingeschränkt. Die ausgewählten Elemente des expandierten mehrteilige Werkes „Hundert Jahre Deutschland“ werden in der Merkliste geführt.

Da Evelyn nun die für sie relevanten und verfügbaren Titel extrahiert hat, selektiert sie nun diese durch Anklicken und druckt die Auswahl über die integrierte Merkliste (siehe Kapitel 2.5.4) aus. Mit ausgedruckter Titel-, Signatur- und Bildbeschreibung kann Evelyn nun gezielt die verfügbaren Exemplare in der Mediothek ansteuern und gegebenenfalls ausleihen.

Dieses Szenario zeigt nur einen Teilausschnitt der Anwendungsmöglichkeiten, Visualisierungen und Funktionen von MedioVis, soll aber hier als Einführung und Kontexterläute-

rung genügen. Im weiteren Verlauf werden die Diversität der angebotenen Datenquelle und ihre spezifischen Eigenheiten und darauf folgend eine Auswahl von Visualisierungen und deren Funktionen näher betrachtet.

## 2.4 Datenquellen

Beim Design des MedioVis Frameworks wurde viel Bedacht auf eine möglichst flexible Datenanbindung gelegt. Damit sollte eine vertikale und horizontale Kompatibilität über Projekte, Domänen und Versionen hinweg gewährleistet werden. So ist es möglich auf der Basis eines einheitlichen Frameworks verschiedene Ausprägungen einer Visualisierung miteinander vergleichen oder den Einsatz einer Visualisierung in verschiedenen Domänen evaluieren zu können.

Im Laufe der Zeit wurde MedioVis an unterschiedlichste Datenbasen, von geographischen Metadaten im Projekt INVISIP, über bibliographische Sammlungen wissenschaftlicher Publikationen, wie DBLP<sup>7</sup>, bis hin zu sehr dynamischen Online-Diensten, wie Yahoo, Ebay oder P2P-Tauschbörsen<sup>8</sup> angekoppelt.

Im Rahmen des Projektes MedioVis wird hauptsächlich der Datenbestand der Universitätsbibliothek Konstanz bzw. deren Untermenge „Mediothek“ und gespiegelte Film- und Schauspielerdaten der Internet Movie Database IMDb<sup>9</sup> verwendet [GGJ<sup>+</sup>05].

Die Bibliotheksdaten werden vom Library Management System LIBERO im MAB2-Format<sup>10</sup>, einem standardisierten Format für Bibliotheksdaten, verwaltet. Regelmäßig<sup>11</sup> wird der gesamte Datenbestand der Bibliothek ausgelesen, in ein relationales Datenschema konvertiert und in eine PostgreSQL<sup>12</sup> Datenbank importiert. Des Weiteren werden diese textuellen Daten durch passendes Bildmaterial aus den Medienbeständen der Online-Dienste Amazon<sup>13</sup> und IMDb angereichert.

Zum Teil findet sogar eine dynamische Vereinigung von mehreren Datenquellen innerhalb

---

<sup>7</sup>DBLP: Digital Bibliography & Library Project, über 600.000 indexierte Artikel, Universität Trier, <http://dblp.uni-trier.de>

<sup>8</sup>P2P: Peer-to-Peer Netzwerke, gleichgestellte, dezentrale Kommunikation ohne klassische Client-Server Rollenverteilung

<sup>9</sup>IMDb: Internet Movie Database, Metadaten zu mehr als 500.000 Filmproduktionen und 1,8 Millionen Personen, <http://www.imdb.com>

<sup>10</sup>MAB2: Maschinelles Austauschformat für Bibliotheken, Version 2

<sup>11</sup>Zum Zeitpunkt der Erstellung dieser Arbeit wurde jeden zweiten Tag ein Update des gesamten Datenbestandes der Bibliothek Konstanz durchgeführt.

<sup>12</sup>PostgreSQL, Open Source Datenbank System, <http://www.postgresql.org>

<sup>13</sup>Amazon: Internet-Versandhaus, Internet-Marktführer für Bücher, CDs und Videos, <http://www.amazon.com>

des MedioVis Frameworks statt. So können beispielsweise zu Filmdaten semantisch in Bezug stehende Personendaten angezeigt werden, wobei Film- und Personendaten aus komplett verschiedenen Datenquellen bzw. Datenbanken stammen können [JGK<sup>+</sup>05].

Die Datenanfragen und der Datentransfer findet mittels der standardisierten Abfragesprache SQL<sup>14</sup> über TCP/IP<sup>15</sup> statt. Dies ermöglicht den Einsatz von verschiedensten Datenbanksystemen und eine weitestgehend physische Unabhängigkeit.

Um eine komplette Unabhängigkeit, d.h. auch eine Anfrage ohne Zugang zum Internet zu ermöglichen, kann eine gespiegelte Datenbank auch lokal auf einem Laptop oder ähnlichem installiert werden. Ebenso können die Daten auch im CSV-Format<sup>16</sup> als Textdateien der Anwendung zur Verfügung gestellt werden, wobei aber die Performance merklich darunter leidet.

Zusammenfassend kann von einer Fülle von Möglichkeiten zur Anbindung von ein oder mehreren parallelen Datenquellen an das MedioVis Framework gesprochen werden. Des Weiteren werden verschiedenste Datentypen, wie Texte, numerische Werte, Bilder oder Filme unterstützt. Einfache Relationen können sogar dynamisch hergestellt und entsprechend behandelt werden. Die interne Abbildung und Auswertung von hierarchischen Strukturen werden jedoch erst jetzt mit den Änderungen, welche Gegenstand dieser Arbeit sind, ermöglicht.

## 2.5 Visualisierungs- und Interaktionskonzepte

Der Berührungspunkt zwischen MedioVis und Mensch ist natürlicherweise die Benutzeroberfläche, d.h. die Hauptanwendung und darin eingebettet die einzelnen Visualisierungsmodule und deren Interaktionsangebot. Hier ist auch der Forschungsschwerpunkt der Arbeitsgruppe Mensch-Computer Interaktion der Universität Konstanz zu finden. Die Beweglichkeit im Innovationsprozess wird durch eine durchgängige Modularisierung der Visualisierungen und einer wohl definierten, aber trotzdem flexiblen Interaktionsschnittstelle unterstützt.

Das Framework MedioVis kann mit einer beachtlichen Visualisierungsvielfalt aufwarten. Im Rahmen dieser Arbeit kann aber nur ein Teilausschnitt davon beschrieben werden, welcher vor allem in der Bibliotheksdomäne Anwendung findet. Im Folgenden werden die schon im Kapitel 2.3 kurz angesprochenen Visualisierungen TreeTable, Detail-Ansicht, Standort-Ansicht und Merkliste einzeln und in ihrem Zusammenspiel thematisiert.

<sup>14</sup>SQL: Deklarative Abfragesprache für Relationale Datenbanken

<sup>15</sup>TCP/IP: Internet Protocol Suite, Basis der Internetkommunikation, Standardprotokolle für Transport- und Vermittlungs-Schicht im OSI-Modell

<sup>16</sup>CSV: Character Separated Values, durch definierte Trennzeichen strukturierte Textdatei

## 2.5.1 TreeTable

Nach dem Starten von MedioVis wird dem Anwender ein einfaches Suchfeld angeboten, mit welchem er, analog zu bekannten Online-Suchsystemen wie Yahoo<sup>17</sup> oder Google<sup>18</sup>, Schlüsselwörter als Suchargumente definieren kann. Nach Betätigen des Suchbuttons wird der gesamte Datenbestand anhand der eingegebenen Schlüsselwörter eingeschränkt und mittels der TreeTable visualisiert.

Grundsätzlich besteht diese aus einer konventionellen Tabelle, welche dem Anwender als Ausgangsbasis eine vertraute Umgebung bieten soll. Die einzelnen Suchtreffer werden in Zeilen angeordnet. Auf die Spalten werden die Attribute der Treffer abgetragen, wobei Auswahl und Reihenfolge mittels des in Kapitel 2.1 angesprochenen Assignment Editors frei definiert werden können. Die Spaltenköpfe tragen den zugeordneten Attributnamen und dienen gleichzeitig als Sortierbutton für die jeweilige Spalte.

The screenshot shows the MedioVis 0.70m search interface. At the top, there is a search bar with the query 'scorsese' and buttons for 'Finde!' and 'Erweitert'. Below the search bar is a TreeTable with columns: Titel, Personen, Jahr, Sprache, Medientyp, Signatur, and Ausleihstatus. The table lists various titles, including 'Aviator', 'Gangs of New York', 'Il mio viaggio in Italia', 'Kap der Angst', 'Martin Scorsese presents: The Blues', 'Feeling like going home', 'Godfathers and sons', 'Piano Blues', 'Red, White & Blues', 'The road to Memphis', 'The soul of a man', 'Warning by the devil's fire', and 'New Yorker Geschichten'. The 'Il mio viaggio in Italia' entry is expanded, showing multiple copies and their availability status. The 'Martin Scorsese presents: The Blues' entry is also expanded, showing its multi-part nature. Below the table, there are two detail panels. The left panel, titled 'New Yorker Geschichten', shows details for a DVD set, including the director (Francis Coppola), cast, and release information. The right panel, titled 'Il mio viaggio in Italia', shows details for a video set, including its availability and release information. On the far right, there are icons for 'Drucken', 'Mailen', 'Speichern', and 'Leeren'.

Abbildung 2.3: TreeTable (obere Hälfte) mit Detail-Ansicht und Merkliste. Die hierarchischen Titel „Il mio...“ (mehrere Exemplare) und „Martin Scorsese...“ (mehrteiliges Werk) sind expandiert und Untertitel z.T selektiert.

Direkt unter den Spaltenköpfen stehen für jede Spalte Eingabefelder bereit, welche die

<sup>17</sup><http://www.yahoo.com>

<sup>18</sup><http://www.google.com>

Ergebnismenge anhand der dort eingegebenen Schlüsselworte schon während der Eingabe dynamisch filtern. Es werden nur diejenigen Dokumente in der TreeTable dargestellt, welche dem dort formulierten Filterkriterium (bzw. mehreren mit Booleschem<sup>19</sup> „UND“ verknüpften Filterkriterien) entsprechen. Wird beispielsweise unter „Medientyp“ das Schlüsselwort „DVD“ eingegeben und unter „Jahr“ „200“, reduziert sich die Darstellung auf DVDs, welche schon in diesem Jahrzehnt veröffentlicht wurden. Dieser dynamische Filtermechanismus wird im Folgenden als Tabellenfilter bezeichnet.

Speziell im Bibliotheksszenario kann man zwei verschiedene Dokumentarten unterscheiden. Zum Einen sind es einfache Dokumente, die ohne spezifische Relation für sich alleine stehen. Zum anderen kann man „Container“-Objekte identifizieren, welche mehrere in Beziehung stehende Dokumente unter sich vereinen. So werden identische Dokumente (siehe „Il mio...“ in Abbildung 2.3) unter einem Repräsentator zusammengefasst oder die einzelnen Teilobjekte eines mehrteiligen Werkes (siehe „Martin Scorsese...“ in Abbildung 2.3) einem Eintrag für das Gesamtwerk untergeordnet.

Würden diese logischen Hierarchien nicht auch in der Darstellung visuell kommuniziert werden, könnte der Anwender beispielsweise von einem Unterdokument nicht mehr auf das Gesamtwerk schließen und inhärente Informationen, wie der übergeordnete Gesamttitel, würden verloren gehen. Auch eine gemeinsame Darstellung identischer Exemplare erleichtert zuerst durch eine Minimierung der Redundanzen die Übersicht, gibt aber trotzdem bei Bedarf die Information über die Anzahl und jeweilige Position der identischen Dokumente frei.

Um dem Anwender bei der ersten Ansicht der Suchresultate einen guten Gesamtüberblick zu ermöglichen, werden bei hierarchischen Daten immer nur die übergeordneten Repräsentanten angezeigt. Durch einen Klick mit der linken Maustaste in die Zeile einer dieser Repräsentanten werden dann die untergeordneten Elemente dieses Containers in einer Animation nach unten aufgeklappt und nehmen nun visuell gruppiert jeweils eine eigene Zeile in der TreeTable ein. So können detaillierte Informationen zu den Teildokumenten eines Containers auf Wunsch angezeigt werden, ohne jedoch den Anwender bei der Erstante durch Redundanzen zusätzlich kognitiv zu belasten. Ein Klick mit der rechten Maustaste auf das Oberdokument klappt die Unterhierarchien wieder zu.

Einfache Dokumente oder Unterdokumente können per Mausklick selektiert bzw. deselektiert werden, was zugleich auch eine Übertragung des Dokumentes in die Merkliste oder gegebenenfalls dessen Entfernung bewirkt. So kann sich der Anwender aus den dargestellten Dokumenten eine Auswahl erstellen, welche er speichern, drucken oder mailen kann (siehe Kapitel 2.5.4).

---

<sup>19</sup>Boolesche Algebra mit logischen Operatoren UND, ODER und NICHT nach George Boole, 19. Jahrhundert.

The screenshot shows the MedioVis 0.70m search interface. The search term 'heidelberg' has been entered, resulting in 393 hits, with 55 visible. The main window displays a table of search results with columns for Title, Persons, Year, Language, Media Type, Signature, and Availability Status. The table is filtered to show only items from Heidelberg. The selected item is 'Einführung in die Statistik der Finanzmärkte' by Franke, Jürgen; Hürde, Wolfgang, published in 2001. Below the table, a detailed view of this item is shown, including its description, metadata (ISBN, ISSN, etc.), and a floor plan of the library location with an 'EINGANG' (entrance) marked.

Titel	Personen	Jahr	Sprache	Medientyp	Signatur	Ausleihstatus
Die Vögel Europas		2000	dt.	CD-Rom	6 bio 776:e/v62	ausgeliehen bis: 22.11.2005
Einführung in die Statistik der Finanzmärkte	Franke, Jürgen; Hürde, Wolfgang,...	2001	dt.	CD-Rom	6 wrk 499:u/f71a	verfügbar
European limnafauna	Visser, Henk	2000	engl.	CD-Rom	6 bio 712:zc38 /v48	verfügbar
Flamenco	Leblon, Bernard	2001	dt.+span.	diverse	6 tff 543/I21	mehrteiliges Werk
Flamenco		2001	span.	Tonträger	6 tff 543/I21-cd	ausgeliehen bis: 12.12.2005
Flamenco		2001	dt.	Buch	6 tff 543/I21-a	ausgeliehen bis: 12.12.2005
Fundamentals of physics and chemistry...	Visconti, Guido	2001	engl.	Buch	6 erd 164:n/v48-cdrom...	mehrere Exemplare
Graphs and applications	Aldous, Joan; Wilson, Robin J.	2000	dt.	Buch	6 mat 9:a125:ra/g71-cd...	mehrere Exemplare
Introduction to time series and forecast...	Brockwell, Peter J.; Davis, Richard A.	2002	engl.	Buch	6 sta 230/b75c(2)-cdrom...	mehrere Exemplare
Jahrbuch Telekommunikation und Gese...		1999		Zeitschrift		mehrteiliges Werk
Global home		2000		CD-Rom	6 soz 794:i/t25b-2000...	mehrere Exemplare
InternetFuture		2001		Buch	6 kid 920:i/t25b-8,cdrom	verfügbar
Klimafolgen für Mensch und Küste am Be...	Daschkeit, Achim	2002	dt.	Buch	6 soz 794:i/t25b-2001,...	verfügbar
					6 erd 420.34/d18-cdro...	mehrere Exemplare

**Details: Einführung in die Statistik der Finanzmärkte**

**Beschreibung:** J. Franke; W. Hürde; C. Hafner  
**Medientyp:** CD-Rom  
**Details:** Buch-Ausg. u.d.T.: Franke, Jürgen: Einführung in die Statistik der Finanzmärkte  
**Jahr:** 2001  
**Sprache:** dt.  
**Format:** 1 CD-ROM; 12 cm  
**Anmerkung:** 720 ddsu/skü. - Buchausg. s. idn 9449634; 721:L1UB  
**Herausgeber:** Springer  
**Stadt:** Berlin; Heidelberg [u.a.]  
**ISBN:** ISBN 3-540-41722-2  
**Fachgebiet:** Geld  
**Signatur:** 6 wrk 499:u/f71a  
**Ausleihstatus:** verfügbar  
**Beliebtheit:** 7x ausgeliehen

**Standort | Merkliste |**

EINGANG

Abbildung 2.4: MedioVis mit 55 von 393 mittels Tabellenfilter gefilterten Suchtreffer.

Das Sortieren und Filtern in der TreeTable wirkt nicht nur auf die oberste Hierarchie der Dokumente, sondern rekursiv über alle Ebenen hinweg. Bei der Sortierung werden die Unterdokumente jeweils auf gleicher Ebene innerhalb der Gruppierung sortiert – die logische Struktur bleibt daher erhalten. Der Tabellenfilter hat ebenfalls Auswirkung auf alle Ebenen, wobei hier zwei Sonderfälle beachtet werden müssen. Im Normalfall werden Dokumente, Container und Unterdokumente bzgl. der eingegebenen Filterargumente gefiltert bzw. ausgeblendet. Nehmen wir zur besseren Erläuterung der Sonderfälle das vorher aufgeführte Beispiel mit dem Argument „DVD“ in der Spalte „Medientyp“ und dem Argument „200“ in der Spalte „Jahr“. Wenn nun Teile eines mehrteiligen Werkes nicht gleichzeitig, sondern über mehrere Jahre hinweg veröffentlicht wurden, müsste zum Beispiel der Container (siehe Abbildung 2.4 „Jahrbuch Telekommunikation...“ Jahr = 1999) aufgrund des Erscheinungsjahres aus der Ansicht entfernt werden, wobei dessen Unterdokumente (Jahr = 2000/2001) dem Filter jedoch entsprechen. Daher wird in diesem Fall der Container trotzdem angezeigt, um die hierarchische Struktur und damit die inhärenten Informationen zu erhalten. Im gegensätzlichen Fall, wenn der Container dem Filter genügt, aber nicht die Unterdokumente, würde das alleinige Anzeigen des Containers wiederum wesentliche Informationen unterschlagen, da der Container für sich nur eine leere Hülle darstellt. Daher werden in diesem Fall alle Unterdokumente des gefilterten Containers angezeigt.

Als letztes soll noch die Spalte „Ausleihstatus“ näher erläutert werden. Zeitgleich zum Suchprozess wird zu jedem Suchtreffer der aktuelle Verfügbarkeitsstatus vom Libero Server der Universitätsbibliothek abgefragt. Der jeweilige Status des Dokumentes inklusive direktem Link zur Katalogdatenanzeige von Libero wird in der letzten Spalte der TreeTable angezeigt. So kann der Anwender beispielsweise nach der globalen Suche seine daraus resultierende Treffermenge auf nur noch verfügbare Dokumente mittels des gerade beschriebenen Tabellenfilters einschränken oder die Tabelle nach Verfügbarkeit sortieren.

## 2.5.2 Detail-Ansicht

Die TreeTable stellt die Suchtreffer anhand der den Spalten zugeordneten Attribute gegenüber. Um ein horizontales Scrollen zu vermeiden, kann aber nur eine geringe Anzahl von Attributen direkt in der Tabelle angezeigt werden. Die Detail-Ansicht schafft hier mit einem „Overview and Detail“-Konzept Abhilfe, indem sie zu dem gerade fokussierten Dokument alle vorhandenen Attribute formatiert in einer separaten Ansicht aufführt. Hier besteht auch die Möglichkeit, mediale Titel mit erhöhtem Platzbedarf, wie Filmposter oder Bücherfronten abzubilden. Wird direkt ein Unterdokument fokussiert, welches Teil eines mehrteiligen Werkes ist, werden dessen Attribute mit den Informationen des Gesamtwerkes ergänzt.

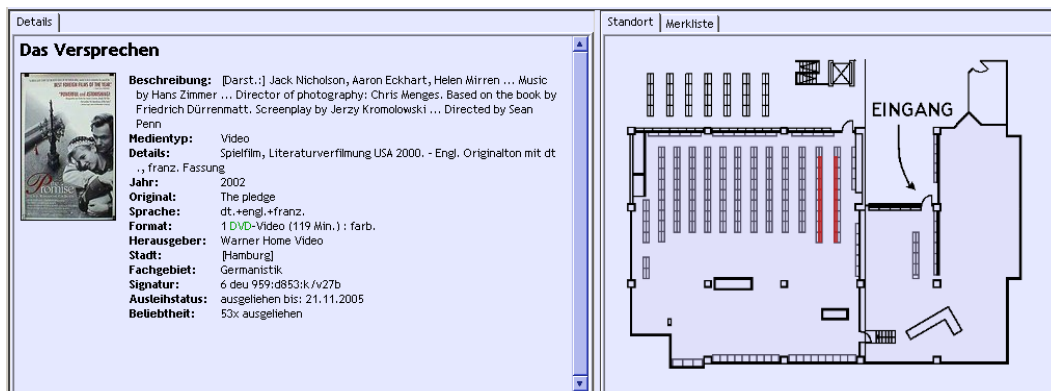


Abbildung 2.5: Detail-Ansicht (links) und Standort-Ansicht (rechts).

## 2.5.3 Standort-Ansicht

Die Standort-Ansicht macht sich zu nutze, dass die eindeutigen Signaturen der Titel im Bibliotheksbestand der Universität Konstanz Aufschluss über den zugeordneten physischen Standort geben. Anhand einer schematischen Umriss-Karte der Mediothek wird

für das gerade fokussierte Dokument jeweils mit farblicher Hervorhebung das Regal markiert, in welchem sich das entsprechende Dokument befinden sollte. Somit bekommt der Anwender für die Lokalisation nicht nur textuell, sondern auch visuell eine Hilfe in Form einer konventionellen und dem Betrachter vertrauten Kartenansicht.



Abbildung 2.6: Druckdialog der Merkliste mit Vorschau.

## 2.5.4 Merkliste

Wie schon in Kapitel 2.5.1 beschrieben, werden die in der TreeTable selektierten Dokumente in die Merkliste übertragen. Diese dient demnach als eine manuelle Zusammenstellung von vermeintlich relevanten Dokumenten. Im Gegensatz zur TreeTable, welche immer nur die aktuellen Suchtreffer anzeigt, führt die Merkliste die ausgewählten Dokumente persistent über mehrere Suchen hinweg. Einmal ausgewählte Dokumente verbleiben also und werden erst durch eine aktive Deselektion in Merkliste oder TreeTable aus der Merkliste entfernt. Technisch ist es sogar möglich die Auswahl über mehrere Sessions, also über die Laufzeit der Applikation hinweg durch Serialisierung vorzuhalten und beim erneuten Starten wieder einzulesen. Im Bibliotheksszenario wird aber von einer wechselnden Benutzerschaft ausgegangen, wodurch eine sessionweite Speicherung ohne weitere Personalisierung der Anwendung nur eingeschränkt sinnvoll und auch da-

tenschutzrechtlich bedenklich ist. Deshalb wird zumindest im Bibliothekseinsatz auf die Serialisierung verzichtet.

Die ausgewählten Dokumente werden in der Merkliste nacheinander aufgeführt. Zum Titel und gegebenenfalls Gesamttitel können noch weitere Attribute, wie Ausleihstatus, Medientyp und Signatur angezeigt werden. Über das Deselektieren einzelner Dokumente hinaus, stehen weitere Aktionsmöglichkeiten, wie das Drucken, Mailen, Speichern und Löschen der gesamten Merkliste zur Verfügung.

Ebenso wie alle im MedioVis integrierten Visualisierungen ist auch die Merkliste Teil des Brushing & Linking Konzeptes. Dokumente, welche in der Merkliste und in der TreeTable angezeigt werden und in einer der beiden Visualisierungen gerade den Fokus besitzen, werden auch in beiden farblich hervorgehoben.

Weitergehende und detailliertere Informationen zu den Visualisierungen und Interaktionsmöglichkeiten von MedioVis werden im nächsten Kapitel, anhand der technischen Umsetzung des Datenmodells und der definierten Schnittstellen, erklärt und veranschaulicht.

## 3 MedioVis Datenmodell

Zentrales und daher auch kritisches Element des MedioVis Frameworks stellt das Datenmodell und dessen Schnittstellen dar. Von Seiten der Datenanbindung werden Informationen aus verschiedenen Quellen gebündelt, aufbereitet und in logischen Strukturen abgebildet. Von Seiten der Visualisierungen müssen eben diese performant zur Verfügung gestellt, Zustände gespeichert, Filter angewendet und die Datenstrukturen verwaltet werden.

Im Falle eines Frameworks, welches per se leicht wieder verwertbar, anpassbar und erweiterbar sein soll, gelten für das integrierte Datenmodell dementsprechend besonders hohe Ansprüche. Im Folgenden werden die Anforderungen an das MedioVis Datenmodell als integrales Element des MedioVis Frameworks und die hieraus resultierende Komplexität thematisiert.

### 3.1 Anforderungen

Nicht nur verschiedenste Datenquellen, wie schon in Kapitel 2.4 aufgeführt, sondern auch unterschiedlichste Datenarten, vom Zahlenwert über Volltexte bis hin zu Bildern oder Filmen, sollen vom Datenmodell gespeichert, verwaltet und verarbeitet werden. Des Weiteren sollen zusätzlich zu den eigentlichen Daten, die inhärent vorherrschenden Beziehungen zwischen den Datenelementen als m:n Relationen oder Hierarchien über das Datenmodell für die Visualisierungen zur Verfügung stehen.

Eine klar definierte, aber individualisierbare Schnittstelle soll eine möglichst hohe Flexibilität in Bezug auf Integration, Modifikation und Erweiterbarkeit im Zusammenspiel von Visualisierungen und Datenmodell bieten. Grundlegende Schnittstellendefinitionen sollen den Anforderungen eines möglichst breiten Spektrums an Visualisierung genügen, wobei Sonderfunktionen einzelner Visualisierungen die Kompatibilität zu den restlichen nicht beeinträchtigen dürfen.

Des Weiteren sollen mit den unterschiedlichsten Visualisierungen auch deren vielfältige Interaktionsmechanismen unterstützt werden. Grundlegend sollen Sortierung, Filterung, Fokussierung, Selektion und Zooming angeboten werden. Die Art der Umsetzung dieser

Funktionalitäten kann in den einzelnen Visualisierungen aber sehr variieren. Deshalb soll eine abstrakte Interaktionsschnittstelle nur einen Rahmen für die jeweilige Umsetzung in den angeschlossenen Visualisierungen bieten.

Getätigte Interaktionen aus einzelnen Visualisierungen können oder müssen Einfluss auf einzelne oder auf alle anderen Visualisierungen ausüben. Das heißt, dass die eben besprochene Interaktionsschnittstelle auch eine Synchronisation zwischen den Visualisierungen untereinander und gegebenenfalls mit dem Datenmodell ermöglichen muss. Die Modularisierung der Komponenten darf dadurch nicht beeinträchtigt werden – um weiterhin eine hohe Unabhängigkeit und Austauschbarkeit gewährleisten zu können, darf die Kommunikation nur über diese eine Schnittstelle stattfinden.

Das MedioVis Framework und damit das integrierte Datenmodell sollen dem Anwender ermöglichen, ein oder mehrere Datenelemente inklusive der zugehörigen Relationen bzw. Hierarchiestrukturen in eine Art parallele, gleichwertige Datenmodell-Instanz zu transferieren und dort beispielsweise als Merkliste zu verwalten. Ein Wechsel zwischen den Instanzen oder das gleichzeitige Anzeigen dieser in verschiedenen Visualisierungen soll direkt in der laufenden Applikation durch Abstraktion und Modularisierung realisierbar sein. Auch sollte eine Synchronisierung der Instanzen bezüglich Brushing & Linking und Selektionsstatus mittels objektspezifischer Hash-Werten<sup>1</sup> vollzogen werden, da vor allem bei flüchtigen Datenquellen (z.B. Yahoo Web-Suche) keine eindeutigen, persistenten Identifikationswerte vorliegen.

Die zur Laufzeit erzeugten zusätzlichen Datenmodell-Instanzen sollen bei Bedarf einschließlich der Datenwerte und Relationen persistent speicherbar sein, damit sie zu einem späteren Zeitpunkt beim erneuten Starten der Anwendung weiter verwendbar sind. Die Schnittstellen müssen daher unabhängig davon, ob es sich um persistente oder temporäre Instanzen handelt, gleiche Funktionen und gleiches Verhalten gegenüber den Visualisierungen und deren Interaktionsmechanismen bieten.

Der beschriebene Funktionsumfang muss weitestgehend ohne Einfluss der anzuzeigenden Datenmenge, also ob 10 oder 10.000 Datenobjekte, und unabhängig von der Multi-dimensionalität der Objekte, ob 2 oder 20 Attribute, bis zu einer gewissen Grenze hin erfüllt sein. Des Weiteren ist die Geschwindigkeit der Funktionsausführung und damit ein schnelles Feedback auf getätigte Benutzeraktionen vor allem bei interaktiven Visualisierungen für das Verständnis des Anwenders von entscheidender Bedeutung.

Hohe Performance bei kompletter Funktionsabdeckung mit der geforderten Flexibilität zu vereinbaren, ähnelt einer Gradwanderung und ist manchmal nur in einem Kompromiss zu finden. Die Kunst liegt hier auch in der richtigen Gewichtung von gegenläufigen Anforderungen und entsprechender Kompromissfindung.

---

<sup>1</sup>Hash-Wert: Aus komplexer Datenstruktur mittels Hash-Funktion berechneter, skalarer Wert.

## 3.2 Gesamtstruktur

Das MedioVis Framework und dessen Datenmodell insbesondere setzen sich aus vielen zumeist unabhängigen Modulen zusammen. Im Folgenden werden die einzelnen Komponenten des Datenmodells und die angrenzenden Funktionseinheiten kurz aufgeführt und in den Gesamtzusammenhang gestellt.

Abstrahiert kann das Framework in vier logische Schichten – Datenquellen, Datenanfrage & Input, Datenmodell und Visualisierungen & Interaktion – eingeteilt werden (siehe Abbildung 3.1). Jede der vier Schichten besitzt zu der nächst höheren bzw. niedrigeren definierte Schnittstellen, welche es erlauben, einzelne Schichten sogar in der laufenden Anwendung auszutauschen, ohne dabei die restlichen Schichten zu beeinträchtigen.

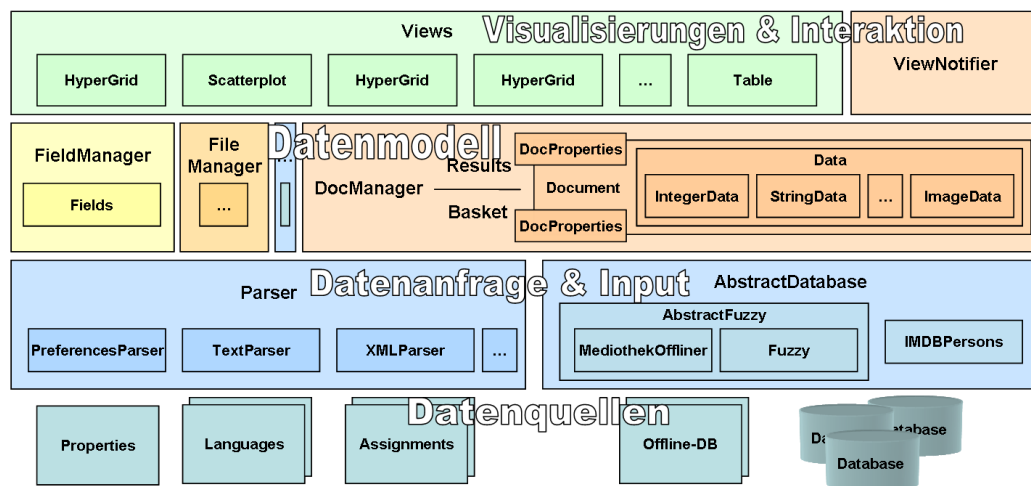


Abbildung 3.1: Schematischer Aufbau des MedioVis Frameworks.

Ausgangsbasis für die Visualisierungen sind natürlicherweise die darzustellenden Daten. Zu der Datenquellen-Schicht gehören verschiedenste Datenbanken und CSV-Dateien, welche die eigentlichen Daten und deren Metadaten beinhalten. Des Weiteren sind hier in XML<sup>2</sup> formatierte Zuordnungsdateien zu finden, welche die Anordnung der Datenattribute je nach Datenquelle auf die MedioVis Visualisierungen regelt. Ebenfalls liegen in dieser Schicht noch Eigenschaftsdateien und Sprachpakete vor, welche die Startparameter der Anwendung und die Dialoggestaltung definieren. Die angesprochenen Dateien können aus der Verzeichnisstruktur, aus einer gepackten JAR-Datei<sup>3</sup> oder über das Internet von einem Server gelesen werden.

<sup>2</sup>XML: EXTensible Markup Language ist ein Standard zur Erstellung von maschinen- und menschenlesbarer Dokumente in Form einer Baumstruktur.

<sup>3</sup>JAR: Java ARchive Dateien sind mit Metainformationen angereicherte, komprimierte Archivdateien, welche direkt als Applikation gestartet oder als Applet in eine Webseite eingebunden werden können.

Die zweite Schicht „Datenanfrage & Input“ besteht aus mehreren Modulen, welche sich mit den in der vorherigen Schicht aufgeführten Datenquellen verbinden, alle oder spezifische Daten auslesen und interpretieren. Hierfür stehen für die CSV- und XML-Dateien Parser und für die Datenbanken abstrakte Verbindungsklassen bereit. Je nach angeschlossener Datenbank kann in den Verbindungsklassen der jeweilige Treiber eingebunden und Abfrage- bzw. Einleseoptimierungen vorgenommen werden. Des Weiteren besteht hier auch die Möglichkeit anstatt der konventionellen datenbankseitigen Suche Fuzzearch als Retrievalkomponente einzubinden. Fuzzearch wurde in Kooperation mit der AG Datenbanken und Informationssysteme der Universität Konstanz entwickelt und bietet eine sehr effiziente, exakte oder auch unscharfe Suche auf großen textuellen Datenmengen anhand von Indizes und der modifizierte Levenshtein-Distanz<sup>4</sup>.

Das Datenmodell stellt die dritte logische Schicht des MedioVis Frameworks dar. Hier werden die Daten und ihre Relationen gespeichert, organisiert und verwaltet. Das Datenmodell ist konsistent objektorientiert umgesetzt. Der abstrakte Datentyp Data als Basistyp wird durch die konkreten Typen IntegerData, StringData, ImageData und weitere überladen. Diese werden in Instanzen vom Typ Document in der im FieldManager definierten Reihenfolge in Arrays gespeichert. Die Documents werden wiederum in einer Instanz der Klasse DocManager verwaltet, welche auch konkrete Methoden für die Selektion und das Filtering bereitstellt. In den nächsten Kapiteln werden die Komponenten des Datenmodells und ihr Zusammenspiel nochmals ausgiebig thematisiert.

Die oberste Schicht „Visualisierungen & Interaktion“ stellt die Benutzeroberfläche und damit die Schnittstelle zum Menschen dar. Hier werden die Daten visuell und textuell kommuniziert, Interaktionen angeboten, entgegengenommen und gegebenenfalls eine Synchronisation zwischen den einzelnen Komponenten hervorgerufen. Nach dem Observer Pattern implementiert jede MedioVis Visualisierung das Interface „Views“, welches Aufrufe für die Initialisierung, Aktualisierung und Statussynchronisierung beinhaltet. Weiterhin melden sich die Visualisierungen auch beim „ViewNotifier“ an, welcher bei einem relevanten Ereignis alle registrierten Visualisierungen benachrichtigt und die konkretisierten Views-Methoden ausführt. Somit können bei völliger Unabhängigkeit der Visualisierungen trotzdem Modul-übergreifende Mechanismen wie Brushing & Linking und Details on Demand in einer Multiple Coordinated Views Umgebung [NS00] realisiert werden.

---

<sup>4</sup>Vladimir Iossifowitsch Levenshtein, russischer Mathematiker, berühmt durch die 1965 von ihm erfundene Levenshtein-Distanz.

### 3.3 Data - abstrakter Datentyp

Die Klasse Data definiert einen abstrakten Basistyp, welcher im Sinne der Objektorientierung als Wurzel für die verschiedenen konkreten Datentypen dient. Hier werden grundlegende Methoden abstrakt deklariert, welche von den abgeleiteten Klassen konkret implementiert werden müssen. Dadurch kann eine Basis-Kompatibilität für die Methoden „isEmpty():boolean“, „toString():String“ und „compareTo(Object):int“ gewährleistet werden. Somit ist jede Instanz eines abgeleiteten Datentyps mit dem gleichen oder einem anderen Typ vergleichbar.

Abgeleitete Typen sind StringData (Textdaten), IntegerData (Ganzzahlen), BooleanData (Wahrheitswerte), ImageData (Bildaten) und Weitere. Diese konkretisierten Datentypen können spezifische Methoden enthalten, welche nur für diese relevant sind. Zum Beispiel kann bei ImageData mit „get():void“ das Bild vom Server geladen und mit „loaded():boolean“ abgefragt werden, ob der Ladevorgang schon erfolgreich abgeschlossen wurde.

Jeder Datenwert in MedioVis ist in eine Instanz eines von Data abgeleiteten Datentyps gespeichert. Diese werden wiederum in einem Objekt der Klasse Document verwaltet.

### 3.4 Document - Dokumentenobjekt

Im Bibliotheksszenario stellt die einzelne DVD, CD oder ein Buch die kleinste logische Einheit dar. Ein reelles Exemplar entspricht im MedioVis Datenmodell einer Instanz der Klasse Document<sup>5</sup>. Die Metainformationen zu dem Exemplar, also die verschiedenen Attribute werden in einem Array des abstrakten Typs Data gespeichert. Somit können beliebige von Data abgeleitete Instanzen in dem Document innerhalb eines Arrays und über eine gemeinsame Schnittstelle verwaltet werden. Die Reihenfolge der Attribute im besagten Array ist durch den FieldManager gegeben und über alle Documents hinweg identisch.

Beispielsweise würde in Abbildung 3.2 ein Document eine ganze Zeile repräsentieren und der Inhalt der Spalten an der durch den FieldManager spezifizierten Stellen im Data Array des Documents zu finden sein. Der FieldManager hat hier die Aufgabe die Namen der Attribute, z.B. Titel, dem entsprechenden Index im Data Array zuzuordnen und darüber Auskunft zu geben.

---

<sup>5</sup>Anmerkung: Im Folgenden wird mit Document die Klasse Document selbst oder ein instanziiertes Objekt von Document bezeichnet. Das deutsche Wort Dokument wird für ein einzelnes Exemplar des Datenbestandes, z.B. für ein Buch oder DVD, verwendet. Da jedes Dokument durch ein Objekt von Document im Datenmodell repräsentiert wird, ist die Unterscheidung eher theoretischer Natur.

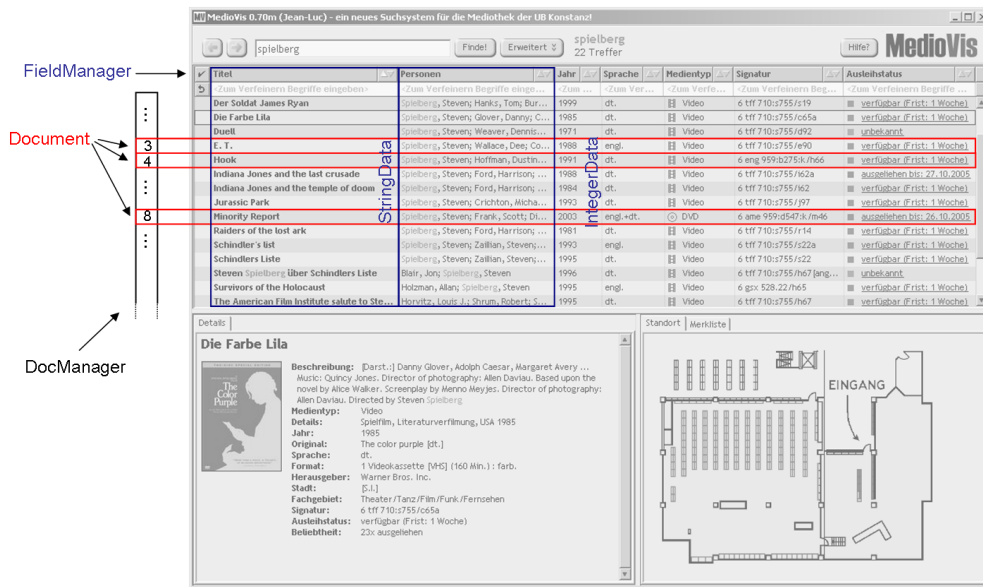


Abbildung 3.2: Zusammenspiel von DocManager, Documents und von Data abgeleiteten Datentypen im MedioVis Datenmodell.

### 3.5 DocManager - Datenverwaltung

Die Referenzen auf die einzelnen Documents werden wiederum in einem Array des Typs Document in einer Instanz des DocManagers gespeichert. Dadurch ergibt sich ein zwei-dimensionaler Aufbau, welcher dem einer Tabelle mit Spalten (Attribute) und Zeilen (Documents) ähnelt, wobei die Zeilennummer bei entsprechender Sortierung dem Index des Documents im Array des DocManagers entspricht.

Vernachlässigen wir kurz den Einfluss der Sortierung, Verfügbarkeit etc. auf die Datenanordnung und schauen uns nochmals die Abbildung 3.2 an. Möchte man zum Beispiel das Document mit dem Filmtitel „Hook“ erhalten, könnte man im Array des DocManagers das Document mit dem Index „4“ anfordern. Im Code lautet diese Anfrage wie folgt:

```
Document hook = DocManager.get().getDoc(4);
```

Würde man sich nun für das Entstehungsjahr des durch das Objekt „hook“ repräsentierten Exemplars interessieren, müsste man zuerst in Erfahrung bringen, welcher Index dem Attribut „year“ im Data Array zugeordnet ist:

```
int field = FieldManager.getID('year');
```

Mit dieser Information kann nun das Metadatum Jahr vom Typ IntegerData aus der Position „field“ im Data Array ausgelesen werden:

```
IntegerData my_year = hook.getData(field);
```

Die aufgeführten Schritte sind notwendig, da jedes Document mehrere Attributwerte an spezifischer Position beinhalten kann und diese wieder in beliebiger Anzahl im DocManager gespeichert werden.

## 3.6 Unabhängigkeit des Datenmodells

Nun fällt vielleicht auf, dass beim DocManager im ersten Code-Beispiel über die statische Methode `get():DocManager` die aktuelle Instanz des DocManagers angefordert wurde. Dieser Zwischenschritt ermöglicht mehrere Instanzen des Datenmodells parallel zu halten. Zum Beispiel kann so die Merkliste im Bibliotheksszenario realisiert werden, welche im Gegensatz zu den Suchergebnissen in der TreeTable nicht bei jeder Suche erneut befüllt wird. Das heißt, während der Laufzeit werden die Instanzen der Datenmodelle ausgetauscht, Documents untereinander transferiert und komplette Instanzen geleert und neu befüllt. Trotz dieser tief greifenden Änderungen der Daten können die Visualisierungen über die gleichen Schnittstellen auf die Datenobjekte zugreifen – kommunizieren, aber gegebenenfalls mit unterschiedlichen Instanzen des Datenmodells.

```
public static DocManager get() {
    return get(currentMode);
}

public static DocManager get(int mode) {
    if(manager[mode] == null) {
        manager[mode] = new DocManager(mode);
    }
    return manager[mode];
}

public static void setViewMode(int mode) {
    currentMode = mode;
}
```

Der Code-Ausschnitt der Klasse DocManager führt die modifizierten getter- und setter-Methoden auf, welche für die Verwaltung der Instanzen des Datenmodells zuständig

sind. Mit der Methode `get():DocManager` wird immer das aktuelle oder beim Start ein erstes Datenmodell erzeugt. Will man auf ein spezifisches Datenmodell zugreifen ist dies über die Methode `get(int):DocManager` möglich. Hierbei wird eine neue Instanz erstellt, wenn diese noch nicht existiert oder andernfalls einfach zurückgegeben. Mit `setViewMode(int):void` kann global die Datenmodell Instanz gewechselt werden.

## 3.7 Serialisierung des Datenmodells

In Kapitel 3.1 wurde die Anforderung gestellt, dass bestimmte Instanzen des Datenmodells auch über die Laufzeit der Anwendung hinweg zur Verfügung stehen sollen. Durch den objektorientierten Ansatz der Datenmodellverwaltung sind nur kleine Änderungen notwendig, um die Speicherung und das erneute Einlesen von serialisierten Objekten zu ermöglichen. Im Folgenden ist die Methode `get(int):DocManager` der Klasse `DocManager` in modifizierter Form zu sehen, wobei bei dieser Variante die Instanz der Merkliste (BASKET) serialisiert vorliegt und wieder integriert werden soll. Wird nun die Instanz BASKET angefordert und existiert diese noch nicht im Arbeitsspeicher, so wird diese aus der Verzeichnisstruktur ausgelesen und bei Erfolg eingebunden. Existiert keine serialisierte Instanz des BASKETs im Verzeichnis oder ist diese korrupt, so wird eine neue Instanz erzeugt.

```
public static DocManager get(int mode) {
    if(manager[mode] == null) {
        if(mode == BASKET) {
            manager[BASKET] = (DocManager)
                FileManager.deserializeObj(Preferences.DB);
            if(manager[BASKET] != null) {
                return manager[BASKET];
            }
        }
        manager[mode] = new DocManager(mode);
    }
    return manager[mode];
}
```

## 3.8 Hash Identifikation

Stehen verschiedene Datenmodell Instanzen zur Verfügung, so ist gerade für Brushing & Linking Konzepte ein Mechanismus wichtig, welcher überprüft, ob in zwei zu vergleichenden Instanzen gleiche Documents Objekte existieren oder nicht. Stammen die Documents aus flüchtigen Datenquellen, z.B. Nachrichten aus dem Web, so wird zu meist kein Schlüsselwert, also kein eindeutiger Identifikator mitgeliefert. Der Vergleich von Documents gestaltet sich in diesem Falle schwierig, zumal die Zeitpunkte der Objektgenerierung durch eine mögliche Serialisierung sehr differieren können.

Aufgrund dieser Problematik besitzt im MedioVis Datenmodell jedes Document einen eindeutigen Hash-Wert, welcher sich aus mehreren Hash-Werten, der im Assignment Editor spezifizierten Attributausprägungen des jeweiligen Documents, zusammensetzt. Mittels der eindeutigen Hash-Werte kann somit jedes einzelne Document unabhängig vom Generierungszeitpunkt und einer etwaigen Serialisierung identifiziert und verglichen werden.

## 3.9 Hierarchisierung des Datenmodells

Bei der Betrachtung des soweit beschrieben Datenmodells stellt sich die Frage, wie hierarchische Strukturen im Datenmodell sinnerhaltend abgebildet werden können. Der objektorientierte Ansatz des geschilderten Datenmodells liefert zur Lösung der Problematik eine gute Ausgangsbasis. Die Documents Objekte sind bisher linear in Instanzen des DocManagers angeordnet. Die Klasse Documents wird nun durch einen Array des Typs Documents erweitert, welcher Referenzen auf weitere Documents als 1:n Relationen verwaltet (siehe Code-Fragment Array „childs“). Die referenzierten Documents können wiederum andere Documents referenzieren. Durch die Schachtelung der Documents kann jede beliebige Baumstruktur bzw. Hierarchie im Datenmodell abgebildet werden (siehe Abbildung 3.3).

```
public class Document implements Serializable {
    private String hash;
    private Data[] data;
    private Document[] childs;
    private Document parent = null;
    private DocProperties[] docProps;
    private boolean focused = false;
    ...
}
```

Um performante Zugriffe auf die Hierarchien zu ermöglichen, trägt das Document nicht nur den Array mit den Referenzen zu den Kind-Knoten, sondern auch eine Referenz zu dem Vater-Knoten (siehe Code-Fragment Variable „parent“). Somit kann auch vom Blattknoten aus auf dessen Wurzel in linearer Zeit zugegriffen werden.

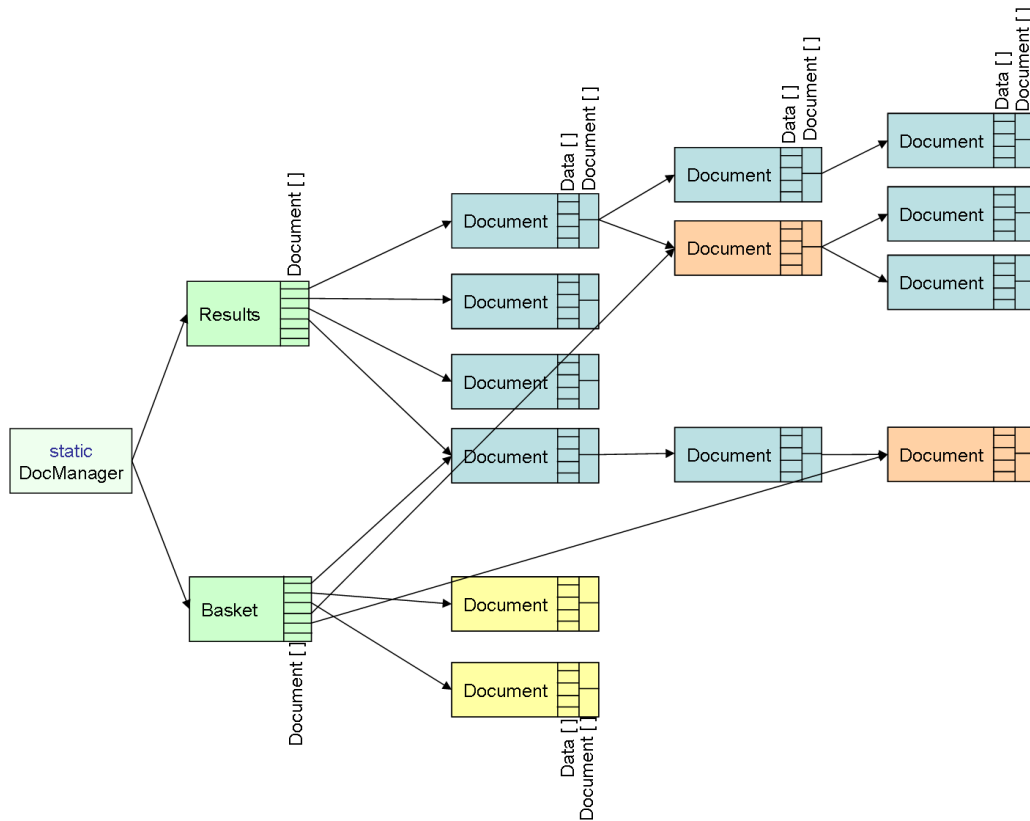


Abbildung 3.3: Schachtelung und Referenzierung von Documents Objekten zur Abbildung von Hierarchien in parallelen DocManager Instanzen.

Ein zusätzlicher Vorteil, welcher aus der Hierarchisierung mittels Referenzen resultiert, zeigt sich, wenn Knoten zwischen Instanzen des Datenmodells transferiert werden sollen. In Abbildung 3.3 sind die parallelen Datenmodellinstanzen Results (aktuelle Suche) und Basket (Merkliste) dargestellt. Wenn ein Document aus Results auch im Basket visualisiert werden soll, muss nur im Basket eine Referenz auf diesen Knoten erzeugt werden. Der entscheidende Punkt ist, dass über diese Referenz auch auf alle Unterknoten des in den Basket übertragenen Documents zugegriffen werden kann. Auch der Pfad bis zum Wurzelknoten bleibt erhalten, da jedes Document eine Referenz auf den Vaterknoten vorhält. Selbst wenn Results bei einer erneuten Suche überschrieben oder durch Serialisierung nicht mehr existent ist, bleiben die referenzierten Documents mit allen Unterhierarchien inklusive Wurzelfpfad im Basket erhalten. Würde man sich nicht der Referenzen bedienen, sondern Kopien der Documents anlegen, müssten auch alle untergeordneten Knoten und Vaterknoten jeweils kopiert und neu referenziert werden.

Operationen wie die Suche, Sortierung und das Filtern können bei hierarchischen Strukturen mittels Rekursion sehr elegant durchgeführt werden. Als Invariante<sup>6</sup> der Rekursion dient zumeist die Länge des „childs“-Arrays bzw. dessen Existenz, d.h. die Rekursion endet bei Erreichen des Blattknotens.

Das folgende Code-Fragment mit der Methode `searchByHash(String):Document` der Klasse `Documents` zeigt exemplarisch eine Suche nach einem mittels Hash-Wert spezifizierten `Document` über alle Knoten, welche dem aktuellen Objekt hierarchisch untergeordnet sind. Hierbei werden nach dem Prinzip der Tiefensuche (`depth-first search`) rekursiv immer zuerst die Kindknoten untersucht, bevor die Knoten der gleichen Ebene verglichen werden. Die aufgeführte Methode wird vor allem bei der Synchronisierung von verschiedenen Datenmodell Instanzen verwendet, wobei im Falle des Fundes, das entsprechende `Document` hervorgehoben oder markiert wird (z.B. `Brushing & Linking` oder `Fokussierung`).

```
public Document searchByHash(String hash) {
    if (this.getHash().equals(hash)) {
        return this;
    } else {
        if (childs == null) {
            return null;
        } else {
            for (int i = 0; i < childs.length; i++) {
                Document sub = childs[i].searchByHash(hash);
                if (sub != null) {
                    return sub;
                }
            }
            return null;
        }
    }
}
```

Die lineare Vorgehensweise der überwiegenden Anzahl von Visualisierungen des MedioVis Frameworks impliziert die Notwendigkeit, über eine Methode einen linear iterativen Zugang zu den hierarchisch angelegten Daten zu erhalten. Beispielsweise baut die `TreeTable` die Darstellung zeilenbasiert auf, d.h. es werden die Daten zeilenweise ausgelesen und gezeichnet.

---

<sup>6</sup>Eine (Schleifen)-Invariante definiert bei Rekursionen die Abbruchbedingung, ohne welche ein infinites Regress bzw. eine Endlosschleife auftreten würde.

Hierfür bietet die Methode `getAllDocuments():Document[]` eine einfache Schnittstelle, da sie die untergeordneten Hierarchien auf einen linearen Array flachdrückt. Hinzu kommt, dass bei der Sortierung die Hierarchien umsortiert werden und daher die Documents im linearen Array schon in der richtigen Reihenfolge vorliegen.

Die Methode ist wieder rekursiv angelegt und vereinigt auf jeder Hierarchieebene die einzelnen Teil-Arrays bis zuletzt die Hierarchien linear in einem Gesamt-Array gebündelt zurückgegeben werden.

```
public Document[] getAllDocuments() {
    if (childs == null) {
        return new Document[] { this };
    } else {
        Document[] tmpAll = new Document[] { this };
        for (int i = 0; i < childs.length; i++) {
            Document[] tmp = childs[i].getAllDocuments();
            if(tmp.length==0) continue;
            Document[] tmp2 = new Document[tmp.length + tmpAll.length];
            System.arraycopy(tmpAll, 0, tmp2, 0, tmpAll.length);
            System.arraycopy(tmp, 0, tmp2, tmpAll.length, tmp.length);
            tmpAll = (Document[]) tmp2;
        }
        return tmpAll;
    }
}
```

Die in diesem Code-Fragment verwendete System-Methode `System.arraycopy(Object[], int, int, int, int):Object[]` stellt eine sehr performante Methode dar, Arrays zu kopieren. Man könnte das vielleicht subjektiv „unschöne“ Kopieren durch die Verwendung von Vektoren umgehen, müsste aber dabei mit einem erheblichen Performanceverlust rechnen.

Insgesamt wurde bei der Entwicklung des MedioVis Frameworks hohen Wert auf eine Verbindung von sauberem Architektur-Design und bestmöglicher Performance gelegt. Dies bestätigt sich einerseits in der Flexibilität des Systems in Bezug auf Anpassungen und Erweiterungen und andererseits in den minimalen Anforderungen an die Hardware. Trotz der interaktiven Visualisierungen und der ausgiebigen Verwendung von Animationen läuft MedioVis auch noch auf Desktop-PCs angenehm flüssig, welche vor drei bis vier Jahren zum Standard<sup>7</sup> gehörten. Für den reellen Einsatz von MedioVis in der

<sup>7</sup>Mindestanforderung: Pentium III 600 MHz, 256 MB RAM, Grafikkarte incl. 8 MB RAM, Online-Verbindung oder 50 MB Speicherplatz, Java Runtime Environment 1.4

Universitätsbibliothek Konstanz ist diese Hardware-Unabhängigkeit von erschreckender Bedeutung, da die PC-Arbeitsplätze der Bibliothek die komplette Bandbreite der Computertechnik der letzten Jahre bietet.

## 3.10 Sortierung

Die Documents sollen bezüglich eines spezifizierten Feldes (Spalte) jeweils innerhalb der Hierarchieebenen, auf- bzw. absteigend sortiert werden. Die Hierarchieebenen bleiben dabei in sich geschlossen.

Der Array „DOCS“ des DocManagers, welcher die Documents der obersten Ebene enthält, kann nicht direkt umsortiert werden, da die Indizes intern auch zur Identifikation genutzt werden. Daher wird ein Integer Array „sortedIDs“ erzeugt, welcher in der Sortierreihenfolge den jeweiligen Index des entsprechenden Documents enthält.

```
public class DocManager implements DataTypes, Serializable {
    private Document[] DOCS;
    private int[] sortedIDs;
    private int[] sortedRows;
    ...
}
```

Ein Document, welches in DOCS an dritter Stelle gespeichert ist, sollte nach der Sortierung zum Beispiel an siebter Stelle stehen, weshalb `sortedIDs[7]=3` gilt. Die folgende Methode `example():void` zeigt beispielsweise, wie man die Documents in Sortierreihenfolge mithilfe der Methode `getID(int):int` aus der Klasse DocManager ausgeben kann:

```
private void example(){
    for(int i=0;i<DOCS.length();i++){
        System.out.println('row '+i+ ' = '+ DOCS[ getID(i) ].toString );
    }
}
public int getID(int row) {
    return sortedIDs[row];
}
public int getRow(int id) {
    return sortedRows[id];
}
}
```

In manchen Fällen (z.B. Fokus) möchte man zu einer Zeile in der Visualisierung das Document bzw. den entsprechenden Index für DOCS geliefert bekommen. Hierfür wird ein weiterer Integer Array `sortedRows` verwaltet, welcher für jeden Index in DOCS die momentane Zeilennummer speichert. Im vorherigen Beispiel würde dann `sortedRows[3]=7` gelten. Als Schnittstelle steht die Methode `getRow(int):int` in der Klasse `DocManager` zur Verfügung, mit der zu einem Index in DOCS bzw. Document die zugeteilte Zeilennummer in Erfahrung gebracht werden kann.

Die beiden Integer Arrays werden während des Sortierens innerhalb des Sortieralgorithmus `MergeSort`<sup>8</sup> jeweils neu gesetzt. Im Grunde könnte man sagen, dass die beiden Arrays die gleichen Daten beinhalten, nur fungiert die sortierte Position des Documents bei `sortedIDs` als Index und bei `sortedRows` als Wert. Theoretisch könnte man auf den `sortedRows` Array verzichten und dessen Funktionalität durch einen zusätzlichen Schleifendurchlauf über alle Documents ersetzen. Damit würde man minimal Arbeitsspeicher sparen, aber bei einem Fokuswechsel oder ähnlichem würde die Zeilenzuordnung anstatt einer konstanten Laufzeit von  $O(1)$ , eine lineare Laufzeit von  $O(n)$  benötigen. Es ist klar, dass im Sinne der Skalierbarkeit die Variante mit konstanter Laufzeit präferiert wird.

Im Gegensatz zur Sortierung der obersten Ebene können die untergeordneten Hierarchien in sich relativ leicht durch einen rekursiven Ansatz sortiert werden. Dabei wird zuerst der Array mit den Referenzen anhand eines Vergleichskonstrukts sortiert und danach für jeden Kindknoten erneut die Sortierung angestoßen.

```
public void sortSubs(int field, int order) {
    if(childs == null || childs.length == 0) return;
    Arrays.sort(childs, new DocComparator(field, order));
    for(int i=0;i<childs.length;i++){
        childs[i].sortSubs(field, order);
    }
}
```

### 3.11 Filter- und Selektionsstatus

Schränkt der Anwender die Gesamtmenge der darzustellenden Dokumente über den Tabellenfilter (siehe Kapitel 2.5.1) ein, so werden die restlichen Documents nicht etwa gelöscht, sondern nur deaktiviert. Hierfür ist jedem Document ein Status „`isAvailable`“ zugeordnet. Da ein Document von mehreren Datenmodell Instanzen gleichzeitig referen-

---

<sup>8</sup>`Mergesort`: Rekursiver Sortieralgorithmus nach dem Divide and Conquer Prinzip. 1945 erstmals von John v. Neumann veröffentlicht.

ziert werden kann, wird dieser Verfügbarkeitsstatus in eine Eigenschaftsklasse DocProperties ausgelagert. Analog wird mit dem Selektionsstatus verfahren. Pro Datenmodell Instanz besitzt dementsprechend jedes Document ein DocProperties Objekt. Somit kann völlig unabhängig gefiltert und selektiert werden. Beim Zeichnen der Documents wird dann jeweils abgefragt, ob das entsprechende Document für die aktuelle Datenmodell Instanz verfügbar ist und im positiven Falle dann auch dargestellt.

```
public class Document implements Serializable {
    private DocProperties[] docProps;
    ...
    public boolean isSelected(int mode) {
        return docProps[mode].isSelected();
    }
}

public class DocProperties implements Serializable {
    private boolean isAvailable = true;
    private boolean isSelected = false;
    ...
}
```

## 3.12 Fokussierung

Per Definition kann innerhalb des MedioVis Frameworks immer nur ein oder kein Objekt den Fokus besitzen. Dies ergibt sich aus der Tatsache, dass zumindest bei konventionellen Betriebssystemen auch nur ein Pointer (Mauspfeil) existiert und dieser nur auf ein Objekt zeigen kann. Theoretisch könnte also global eine Variable mit dem Hash-Wert des aktuell fokussierten Documents gespeichert werden. Da aber die meisten Visualisierungen beim Zeichnen jedes Documents überprüfen, ob dieses gerade fokussiert ist und damit jeweils einen String-Vergleich (Hash-Identifikation) mit sich bringen würde, ist das Speichern des Selektionsstatus als Boolesche Variable<sup>9</sup> direkt im zugehörigen Document im Hinblick auf die Performance wesentlich günstiger. So ist im letzteren Fall nur ein Zugriff auf eine Variable, im Gegensatz zu zwei Zugriffen und einem String-Vergleich bei der globalen Variante, nötig. Dies scheinen auf den ersten Blick marginale Differenzen zu sein, aber bei Animationen mit 20 oder mehr Zeichenvorgängen pro Sekunde über alle Dokumente und mehrere Visualisierungen hinweg potenzieren sich die Unterschiede und wachsen in ihrer Bedeutung.

---

<sup>9</sup>Boolesche Variablen sind Elemente der booleschen Algebra und nehmen entweder den Wert wahr (true) oder falsch (false) an.

## 4 Zusammenfassung

Das MedioVis Framework wurde über mehrere Iterationen hinweg immer wieder umstrukturiert, angepasst und erweitert, um den verschiedensten Anwendungsdomänen und Anforderungen der Forschungsprojekte gerecht zu werden. Die Tatsache, dass der zu verändernde Codeanteil im Laufe der Zeit immer weiter sank, spricht für die Modularisierung, Flexibilität und den Reifegrad des gesamten Frameworks. Dennoch verlangte das in dieser Arbeit thematisierte Redesign zur Unterstützung von beliebigen Hierarchiestrukturen tief greifende Anpassungen auf allen Ebenen.

Über die eigentlichen Datenobjekte hinaus, tragen nun auch deren Konstellationen bzw. Relationen relevante Informationen, welche interpretiert und verwaltet, aber vor allem auch dem Anwender kommuniziert werden sollen. Von der Datenanfrage über das Datenmodell bis hin zu den Visualisierungen und Interaktionen musste dem zusätzlichen Informationsträger „Relation“ Rechnung getragen werden.

Das durchgehend objektorientierte Datenmodell bildete eine gute Grundlage für die nötigen Anpassungen. Auf Basis der zwei-dimensionalen Organisation von Dokumenten und deren Attribute wurden durch Objekt-Referenzierung sinntragende Hierarchien ermöglicht, welche performant verwaltet und persistent serialisiert werden können. Die konsistente Verwendung von rekursiven Ansätzen lässt Strukturen beliebiger Tiefe und Breite zu, wodurch MedioVis auch für hierarchische Daten außerhalb der Bibliotheksdomäne bestens geeignet ist. Trotz der erheblichen Anpassungen wurde die Kompatibilität zu bestehenden nicht hierarchischen Visualisierungen auf oberster Schicht und konventionellen Datenquellen auf unterster Schicht bewahrt.

Dieses Framework bzw. das Datenmodell als vollkommen oder final zu bezeichnen, wäre sicherlich unzutreffend, da bei einer Anwendung dieser Komplexität für eine Problematik zumeist keine optimale Lösung gefunden werden kann, sondern die richtige Gewichtung von eventuell gegenläufigen Anforderungen entscheidend für einen guten Kompromiss und somit praktikable Lösung ist.

Derzeit ist MedioVis in der Version „Jean-Luc“ 0.70m in der Universitätsbibliothek Konstanz im reellen Einsatz. Bis zum Zeitpunkt der Fertigstellung dieser Arbeit musste noch keine Änderung am umstrukturierten Datenmodell vorgenommen werden.

# Abbildungsverzeichnis

2.1	MedioVis Oberfläche mit TreeTable, Detail- und Standort-Ansicht. Bezüglich „Deutschland 2005“ wurden 80 Suchtreffer gefunden. . . . .	9
2.2	Die Suchtreffer wurden mittels Tabellenfilter auf 10 DVDs eingeschränkt. Die ausgewählten Elemente des expandierten mehrteilige Werkes „Hundert Jahre Deutschland“ werden in der Merkliste geführt. . . . .	10
2.3	TreeTable (obere Hälfte) mit Detail-Ansicht und Merkliste. Die hierarchischen Titel „Il mio...“ (mehrere Exemplare) und „Martin Scorsese...“ (mehrteiliges Werk) sind expandiert und Untertitel z.T selektiert. . . . .	13
2.4	MedioVis mit 55 von 393 mittels Tabellenfilter gefilterten Suchtreffer. . .	15
2.5	Detail-Ansicht (links) und Standort-Ansicht (rechts). . . . .	16
2.6	Druckdialog der Merkliste mit Voransicht. . . . .	17
3.1	Schematischer Aufbau des MedioVis Frameworks. . . . .	21
3.2	Zusammenspiel von DocManager, Documents und von Data abgeleiteten Datentypen im MedioVis Datenmodell. . . . .	24
3.3	Schachtelung und Referenzierung von Documents Objekten zur Abbildung von Hierarchien in parallelen DocManager Instanzen. . . . .	28

# Literaturverzeichnis

- [BC87] Richard A. Becker and William S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.
- [BR91] V. R. Basili and H. D. Rombach. Support for comprehensive reuse. *Softw. Eng. J.*, 6(5):303–316, 1991.
- [Eng02] Florian Engster, editor. *Evaluierung des WWW-Angebotes der Bibliothek der Universität Konstanz*. Universität Konstanz, Bibliothek, <http://www.ub.uni-konstanz.de/kops/volltexte/2002/754/index.html>, 2002.
- [GGJ<sup>+</sup>05] Christian Grün, Jens Gerken, Hans-Christian Jetter, Werner König, and Harald Reiterer. Mediovis - a user-centred library metadata browser. In *ECDL*, pages 174–185, 2005.
- [GHRM02] Stefan Göbel, Jörg Haist, Harald Reiterer, and Frank Müller. Invisip: Metadata-based information visualization techniques to access geodata archives and to support the site planning process, 2002.
- [Grü04] Christian Grün, editor. *Entwicklung eines visuellen Metadaten-Browsers für die Mediothek Konstanz. Adaption der VISMEB-Architektur an eine konkrete Anwendungsdomäne*. Universität Konstanz, Fachbereich für Informatik und Informationswissenschaft, 2004.
- [JGK<sup>+</sup>05] Hans-Christian Jetter, Jens Gerken, Werner König, Christian Grün, and Harald Reiterer. Hypergrid - accessing complex information spaces. In *People and Computers XIX - The Bigger Picture, Proceedings of HCI 2005*, volume 1, Edinburgh, UK, 2005. Springer Verlag.
- [KRML03] Peter Klein, Harald Reiterer, Frank Müller, and Tobias Limbach. Metadata visualisation with vismeb. In *IV '03: Proceedings of the Seventh International Conference on Information Visualization*, page 600, Washington, DC, USA, 2003. IEEE Computer Society.

- [NS00] Chris North and Ben Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 128–135, New York, NY, USA, 2000. ACM Press.
- [RMMH00] Harald Reiterer, Gabriela Mußler, Thomas M. Mann, and Siegfried Handschuh. Insyder – an information assistant for business intelligence. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 112–119, New York, NY, USA, 2000. ACM Press.