Bachelorarbeit

Visualisierung von Multi-Data-Points in einem 3D-Scatterplot

Konzeption & Implementierung
innerhalb eines Metadaten Browsers

Philipp Liebrenz

1. Gutachter: Prof. Dr. Harald Reiterer
2. Gutachter: Prof. Dr. Oliver Deussen

Universität Konstanz
FB Informatik und Informationswissenschaft
Arbeitsgruppe Mensch-Computer Interaktion

KURZFASSUNG

Als Multi-Data-Point bezeichnet man die Überlagerung mehrerer Datenpunkte in einer Visualisierung, welche aufgrund gleicher Eigenschaften auf die gleiche Position im Raum abgebildet werden. Die Datenpunkte überlappen sich und sind für den Betrachter nicht mehr gesondert erkennbar und identifizierbar. Diese Arbeit stellt ein Konzept zur Visualisierung solcher Multi-Data-Points vor und dokumentiert die Implementierung dieses Konzepts innerhalb des VisMeB 3D-Scatterplots. Der 3D-Scatterplot ist eine Visualisierungskomponente des **Vis**uellen **Me**tadaten **Br**owsers (VisMeB)[1], der unabhängig einer Anwendungsdomäne den Benutzern bei der Suche und Exploration auf großen Datenmengen unterstützen soll. Die neue Multi-Data-Point-Visualisierung macht dem Benutzer die Elemente des Multi-Data-Points zugänglich und ermöglicht ein effektives und effizientes Explorieren der Elemente.

ABSTRACT

A Multi-Data-Point describes the overlapping of several data points within visualizations, which due to their identical properties were represented on the same position in a scatterplot. The data points overlap themselves and therefor the viewer does not recognize them as different. This work presents a concept for the visualization of such Multi-Data-Points and documents the implementation of this concept within the VisMeB 3D-Scatterplot. The 3D-Scatterplot is a visualization component of the **Vi**sual **Me**ta Data **B**rowsers (VisMeB)[1], that independently from an application domain supports users in large data sets research. The new Multi-Data-Point visualization makes the elements of the Multi-Data-Point accessible for the user and enables an effective and efficient browsing through the elements.

[1] VisMeB - Visueller Metadaten Browser, Forschungsprojekt der AG Mensch-Computer Interaktion an der Universität Konstanz unter der Leitung von Prof. Dr. Reiterer.

Meiner Familie, Werner König, Frank Müller, Stefanie Haber, Fabian Arnold und meinen Kommilitonen und Freunden:

Danke für Rat, Kritik, Unterstützung und Hilfe.

INHALTSVERZEICHNIS

1 EINLEITUNG

Das visuelle Darstellen von Fakten und Informationen ist keine Disziplin der Neuzeit. Schon lange bevor es Computer gab, versuchten die Menschen gesammeltes Wissen bildlich darzustellen, um dem Betrachter das Verständnis zu erleichtern. So wurde etwa bereits ca. 6200 Jahre. v. Chr. ein erster Stadtplan einer babylonischen Siedlung angefertigt[2]. Aber auch in Wissenschaft und Forschung bediente man sich früh grafischer Darstellungen um Erkenntnisse festzuhalten, zu präsentieren und zu erklären. Alexander von Humboldt benutzte schon im Jahre 1817 ein Diagramm mit Isothermen, um die Korrelation zwischen mittlerer Temperatur und Breitengrad darzulegen.

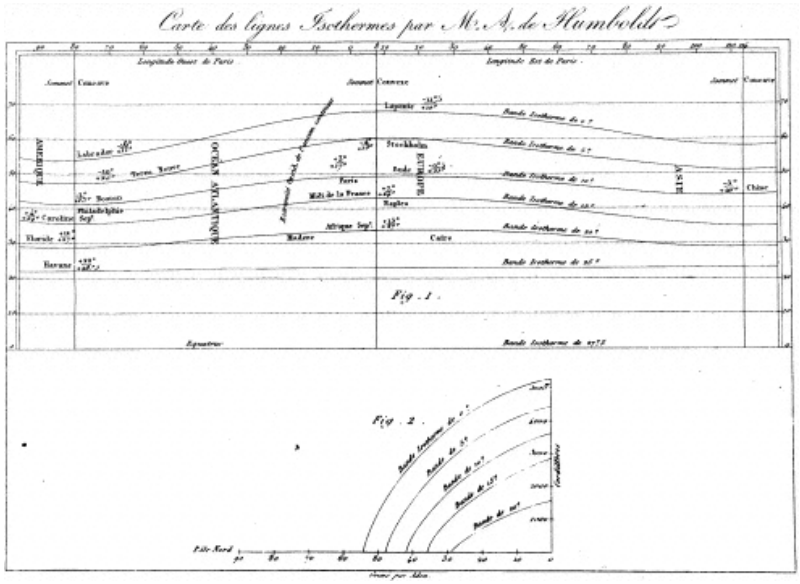


Abbildung 1: von Humboldt, A. (1817). Sur les lignes isothermes. *Annales de Chimie et de Physique*, aus : Friendly, M., Denis, D.J., Milestones in the history of thematic cartography, statistical graphics, and data visualization.

[2] http://www.atamanhotel.com/catalhoyuk/oldest-map.html

Eine der grundlegendsten Visualisierungsformen ist das kartesische Koordinatensystem. Seine Wurzeln liegen in der analytischen Geometrie, als zum ersten Mal versucht wurde, den Raum in Zahlen und Koordinaten zu fassen. Der interaktive 3D-Scatterplot ist ein Nachkomme dieser Koordinatensysteme und in Funktion, Lesbarkeit und Aussagekraft um ein Vielfaches mächtiger. Trotzdem findet er heute noch relativ wenig Verbreitung. Das liegt zum einen daran, dass erst vor kurzem die Möglichkeiten zur technischen Umsetzung geschaffen wurden und zum anderen, dass seine Anwendung hohe Ansprüche an den Benutzer stellt.

Eine weitere Schwierigkeit bei der Visualisierung von Informationen im 3D-Scatterplot ist die Darstellung sogenannter Multi-Data-Points. Sie entstehen, wenn mehrere dazustellende Objekte auf allen Achsen die gleichen Ausprägungen haben. In der Visualisierung überdecken sich diese Datenpunkte und sind daher nicht mehr gesondert erkennbar.

Vor dieser Problematik stand man auch bei der Entwicklung und Implementierung eines 3D-Scatterplot als Visualisierungskomponente des VisMeB Metadaten Browsers, einem Framework verschiedener Visualisierungen zur Darstellung von Metadaten[3], das im Zuge des Projekts VisMeB an der Universität Konstanz entwickelt wurde (siehe auch Kapitel 3). Der Browser soll die Benutzer bei der Suche auf großen Datenmengen aus verschiedensten Anwendungsbereichen unterstützen und ihnen helfen, die relevanten Objekte zu identifizieren.

Diese Arbeit stellt die Multi-Data-Point Problematik vor und dokumentiert Konzept und Implementierung des für den VisMeB 3D-Scatterplot gefundenen Lösungsansatzes.

[3] Metadaten („Daten über Daten“) sind strukturierte Daten, die Datenobjekte näher beschreiben. Dies können Eigenschaften der Objekte oder auch andere weitergehende Informationen über die Objekte sein. Mit Hilfe von Metadaten kann das beschriebene Objekt besser charakterisiert und deshalb in einer Menge anderer Objekte besser identifiziert werden.

2 VISUALISIERUNG VON INFORMATIONEN

2.1 Visualization

In der heutigen Welt sind grafische Repräsentationen allgegenwärtig. Nicht nur in den Medien, auch im Berufsalltag werden uns Informationen verschiedenster Art in Form von Grafiken oder Diagrammen unterbreitet. Sie sollen durch die visuelle Repräsentation ein effektives Auswerten der zugrunde liegenden Daten ermöglichen.

Dem Betrachter soll

- das Verständnis der Daten erleichtert,
- die enthaltene Information besser kommuniziert und
- eine effektive Analyse ermöglicht werden.

In „*Readings in Information Visualization*“ [2], herausgegeben von Stuart Card, Jock Mackinlay und Ben Schneiderman, wird Visualisierung (engl. Visualization) definiert als

> *“The use of computer-supported, interactive, visual representation of data* ***to amplify cognition****.”*

Zwar kommt dieses Zitat aus dem Zusammenhang der *Information Visualization* (siehe Kapitel 2.2) und beschränkt sich daher auf computergestützte Visualisierung, aber es verdeutlicht, dass das Ziel einer geeigneten visuellen Repräsentation nicht das einfache „Verbildlichen“ von Daten ist, sondern den kognitiven Prozess der Entstehung, des Erlernens und des Benutzens von Wissen im Betrachter zu unterstützen.

Wie wichtig dieser illustrative Aspekt einer Visualisierung sein kann, verdeutlicht das bekannte Beispiel der Choleraverbreitung in London im Jahre 1854 (aus [29]). Damals konnte durch das geeignete Visualisieren statistischer Daten die Ursache der Epidemie gefunden und die weitere Verbreitung verhindert werden.

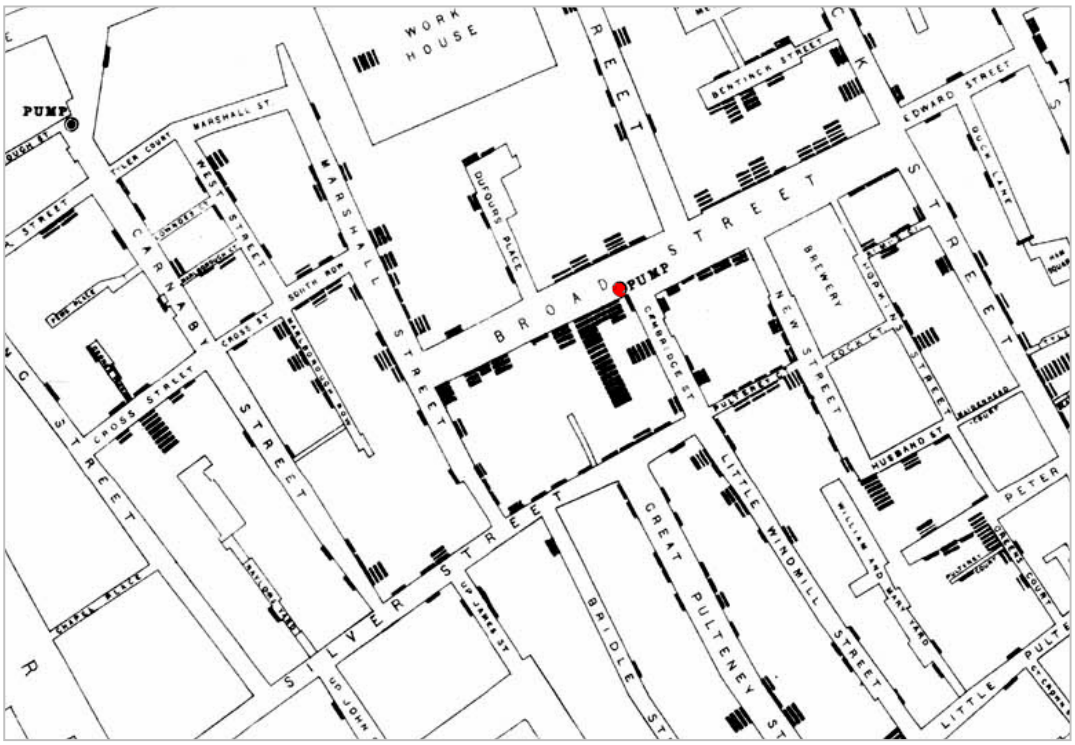


Abbildung 2: Ausschnitt aus der Visualisierung, die die Verbreitung der Cholera in London (1854) dokumentiert. Durch das Visualisieren der Todesfälle (durch Balken parallel zu dem Gebäude in dem die Personen gestorben sind) konnte eine Frischwasserpumpe (rot markiert) als verbreitender Faktor der Cholerabakterien identifiziert werden. Aus: John Snow, The historical treatise, London, 1855

2.2 Information Visualization

Eine besondere Form der Visualisierung von Daten ist die *Informationsvisualisierung* (eng. „*Information Visualization*“). Sie entstand aus dem Bedarf, neue Datenformen, wie sie in modernen Informationssystemen und in großen Mengen vorkommen, grafisch zu veranschaulichen, um die enthaltenen Informationen zugänglich zu machen.

Der Begriff Information Visualization wurde Anfang der 90er Jahre am XEROX-Palo Alto Research Center in den USA geprägt. Dort beschäftigte man sich mit der Entwicklung neuer visueller Metaphern für Informationsräume. Zu den bekanntesten Entwicklungen dieser Arbeitsgruppe gehören die „*Perspective Wall*“[16] und der „*Cone Tree*“[25]. Mit dem Ausdruck Information Visualization wurden die Konzepte zur Visualisierung abstrakter Daten bezeichnet.

Definition:

Informationsvisualisierung (engl. Information Visualization) umfasst alle Werkzeuge und Methoden zur visuellen Darstellung abstrakter Informationen, wie sie in Datenbanken, digitalen Bibliotheken oder anderen großen Dokumentsammlungen auftreten.

Nach Rolf Däßler, in: Virtuelle Informationsräume mit VRML [5]

Gegenstand der Informationsvisualisierung ist also die visuelle Repräsentation von Daten ohne Raumbezug und ohne physische Abbildung. Dies können zum Beispiel Finanzdaten, Verkaufsdaten eines Unternehmens oder Spielfilm-Metadaten sein. Ein besonders Merkmal dieser Form von Daten ist, dass sie meist viele Datenelemente (Tupel) besitzen und die Elemente ihrerseits viele Attribute (Dimensionen). Oftmals sind dies zentrale Probleme der Informationsvisualisierung, die es zu lösen gilt.

3 VISMEB – EIN METADATEN BROWSER

3.1 Das VisMeB Projekt

Unter der Leitung von Prof. Reiterer arbeitet die AG Mensch-Computer Interaktion der Universität Konstanz am Forschungsprojekt „VisMeB“. Gegenstand des Projekts ist die Entwicklung und Evaluation eines **Vis**uellen **Me**tadaten **B**rowsers. Der Browser soll die Benutzer bei der Exploration und der Suche auf großen Datenmengen unterstützen und ihnen helfen, in großen Treffermengen die für sie relevanten Objekte, zu identifizieren. Die zentrale Forschungsidee ist, durch das Verknüpfen von bewährten und neuen Visualisierungstechniken und dem Wissen aus der Mensch-Computer Interaktion eine neue Form der visuellen Interaktion mit dem Medium Computer zu finden.

VisMeB baut auf den Erfahrungen und Vorarbeiten bereits zweier Projekte auf:

Im Rahmen des Projekts **INSYDER**[4] **(Internet Systeme de Recherche)** wurde ein visuelles Suchsystem für das World Wide Web entwickelt (siehe auch [22] und [23]). Ziel des Projekts war, kleine und mittelständische Unternehmen mit geschäftsrelevanten Informationen aus dem Internet zu versorgen. Neben der klassischen Listendarstellung der Ergebnismenge bietet das System eine Reihe von weiteren Visualisierungen, wie zum Beispiel Scatterplot, Barcharts oder Tilebars. Ein weiterer Bestandteil dieses Projekts war die umfassende Evaluation des Suchsystems bezüglich der Gebrauchstauglichkeit der Visualisierungen (siehe [17]). Die Erkenntnisse der Untersuchung flossen in das Projekt INVISIP.

[4] EU ESPRIT Project No. 29232 INSYDER

Das EU-geförderte Projekt **INVISIP**[5] **(Information Visualization for Site Planning)** verfolgt das Ziel, Standortentscheidungen zu begleiten und die beteiligten Parteien bei der Entscheidungsfindung und den nachfolgenden Prozessen zu unterstützen (siehe auch [8]). Die Arbeitsgruppe an der Universität Konstanz entwickelt hierfür einen Metadaten Browser zur Suche auf Geometadaten und zur Ergebnisvisualisierung. Das Suchsystem soll Architekten, Planern, und Geografen beim Finden von entscheidungsrelevanten Geometadaten für Standortentscheidungen unterstützen.

VisMeB profitierte von den Erfahrungen beider Projekte. Aus den Erkenntnissen der Evaluation von INSYDER entstanden neue Formen der Visualisierung. Klassische Konzepte, wie die tabellarische Darstellung, wurden mit effektiveren und effizienteren grafischen Visualisierungen kombiniert und in das INVISIP-System integriert (vgl. [13]). Für VisMeB wurden die Visualisierungen erweitert und neue hinzugefügt.

Während das Vorgängerprojekt INSYDER nur Suchergebnisse aus dem World Wide Web visualisierte und auch INVISIP sich auf Metadaten geografischer Objekte beschränkte, verfolgt VisMeB einen universellen Ansatz. Durch einen stark generischen Aufbau des Browsers soll VisMeB in den unterschiedlichsten Anwendungsdomänen zum Einsatz kommen können, zum Beispiel bei der Suche im Web, der Suche in Metadateninformationssystemen für Geodaten, aber auch für die Suche auf pharmakologischen oder medizinischen Datenbanken ist eine Anwendung denkbar.

Zur Entwicklung und Evaluation wurde das System bisher an drei Datenbanken mit Metadaten aus verschiedenen Anwendungsdomänen angebunden. Dabei handelt es sich um eine Datenbank mit Geometadaten-Objekten, eine Filmdatenbank und einer Datenbank mit Metadaten über Webdokumente.

[5] EU IST-2000-29640 INVISIP

3.2 Der VisMeB Metadaten Browser

Der VisMeB Metadaten Browser bietet den Benutzern ein grafisches Interface zur explorativen Suche auf Metadaten-Datenbanken. Es visualisiert die Metadaten gefundener Objekte und stellt die Relevanzen der Objekte bezüglich der Suche grafisch dar. Der Benutzer erhält visuelle Repräsentationen, die es ihm ermöglichen, die Objekte schnell und präzise zu identifizieren, zu bewerten und untereinander zu vergleichen.

Im Folgenden werden die einzelnen Komponenten des Systems anhand eines Szenarios vorgestellt. Es umfasst die Anbindung an eine Datenbank, die Formulierung der Suchanfrage und die einzelnen Suchergebnisvisualisierungen. Dabei werden zur Verdeutlichung sowohl Beispiele bei der Suche und Exploration der Filmdatenbank als Beispiele bei der Suche auf den Webdokumenten angeführt.

3.2.1 Das Assignment Tool und Konfiguration

VisMeB bezieht die Daten zur Visualisierung von einer beliebigen Datenbank. Mit Hilfe des Assignment Tools (siehe Abbildung 3) kann die Anbindung des Systems an die Datenbank konfiguriert und bestehende Konfigurationen editiert werden.

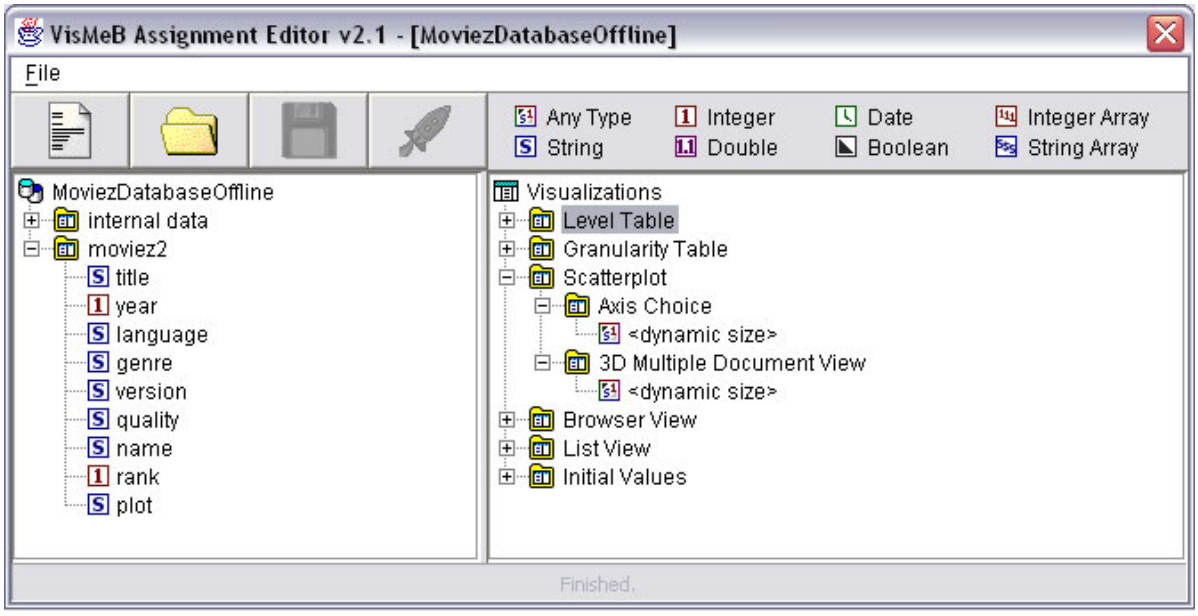


Abbildung 3: Das VisMeB Visual Configuration and Assignment Tool

Durch ein benutzerfreundliches Interface kann der Anwender die vorhandenen Tabellen und Views der Datenbank selbst explorieren und die für ihn interessanten Datenbankfelder auswählen und einer Visualisierung zuordnen. So können auch Benutzer ohne Programmierkenntnisse die Visualisierungen für ihre spezifische Anwendung anpassen.

Das Assignment Tool ist zweigeteilt: Auf der linken Seite wird die Tabellenstruktur der aktuellen Datenbank in Form einer Baumhierachie angezeigt. Jeder Knoten steht für eine Tabelle oder eine View der Datenbank. Die Blätter sind die Attribute der Datenelemente der Tabelle oder der View. Auf der rechten Seite werden die unterschiedlichen VisMeB-Visualisierungen und ihre Zuordnungsmöglichkeiten aufgelistet. Die Zuordnungsmöglichkeiten zeigen an, welcher Datentyp (Wertebereich) für die jeweilige Visualisierung möglich bzw. sinnvoll ist. Die möglichen Zuordnungen werden mit Hilfe kleiner Icons symbolisiert (siehe Tabelle 1). Ebenso wird jedem Feld der Datenbank ein Symbol für dessen Datentyp zugeordnet.

Icon	Datentyp
	Datum
	String
	Boolean
	Integer
	Double
	Sting Array
	Integer Array
	Any Type

Tabelle 1: VisMeB, Datentypen des Assignment Tools

Der Anwender kann nun einfach per drag`n`drop ein Datenbankfeld einer Visualisierung zuordnen. Visualisierungsfelder mit dem Icon „Any Type“ akzeptieren jeden Datentyp. Die Anzahl der Datenbankfelder (Attribute), die einer Visualisierung zugeordnet werden können ist abhängig vom Typ der Visualisierung. Daher haben manche Visualisierung eine feste Anzahl an Visualisierungsfeldern („Fixed Size“), bei anderen ist die Anzahl der Visualisierungsfelder dynamisch erweiterbar („Dynamic Size“). Unter dem

Visualisierungsfeld „Initial Values“ können die Attribute zugeordnet werden, die in einer Visualisierung zu Beginn angezeigt werden sollen.

Die fertigen Zusammenstellungen können als XML-Datei abgespeichert werden und immer wieder benutzt oder andern zugänglich gemacht werden. Für verschiedene Anwendungen können speziell angepasste Konfigurationen zusammengestellt oder Konfigurationen als Standard definiert werden.

3.2.2 Die Suchanfrage

Sind die Visualisierungen konfiguriert, kann der Suchprozess gestartet werden. VisMeB bietet zwei Möglichkeiten, eine Suchanfrage zu formulieren. Zum einen durch die Eingabe von Suchtermen („*Textual Search*“), zum andern durch ein grafisches Interface, der CircleSegmentView.

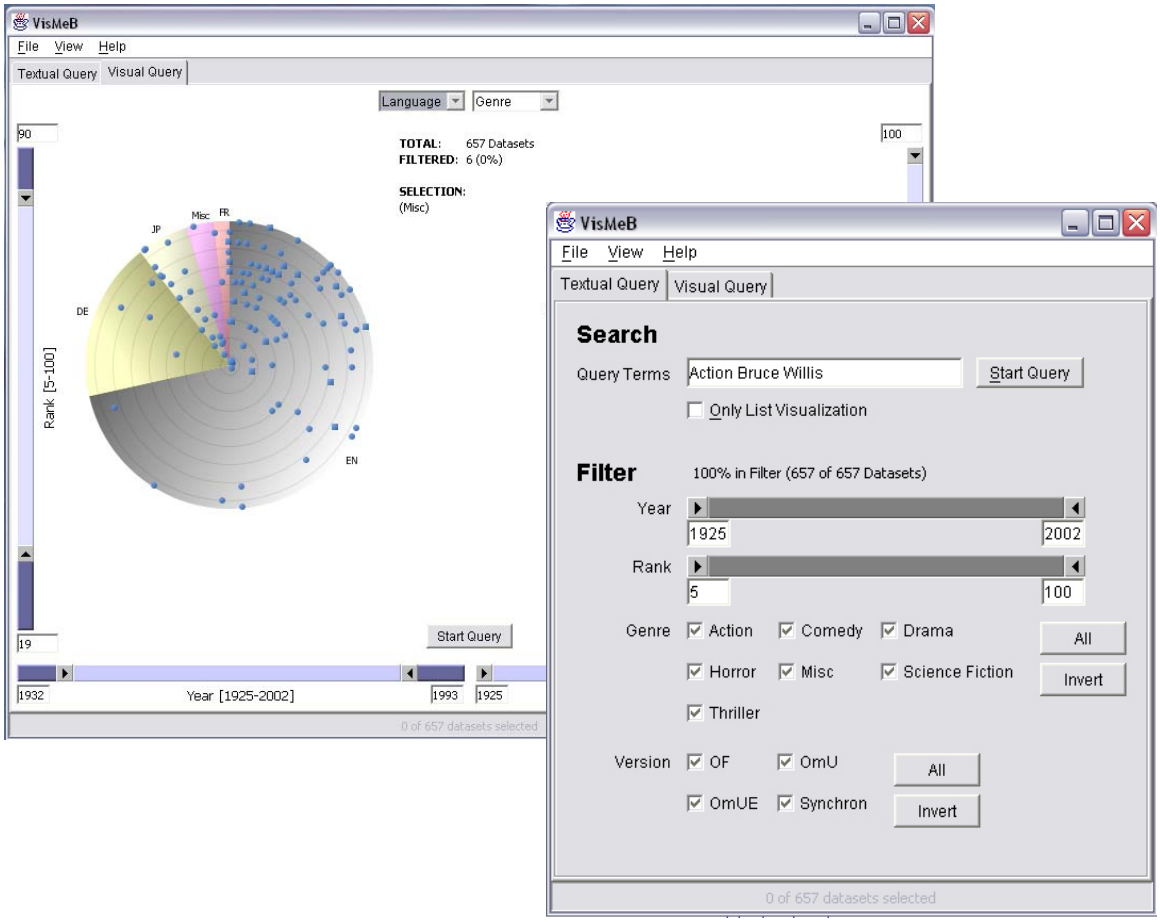


Abbildung 4: VisMeB grafischer (links) und textueller Suchanfragedialog (rechts) bei der Suche auf der Filmdatenbank

Zusätzlich zu den Suchtermen können bei der textuellen Suche Filter definiert werden, die die gesuchten Objekte näher definieren und die Treffermenge einschränken. Numerische Attribute können durch Doppelslider (vergleiche Alphaslider nach Ahlberg et al. [1]) oder durch Eingabe der Grenzwerte, ordinale Attribute können durch Checkboxes eingeschränkt werden. Wieviele Objekte nach Anwendung der Filter in der Treffermenge verbleiben, wird sowohl prozentual als auch absolut in einer Textzeile oberhalb der Filterattribute angezeigt.

Diesem Prinzip der dynamischen Suche bedient sich auch die CircleSegmentView. Die Ausprägungen der Attribute können durch Doppelslider eingeschränkt werden. Dadurch kann die Treffermenge näher spezifiziert und auf ein interessantes Maß an Objekten verringert werden.

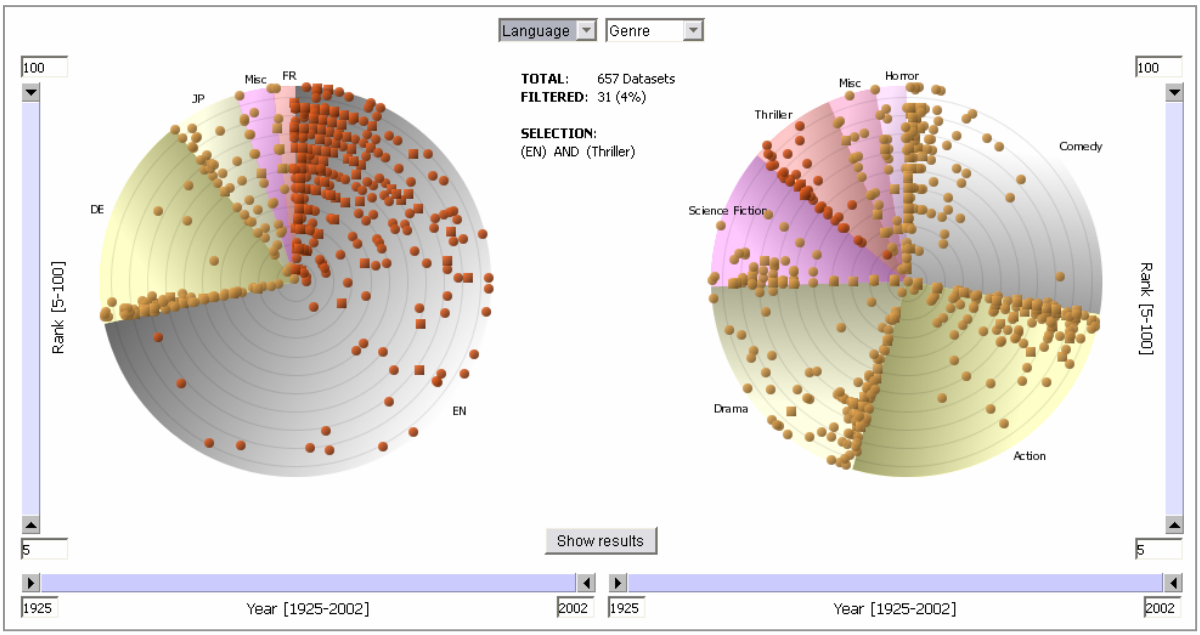


Abbildung 5: VisMeB CircleSegmentView zur Query Preview

Die CircleSegmentView visualisiert die Objekte und deren Attribute innerhalb zweier Pie Charts. Beide Pie Charts sind unabhängig voneinander konfigurierbar. Jedes Metadatenobjekt ist durch einen Punkt symbolisiert. Selektierte Objekte sind dunkler. Multi-Data-Points (siehe Kapitel 4) werden als Quadrat visualisiert.

Die Pie Charts Segmente stellen den Anteil der Ausprägungen eines Metadatenattribut dar. So erkennt man zum Beispiel in Abbildung 5, dass fast 75% aller in der Filmdatenbank vorkommenden Filme englische sind (linker Pie Chart, graues Segment). Die Metadatenobjekte werden innerhalb der jeweiligen Kategorie eingeordnet. Die Beschriftung der Segmente entspricht den Ausprägungen des Attributs. Welches Attribut visualisiert wird, kann durch ein Drop-Down-Menü in der Mitte der Ansicht gewählt werden.

Für die genaue Position eines Objekts innerhalb des Segments sind die Ausprägungen zweier weiterer Attribute verantwortlich. Die Entfernung des Objekts zum Mittelpunkt des Pie Charts kodiert ein Attribut. Ein weiteres Mettadaten-Attribut wird durch den Winkel des Objekts innerhalb des Segments visualisiert. Farbverlauf und radiale Kreise zum Mittelpunkt unterstützen die Wahrnehmung des Benutzers bei dem Einordnen der Datenobjekte. Per Kontextmenü können die Belegungen gewählt werden.

Durch Alphaslider an der Seite bzw. unterhalb des Pie Charts, sowie durch textuelle Eingabefelder an den Enden der Alpha Slider kann die Treffermenge eingeschränkt werden. Das Einschränken hat direkte Auswirkung auf die Visualisierung. Gefilterte Objekte fallen aus der Visualisierung heraus und die Ansicht zoomt in den verbleibenden Wertebereich.

Durch diese Möglichkeit der direkten Manipulation der Treffermenge kann die CircleSegmentView nicht nur zur Query Preview (vergleiche [28]), sondern auch als allein stehende Visualisierung mit Filterfunktionalität verwendet werden. Durch eine boolesche Verknüpfung der beiden Pie Charts lassen sich komplexe Filter bzw. Suchanfragen modellieren.

3.2.3 Die Visualisierungen

Nachdem eine Suche formuliert und ausgeführt wurde, gelangt man zur eigentlichen Arbeitsumgebung von VisMeB, bestehend aus den verschiedenen Visualisierungen.

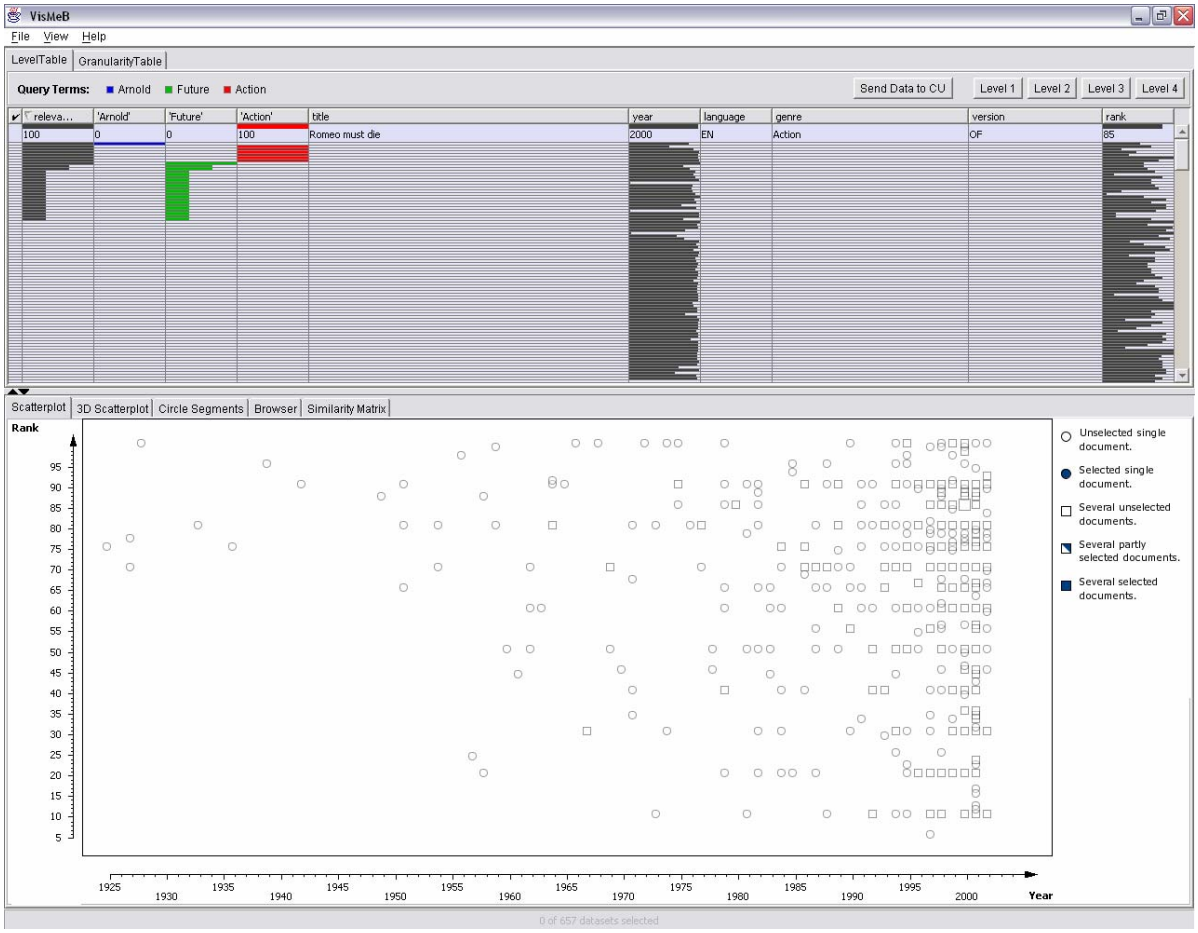


Abbildung 6: VisMeB Visualisierungen

Zwei zentrale Elemente der Visualisierungen des Metadaten Browsers sind die LevelTable und die GranularityTable. Sie befinden sich im oberen Bereich der Ansicht und können durch Karteireiter optional gewählt werden. Im unteren Abschnitt findet man weitere Visualisierungen: Den Scatterplot, 3D-Scatterplot, die CircleSegmentView und den Browser. Auch sie sind optional über Karteikartenreiter aktivierbar.

Die einzelnen Visualisierungen sind eng miteinander verbunden, d.h. Änderungen in der einen Ansicht haben Auswirkungen auf die anderen (vergleiche Snap-Together Visualizations [18]). Nach dem „*Brushing & Linking*“ Prinzip werden zum Beispiel Datensätze, die im Scatterplot selektieren werden, auch in der LevelTable und den anderen Visualisierungen selektiert. Grundsätzlich werden Dokumente, die durch den Mauszeiger in einer Visualisierung fokussiert werden, in den anderen farblich hervorgehoben.

Sowohl in LevelTable als auch in der GranularityTable werden in der obersten Zeile der Visualisierungen die aktuellen Suchterme angezeigt. Jedem Term wird eine Farbe zugeordnet, die in den verschiedenen Visualisierungen als Referenz auf den jeweiligen Suchterm benutzt werden kann.

LevelTable und GranularityTable sind Designvarianten der SuperTable, wie in [13] vorgestellt. Die Usability Studie des INSYDER-Systems hat gezeigt, dass die tabellenbasierte Darstellung von Informationen von den Benutzern besser akzeptiert wurde. Hingegen bieten komplexere Visualisierungen, wie „Bar Charts“, mehr Informationen auf einen Blick und erhöhen den „Joy of Use“. Die Super Table integriert verschiedene Visualisierungen, wie „Bar Charts“, „Tile Bars“ und farblich kodierte Textstellen in eine tabellarische Darstellung der Datenelemente, um die Vorteile der tabellarischen Darstellung und die der komplexeren Visualisierungsformen zu vereinen. Trotzdem soll der Benutzer nicht durch zu viele verschiedene Darstellungsformen überfordert werden. Man entschied sich daher, dass die unterschiedlichen Visualisierungen dem Benutzer ein unterschiedliches Maß an Detailinformationen („*Level of Detail*“) über die Datensätze geben sollten und in Form von Detailstufen wählbar sein sollten. Die Detailstufen sollten von einer rudimentären Sicht, in der es möglich ist, sich viele Datensätze anzeigen zu lassen und diese im Zusammenhang zu betrachten, bis hin zu einer sehr detaillierten Ansicht des einzelnen Datensatzes reichen. Dabei sollte der Benutzer aber stets erkennen, dass die verschiedenen Visualisierungen nur verschiedene Sichten auf den gleichen Datensatz bieten.
Dieses Konzept der unterschiedlichen Detailgrade nannte man „Granularity Concept“. Es wurde in LevelTable und GranularityTable unterschiedlich umgesetzt.

Im Folgenden werden die einzelnen Visualisierungen der Suchergebnisse von VisMeB näher vorgestellt.

Die LevelTable

Die LevelTable ist, wie bereits erwähnt, eine tabellenbasierte Visualisierung mit integrierten grafischen Visualisierungselementen. Jede Zeile der Tabelle steht für einen Datensatz. Die Spalten stellen die Gesamtrelevanz und die Einzelrelevanzen bezüglich der Suchterme sowie die einzelnen Attribute der Metadaten-Objekte dar.

Das „Granularity Concept“ wurde hier durch vier verschiedene detaillierte Ansichten auf die Daten realisiert. Der Detaillevel kann durch vier Buttons in der oberen, rechten Ecke gewählt werden.

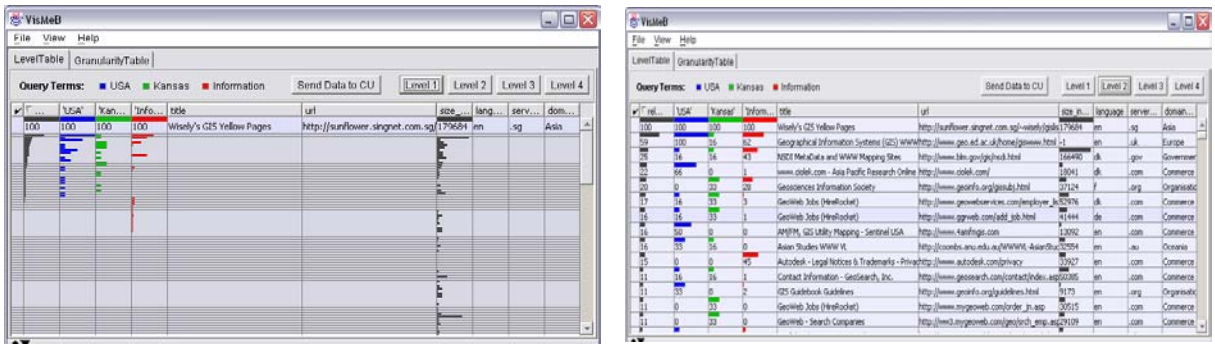


Abbildung 7: LevelTable, erstes Level (links) und zweites Level (rechts)

Im **ersten Level** erhält der Anwender eine nur wenige Pixel hohe Repräsentation der einzelnen Datenelemente. Die Relevanzen und numerische Attribute sind in Form von farblich kodierten Balken erkennbar. Beim Überfahren einer Zeile mit der Maus wird die Zeile vergrößert und die Attribute sind lesbar (Level 2), der Datensatz ist also im „*Focus of Interest*“ (Focus). Die anderen Datenelemente bleiben unverändert (Context) (basierend auf den Ideen der Table Lens [21]).

Im **zweiten Level** sind die Attribute aller Elemente lesbar. Zusätzlich werden unter der Balkendarstellung die Zahlenwerte der Relevanzen und der numerische Attribute textuell angezeigt.

Das **dritte Level** enthält zusätzlich zu den Metadaten-Attributen der Objekte die „*Relevance Curve*“. Sie stellt die Verteilung der Relevanz innerhalb des Objekts oder des beschreibenden Dokuments dar. Das Dokument wird in Segmente aufgeteilt und für jedes Segment die relative Häufigkeit der Suchterme abgetragen. Man erhält eine Art Balkendiagramm, aus dem man die relevanten Stellen des Dokuments ablesen kann.

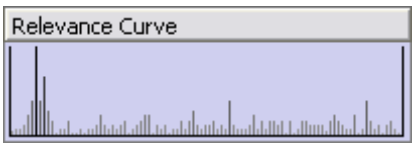


Abbildung 8: Relevance Curve

Dem „*Granularity Concept*“ entsprechend wird die „*Relevance Curve*“ im **vierten Level** um weitere Details erweitert. Statt nur die Gesamtrelevanz eines Dokument-abschnitts anzuzeigen, werden die Balken je nach Häufigkeit der einzelnen Suchterme aufgeteilt und durch die Farbkodierung veranschaulicht. Der Benutzer erkennt nun, wo welcher Suchterm im Dokument vorkommt. Beim Überfahren der einzelnen Segmente springt die BrowserView zu dem jeweiligen Abschnitt und hebt den Abschnitt farblich hervor (siehe Abbildung 10).

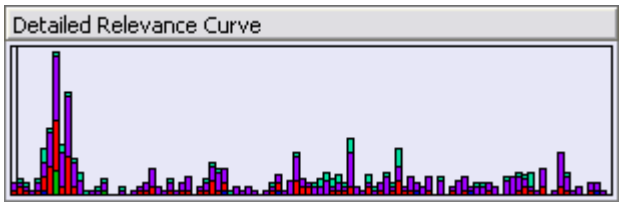


Abbildung 9: Detailed Relevance Curve

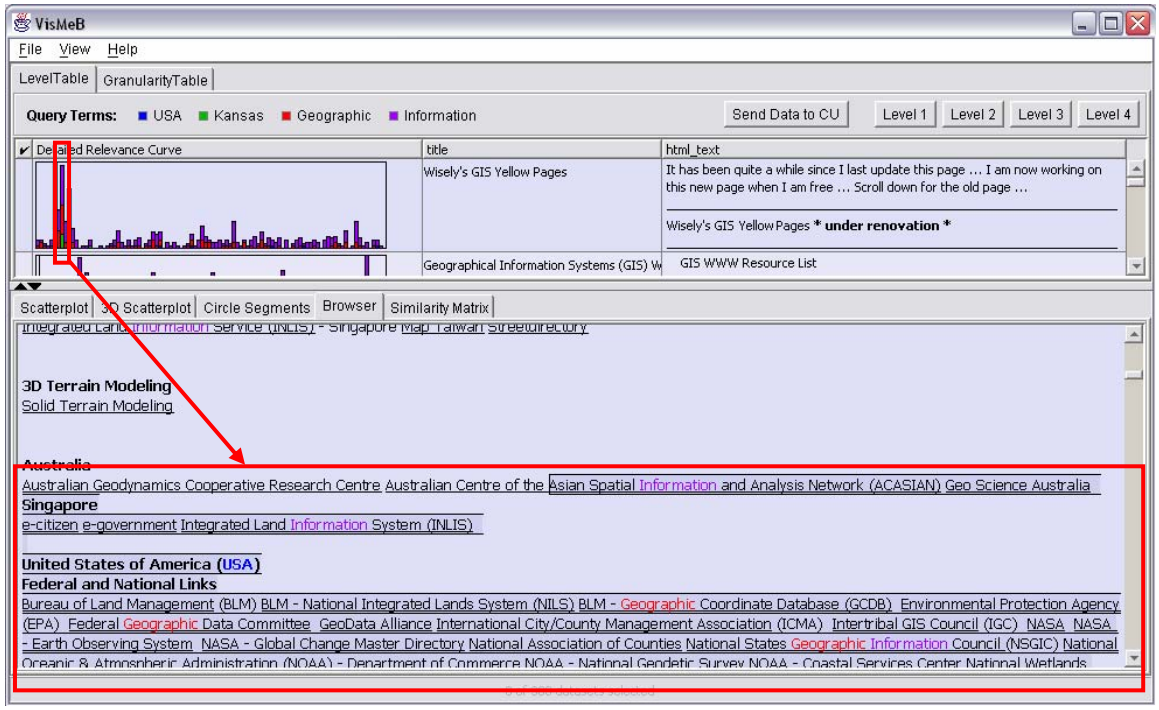


Abbildung 10: LevelTable im vierten Level und Browserview, Abschnittsmarkierung

Die GranularityTable

Der zweite Designentwurf der SuperTable ist die GranularityTable. Auch hier kann der Benutzer zwischen verschiedenen Detailstufen wählen. Statt vier Buttons besitzt die GranularityTable einen Slider. Idealerweise soll durch ihn die Detailmenge der Visualisierung bis hin zum eigentlichen Dokument stufenlos reguliert werden können. Da dies aber nur schwer umsetzbar ist, wurde versucht sich der Idee durch sechs Detailstufen anzunähern. Ein weiterer Unterschied zur LevelTable ist, dass das „*Level of Detail*“ nicht nur global geändert werden kann, sondern auch für jeden Datensatz separat durch einen Slider innerhalb der Tabellenzeile.

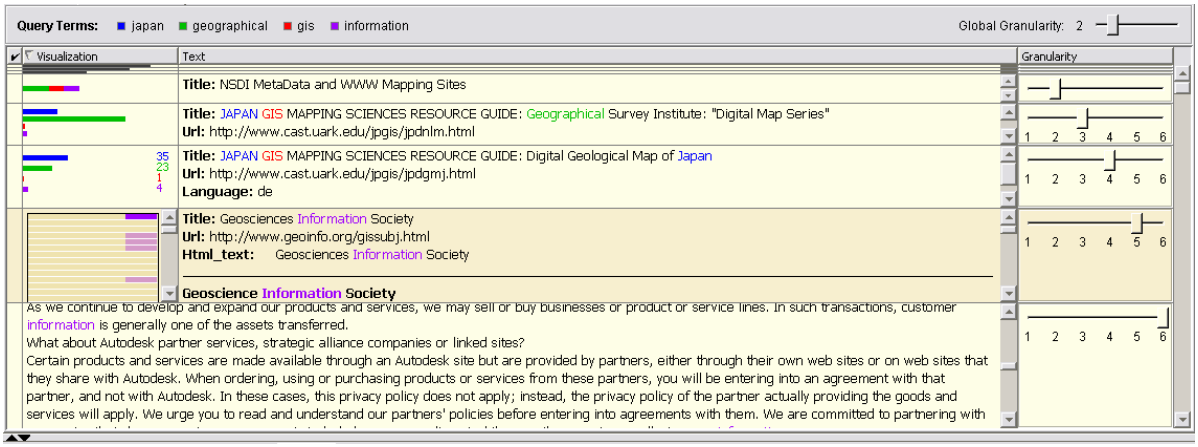


Abbildung 11: VisMeB GranularityTable Visualisierung der Webdokumentdaten, lokale Detailstufen 1 bis 6

Die GranularityTable hat in jeder Detailstufe nur vier Spalten: Selektion, ein integrierte Visualisierung, die je nach Detailstufe die Relevanz oder Relevanzenverteilung innerhalb des Dokuments visualisiert, ein beschreibendes Textfeld und eine Spalte, die den Slider zum lokalen Wechsel der Detailstufe enthält.

In der **ersten, geringsten Detailstufe** wird, vergleichbar der LevelTable, nur eine wenige Pixel hohe Repräsentation der einzelnen Datensätze gegeben. Auch hier ist nur der horizontale Balken, der die Gesamtrelevanz des Objekts darstellt, erkennbar.

Im **zweiten Level** wird der Relevanzbalken je nach Anteil der einzelnen Suchterme an der Gesamtrelevanz farblich kodiert. Die Textspalte enthält ein repräsentatives Attribut (wird im Assignment festgelegt, in Abbildung 12 der Titel des Webdokuments) des Objekts.

Die Einzelrelevanzen werden dann im **dritten Level** übereinander dargestellt und sind dadurch besser vergleichbar. Die Textspalte wird um nähere Angaben zum Metadatenobjekt ergänzt.

Das **vierte Level** ergänzt die Relevanzvisualisierung um die numerische Angabe der Einzelrelevanzen.

Im **fünften Level** werden die „*Bar Charts*“ durch eine neue Visualisierung, genannt „*Tile Bar*“ (in Anlehnung an [9]) ersetzt. Sie teilt das Dokument, ähnlich der „*Relevance Curve*“, in Segmente auf. Jede Zeile der „*Tile Bar*“ steht für ein Segment des Dokuments, jede Spalte für einen Suchterm. Segmente, die den jeweiligen Suchterm enthalten, werden in der „*Tile Bar*“-Darstellung durch die für den Suchterm reservierte Farbe, kenntlich gemacht. Beim Anklicken einer Zeile in der “*Tile Bar*“ wird das entsprechende Textsegment des Dokument fokussiert und markiert.

Im **sechsten Level** fällt die Spalte für die Visualisierung weg und stattdessen wird im Textfeld das komplette Dokument angezeigt.

Die CircleSegmentView

Die CircleSegmentView visualisiert die Objekte und deren Attribute innerhalb zweier Pie Charts, wie in Kapitel 1.4.2. bereits beschrieben wurde. Im Unterschied zur Query Preview Funktionalität im Suchdialog dient die CircleSegmentView hier als Filter.

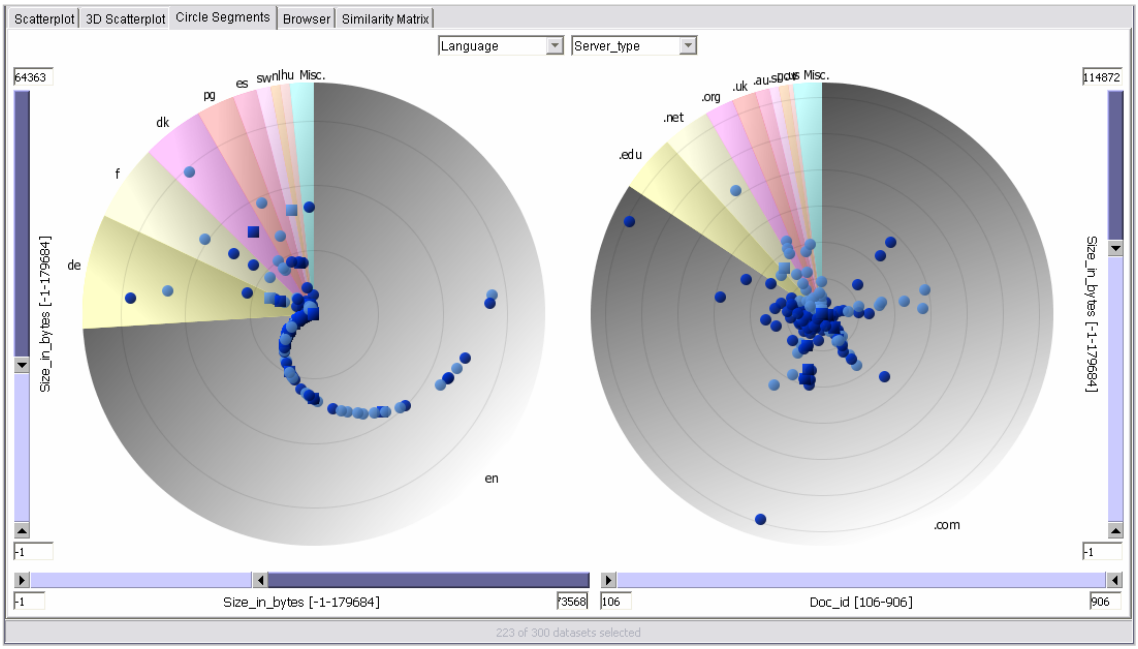


Abbildung 12: VisMeB CircleSegmentView

Durch die Alphaslider kann der Benutzer Wertebereiche mehrerer Attribute einschränken. Objekte, die diesen Kriterien nicht entsprechen fallen aus der Treffermenge und werden in den Visualisierungen nicht mehr angezeigt. Über boolesche Verknüpfung der beiden Pie Charts können die Kriterien logisch kombiniert werden.

Die BrowserView

Die BrowserView bietet eine Vollansicht der beschreibenden Metadatenobjekte. Mehrere Objekte können nebeneinander oder untereinander angeordnet und dadurch miteinander verglichen werden. Außerdem besitzt die BrowserView eine Zoomfunktion durch die die Darstellung vergrößert oder verkleinert werden kann. Auch die BrowserView unterstützt die Interaktion mit LevelTable und GranularityTable, wie z.B. das farbliche Hervorheben des fokussierten Dokuments und Selektion.

Der Scatterplot

Der VisMeB Scatterplot visualisiert die Daten in einem zweidimensionalen Koordinatensystem, wobei jede Achse mit einem Metadatenattribut frei belegt werden kann. Dadurch ist es dem Benutzer möglich, Abhängigkeiten zwischen zwei Attributen und auffällige Verteilungen der Ausprägungen, zum Beispiel Cluster, zu erkennen.

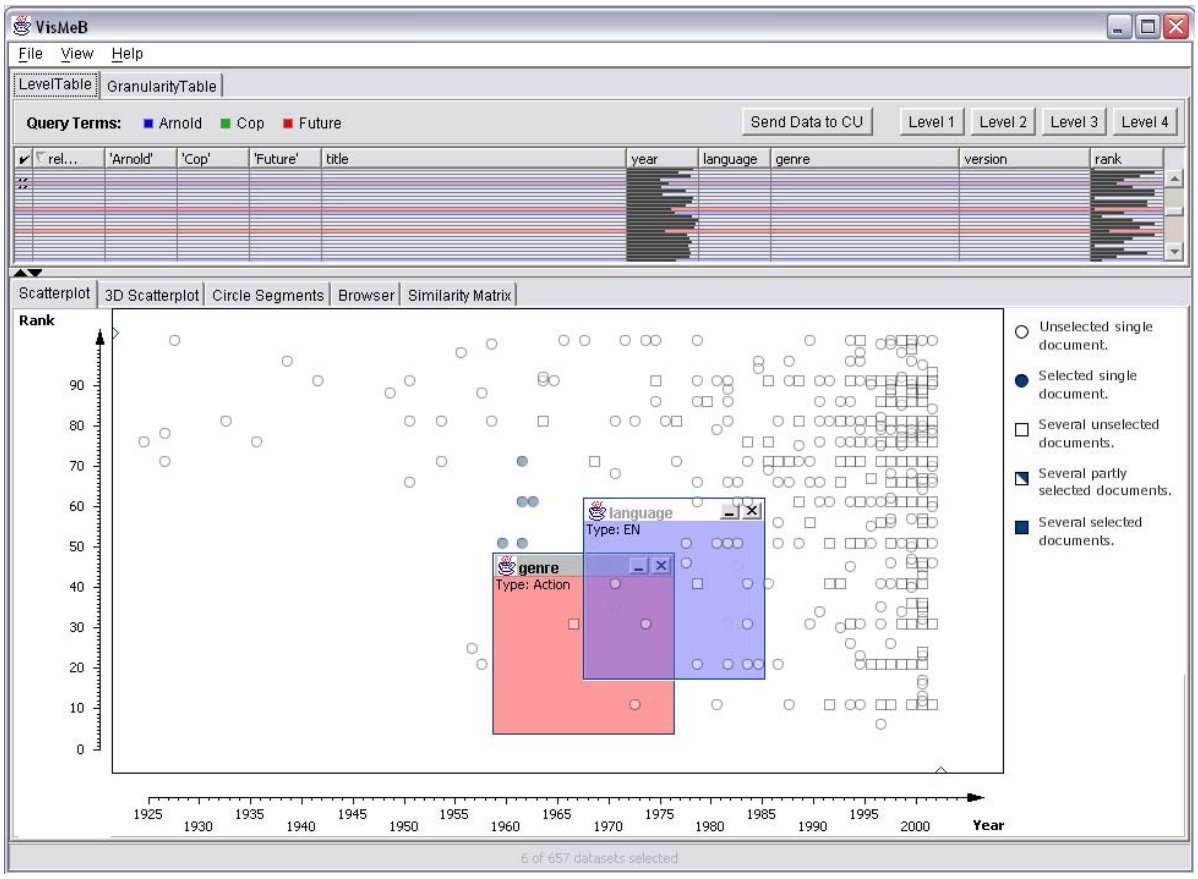


Abbildung 13: VisMeB Scatterplot mit Moveable Filter

Die Datenobjekte werden in Form kleiner Kreise dargestellt, die bei Selektion ausgefüllt gezeichnet werden. Multi-Data-Points sind durch Rechtecke dargestellt. Bei Selektion aller enthaltenen Objekte wird das Rechteck des Multi-Data-Points gefüllt gezeichnet. Sind nur einzelne enthaltene Elemente selektiert, wird das Rechteck nur halb ausgefüllt.

Entsprechend des Brushing&Linking-Konzepts werden die Datenelemente beim Überfahren mit der Maus im Scatterplot selbst und in GranularityTable oder LevelTable farblich

hervorgehoben. Um zu verdeutlichen, welcher Datenpunkt fokussiert ist, wird der jeweilige Kreis im Scatterplot leicht vergrößert. Zum fokussierten Objekt wird ein Tooltip mit näheren Informationen angezeigt.

Durch Rechtsklick der Maus im Bereich des Scatterplots kann der Benutzer ein Kontextmenü aufrufen, über das er sogenannte „*Moveable Filter*“ definieren kann (siehe auch [6]). Die „*Moveable Filter*“ werden in Form beweglicher, halbtransparenter Fenster im Scatterplot dargestellt (siehe Abbildung 13). Alle Dokumente innerhalb dieser Fenster, die den festgelegten Kriterien des Filters nicht entsprechen, werden aus der Ansicht herausgefiltert. Dokumente, die die Kriterien erfüllen, werden in LevelTable und GranularityTable der Farbe des Filters entsprechend, farblich markiert. Mehrere Filter lassen sich in AND- oder OR-Beziehungen miteinander verknüpfen.

Eine weitere Visualisierung innerhalb des VisMeB Metadaten Browsers ist der 3D-Scatterplot. Die Multi-Data-Point-View ist eine Komponente des 3D-Scatterplots. Er wird daher im folgenden Kapitel detailliert vorgestellt.

4 DER VISMEB 3D-SCATTERPLOT

Ein 3D-Scatterplot ist ein dreidimensionales kartesisches Koordinatensystem im Raum. Jede Achse des Systems visualisiert die Ausprägungen einer Variablen eines Datenbestandes.

In diesem Kapitel wird der 3D-Scatterplot des VisMeB Metadaten Browsers vorgestellt. Hierbei handelt es sich um einen interaktiven 3D-Scatterplot. Bei seiner Konzeption wurde versucht, den Problemen mit 3D Visualisierungen Sorge zu tragen. Die Interaktion wurde möglichst einfach und überschaubar gestaltet. Trotzdem sollte er nicht an Funktionalität und Aussagekraft verlieren. Nicht zu letzt wurde dies auch durch eine sinnvolle Behandlung der Multi-Data-Points erreicht.

Weitere Ausführungen bezüglich des Konzeption und Implementierung des VisMeB 3D-Scatterplots sind unter [15] zu finden.

4.1 Die Visualisierung

Der 3D-Scatterplot visualisiert die Datenelemente in einem virtuellen Raum, aufgespannt durch drei Koordinatenachsen (x-, y-, z-Achse). Jede Achse kann mit einem beliebigen Metadatenattribut belegt werden. In der linken, oberen Ecke, in dem so genannten „*Optionpanel*“, befinden sich hierfür für jede Achse Drop-Down-Menüs zur Wahl der zu visualisierenden Attribute. Die aktuelle Belegung der Achsen wird in der unteren, linken Ecke der Visualisierung angezeigt.

Die Länge der Koordinatenachsen ist fest definiert. Die Ausprägungen der Attribute verteilen sich über diese Länge. Datenpunkte werden entsprechend ihrer Ausprägungen im Raum platziert.

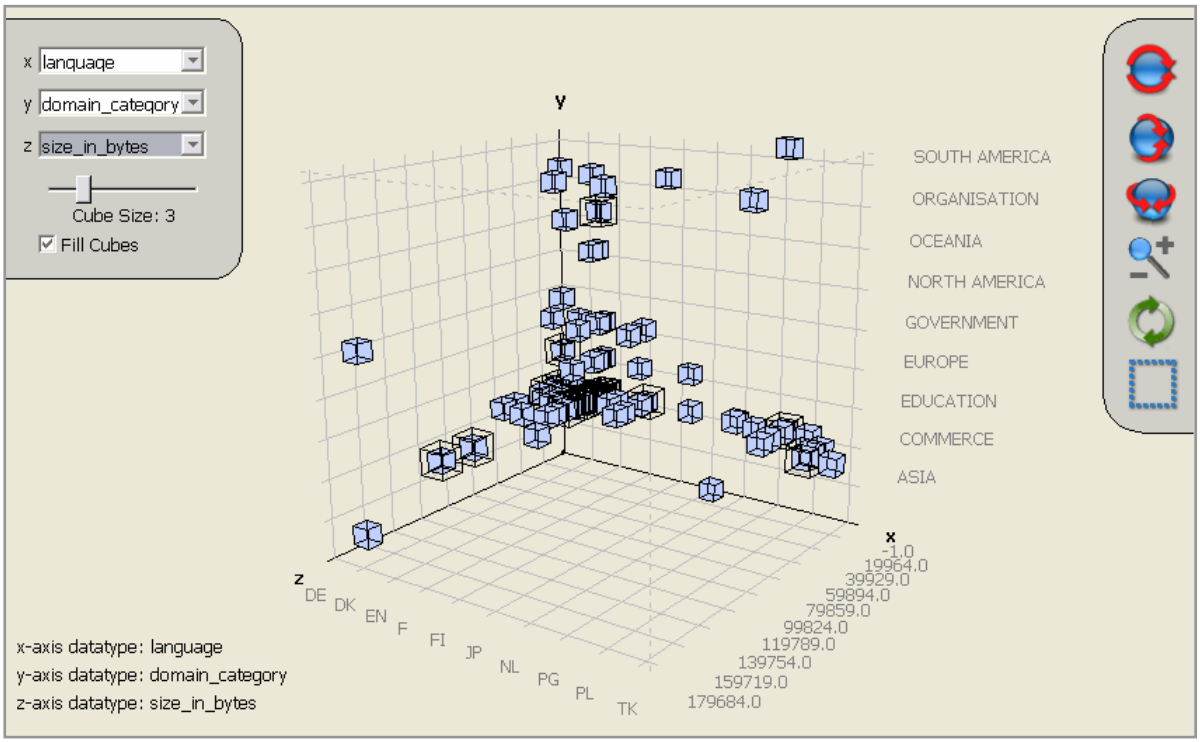


Abbildung 14: VisMeb 3D-Scatterplot Webdokumentdaten

Zum besseren Einschätzen der Positionen der Datenpunkte sind die Koordinatenebenen mit Gitterlinien belegt. Die Anzahl der Gitterlinien richtet sich nach der Anzahl der Ausprägungen des visualisierten Attributs. So werden zum Beispiel bei kategorischen Attributen mit nur fünf Ausprägungen auch nur fünf Gitterlinien gezeichnet. An den Seiten der Koordinatenebenen, an den Enden der Gitterlinien, befindet sich die Beschriftung der korrespondierenden Ausprägungen. Dadurch kann der Betrachter den Datenelementen Werte bzw. Kategorien zuordnen.

Drei gestrichelte Linien gegenüber den Koordinatenachsen komplettieren den virtuellen Raum zu einem dreidimensionalen Würfel. Sie sollen den dreidimensionalen Eindruck verstärken und dem Benutzer helfen, die Lage des Scatterplots im Raum korrekt wahrzunehmen.

Die Datenpunkte werden wahlweise (durch eine Checkbox im „*Optionpanel*“) als gefüllte oder nicht gefüllte Würfel visualisiert. Selektierte Elemente werden dunkler dargestellt. Die Größe der dargestellten Würfel ist durch einen Slider im „*Optionpanel*“ manipulierbar. Wie auch das Koordinatensystem werden die Würfel perspektivisch gezeichnet. Dadurch wird

der dreidimensionale Eindruck verstärkt und die Position der Elemente im Raum ist besser bestimmbar. Auch die korrekte, perspektivische, gegenseitige Verdeckung der Würfel verstärkt die dreidimensionale Wahrnehmung.

4.2 Die Interaktion

Die starre bildliche Darstellung im 3D-Scatterplot wäre nicht geeignet zur Exploration der Daten. Je nach Perspektive wären Datenelemente verdeckt oder Zusammenhänge zwischen den Elementen nicht erkennbar. Der Benutzer muss die Möglichkeit haben, die Perspektive ändern zu können, um verschiedene Sichten auf die Daten zu bekommen.

Der VisMeB 3D-Scatterplot besitzt mehrere Möglichkeiten der Interaktion. Zum einen die direkte Manipulation des Scatterplots, zum andern die Manipulation über die Interaktionselemente im sogenannten „*Interaktion Panel*“ auf der rechten Seite des Scatterplots.

Die direkte Manipulation erlaubt dem Betrachter, bildlich ausgedrückt, den Scatterplot anzufassen und zu drehen. Per drag´n´drop kann er die Ansicht durch horizontale Mausbewegungen nach rechts oder links und durch vertikale Mausbewegungen nach hinten oder nach vorne rotieren. Die Rotationsachsen sind hierbei nicht die Achsen des 3D-Scatterplots, sondern die horizontale bzw. vertikale Sichtachse.

Durch vertikale Bewegungen der Maus bei gedrückter mittlerer Maustaste (oder gedrückter STRG-Taste und einer beliebigen Maustaste) kann die Entfernung zum Koordinatensystem geändert werden. Bewegungen nach oben vergrößern die Ansicht, Bewegungen nach unten verkleinern sie.

Ebenso wie die anderen Visualisierungen interagiert der 3D-Scatterplot auch mit der LevelTable und der GranularityTable. Wird ein Datenwürfel mit der Maus fokussiert, ändert er seine Farbe und wird auch in den anderen Visualisierungen kenntlich gemacht. Zusätzlich wird im 3D-Scatterplot auf Seiten der Beschriftung der korrespondierende Datenwert durch einen roten Punkt gekennzeichnet (siehe Abbildung 15). Diese Hilfestellung unterstützt den

Betrachter auch beim Einordnen des Datenelements im Raum. Bei längerem Verharren auf dem Datenwürfel wird ein Tooltip angezeigt. Er enthält nähere Informationen zu dem Datenelement. Bei Klick auf den Datenwürfel wird das Objekt selektiert.

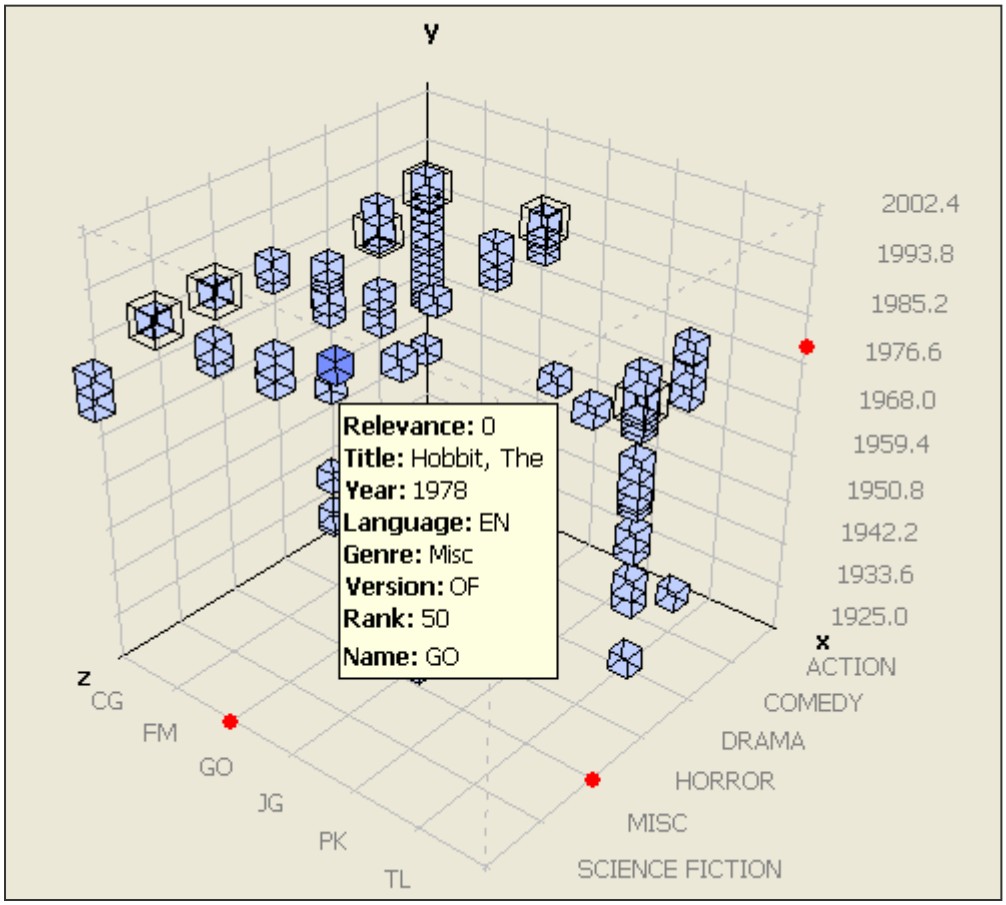


Abbildung 15: VisMeB 3D-Scatterplot Koordinatensystem mit Tooltip

Alternativ zur direkten Manipulation kann der Scatterplot über Buttons im „Interaction Panel“ rotiert werden (siehe Abbildung 16). Neben Buttons zur Rotation um die horizontale und die vertikale Achse findet der Benutzer dort einen Button zur Rotation um die Tiefenachse. Rote Pfeile in den Icons deuten die Drehrichtung an. Je nach dem, welchen Pfeil und damit welchen Teil des Icons der Benutzer drückt, rotiert das System im Uhrzeigersinn oder gegen den Uhrzeigersinn (bzw. rechtsrum oder linksrum, bzw. nach vorne oder nach hinten).

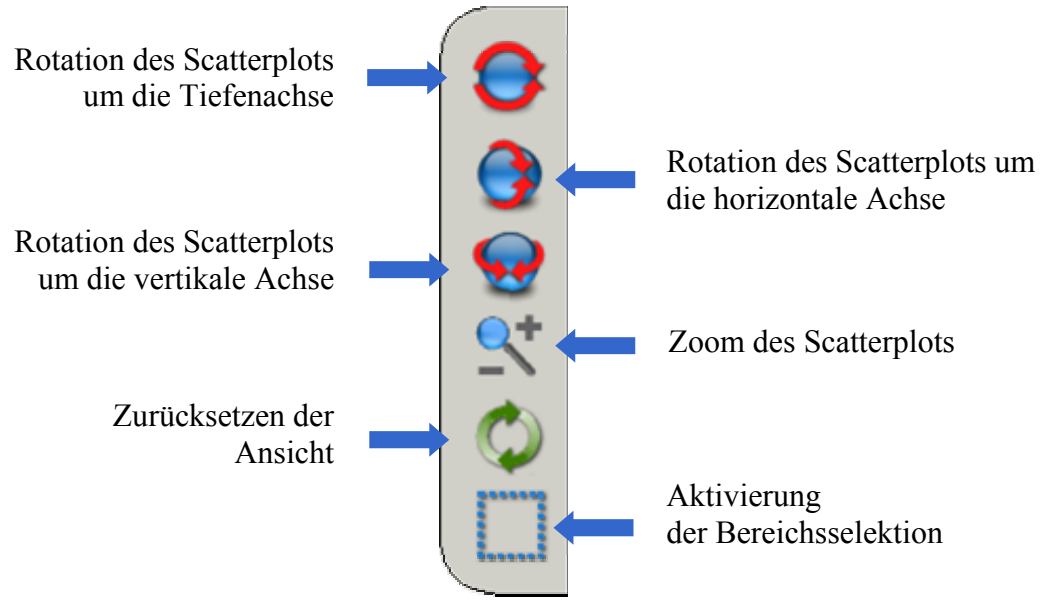


Abbildung 16: VisMeB 3D-Scatterplot Interaction Panel

Das Icon eines Vergrößerungsglases symbolisiert die Zoomfunktion. Über das Plus kann die Ansicht vergrößert, über das Minus verkleinert werden. Darunter findet sich ein Button zum Zurücksetzen der Ansicht. Der Scatterplot wird beim Klicken auf diesen Button in seine ursprüngliche Lage und Größe zurückversetzt.

Das letzte Element des Interaktion Panels ist der Button zur Aktivierung der Bereichsselektion. Ist dieser Modus aktiv, ändert sich der Mauszeiger in ein Fadenkreuz. Innerhalb des Koordinatensystems des Scatterplots lässt sich dann durch Gedrückthalten der Maustaste ein Rechteck aufziehen. Alle Punkte, die sich unabhängig von der Perspektivischen Darstellung innerhalb dieses Rechtecks befinden, werden beim Loslassen der Maus selektiert, bereits selektierte Objekte werden deselektiert.

5 MULTI-DATA-POINTS

Der VisMeB 3D-Scatterplot visualisiert Metadatenobjekte, indem er die Objekte abhängig von den Ausprägungen ihrer Attribute in ein virtuelles räumliches System einordnet. Besitzen unterschiedliche Objekte drei Attribute mit jeweils denselben Ausprägungen und werden diese Attribute im 3D-Scatterplot visualisiert, wird den Objekten im Scatterplot die gleiche Position zugeordnet, d. h. die Objekte besitzen die gleichen X-, Y-, und Z-Koordinaten im Scatterplot. In der Darstellung würden sie sich dann gegenseitig überdecken. Die Objekte bilden einen „*Mulidatenpunkt*“ (engl. Multi-Data-Point).

In diesem Kapitel werden die Problematik der Multi-Data-Points vorgestellt und bereits vorhandene Lösungsansätze diskutiert.

5.1 Die Multi-Data-Point Problematik

Definition:

Als „Multi-Data-Point“ bezeichnet man die Überlagerung mehrerer Datenpunkte in einer Visualisierung, die aufgrund gleicher Eigenschaften auf die gleiche Position im Raum abgebildet werden.

Dies gilt somit nicht nur für den 3D-Scatterplot. Jede Visualisierung, die Datenelemente in den Raum abbildet und durch die Position der Datenpunkte, relativ zu einem gemeinsamen Ausgangspunkt, die Ausprägung eines Attributs codiert, kann potentiell Multi-Data-Points enthalten. Multi-Data-Points können also sowohl in eindimensionalen als auch in zwei-, drei- und höher-dimensionalen Visualisierungen vorkommen.

Ein Beispiel:

Die nachfolgende Tabelle (Tabelle 1) enthält einige Metadaten aus der Filmdatenbank, u.a. das Produktionsland, das Produktionsjahr, oder die Länge des Films. Die Ausprägungen einzelner Attribute sind für einige Filme gleich (farblich gekennzeichnet). Zum Beispiel wurde sowohl „2001 - A Space Odyssey“, als auch „Alien“ und „Time Bandits“ in Großbritannien gedreht.

Titel	**Jahr**	**Sprache**	**Genre**	**Länge (in Minuten)**	**Produktionsland**
2001- A Space Odyssey	1968	Englisch	Si-Fi	139	UK
Alien	1979	Englisch	Horror	117	UK
Beach, The	2000	English	Drama	119	USA
Citizen Kane	1941	Englisch	Drama	119	USA
Le Fabuleux destin d'Amélie Poulain	2001	Französisch	Comedy	122	Frankreich
Memento	2000	Englisch	Action	113	USA
Time Bandits	1981	Englisch	Comedy	116	UK

Tabelle 2: Beispieldatensätze Spielfilmmetadaten

Bei der Visualisierung dieser Metadaten in einen 3D-Scatterplot käme es bei der Achsenbelegung Genre, Länge und Produktionsland zu einem Multi-Data-Point. Die Metadatenobjekte der Filme „The Beach“ und „Citizen Kane“ würden auf die gleiche Position im Raum abgebildet werden. Sie bilden einen Multi-Data-Point.

Bei der Visualisierung in einen 2D-Scatterplot würden schon zwei gleiche Attribute zum Multi-Data-Point genügen, z.B. Genre und Länge des Films. In einer Eindimensionalen Visualisierung genügt sogar ein identisches Attribut dem Multi-Data-Point. Je nach Anzahl der dargestellten Dimensionen müssen also ein, zwei oder drei Attribute übereinstimmen, um einen Multi-Data-Point zu bilden (siehe Abbildung 17).

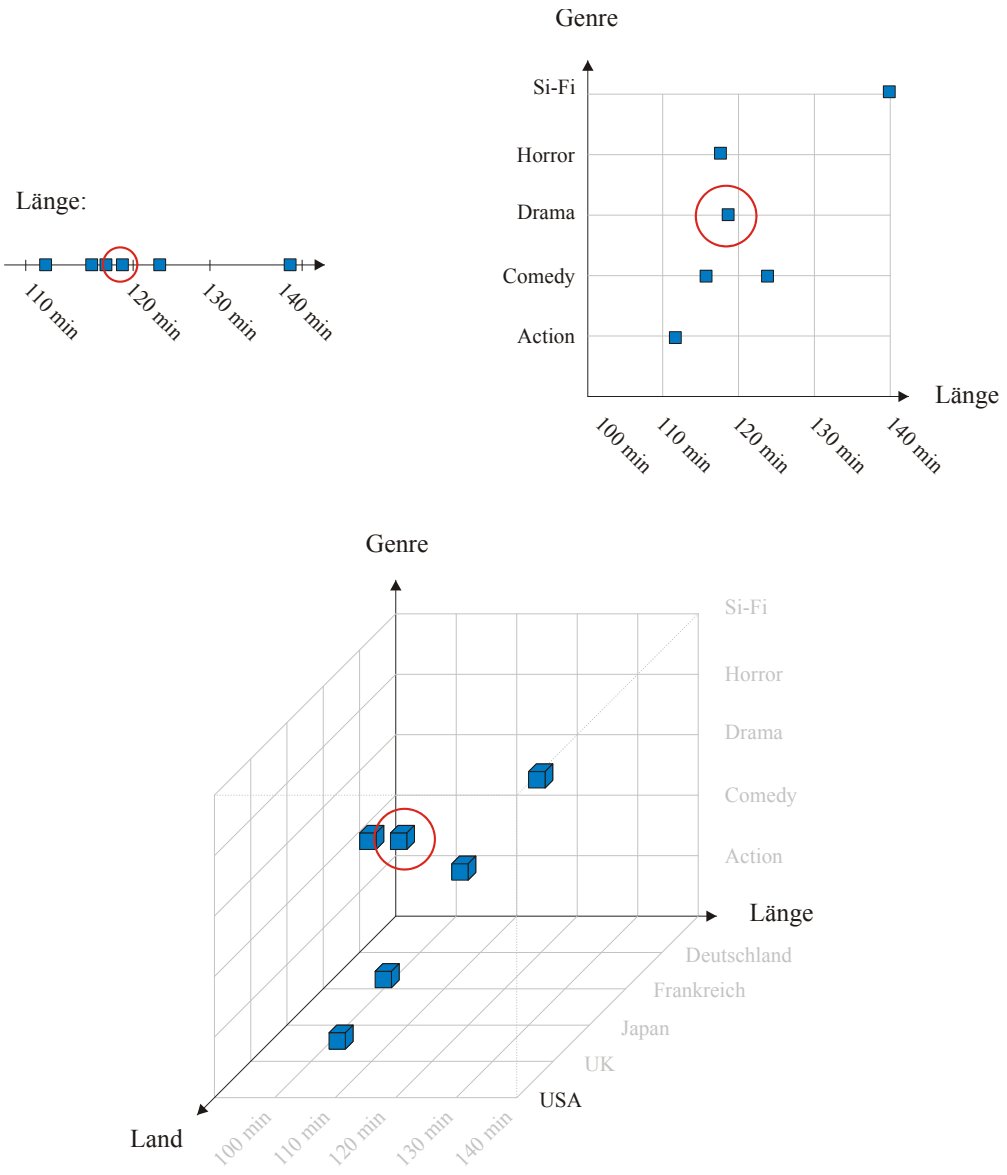


Abbildung 17: Grafische Darstellung der Beispieldaten in 1D (oben links), 2D (oben rechts) und 3D (unten)

Die Wahrscheinlichkeit eines Multi-Data-Points ist neben der Anzahl der dargestellten Dimensionen auch abhängig vom Datentyp der Attribute. Besitzt ein Attribut nur wenige mögliche Ausprägungen, z.B. „männlich“ und „weiblich“, ist es wahrscheinlich, dass verschiedene Objekte gleiche Werte besitzen. Dies gilt im Besonderen für nominale und diskret ordinale Attribute. Ein weiterer Faktor, der die Anzahl bzw. die Wahrscheinlichkeit von Multi-Data-Points beeinflusst, ist natürlich die Anzahl der insgesamt dargestellten Datensätze. Je mehr Daten visualisiert werden, umso wahrscheinlicher werden Multi-Data-Points. Gerade bei der Darstellung großer Datenmengen ist das Vorkommen von Multi-Data-Points daher hochwahrscheinlich.

Aus Sicht des Information Visualization und des Projekts VisMeB ergibt sich daher die Problematik, wie ein solcher Multi-Data-Point in einer Visualisierung dem Betrachter veranschaulicht werden kann. Obwohl die Datenelemente auf der gleichen Position liegen und sie sich gegenseitig überdecken, muss der Betrachter sie als selbständige Objekte erkennen können, ansonsten wäre die Visualisierung nicht vollständig und dadurch nicht zweckgerecht.

Natürlich kann ein Multi-Data-Point auch aus mehr als nur zwei Objekten bestehen. Gerade bei vielen Datenobjekten kommt es vor, dass ein Multi-Data-Point mehrere Hundert Einzelobjekte enthält. Dieses wirft die Frage auf, wie man Hundert Datenpunkte auf ein und derselben Position im Raum visualisiert kann.

5.2 Bestehende Lösungsansätze

Die Problematik des Multi-Data-Points ist nicht neu. Trotzdem gibt es nur wenig Veröffentlichung zu diesem Thema. Die wohl bedeutendste ist „The Elements of Graphing Data“ von W. S. Cleveland [4]. Cleveland beschreibt dort das Problem überlappender Datenpunkte bei der Visualisierung zweier quantitiver Variablen im Scatterplot. Er gibt mögliche Lösungsansätze für die Darstellung teilweise überlappender, sowie vollständig überlappender Datenpunkte und schlägt sechs Hilfsmittel vor:

- Logarithmische Darstellung
- Darstellung der Residuen
- „Moving“
- „Sunflowers“
- „Jittering“
- „Open Circles“.

Die vorgeschlagenen Methoden der logarithmischen Darstellung und der Visualisierung des Residuums dienen der Verzerrung der Ansicht. Dadurch werden teilweise überlappende Datenpunkte entzerrt und können getrennt wahrgenommen werden. Die Entzerrung hat aber keine Auswirkungen auf Multi-Data-Points. Auch das Darstellen von „Open Circles“, also

das Darstellen von nicht gefüllten Kreisen zur Repräsentation der Datenobjekten, bezweckt nur eine verbesserte Visualisierung sich teilweise überlagernden Datenobjekte.
Die verbleibenden Methoden „*Moving*“, „*Sunflowers*“ und „*Jittering*“ bieten Lösungsansätze der Multi-Data-Point Problematik. Sie sollen im Folgenden vorgestellt werden.

„Moving“

Eine Möglichkeit, die einzelnen Elemente eines Multi-Data-Points zu visualisieren ist das geringfügige Verschieben (engl. moving) der einzelnen Datenpunkte um einen festen Wert. Zum Beispiel werden alle Datenpunkte vertikal um die Durchmesser ihrer Repräsentation verschoben. Dabei ist wichtig, dass der Betrachter auf diese Verschiebung, zum Beispiel in der Legende, hingewiesen wird, damit es nicht zur Fehlinterpretationen kommt.

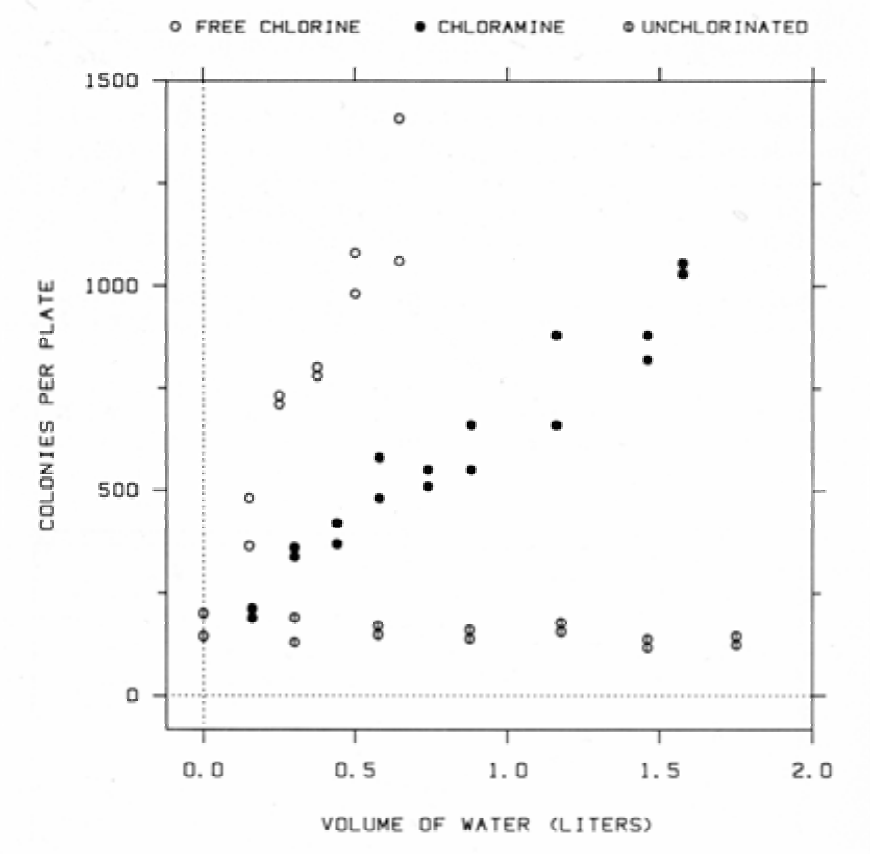


Abbildung 18: Das Moving, Konzept in einem 2D Scatterplot nach Cleveland, aus: Cleveland, W.S., *The Elements of Graphing Data*

Beispiele einer möglichen Anwendung finden sich in den Abbildungen 18 und 19. Abbildung 18 zeigt einen 2D-Scatterplot in dem die verdeckten Punkte vertikal nach oben verschoben wurden. Eine besondere Form der Anwendung des Prinzips zeigt Abbildung 19.

In einem sogenannten Dot-Plot (siehe [31] und [32]) werden Datenpunkte mit gleicher Ausprägung entweder symmetrisch oder asymmetrisch übereinander gezeichnet.

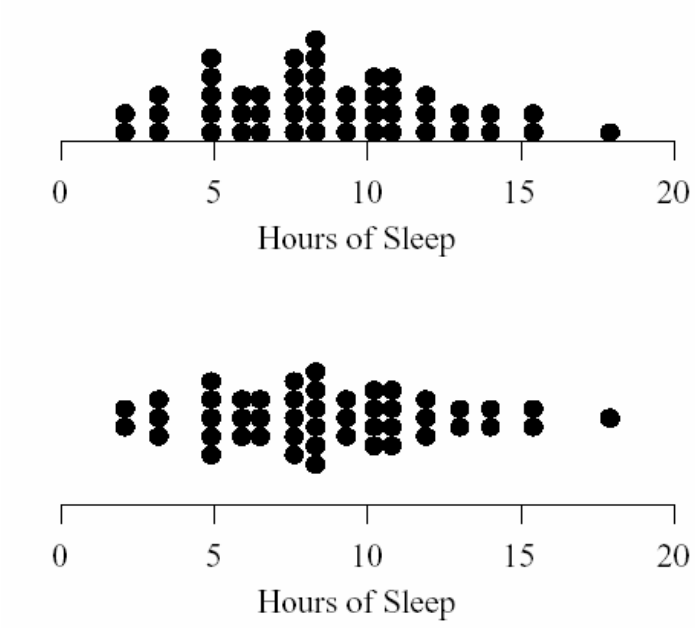


Abbildung 19: Asymmetrischer dot plot (oben), symmetrischer dot plot (unten), aus: Wilkinson, L., *Dot Plots*

Durch das Verschieben können die einzelnen Elemente erkannt werden, aber da die Dimension in der verschoben wird, meist auch ein Attribut codiert kann der Betrachter nicht immer entscheiden ob es sich um einen künstlich verschobenen Datenpunkt handelt oder nicht. Problematisch wird die „*Moving*"-Methode, wenn die Multi-Data-Points viele Objekte enthalten. Die Visualisierung wäre zu unübersichtlich und bei sehr vielen Elementen sind die Punkte aus praktischen Gründen nicht mehr darstellbar.

„Sunflowers"

„*Sunflowers*" (dt. Sonnenblumen) stellen jedes Element des Multi-Data-Points durch eine kleine Linie (Blütenblatt) um den eigentlichen Datenpunkt dar (siehe Abbildung 20 und 21). Enthält der Multi-Data-Point beispielsweise fünf Elemente wird eine kleine Sonnenblume mit fünf Blütenblättern an diese Stelle gezeichnet (✳). Die Anzahl der Blütenblätter zeigt also an, wie viele Datenpunkte der Multi-Data-Point enthält.

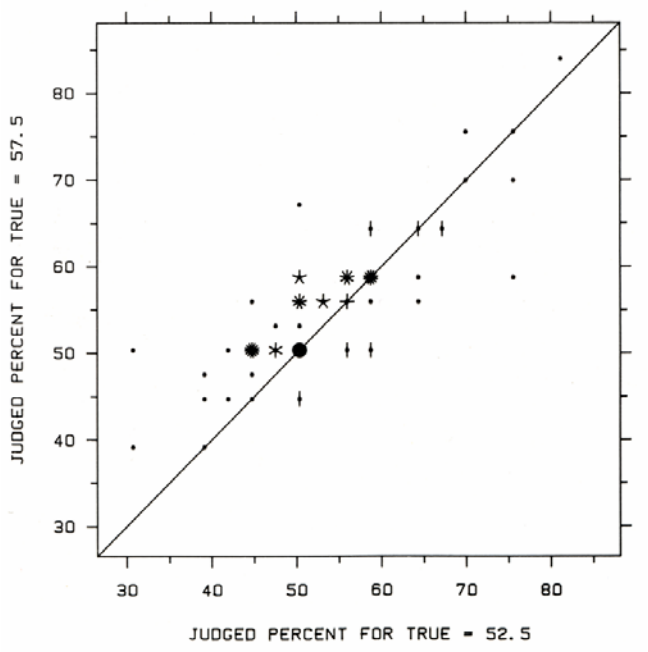


Abbildung 21: Sunflower-Darstellung nach Cleveland, aus: aus: Cleveland, W.S., *The Elements of Graphing Data*

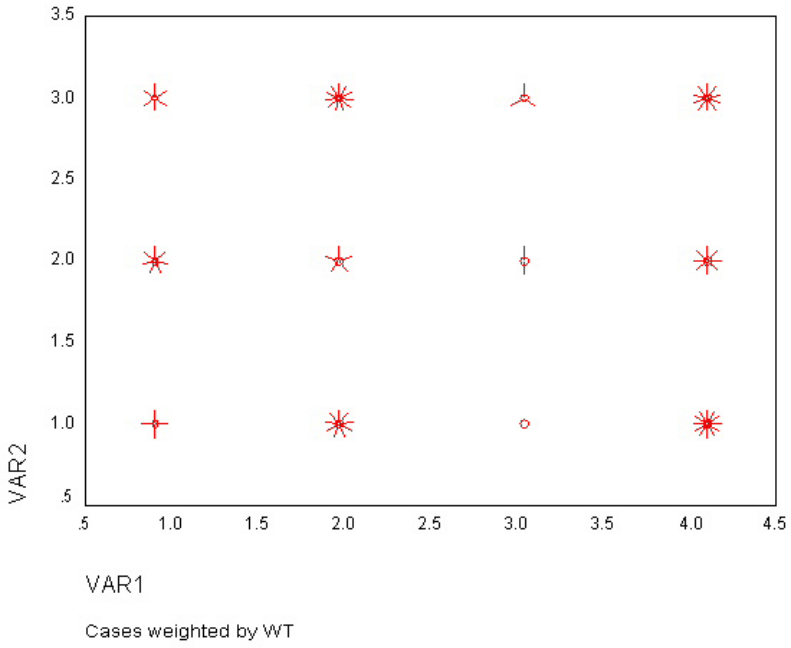


Abbildung 20:
Sunflower-Darstellung der Statistik Software SPSS

Die Idee der „*Sunflowers*“ wurde bereits in einer frühen Version des VisMeB 3D-Scatterplots implementiert. Die überlappenden Dokumente wurden zu so genannte „*Buzzing Beans*“ zusammengefasst, repräsentiert durch zwei ineinander geschachtelte Würfel. Beim Überfahren dieser mit dem Mauszeiger fächerten sich die enthaltenen Dokumente, symbolisiert durch kleine Würfel, kreisförmig auf. Aber schon bei mehr als 6 Objekten konnten die einzelnen Elemente in diese Form der Visualisierung nicht erkannt werden (siehe Abbildung 22).

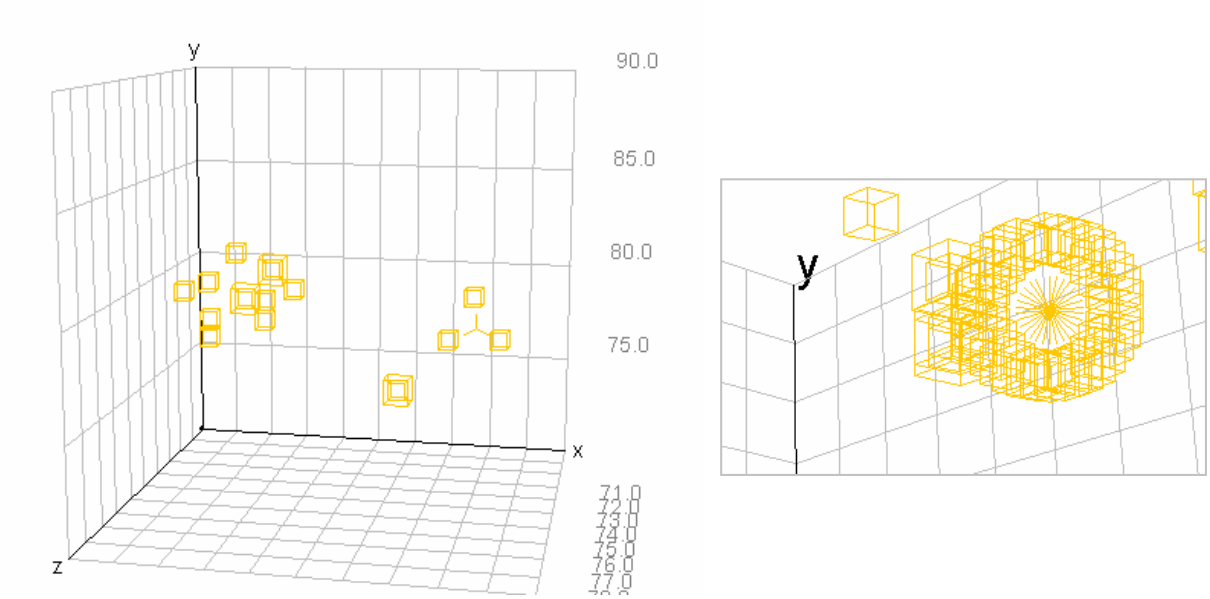


Abbildung 22: INVISIP buzzing beans, rechts: geöffnet mit 26 Elementen

„Jittering“

Ein weiterer Lösungsansatz ist das „Jittering“. Hierbei werden zu den Koordinatenwerten der sich überlappenden Punkte kleine, zufällige Fehler addiert bevor die Datenpunkte visualisiert werden. Dadurch verteilen sich die Elemente des Multi-Data-Point um ihren eigentlichen Wert und sind unterscheidbar (siehe Abbildung 23 und 24).

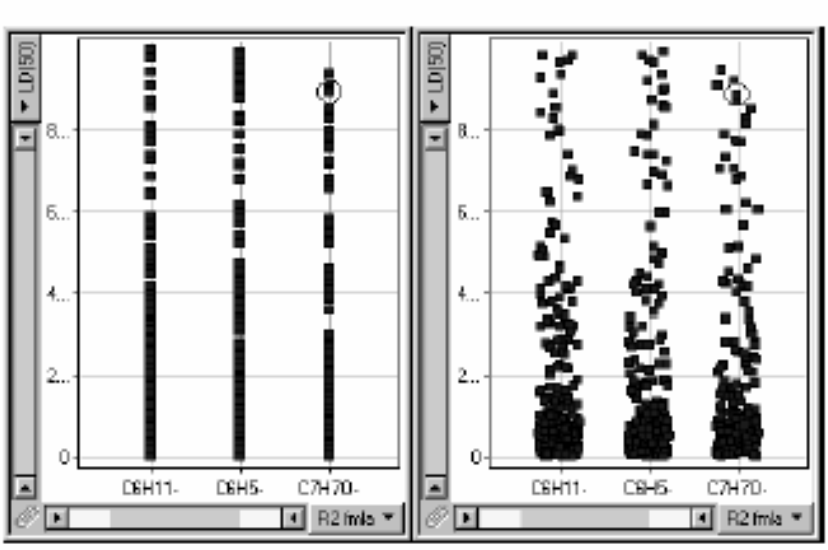


Abbildung 23: Jittering-Darstellung der Visualisierungssoftware Spotfire-Decision Site, aus: Spotfire Decision Site 7.1 - User’s Guide and Reference Manual White Paper

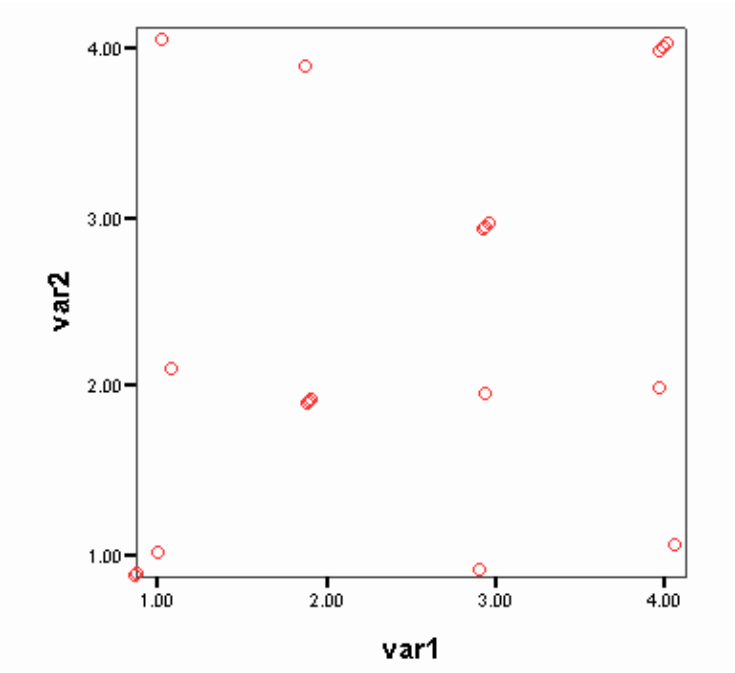


Abbildung 24: Jittering -Darstellung der Statistik Software SPSS

Das Problem bei dieser Methode ist, dass die Position aller Elemente der Multi-Data-Points manipuliert wird und der Betrachter keine Möglichkeit hat, den eigentlichen Wert abzulesen. Außerdem ist auch diese Methode nur bei kleinen Datenmengen sinnvoll, da es ansonsten nach der Anwendung zu weiteren Überdeckungen kommen kann.

Für alle von Cleveland vorgestellten Ansätze hat sich gezeigt, dass sie bei Multi-Data-Points mit vielen Elementen versagen. Das Problem, viele Datenpunkte auf stark begrenztem Raum darzustellen, scheint unlösbar. Aber nicht immer ist das Visualisieren der in den Multi-Data-Points enthaltenen Elemente, notwendig. Stellt z. B. ein Scatterplot zwei Attribute dar, um dem Betrachter nur einen Überblick über Korrelation der beiden Variablen zu geben, wäre das Fehlen einzelner Datenpunkte nur bedeutend, wenn dadurch der Gesamteindruck verfälscht werden würde. Die einzelnen Datenpunkte eines Multi-Data-Points müssen in einer solchen Visualisierung nicht explizit dargestellt werden. Es genügt dem Betrachter, die Größe im Multi-Data-Point anzuzeigen, um diesen im Zusammenhang richtig gewichten zu können. Die Art der Repräsentation der Datenpunkte ist also auch stark abhängig vom Zweck der Visualisierung.

Ein Beispiel für eine Visualisierung, in der nur die Anzahl der Elemente im Multi-Data-Point dargestellt wird, ist der „Frequency Scatterplot“, der in einem elektronischen Statistik-Lehrbuch der Firma Statsoft vorgestellt wird (siehe [27]). Dort codieren unterschiedlich große Kreise die Anzahl der enthaltenen Elemente (siehe Abbildung 25).

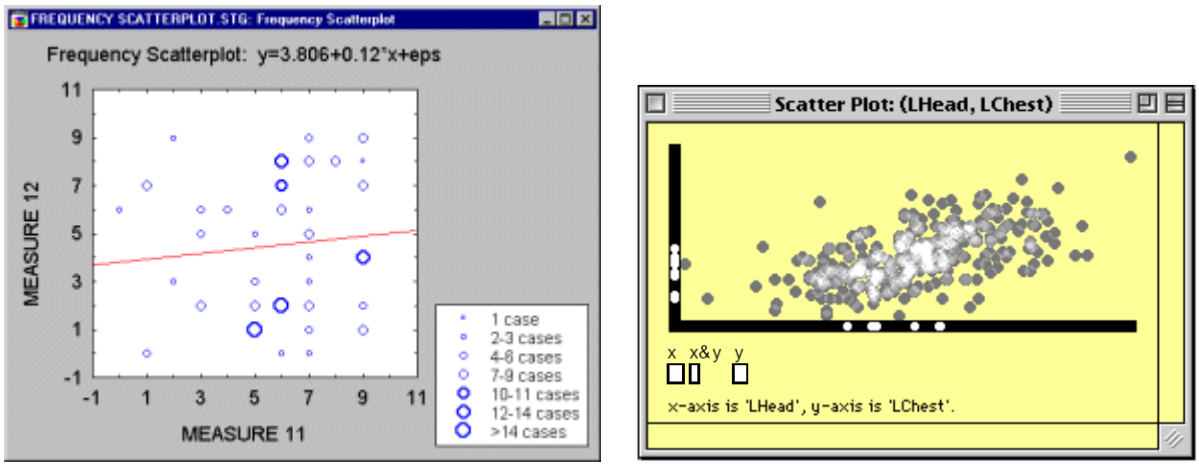


Abbildung 25: Frequency Scatterplot (links) und MANET Scatterplot (rechts)

Eine weitere Möglichkeit die Anzahl der sich überlappenden Punkte dem Betrachter zu veranschaulichen ist, durch ein „vierte“ Dimension, zum Beispiel durch die Helligkeit der Datenpunkte, die Dichte der Datenobjekte zu visualisieren. Heike Hofmanns Software MANET (siehe auch [10] und [30]) setzt dieses Konzept um. Die Helligkeit der dargestellten Datenpunkte steigt linear mit der Anzahl der sich überlappenden Punkte. Zwar werden dabei auch sich teilweise überlappende Punkte mit einbezogen, aber der Benutzer bekommt einen guten Überblick, wie sich die Punkte im Datenraum verteilen.

Aber nicht immer genügt der gute Überblick. Wann immer der einzelne Datenpunkt für den Erfolg der Visualisierung wichtig ist, muss er auch sichtbar sein. In einem Suchsystem ist jeder Datenpunkt ein möglicher Treffer. Ist dort ein Datenpunkt nicht sichtbar, da er durch einen Multi-Data-Point verdeckt, ist dies nicht akzeptabel. Jedes Element und dessen Detailinformationen muss dargestellt werden, damit der Benutzer entscheiden kann wie relevant das Objekt ist. Für den 3D-Scatterplot im Projekt VisMeB musste daher eine Form der Visualisierung der Multi-Data-Points gefunden werden, die dem Benutzer erlaubt die Multi-Data-Points zu explorieren und sich zu den im Multi-Data-Point enthaltenen Objekten, Detailinformationen anzeigen zu lassen,

6 MULTI-DATA-POINT-VIEW IN VISMEB

Der VisMeB 3D-Scatterplot visualisiert drei Dimensionen eines Metadatenobjekts und stellt diese in Relation. Dadurch kann der Benutzer Zusammenhänge zwischen den dargestellten Attributen der Datenpunkte erkennen. Um die Relevanz eines Datenobjekts bewerten zu können, genügen die drei dargestellten Dimensionen aber oftmals nicht. Der Benutzer muss sich Detailinformationen zu dem Objekt anzeigen lassen. Problematisch wird dies, wenn das Datenelement in einem Multi-Data-Point versteckt ist.

Durch Multi-Data-Point-Visualisierung sollen Multi-Data-Points dem Benutzer zugänglich gemacht werden. Alle Elemente eines Multi-Data-Points sollen inklusive Detailinformationen angezeigt werden können, so dass der Benutzer auch die Relevanz dieser Objekte bewerten kann.

6.1 Projektdefinition

Im Zuge einer Neukonzeption und Neuimplementierung des 3D-Scatterplots sollte auch eine neue Methode zur Visualisierung der Multi-Data-Points gefunden werden. Zwar gab es in der vorhandenen Version bereits eine Visualisierung der Multi-Data-Points (die sog. „*Buzzing Beans*“), da diese aber nicht den Ansprüchen eines Suchsystems genügte, sollte ein neues Konzept erdacht werden. Vorrangigstes Ziel der Darstellung sollte sein, dass sowohl alle im Multi-Data-Point enthaltenen Elemente grafisch dargestellt werden, als auch, dass zu jedem Objekt Detailinformationen angezeigt werden können.

Die Elemente des Multi-Data-Points müssen wie normale Datenobjekte identifizierbar, fokussierbar und selektierbar sein. Die Ansicht sollte mit der LevelTable und der GranularityTable synchronisiert werden, d.h. die Elemente der Multi-Data-Points sollten bei

Selektion in der LevelTable und in der GranularityTable farblich markiert werden oder bei Anwendung eines globalen Filters aus der Ansicht entfernt werden.

Eine weitere Vorgabe an die Visualisierung war, dass die im Multi-Data-Point enthaltenen Elemente animiert dargestellt werden. In Analogie zu *„Ease of Use“* soll durch die dynamische Darstellung der Elemente der *„Joy of Use“* des Benutzers gesteigert werden. Dies soll einen positiven Einfluss auf Qualität der Arbeit mit der Software haben (höhere Motivation, gesteigerte Arbeitszufriedenheit, höhere Akzeptanz der Software).

Die Ansicht muss Funktionen zum Explorieren der einzelnen Elemente, zur Anzeige von Detailinformationen und zur Selektion der Objekte bieten. Die Funktionalitäten sollten selbstbeschreibend und leicht erlernbar sein. Die Ansicht sollte sich stets erwartungskonform verhalten. Als Standard Eingabegerät sollte die Maus dienen, allerdings sollte die Ansicht auch durch die Tastatur bedienbar sein.

Die Darstellung sollte für die Verwendung in einer Büroumgebung angemessen sein. Sie sollte des Weiteren übersichtlich, leicht verständlich und benutzerfreundlich sein. Der Benutzer soll die Intention der Darstellung verstehen. Ebenso wie die anderen Komponenten des Systems sollte die Visualisierung dem „SUN Look& Feel“-Standard entsprechen.

Die Visualisierung wird in das VisMeB System eingegliedert und muss auf den vorhandenen Strukturen des Datenmodells arbeiten sowie deren Objekte und der Schnittstellen nützen. Sinnvoll war daher die Visualisierung auch in JAVA zu implementieren, wobei das System mindestens mit Version 1.3.1 der JAVA Virtual Machine (JVM) lauffähig sein musste. Die Plattformunabhängigkeit des Systems sollte gewahrt werden. Der Anspruch an die Hardware durfte, mit Berücksichtigung der anderen Komponenten des VisMeB-Systems, nicht die eines gängigen Personal Computers überschreiten.

6.2 Konzeption

Mit dem Projekt betraut wurden Philipp Liebrenz[6] und Werner König[7], die auch für das Redesign und die Neuimplementierung des 3D-Scatterplots verantwortlich waren. Durch das notwendige Redesign des 3D-Scatterplots und Verwandtheit der Visualisierungen konnte ein strukturiertes Konzept zur sinnvollen Eingliederung der Multi-Data-Point Visualisierung in den 3D-Scatterplot erstellt werden.
Wie bereits erwähnt, existierte in der vorhandenen Version des 3D-Scatterplots bereits eine Multi-Data-Point-Behandlung. Am Beginn des Projekts stand nun die Überlegung, in wie weit die vorhandene Multi-Data-Point-Visualisierung übernommen, bzw. angepasst werden kann.

6.2.1 Ausgangslage

Die vorhandene Multi-Data-Point-Darstellung lehnte sich an das „*Sunflower*"-Konzept an. Die überlappenden Dokumente wurden in so genannten „*Buzzing Beans*" zusammengefasst. Im Gegensatz zu normalen Datenobjekten, die durch ein einfaches Würfeldrahtgestell symbolisiert wurden, visualisiert man die „*Buzzing Bean*" als zwei ineinander geschachtelte Würfel im Koordinatensystem des 3D-Scatterplots. Beim Überfahren der „*Buzzing Beans*" mit dem Mauszeiger fächerten sich die enthaltenen Objekte, symbolisiert durch kleine Würfel, kreisförmig auf. Jeder der kleinen Würfel repräsentierte ein Element des Multi-Data-Points.

Außer dem Auffächern existierten keine weiteren Interaktionsmöglichkeiten mit den Datenelementen. Weder zur „*Buzzing Bean*" selbst, noch zu den enthaltenen Elementen konnte man sich nähere Informationen anzeigen lassen. Die Objekte in dem Multi-Data-Point ließen sich nicht identifizieren, fokussieren oder selektieren. Dem Benutzer wurde lediglich, durch die Anzahl der sich auffächernden Würfel, gezeigt, wie viele Datenelemente

[6] siehe auch: Philipp Liebrenz, State-of-the-Art 3D Visualization on Scatterplots, Seminararbeit, Universität Konstanz, 2003
[7] siehe auch König, W., Konzeption und Implementation eines 3D-Scatterplots zur Visualisierung von Metadaten, Bachelorarbeit, Universität Konstanz, 2003

ungefähr in der „*Buzzing Bean*“ enthalten sind. Aber schon bei mehr als 6 Objekten war das Erkennen und Abzählen der Würfel nur noch schwer möglich. Bei großen Datenmengen und Multi-Data-Point mit mehreren hundert Datenobjekten waren die einzelnen Würfel nicht mehr erkennbar. Die Visualisierung in Form von „*Buzzing Beans*“ brachte dann keinen Mehrwert mehr, sondern machte die Visualisierung unübersichtlich.

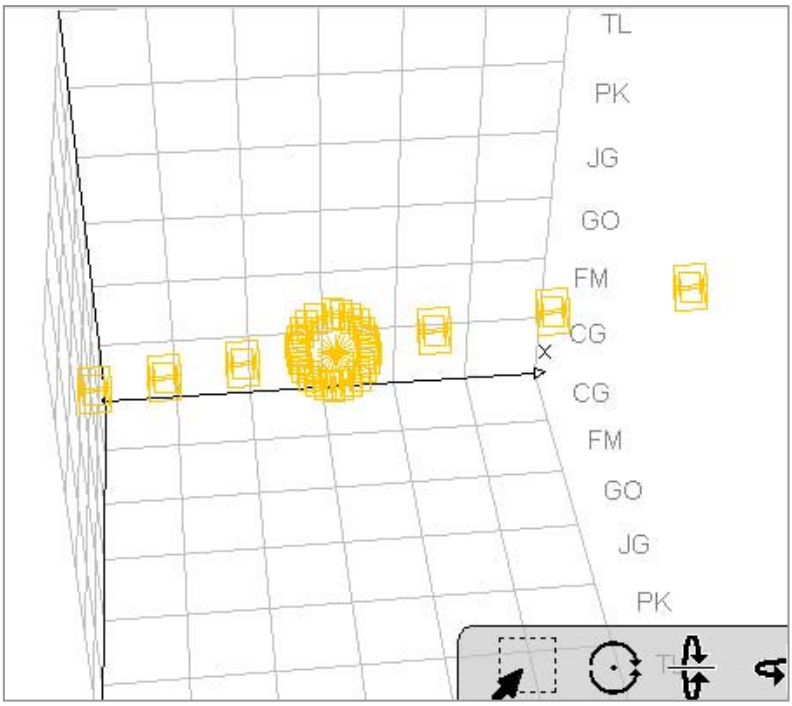


Abbildung 26: Aufgefächerte Buzzing Bean

Auch konnten die Datenpunkte nicht selektiert werden, um interessante Objekte zu kennzeichnen. Es existierte keine Synchronisation mit der LevelTable oder der GranularityTable.

Durch einen Fehler in der Implementierung lasteten aufgefächerte „*Buzzing Beans*“ den Prozessor („Pentium IV, 2,4 GHZ“) komplett aus und ließen sich teilweise nicht mehr schließen. Das System wurde überlastet und musste gegebenenfalls beendet werden. Auch traten Fehler beim Zeichnen der Würfelelemente auf. Kanten wurden nicht dargestellt oder komplette Würfel fehlten.

Des Weiteren wurden die Würfel, die die Datenelemente symbolisierten, im Gegensatz zum Koordinatensystem, das zentral-perspektivisch dargestellt wurde, parallel-perspektivisch

gezeichnet. Dies schwächte die dreidimensionale Wirkung der Darstellung ab und verwirrte den Betrachter.

Zwar boten die „*Buzzing Beans*“ eine interessante Darstellung und veranschaulichten gut, dass es sich um Datenobjekte mit denselben Ausprägungen handelt, aber die Tatsache, dass sich Multi-Data-Points mit viele Elementen in dieser Form nicht mehr darstellen ließen, führten dazu, das Konzept der „*Buzzing Beans*“ komplett fallen zu lassen und einen neuen Ansatz zu suchen.

6.2.2 Konzept

Innerhalb der 3D-Scatterplot-Ansicht stand nur wenig Raum zur Visualisierung der Multi-Data-Points zur Verfügung. Multi-Data-Points mit mehreren hundert Elementen ließen sich dort praktisch nicht sinnvoll darstellen. Auch würde eine integrierte Darstellung das Erscheinungsbild weiter komplizieren. Man entschied sich daher, die Multi-Data-Point-Visualisierung auszugliedern und die Multi-Data-Points in einer neuen Ansicht explorierbar zu machen. Der Multi-Data-Point als solcher sollte im 3D-Scatterplot gekennzeichnet werden. Wünscht der Benutzer nähere Informationen zu einem Multi-Data-Point, klickt er auf diesen und an Stelle des 3D-Scatterplots rückt die neue Ansicht, in der die einzelnen Elemente des Multi-Data-Points explorierbar sind. Optimal sollte der Benutzer schon im 3D-Scatterplot erkennen können, wieviele Elemente in dem Multi-Data-Point enthalten sind. Ein schnelles Wechseln zwischen 3D-Scatterplot und der detaillierten Multi-Data-Point-Ansicht sollte möglich sein.

Man nannte die neue Visualisierung Multi-Data-Point-View (MDPView). Sie sollte im Grunde aus drei Elementen bestehen (siehe auch Designentwurf, Abbildung 27):

1. Übersicht.
   In einer verkleinerten Darstellung des 3D-Scatterplots sollte die Position des aktuell dargestellten Multi-Data-Points erkennbar sein. Der Benutzer sollte dadurch stets den Überblick bewahren und wissen, um welchen Multi-Data-Point im Datenraum des

3D-Scatterplots es sich bei der aktuellen Visualisierung handelt, ganz im Sinne des „*Overview&Detail*“ Konzepts.

2. Darstellung der enthaltenen Datenpunkte.
   Alle im Multi-Data-Point enthaltenen Elemente sollten als Datenobjekt dargestellt werden. Jedes sollte durch eine repräsentative Bezeichnung identifizierbar sein. Außerdem sollte man jedes Objekt fokussieren und selektieren können.

3. Detailinformationen.
   Zu dem jeweils fokussierten Datenelement sollen möglichst alle verfügbaren Detailinformationen angezeigt werden, um den Benutzer zu ermöglichen die Relevanz des Objekts zu bewerten.

Der Anspruch an die Multi-Data-Point-Visualisierung war, dass die Elemente des Multi-Data-Points effizient und effektiv gesichtet werden können, auch wenn der Multi-Data-Point sehr viele Elemente enthält. Es mussten also sowohl viele als auch wenige Elemente sinnvoll darstellbar sein. Gleichzeitig gab es die Vorgabe, dass die Objekte animiert dargestellt werden sollten, um den „*Joy of Use*“ zu steigern.

In Anlehnung an das „*Sunflower*“ Konzept und die „*Buzzing Beans*“ entschied man sich, die Datenelemente des Multi-Data-Points auf einen Kreis um den symbolischen Multi-Data-Point anzuordnen. Nach dem Vorbild der „*Rapid Serial Visual Presentation*“ nach Spence und de Bruijn (siehe auch [26]) sollte sich dieser Kreis der Elemente rotieren und ein Objekt nach dem anderen in den Fokus des Betrachters gerückt werden. Dadurch wäre ein effizientes Durchblättern der Elemente möglich. Die Problematik, dass bei der Visualisierung großer Multi-Data-Points die einzelnen Elemente in den „*Buzzing Beans*“ nicht mehr erkennbar waren, sollte gelöst werden, indem man die Objekte im oberen Teil des Kreises verdichtet und im unteren Bereich nur sehr wenig Objekte darstellt. In einem rotierbaren System könnte dann jedes Objekt nach unten geführt werden und wäre dort sichtbar und fokussierbar.

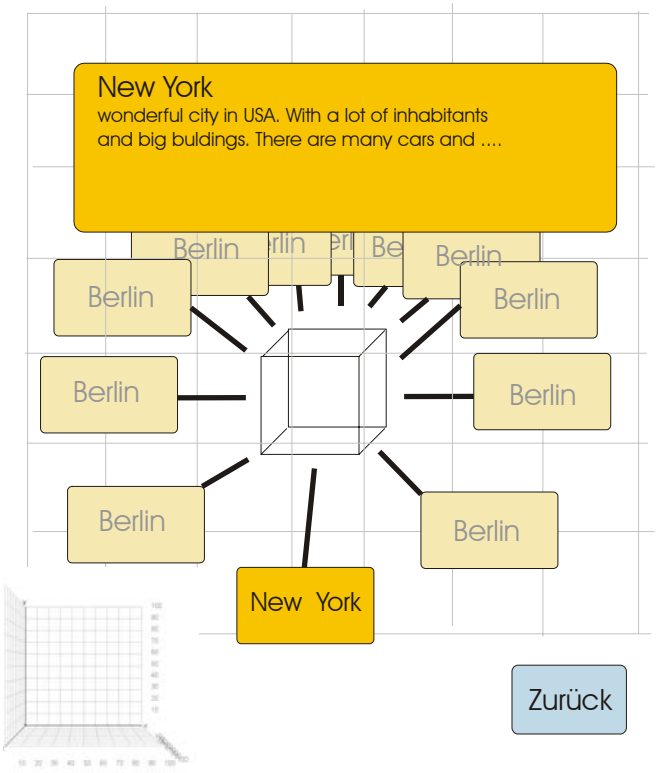


Abbildung 27: Designentwurf Multi-Data-Point View

Ein erster Designentwurf entstand (siehe Abbildung 27). Die aktuellen Detailinformationen wurden ein Feld im oberen Bereich angeordnet. Unten links befand sich die verkleinerte Darstellung des Scatterplots und im Zentrum der Darstellung, die im Kreis angeordneten Elemente des Multi-Data-Points. Die Datenelemente wurden durch rechteckige Felder repräsentiert, die in ihrer Mitte die Bezeichnung des dargestellten Objekts trugen. Im unteren Bereich des Kreises wurden nur drei Elemente angeordnet, der Rest verteilte sich im oberen Bereich. Das unterste Objekt war immer das Objekt im Fokus des Betrachters. Zu ihm wurden im Detailbereich nähere Informationen angezeigt. Würde der Kreis der Elemente rotiert, was in der späteren Ansicht animiert dargestellt werden sollte, rückt das nächste Objekt in den Vordergrund und die Informationen im Detailbereich würden wechseln.

6.3 Die Multi-Data-Point-Visualisierung

6.3.1 Visualisierung und Interaktion innerhalb des 3D-Scatterplots

Die Multi-Data-Point-Visualisierung ist in vielerlei Hinsicht eng verbunden mit dem 3D-Scatterplot, so auch in der Visualisierung. Die Multi-Data-Point-View kann nicht völlig losgelöst vom 3D-Scatterplot existieren. Teilaspekte der Multi-Data-Point-Visualisierung finden schon im 3D-Scatterplot statt. So muss dem Betrachter schon dort verdeutlicht werden, dass es sich bei diesem Datenpunkt um einen Multi-Data-Point handelt.

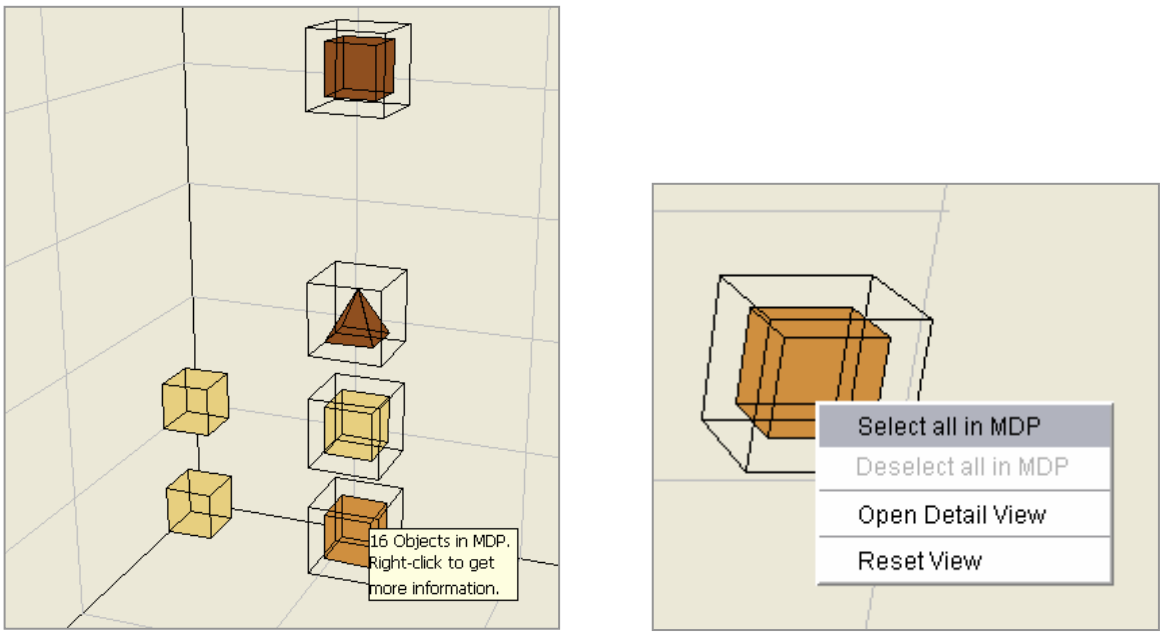


Abbildung 28: Multidatenpunkte im 3D Scatterplot (links) und Kontextmenü (rechts)

Im 3D-Scatterplot werden die Multi-Data-Points durch zwei ineinander geschachtelte, transparente Würfel bzw. durch eine Pyramide in einem Würfel symbolisiert. Dadurch kann der Benutzer einen Multi-Data-Point von einfachen Datenobjekten, die durch einfache gefüllte Würfel dargestellt sind, unterscheiden. Beim Überfahren des Würfels wird der Multi-Data-Point fokussiert und leicht dunkler dargestellt. Nach wenigen Sekunden erscheint zum fokussierten Multi-Data-Point ein Tooltip, der anzeigt, wieviele Elemente im Multi-Data-Point enthalten sind. Sind alle Datenelemente des Multi-Data-Points selektiert, wird der Multi-Data-Point in der deutlich dunkleren Selektionsfarbe dargestellt. Sind nur

einzelne Elemente des Multi-Data-Points selektiert, wird statt des inneren Würfels eine Pyramide in der Selektionsfarbe dargestellt.

Bei Rechtsklick auf einen Multi-Data-Point erscheint ein Interaktionsmenü zum aktuell fokussierten Multi-Data-Point. Hier kann der Benutzer alle Elemente, die im Multi-Data-Point enthalten sind mit einem Klick selektieren, deselektieren oder in die detaillierte Ansicht des Multi-Data-Points, also in die Multi-Data-Point-View wechseln.

Die zweite und wohl üblichere Möglichkeit, die Detailansicht aufzurufen, ist der einfache Klick auf einen Multi-Data-Point. Anstelle des 3D-Scatterplots wird dann die Multi-Data-Point-View angezeigt.

6.3.2 Visualisierung in der Multi-Data-Point-View

Entsprechend den Vorüberlegungen besteht die Multi-Data-Point-View aus drei Bereichen (siehe Abbildung 29): Dem verkleinerten 3D-Scatterplot zum Überblick dem rotierenden Kreis der Einzelelemente und dem Detailbereich, in dem die Detailinformationen zum fokussierten Datenelement angezeigt werden. Da die in VisMeB integrierte Ansicht der Multi-Data-Point-View, eher breit als hoch sein sollte, entschied man sich gegen das erste Konzept und ordnete die Bereiche nebeneinander statt übereinander an. Die Detailinformationen werden nun links des Elementenkreises und die 3D-Scatterplot Übersicht rechts davon dargestellt.

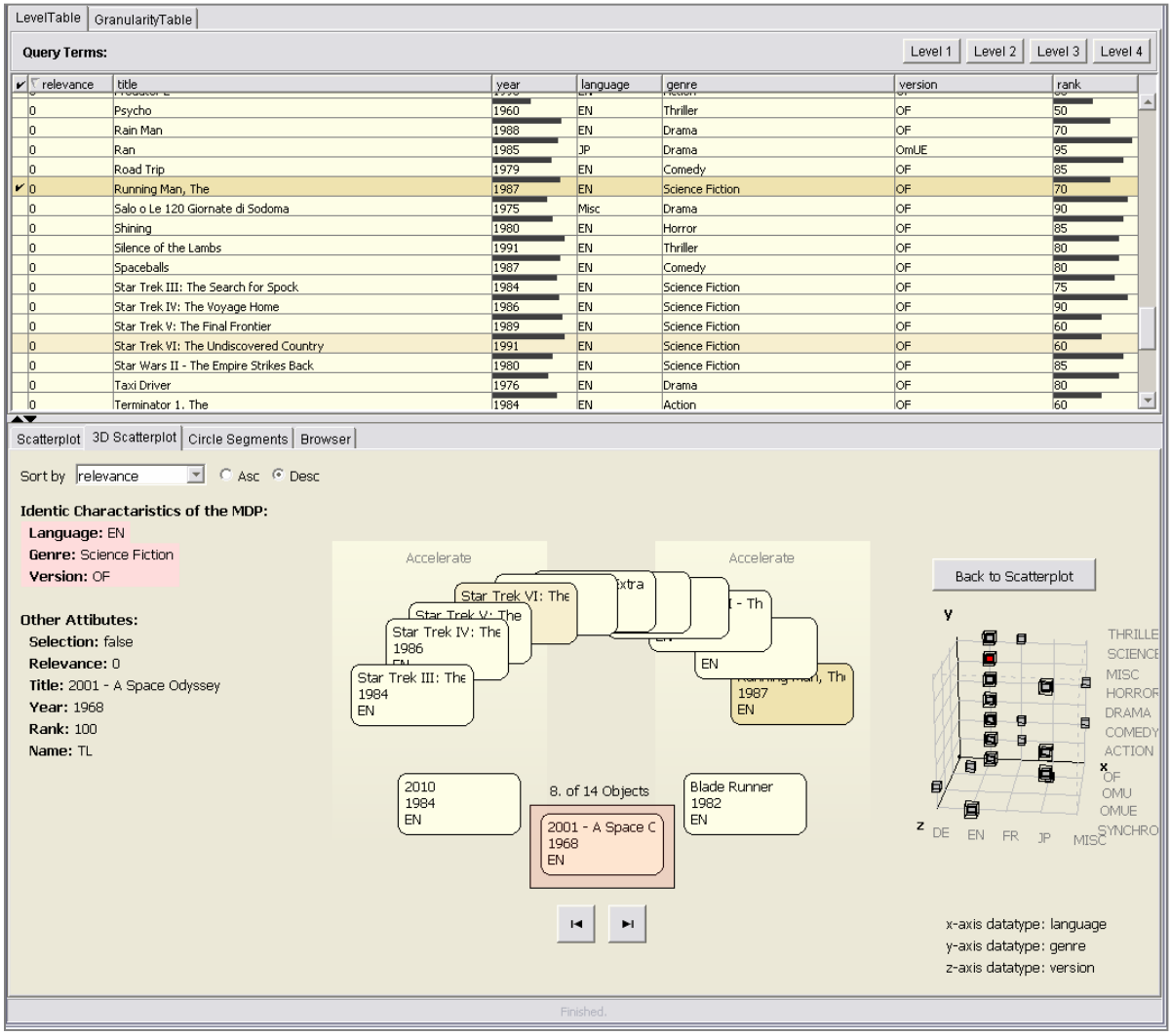


Abbildung 29: VisMeB Multi-Data-Point-View

Der verkleinerte Scatterplot soll dem Betrachter dazu dienen, nicht die Orientierung zu verlieren. Der Betrachter weiß somit stets, wo im Scatterplot und in welcher Relation zu den restlichen Daten sich der aktuell dargestellte Multi-Data-Point befindet. Hierfür ist der aktuelle Multi-Data-Point in der verkleinerten Darstellung rot gekennzeichnet. Ebenso wie der original Scatterplot, kann auch die Scatterplot-Übersicht per drag´n´drop rotiert werden. Unter dem Scatterplot wird die aktuelle Achsenbelegung des Koordinatensystems angezeigt. Oberhalb befindet sich der „Back to Scatterplot“ Button, durch den der Benutzer wieder zurück zur eigentlichen 3D-Scatterplot Visualisierung kommt. Alternativ gelangt der Benutzer auch durch einen einfachen Klick auf verkleinerte Darstellung zurück in die Scatterplot Visualisierung.

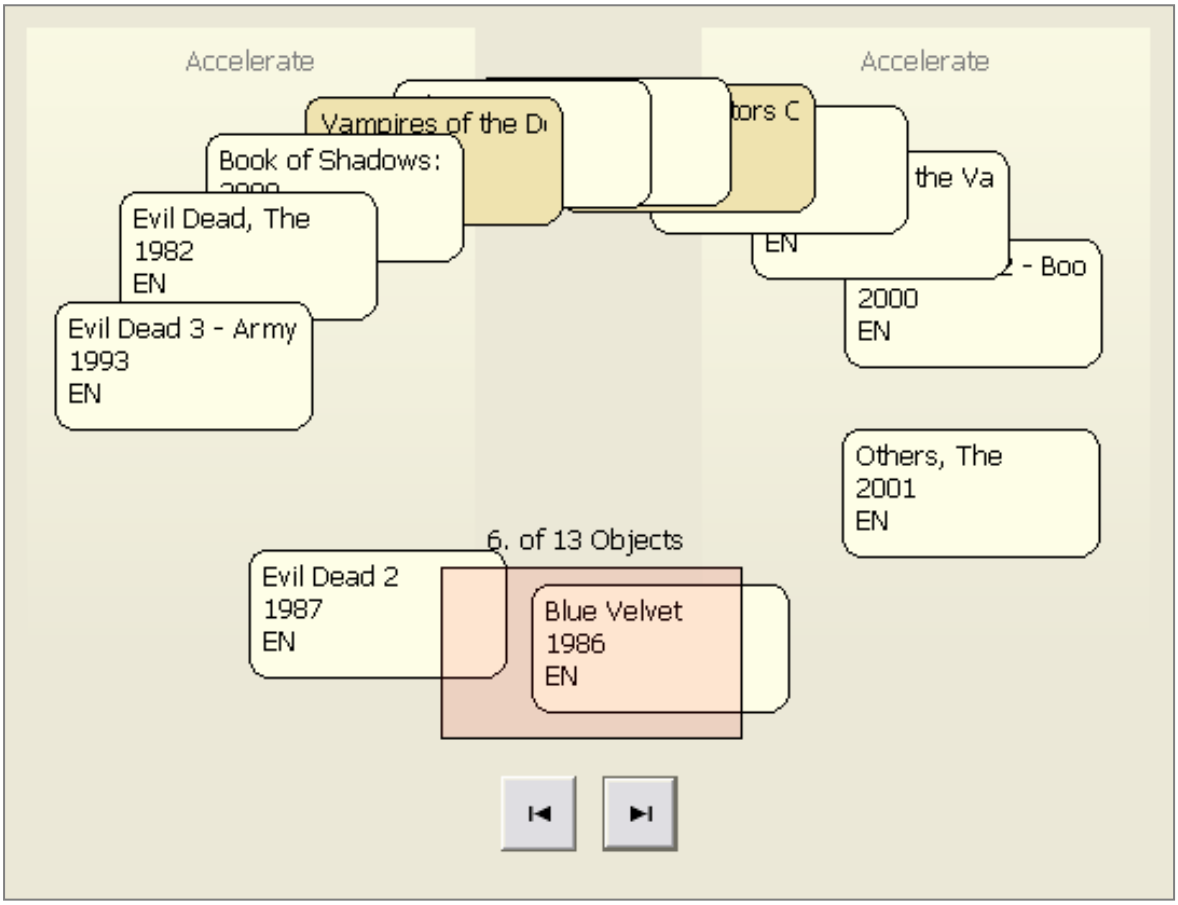


Abbildung 30: MDPView Ellipse der Elemente

In der Mitte werden die einzelnen Elemente eines Multi-Data-Points dargestellt (siehe Abbildung 30). Sie werden durch abgerundete Rechtecke repräsentiert, in denen ihre Bezeichnung und, je nach Darstellungsgröße, die Ausprägungen der wichtigsten Attribute

der Datenobjekte stehen. Selektierte Elemente werden in der dunklen Selektionsfarbe angezeigt, in der LevelTable oder GranularityTable fokussierte Objekte in der helleren Fokusfarbe. Durch Klick auf die Rechtecke können die Objekte selektiert werden.

Die Elemente werden auf einer Ellipse angeordnet, welche einen im Raum liegenden Kreis darstellt. Dadurch scheinen die in der oberen Hälfte dargestellten Objekte, im Hintergrund und die Objekte der unteren Hälfte im Vordergrund zu liegen. Es entsteht ein natürlicher dreidimensionaler Effekt. Je nach Anzahl der dargestellten Elemente, ist diese Ellipse stark bzw. weniger stark ausgeprägt.

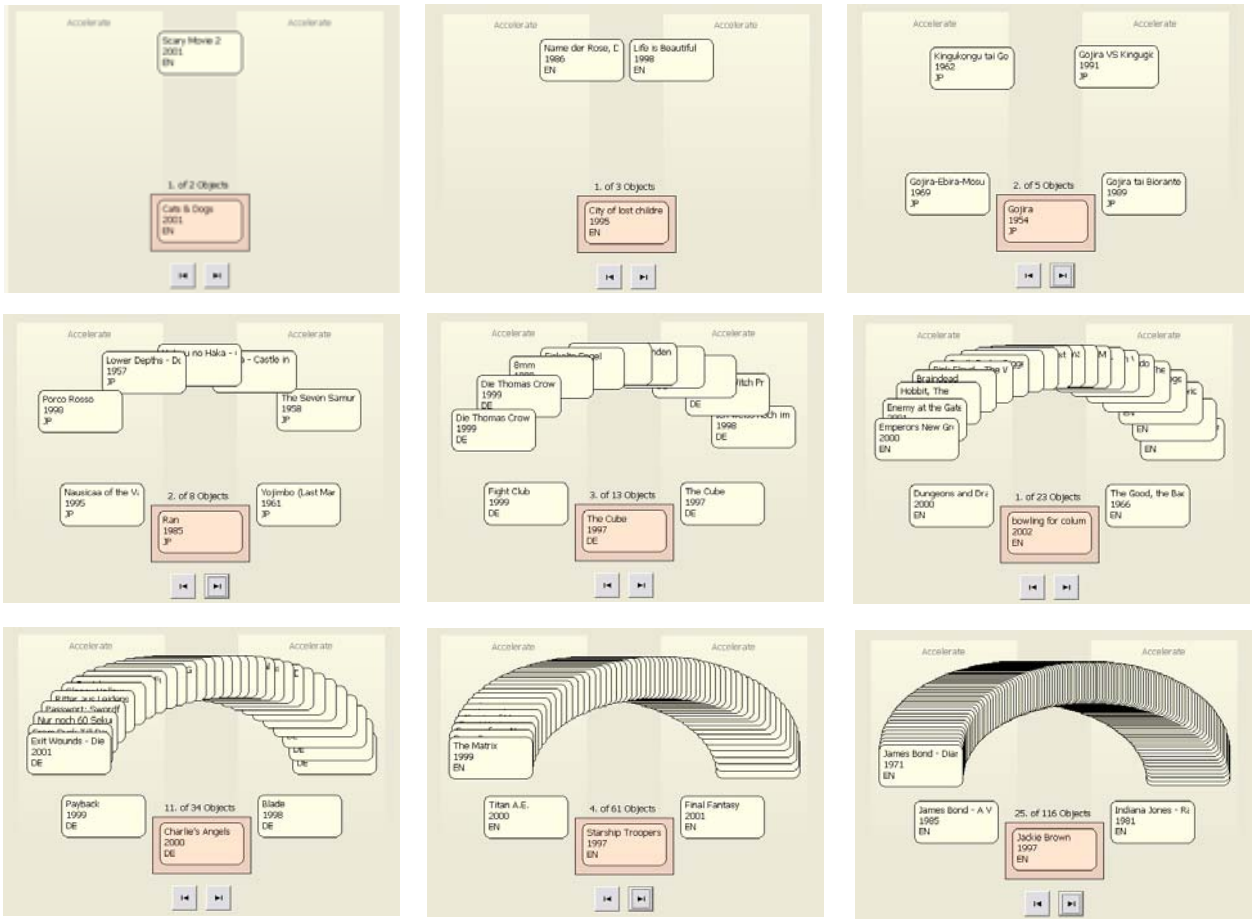


Abbildung 31: MDPView Ellipse der Elemente mit 2, 3, 5, 8, 13, 23, 34, 61 und 116 Elementen

In der unteren Hälfte der Ellipse werden maximal drei Objekte angezeigt, die restlichen werden in der oberen Hälfte gestaffelt. Das System ist rotierbar, d.h. die Elemente können entlang der virtuellen Ellipsenbahn rotiert werden. Vergleichbar mit einem Stapel von Karteikarten, kann das oberste Element ganz links, vom Stapel genommen werden und über

den unteren Bereich zum hinteren Ende des Stapels geführt werden. Dadurch, dass sich im unteren Bereich maximal drei Elemente befinden, sind diese Objekte zu jedem Zeitpunkt lesbar.

In der Mitte der unteren Ellipsenbahn befindet sich ein halbtransparentes, rotes Rechteck, ein wenig größer als ein Datenelement. Es markiert das fokussierte Objekt. Zu dem jeweiligen Objekt, das sich in diesem Fenster befindet und damit fokussiert ist, werden auf der linken Seite der Multi-Data-Point-View Detailinformationen angezeigt (Abbildung 32). Oberhalb des Fokusfensters kann der Benutzer ablesen, wie viele Elemente der Multi-Data-Point insgesamt enthält und um das wievielte es sich bei dem aktuellen Element handelt, z.B. 3. von 13 (Abbildung 31, Mitte).

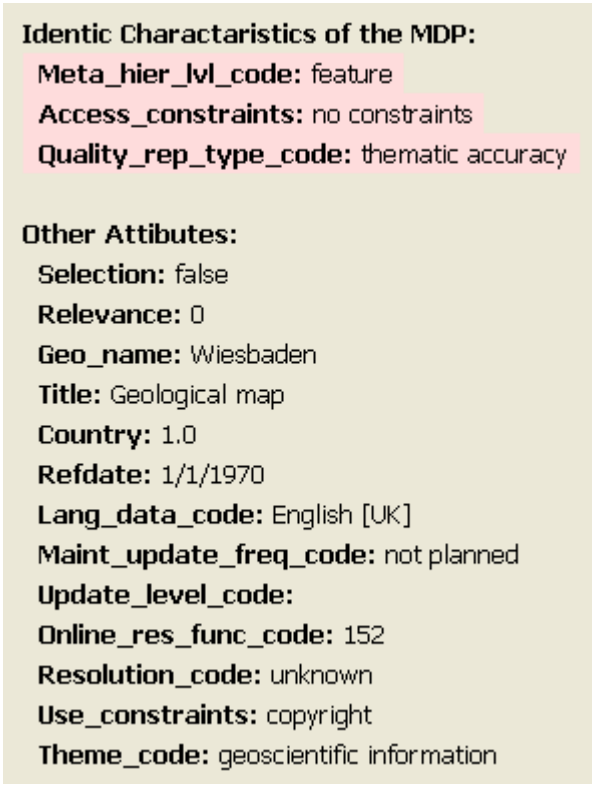


Abbildung 32: MDPView Detailinformationen zu einem fokussierten Element

Im Detailbereich im linken Teil der Visualisierung werden dem Benutzer möglichst alle Attribute des augenblicklich fokussierten Elements angezeigt. Zuoberst werden die für alle Elemente im Multi-Data-Point identischen Attribute, und deren Ausprägungen aufgeführt. Sie sind rot unterlegt. Darunter werden dem Anwender alle weiteren, verfügbaren Attribute der Datenelemente mit Attributsname und Ausprägung angezeigt.

6.3.3 Interaktion mit der Multi-Data-Point-Visualisierung

Die Multi-Data-Point-View bietet dem Benutzer verschiedene Interaktionsmöglichkeiten, insbesondere zur Rotation, aber auch zum Sortieren der Datenelemente.

Die Interaktionselemente zum Sortieren befinden sich oberhalb der Detaildarstellung der Datenelemente. Durch ein Pulldown-Menü kann der Benutzer ein Datenattribut auswählen, nach dem die Daten in der Ellipse sortiert werden sollen. Durch Radiobuttons kann er wählen, ob dies absteigend (engl. descending) oder aufsteigend (engl. ascending) geschehen soll.

Abbildung 33: MDPView Interaktionselemente zum Sortieren

Unterhalb der Elementen-Ellipse sind Buttons zum „Durchblättern“ der Elemente, vorwärts oder rückwärts (siehe Abbildung 30). Bei Klick werden alle Elemente um eine Position in die entsprechende Richtung rotiert und das nächste bzw. letzte Objekt rückt in den Vordergrund. Der Benutzer kann präzise - Element für Element - den Multi-Data-Point durchsuchen, ohne dabei ein Element zu überspringen.

Des Weiteren kann die Ellipse der Elemente auch stufenlos rotiert werden. In der linken und rechten Hälfte der Ellipse befindet sich ein maus-sensibler Bereich, gekennzeichnet durch ein Rechteck mit Farbverlauf und der Beschriftung „Accelerate“ (siehe Abbildung 30). Fährt der Benutzer mit der Maus über diesen Bereich, wird die Ellipse im Uhrzeigersinn (auf der rechten Seite) bzw. gegen den Uhrzeigersinn (auf der linken Seite) rotiert. Die Geschwindigkeit der Rotation ist unterschiedlich. Je weiter oben, im dunkleren Bereich, sich der Mauszeiger befindet, umso schneller rotiert die Ellipse. Ebenso kann die Rotation verlangsamt werden, wenn der Benutzer den Mauszeiger nach unten bewegt. Verlässt der Mauszeiger den sensiblen Bereich versiegt die Rotation und die Ellipse rotiert noch solange vor oder zurück, bis das dem Fokusfenster nächstgelegene Objekt fokussiert ist.
Durch die schnelle Rotation im hinteren Bereich kann der Benutzer selbst viele Dokumente effizient durchblättern.

6.4 Anwendungsbeispiel der Multi-Data-Point-Visualisierung

VisMeB soll dem Benutzer helfen, relevante Objekte in einer Treffermenge zu identifizieren. Hierfür bietet das System dem Benutzer verschiedene Formen der Visualisierung, unter anderem die 3D-Scatterplot Visualisierung. Besonderheit der 3D-Scatterplot-Visualisierung ist, dass drei Attribute der Metadatenobjekte in Bezug gesetzt werden können. Die Multi-Data-Point-Visualisierung kommt immer dann zur Anwendung, wenn die im 3D-Scatterplot visualisierten Attribute nicht genügen, um die Relevanz zu bewerten und Detailinformationen zu den einzelnen Datenpunkten im Multi-Data-Point verborgen liegen.

Dies soll am Beispiel einer Suche auf einer Geometadatenbank gezeigt werden. In diesem Beispiel ist der Anwender auf der Suche nach einer Stadtkarte von Wiesbaden aus den 40er Jahren. Er gibt als Suchbegriffe „city“, „map“ und „Wiesbaden“ ein. Schnell wird deutlich, dass sehr viele Objekte relevant zu sein scheinen (siehe Abbildung 34).

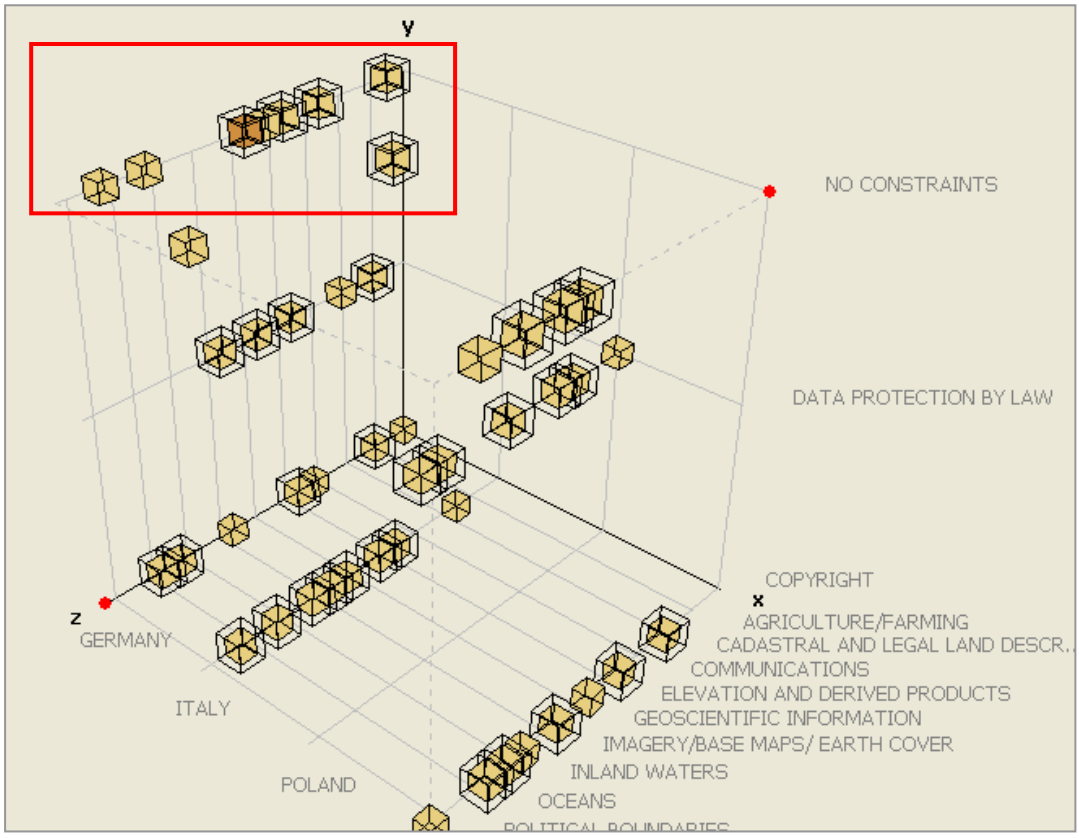


Abbildung 34: Die relevanten Objekte im VisMeB 3D-Scatterplot

In der 3D-Scatterplot Darstellung wählt der Anwender als Achsenbelegung „access Contraints“, „Country“ und „Theme Code“. Der Anwender sucht nur Dokumente ohne Zugangsbeschränkungen und nur Dokumente aus Deutschland. Für die Z-Achse wählt er „Theme Code“ als Achsenbelegung, da er so die Verteilung der Dokumente auf die verschiedenen Themengebiete dargestellt bekommt.

In der Visualisierung erkennt der Benutzer, dass sich die für ihn relevanten Dokumente im oberen linken Bereich des Scatterplots, entlang der Z-Achse verteilen (Abbildung 34, rot umrandet). Allerdings sind nur zwei der interessanten Objekte einfache Datenpunkte. Die anderen sind in Multi-Data-Points zusammengefasst. Nur zu den zwei einfachen Objekten kann er sich Detailinformationen durch den Tooltip anzeigen lassen. Auch das farbliche Hervorheben der Dokumente in der LevelTable, um dort weitere Informationen zu den Objekten zu erlangen, funktioniert nur für diese einfachen Datenpunkte. Trotzdem interessieren den Anwender auch die Objekte in den Multi-Data-Points. Wenn er mit der Maus über die Objekte fährt wird ihm angezeigt, aus wie vielen Elementen der Multi-Data-Point besteht.

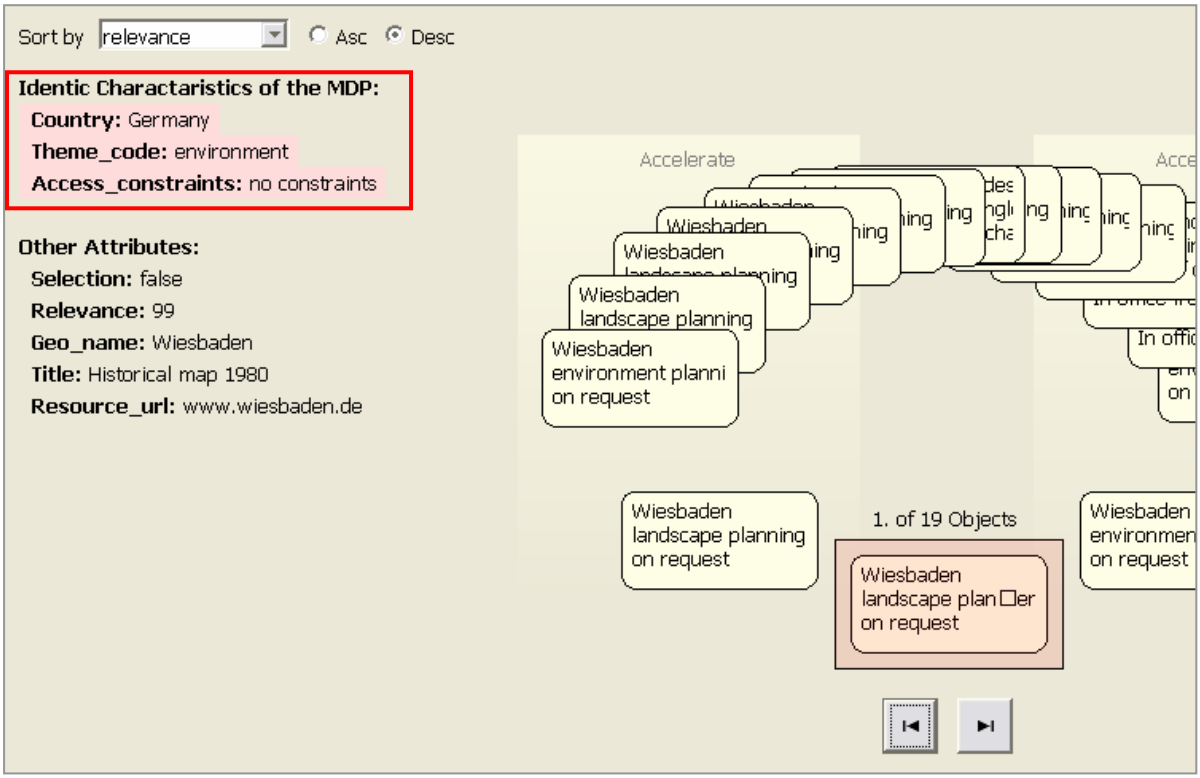


Abbildung 35: Die relevanten Objekte in der MDPView

Der erste Multi-Data-Point von links enthält fünf Datenelemente des Themenbereichs „health“, der zweite 12 Elemente aus dem Gebiet „economy“, der dritte 19 Datenelemente aus dem Gebiet “environment“. Aufgrund der thematischen Nähe entscheidet er sich für den letzteren. Durch einen Klick auf den Datenpunkt gelangt der Benutzer in die Multi-Data-Point- Visualisierung. Dort erkennt er, dass seine Vorgaben von allen Elementen des Multi-Data-Points erfüllt werden (Abbildung 35, rot markiert). Alle Elemente haben keine Zugriffsbeschränkungen, sind aus Deutschland und aus dem Themenbereich „environment“. Der Anwender sortiert die Elemente nach ihrer Relevanz und beginnt die Elemente zu durchblättern. Schon das erste Element ist eine historische Karte aus dem Jahre 1980, wie er am Titel erkennen kann. Über die maus-sensiblen Bereiche kann er die Elemente schnell durchblättern. Dabei achtet er auf die Titelangaben im Detailbereich. Nach wenigen Objekten findet er das gesuchte Element, eine historische Karte aus dem Jahre 1940. Da die Ellipse zu weit rotiert ist, blättert der Anwender mit dem Skip-Button die Elemente einzeln rückwärts durch, bis er das gesuchte Objekt im Fokus hat und markiert es. Um zu überprüfen, ob noch weitere Objekte aus dieser Zeit im Multi-Data-Point vorhanden sind, benutzt er die Sortierfunktion der *MDPView* und sortiert die Objekte absteigend nach dem Attribut „year“. Doch beim Durchblättern stellt er fest, dass die Karte das einzige Objekt aus dieser Zeit ist.

6.5 Umsetzung und Implementierung

Im Folgenden wird die Implementierung der Multi-Data-Point-Visualisierung näher vorgestellt. Wie bereits erwähnt sind alle Klassen der Multi-Data-Point Visualisierung in JAVA[8] implementiert. Als Compiler wird der JAVA 2 SDK Standard Edition in der Version 1.3.1_07 verwendet. Entwicklungsumgebung ist der Bornland JBuilder[9] ab Version 6 in Kombination mit dem Versionsverwaltungssystem CVS[10].

6.5.1 Implementierungsumgebung

VisMeB ist ein Framework aus unterschiedlichen Visualisierungen. Unabhängig von diesen Visualisierungen stellt das System eine Datenstruktur zur Verfügung, in der die Metadaten abgelegt werden. Die Visualisierungen können mittels Interfaces auf diese Datenstruktur zugreifen. Bei der Initialisierung des Systems werden die Metadaten und deren ausgewählte Attribute (konfigurierbar über das Assigment Tool) aus der Datenbank ausgelesen und in die Datenstruktur eingespeist. Für jedes Datenelement wird ein Objekt der Klasse *Dokument* erzeugt. Das Objekt enthält einen Array, indem die Ausprägungen der Attribute abgespeichert werden. Außerdem enthält das Objekt Instanzmerkmale für Selektion, Fokussierung, Relevanz bezüglich der Suchbegriffe und weitere interne Attribute sowie die Zugriffsmethoden auf das Objekt.

Die *Dokument*-Objekte werden durch die Klasse *Dokumentmanager* in einem weiteren Array verwaltet. Jedes Objekt bekommt eine eindeutige ID zugewiesen und kann von den Visualisierungen über den Dokumentmanager angesprochen werden.

Die Visualisierungen arbeiten somit alle auf den gleichen Daten. Änderungen an den Daten haben Auswirkungen auf alle Visualisierungen. Damit die Visualisierungen bei Änderungen benachrichtigt werden können, müssen sich die Visualisierungen zusätzlich beim

[8] JAVA, Sun Microsystems, http://java.sun.com/
[9] Borland JBuilder, Borland Software Corporation, http://www.borland.com/jbuilder/
[10] CVS, Concurrent Versions System, http://www.cvshome.org/

Viewnotifier registrieren. Wird in einer Visualisierung eine Änderung vorgenommen, z.B. ein Element selektiert oder gefiltert, meldet die Visualisierung dies an den *Viewnotifier*. Dieser leitet die Meldung an alle anderen, registrierten Visualisierungen weiter und ruft dort die entsprechende *repaint* Methoden auf. Jede Visualisierung muss daher das Interface *View* implementieren, um zu gewährleisten, dass die *repaint* Methoden vorhanden sind.

6.5.2 Implementierung innerhalb des 3D-Scatterplots

Auch der 3D-Scatterplot ist als Visualisierung registriert und arbeitet auf der VisMeB Datenstruktur. Allerdings erweitert er die Datenstruktur um eigene Elemente, nicht zuletzt auch um Elemente, die für die Visualisierung der Multi-Data-Points notwendig sind. Für jedes Element erzeugt er ein neues Objekt der Klasse *DataDoc*. In diesem wird eine Referenz auf das *Document*-Objekt gespeichert, über die auf die Ausprägungen des Datenelements zugegriffen werden kann. Erweitert wird das Datenelement um Instanzmerkmale zur Visualisierung im 3D-Scatterplot und Merkmale zur Visualisierung von Multi-Data-Points, sowie deren get&set-Methoden.

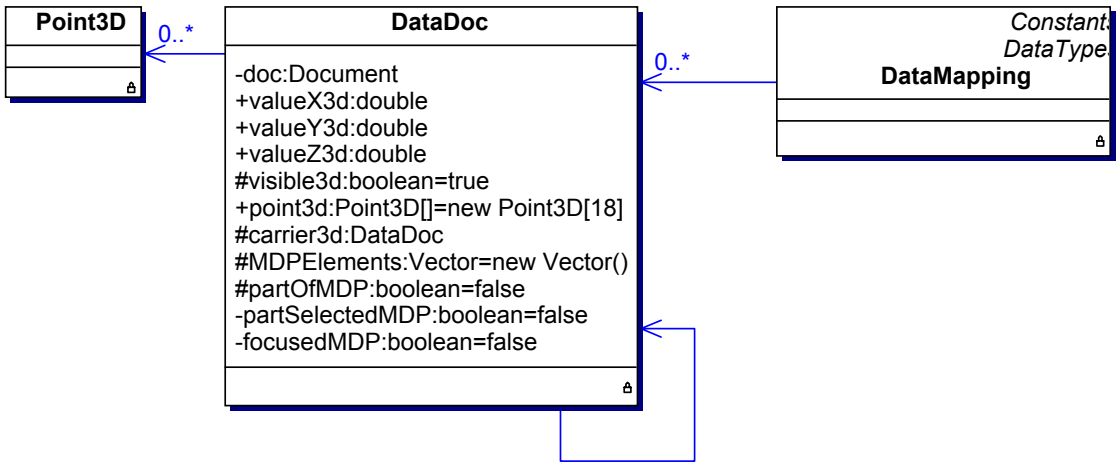


Abbildung 36: UML-Diagramm DataDoc-Klasse

Die *DataDoc*-Objekte werden in der Klasse *DataMapping* in einem Array gespeichert. Dort findet auch das Einordnen der Objekte in das dreidimensionale Koordinatensystem statt. Für jedes Datenelement werden die Koordinaten im virtuellen Raum berechnet. Dabei werden die Ausprägungen der Elemente bezüglich der dargestellten Dimension normalisiert und die

Datenelemente auf die Achsen entsprechend ihrer Ausprägung verteilt. Da die Elemente im 3D-Scatterplot durch einen Würfel, bzw. Multi-Data-Points durch einen doppelten Würfel oder eine Pyramide in einem Würfel repräsentiert werden, müssen für jeden Datenpunkt achtzehn Punkte im Raum berechnet werden (der eigentliche Datenpunkt, acht Punkte für den inneren Würfel, acht Punkte für den äußeren Würfel, einen zusätzlichen Punkt zur Darstellung der Pyramide), die im *DataDoc*-Objekt des Elements in einem Array aus dreidimensionalen Punkten gespeichert werden.

Am Ende des Mapping-Prozesses wird überprüft, ob mehreren Datenelementen die gleiche Position im Raum zugeordnet worden ist, also ob Multi-Data-Points existieren. Dies geschieht durch die Methode *CheckMDP*(siehe Anhang B). Für jedes Element werden X-, Y-, und Z-Koordinaten mit den Koordinaten der anderen Objekte verglichen. Wird ein Objekt mit denselben Koordinaten gefunden, wird das erste Objekt zum Repräsentanten des Multi-Data-Points. In einen Vektor des *DataDoc*-Objekts, genannt *MDPElements*, wird eine Referenz auf das überlappende Objekt gespeichert. Werden weitere Objekte mit denselben Koordinaten gefunden, werden auch diese dem *MDPElement*-Vektor angefügt. Ebenso wird im überlappenden Objekt eine Referenz auf den repräsentierenden Datenpunkt gespeichert. Sowohl der Repräsentant als auch die überlappenden Objekte werden durch das Instanzmerkmal *isPartOfMDP* im *Datadoc* als Teil eines Multi-Data-Points gekennzeichnet. Die überlappenden Objekte werden des Weiteren als nicht sichtbar im 3D-Scatterplot markiert.
Beim späteren Zeichnen wird überprüft, ob es sich bei dem aktuell zu zeichnenden Datenpunkt um einen Multi-Data-Point handelt. Falls ja, wird nur der Repräsentant gezeichnet. Außerdem wird der Selektionsstatus überprüft, also ob der Multi-Data-Point selektiert, teilweise selektiert oder fokussiert ist. Auch hierfür enthält das *DataDoc*-Objekt Merkmale. Dementsprechend wird der Multi-Data-Point in der Selektionsfarbe als doppelter Würfel oder Pyramide bzw. in der Fokusfarbe gezeichnet.

Durch einen Klick auf einen Multi-Data-Point in der 3D-Scatterplot Darstellung wird die Multi-Data-Point-View aktiviert. Allerdings überlagern sich die Datenpunkte in einer perspektivischen Darstellung häufig. Daher wird in einem zweidimensionalen Vektor für jedes auf dem Bildschirm dargestellte Pixel gespeichert, welches Datenelement in Z-Richtung an vorderster Stelle liegt. Klickt der Benutzer in den 3D-Scatterplot wird

überprüft, ob sich an dieser Position ein Datenpunkt befindet und ob es sich um einen Multi-Data-Point handelt. Falls ja, wird die Methode *ActivateMDPView* aufgerufen. In dieser Methode wird eine neue Instanz der Klasse MDPView kreiert, wobei das *DataDoc*-Objekt des angeklickten Multi-Data-Points als Parameter übergeben wird. Die *MDPView* ist von der Klasse *JPanel*[11] abgeleitet und kann als solche der 3D-Scatterplot Visualisierung angefügt werden. Der 3D-Scatterplot wechselt in den MDPView-Modus, wird verkleinert und modifiziert im rechten Teil des Panels gezeichnet.

Insgesamt enthält die 3D-Scatterplot Visualisierung etwa 500 Zeilen Quellcode, die die Multi-Data-Point-Visualisierung betreffen.

[11] Java Container Klasse, Grafisches Element, in das andere grafische Elemente eingefügt werden können und das seinerseits in anderen grafischen Komponenten, wie z.B. Fenster, eingefügt werden kann.

6.5.3 Implementierung der MDPView

Die Klassen der Multi-Data-Point-View sind in einem eigenen Package zusammengefasst. Das Package enthält fünf Klassen und etwa 1400 Zeilen Quellcode. Zentrale Elemente sind die Klassen *MDPView* und *MDPEllipse*. Die Klasse *MDPView* dient der Visualisierung. Sowohl die Datenelemente als auch der Detailtext und die Interaktionselemente werden hier gezeichnet.

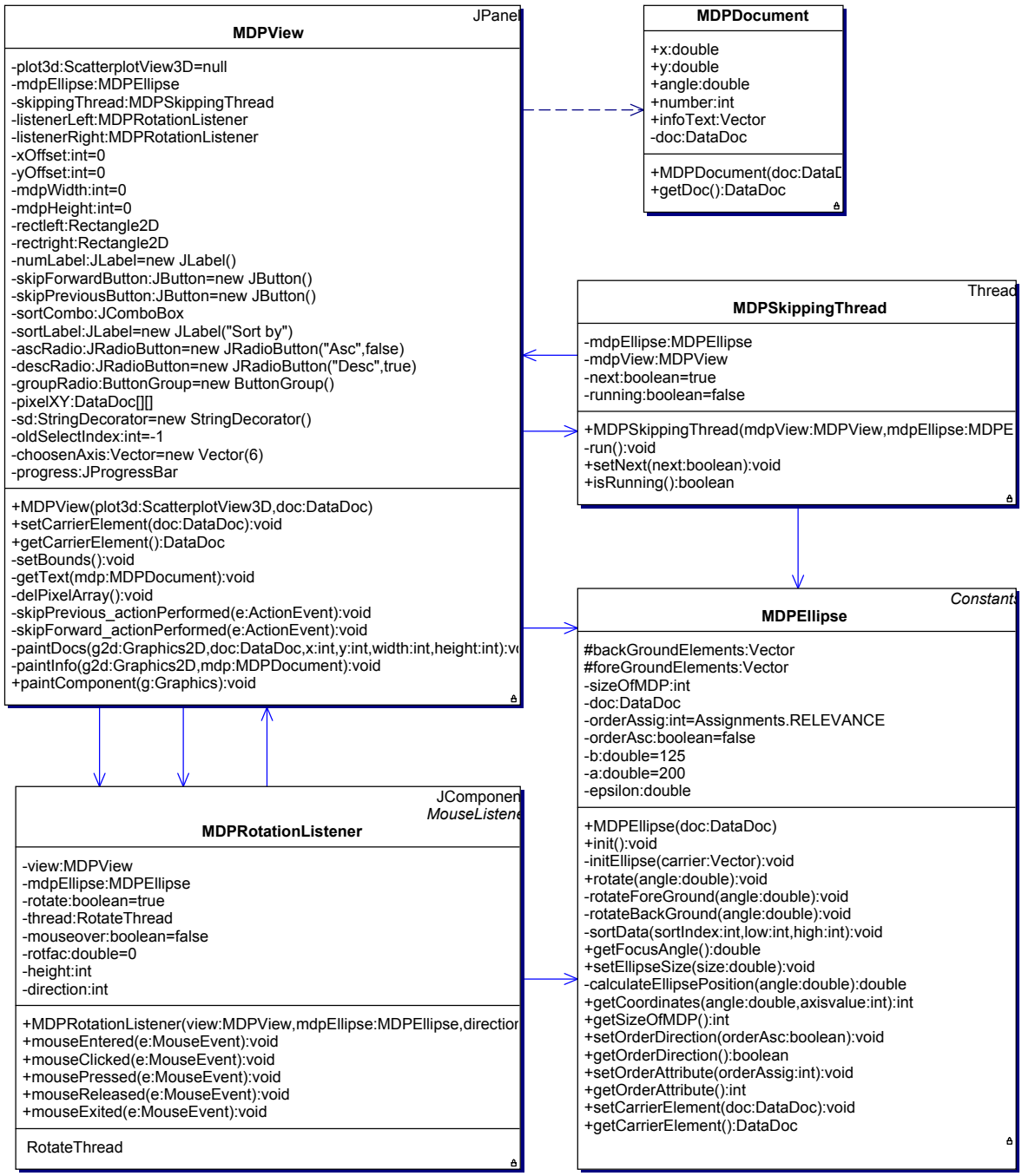


Abbildung 37: UML-Diagramm Multi-Data-Point-Visualisierung

Entsprechend der Idee der Objektorientierten Programmierung ist die Visualisierung streng vom Datenmodel getrennt. Die Datenelemente des Multi-Data-Points werden in der Klasse *MDPEllipse* verwaltet und in die neue Struktur der Ellipse eingegliedert.

Positionierung der MDP-Elemente

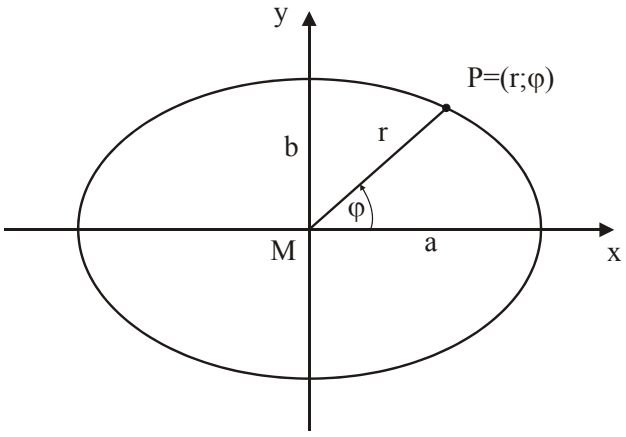
Bei der Instanzierung der Multi-Data-Point-View wird, wie bereits erwähnt, das *DataDoc* Objekt, das den Multi-Data-Point repräsentiert der *MDPView* übergeben. Die *MDPView* wiederum übergibt das Datenobjekt an die *MDPEllipse*. Das *DataDoc*-Objekt enthält Verweise auf die anderen Elemente des Multi-Data-Points in einem Objektvektor, dem *MDPElement*-Vektor. In der Initalisierungsmethode wird der Vektor ausgelesen und für jedes Element ein Objekt der Klasse *MDPDocument*(siehe Anhang B) erstellt. *MDPDocument* erweitert die *DataDoc*-Objekte der Datenelemente um Komponenten zur Visualisierung der Multi-Data-Points auf der Ellipse.

Unter anderem wird im *MDPDocument* auch die Position des Elements auf der Ellipsenbahn gespeichert. Diese Position wird über eine Ellipsengleichung in Polarkoordinaten (siehe Gleichung 12) bestimmt. Dadurch kann jeder Punkt auf einer in Höhe (entspricht der Hauptachse = 2a) und Breite (entspricht der Nebenachse = 2b) definierten Ellipse, allein durch einen Winkel berechnet werden.

$$r = \frac{b}{\sqrt{1 - \varepsilon^2 \cdot \cos^2 \varphi}} \qquad (\varepsilon < 1)$$

M : *Mittelpunkt*
$2a$: *Große Achse* (*Hauptachse*)
$2b$: *Kleine Achse* (*Nebenachse*)

$$\varepsilon = \frac{\sqrt{a^2 - b^2}}{a}$$



Formel 1: Ellipsengleichung in Polarkoordinaten

Es genügt daher für jedes *MDPElement* einen Winkel abzuspeichern, um dessen Position zu bestimmen.

Um die Objekte letztendlich im Panel darstellen zu können, müssen die durch die Winkel bestimmten Positionen, ins kartesische Koordinatensystem transformiert werden. Die resultierende X- und Y-Koordinaten werden ebenfalls im *MDPDocument* gespeichert. Weitere Variablen enthalten den Text mit Detailinformationen des Datenelements sowie einen Verweis auf das ursprüngliche *DataDoc*.

In der Initalisierungsmethode wird also für jedes *MDPElement* ein *MDPDocument* erstellt und dessen Position über einen Winkel definiert. Die Winkel werden im Bogenmaß und im negativen mathematischen Drehsinn angeben, da die Ordinate im Koordinatensystem des Java *JPanels* nach unten abgetragen wird.

Im allgemeinen Fall, d.h. wenn der Multi-Data-Point mehr als drei Elemente enthält, werden drei Elemente in die untere Hälfte der Ellipse gesetzt und die anderen Elemente in der oberen Hälfte verteilt. Das erste Element des sortierten *MDPElement*-Vectors wird an die unterste Stelle der Ellipse positioniert (½ π), das zweite Element links daneben (⅙ π) und das letzte Element des Vectors rechts daneben (⅚ π). Die restlichen Objekte werden (nach Gleichung 2) im oberen Bereich verteilt.

$$\varphi(i) = -\left(\frac{\pi}{2n}\right)\cdot(2i-1)$$

$$mit\ \varphi = Winkel\ der\ MDP\ Elemente$$
$$n = Anzahl\ der\ verbleibenden\ Elemente$$
$$i = Index\ des\ aktuellen\ Elements$$

Formel 2: mathematische Folge zur Anordnung der Hintergrundobjekte

In der mathematische Folge, nach der die Winkel der Objekte im Hintergrund berechnet werden, wird der Ellipsenbogen (½ π) in gleichgroße Anteile aufgeteilt und anhängig vom Index des aktuellen Elements, das ungerade Vielfache gebildet. Dadurch werden die Objekte symmetrisch und gleichmäßig verteilt.

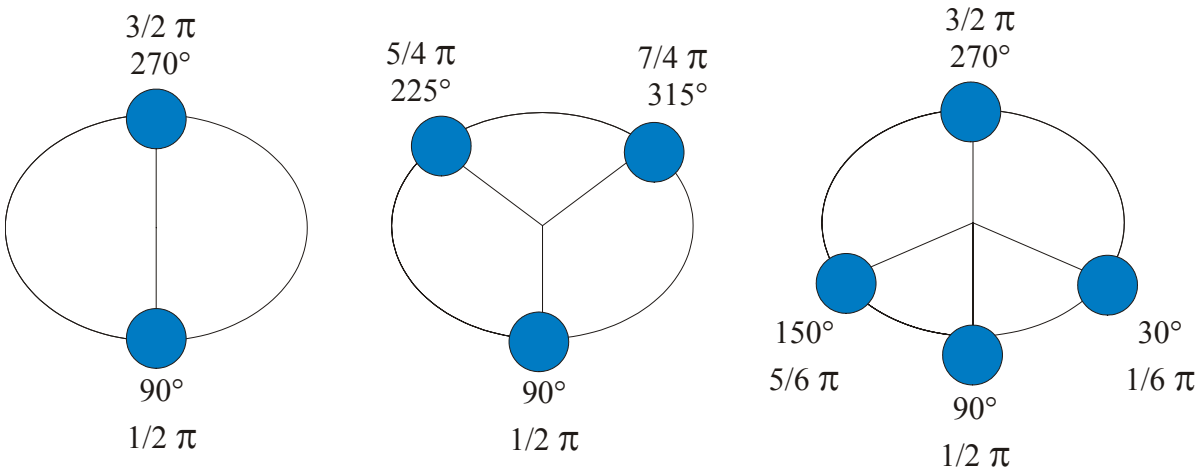


Abbildung 38: Anordnung der Multi-Data-Point-Elemente bei zwei, drei und vier Elementen

Besitzt der Multi-Data-Point drei oder weniger als Objekte, müssen die Elemente anders positioniert werden (Siehe Abbildung 38). Bei zwei Elementen wird das erste Element an die unterste Stelle der Ellipse gesetzt und das zweite an die oberste Position. Bei drei Elementen wird ein Objekt in den Vordergrund und zwei in den Hintergrund gestellt. In jedem Falle ist zu Beginn der Visualisierung ein Element bereits fokussiert.

Nachdem alle Winkel gesetzt und die Polarkoordinaten berechnet wurden (siehe Methode *calculateEllipsePosition* im Anhang B), können die Bildschirmpositionen bestimmt (siehe Methode *getCoordinates* im Anhang B) und die Objekte gezeichnet werden. Dies geschieht in der *PaintComponent* Methode der Klasse *MDPView*.

Rotation der Ellipse

Damit auch die im hinteren Teil der Ellipse dargestellten Objekte des Multi-Data-Points, in den Fokus gerückt werden können, muss die Ellipse der Elemente rotierbar sein. Da die Position der Elemente auf der Ellipse nur durch einen Winkel definiert ist, genügt es, diesen Winkel für alle Elemente um einen bestimmten Summanden zu erhöhen, um das System zu rotieren. Allerdings möchte man die Visualisierungsidee, dass die Objekte des Hintergrunds verdichtet sind und im Vordergrund nur drei Elemente dargestellt werden, sicherstellen. Um dies zu erreichen, müssen die Objekte des Vordergrunds mit einem anderen Winkel rotiert werden als die Objekte des Hintergrunds. Das Verhältnis der beiden Rotationen muss gerade

so sein, dass, wenn ein Objekt des Hintergrunds in den Vordergrund rückt, im Austausch ein Objekt des Vordergrunds dem Hintergrund übergeben wird. Das trifft genau dann zu, wenn das Verhältnis gilt

$$\frac{Rotationswinkel\,des\,Vordergrunds}{Rotationswinkel\,des\,Hintergrunds} = \frac{Anzahl\,der\,Elemente\,des\,Hintergrunds}{Anzahl\,der\,Elemente\,des\,Vordergrunds}$$

Formel 3: Verhältnis zwischen Rotationswinkel des Vordergrunds und des Hintergrunds

Ansonsten würde das System aus dem Gleichgewicht kommen.

In der Implementierung wurde dieses Problem gelöst, indem die *MDPDocument*-Objekte des Vordergrunds in einem anderen Vektor gespeichert werden als die Objekte des Hintergrunds. Die Objekte der beiden Vektoren werden bei der Rotation des Systems unterschiedlich behandelt. So wird zu den aktuellen Winkeln der Objekte des Vordergrunds der Drehwinkel einfach hinzuaddiert, während sich der Summand zur Rotation der Hindergrundelemente, aus obigem Verhältnis wie folgt berechnet:

$$NeuerWinkel = AlterWinkel + \frac{Rotationswinkel \cdot Anzahl\,der\,Objekte\;im\;Vordergrund}{Anzahl\,der\,Objekte\;im\,Hintergrund}$$

Formel 4: Formel zur Berechnung des neuen Winkels der Hintergrundelemente

Kommt ein Objekt an die Schwelle zwischen Vordergrund und Hintergrund, wird es aus dem einen Vektor entfernt und dem anderen angefügt. Gleichzeitig werden die Winkel der Objekte die übergeben werden, neu gesetzt. Dadurch werden Rundungsfehler, die aus der beschränkten Genauigkeit von π und dem Berechnen mit *Double*-Werten entstehen, korrigiert.

Der Benutzer hat zwei Möglichkeiten die Elemente der Ellipse rotieren zu lassen, durch Skip-Buttons unterhalb der Ellipse oder durch die maus-sensiblen Bereiche. Beide Methoden benutzen die gleiche Rotationsmethode in der Klasse *MDPEllipse*, deren Funktionsweise oben erklärt wurde (siehe auch Anhang B). Einziger Übergabeparameter ist der Rotationswinkel.

Durch die Skip-Buttons soll der Benutzer die Elemente des Multi-Data-Points einzeln durchblättern können. Bei Klick auf einen der Skip-Buttons wird das System rotiert bis das nächste bzw. vorherige Element im Fokus ist. Durch das *mouseclicked-Event* des Buttons wird eine neue Instanz der Klasse *SkippingThread* gebildet und ein neuer Thread[12] gestartet. Ein Übergabeparameter teilt der Klasse mit, welcher der beiden Buttons gedrückt wurde und damit, in welche Richtung das System rotiert werden muss. Der Winkel, um den die Elemente auf der Ellipsenbahn rotiert werden müssen, ist abhängig von der Anzahl der Elemente des Multi-Data-Points. Im allgemeinen Fall, bei mehr als drei Elementen, muss das System um ⅓ π rotiert werden. Bei weniger Elementen um π[13]. Im Thread wird dieser Winkel in fünfzig Teilwinkel zerteilt und die Rotationsmethode der Ellipse für jeden dieser Teilwinkel aufgerufen. Dadurch entsteht für den Betrachter der Eindruck einer animierten Bewegung.

Die zweite Möglichkeit die Elemente der Ellipse rotieren zu lassen, ist die Rotation über die maus-sensiblen Bereiche innerhalb der Ellipse. Hierbei handelt es sich um Instanzen der Klasse *MDPRotationListener*, die die Eigenschaften der Klasse *JComponent* erbt und dadurch dem Panel der *MDPView* angefügt werden kann. Sobald der Mauszeiger in einen dieser Bereiche kommt, wird ein Thread gestartet, der als *inner class*[14] in der *MDPRotationListener* Klasse implementiert wurde. Aus der relativen Position des Mauszeigers innerhalb der Komponente wird ein Rotationsfaktor berechnet und die Rotationsmethode der Ellipse aufgerufen. Dies geschieht solange, bis der Mauszeiger den Bereich wieder verlässt. Um sicherzustellen, dass am Ende der Rotation ein Element fokussiert ist, wird abgefragt, wie weit das unterste Element vom Fokus entfernt ist und das System nochmals um diesen Betrag rotiert. Das System pendelt sich ein. Auch bei der Rotation durch die maus-sensiblen Bereiche sind die Einzelwinkel, mit dem die Ellipse rotiert wird, so gewählt, dass der Benutzer die Rotation als fließend wahrnimmt.

[12] Ein Thread ist ein isolierter Prozess innerhalb einer Anwendung.
[13] Sowohl bei zwei als auch bei drei Elementen, ergibt sich auf Grund der unterschiedlichen Rotationsmethoden für Vorder- und Hintergrund der Rotationswinkel π.
[14] inner class: Eine nicht-statische lokale Klasse in einer Klasse.

7 ZUSAMMENFASSUNG

Der VisMeB Metadaten Browser soll Benutzern unterschiedlichster Anwendungsdomänen ein geeignetes Werkzeug zur Exploration und Suche auf großen Datenmengen bieten. Unterschiedlichste Visualisierungen vereint in einem Framework sollen den Betrachter darin unterstützen, auch aus großen Treffermengen, die für ihn relevanten Objekte herauszufiltern. Eine Visualisierungskomponente ist der 3D-Scatterplot. Er ermöglicht dem Anwender mehrdimensionale Korrelationen zwischen den Attributen eines Datenbestandes zu visualisieren und zu erkennen. Dadurch ist der 3D-Scatterplot ein wertvolles Werkzeug bei der Exploration der Daten. Doch gerade bei der Visualisierung abstrakter Daten mit vielen Datensätzen kommt es häufig zur vollständigen Überlappung einzelner Datenelemente, den Multi-Data-Points. Ohne besondere Kennzeichnung wären die Elemente des Multi-Data-Points nicht für den Betrachter gesondert erkennbar. Die Visualisierung wäre nicht vollständig und dadurch nicht zweckgerecht. VisMeB soll den Benutzer unterstützen relevante Objekte zu identifizieren. Ohne Multi-Data-Point-Visualisierung könnten die Elemente eines Multi-Data-Points nicht identifiziert und daher nicht in ihrer Relevanz bewertet werden.

Ziel des in dieser Arbeit vorgestellten Projekts war, eine geeignete Visualisierung der Multi-Data-Points für den 3D-Scatterplot sowie eine Möglichkeit zur effektiven und effizienten Datenexploration von Multi-Data-Points zu finden und diese in die bestehende Umgebung des Projekts VisMeB zu implementieren.

Die gefunden Lösung kennzeichnet Multi-Data-Points bereits im 3D-Scatterplot als solche und erlaubt dem Benutzer sich anzeigen zu lassen, wie viele Objekte im Multi-Data-Point enthalten sind. Außerdem erkennt der Benutzer an der Repräsentation der Multi-Data-Points im Scatterplot, ob enthaltene Objekte selektiert oder fokussiert sind. Wünscht der Anwender nähere Information zu einem Multi-Data-Point oder möchte er einzelne Elemente selektieren

oder fokussieren, kann er die enthaltenen Objekte mit der MDPView explorieren. Die MDPView gibt dem Benutzer einen schnellen Überblick über die Objekte im Multi-Data-Point. Durch die steuerbare und individuell anpassbare Geschwindigkeit beim Durchblättern kann der Benutzer den Multi-Data-Point effizient nach Elementen durchsuchen. Das Springen von Objekt zu Objekt durch die Skip-Buttons erlaubt ein kontrolliertes Durchsuchen der Elemente. Durch die Sortierfunktion kann der Benutzer die Elemente nach eigenen Prioritäten ordnen. Mit Hilfe der Darstellung aller verfügbaren Detailinformationen bezüglich der einzelnen Elemente im Detailbereich kann der Anwender die Relevanz der Objekte bestimmen.

Alles in Allem bietet die implementierte Multi-Data-Point-Visualisierung dem Anwender alle Möglichkeiten zur zweckgerechten Exploration der Multi-Data-Points in dem VisMeB 3D-Scatterplot.

8 ABBILDUNGSVERZEICHNIS

9 TABELLENVERZEICHNIS

10 FORMELVERZEICHNIS

11 QUELLENVERZEICHNIS

[1] Ahlberg, C., Shneiderman, B., *The Alphaslider: A Compact and Rapid Selector*,
[2] Card, S. K., Mackinley, J., Shneiderman B., *Readings in Information Visualization - Using Vision to Think* , Morgan Kaufmann Publishers, San Francisco, 1999
[3] C. Chen, *Information Visualisation and Virtual Environments,* November 1999
[4] Cleveland, W.S., *The Elements of Graphing Data*, Wadsworth Advanced Books, Monterey, 1985

[5] Däßler, R., H. Palm, *Virtuelle Informationsräume mit VRML*, dpunkt-Verlag, Heidelberg, 1998

[6] Fishkin, K., Stone, M.C., *Enhanced Dynamic Queries via Movable Filters*, Proceedings of the CHI'95 Conference, New York, ACM Press, 1995.

[7] Friendly, M., Denis, D.J., *Milestones in the history of thematic cartography, statistical graphics, and data visualization,* http://www.math.yorku.ca/SCS/Gallery/milestone/index.html, 5.11.2003

[8] Göbel S., Haist J., Müller F., Reiterer H.: *INVISIP: Usage of Information Visualization Techniques to Access Geospatial Data Archives*, 13th International Conference on Database and Expert Systems Applications (DEXA 2002), Aix en Provence, 2002

[9] Hearst, M. A., *TileBars: Visualization of Term Distribution Information in Full Text Information Access.* In: Katz, Irvin R.; Mack, Robert L.; Marks, Linn et al. (Eds.): CHI 1995: Conference Proceedings Human Factors in Computing Systems, ACM Press, New York ,1995. pages 59-66

[10] Hofmann, H., *Manet - Missings Are Now Equally Treated*, http://www1.math.uni-augsburg.de/Manet/index.html, 5.11.2003

[11] INSYDER, *Internet Système de Recherche*, http://www.insyder.com, 5.11.2003.

[12] INVISIP, *Information Visualization for Site Planning*, http://www.invisip.de, 5.11.03.2003.

[13] Klein P., Müller, F., Reiterer, H., Eibl, M., *Visual Information Retrieval with the SuperTable + Scatterplot*, Proceedings of the 6th International Conference on Information Visualisation (IV 02), IEEE Computer Society, 2002

[14] Klein, P., Reiterer, H., Müller, F., Limbach, T., *Metadata Visualization with VisMeB*, IV03, 7th International Conference on Information Visualization, London, 2003

[15] König, W., *Konzeption und Implementation eines 3D-Scatterplots zur Visualisierung von Metadaten*, Bachelor-Arbeit, Universität Konstanz, 2003

[16] Mackinlay, J., Robertson, G., Card, S., The *Perspective Wall: Detail and Context Smoothly Integrated*, Proceedings of ACM CHI'91: Human Factors in Computing Systems, 1991, pages 173-179.

[17] Mann, T. M., *Visualization of WWW-Search Results*, Dissertation, Freiburg, 2002

[18] North, C. L.; Shneiderman, B., *Snap-Together Visualizations: Can Users Construct and Operate Coordinated View,International Journal of Human-Computer Studies*, 53 (2000) 5, p. 715

[19] Papula, L., *Mathematik für Ingenieure und Naturwissenschaftler Bd. 1+2*, 10. Auflage, Vieweg, Wiesbaden, 2001

[20] Papula, L., *Mathematische Formelsammlung*, 10. Auflage, Vieweg, Wiesbaden, 2001

[21] Rao, R., Card, S. K., *The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information*, In: Card, S. K., Mackinlay, J. D., Shneiderman, B., eds., Readings in Information Visualization: Using Vision to Think, Morgan Kaufmann Publishers: San Francisco, pp. 343-349, 1999

[22] Reiterer H., Mußler G., Mann T., *Visual Information Retrieval for the WWW*, in: Smith M.J. et al. (eds.), Usability Evaluation and Interface Design, Lawrence Erlbaum, 2001, pp. 1150-1154

[23] Reiterer, H., Mußler, G., Mann, T., Handschuh, S., *Insyder - An Information Assistant for Business Intelligence*, Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, 2000.

[24] Reiterer H., Limbach T., Klein P., Müller F., Jetter C.: *Ein visueller Metadaten Browser für die explorative Erkundung großer Datenmengen*, Mensch & Computer 2003, Stuttgart, 2003

[25] Robertson, G., Mackinlay, J., Card, S., *Cone Trees: Animated 3D Visualizations of Hierarchical Information*, Proceedings of ACM CHI'91: Human Factors in Computing Systems, 1991, pages 189-194.

[26] Spence, R., *Information Visualization*, ACM Press, 2000, Vol. 1 2002, pp. 126-129

[27] StatSoft, Inc., *Electronic Statistics Textbook*, Tulsa, http://www.statsoft.com/textbook/stathome.html, 5.11.2003

[28] Tanin, E., Plaisant, C., Shneiderman, B. *Browsing Large Online Data with Query Previews*. In: Proceedings of the Symposium on New Paradigms in Information Visualization and Manipulation (NPIVM) 2000, ACM Press, Washington D.C., 2000

[29] Tufte, E. R., *Visual Explanation: Images And Quantities, Evidence And Narrative*, Graphics Press, Cheshire, 1997.

[30] Unwin, A., *Scatterplotting*, University of Augsburg, http://www1.math.uni-augsburg.de/~unwin/AntonyArts/Scatters.pdf, 5.11.2003

[31] Wilkinson, L., *Dot Plots*, http://www.spss.com/research/wilkinson/Publications/dots.pdf, SPSS Inc., Chicago, 5.11.2003

[32] Wilkinson, L., *The Grammar of Graphics,* Springer-Verlag, New York, 1999

12 ANHANG

12.1 Anhang A: Pflichtenheft

1. Zielbestimmung

1.1. Musskriterien

1.1.1. Implementierung von Animierten Multi-Data-Points innerhalb des 3D-Scatterplots des Metadaten Browsers im Projekt VisMeB
1.1.2. Repräsentation der einzelnen Data-Points mit einer Bezeichnung (z.B. Titel)
1.1.3. Anzeige von Detailinformationen zu den Data-Points
1.1.4. Selektion der einzelnen Data-Points
1.1.5. Visualisierung der Selektion durch eine Farbkennzeichnung und der Selektion in der LevelTable und der GranularityTable
1.1.6. Synchronisation mit der SuperTable und der GranularityTable

1.2. Wunschkriterien

1.2.1. Overview über die Gesamtansicht des Scatterplots
1.2.2. Steuerung der Animationsrichtung und Geschwindigkeit
1.2.3. Selektion aller Data-Points mit einem Klick

2. Produkteinsatz

2.1. Anwendungsbereiche

2.1.1. Das Produkt wird innerhalb des 3D-Scatterplots im VisMeB Metadaten Browsers zur Visualisierung von Suchergebnissen mit gleichen Dimensionswerten und zur Selektion dieser Ergebnisse eingesetzt.

2.2. Zielgruppen

2.2.1. Benutzer des VisMeB Metadaten Browsers.

2.3. Betriebsbedingungen

2.3.1. Büroumgebung

3. Produkt-Umgebung

3.1. Software

3.1.1. Java(TM) 2 Runtime Environment, ab Standard Edition 1.3.1

4. Produktfunktion

4.1. Wechsel von der bereits vorhandenen 3D-Ansicht des Scatterplots in eine detaillierte Ansicht der Multi-Data-Points durch Mausklick auf den Repräsentanten des Clusters

4.2. Steuerbarkeit innerhalb der Detail-Ansicht mit der Maus

4.2.1. Steuerung der Drehung der Data-Points um die Cluster Repräsentation durch sensitive Bereiche
4.2.2. Selektion der Data-Points durch Klick auf die Repräsentation
4.2.3. Wechsel zur Overview Ansicht durch Klick eines „Zurück“-Buttons
4.2.4. Alternativ zum Zurück-Button: Wechsel zur Overview Ansicht durch Klick auf die Overview-Ansicht

5. Benutzungsoberfläche

5.1. Look&Feel

5.1.1. Sun Java konformes Look&Feel

6. Qualitätsbestimmung (sehr gut / gut / normal / nicht relevant)

6.1. Funktionalität

6.1.1. Angemessenheit	gut
6.1.2. Richtigkeit	sehr gut
6.1.3. Interoperabilität	sehr gut
6.1.4. Ordnungsmäßigkeit	normal
6.1.5. Sicherheit	normal

6.2. Zuverlässigkeit

6.2.1. Reife	gut
6.2.2. Fehlertoleranz	gut
6.2.3. Widerherstellbarkeit	normal

6.3. Benutzbarkeit

6.3.1. Verständlichkeit	sehr gut
6.3.2. Erlernbarkeit	sehr gut
6.3.3. Bedienbarkeit	sehr gut

6.4. Effizienz

6.4.1. Zeitverhalten	gut
6.4.2. Verbrauchsverhalten	gut

6.5. Änderbarkeit

6.5.1.	Analysierbarkeit	normal
6.5.2.	Modifizierbarkeit	sehr gut
6.5.3.	Stabilität	sehr gut
6.5.4.	Prüfbarkeit	normal

6.6. Übertragbarkeit

6.6.1.	Anpassbarkeit	gut
6.6.2.	Installierbarkeit	normal
6.6.3.	Konformität	gut
6.6.4.	Austauschbarkeit	normal

7. Entwicklungsumgebung

7.1. Software

7.1.1. Microsoft Windows XP / Linux
7.1.2. Borland JBuilder 7 Personal

7.2. Hardware

7.2.1. PC mit Intel Pentium IV
7.2.2. 256 – 1024 MB RAM
7.2.3. 3D-fähige Grafikkarte, min. 16 MB RAM

7.3. Orgware

7.3.1. Breitband Internetverbindung
7.3.2. Ethernet 100 Mbit

12.2 Anhang B: Quellcode

12.2.1 checkMDP-Methode innerhalb Klasse MDPMapping

```
/**
 * Checks if documents have the same x-,y-,z-values and register the carriers.
 */
private void checkMDP()
{
  DataDoc doc1;
  DataDoc doc2;
  boolean sumAdd = false;
  int length = docs.length;
  for(int i=0;i<length;i++){
    docs[i].setIsPartMDP(false);
    docs[i].removeAllMDP3D();
    docs[i].setCarrier3D(null);
    docs[i].setVisible3D(true);
  }
  for(int i = 0;i < (length - 1);i++)
  {
    doc1 = docs[i];
    if(!doc1.getDocument().isAvailable()){
      continue;
    }

    for(int j = i + 1;j < length; j++)
    {
      doc2 = docs[j];
      if(!doc2.getDocument().isAvailable()){
        continue;
      }

      if((doc1.valueX3d == doc2.valueX3d) &&
         (doc1.valueY3d == doc2.valueY3d) &&
         (doc1.valueZ3d == doc2.valueZ3d) &&
         (doc2.isPartOfMDP() == false))
      {
        sumAdd = true;
        doc1.addElementMDP3D(doc2);
        doc2.setIsPartMDP(true);
        doc2.setVisible3D(false);
        doc2.setCarrier3D(doc1);
      }
    }

    if(sumAdd)
    {
      doc1.setIsPartMDP(true);
      doc1.setVisible3D(true);
      doc1.addElementMDP3D(doc1);
      doc1.setCarrier3D(doc1);
    }

    sumAdd = false;
  }
}
```

12.2.2 MDPDocument

```
package Views.Scatterplot3D.MDPView;

import java.util.Vector;
import Views.Scatterplot3D.*;

/**
 * Title:        MDPDocument
 * Description:  object of an MDP, stores all the attributes
 * Copyright:    Copyright (c) 2003
 * Company:      University of Constance
 * @author       Philipp Liebrenz, 01/473782, liebrenz@inf.uni-konstanz.de
 * @author       Werner Koenig, 01/458695, koenigw@inf.uni-konstanz.de
 * @version      1.0
 */

public class MDPDocument{

  public double x;
  public double y;
  public double angle;
  public int number;
  public Vector infoText;
  private DataDoc doc;



  /**
   * Constructor
   *
   * @param doc the doc document carrying the informations
   */
  public MDPDocument(DataDoc doc) {
    this.doc=doc;
    number = 0;
    infoText = new Vector();
  }

  /**
   * gives the doc document of the MDP
   * @return the doc document of the MDP
   */
  public DataDoc getDoc(){
    return doc;
  }
}
```

12.2.3 Init-Methode in der Klasse MDPEllipse

```
  /**
   * initalizes the vectors of the MDP elements. Decides whether they are in the
   * background or in the foreground. And puts them at the right position.
   *
   * @param carrier vector that carrys the MDP elements
   */
  private void initEllipse(Vector carrier){


    // The Fore- and the BackGround Vector, definite size to save resources
    backGroundElements = new Vector(sizeOfMDP);
    foreGroundElements = new Vector(5);
```

```
// different Numbers of Element causes different arrangements of the elements
// in the view
if (carrier.size()<4){

  // there must be something wong, if this happens
  if (carrier.size()==0) {
    System.err.println("ERROR: MultiDataPoint is no MultiDataPoint (Size=0)");
  }
  // MDP with only one element ??!?!?!?! never ever
  else if (carrier.size()==1) {
    System.err.println("ERROR: MultiDataPoint is no MultiDataPoint (Size=1)");
  }
  // two element: one in the back, one in the front
  else if (carrier.size()==2) {

    //-------------------------------------------------------------------------
    // take the first element
    MDPDocument mdp = new MDPDocument((DataDoc)carrier.elementAt(0));
    //put it in the very front
    mdp.angle=Math.PI/2;
    //and calculate the screenposition
    mdp.x=getCoordinates(mdp.angle, X_VALUE);
    mdp.y=getCoordinates(mdp.angle, Y_VALUE);
    // its the first
    mdp.number = 1;
    // and belongs to Foreground
    foreGroundElements.add(mdp);
    //-------------------------------------------------------------------------


    //-------------------------------------------------------------------------
    // take the second
    MDPDocument mdp1 = new MDPDocument((DataDoc)carrier.elementAt(1));
    // put it in the very back
    mdp1.angle=3*Math.PI/2;
    //and calculate the screenposition
    mdp1.x=getCoordinates(mdp1.angle, X_VALUE);
    mdp1.y=getCoordinates(mdp1.angle, Y_VALUE);
    // its the secound
    mdp1.number = 2;
    // and belongs to Background
    backGroundElements.add(mdp1);
    //-------------------------------------------------------------------------
  }
  // three elements one in the front, two in the back
  else if (carrier.size()==3){

    MDPDocument mdp2 = new MDPDocument((DataDoc)carrier.elementAt(2));
    mdp2.angle=7*Math.PI/4;
    mdp2.x=getCoordinates(mdp2.angle, X_VALUE);
    mdp2.y=getCoordinates(mdp2.angle, Y_VALUE);
    mdp2.number = 3;
    backGroundElements.add(mdp2);

    MDPDocument mdp = new MDPDocument((DataDoc)carrier.elementAt(0));
    mdp.angle=Math.PI/2;
    mdp.x=getCoordinates(mdp.angle, X_VALUE);
    mdp.y=getCoordinates(mdp.angle, Y_VALUE);
    mdp.number = 1;
    foreGroundElements.add(mdp);

    MDPDocument mdp1 = new MDPDocument((DataDoc)carrier.elementAt(1));
    mdp1.angle=5*Math.PI/4;
    mdp1.x=getCoordinates(mdp1.angle, X_VALUE);
    mdp1.y=getCoordinates(mdp1.angle, Y_VALUE);
    mdp1.number = 2;
```

```
      backGroundElements.add(mdp1);
    }
  }
  // and if there are more than three:
  // put first one in the very front
  // the last one right beside
  // the second one left beside
  else {
    // foreground-----------------------------------------------------------


    MDPDocument mdp3 = new MDPDocument((DataDoc)carrier.elementAt(1));
    mdp3.angle=Math.PI/6;
    mdp3.x=getCoordinates(mdp3.angle, X_VALUE);
    mdp3.y=getCoordinates(mdp3.angle, Y_VALUE);
    mdp3.number = carrier.size();
    foreGroundElements.add(mdp3);

    MDPDocument mdp2 = new MDPDocument((DataDoc)carrier.elementAt(0));
    mdp2.angle=Math.PI/2;
    mdp2.x=getCoordinates(mdp2.angle, X_VALUE);
    mdp2.y=getCoordinates(mdp2.angle, Y_VALUE);
    mdp2.number = 1;
    foreGroundElements.add(mdp2);

    MDPDocument mdp1 =
       new MDPDocument((DataDoc)carrier.elementAt(carrier.size()-1));
    mdp1.angle=Math.PI*5/6;
    mdp1.x=getCoordinates(mdp1.angle, X_VALUE);
    mdp1.y=getCoordinates(mdp1.angle, Y_VALUE);
    mdp1.number =2;
    foreGroundElements.add(mdp1);


    // background ----------------------------------------------------------
    MDPDocument mdpBackground;
    int size = carrier.size()-3;
    double index =1;
    double fac = (Math.PI/(2*(size)));

    for( int i=size; i>=1; i--){

      mdpBackground = new MDPDocument((DataDoc)carrier.elementAt(i));
      mdpBackground.angle =-((((2*index)-1))*fac);
      mdpBackground.x=getCoordinates(mdpBackground.angle, X_VALUE);
      mdpBackground.y=getCoordinates(mdpBackground.angle, Y_VALUE);
      mdpBackground.number = i+1;
      backGroundElements.add(mdpBackground);
      index++;

    }
  }
```

12.2.4 CalculateEllipsePosition und getCoordinates in MDPEllipse

```
/**
 * calculates the position of an element by an given angle
 *
 * @param angle
 * @return
 */
private double calculateEllipsePosition(double angle){

  double r;
```

```
    double cosAngle = Math.cos(angle);


    r = b/Math.sqrt(1-(epsilon*epsilon*cosAngle*cosAngle));
    // epsilon = (Math.sqrt(a*a-b*b))/a;

    return r;
  }


  /**
   * callculates the x or y coordinate by an given angle
   *
   * @param angle
   * @param axisvalue identifier of axis (x or y)
   * @return the calcualted value (x or y)
   */
  public int getCoordinates(double angle,int axisvalue){

    double r;
    r =calculateEllipsePosition(angle);

    if (axisvalue==X_AXIS) return (int)(r*(Math.cos(angle)));
    else if (axisvalue==Y_AXIS) return (int)(r*(Math.sin(angle)));
    else {
      System.err.println("Not and guilty axis value");
      return -1;
    }
```

12.2.5 Rotationsmethoden in MDPEllipse

```
  /**
   * public class to rotate the ellipse
   *
   * @param angle the angle the ellipse should be rotated
   */
  public void rotate(double angle){
    rotateForeGround(angle);
    rotateBackGround(angle);
  }

  /**
   * rotates the elements of the foreground and decides if they should be
   * given to the background
   *
   * @param angle the angle the ellipse should be rotated
   */
  private void rotateForeGround(double angle){

    //rotate left
    if (angle<0){
      if ((((MDPDocument)foreGroundElements.get(0)).angle) <= 0){

        ((MDPDocument)foreGroundElements.elementAt(0)).angle=0;
        backGroundElements.insertElementAt(foreGroundElements.elementAt(0),0);

        // rearrange the elements
        if (foreGroundElements.size()>2){
          ((MDPDocument)foreGroundElements.elementAt(1)).angle=Math.PI/3;
          ((MDPDocument)foreGroundElements.elementAt(2)).angle=Math.PI*2/3;
        }

        foreGroundElements.removeElementAt(0);

        ((MDPDocument)backGroundElements.lastElement()).angle=Math.PI;
```

```
      foreGroundElements.add(backGroundElements.lastElement());
      backGroundElements.removeElementAt(backGroundElements.size()-1);
    }
  }
  // rotate right
  else{
    if ((((MDPDocument)foreGroundElements.get(
              foreGroundElements.size()-1)).angle) >= Math.PI){

      ((MDPDocument)foreGroundElements.lastElement()).angle=Math.PI;
      backGroundElements.add(foreGroundElements.lastElement());

      // rearrange the elements
      if (foreGroundElements.size()>1){
        ((MDPDocument)foreGroundElements.elementAt(0)).angle=Math.PI/3;
        ((MDPDocument)foreGroundElements.elementAt(1)).angle=Math.PI*2/3;
      }

      foreGroundElements.removeElementAt(foreGroundElements.size()-1);

      ((MDPDocument)backGroundElements.get(0)).angle=0;
      foreGroundElements.insertElementAt(backGroundElements.elementAt(0),0);
      backGroundElements.removeElementAt(0);
    }
  }
  MDPDocument mdp;
  double tmp;
  for (int i=0; i<foreGroundElements.size();i++){
    mdp = (MDPDocument)foreGroundElements.get(i);
    tmp = mdp.angle += angle;
    mdp.x=getCoordinates(tmp, X_VALUE);
    mdp.y=getCoordinates(tmp, Y_VALUE);
  }
}



/**
 * rotates the elements of the background
 * @param angle the angle the ellipse should be rotated
 */
private void rotateBackGround( double angle){
  double addAngle = (angle*foreGroundElements.size())/backGroundElements.size();
  MDPDocument mdp;
  double tmp;
  for (int i=0; i<backGroundElements.size();i++){
    mdp = (MDPDocument)backGroundElements.get(i);
    tmp = mdp.angle += addAngle;
    mdp.x = getCoordinates(tmp, X_VALUE);
    mdp.y = getCoordinates(tmp, Y_VALUE);
  }
```

12.3 Anhang C: CD-ROM

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

- Ausführbare Version von VisMeB (Stand 12.11.2003)
- Bachelorarbeit im PDF-Format
- Quellen
- Abbildungen

Erklärung

Ich versichere hiermit, dass ich die anliegende Arbeit mit dem Thema „Visualisierung von Multi-Data-Points in einem 3D-Scatterplot - Konzeption & Implementierung innerhalb eines Metadaten Browsers“ selbständig verfasst und keine anderen Hilfsmittel als die angegebenen benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinne nach entnommen sind, habe ich in jedem einzelnen Falle durch Angabe der Quelle, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht.

Diese Arbeit wird nach Abschluss des Prüfungsverfahrens der Universitätsbibliothek Konstanz übergeben und ist durch Einsicht und Ausleihe somit der Öffentlichkeit zugänglich.
Als Urheber der anliegenden Arbeit stimme ich diesem Verfahren zu / nicht zu *).

Konstanz, ... Unterschrift .....

*) Nichtzutreffendes bitte streichen.