Thomas Memmel

Implementation einer tabellenbasierten Visualisierung für geo-räumliche Metadaten.

INVISIP Projekt Team

Bachelor Arbeit zur Erlangung des Bachelor Abschlusses

Universität Konstanz, Deutschland

Fachbereich Informatik und Informationswissenschaft

Arbeitsgruppe Informationssysteme

1. Gutachter: Prof. Dr. Harald Reiterer

2. Gutachter: Prof. Dr. Rainer Kuhlen

Version vom 26.08.2002

INVISIP

Implementation of a table-based visualization for geo-spatial metadata.

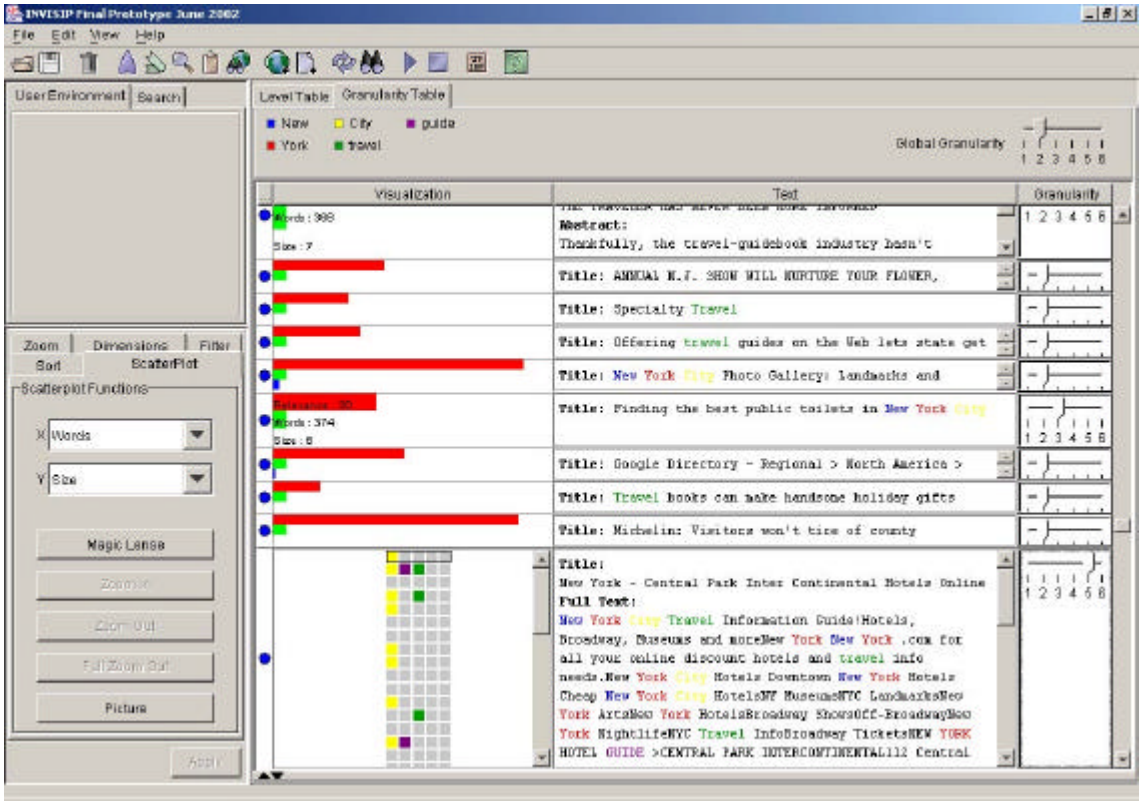

Abbildung 1: INVISIP GranularityTable, Prototyp Version vom Juni 2002

Abstract (Deutsch)

Diese Bachelor-Arbeit dokumentiert die Implementation einer tabellenbasierten Visualisierung für geo-räumliche Metadaten nach der konzeptuellen Redesign Idee von M. Eibl [EM 2001]. Die Eiblsche Designidee wird als Granularitätskonzept bezeichnet und im Rahmen des Prototyping für das INVISIP Projekt, dem praktischen Teil dieser Bachelor-Arbeit, in der Programmiersprache JAVA umgesetzt. Diese Arbeit ist in einen einleitenden Teil und vier Hauptteile gegliedert. Die Einleitung skizziert den thematischen Hintergrund des INVISIP Projektes und stellt geo-räumliche Metadaten in den Zusammenhang zu Geoinformations- und Information Retrieval Systemen.

Der erste Hauptteil beschreibt die bisherigen und aktuellen Projekte für die Visualisierung von Metadaten, fasst deren Intention, Einsatzbereich und Schwächen zusammen und leitet über zu Ideen wie dem Eiblschen Granularitätskonzept, welches als ein Lösungsvorschlag die Probleme älterer Prototypen beheben soll.

Das Pflichtenheft zum praktischen Teil dieser Bachelor-Arbeit stellt den zweiten Hauptteil dar und fasst die Aufgabenstellung für den Implementationsteil zusammen, die sich aus der Evaluation der alten Prototypen und dem Eiblschen Designvorschlag ergibt.

Der dritte Hauptteil beschreibt die ersten praktischen Implementationsarbeiten und die Umsetzung eines Scatterplot als weiteres Visualisierungskonzept in einem begleitenden Projekt [GF 2002], welches mit der praktischen Lösung des Granularitätskonzeptes eng verbunden ist. Im weiteren werden die Überlegungen bei der Planung des Softwareprojektes erklärt. Es werden notwendige Design Patterns und Konzepte der objektorientierten Programmierung in JAVA vorgestellt, das Projekt in den Kontext zu Modellen der Information Visualization gestellt und die Visualisierungsmethoden des tabellenbasierten Granularitätskonzeptes beleuchtet.

Die praktische Umsetzung des Redesign Vorschlages mit Hilfe von Diagrammen und Auszügen aus dem Quellcode der wichtigsten Programmteile wird im vierten Hauptteil dokumentiert. Dieser Teil beschreibt, wie die im dritten Hauptteil erfassten Konzepte softwaretechnisch umgesetzt werden.

Die Arbeit schließt mit einer Zusammenfassung der Probleme bei der Implementation des Eiblschen Granularitätskonzeptes und liefert einen Ausblick auf erforderliche Lösungen und weitere Aufgabenstellungen.

Abstract (Englisch)

This bachelor thesis describes the implementation of a table-based visualization for geo-spatial metadata, following a redesign proposal by M. Eibl [EM 2001]. The design approach is called the concept of granularity, which is implemented in the practical part of this bachelor thesis using the programming language JAVA. The thesis is divided in an introductorily part and four main parts.

Dealing with the background of the INVSISP project, the introduction brings geo-spatial metadata together with information retrieval and geographical information systems.

The first main part presents earlier and recent projects concerning the visualization of metadata and sums up these project's intentions, context of use and weaknesses. This section then leads up to the concept of granularity, which is supposed to solve the evaluated problems of former projects.

The product requirement specification is stated in the second main part. It summarizes the required implementations for overcoming the identified problems and converting the granularity concept.

Main part three describes the first prototyping issues and also points out the implementation of a Scatterplot visualization, done at the same time by [GF 2002] and deeply interconnected to the implemented table-based concept of granularity. It explains the considerations taken place during the preliminary discussions of the project. The considerations deal with the use of software design patterns, some principles of object-oriented programming in JAVA, puts the project into context to basic concepts of information visualization and describes the methods of visualization used in the project.

A documentation of programming the granularity concept as JAVA software project is delivered in main part four. It illustrates the most important parts of the program with the help of unified model language (UML) diagrams and extractions of the project's source code.

This thesis ends with a summary of the problems and obstacles which occurred during implementation and provides an outlook on future tasks.

Danksagung

Diese Arbeit ist meinen Eltern Gisela und Karl-Heinz Memmel gewidmet, die mich zu jedem Zeitpunkt moralisch, finanziell und mit Vertrauen und Respekt dabei unterstützt haben, meine eigenen Ziele zu verfolgen und die mich gelehrt haben, dass man mit Verstand und Ehrgeiz alles erreichen kann.

Außerdem bedanke ich mich bei Jennifer Göpfert, die mich in Höhen und Tiefen meines Studiums mit Liebe begleitet und mir den Rücken gestärkt hat.

Besonderer Dank gilt meinem Teamkollegen Fredrik Gundelsweiler, dessen besondere Charaktereigenschaften wie Ehrlichkeit und Verlässlichkeit, aber auch sein Sinn für Humor, ein Eckpfeiler bei der Bewältigung des Bachelor Studiums Information Engineering gewesen sind. Ich hoffe auf weitere so ergiebige Jahre mit ihm im Master Studium.

Vielen Dank.

Vorwort

„The power of the unaided mind is highly overrated. Without external aids, memory, thought, and reasoning are all constrained. But hum intelligence is highly flexible and adaptive, superb at inventing procedures and objects that overcome its own limits. The real powers come from devising external aids that enhance cognitive abilities. How have we increased memory, thought and reasoning? By the invention of external aids: It is things that make us smart."[1]

Donald Norman beschreibt mit dieser Aussage die Fähigkeit des Menschen, die eigenen kognitiven Kapazitäten zu erhöhen. Die dazu notwendige Unterstützung erhält der menschliche Intellekt in Form von externen Hilfestellungen. Diese ermöglichen es, bestimmte Erkenntnisse und Ergebnisse außerhalb des Verstandes abzuspeichern und entlasten so den Menschen. Einfaches Beispiel ist die Verwendung einer externen Hilfe in Form von Stift und Zettel zur Berechnung einer Multiplikation. Diese Methode ist bis zu fünf mal schneller als Kopfrechnung[2]. Durch das Anschreiben von Zwischenergebnissen bei der Berechnung einer mathematischen Aufgabe werden visuelle Adressräume geschaffen, die die Zugriffszeit auf Information verringern. Eine Aufgabenstellung, die zunächst Anforderungen an die Gedächtnisleistung stellt, wird so in eine Anforderung an die visuelle Informationsverarbeitung umgewandelt.

Eine wichtige Klasse visueller Hilfen sind Diagramme. Sie können für den Anwender eine sehr große Unterstützung sein, jedoch können schlechte Diagramme auch Information verbergen oder Fehlinterpretationen provozieren. Die Verwendung eines gut gestalteten Diagramms hätte beispielsweise den Unfall der Raumfähre Challenger im Jahr 1986 verhindern können. Der Hersteller der Startraketen hatte ein Diagramm verwendet, in dem die wichtigsten Variablen wie Temperatur und Beschädigungen der Raketen bei vorigen Starts nur unzureichend visualisiert waren. So wurden relevante Indikatoren für Gefahren beim Shuttlestart [NHM 1997] nicht identifiziert. Eine Darstellung in einem Scatterplot hingegen hätte leicht erkennen lassen, dass bei vorigen Raketenstarts unterhalb von 65° Fahrenheit Außentemperatur, signifikante Fehler an den Halterungen der Startraketen festgestellt wurden [TR 1997]. Die Ingenieure hätten auf einen Start, bei Temperaturen weit unter dieser Marke, verzichtet.

Visualisierungen spielen eine zunehmend größere Rolle, wenn der Mensch immer mehr Daten verarbeiten muss und aus Daten relevante Informationen extrahieren muss. Um

[1] Norman, Donald A. – „**Things that make us smart**" – Addison Wesley, 1993. Originaltext in Englisch.
[2] Aus: [CMS 1999], Seite 2.

den Benutzer mental zu entlasten und die Erkennung von Ergebnissen und Mustern zu beschleunigen, können graphische Darstellungen auch in Softwareprodukten verwendet werden. Computeranwendungen wie Geoinformationssysteme können mit Visualisierungen Daten und Metainformationen über Geo-Objekte für den Benutzer in vorteilhafter Form aufbereiten. Durch geeignete Visualisierungsmethoden kann sowohl die Benutzbarkeit der Softwareanwendung, als auch die Effektivität des Retrievalprozesses gesteigert werden.

Das INVISIP Projekt ist die Erschaffung eines Softwareproduktes für die geo-räumliche Ortsplanung und somit die Konzeption eines Geoinformationssystems, welches umfangreiche Datenbestände mit Hilfe bewährter, aber auch neuartiger Konzepte der Informationsvisualisierung darstellen soll.

Inhalt

1. Einleitung

1.1. Geoinformationssysteme

„Ein Geoinformationssystem (GIS) ist ein DV-gestütztes Informationssystem zur Erfassung, Verwaltung, Analyse, Modellierung und Visualisierung von Geoinformationen (Abbildung 2). Die zugrundeliegenden Geodaten beschreiben die Geometrie, Topologie, Thematik und Dynamik der Geoobjekte“[3]. Für die Analyse, Modellierung und Visualisierung raumbezogener Daten sind GIS heute das dominierende Instrument, obwohl derartige Systeme (Abbildung 2) sehr hohe Ansprüche an die Hardware stellen. Viel Rechenleistung und ausreichend Hauptspeicher werden benötigt, um komplexe Operationen auf großen Datenmengen ausführen zu können.

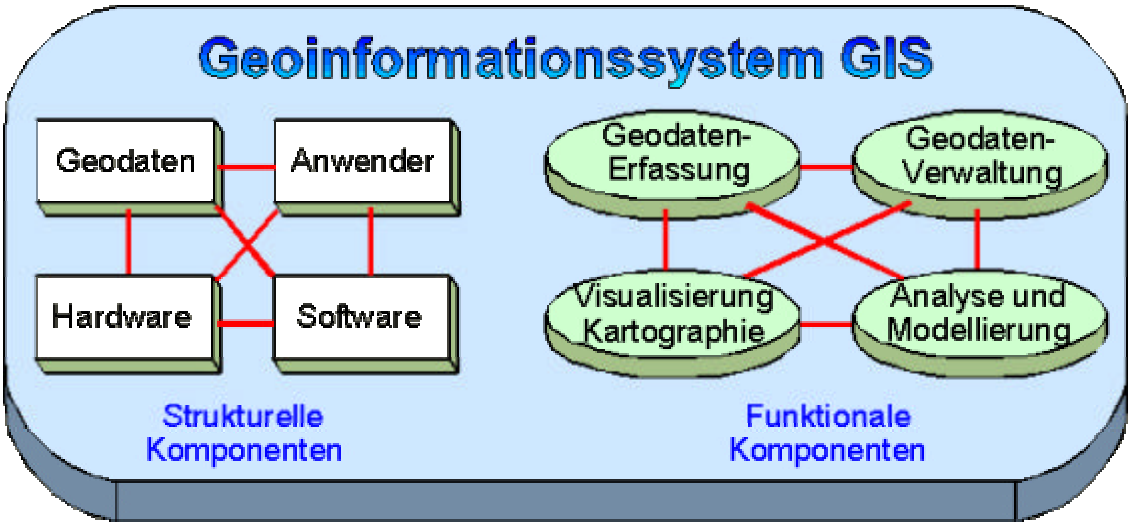

Abbildung 2: Strukturelle und funktionale Komponenten von GIS

Im Unterschied zu einem normalen Datenbanksystem (Tabelle 1), werden dem Anwender in einem Geoinformationssystem interaktive Auswertungsmethoden und Visualisierungen zur Verfügung gestellt. Eine benutzerfreundliche Gestaltung des Geoinformationssystems, insbesondere der Benutzeroberfläche, ist dabei von entscheidender Bedeutung für den Erfolg einer solchen Applikation. Dabei müssen auch

[3] Streit, U. – „**Einführung in die Geoinformatik**“, http://castafiore.uni-muenster.de/vorlesungen/geoinformatik /kap/kap9/k09_01.htm#9.1, online am 10.08.2002.

komplexe Funktionen der Anwendung mit Hilfe grafisch-interaktiver Bedienung leicht ausführbar sein und eine verständliche, wohl strukturierte Onlinehilfe zur Verfügung stehen. Beim Abruf numerischer, textueller, bildhafter und multimedialer Informationen werden Metadaten intensiv zur Recherche in den komplexen Datenbeständen genutzt.

Unterschiede	
Geoinformationssystem	**Datenbanksystem**
Geoobjekte mit explizitem Raumbezug und gekoppelten Sachdaten	Modellierung allgemeiner Objekte, Raumbezug nur als ein Attribut
Selektion von Geoobjekten über Raumbezug und Attribute möglich	Selektion von Objekten nur über Attribute (z.B. Schlüssel) möglich
Datenanalyse interaktiv-grafisch, numerisch - statistisch	Datenanalyse überwiegend mit statistischen Methoden
Visualisierung mit digitaler Kartographie, Tabellen und Diagrammen	Visualisierung durch Tabellen und Diagramme (Business-Grafik)

Tabelle 1: Unterschiede zwischen Geoinformationssystem und Datenbanksystem

1.2. Georäumliche Metadaten

Grundsätzlich sind alle Formen von Informationen, Zusammenfassungen, Indexierungstermen und Abstracts, die die ursprünglichen Objekte auf eine bestimmte Art beschreiben, Metadaten.

Metadaten werden definiert als Informationen über Daten, die in einer bestimmten Form gehalten werden, dass sie die Recherche, das Retrieval und die Nutzung der Primärdokumente ermöglichen, erleichtern und bestimmen [RF 1997]. Metadaten machen aus Daten Informationen, weil ein Dokument durch Suchagenten unter Umständen erst aufgrund dieser Beschreibung gefunden und verwendet werden kann (Abbildung 3).

Metadaten können sowohl für elektronische Dokumente, als auch für nur physisch vorhandene Objekte verwendet werden. Ein solches Objekt ist beispielsweise ein geographischer Raum, der selbst nicht elektronisch gespeichert, aber elektronisch beschrieben werden kann. Metadaten sollen helfen, die Präzision von Recherchen zu

verbessern, indem sie Dokumente strukturieren, für die Indexierung durch Suchmaschinen festgelegte Datenfelder bereitstellen und damit die mangelnde Retrievalpräzision von Suchmaschinen auf dem Volltext eines Dokumentes ausgleichen. Geo-Metadaten sollen dieselbe Leistung für den Einsatz in Geoinformationssystemen erbringen. Erst Metadaten ermöglichen die Umsetzung und den Betrieb eines professionellen Geoinformationssystems, den allen geographischen Daten (Geo-Daten) ist gemeinsam, dass sie in roher Form praktisch unverständlich sind. Alle Metainformationen einer Landkarte (Maßstab, Legende, Aktualität etc) sind zum Beispiel zunächst nicht vorhanden und digitale Geo-Daten ohne diese Beschreibung sind unbrauchbar.

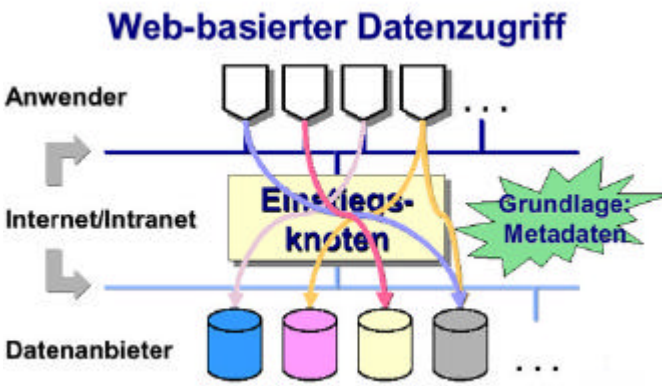

Abbildung 3: Web-basierter Zugriff auf Geodaten

Um dem Anspruch eines Geoinformationssystems gerecht zu werden müssen aber neben Geometrie-, Topologie-, Thematik-, Dynamik- und Sachdaten vor allem auch Metadaten über die Qualität der Geodaten zur Verfügung stehen.

Geographische und geo-räumliche Metadaten ermöglichen das Auffinden, die Verwendung und den Umgang mit digitalen geographischen Daten. Sie sind die formalen Beschreibungen der Eigenschaften von Geo-Objekten in Form von Ziffern und Zeichen zur computergerechten Verarbeitung [HU 2002].

Auf der Suche nach einem Standard für Geo-Metadaten sind vier Grundsätze zu beachten. Eine Beschreibungssprache für Objekte muss einfach sein, damit sie auch ein in der Beschreibung von Objekten ungeübter Autor einsetzen kann. (Simplicity of Creation and Maintenance). Es sollte eine domänen-unabhängige Beschreibungssprache verwendet werden (Commonly Understood Semantics) und damit die Daten international verwertet werden können, ist die Berücksichtigung verschiedener Sprachen unerlässlich (International Scope) . Außerdem muss die Sprache, die für

Metadatensätze verwendet wird, leicht erweiterbar sein (Extensibility) [HD 2002].
Diverse Normierungs- und Standardisierungseinrichtungen übernehmen derzeit die Aufgabe der Standardisierung von geo-räumlichen Metadaten. Einerseits muss festgelegt werden, welche Charakteristika von Geo-Daten überhaupt beschrieben werden, andererseits muss dann das konkrete Format für Speicherung und Transfer festgelegt werden. Die beiden derzeit wichtigsten Metadaten-Standardisierungs-Gremien sind die International Standards Organization (ISO) und das Federal Geographic Data Committee (FGDC). Diese beiden Organisationen werden mittelfristig gemeinsam eine international relevante Metadaten-Norm definieren.

1.3. Information Retrieval Systeme

„Die Disziplin ‚Information Retrieval' entstand aus der zunehmenden Notwendigkeit, aus immer größer werdenden Datenbeständen die jeweils für eine Handlung bzw. Situation relevanten Informationen herauszugreifen"[4].
Prinzipiell ist das Speichern und die Wiedergewinnung von Information einfach: Ein Benutzer sucht Information oder hat eine Frage bezüglich eines bestimmten Themengebietes und es steht ihm eine Anzahl Dokumente für seine Anfrage zur Verfügung. Diese Aufgabe kann erledigt werden, indem alle Dokumente gelesen und alle irrelevanten Dokumente weggelegt werden. Ein solcher Vorgang wäre ein perfektes Information Retrieval (IR), jedoch natürlich mit einem viel zu großen Zeitaufwand verbunden und daher nicht praktikabel. Auch moderne Hochleistungsrechner können zunächst dieser idealen Vorstellung des Retrievalprozesses nicht gerecht werden. Das computerisierte Lesen eines Dokumentes erfordert das syntaktische und semantische Verstehen des Textes. Das Problem ist somit nicht nur die automatische Extraktion der Information, sondern auch die vorherige Entscheidung, welche Textsegmente tatsächlich relevant sind. Ziel des gesamten Prozesses soll sein, dem Benutzer die relevantesten Dokumente anzubieten und möglichst alle irrelevanten Dokumente auszublenden. Die Metadaten eines Dokumentes helfen dem computerisierten Retrieval System zu entscheiden, welche Dokumente mit der Anfrage des Anwenders bearbeitet

[4] Bekavac, Dr. B – **„Skript zum Kurs Information Retrieval"**, Universität Konstanz, Informationswissenschaft, Wintersemester 2001 / 2002. Seite 2, Zeile 11ff.. Quelle: www.inf-wiss.uni-konstanz.de/CURR/winter0102/IR/ir_script_ws01.pdf, online am 10.08.2002.

werden sollen und welche schon zuvor als nicht relevant ausgeschlossen werden können.

Die Effizienz eines Retrievalsystems wird gemessen an Precision (Vollständigkeit) und Recall (Relevanz). Der Precision-Wert errechnet sich aus der Relation gefundener relevanter Dokumente zu der Gesamtzahl aller gefundenen (relevanten und nicht relevanten) Dokumente. Der Recall-Wert definiert sich aus der relativen Beziehung von allen gefundenen, relevanten Dokumenten und der Anzahl aller relevanten (sowohl gefundenen als auch nicht gefundenen) Dokumente. Die Effektivität beschreibt die Fähigkeit des Systems, dem Benutzer die benötigte Information bei möglichst geringen Kosten an Zeit und geringer mentaler Anstrengung anzubieten, sowie den Nutzen des Systems, welcher der Qualität des Suchergebnisses entspricht.

Nach Salton und McGill [SM 1978] ist der Gegenstand des Information Retrieval die Repräsentation, die Speicherung und Organisation von Informationen sowie der Zugriff auf diese. Ein IR-System ist somit eine Anwendung, die den Benutzer bei diesen Operationen unterstützt oder diese für ihn automatisiert.

Im folgenden wird kurz das EU Forschungsprojekt INSYDER, ein IR- und Business Intelligence System für die Suche im Internet, vorgestellt. Kern dieser Arbeit ist das INVISIP GIS-Projekt, insbesondere die Idee und Implementation eines neuartigen Konzeptes zur Visualisierungen von geo-räumlichen Metadaten im Zusammenspiel mit bewährten Methoden der Information Visualization. Ansatzpunkt beider Forschungsprojekte ist die Verbesserung der Effektivität von Information Retrieval Systemen. Dies wird sowohl durch eine hohe Ergonomie der Benutzerschnittstelle, als auch durch eine optimale Visualisierung der Metadaten eines Dokumentes erreicht.

2. Abgeschlossene und Laufende Projekte

2.1. INSYDER

Das Projekt mit den Namen INSYDER (Internet Système de Recherche) wurde von der Europäischen Kommission als ESPRIT Projekt mit der Nummer #29232 finanziert.
Mit INSYDER (Abbildung 4) sollen Unternehmen bei der Suche nach Daten und Informationen im World Wide Web unterstützt werden. Dabei soll INSYDER für den Anwender bei der Suche nach relevanten Dokumenten eine größere und mehr leistende Hilfestellung sein, als dies normale im Internet zur Verfügung stehende Suchmaschinen sind. Durch einen semantischen Agenten und einen Thesaurus soll der Inhalt von Dokumenten effizient auf Relevanz untersucht und Suchergebnisse gefiltert werden.

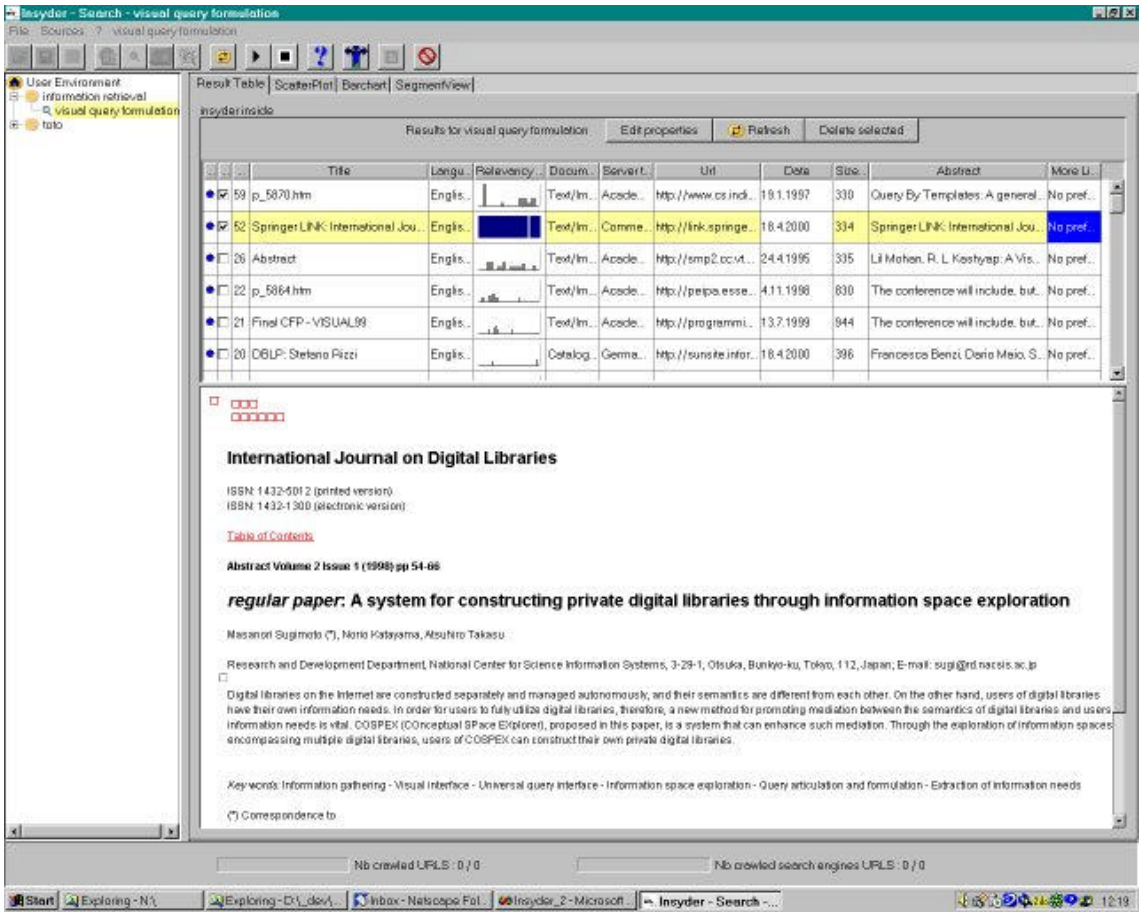

Abbildung 4: Insyder Anwendung

Nicht alle Informationen die im Internet vorhanden und verfügbar sind, sind auch in allen Suchmaschinen indexiert. Der Benutzer wird in der Regel nur eine sehr kleine Anzahl von Suchmaschinen verwenden, um im Internet zu recherchieren. Dadurch sind von Anfang an schon sehr viele Daten von der Ergebnismenge ausgeschlossen, da die verwendeten Suchmaschinen diese Information nicht im Index gespeichert haben. INSYDER soll diese Einschränkung der manuellen Suche aufheben und die Suchanfrage des Benutzers an eine Vielzahl von Suchmaschinen weiterleiten. Die Ergebnismenge wird vom System quantitativ und qualitativ optimiert.

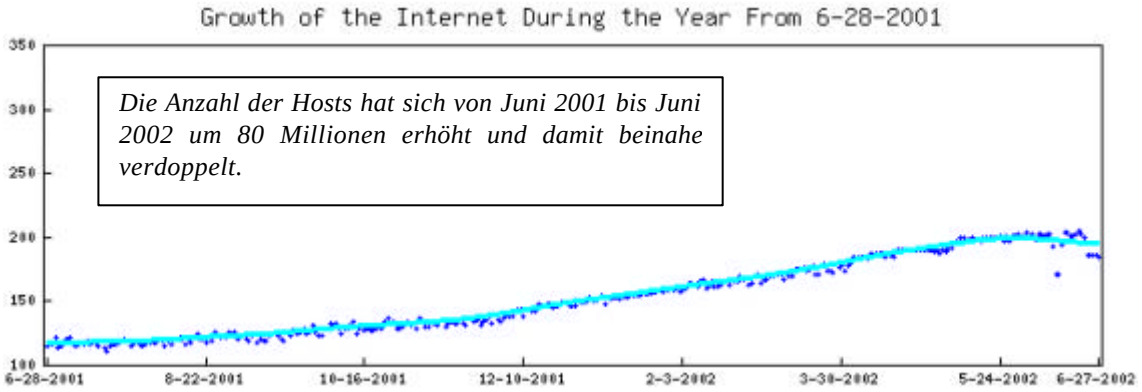

Abbildung 5: Growth Of The Internet 2001/2002

Durch das schnelle Wachstum des Internet (Abbildung 5), die zunehmende Anzahl von Internetseiten (Tabelle 2) und einem damit verbundenen potentiellen Anstieg der Ergebnismenge je Suchanfrage, wird der Benutzer mit sehr vielen Daten konfrontiert, aus denen er die für ihn relevanten Informationen extrahieren muss. In Verbindung mit einer großen Datenvielfalt wird es für den Benutzer immer schwieriger die gesuchte Information auch tatsächlich zu finden. Ist der Retrievalprozess mit einem hohen zeitlichen Aufwand verbunden, kann dies dazu führen, dass Benutzer sich schon mit weniger relevanten Ergebnissen zufrieden geben, da das Auffinden der relevantesten Informationen unter Umständen zu lange dauert. Dies kann auch damit zusammenhängen, dass das Rankingverfahren der Suchmaschine nicht verstanden wird. Suchmaschinen bieten nur sehr wenig Einblick in diese Verfahren. In den meisten Fällen werden zwar Titel, Überschriften und Inhalte berücksichtigt, jedoch nicht semantisch untersucht. Das Ranking wird oft davon beeinflusst, wie viele andere Webseiten auf die in der Ergebnismenge enthaltene Seite verweisen (Link Popularity), wie oft die Webseite in einer Ergebnismenge schon angeklickt worden ist (Direct Hit)

oder die Reihenfolge der Ergebnisse wird gegen Bezahlung zu Gunsten einer Webseite modifiziert. [ST 2002]

Monthly Average Hosts in Millions			
Month	**2000**	**2001**	**2002**
Jan	70.1222	101.649	151.896
Feb	72.8703	105.249	159.920
Mar	75.0431	108.562	168.489
Apr	77.1176	111.254	179.075
May	80.0517	113.538	189.749
Jun	82.6591	116.329	-
Jul	85.5456	118.865	-
Aug	88.2858	121.756	-
Sep	91.1874	124.842	-
Oct	93.8707	129.386	-
Nov	96.1370	137.082	-
Dec	98.7915	145.013	-

Tabelle 2: Zuwachs an Hosts im WWW Januar 2000 bis Mai 2002 in Millionen

INSYDER soll den Benutzer nicht mit zu vielen Ergebnissen überlasten. Suchanfragen sollen vom Benutzer leicht eingegeben und verwaltet werden können. Die Anzeige der Ergebnisse sowie deren Reihenfolge soll möglichst einfach und verständlich sein. Die Visualisierung der Suchergebnisse soll dem Benutzer bei der Extraktion der für ihn relevantesten Daten eine optimale Hilfe sein. Somit legt INSYDER sehr viel Wert auf die Unterstützung des Benutzers auf dem Weg von der Eingabe einer Suchanfrage bis hin zur Filterung und Sortierung der Ergebnismenge. Das System verwendet zur Visualisierung der Ergebnisse eine ResultTable[5], sowie zusätzlich einen Scatterplot, BarCharts, und SegmentViews. Die Benutzeroberfläche ist an die Bedürfnisse des Anwenders anpassbar, die Navigation im Programm einfach und durch Metaphern gestützt.

Bei der Entwicklung des Systems waren die sogenannten „**5 T**“ Kriterien die Anleitung zur Gestaltung der grafischen Benutzeroberfläche. Diese fünf „T“ stehen für eine Konzentration auf die Zielgruppe der Anwendung (**T**arget Group), die typischen

[5] Begriff „ResultTable“ aus [MT 2001]

Aufgaben der Benutzer (**T**ypical Tasks), die technische Umgebung (**T**ypical Environment), die Beschaffenheit und Art der Daten, die visualisiert werden sollen (**T**ype Of Data) und die Benutzbarkeit der Anwendung mit minimalem Trainingsaufwand (**T**raining) [MKRE 2002]/[MR 1999].

Die umfangreiche Funktionalität und die Mehrwerte gegenüber herkömmlichen Suchmaschinen im Internet zeichnen INSYDER nicht nur als Suchagenten aus, sondern im Nutzungskontext der Anwendung kann von einem Business Intelligence System gesprochen werden. INSYDER bietet adaptive Schnittstellen und kann in andere Anwendungsdomänen übertragen werden. So ist beispielsweise der Thesaurus leicht auswechselbar, wodurch der Evaluationsagent bei der Analyse von Dokumenten immer entsprechend des Kontext deren Relevanz bewerten kann.

2.2. INSYDER Redesign und INVISIP

2.2.1. Usability Probleme

Eine Evaluation mit 40 Benutzern des INSYDER Prototypen hinsichtlich der „5 T“ Faktoren und der verwendeten Visualisierungen stellte heraus, dass 50% der Benutzer den ResultTable als visuelle Unterstützung bei der Verwaltung der Suchergebnisse favorisierten und die tabellenbasierte Darstellung demnach auch am meisten und am längsten benutzt wurde. Gleichzeitig zeigte sich, dass die Mehrzahl der Nutzer allgemeine Schwierigkeiten bei der Verwendung der Visualisierungen hatte. Die anderen Visualisierungen, wie Scatterplot oder BarCharts, wurden hauptsächlich nur im Zusammenhang mit der ResultTable verwendet und kaum als alleinstehende Visualisierungen erkannt. Das Umschalten zwischen verschiedenen Visualisierungsformen verwirrte den Benutzer und war somit weiterer Faktor aufgrund dessen die ResultTable am meisten Zuspruch fand [MKRE 2002]/[MT 2001].

2.2.2. SuperTable Konzept

Die Ergebnisse der Evaluation des INSYDER Systems führten zu der Überlegung, eine tabellenbasierte Visualisierung als Basis des Business Intelligence Systems zu verwenden und die ResultTable dann mit den anderen Visualisierungen zu einer SuperTable[6] zu kombinieren, in welche die anderen Visualisierungen zum Teil integriert sind (Abbildung 6).

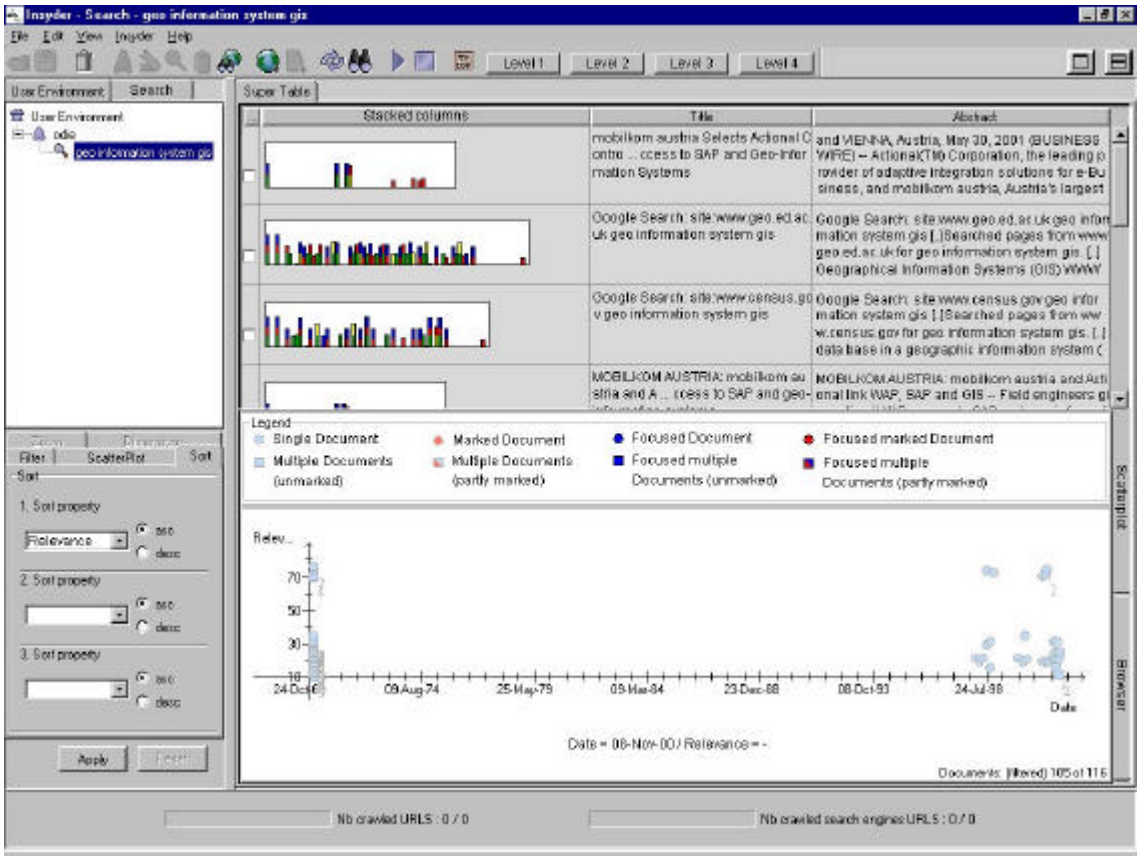

Abbildung 6: Insyder Redesign HTML Prototyp

Die Verbesserungen und neuen Komponenten des INSYDER Redesign sind gleichzeitig Bestandteil des neu aufgesetzten INVISIP Projektes[7]. INVISIP - **In**formation **Vis**ualisation For **S**ite **P**lanning – ist wie INSYDER ein System, welches die Suche nach relevanter Information, sowie deren Auswertung und Verwaltung unterstützen soll. Jedoch ist die Anwendungsdomäne von INVISIP die geo-räumliche Ortsplanung, also unter anderem die Suche nach Bauplätzen oder Erschließungsgebieten, welche die

[6] Begriff „SuperTable“ aus [MKRE 2002]
[7] Finanziert von der Europäischen Kommission, IST Programm, Projekt Nummer IST-2000- 29640.

Applikation als Geoinformationssystem charakterisiert. Das System soll eine Hilfestellung aller betroffenen Benutzergruppen, von der öffentlichen Einrichtung bis hin zum privaten Nutzer, sein. Dabei sollen Funktionen sowohl für den Privatbenutzer, als auch für Experten, wie Techniker oder Planungsbüros, zugeschnitten sein (Abbildung 7). Suchanfragen an das System greifen auf eine Wissensdatenbank zu, die mit Metadaten gefüllt ist.

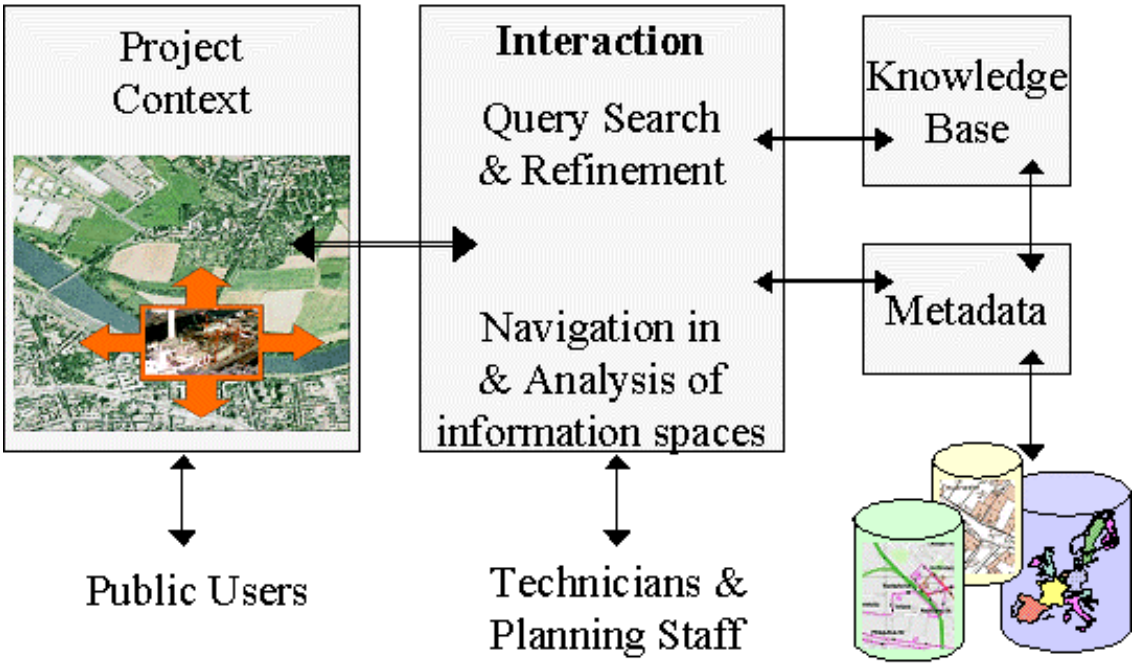

Abbildung 7: INVISP Use Scenario

Bei der Verbesserung der grafischen Oberfläche von INSYDER und der Neugestaltung des User Interface von INVISIP geht das Prototyping in zwei Richtungen. Zum einen wird aus der ResultTable die SuperTable mit Levelkonzept entwickelt, zum anderen eine SuperTable nach einem Granularitätskonzept, **im folgenden als GranularityTable bezeichnet**. Beide tabellenbasierten Visualisierungskonzepte integrieren weitere, schon bekannte und zuvor im INSYDER verwendete Visualisierungsformen. Sie sollen durch einen im Split-Screen eingeblendeten Scatterplot [GF 2002] unterstützt werden.

2.2.3. SuperTable mit Level- und Granularitätskonzept

Die Testpersonen, die an der Evaluation des Insyder Prototypen teilnahmen, bemängelten hauptsächlich die Eigenschaft des Programms, welche ursprünglich dessen Mehrwert sein sollte, nämlich die Vielzahl und Koppelung der verschiedenen Visualisierungen. Eine Lösung dieses Konfliktes kann nun darin bestehen, die Anzahl der Visualisierungen zu reduzieren und die Verbindungen zwischen graphischer Darstellung eines Datensatzes und dessen Detailgrades bezüglich Text und Größe klarer zu gestalten.

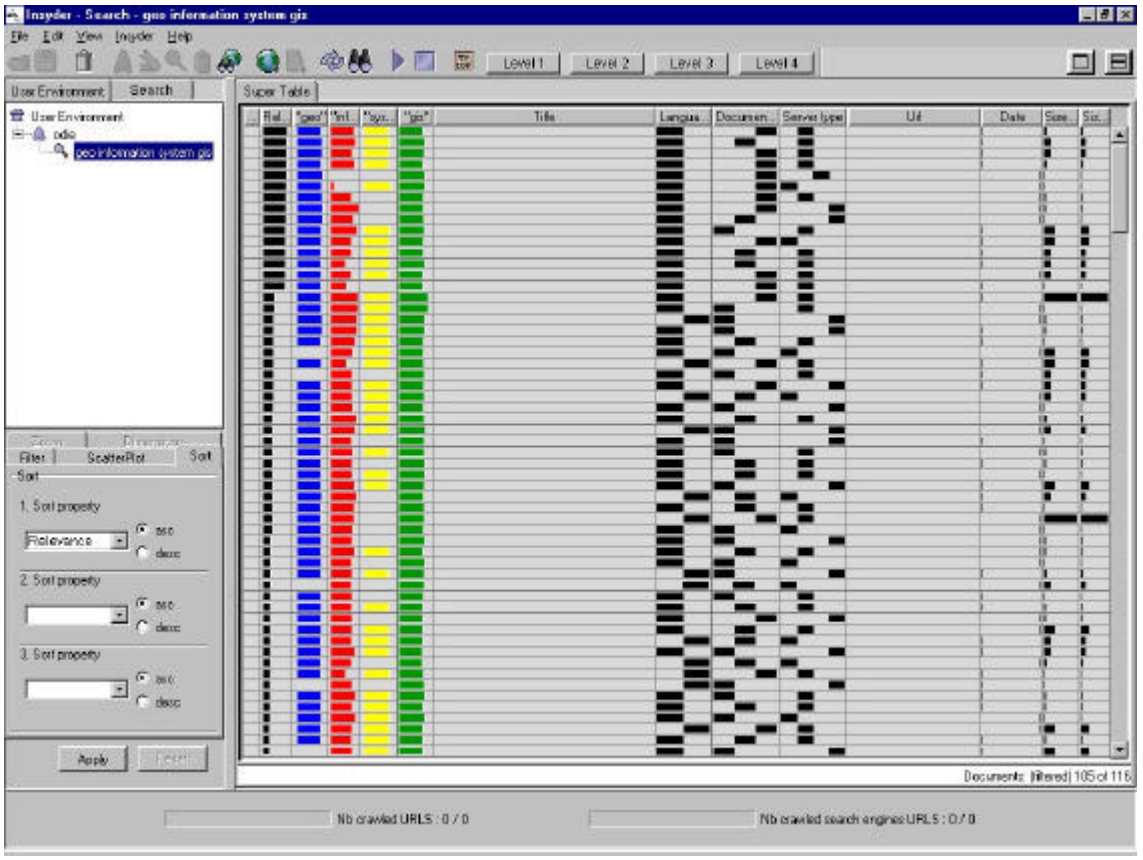

Abbildung 8: Insyder Redesign HTML Prototyp, Level 1

Der INSYDER HTML Redesign Vorschlag beinhaltet ein neues Levelkonzept, mit dem der Benutzer zwischen verschiedenen Stufen von Visualisierung und Detail umschalten kann. Die erste Detailstufe zeigt dabei alle Dokumente in einer großen tabellarischen Übersicht. Es ist kein Text sichtbar, lediglich BarCharts sind je Tabellenspalte zu sehen, die die Relevanz des Dokumentes bezüglich bestimmter Filterkriterien anzeigen (Abbildung 8).

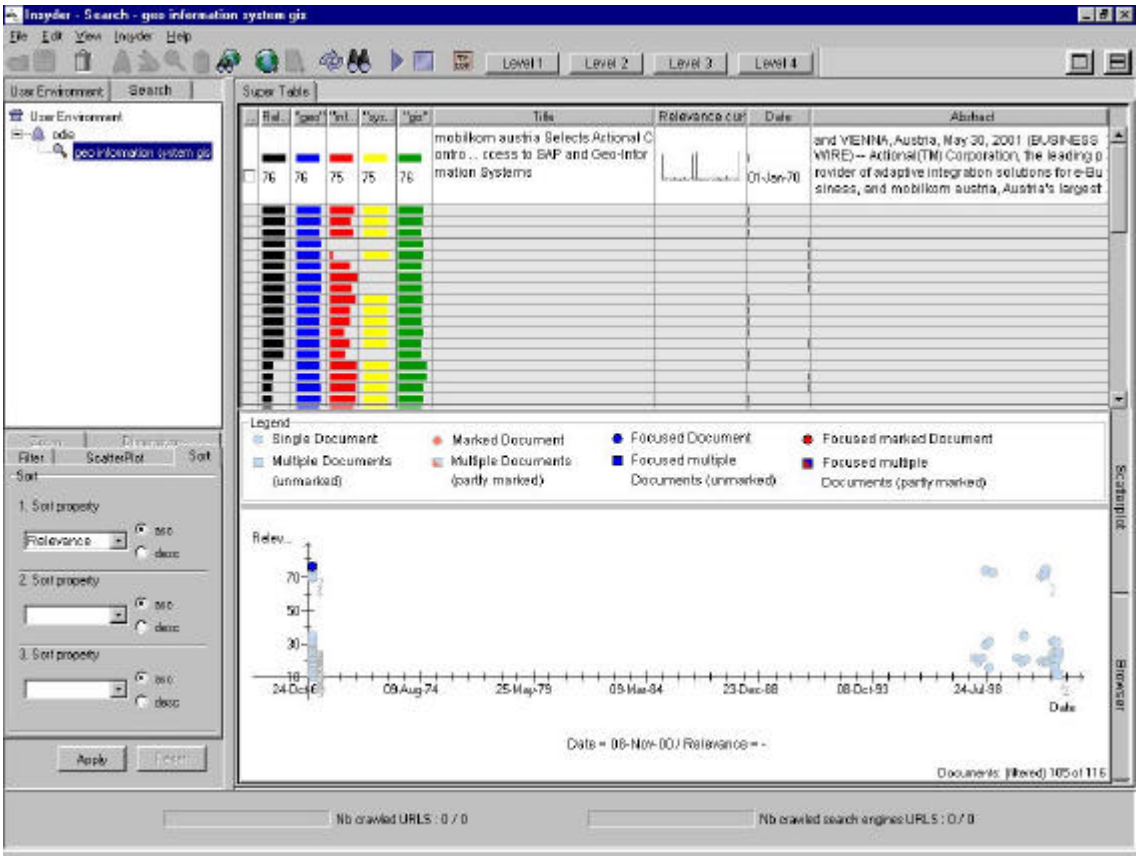

Abbildung 9: Insyder Redesign HTML Prototyp, geteilte Ansicht mit Scatterplot, einzelnes Dokument in Level 3.

Der Benutzer kann per Mausklick auf eine Datensatz- bzw. Tabellenzeile den „Focus Of Interest“ eines Dokumentes verändern, d.h., das einzelne Dokument wird in einen anderen Level umgeschaltet (Abbildung 9) und die angezeigte Information wird dadurch in anderer visueller Struktur dargestellt. Zusätzlich können alle Datensätze über die globalen Level-Buttons simultan im Detailgrad erhöht oder erniedrigt werden (Abbildung 10) [MKRE 2002].

Während der Benutzer Document Retrieval betreibt und die Ergebnismenge seiner Suchanfrage untersucht, schaltet er zwischen textueller und graphischer Informationsverarbeitung um. Bei den meisten Systemen und so auch bei INSYDER werden Suchanfragen textuell, beispielsweise als vereinfachte SQL Query, eingegeben. Ein Informationssystem wie INSYDER oder auch INVISIP bereitet die Anfrageergebnisse graphisch auf, um den Selektionsprozess auf diese Weise für den Benutzer zu beschleunigen und zu vereinfachen. Betrachtet der Benutzer ein einzelnes oder mehrere Dokumente genauer, erhöht sich automatisch der Grad an textueller Information, wie zum Beispiel im Level 4 der INSYDER Applikation (Abbildung 11).

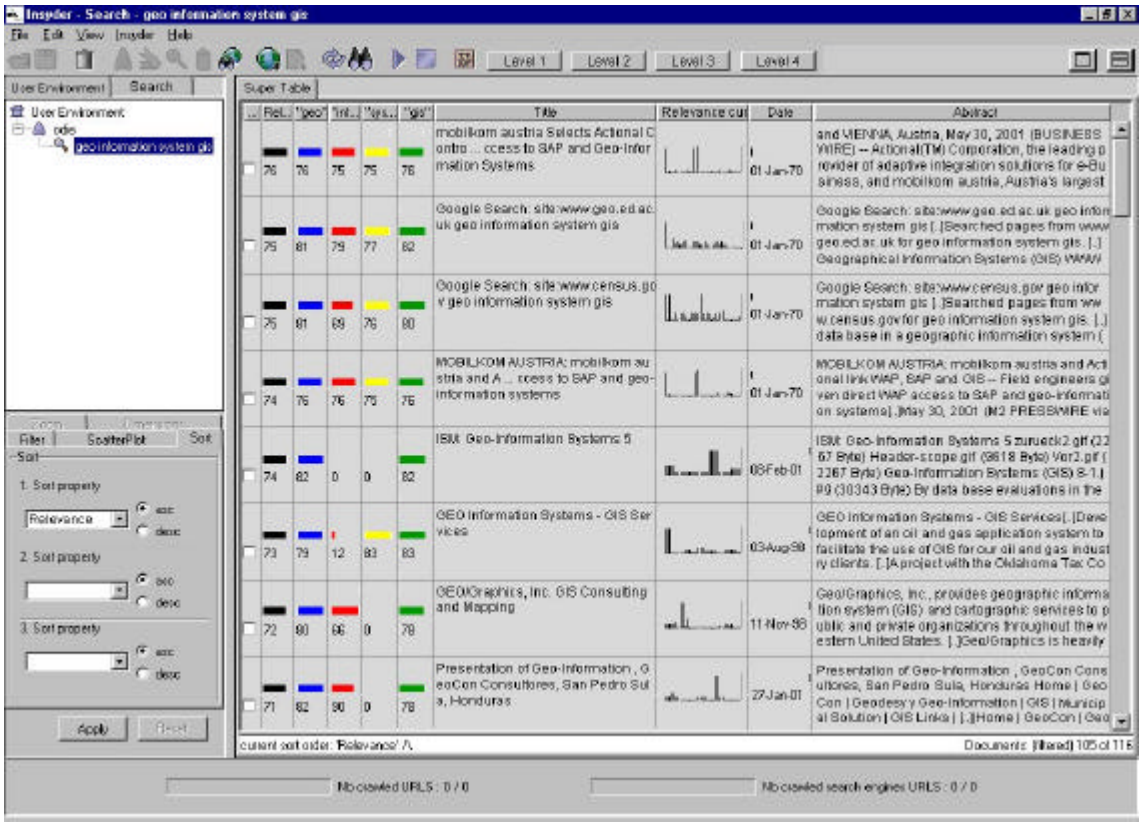

Abbildung 10: Insyder Redesign HTML Prototyp, Level 3

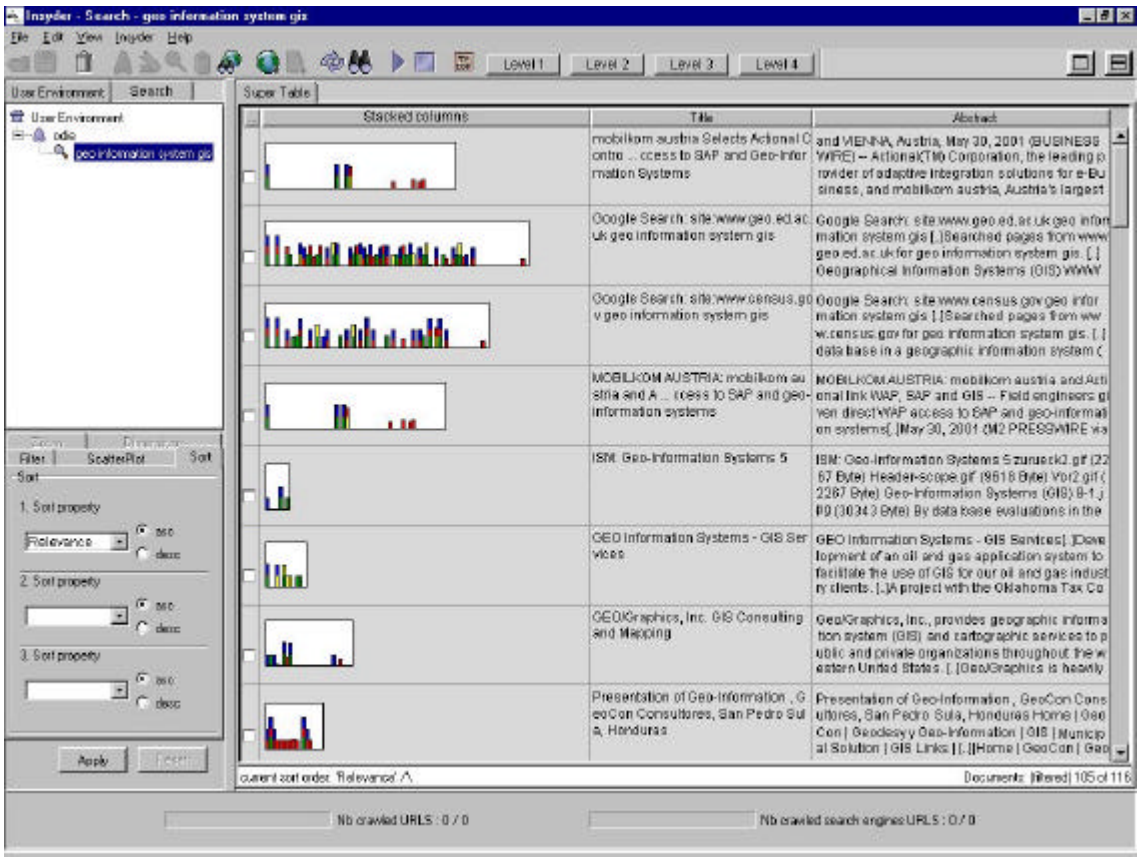

Abbildung 11: Insyder Redesign HTML Prototyp, Level 4

Dem Prozess der Veränderung der Modalität (Abbildung 12), dem Umschalten zwischen graphischer und textueller Informationsverarbeitung, ist der Benutzer bei INSYDER demnach mehrmals ausgesetzt.

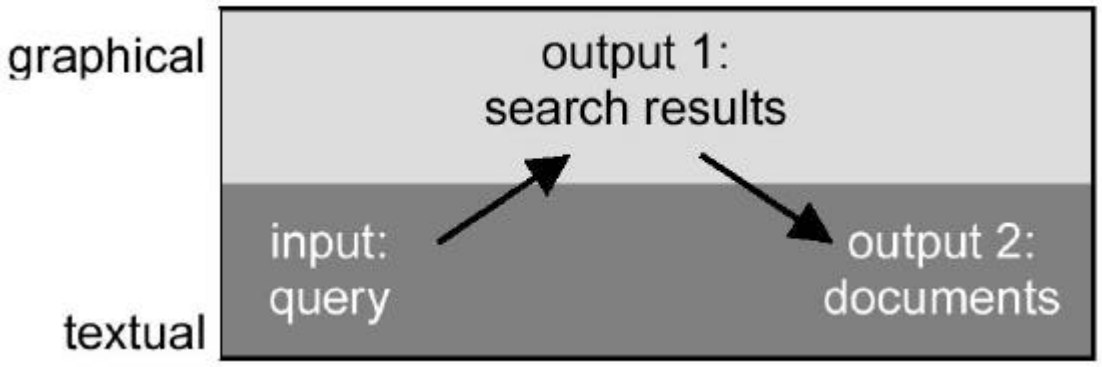

Abbildung 12: Veränderungen der Modalität

Während die Visualisierungen und graphischen Komponenten des Systems das Document Retrieval vereinfachen, stellen textuelle Eingaben und die Analyse von Text eine kognitive Belastung dar. Durch das Granularitätskonzept kann verhindert werden, dass die Vorteile graphischer Darstellung wieder von textueller Dokumentenanalyse verringert oder gar absorbiert werden [MKRE 2002].
Das Konzept der Granularität sieht vor, die Buttons, die im INSYDER HTML Prototypen zum Umschalten zwischen den verschiedenen Detailstufen verwendet werden, durch Granularitätsschieber zu ersetzen. Dabei soll es einen globalen Regler, sowie für jedes Dokument einen einzelnen Regler geben. Der Granularitätsregler wird genauso wie die verschiedenen Visualisierungen eingebettet in eine tabellarische Anordnung. Die entstehende GranularityTable[8] umfasst konzeptuell somit drei Spalten, je eine für Visualisierung, Text und den Regler (Abbildung 13).

Visualisierung	Text	Granularität
Thumbnail mit markierten Suchbegriffen	möglichst Volltext mit markierten Suchbegriffen	Granularitäts-schieber rechts

Abbildung 13: Aufbau der GranularityTable, höchste Detailstufe

[8] GranularityTable ist die Entwicklungsbezeichnung für eine SuperTable mit Granularitätskonzept nach [MKRE 2002]

Der „harte Bruch zwischen Visualisierung und Text“[9] wird durch die Einführung der Granularitätsstufen aufgehoben. Text und Visualisierung verändern sich proportional hinsichtlich des Detailgrades, wodurch der bei INSYDER von vielen Benutzern nicht verstandene Zusammenhang zwischen textueller und graphischer Detailstufe besser hergestellt werden kann und der Übergang von voller graphischer Darstellung zu Volltext fließend ist.

In der ersten und niedrigsten Granularitätsstufe enthält die Tabelle in der Visualisierungsspalte nur schmale BarCharts, wie sie auch im Level 1 des INSYDER Redesign verwendet wurden (Abbildung 14). Die Textspalte kann aufgrund der niedrigen Zellenhöhe noch keinen lesbaren Text anzeigen und bleibt daher leer bzw. zeigt nur Teile von Buchstaben an. Der Granularitätsschieber bleibt in dieser Detailstufe zunächst verborgen und soll erst bei Überfahren mit der Maus sichtbar werden.

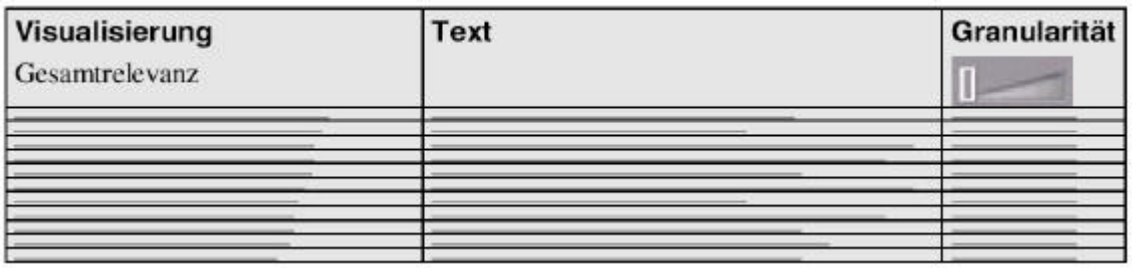

Abbildung 14: GranularityTable, erste Detailstufe

In der zweiten Detailstufe werden anstelle der Gesamtrelevanz diejenigen Werte angezeigt, aus denen sich die Gesamtrelevanz zusammensetzt. Gleichzeitig wird die Tabellenzeile höher, so dass mit der Anzeige mehrerer BarCharts auch eine erste Anzeige von Text einhergeht (Abbildung 15). Dabei soll die Tabellenzelle genau so hoch sein, dass der Titel des Dokumentes sichtbar ist. In der dritten Granularitätsstufe werden neben den BarCharts der Einzelrelevanzen auch deren Werte auch noch numerisch angezeigt. Außerdem erhöht sich wiederum die Höhe der Tabellenzeile, so dass zusätzlich zum Titel des Dokumentes auch der Autor angezeigt werden kann (Abbildung 16). Der Granularitätsschieber ist in Stufe Drei nun vollkommen sichtbar. Schiebt der Anwender den Regler an die vierte Position, so gelangt er in eine Anzeige mit TileBars. Die Höhe des Textbereiches erhöht sich bei dieser Umschaltung nicht oder nur in geringem Maße (Abbildung 17).

[9] Eibl, M. - „**Vorschlag für INSYDER Redesign**“ - IZ Social Science Information Centre, Dezember 2001. Seite 2, Zeile 12ff.

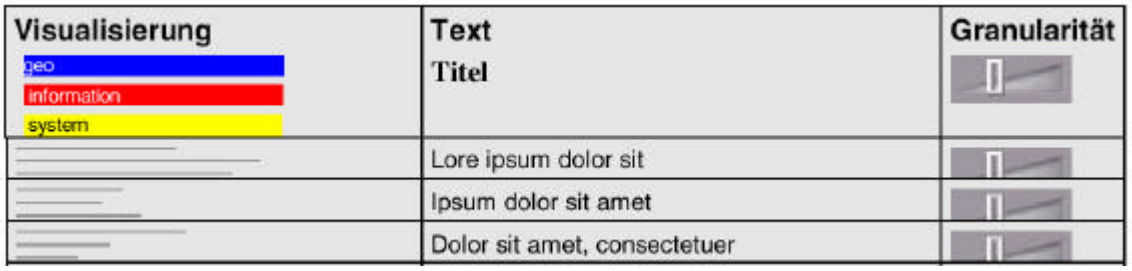

Abbildung 15: GranularityTable, zweite Detailstufe

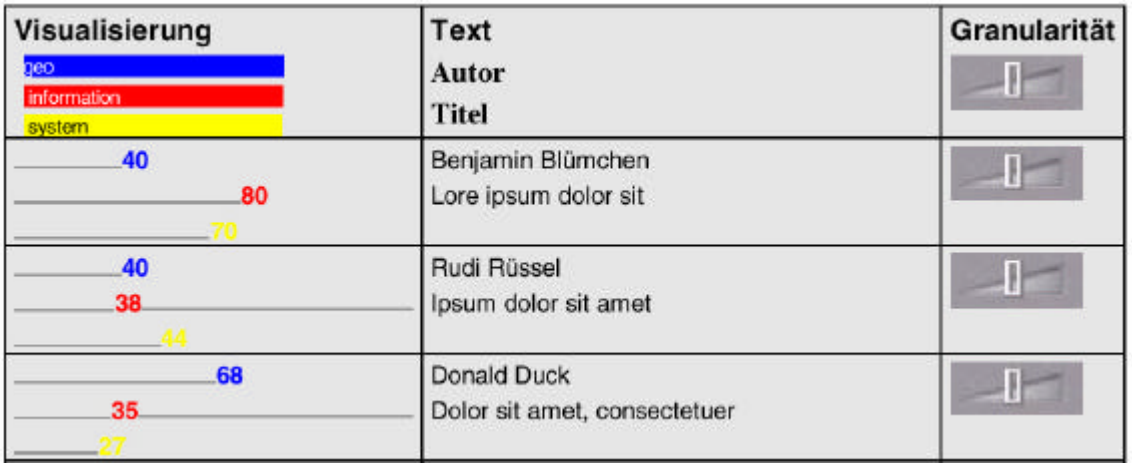

Abbildung 16: GranularityTable, dritte Detailstufe

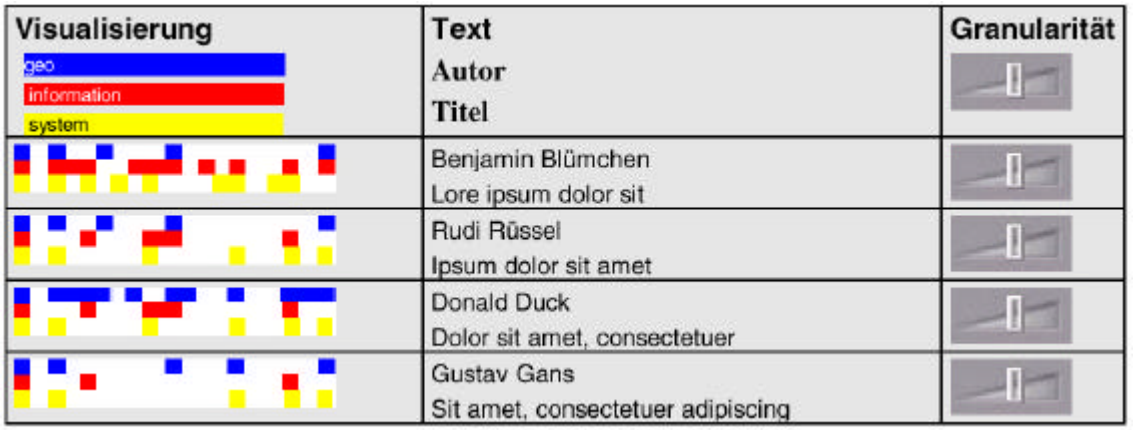

Abbildung 17: GranularityTable, vierte Detailstufe

Im Schritt von Granularitätsstufe Vier zu Fünf vergrößert sich die Zellenhöhe der Tabellenzeile dagegen um so mehr. Die Textspalte hat eine um eine Stufe größere

Granularität und zeigt nun zusätzlich den Abstract des Dokumentes an. Der größere sichtbare Textbereich gibt den Platz für eine um 90° gedrehte TileBar frei (Abbildung 18).

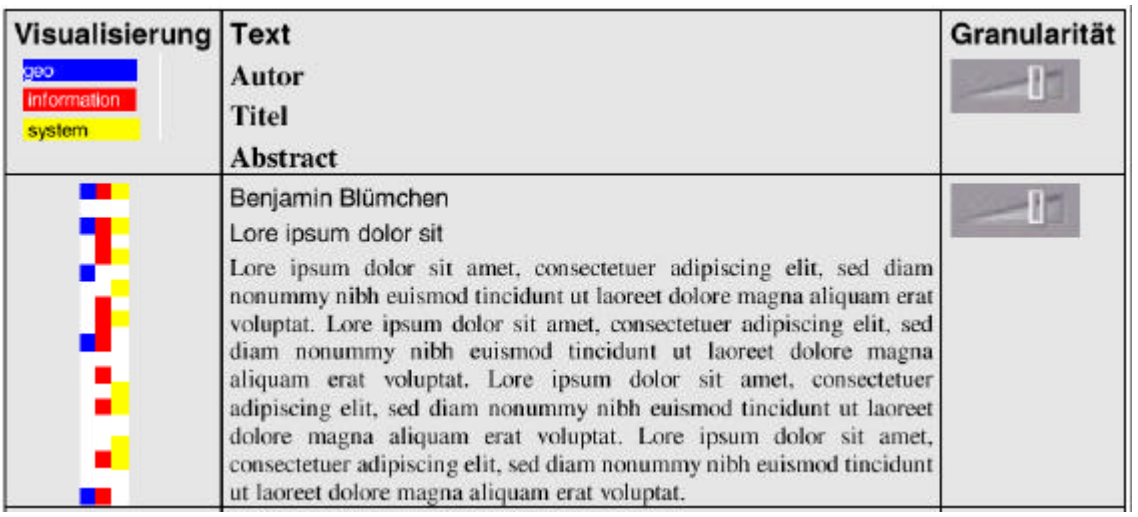

Abbildung 18: GranularityTable, fünfte Detailstufe

Die sechste Detailstufe soll nach der Idee von M. Eibl [EM 2001]/[MKRE 2002] eine Thumbnail des Abstracts oder des Volltextes anzeigen.

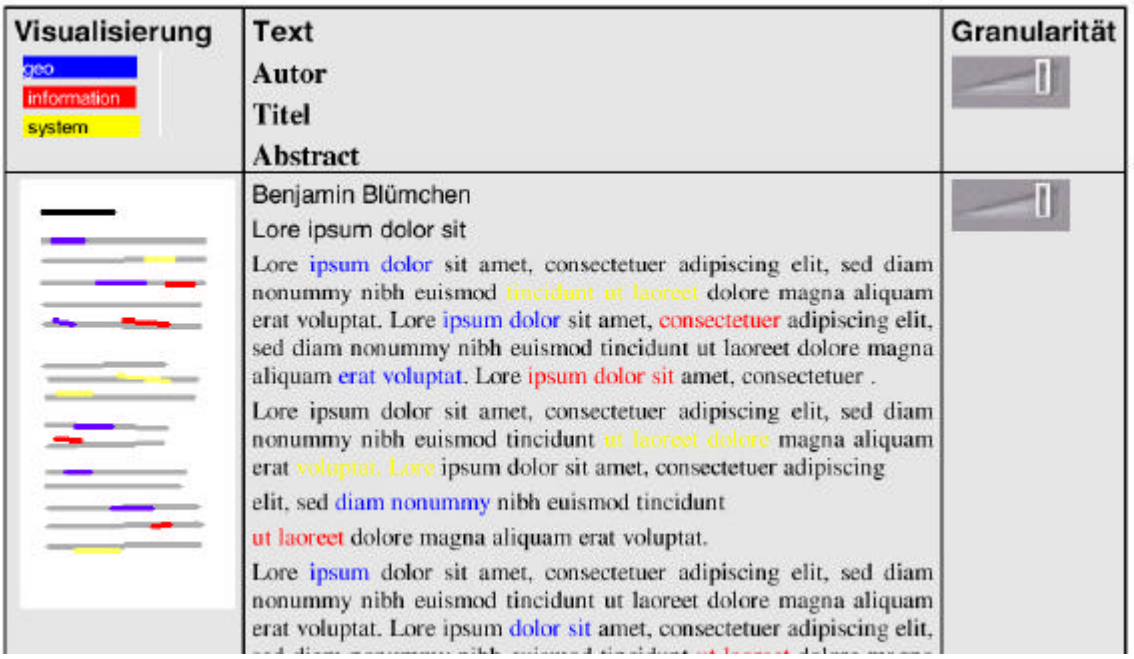

Abbildung 19: GranularityTable, sechste Detailstufe

Im Abstract des Textes sind die relevanten Suchbegriffe ebenso wie im Thumbnail mit den entsprechenden Farben hervorgehoben (Abbildung 19).

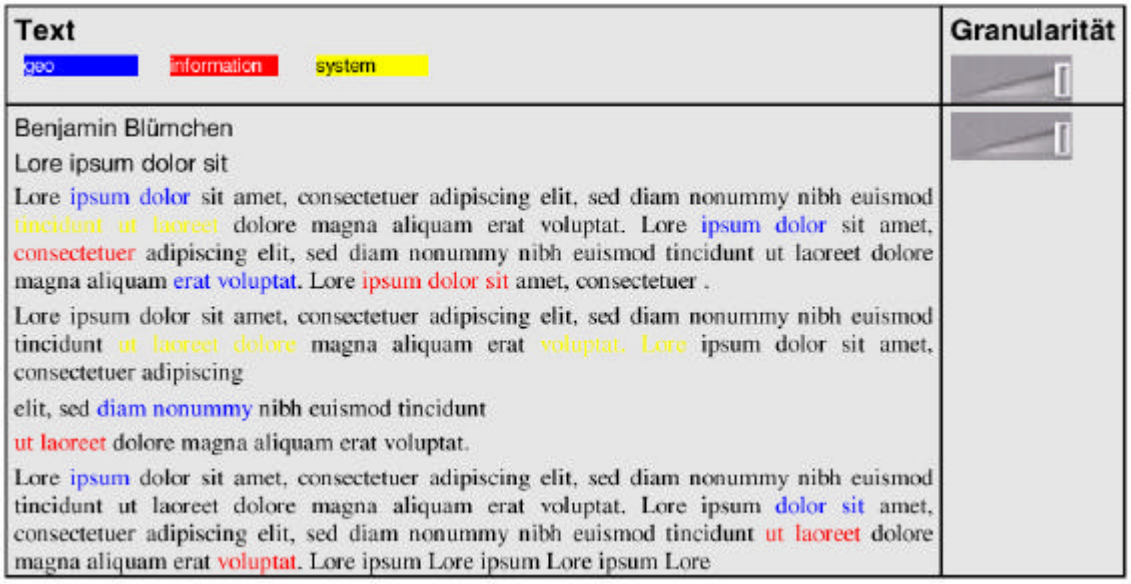

Abbildung 20: GranularityTable, siebte Detailstufe

Die Siebte ist die höchste und letzte Granularitätsstufe. Hier werden die beiden Tabellenspalten „Visualisierung“ und „Text“ verbunden. Es wird nur noch Text angezeigt (Abbildung 20), da „die höchste Granularität eines Textes ... der Text (selbst)“[10] ist [EM 2001].

[10] Eibl, M. - „**Vorschlag für INSYDER Redesign**“ - IZ Social Science Information Centre, Dezember 2001. Seite 10, Zeile 1.

3. Aufgabenstellung und Pflichtenheft

3.1. Aufgabenzielbestimmung

Die Aufgabenstellung des Softwareprojektes „Implementation einer tabellenbasierten Visualisierung für geo-räumliche Metadaten“ ist die Umsetzung des Granularitätskonzeptes in der Programmiersprache JAVA. Die neue GranularityTable[11] soll auf einem bereits bestehenden Datenmodell aufbauen und eng mit einer gleichzeitig entstehenden Scatterplot Visualisierung [GF 2002] verbunden sein.

3.1.1. Musskriterien

Das Redesign der Benutzeroberfläche des bestehenden INSYDER Forschungsprototypen im Rahmen des INVISIP Projektes ist Kern der Aufgabenstellung. Die Implementation neuer, Übernahme und Verbesserung alter Visualisierungen für Metadaten, sowie die Umsetzung des Granularitätskonzeptes und damit des Redesign Vorschlages von M. Eibl [EM 2001]/[MKRE 2002] steht damit im Zusammenhang. Ferner ist also die Darstellung aller Granularitätsstufen direkt in einer neuen Form des SuperTable mit der Entwicklungsbezeichnung „GranularityTable“, die Verknüpfung dieses Visualisierungskonzeptes mit einem neuen Scatterplot und die Kommunikation der Komponenten zu realisieren.

3.1.2. Wunschkriterien

Wünschenswert ist eine Verbesserung der Performanz und der Zuverlässigkeit des INVISIP Prototypen im Vergleich zur INSYDER Anwendung, sowie eine höhere Nutzerzufriedenheit im Umgang mit den Visualisierungen.

3.1.3. Abgrenzungskriterien

Schwerpunkt der Implementation sind Information Visualization und Usability. Die Gesamtperformanz der Applikation hängt von allen Komponenten des Prototypen ab.

[11] GranularityTable ist die Entwicklungsbezeichnung für eine SuperTable mit Granularitätskonzept nach [MKRE 2002]

3.1.4. Beschäftigungsdaten

Projektbeginn 1. November 2001, Beginn der Implementationsarbeit Januar 2002, Abschluss des Teilprojektes GranularityTable bis Juni 2002.

3.2. Produkteinsatz

Die Applikation dient zur Visualisierung geo-räumlicher Metadaten.

3.2.1. Anwendungsbereiche

Das Granularitätskonzept und die damit verbundenen Visualisierungen werden im Geoinformationssystem INVSISP verwendet. Anwendungsbereich des Produktes sind Testszenarien mit exemplarischen Metadaten zur Evaluation der Visualisierungen.

3.2.2. Zielgruppen

Zielgruppe ist somit der Lehrstuhl von Prof. Dr. Harald Reiterer[12] und alle Mitarbeiter, sowie andere beteiligte Forschungsgruppen und INVSISP Projektpartner.

3.3. Produktübersicht

Eine Übersicht über das Produkt liefert Abbildung 7.

3.4. Produktfunktionen

Es handelt sich um ein Information bzw. Document Retrieval System. Die Oberfläche des gesamten Prototypen soll aus einem Application Window, Panels, Menu Bar, Tool Bar, TabbedPanes, Visualisierungen und Interaktionsschaltflächen (Buttons, CheckBoxes, ToggleButtons, etc.) bestehen. Aus Metadaten werden verschiedene in die GranularityTable integrierte Visualisierungen erzeugt. Der Granularitätsslider wird verwendet, um die Darstellung der Datensätze zu verändern.

[12] Arbeitsgruppe „Informationssysteme“, Informatik & Informationswissenschaft, Universität Konstanz.

3.5. Produktdaten

Das Programm basiert auf XML Datensätzen, die aus der INSYDER Datenbank gewonnen worden sind. Die XML Datensätze beinhalten Metadaten über Dokumente aus dem World Wide Web und dienen zur Demonstration der Funktion von Visualisierungen, die später im Speziellen geo-räumliche Metadaten graphisch aufbereiten sollen.

3.6. Benutzungsschnittstelle

Für den gesamten Prototypen ist eine WIMP[13] Benutzeroberfläche vorgesehen. Es werden insbesondere Menüsteuerungen, Fenster, Dialogfenster und Frames implementiert. Die Bedienung soll durch Maus und Tastatur erfolgen.

3.7. Technische Produktumgebung

3.7.1. Software

Als Client-Betriebssystem können alle Betriebssysteme verwendet werden, auf denen eine JAVA VM[14] lauffähig ist.
Die Software ist in der plattformunabhängigen Programmiersprache JAVA entwickelt. Zum Systemstart ist eine JAVA VM und ein JAVA JDK[15] in der Version mit der Kennung 1.3.1_03 notwendig. Optimal ist eine Installation der *Borland JBuilder* Programmierumgebung ab Version 4.

3.7.2. Hardware

Client-seitig wird Personal Computer mit Intel Pentium III oder AMD Athlon Prozessor mit mindestens 128 MB RAM und 16 MB Grafikkarten benötigt.

[13] Windows, Icons, Menus, Pointers als Basiselemente der Interaktion
[14] Enlg. ‚Java Virtual Machine“. Laufzeitumgebung.
[15] Enlg. „ Java Developer Kit“. Java Bibliotheken und Tools.

3.8. Entwicklungsumgebung

Der Prototyp wird auf einem PC mit Windows 2000 Betriebsystem entwickelt. Programmierumgebung ist der *Borland JBuilder* in den Versionen 5 und 6. Bei der Entwicklung des gesamten Prototypen wird ein CVS[16] System verwendet. Ansonsten bestehen keine Abweichungen von der Produktumgebung.

3.9. Testszenarien

Als Testpersonen für eine Evaluation[17] des Prototypen fungieren Experten des Lehrstuhls von Prof. Dr. Harald Reiterer[18].

[16] Engl. „Concurrent Versions System“. System zur Versionenkontrolle bei großen Projekten.
[17] Die Evaluation wurde durchgeführt von [JC 2002].
[18] Arbeitsgruppe „Informationssysteme“, Informatik & Informationswissenschaft, Universität Konstanz.

4. Konzepte zur Umsetzung der Redesign Idee von M. Eibl

4.1. GranularityTable Prototyping

Der erste Prototyp soll die Möglichkeiten und Probleme bei der Gestaltung einer Tabelle beleuchten, die nicht nur Standardelemente wie Text, sondern weitere und verschiedenartige Subkomponenten enthalten soll. Der Prototyp besteht aus einem Frame, in welches zu großen Teilen Abbildungen von Komponenten des INSYDER Programms als Images eingefügt worden sind, um schnell eine bereits bekannte Programmoberfläche darstellen zu können. Hauptbestandteil der Implementationsarbeit ist der exemplarische Entwurf einer Tabelle (JTable). Im ersten Prototypen besteht die Tabelle aus 5 Spalten, wobei je eine Spalte für Visualisierung und Granularitätsschieber, und die übrigen Spalten zur exemplarischen Anzeige von Information vorhanden sind. Spalte eins enthält die Visualisierungen, welche zu Testzwecken zu diesem Zeitpunkt noch Screenshots der INSYDER Grafiken sind.
Der Detailregler ist in JAVA als JSlider umgesetzt. Verschiebungen des Sliders werden über einen Eventlistener registriert und die Reaktion des Programms ist der Austausch der Visualisierungsabbildung analog zur Erhöhung bzw. Verringerung der Granularität, sowie eine Veränderung der Tabellenzelle. Zusätzlich zu den zeilenweisen Schiebereglern ist analog zur Idee von [EM 2001] /[MKRE 2002] ein globaler Regler vorhanden, mit dem alle Tabellenzeilen gleichzeitig manipuliert werden können (Abbildung 21). Der Prototyp enthält durch Split-Screen (JTabbedPane) bereits eine Schablone, in die später der Scatterplot [GF 2002] eingefügt wird (Abbildung 22).
Bei der Implementation des Prototypen ist es erforderlich, eigene Bestandteile der Tabelle in JAVA zu programmieren. Grundsätzlich enthält eine JTable mit zugehörigen TableUI (Table User Interface) und TableModel nur Darstellungsmöglichkeiten für Text. Um andere Elemente in einer JTable anzuzeigen werden eigene TableCellRenderer und TableCellEditoren benötigt. Der TableCellRenderer ist verantwortlich für das Zeichnen einer Tabellenzelle, der Editor ermöglicht die Anwahl und Veränderung einer Komponente, die in eine Tabellenzelle eingefügt ist. Quellcode 1 zeigt zwei wichtige Methoden, die bei der Implementation eigener Renderer und Editoren für Tabellenzellen notwendig sind. Die Paintmethoden der JTable fragen stets an diesen Methoden an, sobald die jeweilige Zelle gezeichnet werden soll.

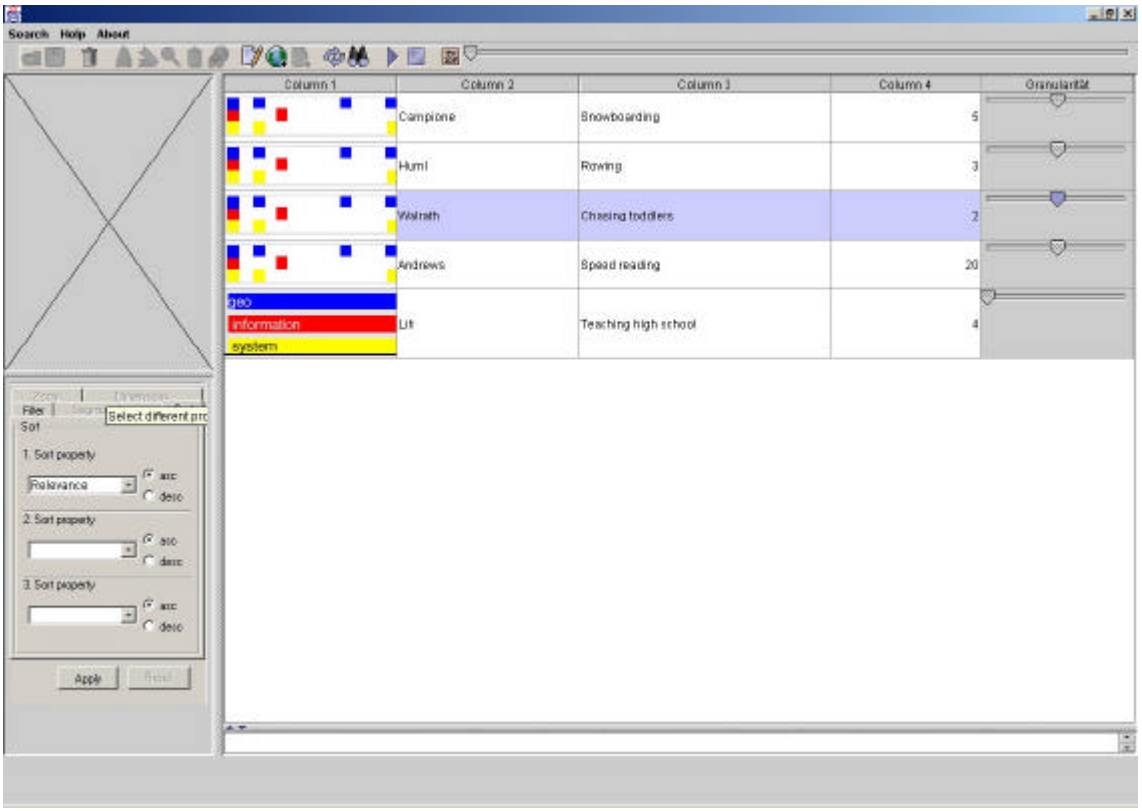

Abbildung 21: GranularityTable, erster Prototyp in JAVA

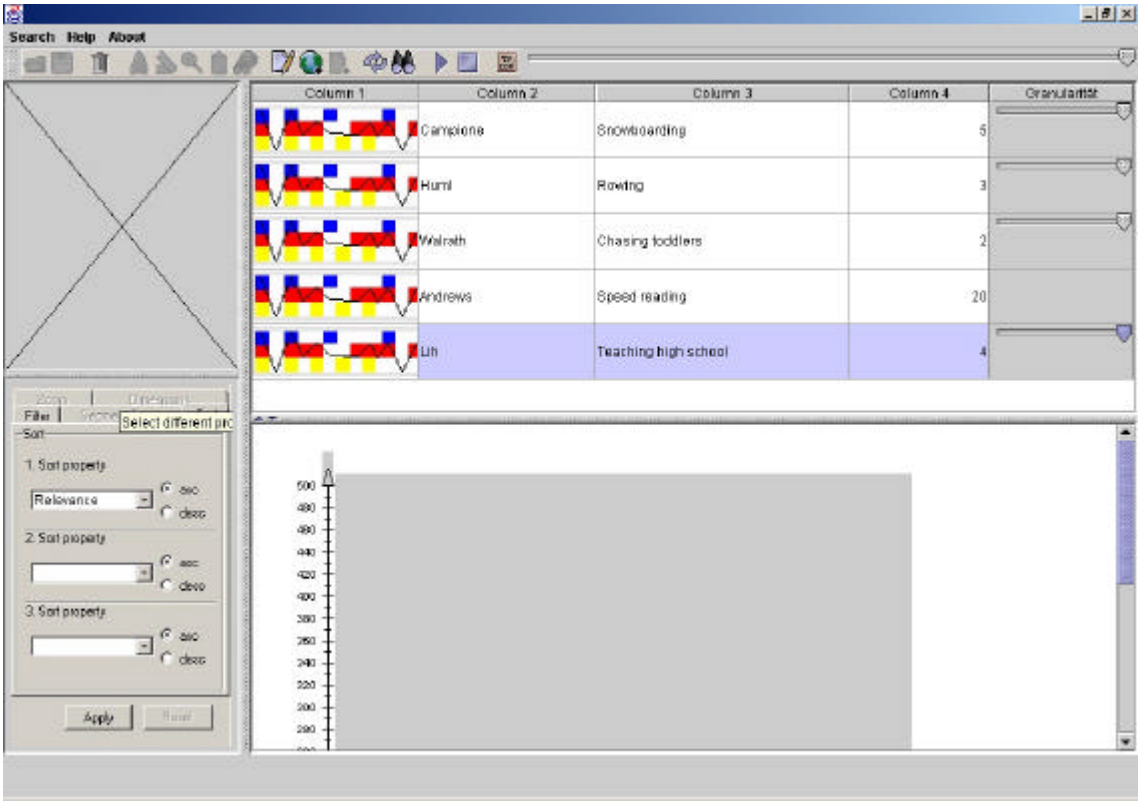

Abbildung 22: GranularityTable mit Scatterplotschablone im Splitscreen, erster Prototyp in JAVA

```
…
public Component getTableCellRendererComponent(JTable table, Object obj, boolean bool1, boolean
bool2, int row, int col) [Codeposition 1.1.]
{
   return (Component)sliderVector.elementAt(row);
}

public Component getTableCellEditorComponent(JTable table, Object obj, boolean bool, int row, int col)
[Codeposition 1.2.]
{
   return (Component)sliderVector.elementAt(row);
}
...
```

Quellcode 1: Auszug aus TableCellRenderer zur Anzeige von Bildern in einer JTable.

Renderer [Codeposition 1.1.] und Editor [Codeposition 1.2.] geben jeweils eine Instanz eines zuvor erzeugten JSliders zurück, welcher somit reihenweise zum Zeichnen an die Tabelle zurückgegeben werden.

4.2. Integration eines Scatterplot

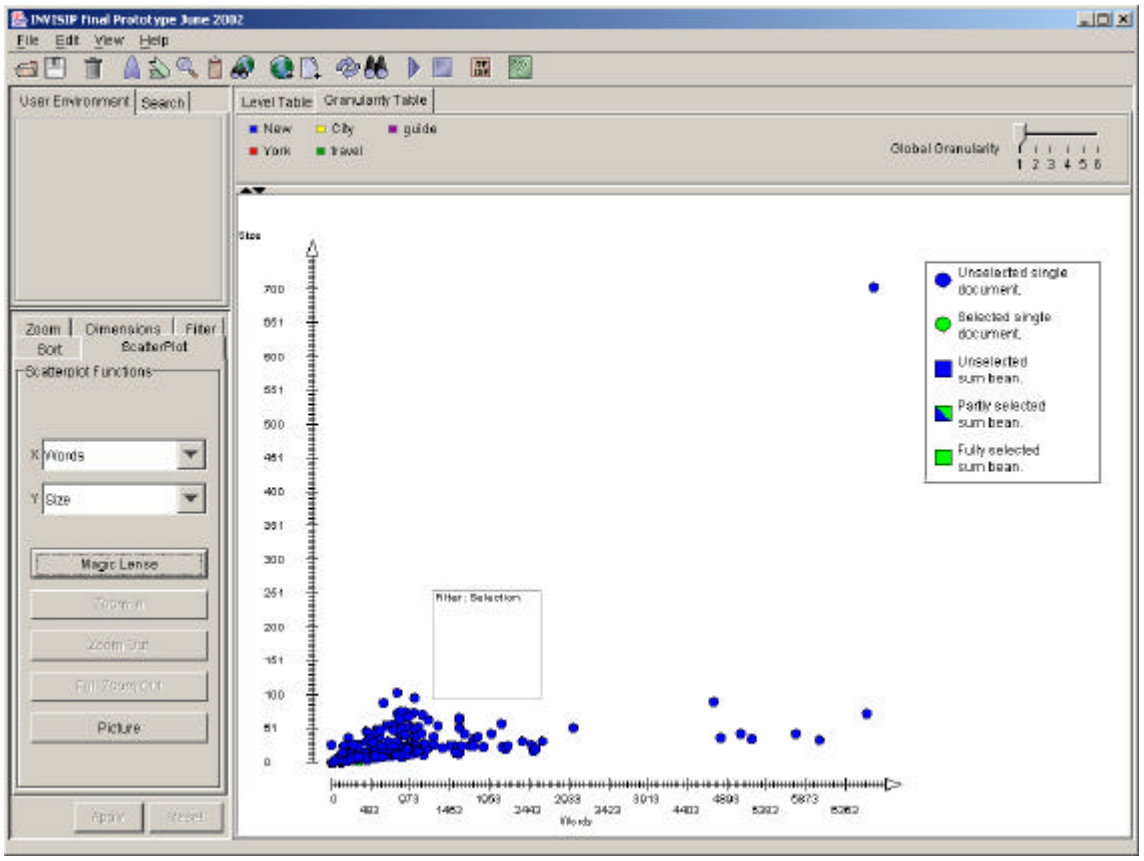

Abbildung 23: Scatterplot, Betaversion. Abbildung mit MagicLens.

Die GranularityTable soll vom Anwender zusammen mit einem Scatterplot [GF 2002] verwendet werden können. Der Scatterplot (Abbildung 23), der durch die Einbettung von JTable und sich selbst in ein JTabbedPane immer in den sichtbaren Bereich der Benutzeroberfläche geschoben werden kann, muss mit der Tabelle verbunden sein, so dass die Visualisierungen sich gegenseitig beeinflussen können. Für die Verbindung zwischen den zwei Komponenten kann auf bekannte Design Patterns zurückgegriffen werden.

4.3. Design Patterns und Objektorientierte Konzepte

Wenn zwei Visualisierungen zusammenspielen sollen, also Aktionen des Anwenders auf einer visuellen Komponente (View) eine Reaktion in allen anderen Views hervorrufen sollen, dann bietet sich eine Implementation des Softwareprojektes auf Basis des „Observer Patterns“[19] (auch „Model-View-Controller Pattern“[20] genannt) an.

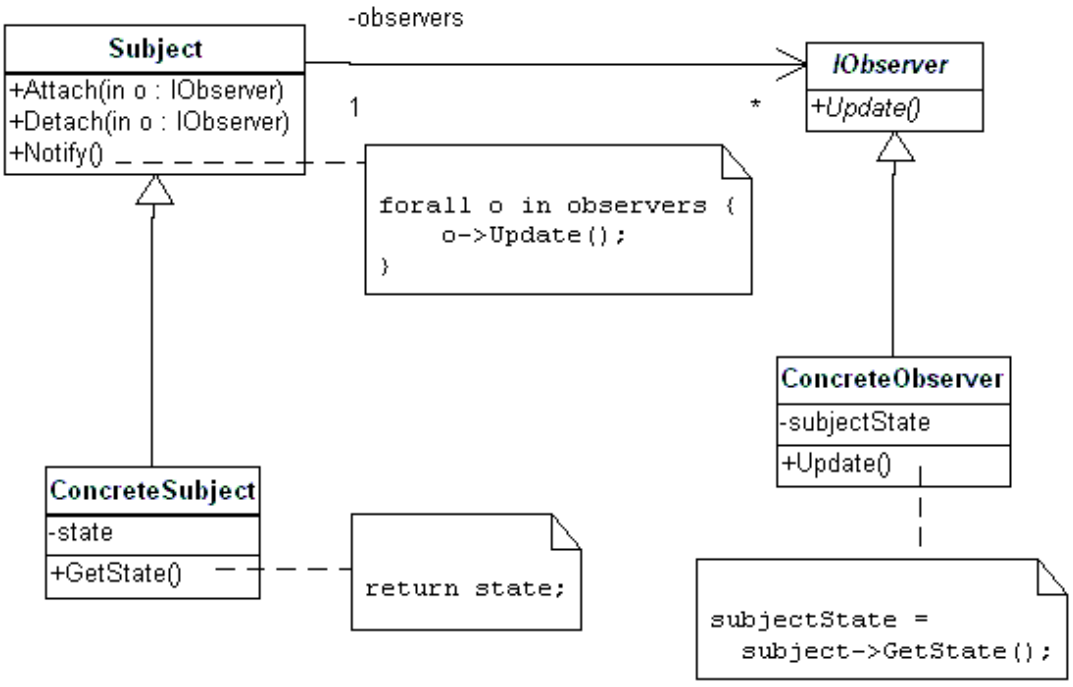

Abbildung 24: Observer Pattern, abstrakte Darstellung

[19] aus [GHJV 1994]
[20] aus [GHJV 1994]

Visualisierungen und das zugrundeliegende Datenmodell werde dabei voneinander getrennt. Über Interaktionen auf den verschiedenen Visualisierungen werden alle anderen Komponenten informiert, wenn diese einem Update unterzogen werden müssen [GHJV 1994]. GranularityTable und Scatterplot sollen konkrete Observer (Abbildung 24) bzw. Views (Abbildung 25) sein. Sie sind mit dem Subject, dem Datenmodell, verbunden, indem sie sich über die Methode „attach()“ an diesem anmelden.

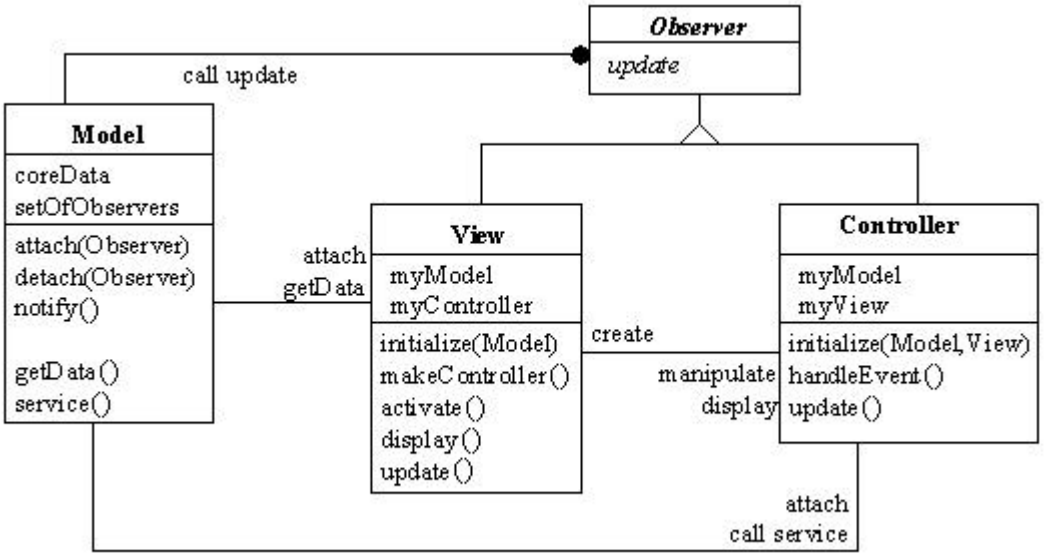

Abbildung 25: Model View Controller Konzept, detaillierte Darstellung

Eventhandler, die innerhalb der Visualisierungen auf Eingaben des Benutzers (wie zum Beispiel Mouseclicks oder Tastatureingaben) hören, sind einer jeden View zugeordnet. Die Eventhandler rufen am Datenmodell (Model) einen Service auf, wenn Manipulationen (Events) an der View ein Update anderer Komponenten zur Konsequenz haben muss. Der Service, der vom Controller via Call-Back Mechanismus aufgerufen wird, löst wiederum ein Update der registrierten Views aus. Das Observer Pattern kann auf zwei verschiedene Arten umgesetzt werden. Im Push-Model sendet das Subject detaillierte Informationen über das erforderliche Update an alle registrierten Observer, wohingegen bei Verwendung des Pull-Model die Observer nur minimale Informationen erhalten und selbst die für ein Update erforderlichen Daten vom Subject (Model) anfordern müssen. Bei der Anbindung der GranularityTable an das Datenmodell können Mechanismen sowohl des Push- als auch des Pull-Model verwendet werden.

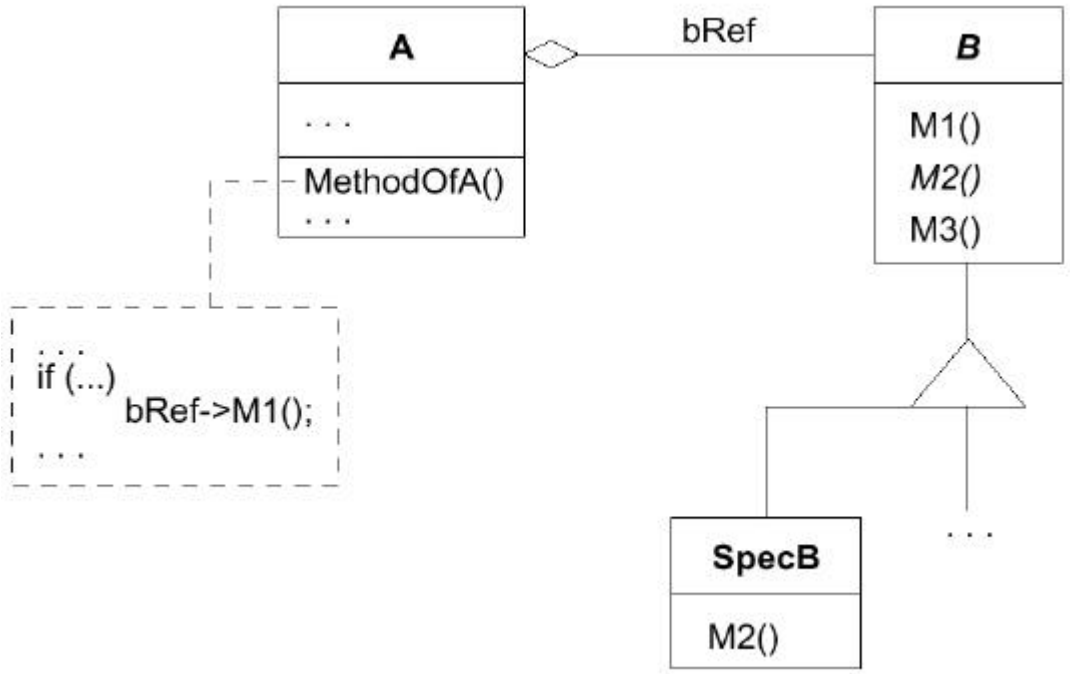

Abbildung 26: Abstrakte Kopplung

Die Subject-Klasse muss die konkreten Observer nicht kennen. Die Visualisierungen sind nur über eine abstrakte Klasse oder eine Interfaceklasse sichtbar. Die Verwendung des Observer Patterns geht deshalb mit der Verwendung des objektorientierten Konzeptes der abstrakten Kopplung[21] einher (Abbildung 26). Visualisierungen (Views) können daher leicht an die bestehende Anwendung angegliedert werden, indem diese als spezifische Klasse „SpecB“ die Eigenschaften der abstrakten Klasse „B“ erben oder das entsprechende Interface implementieren.

Bei einem Projekt mit sehr vielen Klassen kann es zu Problemen kommen, wenn es unerwünscht mehrere Instanzen einer Klasse gibt und unterschiedliche Komponenten mit verschiedenen Instanzen dieser Klassen arbeiten, obwohl alle mit der gleichen Instanz arbeiten sollten. Auch bei der Implementation der GranularityTable und aller zugehörigen anderen Komponenten, wie Datenmodell und Scatterplot, kann es zu solchen Konflikten kommen. Daher empfiehlt sich die Verwendung des „Singleton Patterns“[22]. Das Singleton Pattern (Quellcode 2) stellt sicher, dass es nur genau eine Instanz einer Klasse gibt. Der Konstruktor der betreffenden Klasse wird als private

[21] Abstrakte Kopplung aus [PW 1997]
[22] Aus [GHJV 1994]

deklariert, wodurch die Erzeugung einer Instanz dieser Klasse aus anderen Klassen und Paketen heraus nicht mehr durch den Aufruf des Konstruktors möglich ist.

```
class Singleton
{
	private static Singleton instance = null;

	private Singleton()
	{
		…
	}
	public static Singleton instance()
	{
		if (instance == null)
		{
			instance = new Singleton();
		}
		return instance;
	}
}
```

Quellcode 2: Singleton Pattern Codebeispiel

4.4. Konzepte der Information Visualization

4.4.1. Knowledge Crystallization

Das Konzept der GranularityTable soll zusammen mit den integrierten Visualisierungen (Kapitel 4.4.2 ff.) möglichst effektiv das Information Retrieval auf geo-räumlichen Metadaten unterstützen. Der *Knowledge Crystallization* Zyklus (Abbildung 27) beschreibt den Prozess, der bei einer stark informationslastigen Aufgabe vom Benutzer durchlaufen wird. Im INVISIP Projekt hat der Benutzer Aufgabenstellungen (Task) in der Anwendungsdomäne Geographie und geo-räumlichen Planung zu erledigen. Die Benutzeroberfläche der Anwendung muss die Phasen des Kreislaufes unterstützen, um den Retrievalprozess optimal zu unterstützen.
Die Sammlung (Information Foraging) von relevanten Dokumenten hinsichtlich der Suchanfrage übernimmt ein Crawler mit Hilfe der Geo-Metadaten. Der Benutzer kann zu diesem Zeitpunkt das Granularitätskonzept verwenden, um Überblick (Overview) und Details (Details-On-Demand) anzuzeigen, sowie in Interaktion mit dem Scatterplot [GF 2002] Filter- und Zoomoperationen durchführen. Die Datensätze können innerhalb der GranularityTable mit Hilfe der verschiedenen Visualisierungen oder im Scatterplot

auf Basis ihrer Position im Koordinatensystem verglichen werden (Search For Scheme).

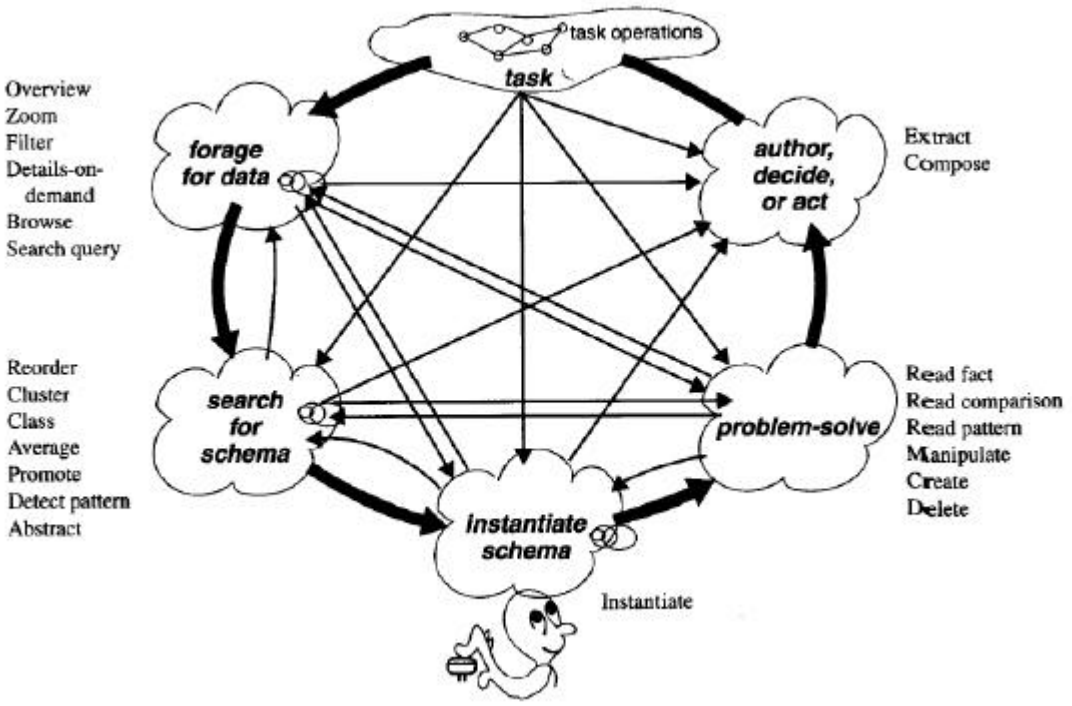

Abbildung 27: Knowledge Crystallization Cycle

Da alle Visualisierungen systemseitig schon für alle Datensätze vorhanden sind, kann das Schema zum Vergleich der Daten sofort angewendet werden (Instantiate Scheme). Datensätze, die vom Benutzer als nicht relevant beurteilt worden sind, können innerhalb der GranularityTable durch Selektion bzw. De-Selektion als solche kenntlich gemacht und dadurch von anderen Datensätzen unterschieden werden (Problem-Solve). Um die Entscheidung für einige wenige der Datensätze fällen zu können, können andere Visualisierungen als Vergleichschemata verwendet werden (Reduce Problem To More Simple Trade Off). Der globale Granularitätsschieber erleichtert das Umschalten in andere Darstellungen für alle Datensätze. Die Datensätze, die im Vergleich am Besten abschneiden, werden extrahiert (Author, Decide, Act).
Die Technik der TableLens ist ein Tool der Information Visualization mit deren Hilfe Schemata gerade auch bei sehr vielen Datensätzen instanziiert und Variablen hinsichtlich einer Problemlösung manipuliert werden können.

4.4.2. Verwendung einer abgewandelten TableLens

Bei der Umsetzung der Eiblschen Redesign Idee erhalten hinsichtlich des Granularitätskonzeptes neue Konzepte des Document- und Information Retrieval Einzug in das INVISIP Projekt. Genauso wird aber auch bezüglich der in die Tabelle integrierten Visualisierungen auf bestehende Konzepte der Information Visualization zurückgegriffen.

Der INVSISP Testdatensatz besteht aus 600 XML Dateien und spiegelt mit dieser Menge diejenige Menge von Dokumenten wider, welche das System dem Anwender in der Regel maximal als Suchergebnismenge anbieten wird. Angelehnt an die Konzeption der ersten Granularitätsstufe nach M. Eibl [EM 2001] /[MKRE 2002] werden die Dokumente zunächst nur in sehr niedrigen Zeilen angezeigt und es werden lediglich schmale BarCharts zu sehen sein, welche dem Benutzer die Relevanz des Dokumentes beispielsweise bezüglich eines einzelnen Wertes anzeigt.

In der ersten Detailstufe ist aufgrund der geringen Tabellenzellenhöhe auch der Granularitätsschieber nicht sichtbar. M. Eibl schlägt vor, den Slider bei Überfahren mit der Maus in einer Art Popup anzuzeigen und so dem Anwender die Bedienung des Reglers und damit verbunden das Umschalten in andere Granularitätsstufen zu ermöglichen. Bei der geringen Zeilenhöhe und der Vielzahl an Datenzeilen in der Tabelle, kann ein Popup für Verwirrung sorgen, wenn der Benutzer nach dem MouseOver nicht genau zuordnen kann, zu welcher Zeile der Popup gehört. Dieser Effekt wird durch die Masse des Granularitätsschiebers zusätzlich unterstützt und ist bereits in der Planungsphase hinsichtlich der Usability eines späteren Prototypen als kritisch anzusehen. Eine bessere Lösung besteht darin, nicht nur den Granularitätsregler aufpoppen zu lassen, sondern die gesamte Datensatzzeile bei Überfahrt des Mauszeigers so zu vergrößern, dass der Regler in erforderlicher Größe verfügbar ist. Ein vorteilhafter Nebeneffekt einer Größenveränderung der ganzen Tabellenzeile ist, dass bei entsprechender Vorgabe einer passenden MouseOver-Zeilenhöhe bereits Text zu erkennen ist, wie zum Beispiel der Titel des Dokumentes. So kann bereits ohne Umschalten der Granularität eine benutzerseitige Filterung relevanter Dokumente aufgrund der BarCharts und des Titels erfolgen. Die Integration eines solchen MouseOver Effektes ist in der Information Visualization als TableLens [CR 1999] bekannt.

Die TableLens ist eine neue Form der Visualisierung, die [CR 1999] für die Verwendung in Tabellen entwickelt hat. Normale Tabellen können in Abhängigkeit von der Größe und der Auflösung des Computerbildschirmes nur eine beschränkte Anzahl von Tabellenzeilen und Spalten anzeigen. Die Zeilenhöhe kann in normalen Anwendungen nach dem Einfügen der Inhalte nicht verändert werden, so dass die Tabelle und deren Inhalte in statischer Form vorliegen und die Anzeige der Tabellenzeilen in der Größe nicht mehr veränderbar ist.
Die TableLens ist eine „Focus+Context-“ oder „Fisheye-“ Technik, mit deren Hilfe mehr Tabellenzeilen pro Bildschirm angezeigt werden können, indem das Layout der Tabelle dynamisch geändert wird.

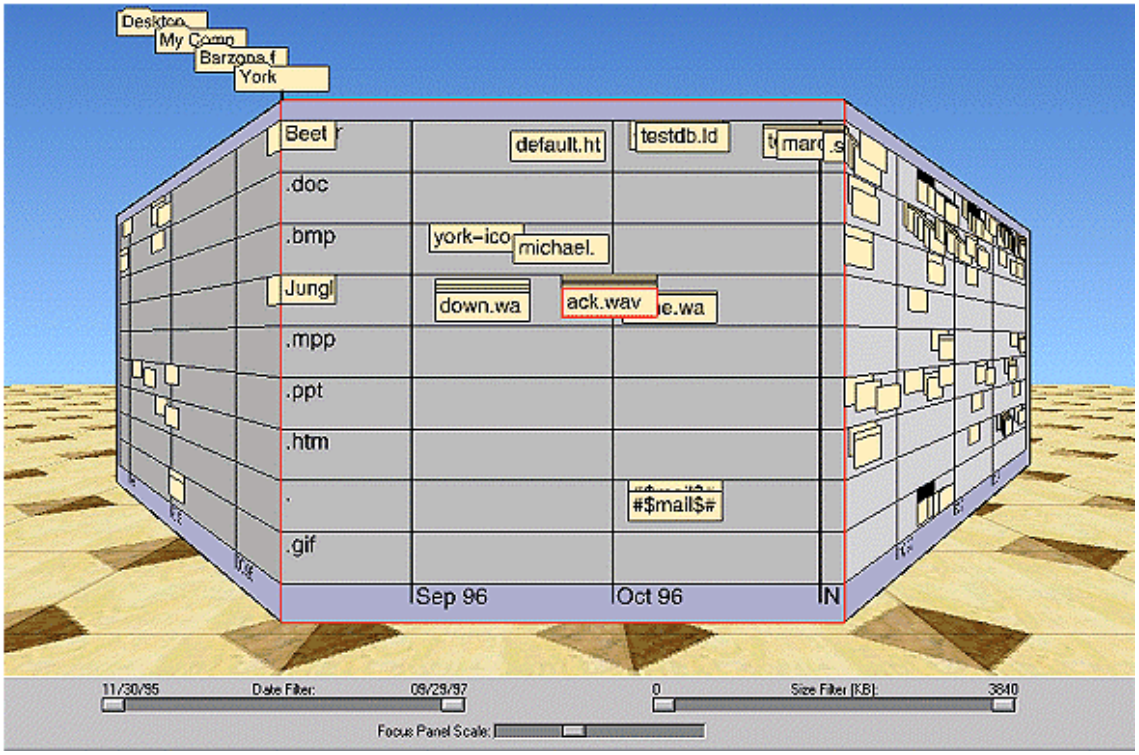

Abbildung 28: Perspective Wall

Focus+Context Techniken ermöglichen die Darstellung einer gesamten Informationsstruktur, wie z.B. einer Tabelle, auf einem Bildschirm. Sie helfen bei der Suche nach relevanten Datensätzen durch Visualisierung, wie z.B. BarCharts mit bestimmter Länge und es können Zoom-Operationen auf ausgewählten Elementen der Tabelle ausgeführt werden. Es existieren bereits mehrere solcher Techniken, umgesetzt beispielsweise in der Document Lens [RM 1993] oder der Perspective Wall (Abbildung 28) [MRC 1986].

Der TableLens Mechanismus (Abbildung 29) verändert die Tabellezellen in Ihrer Höhe und Breite unter Verwendung der DOI Funktion [FG 1986].

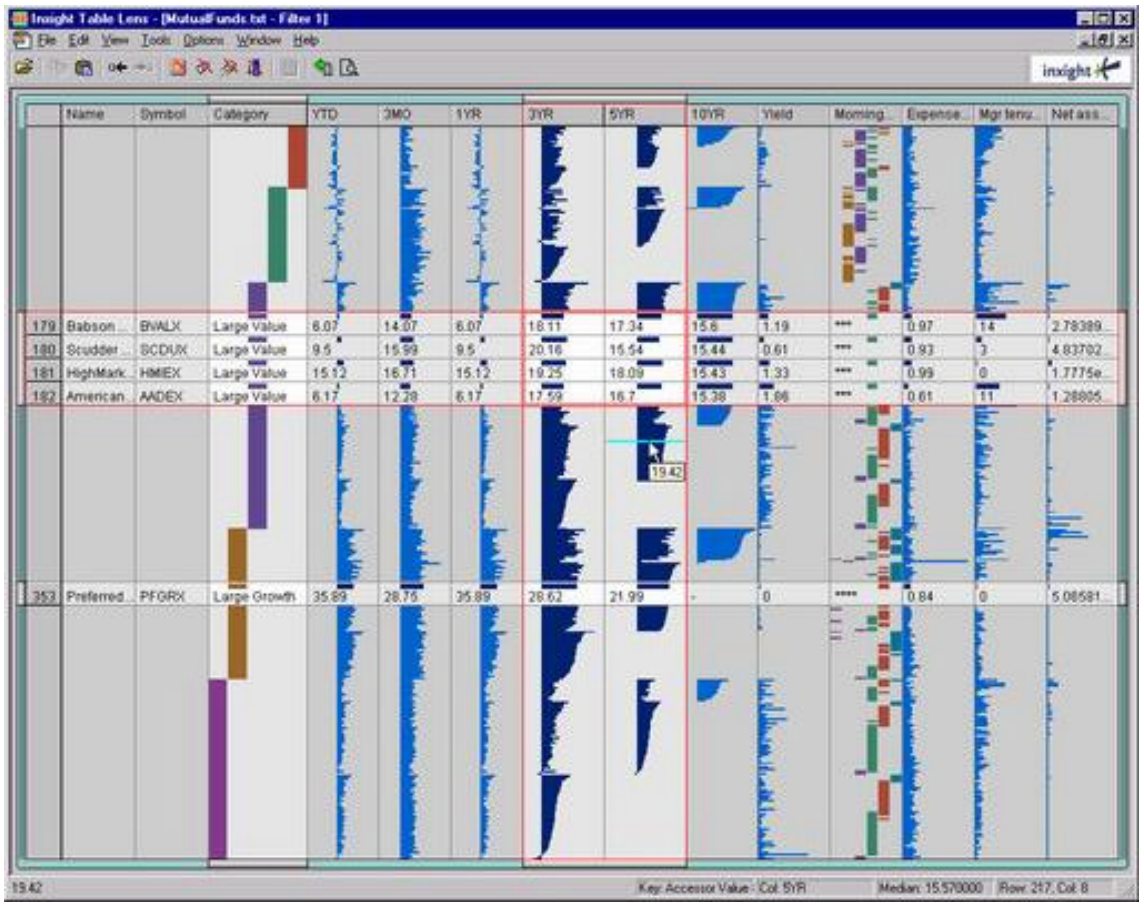

Abbildung 29: INXIGHT Table Lens

Demnach wird bei Überfahren mit dem Mauszeiger ein fokaler Bereich vergrößert und entsprechend müssen die übrigen Tabellenzeilen im nicht-fokalen Bereich in ihrer Größe angepasst werden, damit die Tabelle weiterhin mit dem zur Verfügung stehenden Platz auskommt. Die TableLens bietet im Besonderen drei kanonische Manipulationsmöglichkeiten an. Mit der Zoom Funktion (Zooming) können die Elemente innerhalb des fokalen Bereiches nochmals vergrößert werden, d.h. die Tabellezeilen werden höher, der Inhalt wird erweitert oder nutzt das Plus an Platz sinnvoll aus. Mit einer Anpassungsfunktion (Adjustment) können innerhalb eines fokalen Bereichs gleichbleibender Größe mehr Elemente angezeigt werden. Durch die Slide-Funktion (Sliding) kann der Anwender den fokalen Bereich innerhalb der Tabelle zu anderen Zeilen hin verschieben. Außerdem können mehrere fokale Bereiche

gleichzeitig unterstützt werden, indem ein vergrößerter Bereich an einer bestimmten Stelle verbleibt und der Benutzer an einer anderen Stelle eine neue Linse erzeugt.
Die TableLens Technik kann auch in der GranularityTable verwendet werden. Im Zusammenspiel mit dem Granularitätskonzept und dem Granularitätsschieber verändert sich jedoch der erforderliche Funktionsumfang der TableLens Technik.
Bei einer sehr großen Anzahl von Datensätzen wird es nicht mehr möglich sein, alle Tabellenzeilen auf nur einer Bildschirmseite und somit ohne Scrollbalken darzustellen. Aus diesem Grund kann bei der GranularityTable auf die automatische Größenveränderung der Zellen im non-fokalen Bereich unter Berechnung durch die DOI Funktion verzichtet werden. Stattdessen kann für Zellen bzw. Zeilen außerhalb des Benutzerfokus eine feste Größe definiert werden. Eine Tabellenzeile soll durch die Linse so vergrößert werden, dass der Granularitätsschieber in einer Größe erscheint, die dessen Benutzung erlaubt. Der fokale Bereich in der GranularityTable soll sich zunächst nur auf eine einzelne Tabellenzeile und nicht auf mehrere beziehen. Die Adjustment-Funktion fällt dadurch aus dem Manipulationspotential heraus. Das hinein- und herauszoomen in Tabellenzeilen mit Hilfe der TableLens wird innerhalb des GranularityTable ebenfalls in anderer Form umgesetzt sein. Die Zoom Operation übernimmt in diesem Fall der Granularitätsschieber, der durch die einfache Zoomfunktion so freigelegt ist, dass der diese Manipulationsmöglichkeit in Vertretung für die TableLens übernehmen kann.
Mehrfache fokale Bereiche kann die GranularityTable dadurch unterstützen, indem Tabellenzeilen durch Manipulation des Granularitätsschiebers in die nächstgrößere Detailstufe versetzt werden und dadurch vergrößert bleiben, auch wenn der Anwender den fokalen Bereich anschließend von der Tabellenzeile wegbewegt.
Die Konzepte von TableLens und GranularityTable überschneiden sich hinsichtlich der Manipulationsmöglichkeiten für die zugrundeliegende tabellenbasierte Visualisierung. Da der Schwerpunkt des INVSISP Projektteils auf der Umsetzung des neuen Granularitätskonzeptes liegt, wird die TableLens im Funktionsumfang zugunsten des Granularitätsschiebers eingeschränkt. Die TableLens bezieht sich bei der GranularityTable nur auf ganze Tabellenzeilen und insgesamt kann im Zusammenhang mit der GranularityTable nur von einer veränderten und vereinfachten TableLens gesprochen werden.

4.4.3. Referenzmodell für Visualisierung

Beim Information Retrieval werden Teilmengen von rohen Daten aufgrund Ihrer Relevanz bezüglich einer Suchanfrage extrahiert. Ein Information Retrieval System ist dann um so effektiver, wenn die Suchergebnisse für den Benutzer so aufbereitet werden, dass dieser möglichst schnell und ohne hohe Belastung eine Auswahl treffen kann. Den Prozess der Umwandlung der Daten in eine benutzerfreundliche und unterstützende Ansicht beschreibt das Referenzmodell für Visualisierung (Abbildung 30) [CMS 1999]. Wichtig bei der Umsetzung der GranularityTable sind die visuellen Strukturen (Visual Structures), die später in die Tabelle (View) integriert werden (Kapitel 4.4.4 ff.). Diese müssen vom Benutzer schnell und einfach verstanden, sowie verarbeitet werden können. Eine visuelle Struktur ist um so effektiver, um so schneller sie interpretiert werden kann und um so mehr Möglichkeit des Vergleiches sie offeriert. Durch Transformationen (View Transformations, siehe auch Kapitel 5.3.4.2) können die Views modifiziert werden.

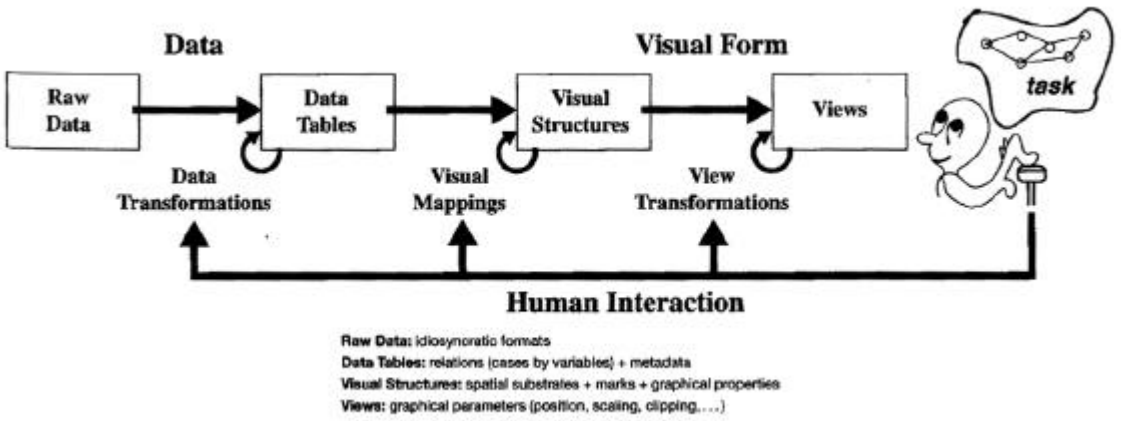

Abbildung 30: Reference Model For Visualization

Der TableLens Mechanismus aus Kapitel 4.4.2 oder die MagicLens des Scatterplot [GF 2002] sind solche View Transformation Techniken zur Modifikation der Visualisierung.

4.4.4. Verwendung von BarCharts

BarCharts sind auch als BarGraphs bekannt. Sie stellen eine Visualisierung dar, die besonders für das Ranking von Datensätzen hilfreich ist. BarCharts werden verwendet, um Information über einzelne Dokumente und die gesamte Ergebnismenge in Erfahrung zu bringen. Für jeden einzelnen Datensatz zeigen vertikal oder horizontal angeordnete Balken beispielsweise die Gesamtrelevanz eines Dokumentes an, sowie dessen Einzelrelevanzen hinsichtlich einzelner Keywords. Umso höher bzw. länger ein Balken ist, desto relevanter ist das Dokument bezüglich des abgetragenen Wertes in Relation zur gesamten Ergebnismenge. Wird ein Balken für einen Wert nicht angezeigt, bedeutet dies, dass das Dokument für diese Eigenschaft keine Relevanz aufweist und/oder keine Information zu dem entsprechenden Keyword beinhaltet.
Wie auch die SuperTable des INSYDER Projektes, beinhaltet die GranularityTable BarCharts in den niedrigen Detailstufen der tabellenbasierten Visualisierung. Die Verwendung der BarCharts im INSYDER Projekt (Abbildung 31) wurde durch die Arbeit von Veerasamy und Belkin [VB 1996] inspiriert.

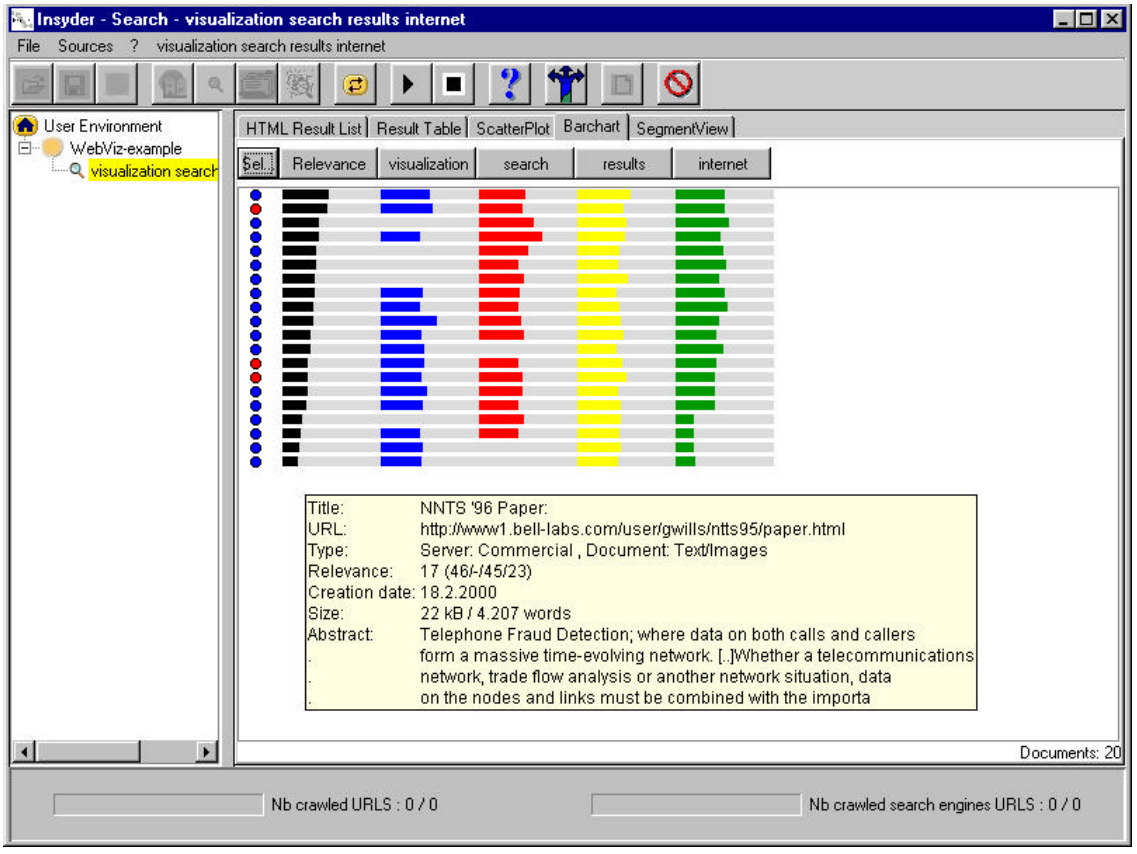

Abbildung 31: BarCharts / BarCharts im INSYDER

4.4.5. Verwendung von Tile Bars

Das Eiblsche Granularitätskonzept integriert verschiedene Visualisierungen in eine Tabelle, darunter auch sogenannte TileBars, die von [HM 1995] entwickelt wurden. Tile Bars unterstützen die einfache Erkennung wichtiger ausgewählter Textmerkmale. Sie sind eine Visualisierung, die die Länge eines Dokumentes darstellt, sowie gleichzeitig die Häufigkeit, Verteilung und Relevanz von Suchbegriffen innerhalb eines Dokumentes und dessen Segmenten. Jede Zeile der TileBar ist dabei unterschiedlich farblich kodiert. Eine TileBar Zeile stellt beispielsweise genau ein Keyword dar, die einzelnen Tiles der Zeile repräsentieren Relevanz und Verteilung. Die Relevanz kann in Stufen abgebildet werden, indem die Tiles in unterschiedlichen Farbstärken gefüllt werden. Die visuelle Kodierung der TileBar kann vom Benutzer überprüft werden, wenn die TileBar eine „Jumpfunktion“ anbietet. Beim Mausklick auf eine einzelne Tile springt der Sichtbarkeitsbereich des zugehörigen textuellen Bereichs dann an die entsprechende Stelle, ähnlich der Umsetzung im INSYDER (Abbildung 32) [MR 1999]. TileBars erhalten ihre Daten und somit ihre visuelle Struktur aufgrund eines Algorithmus bzw. aus den numerischen Ergebnissen einer algorithmischen Berechnung.

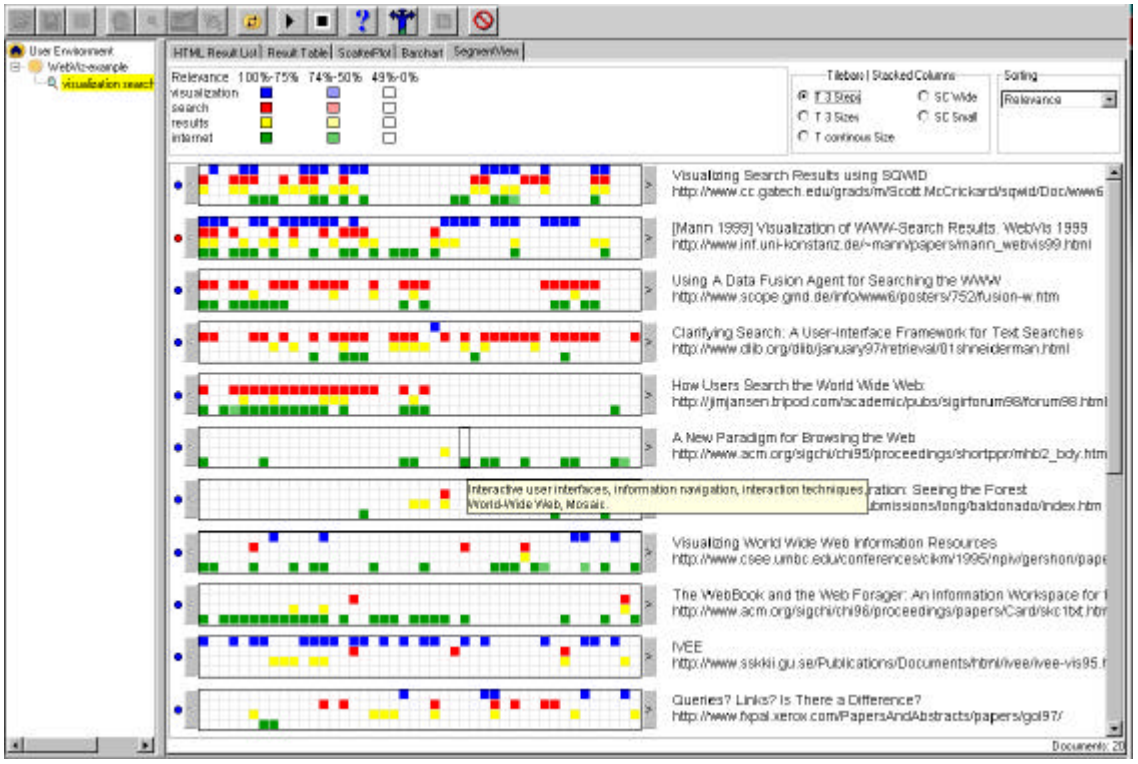

Abbildung 32: TileBars im INSYDER

4.4.6. Verwendung von Thumbnails

In Granularitätsstufe Sechs der tabellenbasierten Visualisierung nach [EM 2001] /[MKRE 2002] soll eine Thumbnail View mit Navigationsfunktion den Retrievalprozess unterstützen. Beispiele für die Verwendung von Thumbnails zur visuellen Identifikation von Keyword Verteilungen innerhalb von Dokumenten sind Systeme wie die Document Lens [RM 1993] oder ein System von Kaugars [KK 1998].

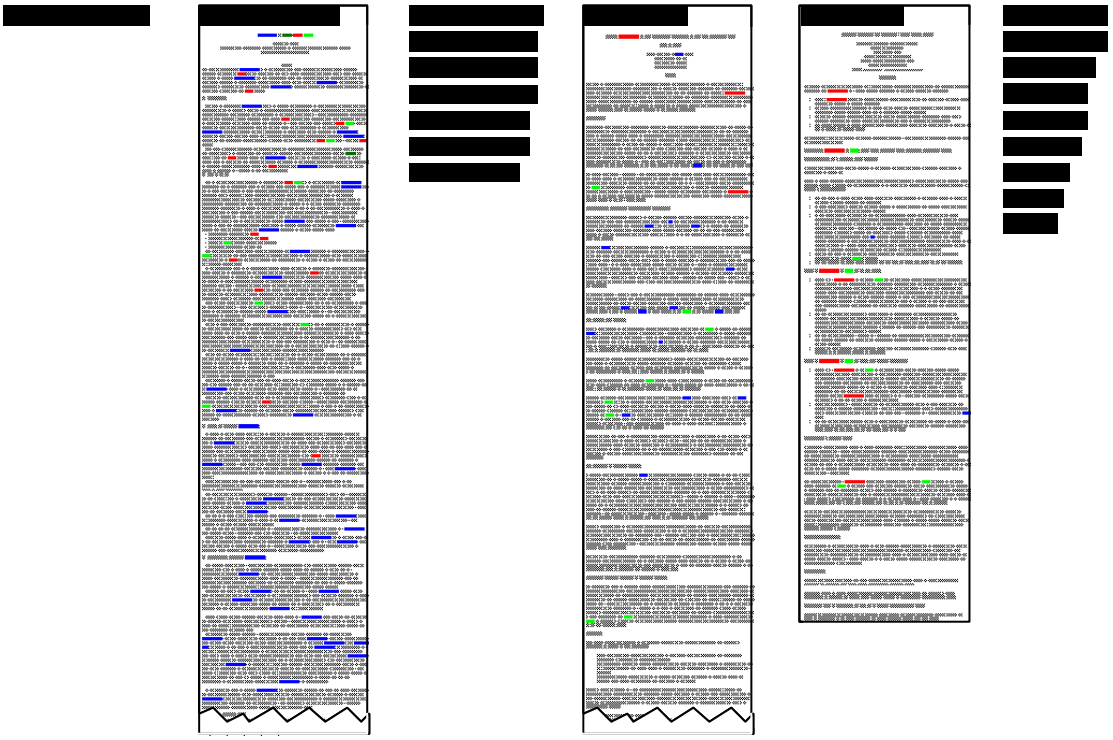

Abbildung 33: System von Kaugars mit Thumbnail View Technik

Kaugars' System verwendet vier verschiedene Abstraktionslevel, in denen Dokumente entweder geschlossen sind, als Thumbnails zur Verfügung stehen, oder zur Hälfte bzw. ganz geöffnet sind. Abbildung 33 zeigt drei geöffnete Dokumente in einer Thumbnail View, in der die im Text gefundenen Keywords jeweils hervorgehoben sind. Eine Thumbnail View nach dieser Idee soll in der GranularityTable dem Benutzer eine Übersicht über den Text verschaffen und die im Text markierten Schlüsselwörter sollen als Anker dienen, welche bei einem Mausklick den textuellen Bereich der Tabellenzeile an die entsprechende Stelle im Dokument springen lassen. Ähnlich wie die TileBars zeigen die Thumbnail Views somit die Struktur des Dokumentes sowie dessen Relevanz an und stellen gleichzeitig eine Navigationsfunktion für den Volltext zur Verfügung.

5. Implementation der GranularityTable

Folgende Ausführungen beschreiben die Implementation der Aufgabenstellung aus dem Pflichtenheft. Da die GranularityTable auf bestehenden Komponenten wie dem Datenmodell aufbaut und mit anderem Komponenten des INVISP Projektes eng gekoppelt ist, werden auch kurze Beschreibungen im Zusammenhang mit den verschiedenen Teilsystemen angeführt, sowie Verweise auf die Systemdokumentation der anderen Komponenten zu finden sein.
Dies geschieht aufgrund der Tatsache, dass durch die Verbundenheit zwischen den einzelnen Projektteilen, regelmäßig Ergänzungen und Veränderungen an den anderen, angeschlossenen Systemkomponenten notwendig waren.

5.1. Datenquelle : XML Datei

Der INVISIP Prototyp dient zur Darstellung von Metadaten, später speziell zur Visualisierung von Geo-Metadaten. Auf Prototypebene und bis zum Zeitpunkt der Festlegung eines Datensatzformates für die Metadaten kann mit einem Testdatensatz beliebigen Inhalts gearbeitet werden. Der INVISIP Prototyp verwendet in der Design- und Entwicklungsphase XML Daten, welche verschiedene Metainformationen über Dokumente aus dem World Wide Web enthalten. Die XML Dateien (Quellcode 3) enthalten sowohl textuelle Metainformationen über das Dokument, als auch numerische Informationen, etwa zum Zeichnen von Stabdiagrammen (BarCharts) oder Relevanzkurven (RelevanceCurve). Alle Metainformationen entsprechen den Auswertungen des INSYDER Evaluationsagenten. Die XML Dateien sind ein Datenbankexport im Anschluss an eine Suchanfrage in der INSYDER Anwendung. Die Zahlenfolgen von Keyword Relevanzen und anderen Werten müssen nicht bei jedem Start des INVISIP Prototypen neu berechnet werden, sondern die Applikation kann auf die bereits durch INSYDER aufbereiteten Daten aus der XML Datei zurückgreifen. Dadurch kann in der Entwicklungsphase eine noch stärkere Konzentration auf Information Visualization und Usability des Softwareproduktes stattfinden.

```
<Insyder_Export>
	<Index>141</Index>
	<Selection>false</Selection>
	<DocumentRead>false</DocumentRead>
	<Relevance>34</Relevance>

	<keywords>
		<New>12</New>
		<York>0</York>
		<City>82</City>
		<travel>68</travel>
		<guide>77</guide>
	</keywords>

	<TFIDF>0</TFIDF>
	<Title>CNY : liens</Title>
	<LocalURL>c:\Insyder2\Repository\f_7\p_2018.htm</LocalURL>
	<URL>http://fr.srd.yahoo.com/goo.fr/…</URL>
	<Date>1.1.1970</Date>
	<Language>English</Language>
	<Size>26</Size>
	<Words>0</Words>
	<DocumentType>Frameset</DocumentType>
	<Servertype>Miscellaneous</Servertype>
	<Abstract>-</Abstract>
	<SegmentText></SegmentText>

	<RelevanceCurve>
		27,27,27,2,2,2,27,16,4,2,2,16,4,2,42,5,7,27,28,5,2,2,2,42,30,27,5,25,3,2,2,3,33,2,2,8,3,
		3,14,2,42,2,42,3,12,2,14,2,3,2,27,11,27,2,3,3,3
	</RelevanceCurve>

	<keyword_segments>
		<NewSegments>
			0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
			,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,…
		</NewSegments>
		<YorkSegments>
			0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
			,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,…
		</YorkSegments>
		<CitySegments>
			100,2,2,4,4,100,2,4,4,4,4,4,4,4,4,10,4,4,4,100,4,10,4,4,4,4,100,4,4,4,4,28,4,2,
			2,100,100,4,100,100,100,100,100,…
		</CitySegments>
		<travelSegments>
			99,2,2,94,99,99,17,2,2,2,2,2,2,2,2,2,2,2,2,2,2,16,2,87,87,87,2,2,2,2,87,99,9,2,
			2,99,94,87,9,2,9,90,94,99,9,99,9,…
		</travelSegments>
		<guideSegments>
			2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,74,74,2,2,
			2,74,74,74,74,74,2,2,2,2,74,74,2,…
		</guideSegments>
	</keyword_segments>

</Insyder_Export>
```

Quellcode 3: INVISIP Prototyp XML Datei

5.2. Das Datenmodell

Herzstück des gesamten INVISIP Projektes und somit auch die Aorta der GranularityTable ist das zugrundeliegende Datenmodell.

5.2.1. Der XML Parser und das FileIO Package

Die Metadaten über Dokumente aus dem World Wide Web liegen als XML Dateien vor und werden vom „XMLParser“ des Datenmodells ausgelesen. Für verschiedene Bestandteile (Metatags) des XML Files existieren auch spezielle Datentypen im Datenmodell, welche mit dem Inhalt von passenden Tags gespeist werden.
Der XML Parser besitzt einen „ParseManager“. Dieser beauftragt weitere Klassen, wie den „DataElementBuilder“, mit dem Aufbau eines sogenannten „WWWDocument“. Dieses setzt sich nun aus solchen speziellen Metadatentypen, wie zum Beispiel „IntegerMetaData“ (für Integer Werte) oder „StringMetaData“ (für kleinere Texte), zusammen und repräsentiert genau eine XML Datei mit allen enthaltenen Daten.
Wie im Verlaufe des Projektes festgestellt wurde, ist die farbige Darstellung von Text zur Laufzeit der Visualisierung zu langsam und erzeugt zu lange Wartezeiten für den Benutzer. Daher dekoriert der ParseManager den Text des Dokumentes (Volltext und Abstract) bereits dann, wenn die Daten aus dem XML Dateien ausgelesen und zu WWWDocuments zusammengefügt werden. Dabei steht dem ParseManager ein komplexerer Datentyp namens „DecoratedStringArrayDocument“ zu Verfügung, welcher die dekorierten, textuellen Metadaten eines Dokumentes direkt in einem „DefaultStyledDocument“ abspeichert.
Der ParseManager steuert während des Aufbau- und Dekorationsprozesses einen „ProgressMonitor“, der dem Benutzer nach Systemstart den Einlesevorgang der XML Dateien und dessen Fortschritt anzeigt.
Alle gespeicherten Dokument Datensätze werden über den „DataManager“ (Abbildung 34) zugänglich gemacht, indem der Caller über die Methode „getDataContainer()“ auf den „DataElementContainer“ zugreift. Alle Komponenten, die mit den Datensätzen arbeiten, kennen und nutzen den DataManager für den Datenzugriff, wie zum Beispiel auch der Scatterplot [GF 2002].
Eine Zeile der späteren GranularityTable wird später genau einem WWWDocument entsprechen, die Tabellenspalten für Visualisierung und Textanzeige werden durch die

im WWWDocument enthaltenen Daten gespeist und gezeichnet. Der Zugriff auf die Metadaten, die angezeigt werden sollen, erfolgt nach diesem Prinzip aus den Visualisierungen heraus über TableModel, TableCellRenderer und DataManager.

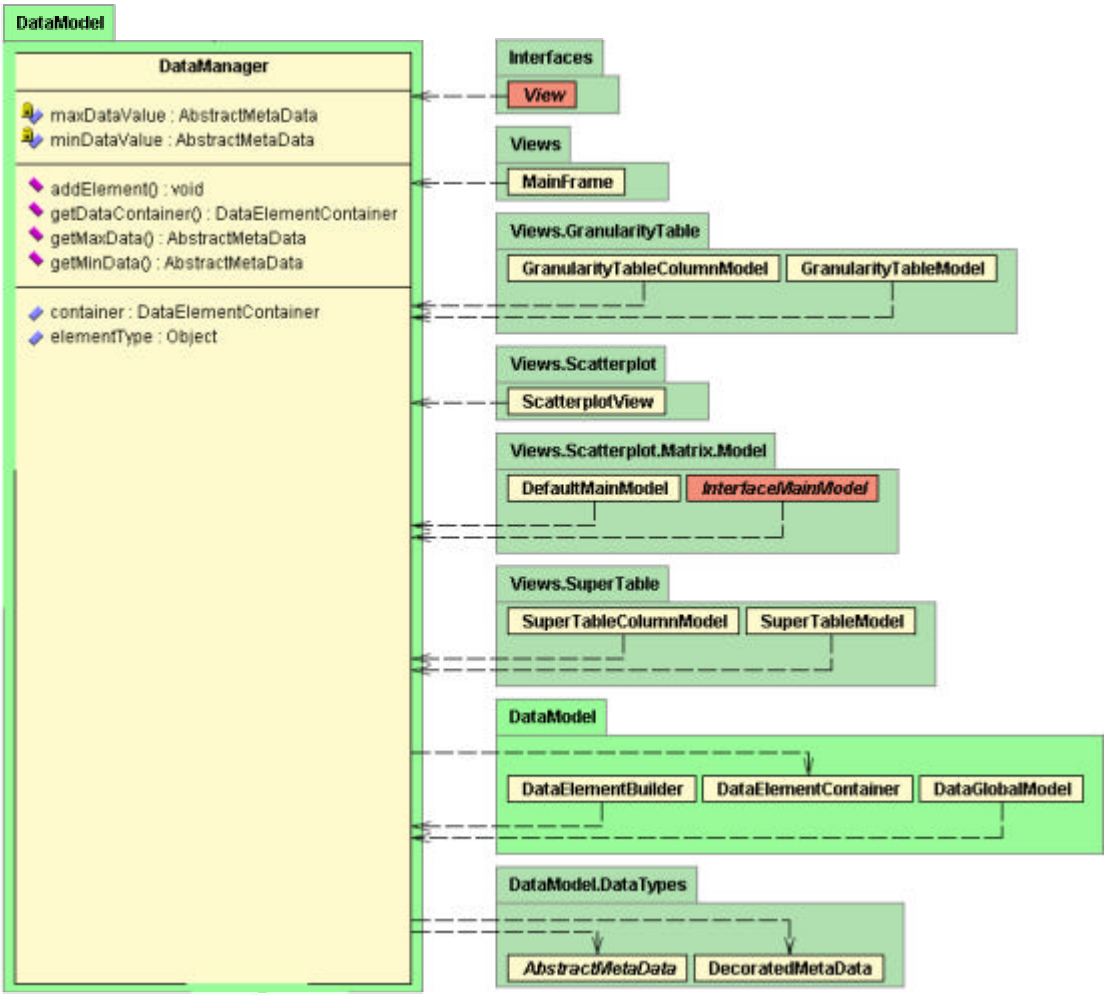

Abbildung 34: Klasse DataManager.java UML Ausschnitt

5.2.2. Das DataGlobalModel im DataModel Package

Das „DataGlobalModel“ ist das eigentliche Kernstück des gesamten Datenmodells. Entsprechend der Entscheidung, die Applikation nach dem Model-View-Controller Konzept aufzubauen, wird ein Model (Subject) benötigt, welches angemeldete Views über erforderliche Updates informiert. Eben dieses Subject ist die Klasse DataGlobalModel. Unmittelbar nach Programmstart melden sich die erzeugten Views am DataGlobalModel an. Nachdem der XMLParser gestartet und die Datensätze eingelesen worden sind, sendet das DataGlobalModel eine Instanz des aktuellen

DataManager an alle Views, die anschließend ihre Oberfläche entsprechend der im DataManager geänderten Daten umbauen und neu zeichnen.
Das DataGlobalModel (Abbildung 35) implementiert das Interface „Model“ und setzt somit die Idee der abstrakten Kopplung um. Einige Komponenten werden das Kernstück der Applikation nur durch dieses Interface ansprechen können.

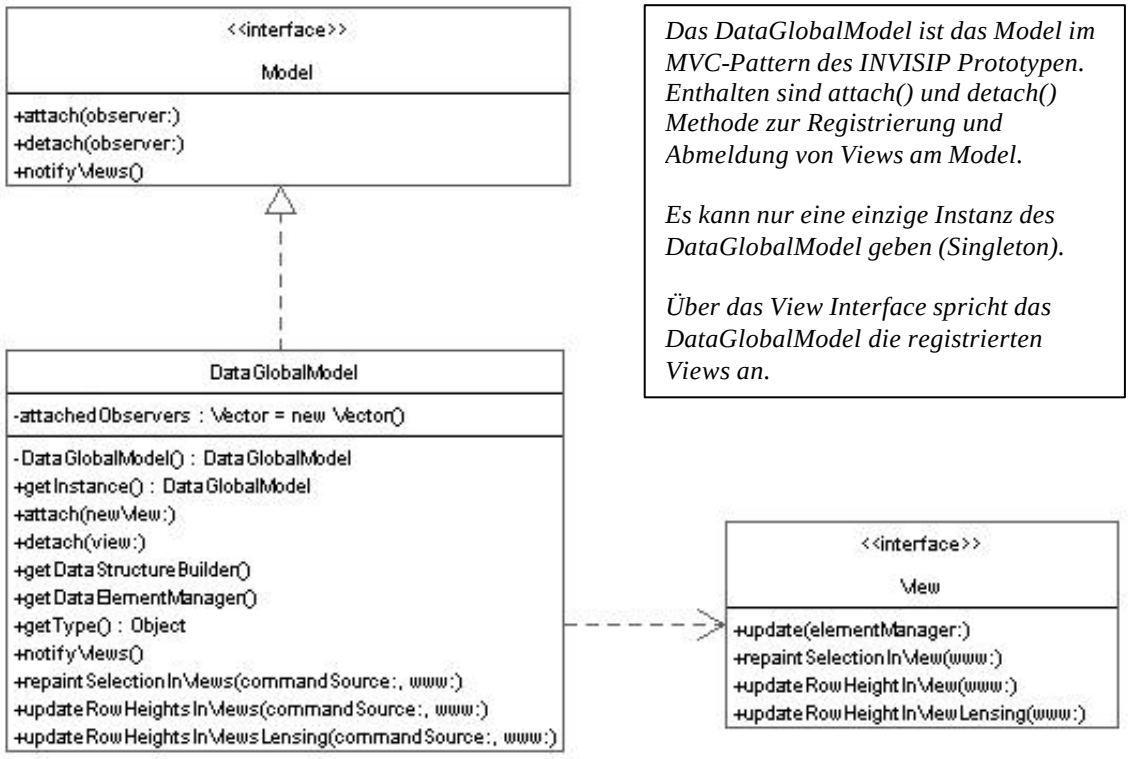

Abbildung 35: Klasse DataGlobalModel.java UML

Außerdem kann nur eine einzige Instanz vom DataGlobalModel erzeugt werden, da die Instanziierung der Klasse, entsprechend der Überlegungen während der Konzeptionsphase, nur über das Singleton Pattern möglich ist.
Das DataGlobalModel informiert andere Views, wenn in einer Visualisierung ein Dokument zum Beispiel per Mausklick selektiert worden ist. Dies geschieht vor allem aufgrund der engen Verbindung zwischen Scatterplot und GranularityTable. Werden im Scatterplot ein oder mehrere Dokumente ausgewählt, wird die Tabelle über diese Änderung des Status eines jeden betroffenen Dokumentes informiert und zeichnet Elemente der Tabelle neu, so dass die Selektion des Dokumentes auch in der Tabelle sichtbar wird. Die Methode „repaintSelectionInViews(...)“ ruft der Reihe nach die Services in den registrierten Views auf, die dieses Update durchführen. Der Scatterplot

unterstützt mehrere Funktionen, die dem Prinzip der direkten Manipulation entsprechen. Dazu gehört zum Beispiel die Zoom Funktion, mit der im Koordinatensystem des Scatterplot Punktemengen und deren Umgebung vergrößert, während andere Punkte Cluster, die nicht im vom Benutzer ausgewählten Zoombereich liegen, ausgeblendet werden. Erfolgt im Scatterplot ein solches Zoom-Event, ist ebenfalls eine Reaktion in der tabellenbasierten Visualisierung erforderlich (Kapitel 5.3.4.2). Das DataGlobalModel enthält somit auch solche Methoden, die iterativ in allen Visualisierungen Reaktionen auf Änderungen des sichtbaren Bereichs im Scatterplot auslösen (Kapitel ...)

5.2.3. Probleme bei der Angliederung an das Datenmodell

Die dekorierten DefaultStyledDocuments haben vor allem bei langen Texten einen sehr hohen Arbeitsspeicherbedarf. Die Java Virtual Machine muss daher bei Programmstart über entsprechende VM Parameter mit mehr Arbeitsspeicher ausgestattet werden. Werden alle 600 XML Testdatensätze eingelesen, werden über 128 Megabyte Arbeitsspeicher für die Java Virtual Machine benötigt. Erfolgt die erhöhte Speicherzuweisung nicht, feuert das Programm einen „java.lang.OutOfMemoryError“.

5.3. GranularityTable

5.3.1. Kernklassen der GranularityTable

Aufgrund des verwendeten Prinzips der abstrakten Kopplung der verschiedenen Views an das Datenmodell, muss auch die Kernklasse der GranularityTable mit dem Namen „GranularityTableView“ das Interface „View“ (Abbildung 36) implementieren. Die Klasse GranularityTableView ist die View des GranularityTable und meldet sich, entsprechend dem Paradigma des Observer Pattern, in dieser Funktion auch am Datenmodell an. Von der GranularityTableView wird die Tabelle über das zugehörige „GranularityTableModel“ sowie das „GranularityTableColumnModel“ aufgebaut. Die Grundlage des Aufbaus der Tabelle sind die Datensätze aus dem Datenmodell. Da die Klasse beim Datenmodell als Empfänger von Updates an den Daten registriert ist, bekommt die GranularityTableView über die „update(..)“ Methode eine Instanz des

DataManager übergeben und kann über diesen auf alle gespeicherten Daten zugreifen. Kommt es zu Änderungen an den Daten, baut sich die Tabelle anschließend über die „updateTable()“ Methode in Struktur und Inhalt neu auf.

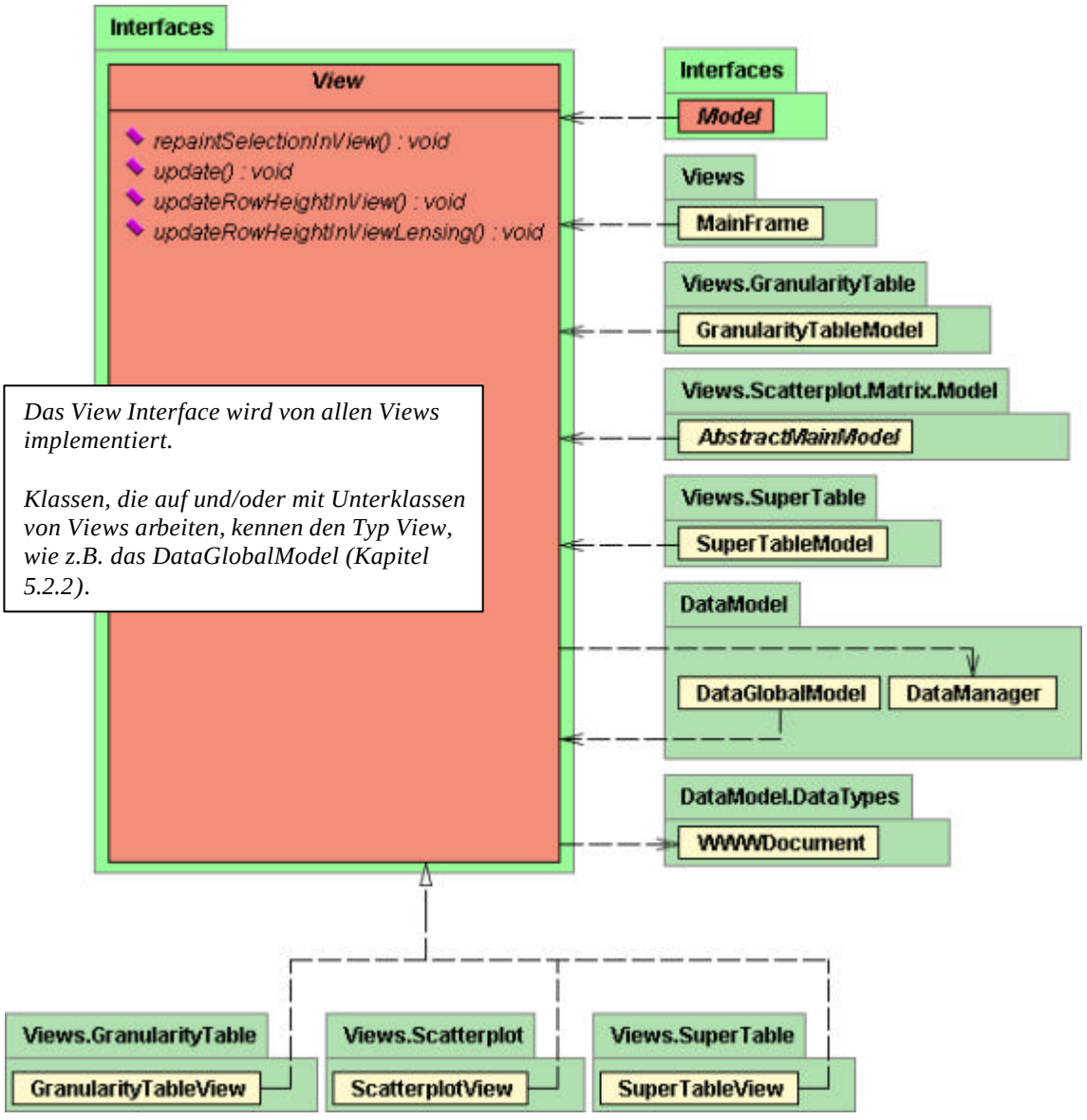

Abbildung 36: Interface View.java, UML Diagramm (JBuilder)

5.3.2. EventListener an der GranularityTable

Die Tabelle benötigt für die Sortierung einen „GranularityTableHeaderListener“ und einen „GranularityTableRolloverListener“, der die TableLens Funktionalität umsetzt.

```
…
public void mouseReleased(MouseEvent e) [Codeposition 4.1.]
{
   clickedIndex [Codeposition 4.2.]
    = ((JTableHeader) e.getComponent()).getColumnModel().getColumnIndexAtX(e.getX());

  columnName
   = ((JTableHeader)
          e.getComponent()).getColumnModel().getColumn(clickedIndex).getIdentifier().toString();

  [Codeposition 4.3.]
  if(columnName.equalsIgnoreCase("Insyder_Export.Visualization") ||
          columnName.equalsIgnoreCase("Insyder_Export.Selection"))
  {
     //-------------------------order the table-----------------------------------------------------------------------
     ((GranularityTableView) ((JTableHeader)
          e.getComponent()).getTable()).setSortOrder(columnName);
  }
}

…
```

Quellcode 4: GranularityTableHeaderListener Codefragment

Quellcode 4 zeigt die Hauptmethode des GranularityTableHeaderListener. Bei einem Mausklick auf dem Tabellenkopf wird die Methode „mouseReleased(...)“ [Codeposition 4.1.] aufgerufen, sobald die Maustaste losgelassen wird. Aufgrund der Mausposition wird festgestellt, welche Tabellenspalte angeklickt worden ist [Codeposition 4.2.]. Anschließend wird eine Sortierfunktion (Kapitel 5.3.3.) aktiviert, jedoch nur dann, wenn die Spalte „Visualization“ oder „Selection“ angeklickt worden ist. Die anderen Spalten, die den Text des Dokumentes und den Granularitätsschieber enthalten, bieten keine Eigenschaften zur Sortierung an, weshalb die Sortierung durch anklicken dieser Spalten nicht aktiviert werden kann [Codeposition 4.3.].

Quellcode 5 zeigt die Methoden des GranularityTableRolloverListeners, die eine Art von TableLens Technik als beweglichen Filter auf der GranularityTable ermöglichen und so die Idee aus Kapitel 4.4.2. umsetzen. Die einfachere der beiden Methoden ist die „mouseExited(...)“ Methode [Codeposition 5.1.]. Diese stellt sicher, dass der TableLens Rollover-Effekt aus der GranularityTable verschwindet, sobald der Benutzer den Mauszeiger aus der Tabelle heraus bewegt. Die zuletzt vergrößerte Tabellenzeile wird in einem solchen Fall wieder auf ihre Standardzeilenhöhe zurückgesetzt [Codeposition 5.3.], wenn sie nicht schon zuvor durch Verwendung des Granularitätskonzeptes in einen anderen Level umgestellt worden ist und ihr dadurch eine andere als die Standardhöhe zugewiesen wurde [Codeposition 5.2.] .

```
…
public void mouseExited(MouseEvent e) [Codeposition 5.1.]
{
   GranularityTableSlider gs = model.getSliderAtRow(recentRow);
   if(gs.getValue()==1 && !(table.getBounds().contains(e.getPoint()))) [Codeposition 5.2.]
   {
         table.setRowHeight(recentRow,DEFAULT_ROW_HEIGHT); [Codeposition 5.3.]
   }
}

public void mouseMoved(MouseEvent e) [Codeposition 5.4.]
{
   recentRow = table.rowAtPoint(e.getPoint());
   GranularityTableSlider gs = model.getSliderAtRow(recentRow);
   if(gs.getValue()==1) [Codeposition 5.5.]
   {
         if(table.getRowHeight(recentRow) == DEFAULT_ROW_HEIGHT) [Codeposition 5.6.]
         {
               table.setRowHeight(recentRow,ROLLOVER_ROW_HEIGHT);
         }
         if(lastRow != -1 && lastRow != recentRow) [Codeposition 5.7.]
         {
               if(table.getRowHeight(lastRow) == ROLLOVER_ROW_HEIGHT)
               {
                     table.setRowHeight(lastRow,DEFAULT_ROW_HEIGHT);
               }
         }
         lastRow = recentRow;
  }
  else
  {
         if(lastRow != -1) [Codeposition 5.8.]
         {
               if(table.getRowHeight(lastRow) == ROLLOVER_ROW_HEIGHT)
               {
                     table.setRowHeight(lastRow,DEFAULT_ROW_HEIGHT);
               }
          }
         lastRow = -1;
   }
}
…
```

Quellcode 5: GranularityTableRolloverListener Codefragment

Die Funktionalität der ersten Methode erklärt bereits die Umsetzung des Konzeptes aus Kapitel 4.4.2. Vergrößert wird immer nur eine Zeile und zwar genau dann, wenn der Anwender mit dem Mauszeiger über die Zeile fährt [Codeposition 5.4.]. Wird eine neue Tabellenzeile überfahren, wird jeweils gleichzeitig mit deren Vergrößerung [Codeposition 5.6.] die zuvor überfahrene Zeile wieder verkleinert [Codeposition 5.7.]. Auch hier wird unterschieden, ob nicht durch Manipulation des Granularitätssliders eine Größenveränderung der Tabellenzeile stattgefunden hat. Dazu wird der Level des Granularitätsschiebers der jeweiligen Zeile abgefragt [Codeposition 5.5.]. Ist der Wert

des Granularitätsschiebers nicht gleich eins, so wird nur die zuletzt vergrößerte Tabellenzeile wieder auf die Defaultgröße zurückgesetzt [Codeposition 5.8.]. Der Funktionskonflikt zwischen TableLens Technik und Granularitätskonzept, wie in Kapitel 4.4.2. festgestellt, wird in dieser Methode gelöst. Konform zu den Überlegungen vor Beginn der Implementation, stehen Veränderungen, die durch den Granularitätsslider hervorgerufen worden sind, über der Funktion der TableLens. Durch den Granularitätsschieber manipulierte Tabellenzeilen werden somit nicht mehr vom TableLens Mechanismus berücksichtigt, außer sie werden zu einem späteren Zeitpunkt auf selbigem Wege wieder in die Standardzellenhöhe zurückversetzt. Eine derartig zurückversetzte Tabellenzeile wird vom Rollover-Listener wieder erkannt und in den TableLens Mechanismus re-integriert. Abbildung 37 zeigt den Roller-Effekt der GranularityTable. Zwei Zeilen der Tabelle sind vergrößert, wobei die obere Tabellenzeile durch den TableLens Mechanismus und die untere Zeile durch den Granularitätsschieber vergrößert worden ist.

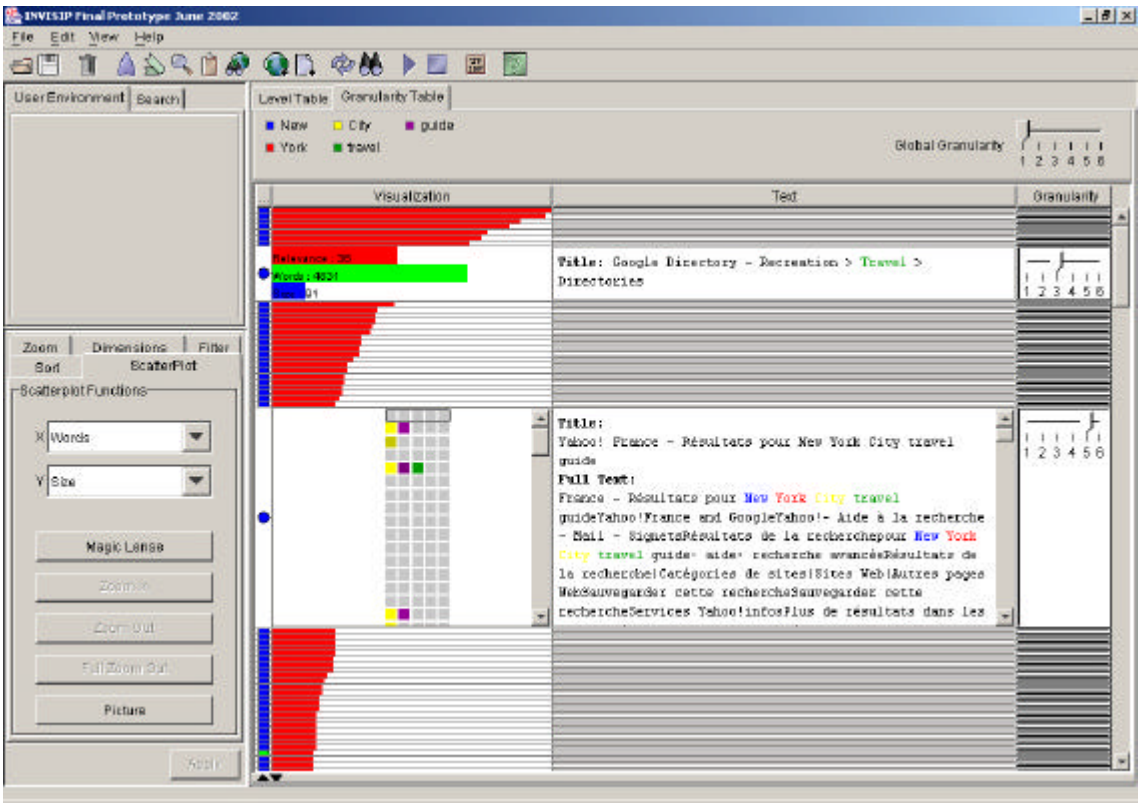

Abbildung 37: TableLens / Rollover Mechanismus der GranularityTable

5.3.3. Sortierfunktion der GranularityTable

Die Sortierfunktion des GranularityTable basiert auf einer in Java implementierten Version des QuickSort Algorithmus. Eine der Kernprogrammmethoden des QuickSort ist die „exchange(...)“ bzw. „swap(...)“ Methode, die Elemente des Datenarrays untereinander in der Position austauscht. Quellcode 6 zeigt die „swap(..)“ Methode in deren Ausprägung für die Sortierung der Daten des GranularityTable.

```
...
private void swap(DataElementContainer array, int i, int j)
{
   WWWDocument wwwTemp = (WWWDocument) array.getElementAt(i); [Codeposition 6.1.]
   WWWDocument wwwNew = (WWWDocument) array.getElementAt(j);

   if(GranularityTableModel.getModel() != null)
   {
    GranularityTableModel model = GranularityTableModel.getModel();
    Vector sliderList = model.getSliderList();
    Vector textList = model.getTextList();
    Vector barList = model.getBarList();
    //-------------------------------------------------------
    Vector tempV  = (Vector)model.getDataVector().elementAt(i); [Codeposition 6.2.]
    Vector newV   = (Vector)model.getDataVector().elementAt(j);
    model.getDataVector().setElementAt(newV,i);
    model.getDataVector().setElementAt(tempV,j);
    //-------------------------------------------------------
    GranularityTableSlider jSliderTemp = model.getSliderAtRow(i); [Codeposition 6.3.]
    GranularityTableSlider jSliderNew = model.getSliderAtRow(j);
    sliderList.setElementAt(jSliderNew,i);
    jSliderNew.setRow(i);
    sliderList.setElementAt(jSliderTemp,j);
    jSliderTemp.setRow(j);
    //-------------------------------------------------------
    GranularityTableTextView jTextTemp = model.getTextViewAtRow(i); [Codeposition 6.4.]
    GranularityTableTextView jTextNew = model.getTextViewAtRow(j);
    textList.setElementAt(jTextNew,i);
    textList.setElementAt(jTextTemp,j);
    //-------------------------------------------------------
    GranularityTableBar jBarTemp = model.getBarAtRow(i); [Codeposition 6.5.]
    GranularityTableBar jBarNew = model.getBarAtRow(j);
    barList.setElementAt(jBarNew,i);
    barList.setElementAt(jBarTemp,j);
    //-------------------------------------------------------
    wwwNew.setRow(i);
    wwwTemp.setRow(j);
   }
  //---------------------------------------------------------
  array.setElementAt(wwwNew, i);
   array.setElementAt(wwwTemp, j);
}

...
```

Quellcode 6: Ausschnitt aus der QuickSort basierten Sortierfunktion des GranularityTable

Nicht nur die Datensätze im Datenmodell [Codeposition 6.1.] werden nach einem Sortierbefehl umsortiert, sondern auch die in die Tabelle eingebauten visuellen Komponenten [Codepositionen 6.3. - 6.5.] müssen sortiert werden. So wird sichergestellt, dass auch nach dem Sortiervorgang Visualisierung, Textansicht und Granularitätsschieber zusammenpassen. Zusätzlich wird direkt in der Sortierfunktion auch der Datenvektor der Tabelle mitsortiert [Codeposition 6.2.]. Alternativ zu diesem Vorgang könnte nach abgeschlossener Sortierung der Datenvektor aus dem sortierten Datenmodell neu aufgebaut werden. Beide Vorgehensweisen entsprechen sich in ihrer Funktion.

5.3.4. Kommunikation der GranularityTable mit anderen Views

Nach dem Model-View-Controller Konzept läuft die Kommunikation der GranularityTableView mit anderen registrierten Views über das DataGlobalModel des Datenmodells ab (Abbildung 38).

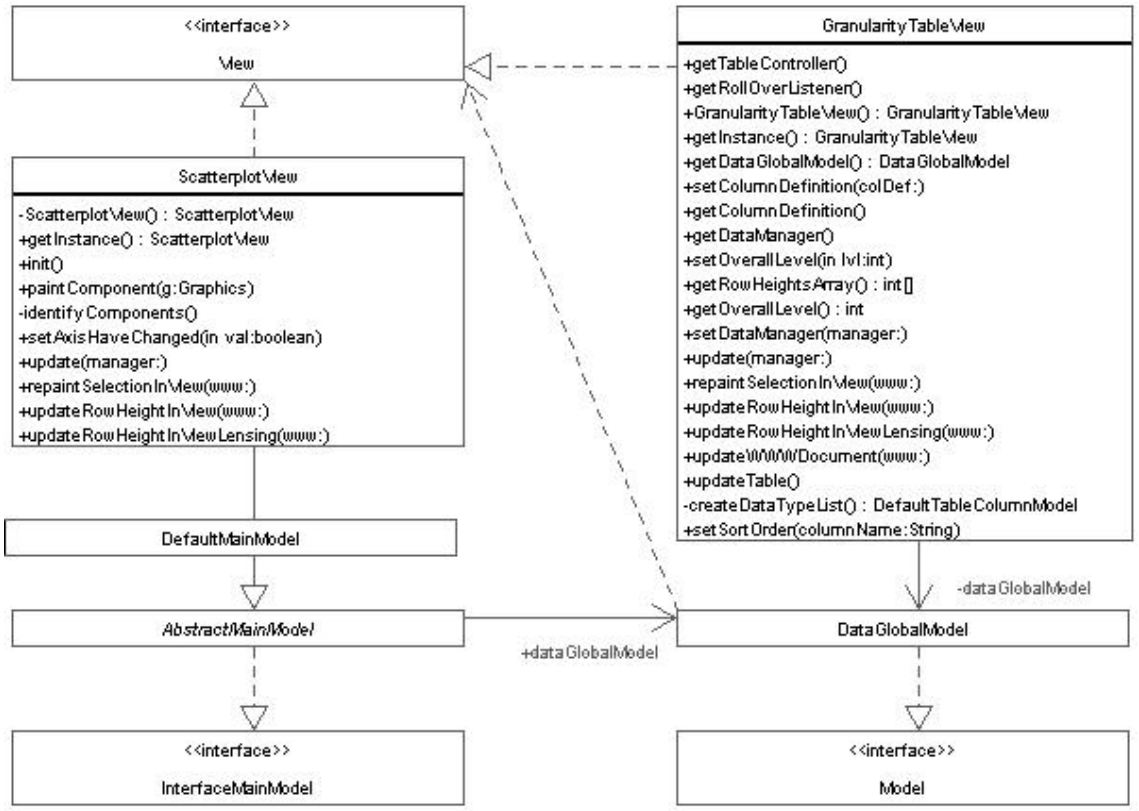

Abbildung 38: Kommunikation zwischen Views und DataGlobalModel nach dem MVC-Konzept

Die GranularityTableView muss, wie alle Views, die durch ein Interface zur Kommunikation vorgegebenen Methoden implementieren. Diese sind sowohl für den Empfang, als auch für den Versand von Nachrichten über Updates an andere Views verantwortlich. Die registrierten Views kommunizieren ausschließlich über das DataGlobalModel miteinander. Das angesprochene DataGlobalModel wiederum spricht die Views über die von allen implementieren Interface Methoden an und sendet dabei nach den Konzepten des MVC-Pull- und Push-Models Daten an diese. Die GranularityTable reagiert auf Selektions- und Zoomoperationen und die Verwendung

der MagicLens[23] im Scatterplot [GF 2002].

5.3.4.1. Dokument- bzw. Punktselektionen und deren Kommunikation unter den Views

Wird in einer am Datenmodell registrierten View eine benutzerseitige Selektion eines Dokumentes ausgeführt, so muss zur Bewahrung der Konsistenz der Datendarstellung die Selektionsoperation auch in allen anderen Views dargestellt werden. Wählt der Benutzer beispielsweise im Scatterplot [GF 2002] einen Dokument-Punkt aus, wird diese Selektion auch in der GranularityTable sichtbar. Abbildung 39 zeigt die Sichtbarkeit der Selektion eines Dokumentes in beiden Visualisierungen. Das selektierte Dokument erscheint im Scatterplot als grüner Punkt. Im GranularityTable erscheint ein Pendant dieses grünen Punktes in der dem Dokument zugehörigen Tabellenzeile in der Spalte „Selection“.

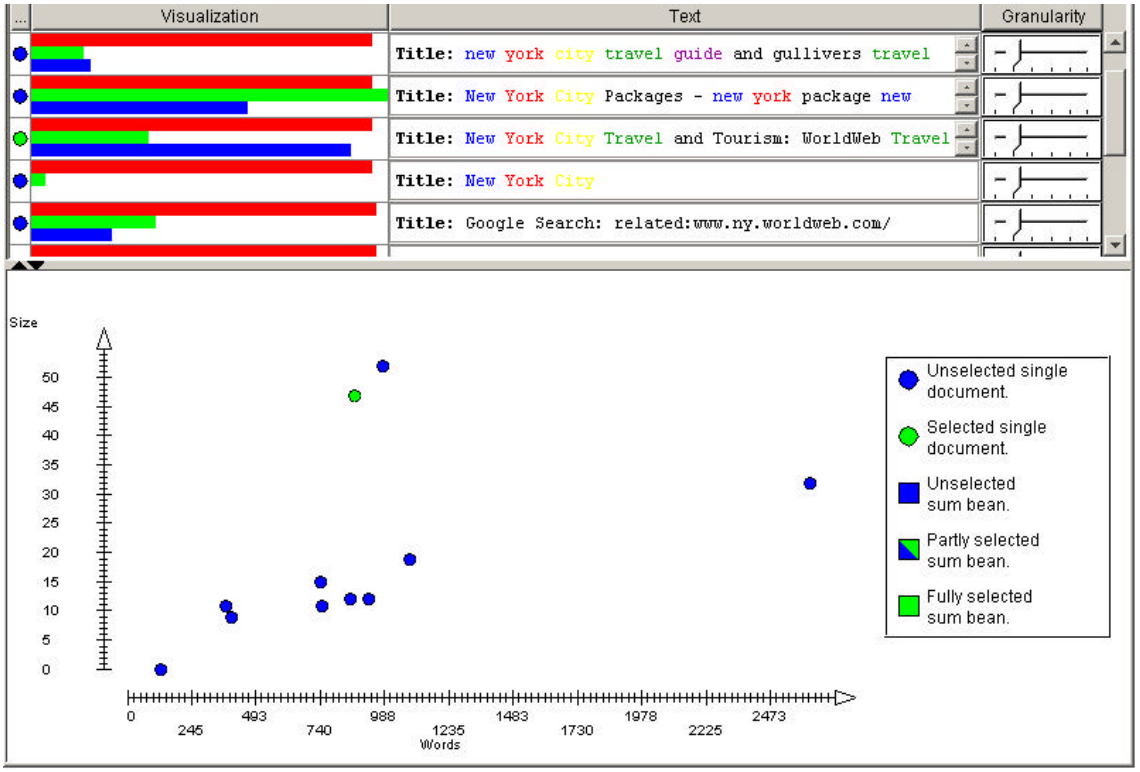

Abbildung 39: Sichtbarkeit der Selektion eines Dokumentes (Punktes) in den Visualisierungen Scatterplot und GranularityTable

[23] Begriff geprägt durch: Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, Tony D. DeRose, Xerox PARC, Palo Alto, CA 94304, University of Toronto, University of Washington

Der Mechanismus hinter dieser Interaktion zwischen den Views, ist der Versand einer Nachricht über die Selektionsveränderung seitens derjenigen View, in der die Selektion vorgenommen wird. Quellcode 7 zeigt die Methodenaufrufe, die als Reaktion auf das Anklicken eines Dokument-Punktes im Scatterplot stattfinden. Zunächst wird eine andere Methode (Quellcode 8) angewiesen den Selektionsstatus des Dokumentes zu ändern [Codeposition 7.1.], indem im Quellcode der Methode „setSelected(...)“ auf das Dokument im Datenmodell zugegriffen wird [Codeposition 8.1.]. Anschließend findet ein Repaint des Scatterplot statt [Codeposition 7.2.], um die Selektionsänderung anzuzeigen. Unmittelbar damit geht der Befehl zum Repaint des Selektionsstatus in allen anderen Views [Codeposition 7.3.] einher, der über das Model des Scatterplot an das DataGlobalModel weitergereicht wird.

```
…
public void mouseClicked(MouseEvent e)
{
   if(e.getModifiers() == e.BUTTON1_MASK)
   {
    if(isSelected()) setSelected(false); [Codeposition 7.1.]
    else setSelected(true);
    mainModel.fireRepaintToMatrix();[Codeposition 7.2.]
    mainModel.fireRepaintSelectionToOtherViews(wwwDoc, true, true); [Codeposition 7.3.]
   }
   …
}
…
```

Quellcode 7: Reaktion auf das Anklicken eines Punktes im Scatterplot [GF 2002]

```
…
public void setSelected(boolean sel)
{
        …
        [Codeposition 8.1.]
        ((DecoratedMetaData)wwwDoc.getReferenceTo("Insyder_Export.Selection")).getCore().
        setData(new Boolean(sel));
}
…
```

Quellcode 8: Änderung des Selektionsstatus eines Dokumentes

Der Befehl enthält alle zum Update notwendigen Informationen (MVC-Push-Model) und wird in der zuständigen Methode des DataGlobalModel verarbeitet (Quellcode 9). Iterativ [Codeposition 9.1.] wird die Interface Methode aller registrierten Views, bis auf die derjenigen View, die Urheber des Kommandos gewesen ist, aufgerufen und die Updateinformation an diese weitergeleitet [Codeposition 9.2.].

```
…
public void repaintSelectionInViews(View cmdSrc, WWWDocument www, boolean j,   boolean s)
{
      for (int i = 0; i < attachedObservers.size(); i++) [Codeposition 9.1.]
         if(!attachedObservers.elementAt(i).equals(cmdSrc)) [Codeposition 9.2.]
           ((View) attachedObservers.elementAt(i)).repaintSelectionInView(www, j, s);
}
…
```

Quellcode 9: Verarbeitung eines Selektionsupdates, Benachrichtigung aller Views durch DataGlobalModel.

Die GranularityTableView Klasse ist der Empfänger des Updates und reicht dieses an das Model der GranularityTable weiter. Diese reagiert zum einen mit einem Aufruf der in der Mutterklasse des Models, dem DefaultTableModel, enthaltenen Methode zum Repaint der Tabellenzelle, die den Selektionspunkt enthält [Codeposition 10.1.]. Der Repaint hat einen Aufruf des TableCellRenderer der Tabellenzelle zur Folge. Dieser greift auf die Daten des Dokumentes zu und zeichnet den Selektionspunkt entsprechend dessen geänderten Selektionsstatus neu. Zum anderen wird die Tabellenzeile des Dokumentes in den sichtbaren Bereich der Tabelle verschoben, indem das ScrollPane der GranularityTable automatisch entsprechend der Position der Zeile bewegt wird (Quellcode 11).

```
…
public void updateWWWDocument(WWWDocument www, boolean jump, boolean select)
{
          if(select)fireTableCellUpdated(www.getRow(),table.getColumnModel().
                    getColumnIndex("Insyder_Export.Selection")); [Codeposition 10.1.]
           if(jump)fireSetScrollFocus(www); [Codeposition 10.2.]
}
```

Quellcode 10: Reaktion der GranularityTable auf Selektionsänderung eines Dokumentes

Jedes Dokument aus dem Datenmodell speichert seine aktuelle Reihe innerhalb der GranularityTable, wodurch der Zugriff auf Positionsdaten einzelner Datensätze und deren visueller Repräsentation abgefragt werden kann.

```
...
public void fireSetScrollFocus(WWWDocument www)
 {
        MyJScrollPane.getInstance().getVerticalScrollBar().
                setValue((int)(table.getCellRect(www.getRow(),table.getColumnModel().
                        getColumnIndex("Insyder_Export.Visualization"),true)).getY());
 }
...
```

Quellcode 11: Fokussierung des Punktes, dessen Selektion verändert worden ist, in der Tabelle

5.3.4.2. Das Verhalten der GranularityTable bei Zoom- und Filteroperationen

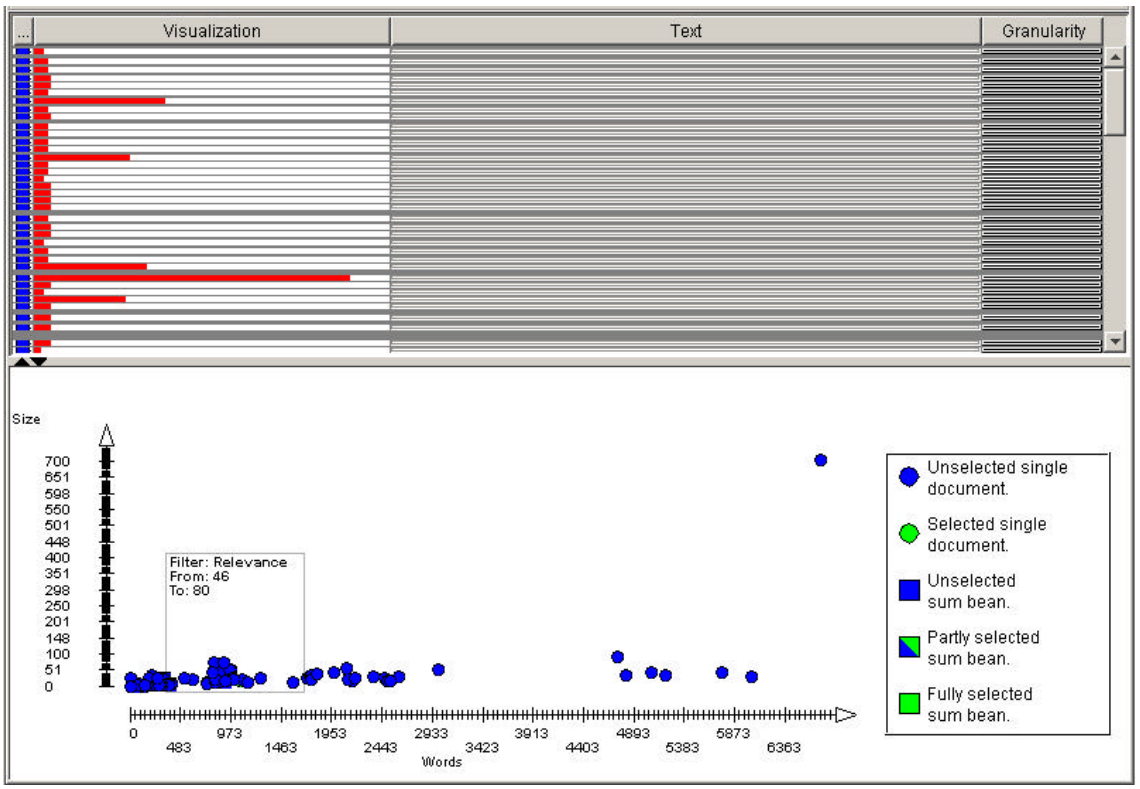

Abbildung 40: Operation mit MagicLens im Scatterplot und Reaktion in der GranularityTable

Im Scatterplot von [GF 2002] können View Transformationen[24] in Form von Zoomoperationen oder durch die Verwendung einer MagicLens ausgeführt werden. Die GranularityTable reagiert auch auf solche Benutzeraktionen mit Veränderungen. Abbildung 40 zeigt die Verwendung einer MagicLens im Scatterplot. Die Filterung der Punkte im Scatterplot führt in der GranularityTable zu einer maximalen Verkleinerung der Tabellenzeilen[25] der betroffenen, herausgefilterten Dokumente [Codeposition 12.4.]. In Abbildung 40 repräsentieren die grauen Balken diese modifizierten Zeilen der GranularityTable. Die Größenveränderung der Zeilen wird solange aufrecht erhalten, bis die Linse vom Dokument-Punkt wegbewegt wird.

```
public void updateRowHeightInViewLensing(WWWDocument www)
{
  if(www.isVisibleIn Lens())[Codeposition 12.1.]
  {
    table.setRowHeight(www.getRow(),levelRowHeights[0]); [Codeposition 12.2.]
    getSliderAtRow(www.getRow()).setValue(1); [Codeposition 12.3.]
  }
  else
  {
    table.setRowHeight(www.getRow(),1); [Codeposition 12.4.]
  }
}
```

Quellcode 12: Reaktion der GranularityTable auf Filteroperationen mit der MagicLens

Dokumente, die innerhalb der Linse sichtbar bleiben [Codeposition 12.1.], weil Sie den Filterkriterien entsprechen, werden auf eine feste Zeilenhöhe eingestellt [Codeposition 12.2.] und der Granularitätslevel der Tabellenzeile des Dokumentes auf die erste Detailstufe zurückgesetzt [Codeposition 12.3.].
Die Reaktion der GranularityTable auf eine Zoomanweisung im Scatterplot funktioniert analog. Die Tabellenzeilen der Dokumente, die außerhalb des gezoomten, sichtbaren Bereichs des Scatterplot liegen, bleiben verkleinert, bis die Dimensionen des Scatterplot den Punkt in ihrem Wertbereich wieder einschließen und dieser innerhalb der Grenzen wieder angezeigt wird.

[24] siehe dazu: Referenzmodell der Visualisierung, Kapitel 4.4.3
[25] siehe dazu auch in Kapitel 5.3.8

5.3.5. Kreation eigener TableCellRenderer

Wie während des Prototyping zur Umsetzung der Granularitätsidee festgestellt wurde, werden zur Integration von Visualisierungen und Granularitätsschieber eigene TableCellRenderer und TableCellEditoren benötigt (Kapitel 4.1, Quellcode 1).
Die Klassen der GranularityTable, die als TableCellRenderer fungieren, beinhalten alle Methoden zur Rückgabe des TableCellRenderer [Codeposition 1.1., Codeposition 13.1.] und des TableCellEditor [Codeposition 1.2.] für die jeweilige Tabellenspalte. Diese Klassen enthalten jedoch nicht selbst die Komponente, die zurückgegeben werden soll, sondern sie geben die Instanz einer Komponente zurück, die im GranularityTableModel gespeichert ist [Codeposition 13.2.]. Bei Aufruf durch die Renderer- oder Editor-Methode werden diese Komponenten mit den Daten, die visualisiert werden sollen, gespeist [Codeposition 13.3.].

```
public Component getTableCellRendererComponent(JTable granularityTable, Object obj, boolean
isSelected, boolean hasFocus, int row, int column) [Codeposition 13.1.]
{
  bar = ((GranularityTableModel)((GranularityTableView)granularityTable).getModel()).getBarAtRow(
        row); [Codeposition 13.2.]
  bar.setWWWDocument((WWWDocument)obj); [Codeposition 13.3.]
  if(multiple)
  {
   bar.setKeys(keys); [Codeposition 13.4.]
   bar.setMaxValues(maxValues); [Codeposition 13.5.]
   bar.setMultiple(true);
  }
  else
  {
        …
  }
  bar.init();
  return (Component)bar;
}
```

Quellcode 13: Rückgabe einer BarChart Visualisierung durch einen TableCellRenderer

Quellcode 13 zeigt diese Methodik innerhalb des TableCellRenderer für BarCharts. Zusätzlich zur Übergabe der Daten in Form des Dokumentes an die BarChart

Komponente [Codeposition 13.3.], werden für die Darstellung auch das Schlüsselwort bzw. die Schlüsselworte [Codeposition 13.4.] an die Visualisierungskomponente übergeben, so dass diese BarCharts genau die Werte dieser Schlüssel repräsentieren. Damit die BarCharts in Relation zum ermittelten Maximalwert über alle Schlüsselwerte gezeichnet werden, wird der Maximalwert an jede Visualisierungskomponente gesendet [Codeposition 13.5.]. Abbildung 41 modelliert die beschriebene Verbindung zwischen Renderer-Klasse und Visualisierungskomponente.

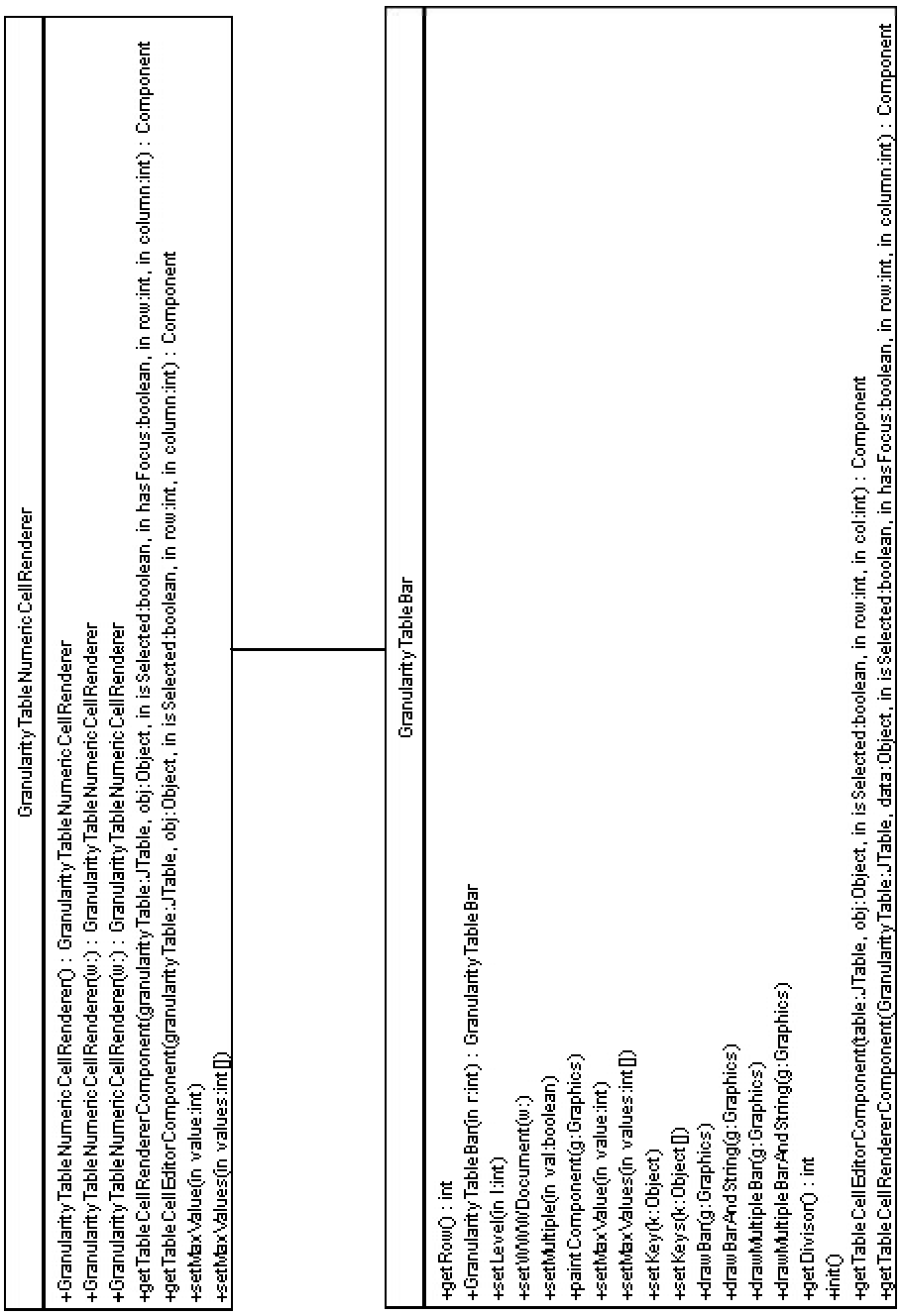

Abbildung 41: GranularityTableNumericCellRenderer mit GranularityTableBar, UML Diagramm

5.3.6. Implementation der Visualisierungen

5.3.6.1. BarCharts

Die im Erklärungsbeispiel durch den Renderer (Kapitel 5.3.5) gespeiste BarChart Visualisierungskomponente extrahiert aus den Dokumentdaten die Werte der Schlüssel, die graphisch dargestellt werden sollen. Während in Granularitätsstufe Eins nur ein Wertebalken gezeichnet wird, enthalten die BarChart Visualisierungen in den Stufen Zwei bis Vier mehrere Balken. In den Detailstufen Drei und Vier enthalten die Grafiken zusätzlich auch Beschriftungen, die den Benutzer über den angezeigten Datentyp, sowie dessen numerischen Wert informieren (Abbildung 42).

Abbildung 42: BarCharts im GranularityTable, Level 2,3 und 4.

Quellcode 14 zeigt die Methode zur Zeichnung der einzelnen BarChart in der ersten Granularitätsstufe. Diejenigen Werte, die im Datentyp Double vorliegen, werden in Integer Werte konvertiert [Codeposition 14.1.] und anschließend als Balken relativ zum existierenden Maximalwert [Codeposition 14.2.] für den zu visualisierenden Schlüssel gezeichnet [Codeposition 14.3.].

```
public void drawBar(Graphics g)
{
   g.setColor(Color.red);
   int rectWidth = 0;
   int value = 0;
   if (dmdata == null && imdata == null) g.drawString("No Data Error",0,0);
   else
   {
    if (dmdata != null) value = ((Double)dmdata.getData()).intValue(); [Codeposition 14.1.]
    else value = ((Integer)imdata.getData()).intValue();
    rectWidth = value * 100 / maxValue * getWidth() / 100; [Codeposition 14.2.]
    g.fillRect(0,0, rectWidth, getHeight()); [Codeposition 14.3.]
   }
}
```

Quellcode 14: Methode zur Zeichnung einer BarChart

5.3.6.2. TileBars

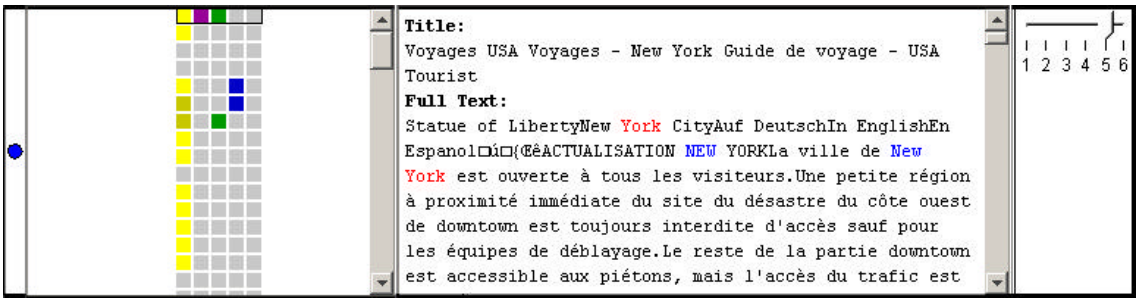

Abbildung 43: TileBar im GranularityTable

Der XML Datensatz (Kapitel 5.1.) enthält fünf Keywords. Die TileBar Visualisierung unterstützt das Textretrieval in Granularitätsstufe Fünf nach dem Konzept aus Kapitel 4.4.5. und ist vertikal ausgereichtet[26].

```
…
private void drawVerticalTileBar(Graphics g)
{
   …
   for(int i = 0; i < segments.length; i++) [Codeposition 15.1.]
   {
    x = (int[])segments[i]; [Codeposition 15.2.]
    for(int j = 0; j < x.length; j++) [Codeposition 15.3.]
    {
          …
           g.fillRect(xPos,yPos, rectWidth, rectHeight);
          …
           yPos = yPos + rectHeight + space;
    }
    …
    xPos = xPos + rectWidth + space;
   …
   }
}
…
```

Quellcode 15: Methode zur Zeichnung der vertikalen TileBar für Granularitätsstufe Fünf

[26] Hintergrundinformation zu dieser Designentscheidung siehe Kapitel 5.3.8.

Da die TileBars unter Umständen länger werden können, als die Tabellenzeile hoch ist, sind die TileBar Visualisierungen in ein ScrollPane eingelagert. Für jedes Keyword ist eine vertikale Reihe von Rechtecken vorgesehen. Jede Längsreihe hat einen eigenen Farbcode, um die unterschiedlichen Verteilungen und Keyword Relevanzen optisch trennen zu können.

Die numerischen Werte, die zum Zeichnen der Tiles vorhanden sein müssen, sind als Segmente in jedem Dokumentdatensatz gespeichert [Codeposition 15.2.]. Iterativ werden die einzelnen Tiles entlang der Keywordsegmente [Codeposition 15.1.] und der Länge deren Wertelisten [Codeposition 15.3.] gezeichnet. Die Größe der Tiles und die Abstände zwischen allen Rechtecken wird zuvor einheitlich festgelegt.

Die TileBar Visualisierungen stellen eine Jumpfunktion (Kapitel 4.4.5.) zur Verfügung. Ein schwarzes Rechteck, welches jeweils eine horizontale Reihe umschließt und damit eine Textposition repräsentiert, kann per Mausklick längs der TileBar verschoben werden. Bei einer solchen Benutzeraktion wird der Text in der Textspalte an die entsprechende Stelle verschoben[27].

[27] zur Jumpfunktion siehe Kapitel 5.3.8.

5.3.6.3. Textansicht

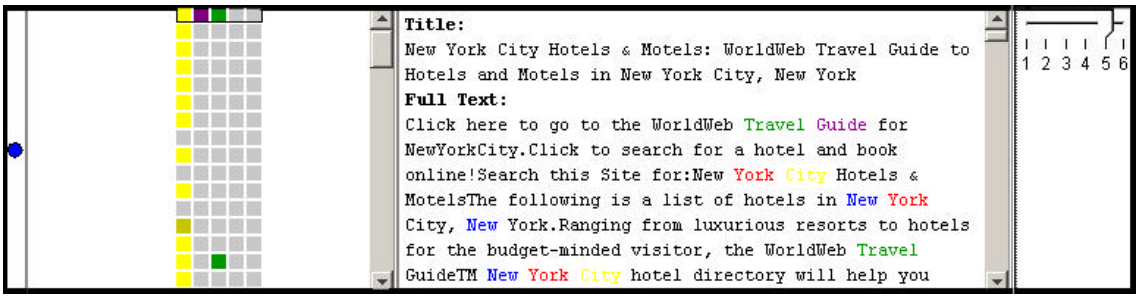

Abbildung 44: Dekorierter Volltext im GranularityTable

Die Textansicht besteht aus einer TextArea, die in ein ScrollPane eingelagert ist. In den verschiedenen Granularitätsstufen werden in der TextArea verschiedene textuelle Elemente angezeigt, wie zum Beispiel der Titel des Dokumentes, der Titel in Kombination mit dem Abstract oder zusammen mit dem Volltext des Dokumentes. In Detailstufe fünf sind die Keywords der Suchanfrage farblich markiert, passend zu der Farbgebung der TileBars. Die Markierung des Textes und der Zusammenbau eines DefaultStyledDocument hat bereits zum Zeitpunkt der Dateneinlese bei Programmstart stattgefunden (Kapitel 5.2.1), so dass bei Aufruf des zugehörigen TableCellRenderer diese aufwendige Prozedur nicht mehr stattfinden muss. Der Renderer übergibt der GranularityTableTextView Klasse (Quellcode 16) das Dokument der entsprechenden Tabellenzeile. Die GranularityTableTextView implementiert das Runnable Interface und kann daher als Thread behandelt werden. Entsprechend den Vorschriften zum Zeichnen von größeren Swing Komponenten wird daher das Zeichnen der TextArea nicht unmittelbar befohlen, sondern die Initialisierung der Zeichenoperationen an den „EventDispatchThread“ der Laufzeitumgebung abdirigiert [Codeposition 16.1.]. Da beim schnellen Scrollen durch die GranularityTable viele Renderer der Tabellenzeilen hintereinander aufgerufen werden und diese jeweils Rechenleistung und Arbeitsspeicher beanspruchen, ermöglicht diese Methodik, im Gegensatz zu jeweils direkt stattfindenden Befehlen zum Zeichnen, ein flüssiges und ruckelfreies navigieren innerhalb der tabellenbasierten Visualisierung. Die Berechnungen der Renderer und das Zeichnen der Grafiken laufen im Hintergrund ab. Dies kann zwar, abhängig von der Belastung des EventDispatchThread und dessen Entscheidung zur Erteilung des

Zeichenbefehls, zu Verzögerungen bei der Anzeige einer Grafik führen, jedoch sind diese im Vergleich zu sekundenlangen Systemstillständen die bessere Alternative.
Wird die Methode run() der zu zeichnenden GranularityTableTextView vom EventDispatchThread aufgerufen, wird über das Dokument auf dessen DefaultStyledDocument zugegriffen und dieses direkt in die TextArea eingefügt und gezeichnet [Codeposition 16.2.].

```
…
public void drawDecoratedText(Object obj)
{
   www = (WWWDocument)obj;
   SwingUtilities.invokeLater(this); [Codeposition 16.1.]
}
…
public void run()
{
   if (level >= 5)//full text
   {
    dmd = (DecoratedMetaData)
          www.getReferenceTo("Insyder_Export.SegmentText_decoratedFullText");
          textArea.setDocument((DefaultStyledDocument) dmd.getData());[Codeposition 16.2.]
   }
   ...
}
...
```

Quellcode 16: Anzeige der Textkomponente mit dekorierten Keywords in Klasse GranularityTableTextView

5.3.6.4. Volltextansicht

```
...
public void paint(Graphics g, JComponent c)
{
   model = GranularityTableModel.getModel();
   super.paint(g,c); [Codeposition 17.1.]
   for(int i=0;i<model.getRowCount();i++)
   {
    paintRow(g,i,2);//only do it for col 2, which is text view
   }
 }

private void paintRow(Graphics g, int row, int col)
{
   int level = model.getSliderAtRow(row).getLevel();
   if(level == 6)
   {
    Rectangle cellRect = table.getCellRect(row,col,true);
    Rectangle mergedRect = table.getCellRect(row,col-1,true);
    Rectangle sumrect = new Rectangle(mergedRect.x,mergedRect.y, mergedRect.width +
          cellRect.width,mergedRect.height); [Codeposition 17.2.]

    GranularityTableCellRenderer renderer =
          (GranularityTableCellRenderer)table.getCellRenderer(row, col);

    if (table.isEditing() && table.getEditingRow()==row && table.getEditingColumn()==col)
    {
     Component component = table.getEditorComponent();
     component.setBounds(sumrect);
     component.validate();
    }
    else
    {
     Component component = table.prepareRenderer(renderer, row, col);
     rendererPane.add(component);
     rendererPane.paintComponent(g, component, table, sumrect.x, sumrect.y, sumrect.width,
          sumrect.height, true); [Codeposition 17.3.]
    }
   }
}
...
```

Quellcode 17: Eigene TableUI der GranularityTable

Die letzte Granularitätsstufe sieht vor, dass die Tabellenspalten „Visualization“ und „Text“ zusammenfallen und der Volltext des Dokumentes über die volle Größe der beiden Spalten angezeigt wird. Die JTable und das Table(Column-)Model sehen ein Merging von Tabellenspalten nicht vor. Für die Umsetzung der letzten Granularitätsstufe ist es deshalb notwendig, eine eigene TableUI, d.h. ein eignes User Interface, für die GranularityTable zu schreiben, so dass das Verbinden von Tabellenspalten mit dieser neuen TableUI emuliert werden kann.

Die „MultSpanCellTableUI“ (Quellcode 17) der GranularityTable zeichnet die Tabellenspalten und -zeilen, die sich in einer kleineren Granularitätsstufe als der Höchsten befinden, mit der normalen Standard TableUI der Java Bibliothek [Codeposition 17.1.]. Iterativ wird für solche Dokumentzeilen der Tabelle, die sich im höchsten Detaillevel befinden, die Darstellung des Panels einer Tabellenzelle in der Größe so verändert, dass es genau die Masse der beiden zu verbindenden Spalten annimmt [Codeposition 17.2.]. Dazu wird eine neue Komponente in der neu berechneten Größe erstellt und mit dem TableCellRenderer der Textspalte gefüllt. In der vergrößerten und verbundenen Zelle wird der Text durch expliziten Aufruf der Paint Anweisung innerhalb der neuen Granzen angezeigt [Codeposition 17.3.]. Abbildung 45 zeigt die neue TableUI im praktischen Einsatz.

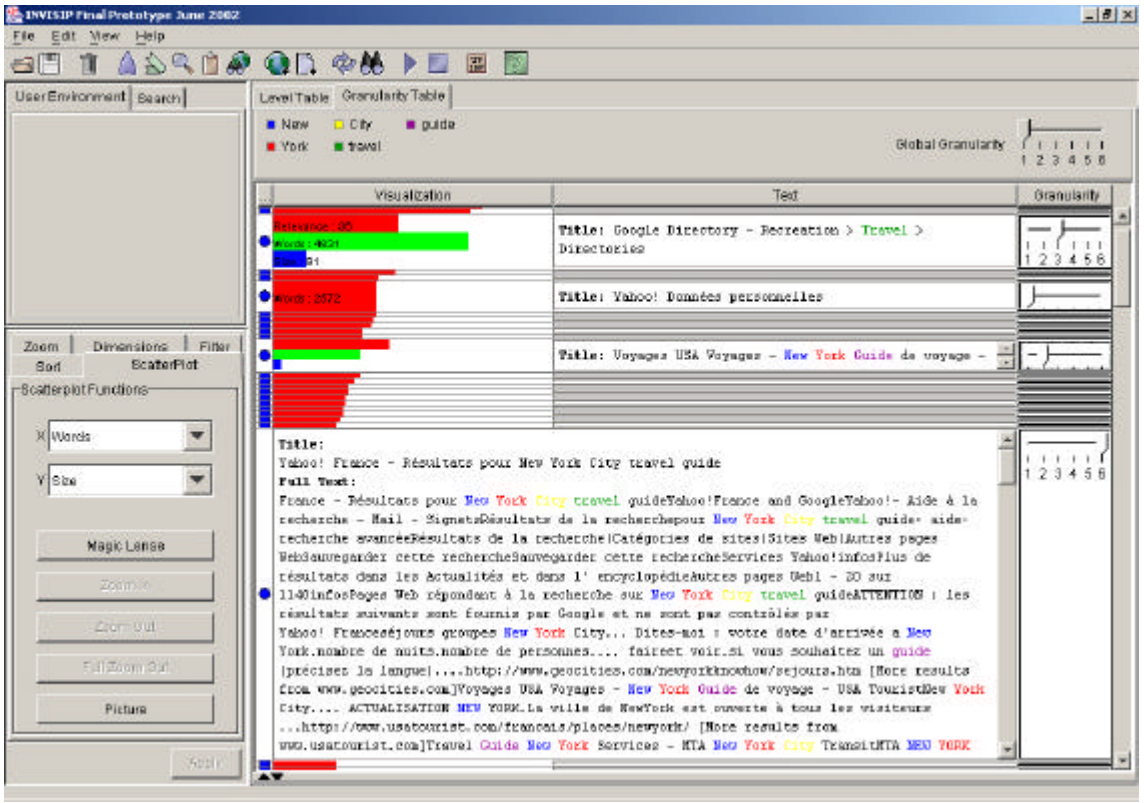

Abbildung 45: GranularityTable mit einem Dokument in letzter Detailstufe

Die Zeilen, die sich in der sechsten Granularitätsstufe befinden, haben nun sowohl die Tabellenspalte „Visualization“, also auch die Spalte „Text“ zur Verfügung.

5.3.7. Der Granularitätsslider

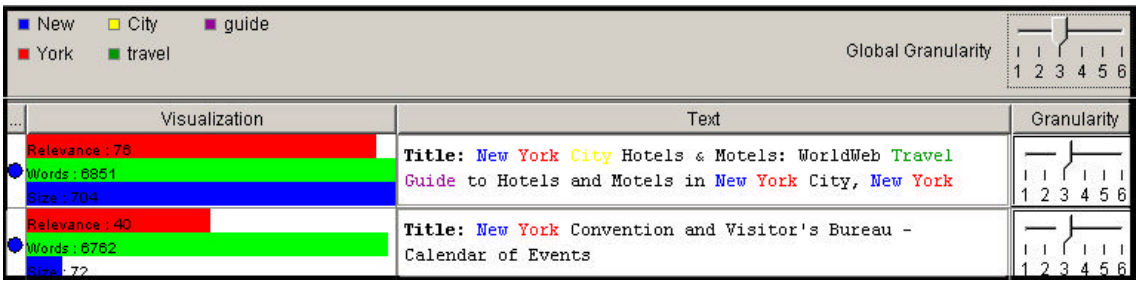

Abbildung 46: Globaler und zeilenweise Granularitätsslider

Der Granularitätsschieber existiert konform zu den Vorgaben des Redesign Vorschlags von M. Eibl [EM 2001] in zwei Varianten. Zum einen als globaler Regler [Codeposition 18.2.], der bei Benutzung die Granularitätsstufe für alle Tabellenzeilen ändert [Codeposition 18.3.], zum anderen als zeilenweiser Regler, dessen Benutzung nur Auswirkungen auf die jeweilige Zeile hat [Codeposition 18.5.]. Der Slider speichert die für die Modifikation notwendige Nummer der Reihe, die verändert werden muss, sowie die neue Granularitätsstufe [Codeposition 18.1.]. Die Reaktion auf die Verwendung des Granularitätssliders ist eine Veränderung der Zellenhöhe einer einzelnen [Codeposition 18.4.] oder (iterativ) mehrerer Tabellenzeilen [Codeposition 18.6.].

```
public void changeRow(GranularityTableSlider slider)
{
   try
   {
     int level = slider.getLevel();[Codeposition 18.1.]
     int row = slider.getRow();
     if(slider.isMainSlider()) [Codeposition 18.2.]
     {
       int size = manager.getContainer().size();
       for(int i=0;i < size;i++) [Codeposition 18.3.]
       {
         ((WWWDocument)manager.getContainer().getElementAt(i)).setLevel(level);
         getSliderAtRow(i).setValue(level); [Codeposition 18.4.]
       }
       table.setOverallLevel(level);//changes row heights for all levels
     }
     else [Codeposition 18.5.]
     {
          …
        ((WWWDocument)manager.getContainer().getElementAt(row)).setLevel(level);
        table.setRowHeight(row,getValue(level)); [Codeposition 18.6.]
     }
   }
}
```

Quellcode 18: Änderungen der Höhe der Tabellenzeilen nach ChangeEvent am Slider

Da das Projekt nach den Vorgaben der Aufgabenstellung in Java Version 1.3.1_03 implementiert wird, steht für den Granularitätsslider noch keine Unterstützung des Mausrades zur Verfügung[28]. Auch die stufenweise Umschaltung auf die jeweiligen Sliderwerte, durch Anklicken des ungefähren Wertebereichs in der Sliderkomponente mit der Maus, wird seitens der Java JSlider Klasse nicht unterstützt, wenn der Gesamtwertebereich des Sliders wie in der GranularityTable sehr klein ist.

```
…
public void mouseClicked(MouseEvent e)
{
   int space = (getWidth()-18)/(getMaximum()-1);
   int pos = getValue();
   if(e.getX()<(((pos-1)*space)+9)) [Codeposition 19.1.]
   {
    setValue(pos-1); [Codeposition 19.2.]
   }
   else
   {
    setValue(pos+1);
   }
}
...
```

Quellcode 19: Methode zur Umschaltung der Granularität durch Mausklick auf die Detailstufe

Daher enthält die Klasse GranularityTableSlider als Unterklasse von JSlider eine eigene Methode, die diese offensichtliche Schwachstelle in der Benutzbarkeit des Reglers ausgleichen soll. Aufgrund einer Berechnung der Größe der Sliderkomponente in der Spalte der GranularityTable und der Einteilung der Komponente in 6 Bereiche für jede Detailstufe [Codeposition 19.1.], werden Mausklicks innerhalb der Sliderkomponente aufgefangen und der Scheiberegler in die Detailstufe umgeschalten [Codeposition 19.2.], die im Einzugsbereich des Mausklicks liegt.

[28] Ein „MouseWheelListener“ steht erst ab Java Version 1.4. zur Verfügung

5.3.8. Implementations- und Konzeptprobleme

Bei der Umsetzung des Granularitätskonzeptes traten Schwierigkeiten bei zwei Granularitätsstufen auf. Konsequenz des ersten Problems ist, dass Detailstufe vier in der implementierten Version keine horizontale TileBar, sondern die gleiche Visualisierung wie Detailstufe drei enthält. Der Unterschied zwischen den Detailstufen besteht nur in einer vergrößerten Zeilenhöhe und dadurch größeren Darstellung der BarCharts. Hintergrund der Entscheidung zum Wegfall der TileBar in diesem Level ist, dass nach übereinstimmender Meinung der Projektmitglieder[29] nur die vertikale TileBar aus Detailstufe fünf einen Zusammenhang zwischen Visualisierung und Volltext erkennen lässt. In Detailstufe vier wird nach dem Eiblschen Konzept [EM 2001] /[MKRE 2002] nur der Abstract angezeigt, zu dem die angezeigten Werte einer TileBar aber nicht passen, weil diese basierend auf aus dem Volltext errechneten Werten gezeichnet worden ist. Alternativ hätte man auch die horizontale TileBar verwenden können, indem man bereits in Level vier Volltext angezeigt hätte. Doch in jedem der beiden Fälle muss das Konzept von M. Eibl zum Teil gebrochen werden. Welche der Alternativen tatsächlich vorzuziehen ist, können zu einem späteren Zeitpunkt Benutzertests herauskristallisieren.

Problem zwei besteht in der Granularitätsstufe Sechs des Redesign Konzeptes. Diese ist in der implementierten Version nicht enthalten, da der Rechenaufwand für die Erstellung von Thumbnails von der Textansicht eines Dokumentes zu groß wäre. Außerdem könnten die Thumbnails von mehreren hundert TextPanels nicht zusätzlich, wie die anderen Dokumentdaten, ebenfalls im Arbeitsspeicher gehalten werden, sondern müssten als Files abgelegt werden. Die dabei entstehenden Datenoutput- und Dateninputströmungen würden bei der Arbeit mit der GranularityTable enorme Wartezeiten bei der Aufbereitung der Daten und Granularitätsstufen zur Konsequenz haben. Deswegen wurde in dieser Version der GranularityTable (noch) auf die Thumbnail View verzichtet, so dass es nur sechs statt sieben Granularitätsstufen gibt.

Weitere Probleme entstehen durch die noch unausgereiften Klassen und Methoden, die die Java Bibliothek für das Anzeigen von Texten und der Navigation in diesen zur Verfügung stellt. So bieten sich bei Verwendung des DefaultStyledDocument in Verbindung mit einer TextArea und einem ScrollPane noch keine bereits vorhandenen

[29] Arbeitsgruppe „Informationssysteme“, Informatik & Informationswissenschaft, Universität Konstanz.

Funktionen an, die die Jumpfunktion der TileBar aus Granularitätsstufe Fünf ermöglichen würden. Diese bedarf nämlich einer Methode, die das ScrollPane direkt an die Stelle des Textsegmentes verschiebt, welches der Benutzer durch Auswahl auf der TileBar ansteuern will. Die Jumpfunktion der GranularityTable ist somit noch nicht voll funktionsfähig, da hier erst noch eine geeignete Lösung gefunden werden muss. Problematisch bei der Jumpfunktion ist auch, dass die TileBars sich aufgrund der Werte von Textsegmenten zeichnen, die sehr zum Nachteil der Jumpfunktion unterschiedliche Stringlänge haben. Dies ist ein Fauxpas, der zuvor bei der Programmierung des INSYDER Agenten begangen wurde und der sich nun in diesem Projekt, welches mit Exportdaten aus INSYDER arbeitet, bemerkbar gemacht hat. Somit ist das Auffinden des Segmentes im Volltext algorithmisch sehr aufwendig, da die Textstelle nicht einfach durch Abzählen gefunden werden kann. Beide Defizite erschweren und verlangsamen eine Jumpfunktion und die Behebung dieses Problems dürfte sehr hohe Kosten verursachen.

Zoom- und Filteroperationen im Scatterplot haben in der Tabelle eine Verkleinerung der Tabellenzeilen zur Folge (Kapitel 5.3.4.2). Java und die JTable unterstützen nur eine Verkleinerung von Tabellezeilen auf eine Höhe von einem Pixel plus der Begrenzung der Zeile, so dass sich bei vielen gefilterten Dokumenten bzw. Tabellenzeilen farbige Balken ergeben. Grundidee beim Verkleinerungsmechanismus war die Tabellenzeile völlig verschwinden zu lassen, was aber durch eine Verkleinerung somit nicht möglich ist. Einziger Lösungsweg ist hier die Entfernung der gefilterten Dokumente aus dem Datenbestand des Datenmodells und anschließend das TableModel anzuweisen, die Tabelle mit dem neuen Datensatz wieder aufzubauen. Damit aber die Dokumente wieder in der Tabelle angezeigt werden, wenn beispielsweise die MagicLens im Scatterplot bewegt und die Filterung für Dokumente dadurch aufgehoben wird, müsste der vorherige Datenbestand separat gespeichert werden. Dadurch ergäbe sich einer Art History von Datenbeständen und somit die Basis einer Undo/Redo Funktionalität für derartige Operationen. Dies würde es auch erleichtern, für die gefilterten Dokumente zu speichern, in welcher Granularitätsstufe sie sich befanden, bevor sie entfernt wurden. Somit könnten diese bei einem Undo wieder in den gleichen Zustand zurückversetzt werden, anstatt auf eine feste voreingestellte Detailstufe und Zeilenhöhe eingestellt zu werden (Kapitel 5.3.4.2). Auf die Integration einer solchen History wurde in der vorliegenden Version der GranularityTable jedoch vorerst verzichtet, da auch hier,

ähnlich wie bei der Thumbnail View, zur Entlastung des Arbeitsspeichers eine Auslagerung der Datenbestände in eine Datei notwendig wäre.

Die Entscheidung zum Verzicht auf Funktionen, die Zugriff auf das Dateisystem des Computers notwendig machen, ist eng mit den Leistungsdefiziten des Datenmodells verbunden, welches zuerst verbessert werden muss, bevor an neue, anspruchsvolle, rechen- und zeitintensive Konzepte innerhalb der GranularityTable gedacht werden kann.[30]

[30] zu den Defiziten des Datenmodells : Kapitel 6.3 Verbesserung der Performanz

6. Ausblick

An dieser Stelle soll auf Erweiterungs- und Verbesserungsmöglichkeiten der erstellten Software (in der Version vom Juni 2002, Abbildung 47) hingewiesen werden.

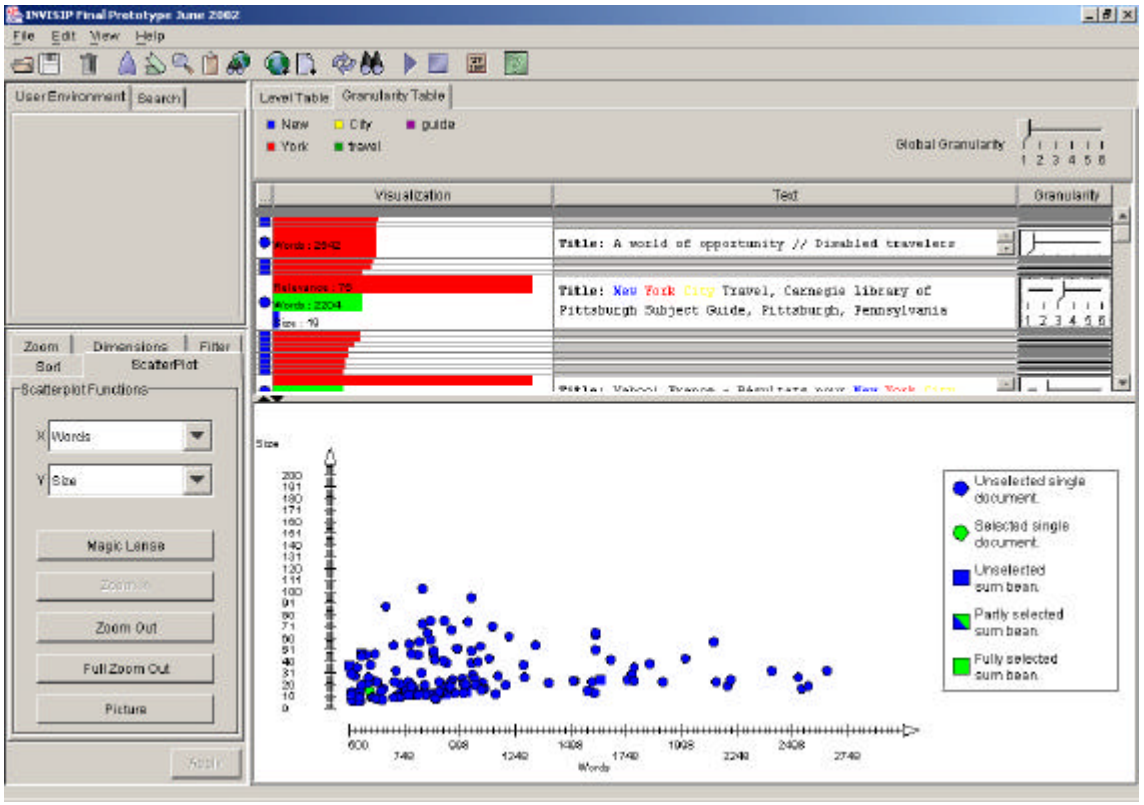

Abbildung 47: INVISIP Finaler Teil-Prototyp Stand Juni 2002

6.1. Erkenntnisse aus der Implementationsarbeit

Schwerpunkt bei der Umsetzung der GranularityTable und der anderen visuellen Komponenten des INVISIP Prototypen ist der Umgang mit den Java Swing Widgets.
Swing Komponenten sind sehr mächtige Konstrukte, d.h. sie beinhalten sehr viel Funktionalität. Die Vielzahl der Innereien einer Swing Komponente schleppt man sofort in ein Java Programm ein, sobald man dieser ein solches Widget hinzufügt, auch wenn man einen großen Teil der Funktionen nicht benötigt. Dies hat zur Konsequenz, dass der Arbeitsspeicher stärker belastet wird, als dies eigentlich der Fall sein müsste. Die GranularityTable mit ihren vielen Swing Komponenten, die sie zur Visualisierung der geo-räumlichen Metadaten verwendet, kann – stark abstrahiert – mit einem PKW

verglichen werden, der vollgepackt oder sogar überladen ist, obwohl nur ein Bruchteil der Gepäckstücke benötigt wird. Unter der Last des hohen Gewichtes wird die GranularityTable sehr langsam.

6.2. Evaluationsergebnisse

Die Evaluation des Prototypen [JC 2002] in der Version vom Juni 2002, wird von drei Experten des Lehrstuhls Prof. Dr. Reiterer, Universität Konstanz, durchgeführt. Der Test basiert auf einer XEROX Checkliste für heuristische Evaluationen[31]. Der Prototyp wurde hinsichtlich Usability untersucht und die festgestellten Konflikte in drei Kategorien eingeteilt. Kleine Usability Konflikte (minor usability problem) können mit niedriger Priorität behoben werden, an größeren Problemen (major usability problem) sollte unbedingt gearbeitet werden und Usability Katastrophen (usability catastrophe) müssen in jedem Fall vor dem ersten Release der Anwendung behoben werden. Tabelle 3 zeigt Anzahl und Verteilung der in der Evaluation identifizierten Usability Konflikte.

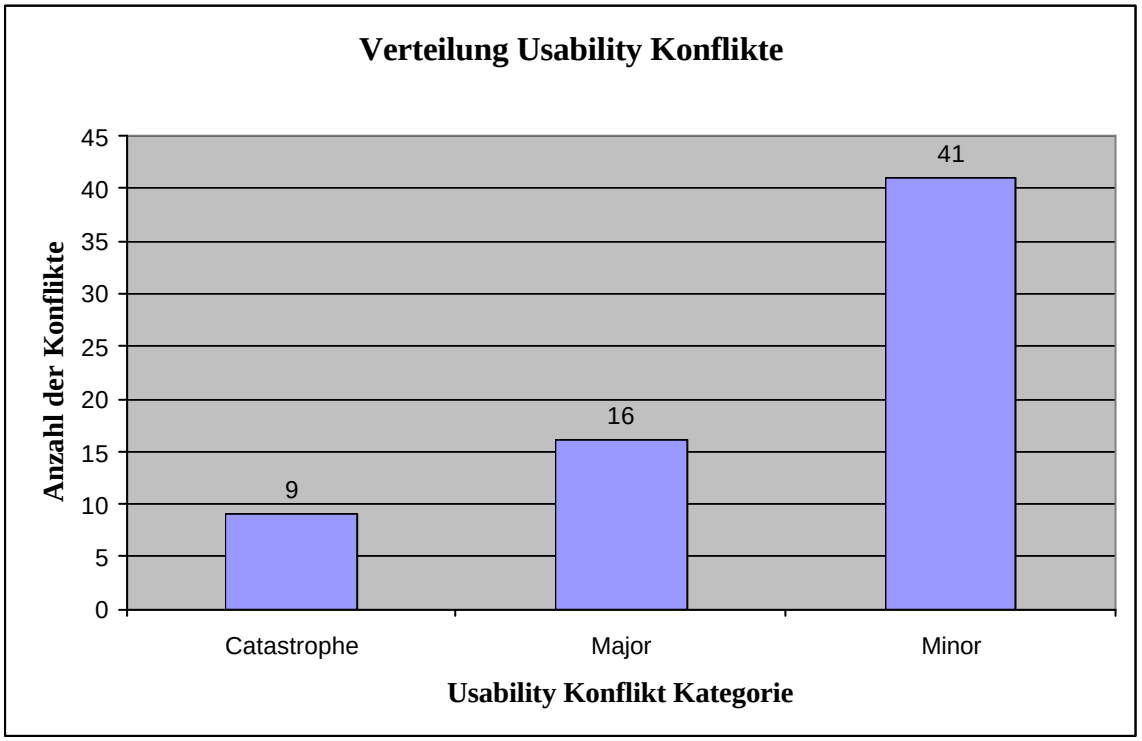

Tabelle 3: Statistik der heuristischen Evaluation des JAVA Prototypen vom Juni 2002

[31] Xerox, Heuristic Evaluation - A System Checklist, http://www.stcsig.org/usability/resources/toolkit/he_cklst.doc, online am 21.08.2002.

Die Experten kritisieren Ungenauigkeiten und ergonomische Schwächen bei der Farbgestaltung der Programmoberfläche des gesamten Prototypen. Diese Kritikpunkte, sowie auch diejenigen, die die Funktionalität und Manipulierbarkeit der Programmoberfläche betreffen, wie zum Beispiel eine noch nicht vorhandene Resize-Funktion für die Tabellenspalten der GranularityTable oder die Abstinenz von Keyboard Shortcuts, können im weiteren Verlauf der Arbeiten am INVISIP Projekt berücksichtigt werden und dienen als wichtige Anregungen und Verbesserungsvorschläge. Usability Konflikte dieser Art sind hauptsächlich gering bis schwerer gewichtet. Sehr viel problematischer und tiefgreifender ist, dass alle Teilnehmer der Evaluation die zu langen Reaktionszeiten der GranularityTable bemängeln. Das System benötigt für Veränderungen der Granularität gerade in höheren Detailstufen sehr viel Zeit. Wie schwerwiegend dieses Usability Problem ist, zeigt Tabelle 4. Zwei Drittel der Usability Katastrophen werden durch Geschwindigkeitsprobleme der GranularityTable und Defizite der Java Swing Komponenten verursacht. Wichtigster Kritikpunkt aus entwicklungstechnischer Sicht und auch das wohl am aufwendigsten zu behebende Problem ist somit das schon während der Implementation festgestellte Performanceproblem der GranularityTable.

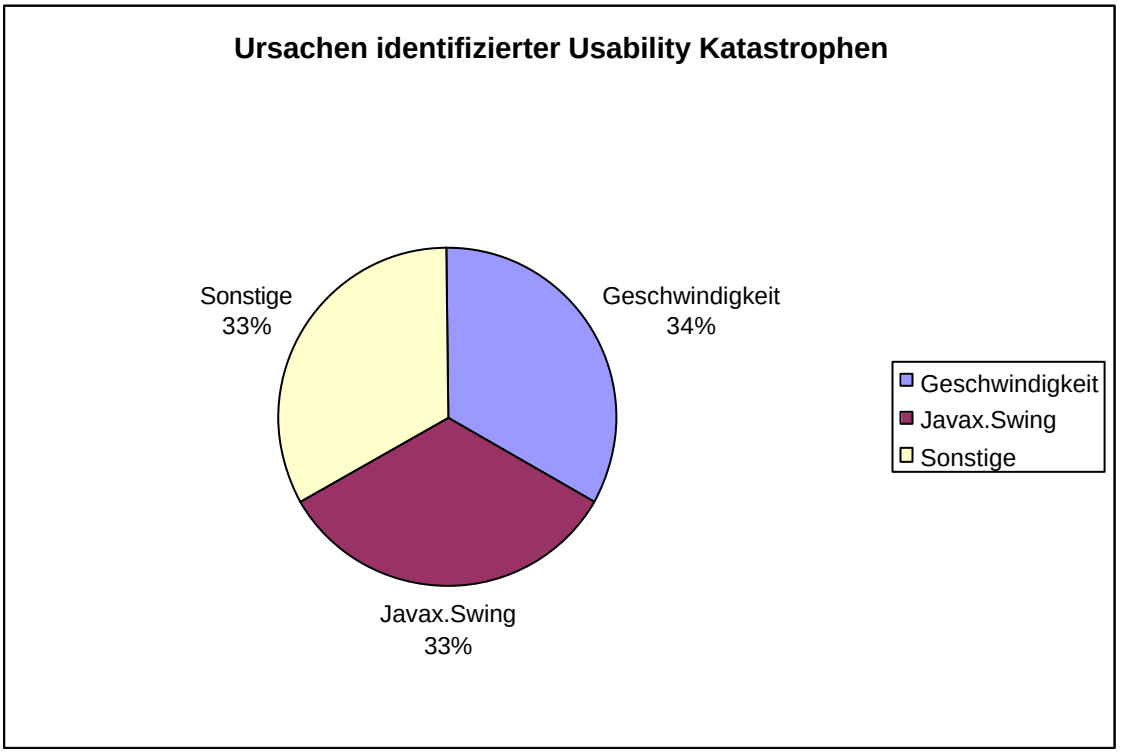

Tabelle 4: Ursachen identifizierter Usability Katastrophen

6.3. Verbesserung der Performanz

Die primäre Strategie zur Verbesserung der Geschwindigkeit der GranularityTable besteht darin, das Datenmodell zu verbessern. Das Datenmodell besitzt an sehr vielen Stellen Funktionalitäten, die von keiner View des Projektes benötigt werden, andererseits fehlen dafür aber elementare Dienste, so dass Views zum Teil Berechnungen durchführen müssen, die eigentlich bereits zu einem früheren Zeitpunkt vom Datenmodell hätten erledigt werden sollen. Gerade die verschiedenen TableCellRenderer, die bei jedem Repaint der Tabelle aufgerufen werden, um den Zelleninhalt zu zeichnen, müssen frei sein von jeglicher algorithmischer Berechnung. Ist dies nicht der Fall, werden die Renderer zum Flaschenhals der gesamten Applikation. Dazu kommt, dass das Datenmodell sehr stark die objektorientierten Konzepte der Programmiersprache JAVA nutzt. Die Klassen des Datenmodells führen sehr viele Referenzen auf andere teilnehmende Komponenten und die Verwaltung dieser Referenzen über die JAVA Referenztabelle kostet sehr viel Zeit, was mit dem Erfordernis eines schnellen Datenzugriffs konfligiert. Das Datenmodell sollte daher so umgebaut werden, so dass es nach der Revision zum einen schlanker und leistungsfähiger ist und zum anderen die Views stärker als bisher entlastet.
Die Geschwindigkeit der GranularityTable könnte auch durch die Verwendung der neuen SWT Technologie von IBM gesteigert werden. SWT steht für „Standard Widget Toolkit“ und basiert grundsätzlich auf JAVA. Jedoch wird durch entsprechende Compiler aus den JAVA Sourcen nativer, plattformspezifischer Quellcode erzeugt. Dadurch können Anwendungen, die in der plattformunabhängigen Programmiersprache JAVA implementiert worden sind, an die Leistungsfähigkeit von nativer Software heranreichen. Grund dafür ist die eigene API von SWT. Diese versucht die Defizite von Java AWT und Swing aufzuholen. Während AWT nur Widgets anbietet, die auf allen Plattformen vorhanden sind und dadurch nur sehr eingeschränkt für die Erstellung von richtigen Applikationen geeignet ist, verfügt Swing über ein sehr großes Repertoire an Widgets und ist sehr flexibel, weil plattformnative Beschränkungen übergangen werden können. Jedoch muss Swing diese Widgets selbst zeichnen, was zum einen dazu führt, dass Anwendungen nicht immer zu 100% dem Look & Feel des Systems entsprechen, zum anderen erhöht dies den Speicherbedarf der Software sehr stark. SWT verfolgt die Strategie, diejenigen Widgets, die auf dem System vorhanden sind auch zu nutzen und nur diejenigen Widgets selbst zu erstellen und zu emulieren, die es nicht sind.

Letztendlich resultiert dies in der Praxis in einer besseren Performanz des Softwareproduktes [VK 2002]. Gerade die GranularityTable, die für jeden Datensatz mehrere „schwergewichtige“ Swing Komponenten, darunter Panels, ScrollPanels, eine TextArea und einen Slider, verwendet, könnte von dieser Technologie sehr stark profitieren und so die auch in der Evaluation festgestellten Defizite bezüglich der Geschwindigkeit absorbieren.

6.4. Zusammenfassung

Die Redesign Idee von M. Eibl stößt bei der Umsetzung an die Grenzen der Leistungsfähigkeit der Programmiersprache JAVA und der Virtual Machine, selbst auf modernen Hochleistungscomputern. Ein nicht unerheblicher Teil künftiger Implementationsarbeit wird daher zunächst dafür aufzubringen sein, die durch die Verwendung von Swing eingekauften Performancedefizite mit Hilfe spezieller von Sun Microsystems veröffentlichten Tutorials zumindest teilweise zu beheben oder Swing durch die SWT Technologie von IBM zu ersetzen. Gleichzeitig muss durch eine Revision des Projektkernes - dem Datenmodell - die Geschwindigkeit von einem anderen Ansatzpunkt aus gesteigert werden. Das Datenmodell hält die Renderer der GranularityTable nicht frei von diversen Rechenoperationen, da es selbst diese nicht anbietet. Dies ist ein sehr entscheidendes Defizit des Datenmodells. Das Datenmodell muss an diese Erfordernisse angeglichen und an vielen entscheidenden Stellen verändert werden, so dass die GranularityTable schneller wird.
Das Granularitätskonzept ist eine interessante und vielversprechende Idee für das Document Retrieval. Doch erst nach der Behebung der Leistungsprobleme der GranularityTable können die in Kapitel 5.3.8 beschrieben Zusatzfunktionen eingebaut und ein direkter Vergleich gegenüber anderen tabellenbasierten Visualisierungen durchgeführt werden. Auf dieser Basis stattfindende Gegenüberstellungen werden dann zeigen, ob das Granularitätskonzept bei den potentiellen Anwendergruppen erfolgreicher ist als die Visualisierungsmethoden von INSYDER und ob mit der Redesign Idee von M. Eibl ein effektives Konzept gefunden worden ist.

7. Quellenverzeichnis

Quellen ID	Metadaten
[AGIS 1998]	Universität Konstanz, Arbeitsgruppe Informationsysteme - „**Insyder Overview**“ - 1998.
[AP 1999]	Althammer, E.; Pree, W. – “**An Architecture for a Strict Model-View Separation in Java**” - Software Engineering and Applications (SEA‘99) Conference , Scottsdale, Arizona, 6-8 October 1999.
[BB 2001]	Bekavac, Dr. B **– „Skript zum Kurs Information Retrieval**“, Universität Konstanz, Inf.wissenschaft, WS 01 / 02. Quelle: www.inf-wiss.uni-konstanz.de/CURR/winter0102/IR/ir_script_ws01.pdf, online am 10.08.2002.
[BH 2000]	Balzert, Helmut – “**Lehrbuch der Softwaretechnik**” - Spektrum Akad. Vlg, 2000.
[BJ 1998]	Bosch, Jan – “**Design Patterns as Language Constructs**” - University of Karlskrona/Ronneby, Department of Computer Science and Business Administration, 1998.
[BRS 1996]	Buder, M; Rehfeld, W.; Seeger, T. – „**Grundlagen der praktischen Information und Dokumentation“** - Saur, K.G., Müchen, November 1996.
[BSFBB 1994]	Bier, Eric A.; Stone, Maureen C; Fishkin, Ken; Buxton, William; Baudel, Thomas - „**A Taxonomy of See-Through Tools**“ - Xerox Parc; University Of Toronto; University Of Paris-Sud; 1994.
[BSPBD 1993]	Bier, Eric A.; Stone, Maureen C; Pier, Ken; Buxton, William; DeRose, Tony D. - „**Toolglass and Magic Lenses: The See-**

Through Interface“ - Xerox Parc; University Of Toronto; University Of Washington; 1993.

[CMS 1999] Card, S.; Mackinlay, J.; Schneiderman, B. – “**Readings in Information Visualization” -** Morgan Kaufmann Verlag, Februar 1999.

[CR 1999] Card, Stuart K.; Rao Ramana; - „**The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Conext Visualization for Tabular Information**” – Xerox Palo Alto Research Center, Palo Alto, CA, USA. In: Card, S.; Mackinlay, J.; Schneiderman, B. – “Readings in Information Visualization**” -** Morgan Kaufmann Verlag, Februar 1999.

[EM 2001] Eibl, M. - „**Vorschlag für INSYDER Redesign**“ - IZ Social Science Information Centre, Dezember 2001. [Unveröffentlicht]. Siehe auch [MKRE 2002].

[FD 2002] Flanagan, David – „**Java In A Nutshell**“ - O'Reilly UK, 4. Auflage, März 2002.

[FG 1986] Furnas; George W. – „**Generalizes Fisheye Views**“ – In „Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems“, Seiten 16-23, ACM, April 1986.

[FGDC 1998] Federal Geographic Data Committee (FGDC) - **Content Standard for Digital Geospatial Metadata (CSDGM)** - FGDC-STD-001-1998

[GF 2002] Gundelsweiler, Fredrik – „**Implementation eines Scatterplots zur Visualisierung von geo-räumlichen Metadaten“** – Bachelorarbeit, Universität Konstanz, , 2002.

[GHJV 1993] Gamma, E; Helm, R.; Johnson, R; Vlissides John – "**Design Patterns: Abstraction and Reuse of Object-Oriented Design"** - Lecture Notes in Computer Science; IBM Research Center; Universitiy Of Illinois; 1993.

[GHJV 1994] Gamma, E; Helm, R.; Johnson, R; Vlissides John – "**Design Patterns**" – Addison Wesley Verlag, 1994.

[HD 2002] Hillmann, Diane I. - **Using Dublin Core** - National Science Digital Library Project at Cornell, Department of Computer Science, Cornell University, Ithaca, New York, USA; 16.07.2002; Online am 05.08.2002, http://dublincore.org/documents/2000/07/16/usageguide/

[HM 1995] Hearst, Marti A. - „**Visualization Of Term Distribution In Full Text Information Access**" - Xerox Palo Alto Research Centre, 1995.

[HMMR 1999] Handschuh, Siegfried; Mann, Thomas M.; Mußler, Gabriela; Reiterer, Harald - „**Die Entwicklung eines Business Intelligence Systems zur Beschaffung von Geschäftsinformationen im WWW**" - aus: Semar, Wolfgang; Kuhlen, Rainer (Hrsg.): Information Engineering. Proceedings des 4. Konstanzer Informationswissenschaftlichen Kolloquiums (KIK '99) Konstanz (Universitätsverlag Konstanz) 1999. (=Schriften zur Informationswissenschaft 36) S. 171-182.

[HU 2002] Huber, Ulrich - „**Metadaten in der Geoinformatik**" - Technische Universität München, Institut für Geodäsie, GIS und Landesmanagement, März 2002.

[IP 2002] INVISIP Project - „**Information Visualization For Site Planning**" - http://www.invisip.de, online am 31.7.2002.

[JC 2002] Jetter, C. – "**Heuristische Evaluation des Java Prototypen**" – Universität Konstanz, 2002. [Unveröffentlicht]

[KK 1998] Kaugars, Karlis J. – "**Integrated multi scale text retrieval visualization**" - In: Karat, Clare-Marie; Karat, John; Horrocks, Ian (Eds.): CHI 1998: Conference Proceedings Human Factors in Computing Systems. Conference: Los Angeles, CA, April 21-23 1998. New York (Addison-Wesley) 1998. p. 307-308.

[MD 1999] Mayhew, D. J. – "**The Usability Engineering Life Cycle" -** Morgan Kaufmann Verlag, März 1999.

[MKRE 2002] Müller, F.; Klein, P.; Reiterer, H.; Eibl, M. - „**The Supertable + Scatterplot Visualization For Metadata Retrieval**" - University Of Konstanz, Department Of Computer & Information Science; IZ Social Science Information Centre; 2002.

[MR 1999] Mann, T. M.; Reiterer, H. – „**Case Study: A Combined Visualization Approach for WWW-Search Results**" - IEEE Information Visualization Symposium 1999 Late Breaking Hot Topics Proceedings

[MRC 1986] Mackinlay, J.D.; Robertsen G.G; Card S.K. – "**The perspective wall : Detail and context smoothly integrated**" – In "Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems", Seiten 16 – 23, ACM, April 1986

[MRM 2000] Mußler, G.; Reiterer, H.; Mann, T. M. - **INSYDER - Information Retrieval Aspects of *a* Business Intelligence System** - Knorz, G.; Kuhlen, R. (Eds.): Proceedings des 7. Internationalen Symposiums für Informationswissenschaft. Conference: Darmstadt, Germany, November 8-10 2000. Konstanz. Germany (UKV Universitätsverlag Konstanz) 2000. p. 127-143.

[MT 1999] Mann, Thomas M - "**Visualization of WWW-Search Results**" - Proceedings Tenth International Workshop on Database and Expert Systems Applications. Conference: September 1-3, 1999

[MT 2001] Mann, Thomas M; "**Visualization of Search Results from the World Wide Web**" – Disseratation, Universität Konstanz, Fachbereich Informatik und Informationswissenschaft, 7.10.2001.

[NDA 1993] Norman, D. A. – „**Things that make us smart**" – Addison Wesley, 1993

[NHM 1997] Nielson, G. M.; Hagen, H.; Muller, H. – „**Scientific Visualization: Overviews, Methodologies, and Techniques**" – Los Alamitos, CA: IEEE Computer Society Press, 1997.

[NS 2002] **Netzsizer**, http://www.netsizer.com/, online am 3.7.2002

[PA 2001] Pasetti, A. – Lecture **Software Frameworks** (Modules) – Universität Konstanz, Wintersemester 2001/2002.

[PW 1994] Pree, Wolfgang C. – "**Meta Patterns-A Means For Capturing the Essentials of Reusable Object-Oriented Design**", C. Doppler Laboratory for Software Engineering, Universität Linz, 1994.

[PW 1995] Pree, Wolfgang C. – "**Design Patterns for Object-Oriented Software Development**" - Addison Wesley/ACM Press, 1995

[PW 1995] Pree, Wolfgang C. – "**State-of-the-art Design Pattern Approaches – An Overview**" - TOOLS , Paris, 6-9 March 1995

[PW 1997] Pree, Wolfgang C. **– "Component-Based Software Development - A New Paradigm in Software Engineering? " –** 1997.

[RF 1997] Rusch-Feja, D. - **Mehr Qualität im Internet: Entwicklung und Implementierung von Metadaten** - In: Ockenfeld, Marlies/Schmidt, Ralph (Hrsg.): 19. Online-Tagung der DGD. Die Zukunft der Recherche – Rechte, Ressourcen und Referenzen. Frankfurt am Main 14. bis 16. Mai 1997. Frankfurt/Main: DGD. 113-130.

[RM 1993] Robertsen, George G.; Mackinlay J.D.; - „**The document lens**“ – In „Proceedings of the ACM Symposium on User Interface Software and Technology”, ACM Press, November 1993

[RMMH 2000] Reiterer, H.; Mußler, G.; Mann, T. M.; Handschuh, Siegfried – “**Insyder - An Information Assistant for Business Intelligence**” - to appear in: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Developement in Information Retrieval, July 14-18, 2000 Athens, Greece

[SF 1993] Stone, Maureen C; Fishkin Ken, Bier, Eric A. - „**The Moveable Filter As A User Interface Tool**“ - XEROX Parc, 1993.

[SF 1995] Stone, M. C; Fishkin K. - „**Enhanced Dynamic Queries Via Moveable Filters**“ - XEROX Parc, 1995.

[SK 2002] Schumann, Heidrun; Kreuseler, Matthias – “**A Flexible Approach for Visual Data Mining**” - IEEE Transactions on visualization and computer graphics, vol. 8, Nummer 1, Januar-März 2002

[SM 1987] Salton G.; McGill, M.J. – “**Information Retrieval - Grundlegendes für Informationswissenschaftler**“ - McGraw-Hill: Hamburg etc., 1987

[ST 2002] Suchmaschinentricks, http://www.suchmaschinentricks.de/, Certo IT Solutions & Training GmbH, online am 01.07.2002

[SU 2002] Streit, U. – „**Einführung in Geoinformationssysteme**“ - Institut für Geoinformatik der Universität Münster, http://castafiore.uni-muenster.de/vorlesungen/geoinformatik/, online am 10.08.2002.

[TR 1997] Tufte, E. R. – „**Visual Explanation: Images And Quantities, Evidence And Narrative**“ – Cheshire, CT: Graphics Press, 1997.

[UK 1998] Universität Konstanz, AG Informationsysteme – „**INSYDER, Software Requirements Analysis**“ – Dezember 1998.

[UK 1998] Universität Konstanz, AG Informationsysteme – „**INSYDER, System Design**“ – ARISEM, Juni 1998.

[UK 1999] Universität Konstanz, AG Informationsysteme - „**Application Styleguide**“ - 1999.

[UK 1999] Universität Konstanz, AG Informationsysteme – „**INSYDER, Design Of The Evaluation Agent**“ – Universität Konstanz; ARISEM, Juni 1999.

[UK 1999] Universität Konstanz, AG Informationsysteme - „**Internet Systeme De Recherche - Technical Specification**“ - Januar 1999.

[UK 1999] Universität Konstanz, AG Informationssysteme – „**INSYDER, Analysis of alternative technical solutions“ – Cybion, Paris 1999.**

[VB 1996] Veerasamy, A.; Belkin, N. J. – “**Evaluation of a Tool for Visualization of Information Retrieval Results**” - In: Frei, Hans-P. et al. (Eds.): SIGIR 1996: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval Conference: Zürich, CH, August 18 -22 1996. New York (ACM Press) 1996. p. 85-92.

[VK 2002] Vogen, Kirk - "**Create native, cross-platform GUI applications - How GCJ, Linux, and the SWT come together to solve the Java UI conundrum**" - I/T Specialist, IBM Global Services April 2002

[YR 1999] Yates-Baeza, Ricardo; Ribeiro-Neto, Berthier – "**Modern Information Retrieval**" – Addison Wesley Verlag, 1999.

8. Abbildungs-, Tabellen- und Quellcodeverzeichnis

8.1. Abbildungsverzeichnis

Reusable Object-Oriented Design", C. Doppler Laboratory for Software Engineering, Universität Linz, 1994. Seite 6.

8.2. Tabellenverzeichnis

8.3. Quellcodeverzeichnis

9. Anhang

Dieser Bachelor Arbeit ist eine CD-ROM mit folgendem Inhalt beigefügt:

- Bachelor Arbeit in Version für einseitigen und zweiseitigen Druck
- Quellen
- Bilder
- ScreenCam Film zur Präsentation von GranularityTable und Scatterplot