CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Masterarbeit

Entwicklung einer Abstraktionsschicht zwischen Darstellungen, Eingabeformen und Datenquellen zur visuellen Analyse

vorgelegt von

B.Sc. Arne Uphoff

Gutachter:

Prof. Dr. Dr. h.c. H.-J. Appelrath
Dipl.-Inform. Stefan Flöring

19. April 2010

Zusammenfassung

Die zur Verfügung stehende Datenmenge steigt stetig an. Dies ist beispielsweise in der Einführung von Überwachungssystemen oder gesetzlichen Aufzeichnungspflichten begründet. Die Daten werden in divergenten Formaten in verschiedenen Datenbanken hinterlegt. Um aus der Datenflut Informationen zu gewinnen muss eine Analyse durchgeführt werden. Allerdings steigen im Vergleich mit der Datenmenge die Analysefähigkeiten nicht so stark an. Dieses Problem wird als Information-Overload-Problem bezeichnet. Die visuelle Analyse wird als eine Lösung für das Information-Overload-Problem angesehen, in dem der Mensch stärker in einen integrierten Analyseprozess eingebunden wird. Für den Zugang zu der Datenmenge werden Visualisierungen verwendet, welche mit Hilfe von Interaktionen beeinflusst werden können. Für eine effektive Analyse ist die Ansicht der Datenmenge mit Hilfe von verschiedenen Visualisierungen notwendig. Die Herausforderung besteht in der Verwendung dieser Visualisierungen, die um Korrelationen leichter zu erkennen, miteinander verknüpft werden müssen. Verschärft wird das Problem durch die Notwendigkeit verschiedene Datenquellen in die Analyse einzubeziehen.

Der Lösungsansatz, der in dieser Arbeit verfolgt wird, ist die Entwicklung einer Abstraktionsschicht zwischen Visualisierungen, Interaktionen und Datenquellen. Auf dieser Basis wird eine Verknüpfung unabhängig vom verwendeten Modell zur Datenbeschreibung ermöglicht. Zudem wird durch die Abstraktionsschicht erreicht, dass verschiedenartige Datenquellen in ein System eingebunden und zur Analyse mit den zur Verfügung stehenden Visualisierungen verwendet werden können. Des Weiteren wird die Einbindung von Visualisierungskonzepten und Datenquellen ermöglicht, die nicht explizit für das spezifische System entwickelt wurden. Diese können auch nach Fertigstellung eingebunden werden.

Die Abstraktionsschicht wird mit Hilfe einer domänenspezifischen Sprache entwickelt. Um die Elemente der Sprache zu ermitteln, wird ein Klassifikationssystem für Visualisierungen erstellt. Im Anschluss wird das hieraus resultierende Modell in ein prototypisches System, genannt *Visual-Analytics-Transformation (VAT)*-System, überführt. Bei der Entwicklung des *VAT*-Systems liegt der Fokus auf der Erweiterbarkeit und Verknüpfung von Analyseelementen auch über den Kontext einer Datenquelle hinaus. Diese Ziele werden mit Hilfe von Transformationen erreicht.

Inhalt

1 Einführung

Die technologische Entwicklung im Bereich der Informationstechnologie ermöglicht die Speicherung immer größerer Datenmengen, allerdings werden im Schnitt 80% der gespeicherten Daten nie verwendet [Car03]. Um aus den Daten Schlussfolgerungen ziehen zu können, müssen diese einem systematischen Erkenntnisprozess unterzogen werden [AN95]. Hierbei wird ausgehend von Daten und Informationen, Wissen aus den Daten gewonnen. In großen Datenmengen und bei schlechter Datenqualität gestaltet sich dieses als besonders schwierig [Kei02b].

Das Problem wird durch das *Information-Overload* Phänomen verschärft. Die Fähigkeit Daten zu speichern wächst schneller als die Rechnenleistung und damit schneller als die Auswertungsmöglichkeiten [KMS+08]. Nach einer Studie des Marktforschungs- und Beratungsunternehmens International Data Corporation (IDC) wird die digital produzierte Datenmenge 2011 etwa 1.800 Exabyte betragen und sich damit im Vergleich zum Jahr 2006 verzehnfacht haben [Gan08]. Die Suche nach den wirklich relevanten Informationen aus der großen Datenmenge wird immer komplexer und kostenintensiver.

Manuelle Datenauswertungsverfahren zur Informationsgewinnung sind aufgrund der hohen Fehlerwahrscheinlichkeit häufig nicht praktikabel. Außerdem sind die kognitiven Fähigkeiten eines Menschen bei großen Datenmengen schnell erschöpft. [GE97]

Auch klassische Data-Mining-Verfahren stoßen bei großen Datenmengen schnell an ihre Grenzen. Die Datenqualität und die Komplexibilität der Algorithmen, die zur Auswertung verwendet werden, sind hierfür die wichtigsten Gründe [Sch07]. Die Algorithmen müssen zudem in den meisten Fällen für einen Datenbestand speziell parametrisiert werden. Für die Ermittlung dieser Parameter ist allerdings Verständnis für die Funktionsweise der Algorithmen notwendig, das in der Praxis nicht immer gegeben ist [SN07].

Es ist ersichtlich, dass für die Auswertung neue Analyseverfahren zur explorativen Datenanalyse entwickelt werden müssen. Als eine Lösung wird die visuelle, explorative Datenanalyse (*Visual Analytics*) gesehen [KMS+08], welche im Folgenden vorgestellt wird.

1.1 Visual-Analytics

Mit Hilfe von Visual-Analytics wird der Mensch mit seiner Flexibilität, Kreativität und seinem Allgemeinwissen in den Analyseprozess eingebunden, wodurch eine Ergänzung zu algorithmischen Verfahren, wie sie im Data-Mining auftreten, angestrebt wird. Ziel von Visual-Analytics ist es, den Information-Overload in einen Vorteil umzukehren [KKS+09]. Die visuelle Exploration von Daten hielt bereits in der zweiten Hälfte der 70er Jahre Einzug in die Statistik. Die Grundlagen hat J.W. Tukey mit der Idee „*Looking at data to see what it seems to say*“ [Tuk77] gelegt. Das Konzept von Tukey besteht darin, die Daten zunächst graphisch darzustellen und anhand dieser auf eine Hypothese zu schließen, welche durch die konfirmatorische Statistik bestätigt oder widerlegt wird. Aus diesen Grundzügen hat sich die explorative Datenanalyse entwickelt.

Definition 1 (Visual-Analytics) *Visual-Analytics ist die Bildung von abstrakten visuellen Metaphern in Kombination mit einer vom Menschen geführten Informationserörterung (gewöhnlich eine Interakti-*

onsform). Das ermöglicht die Erkennung von Erwartetem und die Entdeckung von Unerwartetem in großen sich dynamisch ändernden Informationsräumen. [CES07]

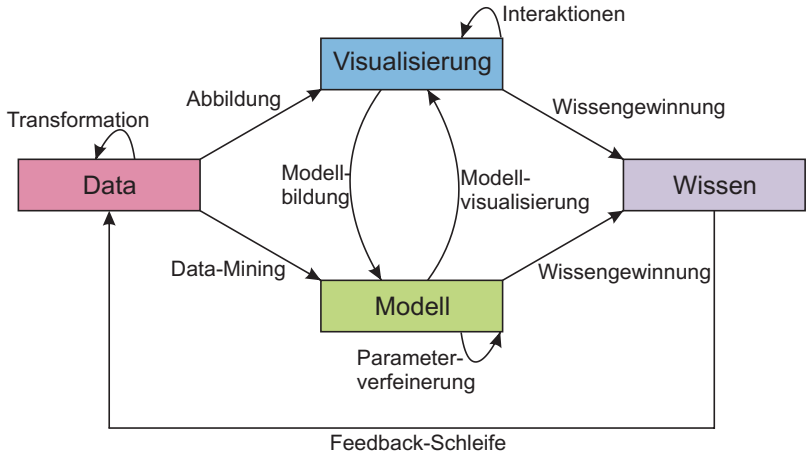


Abbildung 1.1: *Der Visual Analytics-Prozess (vgl. [KMSZ09])*

Visual-Analytics wird nach [KMS+08] durch einen Prozess beschrieben. Abbildung 1.1 gibt eine Übersicht über die Komponenten des Prozesses. Dieser besteht aus unterschiedlichen Zuständen (dargestellt als Blöcke) und Transitionen (Pfeile).

In vielen Visual-Analytics-Szenarien müssen zunächst heterogene Datenquellen integriert werden, bevor visuelle oder automatische Analysemethoden angewendet werden können. Deshalb ist der erste Schritt oftmals die Transformation der Daten in aussagekräftige Einheiten zur weiteren Datenverarbeitung. Aufbauend auf den transformierten Daten kann ein Analyst zwischen einer visuellen oder automatischen Analysemethode wählen. Nach Auswahl kann der Analyst evtl. direkt das gewünschte Wissen erhalten, allerdings ist es wahrscheinlich, dass eine initiale Visualisierung für die Analyse nicht ausreicht. Benutzerinteraktionen auf Visualisierungen sind notwendig, um aufschlussreiche Informationen zu enthüllen. Zum Beispiel durch Zoomen in verschiedenen Bereichen der Darstellung können Details zu einer Datenmenge betrachtet werden. Im Gegensatz zu klassischen Informationsvisualisierungen können Visualisierungsmodelle, die durch Interaktionen verändert wurden, zur automatischen Analyse wiederverwendet werden. Diese Modelle können auch aus den originalen Daten erstellt werden, indem Data-Mining-Techniken verwendet werden. Nachdem ein Modell erstellt wurde, hat der Analyst die Möglichkeit mit den automatischen Analysemethoden zu interagieren, indem Parameter modifiziert oder andere Typen von Analysealgorithmen ausgewählt werden. Eine Modellvisualisierung kann anschließend verwendet werden, um das gefundene Modell zu verifizieren. Der Wechsel zwischen visuellen und automatischen Methoden ist eine wesentliche Eigenschaft des Visual-Analytics-Prozesses und führt zu einer kontinuierlichen Verfeinerung und Verifikation von vorläufigen Ergebnissen. Irreführende Ergebnisse in einem Zwischenschritt können in einem frühen Status der Analyse entdeckt werden, was zu einem größeren Vertrauen in die Endergebnisse führt. Im Visual-Analytics-Prozess kann Wissen durch Visualisierungen oder automatische Analysen gewonnen werden. Durch die Feedback-Schleife wird sichergestellt, dass das durch die Analyse gefundene Wissen nicht verloren geht.

Zusammenfassend ergeben sich die folgend aufgeführten Vorteile, die durch die Einbindung des Menschen in die Analyse im Vergleich zum klassischen Data-Mining entstehen:

- Der Mensch kann bei geeigneter Visualisierung sehr leicht inhomogene und „verrauschte“ Daten von signifikanten Daten unterscheiden [Kei01].
- Data Mining basiert auf komplexen Algorithmen, die von – nicht immer verfügbaren und mitunter widersprechenden – Experten parametrisiert werden müssen. Weil die visuelle Datenexploration nicht auf komplexen Algorithmen basiert, werden diese Experten nicht benötigt. [Kei01]
- Da der Mensch selbst in den Explorationsprozess eingebunden ist, kann er das Explorationsziel auch während der Analyse ändern. Außerdem kann dem Ergebnis ein größeres Vertrauen entgegengebracht werden, da Zwischenergebnisse vom Analysten verifiziert oder zumindest auf Plausibilität hin geprüft werden können [SN07].
- Im Gegensatz zu klassischen Data-Mining-Verfahren ist die Auswertung von Datenbeständen möglich, über deren verborgene Informationen sehr wenig bekannt ist, weshalb die Auswertungsziele noch nicht eindeutig definiert werden können [Kei02b].

Die Eigenschaften der visuellen Analyse sollen auch in der Anwendungsdomäne, die im Folgenden vorgestellt wird, genutzt werden.

1.2 Anwendungsdomäne: Bevölkerungsbezogene Epidemiologie

Im Gesundheitswesen, speziell im Bereich der bevölkerungsbezogenen Epidemiologie, spielt die Datenauswertung eine wichtige Rolle [FHTA09]. Die Epidemiologie ist eine Wissenschaft, die sich mit der Verteilung von Krankheiten einer Bevölkerung und den Einflussfaktoren für die Verteilung beschäftigt. Epidemiologische Studien sind im Gegensatz zu experimentellen Studien reine Beobachtungsstudien am Menschen unter realen Umweltbedingungen. Sie stellen die Basis für die Abschätzung des Erkrankungsrisikos beim Menschen dar, weil aus ethischen Gründen bei Menschen häufig keine experimentellen Studien durchgeführt werden können [Sob09]. Während die deskriptive Epidemiologie das Auftreten von Krankheiten in einer bestimmten Bevölkerung mit soziodemographischen Attributen, wie Alter, Geschlecht, Beruf oder Umwelt, beschreibt, beschäftigt sich die analytische Epidemiologie mit den Hintergründen der Erkrankungen und versucht Erkrankungsursachen zu ermitteln. Zusätzlich ist die Entdeckung von Trends zwischen Krankheiten und potenziellen Einflussfaktoren basierend auf geographischen Regionen besonders von Interesse [MRAK03]. Eine deskriptive epidemiologische Datenbank stellt das *Epidemiologisches Krebsregister Niedersachsen (EKN)* zur Verfügung. In der Registerstelle des *EKN* sind zurzeit weit über eine Millionen Datensätze zu Tumorerkrankungen und krebsbedingten Todesfällen verzeichnet [Tho09]. Die Datenmenge wird durch Vorsorgeuntersuchungen, wie zum Beispiel landesweite Mammographie-Screenings, weiter rapide ansteigen. Ziel bei der Analyse der Datenmenge ist es unter anderem Erkenntnisse über die Zusammenhänge zwischen ortsbezogenen Faktoren und Erkrankungshäufigkeiten zu gewinnen [FHTA09].

Typischerweise benötigen Analysten verschiedene Arten von Darstellungsformen. Für Daten mit Raumbezug, wie sie im *EKN*-Register vorliegen, sind thematsiche Karten eine verbreitete Form der Darstellung. Allerdings ergibt es keinen Sinn diese Darstellungsform auf Daten ohne Raumbezug anzuwenden. Zudem ist eine ausschließliche Darstellung mit thematischen Karten nicht immer vorteilhaft, da andere Visualisierungsformen sich für spezielle Fragestellungen besser eignen können. Um zum Beispiel Anomalien in einem Datenbestand entdecken zu können, werden häufig Punktdiagramme

verwendet, um ein Verständnis für zeitabhängige Trends zu erlangen, können Liniendiagramme besser sein [HSM09]. Programme zur visuellen Exploration müssen eine Kombination verschiedener Visualisierungsarten unterstützen [Shn96]. Dabei sollten die Programme sowohl bekannte als auch neuartige, weniger bekannte Darstellungsformen unterstützen[LO95]. Gerade die Einbindung weiterer Darstellungen gestaltet sich in einem bestehenden System oft schwierig, da nicht vorhergesehen werden kann, wie neue Visualisierungen die Daten darstellen und welche Möglichkeiten zur Interaktion bestehen.

1.3 Analyseszenario

Das Verfahren zur visuellen, explorativen Datenanalyse lässt sich durch den Visual-Analytics-Prozess (vgl. Abbildung 1.1) beschreiben. Im Folgenden Analyseszenario wird der Prozess anhand eines Szenarios in der Anwendungsdomäne vorgestellt. Hierbei wird ein Verweis auf die Elemente des Prozesses in Klammern gegeben.

Zu untersuchen sind epidemiologischen Daten mit Raumbezug, welche speziell für Krebserkrankungen mit Todesfolge in Niedersachsen aus einem großen Datenbestand extrahiert wurden (*Transformation*). Mit Hilfe einer automatischen Verteilungsanalyse (*Data-Mining*) ist die Hypothese aufgestellt worden, dass in den letzten Jahren die Anzahl der Krebsfälle in Südniedersachsen stark zugenommen hat. Diese Hypothese soll verifiziert werden. Hierzu verschafft sich der Analyst zunächst mit Hilfe einer Choroplethenkarte einen Überblick (*Modelvisualisierung*) über die Verteilung der Fälle in Niedersachsen. Eine Choroplethenkarte ist eine thematische Karte, bei der die Gebiete im Verhältnis zur Verteilung des beobachteten Merkmales – meistens durch Farbe – hervorgehoben werden. Der Analyst stellt fest, dass in Südniedersachsen am meisten Fälle aufgetreten sind. Weil dieses Gebiet das bevölkerungsreichste ist, kann hieraus noch kein Schluss gezogen werden. Es muss der Faktor der Bevölkerungsverteilung herausgerechnet werden. Hierzu verknüpft der Analyst die Darstellung mit der statistischen Maßzahl zur Bevölkerungsdichte (*Modellbildung und anschließende Modellvisualisierung*). Hierdurch verschiebt sich die Ballung der Krebserkrankungen in das Gebiet Nord-West-Niedersachsen. Um die Verteilung in diesem Gebiet direkt zu betrachten, wird die Visualisierung vergrößert und verfeinert (*Interaktion*). Durch eine genaue Betrachtung der Anzahl der Krebsfälle, stellt der Analyst fest, dass in der Region Ostfriesland Krebsfälle gehäuft auftreten (*Wissen*). In einem weiteren Schritt soll untersucht werden, ob diese Häufung in einer bestimmten Krebsart begründet ist oder ob alle Krebsarten gleichmäßig häufig vorkommen (*Feedback-Schleife*). Zu diesem Zweck möchte sich der Analyst die Verteilung der einzelnen Krebsarten in den Städten und Regionen Ostfrieslands anschauen. Eine Choroplethenkarte kann jedoch nur die Verteilung eines Merkmales darstellen. Aus diesem Grund wählt der Analyst eine Punktdarstellung für die Anzeige der Verteilung der verschiedenen Krebsarten (*Abbildung*). Es soll im speziellen untersucht werden, welche Krebsart in den verschiedenen Teilen Ostfrieslands besonders häufig auftritt. Deshalb verfeinert der Analyst die Ansicht der Krebsarten und wählt nacheinander verschiedene Regionen zur Hervorhebung aus (*Interaktion*). Damit der Analyst in der thematischen Karte sehen kann, welche Region er betrachtet und eine Korrelation schnell erkennen kann, wird das jeweils aktuell betrachtete Gebiet der Punktdarstellung in der Karte hervorgehoben. Der Analyst muss weitere Einflussfaktoren betrachten, wie zum Beispiel die Altersverteilung, um die Annahme zu verifizieren.

Es ist ersichtlich, dass die visuelle Analyse ein Ablauf von Schritten, bestehend aus der Betrachtung von Visualisierungen und Interaktionen ist. Hierbei kommen verschiedene Darstellungsformen zum Einsatz. Jede Interaktion ändert die Ansicht einer Darstellung, welche evtl. auf eine andere Darstellung mit einer anderen Sicht auf die Daten abgebildet werden kann bzw. muss. Aus der Verknüpfung von

verschiedenen oder gleichen Visualisierungsarten ergeben sich die nachfolgend diskutieren Herausforderungen.

1.4 Herausforderungen im Bereich von Visual-Analytics

Ausgehend von der Anforderung, dass ein System zur explorativen, visuellen Datenanalyse von epidemiologischen Daten unterschiedliche Visualisierungsformen benötigt und diese miteinander verknüpft werden müssen, entstehen Herausforderungen, die in diesem Kapitel aufgezeigt werden.

Durch die Verknüpfung von verschiedenen Visualisierungstechniken wird versucht die Schwächen einer Visualisierungsform durch die Stärken einer anderen Visualisierungsform auszugleichen [Kei02a]. Es stellt sich allerdings die Frage, wie diese Verknüpfung durchgeführt werden kann, da jede Visualisierungsform in der Regel unterschiedliche Eigenschaften aufweist. Die Herausforderungen bestehen also darin, ähnliche Teilbereiche in Visualisierungen zu ermitteln und zu bestimmen wie ein Teilbereich einer Visualisierung auf eine andere Visualisierung abgebildet werden kann. Die reine Abbildung von Teilbereichen einer Visualisierung auf eine andere fördert den Analyseprozess nur unzureichend, da diese Abbildung statisch wäre. Änderungen, wie zum Beispiel die Selektion eines Elementes in einer Visualisierung, würden nicht auf die verknüpften Visualisierungen abgebildet werden. Es ist erforderlich, dass durch Benutzerinteraktionen ausgelöste dynamische Änderungen ebenfalls auf die verknüpften Visualisierungen abgebildet werden. Damit ergibt sich die Herausforderung, Interaktionen auch auf verknüpfte Visualisierungen abzubilden.

Interaktionen sind zum Beispiel nötig, weil selten der gesamte Datenbestand auf einmal angezeigt werden kann. Der Benutzer muss in der Lage sein, den sichtbaren Bereich einer Visualisierung individuell zu ändern. Um diese Interaktionen auszuführen, sind interaktive visuelle Schnittstellen nötig, die eine intuitive Bedienung ermöglichen, um den Analyseprozess nicht durch eine komplexe Bedienung zu behindern. Neuartige Bedienkonzepte und Eingabegeräte für die visuelle Analyse, wie in [FH09] und [IC07] beschrieben, können die intuitive Bedienung eines Systems steigern. Es stellt sich jedoch die Frage, wie diese unterschiedlichen Eingabegeräte in ein System eingebunden werden können, gerade auch dann, wenn diese Eingabegeräte noch nicht in der Entwicklungsphase des Systems zur Verfügung stehen. Eine weitere Komponente, die nicht unbedingt in der Entwicklungsphase zur Verfügung steht, sind Visualisierungsformen. In der Wissenschaft werden regelmäßig neue oder angepasste Visualisierungsformen entwickelt. Die Herausforderung an ein Analysesystem besteht darin, diese neuartigen Visualisierungsformen ohne eine vollständige Reimplementierung einzubinden und mit der vollen Funktionsvielfalt (Verknüpfung, Interaktionsabstraktion und Datenabstraktion) dem Analysten zur Verfügung zu stellen.

Eine Lösung dieser Herausforderungen wird in einer Abstraktionsschicht zwischen Daten, Eingaben und visuellen Repräsentationen gesehen, die in dieser Arbeit entwickelt wird.

1.5 Aufbau der Arbeit

Die weitere Arbeit ist wie folgt gegliedert. Zunächst wird im Grundlagenabschnitt die domänenspezifische modellgetriebene Softwareentwicklung vorgestellt. Vor dem Hintergrund dieser Grundlagen wird ein Softwaresystem zur Modellierung und Einbindung von *domänenspezifische Sprache (Domain Specific Language) (DSL)*s vorgestellt. Zudem wird die *Multidimensional Statistical Data Analysis*

Engine (MUSTANG)-Plattform vorgestellt, welche als eine Abstraktionsschicht zu einer *Online Analytical Processing (OLAP)*-Datenbank verwendet werden kann. Im Anschluss an diesen Abschnitt wird die Modellierung von Visualisierungen und Interaktionen vorgestellt. Diese Eigenschaften dienen zur Bildung eines Modells für das zu entwickelnden System. Nachdem in einem weiteren Abschnitt aktuelle Visual-Analytics-Systeme vergleichend vorgestellt werden, wird ein Klassifikationssystem für Visualisierungen im Bereich der visuellen, explorativen Analyse erstellt. Die hieraus gewonnen Erkenntnisse in Kombination mit den Modellierungskonzepten fließen in das Konzept und dem Entwurf des *VAT*-Systems ein. Die Vorstellung des *VAT*-Systems erfolgt nach der Identifizierung der Anwendungsfälle und funktionalen Anforderungen. Der anschließende Entwurfsabschnitt gliedert sich in den Grob- und Feinentwurf. Er wird durch ausgesuchte Implementierungsaspekte in dem folgenden Abschnitt ergänzt. Abgeschlossen wird diese Arbeit mit einem Vergleich des *VAT*-Systems mit den vorgestellten Visual-Analytics-Systemen und einem Fazit.

2 Grundlagen

Das zu entwickelnde System hat das Ziel den visuellen Explorationsprozess zu unterstützen. Diese Anwendungsdomäne schränkt das zu entwickelnde System ein. Die Anwendungsdomäne wird im System mit Hilfe der domänenspezifischen modellgetriebenen Softwareentwicklung abgebildet. In diesem Abschnitt wird diese Technik vorgestellt und anschließend ein System zur Verwendung einer *DSL* in einem System beschrieben. Mit Hilfe dieser Eigenschaften wird ein Modell für das *VAT*-System entwickelt. Eine weitere wichtige Eigenschaft eines Analysesystems ist neben einem geeigneten Modell der Zugriff auf die Datenmenge. Aus diesem Grund wird im anschließenden Kapitel ein System zur Abstraktion eines Zugriffes auf multidimensionale Datenbanken, die sich für analytische Systeme etabliert haben, vorgestellt.

2.1 Domänenspezifische modellgetriebene Softwareentwicklung

Die *domänenspezifische Modellierung (Domain Specific Modeling) (DSM)* ist eine Technik, um die domänengetriebene Entwicklung zu unterstützen. Ziel der domänengetriebenen Entwicklung ist eine an die Anwendungsdomäne angepasste Umgebung in der die Konzepte (einschließlich der Semantik) direkt verwendet werden können. Zum Beispiel ist es bei der Entwicklung einer Plattform für Versicherungen, einfacher mit den Begriffen „Riskio“, „Bonus“ und „Schaden“ zu agieren, als mit entsprechendem C#-Code. Das Modell repräsentiert keine Konzepte aus der Programmierumgebung, sondern aus der Anwendungsumgebung, wodurch es evtl. auch von Nicht-Programmierern entwickelt werden kann [VC09]. Des Weiteren ist die Spezifikation des Modells auf einem signifikant höheren Niveau als traditioneller Quellcode und Klassendiagramme. Dies bedeutet, dass die Spezifizierung leichter und dadurch schneller durchgeführt werden kann. Außerdem fällt das Modell auf Grundlage seiner Anwendungsdomänengültigkeit sehr schlank aus. Es müssen keine Techniken oder Konstrukte modelliert werden, die nicht benötigt werden [TK04]. Das Endprodukt wird aus der abstrakten Spezifikation der Anwendungsdomäne mit Hilfe von Quellcodegeneratoren zu weiten Teilen, in einigen Fällen auch komplett, generiert [VC09]. Die Eigenschaft des höheren Abstraktionsgrades der domänenspezifischen, modellgetriebenen Softwareentwicklung wird genutzt, um das Abstraktionsmodell zu entwickeln und damit die genannten Vorteile einer *DSM* zu nutzen. Zur domänengetriebenen Entwicklung können allerdings nicht allgemeingültige Sprachen verwendet werden, da sie nicht an die Anwendungsdomäne angepasst werden können. Ausgehend von der Assemblersprache ist zwar der Abstraktionsgrad der Programmiersprache immer weiter erhöht worden, allerdings können zum Beispiel die Notationen nicht an eine Domäne angepasst werden[VC09]. Es ist somit eine spezielle Technik notwendig, die im Folgenden vorgestellt wird. Im Anschluss wird ein Tool zur Umsetzung dieser Technik vorgestellt.

Zur Nutzung der Vorteile des *DSM*, wird eine *DSL* benötigt, die eine speziell für eine Domäne entworfene formale Sprache ist. Eine *DSL* umfasst nach [VC09] dabei mindestens die folgenden Aspekte:

Abstrakte Syntax: Die *abstrakte Syntax* ist eine Datenstruktur, die den semantisch relevanten Inhalt eines mit Hilfe einer *DSL* definierten Systems darstellt. Sie ist relevant für die Verarbeitung mit Werkzeugen, wie zum Beispiel Codegeneratoren, deren Aufgabe es ist, semantisch äquivalenten Code in der Zielsprache zu generieren.

Konkrete Syntax: Anwender, welche die Sprache verwenden möchten, interagieren mit der *konkreten Syntax*. Diese ist eine Darstellung der abstrakten Syntax. Die Interaktion mit der konkreten Syntax kann entweder graphisch mit Hilfe eines Modellierungstools geschehen, oder durch Texteingabe. Bei der textuellen Definition ist eine Grammatik notwendig, die eine formale Beschreibung der konkreten Syntax ist und somit eine Abbildung zur abstrakten Syntax ermöglicht.

Semantik: Die Semantik ist die Bedeutung dessen, was mit der konkreten Syntax geschrieben wird. Sie kann auf unterschiedliche Weise definiert werden. In der Praxis wird allerdings häufig die Sprache mit Hilfe einer Dokumentation in Prosa oder mittels Interpretern und Codegeneratoren definiert.

Einschränkungen: Bei den meisten *DSL*s existieren implizite Randbedingungen, die nicht durch gängige Formalismen zur Beschreibung einer abstrakten Syntax definiert werden können, wie zum Beispiel, dass Attribute einer Datenstruktur einen eindeutigen Namen haben müssen. *Einschränkungen*, auch Constraints genannt, dienen dazu, solche Regeln sicherzustellen.

Um das Modell, definiert mit Hilfe einer *DSL*, in einem Softwaresystem verwenden zu können, ist in der Regel eine Integration in eine Laufzeitinfrastruktur notwendig. Hierzu existieren generell zwei unterschiedliche Ansätze: Interpreter und Codegenerator [VC09]. Ein Interpreter interpretiert den abstrakten Syntaxbaum einer *DSL* und führt entsprechend der Semantik Aktionen aus. Hingegen wandeln Codegeneratoren den Syntaxbaum in einen semantisch äquivalenten Quellcode einer bestimmten Programmiersprache. Der so generierte Quellcode wird eingebunden und evtl. kompiliert.

Nachfolgend wird eine *Language Workbench* [Fow05] vorgestellt und anhand eines konkreten Szenarios auf ihre Tauglichkeit zur Modellierung des Gesamtsystems überprüft. Eine *Language Workbench* ist eine Werkzeugsammlung mit den benötigten Tools zur Erstellung (zum Beispiel Editoren oder Verifizierer) und Verwendung (zum Beispiel Generatoren oder Interpreter und angepasste Editoren) von *DSL*s.

2.1.1 VMTS Presentation Framwork

Visual Modeling and Transformation System (VMTS)[1] ist eine domänenspezifische Modellierungs- und Modelltransformationsumgebung die mit Hilfe der .NET-Technologie entwickelt wurde [VMT08]. Das System wird seit 2003 an der technischen Universität in Budapest entwickelt und kann für den akademischen Bereich kostenlos verwendet werden. *VMTS* unterstützt die graphische Modellierung von domänenspezifischen Modellen, die Repräsentation des Modells in einem angepassten Editor und das dynamische Verhalten des Modells.

[1] Die Dokumentation und das Framework ist unter der Adresse `http://www.aut.bme.hu/portal/Vmts.aspx`, letzter Abruf 02.11.2009, verfügbar.

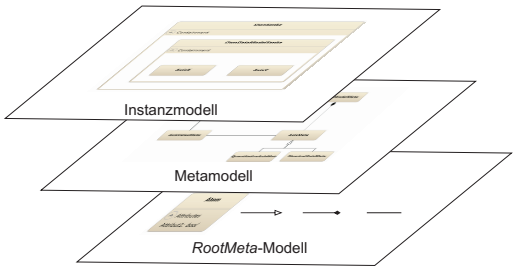


Abbildung 2.1: *N-Layer Hierarchie*

Die Modellierung in *VMTS* basiert auf unterschiedlichen Schichten (vgl. Abbildung 2.1), wobei eine Schicht (n) das Instanzmodell der darunterliegenden Schicht (n-1) ist [MMC07]. In der aktuellen Modellierungsschicht (n) können nur Modellelemente der Schicht n-1 zur Modellierung verwendet werden. Die Schicht n-1 wird Metamodell genannt, da sie ein Modell für ein Modell (die Schicht n) ist. Die unterste Modellschicht ist in *VMTS* immer das `RootMeta`-Modell. Hierdurch können Modellhierarchien, N-Layer Hierarchie genannt, entstehen, wie zum Beispiel eine Hierarchie bestehend aus `RootMeta`-Modell → Metaklassendiagramm → Klassendiagramm → Objektdiagramm. Das Metaklassendiagramm definiert die Modellelemente des Klassendiagramms (Klassen, Beziehungen, Attribute, Aggregationen, etc.) aufbauend auf dem `RootMeta`-Modell, während im Klassendiagramm diese Modellelemente verwendet werden um ein Diagramm für ein konkretes Programm zu entwickeln. Im Objektdiagramm werden anschließend die Instanzen dieser modellierten Klassen erstellt. Die maximale Tiefe der Hierarchie ist in *VMTS* nicht limitiert. Im Folgenden wird zur Einführung des Modellierungskonzeptes von *VMTS* und zur Beschreibung der Sprachkonzepte ein Metamodell für ein Beispielszenario modelliert. Das Sprachkonzept wird bei der Erstellung der Abstraktionsschicht verwendet. Es werden zunächst die Anforderungen an das zu entwickelnde Modell gestellt, bevor dieses im anschließenden Abschnitt mit Hilfe von *VMTS* umgesetzt wird.

2.1.2 Beispielszenario

Zur Einführung in die Modellierung mit *VMTS* wird ein Beispielprogramm modelliert. Die Anforderungen an dieses Programm aus der Domäne der Datenvisualisierung werden in diesem Abschnitt beschrieben. Sie sind angelehnt an ein späteres komplexeres Modell für Visualisierungen. Das Szenario besteht aus zwei Diagrammdatensätzen, die in einem Punktdiagramm, bestehend aus X- und Y-Achse, dargestellt werden sollen. Die darzustellenden Daten sind konstant (siehe Tabelle 2.1). Sie sind so gewählt, dass die Daten eine gemeinsame ordinale Achse besitzen, die Zeitachse.

	Zeitraum			
	2005	2006	2007	2008
Krankheitsfälle	20	40	80	20
Todesfälle	0	10	20	40

Tabelle 2.1: *Szenariodatensätze*

Das zu entwickelnde Modell soll diese Daten für ein X-Y-Diagramm bereitstellen. Zudem soll das Modell die Möglichkeit bieten, mehrere Diagramme miteinander zu verknüpfen, indem einzelne Punkte selektiert und diese im verknüpften Diagramm mit der gleichen Achse ebenso selektiert dargestellt

werden. Des Weiteren soll eine Selektion einzelner Punkte möglich sein, welches graphisch angezeigt wird. Es ergeben sich folgende funktionalen Anforderungen:

[FA.1] Datenhaltung: Das Modell muss in der Lage sein, Datensätze für ein X-Y-Diagramm zur Verfügung zu halten. Jeder Datensatz besteht aus einem ordinalen (dem Zeitraum) und quantitativen (Krankheitsfälle oder Todesfälle) Anteil.

[FA.2] Selektion: Einzelne Datensätze können selektiert werden. Die Information, welcher Datensatz selektiert ist, soll im Modell gehalten werden.

[FA.3] Achsen: Ein Punktdiagramm besteht aus zwei Achsen, die jeweils eine Beschriftung tragen. Die Information über die Beschriftungen der Achsen (Zeitraum, Krankheitsfälle oder Todesfälle) soll das Modell bereitstellen.

[FA.4] Verknüpfung: Durch die Verknüpfung zweier Diagramme wird gewährleistet, dass Selektionen in einem Diagramm auf die verknüpften Diagrammpunkte in einem anderen Diagramm übertragen werden. Das Modell muss gewährleisten, dass diese Verknüpfungsart möglich ist.

Diese Anforderungen werden im weiteren Verlauf Schrittweise in ein konkretes Modell mit dem Tool umgesetzt und darauf aufbauend die Sicht (das Punktdiagramm) für dieses Modell entwickelt.

2.1.3 Umsetzung des Beispielszenarios in VMTS

Ein Modell wird in *VMTS* graphisch durch Elemente und Knoten, angelehnt an die *Unified Modeling Language (UML)*-Notation gebildet. Alle Modellelemente können Attribute enthalten. Die Modellierung von Modellen wird in der Komponente Adaptive Modeler gekapselt. Mit Hilfe dieses Moduls ist der Zugriff auf alle Komponenten der *VMTS*-Architektur, wie sie in der Abbildung 2.2 dargestellt ist, möglich. Für die graphische Modellierung ist die VPF-Schicht [MML08] verantwortlich. Es stehen

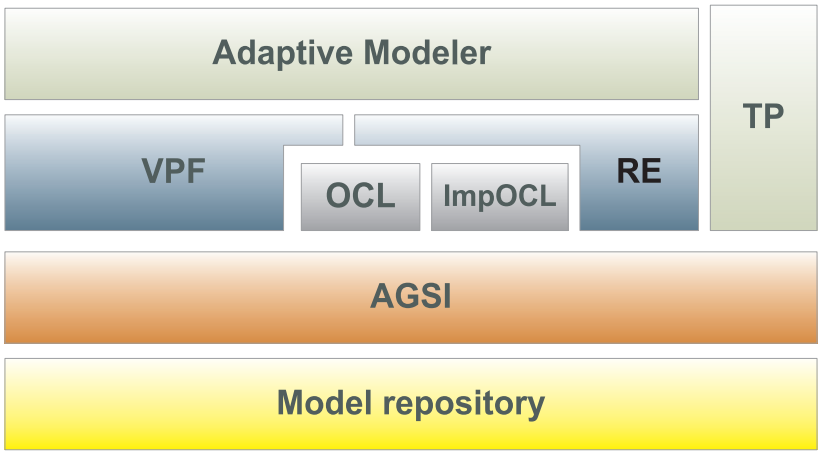


Abbildung 2.2: *VMTS Systemarchitektur [MMC09]*

aktuell zwei Versionen der VPF-Engine zur Verfügung. Eine Version mit allen Features, die allerdings instabil und nicht fehlerfrei ist [MLC05] und eine in C# und *Windows Presentation Foundation (WPF)* neuentwickelte Version [MML08], die jedoch noch nicht alle Features von *VMTS* auf die AGSI-Schicht

abbilden kann. Die AGSI-Ebene unterstützt die Hüllklassen zur Erstellung und Manipulation von Modellen, Modellelementen und Elementeigenschaften, die im Modellrepository persistent in einer Datenbank oder einem propreitären Format gespeichert werden können. Im Auslieferungszustand von *VMTS* ist das `RootMeta`-Modell im Repository hinterlegt. Auf diesem Modell basieren alle mit *VMTS* erstellten Modelle (vgl. Abbildung 2.1), auch dieses Szenariomodell und das Abstraktionsmodell, weshalb das Metamodel vorgestellt wird.

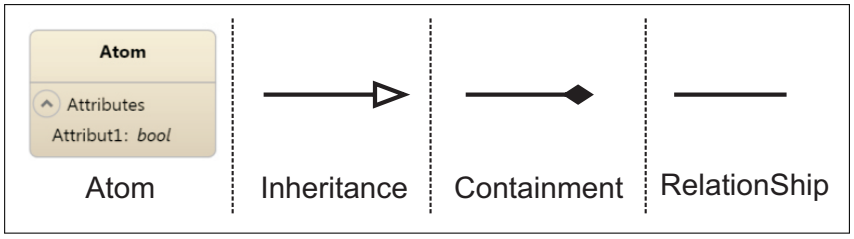


Abbildung 2.3: *Modellkomponenten des `RootMeta`-Modells*

Die Modellkomponenten, die das `RootMeta`-Modell zur Verfügung stellt, sind in der Abbildung 2.3 dargestellt. Das Metamodell stellt die Modellkomponente `Atom` zur Verfügung, zur Modellierung von Modellelementen (graphisch dargestellt als Box), und Modellkomponenten um Beziehungen zwischen Modellelementen abzubilden. Die graphischen Repräsentationen der Modellkomponenten sind an die *UML*-Notation in Klassendiagrammen angelehnt. Ein Modellelement wird durch eine Box (ähnlich einer Klasse) mit einem Attributbereich angezeigt, während Beziehungen als Verbindungslinie zwischen zwei Modellelementen dargestellt werden. Eine Beziehung ist eine Art von Verbindung zwischen den verbundenen Elementen. Es existieren drei Typen von Beziehungen in *VMTS*. Die `Inheritance` drückt eine Vererbungshierarchie zwischen Modellelementen aus. Die Eigenschaften von dem Basismodellelement werden zu den Eigenschaften der Kindelemente hinzugefügt. Die Beziehung wird durch einen Pfeil dargestellt. Die `Containment`-Beziehung, repräsentiert durch eine ausgefüllte Raute als Pfeilspitze, stellt eine „besteht aus“-Beziehung dar. Diese beiden speziellen Beziehungsarten können durch den allgemeineren Beziehungstyp `RelationShip` ausgedrückt werden. Mit dieser Modellkomponente ist es möglich, Verbindungen zwischen Elementen zu modellieren, die zur Kommunikation verwendet werden können. Die Darstellung ist der *UML*-Notation einer Assoziation gleich.

Modellelemente werden in einem Diagramm, ähnlich einem Klassendiagramm, bestehend Modellelementen mit den verschiedenen Relationen, dargestellt. Ein Modell kann aus mehreren Diagrammen bestehen, um das Modell bei hoher Komplexität in einzelne Teile gliedern zu können. Damit die Elemente in dem Diagramm leichter identifiziert werden können, ist es notwendig Namen für Modellelemente zu vergeben. Jedes Modellelement besitzt zwei Namensbereiche, die definiert werden müssen. Zum einen die Eigenschaft `Name`, die den Namen des Elementes im aktuellen Kontext angibt und den `InstanceName`, welcher für abgeleitete Modelle (vgl. Abbildung 2.1) den Namen dieses Elementes angibt.

Neben der Namenseigenschaft kann jede Modellkomponente in *VMTS* Attribute beinhalten. Attribute werden durch Felder, welche die genaueren Details von einem Attribut definieren, angelegt (vgl. Abbildung 2.4). Ein Attribut wird neben den Namen und dem Typ durch das Tag `Multiplicity` beschrieben. Dieses Tag enthält die minimale und maximale Anzahl von Attributinstanzen, zum Beispiel `1..1` oder `0..*`. Komplexe Attribute sind Kompositionen aus mehreren einzelnen Attributen, wobei diese die identischen Felder aufweisen wie einfache Attribute. Des Weiteren kann mit Hilfe

des Feldes `ReadOnly` eingeschränkt werden, dass ein Attribut nur gelesen, aber nicht geändert werden kann. Weitere Einschränkungen können in *VMTS* durch die Verwendung der *Object Constraint Language (OCL)*-Sprache [OMG09], für textuelle Einschränkungen im Modell, definiert werden. Unter Verwendung dieser Sprache ist es zum Beispiel möglich die Anzahl der eingehenden oder ausgehenden Kanten eines Modellelementes zu beschränken. *OCL* ist eine Modellierungssprache mit der es möglich ist Softwaremodule zu erstellen. Sie ist seit *UML* 2 Bestandteil der *UML*-Definition und wird hauptsächlich für textuelle Spezifikationen von Bedingungen in *UML*-Diagrammen verwendet [WK03]. Um Einschränkungen zu definieren, muss eine Methode und ein einschränkender Quellcode zur Überprüfung der Regel geschrieben werden. Als Sprache zur Definition der Regel kann die *OCL*-Syntax oder C#-Syntax verwendet werden. Zur Übersetzung der *OCL*-Syntax wird der Imperative *OCL*-Übersetzer verwendet (vgl. Abbildung 2.2).

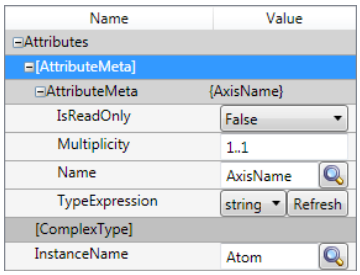


Abbildung 2.4: *Attribut-Definition in VMTS*

Anhand dieser Einführung in *VMTS* wird das in 2.1.2 beschriebene Szenario durch mehrere Modellelemente, dargestellt in der Abbildung 2.5, modelliert. Das zentrale Element ist `DiagrammMeta`. Es stellt ein Modell eines Punktdiagrammes dar. `DiagrammMeta` besteht aus einem `WertMeta`-Modellelement, wobei durch Vergabe entsprechender Multiplizitäten sichergestellt ist, dass ein Diagramm aus mehreren Werten bestehen kann. `WertMeta` stellt einen Datenpunkt im Diagramm dar. Ein Datenpunkt hat einen ordinalen und quantitativen Anteil (vgl. [FA.1]). Diese Eigenschaft wird durch zwei Attribute modelliert. Als weiteres Attribut ist `IstSelektiert` als boolsches Attribut zum Modellelement hinzugefügt worden, um die Anforderung zu erfüllen, dass die Information über eine Selektion eines Datenwertes im Modell abgebildet werden kann (vgl. [FA.2]). Des Weiteren besteht ein Punktdiagramm aus zwei Achsen (vgl. [FA.3]), die durch eine „besteht aus“- Beziehung mit dem Modellelement `AchseMeta` modelliert ist. Das `AchseMeta`-Element besitzt das Attribut `Achsenbeschriftung`, um die Anforderung zu erfüllen, dass jede Achse einen Namen trägt (vgl. [FA.3]). Achsen können vom Typ ordinal oder quantitativ sein. Diese Eigenschaft wird durch Vererbung von dem Element `AchseMeta` abgebildet. Das Modell ist speziell für ein Punktdiagramm mit zwei Achsen entwickelt, deshalb sind die Multiplizitäten der Beziehung entsprechend angepasst, wobei eine Achse mehreren Diagrammen zugeordnet sein kann (die Zeitachse), ein Diagramm allerdings aus zwei Achsen besteht. Damit ist das Modell für ein Punktdiagramm modelliert, allerdings kann der komplette Datensatz noch nicht gleichzeitig angezeigt werden, da dieser durch zwei Diagramme der beschriebenen Art dargestellt werden muss. Aus diesem Grund ist das Modellelement `SzenarioProgrammMeta` hinzugefügt worden, das in Beziehung mit dem Element `DiagrammMeta` steht und als Container für mehrere Diagramme dient.

Das erstellte Metamodell kann in *VMTS* direkt in C#-Code transformiert werden und in einer Anwendung eingebunden werden. Bevor die Generierung durchgeführt wird, findet eine Modellvalidierung mit Hilfe des *OCL*-Moduls (vgl. Abbildung 2.2) statt. Für die Transformation von Modellen ist das RE-Modul (vgl. Abbildung 2.2) in *VMTS* eingepflegt worden. Das RE-Modul kann verwendet werden um frei wählbare Transformationen zwischen Modellen zu beschreiben. Zurzeit ist ein Transformationsmodul zur Generierung von C#-Quellcode implementiert, dass zwei Klassenmodelle erstellt. Ein statisches Modell, bei dem keine Benachrichtigungen bei Datenänderungen gefeuert werden, und ein dynamisches Modell, welches auf Grundlage der Verwendung von Dependency-Properties implementierte Benachrichtigungen bietet. Diese Klassen werden zur Laufzeit nachgeladen. Weiterhin

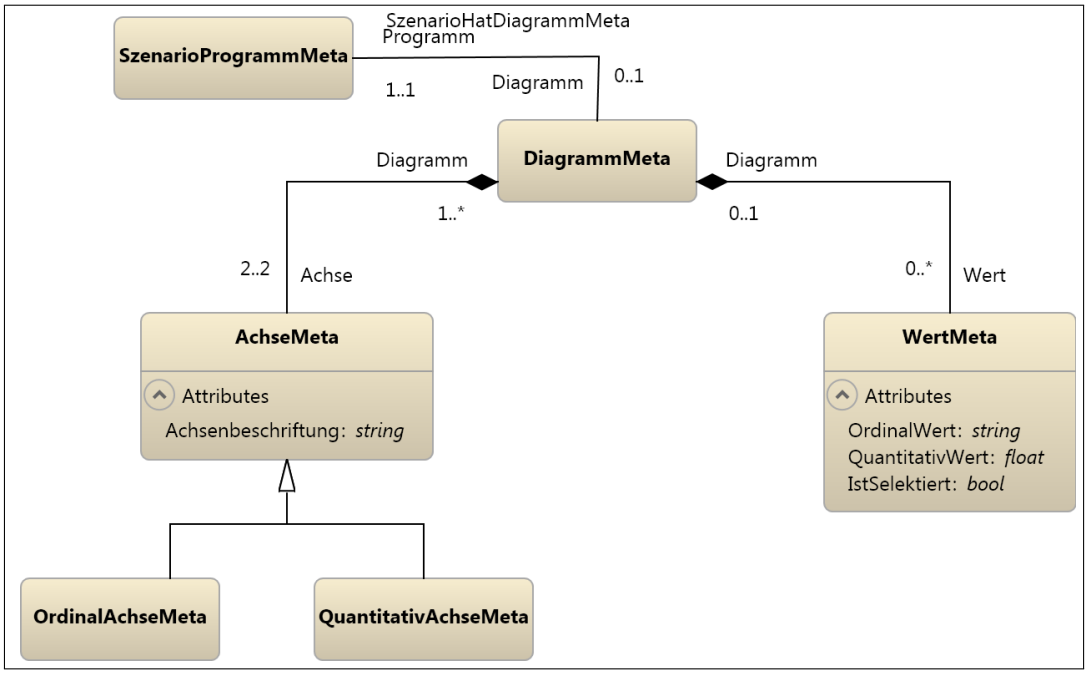


Abbildung 2.5: *Das Modell zum Beispielszenario*

stellt *VMTS* das Modell und alle Unterelemente als Interfaces zur Verfügung, wodurch ein Programmierer das Modell typkonform und effektiv verwenden kann. Durch diese zwei Möglichkeiten ein *VMTS*-Modell in ein System einzubinden, ist ein Systementwickler nicht an ein Darstellungsframework gebunden. Es bietet sich jedoch an, das dynamische Modell direkt als View-Model-Komponente einer *WPF*-Anwendung mit einer *Model-View-Viewmodel (MVVM)*-Architektur zu verwenden. Das entwickelte Beispielprogramm nutzt diese Möglichkeit. Als weitere Möglichkeit zum Zugriff auf ein Modell kann in *VMTS* das TP-Modul verwendet werden, wodurch auch externe Tools Zugang zum Modell erhalten.

Allerdings sind noch nicht alle Anforderungen an das Beispielprogramm erfüllt. Die Anforderung [FA.2] der Selektion eines Elementes erfordert, dass das Modell durch den Anwender beeinflusst werden kann, um das Attribut `IstSelektiert` zu ändern. Die Beeinflussung kann entweder direkt durch Abbildung von Benutzerereignissen und dementsprechenden Anpassungen von Modellkomponenten durchgeführt werden oder es wird das in *VMTS* integrierte Model-Transformation-Framework verwendet, mit dessen Hilfe eine graphische Beschreibung einer Transformation eines Modells möglich ist. Für das Beispielprogramm wird diese Möglichkeit gewählt, da durch die graphische Repräsentation und Umwandlung der Modellelemente ein höherer Abstraktionsgrad als die direkte Umsetzung in Quellcode erreicht wird. Die Selektion eines spezifizierten Modellelementes erfordert, wie jede Modelltransformation in *VMTS*, die Definition einer oder mehrerer Regeln und ein Kontrollflussmodell anhand spezieller Metamodelle. Regeln werden in *VMTS* als Muster angelegt. Wenn eine Regel angewendet wird, sucht das *VMTS*-System im Instanzmodell, auf dass die Transformation angewendet wird, nach diesem Muster und führt anschließend die in der Regel definierten Transformationen auf den Teil des Modells aus. Mit Hilfe einer Regel können Transformationen wie das Löschen und Erstellen von Knoten und Kanten, die Änderung von Attributen durch imperativen Quellcode und die Umlenkung von Referenzen auf andere Modellelemente definiert werden. Für das Beispielprogramm wird die

Änderung von Attributen, dem Attribut `IstSelektiert`, benötigt. Für die Änderung des Attributes sind die Regeln `Selektiere`, zur Selektion eines Elementes und `DeSelektiere`, zur Deselektion eines Elementes, erstellt worden. Das Kontrollflussmodell beschreibt, in welcher Abhängigkeit und Reihenfolge die Regeln ausgeführt werden. Es wird durch einen Graphen bestehend aus Kanten und Knoten definiert. Ein Knoten kann mehrere Vorgänger- und Folgeknoten haben. Hierdurch entstehen Kontrollflusspfade, die im Startknoten (dargestellt durch einen grünen Pfeil) beginnen und im Endknoten (dargestellt durch ein rotes Rechteck) enden. Welche der evtl. komplexen Pfade vom Start- zum Endknoten gewählt wird, kann durch Guards an den Kanten definiert werden. Ein Guard wird durch einen booleschen Ausdruck definiert. Im Beispielprogramm wird diese Technik verwendet, um unter anderem eine Unterscheidung zu erreichen, welche der beiden erstellten Regeln zur Änderung des Selektions-Attributes verwendet wird. Ist das Element bereits selektiert, wird der Pfad mit der Regel `DeSelektiere` gewählt. Für den Fall, dass das Element noch nicht selektiert ist, wird die Regel `Selektiere` angewandt. Eine Regel wird im Kontrollflussgraphen durch einen Knoten repräsentiert. Durch diesen beschriebenen Graphen und diese Regeln, wird eine Selektion eines Elementes im Punktdiagramm durchgeführt. Somit kann die Anforderung [FA.2], die Selektion eines Elementes, mit Hilfe von *VMTS* modelliert werden. Der Kontrollflussgraph ist mit der Darstellung der Guards an

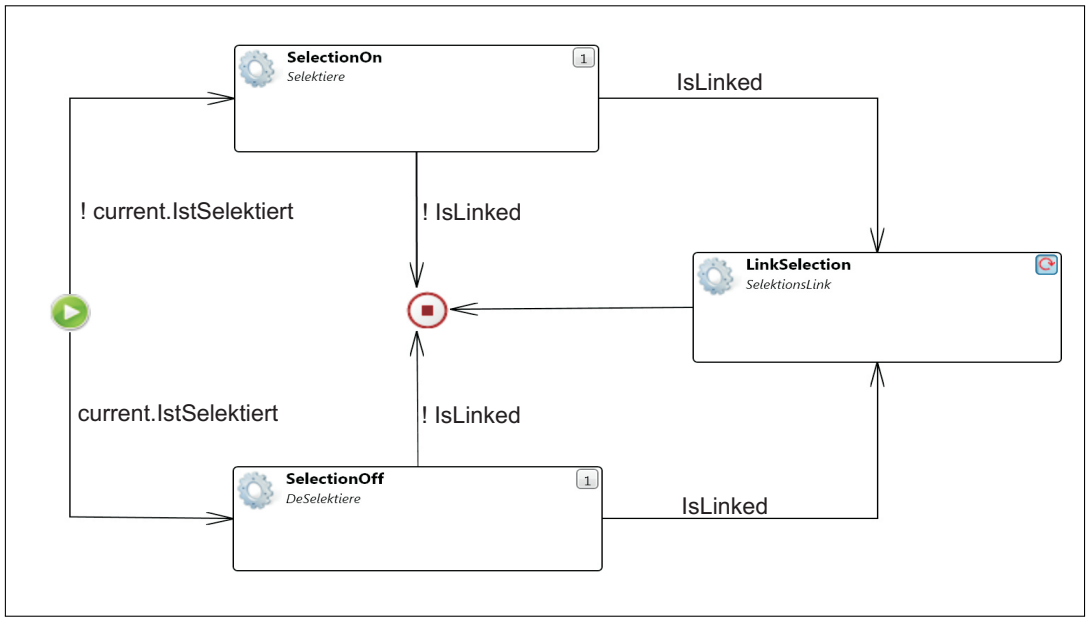


Abbildung 2.6: *Der Kontrollflussgraph des Beispielssystems*

den Kanten in der Abbildung 2.6 dargestellt. In dieser Abbildung ist der Graph dahingehend erweitert worden, dass die Anforderung [FA.4], die Verknüpfung der Diagramme, erfüllt wird. Dieses wird durch das Einfügen der Regel `SelektionsLink` erreicht. Die Regel beschreibt die Anpassung des Attributes `IstSelektiert` für Modellelemente dahingehend, dass dieses Attribut mit dem übergebenen Modellelement übereinstimmt. Die Ausführung der Regel ist in Abhängigkeit der Verknüpfung der Diagramme modelliert worden. Es sollen allerdings nicht alle im Instanzmodell enthaltenen Modellelemente selektiert werden, sondern nur Elemente mit der gleichen ordinalen Attributsausprägung (das gleiche Jahr). Diese Einschränkung kann durch Definition von Constraints in der Regel durchgeführt werden. Es ist zu beachten, dass evtl. mehrere Modellelemente die gleiche Ausprägung haben. Aus diesem Grund ist der Knoten der Regel `SelektionsLink` dahingehend modifiziert worden, dass

die Regel so lange ausgeführt wird, bis keine Modellelemente mit dem Muster der Regel vorhanden sind (in der Abbildung ein roter Pfeil im Knoten). Ansonsten würde die Regel nur einmalig ausgeführt werden.

Das entwickelte Beispielsystem mit den Daten aus dem Szenario (vgl. Tabelle 2.1) ist in der Abbildung 2.7 dargestellt. In blau werden selektierte Elemente dargestellt. Die Verknüpfung der Diagramme, kann durch eine Checkbox ein- bzw. ausgeschaltet werden.

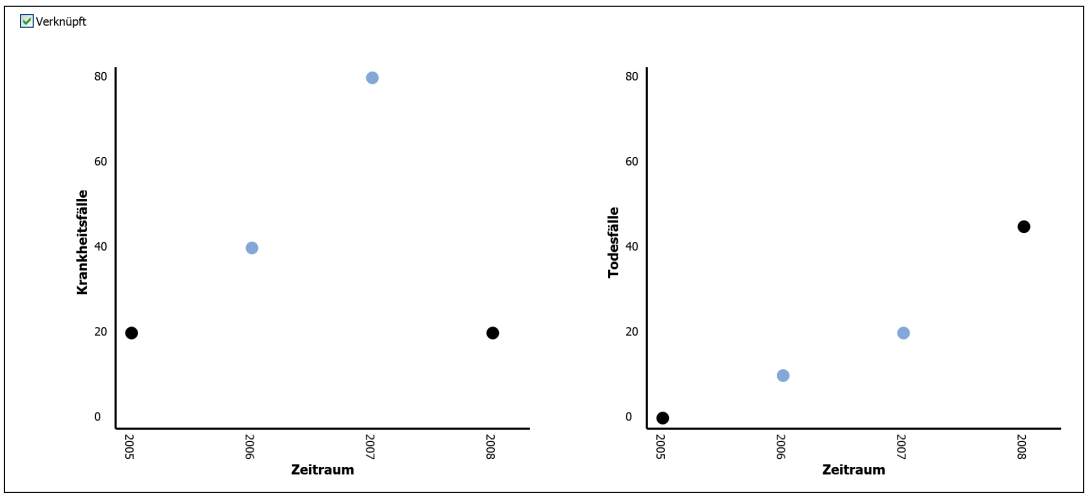


Abbildung 2.7: *Das Beispielsystem zum Szenario entwickelt mit Hilfe von VMTS*

Eine weitere wichtige Eigenschaft neben einem spezifischen Modell sind die Daten selbst und wie diese dem Analysten zur Verfügung gestellt werden. Im Bereich der analytischen Informationssysteme hat sich zu diesem Zweck das multidimensionale Modell etabliert. Die im Folgenden betrachte *MUSTANG*-Plattform bietet Dienste für den Zugriff auf solch ein Modell.

2.2 Die MUSTANG Plattform

MUSTANG ist eine geographisch, statistische Datenanalyseplattform zur Erstellung von Anwendungen zur Analyse von multidimensionalen Daten. In [MTA09] wird diese Plattform vorgestellt. Die Plattform ist im OFFIS[2] mit dem Analyseschwerpunkt epidemiologischer Datenbestände entstanden. Aus diesem Grund sind im System erweiterte statistische Verfahren eingebunden. Die von *MUSTANG* verarbeitete Datenmenge muss multidimensional aufbereitet sein. Die Daten können dabei einen geographischen Bezug besitzen, der zur räumlichen, statistischen Analyse herangezogen werden kann. Diese drei Eigenschaften (multidimensionale Daten, statistische Verfahren und Daten mit Geographiebezug) bilden die zentralen Komponenten von *MUSTANG*. Im Folgenden wird näher auf die Architektur und die Eigenschaften der verschiedenen Komponenten eingegangen.

[2] `http://www.offis.de/`, letzter Abruf 08.04.1010

2.2.1 Online Analytical Processing

MUSTANG ist eine Plattform, die zum Zugriff auf *OLAP*-Systeme verwendet wird. *OLAP* zählt zu den Methoden der analytischen Informationssysteme. Im Gegensatz zu klassischen transaktionellen Systemen steht die Analyse von großen Datenmengen im Vordergrund [CCS93]. Das Ziel ist die Unterstützung des Analysevorhabens durch die multidimensionale Betrachtung der zugrunde liegenden Daten. Hierzu werden unter dem Begriff *OLAP* verschiedene Anwendungen und Technologien zur Erfassung, Verwaltung, Verarbeitung und Darstellung multidimensionaler Daten für Analyse- und Managementzwecke zusammengefasst. Eine weit verbreitete Definition der Eigenschaften von *OLAP*-Systemen ist *Fast Analysis of Shared Multidimensional Information (FASMI)* [Pen08]:

Fast: Diese Eigenschaft bezieht sich auf die Reaktionsgeschwindigkeit des Systems. Ein *OLAP*-System führt Anfragen in der Regel innerhalb von einer Sekunde aus. Sehr große Anfragen können bis zu 20 Sekunden dauern.

Analysis: Ein *OLAP*-System kann nach der Definition von Pendse jede Art von Geschäftslogik und statistischen Daten zur Analyse für eine Anwendung verwalten. Darüber hinaus wird die Möglichkeit gegeben selbstdefinierte Berechnungen im Rahmen der Analyse durchzuführen.

Shared: Das System bietet ein ausreichendes Regelwerk zur Durchsetzung von Sicherheitsanforderungen an. Dieses ist für die Vertraulichkeit und, falls ein Schreibzugriff gegeben ist, für Sperren erforderlich.

Multdimensional: Dies ist die wichtigste Eigenschaft eines *OLAP*-Systems. Daten werden in *OLAP*-Systemen in einer multidimensionalen Struktur mit der Unterstützung von mehreren Hierarchien vorgehalten.

Information: Die im *OLAP*-System vorgehaltenen Daten und Metadaten stehen dem Anwender jederzeit zur Verfügung, unabhängig wie diese im System gespeichert sind. Hieraus resultierende Beschänkungen beeinflussen den Analysten nicht.

Zentrale Eigenschaft eines *OLAP*-Systems ist die zugrunde liegende multidimensionale Datenstruktur, genannt *Cube*. Ein Cube ist vergleichbar mit einer Tabelle in einer relationalen Datenbank, allerdings ist die Struktur nicht für eine effiziente Datenspeicherung sondern für eine effiziente Datenbeschaffung ausgelegt. Der Cube (vgl. Abbildung 2.8) wird durch Dimensionen aufgespannt (Kanten des Würfels) und besteht aus einzelnen Zellen. Eine *Dimension*, beispielsweise `Ort` oder `Datum`, beschreibt eine mögliche Sicht auf die assoziierte Kennzahl. Hierbei ist eine Dimension in einer baumartigen Struktur in einzelne *Ebenen* (engl. Level) untergliedert. Zum Beispiel die Dimension `Ort` kann in einem konkreten Modell in die Ebenen `Länder`, `Bundesländer` sowie `Regionen` untergliedert werden. Die konkreten Elemente in jeder Ebene, beispielsweise `Niedersachen` in der Ebene `Bundesländer`, sind Knoten (engl. Nodes). Diese Knoten repräsentieren die Koordinaten einer Zelle in einem Cube. Innerhalb der Baumstruktur einer Dimension können unterschiedliche Pfade zu einer Ebene bestehen. Zum Beispiel in der Dimension `Datum` kann die Ebene `Tag` durch den Pfad `Datum` → `Jahr` → `Monat` → `Tag` oder `Datum` → `Woche` → `Tag` erreicht werden. Diese unterschiedlichen Pfade werden im multidimensionalen Modell *Hierarchien* genannt. Ein Cube besteht aus mindestens einer Dimension, kann aber auch durch wesentlich mehr Dimensionen aufgespannt sein, wobei die gewählten Dimensionen orthogonal sein müssen. Das heißt, eine Dimension darf keine funktionale Abhängigkeit zu einer anderen haben. Ein Cube besteht neben den aufspannenden Dimensionen aus Zellen, welche die eigentlichen Daten, wie zum Beispiel `Einwohneranzahl`,

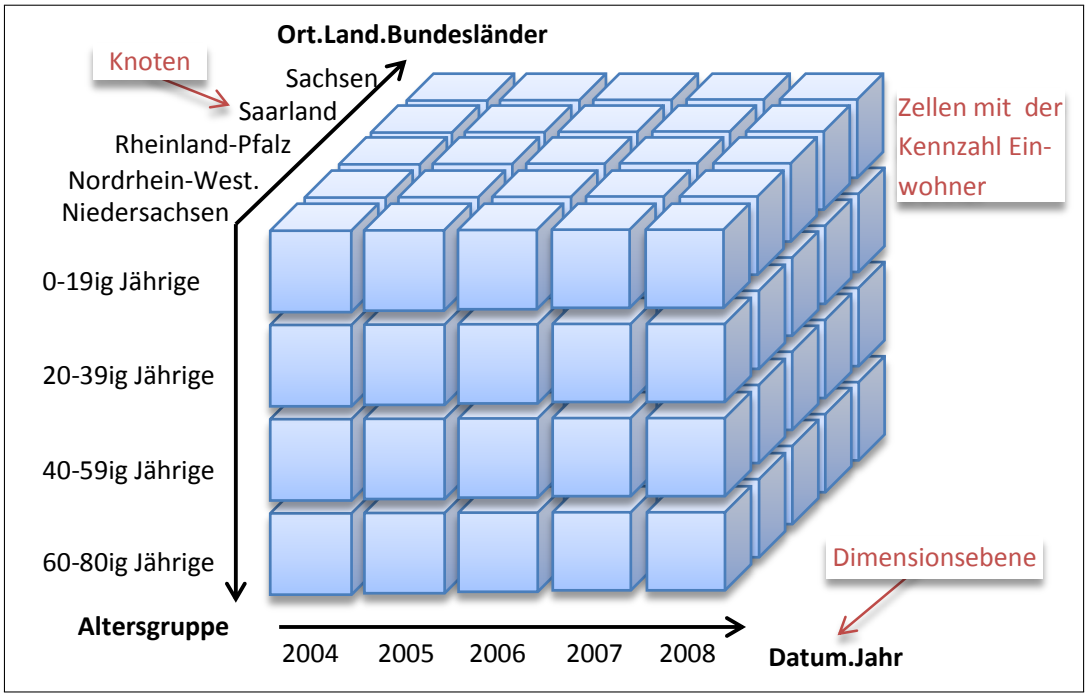


Abbildung 2.8: *Schematischer Cube eines OLAP-System*

beinhalten. Eine Zelle besteht aus einer oder mehreren Kennzahlen, die den Wert für die gewählte Ebene der Dimensionen enthält.

Auf Cubes sind eine Reihe von Operationen zur Unterstützung der Datenanalyse möglich [PJ01]. Diese werden vorgestellt und anhand von Beispielen verdeutlicht. Für die Beispiele wird der in der Abbildung 2.8 dargestellte Cube mit den Dimensionen `Ort`, `Datum` sowie `Altersgruppe` und der Kennzahl `Einwohneranzahl` verwendet.

Drill-Down und Roll-Up: Diese beiden Operationen ändern die verwendete Ebene einer Dimension. Die Dimensionalität des Cubes bleibt bei der Durchführung der Operationen erhalten. Bei der Roll-Up-Operation werden neue Informationen durch Aggregation der Daten erzeugt. Die Operation kann zum Beispiel verwendet werden um von der Quartalsebene eines Würfels auf die Jahresebene zu wechseln. Die Drill-Down-Operation ist die komplementäre Operation. Einem Anwender wird es somit ermöglicht, die aggregierten Daten im Detail zu betrachten.

Drill-Across: Die Drill-Across-Operation wird zum Wechseln des Cubes verwendet. Die Operation wird durchgeführt, um eine Navigation auf gleicher Hierarchieebene durchzuführen. Es werden also für jeden Knoten die jeweiligen Knoten einer anderen Hierarchie (Geschwister) der gleichen Dimension betrachtet und nicht die Kindknoten (wie beim Drilldown) oder den Elternknoten (wie beim Roll-Up). Diese Operation ist nur möglich, wenn es mindestens zwei Hierarchieebenen gibt, wie beispielsweise bei der Dimension Datum.

Slice und Dice: Slice- und Dice-Operationen ermöglichen die Bildung einer individuellen Sicht auf die Datenmenge. Die Slice-Abfrage kann mit dem Herausschneiden von „Scheiben“ aus dem Würfel verglichen werden. Die Scheiben werden durch die Auswahl eines Knotens, beispielsweise das Bundesland Niedersachsen, einer aufspannenden Dimension des Würfels bestimmt. Es wird somit

die Dimensionalität verringert. Ein Dice ist hingegen mit dem Herausschneiden eines Teilwürfels vergleichbar. Es wird für eine Zelle des Würfels für alle beteiligen Dimensionen ein Drill-Down durchgeführt. Somit bleibt die Dimensionalität erhalten, aber es ändern sich die Hierarchieobjekte. Diese Operation kann zum Beispiel verwendet werden, um für das Bundesland Niedersachsen mit der Altersgruppe der 20-39-jährigen und dem Jahr 2006 die Details zu betrachten.

Rotating: Mit Hilfe der Rotating-Operation, auch Pivotierung genannt, werden die Dimensionen des Würfels aus einer anderen Perspektive dargestellt. Die Daten werden somit anhand einer anderen Menge von Daten gruppiert, bzw. berechnet. Im Beispiel Cube kann diese Operation verwendet werden, um beispielsweise Datum und Ort zu tauschen. In die Berechnung der Einwohnerzahlen für eine Zelle fließen andere Knoten der beteiligten Dimensionen ein.

Drill-Down- und Roll-Up- Abfragen können mit Slice- und Dice- Abfragen kombiniert werden, wodurch die Kindknoten eines bestimmten Knotens bei der Kombination von Drill-Down- und Slice-Operation abgefragt werden können. Um diese Möglichkeiten in die *MUSTANG*-Plattform einzubinden, ist folgende Architektur gewählt worden.

2.2.2 Die Architektur von MUSTANG

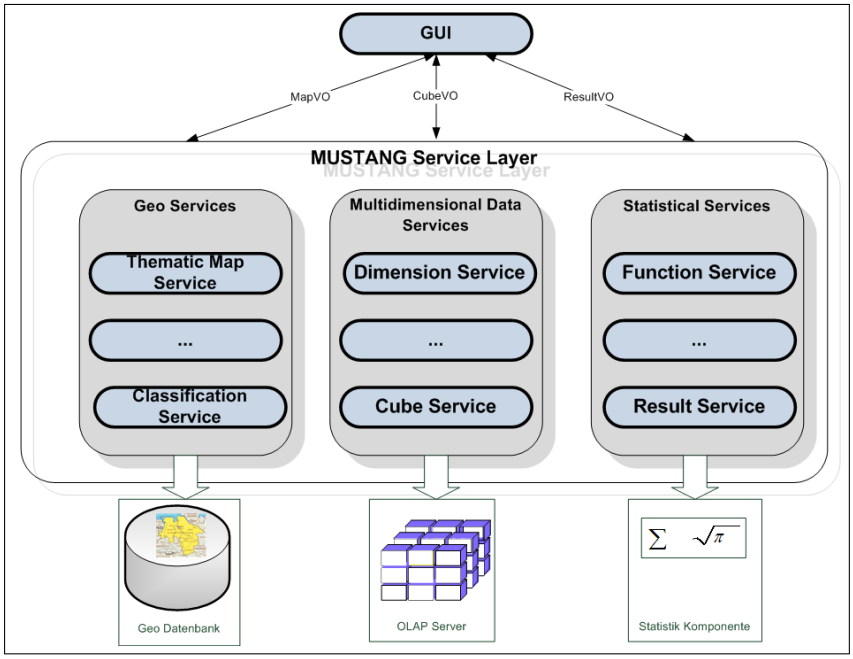


Abbildung 2.9: *Architektur der MUSTANG Services [FHTA09]*

Die *MUSTANG*-Plattform basiert auf einer rekonfigurierbaren serviceorientierten Architektur, wobei die Services zustandslos sind. Die als abstrakt konzipierten Services lassen sich zu Anwendungen zusammenfassen, wobei jedem Service ein Datentransferobjekt zugewiesen wird. Die Architektur mit den unterschiedlichen Schichten von *MUSTANG* ist in der Abbildung 2.9 abgebildet. Auf der untersten Ebene sind die Standardkomponenten Geo-Datenbank, *OLAP*-Server und Statistikkomponente angesiedelt. Mit Hilfe der darüber liegenden Serviceschicht ist ein einheitlicher Zugriff auf diese Komponenten möglich, wodurch auf die Fremdkomponenten transparent zugegriffen werden kann. Die

oberste Schicht bildet das *Graphical User Interface (GUI)*, das auf die *MUSTANG*-Services zugreift. Die *MUSTANG*-Services lassen sich zu drei voneinander abgrenzbaren Anwendungen zusammenfassen. Im Folgenden werden diese drei Anwendungen erläutert:

GEO Services: Mit Hilfe dieser Services wird der Umgang mit geographischen Daten realisiert. Zur Verfügung stehen ein Service zum Zugriff auf eine Geodatenbank, sowie ein thematischer Kartenservice zum Erzeugen von thematischen Karten. Als Geodatenbank kann zurzeit PostGIS integriert werden.

Multidimensional Data Services: Diese Services ermöglichen den Zugriff auf den *OLAP*-Server. Der eigentliche Zugriff auf den *OLAP*-Server wird mit Hilfe der Abfragesprache *XML for Analysis (XMLA)* durchgeführt, wodurch unterschiedliche Server angesprochen werden können. Unterstützte Standard-*OLAP*-Server sind zurzeit Microsoft Analysis Services und Mondrian. Aufgeteilt ist der Zugriff auf den *OLAP*-Server unter anderem in den Dimension-Service zur Abfrage der Dimensionen und deren Untergliederungen, dem Cube-Service zum Zugriff auf *OLAP*-Cubes sowie dem Summary-Attribute-Service zur Ermittlung kompatibler Kennzahlen.

Statistical-Services: Die Services zur statistischen Auswertung und Berechnung von *OLAP*-Cubes sind in diesem Bereich zusammengefasst. Es stehen komplexe, statistische Berechnungsmethoden zur Verfügung, deren Großteil auch über die Grenzen eines Cubes hinaus angewandt werden können. Im Function-Service sind die im System integrierten statistischen Verfahren gekapselt. Der Result-Service wird zum Abruf von *OLAP*-Cubes mit statistischen Kennzahlen verwendet. Die eigentlichen Berechnungen der statistischen Services werden an die Statistikkomponente R-Project[3] weitergeleitet.

Die Architektur ist so flexibel gehalten, dass weitere Standardkomponenten, wie zum Beispiel weitere GEO-Datenbanken oder andere *OLAP*-Server integriert werden können.

Die *MUSTANG*-Plattform abstrahiert Daten und Verfahren zum Zugriff auf *OLAP*-Server. Diese Abstraktion umfasst Dimensionen und Kennzahlen. Als weitere Eigenschaft können mit statistischen Maßzahlen verknüpfte Kennzahlen und Cubes berechnet werden, wobei der Übergang zwischen diesen Cubes für den Anwender der Plattform transparent ist. *MUSTANG* ermöglicht somit die Vernachlässigung der Aspekte Datenbeschaffung und Verarbeitung in einer Analyseanwendung, da dieses bereits durch die Plattform realisiert wird. [FHTA09]

2.3 Zusammenfassung

Die vorangegangen Kapitel haben beispielhaft aufgezeigt wie Systeme mit *VMTS* entwickelt werden. Ausgegangen von den Anforderungen an ein System, welche in diesem Fall durch ein Beispieldatenszenario entstanden sind, sind diese Anforderungen auf ein Modell abgebildet worden. Dieses Modell dient als Grundlage für die entwickelte Ansicht (in diesem Fall das Punktdiagramm). *VMTS* bietet des Weiteren durch das Model-Transformation-Framework die Möglichkeit Modelle anzupassen und in andere Modelle zu transformieren. Hierzu kann der Entwickler auf graphische Modellierungssprachen zurückgreifen, in denen er zunächst Regeln zur Transformation und anschließend einen Kontrollflussgraphen zur Festlegung der Reihenfolge der Ausführung von Regeln modelliert.

[3] Die Daten zum RProject können unter der URL `http://www.r-project.org/`, letzter Abruf 01.02.1010, abgerufen werden.

VMTS ermöglicht es, die Ziele der domänenspezifischen modellgetriebenen Entwicklung zu erreichen, indem Teilkomponenten eines Modells modelliert und anschließend diese Teilkomponenten zu einem Gesamtsystem verbunden werden können (N-Layer-Struktur). Metamodelle bilden die zentralen Elemente eines Modells, während Instanzmodelle diese Elemente verwenden, um ein konkretes System zu modellieren. Zusätzlich zur rein graphischen Modellierung besteht auch die Möglichkeit Randbedingungen, die nicht direkt modelliert werden können, mit Hilfe der *OCL*-Sprache abzubilden. Das Exportieren in C#-Code ermöglicht anschließend die Verwendung des Modells als View-Modell oder statisches Modell. Diese Eigenschaften von *VMTS* werden im Folgenden verwendet, um ein Modell für Visualisierungen in der visuellen Exploration zu modellieren.

Des Weiteren ist die *MUSTANG*-Plattform vorgestellt worden. Diese ermöglicht die Abstraktion von Abfragen einer multidimensionalen Datenbank. Solch eine Datenbank ist für die effektive Auswertung einer Datenmenge optimiert und nicht die Speicherung. Aus diesem Grund wird eine spezielle Datenstruktur, genannt Cube, zur Datenhaltung verwendet. Ein Cube wird durch Dimensionen aufgespannt und mit Hilfe von Kennzahlen werden den Zellen des Cubes Datenwerte zugewiesen. Die *MUSTANG*-Plattform ermöglicht neben dem Zugriff auf diese Datenstruktur ein *Geographic Information System (GIS)* einzubinden und mit Hilfe der Statistikkomponente statistische Eigenschaften, wie beispielsweise die Bevölkerungsanzahl, aus einem entsprechenden Cube zu kompensieren.

3 Modellierung von Visualisierungen und Interaktionen

Visualisierungen bilden die zentrale Komponente um einen Zugang zu den Daten zu ermöglichen. Um die Abstraktionsschicht zu bilden werden in diesem Abschnitt bestehende Visualisierungs- und Interaktionsmodelle bezüglich der Tauglichkeit als Abstraktionsmodell untersucht. Die Konzepte werden jeweils an Beispielsystemen dargestellt. Zunächst wird ein Überblick zu Visualisierungsmodellen gegeben und anschließend zu Interaktionsmodellen. Am Seitenrand sind die wichtigsten Konzepte notiert.

3.1 Visualisierungsmodelle

Die Wahl eines Modells beeinflusst den Datenzugriff, die möglichen Datentransformationen, die Anzahl der Metadaten und die Eigenschaft der Modularisierung des Systems [TSB04]. In der Literatur werden verschiedene Modelle vorgestellt. Ein weit verbreitetes Datenmodell ist das relationale Modell. In [TSB04] wird die Möglichkeit diskutiert dieses auch für analytische Systeme einzuführen. Das relationale Modell ist nach [TSB04] einfach und flexibel für das Abbilden von Daten auf Visualisierungen, wie zum Beispiel Balkendiagramme, Punktdiagramme oder Liniendiagramme. Das System Snap-Together [NS00] baut auf dem relationalen Datenbankmodell auf. Snap-Together ermöglicht es Visualisierungen dynamisch zu mischen und abzugleichen, um eine eigene Explorationsschnittstelle zu erstellen, ohne diese programmieren zu müssen. Hierzu muss ein Anwender vorher die Visualisierungen, die Daten aus einer relationalen Datenbank und die integrierten Abgleichsoperatoren koordinieren. Die Koordination der Visualisierung und der Daten erfolgt, indem eine oder mehrere Visualisierungstypen für eine relationale Tabelle gewählt werden. Zwischen den gewählten Visualisierungen bestehen in Snap-Together somit die gleichen Beziehungen, wie im relationalen Schema. Die Beziehungstypen können dabei one-to-one, one-to-many, many-to-many oder keine sein. Bei einer one-to-one-Beziehung werden Benutzeraktionen auf einem Datentupel einer Ansicht automatisch an das in Beziehung stehende Tupel weitergeleitet. Diese Beziehung wird durch die Primärschlüssel des relationalen Datenbankmodells identifiziert. Besteht eine Beziehung zwischen einem Primärschlüssel und einem Fremdschlüssel in der relationalen Datenbank wird diese Beziehung one-to-many genannt. Die Beziehung weist auf eine hierarchische Beziehung zwischen den Daten hin. Eine Beziehung zwischen Fremdschlüsseln, kann durch zwei one-to-many-Beziehungen ausgedrückt werden. Außerdem kann es sein, dass zwischen den relationalen Tabellen keine Beziehungen existieren. Nachdem die Beziehungen zwischen den Visualisierungen gewählt worden sind, müssen noch die Typen der Abgleichsoperatoren ausgewählt werden. Benutzer können die folgenden Abgleichsoperatoren, basierend auf den Zusammenhängen in dem relationalen Datenbankschema, erstellen:

relationales Modell

- brushing-and-linking: Wird ein Element in einer Visualisierung selektiert, wird das übereinstimmende Element in anderen Visualisierungen ebenso selektiert.
- overview and detail: Wird ein Element in einer Visualisierung selektiert, navigiert die andere Visualisierung zu diesem Element und zeigt das Element ebenfalls an.
- drill-down: Durch Selektion eines Elementes in einer Visualisierung, werden die dazugehörigen Elemente in der anderen Visualisierung dargestellt. Es werden somit die Details dargestellt.
- synchronized scrolling: Das scrollen in einer Visualisierung verursacht ein scrollen in der verknüpften Visualisierung.

- details on demand: Durch Selektion eines Elementes in einer Visualisierung werden Details dieses Elementes in einer angrenzenden textuellen Anzeige dargestellt.

Durch vorherige Definition von Beziehungen und Abgleichsoperatoren kann das relationale Modell auch für mehrere – eventuell unterschiedliche – Sichten auf Daten verwendet werden. Allerdings ist ein Nachteil bei einem relationalen Modell, dass bestimmte Datenanalysen schwer durchzuführen sind. Während räumliche oder dynamische Abfragen, wie zum Beispiel Filtern, mit Hilfe von speziellen Indizes oder Datenstrukturen durchgeführt werden können, existiert ein Typ von Interaktionen, der im relationalen Modell schwierig umzusetzen ist [TSB04]. Dieser Typ ist die Umgestaltung, wie zum Beispiel die Umsetzung einer relationalen Repräsentation in eine korrelierende Matrix. Das Problem wird durch die Betrachtung des folgenden Beispiels verdeutlicht: Angenommen es soll eine Gendatenbank, bestehend aus verschiedenen Sätzen von Genen, analysiert werden. Eine Möglichkeit der Strukturierung der Daten besteht darin, für jeden Gentyp eine Spalte und für jeden einzelnen Datensatz eine Zeile im relationalen Modell zu verwenden. Die Struktur ist kompakt, aber beeinträchtigt in der Analyse, da nur Tupel (Zeilen) gruppiert und gefiltert werden können, allerdings nicht die Spalten. Um dieses Problem für Visualisierungen lösen zu können muss das Modell eine Unterstützung von Datentransformationen bieten. Solche Modelle werden *Visualization-Transform-Model (VTM)* genannt [TSB04].

*VTM*s beschreiben den Typ der Visualisierung der angezeigt wird und bilden somit eine Funktion, welche die Daten auf den Bildschirm abbildet. Jeder Visualisierungstyp hat ein unterliegendes *VTM*, auch wenn dieses nicht explizit als Modell ausgeführt ist [JK03]. Ein *VTM* hat die folgend aufgelisteten drei Eigenschaften:

- Die Visualisierungstransformation, die die Abbildung der Daten auf die Ergebnisse bestimmt.
- Die Parameter, welche die Visualisierungstransformation beeinflussen.
- Den Ergebnistyp, der durch die Visualisierungstransformation generiert wird.

Data-Flow-Modell

Jedes der folgenden Modelle charakterisiert diese drei Komponenten unterschiedlich. *Data-Flow*-Modelle , eingeführt von Haber und McNabb [HM90], sind die am weitesten verbreiteten Transformationsmodelle in wissenschaftlichen Visualisierungsanwendungen [JK03]. Dieses Modell betrachtet die Visualisierungstransformation als eine Pipeline, in der jede Stufe eine Transformation darstellt (vgl. Abbildung 3.1). Die Kombination der Transformationen bildet ein Netzwerk über den Datenfluss. Ein System, dass dieses Modell verwendet, ist VisIt [CBB+05], welches verwendet wird um große Simulationen mit bis zu mehreren Milliarden von Datenelementen zu visualisieren. Auf Grundlage der großen Datenmenge sind Optimierungen unabdingbar. Allerdings ist der Satz von anwendbaren Optimierungen abhängig von der Operatormenge. Erschwerend zu dieser Problematik ist die Anforderung an die Erweiterungsfähigkeit durch Plugins, die es erlaubt neue Komponenten einzubinden, wodurch die Komplexibilität zur Bestimmung von Optimierungen weiter erhöht wird. Komponenten können in VisIt Filter, Quellen oder Senken sein. Filter haben als Ein- und Ausgaben Datenobjekte, während Quellen nur Ausgabeobjekte und Senken nur Eingabeobjekte haben können. Jede Komponente kann die Menge der durchführbaren Optimierungen modifizieren. Am Ende der Pipeline (von der Quelle über evtl. mehrere Filter bis zur Senke) steht dann für eine spezifizierte Anordnung von Komponenten eine angepasste Menge an Optimierungen für einen Datensatz zur Verfügung. Da die Anzahl der nacheinander geschalteten Filter nicht begrenzt ist, können Pipelines sehr komplex werden. Im System VisMashup [SLA+09] können diese deshalb für die Sicht des Anwenders zusammengefasst und gegebenenfalls auch nicht benötigte Attribute abstrahiert werden. VisMashup ist ein Framework zur Ra-

tionalisierung der Erstellung von angepassten Visualisierungsanwendungen. Weil diese Applikationen für sehr spezielle Aufgaben erstellt werden können, verstecken sie viel von der Komplexibilität einer Visualisierungsspezifikation und machen es damit einfacher für Benutzer, Visualisierungen zu erstellen, indem ein kleiner Satz von Parametern verändert wird. Hierdurch können auch Benutzer mit keinen oder nur geringen Kenntnissen von Visualisierungstechnik und oder wenig Programmiererfahrung Visualisierungsapplikationen erstellen. Das Datenmodell von VisMashup basiert auf einem angepassten Data-Flow-Modell von [LP95]. Im Gegensatz zum Data-Flow-Modell von Haber und McNabb kann ein Knotenelement mehrere Vorgängerknoten besitzen (bei Haber und McNabb nur einen), wodurch die Pipeline nicht mehr aus einem linearen Verlauf besteht, sondern ein azyklischer Graph ist. Wenn eine Verbindung vom Knoten 'm' zum Knoten 'n' existiert, erfordert 'n' die vorherige Ausführung von 'm'. Die Verbindung zwischen den Knoten ist eine Datenabhängigkeit. Knoten können Parameter als Eingaben haben. Die Kombination von unterschiedlichen Pipelines wird „medley" genannt. Eine Kombination ist zum Beispiel möglich, indem eine Variable mit einer anderen Variablen in einer anderen Pipeline synchronisiert wird. Dies führt dazu, dass beide Variablen den gleichen Wert haben, wodurch beispielsweise Vergleiche leichter realisiert werden können. Eine weitere Möglichkeit der Synchronisation ist die Komposition zweier oder mehrerer Pipelines. Dieses ist möglich indem die zu synchronisierenden Pipelines als Eingabe für eine weitere Pipeline dienen. Das System DataMeadow [EST08] nutzt das Data-Flow-Modell nicht nur als internes Modell, sondern zeigt es dem Anwender direkt an. DataMeadow ist ein System zur dynamischen Änderung von Abfragen und Filtern auf multidimensionalen Datensätzen. Die graphischen Elemente werden auf einer unendlich großen, zweidimensionalen, virtuellen Fläche dargestellt. Diese stellt einen Container für visuelle, analytische Elemente dar. Der Container enthält zusätzlich Metadaten zu den einzelnen Elementen und den verfügbaren Datendimensionen und deren Daten. In DataMeadow werden drei unterschiedliche Typen von visuellen Elementen verwendet: Quellen (Datenbankleser oder Zahlengeneratoren), Senken (Betrachter und Beschriftungen) und Transformer (Ein- und Ausgabeelemente), welche die eingehenden Daten mit Operationen transformieren und ausgeben. Die gerichteten Verbindungen zwischen Elementen werden Abhängigkeiten genannt. Daten werden durch die Verbindungen von der Quelle zum Ziel geleitet, allerdings nicht verändert. Ändert sich die Quelle, ändern sich ebenfalls alle nachgelagerten Ziele ohne Einfluss des Benutzers.

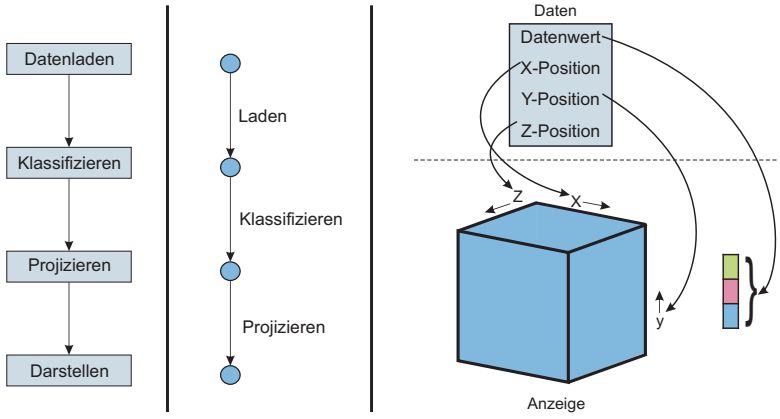


Abbildung 3.1: *Links:Data-Flow Modell; Mitte: Data-State Modell; Rechts: direktes Mapping*

Im Data-Flow-Modell werden Transformationen durch die Knoten durchgeführt. Im Gegensatz hierzu steht das *Data-State*-Modell [CR98], das sich auf die Transformation von Daten durch die

Data-State-Modell

Visualisierungspipeline konzentriert. Wie bei Data-Flow-Modellen wird ein Netzwerk verwendet um die Visualisierungstransformation zu beschreiben (vgl. Abbildung 3.1). In einem Data-State-Netzwerk repräsentieren die Knoten einen Datenstatus und Kanten Operationen auf den Daten. Ein System welches beispielsweise dieses Modell verwendet ist Lark [TIC09], ein System zur Erleichterung der Koordination von Interaktionen auf Informationsvisualisierungen bei geteilten digitalen Arbeitsplätzen. Jeder Knoten speichert die Daten, verändert diese allerdings nicht, so dass bestimmte Interaktionen, wie zum Beispiel ein Zoom, keinen Einfluss auf die Daten in anderen Sichten haben. Interaktionen des Analysten beeinflussen in der Regel bestimmte Parameter in der Visualisierungspipeline. In [JKMG07] wird das Data-State-Modell um Informationen über diese Parameter erweitert. Aufbauend auf einen spezialisierten Ansatz in [JKMG02] wird das Modell verallgemeinert und formal beschrieben. Das Modell besteht aus einer Visualisierungstransformation als eine Funktion $t : D_0 \times \ldots \times D_n \times P_0 \times \ldots \times P_m \mapsto R$, welche das Abbilden von Werten (die Datensatztypen D_0 bis D_n) zur Ansicht (das Ergebnis der Visualisierungstransformation R) definiert. P_0 bis P_m sind die Parameter der Transformation.

Direktes Mapping

Im Darstellungsmodell, dass von dem System VisAD [HDP94, HDP93, Hib98] verwendet wird, werden die Datenattribute direkt auf graphische Elemente abgebildet (vgl. Abbildung 3.1) . Die Visualisierung erfolgt mit Hilfe einer Funktion zur Abbildung der Eigenschaften. Ebenso wie im System VQE/Visage [DKR97] werden die visuellen Eigenschaften der Visualisierungen, wie zum Beispiel Farbe und Größe eines Punktes, in Objekten codiert, die anschließend dargestellt werden. Ein Objekt kann von verschiedenen Visualisierungen unterschiedlich angezeigt werden. Beide Systeme verwenden damit ein Modell, welches sowohl die Daten als auch die Darstellungen beschreibt. Damit ist das Modell gleichzeitig ein *VTM* (Abbildung von Werten auf die Sicht) wie auch ein Visualisierungsdatenmodell (die Struktur der Daten). Ein Visualisierungsdatenmodell ist nach [JK03] ein Modell, welches die Datentypen, Transformationsparameter und Transformationsergebnisse bereitstellt. Für Dokumentationszwecke können einfache Beschriftungen der Typen genügen. Für Transformationsergebnisse könnte die Beschriftung zum Beispiel ein *Multipurpose Internet Mail Extension (MIME)*-Typ [FB96c] sein. Jedoch ist zur Beschreibung der Daten, Parameter und generierten Ergebnisse während des Visualisierungsprozesses ein robusteres Datenmodell erforderlich [JK03]. Es sind Datenmodelle für wissenschaftliche Visualisierungen und Informationsvisualisierungen entwickelt worden, allerdings sind diese häufig nicht verwendbar für Transformationsparameter und -ergebnisse.

Geografisches Modell

In [Men00] wird ein Modell speziell für die Repräsentation von geographischen Phänomenen in *GIS*-Datenbanken vorgestellt. Ausgehend von dem Mangel bisheriger für geographische Daten entwickelter Modelle, Informationen in einer natürlichen Art und Weise zu repräsentieren [GYC07], ist ein Modell entwickelt worden. Dieses integriert die Prinzipien der Wahrnehmung in die geographische Datenbankrepräsentation. Hierbei wird nicht versucht jeden bekannten Wahrnehmungsaspekt in der Visualisierung zu imitieren, sondern die für Mensch-Maschinen-Interaktionen wichtigen Eigenschaften abzubilden und dadurch gleichzeitig eine höhere Recheneffizienz zu erreichen. Das Modell fördert durch seinen Aufbau die Exploration von neuen Daten durch schnelle Identifizierung von Mustern und Anomalien. Es besteht aus zum einen aus der Datenkomponente, welches das beobachtete Objekt, Zeit und Themendaten umfasst und zum anderen aus der Wissenskomponente, die kategorische Kriterien und die Beziehungen zwischen den Daten beinhaltet. Die Datenkomponente bietet die Daten immer unverändert im nicht interpretierten Zustand an. Um die Kategoriekriterien festzulegen, kann der Analyst sowohl einen top-down-, bottom-up- oder einen gemischten Ansatz verfolgen. Im top-down-Ansatz erstellt ein Datenexperte Klassen mit vorbestimmten Zugehörigkeitskriterien, bevor die Objekte innerhalb jeder Klasse definiert werden. In diesem Klassifikationsansatz ist ein vorheriges Expertenwissen von der modellierten Domäne notwendig. Der bottom-up-Ansatz ist explorativer und ist

daher unter anderem für unbekannte Zusammenhänge in Daten geeignet. Hier können Clustertechniken oder induktive Klassifikationsmethoden verwendet werden, um Klassen zu entwickeln und Objekte des Datenraumes zu bestimmen. Anstatt der Integration von Expertenwissen zielt dieser Ansatz darauf ab, Wissensentdeckung zu vereinfachen indem a priori Annahmen über die Natur der Daten minimiert werden.

3.2 Interaktionsabstraktion

Ein wesentlicher Schritt bei der visuellen, explorativen Analyse ist die Kommunikation des Analysten mit dem Visualisierungssystem. Hierzu werden Visualisierungsschnittstellen verwendet, die es dem Nutzer erlauben Interaktionen mit Visualisierungen durchzuführen. Folglich ist es wichtig zu verstehen, wie ein Anwender mit einem System interagieren kann [JKMG07]. Hierbei muss nicht nur die Abbildung einer Benutzerinteraktion auf eine einzige Visualisierung betrachtet werden, sondern der gesamte Einfluss einer bestimmten Interaktion auf das System, wie an folgendem Beispiel verdeutlicht wird [CR98]. Beispielsweise wird ein Quelldatensatz von zwei unterschiedlichen Visualisierungen gleichzeitig dargestellt. In einer Visualisierung werden die Daten in einem Scatterplot dargestellt, während sie in einer Anderen mit Hilfe einer sortierten Tabelle präsentiert werden. Es stellt sich nun die Frage, was passieren soll, wenn der Benutzer durch eine Interaktion im Scatterplot die Daten filtert. Eine mögliche Interpretation ist, dass die Tabelle eine völlig unabhängige Sicht der Originaldaten ist und sich deshalb nicht anpassen sollte. Eine andere Interpretation ist, dass die original Datenquelle bei der Filterinteraktion modifiziert wird, was bedeutet, dass die Tabelle ihre Ansicht anpassen muss. Es ist somit ein Vermittlerwerk, welches diese Problematik auflöst, notwendig [CR98]. In der aktuellen Forschung werden zwei – nicht unbedingt gegensätzliche – Ansätze für Visualisierungsschnittstellen verfolgt: Zum einen syntaktische Ansätze, in denen untersucht wird, wie Benutzerschnittstellenereignisse Visualisierungen modifizieren, und zum anderen semantische Ansätze, in denen versucht wird, die Ziele die ein Anwender bei der Interaktion mit einem Benutzungsschnittstellenelement hat, zu erkennen und entsprechend durchzuführen [JK03]. Diese beiden Ansätze werden im Folgenden vorgestellt.

syntaktische Modelle

Syntaktisches modellieren von Benutzerinteraktionen fällt in den Bereich der Mensch-Maschinen-Interaktion (HCI). Dieser Forschungsbereich ist allgemein für alle Mensch-Maschinen-Interaktionen zuständig und nicht nur für Visualisierungsschnittstellen. In [HR00] wird ein Überblick über verschiedene Benutzungsschnittstellenereignisse und ihre Effekte gegeben. Diese Modelle fokussieren auf Interaktionen mit spezifizierten Benutzungsschnittstellenelementen (zum Beispiel Buttons oder Scrollbars) und wie sie auf Systemfunktionen abgebildet werden können. Das System Rivet [BST+00, TSB04], welches ursprünglich ein System zur Analyse von komplexen Computersystemen war und anschließend für allgemeine Informationsdatenanalysen erweitert wurde, enthält ein syntaktisches Interaktionsmodell. Rivet nutzt ein Eventmodell basierend auf Objekten, die Events abfeuern, wenn ein Benutzer eine Aktion ausführt, wie zum Beispiel einen Mausklick. Um diese Funktionalität zu gewährleisten verwendet Rivet Skripte, die bei der Erstellung einer Visualisierung zusätzlich zu entwickeln sind. Ein Skript beschreibt, wie Daten importiert werden, wie Daten in visuellen Objekten abgebildet werden und wie Interaktionen des Nutzers auf Events abgebildet werden. Aktionen die im Skript festgelegt werden, können aus einer beliebigen Folge von Operationen bestehen. Ein Skriptentwickler kann somit selbst entscheiden, ob zum Beispiel ein Zoomereignis einen optischen (Vergrößerung) oder semantischen (zum Beispiel eine Verfeinerung der Daten) Zoom auslöst. Events werden durch ein Overlay für jede Eingabeform, zum Beispiel Maus oder Tastaturereignisse abgefangen, die daraufhin

entsprechende Systemereignisse auslösen. Anschließend werden diese Systemereignisse mit Hilfe des Skriptes auf Systemfunktionen abgebildet.

semantische Modelle

Allerdings bilden syntaktische Modelle nicht die Absichten des Benutzers – die Semantik einer Interaktion – ab. Semantischen Modellen versuchen diese abzubilden. Um zu erkennen, welche Absicht der Benutzer mit einer bestimmten Interaktion verfolgt, gibt es verschiedene Ansätze. Für das GADGET System [FTIN97, FFIT00] ist die Erkenntnis über die Semantik einer Interaktion entwickelt worden, indem vorher ein Klassifizierungsansatz für Problemstellungen [WL90] und ein aufgabenorientierter Klassifizierungsansatz von Shneiderman [Shn96] untersucht wurde. Das GADGET System ist ein Informationsvisualisierungssystem, welches automatisch interaktive Visualisierungen basierend auf der Benutzerabsicht erstellen kann. Dieses System verwendet ein Modell, das nicht an Interaktionen des Benutzers mit Visualisierungen gebunden ist [JKMG07]. Chuah und Eick [CR96] verfolgen einen anderen Ansatz für ein semantisches Modell. Sie zerlegen eine Visualisierungstransformation in atomare Einheiten, *Basic Visualization Interactions (BVI)* genannt. Eine *BVI* besteht aus einem visuellen Kontrollelement (eine Benutzungsschnittstelle für den Anwender) und Methoden zur Modifikation der Visualisierung. Der Anwender interagiert mit *BVI*s um Visualisierungen zu kontrollieren, während die *BVI*s Parameter beeinflussen. Dieses Modell stellt die Wichtigkeit von Visualisierungsparametern und die Beeinflussung dieser Parameter im Analyseprozess heraus. In [Rhe02] werden zwei Typen von Parametermanipulation identifiziert: Interaktive Parameterkontrolle und dynamische Manipulation. Bei einer interaktiven Manipulation von Parameterwerten wird die Ansicht nicht interaktiv geändert. Das Ergebnis der Parameteränderung wird nur auf Benutzeranfrage erstellt. Im Gegensatz hierzu werden dynamische Manipulationen in einem kontinuierlichen Intervall durchgeführt und direkt an die Visualisierung propagiert. Allerdings können Parameter auch indirekt durch eine Funktionsabstammung beeinflusst werden [JK03]. Funktionsabstammungen sind abgeleitete Parameter durch eine Art von Benutzeroperator, wie beispielsweise in Image Graph [Ma99] und im Parallel Coordinate Interface [TPM05], oder durch eine Funktion, wie zum Beispiel in VisSheet [JKM02]).

In der Arbeit von Chuah und Eick wird allerdings nicht verdeutlicht, wie die Parameter gewechselt werden können. Dies ist eine wichtige Eigenschaft in der visuellen, explorativen Analyse [JKMG07]. Dieses Problem wird unter anderem in [CR98] erkannt. Als Lösung wird vorgeschlagen ein Datentransformationsprozessmodell, wie beispielsweise eine Visualisierungspipeline, und ein Modell für Datenzustände zu verwenden. Ausgehend von der Erkenntnis der Existenz von domänenabhängigen und -unabhängigen Operationen wird ein Modell vorgestellt, welches beschreibt wie Graphiken, Daten und der Kontrollfluss von Operationen im Programm beeinflusst werden. Als Operationen werden direkte oder indirekte Manipulationen und Interaktionen verstanden. Zur Realisierung des Modells sind Operationen in Sicht- und Wertoperatoren untergliedert worden. Diese Unterscheidung ist darin begründet, dass hinsichtlich der Implementation große Unterschiede bei diesen Operationen bestehen. Wertoperationen beeinflussen die Datenquelle durch einen Prozess, wie beispielsweise Addition oder Löschung von Datensätzen, Filterung oder Modifizierung von Rohdaten. Eine Wertoperation generiert im Grunde einen neuen Datensatz. Eine Sichtoperation wechselt nur die Visualisierungsansicht, zum Beispiel durch Rotation, Translation oder Zoomen und ändert nicht die darunterliegenden Datensätze. Das konzeptionelle Modell für Visualisierungsoperationen des Systems Lark [TIC09] basiert auf der Trennung von Sicht- und Wertoperationen. Lark ist ein System, dass die Koordination von Interaktionen mit Informationsvisualisierungen auf geteilten Arbeitsplätzen unterstützt. Der Schwerpunkt der Unterstützung liegt auf den Merkmalen: Abgrenzung von Interaktionen, zeitliche und räumliche Flexibilität sowie änderbare Zusammenarbeitsstile. Dies wird erreicht durch die Integration einer Repräsentation von Informationsvisualisierungspipelines basierend auf dem Data-State-Modell und

der zusätzlichen Information von Koordinatenpunkten nach der Idee von [HA08]. Diese Daten werden in einem Meta-Datenmodell angelegt, welches ebenfalls visualisiert wird, wodurch der Analyst einen Pfad von Quelldaten bis zur Visualisierung erkennen kann.

3.3 Zusammenfassung

In den vorangegangenen Kapiteln sind Visualisierungsmodelle und Modelle zur Abbildung von Interaktionen auf Visualisierungen vorgestellt worden. Angefangen von einem Visualisierungsmodell, angelehnt an dem relationalen Modell aus der Informationstechnologie, sind Visualisierungspipelinemodelle vorgestellt worden. In wissenschaftlichen Visualisierungen wird häufig das Data-Flow-Pipelinemodell verwendet. Dieses ist in der großen Anzahl von vorhandenen Data-Flow-Schnittstellen begründet [JK03]. Im Bereich der Informationsvisualisierung werden häufig Data-State-Modelle verwendet, obwohl diese auch für wissenschaftliche Daten geeignet sind [Chi00]. Diese basieren ebenfalls auf einer Visualisierungspipeline zur Bestimmung des Visualisierungsergebnisses, im Gegensatz zu Data-Flow-Modellen, findet allerdings eine Änderung der Daten in den Transaktionen zwischen Elementen der Pipeline statt. Beim Data-Flow-Modell werden diese Änderungen in den Knoten der Pipeline durchgeführt. Auf Pipelines basierende Modelle gestatten in der Regel sehr leicht die Einbindung von weiteren Komponenten, ohne große Änderungen an den Bestehenden durchführen zu müssen, da die Struktur sehr flexibel ist. Diese Flexibilität ist bei dem weiteren vorgestellten Visualisierungsmodell, der direkten Abbildung von Dateneigenschaften auf Visualisierungseigenschaften, nicht unbedingt gegeben. Allerdings sind diese Modelle sehr leistungsfähig, da kein Verwaltungsaufwand für eine Pipeline anfällt und evtl. Berechnungen nicht wiederholt durchgeführt werden müssen. Des Weiteren ist ein Modell speziell für geographische Informationsvisualisierungen vorgestellt worden, welches die Prinzipien der Wahrnehmung des Menschen integriert.

Im Bereich der Interaktionsabstraktion sind die Modellformen des syntaktischen und semantischen Interaktionsmodells vorgestellt worden. Syntaktische Modelle beschreiben die Abbildung von Ereignissen, die durch den Benutzer ausgelöst werden, auf ein darunterliegendes Modell. Indes versuchen semantische Modelle die Absicht, die ein Nutzer mit einer Interaktion verfolgt, zu erfassen und diese auf ein Modell abzubilden. Ein Ansatz hierfür sind zum Beispiel die von Chuah und Eick vorgestellten *BVI*s, in denen eine Interaktion aus kleineren atomaren Ereignissen besteht [CR96] und somit die Absicht des Benutzers besser ermittelt werden kann. Allerdings gestaltet es sich schwierig immer die genauen Absichten – die Semantik einer Interaktion – eines Benutzers zu erkennen, auch dann, wenn der logische Ablauf von Benutzerinteraktionen nicht eindeutig ist, wie dies der Fall in der visuellen, explorativen Datenanalyse ist.

Neben diesem Problem stellt sich die Frage, wie hierarchische Daten, die häufig in der Anwendungsdomäne auftreten, modelliert werden können. Data-Flow- und Data-State-Modelle beschreiben das Zusammenwirken von verschiedenen Modellierungskomponenten untereinander, allerdings nicht die eigentliche Datenhaltung. Auch das relationale Modell bietet nur eingeschränkte Möglichkeiten um diesen Beziehungstyp zwischen Daten zu modellieren. Es ist somit nicht nur ein Modell für den Visualisierungsprozess an sich notwendig, sondern auch eine Beschreibung der Daten in Form eines Modells [JKMG07] und das Zusammenspiel dessen mit einem Modell zur Interaktionsabstraktion. Keines der betrachteten Modelle bietet diese Eigenschaften, weshalb im Kapitel 6 ein eigenes Modell entwickelt wird.

4 Visual-Analytics-Systeme

Im Bereich der explorativen, visuellen Datenanalyse stehen einige Systeme zur Verfügung. Eine ausgewählte Menge wird in diesem Kapitel beschrieben und im Anschluss bewertet. Die Bewertungskriterien ergeben sich aus Besonderheiten der Systeme, die jeweils am Rand der Seite notiert sind. Das hieraus entwickelte Bewertungssystem wird nach Entwicklung des *VAT*-Systems verwendet, um einen Vergleich zwischen den Systemen und *VAT* zu erzielen. Als Vergleichssysteme werden HD-Eye, SellTrend, DataMeadow, MineSet und Advizor gewählt. Das System HD-Eye ist gewählt worden, da es auf das Ziel Clusterung und der damit verbundenen Musteranalyse spezialisiert ist. Dieses Ziel überschneidet sich mit dem Ziel in der Anwendungsdomäne. Der Vergleich mit SellTrend wird gewählt, da es ebenso wie *VAT* die gleichzeitige Anzeige einer Datenmenge in verschiedenen Visualisierungsformen unterstützt und DataMeadow, weil eine große Ähnlichkeit mit den vorgestellten Visualisierungsprozesses (vgl.Kapitel 3.1) besteht. DataMeadow basiert auf einem individuell angepassten Data-Flow-Konzept und weist damit ein explizites Modell auf, wie dieses auch im MineSet-System der Fall ist. Das kostenpflichtige System Advizor ist nach Angaben des Herstellers das führende Produkt zur Analyse [Eic09b] und wird aus diesem Grund zum Vergleich herangezogen. Im weiteren Verlauf werden zunächst die Systeme beschrieben und im Anschluss anhand der identifizierten Besonderheiten aller beschriebenen Systeme miteinander verglichen. Begonnen wird mit dem HD-Eye-System.

4.1 HD-Eye

Ziel: Clusterung

Das HD-Eye-System [HKW99] unterstützt einen Analysten bei einer Clusterung, indem es eine interaktive Möglichkeit bietet, den Clusterungs-Prozess nachzuvollziehen und zu beeinflussen. Clusterung von großen Datenbanken mit hoch-dimensionalen Daten ist eine Herausforderung, da die Algorithmen häufig komplex und das Ergebnis dadurch nicht nachvollziehbar ist [HKW03]. Zusätzlich wird das Problem der Clusterung verstärkt indem Daten analysiert werden, die nicht zum Zweck der Analyse gesammelt worden sind. Das Clustering-Problem kann als die Aufteilung von hoch-dimensionalen Datensatzvektoren in eine Anzahl von Cluster und einem Rauschen definiert werden, mit dem Ziel, dass die Datenvektoren in einem Cluster ähnlich sind. Cluster-Algorithmen benötigen deshalb meistens ein Entscheidungsmodell um die Ähnlichkeit zwischen Datensätzen zu bestimmen. In einfachen Fällen kann dieses eine simple Funktion sein, in der Praxis wird die Ähnlichkeit zwischen zwei Datensätzen jedoch häufig in Abhängigkeit der Aufgabe und der Anwendung definiert. Das HD-Eye-System integriert Clustering-Algorithmen und Visualisierungstechniken, um ein besseres Verständnis und eine effektivere Clusterbildung zu ermöglichen.

Die umgesetzte Idee im HD-Eye-System ist es, den Clusterungsprozess zu verbessern, indem der Anwender die Möglichkeit erhält direkt in die entscheidenden Schritte des Prozesses einzugreifen. Diese entscheidenden Schritte sind die Selektion der betrachteten Attribute, die Wahl des Clustering-Algorithmus und die Aufteilung der Datensätze in Cluster. In HD-Eye sind Visualisierungstechniken und Clustering-Algorithmen eng miteinander verknüpft. Es können die Parameter der Clusteralgorithmen interaktiv angepasst werden und die Daten schrittweise in einzelne Teile aufgeteilt werden. Um die Daten zu analysieren stehen verschiedene Visualisierungstechniken zur Verfügung (siehe Abbildung 4.1). Diese werden im Folgenden vorgestellt. Die Nummern in Klammern verweisen auf die entsprechenden Zahlen in der Abbildung 4.1.

Selektion

Parameteranpassung

Die verwendeten Visualisierungstechniken sind Kombination von pixel-orientierten Dichtediagram-

Pixeldarstellung

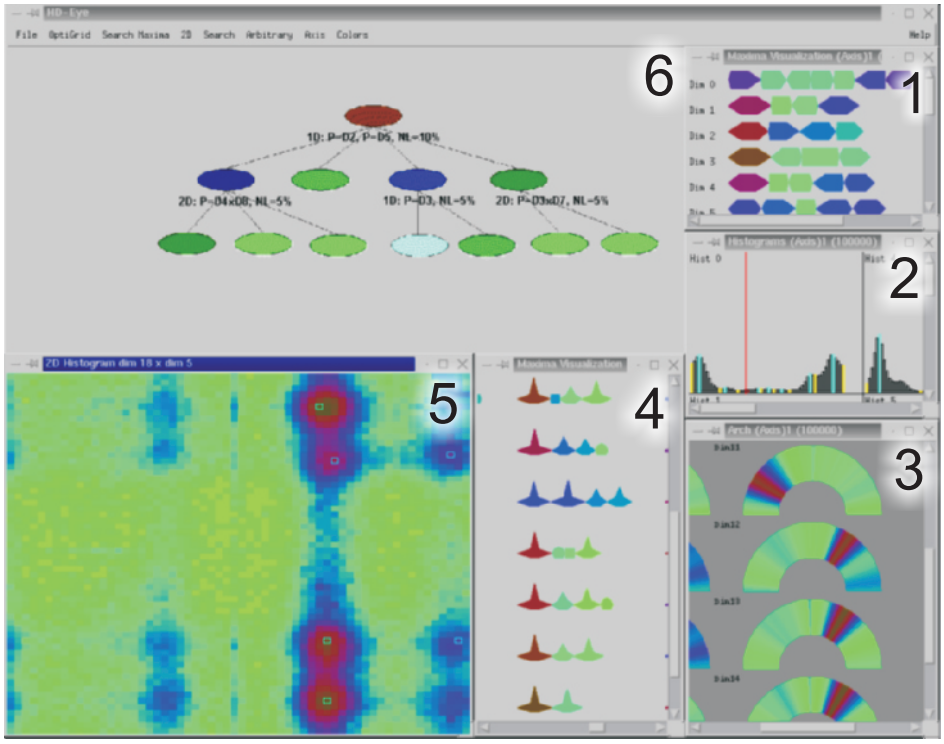


Abbildung 4.1: *Das HD-Eye-System (vgl. [HKW03])*

Icondarstellung

men und Icon-Repräsentationen von Daten. In der Icon-Visualisierung (1) wird für jede Dimension der Daten eine eigene Clusterung durchgeführt. Ein Analyst kann erkennen, wie weit sich in den einzelnen Gruppen die Daten unterscheiden, welches durch die Form der Icons verdeutlicht wird. Mit Hilfe der Farbe der Icons wird die Anzahl der Elemente in dem Cluster visualisiert. Dieses wird durch die Darstellung der Daten als Histogramm (2) verstärkt. Außerdem kann gleichzeitig in einem Pixel-Dichtediagramm (3) die Verteilung der Daten für jedes Attribut einzeln dargestellt werden. Das Ergebnis einer Clusterung einer Attributmenge wird mit Hilfe einer Icon-Visualisierung (4) dargestellt. Es ist erkennbar, welches Ergebnis einer Clusterung von Kombinationen von Attributen ergeben würde. Zur Erstellung von Clustern ist die Definition von Trennungen zwischen diesen unabdingbar. Dies kann in HD-Eye zum einen durch eine interaktive Auswahl der Cluster in einem Pixel-Dichtediagramm erfolgen (5) oder durch Beeinflussung der Clusterungs-Parameter im resultierenden Entscheidungsbaum (6) für die Clusterung. Die Visualisierungen (1-6) sind miteinander verknüpft, wodurch sich Änderungen an einer Visualisierung auf alle anderen auswirken können.

Verknüpfung

4.2 SellTrend

SellTrend [LSS09] ist ein System zur Analyse von Flugbuchungsanfragen und ist in Kooperation mit Travelport, einem globalen Flugreservierungsunternehmen, entstanden. Travelport ist ein Vermittler zwischen Reiseagenturen, internetbasierten Reisediensten und Fluggesellschaften. Als Vermittler erhält Travelport von verschiedenen Fluggesellschaften Fluginformationen, die für mögliche Buchungen zur Verfügung stehen. Wenn ein Kunde eine Reiseagentur kontaktiert und versucht einen Flug zu buchen, entsteht eine Menge von Daten, wie zum Beispiel gewünschter Abflug- und Ankunftsflughafen,

Reisezeit oder gewünschte Flugairline. Die Schwierigkeit besteht in der Analyse dieser multivariaten zeitlichen Daten, um Vorhersagen und Gründe für Flugbuchungen zu ermitteln, wodurch eine bessere Kundenanpassung und Auslastung der Flüge ermöglicht werden würde. Des Weiteren wird besonderer Wert auf die Auswertung von abgebrochenen Buchungsanfragen gelegt, um zu ermitteln, aus welchem Grund der Abbruch der Buchung stattfand. Zur Unterstützung einer Analyse dieser komplexen Daten ist SellTrend entwickelt worden.

zeitliche Daten

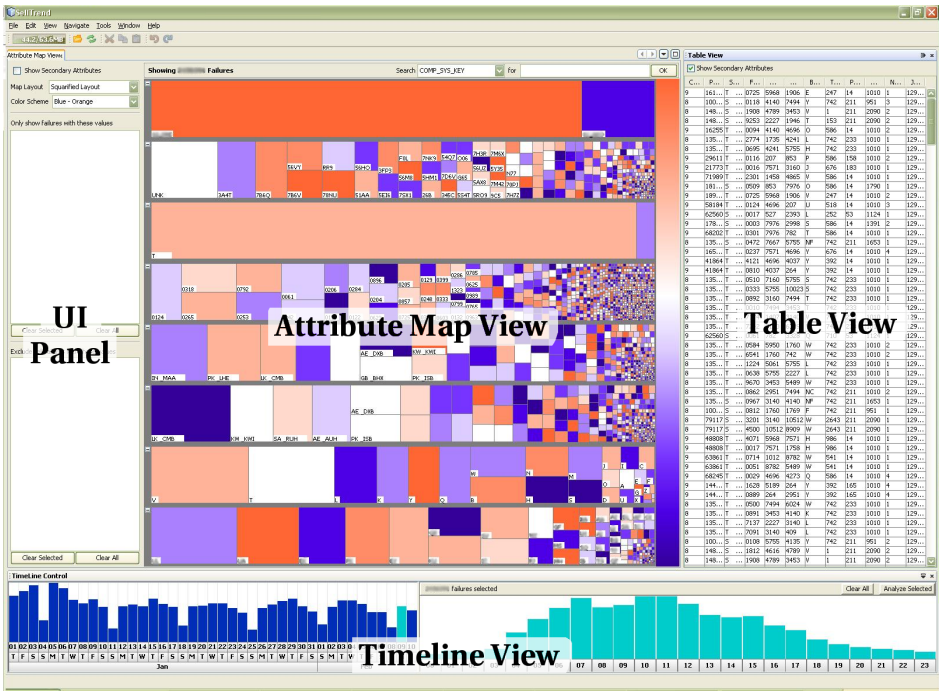


Abbildung 4.2: *Das SellTrend-System [LSS09]*

Das System SellTrend stellt Daten gleichzeitig in verschiedenen Ansichten dar (vgl. Abbildung 4.2). Im unteren Bereich des Systems wird eine Zeitansicht in Form eines Balkendiagramms von fehlgeschlagenen Buchungen pro Tag und Stunde angezeigt. Ein Analyst kann hier die betrachtete Zeitperiode interaktiv auswählen. Alle fehlgeschlagenen Buchungen von der gewählten Zeitperiode werden dann in einer Tabellenansicht dargestellt. Gleichzeitig werden dieselben ausgewählten Daten in einer TreeMap [Shn92] dargestellt. Eine TreeMap ist eine geschachtelte Visualisierung, in der hierarchische Daten in Form untereinander aufgeteilter Flächen dargestellt werden. Die einzelnen Teilflächen werden anhand eines Kategorisierungsparameters bestimmt. Die Größe der Fläche wird implizit durch die Mächtigkeit der enthaltenen Subelemente einer Einheit definiert. Außerdem wird dieser Effekt häufig durch eine entsprechende Einfärbung der Elemente unterstützt. In SellTrend wird diese Form der Visualisierung genutzt um ein Attribut (zum Beispiel Fluggesellschaft oder Abflugdatum) der ausgewählten Transaktionen darzustellen. Für jedes Attribut der Transaktionen wird eine eigene TreeMap erstellt, welche zeilenweise dargestellt werden. Wird die Maus über ein Element in der TreeMap gehalten, wird ein Tooltip mit der Anzahl der Unterelemente und die Anzahl von abgebrochenen Buchungen, die mit diesem Attribut in Verbindung stehen, angezeigt. Als weitere Funktionalität bietet das System die Möglichkeit, während des Analyseprozesses die Anzeige der Attribute von abgebrochenen Buchungen auf eine bestimmte Menge von Attributen oder Attributwerten einzuschränken. Hierzu stehen zwei Filter zur Verfügung. Zum einen ein Fokusfilter, der nur Elemente

Geschachtelte Darstellung

Filterung

mit einem bestimmten Wert für ein Attribut anzeigt und zum anderen ein Ausgrenzungsfilter, der nur die Elemente filtert und damit nicht anzeigt, die genau dem spezifizierten Wert für das Attribut entsprechen.

4.3 DataMeadow

multi-dimen-sionale Daten

Vergleich von Daten-sätzen

DataMeadow [EST08] ist für die Analyse von multidimensionalen Datensätzen mit dem Hauptaugenmerk auf die Unterstützung von Vergleichen zwischen verschiedenen Datensätzen oder Teilmengen von Datensätzen erstellt worden. Die Einbindung von großen, multidimensionalen Datensätzen in der visuellen Analyse erfordert einen hohen Grad an Interaktivität und Benutzerkontrolle. Aus diesem Grund ist in DataMeadow ein neues Konzept zur interaktiven Abfrage und Filterung von Datenquellen eingeführt worden. Abfragen werden graphisch mit Hilfe von Daten-Rosen (vgl. Abbildung 4.3) erstellt. Eine Daten-Rose ist ein farbcodierter Starplot, mit dessen Hilfe Datensätze angezeigt und gleichzeitig durch Slider an jeder Achse gefiltert werden können. Daten-Rosen können miteinander verbunden werden, was durch einen Pfeil von der Quell-Daten-Rose zur Ziel-Daten-Rose dargestellt wird. Durch diesen Weg kann ein Anwender sukzessiv eine oder mehrere komplexe Abfragen erstellen. Daten-Rosen können in einer 2D-Canvas frei bewegt und skaliert werden, was einen einfacheren Vergleich zwischen verschiedenen Daten-Rosen ermöglicht. Außerdem können einzelne Datenelemente selektiert werden. Die Selektionsinteraktion wird an die folgenden visuellen Komponenten weitergereicht, die daraufhin das selektierte Datenelement ebenso selektiert darstellen. Die 2D-Canvas ist ein Container, der neben Daten-Rosen Meta-Daten über die zur Verfügung stehenden Datenquellen bereitstellt. Außerdem besteht die Möglichkeit alle Analyseschritte in einer History, die von der Canvas zur Verfügung gestellt wird, nachzuvollziehen. Datenquellen werden auch in Form einer Daten-Rose visuell dargestellt, allerdings in einer anderen Farbe, damit diese von anderen Daten-Rose-Typen unterschieden werden können. Weiterhin stehen noch Daten-Rose-Typen für die Mengenoperationen Vereinigung und Schnitt zur Verfügung, um verschiedene Pfade bestehend aus Daten-Rosen miteinander zu verbinden.

Selektions-verknüpf-ung

Speiche-rung des Analyseer-gebnisses

Für die visuelle Repräsentation der Daten innerhalb der Daten-Rose existieren die Modi *color-histogram*, *opacity band* und *parallel coordinate* (vgl. Abbildung 4.3). Im color-histogram-Modus

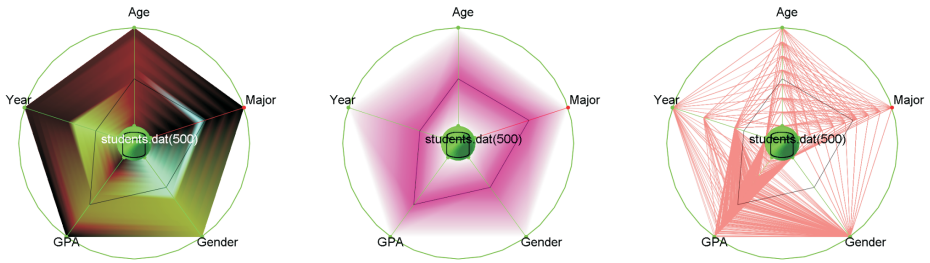


Abbildung 4.3: *Beispielhafte Darstellung von Daten-Rose-Visualisierungen im DataMeadow-System (vgl. [EST08])*

wird die Verteilung der Daten für jede Dimension auf der Oberfläche der Daten-Rose mit einer kontinuierlichen Farbskala dargestellt. Die Farbübergänge zwischen Farbwerten benachbarter Achsen werden interpoliert. Eine hohe Helligkeit zeigt eine hohe Dichte in der zugrunde liegenden Verteilung der Daten an. Mit Hilfe der grünen Punkte jeder Achse können die Daten interaktiv gefiltert werden.

Diese Filterung beeinflusst direkt alle folgenden Komponenten, wodurch ein Analyst sofort den Effekt einer Parameterbeeinflussung erkennen kann. Im opacity-band-Modus werden die Daten durch ein Transparenzband von voller Deckkraft im Durchschnitt der Daten bis zur vollständigen Transparenz für die Extremwerte (Minimum und Maximum) dargestellt. Übergänge zwischen den Achsen werden wiederum interpoliert. Es sind die gleichen Trends erkennbar, wie in der Farbdarstellung, allerdings ist das Abstraktionslevel höher. Darüber hinaus ist die Dichte der Daten für verschiedene Werte weniger offensichtlich. Im parallel-coordinate-Modus wird eine Parallelkoordinatendarstellung [Ins85] zur Abstraktion der Daten verwendet, wobei jeder Datensatz durch ein Polygon, welches durch die Achsenschnittpunkte bestimmt wird, visualisiert wird. Der Nachteil liegt darin, dass die Verteilung der Daten schwer erkennbar ist. Neben Daten-Rosen besteht die Möglichkeit Daten in Balken- oder Tortendiagrammen oder in einem Histogramm explizit darzustellen.

geometrie-basierende Darstellung

Standard 2-D/3-D Darstellung

4.4 MineSet

MineSet [BKK97] integriert Datenbankzugriffe, analytisches Data-Mining und Datenvisualisierungen in einem System. Hierdurch wird der Erkenntnisprozess von der Vorbereitung der Daten bis zur iterativen Analyse mit Hilfe von 2D- und 3D-Visualisierungen ermöglicht. Die Visualisierungen ermöglichen die direkte Datenvisualisierung zur explorativen Analyse von hochdimensionalen, geographischen oder hierarchischen Daten. Des Weiteren stehen Data-Mining-Algorithmen zur Verfügung, um interessante Aspekte einer Datenmenge leichter identifizieren zu können. Die Ergebnisse der Algorithmen können mit Hilfe von Visualisierungen verifiziert werden. Gegebenenfalls können die Parameter angepasst und anschließend die Analyse erneut durchgeführt werden.

Datenvorbereitung

räumliche Daten

hierarchische Daten

Damit diese Funktionalität zur Verfügung gestellt werden kann, besteht das MineSet-System aus einer dreistufigen Client-Server-Architektur. Die erste Stufe ist der Client, welcher den Tool-Manager und Visualisierungstools enthält. Der Tool-Manager ist die graphische Schnittstelle, durch die ein Anwender mit dem MineSet-System interagieren kann. Außerdem stellt er eine graphische History zum Verwalten der durchgeführten Benutzeroperationen bereit und ermöglicht es diese in zukünftigen Analyseszenarien wieder anzuwenden. Visualisierungstools werden zur Visualisierung von Daten und Modellen von Daten, die durch Data-Mining-Algorithmen generiert wurden, verwendet. Nachdem Aufruf eines Visualisierungstools mit dem Tool-Manager, kann der Analyst direkt mit dem Tool interagieren und Informationen an das Tool senden, bzw. an andere Tools weiterleiten. Die zweite Schicht ist der Server, welche den Data-Mover und die Data-Mining-Engine beinhaltet. Mit Hilfe des Data Movers ist der Zugriff auf die Datenbankschicht, die häufig aus einem Datenbankmanagementsystem besteht, möglich. Mining-Tools werden zur Generierung von Modellen verwendet, die auf neue Daten oder visualisierte Daten angewendet werden können. Zur Unterstützung von unterschiedlichen Zielen bei der Analyse mit verschiedenen Datenformen sind mehrere Visualisierungsformen in MineSet integriert worden. Beispielsweise können statistische Informationen über Daten mit Hilfe von Histogrammen oder in Boxplots dargestellt werden. Boxplots fassen Werte in einem Feld inkl. des Medians und Quantile zusammen. Extremwerte werden als Punkte außerhalb der Box dargestellt. Außerdem existiert ein mit bis zu acht Attributen belegbares 3D-Scatterplot. Neben den Koordinatenachsen können die Farb-, Größen- und Drehungseigenschaften der dargestellten Objekte mit Attributen belegt werden. Zwei unabhängige Attribute können des Weiteren mit Hilfe einer Animation dargestellt werden. Das Scatterplot kann auch mehrere tausend Datensätze darstellen. Das häufig bei dieser hohen Anzahl von dargestellten Datensätzen auftretende Problem der Überdeckung wird durch einen speziellen Verwischungseffekt gelöst, wodurch der Eindruck einer Fläche von Daten entsteht. Für Datensätze mit

Data-Mining

Wiederanwendung des Wissens

Modellbildung

diskrete Animation

räumlichen Informationen kann eine Kartenvisualisierung gewählt werden. Die Choroplethenkarte kann zwei unabhängige, quantitative Attribute mit Hilfe der Farbe und Höhe der gezeichneten Fläche visualisieren. Eine Verknüpfung von verschiedenen Karten ist möglich. Existieren in den Datensätzen hierarchische Beziehungen, können diese durch eine Baumdarstellung visualisiert werden, in der jede Ebene des Baumes zusätzlich auf ein Histogramm abgebildet werden kann. Innerhalb einer Visualisierungsform kann interaktiv mit Hilfe der Maus eine Verschiebung, Rotation oder ein (optischer und semantischer) Zoomen erfolgen.

Verschiebung

Zoom

Neben den Visualisierungsformen werden durch die Data-Mining-Engine verschiedene Funktionen zur Analyse von Daten bereitgestellt, wie zum Beispiel Klassifikationsalgorithmen oder Assoziationsgeneratoren. Außerdem stehen Algorithmen zur automatischen Auswahl relevanter Attributmengen zur Verfügung. Ein Analyst kann zum Beispiel bei der Betrachtung eines Phänomens in einem Scatterplot die Anfrage an das System stellen, welche Attribute interessant für die Ursache dieses Phänomens sind. MineSet identifiziert die wichtigen Attribute mit Hilfe der bedingten Entropieminimierung [KL95] auf Grundlage von Entscheidungsbäumen. Zusätzlich bietet MineSet Filter- und Suchfunktionen innerhalb einer Visualisierung, durch die ein Analyst in die Lage versetzt wird, schnell die Anzahl der Elemente auf die Wichtigen zu reduzieren.

Ziel: Klassifizierung

Ziel: Assoziationen

4.5 Advizor

Advizor [Eic09b] ist entwickelt worden, um eine eigenständige Datenanalyse für Analysten und Anwender in einem Unternehmen zu ermöglichen. Das System ermöglicht das Erstellen von Visualisierungskompositionen für Analyse- und Präsentationszwecke. Die Datenanalyse ist auf Wirtschaftsdaten spezialisiert, kann jedoch auch für andere Datendomänen verwendet werden [G.E00]. Das System unterstützt die Analyse durch verschiedene Visualisierungsformen, die interaktiv miteinander verknüpft werden können. In Abhängigkeit der Datendomäne und des Analysezwecks muss der Analyst zunächst eine Sicht (Page) bestehend aus einer oder evtl. mehreren Visualisierungsformen zusammenstellen. Diese Sicht stellt die Analysedaten in unterschiedlichen Visualisierungen mit verschiedenen Kodierungen zur Repräsentation der Daten dar. Innerhalb dieser Sicht kann der Analyst neben Verschiebungen und Rotationen häufig auch direkt eine Verfeinerung oder Aggregation der Daten vornehmen.

Kodierung

Advizor enthält 15 interaktive Visualisierungsformen, die jeweils unterschiedlich parametrisiert werden können, wodurch verschiedene Darstellungen entstehen [Eic09a]. Eine Visualisierungsform sind Balkendiagramme, in denen Daten gruppiert in Kategorien dargestellt werden. Neben der Orientierung und der Größe der Balken kann die Gruppierungsfunktion zur Bestimmung der Gruppen angepasst werden. Ist die Verteilung der Daten von Interesse, kann beispielsweise ein Tortendiagramm oder ein Histogramm verwendet werden. Durch die Wahl eines Animationsattributes kann ein Tortendiagramm animiert werden, so dass ein Verlauf über das spezifizierte Attribut, zum Beispiel die Zeit, möglich ist. Histogramme zeigen die Verteilung einer numerischen Variablen mit einer Glättung an. Es kann sowohl die Ausrichtung (horizontal oder vertikal) als auch die bestimmende Dichtefunktion und das Skalenniveau (linear, logarithmisch, usw.) gewählt werden. Außerdem stehen Datentabellen zur Verfügung. Diese Visualisierungsform erweitert klassische Tabellen um farbcodierte Zellenwerte und unterstützt eine Sortierung wie auch eine Filterung. Durch Zoomen wird die Schriftgröße der Datenwerte bestimmt, wodurch mehr Informationen dargestellt werden können. Bei weiterem Zoomen, über die Lesbarkeitsschwelle hinaus, ändert sich die Tabelle zu einem Balkendiagramm und schließlich zu einer Dichtedarstellung. Weiterhin steht eine Zusammenfassungstabelle, welche Daten nach spezifizierten Attributen und Funktionen aggregiert und nur die Aggregationen der Werte darstellt,

im System bereit. Hierdurch wird allerdings nicht die analysierte Datenmenge eingeschränkt. Sollen bei allen in der Page enthaltenen Visualisierungen die Daten gefiltert werden, kann ein Textfilter verwendet werden. Eine Textfiltervisualisierung ermöglicht die Selektion einer Teilmenge von Daten. Die gesamte Datenmenge ist weiterhin verfügbar, allerdings werden gefilterte Daten ignoriert. Die gefilterte Datenmenge kann interaktiv geändert werden. Um Beziehungen zwischen zwei Attributen zu ermitteln, kann ein Scatterplot verwendet werden, in dem zusätzlich zur Vereinfachung der Analyse eines Trends eine Trendlinie, die beispielsweise mit Hilfe der linearen Regression ermittelt wird, eingefügt werden kann. Als weitere Visualisierungsform zur Ermittlung von Trends existieren im System Liniendiagramme. Diese Visualisierung kann animiert werden, um ein weiteres Attribut darzustellen. Neben diesen für allgemeine Daten zur Verfügung stehenden Visualisierungstypen sind in Advizor für spezielle Analysezwecke weitere Visualisierungstypen eingebunden. Zum Beispiel Time-Tables [EL96] für die Anzeige von Daten, die einem Zeitstempel zugeordnet werden können. In einer Time-Table wird für jedes Ereignis eine Markierung in einem Zeit-Typ-Gitter eingefügt. Die Markierungen unterscheiden sich in Farbe, Form und Rotation um den Ereignistyp und andere in Beziehung stehende Eigenschaften darzustellen. Außerdem kann die Dauer eines Ereignisses durch eine Linie, die an der Markierung beginnt, dargestellt werden. Eine weitere spezielle Visualisierungsform ist die Multiscape-Visualisierung zur Darstellung von Beziehungen zwischen zwei Kategorien oder die Parabox-Visualisierung zur Darstellung von n-dimensionalen, kontinuierlichen oder kategorischen Daten. Eine Parabox ist eine Kombination aus Boxplot und Parallel-Koordinatenvisualisierung. Die Stärke einer Boxplot-Visualisierung besteht in der Darstellung von Verteilungen einer Menge, während mit Hilfe der Parallel-Koordinatenvisualisierung hochdimensionale Datensätzen dargestellt werden können. In Parabox-Visualisierungen sind diese beiden Datenrepräsentationen vereint. Des Weiteren steht eine Graphvisualisierung zur Darstellung von Beziehungen innerhalb von Datensätzen zur Verfügung. Wenn die Daten einen Raumbezug haben, kann diese Visualisierungsform mit verschiedenen Kartentypen kombiniert werden. Die Multiscape-Visualisierung stellt zwei kategorische Attribute in einer 2D- oder 3D-Form dar. Ist die Beziehungsart innerhalb der Datensätze hierarchisch können diese auch in einer TreeMap dargestellt werden.

Zusätzlich zu den Visualisierungstypen sind in Advizor Data-Mining-Verfahren und Algorithmen eingebunden, um eine vorherige Analyse von Daten zu ermöglichen. Die hieraus erstellten Modelle können mit allen Visualisierungstools verwendet werden. Die Parameter und die verwendeten Attribute im Verfahren können interaktiv angepasst werden. Das Ergebnis der Analyse wird graphisch dargestellt und kann in den aufgeführten Visualisierungen weiter analysiert werden.

Modellvisualisierung

4.6 Bewertung und Zusammenfassung

Anhand der identifizierten Merkmale der betrachteten Systeme können diese miteinander verglichen werden. In der Tabelle 4.1 sind diese Daten gegenübergestellt. Wenn eine Eigenschaft von einem System nicht erfüllt ist, wird das ✗-Symbol eingefügt, falls nur teilweise bzw. eine unzureichende Umsetzung eines Merkmales existiert, wird das ❍-Symbol eingefügt. Für die Unterstützung einer Eigenschaft wird das ✓-Symbol verwendet.

Nr.	Eigenschaft	HD-Eye	SellTrend	DataMeadow	MineSet	Advizor
1	Pixeldarstellung	✓	✗	✗	✗	✗
2	Icondarstellung	✓	✗	✗	✗	✓
3	Geschachtelte Darstellung	✗	✓	✗	✗	✓

4	Geometriebasierende Darstellung	✗	✗	✓	✗	✓
5	Standard 2D-/3D-Darstellung	✗	✓	✓	✓	✓
6	Vergleich von Datensätzen	✗	✓	✓	✗	✓
7	Datenvorbereitung	✗	✗	✗	✓	✗
8	Räumliche Daten	✗	✗	✗	✓	✓
9	Hierarchische Daten	✗	✗	✗	✓	✓
10	Zeitliche Daten	✗	✓	✗	✗	✓
11	Multdimensionale Daten	✗	✗	✓	✓	✓
12	Selektion	❍	✓	✓	❍	✓
13	Verknüpfung	✓	✓	✗	✗	✗
14	Selektionsverknüpfung	✗	✗	✓	❍	✓
15	Verschiebung	✗	✗	✓	✓	✓
16	Zoom	✗	✗	❍	✓	✓
17	Filterung	✗	✓	✓	✓	✓
18	Kodierung	✗	✗	✓	✓	❍
19	Parameteranpassung	✓	✗	✗	✓	✓
20	Data-Mining	✗	✗	✗	✓	✓
21	Modellbildung	✗	✗	✗	✓	✗
22	Modellvisualisierung	✗	✗	✗	❍	✓
23	Ziel: Clusterung	✓	✗	✗	✓	✗
24	Ziel: Klassifizierung	✗	✗	✗	✓	✗
25	Ziel: Assoziationen	✗	✗	✗	✗	✓
26	Diskrete Animation	✗	✗	✗	✓	✓
27	Speicherung des Analyseergebnisses	✗	✗	✓	✓	✓
28	Wiederanwendung des Wissens	✗	✗	✗	✓	✗

Tabelle 4.1: *Identifizierte Merkmale der Systeme im Vergleich*

Die Pixeldarstellung ist als einzige im HD-Eye-System integriert. Außerdem bietet dieses System, wie Advizor eine Icondarstellung. Weiterhin bietet Advizor die Möglichkeit einer geschachtelten Darstellung. Diese wird, wie bei SellTrend, mit Hilfe einer TreeMap ermöglicht. Die Parallelkoordinatendarstellung des DataMeadow-Systems ist eine geometriebasierte Darstellung. Diese wird in Advizor als Parabox-Visualisierung erweitert und dem Anwender zu Analysezwecken zur Verfügung gestellt. Das Merkmal Standard 2D-/3D-Darstellung wird nur von dem System HD-Eye nicht erfüllt. In SellTrend werden Balkendiagramme zur Darstellung von aggregierten Zeitdaten verwendet. In DataMeadow wird diese Visualisierungstechnik verwendet, um einen ausgewählten Datensatz darzustellen. Aber auch MineSet und Advizor bieten die Möglichkeit eine Datenmenge zum Beispiel mit Hilfe eines Scatterplottes darzustellen. Auf Grundlage einer Menge von Visualisierungen bietet es sich an, eine Datenmenge in verschiedenen Visualisierungen gleichzeitig darzustellen, um die Schwächen einer Visualisierungsform durch eine andere zu beheben und einen Vergleich von Datensätzen zu vereinfachen. Diese Eigenschaft ist in HD-Eye und Advizor umgesetzt. Zudem wird diese Technik im System SellTrend verwendet, um in der TreeMap eine graphische Repräsentation und in einer Tabelle die konkreten Daten darstellen zu können. DataMeadow bietet die Möglichkeit an, die in Daten-Rosen visualisierten Daten, graphisch nebeneinander zu verschieben. Gleichzeitig können diese auch in der Größe individuell angepasst und somit miteinander verglichen werden. In der Regel müssen die zu analysierenden Daten vorbereitet werden und in ein einheitliches Format überführt werden. Dieses wird nur von MineSet durch die Integration von verschiedenen Datenbankzugriffen und Werkzeugen ermög-

licht. Zudem stellt das MineSet-System Transformationen zwischen unterschiedlichen Datenquellen bereit. Als Analysedaten können hierbei sowohl räumliche, wie auch hierarchische Daten verwendet werden. Zusätzlich bietet Advizor die Möglichkeit zeitliche Daten zu analysieren. Diese Eigenschaft bietet auch das System SellTrend, welches zur Analyse von Flugbuchungsdaten mit zeitlichem Aspekt entwickelt wurde. Als weitere Dateneigenschaft können mit Hilfe von MineSet, DataMeadow und Advizor multidimensionale Daten analysiert werden.

Die Selektion von Daten wird von allen Systemen unterstützt, auch wenn die Selektion auf unterschiedlicher Ebene ermöglicht wird. Im HD-Eye ist die Selektion von Attributen möglich, während in den anderen Systemen die Auswahl eines spezifischen Datensatzes möglich ist. In MineSet ist die Selektion von Datensätzen allerdings auf die Kartendarstellung begrenzt. Die Änderungen von Selektionen werden in DataMeadow an alle verknüpften Daten-Rosen weitergereicht. Die Selektionsverknüpfung wird des Weiteren von Advizor in der dargestellten Page ermöglicht. Innerhalb der Page werden Selektionen auf alle Visualisierungen übertragen, wie dies auch in MineSet der Fall ist, allerdings können hier nur Karten miteinander verknüpft werden. Eine allgemeinere Art der Verknüpfung ist in HD-Eye und SellTrend implementiert. In HD-Eye wird die Auswahl von Bereichen auf die dargestellten Visualisierungen übertragen und in SellTrend ist das Balkendiagramm als Aggregation der verfügbaren Daten mit der TreeMap und der Detailtabelle verknüpft. Durch Auswahl der Daten im Balkendiagramm werden die Daten für den ausgewählten Bereich in diesen Visualisierungen dargestellt. Für den Fall, dass eine Datenmenge zu dicht ist, um Details zu betrachten, können in DataMeadow, MineSet und Advizor optische Zooms und Verschiebungen durchgeführt werden. Außerdem bieten MineSet und Advizor die Möglichkeit an, einen semantischen Zoom durchzuführen. Damit die betrachtete Datenmenge weiter eingeschränkt werden kann, sind in den Systemen (außer in HD-Eye) Filterungen eingebunden. Diese können beispielsweise, wie in SellTrend zur Ausgrenzung eines Datensatzes oder zur expliziten Betrachtung einer Datenmenge verwendet werden, oder wie im Fall von DataMeadow, für ein interaktives Filtern innerhalb jeder Daten-Rose. Zusätzlich ermöglichen es die Systeme die angezeigten Daten in einer anderen Visualisierungsform darzustellen und damit die Kodierung zu ändern. Bei den Systemen DataMeadow und MineSet kann die Kodierung während der Analysezeit geändert werden. In Advizor wird die Kodierung durch eine Page vorher definiert. Zusätzlich sind in MineSet und Advizor Data-Mining-Algorithmen für eine automatische Analyse einer Datenmenge eingebunden, um interessante Aspekte der Menge leichter zu identifizieren. Die hieraus resultierenden Modelle können in den Visualisierungen der Systeme dargestellt werden. Anpassung der Data-Mining-Parameter werden interaktiv auf die Modelle übertragen. In Advizor werden die Modelle automatisch den Visualisierungen zugewiesen, während dieses in MineSet manuell durchgeführt werden muss. Eine Modellbildung aus einer bestehenden Visualisierung, um einen Data-Mining-Algorithmus anzuwenden, ist allerdings nur in MineSet möglich.

Das Ziel Clustering kann mit Hilfe der Systeme MineSet und HD-Eye erreicht werden. Das zuletzt genannte System ist speziell zu diesem Zweck entwickelt worden. Eine Änderung der Parameter des Cluster-Algorithmus wird interaktiv in beiden Systemen ermöglicht. Zusätzlich kann in MineSet das Ziel Klassifizierung erreicht werden. Hingegen unterstützt Advizor das Ziel Assoziation, indem Trendlinien, zum Beispiel in einem Scatterplot, hinzugefügt werden. Des Weiteren kann in MineSet und Advizor eine diskrete Animation verwendet werden, um die Analyse zu vereinfachen und einem Analyseziel näher zu kommen. Die Speicherung des Analyseergebnisses wird im System Advizor durch das Speichern einer Visualisierung in einer Datenbank realisiert. MineSet und SellTrend sichern den gesamten Analyseverlauf in einer History, der in MineSet auf eine neue Datenmenge wieder angewandt werden kann.

5 Klassifizierung von Systemen zur visuellen Analyse

Die Entwicklung eines Modells erfordert die Kenntnis über spezifische und detaillierte Informationen über das zu modellierende System. Diese Kenntnisse werden in der vorliegenden Arbeit durch die Betrachtung von bestehenden Klassifikationssystemen erlangt. Die Eigenschaften werden in ein Klassifikationssystem für Visualisierungen überführt, wodurch detaillierte Informationen zu Visualisierungen ermittelt werden. Als Visualisierungen werden dabei Darstellungen nach der Definition von Card, Mackinlay und Shneiderman gesehen:

Definition 2 (Visualisierung) *„Visualization is the use of computer-supported, interactive, visual representations of data to amplify cognition (...)“ [CMS99]*

In der Literatur werden verschiedene Ansätze der Klassifikation von Visualisierungstypen aus verschiedenen Sichtweisen (daten- [RM90], ziel-/aufgaben- [WL90, VPF06] und phasenorientiert [PHP03, SBC97]) beschrieben [Man99]. Des Weiteren existieren verschiedene Ansätze der Kombination von unterschiedlichen Sichtweisen, wie zum Beispiel [WBJ03], in der die Daten- und Aufgabensicht in einem Klassifikationssystem kombiniert wird. In [Kei01], aufbauend auf dem Datenklassifizierungsansatz in [Shn96], hat Keim ein Klassifizierungssystem mit Sicht auf die Aufgaben in der visuellen explorativen Analyse vorgestellt. Aufbauend auf diesem Klassifizierungsansatz (vgl. Abbildung 5.1), bestehend aus den orthogonalen Klassen *Visualisierungstechnik*, *Interaktions- und Verzerrungstechnik* sowie der Klasse der darzustellenden *Daten*, werden im Folgenden die verschiedenen Klassifizierungssysteme eingeordnet. Hierdurch wird das von Keim vorgestellte System um nötige Eigenschaften und die Klassen *Dynamische Visualisierungen und Animation* und *Zielorientierung* zur Modellierung ergänzt. Wichtige Eigenschaften sind als Piktogramme am Rand dargestellt und werden im anschließenden Abschnitt zu einem Klassifizierungssystem für Visualisierungen zusammengefasst. Zunächst wird auf die Klasse der zu visualisierenden Daten eingegangen.

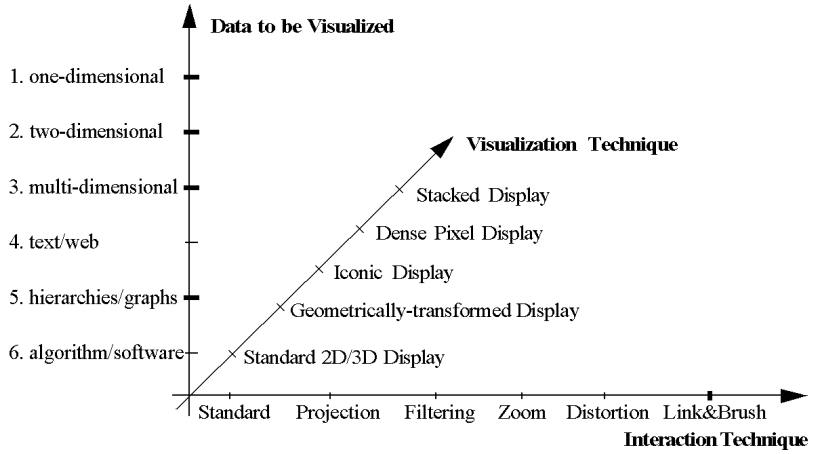
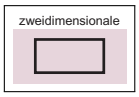
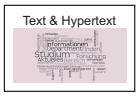


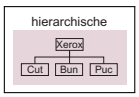
Abbildung 5.1: *Klassifizierungssystem für Visualisierungen nach Keim (vgl. [Kei01])*

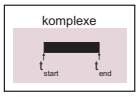
5.1 Art der zu visualisierenden Daten



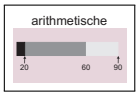


Datenmengen, wie sie in der visuellen Exploration verwendet werden, haben die Eigenschaft, dass sie meist aus sehr vielen Datensätzen bestehen. Ein Datensatz entspricht einer Beobachtung mit einer bestimmten Anzahl von Attributen. Die Anzahl der Attribute variiert in unterschiedlichen Datenmengen, von einigen wenigen bis hunderten, in einigen Fällen über tausende von Attributen [Pan05, Kei02a]. Datenmengen mit nur einem Attribut werden *eindimensionale* Daten genannt. Sie besitzen typischerweise ein kontinuierliches Attribut, welches eine vollständige Ordnung der Daten zulässt. Zeitabhängige Datensätze sind ein typisches Beispiel für *eindimensionale* Daten. Ein Punkt auf der Skala kann ein oder mehreren Datensätzen zugeordnet werden. Enthält der betrachtete Datensatz zum Beispiel Koordinatendaten (Längen- und Breitengrad), so wird er *zweidimensonaler* Datensatz genannt. Diese Form von Datensätzen hat zwei Attribute. Für Datensätze, die mehr Attribute aufweisen, wie zum Beispiel in relationalen Datenbanken, hat Keim die Eigenschaft von *mehrdimensionalen* Datensätzen eingeführt. Für diese Form der Daten, welche selten eine eigene auf den Raum bezogene Semantik besitzen, werden neuartige Visualisierungstechniken benötigt, da das transformieren auf die zweidimensionale Darstellung eines Bildschirms schwierig ist. Allerdings können nicht alle Datensätze, wie zum Beispiel *Hypertext Markup Language (HTML)*-Dokumente, durch Angabe einer festen Dimensionalität beschrieben werden. *Text-und Hypertext-* Datensätze sind im Zeitalter des World-Wide-Web wichtige Daten [Kei02a]. Viele der bekannten Visualisierungstechniken können solche Daten nicht darstellen, weshalb häufig eine Transformation, zum Beispiel durch eine Häufigkeitsanalyse der vorkommenden Wörter, in Beschreibungsvektoren durchgeführt wird, welche anschließend dargestellt werden.





Daten, die Beziehungen enthalten, lassen sich nicht durch die bisher vorgestellten Eigenschaften beschreiben. Diese Daten, wie beispielsweise Verbindungsdatensätze in Netzwerken, das Kaufverhalten von Kunden oder Dateisysteme, werden *hierarchische* Daten genannt. In [RM90] sind neben einer *hierarchischen* Beziehung zwei weitere Beziehungsarten innerhalb von Datensätzen vorgestellt worden. Zum Beispiel kann bei der Betrachtung zum Infektionstagen und Gesundungstagen für verschiedene Krankheiten, der Zeitraum zwischen den beiden Tagen als Heilungsdauer bezeichnet werden. Zur Darstellung dieser *komplexen* Datensätze sind spezielle Visualisierungstypen notwendig, die den Datensatz interpretieren können. Die Daten für den Infektionstag und Gesundungstag müssten als Anfangs- und Endpunkte interpretiert werden, um die Dauer darzustellen. Für diese Art von Daten sind jeweils angepasste Darstellungen notwendig. Deshalb wird häufig versucht *komplexe* Datensätze in einfachere umzuwandeln. Dies kann zum Beispiel dadurch erreicht werden, dass jeder Datensatz einzeln interpretiert wird. Also die Darstellung des Infektionstages und Gesundungstages als einzelne Datenpunkte. In der Abbildung 5.2 sind vergleichend zwei Diagrammformen dargestellt. Die linke Darstellung kann *komplexe* Datensätze darstellen, die Rechte ist um die Logik zur Darstellung von komplexen Datensätzen nicht ergänzt worden. Bei dieser sind die Daten eine flache Struktur umgewandelt. Der Zusammenhang zwischen den Daten, die Heilungsdauer, wird allerdings nicht explizit dargestellt. Als weitere Beziehungsart in Datensätzen wird in [RM90] ein *algebraischer* Zusammenhang zwischen Daten aufgeführt. Diese Beziehungsart wird anhand der Betrachtung des folgenden Beispiels deutlich. Angenommen ein Datensatz besteht aus den Attributen Forschungskosten, Herstellungskosten und Produktionskosten bezogen jeweils auf ein Heilmittel. Die *arithmetische* Beziehung in dem Datensatz besteht darin, dass die Gesamtkosten für ein Heilmittel sich aus der Addition der drei Einzelattribute, bzw. Einzelkosten, ergeben könnten. In der Abbildung 5.3 wird dieses Beispiel dargestellt. Wenn ein Visualisierungstyp diese Form der Beziehung darstellen kann, können leicht *arithmetische* Zusammenhänge erkannt werden. Im Beispiel sind dies die Verteilung der einzelnen Kostenfaktoren für



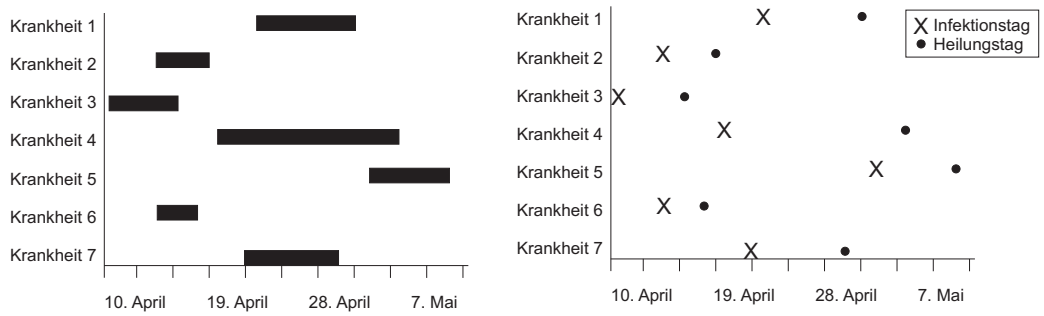


Abbildung 5.2: *Darstellung komplexer Beziehungen in Balken- und Punktdiagramm*

jedes Heilmittel und gleichzeitig die Darstellung der Gesamtkosten. Diese Beziehungsarten können

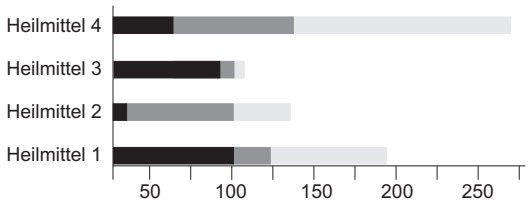
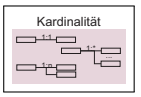


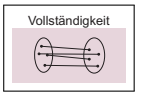
Abbildung 5.3: *Darstellung algebraischer Beziehungen mit Hilfe eines Balkendiagramms*

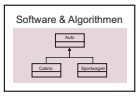


durch die Eigenschaft der *Kardinalität*, angelehnt an die Definition in Informationssystemen, genauer beschrieben werden [RM90].

Definition 3 (Kardinalität in Informationssystemen) *„Die Kardinalität gibt an, wie viele Ausprägungen einer Entität an einer Beziehung (. . .) beteiligt sein können.“ [Wie08]*

In Abhängigkeit des Beziehungsgrades, können die Daten von unterschiedlichen Visualisierungen dargestellt werden. Ein Datenwert kann zu Einzelwerten, einer festen Anzahl von Werten oder einer variablen Anzahl von Werten in Beziehung stehen. Die in der Abbildung 5.2 dargestellten Krankheiten und Heilungstage stehen in einer festen Anzahl zu Krankheiten in Beziehung. Für jede Krankheit gibt es exakt einen Heilungstag und einen Infektionstag. Auch die arithmetischen Datensätze der Abbildung 5.3 stehen zu einer festen Anzahl von Werten in Beziehung. Bei *hierarchischen* Beziehungen stehen oftmals variable Anzahlen von Werten in Beziehung, wie beispielsweise bei Dateisystemen, in denen die Verschachtelungstiefe von Ordnern variabel ist. Eine andere Eigenschaft von Beziehungen zwischen Daten ist die *Vollständigkeit* [RM90], welche in Anlehnung an die mathematische Eigenschaft Surjektivität angibt, ob ein Element einer Dimension auf ein anderes Element einer anderen Dimension abgebildet werden kann. Die beispielhafte Darstellung der Daten von Infektionstagen und Heilungstagen ist *vollständig*, da jeder Infektionstag wie auch Heilungstag einer Krankheit zugeordnet werden kann. Es sind allerdings auch unvollständige Datensätze vorstellbar, in denen nicht jedes Element einer Achse auf ein Element einer anderen Achse dargestellt werden kann. Dies kann zum Beispiel in fehlenden Datensätzen in der Datenbank oder einer fehlenden Semantik der anzuzeigenden

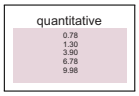




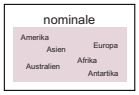
Daten begründet sein. Eine mehr semantische Abhängigkeit zwischen Datensätzen ist in *Software und Algorithmen* zu finden. Quellcode besteht aus vielen voneinander abhängigen Quellcodezeilen. Diese Datenform kann zum Beispiel durch die Analyse der Methodenaufrufe, durch die Visualisierung aller von einem Fehler betroffenen Teile eines Programms oder durch Analyse des Versionsverwaltungssystems visualisiert werden, um somit Fehler, Auswirkungen oder Optimierungsmöglichkeiten des analysierten Programms zu erkennen.

In der visuellen, explorativen Analyse werden Daten nicht nur durch Darstellungen repräsentiert, sondern es sind auch vielfältige Interaktionen möglich. Diese vom Benutzer ausgeführten Aktionen münden in Operationen, die vom System ausgeführt werden. Operationen, die bei der Analyse oftmals durchgeführt werden, sind unter anderem das Suchen von Datensätzen in Datenmengen, das Vergleichen von unterschiedlichen Datensätzen und das Sortieren dieser [VPF06, WL90]. Dies erfordert, dass Datensätze untereinander verglichen werden können. Es ist also eine Ordnungsrelation erforderlich, die es ermöglicht Elemente in einer Datenmenge miteinander zu vergleichen und einzuordnen.

Definition 4 (Ordnungsrelationen) *„Eine reflexive, antisymmetrische und transitive binäre Relation auf einer Menge M wird Ordnungsrelation genannt.“ [Pri00]*



Roth und Mattis unterscheiden in [RM90] die drei Ordnungsrelationen *quantitativ*, *ordinal* und *nominal*. In [CSCC99] wird weiterhin die Relation *kategorisch* eingeführt. In *quantitativen* Datensätzen sind die Daten numerisch geordnet, zum Beispiel nach Anzahl der Krankheitsfälle von 0 bis 100. Darstellungen, die diese Daten repräsentieren, können sehr effektiv eine Abbildung auf zum Beispiel eine Größenachse, Positionsachse oder Farbachse durchführen. Allerdings ist eine Darstellung in Visualisierungen, die Daten als Formen (zum Beispiel als Stern, Kreis, Quadrat, Dreieck) repräsentieren, unvorteilhaft. Ein Nutzer kann die Formen nur sehr schwierig einem Datenwert zuordnen. *Ordinale* Datensätze sind immer vergleichbar nach ihrer Semantik, zum Beispiel Ansteckungsgefahr (gering, mittel, hoch, sehr hoch). Im Gegensatz zu *quantitativen* Daten ist es notwendig, dass die Visualisierung jedes Element darstellt, da Zwischenwerte nicht berechnet werden können. Ungeordnete Datensätze, wie zum Beispiel die Namen der Krankheiten, werden *nominale* Datensätze genannt. Diese Daten können effektiv durch Visualisierungen dargestellt werden, welche keine Ordnung in die Daten hinzufügen, wie zum Beispiel Farb- oder Formrepräsentationen, vorausgesetzt die Datenmenge ist gering. Hierdurch wird eine Fehlinterpretation der Daten vermieden. *Kategorische* Datensätze sind das Ergebnis einer vorherigen Kategorisierung. In Abhängigkeit des Kategorisierungsparameters können die Daten wie *ordinale* Daten geordnet werden. Zum Beispiel bei einer Kategorisierung nach Anzahl der Todesfälle in die Gruppen `keine`, `wenig`, `viele`. Bei anderen Kategorisierungsparametern, wie zum Beispiel Kontinente, entstehen Datensätze, die wie *nominale* Daten nicht geordnet werden können. Die Tabelle 5.1 stellt die Ordnungsrelation und mögliche visuelle Repräsentationen zusammengefasst gegenüber.

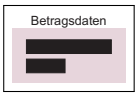


Quantitative Datensätze können weiterhin in *Koordinaten-* und *Betragsdaten* unterschieden werden [RM90]. *Koordinatendaten* sind Datensätze, in denen jedes Element einen bestimmten Punkt oder eine Position zeitlich, räumlich, etc. definiert, wie zum Beispiel Kalendertage, Breitengrad oder Zeitzone.

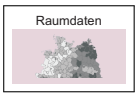
Quantitativ	Ordinal	Nominal	Kategorisch
Position	Position	Position	Farbton
Länge	Farbsättigung	Farbton	Form
Orientierung	Farbton	Textur	Farbsättigung
Größe	Textur	Farbsättigung	Textur
Farbsättigung	Länge	Form	Größe
Farbton	Orientierung	Länge	
Textur	Größe	Orientierung	
Form	Form		

Tabelle 5.1: *Vorgeschlagene visuelle Repräsentationen zu Ordnungsrealtionen (vgl. [CSCC99])*

Im Gegensatz dazu, bestimmen *Betragsdaten* einen Wertebereich, wie zum Beispiel Zahl von Tagen oder Gewicht. Eine Visualisierung welche für *Betragsdaten* entwickelt wurde, stellt *Koordinatendaten* im Regelfall inkorrekt dar. Es kann zum Beispiel vorkommen, dass *Koordinatendaten* nicht mehr als Zeitpunkte angezeigt werden, sondern fälschlicherweise als Zeiträume.

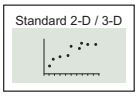


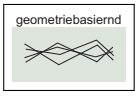
Anknüpfend an das vorherige Beispiel, wird bereits ein Eindruck über die Wichtigkeit der Dateneigenschaft *Zeit* gegeben. *Zeit* ist in vielen Anwendungsdomänen, wie zum Beispiel Transport, Produktion oder Gesundheit, eine wichtige Eigenschaft [ABM+07]. Im Gegensatz zu anderen *quantitativen* Datendimensionen hat *Zeit* eine eigene semantische Struktur, was die Komplexibilität erhöht. Die hierarchische Struktur und Granularität in der *Zeit*, wie zum Beispiel Minuten, Stunden oder Tage, ist anders als bei den meisten anderen *quantitativen* Dimensionen [ABM+07]. Dies erfordert eine besondere Betrachtung bei der Erstellung des Modells. Eine weitere Dateneigenschaft, die besondere Bedeutung besitzt, sind *räumliche* Datensätze [KPS03]. *Räumliche* Daten beschreiben Objekte oder Phänomene mit einer spezifizierten Position und evtl. einer Ausdehnung in der echten Welt. *Räumliche* Daten können keine Ausdehnungen haben, wie beispielsweise Positionsdaten, eine eindimensionale, wie zum Beispiel Grenzen oder Telekommunikationsnetze, oder eine zweidimensionale. Zweidimensionale Ausdehnungen sind Flächen, wie zum Beispiel Flüsse oder Länder [KPS03].



5.2 Visualisierungstechnik

Keim schlägt als weitere Klassifizierungseigenschaft die Einteilung nach der Visualisierungstechnik vor, die zur Darstellung der Daten verwendet wird. Weit verbreitete 2-D- und 3-D-Visualisierungstechniken, wie zum Beispiel X-Y-Graphen, Karten oder Liniendiagramme, fasst Keim in der Klassifizierungseigenschaft *Standard 2-D-/ 3-D*-Technik zusammen. In [WBJ03] wird diese Gruppe um $2\frac{1}{2}$-D-Techniken ergänzt. $2\frac{1}{2}$-D-Visualisierungen sind 2-D-projizierte 3-D-Darstellungen. 3-D-Techniken ermöglichen die freie Wahl der Perspektive im virtuellen Raum. *Standard 2-D- /3-D*-Techniken führen keine Transformation der Daten durch, sondern stellen diese möglichst direkt dar. Es kann jedoch vorkommen, dass auf Grundlage der hohen Dimensionalität der Daten eine Transformation der Daten erfolgen muss, wie es bei *geometriebasierten*-Techniken geschieht. Die Grundidee dieser Techniken ist die Transformation und Projektion multidimensionaler Datensätze, um diese visuell darzustellen. Es können dabei sowohl 2-D- als auch 3-D-Visualisierungen verwendet werden. Als weitere Technik für hochdimensionale Daten können *Icon und Glyph*-Techniken aufgeführt werden. Diese Zielen auf





die Fähigkeit des Menschen Formen von Objekten schnell unterscheiden und wiedererkennen zu können. Bei *Icon und Glyph*-Techniken wird die visuelle Darstellung der Daten dadurch erreicht, dass Eigenschaften eines Icons oder Glyphs, wie zum Beispiel Form oder Farbe, auf Attribute der Datenmenge abgebildet werden. Für jeden Datensatz der Datenmenge entsteht genau eine visuelle Form, die alle Attribute gleichzeitig darstellt. Sind Datensätze bezüglich der Darstellungsdimensionen dicht zusammen, entstehen in der resultierenden Visualisierung ähnliche Darstellungen. Die Schwierigkeit besteht darin, die Zuordnung zwischen den Attributen und den Eigenschaften der Form so zu wählen, dass korrelierende Dimensionen ebenfalls korreliert dargestellt werden. Außerdem besteht das Problem, dass bei einer großen Anzahl von Datensätzen die Anzahl der Figuren entsprechend steigt, und somit der Vorteil der leichten Identifikation in der Menge der Figuren untergeht. Dieses Problem wird durch die *Pixeltechniken* umgangen, die die höchste Darstellungsdichte von Datensätzen besitzen und somit die meisten Datensätze auf einmal darstellen können [KK94]. Die Darstellungstechnik basiert darauf, dass jedem Datenpunkt ein Bildpunkt zugeordnet wird, indem der Bildpunkt entsprechend des Wertes des Datenpunktes eingefärbt wird. Es können zum Beispiel eckige [KAK95] oder runde [AKK96] Flächen gefüllt werden. Die Abbildung 5.4 zeigt das Schemata beider Pixelzurordnungstechniken. Bei einer Auflösung von $1680 * 1050$ Pixeln können über 1,7 Millionen Pixel und damit Datenpunkte

Pixel

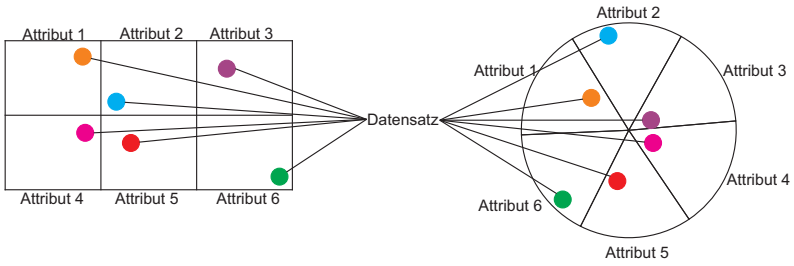


Abbildung 5.4: *Pixelzuordnungstechniken: Links eckige und rechts runde Flächen (vgl. [BH02])*

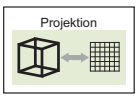
dargestellt werden. Die Position des Datenpunktes wird durch das Attribut, welches dieser Bildpunkt darstellt, definiert. Datenpunkte werden nach ihrem Attribut gruppiert. Ein Datensatz wird somit über das Bild verteilt und steht nur über die relative Position innerhalb der Teilbereiche mit anderen Datensätzen in Beziehung. Dieser zweidimensionale Ansatz wird durch die Voxel-Technik [FIET01] auf drei Dimensionen erweitert, wodurch mehr Datensätze dargestellt werden können. Durch diese Visualisierungstechnik ist es möglich Beziehungen, Korrelationen und Ausnahmen in den Datensätzen zu finden. Bestehen in der Datenmenge allerdings hierarchische Beziehungen können diese durch die bisher beschriebenen Visualisierungstechniken nicht dargestellt werden. Zur Darstellung von hierarchischen Daten werden *geschachtelte* Visualisierungstechniken benötigt. Diese stellt Daten in Form hierarchischer, untereinander aufgeteilter Untereinheiten dar, wobei die Wertebereiche der Attribute ineinander geschachtelt werden. Die Wahl der Attribute ist für die Qualität der Visualisierung entscheidend und muss deshalb sorgfältig gewählt werden.

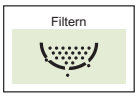
geschachtelt

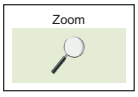
5.3 Interaktions- und Verzerrungstechnik

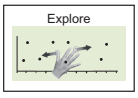
Selbst bei sehr komplexen Visualisierungsverfahren ergibt sich das Problem, dass bei großen Datenmengen die zur Verfügung stehende Darstellungsfläche zu gering ist, um alle Daten gleichzeitig und verständlich darzustellen. Es besteht somit die Notwendigkeit zielgerichtet in den Daten zu navigieren

und einzelne Aspekte der Datenmenge gezielt anzeigen zu lassen [SBM08a]. Um dieses Ziel zu erreichen, stehen nach [Kei01] in dieser Klassifizierungseigenschaft fünf verschiedene Interaktions- und Verzerrungstechniken zur Verfügung. Die Techniken *Projektion*, *Filterung*, *Zoomen* und *Verzerrung* sind Interaktionsformen bezogen auf eine Visualisierung. Zur Verknüpfung verschiedener Diagramme stehen *Linking- und Brushing*-Techniken zur Verfügung. Mit Hilfe der *Projektionstechnik* wird der Benutzer in die Lage versetzt, Datenprojektionen dynamisch zu verändern, um so eine andere Sicht auf multidimensionale Datensätze zu erhalten. Beispielsweise können multidimensionale Datensätze als eine Kombination aus zweidimensionalen Datensätzen dargestellt werden. Da die Anzahl der möglichen Kombinationen exponentiell ansteigt, ist es sinnvoll den Analysten die Anzeige der Kombinationen dynamisch ändern zulassen. Bei großen Datenmengen besteht allerdings das Problem der Übersichtlichkeit. Mit Hilfe von *Filtertechniken* wird versucht dieses Problem zu lösen. Die durch die *Filterung* entstehenden Teilmengen können entweder durch direkte Interaktionen mit der Visualisierung oder durch spezielle Anfragesprachen gebildet werden. Eine direkte Auswahl der gewünschten Teilmenge ist bei großen Datenmengen schwierig. Ein Problem bei der Spezifikation mit Hilfe von Abfragen ist, dass die Teilmenge oft nicht das gewünschte Ergebnis liefert [Kei02a]. Eine andere Möglichkeit, um eine Übersicht über die Datenmenge zu erhalten, ist es, die Daten in einer reduzierten bzw. aggregierten Form darzustellen. Mit Hilfe von *Zoom*-Techniken kann der Analyst die Details, ausgehend von einer Überblicksdarstellung, betrachten. *Zoom* bedeutet jedoch nicht nur, dass die visuelle Repräsentation optisch vergrößert dargestellt wird, sondern auch, dass der vergrößerte Bereich mit mehr Details dargestellt wird. Zum Beispiel bei der Betrachtung von Heilmitteln gegenüber Tumorerkrankungen, welche nach dem *International Statistical Classification of Diseases and Related Health Problems (ICD)*-Klassifikationssystem gruppiert sind. Tumorerkrankungen werden in verschiedene Gruppen eingeteilt, die als Überblick dargestellt werden können. Möchte der Analyst nun genau wissen, welches Heilmittel für eine bestimmte Tumorerkrankung verwendet werden kann, so kann dieses durch eine *Zoom*-Operation durchgeführt werden. Allerdings kann es vorkommen, dass dem Analysten nicht mehr der gesamte Datenbestand aufgezeigt werden kann. Damit zu den nicht sichtbaren Bereichen vorgedrungen werden kann, sind in [YKSJ07] *Explore*-Techniken vorgestellt worden. In der Regel verändern diese Techniken die Daten nicht, sondern zeigen nur einige bisher verdeckte Elemente an. Die bekannteste Technik ist das Verschieben (Panning). Panning ist vergleichbar mit dem verschieben einer Kamera, welche die Ansicht der Daten aufzeigt. Durch Verschieben der Kameraposition erhält der Analyst Einblick in bisher nicht gezeigte Daten. Um den Überblick über einen Datenbestand zu behalten und gleichzeitig Details zu betrachten, können auch *Verzerrungstechniken* verwendet werden. Diese Techniken ermöglichen es einen Bereich der Daten detaillierter darzustellen, während der Gesamtüberblick bestehen bleibt. Hyperbolische oder sphärische Verzerrungen sind bekannte Vertreter dieser Technik.

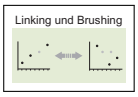




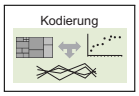


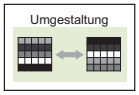


Oftmals haben Visualisierungstypen bestimmte Stärken, aber in vielen Bereichen auch einige Nachteile. Im Abschnitt 5.2 sind verschiedene Visualisierungstechniken aufgezeigt worden. Diese haben unterschiedliche Schwächen und Stärken. Pixeltechniken können zum Beispiel hierarchische Daten nicht darstellen. Es kann aber durchaus sinnvoll sein, hierarchische Daten einer Ebene mit Hilfe der Pixeltechnik zu visualisieren, um somit lokale Beziehungen erkennen zu können. Aus diesem Grund ist es sinnvoll, Datensätze mit verschiedenen Visualisierungstypen zu betrachten. Die *Linking- und Brushing*-Techniken klassifizieren Techniken, die eine Verknüpfung von verschiedenen Visualisierungen beschreiben. Eine mögliche Verknüpfungsform ist zum Beispiel die Selektionsverknüpfung. *Selektionstechniken* bieten dem Anwender die Möglichkeit, interessante Datenelemente zu markieren. Mit Hilfe einer Markierung können Analysten Datensätze leicht in einer großen Datenmenge nachverfolgen oder bei Änderungen wiedererkennen [YKSJ07]. Wenn ein Datenpunkt in einer Visualisierung



selektiert wird, werden diese auch in den verknüpften Visualisierungen ausgewählt dargestellt. Somit sind Korrelationen und Zusammenhänge zwischen den evtl. auch unterschiedlichen Visualisierungstypen oder in unterschiedlichen Aggregationsstufen dargestellten Daten möglich. Es hat sich gezeigt, dass durch die Verknüpfung von verschiedenen Visualisierungen mehr erkannt werden kann, als bei Betrachtung der einzelnen Visualisierungen getrennt voneinander [Kei02a]. Eine weitere Technik, welche unmittelbar beeinflusst, wie der Nutzer Beziehungen und Verteilungen von Daten erkennen kann, sind *Kodierungstechniken* [YKSJ07]. Diese ermöglichen dem Benutzer die grundlegende visuelle Repräsentation der Daten einschließlich der Optik, wie zum Beispiel Farbe, Größe oder Form, von jedem Datenelement zu ändern. Eine Möglichkeit der *Kodierungstechnik* ist der Ansichtswechsel zwischen einem Punktdiagramm und einem Tortendiagramm. Durch den Wechsel der Ansicht in eine Tortendarstellung können zum Beispiel besser prozentuale Verteilungen analysiert werden. Eine andere Möglichkeit der Interaktion ist die Änderung der Farbkodierung oder Anpassung der Größen von Darstellungselementen, wodurch ein Vergleich der Elemente untereinander erleichtert werden kann. Das Vergleichen von Datenelementen wird ebenfalls durch *Umgestaltungstechniken* erleichtert, die dem Analysten die Möglichkeit bieten, unterschiedliche Perspektiven auf die Daten durch Änderung der räumlichen Anordnung der visuellen Elemente zu erhalten [YKSJ07]. *Umgestaltungstechniken* können zum Beispiel die Verschiebung von Spalten und Zeilen in Tabellen, ein Achsentausch in einem Punktdiagramm oder die freie Verschiebung von Datenwerten in einer Darstellung sein.





5.4 Dynamische Visualisierungen und Animation

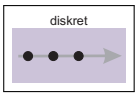
Jain beschreibt in seinem Ausblick zum nächsten Millennium [Jai99] die steigende Wichtigkeit von Simulationen und Animationen. Die von Keim vorgestellten drei Achsen zur Klassifizierung von Visualisierungen vernachlässigen den Aspekt der Animation gänzlich. Somit könnten bestehende Systeme, die eine Animation ermöglichen, nicht eingeordnet werden. Mit Hilfe der Klasse *Animation* kann auch diese Eigenschaft einer Visualisierung klassifiziert werden. Animationen können zum Beispiel für Präsentationszwecke verwendet werden, indem der Moderator ein vorher analysiertes Szenario vorstellt. Im Bereich der explorativen Analyse ist die Entdeckung von Trends ein Ziel der Analyse [KMS+08].

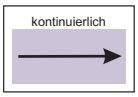
Definition 5 (Trend) *„The general direction in which something tends to move.“ [Dic09]*

Die Animation über die Zeit ist eine Möglichkeit dieses Ziel zu erreichen [RFF+08]. Dieses kann am folgenden Beispiel verdeutlicht werden. Wird die Lebenserwartung (Y-Achse) gegenüber der Kindersterblichkeit (X-Achse) für verschiedene Länder (Punkte) und deren Bevölkerungsdichte (Farbe der Punkte) bezogen auf ein Jahr betrachtet, können Aussagen über die zu diesem Jahr gehörigen Datensätze getroffen werden. Möchte der Analyst allerdings eine Aussage über den Trend der Lebenserwartung und Kindersterblichkeit treffen, muss er entweder alle Jahre gleichzeitig betrachten oder die Jahre einzeln nacheinander. Wenn alle Jahre in einer Visualisierung dargestellt werden, ist nicht klar, welcher Datenpunkt zu welchem Jahr gehört. Wenn der Analyst die einzelnen Jahre manuell ausgewählt betrachtet, vergeht immer sehr viel Zeit zwischen den einzelnen Darstellungen, so dass der Zusammenhang nicht unbedingt klar wird. Bei Verwendung einer Animation kann der Analyst in kurzen Zeitabständen die jeweiligen Datenpunkte für die einzelnen Jahre nacheinander betrachten, und

somit den Trend zu einer höheren Lebenserwartung bei einer geringeren Kindersterblichkeit erkennen. In der Abbildung 5.5 ist die Animation dieser Daten anhand von drei Einzelbildern beispielhaft dargestellt.

Abbildung 5.5: *Darstellung einer diskreten über die Zeit animierten Visualisierung [RFF+08]*





Werden die Punkte nacheinander, also jeweils nach einer bestimmten Zeitspanne, eingeblendet ist dies eine *diskrete* Animation [WBJ03]. Die Änderungen können proportional oder nicht proportional zu einer Zeitachse durchgeführt werden. Eine *diskrete* Animation von Daten kann zum Beispiel mit Hilfe eines Punktdiagramms durchgeführt werden. Liegen quantitative Daten vor, kann auch eine *kontinuierliche* Animation durchgeführt werden. In *kontinuierlichen* Animationen findet eine fortlaufende Datenänderung statt. Der Verlauf wird noch deutlicher, wenn ein Trace der zukünftigen und vergangenen Datenänderungen hinzugefügt ist [RFF+08].

5.5 Zielorientierung

Wehrend und Lewis stellen in [WL90] Ziele als eine Klassifizierungseigenschaft, die mit einer Visualisierung erreicht werden sollen, heraus. Abhängig davon welches Ziel der Analyst verfolgt, sind andere Visualisierungstypen notwendig. In [BH02] werden speziell für den Bereich der visuellen Analyse die Ziele *Assoziationsregelfindung*, *Klassifizierung* und *Clusterung* eingeführt. Diese bilden eine weitere Klassifizierungseigenschaft, die von den vorherigen nicht erfasst wurde. Es gibt eine große Anzahl von Visualisierungstechniken, welche die unterschiedlichen Data-Mining-Ziele unterstützen. Bei dem Ziel *Assoziationsregelfindung* ist es von Interesse, Muster und Trends in transaktionalen Datenbanken zu finden. Assoziationsregeln sind statische Beziehungen zwischen zwei oder mehr Elementen in einem Datensatz. Zum Beispiel in einer Supermarktwarenkorbanwendung drücken Assoziationen die Beziehungen zu den Waren aus, die zusammen gekauft wurden. Es ist interessant, wenn Brot gekauft wird, dass in 70% der Fälle ebenso Milch eingekauft wird. Die Wahrscheinlichkeit in der das Ereignis auftritt wird Vertrauen bzw. Vertrauensintervall genannt. Für ein gegebenes Problem und einen definierten Vertrauensbereich gibt es effiziente Algorithmen um alle Assoziationsregeln zu bestimmen [AMS+96]. Ein Problem ist jedoch, dass der resultierende Satz von Regeln gewöhnlich ziemlich groß ist, vor allem wenn das Vertrauensintervall groß ist. Die Verwendung eines kleinen Vertrauensintervalls ist nicht effizient, da nützliche Regeln evtl. übersehen werden können [BH02]. Visualisierungstechniken unterstützen den Anwender bei der Verfolgung des Zieles, indem sie eine interaktive Selektion von Vertrauensintervallen ermöglichen. Eine Möglichkeit der Visualisierung von Assoziationen ist es, die Elemente auf die X und Y Achse aufzutragen und den Zusammenhang

zwischen diesen durch eine Balkenhöhe darzustellen. Im Beispiel könnte das Ziel aber ebenso darin bestehen, verschiedene Klassen von Einkäufern zu identifizieren. *Klassifikation* wird in [BH02] als weiteres Ziel definiert. *Klassifikation* ist der Prozess der Entdeckung eines Klassifikationsmodells basierend auf einem Trainingssatz mit bekannten Klassendaten. Um das Klassifikationsmodell zu erstellen, werden die Attribute vom Trainingsdatensatz analysiert, sowie eine zielgenaue Beschreibung oder ein Modell von den Klassen basierend auf den Attributen entwickelt. Diese Beschreibung wird benutzt um einen unbekannten Datensatz zu klassifizieren. Es gibt eine große Menge von Algorithmen zur Lösung von Klassifikationsaufgaben, wie zum Beispiel Ansätze mit Entscheidungsbäumen, neuronalen Netzen oder genetischen Algorithmen. Allerdings werden die meisten Algorithmen als Black-Box-Ansätze verwendet, wodurch es oftmals schwierig ist das Entscheidungsmodell zur Klassifizierung anzupassen und zu optimieren [BH02]. Visualisierungstechniken unterstützen den Analysten bei der Wahl der Parameter, indem sie gleichzeitig die Einsicht in das Klassifikationsergebnis und die Parametereinstellung ermöglichen. Im Gegensatz zur *Klassifikation* wird bei der *Clusterung* in den meisten Fällen kein Trainingssatz benötigt. Durch das Clustern können unter anderem Muster in den Daten erkannt werden. Bei der *Clusterung* werden die Daten als Zufallsvariablen aufgefasst und anhand von bestimmten Eigenschaften, die durch Algorithmen bestimmt werden, klassifiziert. In der Literatur wird ein breites Spektrum von Clusteralgorithmen vorgeschlagen, die meistens auf Annahmen über Eigenschaften von Clustern basieren. Abhängig von der Parametrisierung erhält der Analyst verschiedene Clusterresultate. Im zwei- und dreidimensionalen Raum können die Ergebnisse leicht erforscht werden, zum Beispiel durch die Anzeige in einem X-Y-Diagramm. In höherdimensionalen Räumen ist der Effekt der Änderung der Parameter allerdings viel schwieriger zu verstehen. Einige höher dimensionale Techniken versuchen die Eigenschaften in zwei- oder dreidimensionale Projektionen von Daten wiederzugeben. Diese Techniken sind aber schwierig für hoch-dimensionale Datensätze zu übernehmen, speziell wenn die Cluster nicht klar voneinander getrennt sind oder Rauschdaten enthalten sind. Rauschdaten sind in diesem Fall Daten, die zu keinem Cluster passen. Für solche Datensätze werden anspruchsvollere Visualisierungstechniken benötigt [BH02].

5.6 Eine graphische Notation zur Klassifizierung von visuellen Analysesystemen

Im vorangegangenen Abschnitt ist ausgehend von dem Klassifikationssystem in [Kei01] mit den Klassen *zu visualisierende Daten*, *Visualisierungstechnik* sowie *Interaktions- und Verzerrungstechnik* ein Klassifikationssystem bestehend aus fünf Eigenschaften beschrieben worden. Keim klassifiziert dabei nach den Eigenschaften bezogen auf die Daten, mit welcher Technik diese grundlegend dargestellt werden und möglichen Interaktionen auf bzw. mit Visualisierungen. Es hat sich gezeigt, dass diese drei Klassen unzureichend für ein Visualisierungsmodell sind, weshalb die Klasse *Zielorientierung*, um zu klassifizieren, welches übergeordnete Data-Mining-Ziel verfolgt wird, hinzugefügt wurde. Außerdem ist die Wichtigkeit von *Dynamischen Visualisierungen und Animationen* herausgestellt worden, die aus diesem Grund als weitere Klassifizierungsklasse eingeführt wurde. Die Klasse der *zu visualisierenden Daten* kann untergliedert werden in Eigenschaften von Daten, den Beziehungsarten zwischen diesen und wie die Daten geordnet werden können. Die Klassen sind in der Abbildung 5.6 dargestellt.

Bei der Definition der Klassen ist darauf geachtet worden, dass diese unabhängig von anderen Klassen sind. Somit bleibt die in [KPS03] beschriebene Orthogonalität der Klassen bestehen. Unter Orthogonalität wird verstanden, dass jede Klassifizierungseigenschaft in Verbindung mit jeder anderen Klassifizierungseigenschaft verwendet werden kann und beinhaltet damit auch eine vollständige

Disjunktivität der Klassen. Dies hat zur Folge, dass die unterschiedlichen Klassifikationseigenschaften auch unabhängig voneinander betrachtet werden können. Sollen zum Beispiel nur *Interaktions- und Verzerrungstechniken* von verschiedenen Visualisierungen klassifiziert werden, reicht es aus, nur diese Klasse zu betrachten. Die Merkmale für das Klassifizierungssystem und die einzelnen Klassen sind in der Abbildung 5.6 auf der folgenden Seite zusammengefasst. Mit Hilfe dieser Klassifizierung wird im nachfolgenden Abschnitt das *VAT*-System entwickelt.

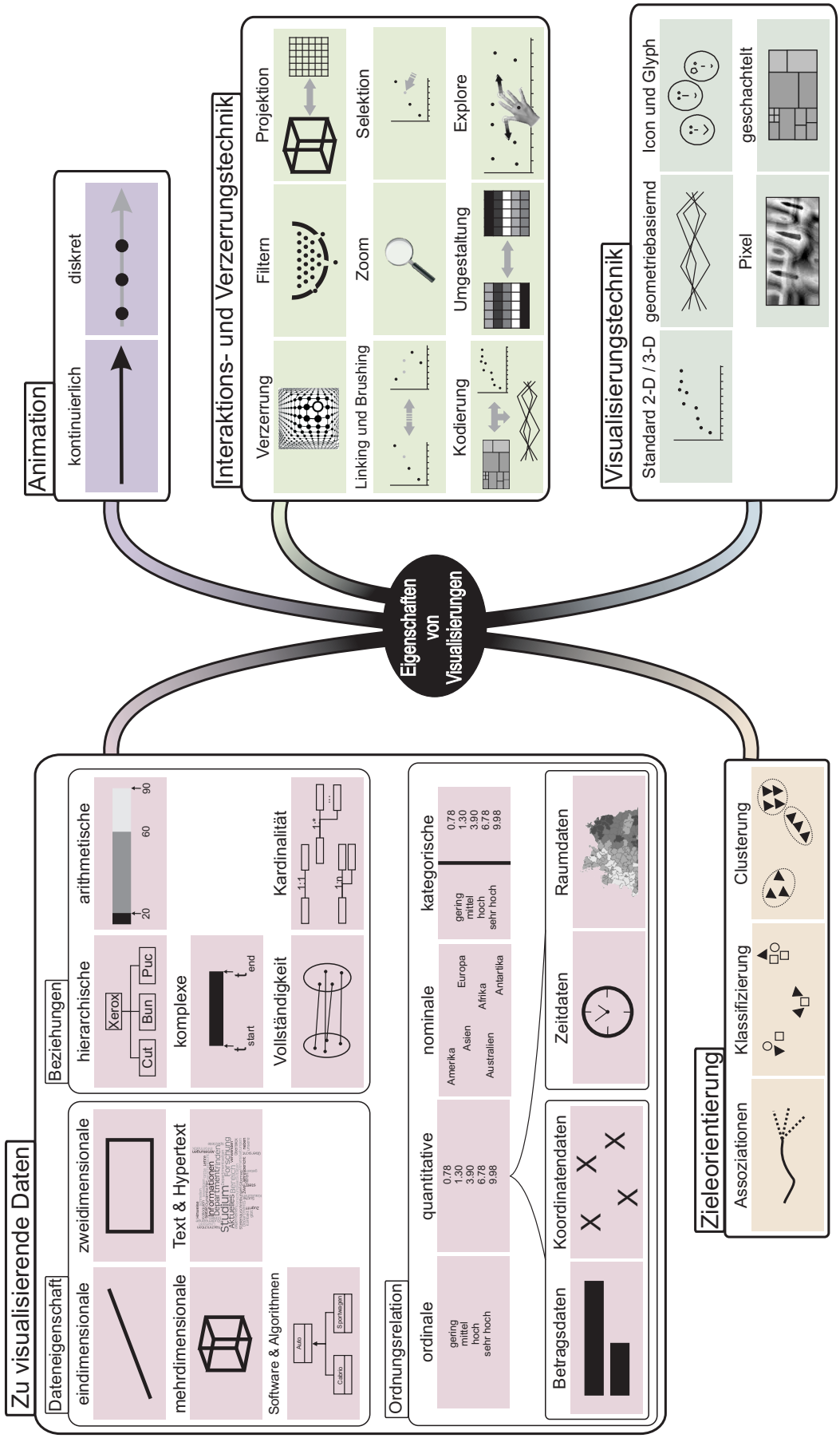


Abbildung 5.6: *Entwickeltes Klassifikationssystem*

6 Das Visual-Analytics-Transformation-System

Die Anforderungen an das *VAT*-System ergeben sich aus dem Analyseszenario, welches im Kapitel 1.3 vorgestellt wurde. Anhand dieses Leitszenarios werden Teilszenarien extrahiert und Anwendungsfälle abgeleitet. Anschließend werden hieraus die Anforderungen an ein System zur visuellen Analyse aufgestellt. Aufbauend auf diesen Anforderungen, wird das *VAT*-System zur Umsetzung dieser vorgestellt. Hierbei wird jeweils auf die Anforderungen, die mit der jeweiligen Eigenschaft des Entwurfs erfüllt werden, verwiesen. Einfluss auf das Gesamtkonzept haben die im Kapitel 3.1 vorgestellten Modelle für Visualisierungsssysteme und die hieraus gewonnenen Erkenntnisse. Zunächst wird dargestellt, welches Visualisierungsmodell als Grundlage für das Konzept verwendet wird. Anschließend wird das Modell mit Hilfe der im Kapitel 2.1 verwendeten Notation entwickelt. Begonnen wird mit Beschreibung der Anwendungsfälle.

6.1 Anwendungsfälle

Anwendungsfälle beschreiben mit welchen Aktionen Akteure ein vorgegebenes Ziel in einem System erreichen. Der Akteur ist in diesem Fall ein Analyst, welcher eine Datenmenge explorativ analysieren möchte. Bei der Analyse wird nach dem Visual-Analytics-Prozess (vgl. Kapitel 1.1) vorgegangen. Bei der Durchführung der Analyse kann der Analyst unterschiedliche Visualisierungen und eine Menge von Interaktionen anwenden, die in den nachfolgenden Anwendungsfällen beschrieben sind.

[UC.1] Datenquelle auswählen

Bevor eine Analyse durchgeführt werden kann, ist es notwendig eine Datenquelle auszuwählen. Diese stellt alle benötigten Metainformationen (Kennzahl- und Dimensionseigenschaften) und die Datenmenge bereit. Im Analyseszenario wählt der Analyst eine raumbezogene epidemiologische Datenquelle mit Daten über Krebserkrankungen mit Todesfolge und führt damit den ersten Schritt in dem Analyseprozess aus.

Beschreibung	Der Akteur wählt eine Datenquelle mit evtl. epidemiologischen Daten für die Analyse aus. Diese Daten stehen nach Auswahl für die Analyse zur Verfügung und können für Visualisierungen oder zur Anwendung von statistischen Verfahren verwendet werden.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Das Programm ist gestartet. 2. Die Datenquelle ist im System eingebunden. 3. Eine Verbindung kann zu dieser Quelle hergestellt werden.
Nachbedingungen	Der Akteur hat für die weitere Analyse eine Datenquelle ausgewählt.
Ausnahmen	–
Erweiterungen	Ein Analyst kann mehrere unterschiedliche Datenquellen zur Analyse gleichzeitig verwenden.

Standardablauf	1. Ein Analyst wählt im Menü eine Datenquelle. 2. Die Quelle wird geladen und steht anschließend zur Analyse zur Verfügung.

[UC.2] Anwendung eines statistischen Verfahrens

Durch die Anwendung von statistischen Verfahren kann ein Benutzer statistische Eigenschaften einer Datenmenge ermitteln, die zur weiteren Analyse zielführend sind. Das Ergebnis des Verfahrens wird visuell dargestellt. Hierdurch ist es beispielsweise möglich, durch eine Verteilungsanalyse interessante Gebiete in einer raumbezogenen Datenmenge zu identifizieren.

Beschreibung	Es soll ein statistisches Verfahren auf eine ausgewählte Datenmenge angewandt werden und das berechnete Ergebnis visuell dargestellt werden. Hierzu wird zunächst die entsprechende statistische Funktion ausgewählt und anschließend eine Visualisierungsform zur Darstellung der Datenmenge.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Es ist eine kompatible Teilmenge von Daten ausgewählt.
Nachbedingungen	Die statistische Funktion ist auf die gewählte Datenteilmenge angewendet, und die Ergebnisse werden in der ausgewählten Visualisierung dargestellt.
Ausnahmen	–
Erweiterungen	–
Standardablauf	1. Der Akteur wählt ein Verfahren aus. 2. Er verbindet dieses mit einer kompatiblen Teilmenge von Daten. 3. Der Analyst wählt eine Visualisierung zur Darstellung der berechneten Ergebnisse und verbindet diese mit dem ausgewählten statistischen Modul.

[UC.3] Änderung von Funktionsparametern

Funktionen, die auf eine Datenmenge angewendet werden, beinhalten häufig Parameter. Die Wahl dieser Parameter gestaltet sich vor dem Hintergrund einer bisher unbekannten Datenmenge als schwierig. Aus diesem Grund können die Parameter interaktiv angepasst werden, um hieraus durch eine sukzessive Verfeinerung eine Hypothese aufstellen zu können, die im weiteren Analysevorgehen verifiziert wird.

Beschreibung	Der Akteur möchte die Parameter einer Funktion ändern. Nach Änderung der Parameter wird der Algorithmus noch einmal auf die Daten angewendet und die Visualisierung mit den evtl. geänderten Ergebnissen aktualisiert.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Eine Analysefunktion ist ausgewählt. 2. Eine Visualisierung zur Darstellung der Ergebnisse ist mit der Analysefunktion verbunden.
Nachbedingungen	Die Funktion wird mit den geänderten Parametern erneut ausgeführt und die Ergebnisse werden in der Visualisierung dargestellt.
Ausnahmen	–
Erweiterungen	–
Standardablauf	1. Die Parameter des Algorithmus werden vom Analysten an den Analysefall angepasst.

[UC.4] Visualisierung verwenden

Um eine aufgestellte Hypothese zu verifizieren wird eine Visualisierung der Datenmenge verwendet, welches dem Analyseschritt Modellvisualisierung des Analyseprozesses entspricht. Weil die ausgewählte Datenmenge raumbezogene Attribute enthält, wird als Visualisierungsform eine Choroplethenkarte gewählt. Im Analyseszenario wird hierdurch eine Häufung der Krebsfälle in Südniedersachsen ermittelt.

Beschreibung	Durch Zuweisung einer Visualisierung zu einer Datenmenge, oder einer Teilmenge von Daten werden diese in der entsprechenden Visualisierung dargestellt. Der Akteur kann anschließend mit dieser Visualisierung interagieren.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Eine Datenquelle wurde ausgewählt. 2. Die zu verwendende Teilmenge der Gesamtmenge von Daten der Datenquelle wurde dem Analysefall angepasst. Es muss mindestens ein Raumbezognes- und ein Quantitativesattribut gewählt werden. 3. Die durchgeführte Auswahl der Teilmenge der Attribute ist miteinander kompatibel.
Nachbedingungen	Der Analyst erhält eine Darstellung der Daten und kann die von der Visualisierung unterstützen Operationen mit Hilfe von Interaktionen aufrufen.

Ausnahmen	–
Erweiterungen	–
Standardablauf	1. Der Analyst wählt eine Visualisierungsform aus, welche die gewählte Datenmenge darstellen kann. 2. Er verknüpft die Visualisierung mit der ausgewählten Datenmenge.

[UC.5] Verknüpfung mit statistischen Maßzahlen

Bei der Anzeige von Visualisierungen kann es vorkommen, dass Einflüsse, wie zum Beispiel die Einwohneranzahl, die dargestellte Visualisierung stark beeinflussen. Dieses ist auch bei der Betrachtung der Choroplethenkarte der Fall. Aus diesem Grund wird die ausgewählte Datenmenge mit einer statistischen Maßzahl verknüpft, wodurch bestimmte Einflussfaktoren herausberechnet werden können. Es kommt zu einer Verschiebung der Häufung der betrachteten Eigenschaft. Im Szenario wird die Datenmenge mit der Maßzahl zur Bevölkerungsdichte verknüpft, wodurch sich die Ballung nach Nord-West-Niedersachsen verschiebt.

Beschreibung	Durch Verknüpfung einer Datenmenge mit einer statistischen Maßzahl wird die Datenmenge entsprechend angepasst. Außerdem wird die zur Darstellung der Daten ausgewählte Visualisierung aktualisiert.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Eine Datenquelle wurde ausgewählt. 2. Die zu verwendenden Teilmengen der Gesamtmenge von Daten der Datenquelle wurde dem Analysefall angepasst. 3. Die durchgeführte Auswahl der Teilmenge der Attribute ist miteinander kompatibel.
Nachbedingungen	Der Analyst erhält eine Darstellung der Datenmenge mit der verknüpften statistischen Maßzahl.
Ausnahmen	–
Erweiterungen	–
Standardablauf	1. Der Analyst wählt eine statistische Maßzahl aus. 2. Er weist diese der Datenteilmenge zu.

[UC.6] Verfeinerung der Sicht

Mit Hilfe der Choroplethenkarte ist ein Ballungsgebiet ermittelt worden. Im weiteren Analysevorgehen wird dieses Gebiet genauer betrachtet, indem eine Verfeinerung der Sicht durchgeführt wird. Die

Verfeinerung wird in einzelnen Stufen durchgeführt, wodurch dass Ballungszentrum der gewählten Eigenschaft genau ermittelt werden kann.

Beschreibung	Mit Hilfe einer Interaktion kann der Akteur eine Verfeinerung durchführen. Hierdurch werden die Daten der hierarchisch tiefer angeordneten Elemente angezeigt.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Eine Visualisierung wurde mit einer Datenmenge verknüpft. 2. Die ausgewählte Visualisierung unterstützt eine Verfeinerungsinteraktion. 3. Es ist nicht die feinste Ebene der Daten dargestellt. 4. Die Daten haben eine hierarchische Gliederung.
Nachbedingungen	Der Analyst sieht die ausgewählte Ebene der Daten.
Ausnahmen	–
Erweiterungen	–
Standardablauf	1. Das zu verfeinernde Attribut wird ausgewählt. 2. Eine Interaktion zur Verfeinerung wird ausgeführt.

[UC.7] Vergröberung der Sicht

Auf Grundlage einer sukzessiven Verfeinerung wird eine Ballung ermittelt. Da jedoch eine Ballung auf unterschiedlichen Gebieten stattfindet, wird zwischen zwei Verfeinerungsstufen gewechselt. Hierbei werden abwechselnd Vergröberungs- und Verfeinerungsinteraktionen ausgeführt. Durch die Kombination aus Vergröberung und Verfeinerung gelingt es dem Analysten im Szenario das Ballungsgebiet Ostfriesland als besonders interessant zu ermitteln.

Beschreibung	Durch eine ausgeführte Vergröberungsinteraktion wird die gröbere Ebene der hierarchisch gegliederten Datenmenge dargestellt.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Eine Visualisierung wurde mit einer Datenmenge verknüpft. 2. Die ausgewählte Visualisierung unterstützt eine Vergröberungsinteraktion. 3. Es ist nicht die gröbste Ebene dargestellt. 4. Die Daten haben eine hierarchische Gliederung.

Nachbedingungen	Der Analyst sieht die ausgewählte Ebene der Daten.
Ausnahmen	–
Erweiterungen	–
Standardablauf	1. Es wird die Vergröberungsinteraktion ausgeführt.

[UC.8] Datenteilmenge auswählen

Die Betrachtung der Gesamtendatenmenge ist nicht immer zweckdienlich. Aus diesem Grund wird eine Teilmenge der Attributmenge, die von der Datenquelle zur Verfügung gestellt wird, ausgewählt. Diese Teilmenge wird im weiteren Analysevorgehen weiter verwendet. Bezogen auf das Anlyseszenario werden die Krebsarten innnerhalb einer Region durch eine Einschränkung der Attribute dargestellt.

Beschreibung	Der Benutzer möchte die Menge der darzustellenden Attribute einschränken. Hierzu kann der Analyst ein Attributsauswahlwerkzeug verwenden, wodurch nur die ausgewählten Attribute bei der Durchführung von statistischen Funktionen und Darstellung der Daten einfließen.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Eine Datenquelle ist ausgewählt.
Nachbedingungen	Der Akteur hat eine Teilmenge der durch die Datenquelle zur Verfügung stehenden Datenmenge ausgewählt.
Ausnahmen	Die gewählten Attribute sind nicht miteinander kompatibel.
Erweiterungen	–
Standardablauf	1. Der Analyst wählt Schrittweise Attribute aus, die für die Teilmenge bestimmt sind.

[UC.9] Änderung der Visualisierungsform

Bisher sind die gewählten Daten in einer Choroplethenkarte dargestellt. Diese Darstellungsform kann jedoch nur räumliche Attribute darstellen. Weil allerdings eine Kombinationen von Attributen ohne Raumbezug ausgewählt wurde, können diese nicht in der gewählten Visualisierungsform dargestellt werde. Es ist notwendig eine andere Form zu wählen. Eine Punktdarstellung wird für die weitere Betrachtung der Datenmenge ausgewählt.

Beschreibung	Der Anwender ändert für eine Datenteilmenge die Visualisierungsform. Hierdurch werden die Daten in der gewählten Visualisierung dargestellt. Eventuell bestehende Visualisierungen der Daten werden durch diese Operation nicht beeinflusst.

Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Eine Datenteilmenge wurde ausgewählt.
Nachbedingungen	Der Akteur hat für die Teilmenge eine andere Visualisierungsform ausgewählt.
Ausnahmen	–
Erweiterungen	–
Standardablauf	1. Der Analyst wählt aus der Menge zur Verfügung stehender Visualisierungsformen eine aus. 2. Durch Verknüpfung mit der Datenmenge wird diese in der Visualisierung dargestellt.

[UC.10] Selektion von Daten

Innerhalb der Punktdarstellung der Datenmenge werden nacheinander verschiedene Datenpunkte ausgewählt. Hierduch kann sich ein Anwender auf diese Punkte konzentrieren und einen Zusammenhang zwischen den ausgewählten Datensätzen ermitteln.

Beschreibung	Der Anwender selektiert einen Datenpunkt in einer Visualisierung. Dieser wird anschließend hervorgehoben, um die Selektion dieses Punktes für den Analysten zu verdeutlichen.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Eine Visualisierung wurde ausgewählt 2. Es ist eine Datenmenge vorhanden, die dargestellt werden kann.
Nachbedingungen	Der Datenpunkt ist hervorgehoben.
Ausnahmen	–
Erweiterungen	–
Standardablauf	1. Der Analyst wählt einen Datenpunkt aus und selektiert diesen. 2. Der Punkt wird durch eine spezielle Einfärbung selektiert dargestellt.

[UC.11] Visualisierung verknüpfen

Das System stellt zwei Visualisierungen (Choroplethenkarte und Punktdarstellung) dar. Innerhalb beider Visualisierungsformen existiert ein logischer Zusammenhang über das Raumattribut. Auf Grund-

lage dieser Eigenschaft werden beide Visualisierungen miteinander verknüpft. Selektionen innerhalb einer Visualisierung werden an die jeweils verknüpfte Visualisierung, mit teilweise unterschiedlichen dargestellten Attributen, weitergereicht, wodurch der Analyst Beziehungen zwischen diesen erkennen kann. Mit Hilfe dieser Systemeigenschaft ermittelt der Analyst, dass die Krebsart Lungenkrebs im Analyseszenario in der betrachteten Region gehäuft auftrit.

Beschreibung	Der Akteur möchte zwei Visualisierungen miteinander verknüpfen, um Selektionen auf einer der beiden Visualisierungen auch in der anderen anzuzeigen. Hierzu stellt er mit Hilfe einer Operation eine Verknüpfung zwischen beiden Visualisierungen her.
Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Mindestens zwei Visualisierungen sind dargestellt. 2. Die für die Verknüpfung ausgewählten Visualisierungen sind miteinander kompatibel (teilweise übereinstimmende Attribute) und stammen von der gleichen Datenquelle.
Nachbedingungen	Der Akteur hat zwei Visualisierungen miteinander verknüpft. Selektionen einer Visualisierung werden an die verknüpfte Visualisierung weitergereicht.
Ausnahmen	–
Erweiterungen	Auf Grundlage der besonderen Eigenschaft von Zeit- und Ortsattributen können diese auch miteinander verknüpft werden, wenn die Datenquellen der Visualisierungen unterschiedlich sind.
Standardablauf	1. Der Analyst wählt das Verknüpfungswerkzeug aus. 2. Der Analyst spezifiziert die erste Visualisierung zur Verknüpfung. 3. Er bestimmt die zweite Visualisierung zur Verknüpfung.

[UC.12] Auswahl weiterer Datenquellen

Durch die Einbeziehung weiterer Datenquellen in das Analysevorgehen wird die Betrachtung weiterer Eigenschaften ermöglicht, die nicht von einer einzelnen Datenquelle bereit gestellt werden. Es kann somit eine gegenseitige Ergänzung stattfinden. Diese Möglichkeit wird genutzt, um das ermittelte Ergebnis einer vorherigen Analyse zu bestätigen und auf Grundlage einer anderen Datenquelle mit anderen Attributen zu begründen. Im Analyseszenario wird eine weitere Datenquelle mit regionalen Einflussfaktoren genutzt, um den Grund für die Häufung der Krebsart zu bestimmen.

Beschreibung	Durch Auswahl einer anderen Datenquelle steht diese im System zur Verfügung. Falls bereits eine vorherige Datenquelle für eine Analyse verwendet wird, hat diese auch weiterhin für die verbundenen Visualisierungen und Funktionen Gültigkeit.

Akteure	Analyst
Auslöser	–
Vorbedingungen	1. Die weitere Datenquelle ist im System eingebunden. 2. Die Verbindung zur Quelle kann hergestellt werden.
Nachbedingungen	Der Analyst kann eine weitere Datenquelle zur Analyse verwenden. Es stehen die Möglichkeiten wie bei der vorherigen Quelle zur Verfügung.
Ausnahmen	–
Erweiterungen	–
Standardablauf	1. Durch Auswahl der Quelle steht diese im System zur Verfügung.

6.2 Funktionale Anforderungen

Das System benötigt eine Vielzahl von Funktionen, die der Analyse von Daten dienen. Aus diesen Funktionen ergeben sich Anforderungen, die im Folgenden spezifiziert werden. Hierbei wird in eckigen Klammern am Ende der Anforderung auf die jeweiligen Anwendungsfälle des vorherigen Kapitels verwiesen, um einen Zusammenhang zwischen Anwendungsfall und Anforderung zu erzielen. Im anschließenden Abschnitt wird ein Konzept vorgestellt, welches diese Anforderungen erfüllt.

[FA.1] Anbindung multidimensionaler Datenmengen: Aus den Anwendungsfällen geht hervor, dass multidimensionale Daten analysiert werden sollen. Das System soll die Möglichkeit bieten, diese Daten darzustellen und auf die hierarchische Struktur der Daten Zugriff zu erhalten, um Teilmengen für einen spezifischen Analysefall auswählen zu können.[UC.1]

[FA.2] Analyse zeit- und ortsbezogener Daten: Bei der Analyse werden epidemiologische Daten untersucht. Diese weisen zeit- und ortsbezogene Attribute auf. Die besonderen Eigenschaften dieser Attributstypen sollen im System abgebildet werden können.[UC.1]

[FA.3] Möglichkeit der Integration von Funktionen zur Auswertung einer Datenmenge: In den Anwendungsfällen werden Funktionen verwendet, um einen Überblick über die Daten zu erhalten und erste Erkenntnisse über diese zu gewinnen. Das Analysesystem soll die Möglichkeit bieten, Funktionen, wie beispielsweise statistische Methoden oder Data-Mining-Algorithmen, einzubinden und diese auf eine gewählte Datenmenge anzuwenden.[UC.2]

[FA.4] Modellvisualisierung: Auf Grundlage von durchgeführten Funktionen können unterschiedliche Modellinstanzen entstehen. Die Darstellung des berechneten Modells ist für die weitere Analyse wichtig. Das System soll die Visualisierung von berechneten Modellen ermöglichen.[UC.2]

[FA.5] Parameteränderungen beeinflussen Visualisierungen: Häufig haben Data-Mining-Algorithmen und statistische Verfahren Parameter, welche die Berechnungen beeinflussen. Sofern

diese Parameter geändert werden, soll eine Visualisierung der Datenmenge entsprechend angepasst werden.[UC.3]

[FA.6] Visualisierungsform: Choroplethenkarte: Daten mit raumbezogenen Attributen können in thematischen Karten dargestellt werden. Das System soll für diesen Analysezweck eine Choroplethenkarte bereit stellen.[UC.4]

[FA.7] Freie Auswahl der Analysereihenfolge: Zentraler Bestandteil des Visual Analytics Prozesses ist die Möglichkeit frei zwischen Visualisierung und Modell zu wechseln. Dies bedeutet, dass die Analyse nicht einem festen Ablauf aus Modellbildung und Modellvisualisierung unterliegt. Ein Analyst soll die Möglichkeit haben, diesen Analyseablauf frei zu wählen und frei zwischen den Prozesszuständen zu wechseln. Dies kann unter anderem auch bedeuten, dass mehrere Visualisierungen oder Data-Mining-Algorithmen hintereinander ausgeführt werden, ohne ein vorheriges Wechseln in den anderen Prozesszustand.[UC.2, UC.4]

[FA.8] Verknüpfung von Daten mit statistischen Maßzahlen: Eine statistische Maßzahl dient im Szenario zur Herausberechnung der unterschiedlichen Bevölkerungsdichte bei der Betrachtung der Datenmenge. Das System soll die Möglichkeit bieten, vorher definierte Maßzahlen mit der Datenmenge dynamisch zu verknüpfen.[UC.5]

[FA.9] Verfeinerung und Vergröberung der Sicht: Die verwendete Datenmenge weist hierarchische Strukturen auf. Während der Analyse ist die Betrachtung von unterschiedlichen Hierarchiestufen in der Datenmenge erforderlich. Das zu entwickelnde Analysesystem soll den Wechsel der dargestellten Hierarchiestufe dynamisch ermöglichen.[UC.6, UC.7]

[FA.10] Direkte Interaktion mit einer Visualisierung: Ein weiterer Schritt bei der Analyse ist eine Interaktion mit einer Visualisierung, um beispielsweise einen semantischen Zoom durchzuführen. Ein Analyst soll durch eine Interaktion die Visualisierung direkt beeinflussen können.[UC.6]

[FA.11] Attributsauswahl: In multidimensionalen Mengen sind in der Regel weitaus mehr Attribute vorhanden, als gleichzeitig vom System dargestellt werden können. Ein Anwender des Systems soll in die Lage versetzt werden, die insgesamt zur Verfügung stehenden Attribute auf eine bestimmte wählbare Menge einzugrenzen. Alle mit dieser Quelle verbundenen Visualisierungen verwenden anschließend nur diese Teilmenge zur Anzeige der Daten.[UC.8]

[FA.12] Visualisierungsform: Punktdarstellung: Nur die Verwendung einer spezifischen Visualisierungsform führt nicht unbedingt bei allen Attributtypen zur optimalen Anzeige. Außerdem können nicht alle Teildatenmengen in einer Choroplethenkarte dargestellt werden. Aus diesem Grund soll als weitere Visualisierungsform eine Punktdarstellung in das System eingebunden werden.[UC.9]

[FA.13] Gleichzeitige Betrachtung einer Datenmenge mit unterschiedlichen Visualisierungsformen: Weil nicht jede Visualisierungsform für alle Attribute gleichartig geeignet ist, soll das System die Möglichkeit der Anzeige einer spezifischen Datenmenge in unterschiedlichen Visualisierungsformen bieten. Somit kann die Geeignetste dynamisch anhand der Darstellung der Datenmenge bestimmt werden.[UC.9]

[FA.14] Erweiterung um Visualisierungsformen: Im Bereich von Visual-Analytics werden unterschiedliche Visualisierungensformen zur Darstellung von Datenmengen verwendet, die zum

Teil verschiedene Interaktionen ermöglichen. In der Wissenschaft ist der Trend zu beobachten, dass immer weitere und speziellere Visualisierungen erforscht werden. Diese entstehen unter anderem auch nach der Entwicklungsphase des Systemes. Aus diesem Grund soll das System eine Möglichkeit bieten, diese neuartigen Visualisierungen einzubinden und für den Analysten zugänglich zu machen.[UC.9]

[FA.15] Selektion von Daten in Visualisierungen: Durch eine Selektion von Daten in einer Visualisierung wird der Analyst in die Lage versetzt, einen Datensatz zum Beispiel bei einer Animation nachzuverfolgen, um somit den Trend eines bestimmten Datenpunktes zu ermitteln. Das System soll diese Möglichkeit bieten.[UC.10]

[FA.16] Selektionsverknüpfung zwischen Visualisierungen: Sofern gleichzeitig mehrere Visualisierungen dargestellt werden, soll die Möglichkeit bestehen, eine Verknüpfung zwischen diesen Visualisierungen – gegebenenfalls auch unterschiedlichen Visualisierungsformen – zu erstellen. Durch diese Verknüpfungsart sollen Selektionen, die in einer Visualisierung durchgeführt werden, auf verknüpfte Visualisierungen repliziert werden. Die Verknüpfung soll auch bei räumlichen bzw. zeitlich abhängigen Attributen aus unterschiedlichen Datenquellen ermöglicht werden.[UC.11]

[FA.17] Einbindung unterschiedlicher Datenquellen: Durch die Einbindung von unterschiedlichen Datenquellen mit evtl. auch unterschiedlichen Datenhaltungen können quellübergreifende Korrelationen zwischen Datenquellen erkannt werden. Aus diesem Grund soll das System die Möglichkeit bieten, unterschiedliche Datenquellen und Datenformen zur Analyse zur Verfügung zu stellen.[UC.12]

6.3 Konzept

Das zu entwickelnde System erfordert eine dynamische Anpassung des Analyseverlaufes [FA.7] mit der Möglichkeit gleichzeitig weitere Datenquellen [FA.17] einzubinden und das System um Visualisierungsformen zu ergänzen [FA.14]. Zusammengefasst führen diese Anforderungen zu dem Ergebnis, dass das System flexibel und dynamisch Komponenten binden muss. Diese Eigenschaften weisen die im Grundlagenteil vorgestellten Visualisierungsprozessmodelle auf. Ausgehend von den hier erlangten Erkenntnissen, dass sowohl ein Visualisierungsprozess an sich notwendig ist, als auch eine Beschreibung der Daten in Form eines Modells, umfasst das Konzept zwei Modellebenen. Eine Ebene umfasst die Beschreibung der analysierten Daten, die Zweite stellt Modellelemente für die Modellierung des Visualisierungsprozesses bereit. Innerhalb der Komponenten des Visualisierungsprozesses wird das Modell zur Beschreibung der Daten verwendet. Es wird deshalb zunächst auf das verwendete Konzept zur Visualisierung eingegangen.

Visualisierung kann als ein Prozess gesehen werden, der Rohdaten in Sichten transformiert. Im Grundlagenabschnitt sind zwei grundsätzlich unterschiedliche Modelle zur Modellierung dieses Visualisierungsprozesses vorgestellt worden. Zum einen das Data-Flow-Modell und zum anderen das Data-State-Modell. Chi beweist in [Chi02], dass beide Modelle die gleiche Ausdrucksstärke haben und ein Modell in das jeweils andere Modell umgeformt werden kann. Die Wahl welches dieser beiden grundsätzlichen Modellarten genutzt wird, ist unabhängig von der Ausdrucksstärke des Modells, da beide äquivalent sind. In der Modellierung mit dem vorgestellten Modellierungstool *VMTS* (vgl. Kapitel 2.1) gestaltet es sich einfacher das Data-Flow-Modell umzusetzen, weil Kanten als Kommunikation

zwischen Knoten interpretiert werden, was dem Data-Flow-Modell entspricht. Aus diesem Grund wird eine angepasste Form des Data-Flow-Modells verwendet, die im Folgenden näher betrachtet wird.

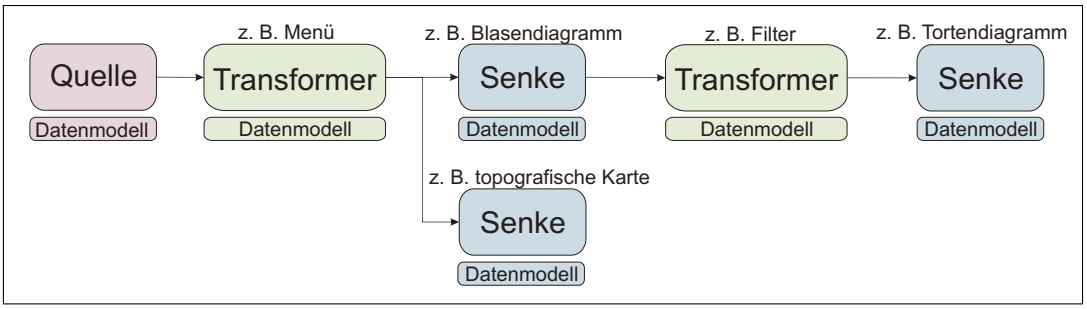


Abbildung 6.1: *Visualisierungsmodell bestehend aus Quelle, Transformer und Senke.*

Das Data-Flow-Modell wird in vielen wissenschaftlichen Anwendungen verwendet. Das Modell besteht im Wesentlichen aus drei Komponenten. Knoten, die Prozesszustände repräsentieren und durch Operationen das Datenmodell verändern können. Kanten zwischen diesen Knoten bilden einen gerichteten Datenfluss (dargestellt durch Pfeile). Zudem werden typgebundene Ein- und Ausgabe-Ports verwendet, um Assoziationen zwischen zwei Knoten nur mit einem bestimmten Datentyp zu ermöglichen. Um die unterschiedlichen Modellierungsanforderungen an Datenquellen und Visualisierungen zu ermöglichen, werden die Knotentypen *Quelle*, *Transformer* und *Senke* verwendet, welche in einer beispielhafen Zusammenstellung in der Visualisierungspipeline in der Abbildung 6.1 dargestellt sind:

Quelle: Diese Modellkomponente stellt im Visualisierungsprozessmodell eine Datenquelle bereit. Um die Anforderung der Anbindung von multidimensionalen Datenmengen [FA.1] zu erfüllen, können Quellen multidimensionale Daten und die dazugehörigen Metadaten zur weiteren Analyse bereit stellen. Diese Modellkomponente weist nur einen Ausgabe-Port auf.

Transformer: Transformer, zum Beispiel Filter, Selektionsmenüs oder Data-Mining-Module, wie sie in den Anforderungen [FA.3] und [FA.11] gefordert werden, ermöglichen die Erstellung von funktional abhängigen Datenmodellen. Diese können mit Hilfe von Senken dargestellt werden. Änderungen der evtl. enthaltenen Parameter eines Transformelementes [FA.5] werden durch die Visualisierungspipline an nachfolgende Visualisierungsprozessmodellelemente weitergereicht. Dieses Knotenelement kann Daten von anderen Quellen oder anderen Transformer-Knoten aufnehmen und gibt diese an andere Transformer oder an Senken weiter.

Senke: Senken stellen Modelldaten graphisch dar. Es spielt hierbei keine Rolle, von welchem Prozesselement (Quelle, Transformer oder Senke) das Datenmodell weitergeleitet wurde. Die eigentliche Darstellungsform ist nicht im Modell festgehalten. Es können deshalb unterschiedliche Visualisierungsformen, wie beispielsweise ein Punktdiagramm [FA.12] oder eine Choroplethenkarte [FA.6] zur Darstellung verwendet werden.

Eine weitere Anforderung an das System wird durch die Möglichkeit der gleichzeitigen Betrachtung einer Datenmenge in unterschiedlichen Senken gegeben [FA.13]. Aus diesem Grund wird das im Grundlagenabschnitt vorgestellte Data-Flow-Modell um die Möglichkeit erweitert, dass Knotenelemente mehrere Nachfolger besitzen können. Außerdem muss das Modell dahingehend ergänzt werden, dass auch der Knotentyp Senke Ausgabeports hat, wodurch die freie Wahl der Komponentenreihenfolge, und damit der Analysereihenfolge, unterstützt wird. Das verwendete Modell ist somit kein linearer Verlauf von der Quelle zur Senke, wie das vorgestellte Data-Flow-Modell, sondern ein azyklischer Graph,

der dynamisch während der Laufzeit gebildet wird. Die Folge und Anzahl der Modellkomponenten ist nicht festgeschrieben.

Angelehnt an die formale Definition des Data-Flow-Modells in [JK03] wird das Visualisierungsprozessmodell folgendermaßen definiert. Das Modell ist ein Visualisierungstransformationsnetzwerk $G = (M, D)$ bestehend aus einem gerichteten Graphen von Modellkomponenten M und den damit verbundenen Datenflusskanten D. Jede Komponente $M_i = (L_i, P_i, O_i, V_i, f_i) \in M$ besteht aus einem Satz von Eingaben L_i, Parametern P_i, Ausgaben O_i, verknüpften Komponenten $V_i = \{v|v \in M\}$ und einer Transformationsfunktion $f_i(P_i) : L_i \mapsto O_i$, welche die Eingaben unter Beeinflussung der Parameter auf die Ausgaben transformiert. Es gilt dabei, dass jede Komponente maximal eine Eingabe aufweist $|L_i| < 2$. Eine Eingabe ist die im gerichteten Graphen vorher definierte Ausgabe. Innerhalb jeder Modellkomponente kann ein Satz von Parametern P_i die Ausgabe O_i beeinflussen. Ausgaben O_i sind die Ergebnisse der Modellkomponente. Modellkomponenten mit keiner Eingabe und mindestens einer Ausgabe sind Quellen, während eine Komponente mit mindestens einer Eingabe aber keiner Ausgabe das Ergebnis der Transformation (meistens eine Senke) ist. Die aktuelle Visualisierungstransformation $t : M_{start} \mapsto M_{ergebnis}$ erfolgt mit Hilfe der Berechnung von der Startkomponente $M_{start} = \{m = (O, P, \emptyset, V) \in M : |O| > 0\}$ im gerichteten Graphen zur Zielkomponente $M_{ergebnis} = \{m = (\emptyset, P, I, V) \in M : |I| > 0\}$. Wenn nur eine Ergebniskomponente existiert, können die Funktionen f_i, die in jeder Modellkomponente M_i berechnet werden, zu einer einzelnen Funktion zusammengefasst werden, welches die Gesamttransformation repräsentiert. Wenn mehrere Ausgaben vorhanden sind (zum Beispiel bei der Anzeige von Quelldaten in unterschiedlichen Ansichten) führt jeder Pfad des Graphen zu einem Ergebnis und definiert damit eine Unterfunktion der übergreifenden Transformationsfunktion. Das Gesamtsystemmodell S besteht aus einer Menge von Modellen $S = (G)$.

Es entstehen damit Visualisierungspfade von einer Quelle über evtl. verschiedene Transformationselemente und Senken zu einer Zielvisualisierung. Die Verknüpfung der Modellelemente hat den Vorteil, dass Änderungen an einer Modellkomponente in der Visualisierungskette weitergereicht werden [FA.5]. Außerdem lässt das Modell die Möglichkeit offen, mehrere Visualisierungspfade an jedem Punkt im gerichteten Graphen zu eröffnen. Damit ist es möglich, dass eine betrachtete Datenmenge in unterschiedlichen Pfaden weiter verarbeitet oder betrachtet wird, wie dies in der Anforderung [FA.13] gefordert wird.

6.4 Modellbildung

Das im vorigen Kapitel beschriebene Konzept führt zwei Modelle ein. Zum einen das Visualisierungsprozessmodell zur Abbildung von Visualisierungspfaden und zum anderen Datenmodell, dass die Daten und Metadaten zu einer Modellkomponente kapselt. In diesem Kapitel werden diese beiden Modelle genauer beschrieben. Zunächst wird das Systemmodell als Realisierung des Visualisierungsprozessmodells beschrieben und im Anschluss das Datenmodell. Die Notation der dargestellten Modelle entspricht dabei der im Kapitel 2.1 vorgestellten.

6.4.1 Das Systemmodell als Realisierung des Prozessmodells

Der verwendete Visualisierungsprozess wird durch das in Abbildung 6.2 dargestellte Modell, im Folgenden Systemmodell genannt, realisiert. Wie im Prozess beschrieben (vgl. Abbildung 6.1) besteht

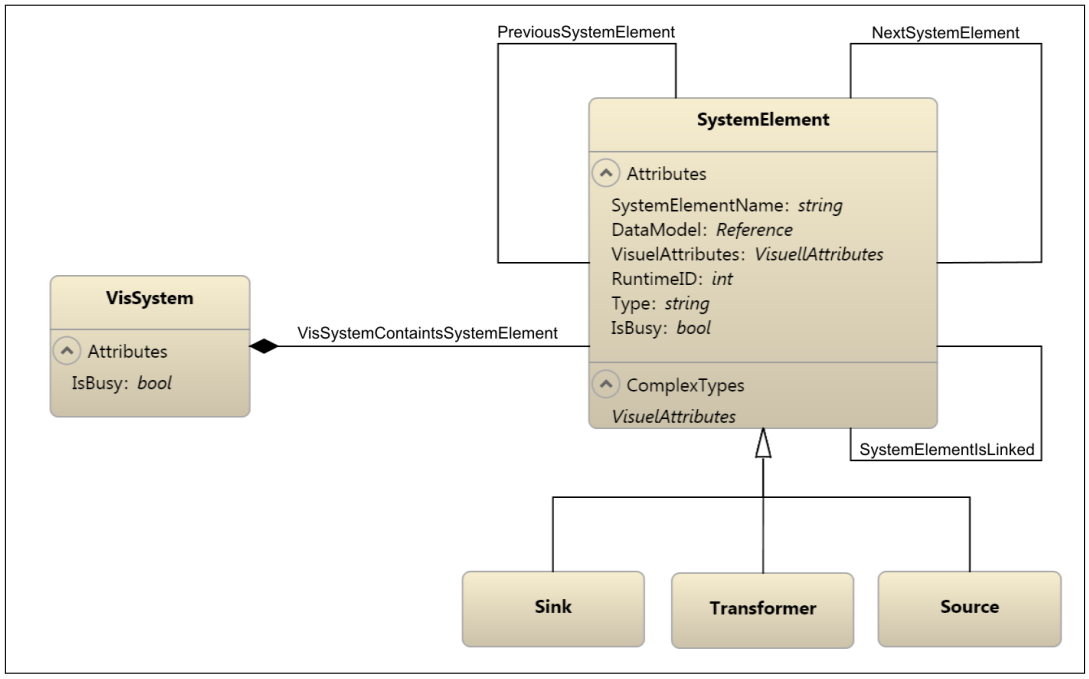


Abbildung 6.2: *Das Systemmodell als Realisierung des Prozessmodells*

er aus einer Menge von Prozesskomponenten, die miteinander verbunden sind. Diese Eigenschaft des Prozesses wird mit Hilfe des Modellelementes `SystemElement` abgebildet. Die Verknüpfung der Elemente wird durch eine Verkettung realisiert. Damit Daten dynamisch nachgeladen werden können, ist es notwendig, dass jedes `SystemElement` nicht nur einen Verweis auf die nachfolgenden Elemente hat, sondern auch auf das Vorgängerelement. Aus diesem Grund wird als Verkettungstyp eine doppeltverkette Liste, die mit Hilfe der Beziehungen `PreviousSystemElement` und `NextSystemElement` realisiert wird, verwendet. Durch diese Elemente kann ein gerichteter Graph bestehend aus `SystemElementen` gebildet werden. Die im Prozess getroffene Unterscheidung der Modellelemente von Senken, Transformer und Quellen, wird durch eine Vererbungshierarchie der gleichnamigen Systemmodellelemente abgebildet. Mit Hilfe von zusätzlich eingefügten Regeln, wird die Eigenschaft des Prozesses sichergestellt, dass Quellen keinen und Transformer sowie Senken nur ein Vorgängerelement haben können.

Das zu entwickelnde System soll die Möglichkeit bieten, gleichzeitig unterschiedliche Analysepfade (von der Quelle bis zu den Zielvisualisierungen) zu ermöglichen [FA.17]. Das bisher beschriebene Systemmodell kann diese Eigenschaft durch die Erstellung von mehreren Pfaden abbilden. Allerdings ist für die Nutzung in einem System ein zentraler Zugriffspunkt auf alle Visualisierungspfade oder Systemelemente notwendig, da ansonsten nicht auf die einzelnen Elemente, bzw. Pfade zugegriffen werden kann. Dieser Zugriffspunkt wird durch das Modellelement `VisSystem` ermöglicht. Das `VisSystem` besteht aus einer Menge von `SystemElementen` und erweitert das bestehende Modell um die Möglichkeit Visualisierungspfade mit unterschiedlichen Quellen als Ausgangspunkt zu erstellen. Aufbauend auf der Eigenschaft, dass eine Menge von unterschiedlichen Visualisierungspfaden existieren, ist die Anforderung der Selektionsverknüpfung [FA.16]. Diese bedingt eine Verbindung zwischen Visualisierungspfaden. Diese Anforderung wird mit Hilfe der Modellbeziehung `SystemElementIs-`

`Linked` realisiert. Die Beziehung ermöglicht die Verknüpfung von `SystemElementen` innerhalb eines Pfades oder auch über einen Pfad hinaus.

Ein Visualisierungsprozess besteht nicht nur aus der Bildung von gerichteten Graphen. Ein wichtiger Parameter ist das zu einer Modellkomponente zugeordnete Datenmodell. Das Systemmodell muss diese Eigenschaft abbilden. Aus diesem Grund verfügt jedes `SystemElement` über das Attribut `DataModel`. Diese Referenz verweist auf das entsprechende Datenmodell. Der Typ des Datenmodells ist einem Entwickler freigestellt. Als weitere Attribute sind `RunTimeID` und `SystemElementName` hinzugefügt, wodurch eine Identifikation für einen Analysten (textuelle Beschreibung des Elementes) und innerhalb des Systems (fortlaufende ganze Zahl) möglich wird. Die visuellen Eigenschaften des Elementes, wie Größe und Position, sind im komplexen Attribut `VisuelAttributes` gekapselt.

6.4.2 Datenmodell

Die Visualisierungsprozesskomponenten Quelle, Senke und Transformer agieren mit Daten, die transformiert, dargestellt oder zur Verfügung gestellt werden. Die Daten können multidimensionale Eigenschaften aufweisen [FA.1] und von unterschiedlichen Datenquellen stammen [FA.17]. Es ist notwendig, dass diese Daten einer allgemeinen zugreifbaren Beschreibung unterliegen, um über eine Prozesskomponente hinaus die Datenmenge zu verwenden. Aus diesem Grund wird das Datenmodell entwickelt, welches eine Beschreibung der aktuellen Datenmenge und der Eigenschaften dieser Menge, wie beispielsweise Beziehungen und Ordnungsrelationen, ermöglicht. Das Datenmodell und die hierauf abbildbaren Interaktionen werden im Folgenden dargestellt.

Die Modellierung des Datenmodells wird durch die Anforderung der Erweiterungsmöglichkeit um weitere Visualisierungsformen geprägt [FA.14]. Aus dieser Anforderung geht hervor, dass nicht ein spezielles Modell für ein Punktdiagramm und eine Choroplethenkarte den Anforderungen genügt, sondern es ist ein von der Visualisierungsform unabhängiges Datenmodell erforderlich. Im Kapitel 5 ist ein Klassifikationssystem erstellt worden, welches mögliche Eigenschaften von Visualisierungen umfasst. Auf Grundlage der hier erlangten Erkenntnisse wird das Datenmodell, das in der Abbildung 6.3 dargestellt ist, entwickelt.

Die erste Erkenntnis aus dem Klassifikationssystem ist, dass unterschiedliche Arten von Ordnungsrelationen in Visualisierungen existieren und Visualisierungstechniken in der Regel Achsen und Werte zur Darstellung verwenden. Die unterschiedlichen Ordnungsrelationen spiegeln sich im Achsentyp wieder. Für jede identifizierte Ordnungsrelation im Klassifikationssystem (ordinal, quantitativ, nominal und kategorisch) wird ein Achsentyp eingefügt. Eine quantitative Achse kann weiterhin in eine Betragsachse, Koordinatenachse, Zeitachse oder Raumachse untergliedert werden, weshalb entsprechend weitere Elemente in das Modell eingefügt werden. Die Art der Ordnungsrelation wird somit durch den Achsentyp abgebildet. Jeder Achse können Werte zugeordnet werden, weshalb für jeden Achsentyp ein spezieller Wertetyp in das Datenmodell eingefügt wird. Diese Zuordnung von Werten zu einer Achse wird durch die Relation `AxisValueOnAxis` des Datenmodells ermöglicht. In Visualisierungen werden nicht nur Werte angezeigt, sondern auch Wertepaare, die einen semantischen Zusammenhang aufweisen. Diese Eigenschaft wird mit Hilfe des Modellelements `ValuePair` realisiert. Ein Wertepaar, bestehend aus einer Menge von Werten aus unterschiedlichen Achsen, stellt im Vergleich mit einem *OLAP*-Cube eine Zelle da. Ein Datenmodell kann aus mehreren Zellen bestehen, wobei ein Wert einer Achse in mehreren Zellen verwendet werden kann. Eine Besonderheit einer multidimensionalen Datenmenge ist die hierarchische Struktur, in der die Daten vorliegen. Die Eigenschaft soll in dem

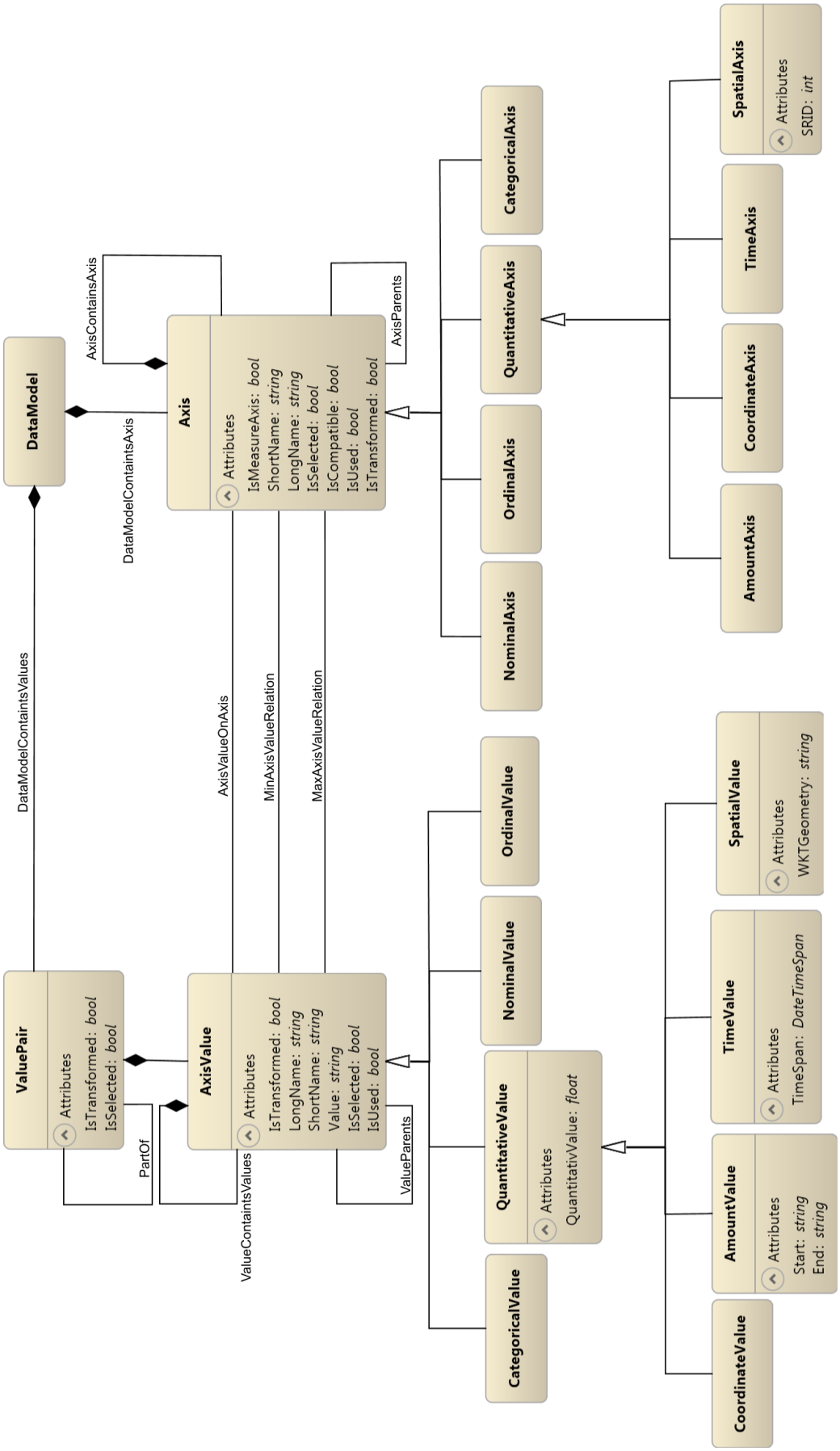


Abbildung 6.3: *Das Datemodell*

Datenmodell abgebildet werden können [FA.1]. Im Klassifizierungssystem wird diese als hierarchische Beziehung identifiziert und im Datenmodell durch die Vater-Kind-Beziehung `AxisContaintsAxis` und Kind-Vater-Beziehung `AxisParents` realisiert. Für hierarchische Beziehungen innerhalb von `AxisValues` werden die Beziehungen `ValueContaintsValues` und `ValueParents` analog eingefügt. Aus dem Klassifizierungssystem gehen des Weiteren Ziele (Assoziationsfindung, Klassifizierung und Clusterung), die durch die Analyse mit Hilfe von Visualisierungen erreicht werden sollen, hervor. Diese Ziele erfordern, dass neben der hierarchischen Beziehung zwischen Daten die Eigenschaft des Zusammenhangs zwischen unterschiedlichen Wertepaaren ausgedrückt werden kann. In [Men00] ist für diesen Zusammenhang die „Is a part of"- Beziehung identifiziert worden. In dem entwickelten Datenmodell wird angelehnt an diese Modellierungseigenschaft die Relation `PartOf` eingefügt, wodurch ein Zusammenhang zwischen unterschiedlichen Wertepaaren ausgedrückt werden kann. Dieser Zusammenhang kann beispielsweise durch Klassifizierungs- oder Clusterungsfunktionen ermittelt werden. Bei der Anwendung von Funktionen und der Erstellung von Visualisierungen besteht häufig die Notwendigkeit die Extremwerte eines Datensatzes zu ermitteln. Beispielsweise im Falle einer Visualisierung können die Extremwerte für die Erstellung der Skala verwendet werden. Damit die Extremwerte nicht bei jeder Verwendung neu bestimmt werden müssen, ist es sinnvoll diese Eigenschaft in das Datenmodell einzufügen. Aus diesem Grund werden die Beziehungen `MinAxisValueRelation` und `MaxAxisValueRelation` in das Modell eingefügt, um somit ein performantes System zu ermöglichen. Außerdem kann aus Sicht des Analysten eine Unterscheidung zwischen den *OLAP*-Typen Dimension und Kennzahl wichtig sein. Dieses wird durch das boolesche Attribut `IsMeasureAxis` einer Achse im Datenmodell abstrahiert.

Auf Grundlage der Einbindung von unterschiedlichen Visualisierungsformen [FA.14], [FA.12], [FA.6] können verschiedene Datenmodelle, die einer Visualisierung zugeordnet sind, begründet sein. Damit alle Visualisierungskomponenten auf die Daten zugreifen können, ist evtl. eine Transformation zwischen verschiedenen Datenmodellen notwendig. Damit die Entwicklung von Transformationen erleichtert wird, ist das Attribut `IsTransformed` jedem Element in dem Datenmodell hinzugefügt worden. Ein Entwickler einer Transformation kann diese Eigenschaft nutzen, um zu prüfen, welche Elemente noch in das entsprechende Datenmodell transformiert werden müssen. Durch diese Eigenschaft wird eine Transformation zwischen unterschiedlichen Datenmodellen erleichtert. Die in den Datenmodellen enthaltene Datenmenge kann rasant ansteigen, wenn alle Daten aller Hierarchien geladen werden. Dieses würde das System unnötig belasten, weil in vielen Fällen nicht jede Aggregationsstufe betrachtet wird. Es besteht also das Bedürfnis, dass Daten je nach Verwendungsfall dynamisch nachgeladen werden können. Diese Eigenschaft wird durch das Attribut `IsUsed` einer Achse bzw. eines Achsenwertes modelliert. Wenn eine Achse in einer Visualisierung verwendet wird (zum Beispiel durch Auswahl in einem Menü) wird dieses Attribut gesetzt. Bei der Weiterreichung des Datenmodells in der Visualisierungskette wird geprüft, ob Achsen verwendet werden, die noch nachgeladen werden müssen. Für den Fall, dass Werte geladen werden müssen, können diese dynamisch von der Quelle nachgeladen werden und in das Datenmodell eingefügt werden.

Die evtl. nachgeladene Datenmenge kann Zeit- und Ortsdaten enthalten. Die besondere Bedeutung dieser Daten wird bei der Betrachtung des Klassifikationssystems (vgl. Kapitel 5.6) deutlich. Es besteht die Anforderung diese Eigenschaften auch im System abzubilden [FA.2]. Zeitdaten haben eine eigene semantische Struktur. Die hierarchische Struktur und Granularität der Zeit ist anders als bei den meisten anderen quantitativen Achsen. Zeit wird aus diesem Grund nicht als ein Zeitpunkt, sondern immer als Zeitspanne aufgefasst. Wird beispielsweise in einem Datenwert ein Zeitpunkt auf einer Minute genau angegeben, kann diese Minute als eine Zeitspanne von Sekunden interpretiert werden.

Diese Idee führt zu der Modellierung, dass Zeit immer einen Start und Endpunkt besitzt, aus dem die Zeitspanne bestimmt wird. Im Datenmodell wird hierzu der komplexe Attributtyp `TimeSpan` im Modellelement `TimeValue` eingefügt. Der Attributtyp kapselt einen Startzeitpunkt und Endzeitpunkt. Räumliche Daten werden durch das Modellelement `SpatialValue` abgebildet. Dieser Datentyp hat die Eigenschaft, dass Objekte oder Phänomene mit einer spezifizierten Position und evtl. einer Ausdehnung angeben ist. Es ist allerdings nicht zwingend erforderlich, dass Objekte eine Ausdehnung haben. Räumliche Daten können sowohl keine, eine 1-D (beispielsweise Grenzen) oder eine 2-D (beispielsweise Länder) Ausdehnung aufweisen. Mit der *Well-known Text (WKT)*-Spezifikation ist für geographische Objekte eine Beschreibungssprache entwickelt worden, welche diese Eigenschaften umfasst. Die *WKT*-Spezifikation [Her06] umfasst eine Beschreibung für Punkte, Linien und Flächen in textueller Weise. Deshalb reicht die Einfügung des Attributes `WKTGeometry` zur Modellierung von räumlichen Daten. Allerdings existieren verschiedene räumliche Referenzsysteme, auf die sich die Koordinaten des Geometrieobjektes beziehen. Um diese unterschiedlichen Systeme trotzdem verarbeiten zu können, ist es notwendig eine Beschreibung des verwendeten Referenzsystems einzufügen. Das Oil and Gas Producers Surveying and Positioning Committee hat zu diesem Zweck *Spatial Reference System Identifier (SRID)*s entwickelt, die zur eindeutigen Identifizierung der für geodätische Daten angewendeten Referenzsysteme verwendet wird [OGP09]. Diese Eigenschaft wird im Datenmodell durch das hinzufügen der *SRID* zur `SpatialAxis` abgebildet.

6.4.3 Interaktionsabbildung auf das Datenmodell

Visualisierungsformen unterstützen eine unterschiedliche Menge an Interaktionen. Beispielsweise ermöglichen einige Formen eine Änderung der Hierarchieebene, während andere eine Auswahl der Daten oder einen optischen Zoom, um einen spezifizierten Darstellungsbereich für einen Anwender vergrößert darzustellen, ermöglichen. Anhand dieser kurzen Auflistung von Operationen wird bereits deutlich, dass einige Interaktionen die Datenmenge (Änderung der Hierarchieebene) beeinflussen, während andere die optische Repräsentation (optischer Zoom) verändern. In der Literatur wird deshalb eine Unterscheidung in Sicht- und Wertoperatoren vorgenommen (vgl. Kapitel 3.2). Auf Grundlage der Anforderung, dass Interaktionen mit einer Visualisierung nicht durch das zu entwickelnde System eingeschränkt werden sollen [FA.10], ist eine Betrachtung der Wertoperationen notwendig. Nachfolgend wird diese Betrachtung durchgeführt und beschrieben, wie Wertoperationen auf das Datenmodell abgebildet werden können.

Zunächst ist festzustellen, welche Interaktionen auf Visualisierungen möglich sind, um zu ermitteln, welche Ergänzungen an dem Datenmodell durchgeführt werden müssen. Im Klassifikationssystem für Visualisierungen (vgl. Kapitel 5.6) sind die folgenden neun Interaktionsgruppen identifiziert worden, die auf Operationen abgebildet werden können:

Verzerrung: Eine Verzerrung der Daten ändert die Ansicht der Daten dahingehend, dass eine ausgewählte Datenmenge optisch besonders hervorgehoben wird. Diese Interaktion kann mit Hilfe einer Sichtoperation realisiert werden, da die Datenmenge nicht verändert wird.

Filtern: Durch eine Filterung wird eine Teilmenge der Daten gebildet, indem nur spezifizierte Daten betrachtet werden. Eine Filterinteraktion wird auf eine Wertoperation abgebildet. Dies ist in der Eigenschaft begründet, dass die Datenmenge verändert wird.

Projektion: Im Vergleich mit den vorgestellten *OLAP*-Operationen (vgl. Kapitel 2.2.1) ist eine Projektion eine Slice-Operation. *OLAP*-Operationen sind grundsätzlich Wertoperationen.

Linking und Brushing: Die Möglichkeit der Verknüpfung von Visualisierungen wird in der Anforderung [FA.16] gefordert und kann durch das beschriebene Systemmodell abgebildet werden. Des Weiteren erfordert eine Verknüpfung zwischen verschiedenen Visualisierungen eine Änderung des Modells. Aus diesem Grund wird eine Einstufung als Wertoperation durchgeführt.

Zoom: Bei der Zoominteraktion muss zwischen einem semantischen und einem optischen Zoom unterschieden werden. Ein semantisches Zoomen ändert die Datenansicht dahingehend, dass die dargestellte Hierarchieebene geändert wird. Diese Interaktion kann auf die *OLAP*-Operationen Drill-Down und Roll-Up abgebildet werden. Optisches Zoomen verändert einen dargestellten Datenbereich. Es findet jedoch keine Änderung an dem Datenmodell statt. Eine Zoomoperation kann somit sowohl die Daten als auch die Ansicht ändern, weshalb keine eindeutige Einordnung durchgeführt werden kann.

Selektion: Mit Hilfe einer Selektion wird ein spezifischer Datensatz in einer Visualisierung hervorgehoben. Es ist möglich diese Hervorhebung nur auf Sichtebene durchzuführen, indem die Ansicht entsprechend angepasst wird. Da allerdings gefordert ist, dass eine Verknüpfung auf Basis der Selektion durchgeführt wird [FA.16] ist eine Modelländerung notwendig.

Kodierung: Mit Hilfe der Kodierung wird die visuelle Repräsentation der Daten geändert. Bereits die Beschreibung der Interaktion führt zu der Erkenntnis, dass die Interaktion auf eine Sichtoperation abgebildet werden kann.

Umgestaltung: Die Umgestaltung ermöglicht es einem Anwender die Daten aus einer anderen Sicht zu betrachten. Dies kann zum Beispiel eine Umsortierung der Daten in einer Tabelle oder die Anwendung der *OLAP*-Rotating-Operation sein.

Explore: Mit Hilfe von Explore-Interaktionen kann der Benutzer in der dargestellten Datenmenge navigieren und unterschiedliche Bereiche anzeigen. Dies erfordert allerdings keine Änderung der Datenmenge, weshalb die Interaktion auf eine Sichtoperation abgebildet werden kann.

	Abbildung auf:	
Interaktion	Wertoperation	Sichtoperation
Verzerrung		✗
Filtern	✗	
Projektion	✗	
Linking und Brushing	✗	
Zoom	semantischer Zoom	optischer Zoom
Selektion	✗	
Kodierung		✗
Umgestaltung	✗	
Explore		✗

Tabelle 6.13: *Abbildung von Interaktionen auf Wert- und Sichtoperationen*

Zusammenfassend ergeben sich die aus der Tabelle 6.13 ersichtlichen Wertoperationen, die auf das Datenmodell abgebildet werden müssen. In der Tabelle sind Interaktionen und Wert- sowie

Sichtoperationen gegenübergestellt. Die Spalte auf die eine Interaktion abgebildet werden kann, ist mit einem ✗-Symbol gekennzeichnet. Im weiteren Verlauf wird beschrieben, wie diese Operationen auf das Modell abgebildet werden. Die Elementbezeichnungen des Modells beziehen sich auf die Abbildung 6.3, in der das Datenmodell dargestellt ist.

Filterung: Eine Filterung der Daten kann im Datenmodell durchgeführt werden, indem die bei der weiteren Betrachtung nicht mehr relevanten Wertepaare aus dem Datenmodell gelöscht werden.

Projektion: Bei der Projektion, bzw. der Dicing *OLAP*-Operation, wird ein mehrdimensionales Modell um eine oder mehrere Dimensionen verringert. Da Dimensionen mit den Achsen des Modells vergleichbar sind, kann diese Operation durchgeführt werden, indem eine Achse aus dem Datenmodell entfernt wird.

Semantischer Zoom: Der durch den semantischen Zoom hervorgerufene Wechsel der Hierarchieebene, sowie die Drill-Down und Roll-Up *OLAP*-Operationen, können durch einen Wechsel der betrachteten Achsen im Datenmodell durchgeführt werden. Eine Achse enthält einen Verweis auf die Kind- und Vaterelemente. Bei einem Roll-Up wird die Achse, welche durch die Kind-Vater-Beziehung `AxisParents` angegeben wurde, verwendet. Ein Drill-Down wird durch den Wechsel zu den durch die Beziehung `AxisContainsAxis` verwiesenen Achsen, durchgeführt.

Selektion: Die Selektion eines Elementes ist in der Anforderung [FA.15] gefordert. Damit diese Operation auf das Modell abgebildet werden kann, wird in den Modellelementen `ValuePair`, `AxisValue` und `Axis` das zusätzliche Attribut `IsSelected` eingefügt. Dieses boolsche Attribut wird gesetzt, sobald eine Selektion der Achse oder des Wertepaares durchgeführt wird.

Linking und Brushing: Die Abbildung von dieser Interaktion, durch die Anforderung der Selektionsverknüpfung [FA.16] besonders herausgestellt, erfordert eine Anpassung des Datenmodells und des Systemmodells. Das Datenmodell muss eine Selektionsoperation abbilden können. Diese Eigenschaft wird durch das Attribut `IsSelected` gegeben. Außerdem wird das Systemmodell (vgl. Abbildung 6.2) mit der Beziehung `SystemElementIsLinked` um die Möglichkeit erweitert, Verknüpfungen zwischen Systemmodellen abzubilden.

Umgestaltung: Die Umgestaltung, also die Änderung der Reihenfolge der Elemente, kann direkt auf das bestehende Modell abgebildet werden. Vor dem Hintergrund, dass die Implementierung der Beziehungen als Liste durchgeführt wird, ist die Umgestaltung ein Positionswechsel in der Liste.

Die identifizierten Wertoperationen können auf das Datenmodell abgebildet werden, womit die Anforderung [FA.10] erfüllt wird. Einige Wertoperationen werden direkt auf Grundlage der gewählten Konstellation des Datenmodells umgesetzt. Für die Selektionsoperation ist speziell ein Attribut in das Modell eingefügt worden. Diese führt zu der Anforderung, dass eine Verknüpfung zwischen Datenmodellen möglich ist [FA.16].

6.4.4 Abgleich zwischen Datenmodellen

Eine Verknüpfung zwischen Visualisierungen [FA.16], deren Daten gleiche oder ähnliche Eigenschaften aufweisen, erfordert einen vorherigen Abgleich, um Gemeinsamkeiten zu identifizieren. Davon ausgehend, dass verschiedene Visualisierungsformen mit einer unterschiedlichen Menge an möglichen Interaktionen die Daten darstellen, ist eine Verknüpfung auf der visuellen Ebene nicht möglich. Es ist erforderlich, dass die Datenmodelle abgeglichen werden, wodurch eine Verknüpfung auf Datenebene

erreicht wird. Der Abgleich zwischen Datenmodellen wird in diesem Kapitel beschrieben. Er basiert auf dem im vorherigen Kapitel beschriebenen Datenmodell. Die weitere Betrachtung beschränkt sich auf die Selektionsverknüpfung, welche für dieses System gefordert ist [FA.16].

Auf Grundlage der Erkenntnis, dass Visualisierungen in der Regel aus Achsen und Werten bestehen, werden die Modellelemente `Axis`, `ValuePair` und `AxisValue` im Datenmodell zum Abgleich identifiziert (vgl. Abbildung 6.3). Eine Schwierigkeit tritt auf, wenn hierarchische Daten im Modell eingefügt werden. Beispielsweise kann eine Achse in einer Visualisierung eine Kindachse, bezogen auf die Vater-Kind-Beziehung, einer anderen Visualisierung sein. Es besteht aber auch die Möglichkeit, dass ein Datenmodell gleiche oder nicht in Beziehung zueinander stehende Achsen enthält. Für die Modellelemente `ValuePair` und `AxisValue` resultieren aus der hierarchischen Achseneigenschaft die gleichen Untergliederungsmöglichkeiten. Diese möglichen Beziehungen sind in der Tabelle 6.14 mit den Operationen, die bei einer Selektionsänderung durchgeführt werden müssen, dargestellt. Eine bidirektionale Verknüpfung ist nur möglich, wenn die abgeglichenen Elemente identisch sind. Dies hat den Grund, dass Änderungen eines Kindelementes nicht eindeutig auf das Vaterelement abgebildet werden können. Das Problem wird an dem folgenden Beispiel verdeutlicht. Sind einer Achse drei Kindachsen zugeordnet und es wird eine Kindachse selektiert, ist nicht eindeutig, welchen Selektionszustand die Vaterachse haben soll. Es sind unterschiedliche Interpretationen denkbar, wie beispielsweise, dass die Vaterachse selektiert wird, sobald die Mehrheit der Kindachsen selektiert wurde, oder dass die Vaterachse den Selektionszustand der letzten Änderung annimmt. Diese Fälle sind Domänenabhängig, weshalb keine allgemeine Lösung für dieses Problem gefunden werden kann. Für die Selektion eines Vaterelementes gilt, dass die Kindelemente auch selektiert dargestellt werden.

Quelldatenmodell	Zieldatenmodell	Operation bei Selektion eines Quellelementes
Kindachse	Vaterachse	Keine Operation
Vaterachse	Kindachse	Alle Kindachsen werden selektiert.
Gleiche Achse		Die spezifische Achse wird selektiert.
Kindwert	Vaterwert	Keine Operation
Vaterwert	Kindwert	Alle Werte des Zieldatenmodells der Achse des Vaterelementes werden selektiert.
Gleicher Wert		Der spezifische Wert wird selektiert.

Tabelle 6.14: *Die Abbildung von Selektionen bei einer Verknüpfung*

In der vorherigen Betrachtung ist implizit davon ausgegangen worden, dass den Datenmodellen, die in unterschiedlichen Visualisierungen dargestellt werden, in der Visualisierungspipeline die gleiche Quelle zugewiesen wurde. Ansonsten ist in der Regel keine Verknüpfung möglich. Die Ausnahme bilden Attribute mit räumlichen oder zeitlichen Eigenschaften. Raum und Zeit sind quellübergreifend miteinander Vergleichbar, weshalb diese Attribute eine besondere Aufmerksamkeit bedingen. Zeit wird im Datenmodell immer als Zeitraum repräsentiert. Hieraus ergeben sich die Möglichkeiten, dass ein Zeitwert innerhalb oder außerhalb eines spezifizierten Zeitraumes liegt (vgl. Abbildung 6.4 Zeitraum a und b).

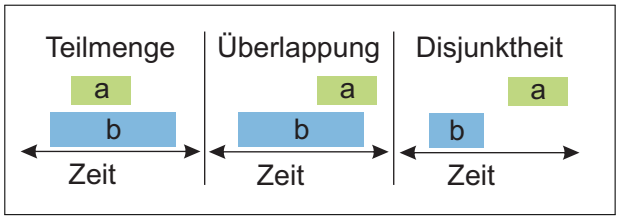


Abbildung 6.4: *Mögliche Beziehungen in Zeiträumen*

Für den Fall, dass der Wert (b) innerhalb des Zeitraumes (a) ist, wird eine Selektion vorgenommen, wenn der Zeitraum (a) selektiert wird, ansonsten wird keine Selektion durchgeführt. Des Weiteren kann ein Zeitraum sich mit einem anderen Zeitraum überlappen. Die Abbildung einer Selektion eines Zeitraumes ist Vergleichbar mit der Kind-Vater-Selektionsabbildung. Aus diesem Grund wird keine Selektion durchgeführt. Für einen räumlichen Abgleich können die gleichen Eigenschaften der Überlappung, vollständige Disjunktheit und Teilmenge festgestellt werden. Deshalb gelten die gleichen Selektionsverknüpfungsregeln. Eine Zusammenfassung ist in der Tabelle 6.15 dargestellt.

Beziehung zwischen den Attributen	Operation bei Selektion eines Quellelementes
Überlappung	Keine Operation
vollständige Disjunktheit	Keine Operation
Teilmenge	Die Teilmenge wird in Abhängigkeit der Obermenge selektiert.

Tabelle 6.15: *Die Abbildung von Selektionen bei einer Verknüpfung: Behandlung von Raum und Zeitattributen*

6.4.5 Einordnung der MUSTANG-Plattform zu den Anforderungen

Im Grundlagenabschnitt ist die *MUSTANG*-Plattform vorgestellt worden (vgl. Kapitel 2.2). Sie weist mit den aufgeführten Anforderungen in einigen Bereichen Übereinstimmungen auf, weshalb die Plattform im weiteren Verlauf näher in diesem Zusammenhang betrachtet wird, mit dem Ziel evtl. Anforderungen mit Hilfe der Plattform zu erfüllen.

Die *MUSTANG*-Plattform ist für den Zugriff auf multidimensionale Datenbanken konzipiert und stellt Möglichkeiten zur Einbindung solcher Datenbanken in ein System zur Verfügung. Diese erste Eigenschaft stimmt mit der Anforderung [FA.1] überein. Die Plattform abstrahiert dabei, die für die Datenbankabfrage nötigen individuellen Befehle. Stattdessen wird ein einheitlicher Service zum Zugriff auf die Datenmenge zur Verfügung gestellt. Dieser Service bietet die Möglichkeit an, eine spezifische Teilmenge der Datenmenge abzufragen und stellt das Ergebnis in einem multidimensionalen Modell zur Verfügung. Des Weiteren ist mit Hilfe des Services möglich, *OLAP*-Operationen abzubilden, indem eine angepasste Abfrage entsprechend der ausgeführten *OLAP*-Operation erstellt wird. Eine weitere Anforderung ist die Möglichkeit der Verknüpfung mit statistischen Maßzahlen [FA.8]. *MUSTANG* bietet neben der Serverkomponente die Statistikkomponente an. Mit Hilfe dieser Komponente, ist die

Verbindung von Maßzahlen und *OLAP*-Kennzahlen, genannt berechnete Kennzahlen, möglich. Diese Verbindungen müssen vor der Erstellung explizit definiert werden. Anschließend stehen sie im weiteren Analyseverlauf zur Verfügung. Für einen Systementwickler, der die *MUSTANG*-Plattform einbindet, ist die Verwendung von berechneten Kennzahlen im Vergleich mit *OLAP*-Kennzahlen transparent. Die letzte vorgestellte Komponente von *MUSTANG* ist die GEO-Datenbank. Die GEO-Datenbank enthält Informationen über geographische Ausprägungen, wie zum Beispiel die geographische Beschreibung der Postleizahlregionen. Mit Hilfe dieser Daten ist der Aufbau einer thematischen Karte [FA.6] möglich.

MUSTANG hilft bei der Erfüllung einiger Anforderungen. Aus diesem Grund wird die Plattform als ein Systemelement in das zu entwickelnde System eingepflegt. Da die Plattform Daten bereitstellt, wird sie im weiteren Verlauf als Quellelement behandelt. Als wichtige Eigenschaften der *MUSTANG*-Plattform sind dabei die Folgenden identifiziert worden:

- Zugriff auf multidimensionale Daten
- Verknüpfung von Daten mit statistischen Maßzahlen
- Anbindung einer GEO-Datenbank
- Abstraktion von *OLAP*-Operationen

MUSTANG ist eine Datenquelle im Gesamtsystem. Sie besteht aus einem Systemelement und einem Datenmodell. Damit das System von einem Analysten verwendet werden kann, sind neben dem Konzept und den Modellen weitere Komponenten, wie beispielsweise eine *GUI* notwendig. Die Komponenten werden im folgenden Architekturentwurf vorgestellt.

6.5 Zusammenfassung

In diesem Abschnitt ist neben den Anforderungsfällen und den funktionalen Anforderungen das *VAT*-System vorgestellt worden, welches die vorher identifizierten Anforderungen erfüllt. Das *VAT*-System basiert auf einem Data-Flow-Modell, welches für die speziellen Anforderungen im Bereich der visuellen Analyse, um die Möglichkeit an einem Knoten mehrere Kanten zu eröffnen, erweitert wurde. Des Weiteren ist die Unterscheidung zwischen Quellen, zur Bereitstellung einer Datenmenge, Transformern, zur Bearbeitung einer Datenmenge, und Senken, zur Darstellung, eingeführt worden. Dieses Konzept mündet in das Systemmodell und das Datenmodell. Das Systemmodell wird für die Abbildung des angepassten Data-Flow-Konzeptes modelliert. Besonderes Augenmerk ist auf das Datenmodell gelegt worden, das die zu analysierenden Daten sichert. Es ist vor dem Hintergrund des Klassifikationssystems (vgl. Kapitel 5.6) und den Anforderungen, die sich aus der Behandlung von multidimensionalen Daten ergeben, entwickelt worden. Im nächsten Schritt sind die Operationen, die auf das Datenmodell abgebildet werden müssen, identifiziert worden. Mit Hilfe der Aufteilung in Sicht- und Wertoperationen ergibt sich, dass Filterung, Projektion, Linking und Brushing, semantischer Zoom, Selektion sowie Umgestaltung auf das Datenmodell als Wertoperationen abgebildet werden müssen. Von diesen Operationen können die Projektion und der semantische Zoom auf *OLAP*-Operationen abgebildet werden. Die Anderen können auf das Datenmodell reflektiert werden. Abgeschlossen wird der Abschnitt durch die Behandlung der Anforderung Selektionsverknüpfung. In Bezug auf das Datenmodell sind die Möglichkeiten eines Abgleiches beschrieben worden mit der speziellen Behandlung von Zeit- und Ortsattributen, die nicht als Zeitpunkte bzw. Ortspunkte interpretiert werden, sondern immer als Zeitspanne und Fläche, wodurch ein Vergleich ermöglicht wird. Die

MUSTANG-Plattform unterstützt bei der Bereitstellung von geographischen und multidimensionalen Datensätzen, ebenso wie die Verknüpfung von Daten mit statistischen Maßzahlen. Auf Grundlage dieses Konzeptes können viele Anforderungen des Kapitels 6.2 umgesetzt werden. Das *VAT*-System ist durch Visualisierungen und Quellen als Systemelemente so erweiterbar, dass alle Anforderungen erfüllt werden können. Die Umsetzung, wie diese Systemelemente eingebunden werden sowie der Entwurf des *VAT*-Systems werden in den nachfolgenden Abschnitten durchgeführt.

7 Entwurf

Der Entwurf des *VAT*-Systems folgt dem Top-Down-Ansatz. Bei diesem Ansatz wird ein bestehendes Problem sukzessiv in kleinere Teilprobleme aufgeteilt, bis diese gelöst werden können. Hierzu wird zunächst ein Grobentwurf erstellt, in dem die Architektur und die Benutzungsoberfläche beschrieben wird. Im abschließenden Feinentwurfskapitel wird mit Hilfe von Klassendaigrammen und Sequenzdiagrammen auf die einzelnen Teile des *VAT*-Systems näher eingegangen. Der folgende Abschnitt Implementierungsaspekete baut auf den Grobentwurf auf und stellt besondere Eigenschaften bezogen auf die Implementierung dar. Zunächst wird der Grobentwurf vorgestellt.

7.1 Grobentwurf

Zentraler Gedanke bei der Entwicklung der Architektur ist das Einbinden von Analyseelementen zu ermöglichen, die auch unterschiedliche Datenmodelle verwenden. Damit diese zusammen in einem Visualisierungspfad (vgl. Kapitel 6.3), der im Systemmodell abgebildet wird, interagieren können, sind Transformationen zwischen den evtl. unterschiedlichen Datenmodellen notwendig. Würde jedes Analyseelement zu jedem eine Transformation bereitstellen, würde sich die Anzahl der zu entwickelnden Transformationen im Vergleich mit der Menge an Analyseelementen rasant erhöhen. Beispielsweise wären bei fünf Analyseelementen bereits zehn Transformationen für jedes Element zu entwickeln. Aus diesem Grund wird das im Kapitel 6.4.2 entwickelte Datenmodell als Abstraktionsschicht zwischen den Analyseelementen verwendet, wodurch die Entwicklung auf zwei Transformationen beschränkt wird. Allerdings bedingt diese Eigenschaft eine spezielle Architekur (vgl. Abbildung 7.1).

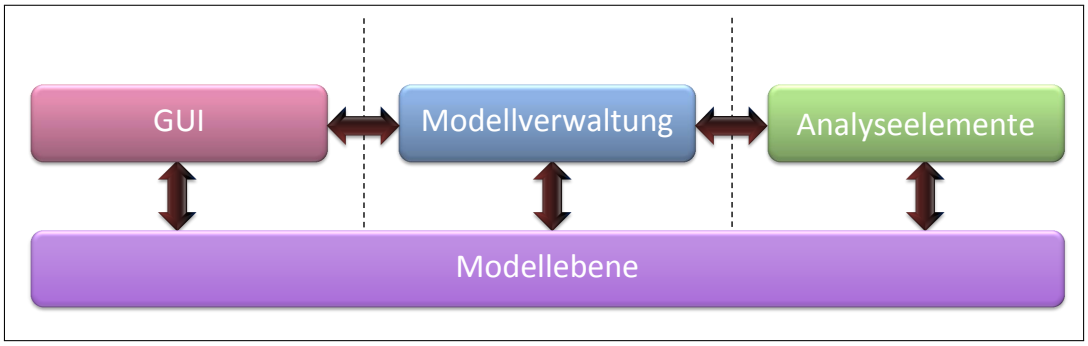


Abbildung 7.1: *Die Grobarchitektur des VAT-Systems*

Die Grobarchitektur besteht aus den horizontalen Schichten `GUI`, `Modellverwaltung` und `Analyseelemente`, die auf die `Modellebene` zugreifen. Die einzelnen Schichten werden im Folgenden detaillierter dargestellt.

GUI: In der *GUI*-Schicht werden Komponenten für die Darstellung des Gesamtsystems eingeordnet. Die Aufgabe dieser Schicht ist es dem Anwender einen ersten Zugang zum System zu ermöglichen und mit dem System zu interagieren.

Modellverwaltung: Die mittlere Schicht bildet die Modellverwaltung. In dieser Schicht werden alle Komponenten zur Erstellung, Verwaltung und Transformation von Modellen und Modellelementen

eingefügt. In der *GUI*-Schicht werden diese Informationen zur Darstellung der Analyseelemente verwendet. Außerdem stellt die Schicht Transformationen bereit, um zwischen unterschiedlichen Modellen zur Datenhaltung zu wechseln und somit unterschiedliche Analyseelemente miteinander zu verbinden.

Analyseelemente: In der Analyseelementschicht werden alle Werkzeuge zur eigentlichen Analyse eingegliedert. Hierunter sind Quellen, Transformer und Senken (vgl. Kapitel 6.3) zu verstehen. Komponenten dieser Schicht können auf die Modellverwaltungsschicht zugreifen, um Modelle zu erstellen sowie Transformationen durchzuführen. Zudem ist ein Zugriff auf auf die Modellebene möglich, um die Beschreibung eines verwendeten Modells, ohne die eine effektive Verwendung eines Modells nicht möglich ist, aufzurufen.

Modellebene: Die übergreifende Modellebene ergibt sich aus den im vorherigen Abschnitt beschriebenen Modellen (Datenmodell und Systemmodell). Sie fasst die Systemelemente, sowie den visuellen Pfad zusammen und bildet eine Abstraktionsschicht zwischen unterschiedlichen Datenmodellen. Die Schichten können auf diese Ebene zugreifen, um Änderungen am Systemmodell durchzuführen oder einen aktuellen Zustand eines Modellelementes abzufragen. Diese Eigenschaft wird genutzt, um in der *GUI*-Schicht den Visualisierungspfad darzustellen.

Nachdem die Schichten im System identifiziert sind, kann aufbauend auf diesem Wissen, die Architektur weiter verfeinert werden. Dieser Schritt wird im weiteren Verlauf durchgeführt.

7.1.1 Verfeinerung des Grobentwurfs

Die Verfeinerung des Systems wird anhand der identifizierten Schichten durchgeführt. In diesem Kapitel werden die Schichten sukzessiv auf Komponenten-Ebene verfeinert. Für jede Schicht wird separat die Verfeinerung durchgeführt. Eine farbliche Korrelation mit den Farben der Schichten im Grobentwurf ermöglicht die Zuordnung der Komponenten. Begonnen wird mit der Verfeinerung der *GUI*.

GUI-Schicht

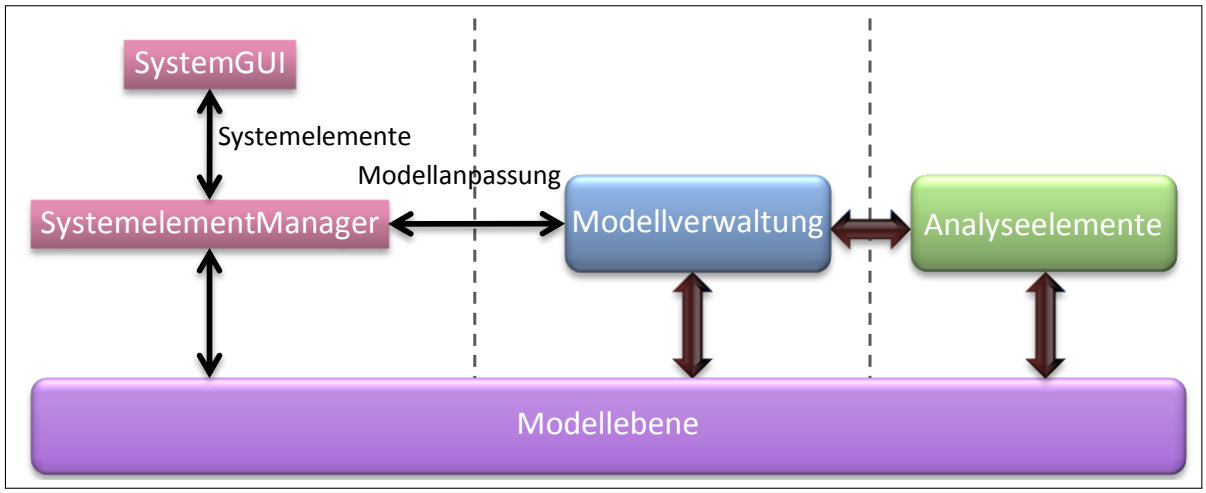


Abbildung 7.2: *Verfeinerung der GUI-Schicht*

Die Aufgabe der *GUI*-Schicht ist, dem Anwender den Zugang zu den verfügbaren Analysewerkzeugen zu ermöglichen und eine Analyse durchzuführen. Die Analyse bedingt allerdings die Möglichkeit, Analysepfade zu erstellen, da sonst keine Verbindung zwischen den verschiedenen Analysewerkzeugen erstellt werden kann. Änderungen an dem Analysepfad werden durch die Modellverwaltungsschicht ermöglicht. Die *GUI*-Schicht dient somit nicht nur als reine graphische Repräsentation, sondern ermöglicht auch Änderungen an dem Modell, bzw. dem Visualisierungspfad. Um eine klare Trennung zwischen *GUI* und Interaktionslogik zu erzielen, wird die *GUI*-Schicht in zwei Komponenten aufgeteilt (vgl. Abbildung 7.2). Die `SystemGUI`-Komponente zur Darstellung der verfügbaren Analysewerkzeuge und des Analysepfades und die `SystelementManager`-Komponente, die Interaktionen zur Änderung des Pfades auf die Modellverwaltungsschicht abbildet.

Modellverwaltungsschicht

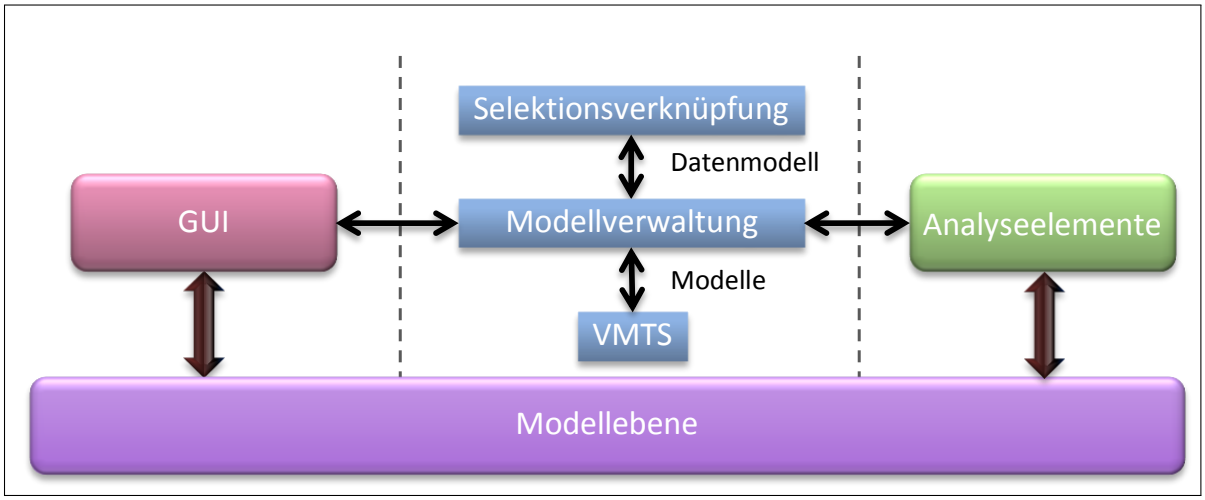


Abbildung 7.3: *Verfeinerung der Modellverwaltungsschicht*

Die Modellverwaltungsschicht ist zwischen der *GUI*-Schicht und den Analyseelementen eingeordnet. Die Schicht ist für die Verwaltung der zur Verfügung stehenden Modelle und Transformationen verantwortlich. Die *GUI*-Schicht greift auf diese zu, um den Visualisierungspfad zu ändern. Eine Änderung des Visualisierungspfades kann sowohl das Einfügen und Löschen eines Systemelementes, wie auch die Pfaderstellung oder die Erstellung von Selektionsverknüpfungen sein. Von der Analyseelementschicht wird der Zugriff auf Transformationen und Modellerstellungsmöglichkeiten gefordert. Aus diesen Anforderungen ergibt sich, dass die Schicht in drei Komponenten untergliedert wird (vgl. Abbildung 7.3). Das Datenmodell und das Systemmodell (vgl. Kapitel 6.4) werden mit Hilfe von *VMTS* modelliert. Um Modelle verwebden zu können, ist eine Initialisierung mit Hilfe von *VMTS* notwendig. Damit nicht jede Komponente in der Modellverwaltungsschicht einen eigenen Zugriff auf *VMTS* implementieren muss, wird dieser von der *VMTS*-Komponente gekapselt. Als weitere Aufgabe ist die Anpassung des Visualisierungspfades und die Verwaltung von Transformationen identifiziert worden. Für diese wird die `Modellverwaltungs`-Komponente eingefügt. Die dritte Komponente ergibt sich aus der Forderung der Selektionsverknüpfung, welche als `SelektionsVerknüpfung`-Komponente in diese Schicht eingefügt wird.

Analyseelementschicht

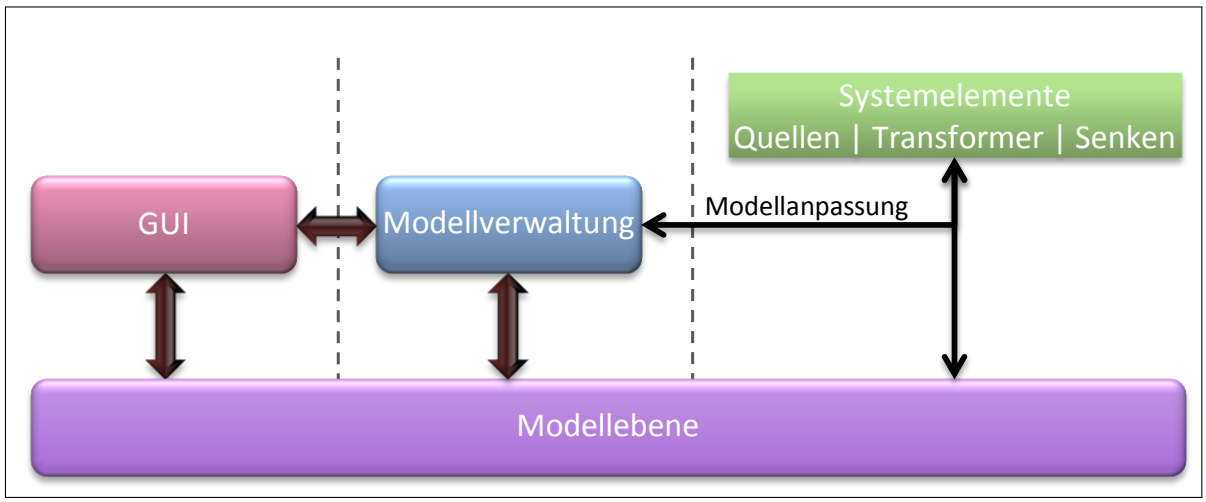


Abbildung 7.4: *Verfeinerung der Analyseelementschicht*

Die Analyseelementschicht stellt Werkzeuge zur Analyse bereit (vgl. Abbildung 7.4). Diese Werkzeuge greifen auf die Modellverwaltungsschicht zurück, um Änderungen am Modell durchzuführen oder neue Modellinstanzen zu erstellen. Werkzeuge werden im *VAT*-System auf `Systemelemente` des Systemelementmodells (vgl. Kapitel 6.4.1) abgebildet. Hierdurch können Quellen, Senken und Transformer eingefügt werden. Für jedes Analysewerkzeug wird ein Systemelement erstellt, welches dieses Werkzeug im Gesamtsystem beschreibt. Zusätzlich zu der Beschreibung des Systemelementes weißt jedes Element ein Datenmodell auf. Damit ein effektiver Zugriff auf die Daten eines Systemelementes ermöglicht wird, findet innerhalb von Analysewerkzeugen ein Zugriff auf die Modellbeschreibungsschicht statt. In dieser ist die Beschreibung des Datenmodells hinterlegt.

Modellebene

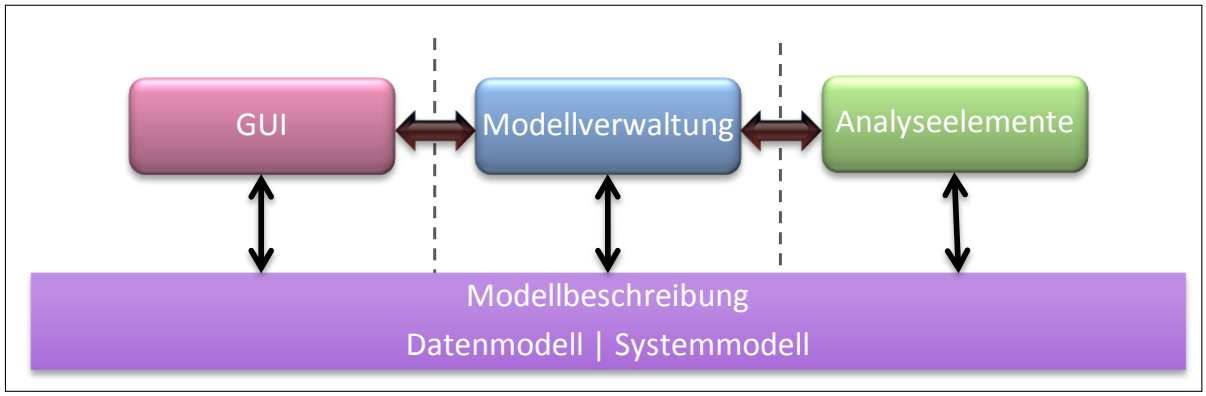


Abbildung 7.5: *Verfeinerung der Modellebene*

Auf die Modellebene können alle Schichten zugreifen. Die Aufgabe ist es, Beschreibungen des Daten- und Systemmodells (vgl. Kapitel 6.4) zur Verfügung zu stellen (vgl. Abbildung 7.5). Diese werden von *VMTS* automatisch generiert, weshalb die Ebene direkt auf die generierten Komponenten abgebildet

werden kann. Die *GUI*-Schicht greift auf das Systemmodell zurück, um den Visualisierungspfad darzustellen. Die Modellverwaltungsschicht initialisiert die Modelle und fügt neue Systemelemente in das Modell ein. Außerdem wird auf die Beschreibung des Datenmodells zur Erstellung von Verknüpfung zugegriffen. Die Analyseelementschicht greift auf diese Ebene zu, um eine Beschreibung des Datenmodells oder andere Fremdmodelle auszulesen.

Die vorgestellten Komponenten der verfeinerten Architektur werden im weiteren Verlauf näher beschrieben. Zunächst wird explizit auf die `SystemGUI`-Komponente eingegangen, welche die graphische Benutzungsoberfläche des Systems bereitstellt.

7.1.2 Entwurf der Benutzungsoberfläche

Zentrale Idee bei der Entwicklung der Benutzungsoberfläche ist die Darstellung der Visualisierungspfade (vgl. Kapitel 6.3), wodurch gleichzeitig ein direkter Einfluss auf diese ermöglicht wird. Um auch große Pfade für die Analyse zu ermöglichen, wird das Design des *VAT*-Systems auf eine größtmögliche Darstellungsfläche für Visualisierungspfade ausgelegt. Aus diesen Überlegungen ergibt sich das in der Abbildung 7.6 dargestellte Konzept der *GUI*.

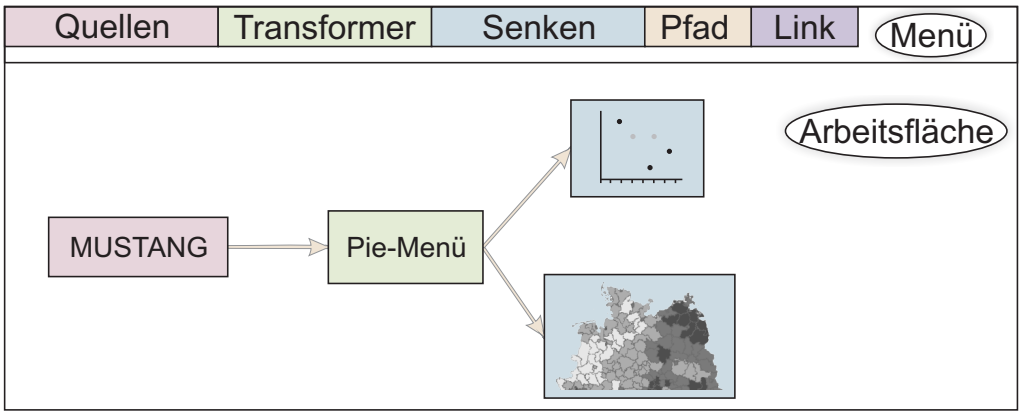


Abbildung 7.6: *Der GUI-Entwurf des VAT-Systems*

Die *GUI* ist in die Bereiche *Menü* und *Arbeitsfläche* aufgeteilt. Das Menü fasst farblich unterteilt die für eine Analyse bereitgestellten Systemelemente zusammen. Quellen werden in Rosa, Transfomer in Grün und Senken in Blau dargestellt. Sobald auf eines der Elemente geklickt wird, wird dieses im zweiten Bereich, der Arbeitsfläche, angezeigt und kann an eine benutzerspezifische Position verschoben und entsprechend den Wünschen des Analysten skaliert werden. Die weiteren Menüpunkte Pfad und Link werden zur Bildung des Visualisierungspfades benötigt. Mit dem Menüpunkt Pfad kann ein Anwender ein Systemelement mit einem anderen verbinden. Mit Hilfe einer Kette von Pfaden wird ein Visualisierungspfad erstellt. Der Menüpunkt Link ermöglicht einem Anwender die Selektionsverknüpfung zwischen zwei Systemelementen zu erstellen. Diese wird auch in der Arbeitsfläche durch einen Pfeil dargestellt.

7.1.3 Paketaufteilung

Die Pakete des *VAT*-Systems sind in der Abbildung 7.7 dargestellt. Hierbei ist der Übersichtshalber der Präfix `UniOldenburg.VATS` mit `UV` abgekürzt worden. Nicht selbstentwickelte Pakete sind in Grau dargestellt. Damit eine Assoziation zwischen den Schichten und den Paketen erfolgen kann, ist der Farbton der Pakete an den Farbton der Schichten der Verfeinerung und Grobarchitektur (vgl. Abbildung 7.1) angelehnt.

Die Paketstruktur ergibt sich aus den Komponenten, die in der ersten Verfeinerungsstufe der Architektur identifiziert wurden (vgl. Kapitel 7.1.1). Zunächst ist der Namespace `UV.SystemGUI` zu erwähnen. Zu diesem Paket werden alle für die Darstellung des Gesamtsystems benötigten Klassen eingefügt. Diese können wiederum in `Menu`-Elemente und `Workarea`-Elemente gegliedert werden. Innerhalb der `Workarea` werden die Klassen zur Darstellung der Arbeitsfläche (vgl. Abbildung 7.6) eingefügt. Zur Kapselung der Menüelemente des Programmes wird der Namespace `UV.SystemGUI.Menu` erstellt. Innerhalb der Arbeitsfläche werden die instanziierten Systemelemente dargestellt. Damit eine Trennung zwischen der graphischen Ansicht und der Interaktionslogik bestehen bleibt, ist die Komponente `UV.SystemelementManager`, welche im gleichnamigen Paket eingefügt wird, eingeführt worden. Innerhalb des Namespaces werden die Klassen zum Zugriff auf die Modellverwaltungsschicht vorgehalten. Zum Zugriff gehört neben der Möglichkeit den Analysepfad zu ändern, der Zugriff auf die Selektionsverknüpfungskomponente zur Erstellung von Verknüpfungen. Es wird somit erreicht, dass das Paket `UV.SystemGUI` nur vom Paket `UV.SystemelementManager` abhängig ist. Das Paket `UV.Modelmanagement` entspricht der Modellverwaltungsschicht. Es wird in die Unterpakete `Handling`, zur Kapselung des Zugriffs auf Modelle, `Transformation` für Transformationen zwischen Modellen und `VMTSAccess`, zum Zugriff auf die VMTS-Komponente unterteilt. Zusätzlich enthält wird das Paket `SelectionLink` zur Abbildung von Selektionsverknüpfungen eingefügt. Das `UV.Systemelement` ist das größte Paket im *VAT*-System. In diesem Paket werden die Zugriffe auf die Quellen, Transformer und Senken eingeordnet. Eine direkte Untergliederung des Paketes in diese Systemelementgruppen ist deshalb sinnvoll. Da von anderen Systemen Komponenten verwendet werden, hängt das `UV.Systemelement` auch von den entsprechend eingebundenen Teilsystemen ab. Diese sind im Einzelnen die *MUSTANG*-Plattform, die als Datenquelle eingebunden wird sowie eine thematische Karte bereitstellt, das `Tap.Controls`-Paket, in welchem ein Blasendiagramm und ein Pie-Menü eingegliedert ist, und `SharpMap` zur Darstellung einer thematischen Karte. Außerdem besteht eine Abhängigkeit zum Paket `UV.Punktdiagramm`. In diesem Namespace wird das beispielhafte Punktdiagramm aus dem Grundlagenabschnitt eingeordnet (vgl. Kapitel 2.1.3). Die Modellbeschreibungen für das Datenmodell und das Systemmodell werden von *VMTS* im Paket `VMTS.Domains` eingefügt. Diese enthalten die Beschreibungen der Modelle, welche für einen effektiven Zugriff auf das Modell notwendig sind.

7.2 Feinentwurf

Nachdem die Pakete im vorangegangenen Kapitel identifiziert wurden, werden sie in diesem Kapitel weiter verfeinert. Als Hilfsmittel zur Verfeinerung werden Klassen- und Sequenzdiagramme verwendet. Der Aufbau der weiteren Unterabschnitte gliedert sich folgendermaßen: Zunächst wird die benötigte Funktionalität des Paketes beschrieben und anschließend eine Gruppierung in Klassen und Methoden vorgenommen. Mit der Entwicklung der Pakete der Modellverwaltungsschicht wird begonnen.

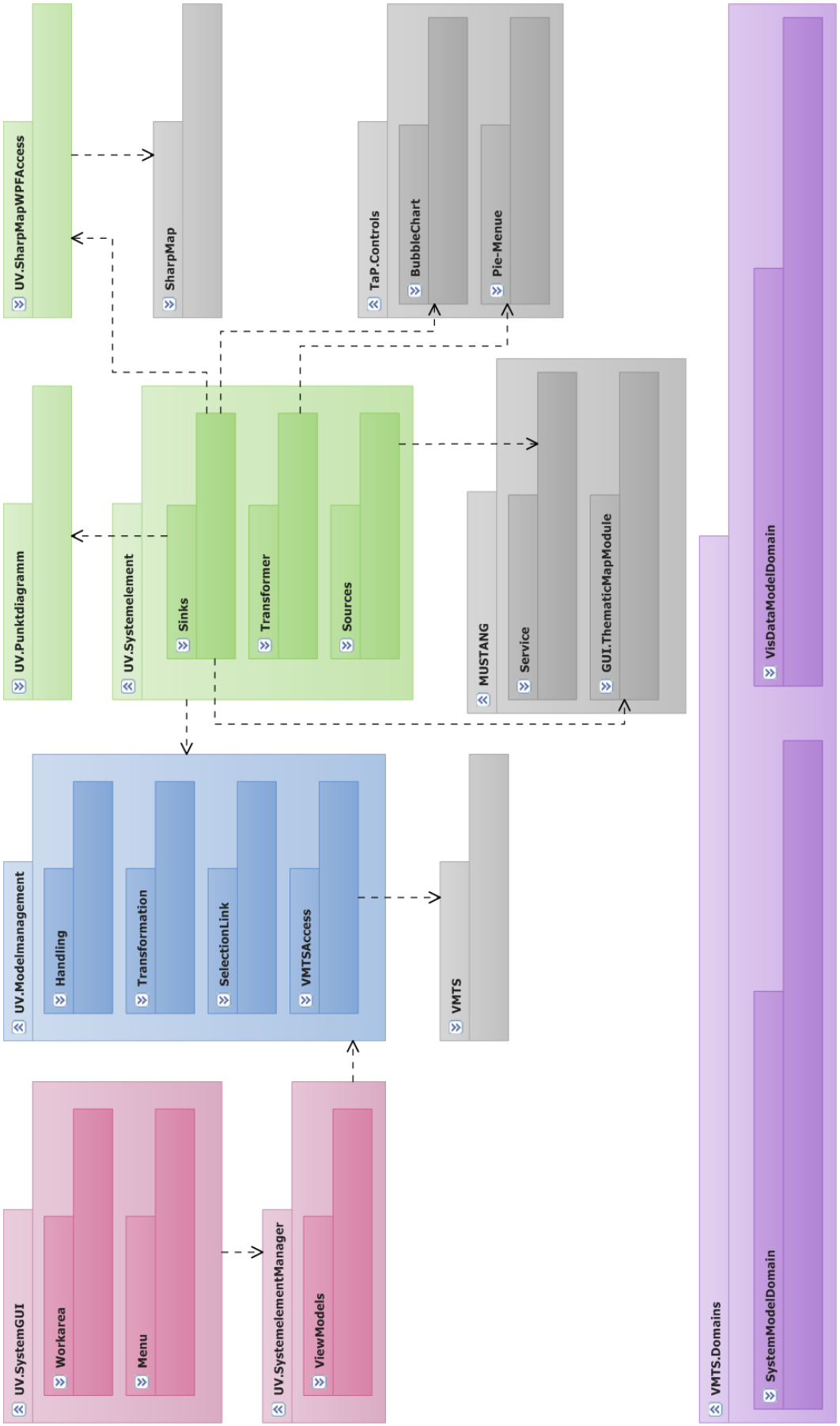


Abbildung 7.7: *Die Namespaces des VAT-Systems*

7.2.1 Das Paket UniOldenburg.VATS.Modelmanagement

Das Paket `Modelmanagement` des *VAT*-Systems soll die Modellverwaltung innerhalb des Systems ermöglichen. Hierunter ist die Erstellung und Verwaltung von Modellen, die Selektionsverknüpfung, die Verwaltung von Transformationen und der Zugriff auf den Analysepfad (vgl. Kapitel 6.3) zu verstehen. Das Paket ist in weitere Subpakete untergliedert, wie dem Paketdiagramm entnommen werden kann (vgl. Abbildung 7.7). Damit die Verwendung der Subpakete vereinfacht wird, zum Beispiel die Registrierung von Transformationen, werden zusätzlich Klassen eingefügt. Diese werden zunächst dargestellt (vgl. Abbildung 7.8), bevor auf die Subpakete im Einzelnen eingegangen wird.

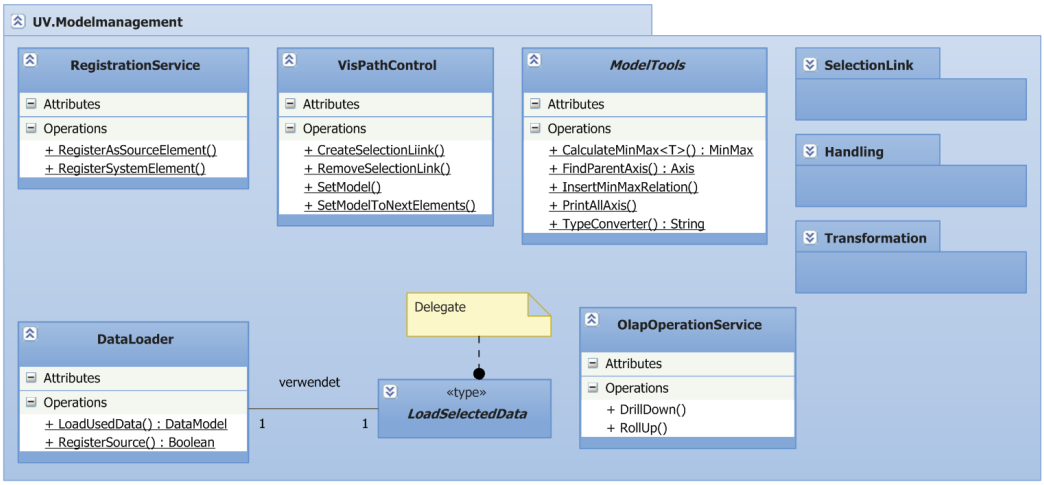


Abbildung 7.8: *Das Paket UniOldenburg.VATS.Modelmanagement*

Das *VAT*-System soll das dynamische Nachladen einer ausgewählten Datenmenge unterstützen. Diese Daten werden von einer Quelle, die in der Regel eine multidimensionale Datenbank abstrahiert, bereitgestellt, bzw. nachgeladen. Hierzu ist es notwendig, dass das Quellsystemelement dem System eine entsprechende Funktionalität bekannt gibt. Zu diesem Zweck wird die Klasse `DataLoader` in das System eingefügt. Mit Hilfe eines Delegaten wird erreicht, dass Quellen eine Nachlademethode innerhalb der Klasse registrieren können. Diese wird für die entsprechende Quellkomponente gesichert und sofern Daten nachgeladen werden müssen, kann die Methode `LoadUsedData` aufgerufen werden. Es ist eine Unterscheidung notwendig, ob ein Quellsystemelement oder ein Senken- bzw. Transformsystemelement verwendet wird, weil Quellen eine Nachlademethode aufweisen müssen. Diese Unterscheidung wird auch im `RegistrationService` durchgeführt, welcher neben der Registrierung der Nachlademethode für Quellen die Registrierung für Transformationen in den entsprechenden Unterpaketen übernimmt. Die Klasse soll einen Dienst zur Verfügung stellen, der die Anmeldung von Systemelementen (Quellen, Senken oder Transformern) erleichtert. Neben der Erstellung und Verwaltung von Modellen soll die Manipulation des Analysepfades (vgl. Kapitel 6.3) eine Hauptaufgabe des Paketes sein. Aus diesem Grund wird die Klasse `VisPathControl` entwickelt. Die Klasse bietet einen Dienst an, um Selektionsverknüpfungen zu verwalten und Modelländerungen eines Systemelementes innerhalb des Analysepfades weiterzureichen. Selektionsverknüpfungen werden mit Hilfe des Subpaketes `SelectionLink` bereitgestellt. Um Modelländerungen, ausgehend von einem Systemelement im Analysepfad weiterzureichen, wird die Methode `SetModel` entwickelt. Diese setzt das Modell des übergebenen Systemelementes zu allen im Analysepfad folgenden Systemelementen. Weiterhin kann es sinnvoll sein, ein Modell direkt einem Systemelement zuzuweisen.

Hierzu wird die Methode `SetModelToNextElements` in die Klasse eingefügt. In der Klasse `ModelTools` werden Methoden eingefügt, die nicht direkt einer bisher beschriebenen Funktionalität zugeordnet werden können und in unterschiedlichen Bereichen benötigt werden. Da sie zustandlos ist, wird die Klasse statisch ausgeführt. In der Beschreibung des Datenmodells (vgl. Kapitel 6.4.2) ist ein Verweis auf die Extremwerte einer Achse dargestellt. Diese müssen immer dann berechnet werden, wenn ein Datenmodell erstellt oder die Achsenwerte geändert werden. Diese Funktionalität wird in der Methode `InsertMinMaxRelation` eingefügt. Des Weiteren wird eine Unterstützung zur Abbildung von Drill-Down und Roll-Up Operationen eingefügt. Zu diesem Zweck wird die statische Klasse `OlapOperationService` eingefügt.

UniOldenburg.VATS.Modelmanagement.SelectionLink

Das Paket soll die Funktionalität einer Selektionsverknüpfung zwischen Datenmodellen zur Verfügung stellen. Unter einer Selektionsverknüpfung ist zu verstehen, dass Selektionen von Werten und Wertepaaren eines Datenmodells auf ein verknüpftes Datenmodell abgebildet werden. Eine Visualisierungsform kann diese Selektion beispielsweise durch farbliche Hervorhebung darstellen. Bereits die Beschreibung, dass Änderungen an einem Objekt überwacht und weitergereicht werden, legt die Verwendung des Entwurfsmusters *Observer* nahe. Das Muster dient zur Weitergabe von Änderungen an einem Objekt und deren abhängige Strukturen. Aufbauend auf diesem Muster wird das Paket umgesetzt. Allerdings kann das Muster nicht in der Reinform verwendet werden, da eine Selektionsverknüpfung zwischen zwei gleichwertigen Datenmodellen durchgeführt wird. Es müssen nicht nur die Änderungen eines Datenmodells auf ein anderes übertragen werden, sondern es ist ein bidirektionaler Austausch der Selektionen gefordert. Aus diesem Grund wird die Erstellung einer Selektion in zwei einzelnen Schritten durchgeführt, bei der die beteiligten Datenmodelle jeweils einmal in der Rolle des Subjekts und Beobachters auf Verknüpfungseigenschaften analysiert werden. Das Subjekt gibt Änderungen an Selektionen bekannt, während der Beobachter auf diese Änderungen reagiert und in dem Fall der Selektionsänderung überträgt.

Dieses Konzept wird auf die Aufteilung der Klassen abgebildet. Das Klassendiagramm ist in der Abbildung 7.9 dargestellt. Die Klasse `SelectionLink` bildet eine Selektionsverknüpfung zwischen zwei `Systemelementen`, denen ein Datenmodell zugewiesen ist, ab. Hierzu stellt die Klasse Registrierungs- und Deregistrierungsmethoden zur Verfügung. Sofern beide Systemelemente zugewiesen wurden, werden zwei `TargetSourceSelectionLink`-Klassen instanziiert, bei dem die Quell- und Zielsystemelemente vertauscht sind. Zur Registrierung der Elemente werden die Methoden `RegisterSource` und `RegisterTarget` in die Klasse eingefügt. Außerdem stellt die Klasse eine Methode zur Analyse der Gemeinsamkeiten zwischen zwei Datenmodellen zur Verfügung. Hierbei werden basierend auf den Erkenntnissen des Kapitels 6.4.4, zum Abgleich des Datenmodells, Werte, Wertepaare und Achsen ermittelt, die miteinander verknüpft werden müssen. Bei der Bestimmung wird auf die Dienste `SelectionLinkUtils` zur Ermittlung von zeitlichen und räumlichen Abhängigkeiten und `ContainmentChecker` zur Ermittlung von hierarchischen Beziehungen zurückgegriffen. Die Verknüpfung wird mit Hilfe der generischen Klasse `Notification` durchgeführt. Diese stellt somit den Beobachter dar, während ein Wert, Wertepaar oder eine Achse das Subjekt ist. Falls Selektionsänderungen an dem Subjekt durchgeführt werden, erfolgt eine Benachrichtigung der Klasse `Notification`, welche die Änderungen an das verknüpfte Modellelement abbildet. Durch diese spezifische Kopplung zwischen Beobachter und Modellelement wird erreicht, dass nur Beobachter Änderungsinformationen erhalten, die diese auch benötigen. Hierdurch wird ein Nachteil des Observer-Entwurfsmusters umgangen. Damit bei Modelländerung die Kopplung wieder entfernt werden kann,

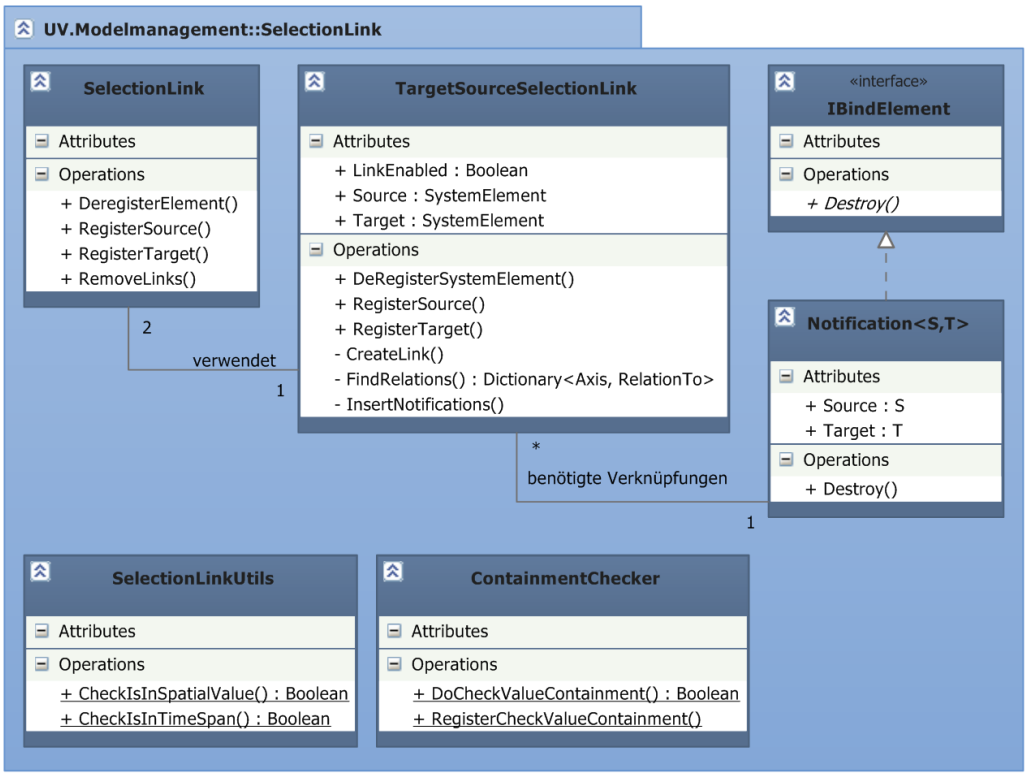


Abbildung 7.9: *Das Paket UniOldenburg.VATS.Modelmanagement.SelectionLink*

implementiert die Klasse `Notification` die Schnittstelle `IBindElement`. Der Status, ob eine Verknüpfung zwischen den beteiligten Datenmodellen durchgeführt wurde, wird in der Eigenschaft `LinkEnabled` zur weiteren Verarbeitung, zum Beispiel zur optischen Darstellung, veröffentlicht.

UniOldenburg.VATS.Modelmanagement.VMTSAccess

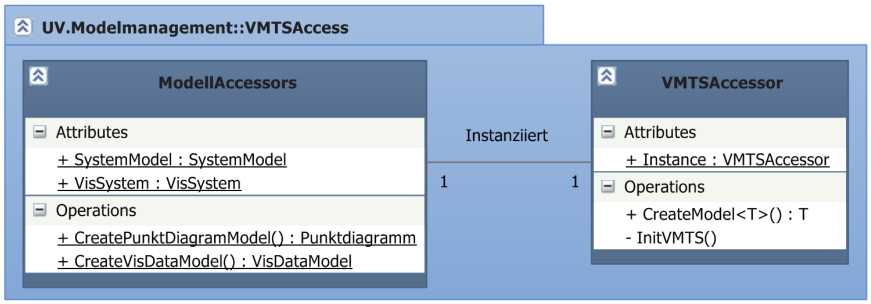


Abbildung 7.10: *Das Paket UniOldenburg.VATS.Modelmanagement.VMTSAccess*

Das `VMTSAccess`-Paket (vgl. Abbildung 7.10) soll den Zugriff auf die *VMTS*-Komponenten kapseln und einen Dienst bereitstellen, um Modelle zu erstellen. Die geforderte Funktionalität wird in zwei Klassen aufgeteilt. Eine, die für die Initialisierung und den Zugriff auf *VMTS* zuständig ist und Eine, welche für die abhängigen Komponenten und Pakete einen Zugriff auf diese Klasse bietet.

Der Zugriff von *VMTS* auf die gespeicherten Modelle und das Modellrepository ist nur mit einer Instanz zur Laufzeit möglich, die zuvor instanziiert werden muss. Aus diesem Grund wird für den Dienst `VMTSAccessor` das Entwurfsmuster *Singleton* verwendet, wodurch gleichzeitig verhindert wird, dass von einer Klasse mehrere Instanzen erzeugt werden können. Die einzige öffentliche Instanz der Klasse steht in der Eigenschaft `Instance` bereit. Durch Zuweisung von Sichtbarkeitsregeln wird erreicht, dass auf diese nur innerhalb des Paketes zugegriffen werden kann. Die generische Methode `CreateModel` ermöglicht die Instanziierung von *VMTS*-Modellen. Diese wird von der Klasse `ModellAccessoirs` verwendet, um den Zugriff auf Modelle zu ermöglichen. Die Klasse stellt hierzu einen nicht zustandsbehafteten Dienst bereit, weshalb eine statische Klasse verwendet wird. Die Eigenschaft `SystemModel` ermöglicht die Erstellung der Modellelemente des Systemmodells (vgl. Kapitel 6.4.1). Auf Basis dieses Zugriffes wird die Eigenschaft `VisSystem` zugewiesen, die eine Instanz des Modellelementes `VisSystem` bildet und als Container für Systemelemente dient. Des Weiteren stellt die Klasse eine Methode zur Erstellung eines Datenmodells (vgl. 6.4.2) und eines Punktdiagrammmodels zur Verfügung.

UniOldenburg.VATS.Modelmanagement.Transformation

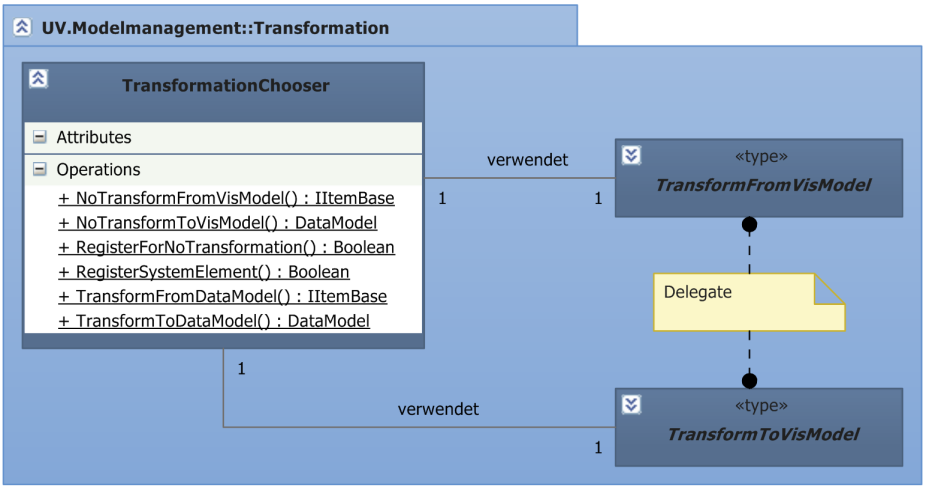


Abbildung 7.11: *Das Paket UniOldenburg.VATS.Transformation*

Transformationen werden im *VAT*-System verwendet, um zwischen unterschiedlichen Modellformen von Quellen, Transformern oder Senken zu wechseln. Die Verwaltung der Transformationen soll in diesem Paket durchgeführt werden. Hierzu wird die Klasse `TransformationChooser` als statischer Dienst in das System eingefügt. Eine Transformation besteht im System immer aus einer Hintransformation zum Fremdmodell und – damit im Visualisierungspfad auch andere Systemelemente das evtl. veränderte Modell verwenden können – aus einer Rücktransformation. Die Angabe einer Transformation wird mit Hilfe von Delegaten realisiert. Hierdurch wird erreicht, dass eine Methode einer Klasseninstanz oder statische Methode mit einer vorgegebenen Methodensignatur angegeben werden kann. Falls keine Transformation notwendig ist, wird die Methode `RegisterForNoTransformation` zur Registrierung verwendet, wodurch alle instanziierten Systemelemente einen Verweis in der Datenstruktur zur Sicherung der Transformationen aufweisen. Mit Hilfe der Methoden `TransformFromDataModel` und `TransformToDataModel` kann auf die hinterlegten Transformationsdaten zugegriffen werden. Die Methode `TransformToDataModel` führt eine

Transformation des angegeben Fremdmodells in das Datenmodell durch und stellt damit die Möglichkeit der Rücktransformation zur Verfügung. Die Hintransformation wird mit Hilfe der Methode `TransformFromDataModel` durchgeführt.

UniOldenburg.VATS.Modelmanagement.Handling

Die Klassen des `Handling`-Paketes sollen den Zugriff und die Verwendung des Daten- und Systemmodells vereinfachen, indem für häufig benötigte Aktionen, wie beispielsweise das Hinzufügen, Erstellen oder Löschen von Systemelementen, Dienste bereitgestellt werden. Hierdurch wird zusätzlich erreicht, dass das Paket `UniOldenburg.VATS.Modelmanagement.VMTSAccess` von anderen Komponenten unabhängig ist, da die Funktionalität durch das hier betrachtete Paket gekapselt wird. Die Klassen des Paketes sind in der Abbildung 7.12 dargestellt.

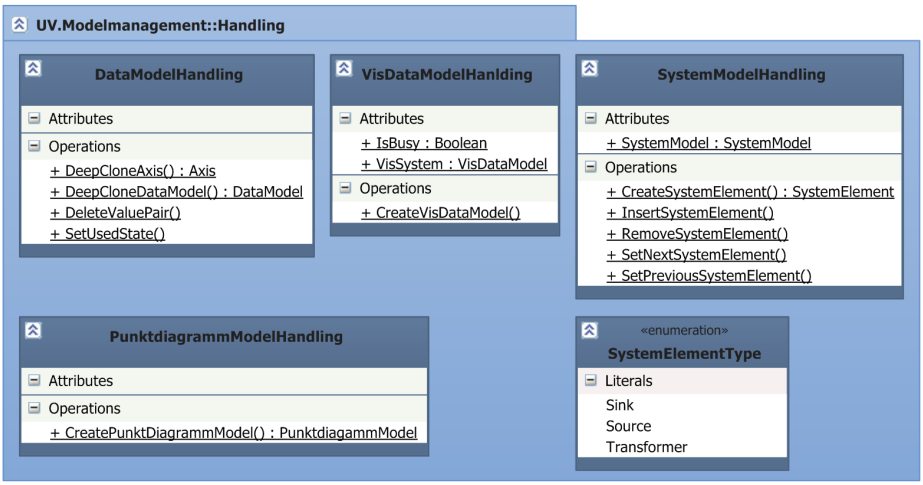


Abbildung 7.12: *Das Paket UniOldenburg.VATS.Modelmanagement.Handling*

Da ein Dienst angeboten werden soll, der nicht zustandsbehaftet ist, bietet es sich an, statische Klassen zu verwenden. Für das Einfügen, Löschen und Erstellen von Systemelementen wird die Klasse `SystemModelHandling` eingefügt. Neben dieser Funktionalität werden der Klasse Methoden zum Einordnen von Systemelementen in den Analysepfad (vgl. Kapitel 6.3) eingefügt. Die Klasse `PunktdiagrammModelHandling` stellt einen Zugriff auf das Punktdiagrammmodell (vgl. Kapitel 2.1.3) zur Verfügung. Die Klassen `VisDataModelHandling` und `DataModelHandling` werden für den Zugriff auf das Datenmodell (vgl. Kapitel 6.4.2) verwendet. Die erst genannte Klasse stellt den Zugriff auf das Modell und die `IsBusy`-Eigenschaft zur Verfügung. Letztere kann verwendet werden, um dem Anwender anzuzeigen, dass das System gerade mit einer Berechnung beschäftigt ist, und deshalb nicht weiterverwendet werden kann. In der statischen Klasse `DataModelHandling` werden Methoden zum Anpassen und Ändern des Datenmodells eingefügt. Im Einzelnen sind dies die Methode `DeepCloneAxis` zum tiefen Klonen von Achsen, die Methode `DeepCloneDataModel` zum tiefen Klonen eines Datenmodels, die Methode `DeleteValuePair` um eine Menge von Wertepaaren zu entfernen und die Methode `SetUsedState` um den Verwendungsstatus einer Achse zu ändern.

7.2.2 Das Paket UniOldenburg.VATS.Systemelement

Innerhalb dieses Paketes werden alle Systemelemente, die zur Analyse verwendet werden, eingefügt. Die Quellen, Transformer oder Senken sollen aus unterschiedlichen Fremdsystemen eingefügt werden können. Diese sind in den entsprechenden Subpaketen eingebunden. Weil der Architekturentwurf nicht durch diese Komponenten beeinflusst wird, werden die Elemente im anschließenden Kapitel 8 zur Implementierung detaillierter betrachtet. Es existiert kein einheitlicher Standard zum Import von Fremdkomponenten. Aus diesem Grund wird eine für das *VAT*-System gültige Beschreibung von Systemkomponenten und deren Eigenschaften entwickelt. Hierzu werden Interfaces verwendet, wodurch eine Strukturvorschrift erreicht wird. Die Schnittstellen sind in der Abbildung 7.13 dargestellt.

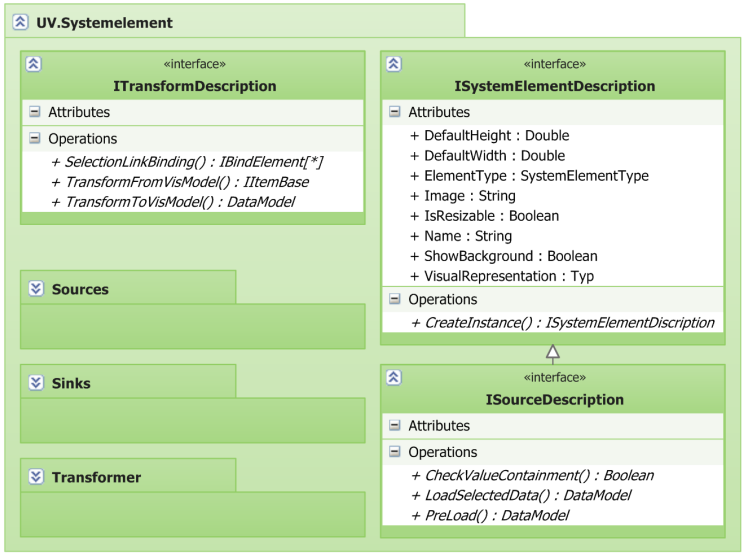


Abbildung 7.13: *Das Paket UniOldenburg.VATS.Systemelement*

Das Interface `ISystemElementDescription` bildet die Basis für eine Beschreibung eines Systemelementes. Mit Hilfe der Attribute `DefaultHeight` und `DefaultWidth` wird die Größe des Elementes bei der ersten Erstellung angegeben. Damit ein Anwender die Größe des Systemelementes an seine Bedürfnisse anpassen kann, besteht die Möglichkeit diese zu ändern. Allerdings exisitieren Analysewerkzeuge, die für diese Operation nicht ausgelegt sind. Aus diesem Grund wird das Attribut `IsResizable` der Beschreibung hinzugefügt. Das `Name`- und `Image`-Attribut dienen zur Identifizierung innerhalb des Menüs. Damit eine Gruppierung innerhalb des Menüs und farbliche Zuordnung im Arbeitsbereich des *VAT*-Systems erfolgen kann, ist es außerdem notwendig, dass bekannt ist, von welchem Typ das Systemelement ist. Diese Eigenschaft wird mit Hilfe des Attributes `ElementType` bekannt gegeben. Als weiteres Merkmal umfasst die Beschreibung die visuelle Repräsentation des Elementes. Diese wird im Arbeitsbereich des *VAT*-Systems nach Instanziierung des Systemelementes dargestellt, um den Anwender die Möglichkeit zu geben, Eigenschaften an dem Modell ändern zu können. Die Verknüpfung zwischen Modell und visueller Repräsentation wird mit Hilfe des `DataContextes` der visuellen Repräsentation erzielt. Diese Beschreibung ist für Transformer und Senken ausreichend. Allerdings ist für Quellen eine erweiterte Beschreibung notwendig, weil Daten dynamisch nachgeladen werden sollen. Die Erweiterung für Quellelemente wird mit Hilfe des Interfaces `ISourceDescription` durchgeführt, welche die Methode `LoadSelectedData` zu diesem Zweck deklariert. Quellen bilden den Ausgangspunkt eines Analysepfades. Aus diesem Grund ist es

notwendig, dass hier Daten vorgeladen werden. Diese umfassen in der Regel ein Datenmodell, welches die Dimensionen und Kennzahlen der Datenquelle beinhaltet. Das Vorladen wird, in Abhängigkeit der Datenhaltung einer Quelle, innerhalb der Methode `PreLoad` implementiert. Weiterhin kann es notwendig sein, dass auf Grundlage von unterschiedlichen Modellen, Transformationen angeben werden müssen. Zu diesem Zweck wird das Interface `ITransformDescription` eingefügt, wodurch die benötigten Transformationen zwischen Datenmodell und Fremdmodell deklariert werden. Zusätzlich wird die Methode `SelectionLinkBinding` eingefügt, welche verwendet wird, um Selektionsänderungen von anderen Datenmodellen auf das transformierte Modell abzubilden.

7.2.3 Das Paket UniOldenburg.VATS.SystemelementManager

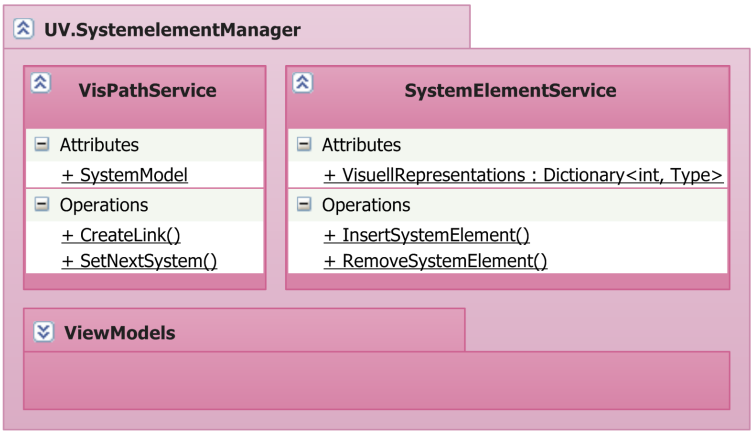


Abbildung 7.14: *Das Paket UniOldenburg.VATS.SystemelementManager*

Die Komponente Systemelementmanager soll die Trennung zwischen *GUI* und Interaktionslogik ermöglichen, wie sie in modernen Systemen angestrebt ist. Die Aufgabe des `Systemelement-Manager`-Paketes besteht in der Abbildung von Interaktionen des Anwenders auf Operationen des Modells, welche von der Modellverwaltungsschicht zur Verfügung gestellt werden. Des Weiteren sollen die geladenen und zur Verfügung stehenden Systemelemente zur Darstellung in der *GUI* bereitgestellt werden. Diese Funktionalität wird im Subpaket `ViewModels` eingefügt. Zunächst wird auf die Abbildung von Interaktionen eingegangen. Die hierzu nötigen Klassen sind in der Abbildung 7.14 dargestellt.

Das Paket enthält die Klassen `VisPathService` und `SystemElementService`. Beide werden als zustandslose Services entworfen, weshalb die Klassen statisch ausgeführt werden. Die Klasse `VisPathService` kapselt den Zugriff auf den Analysepfad (vgl. Kapitel 6.3) des *VAT*-Systems und stellt hierzu Methoden bereit, um Systemelemente in den Pfad einzufügen und Selektionsverknüpfungen zwischen zwei Systemelementen zu erstellen. Die Klasse `SystemelementService` wird für Interaktionsabbildungen, die ein Systemelement beeinflussen, erstellt. Die Methoden `InsertSystemElement` instanziiert ein Systemelement anhand der zugewiesenen Beschreibung (vgl. Kapitel 7.2.2). Hierzu wird zunächst die graphische Repräsentation im Dictionary `VisuellRepresentations` hinterlegt und anschließend das Systemelement instanziiert. Die Daten des Dictionaries werden von der *GUI*-Komponente verwendet, um die visuelle Repräsentation darzustellen. Außerdem stellt

die Klasse die Methode `RemoveSystemElement` zu Verfügung, um eine Entfernen-Interaktion abzubilden.

UniOldenburg.VATS.SystemelementManager.ViewModels

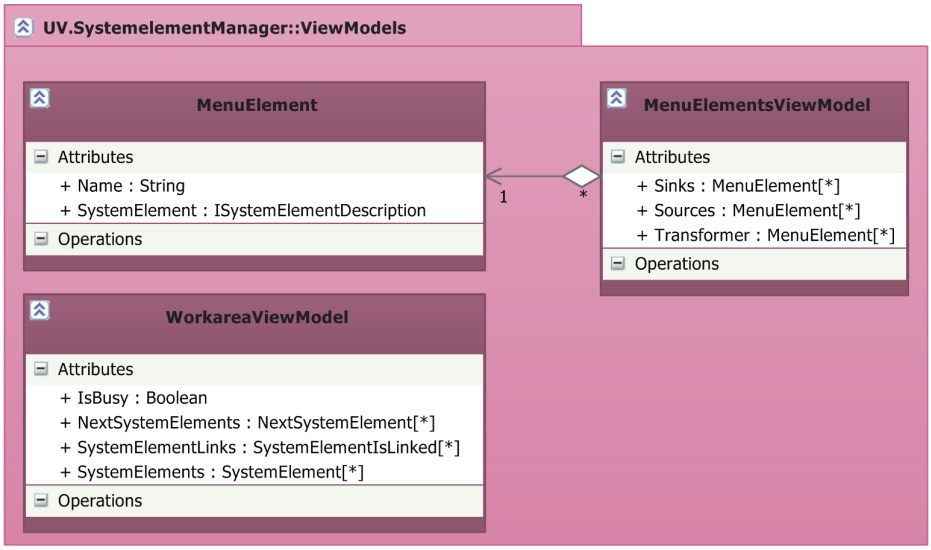


Abbildung 7.15: *Das Paket UniOldenburg.VATS.SystemelementManager.ViewModels*

Die Information über die gerade geladenen Systemelemente und die für eine Analyse verwendbaren Systemelemente werden in diesem Paket als *WPF*-View-Modelle eingefügt. Aus diesem Grund werden für alle Eigenschaften DependencyPropertys verwendet, die eine Benachrichtigungsfunktion bei Änderungen der Eigenschaft zur Verfügung stellen, die von der *GUI* aufgegriffen werden. Die Klassen des Paketes sind in der Abbildung 7.15 dargestellt. Im Wesentlichen werden zwei View-Modelle entwickelt. Die Klasse `WorkareaViewModel` wird zur Verwendung innerhalb der Arbeitsfläche der *GUI* und die Klasse `MenuElementsViewModel` für das Menü entworfen. Das Modell der Arbeitsfläche stellt die geladenen Systemelemente und die Elemente des visuellen Pfades innerhalb einer Liste zur Verfügung. Diese können dann von der *GUI* ausgelesen und einzeln dargestellt werden. Für das Modell des Menüs werden auf Basis der Systemelementbeschreibungen (vgl. Kapitel 7.2.2) Listen der zur Verfügung stehenden Quellen, Transformer und Senken erstellt. Hierbei werden die Elementbeschreibungen von der Klasse `MenuElement` gekapselt.

7.2.4 Das Paket UniOldenburg.VATS.SystemGUI

Die Klassen zur Darstellung der Benutzungsoberfläche des *VAT*-Systems werden in diesem Paket eingefügt. Die Klasse `MainWindow` stellt diese dar. Die Teilung der *GUI* in ein Menü und eine Arbeitsfläche, wie sie aus dem *GUI*-Entwurf hervorgeht (vgl. Kapitel 7.1.2), wird auf die Einteilung der Klassen reflektiert. Das Paket `Menu` enthält die Klassen, welche für den Aufbau und die Interaktion mit dem Menü zuständig sind, sowie das Paket `Workarea` die Klassen für den Arbeitsbereich. Das Klassendiagramm des Paketes kann der Abbildung 7.16 entnommen werden.

Das Menü wird in der Klasse `HeaderMenu` eingefügt. Es stellt gruppiert die verfügbaren Systemelemente des *VAT*-Systems dar. Hierzu wird auf die Klasse `MenuElementsViewModel` der

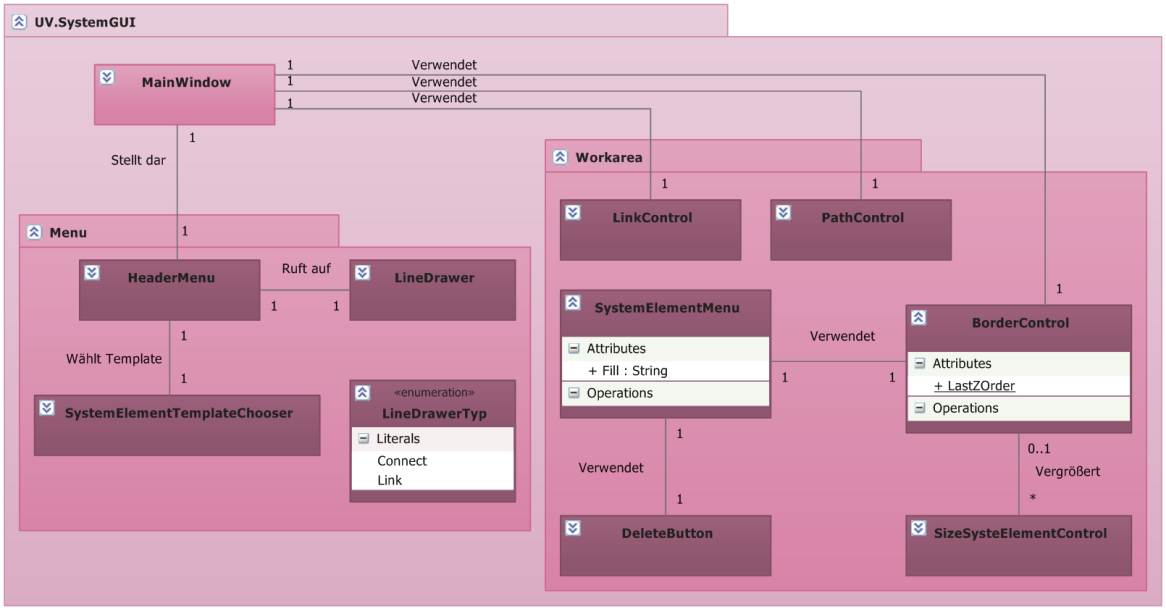


Abbildung 7.16: *Das Paket UniOldenburg.VATS.SystemGUI*

Systemelementverwaltungs-Komponente zugegriffen. Sofern ein Menüelement ausgewählt wurde (und damit die Erstellung eines Systemelementes) wird das Element mit Hilfe des Systemelementservices instanziiert und die visuelle Repräsentation des Systemelementes in dem Arbeitsbereich dargestellt. Die Darstellung des Elementes wird mit Hilfe der Klasse `SystemElementTemplateChooser` durchgeführt. Dieser ermöglicht es, zur Laufzeit ein Datentemplate auf Grundlage des angegeben Typs in der Beschreibung des Systemelementes (vgl. Kapitel 7.2.2) zuerstellen. Um zwei Systemelemente miteinander zu verbinden und hierdurch den Analysepfad (vgl. Kapitel 6.3) anzupassen, wird die Klasse `LineDrawer` eingefügt. Die Art der Verbindung der Elemente, Selektionsverknüpfung oder Pfadverbindung, wird mit Hilfe des Aufzählungstyps `LineDrawerTyp` angegeben. Im Paket `Workarea` werden die für die Darstellung des Arbeitsbereiches nötigen Klassen eingefügt. Hierzu zählen die Klassen `LinkControl` und `PathControl` um den Analysepfad (Verknüpfungen und Verbindungen) darzustellen. Damit Systemelemente auch wieder aus dem Analysepfad entfernt werden können, wird an jedes Element ein Menü angefügt. Dieses wird mit Hilfe der Klasse `BorderControl` erreicht, die im visuellen Baum von *WPF* eine Ebene tiefer als die graphische Repräsentation des Systemelementes eingefügt wird. Mit Hilfe der Klasse wird das Menü `SystemElementMenu` unterhalb jedes Systemelementes dargestellt. Dem Menü wird ein `DeleteButton` zugeordnet, um das entsprechende Element zu entfernen. Außerdem kann es notwendig sein, die Größe eines Systemelementes anzupassen. Aus diesem Grund wird die Klasse `BorderControl` um die Funktionalität erweitert, die Größe anzupassen. Diese Eigenschaft wird innerhalb der Klasse `SizeSystemElementControl` eingefügt.

7.2.5 Wichtige Anwendungsfälle

Nachdem im vorherigen Kapitel die Klassen und Pakete und deren Methoden einzeln dargestellt wurden, wird in diesem Kapitel mit Hilfe von Beispielen auf das Zusammenspiel zwischen diesen ein-

gegangen. Als graphisches Hilfsmittel werden hierzu *UML*-Sequenzdiagramme verwendet. Zunächst wird auf die Erstellung eines Systemelementes durch Auswahl des Elementes im Menü eingegangen. Dieses Vorgehen wird im Anschluss um die Erstellung einer Verbindung zwischen zwei Systemelementen ergänzt und um die Zuweisung von Änderungen im Datenmodell. Damit wird ein einfacher Analysefall dargestellt, der zum Schluss um die Verknüpfung zwischen zwei Visualisierungen ergänzt wird.

Erstellung eines Systemelementes

Beispielhaft für die Auswahl des Anwenders zur Verwendung eines Systemelementes wird im Sequenzdiagramm der Abbildung 7.17 die Methodensequenz dargestellt, die ausgeführt wird, wenn der Benutzer ein Senkenelement im Menü der *GUI* auswählt. Diese wird im Folgenden beschrieben.

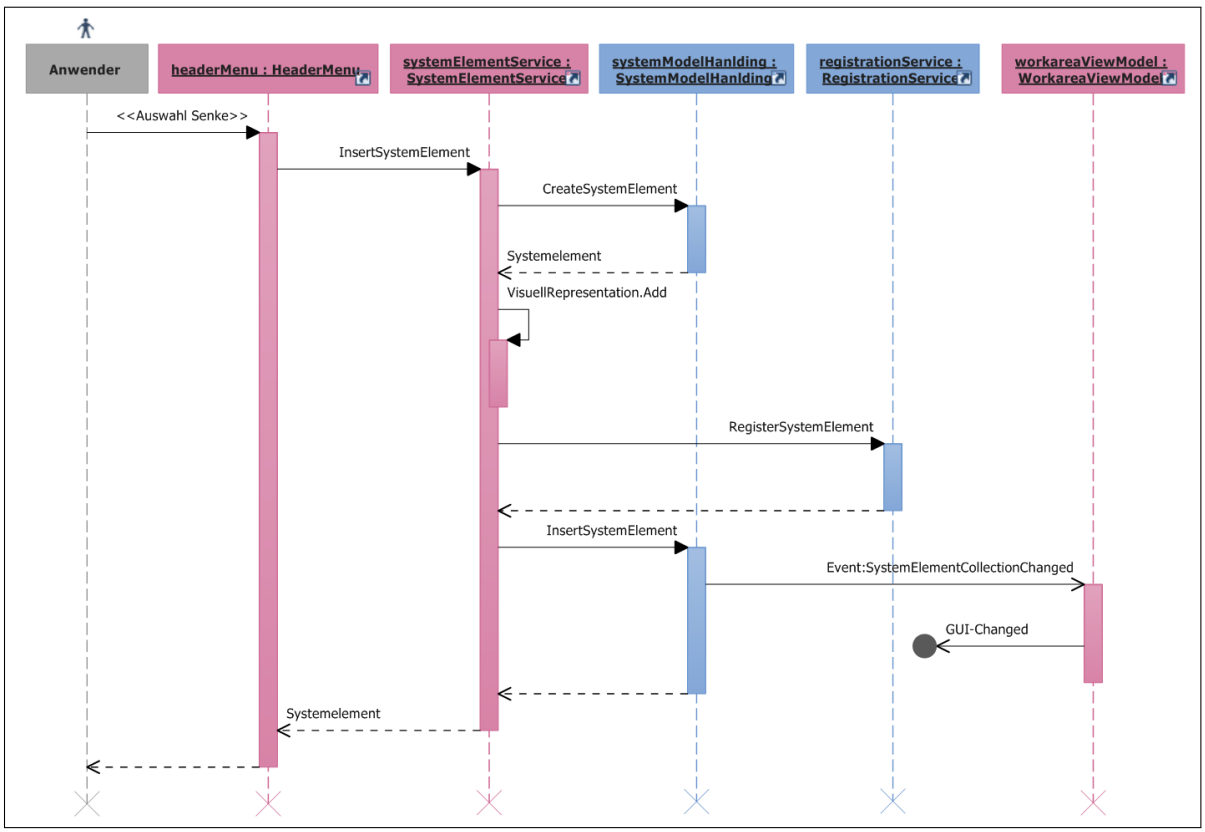


Abbildung 7.17: *Sequenzdiagramm zur Erstellung einer Senke*

Das Menü des *VAT*-Systems wird von der Klasse `HeaderMenu` bereitgestellt. Ein Anwender wählt in diesem Fall den Menüpunkt zur Erstellung einer Senke aus. Hieraufhin werden die Beschreibungsdaten für das Systemelement ausgelesen und mit Hilfe des Methodenaufrufes `InsertSystemElement` an den `SystemElementService` weitergereicht. Dieser Service wird hierdurch angewiesen, das Systemelement anhand der Beschreibung zu erstellen und in das Visualisierungssystem einzufügen. Hierzu wird zunächst die `CreateSystemElement`-Methode des `SystemModelHandling`-Dienstes aufgerufen, um ein konkretes Systemelement zu erstellen. Anschließend werden die Eigenschaften des instanziierten Systemelementes anhand der übergebenen Beschreibung gesetzt. Ein Schritt hierbei ist die visuelle Repräsentation aus der Beschreibung in das `VisuellRepresentati-`

on-Dictionary zu übernehmen. Diese Daten werden zur Darstellung des Systemelementes verwendet. Anschließend werden die evtl. nötigen Transformationen der Senke in der Datenverwaltungsschicht registriert. Der Dienst `RegistrationService` stellt hierzu die Schnittstelle `RegisterSystemElement` zur Verfügung und bildet die Registrierung auf die Unterklassen der Schicht ab, wodurch diese im weiteren Analyseverlauf zur Verfügung stehen. Der letzte Schritt bei der Erstellung des Systemelementes besteht darin, dass Element im *VAT*-System einzufügen. Da bei der Erstellung des Elementes kein Vor- bzw. Nachfolgeelement angegeben werden kann, ist das Einfügen mit Hilfe der Methode `InsertSystemElement` ausreichend. Durch diese Aktion wird das Event `SystemElementCollectionChanged` gefeuert, wodurch das View-Modell `WorkareaViewModel` benachrichtigt wird, dass ein neues Systemelement eingefügt wurde. Das Modell übernimmt die Änderung und benachrichtigt die *GUI*, woraufhin eine Neuzeichnung der *GUI* durch den *GUI*-Thread veranlasst wird. Anschließend steht dem Anwender das Element für Analysezwecke zur Verfügung.

Verbindung von zwei Systemelementen

Das Benutzungskonzept des *VAT*-Systems sieht vor, dass Systemelemente zu einem Analysepfad zusammengeführt werden und hierdurch Abhängigkeiten entstehen (vgl. Kapitel 6.3). Angenommen es werden mit Hilfe der im vorigen Kapitel dargelegten Schritte eine Quelle zur Bereitstellung einer Datenmenge und eine Senke für die Anzeige der Datenmenge erstellt, dafür ist die Erstellung einer Verbindung zwischen diesen notwendig, um die Abhängigkeit zu erstellen. Diese Interaktion des Benutzers wird auf die Änderung des Analysepfades abgebildet. Im weiteren Verlauf wird an dem Beispiel dargelegt, welche Klassen und Methoden zur Erstellung einer Verbindung zwischen Systemelementen verwendet werden. Das Sequenzdiagramm in der Abbildung 7.18 stellt diesen Zusammenhang graphisch dar.

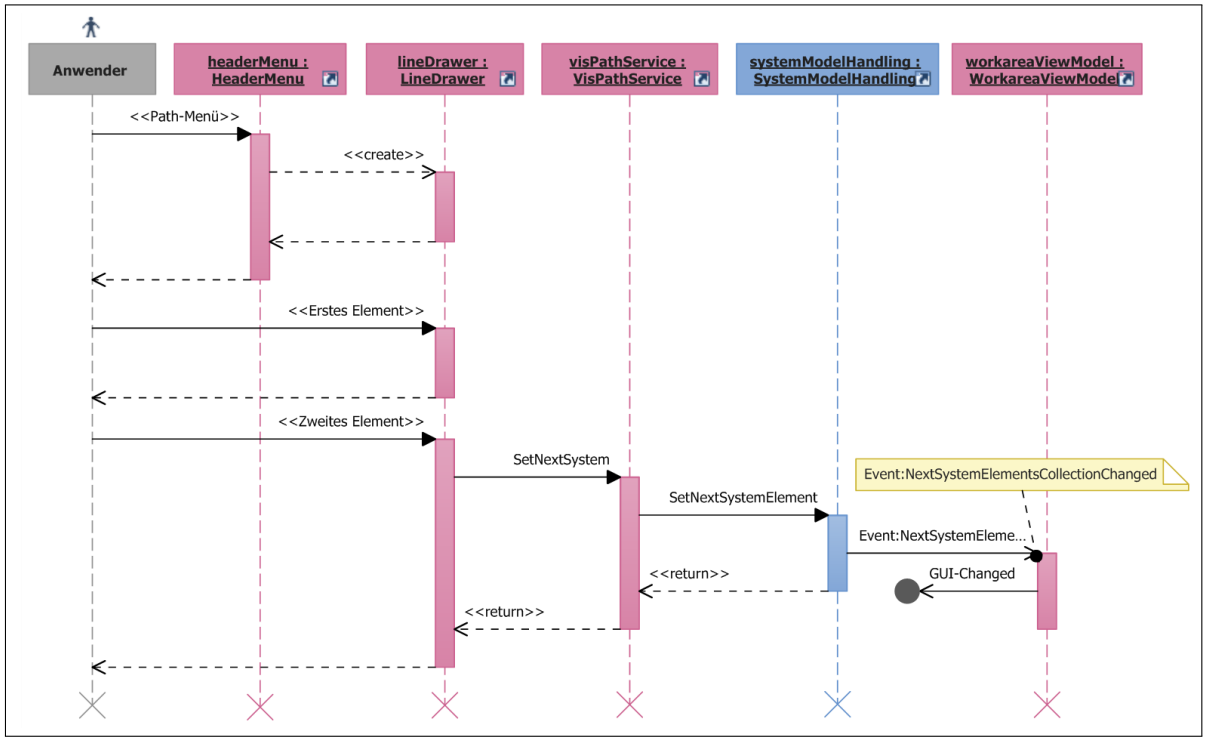


Abbildung 7.18: *Sequenzdiagramm zur Erstellung eines Pfades*

Der Anwender wählt zur Erstellung eines Pfades den Menüpunkt aus dem `HeaderMenu` aus. Hierdurch wird das Linienzeichenwerkzeug `LineDrawer` in den visuellen Baum von *WPF* eingefügt und steht dem Anwender zur Verfügung. Welche Linienart (Selektionsverknüpfung oder Verbindung von Systemelementen) gezeichnet werden soll, wird dem Werkzeug im Konstruktor übergeben. Anschließend wählt der Anwender nacheinander die Systemelemente aus, die miteinander verbunden werden sollen. Diese werden im `LineDrawer` registriert und sofern zwei Systemelemente ausgewählt wurden, wird die Verbindung erstellt. Hierzu wird der `VisPathService` verwendet, der die Methode `SetNextSystem` zur Verfügung stellt. Als Übergabeparameter werden die zwei ausgewählten Systemelemente übergeben. Die Methode kann direkt auf die Modellverwaltungsschicht abgebildet werden. Hierzu wird in der Klasse `SystemModelHandling` die Methode `SetNextSystemElement` aufgerufen. In dieser wird die Verbindung zwischen den beiden Systemelementen erstellt und falls der Erstausgewählten bereits ein Modell zugewiesen wurde, wird dieses dem zweiten Systemelement und evtl. nachfolgenden Elementen zugewiesen. Durch das Einfügen eines Pfadelementes wird die Klasse `Workarea`, die das View-Modell des Arbeitsbereiches bildet, über die Änderung informiert. Das Modell übernimmt die Änderungen und benachrichtigt die *GUI* über diese. Hierdurch wird nicht nur die Verknüpfung auf Datenebene durchgeführt, sondern dem Anwender wird gleichzeitig der Zusammenhang zwischen den Systemelementen im Arbeitsbereich visuell dargestellt.

Zuweisung eines Datenmodells

Durch Interaktionen des Anwenders mit einem Systemelement werden evtl. Änderungen an dem darunterliegenden Datenmodell durchgeführt, wie beispielsweise die Auswahl einer Attributgruppe eines Quellelementes zur weiteren Verwendung im Analysekontext. Die Änderung muss im Analysepfad des *VAT*-Systems weitergereicht werden. Hierzu werden im System unterschiedliche Methoden und Klassen verwendet. Am Beispiel der Auswahl einer Attributmenge einer Datenquelle zur Anzeige in einer Senke wird dieser Zusammenhang im weiteren Verlauf dargestellt. Mit Hilfe des Sequenzdiagramms (vgl. Abbildung 7.19) wird der Ablauf beschrieben.

Der Anwender ändert die Auswahl von Attributen für einen Analysekontext. Dieses wird direkt in der Visualisierung des Systemelementes durchgeführt und auf das Datenmodell des Quellsystemelementes abgebildet. Nachdem die Änderung durchgeführt wurde, wird von der Klasse `Source`, welche eine Instanz von Systemelement ist und die Quelle in dem betrachteten Fall bildet, die Methode `SetModelToNextElements` der Klasse `VisPathControl` aufgerufen. Durch die Operation wird die Modellverwaltungsschicht angewiesen, dass Modell der Quelle an die nachfolgenden Systemelemente, in diesem Fall ein Senkenelement, weiterzureichen. Hierzu wird zunächst eine Rücktransformation des Quelldatenmodells durchgeführt. Die Klasse `TransformationChooser` stellt zu diesem Zweck die Schnittstelle `TransformToDataModel` zur Verfügung. Innerhalb der Methode wird auf die registrierten Transformationen, die mit Hilfe der Systemelementbeschreibung (vgl. Kapitel 7.2.2) hinterlegt wurden, zurückgegriffen. Da im Anwendungsfall, die Quelle direkt das Datenmodell verwendet, ist keine Transformation notwendig. Im Anschluss werden die Daten, die vom Anwender mit Hilfe der Attributsauswahl ausgewählt wurden, nachgeladen. Zu diesem Zweck wird die Methode `LoadUsedData` des `DataLoader`-Dienstes aufgerufen. Dieser greift auf die in der Beschreibung des Quellelementes angegebene dynamische Nachlademethode `LoadSelectedData` zurück und gibt anschließend das Datenmodell an die Klasse `VisPathControl` mit der ausgewählten Datenmenge zurück. Die Daten stehen nun zur Verfügung. Im nächsten Schritt werden diese den nachfolgenden Elementen zugewiesen. Diese können allerdings ein anderes Datenschema aufweisen, weshalb der Dienst `TransformationChooser` zu einer Transformation in das Fremdmodell angewiesen

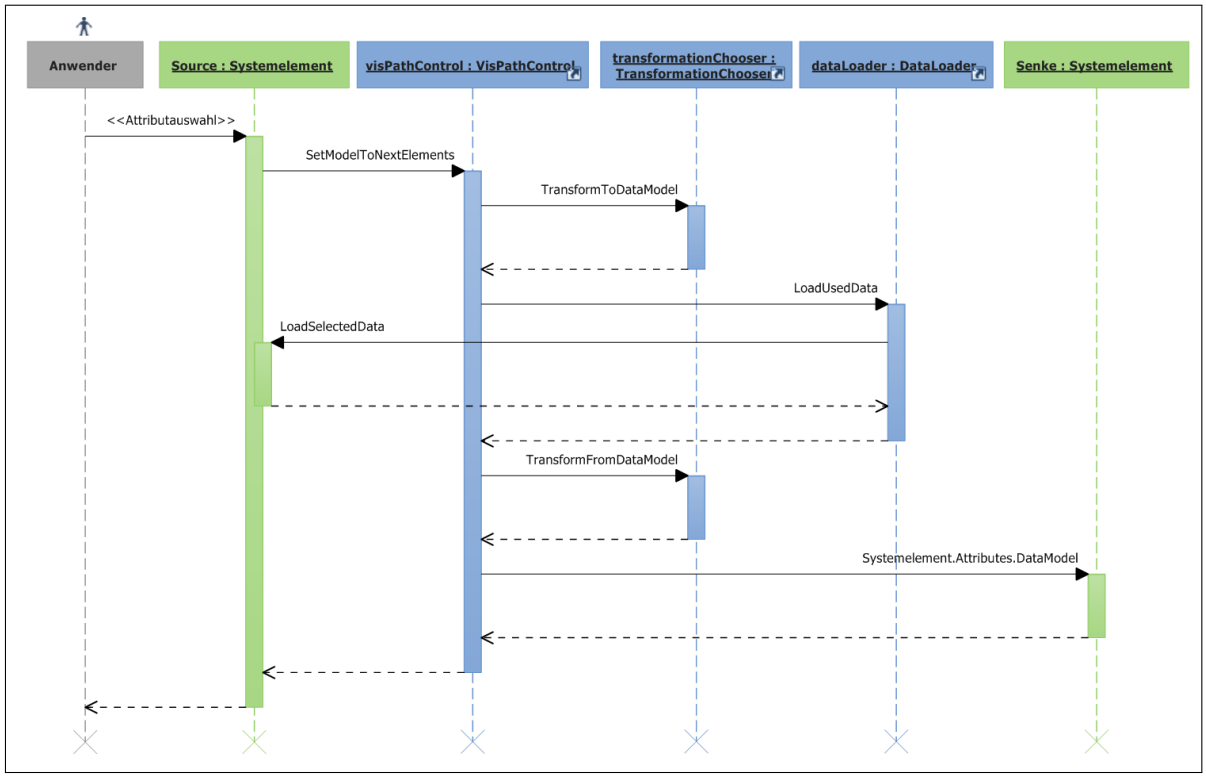


Abbildung 7.19: *Sequenzdiagramm zum Zuweisen eines Datenmodells*

wird. Im Anschluss wird dieses Modell für das `Senken`-Systemelement gesetzt, indem das Attribut `DataModel` zugewiesen wird.

Selektionsverknüpfung

Nachdem ein Anwender mehrere Systemelemente erstellt hat, sollen diese evtl. miteinander verknüpft werden. Hierzu wird die Selektionsverknüpfungskomponente entwickelt. Angenommen ein Anwender möchte zwei Senken miteinander verknüpfen, wird der folgende Ablauf im *VAT*-System durchgeführt (vgl. Abbildung 7.20 und 7.21).

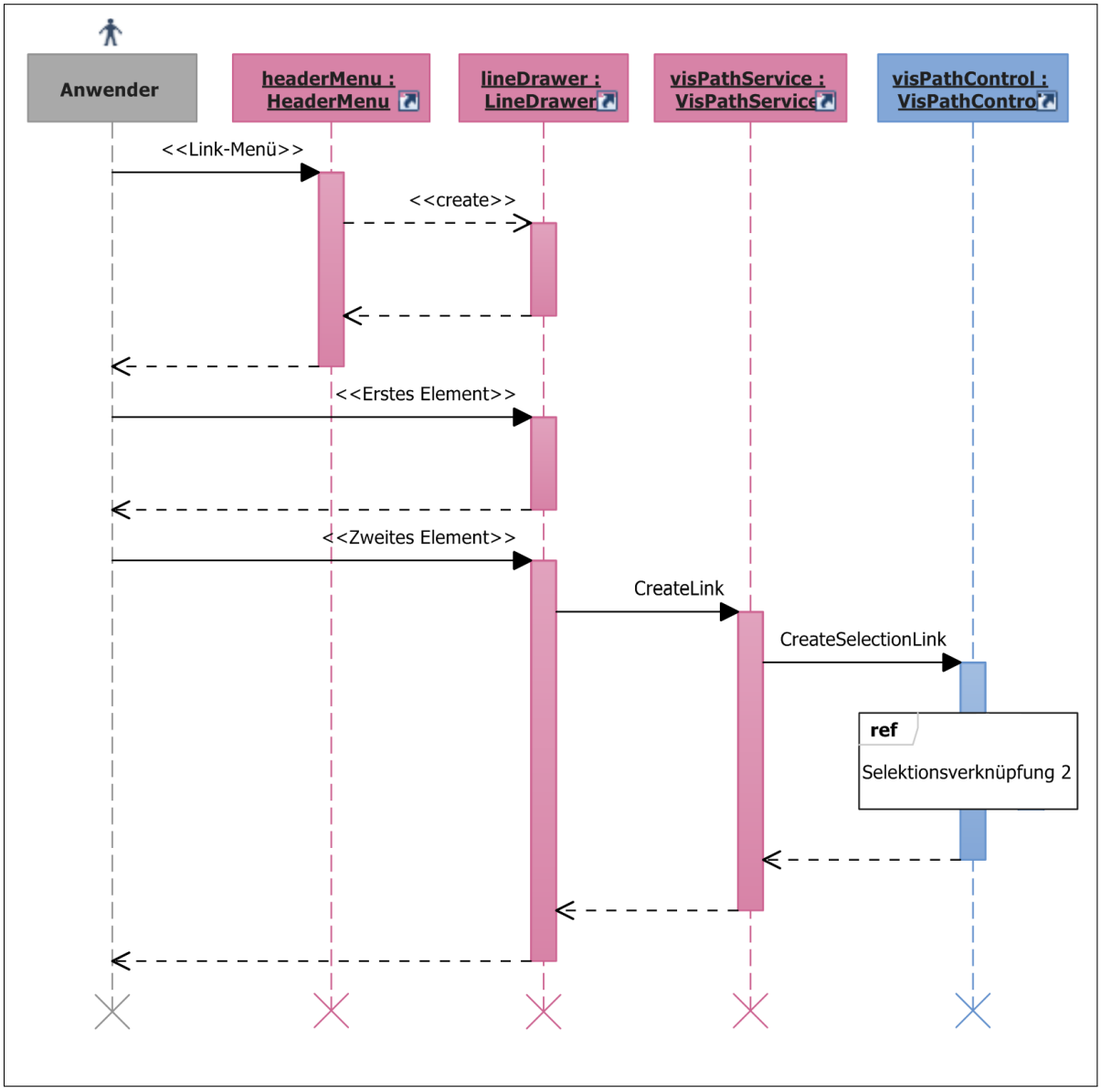


Abbildung 7.20: *Sequenzdiagramm zur Erstellung einer Selektionsverknüpfung (1)*

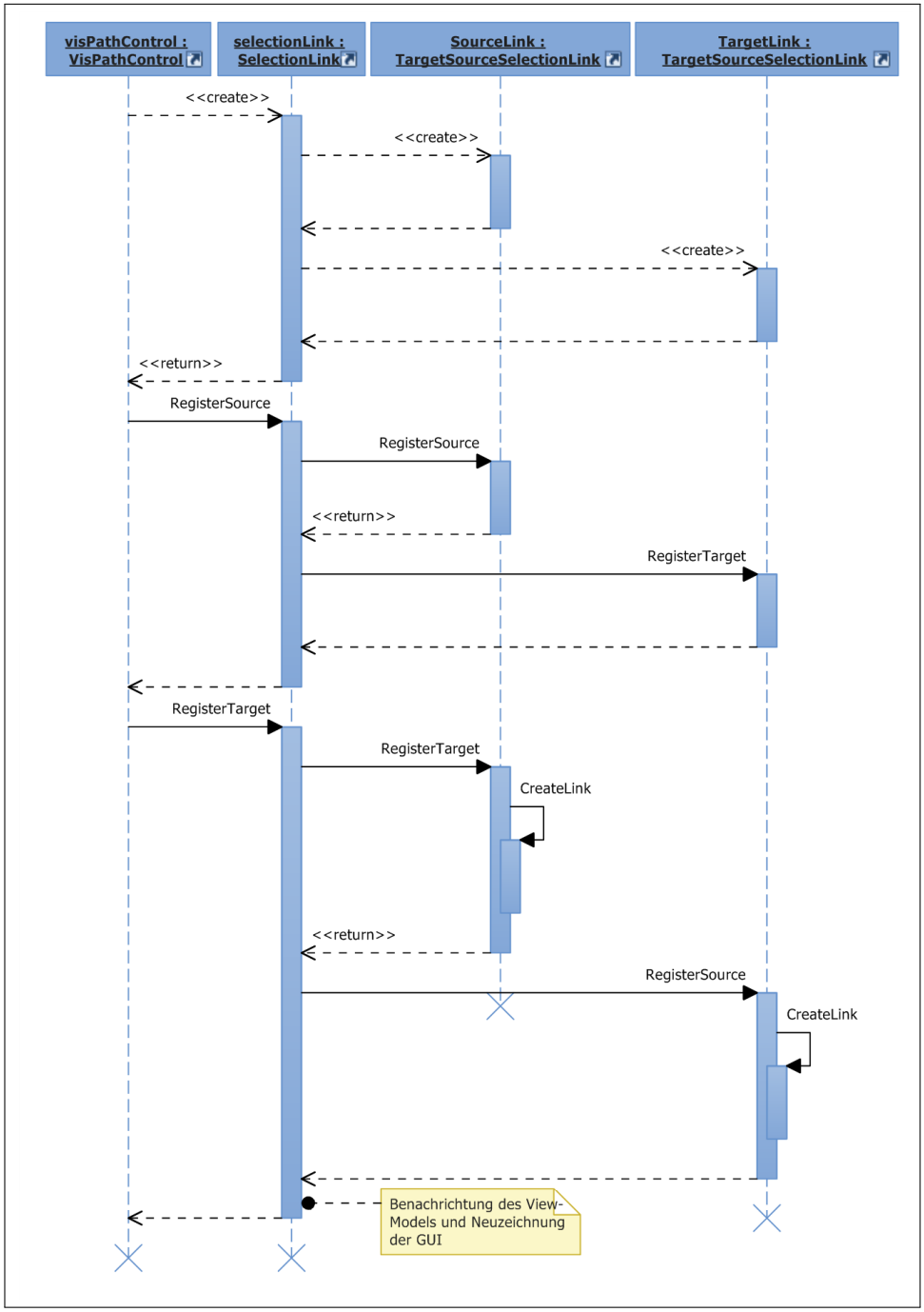


Abbildung 7.21: *Sequenzdiagramm zur Erstellung einer Selektionsverknüpfung (2)*

Für den Fall der Selektionsverknüpfung wird in Anlehnung an die Erstellung einer Verbindung zwischen zwei Systemelementen ein ähnliches Verfahren angewandt. Der Anwender wählt zunächst im Menü das Element zur Verknüpfung von Systemelementen aus. Die Klasse `LineDrawer` wird wiederum zur Erstellung der Verbindung verwendet. Allerdings wird diese mit Hilfe eines Parameters dazu angewiesen, keine Verbindung sondern eine Verknüpfung zwischen den ausgewählten Systemelementen durchzuführen. Aus diesem Grund wird nach Auswahl zweier Systemelemente durch den Anwender die Methode `CreateLink` der Klasse `VisPathService` aufgerufen. Diese bildet die Erstellung der Verknüpfung auf die Methode `CreateSelectionLink` der Modellverwaltungsschicht ab. Innerhalb der Methode wird zunächst eine Instanz der Klasse `SelectionLink` erzeugt. Da eine Selektionsverknüpfung aus zwei `TargetSourceLink`-Klassen zusammengesetzt wird (vgl. Kapitel 7.2.1), werden diese Instanzen bei der Erstellung der `SelectionLink`-Instanz ebenfalls erstellt. Nachdem die Strukturen instanziiert wurden, werden die ausgewählten Systemelemente in der Klasse `SelectionLink` registriert. Zunächst wird mit Hilfe der Methode `RegisterSource` das vom Anwender zuerst ausgewählte Systemelement gesetzt. Diese Operation wird im Objekt `SourceLink` auf das Setzen der Quelle und im Objekt `TargetLink` auf das Zuweisen des Zieles abgebildet. Im Anschluss wird von der Klasse `VisPathControl` das vom Anwender zuletzt ausgewählte Systemelement als Ziel der Verknüpfung registriert. Dadurch werden die Quellen und Ziele in den Objekten der Klasse `TargetSourceLink` gesetzt. Um jedoch eine bidirektionale Verknüpfung zu erreichen werden die Zuweisungen vertauscht. Dies bedeutet, für das Objekt `SourceLink` wird das Systemelement als Ziel und für das Objekt `TargetLink` als Quelle eingefügt. Da in beiden Objekten der Klasse `TargetSourceLink` nun Quell- und Zielelement vorhanden sind, kann mit Hilfe der Methode `CreateLink` die eigentliche Verknüpfung der Elemente erstellt werden. Die Benachrichtigung der *GUI*, mit Hilfe des `Workarea`-View-Modells erfolgt vergleichbar mit den vorherigen.

7.3 Zusammenfassung

Innerhalb dieses Abschnittes ist die Architektur des *VAT*-Systems vorgestellt worden. Wichtigster Aspekt bei der Entwicklung des Systems ist die Erweiterbarkeit um weitere Systemkomponenten. Aus diesem Grund sind die Systemkomponenten in eine eigene Komponente ausgegliedert und können vom Analysten mit Hilfe der *GUI* und des `SystemelementManagers` zur Analyse verwendet werden. Weiterer wichtiger Baustein der Architektur ist *VMTS*, welches die Verwendung der im Kapitel 6.4 vorgestellten Modelle, ermöglicht. Die Einbindung des Datenmodells als Abstraktion zwischen unterschiedlichen Visualisierungsmodellen ermöglicht eine Selektionsverknüpfung zwischen Systemelementen. Damit die Verknüpfung und die Abhängigkeit zwischen Analyseelementen dargestellt werden kann, ist im weiteren Verlauf eine *GUI* entwickelt worden. Sie besteht aus den Bereichen für ein Menü und einer Arbeitsfläche. Die Arbeitsfläche stellt den Visualisierungspfad und die verwendeten Analysewerkzeuge dar, wie auch eine evtl. Verknüpfung zwischen Systemelementen. Somit kann ein Analyst direkten Einfluss auf den Visualisierungspfad und Systemelemente nehmen. Die *GUI* ist im System von den Modellierungsebenen entkoppelt, wie aus der Darstellung der Paketstruktur entnommen werden kann. In der Paketstruktur ist des Weiteren ersichtlich, welche Komponenten anderer Systeme in das *VAT*-System eingebunden werden. Der Namespace `Systemkomponente` hängt im Vergleich mit den anderen Paketen von vielen Komponenten ab. Diese Eigenschaft ist darin begründet, dass die eingebundenen Systemelemente in diesem Paket gekapselt sind.

Im weiteren Verlauf ist dieser Grobentwurf weiter verfeinert worden, wodurch Klassen und Methoden identifiziert wurden. Die Klassen des Paketes `Modelmanagement` stellen die Modellverwaltungsschicht dar. Innerhalb des Paketes sind verschiedene Dienste eingefügt worden, die sofern sie zustandslos sind, statisch ausgeführt sind. Diese Dienste werden vom `SystemElementManager` verwendet um View-Modelle zur Verfügung zu stellen, die wiederum in der *GUI* dargestellt werden. Des Weiteren ist eine Beschreibung für Systemelemente eingeführt worden, die Daten zur visuellen Repräsentation und zu evtl. benötigten Transformationen enthält. Die Beschreibung wird als Interface entworfen, wodurch ein Entwickler von Systemelementen auf Grundlage der fehlenden Mehrfachvererbung im Vergleich zu einer abstrakten Klasse nicht eingeschränkt wird. Abgeschlossen wurde der Abschnitt mit der Beschreibung von wichtigen Anwendungsfällen, um den Zusammenhang zwischen den unterschiedlichen Paketen darzustellen. Hierzu ist ein einfacher Analysefall ausgehend von der Instanziierung eines Systemelementes bis zu der Erstellung einer Selektionsverknüpfung mit Hilfe von *UML*-Sequenzdiagrammen beschrieben worden. Auf Grundlage dieses Entwurfes wird das *VAT*-System, als prototypische Implementierung der Abstraktionschicht implementiert. Besondere Aspekte der Umsetzung werden im anschließenden Abschnitt vorgestellt.

8 Implementierungsaspekte

Aufbauend auf den Feinentwürfen werden in diesem Kapitel ausgesuchte Implementierungseigenschaften der entworfenen Klassen und Pakete dargelegt. Das *VAT*-System baut im Wesentlichen auf das Systemmodell und Datenmodell auf (vgl. Abschnitt 6). Diese werden mit Hilfe von *VMTS* erstellt. Damit die Modelle effektiv verwendet werden, sind einige Änderungen notwendig, die zunächst beschrieben werden. Im Anschluss wird aufbauend auf diesen Änderungen die Implementierung der Selektionsverknüpfung dargelegt. Abgeschlossen wird das Kapitel durch die Beschreibung der eingefügten Systemelemente, um alle Anforderungen (vgl. Abschnitt 6.2) an das *VAT*-System zu erfüllen.

8.1 Änderung am VMTS-Quellcodegenerator

Im *VAT*-System werden die Analysedaten in der Regel aus *OLAP*-Datenbanken geladen. Diese Operation benötigt je nach Datenmenge eine unterschiedliche Zeitspanne. Außerdem können Berechnungen, wie zum Beispiel von Filtern, die Reaktion des Systems verlangsamen. Damit der Anwender über evtl. laufende Berechnungen oder das Laden von Daten informiert wird, ist es notwendig, dass diese zeitaufwendigen Operationen nicht im *WPF*-Thread durchgeführt werden, da dieser sonst blockiert werden würde und eine Ladeanzeige nicht möglich ist. Im Systemmodell (vgl. Kapitel 6.4.1) wird diese Eigenschaft durch das Attribut `IsBusy` des Elementes `VisSystem` abgebildet. Das Attribut kann von unterschiedlichen Komponenten geändert werden, wodurch es notwendig wird, eine Möglichkeit der Benachrichtigung über die Änderung des Attributes bereitzustellen.

Das verwendete *DSL*-Tool generiert aus der graphischen Beschreibung des Systemmodells ein C#-Projekt, bestehend aus Interfaces, welche das Modell abbilden, und Klassen als Realisierungen dieser Schnittstellen. Die Klassen werden zur Laufzeit von *VMTS* geladen. Zur Verwendung stehen die Realisierungsmodelle „Perf“, „UI“ und „NonUI“ zur Verfügung. Die Modelle bieten auf Grundlage der verwendeten Schnittstellen die gleiche Methoden- und Eigenschaftenvielfalt, allerdings ist die interne Implementierung unterschiedlich. Das „UI“-Modell bietet im Gegensatz zum „Perf“-Modell zusätzlich die automatische Benachrichtigung von Änderungen an einem Modellelement. Diese Eigenschaft wird mit Hilfe von DependecyPropertys und ObservableCollections erreicht. Im „Perf“-Modell werden stattdessen Propertys und Listen verwendet, die allerdings keine Benachrichtigungsmöglichkeiten über Änderungen an den Eigenschaften bieten. Allerdings können „UI“-Modellelemente, die in einem Thread erstellt werden, nicht effektiv in einem anderen Thread verwendet werden. Dieses ist in der Realisierung mit Hilfe von DependencyPropertys begründet, die nicht performant Thread-übergreifend verwendet werden können. Zudem können „UI“-Modellelemente im Modellrepository gespeichert werden. Das „NonUI“-Modell ist eine Erweiterung des „Perf“-Modells. Es ermöglicht zusätzlich die Speicherung von Modellelementen im Modellrepository. Allerdings bietet es, ebneso wie das „Perf“-Modell, keine Benachrichtigungsmöglichkeiten. Somit kann keines der aufgeführten Realisierungsmodelle in das *VAT*-System eingefügt werden. Allerdings besteht die Möglichkeit, die für die Generierung des Quellcodes verwendeten Vorlagen anzupassen. Diese Möglichkeit von *VMTS* wird mit dem Ziel genutzt Benachrichtigungen von Änderungen an Modellelementen zu ermöglichen, ohne auf DependencyPropertys zurückzugreifen. In C# wird für die Bekanntgabe von Änderungen an Properties die Schnittstelle `INotifyPropertyChanged` implementiert. Diese deklariert das Event `PropertyChanged`, das bei Änderung einer Eigenschaft gefeuert werden muss. Im weiteren Verlauf werden die nötigen Änderungen an den Quellcode-Vorlagen dargestellt, um das Interface zu implementieren.

Als Grundlage wird das „NonUI“-Modell verwendet, da dieses eine einfachere Anpassung ermöglicht und die Speicherung unterstützt. Zunächst wird anhand des Modellelementes `SystemElement` des Systemmodells dargestellt, welche Klassen und Schnittstellen vom Quellcodegenerator erstellt werden, um darauf aufbauend die erforderlichen Änderungen zu ermitteln.

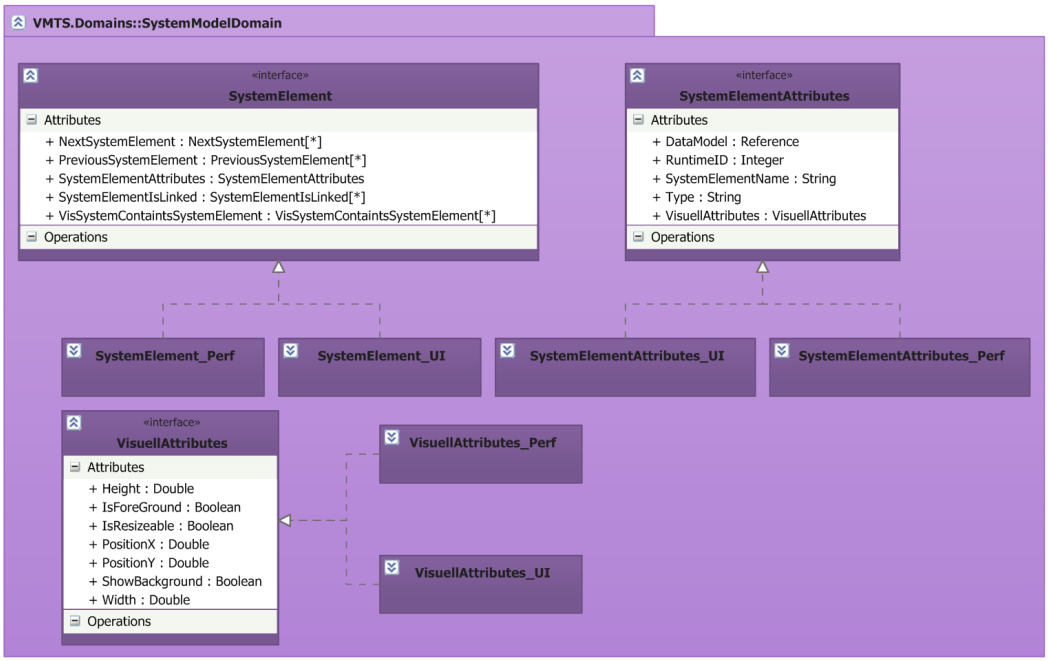


Abbildung 8.1: *Der Klassen und Schnittsteleln des Systemelement-Modellelementes*

Die generierten Klassen sind in der Abbildung 8.1 dargestellt.

Die Schnittstelle `SystemElement` deklariert ein Property zu dem Attribut-Interface `SystemElementAttributes`. Das Attribut-Interface deklariert für jedes Attribut des Modellelementes ein Property. Für komplexe Attribute wird ein eigenes Interface erstellt, das die Eigenschaften des Attributelementes vorschreibt. Außerdem werden von der Schnittstelle `Systemelement` für jede Beziehung Listen mit den Beziehungsmodellelementen deklariert. Jede Schnittstelle wird durch eine eigene Klasse realisiert, welche die Endung `Perf` trägt. Für die Verwendung des „UI“-Modells bestehen ebenso Klassen.

Damit bei Änderungen von Attributen an einem Modellelement eine Benachrichtigung erfolgt, wird für alle Klassen das Interface `INotifyPropertyChanged` implementiert. Dies bedeutet, dass die Vorlagen für die Schnittstellen um das Interface erweitert werden. Zusätzlich wird der Codegenerator bezüglich der Erstellung von Propertys dahingehend angepasst, dass bei Änderungen an einem Attribut das Event `PropertyChanged` gefeuert wird. Im Quelltext 8.1 ist dieses dargestellt.

```
1 <# foreach(ModelGenerator.CodeModelElements.Attribute a in acb.AllAttributes) { #>
2     private static readonly PropertyChangedEventArgs <#=a.Name#>EventArgs = new
          PropertyChangedEventArgs("<#=a.Name#>");
3     private <#=a.Type#> _<#=a.Name#>;
4     public <#=a.Type#> <#=a.Name#> {
5         get{
6                 return _<#=a.Name#>;
7         }
8         <#=a.IsReadOnly?"internal ":""#> set{
```

```
9               if(_<#=a.Name#> != value)
10              {
11                  _<#=a.Name#> = value;
12                  OnPropertyChanged(<#=a.Name#>EventArgs);
13              }
14          }}
15  <#}#>
```

Listing 8.1: *Einfügung von Benachrichtigungen in Propertys*

Die erste Präprozessoranweisung gibt an, dass für jedes Attribut ein privates Feld und ein öffentliches Property erstellt werden soll. Mit `<#=a.Type#>` wird auf den Typ des zu erstellenden Attributes zugegriffen und mit `<#=a.Name#>` auf den Namen. Mit Hilfe dieser Daten wird ein Property erstellt, das evtl. nicht von anderen Instanzen gesetzt werden kann. Diese Eigenschaft wird durch `IsReadOnly` des zu erstellenden Attributes abgefragt. Zusätzlich wird die Methode `OnPropertyChanged` bei Änderungen aufgerufen. Diese feuert für die übergebenen `Eventargs`, die für jede Property explizit erstellt werden, das Event `PropertyChanged` mit dem jeweiligen Instanzobjekt als Senderobjekt. Mit Hilfe dieser Anpassung wird erreicht, dass Änderungen an Attributen eines Modellelementes bekannt gegeben werden. Allerdings werden für Änderungen innerhalb von Listen, beispielsweise das Hinzufügen oder Löschen eines Elementes, keine Benachrichtigungen erstellt. Aus diesem Grund werden die Templates weiterhin verändert, indem Listen durch `ObservableCollections` realisiert werden, welche die geforderte Benachrichtigungseigenschaft aufweisen.

Durch diese Änderungen wird die benötigte Benachrichtigungsfunktionalität in das „NonUI“-Modell implementiert. Allerdings wird bei dem Feuern des Events `PropertyChanged` immer das Instanzobjekt als Sender verwendet. Dies bedeutet, dass bei Änderungen an der Attributsklasse (vgl. `SystemElementAttributes`) immer dieses Objekt angegeben wird. Es besteht allerdings keine Möglichkeit, dass Modellobjekt (vgl. `SystemElement`) zu diesem Attributsobjekt zu erhalten. Zur Weiterverwendung in Beziehungen ist dieses aber erforderlich. Aus diesem Grund wird weiterhin in die Modellklassen das Event `AttributChanged` eingefügt, das bei Änderungen an Properties der Attributsklasse gefeuert wird.

Neben einer Ladeanzeige für den Nutzer baut die Selektionsverknüpfung auf diese Eigenschaft auf. Diese wird im weiteren Verlauf dargestellt.

8.2 Implementation der Selektionsverknüpfung

Ziel der Selektionsverknüpfung ist die Übertragung von Selektionsänderungen zwischen Datenmodellen. Selektionsattribute sind im Datenmodell für die Elemente Achse, Wert und Wertepaar eingefügt worden. Der Selektionsstatus der Elemente wird jeweils durch das Attribut `IsSelected` des Elementes angegeben (vgl. Kapitel 6.4.2). In diesem Kapitel wird die Implementation der Klasse `TargetSourceSelectionLink` beschrieben, die im Wesentlichen diese Verknüpfung durchführt (siehe Klassendiagramm in der Abbildung 7.9). Die Klasse erstellt eine einseitige Verknüpfung zwischen zwei Systemelementen. Im Falle einer gewünschten bidirektionalen Verknüpfung werden zwei Objekte mit vertauschten Quellen bzw. Zielen zur Realisierung verwendet.

Die zentrale Idee zur Umsetzung der Selektionsverknüpfung besteht aus den drei Schritten:

1. **Ermittlung verknüpfungsfähiger Achsen:** Sofern eine Verknüpfung zwischen zwei Datenmodellen möglich ist, müssen Achsenpaare existieren, welche in Beziehung zueinander stehen. Das heißt,

sie entstammen der gleichen Quelle und stehen in einer evtl. hierarchischen Beziehung zueinander. Unabhängig von der Quelleigenschaft können Achsen mit zeitlichen oder räumlichen Eigenschaften miteinander verknüpft werden.

2. **Ermittlung der betroffenen Werte und Wertepaare:** Wenn zwei Achsen zueinander in Beziehung stehen, stellt sich die Frage, welche Werte der einzelnen Achsen in Beziehung stehen. Es ist nicht gegeben, dass jedes Element mit jedem anderen verknüpft werden kann. Beispielsweise bei einer hierarchischen Beziehung zwischen der Quellachse „Jahre“ und der Zielachse „Quartale“, wird erwartet, dass bei der Selektion des Wertes für das Jahr 2007 alle vier Quartale dieses Jahres in der Zielachse selektiert werden, aber nicht die Quartale für das Jahr 2008.
3. **Erstellung der Verknüpfungen:** Nachdem sowohl die verknüpfenden Werte, die Wertepaare und die Achsen ermittelt wurden, ist es notwendig eine Lösung für die Übermittelung von Änderungen zu implementieren. Um Änderungen zu Übertragen wird das Observer-Entwurfsmuster zur Hilfe genommen.

Im weiteren Verlauf werden die einzelnen Schritte im Detail dargestellt. Der erste Schritt wird mit Hilfe der privaten Methode `CreateRelationDic` ausgeführt. Ziel der Methode ist es, Übereinstimmungen zwischen Achsen zu ermitteln. Diese wird von der Methode `CreateLink` aufgerufen, sobald Quell-, wie auch Zielsystemelemente registriert wurden. Um die gefundenen Beziehungen zwischen Achsen im weiteren Verlauf zu sichern, wird ein Dictionary verwendet.

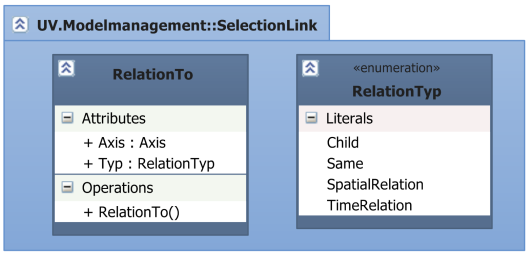


Abbildung 8.2: *RelationTo*

Es soll die Quellachsen als Schlüssel und die Zielachsen als Werte enthalten. Allerdings ist diese Information noch nicht ausreichend, da es unterschiedliche Typen von Beziehungen geben kann, wie im Kapitel zum Abgleich zwischen Datenmodellen dargelegt wurde (vgl. Kapitel 6.4.4). Aus diesem Grund wird die Struktur `RelationTo` eingeführt (siehe Abbildung 8.2). Diese Struktur enthält neben der Achse, den Typ der Beziehung, der durch eine Aufzählung angegeben wird. Es sind folgende Typen möglich:

Child: Die Achse ist in der hierarchischen Beziehung eine Kindachse des im Dictionary angegeben Schlüssels.

Same: Die Achse, die im Dictionary als Schlüssel angegeben ist, und die in der Struktur verwendete Achse sind identisch.

SpatialRelation: Es besteht eine räumliche Beziehung zwischen beiden Achsen. Hierzu müssen beide Achsen vom Typ `SpatialAxis` sein.

TimeRelation: Ähnlich wie bei der `SpatialRelation`-Beziehung sind beide Achsen vom Typ `TimeAxis` und weisen damit eine zeitliche Beziehung zueinander auf.

Die Methode `CreateRelationDic` verwendet die Methode `FindRelations` zur Suche nach in Beziehung stehenden Achsen. Diese füllt das erstellte Dictionary mit den Beziehungen. Hierzu wird jede Quellachse mit jeder Zielachse verglichen. Im Einzelnen wird ein Paar aus Quell- und Zielachse zunächst auf räumliche und zeitliche Eigenschaft geprüft. Anschließend erfolgt eine Prüfung auf Gleichheit und zum Schluss wird die Hierarchie der Quellachse überprüft. Mit dieser Reihenfolge ist sicher gestellt, dass spezielle Attributseigenschaften (räumliche oder zeitliche) immer Vorrang vor

Achseneigenschaften aufweisen. Der zweite Schritt, das Finden von Werten und Wertepaaren, wird innerhalb der Methode `InsertNotifications` durchgeführt. Als Übergabeparameter wird das Beziehungs-Dictionary angegeben. Innerhalb der Methode werden sowohl der Schritt zur Ermittlung der Wertepaare, wie auch die Erstellung von Benachrichtigungen ausgeführt.

Um Benachrichtigungen zu erstellen, wird die generische Klasse `Notification` verwendet. Der Konstruktor dieser Klasse weist zwei generische Parameter auf, die ein Quellelement und Zielelement angeben. Ein Element kann hierbei eine Achse, ein Wert oder ein Wertepaar sein. Diese Modellelemente weisen das Event `AttributChanged` auf, welches gefeuert wird, wenn sich ein Attribut des Elementes ändert. Bei der Instanziierung der Klasse `Notification` wird ein Eventhandler zu diesem Event gebunden. Es wird somit erreicht, dass bei jeder Änderung eines Attributes des Quellelementes die Methode `SourceAttributChanged` ausgeführt wird. Nach Filterung des Selektionsattributs können die Änderungen direkt auf das Selektionsattribut des Zielelementes abgebildet werden.

Beim Einfügen der Benachrichtigungen werden zunächst die ermittelten Achsenpaare behandelt, die identische Eigenschaften aufweisen. Hierzu werden zu allen Werten der Quellachse die identischen Werte der Zielachse gesucht und eine Instanz der Klasse `Notification` mit beiden Werten als Parameter erstellt. Zusätzlich wird eine Benachrichtigung für den Fall, dass sich der Status der Selektion des Wertepaares ändert, eingefügt. Für den Fall, dass eine Kind-Beziehung zwischen Quell- und Zielachse besteht, wird für jedes Element der Quellachse in Kombination mit jedem Element der Zielachse geprüft, ob diese in Beziehung stehen. Als Hilfsmittel wird hierzu der Dienst `ContainmentChecker` verwendet, mit dessen Hilfe geprüft wird, ob ein Wert der Zielachse ein Kind-Wert des betrachteten Quell-Wertes ist. Falls dies der Fall ist, werden entsprechend die Benachrichtigungen für den Wert und das Wertepaar erstellt. Ähnliches wird für den Fall einer zeitlichen oder räumlichen Beziehung zwischen zwei Achsen durchgeführt. Allerdings wird nicht der Dienst `ContainmentChecker` zur Prüfung einer Abhängigkeit verwendet, sondern im Fall einer Zeitbeziehung die Methode `CheckIsInTimeSpan` der Klasse `SelectionLinkUtils` und im Fall einer Raumbeziehung die Methode `CheckIsInSpatialValue` der gleichen Klasse. Zum Abschluss der Methode `InsertNotifications` werden für jedes Achsenpaar in dem übergebenen Dictionary eine Instanz der Klasse `Notification` erstellt, wodurch diese Selektonsänderungen auch übertragen werden.

8.3 Bestehende Systemelemente

Die Architektur des *VAT*-Systems ist darauf ausgelegt, dynamisch um Systemelemente erweitert zu werden. Aus diesem Grund besteht keine feste Verbindung zwischen Systemelementen und den anderen Komponenten der Architektur, sondern die visuelle Repräsentation, das Datenmodell und evtl. nötige Transformationen werden zur Laufzeit zugewiesen. Um die Anforderungen (vgl. Kapitel 6.2) an das System zu erfüllen, werden exemplarisch Quellen, Senken und Transformer (vgl. Kapitel 6.3) als Analysewerkzeuge in das System eingebunden. Diese werden in das Paket `UV.Systemelement` eingeordnet. Als Datenquelle wird die *MUSTANG*-Plattform eingebunden, für die Beschränkung multidimensionaler Datenmengen wird ein Pie-Menü eingesetzt und ein Filter für quantitative Werte zur Verfügung gestellt. Des Weiteren werden als Senken das aus dem Kapitel 2.1.3 bekannte Punktdiagramm, ein Blasendiagramm, eine thematische Karte und eine Mischform aus thematischer Karte und Pie-Chart hinzugefügt. Zunächst wird die *MUSTANG*-Datenquelle vorgestellt.

8.3.1 Quelle: MUSTANG

Die *MUSTANG*-Quelle stellt Daten in einem multidimensionalen Modell zur Verfügung (vgl. Kapitel 2.2). Außerdem kann mit Hilfe von *MUSTANG* auf eine Geo-Datenbank und eine Statistikkomponente zugegriffen werden. Der Zugriff auf die Statistikkomponente erfolgt transparent durch die Wahl von statistischen Kennzahlen, die den *OLAP*-Kennzahlen hinzugefügt werden. Zusammen mit den verfügbaren *OLAP*-Dimensionen stellt *MUSTANG* damit eine Beschreibung der zur Verfügung stehenden Datenmenge bereit, welche mit Hilfe der Methode `PreLoad`, die in der Beschreibung einer Quelle angegeben werden muss (vgl. Kapitel 7.2.2), geladen wird. Das hieraus resultierende Datenmodell wird in einem Pie-Menü dargestellt. Diese Komponente ermöglicht die Auswahl der zu analysierenden Teildatenmenge. Eine Auswahl einer Achse wird durch das `IsUsed`-Attribut eines Achsenmodellelementes des Datenmodells (vgl. Kapitel 6.4.2) symbolisiert. Nachdem eine Datenmenge ausgewählt wird, muss diese noch geladen werden. Hierzu implementiert die Komponente die Methode `LoadSelectedData`, in der die Daten von der *OLAP*-Datenbank geladen werden. Gleichzeitig werden evtl. nötige räumliche Eigenschaften in der *WKT*-Beschreibungssprache von der Geo-Datenbank geladen und einem `AxisValue` hinzugefügt. Weiterhin wird die Möglichkeit bereitgestellt nur bestimmte Nodes der multidimensionalen Datenbank zu laden. Dieses wird erreicht, indem das `IsUsed`-Attribut eines `AxisValues` im Datenmodell, welches der Ladenmethode übergeben wird, gesetzt wird. Diese Eigenschaft wird bei einer Drill-Down-*OLAP*-Operation berücksichtigt, wodurch eine Kombination aus den *OLAP*-Operationen Slice und Drill-Down ermöglicht wird. Das Laden von nicht benötigten Daten wird somit vermieden.

8.3.2 Transformer: Pie-Menü

Die Anforderung der Attributsauswahl [FA.10] bedingt eine geeignete Struktur, um die hierarchischen Daten darzustellen und auszuwählen, bzw. zu Filtern. Ein Pie-Menü, vorgestellt in [Hop91], ist eine Möglichkeit Menüelemente darzustellen. Diese werden um den Mittelpunkt eines Kreises dargestellt. Pie-Menüs haben den Vorteil, dass ein Anwender bei der Auswahl, im Vergleich mit einer klassischen Baumstruktur, eine geringere Strecke zurücklegen muss [Hop91]. In [HFS09] ist dieser Ansatz zur Darstellung von hierarchischen Daten erweitert worden (vgl. Abbildung 8.3). Die Benutzungsoberfläche des Pie-Menüs verwendet ein eigenes Modell, dass Pie-Menü-Modell. Aus diesem Grund ist eine Transformation notwendig, die das bestehende Datenmodell des im Visualisierungspfad vorgelagerten Systemelementes in das proprietäre Modell transformiert und Selektionen vom Pie-Menü auf das Datenmodell abbildet. Auf das Pie-Menü-Modell werden die Achsen des Datenmodells abgebildet, wodurch diese ausgewählt werden können. Durch das Setzen des Datenmodells werden entsprechend die Daten von der Quelle für die selektierten Achsen geladen und im Visualisierungspfad weitergereicht. Die Auswahl eines Elementes erfolgt im Gegensatz zum Ansatz in [HFS09] nicht durch eine Drop-Operation auf die Achse einer Visualisierung, sondern durch

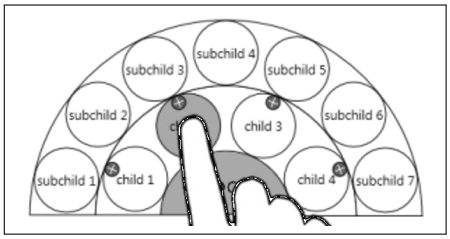


Abbildung 8.3: *Pie-Menü [HFS09]*

eine Drop-Operation auf einen Auswahlring, der nach einer Drag-Operation eines Elementes im Menü als äußere Ringebene dargestellt wird.

8.3.3 Transformer: Quantitativer-Filter

Diese Komponente wird beispielhaft für die Einbindung von Funktionen und statistischen Komponenten zur Verfügung gestellt, wie dies in der Anforderung [FA.3] gefordert ist. Das Systemelement ermöglicht die Filterung von quantitativen Werten durch Angabe eines gültigen Bereiches, der nicht gefiltert werden soll. Es wird somit im Vergleich mit einem Bandpassfilter für Frequenzen ein Tief- und Hochpassfilter bereitgestellt. Hierzu stellt die *GUI* der Komponente die Möglichkeiten zur Einstellung der Filterparameter zur Verfügung. In diesem Fall kann das Minimum und Maximum geändert werden, um einen Bereich anzugeben und die Filterfunktion zu aktivieren bzw. zu deaktivieren. Die Filterung der Datenmenge wird durch eine angepasste Transformation verwendet, die basierend auf den aktuellen Parametern das Datenmodell filtert. Die Schnittstelle `ITransformDescription` deklariert unter anderem die Methode `TransformToVisModel`. Diese wird aufgerufen, bevor ein im Visualisierungspfad folgendes Systemelement das Datenmodell zugewiesen wird. Diese Eigenschaft wird im Filter genutzt um die Filterung durchzuführen. Die Rückgabe der `TransformToVisModel`-Methode ist das gefilterte Datenmodell, welches den nachfolgenden Systemelementen zugewiesen wird.

8.3.4 Senke: Punktdiagramm

Im Grundlagenabschnitt ist ein Punktdiagramm basierend auf einem *VMTS*-Modell vorgestellt worden (siehe Kapitel 2.1.3). Dieses Punktdiagramm wird als eine Senke in das *VAT*-System eingebunden und steht dem Anwender anschließend zur Verfügung. Da für die Visualisierung ein eigenes Modell verwendet wird, ist eine Transformation zu dem Datenmodell notwendig. Diese wird mit Hilfe von *VMTS* modelliert. Sie ist in den Abbildungen 8.4 und 8.5 dargestellt. Die Notation folgt der im Kapitel 2.1.3 vorgestellten. Zunächst wird die Transformation zum Punktdiagrammmodell beschrieben. Im ersten Schritt dieser Transformation wird das eigentliche Modell erstellt und geprüft, welche Achsenpaare zur Abbildung in das Modell ausgewählt werden. Die Regel `CreateCoordinate-PunktdiagramModel` bildet diese Auswahl ab. Innerhalb der Regel wird das Punktdiagrammmodel instanziiert und sofern eine Ordinale- und Koordinatenachse vorhanden sind, wird die Regel `Create-Values` ausgeführt. Ist die Ausführung der Regel nicht erfolgreich (dargestellt durch eine gestrichelte Linie), wird mit Hilfe der Regel `CreateTimeAxis` geprüft, ob eine Kombination aus Zeit- und Koordinatenachse vorhanden ist, oder eine Kombination aus Raum- und Koordinatenachse durch die Regel `CreateSpatialAxis`. Innerhalb der Regel `CreateAmountAxis` wird geprüft ob eine Kombination aus Betragsachse und Koordinatenachse vorliegt. Im Anschluss wird die Regel `CreateValues` zur Erstellung von Wertepaaren des Punktdiagrammmodells und `CreateAxis` zur Abbildung der Datenmodellachsen auf Punktdiagrammmodellachsen ausgeführt. Die abschließende Regel `ResetTransformed` setzt das Hilfsattribut `IsTransformed` des Datenmodells wieder in den Ursprungszustand zurück.

Bei der Rücktransformation (vgl. Abbildung 8.5) muss die Unterscheidung, welche Kombination von Achsenpaaren ursprünglich im Datenmodell vorhanden war, wieder zurückgeführt werden. Hierfür werden jeweils zwei Regeln erstellt, bei der die erste Regel die Achsen des Datenmodells erstellt und die zweite die Werte instanziiert. Die Regel `ResetIsTransformed` führt wie bei der Hintransformation das Zurücksetzten des `IsTransformed`-Attributes aus.

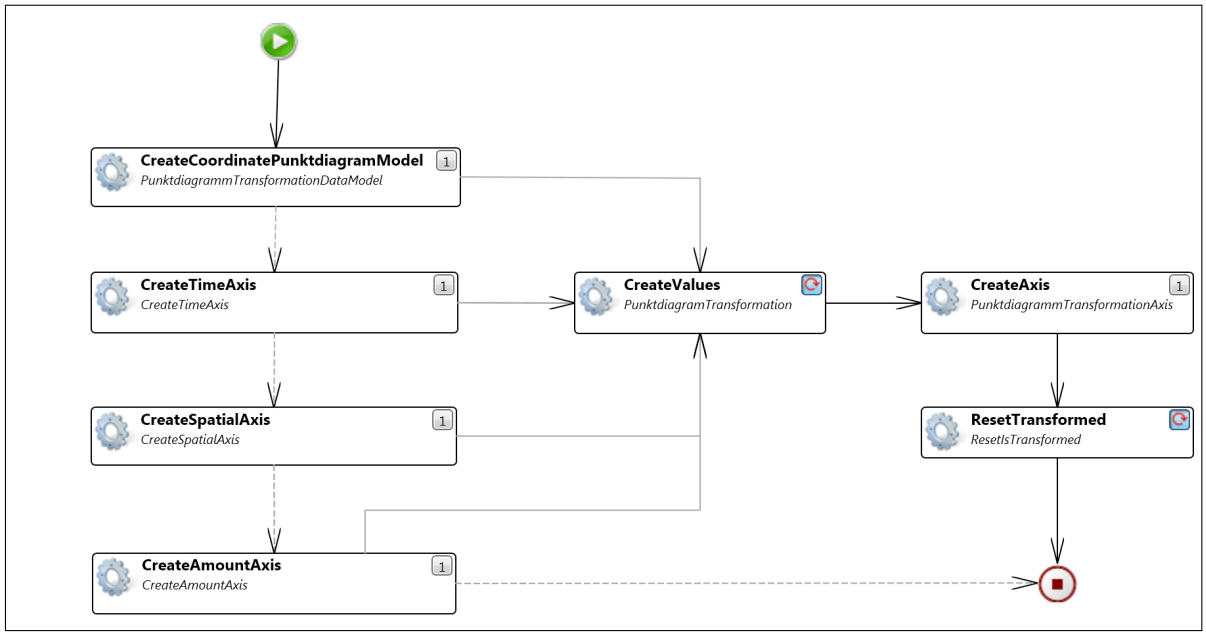


Abbildung 8.4: *Transformation des Datenmodells zum Punktdiagrammmodell*

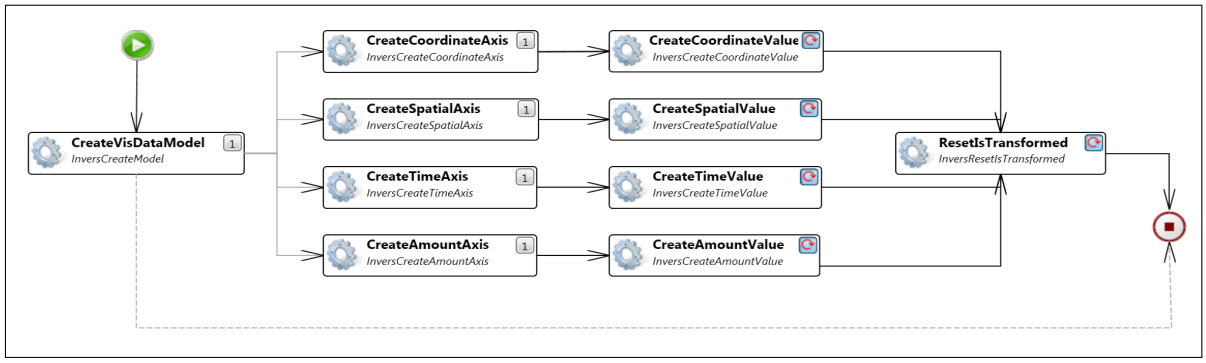


Abbildung 8.5: *Transformation des Punktdiagrammmodells zum Datenmodell*

8.3.5 Senke: Blasendiagramm

Im *VAT*-System sollen zeitbezogene Daten analysiert werden können [FA.2]. Zu diesem Zweck wird die Blasendiagrammkomponente, die dem TaP-System (vorgestellt in [FHTA09]) entstammt, eingefügt. Die Visualisierungsform ermöglicht die Darstellung einer Datenmenge auf fünf Achsen. Neben der X- und Y-Achse stehen eine Größenachse, zur Darstellung eines Datenpunktes in einer relativen Größe zu den anderen Datenpunkten zur Verfügung. Zudem ist eine Farbachse, die Werte mit Hilfe einer chromatischen Verteilung darstellt, eingebunden. Außerdem verfügt das Diagramm über eine Animationsachse, die Attribute mit einer zeitlichen Ordnung darstellt. Die Zuweisung, welches Element auf welcher Achse dargestellt wird, führt der Anwender mit Verschiebungsgesten auf einem Menü durch. Auf Grundlage der besonderen Behandlung von Zeitwerten (dargestellt durch die Animationsachse) wird das Diagramm im *VAT*-System eingebunden und dem Anwender für eine Analyse bereitgestellt. Die Visualisierung wird um den Aspekt der Selektion [FA.15] und der automatischen Achsenzuweisung

ergänzt. Die Achsenzuweisung erleichtert die Wahl der geeigneten Achsenbelegungen, indem ein Regelwerk zur Belegung der Achsen verwendet wird. Bei der automatischen Zuweisung wird sichergestellt, dass auf der X-Achse immer die erste Koordinatenachse dargestellt wird, wie dieses in vielen Visualisierungsformen üblich ist. Die zweite Koordinatenachse eines Datenmodells wird mit Hilfe der Größenachse dargestellt. Die Zeitachse wird vorzugsweise mit der Animationsachse kombiniert, während die zweite Ordinalachse eines Datenmodells durch Farbe symbolisiert wird. Die *GUI* des Blasendiagrammes ermöglicht die Interaktionen Zoomen (semantisch und optisch), Verschieben und Selektion.

8.3.6 Senke: Thematische Karte

Aus der Anforderung [FA.6] geht direkt hervor, dass eine dem Analysten Choroplethenkarte zur Analyse bereitgestellt werden soll. Die Anforderung wird mit dieser Systemkomponente erfüllt. Als Kartenvisualisierung wird eine Karte des *MUSTANG*-Projektes verwendet. Sie ist in *WPF* implementiert. Außerdem stellt das *MUSTANG*-Projekt einen Klassifizierungsdienst zur Verfügung. Dieser Dienst kann nach unterschiedlichen Merkmalen verschiedene Klassifizierungen vornehmen. Zur Verwendung des Dienstes ist allerdings eine Transformation des Datenmodells in das *MUSTANG*-Modell `CubeVO` notwendig. Anhand der Klassifizierung wird ein Farbmodell zugewiesen und die geographischen Daten des Datenmodells auf das *MUSTANG*-Modell abgebildet. Auf Basis dieser Werte wird ein View-Modell für die `MustangThematicMap`-Komponenten erstellt, welches daraufhin eine Choroplethenkarte darstellt. Da auch eine Selektion gefordert ist [FA.16], wird die Kartenkomponente um Eigenschaften zur Selektionsabbildung ergänzt.

8.3.7 Senke: Thematische Karte mit kombiniertem Pie-Chart

Neben zeitbezogenen Daten, die durch das Blasendiagramm dargestellt werden können, sollen im *VAT*-System raumbezogene Daten analysiert werden [FA.2]. Diese können mit Hilfe einer thematischen Karte visualisiert werden. Allerdings kann hiermit immer nur eine Ausprägung dargestellt werden. In der Analyse ist jedoch häufig die Betrachtung von mehreren Einflussfaktoren gleichzeitig notwendig. Aus diesem Grund wird eine Erweiterung einer thematischen Karte vorgenommen, in der ein Pie-Chart als Darstellung einer weiteren ordinalen Achse verwendet wird. Die Architektur dieser Systemkomponente wird im weiteren Verlauf verdeutlicht. Sie ist dargestellt in der Abbildung 8.6.

Als Kartendarstellung wird auf das OpenSource Projekt SharpMap[1] zurückgegriffen. Innerhalb des Projektes wird eine .Net-Kompatible Kartendarstellung entwickelt. Die Grundidee bei der Entwicklung dieser Komponente ist, dass geographische Daten in verschiedenen Ebenen, genannt `Layer`, dargestellt werden. Es besteht die Möglichkeit die `Layer` überlappen zu lassen, wodurch eine Karte dynamisch um Informationen, wie beispielsweise Ländergrenzen, Städte oder topographische Daten ergänzt werden kann. Ein Renderer erstellt die unterschiedlichen `Layer` und fügt diese zu einer Darstellung zusammen. Zurzeit ist jedoch noch kein *WPF*-Renderer implementiert. Aus diesem Grund wird der zur Verfügung stehende WindowsForms-Renderer mit Hilfe des `SharpMapWPFAccess`-Paketes eingebunden. Innerhalb des Paketes wird ein `LayerModel` eingefügt, die eine Schicht innerhalb der Karte beschreibt und von `DefaultThematicMap` mit Hilfe von SharpMap dargestellt werden kann.

[1] Das SharpMap-Projekt ist unter der Adresse `http://www.codeplex.com/SharpMap`, letzter Abruf 25.02.2010, zu erreichen.

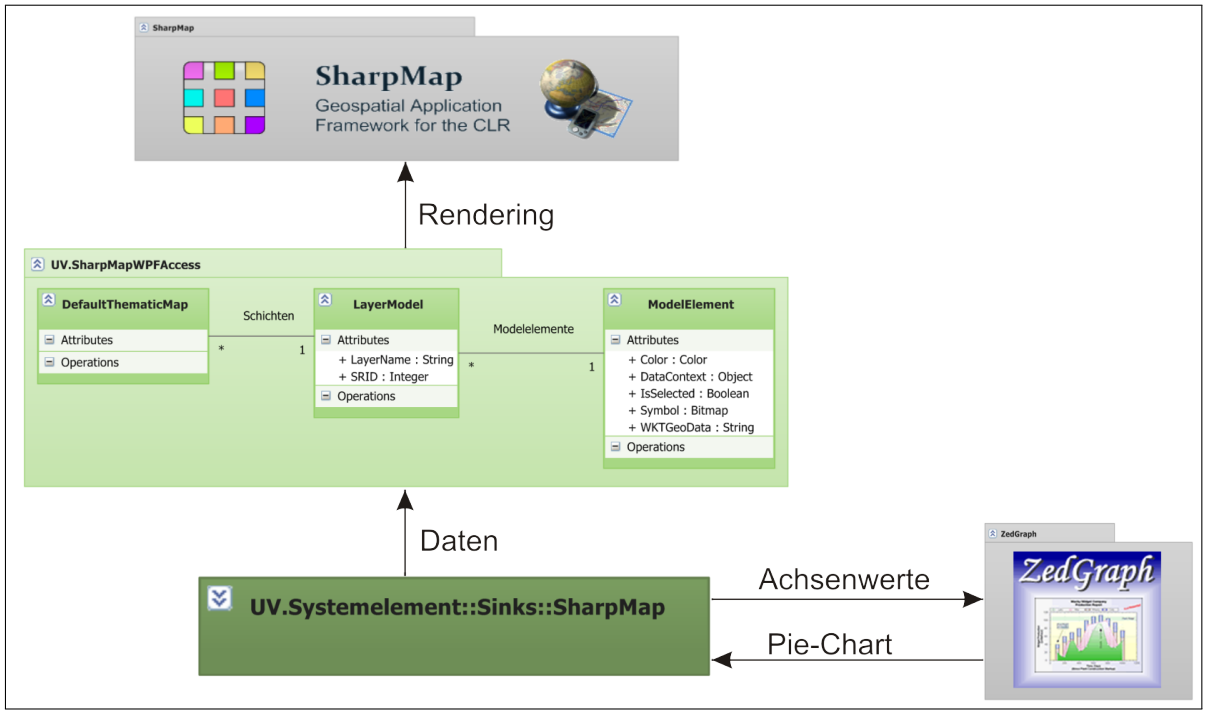


Abbildung 8.6: *Senke: Thematische Karte mit Pie-Charts*

Das `LayerModel` bildet das geographische Modell zu einem `Layer` und besteht aus einer Menge an `ModelElementen`, welche ein räumliches Element in einem Layer darstellt. Die Eigenschaft `Symbol` ermöglicht die Einbindung einer Graphik, die im Schwerpunkt des räumlichen Elementes proportional zu der Fläche dargestellt wird. Hierdurch wird erreicht, dass die thematische Karte nicht nur eine räumliche und eine Koordinatenachse darstellen kann, sondern zusätzlich noch Informationen. Diese Eigenschaft wird genutzt, um eine zusätzliche Ordinalachse durch eine Tortenvisualisierung darzustellen. Die Tortenvisualisierung wird mit Hilfe der freien Diagrammbibliothek ZedGraph[2] erstellt.

8.4 Zusammenfassung

Ausgegangen von den Änderungen an *VMTS* und der darauf aufbauenden Selektionsverknüpfung bis hin zu den eingebundenen Systemelementen stellt dieses Kapitel ausgewählte Implementierungsaspekte des *VAT*-Systems vor. Die Änderung an *VMTS* betreffen den Quellcodegenerator, welcher aus der graphisch erstellten *DSL* eine .Net-Klassenbibliothek erzeugt. Für die Verwendung in dem *VAT*-System sind die Vorlagenklassen umgeschrieben worden, um eine Benachrichtigungsfunktionalität auch bei Nichtverwendung von DependecyPropertys zu ermöglichen. Dieses Ziel ist mit der Implementierung der Schnittstelle `INotifyPropertyChanged` erreicht worden, wodurch gleichzeitig im Vergleich mit der Verwendung von DependencyPropertys ein Performanzvorteil erzielt wurde. Im Anschluss wurde die Selektionsverknüpfung zwischen zwei Datenmodellen vorgestellt. Die Verknüpfung wird in drei Schritten durchgeführt. Im ersten Schritt werden die Gemeinsamkeiten zwischen den Achsen des

[2] Das ZedGraph-Projekt ist unter der Adresse `http://zedgraph.org/`, letzter Abruf 25.02.2010, zu erreichen.

Modells und im Zweiten die Abhängigkeiten zwischen den Werten und Wertepaaren ermittelt. Der letzte Schritt ist die Erstellung von Benachrichtigungen. Diese werden spezifisch für jedes Element erstellt, wodurch die Anzahl der Benachrichtigungen auf ein nötiges beschränkt wird. Abgeschlossen wird das Kapitel durch die Beschreibung der eingebundenen Systemelemente.

Ausgehend von einer Quelle, die *MUSTANG*-Plattform wird zu diesem Zweck eingebunden, können mit Hilfe der Systemelemente ein oder mehrere Visualisierungspfade erstellt werden. Dieser Pfad kann Filter beinhalten. Als Filter steht ein Pie-Menü als Werkzeug zur Auswahl von Attributen [FA.11] und ein quantitativer Wertefilter zur Verfügung. Das Pie-Menü ist um einen speziellen Auswahlring ergänzt worden, der direkt oberhalb des auszuwählenden Elementes erscheint. Durch diese Eigenschaft ist im Vergleich zur Ursprungsversion eine schnellere Auswahl möglich, da der Weg zur Auswahl kürzer ist. Der quantitative Filter wird beispielhaft für weitere Funktionen, wie beispielsweise Data-Mining-Algorithmen oder statistische Methoden eingebunden [FA.3]. Als Visualisierungen der zu analysierenden Datenmenge steht zum einen das aus dem Grundlagenabschnitt bekannte Punktdiagramm zur Verfügung [FA.12]. Zum anderen wird ein Blasendiagramm eingefügt. Das Blasendiagramm kann zusätzlich weitere Achsenwerte darstellen. Besonders hervorzuheben ist die Animationsachse des Blasendiagramms. Auf Grundlage dieser Eigenschaft ist die Visualisierung besonders für Daten mit einem zeitlichen Attribut geeignet [FA.2]. Ein weiterer Attributtyp, der besondere Merkmale aufweist, ist das Raumattribut [FA.2]. Daten mit räumlichen Aspekten können in Karten dargestellt werden. Zu diesem Zweck sind zwei unterschiedliche Kartenvisualisierungen in das System eingebunden. Eine Choroplethenkarte [FA.6], die vom *MUSTANG*-Projekt zur Verfügung gestellt wurde, und eine thematische Karte, die zusätzlich um die Fähigkeit zur Darstellung von Symbolen innerhalb der Fläche ergänzt wurde. Die *MUSTANG*-Karte hat gegenüber der SharpMap-Karte den Vorteil, dass sie in *WPF* entwickelt wurde und mit dem Klassifizierungsdienst des Projektes zusammenarbeitet. Im Gegensatz hierzu werden in der SharpMap-Karte verschiedene Ebenen verwendet, wodurch der Anwender zusätzliche Informationen, die nicht direkt aus der Datenquelle stammen, wie beispielsweise Ländergrenzen, dargestellt bekommen kann. Hierdurch wird eine bessere Orientierung ermöglicht. Außerdem entspricht das verwendete Farbmodell zur Darstellung der Karte dem des Blasendiagramms, wodurch ein Zusammenhang leichter erkennbar ist.

9 Zusammenfassung und Ausblick

Das *VAT*-System, welches im Abschnitt 7 entwickelt wurde, wird in diesem Abschnitt zunächst im Vergleich mit den vorgestellten Systemen im Kapitel 4 eingeordnet, um hierdurch einen Vergleich zu ermöglichen. Im Anschluss wird das Ergebnis dieser Masterarbeit im Verhältnis zu den gestellten Herausforderungen im Bereich von Visual Analytics (vgl. Kapitel 1.4) vorgestellt und das entwickelte System mit den vorgestellten Verglichen. Im Anschluss wird die Arbeit zusammengefasst. Abgeschlossen wird der Abschnitt mit einem Ausblick auf weitere mögliche Schritte in der Entwicklung des Modells, die im *VAT*-System umgesetzt werden könnten.

9.1 Einordnung des Visual-Analytics-Transformation-Systems

Damit das *VAT*-System bewertet werden kann, wird es zunächst nach den Bewertungskriterien der vorgestellten Vergleichssysteme (vgl. Kapitel 4.6) eingeordnet. Um einen Vergleich zu ermöglichen, fließen alle Eigenschaften des *VAT*-Systems ein. Dies bedeutet, die eingebundenen Systemelemente (vgl. Kapitel 8.3) und die Eigenschaften, die auf Grundlage der Verwendung der Modelle (Datenmodell und Systemmodell vgl. Kapitel 6.4) gegeben sind, beeinflussen die Bewertung.

In der Tabelle 9.1 ist das *VAT*-System dem Bewertungsschema hinzugefügt worden. Um einen Verweis auf einzelne Eigenschaften zu ermöglichen, ist jede Zeile zusätzlich mit einer Nummer gekennzeichnet worden. Die Bewertung des *VAT*-Systems erfolgt mit den gleichen Bewertungsausprägungen. Dies bedeutet, wenn eine Eigenschaft erfüllt ist, wird das ✓-Symbol verwendet. Für den Fall, dass eine Eigenschaft nicht erfüllt ist, wird das ✗-Symbol in die Tabelle eingefügt und wenn eine Eigenschaft teilweise zur Verfügung steht, wird das ❍-Symbol gewählt.

Nr.	Eigenschaft	HD-Eye	SellTrend	DataMeadow	MineSet	Advizor	*VAT*
1	Pixeldarstellung	✓	✗	✗	✗	✗	✗
2	Icondarstellung	✓	✗	✗	✗	✓	✗
3	Geschachtelte Darstellung	✗	✓	✗	✗	✓	✓
4	Geometriebasierende Darstellung	✗	✗	✓	✗	✓	✗
5	Standard 2D-/3D-Darstellung	✗	✓	✓	✓	✓	✓
6	Vergleich von Datensätzen	✗	✓	✓	✗	✓	✓
7	Datenvorbereitung	✗	✗	✗	✓	✗	✓
8	Räumliche Daten	✗	✗	✗	✓	✓	✓
9	Hierarchische Daten	✗	✗	✗	✓	✓	✓
10	Zeitliche Daten	✗	✓	✗	✗	✓	✓
11	Multdimensionale Daten	✗	✗	✓	✓	✓	✓
12	Selektion	❍	✓	✓	❍	✓	✓
13	Verknüpfung	✓	✓	✗	✗	✗	✓
14	Selektionsverknüpfung	✗	✗	✓	❍	✓	✓
15	Verschiebung	✗	✗	✓	✓	✓	✓
16	Zoom	✗	✗	❍	✓	✓	✓
17	Filterung	✗	✓	✓	✓	✓	✓
18	Kodierung	✗	✗	✓	✓	❍	✓
19	Parameteranpassung	✓	✗	✗	✓	✓	✓
20	Data-Mining	✗	✗	✗	✓	✓	❍
21	Modellbildung	✗	✗	✗	✓	✗	❍

22	Modellvisualisierung	✗	✗	✗	❍	✓	❍
23	Ziel: Clusterung	✓	✗	✗	✓	✗	❍
24	Ziel: Klassifizierung	✗	✗	✗	✓	✗	❍
25	Ziel: Assoziationen	✗	✗	✗	✗	✓	✗
26	diskrete Animation	✗	✗	✗	✓	✓	✓
27	Speicherung des Analyseergebnisses	✗	✗	✓	✓	✓	❍
28	Wiederanwendung des Wissens	✗	✗	✗	✓	✗	✗

Tabelle 9.1: *Identifizierte Merkmale der Systeme inkl. dem VAT-System im Vergleich*

Das *VAT*-System ist zur Analyse von mehrdimensionalen, hierarchischen Daten entwickelt worden. Insbesondere ist bei der Entwicklung des Systems auf Zeit- und Raumdaten ein besonderer Wert gelegt worden. Für diese Daten stehen im Datenmodell eigene Modellelemente zur Verfügung, die die Besonderheiten von Raum- und Zeitattributen abbilden können. Die Darstellung dieser Daten kann durch die Darstellung in einem Systemelement erfolgen. Eingebunden sind thematische Karten, eine Punktvisualisierung und ein Blasendiagramm, weshalb das Merkmal Standard 2-D/3-D erfüllt ist. Das Blasendiagramm kann Zeitdaten durch eine diskrete Animation darstellen. Außerdem ist das Pie-Menü als geschachtelte Visualisierung eingebunden. Um die Ziele Klassifizierung und Clusterung zu erreichen, ist das Datenmodell um die Beziehung `RelationTo` ergänzt worden. Es stehen jedoch keine Systemelemente zur Verfügung, die diese Beziehung erstellen können. Aus diesem Grund erfolgt eine ❍-Bewertung. Als weiteres Merkmal weist das *VAT*-System zwei Filter auf. Der quantitative Filter dient zur Reduzierung einer Koordinatendatenmenge und das Pie-Menü, um die Menge der betrachteten Achsen eines Datenmodells zu verringern. Werden an einem Systemelement Parameter geändert, können diese, je nach Implementierung des Systemelementes im Visualisierungspfad weitergereicht werden. Das Merkmal der Parameteranpassung ist somit erfüllt. Des Weiteren resultiert aus dem Visualisierungspfad eine Verknüpfung zwischen den Systemelementen eines Pfades. Besteht ein Pfad beispielsweise aus einer Quelle, einem Pie-Menü und einem Blasendiagramm ist sichergestellt, dass im Blasendiagramm nur Daten angezeigt werden, die auch im Pie-Menü ausgewählt wurden. Wird im Pie-Menü eine andere Datenmenge ausgewählt, oder die Quelle wird gewechselt, wird diese Änderung im Visualisierungspfad weitergereicht. Die Selektionsverknüpfung, welche das Merkmal Selektion voraussetzt, ist eine spezielle Form der Verknüpfung. Außerdem bietet das *VAT*-System den `OLAPOperationService` (vgl. Kapitel 7.2.1) um einen semantischen Zoom auszuführen. Dieser Dienst wird dem Anwender von Systemelementen, wie beispielsweise dem Blasendiagramm, in Kombination mit einem optischen Zoom zur Verfügung gestellt. Das Merkmal der Kodierung wird erzielt, indem in jedem Punkt eines Visualisierungspfades (vgl. Kapitel 6.3) ein neuer Pfad eröffnet werden kann, wodurch eine Datenmenge mit Hilfe von verschiedenen Visualisierungsarten gleichzeitig dargestellt werden kann. Zudem können sowohl die Systemelemente selbst, als auch innerhalb der Senken eine Verschiebung durchgeführt werden, wodurch ein bestimmter Bereich dargestellt werden kann. Des Weiteren kann ein Systemelement für einen besseren Vergleich mit einem anderen Systemelement an eine andere Position verschoben werden. Dieser Vergleich wird des Weiteren unterstüzt, indem Systemelemente skaliert werden können. Die Bildung eines Modells auf Basis von Data-Mining-Algorithmen, auch aus bestehenden Visualisierungen, wird ebenfalls durch das *VAT*-System unterstützt. Diese Eigenschaft wird dadurch erreicht, dass das Datenmodell als Abstraktionsschicht zwischen verschiedenen Systemelementen verwendet wird. Allerdings sind noch keine Systemelemente eingebunden, die einen Data-Mining-Algorithmus implementieren. Aus diesem

Grund erfolgt für die Bewertungskriterien Data-Mining, Modellbildung und Modellvisualisierung eine ❍ Bewertung. Eine Speicherung des aktuellen Systemzustandes wird zurzeit nicht unterstützt, allerdings kann dieses auf Grundlage der zentralen Modellhaltung und der Verwendung von *VMTS* nachträglich eingefügt werden. Die hierauf aufbauende Eigenschaft der Wiederanwendung des Wissens kann auf Grundlage der fehlenden Speichereigenschaft zur Zeit nicht geboten werden. Des Weiteren wird auf Grundlage des Datenmodells die Verwendung von unterschiedlichen Datenquellen, in denen die Datenmenge evtl. nach unterschiedlichen Verfahren hinterlegt wird, ermöglicht. Das Datenmodell bildet hierzu eine Abstraktionsschicht, welche durch die Verwendung von Transformationen auch für solche Datenquellen Anwendung finden kann. Die Datenvorbereitung wird damit unterstützt.

9.2 Ergebnis der Arbeit

In der Einführung sind Herausforderungen an Systeme zur visuellen, explorativen Analyse aufgezeigt worden (vgl. Kapitel 1.4). Die Herausforderungen im Bereich Visual-Analytics bestehen in der Verknüpfung von verschiedenen Visualisierungen, wodurch die Schwächen einer Visualisierungsform behoben werden können. In diesem Bereich stellen sich die Fragen, wie die zu verknüpfenden Elemente ermittelt werden und wie diese Verknüpfung abgebildet werden kann, wenn eine Visualisierung nur einen Teilbereich einer anderen Visualisierung darstellt. Als weitere Herausforderung ist die Einbindung von weiteren Datenquellen, Visualisierungen oder Eingabeformen zur Erstellung von Interaktionen identifiziert worden, gerade auch dann wenn eine der Komponenten nicht zur Entwicklungszeit zur Verfügung steht.

Der im *VAT*-System umgesetzte Lösungsansatz ist ein Abstraktionsmodell zwischen Daten, Interaktionen und der visuellen Repräsentation. Dieser Ansatz ist in der Abbildung 9.1 graphisch dargestellt. Im Folgenden wird beschrieben, wie der Lösungsansatz die beschriebenen Herausforderungen löst.

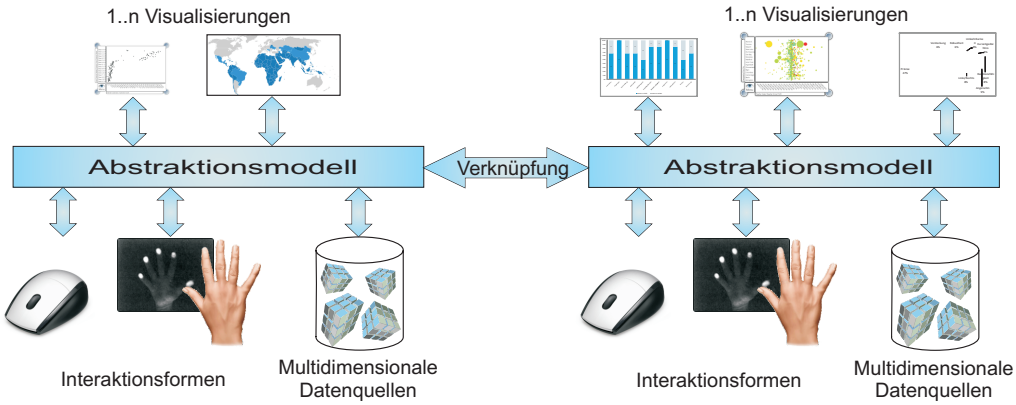


Abbildung 9.1: *Einordnung des Abstraktionsmodells*

Durch das Einfügen einer Abstraktionsschicht zwischen den Visualisierungsformen im System, den Datenquellen und Interaktionen wird erreicht, dass Visualisierungen und Datenquellen unabhängig voneinander entwickelt werden können. Die Abstraktionsschicht, im *VAT*-System ist diese durch das Datenmodell (vgl. Kapitel 6.4.2) gegeben, gleicht Unterschiede zwischen verschiedenen Modellen zur Datenhaltung aus. Zudem wird ein einfacherer Abgleich zwischen unterschiedlichen Instanzen des Datenmodells ermöglicht. Eine Verknüpfung zwischen Instanzen erfordert einen Abgleich der

Datenmodelle als Abstraktion. Da diese Modellstruktur bekannt und für alle Elemente im System gleich ist, können Algorithmen zur Bestimmung der übereinstimmenden Teilbereiche erstellt werden. Im *VAT*-System ist dieses beispielhaft durch die Selektionsverknüpfung demonstriert worden. Ebenso wird die Entwicklung von neuen oder angepassten Visualisierungsformen erleichtert. Neue Diagrammformen können gegen das Abstraktionsmodell entwickelt werden oder es wird eine Transformation zu dem Datenmodell angegeben. Durch anschließendes Einbinden in das Gesamtsystem steht die neu-entwickelte Visualisierung zur Verfügung. Des Weiteren wird von der eigentlichen Datenquelle abstrahiert. Dies hat den Vorteil, dass unterschiedliche Datenquellentypen eingesetzt werden können, um Daten mit Hilfe von Visualisierungen explorativ zu analysieren.

Das *VAT*-System basiert auf dieser Abstraktionsschicht. Im Vergleich (vgl. Tablle 9.1) mit Advizor unterstützt *VAT* die identifizierten Datenformen (räumliche, hierarchische, zeitliche und multidimensionale). Zusätzlich bietet das *VAT* die Datenvorbereitung und ist damit im Vergleich dieser fünf datenbezogenen Eigenschaften (Eigenschaften 7-11) das einzige der betrachteten Systeme, welches alle Merkmale aufweist. Im Bereich der Visualisierungsformen sticht Advizor besonders hervor. Dieses System unterstützt nahezu alle verglichenen Visualisierungstechniken (Eigenschaften 1-5). Nur die Pixeldarstellung wird nicht von diesem System geboten. Diese Technik wird als einzige vom HD-Eye-System verwendet. Die geringe Anzahl von verwendeten Visualisierungstechniken im *VAT*-System kann durch die Einbindung von weiteren Systemelementen erhöht werden. Da das System von der eigentlichen Visualisierungsform unabhängig ist, können auch Pixel-, Icon- oder geometriebasierte Darstellungen eingefügt werden. Des Weiteren fällt bei der Betrachtung der Bewertung auf, dass bei allen Eigenschaften, die direkt mit Interaktionen in Verbindung gebracht werden können (12-18), das *VAT*-System diese Eigenschaften alle erfüllt. Dieses ist unter anderem dem Gesamtkonzept (vgl. Kapitel 6.3) zu verdanken, auf dessen Grundlage die Verknüpfung, Selektionsverknüpfung, Filterung und Kodierung durchgeführt werden kann. MineSet und *VAT* bieten als einzige Systeme die Speicherung und Wiederanwendung des Wissens und unterstützen damit die Feedback-Schleife des Visual Analytics-Prozesses (vgl. Kapitel 1.1). Die anderen Systeme weisen in diesem Bereich Schwächen auf. Im Vergleich mit DataMeadow, das ausgewählt wurde, weil es auch auf einem ähnlichen Visualisierungsprozess basiert, erfüllt *VAT* wesentlich mehr Eigenschaften. Zudem kann es um die Clusterung erweitert werden und kann sich für diesen Fall mit dem HD-Eye-System messen. Besondere Eigenschaften im Vergleich zwischen SellTrend und *VAT* sind nicht auszumachen. Alle Eigenschaften, die von SellTrend erfüllt werden, können ebenso in *VAT* umgesetzt werden. Im Vergleich stechen einzig MineSet, auf Grundlage der Eigenschaften 27 und 28 zur Speicherung und Wiederanwendung des Analysewissens, und Advizor mit der großen Anzahl von unterstützen Visualisierungstechniken im direkten Vergleich mit *VAT* hervor. *VAT* kann allerdings durch die Einbindung von weiteren Systemelementen um weitere Analysewerkzeuge, wie beispielsweise Visualisierungen, Data-Mining-Module oder Speichermodule erweitert werden.

Das *VAT*-System ist somit nicht als ein abgeschlossenes Analysesystem zu verstehen, sondern kann als Basis für die Entwicklung weiterer Analysewerkzeuge verwendet werden. Diese Eigenschaft baut auf der Abstraktionsschicht auf. Im Vergleich mit den betrachteten Systemen ist diese Erweiterbarkeit ein Alleinstellungsmerkmal. Die Einbindung von anderen Modellen zur Datenhaltung, die beispielsweise von Visualisierungen oder Datenquellen verwendet werden, ist in keinem anderen betrachteten System möglich. Das System weist damit Eigenschaften einer Plattform auf. Als weitere Besonderheit ermöglicht das *VAT*-System auf Grundlage der expliziten Behandlung von Raum- und Zeitattributen eine Selektionsverknüpfung zwischen Visualisierungen, deren Visualisierungspfade unterschiedliche

Quellen aufweisen. Dass heißt, es ist möglich eine Verknüpfung unabhängig von der eigentlichen Datenquelle durchzuführen. Dieses ist bei keinem der betrachteten Analysesysteme möglich.

9.3 Zusammenfassung

Auf Grundlage des Information-Overload-Phänomens besteht die Notwendigkeit neue Analyseverfahren zu entwickeln. Das Phänomen stellt die anwachsende Speicherkapazität, im Vergleich zur langsamer zunehmenden Möglichkeit die Datenmenge zu analysieren, heraus. Als eine Lösung wird die Visual-Analytics gesehen. In Visual-Analytics wird der Mensch stärker mit seiner Flexibilität, Kreativität und seinem Allgemeinwissen in den Analyseprozess eingebunden. Dieses wird erreicht, indem der Nutzer über Visualisierungen einen Zugang zu den Daten erhält, wodurch er Muster und Besonderheiten erkennen kann. In der gewählten Anwendungsdomäne, der bevölkerungsbezogenen Epidemiologie, ist das Information-Overload-Phänomen ebenfalls zu beobachten. Die zu analysierende Datenmenge wächst rapide, beispielsweise durch Mammographie-Screenings an. Um diese großen raum- und zeitbezogenen Datenmengen zu analysieren sind unterschiedliche Visualisierungsformen notwendig. Für Zeitdaten können beispielsweise Punktdiagramme verwendet werden und zur Darstellung von räumlichen Daten eignen sich besonders thematische Karten. Als Herausforderung ist zu sehen, diese Daten auch miteinander zu verknüpfen um Korrelationen erkennen zu können. Als weitere Herausforderung ist herauszustellen, dass evtl. durch Forschungen geeignete Visualisierungsformen für ein Analyseziel erst nach Fertigstellung eines Systems entwickelt werden.

Vor diesem Hintergrund wurde in dieser Arbeit das *VAT* entwickelt. Es wurde dabei ein modellgetriebener Entwicklungsansatz verfolgt, wodurch eine leichtere Umsetzung erfolgt. Ziel dieser Entwicklung ist eine an die Anwendungsdomäne (in diesem Fall die visuelle Datenanalyse) angepasste Umgebung. Durch die Domänenabhängigkeit des Modells wird erreicht, dass das Modell kleiner ist, da es keine Allgemeingültigkeit aufweisen muss und aus diesem Grund auch leichter zu warten und zu verwenden ist. Das Tool *VMTS* ist speziell zu diesem Zweck entwickelt worden. Neben der eigentlichen Modellierung und dem Erzeugen von Quellcode bietet dieses Programm die Möglichkeit an, domänenspezifische Datenflüsse zu modellieren und auf ein Modell anzuwenden. Hierzu wird eine Unterscheidung zwischen Metamodell und Instanzmodell getroffen. Ein Metamodell beschreibt die Domäne während ein Instanzmodell konkrete Elemente aus dieser Domäne bereitstellt. Diese Eigenschaften sind an einem Punktdiagramm beispielhaft dargestellt worden. Im weiteren Verlauf ist die *MUSTANG*-Plattform vorgestellt worden. Diese kann verwendet werden, um eine *OLAP*-Datenbank auszulesen. Im Gegensatz zu relationen Datenbanksystemen steht bei diesen Systemen die Analyse von großen Datenmengen im Vordergrund. Hierzu wird eine Datenmenge nicht als flache Struktur, bestehend aus relationalen Tabellen interpretiert, sondern es wird das multidimensionale Modell verwendet. In einer *OLAP*-Datenbank werden die Daten nicht in Tabellenform bereitgestellt, sondern in Cubes, welche von Dimensionen aufgespannt werden und deren Zellwerte mit Hilfe von Kennzahlen bestimmt werden. *MUSTANG* bietet die Möglichkeit an, solche Datenbanken in einem System anzubinden. Des Weiteren stellt *MUSTANG* den Zugriff auf einen Geo-Server zur Verfügung und die Möglichkeit statistische Funktionen einzufügen.

Als nächsten Schritt sind Visualisierungsmodelle und Interaktionsmodelle vorgestellt worden. Diese State-of-the-Art-Betrachtung dient dazu, ein geeignetes Modell zur Verwendung im *VAT*-System zu finden. Im Bereich Visualisierungsmodelle ist das Data-State- und Data-Flow-Modell hervorzuheben. Die Modelle basieren auf dem Pipelineprinzip und verwenden eine Menge an Einzelkomponenten. Das Analyseergebnis wird durch das Durchlaufen aller Einzelkomponenten erreicht. Gleichzeitig ist

festgestellt worden, dass die Modelle keine Dateneigenschaften beschreiben. Vielmehr beschreiben sie das Zusammenspiel einer Menge von Komponenten, welche alle eine eigene Datenmenge aufweisen. Die Notwendigkeit der Bildung eines Datenmodells kristallisiert sich aus dieser Eigenschaft heraus. Für das System sind somit zwei Modelle notwendig. Ein Datenmodell zur Datenhaltung und ein Modell zur Abbildung des Analyseverlaufes.

Damit die Eigenschaften des Datenmodells ermittelt werden können, ist im weiteren Verlauf ein Klassifizierungssystem für Systeme zur visuellen Analyse erstellt worden. In diesem Klassifikationssystem können Analysesysteme aus der Sicht der verfügbaren Visualisierungen eingeordnet werden. Visualisierungen sind im Bereich der visuellen, explorativen Analyse besonders wichtig, da sie für den Anwender den Zugang zu der Datenmenge ermöglichen. Auf Grundlage des Klassifikationssystems von Keim ist das System mit Hilfe von Literaturrecherche erweitert worden. Dieses besteht aus fünf Klassen. Die Klasse *zu visualisierende Daten* umfasst die datenbezogenen Eigenschaften. Mit Hilfe der *Zielorientierungsklasse* werden die Ziele, welche in der Regel mit der Analyse verfolgt werden, klassifiziert. Hierzu stehen in der Klasse *Visualiserungstechnik* verschiedene Visualisierungsformen zur Verfügung, die durch die Elemente der Klasse *Interaktions- und Verzerrungstechnik* detaillierter betrachtet werden können. In der *Animationsklasse* werden Animationsarten eingegliedert, um beispielsweise einen zeitlichen Verlauf darzustellen.

Basierend auf dem Wissen über der Modellkonzepte und den Visualisierungseigenschaften ist anschließend ein Konzept zur Erfüllung der Anforderungen an ein System zur visuellen Datenanalyse vorgestellt worden. Dieses basiert auf einem angepassten Data-Flow-Modell. Das Modell beschreibt einen Pfad bestehend aus einer Menge von Einzelkomponenten, die jeweils Ein- und Ausgänge aufweisen. Dieses Modell ist dahingehend verfeinert worden, dass als Komponenten Quellen, Senken und Transformer möglich sind. Quellen stellen in dem Pfad die Daten bereit, während diese von Transformern umgewandelt bzw. manipuliert und von Senken dargestellt werden. Der hieraus resultierende Pfad wird Visualisierungspfad genannt, wobei jede Komponente, Systemelement genannt, eine eigene Datenhaltung aufweist. Dieses Modell ist anschließend mit Hilfe von *VMTS* in das Metamodell *Systemmodell* umgesetzt worden, welches die Eigenschaften des Konzeptes abbildet. Zusätzlich ist das *Datenmodell* entwickelt worden, um auch eine Datenmenge in jedem Systemelement des Systemmodells zu beschreiben. Das Modell ist auf Grundlage der identifizierten Eigenschaften im Klassifikationssystem und den Eigenschaften der Anwendungsdomäne der bevölkerungsbezogenen Epidemiologie entwickelt worden. Es ermöglicht die Beschreibung von multidimensionalen hierarchischen Daten, die sowohl Zeit- als auch Raumattribute aufweisen können.

Das Konzept mit den dazugehörigen Modellen ist daraufhin in ein prototypisches, modellgetriebenes System für Visual-Analytics überführt worden, bei dem das Hauptaugenmerk auf der Erweiterungsfähigkeit liegt. Die Architektur des *VAT*-Systems besteht aus einer *GUI*, die einem Anwender die Möglichkeit der Interaktion mit dem System gibt, einer Modellverwaltungskomponente, um das Datenmodell und Systemmodell zu verwalten und der Analyseelementkomponente, in der alle Analysewerkzeuge zusammengefasst sind. Ein Analyseelement kann im Vergleich mit dem Systemmodell ein Quelle-, Senken- oder Transformelement darstellen. Diese Systemelemente können auch aus Fremdsystemen zur Analyse verwendet werden, allerdings weisen diese in der Regel ein anderes Modell zur Datenhaltung auf. Damit die Systemelemente trotzdem mit anderen Elementen verknüpft werden können, besteht die Möglichkeit in der Modellverwaltungskomponente Transformationen zwischen dem Fremdmodell und dem entwickelten Datenmodell als Abstraktionsschicht anzugeben. Zu den Komponenten der Architektur des *VAT*-Systems ist parallel die Modellebene eingeordnet, welche die

Beschreibung der Modelle bereitstellt und von allen Komponenten verwendet werden kann, wodurch eine effektive Nutzung von Modellen ermöglicht wird.

Damit die Anforderungen erfüllt werden können, sind in dem *VAT*-System prototypische Systemelemente eingebunden. Mit Hilfe dieser Elemente ist eine Analyse möglich. Eingebunden ist die *MUSTANG*-Plattform als Quelle, die den Zugang zu den bevölkerungsbezogenen epidemiologischen Daten ermöglicht. Des Weiteren sind als Transformer ein Pie-Menü zur Auswahl von Attributen und ein Filter, der auf Basis von quantitativen Werten filtert, eingebunden. Als Senken stehen zur Darstellung von räumlichen Daten Karten zur Verfügung, sowie zur Darstellung von zeitlichen Daten ein Blasendiagramm. Das um diese Systemkomponenten ergänzte *VAT*-System ist anschließend einem Vergleich mit bestehenden Analysesystemen unterzogen worden. Zum Vergleich sind fünf ausgesuchte Systeme mit Besonderheiten in speziellen Gebieten gewählt worden. Anhand dieser Besonderheiten sind Bewertungskriterien ermittelt worden, welche als Vergleichsgrundlage verwendet wurden. Hierbei hat sich gezeigt, dass *VAT* in der Unterstützung der Dateneigenschaften den anderen Systemen überlegen ist. Ebenfalls ermöglicht es die Abbildung der meisten Interaktionen. Allerdings sind Lücken bei der Anzahl der Visualisierungsformen identifiziert worden, welche aber durch die Einbindung von weiteren Systemelementen behoben werden können. Als Besonderheit ist die Erweiterungsfähigkeit identifiziert worden, die in diesem Ausmaß von keinem betrachteten System gegeben ist. Zudem ist die Möglichkeit der Verknüpfung zwischen unterschiedlichen Datenquellen herausgestellt worden. Diese Eigenschaften basieren auf dem Konzept der Abstraktionsschicht.

9.4 Ausblick

Das vorgestellte System ermöglicht die Analyse einer Datenmenge mit zeitlichen und räumlichen Attributen basierend auf Visualisierungspfaden, die aus unterschiedlichen Analysewerkzeugen bestehen. Das System ist flexibel aufgebaut, wodurch weitere Analysewerkzeuge, wie beispielsweise die Einbindung von Pixeldarstellungen, um eine große Datenmenge darstellen zu können, oder Data-Mining-Module, um die Analyseziele der Clusterung und Klassifikation zu erreichen, möglich sind. Gruppierungen von Datenelementen können im Datenmodell mit Hilfe der Relation `PartOf` modelliert werden.

Um den Visual-Analytics-Prozess (vgl. Kapitel 1.1) weiter einzubinden, ist zu evaluieren, ob mit Hilfe des Modellrepositories von *VMTS* eine History-Funktion realisiert werden kann, die den aktuellen Zustand vom *VAT*-System sichert. Auf Grundlage der zentralen Modellhaltung kann dieses geprüft werden. Durch Wiederherstellung eines Zustandes wird die Feedback-Schleife des Prozesses in das System abgebildet, wie dieses im MineSet-System der Fall ist. Eine weitere Möglichkeit die Feedback-Schleife abzubilden, besteht in der Möglichkeit Quellelemente Eingaben zuzuweisen. Allerdings ist dann zu klären, wie das neu gewonnene Wissen wiederverwendet und mit den Originaldaten der Quelle verknüpft werden kann.

Um dem Anwender eine größtmögliche Freiheit bei der Erstellung und Zusammenstellung des Visualisierungspfades zu ermöglichen, kann es sinnvoll sein, dass Analyseelemente mehrere Eingabeelemente aufweisen können. Hierdurch können Daten aus unterschiedlichen Datenpfaden zusammengeführt werden und im weiteren Analyseverlauf gemeinsam betrachtet werden. Hierzu sind zum einen Regeln zum Zusammenfügen der Daten (vergleichbar mit dem Kreuzprodukt der relationalen Algebra) notwendig und zum anderen Visualisierungsformen, durch die der Anwender die Datenmenge explorieren kann. Beispielsweise kann in einer thematischen Karte jede Eingabe durch eine separate

Ebene dargestellt werden. Die Ebenen würden sich evtl. überlagern und sofern ein Farbmodell mit semitransparenten Farben verwendet wird, würden neue Farbverläufe entstehen, wodurch räumliche Korrelationen zwischen Attributen die an den gleichen Orten Extremwerte aufweisen, besonders hervorgehoben werden.

Zudem kann das System um andere Verknüpfungstechniken ergänzt werden. Denkbar sind beispielsweise eine Verknüpfung auf Basis des dargestellten Bildausschnittes, wodurch in verknüpften Visualisierungen, beispielsweise eine thematische Karte und ein Blasendiagramm, immer die gleiche Datenmenge dargestellt wird, auch wenn ein optischer Zoom oder eine Verschiebung durchgeführt wird. Aber auch der Magic-Lens-Ansatz [Dom91], bei dem in einer Visualisierung die Verfeinerungsstufe der ausgewählten Daten dargestellt wird, ist als eine Verknüpfungsform denkbar.

Ein weiterer Ansatz zur Weiterentwicklung des Systems wird in der automatischen Auswahl von passenden Analysewerkzeugen auf Basis des Datenmodells gesehen. Das Datenmodell enthält detailreiche Informationen über die zu analysierende Datenmenge. Durch Auswertung dieser kann erreicht werden, dass einem Anwender nur die Analysewerkzeuge zur Verfügung gestellt werden, die zu dieser Menge kompatibel sind. Beispielsweise kann eine thematische Karte als Visualisierungsform ausgeschlossen werden, wenn die Analysedaten keine räumlichen Eigenschaften aufweisen.

Glossar

Nachfolgend sind noch einmal wesentliche Begriffe dieser Arbeit zusammengefasst und erläutert. Eine ausführliche Erklärung findet sich jeweils in den einführenden Abschnitten sowie der jeweils angegebenen Literatur. Das im Folgenden verwendete Symbol $\sim$ bezieht sich jeweils auf den im Einzelnen vorgestellten Begriff, das Symbol $\uparrow$ verweist auf einen ebenfalls innerhalb dieses Glossars erklärten Begriff.

abstrakte Syntax Die $\sim$ ist eine Datenstruktur, die den $\uparrow$semantisch relevanten Inhalt eines mit Hilfe einer $\uparrow$Domänen spezifischen Sprache definierten Systems darstellt. Sie ist relevant für die Verarbeitung mit Werkzeugen, wie zum Beispiel $\uparrow$Codegeneratoren.

Anwendungsdomäne Der Begriff $\sim$ wird in der Informatik häufig als Synonym für $\uparrow$Domäne verwendet.

Basic Visualization Interactions $\sim$ finden Anwendung in Bereich der $\uparrow$Interaktionsmodellierung. Eine $\sim$ ist eine atomare, unteilbare Einheit einer $\uparrow$Interaktion. Die Einheit besteht aus einem visuellen Kontrollelement und Methoden zur Modifikation von Visualisierungen. Das visuelle Kontrollelement stellt die $\uparrow$Benutzungsschnittstelle dar. Ein Anwender interagiert mit $\sim$ um Visualisierungen zu beeinflussen, wobei die $\sim$ Parameter verändern und damit die $\uparrow$Interaktion abbilden.

Benutzungsschnittstelle Mit Hilfe einer $\sim$ kann ein Anwender $\uparrow$Interaktionen ausführen. Es wird zwischen einer indirekten (beispielsweise mit Hilfe einer Maus und Tastatur) und direkten (beispielsweise durch Touch-Bedienung) $\sim$ unterschieden.

Boxplot Die $\sim$ -Visualisierung ist eine mögliche Darstellung von statistischen Informationen über Daten. Die Stärke besteht in der Darstellung von Verteilungen einer Menge. Hierzu wird durch eine Box ein Datensatz dargestellt, deren Grenzen durch Quantile und den Median bestimmt werden. Extremwerte werden als Punkte außerhalb des Bereiches der Box dargestellt.

Choroplethenkarte Eine $\sim$ ist eine $\uparrow$thematische Karte, bei der die Gebiete im Verhältnis zur Verteilung des beobachteten Merkmales hervorgehoben werden. Meistens wird die Hervorhebung durch unterschiedliche Farbtöne vorgenommen.

Clusteralgorithmen $\sim$ werden zur $\uparrow$Clusterung einer Datenmenge verwendet. Ziel der Algorithmen ist es möglichst ähnliche Objekte in einem Cluster zusammenzufassen, so dass die Ähnlichkeit der Objekte innerhalb eines Clusters maximiert und zwischen den Clustern minimiert wird.

Clusterung Ziel der $\sim$ ist die Entdeckung von Strukturen und Gruppen (Cluster) die bestimmte Eigenschaften aufweisen. Damit ähnelt die $\sim$ der $\uparrow$Klassifizierung. Im Gegensatz zu dieser wird kein Trainingssatz benötigt, da die Daten als Zufallsvariablen interpretiert werden und anhand von $\uparrow$Clusteralgorithmen klassifiziert werden. In der Literatur wird ein breites Spektrum von Clusteralgorithmen vorgeschlagen, welche meistens auf Annahmen über Eigenschaften von Clustern basieren.

Codegenerator Ein $\sim$ wandelt einen $\uparrow$Syntaxbaum in einen $\uparrow$semantisch äquivalenten Quellcode einer bestimmten Programmiersprache um. Der hierdurch generierte Quellcode kann in einem System nach vorheriger Kompilierung verwendet werden.

Cube Ein ∼ ist eine logische Datenstruktur für multidimensionale Daten. Er ist für den effizienten Datenabruf ausgelegt und wird durch ↑Dimensionen aufgespannt. Hierbei müssen die aufspannenden ↑Dimensionen unabhängig voneinander sein. Innerhalb des ∼ beschreiben ↑Kennzahlen die ↑Zellen des ∼ , welche die eigentlichen Daten enthalten.

Data-Mining Ziel des ∼ ist die Auswertung einer großen Datenmenge, um Strukturen und Beziehungen in den Daten zu entdecken. Hierbei wird auf methodische Ansätze aus Statistik, künstlicher Intelligenz, maschinellen Lernen und Mustererkennung zurückgegriffen, um effiziente Verfahren zu entwickeln. Die Auswertung setzt dabei in der Regel keine vorherigen Aussagen über eine Datenmenge voraus [Pet05].

Dense-Pixeltechnik In der ∼ werden Datenpunkte durch entsprechend dem Wert des Datenpunktes eingefärbte Pixel dargestellt. Für jeden Datenpunkt wird im Extremfall nur ein Pixel verwendet, weshalb diese Darstellungstechnik die höchste Darstellungsdichte aufweist. Die Datenpunkte werden nach ihrem Attribut gruppiert und können in einem eckigen oder runden, sowie bei der Erweiterung Voxeltechnik in einem dreidimensionalen Raum eingeordnet werden.

Dice-Operation Die ∼ zählt zu den ↑Online Analytical Processing-Operationen. Mit Hilfe einer Dice-Abfrage ist es möglich, einen Teil-↑Cube aus einem bestehenden ↑Cube zu schneiden. Die ∼ wird in der Regel auf eine ↑Zelle ausgeführt. Sie verursacht eine ↑Drilldown-Operation aller beteiligter ↑Dimensionen. Es werden somit die Details einer ↑Zelle dargestellt.

Dimension Eine ∼ beschreibt eine Sicht auf die assoziierte ↑Kennzahl. Hierbei ist eine ∼ in einer baumartigen Struktur in einzelne ↑Ebenen untergliedert. Zum Beispiel die ∼ Land kann in einem konkreten Modell in die ↑Ebenen Länder, Bundesländer und Regionen untergliedert sein. Zwei ∼ sind orthogonal zueinander, wenn keine funktionale Abhängigkeiten zwischen diesen bestehen.

Domain Specific Language siehe ↑Domänenspezifische Sprache.

Domain Specific Modeling siehe ↑Domänenspezifische Modellierung.

Domäne Eine ∼ bezeichnet einen abgegrenzten Bereich. Häufig wird in der Informatik ∼ als Abkürzung für die Anwendungsdomäne verwendet. Eine Anwendungsdomäne ist ein abgegrenztes Problemfeld eines Softwaresystems, woraus spezielle Anforderungen an ein Softwaresystem hervorgehen.

Domänengetriebene Entwicklung Ziel der ∼ ist eine an die ↑Anwendungsdomäne angepasste Umgebung in der die Konzepte (einschließlich der ↑Semantik) direkt verwendet werden können. Das Modell repräsentiert nicht Konzepte aus der Programmierumgebung, sondern aus der Anwendungsumgebung, wodurch es evtl. auch von Nicht-Programmierern entwickelt werden kann. Des Weiteren ist die Spezifikation des Modells auf einem signifikant höheren Niveau als traditioneller Quellcode und Klassendiagramme, wodurch diese leichter und schneller durchgeführt werden kann. Außerdem fällt das Modell auf Grundlage seiner Anwendungsdomänengültigkeit sehr schlank aus.

Domänenspezifische Modellierung Die ∼ ist eine Technik, um die ↑domänengetriebene Entwicklung zu unterstützen. Die ∼ umfasst die systematische Verwendung von graphischen, ↑domänenspezifischen Sprachen um verschiedene Aspekte eines Systems darzustellen. Die ∼ bietet einen höheren Abstraktionsgrad als allgemeingültige Modellierungssprachen, wodurch für die Erstellung ein geringerer Aufwand erforderlich ist.

Domänenspezifische Sprache Eine ∼ ist eine speziell für eine ↑Domäne erstellte Sprache, die nur die Probleme der ↑Domäne darstellen kann. Sie umfasst die ↑abstrakte und ↑konkrete Syntax. Zudem weist die ∼ eine ↑Semantik auf und in vielen Fällen zusätzliche Einschränkungen.

Domänenspezifisches Modell Ein ∼ ist ein Modell für eine ↑Domäne. Im Unterschied zu einem allgemeinen Modell, ist dieses wesentlich kleiner und bildet nur die Umgebung in der ↑Domäne ab. Eine Möglichkeit ein ∼ zu erzeugen besteht in der Verwendung des ↑Visual Modeling and Transformation Systems.

Drill-Across-Operation Die ∼ zählt zu den ↑Online Analytical Processing-Operationen. Mit Hilfe der Operation ist ein Wechseln zwischen ↑Cubes möglich. Sie wird durchgeführt, um eine Navigation auf gleicher Hierarchieebene durchzuführen. Es wird zwischen gleichen ↑Ebenen unterschiedlicher ↑Hierarchien einer ↑Dimension gewechselt und dadurch die Geschwister der gleichen ↑Dimension betrachtet.

Drill-Down-Operation Bei einer ∼ handelt es sich um eine ↑Online Analytical Processing-Operationen auf einem ↑Cube. Mit Hilfe der Operation wird eine Sicht auf eine höhere Detailstufe der Daten durch den Wechsel in eine tiefere Aggregationsstufe einer ↑Hierarchie ermöglicht. Die Operation ist die Komplementäroperation zur ↑Roll-Up-Operation.

Ebene Stufen einer ↑Hierarchie werden im multidimensionalen Modell ∼ genannt. Zu jeder ∼ existieren ein oder evtl. auch mehrere Pfade im Modell. Zum Beispiel kann in der ↑Dimension Datum die ∼ Tag durch den Pfad Datum → Jahr → Monat erreicht werden. Eine ∼ besteht aus einer Menge von ↑Knoten und ist Bestandteil eines ↑Cubes.

Epidemiologie Die ∼ ist eine Wissenschaft, die sich mit der Verteilung von Krankheiten einer Bevölkerung und den Einflussfaktoren für die Verteilung beschäftigt. Es wird dabei zwischen zwei Arten unterschieden. Die deskriptive ∼ untersucht das Auftreten von Krankheiten in einer bestimmten Bevölkerung mit soziodemografischen Attributen. Hingegen werden in der analytischen ∼ die Hintergründe der Erkrankungen untersucht, um die Ursachen für die Erkrankung zu ermitteln.

Epidemiologisches Krebsregister Niedersachsen Das ∼ registriert seit 2003 neu auftretende Krebserkrankungen in einer Datenbank und wertet diese aus. Das ∼ ist in Bereiche zur Erfassung und Auswertung aufgeteilt. Zurzeit sind weit über eine Millionen Datensätze zu Tumorerkrankungen und krebsbedingten Todesfällen in der Registerstelle verzeichnet.

Erkenntnisprozess Um aus einer Datenmenge Schlussfolgerungen ziehen zu können, müssen die Daten einem systematischen ∼ unterzogen werden. Ein ∼ beschreibt einen Weg ausgehend von Daten und Informationen, Wissen über diese Daten zu gewinnen und wiederanzuwenden. Der Prozess wird nach den Aspekten Erkenntnisinteresse, Erkenntnispraxis und Erkenntniskategorie charakterisiert.

Fast Analysis of Shared Multidimensional Information ∼ ist eine weit verbreitete Definition der Eigenschaften eines ↑Online Analytical Processing-Systems. Die Definition umfasst die fünf Eigenschaften Geschwindigkeit (Fast), Analysemöglichkeit (Analysis), Sicherheit (Shared), Multdimensionalität (Multidimensional) und Kapazität (Information). Eine Beschreibung der Merkmale ist im Kapitel 2.2.1 zu finden.

Geographic Information System siehe ↑Geografisches Informationssystem.

Geographisches Informationssystem $\sim$ sind ↑Informationssysteme zur Speicherung und Bearbeitung von räumlich bezogenen Daten. Sie ermöglichen die Verknüpfung der räumlichen Daten mit ↑Kennzahlen und stellen diese in ↑thematischen Karten dar. Eine Möglichkeit ein $\sim$ in einem Analysesystem einzubinden ist die ↑Multidimensional Statistical Data Analysis Engine-Plattform.

Glyph Ein $\sim$ ist eine Repräsentation eines Datensatzes, bei dem die Daten durch spezielle Ausprägungen eines Objektes dargestellt werden. Beispielsweise können ausgehend von einem Mittelpunkt strahlenartige Linien ein Attribut eines Datensatzes durch die Länge der Linie abbilden. Es entsteht für jeden Datensatz in der Datenmenge ein individueller $\sim$.

Graphical User Interface siehe ↑Benutzungsschnittstelle.

Hierarchie Eine $\sim$ beschreibt im multidimensionalen Modell die Ordnung in der Elemente anderen untergeordnet sind. Innerhalb der Baumstruktur einer ↑Dimension im Modell können unterschiedliche Pfade zu ↑Ebenen bestehen. Diese Pfade werden $\sim$ genannt.

Histogramm Ein $\sim$ stellt die Häufigkeitsverteilung von Merkmalen graphisch dar. Dies erfordert die Einteilung der Datenmenge in Klassen nach bestimmten Merkmalen. Eine weit verbreitete Darstellungsform eines $\sim$ ist die Balkendarstellung, wobei die Balkenhöhe durch die Häufigkeit bestimmt wird. Mit Hilfe eines $\sim$ kann ein visueller Eindruck über die Häufigkeitsverteilung großer Datenmengen erreicht werden.

Hypertext Markup Language $\sim$ ist eine Markup-Sprache zur Auszeichnung von textbasierten Dokumenten. In der Struktur können Inhalte, wie Texte und Bilder beschrieben werden. Die mit Hilfe der $\sim$ beschrieben Dokumente bilden die Grundlage von Inhalten im Internet und werden von einem Browser dargestellt.

Information-Overload Im Bereich der Datenanalyse stellt das $\sim$ -Phänomen das Problem der schneller anwachsenden Speicherkapazität im Vergleich zur weniger deutlich zunehmenden Möglichkeit die Datenmenge zu analysieren heraus. Als eine Lösung wird ↑Visual-Analytics gesehen.

Informationssystem Ein $\sim$ ist ein Verbund aus Hardware und Software zur Erfassung, Speicherung und / oder Transformation von Informationen. Zusätzlich umfasst ein $\sim$ menschliche Komponenten mit dem Ziel der optimalen Bereitstellung von Informationen. Ein $\sim$ ist damit ein soziotechnisches System.

Instanzmodell Das $\sim$ bildet eine Instanz eines ↑Metamodells. In einem $\sim$ können nur die Sprachelemente des vorher definierten ↑Metamodells verwendet werden. Das $\sim$ kann im ↑Modellrepository hinterlegt werden und später erneut geladen werden.

Interaktion In der Informatik beschreibt eine $\sim$ eine einseitige Handlung zwischen Mensch und Maschine. Hierdurch entsteht eine Kommunikation zwischen Mensch und Maschine. Eine $\sim$ wird immer auf eine ↑Operation in einem System abgebildet.

Interaktionsmodellierung Die $\sim$ verfolgt das Ziel ↑Interaktionen eines Anwenders auf ein Modell abzubilden. Es wird hierbei zwischen syntaktischen Modellen, zur Abbildung von Benutzerinteraktionen auf ein Modell, und ↑semantische Modelle, zur Abbildung des Zieles der Interaktion auf ein Modell (vgl. Kapitel 3.2).

International Statistical Classification of Diseases and Related Health Problems Die $\sim$ ist eine Verschlüsselung von Diagnosen in der ambulanten und stationären Versorgung. Die $\sim$

ist von der Weltgesundheitsorganisation erstellt worden und wurde ins Deutsche übertragen und herausgegeben. Die Kodierung unterliegt einem stetigen Wandel. Die aktuelle Version ist die Revision 10.

Kennzahl Eine ∼ beschreibt im multidimensionalen Modell eine konkrete Ausprägung und beinhaltet den Datenwert zu den aufgespannten ↑Dimensionen im ↑Cube. Wird in der ↑Hierarchie nicht die tiefste ↑Ebene betrachtet, findet eine Aggregation der ∼ statt.

Klassifizierung ∼ ist der Prozess der Entdeckung eines Klassifikationsmodells basierend auf einem Trainingssatz mit bekannten Klassendaten. Um das Klassifikationsmodell zu erstellen werden die Attribute vom Trainingsdatensatz analysiert, sowie eine zielgenaue Beschreibung oder ein Modell von den Klassen basierend auf den Attributen entwickelt. Diese Beschreibung wird benutzt um einen meist größeren und unbekannten Datensatz zu klassifizieren.

Knoten Die konkreten Elemente in jeder ↑Ebene im multidimensionalen Modell, beispielsweise für die ↑Ebene Bundesland der ∼ Niedersachsen, werden ∼ genannt. Sie repräsentieren damit die Koordinaten einer ↑Zelle innerhalb eines ↑Cubes.

Komplexes Modellierungsattribut Ein ∼ besteht im ↑Visual Modeling and Transformation System aus einer Menge von Attributen, die evtl. auch wieder komplex sein können. Damit ist ein ∼ vergleichbar mit einer Struktur in der Programmierung. ∼ werden im Kapitel 2.1.3 beschrieben.

Konfirmatorische Statisik In der ∼ , auch schließende Statistik genannt, werden allgemeingültige Schlussfolgerungen aus einer Analyse oder Studienergebnissen gezogen. Bei dieser Analyse lassen sich Fehler allerdings nicht ausschließen. Dies kann in einem Unterschied zwischen der untersuchten Teilmenge und der gesamten Teilmenge begründet sein.

Konkrete Syntax Anwender, die eine ↑domänenspezifische Sprache verwenden möchten, interagieren mit der ∼ . Diese ist eine Darstellung der ↑abstrakten Syntax. Die Interaktion mit der ∼ kann entweder grafisch mit Hilfe eines Modellierungstools geschehen, oder durch Texteingabe. Bei der textuellen Definition ist eine Grammatik notwendig, die eine formale Beschreibung der ∼ ist und somit eine Abbildung zur ↑abstrakten Syntax ermöglicht.

Language Workbench Eine ∼ ist eine Werkzeugsammlung mit den benötigten Tools zur Erstellung (zum Beispiel Editoren oder Verifizierer) und Verwendung (zum Beispiel ↑Codegeneratoren oder Interpreter sowie angepasste Editoren) von ↑domänenspezifischen Sprachen.

Level siehe ↑Ebene.

Liniendiagramm Ein ∼ stellt eine Datenmenge als eine Serie von Punkten dar, die durch eine Linie verbunden sind. Damit wird ein funktionaler Zusammenhang zwischen den Daten visualisiert. ∼ werden verwendet, um große Datenmengen darzustellen, die kontinuierlich über einen Zeitraum verteilt sind.

Linking-und-Brushing ∼ klassifizieren Techniken, die eine Verknüpfung von verschiedenen Visualisierungen ermöglichen. Hierdurch wird die Erkennung eines Zusammenhangs zwischen evtl., auch teilweise, unterschiedlichen Datensätzen in verschiedenen Visualisierungsarten erleichtert. Eine ∼ ist die ↑Selektionsverknüpfung.

Metamodell ∼ bilden die Elemente eines ↑Instanzmodells. Sie beschreiben die Eigenschaften eines Modellelementes und stellen diese Beschreibung zur weiteren Verfügung. Im ↑Visual Modeling

and Transformation System ist immer ein ∼ notwendig, damit ein ↑Instanzmodell erstellt werden kann. Es wird mit Hilfe des ↑Quellcoderators aus einer ↑domänenspezifischen Sprache erzeugt. Ein ∼ , beispielsweise das ↑Rootmeta-Modell, bildet im ↑Visual Modeling and Transformation System wiederum eine ↑domänenspezifische Sprache.

Model-View-Viewmodel Bei ∼ handelt es sich um ein Entwurfsmuster zur losen Kopplung von Benutzungssteuerelementen und Daten, welche in Modellen gehalten werden. Es wird häufig in ↑Windows Presentation Foundation-Anwendungen verwendet. Das Entwurfsmuster sieht eine Dreiteilung vor. Die View zur Anzeige der Benutzungselemente, ein Viewmodel als Zwischenschicht zwischen View und dem Model als drittes Element. Im Idealfall wird somit eine Trennung zwischen View und Modell erreicht.

Modellrepository Ein ∼ ist ein Speicher, in dem Modelle zentral abgelegt werden und erneut geladen werden können. Das ∼ erlaubt die Nutzung von Modellen über mehrere Anwender hinaus. Zur Beschreibung der Daten im ∼ sind Metadaten notwendig. Im ↑Visual Modeling and Transformation System wird ein ∼ verwendet, um ↑Instanzmodelle zu sichern, die mit Hilfe von ↑Metamodellen beschrieben werden.

Multidimensional Statistical Data Analysis Engine ∼ ist eine geographisch, statistische Datenanalyseplattform zur Erstellung von Anwendungen zur Analyse von multidimensionalen Daten. Die Plattform bietet zu diesem Zweck verschiedene Services an (siehe Kapitel 2.2).

Multipurpose Internet Mail Extension ∼ ist ein Kodierungsverfahren, das zum Beispiel bei E-Mails Anwendung findet, um beliebige Inhalte zu kodieren und zu übertragen. Des Weiteren wird ∼ bei der Deklaration von verschiedenen Inhalten verwendet. Der Standard wird in RFC 2045 [FB96b] bis RFC 2049 [FB96a] beschrieben.

Node siehe ↑Knoten.

Object Constraint Language ∼ ist eine Modellierungssprache mit der es möglich ist Softwaremodule zu erstellen. Sie ist seit der ↑Unified Modeling Language Version 2 Bestandteil der ↑Unified Modeling Language-Definition und wird hauptsächlich für textuelle Spezifikationen von Bedingungen in der ↑Unified Modeling Language verwendet. Im ↑Visual Modeling and Transformation System kann die ∼ zur Erstellung von Modelleinschränkungen verwendet werden.

Online Analytical Processing Unter dem Begriff ∼ werden verschiedene Anwendungen und Technologien zur Erfassung, Verwaltung, Verarbeitung und Darstellung multidimensionaler Daten für Analyse- und Managementzwecke zusammengefasst. Die Eigenschaften eines ∼ können durch ↑Fast Analysis of Shared Multidimensional Information definiert werden.

Operation Bezogen auf den Zusammenhang in dieser Arbeit ist eine ∼ ein softwaretechnisches Verhaltensmerkmal, welches den Einstiegspunkt zu einem Verhalten darstellt und auf Methoden im System abgebildet wird. Eine ∼ wird durch eine ↑Interaktion eines Anwenders aufgerufen.

Orthogonale Klassen Sind zwei oder mehr Klassen orthogonal zueinander, beinhalten die betrachten Klassen keine Eigenschaften einer anderen Klasse. ∼ setzt Disjunktivität der betrachteten Klassen voraus. Sind zwei Klassen orthogonal, können diese auch unabhängig voneinander analysiert werden.

Parabox Die ∼ -Visualisierung wird zur Darstellung von n-dimensionalen kontinuierlichen oder kategorischen Daten verwendet. Sie ist eine Kombination aus ↑Boxplot und ↑Parallel-Koordinatenvisualisierung. Die ∼ vereint die Stärke der ↑Boxplot-Visualisierung von Verteilungen in einer Menge und der ↑Parallel-Koordinatenvisualisierung zur Darstellung von hochdimensionalen Datensätzen.

Parallel-Koordinatenvisualisierung In ∼ werden Datensätze durch ein Polygon, das durch die Achsenschnittpunkte bestimmt wird, visualisiert. Es wird somit jede ↑Dimension einzeln dargestellt und kann differenziert betrachtet werden. Der Vorteil der Visualisierungstechnik besteht in der Aufdeckung von Korrelationen. Ähnliche bzw. gleiche Datensätze weisen einen parallelen Verlauf auf, während Extremwerte sich von diesen abheben.

Pivotierungs-Operation Synonym für die ↑Rotating-Operation.

Property Eine ∼ ist eine Eigenschaft einer Klasse. Sie stellt Setter- und Gettermethoden zum Zugriff bereit. In der Regel werden öffentliche Felder einer UML-Klassennotation in der Programmiersprache C# auf ∼ abgebildet. Eine Erweiterung sind Dependency-∼ . Sie bieten die Möglichkeit an, eine bidirektionale Bindung zu erstellen.

Punktdiagramm Ein ∼ stellt eine Datenmenge als eine Menge von Datenpunkte dar. Hierbei können zwei Merkmale dargestellt werden. Die Position der Punkte wird durch das bestimmende Wertepaar bestimmt. ∼ dienen im Allgemeinen zum Vergleich aggregierter Daten über verschiedene Kategorien hinweg. Als eine Erweiterung des ∼ ist ein Blasendiagramm zu sehen, in dem zusätzlich Merkmale durch die Größe und in einigen Fällen die Farbe der Punkte dargestellt werden kann.

Roll-Up-Operation Bei der ∼ werden neue Informationen durch Aggregation der Daten innerhalb eines ↑Cubes erzeugt. Die Operation kann zum Beispiel verwendet werden, um von dem Quartalslevel eines ↑Cubes auf die Jahresebene zu wechseln. Die ↑Online Analytical Processing-Operation wird immer auf einer ↑Dimension ausgeführt.

RootMeta-Modell Das ∼ stellt im ↑Visual Modeling and Transformation System das Modell der untersten Ebene dar. Dies bedeutet alle Modelle im ↑Visual Modeling and Transformation System erben die Eigenschaften des ↑Metamodells. Als Modellelemente stehen Atome und Beziehungen zur Verfügung. Eine Beschreibung des Modells ist im Kapitel 2.1.3 zu finden.

Rotating-Operation Mit Hilfe der ∼ werden die ↑Dimensionen eines ↑Cubes aus einer anderen Perspektive dargestellt, indem eine ↑Dimension des ↑Cubes rotiert wird. Es ergibt sich eine andere Zusammenstellung der ↑Ebenen, welche den ↑Cube aufspannen. Die Daten der ↑Kennzahl werden somit anhand einer anderen Menge von ↑Knoten gruppiert, bzw. berechnet.

Selektionsverknüpfung Zu den ↑Linking-und-Brushing-Techniken gehört die ∼ . Die Technik beschreibt eine Verknüpfung zwischen zwei Visualisierungen basierend auf Selektionsänderungen die durch ↑Interaktionen ausgelöst werden. Eine Selektion eines Datensatzes, Datenwertes oder einer Achse innerhalb einer Visualisierung wird auf die verknüpfte Visualisierung abgebildet.

Semantik Die ∼ stellt die Bedeutung einer Sprache in den Vordergrund. Bei einem Vergleich zweier Sprachen kann durchaus die Syntax unterschiedlich sein, beispielsweise zwischen Java- und C#-Quellcode, der Zweck und die Bedeutung können allerdings übereinstimmen.

Slice-Operation ∼ zählen zu den ↑Online Analytical Processing-Operationen. Mit Hilfe einer Slice-Abfrage wird eine „Scheibe“ aus einem ↑Cube geschnitten. Die Scheiben werden durch die

Auswahl eines ↑Knotens einer aufspannenden ↑Dimension des ↑Cubes bestimmt. Bei einer ∼ -Operation wird die Dimensionalität eines ↑Cubes verringert.

Spatial Reference System Identifier ∼ werden zur eindeutigen Identifizierung des für geodätische Daten angewandten Referenzsystems verwendet. Eine ∼ ist Teil der ↑Well-known Text-Spezifikation. Verwaltet werden ∼ von dem Oil and Gas Producers Surveying and Positioning Committee.

Syntaxbaum In der Informatik können Ableitungen einer Grammatik als ∼ dargestellt. Er wird auch als Ableitungs- oder Parsebaum bezeichnet. Häufig wird ein abstrakter ∼ zum Parsen verwendet. In einem abstrakten ∼ sind unbedeutende und überflüssige Zeichen, wie beispielsweise Kommentare, nicht zu enthalten.

Systemelement ∼ werden in einem ↑Visualisierungspfad verwendet. Es wird zwischen Quellen zur Datenbereitstellung, Transfomern zur Datenmanipulation und Senken zur Visualisierung einer Datenmenge unterschieden. Ein Systemelement weist neben der Position innerhalb des ↑Visualisierungspfades ein Datenmodell auf, in dem die Daten zu diesem ∼ strukturiert sind (vgl. Kapitel 6.3).

Thematische Karte ∼ stellen ein raumbezogenes Attribut unterschiedlichster Art in einer Karte dar, ohne eine direkte Abbildung der Erdoberfläche durchzuführen. Somit kann beispielsweise die Bevölkerungsdichte in einem Gebiet dargestellt werden. ∼ werden häufig mit topografischen Karten verbunden, um eine Orientierung in der Karte zu ermöglichen. Eine Form der ∼ ist die ↑Choroplethenkarte.

TreeMap Eine ∼ ist eine geschachtelte Visualisierung, in der hierarchische Daten in Form untereinander aufgeteilter Untereinheiten dargestellt werden. Die einzelnen Teilflächen werden anhand eines Kategorisierungsparameters bestimmt. Die Größe der Teilflächen und häufig auch die Farbe wird implizit durch die Mächtigkeit der enthaltenen Subelemente definiert.

Unified Modeling Language Die ∼ ist eine standardisierte graphische Sprache zur Modellierung von Modellen für Geschäftsprozesse und die objektorientierte Softwareentwicklung. Sie definiert dabei eine Notation und ↑Semantik zur Visualisierung, Konstruktion und Dokumentation der Modelle. Auf Grundlage des großen Umfangs ist die ∼ in verschiedene Teile untergliedert.

Visual Modeling and Transformation System ∼ ist eine domänenspezifische Modellierungs- und Modelltransformationsumgebung, welche an der Universität in Budapest entwickelt wird. ∼ ermöglicht die ↑domänengetriebene Entwicklung, in dem eine graphische Modellierung von domänenspezifischen Modellen ermöglicht wird.

Visual-Analytics siehe ↑visuelle, explorative Datenanalyse.

Visual-Analytics-Prozess Ein ∼ beschreibt ein Vorgehensmuster zur Analyse einer Datenmenge in der ↑visuellen, explorativen Datenanalyse. Ein Prozess ist im Kapitel 1.1 zu finden. Dieser ist eine Kombination von automatischen und visuellen Analysemethoden mit einer engen Kopplung an Benutzerinteraktionen, um das Wissen aus den Daten zu gewinnen. Er besteht aus Transitionen und Zuständen.

Visual-Analytics-Transformation-System Das ∼ ist ein System zur ↑visuellen, explorativen Datenanalyse. Es ermöglicht auf Grundlage der Verwendung des Daten- und Systemmodells (vgl.

Kapitel 6.4) die Einbindung von verschiedenen Datenquellen. Das System basiert auf ↑Visualisierungspfaden, die aus unterschiedlichen ↑Systemelementen bestehen (vgl. Kapitel 6.3).

Visualisierungspfad Ein ∼ ist eine Möglichkeit, Teile des ↑Visual-Analytics-Prozess in einem System abzubilden. Er besteht aus einer Menge an ↑Systemelementen, die miteinander gerichtet verbunden sind. Der ∼ ist eine Form einer Pipeline. Das Modell ist im Kapitel 6.3 näher beschrieben.

Visuelle, explorative Datenanalyse Die ∼ ist die Kombination von interaktiven Visualisierungen mit Analysetechnologien, Datenmanagement und den besonderen Fähigkeiten des Menschen (beispielsweise die Intuition, das Allgemeinwissen oder die Fähigkeit Zusammenhänge zu erkennen und Daten zu filtern) zur Auswertung eines Datenbestandes. Bei der ∼ wird eine Datenmenge dem Anwender visuell aufbereitet dargestellt. Im Vergleich mit ↑Data-Mining können hierdurch verrauschte und inhomogene Daten leichter analysiert werden. Das Vorgehen bei diesem Datenanalyseverfahren wird durch ↑Visual-Analytics-Prozesse beschrieben (siehe auch 1.1).

Well-known Text Die ∼ ist eine Beschreibungssprache für geographische Objekte. Die ∼ -Spezifikation umfasst eine Beschreibung für die geographischen Elemente in textueller Weise. Die Koordinaten werden innerhalb der Beschreibung in Bezug zu einem Referenzsystem, die mit einem ↑Spatial Reference System Identifier kodiert ist, erstellt.

Windows Presentation Foundation ∼ ist ein Graphik-Framework zur Erstellung von *GUI*-Elementen in .Net. Als Beschreibungssprache für die Elemente wird in der Regel eine angepasste XML Notation verwendet, welche mit Hilfe von partiellen Klassendefinitionen um Laufzeitlogik erweitert wird. Der Kern von ∼ basiert auf einem auflösungsunabhängigen vektorbasierten Renderingmodul, das auf Grundlage der Verwendung von DirectX hardwareoptimiert ist.

XML for Analysis ∼ ist ein Standard zum Datenzugriff auf ↑Online Analytical Processing-Systemen. Der Standard unterstützt das multidimensionale Modell. Mit Hilfe der ↑Multidimensional Statistical Data Analysis Engine können ∼ -gültige Anfragen erstellt werden und an einen ↑Online Analytical Processing-Server gesendet werden.

Zelle Im multidimensionalen Modell besteht ein ↑Cube aus ∼ . ∼ fassen die eigentlichen Datenwerte zusammen und werden aus einer oder mehreren ↑Kennzahlen gebildet. Eine ∼ kann innerhalb eines ↑Cubes eindeutig mit Hilfe der beteiligten ↑Nodes identifiziert werden.

Abkürzungen

BVI Basic Visualization Interactions.

DSL domänenspezifische Sprache (Domain Specific Language).

DSM domänenspezifische Modellierung (Domain Specific Modeling).

EKN Epidemiologisches Krebsregister Niedersachsen.

FASMI Fast Analysis of Shared Multidimensional Information.

GIS Geographic Information System.

GUI Graphical User Interface.

HTML Hypertext Markup Language.

ICD International Statistical Classification of Diseases and Related Health Problems.

MIME Multipurpose Internet Mail Extension.

MUSTANG Multidimensional Statistical Data Analysis Engine.

MVVM Model-View-Viewmodel.

OCL Object Constraint Language.

OLAP Online Analytical Processing.

SRID Spatial Reference System Identifier.

UML Unified Modeling Language.

VAT Visual-Analytics-Transformation.

VMTS Visual Modeling and Transformation System.

VTM Visualization-Transform-Model.

WKT Well-known Text.

WPF Windows Presentation Foundation.

XMLA XML for Analysis.

Abbildungen

Literatur

[ABM+07] AIGNER, Wolfgang ; BERTONE, Alessio ; MIKSCH, Silvia ; TOMINSKI, Christian ; SCHUMANN, Heidrun: Towards a conceptual framework for visual analytics of time and time-oriented data. In: *WSC '07: Proceedings of the 39th conference on Winter simulation*. Piscataway, NJ, USA : IEEE Press, 2007. – ISBN 1–4244–1306–0, S. 721–729 (Bezugnahme / zitiert auf Seite 43)

[AKK96] ANKERST, Mihael ; KEIM, Daniel A. ; KRIEGEL, Hans-Peter: Circle Segments: A Technique for Visually Exploring Large Multidimensional Data Sets. In: *Visualization '96, Hot Topic Session*, 1996 (Bezugnahme / zitiert auf Seite 44)

[AMS+96] AGRAWAL, Rakesh ; MANNILA, Heikki ; SRIKANT, Ramakrishnan ; TOIVONEN, Hannu ; VERKAMO, A. I.: Fast discovery of association rules. (1996), S. 307–328. ISBN 0–262–56097–6 (Bezugnahme / zitiert auf Seite 47)

[AN95] AAMODT, Agnar ; NYGÅRD, Mads: Different Roles and Mutual Dependencies of Data, Information, and Knowledge - An AI Perspective on their Integration. In: *Data & Knowledge Engineering* 16 (1995), Nr. 3, S. 191–222 (Bezugnahme / zitiert auf Seite 1)

[BH02] BERTHOLD, Michael (Hrsg.) ; HAND, David J. (Hrsg.): *Intelligent Data Analysis: An Introduction*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2002. – ISBN 3540430601 (Bezugnahme / zitiert auf den Seiten 44, 47, und 48)

[BKK97] BRUNK, Clifford ; KELLY, James ; KOHAVI, Ron: MineSet: An Integrated System for Data Mining. In: *KDD*, 1997, S. 135–138 (Bezugnahme / zitiert auf Seite 33)

[BST+00] BOSCH, Robert ; STOLTE, Chris ; TANG, Diane ; GERTH, John ; ROSENBLUM, Mendel ; HANRAHAN, Pat: Rivet: a flexible environment for computer systems visualization. In: *SIGGRAPH Comput. Graph.* 34 (2000), Nr. 1, S. 68–73. – ISSN 0097–8930 (Bezugnahme / zitiert auf Seite 25)

[Car03] CARLSON, Christopher N.: Information overload, retrieval strategies and Internet user empowerment. In: HADDON, Leslie (Hrsg.): *The Good, the Bad and the Irrelevant (COST 269)* Bd. 1, Media Lab UIAH, 2003, 169–173 (Bezugnahme / zitiert auf Seite 1)

[CBB+05] CHILDS, Hank ; BRUGGER, Eric ; BONNELL, Kathleen ; MEREDITH, Jeremy ; MILLER, Mark ; WHITLOCK, Brad ; MAX, Nelson: A Contract Based System For Large Data Visualization. In: *Visualization Conference, IEEE* 0 (2005), S. 25. ISBN 0–7803–9462–3 (Bezugnahme / zitiert auf Seite 22)

[CCS93] CODD, E. F. ; CODD, S. B. ; SALLEY, C. T.: *Providing OLAP to User-Analysts: An IT Mandate*. 1993 (Bezugnahme / zitiert auf Seite 16)

[CES07] COOK, Kris ; EARNSHAW, Rae A. ; STASKO, John T.: Guest Editors' Introduction: Discovering the Unexpected. In: *IEEE Computer Graphics and Applications* 27 (2007), Nr. 5, S. 15–19 (Bezugnahme / zitiert auf Seite 2)

[Chi00] CHI, Ed H.: A Taxonomy of Visualization Techniques Using the Data State Reference Model. In: *INFOVIS '00: Proceedings of the IEEE Symposium on Information Vizualization 2000*. Washington, DC, USA : IEEE Computer Society, 2000. – ISBN 0–7695–0804–9, S. 69 (Bezugnahme / zitiert auf Seite 27)

[Chi02] CHI, Ed H.: Expressiveness of the data flow and data state models in visualization systems. In: *AVI '02: Proceedings of the Working Conference on Advanced Visual Interfaces*. New York, NY, USA : ACM, 2002. – ISBN 1–58113–537–8, S. 375–378 (Bezugnahme / zitiert auf Seite 61)

[CMS99] CARD, Stuart K. ; MACKINLAY, Jock D. ; SHNEIDERMAN, Ben: *Information visualization*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1999. – 1–34 S. – ISBN 1–55860–533–9 (Bezugnahme / zitiert auf Seite 39)

[CR96] CHUAH, Mei ; ROTH, Steven F.: On the Semantics of Interactive Visualizations. In: *Proceedings of the IEEE Symposium on Information Visualization*, IEEE Computer Society, October 1996, S. 29–36 (Bezugnahme / zitiert auf den Seiten 26 und 27)

[CR98] CHI, Ed Huai-hsin ; RIEDL, John: An Operator Interaction Framework for Visualization Systems. In: *INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization*. Washington, DC, USA : IEEE Computer Society, 1998. – ISBN 0–8186–9093–3, S. 63–70 (Bezugnahme / zitiert auf den Seiten 23, 25, und 26)

[CSCC99] CATARCI, Tiziana ; SANTUCCI, Giuseppe ; COSTABILE, Maria F. ; CRUZ, Isabel: Foundations of the DARE System for Drawing Adequate Representations. In: *Database Applications in Non-Traditional Environments* 0 (1999), S. 461. ISBN 0–7695–0496–5 (Bezugnahme / zitiert auf den Seiten 42 und 43)

[Dic09] The American Heritage® Dictionary of the English Language, Fourth Edition. (2009), Oct. `http://dictionary.reference.com/browse/trend` (Bezugnahme / zitiert auf Seite 46)

[DKR97] DERTHICK, Mark ; KOLOJEJCHICK, John ; ROTH, Steven F.: An interactive visual query environment for exploring data. In: *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM, 1997. – ISBN 0–89791–881–9, S. 189–198 (Bezugnahme / zitiert auf Seite 24)

[Dom91] DOMIK, Gitta: The Role of Visualization in Understanding Data. In: MAURER, Hermann A. (Hrsg.): *New Results and New Trends in Computer Science* Bd. 555, Springer, 1991 (Lecture Notes in Computer Science). – ISBN 3–540–54869–6, S. 91–107 (Bezugnahme / zitiert auf Seite 118)

[Eic09a] EICK, Stephen G. ; ADVIZOR SULTIONS INC. (Hrsg.): *ADVIZOR: A Technical Overview*. ADVIZOR Sultions Inc., 08 2009 (Bezugnahme / zitiert auf Seite 34)

[Eic09b] EICK, Stephen G.: *Data Visualization Software | ADVIZOR Solutions*. „Website“, 2009. – Online verfügbar auf `http://www.advizorsolutions.com/default.htm`; abgerufen am 17.03.2010 (Bezugnahme / zitiert auf den Seiten 29 und 34)

[EL96] EICK, Stephen G. ; LUCAS, Paul J.: Displaying trace files. In: *Softw. Pract. Exper.* 26 (1996), Nr. 4, S. 399–409. – ISSN 0038–0644 (Bezugnahme / zitiert auf Seite 35)

[EST08] ELMQVIST, Niklas ; STASKO, John ; TSIGAS, Philippas: DataMeadow: a visual canvas for analysis of large-scale multivariate data. In: *Information Visualization* 7 (2008), Nr. 1, S. 18–33. – ISSN 1473–8716 (Bezugnahme / zitiert auf den Seiten 23 und 32)

[FB96a] FREED, N. ; BORENSTEIN, N.: *Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples*. United States, 1996 (Bezugnahme / zitiert

auf Seite 124)

[FB96b] FREED, N. ; BORENSTEIN, N.: *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. United States, 1996 (Bezugnahme / zitiert auf Seite 124)

[FB96c] FREED, N. ; BORENSTEIN, N.: *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. United States, 1996 (Bezugnahme / zitiert auf Seite 24)

[FFIT00] FUJISHIRO, Issei ; FURUHATA, Rika ; ICHIKAWA, Yoshihiko ; TAKESHIMA, Yuriko: GADGET/IV: A Taxonomic Approach to Semi-Automatic Design of Information Visualization Applications Using Modular Visualization Environment. In: *Information Visualization* 0 (2000), S. 77. – ISSN 1522–404X (Bezugnahme / zitiert auf Seite 26)

[FH09] FLÖRING, Stefan ; HESSELMANN, Tobias: *TaP: Towards Visual Analytics on Interactive Surfaces*. 2009 (Bezugnahme / zitiert auf Seite 5)

[FHTA09] FLÖRING, Stefan ; HESSELMANN, Tobias ; TEIKEN, Yvette ; APPELRATH, Hans-Jürgen: Kollaborative visuelle Analyse multidimensionaler Daten auf Surface-Computern. In: *Datenbank-Spektrum* 9 (2009), Dezember, Nr. 31, S. 17–25. – ISSN 1618–2162 (Bezugnahme / zitiert auf den Seiten 3, 18, 19, und 106)

[FIET01] F.OELLIEN ; IHLENFELDT, W.D. ; ENGEL, K. ; T.ERTL: Multi-Variate Interactive Visualization of Data from Laboratory Notebooks. In: *ECDL: Workshop 'Generalized Documents'* Sep. 2001 (2001) (Bezugnahme / zitiert auf Seite 44)

[Fow05] FOWLER, Martin: *Language Workbenches: The Killer-App for Domain Specific Languages?* „Website“, Juni 2005. – Online verfügbar auf `http://www.martinfowler.com/articles/languageWorkbench.html`; abgerufen am 08.10.2009 (Bezugnahme / zitiert auf Seite 8)

[FTIN97] FUJISHIRO, I. ; TAKESHIMA, Y. ; ICHIKAWA, Y. ; NAKAMURA, K.: GADGET: goal-oriented application design guidance for modular visualization environments. In: *Visualization Conference, IEEE* 0 (1997), S. 245. – ISSN 1070–2385 (Bezugnahme / zitiert auf Seite 26)

[Gan08] GANTZ, John F.: *The Diverse and Exploding Digital Universe*. 03 2008. – Whitepaper. – Online verfügbar auf `http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf`; abgerufen am 04.02.2010 (Bezugnahme / zitiert auf Seite 1)

[GE97] GERSHON, Nahum ; EICK, Stephen G.: Information Visualization. In: *IEEE Transactions on Visualization and Computer Graphics* 17 (1997), Nr. 4, S. 29–31. – ISSN 0272–1716 (Bezugnahme / zitiert auf Seite 1)

[G.E00] G.EICK, Stephen: Visual Discovery and Analysis. In: *IEEE Transactions on Visualization and Computer Graphics* 6 (2000), Nr. 01, S. 44–58. – ISSN 1077–2626 (Bezugnahme / zitiert auf Seite 34)

[GYC07] GOODCHILD, Michael F. ; YUAN, May ; COVA, Thomas J.: Towards a general theory of geographic representation in GIS. In: *Int. J. Geogr. Inf. Sci.* 21 (2007), Nr. 3, S. 239–260. – ISSN 1365–8816 (Bezugnahme / zitiert auf Seite 24)

[HA08] HEER, Jeffrey ; AGRAWALA, Maneesh: Design considerations for collaborative visual analytics. In: *Information Visualization* 7 (2008), Nr. 1, S. 49–62. – ISSN 1473–8716 (Bezugnahme / zitiert auf Seite 27)

[HDP93] HIBBARD, William L. ; DYER, Charles R. ; PAUL, Brian E.: The VIS-AD Data Model: Integrating Metadata and Polymorphic Display with a Scientific Programming Language. In: *Proceedings of the IEEE Visualization '93 Workshop on Database Issues for Data Visualization*. London, UK : Springer-Verlag, 1993. – ISBN 3–540–58519–2, S. 37–68 (Bezugnahme / zitiert auf Seite 24)

[HDP94] HIBBARD, William L. ; DYER, Charles R. ; PAUL, Brian E.: A Lattice Model for Data Display. In: BERGERON, R. D. (Hrsg.) ; KAUFMAN, Arie E. (Hrsg.): *IEEE Visualization*, IEEE Computer Society, 1994. – ISBN 0–8186–6626–9, S. 310–317 (Bezugnahme / zitiert auf Seite 24)

[Her06] HERRING, John R.: *OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture*. 10 2006 (Bezugnahme / zitiert auf Seite 68)

[HFS09] HESSELMANN, Tobias ; FLÖRING, Stefan ; SCHMITT, Marwin: Stacked Half-Pie Menus - Navigating Nested Menus on Interactive Tabletops. In: *ITS '09 - Interactive Tabletops and Surfaces 2009* ACM, 2009 (Bezugnahme / zitiert auf Seite 104)

[Hib98] HIBBARD, Bill: VisAD: connecting people to computations and people to people. In: *SIGGRAPH Comput. Graph.* 32 (1998), Nr. 3, S. 10–12. – ISSN 0097–8930 (Bezugnahme / zitiert auf Seite 24)

[HKW99] HINNEBURG, Alexander ; KEIM, Daniel A. ; WAWRYNIUK, Markus: HD-Eye: Visual Mining of High-Dimensional Data. In: *IEEE Computer Graphics and Applications* 19 (1999), Nr. 5, S. 22–31. – ISSN 0272–1716 (Bezugnahme / zitiert auf Seite 29)

[HKW03] HINNEBURG, Alexander ; KEIM, Daniel A. ; WAWRYNIUK, Markus: HD-Eye - visual clustering of high dimensional data: a demonstration. In: *IEEE Computer Graphics and Applications* 19 (2003), Nr. 5, S. 735–755. – ISSN 0272–1716 (Bezugnahme / zitiert auf den Seiten 29 und 30)

[HM90] HABER, R.B. ; MCNABB, D. A.: Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In: *Visualization in Scientific Computing*. 1990 (Bezugnahme / zitiert auf Seite 22)

[Hop91] HOPKINS, Don: The Design and Implementation of Pie Menus. In: *Dr. Dobb's Journal* Dec (1991) (Bezugnahme / zitiert auf Seite 104)

[HR00] HILBERT, David M. ; REDMILES, David F.: Extracting usability information from user interface events. In: *ACM Comput. Surv.* 32 (2000), Nr. 4, S. 384–421. – ISSN 0360–0300 (Bezugnahme / zitiert auf Seite 25)

[HSM09] HANRAHAN, Pat ; STOLTE, Chris ; MACKINLAY, Jock: *Selecting a Visual Analytics Application*. April 2009 (Bezugnahme / zitiert auf Seite 4)

[IC07] ISENBERG, Petra ; CARPENDALE, M. Sheelagh T.: Interactive Tree Comparison for Co-located Collaborative Information Visualization. In: *IEEE Transactions on Visualization and Computer Graphics* 13 (2007), Nr. 6, S. 1232–1239 (Bezugnahme / zitiert auf Seite 5)

[Ins85] INSELBERG, Alfred: The plane with parallel coordinates. In: *The Visual Computer* V1 (1985), December, Nr. 4, S. 69–91 (Bezugnahme / zitiert auf Seite 33)

[Jai99] JAIN, Sanjay: Simulation in the next millennium. In: *WSC '99: Proceedings of the 31st conference on Winter simulation*. New York, NY, USA : ACM, 1999. – ISBN 0–7803–5780–9, S. 1478–1484 (Bezugnahme / zitiert auf Seite 46)

[JK03] JANKUN-KELLY, T.J.: *Visualizing Visualization: A Model and Framework for Visualization Exploration*, Center for Image Processing and Integrated Computing, University of California, Davis, Diss., Juni 2003 (Bezugnahme / zitiert auf den Seiten 22, 24, 25, 26, 27, und 63)

[JKM02] JANKUN-KELLY, T. J. ; MA, Kwan-Liu: VisSheet redux: redesigning a visualization exploration spreadsheet for the web. In: *SIGGRAPH '02: ACM SIGGRAPH 2002 conference abstracts and applications*. New York, NY, USA : ACM, 2002. – ISBN 1–58113–525–4, S. 329–329 (Bezugnahme / zitiert auf Seite 26)

[JKMG02] JANKUN-KELLY, T. J. ; MA, Kwan L. ; GERTZ, Michael: A model for the visualization exploration process. In: *VIS '02: Proceedings of the conference on Visualization '02*. Washington, DC, USA : IEEE Computer Society, 2002. – ISBN 0–7803–7498–3, S. 323–330 (Bezugnahme / zitiert auf Seite 24)

[JKMG07] JANKUN-KELLY, T.J. ; MA, Kwan-Liu ; GERTZ, Michael: A Model and Framework for Visualization Exploration. In: *IEEE Transactions on Visualization and Computer Graphics* 13 (2007), Nr. 2, S. 357–369. – ISSN 1077–2626 (Bezugnahme / zitiert auf den Seiten 24, 25, 26, und 27)

[KAK95] KEIM, Daniel A. ; ANKERST, Mihael ; KRIEGEL, Hans-Peter: Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data. In: *IEEE Visualization*, 1995, S. 279– (Bezugnahme / zitiert auf Seite 44)

[Kei01] KEIM, Daniel A.: Visual exploration of large data sets. In: *Commun. ACM* 44 (2001), Nr. 8, S. 38–44. – ISSN 0001–0782 (Bezugnahme / zitiert auf den Seiten 3, 39, 45, und 48)

[Kei02a] KEIM, Daniel A.: Datenvisualisierung und Data Mining. In: *Datenbank-Spektrum* 2 (2002), S. 30–39 (Bezugnahme / zitiert auf den Seiten 5, 40, 45, und 46)

[Kei02b] KEIM, Daniel A.: Information Visualization and Visual Data Mining. In: *IEEE Transactions on Visualization and Computer Graphics* 8 (2002), Nr. 1, S. 1–8. – ISSN 1077–2626 (Bezugnahme / zitiert auf den Seiten 1 und 3)

[KK94] KEIM, Daniel A. ; KRIGEL, Hans-Peter: VisDB: Database Exploration Using Multidimensional Visualization. In: *IEEE Transactions on Visualization and Computer Graphics* 14 (1994), Nr. 5, S. 40–49. – ISSN 0272–1716 (Bezugnahme / zitiert auf Seite 44)

[KKS+09] KEIM, Daniel A. ; KOHLHAMMER, Joern ; SANTUCCI, Giuseppe ; MANSMANN, Florian ; WANNER, Franz ; SCHAEFER, Matthias: Visual Analytics Challenges. Istanbul, Turkey : eChallenges 2009, 2009 (Bezugnahme / zitiert auf Seite 1)

[KL95] KOHAVI, Ron ; LI, Chia-Hsin: Oblivious Decision Trees, Graphs, and Top-Down Pruning. In: *IJCAI*, 1995, S. 1071–1079 (Bezugnahme / zitiert auf Seite 34)

[KMS+08] KEIM, Daniel A. ; MANSMANN, Florian ; SCHNEIDEWIND, Jörn ; THOMAS, Jim ;

ZIEGLER, Hartmut: Visual Analytics: Scope and Challenges. In: [SBM08b], S. 76–90 (Bezugnahme / zitiert auf den Seiten 1, 2, und 46)

[KMSZ09] KEIM, Daniel A. ; MANSMANN, Florian ; STOFFEL, Andreas ; ZIEGLER, Hartmut: Visual Analytics. In: *Encyclopedia of Database Systems*. Springer, 2009 (Bezugnahme / zitiert auf Seite 2)

[KPS03] KEIM, Daniel A. ; PANSE, Christian ; SIPS, Mike: Visual Data Mining of Large Spatial Data Sets. In: BIANCHI-BERTHOUZE, Nadia (Hrsg.): *DNIS* Bd. 2822, Springer, 2003 (Lecture Notes in Computer Science). – ISBN 3–540–20111–4, S. 201–215 (Bezugnahme / zitiert auf den Seiten 43 und 48)

[LO95] LEE, Hing-Yan ; ONG, Hwee-Leng: A New Visualisation Technique for Knowledge Discovery in OLAP (Abstract). In: *KDOOD/TDOOD*, 1995, S. 23–25 (Bezugnahme / zitiert auf Seite 4)

[LP95] LEE, Edward A. ; PARKS, Thomas M.: Dataflow Process Networks. In: *Proceedings of the IEEE* 83 (1995), 05, Nr. 5, S. 773–801 (Bezugnahme / zitiert auf Seite 23)

[LSS09] LIU, Zhicheng ; STASKO, John ; SULLIVAN, Timothy: SellTrend: Inter-Attribute Visual Analysis of Temporal Transaction Data. In: *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), Nr. 6, S. 1025–1032. – ISSN 1077–2626 (Bezugnahme / zitiert auf den Seiten 30 und 31)

[Ma99] MA, Kwan-Liu: Image graphs—a novel approach to visual data exploration. In: *VIS '99: Proceedings of the conference on Visualization '99*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1999. – ISBN 0–7803–5897, S. 81–88 (Bezugnahme / zitiert auf Seite 26)

[Man99] MANN, Thomas M.: Visualization of WWW-Search Results. In: *DEXA Workshop*, 1999, S. 264–268 (Bezugnahme / zitiert auf Seite 39)

[Men00] MENNIS: A conceptual framework for incorporating cognitive principles into geographical database representation. In: *International Journal of Geographical Information Science* 14 (2000), Nr. 6, S. 501 (Bezugnahme / zitiert auf den Seiten 24 und 67)

[MLC05] MEZEI, Gergely ; LEVENDOVSZKY, Tihamér ; CHARAF, Hassan: A Presentation Framework for Metamodeling Environments. Montego Bay, Jamaica, October 2005 (Bezugnahme / zitiert auf Seite 10)

[MMC07] MÉSZÁROS, Tamás ; MEZEI, Gergely ; CHARAF, Hassan: *A General Purpose Model Visualization Environment*. 8th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, 11 2007 (Bezugnahme / zitiert auf Seite 9)

[MMC09] MÉSZÁROS, Tamás ; MEZEI, Gergely ; CHARAF, Hassan: Engineering the Dynamic Behavior of Metamodeled Languages. In: *Simulation, Special Issue on Multi-Paradigm Modeling* 89 (2009), S. 793–810 (Bezugnahme / zitiert auf Seite 10)

[MML08] MÉSZÁROS, Tamás ; MEZEI, Gergely ; LEVENDOVSZKY, Tihamér: A flexible, declarative presentation framework for domain-specific modeling. In: *AVI '08: Proceedings of the working conference on Advanced visual interfaces*. New York, NY, USA : ACM, 2008. – ISBN 1–978–60558–141–5, S. 309–312 (Bezugnahme / zitiert auf Seite 10)

[MRAK03] MEISTER, Jürgen ; ROHDE, Martin ; APPELRATH, Hans-Jürgen ; KAMP, Vera: Data-Warehousing im Gesundheitswesen. In: *it - Information Technology* 45 (2003), Nr. 4, S. 179–185 (Bezugnahme / zitiert auf Seite 3)

[MTA09] MERTENS, Matthias ; TEIKEN, Yvette ; APPELRATH, Jürgen: Semantische Anreicherung von strukturierten Daten und Prozessen in analytischen Informationssystemen am Beispiel von MUSTANG. In: BAARS, Henning (Hrsg.) ; RIEGER, Bodo (Hrsg.): *CEUR Workshop Proceedings* Bd. 542. Technische Universität Dortmund, Deutschland, Oktober 2009 (Forschungskolloquium Business Intelligence 2009 der GI Fachgruppe 5.8, Perspektiven der betrieblichen Management- und Entscheidungsunterstützung). – ISSN 1613–0073, S. 53–67 (Bezugnahme / zitiert auf Seite 15)

[NS00] NORTH, Chris ; SHNEIDERMAN, Ben: Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In: *AVI '00: Proceedings of the working conference on Advanced visual interfaces*. New York, NY, USA : ACM, 2000. – ISBN 1–58113–252–2, S. 128–135 (Bezugnahme / zitiert auf Seite 21)

[OGP09] OGP SURVEYING AND POSITIONING (Hrsg.): *Coordinate Conversions and Transformations including Formulas*. OGP Surveying and Positioning, November 2009 (Bezugnahme / zitiert auf Seite 68)

[OMG09] OMG: *Object Constraint Language*. 05 2009. – Online verfügbar auf `http://www.omg.org/spec/OCL/2.1/Beta2/PDF`; abgerufen am 02.12.2009 (Bezugnahme / zitiert auf Seite 12)

[Pan05] PANSE, Christian: *Visualizing Geo-Related Data Using Cartograms*. Universitätsstr. 10, 78457 Konstanz, Universität Konstanz, Diss., 2005 (Bezugnahme / zitiert auf Seite 40)

[Pen08] PENDSE, Nigel: *What is OLAP?* „Website“, 03 2008. – Online verfügbar auf `http://www.bi-verdict.com/fileadmin/FreeAnalyses/fasmi.htm`; abgerufen am 04.02.2010 (Bezugnahme / zitiert auf Seite 16)

[Pet05] PETERSOHN, Helge: *Data Mining: Verfahren, Prozesse, Anwendungsarchitektur*. Oldenbourg Wissenschaftsverlag GmbH, 2005 (Bezugnahme / zitiert auf Seite 120)

[PHP03] PFITZNER, Darius ; HOBBS, Vaughan ; POWERS, David M. W.: A Unified Taxonomic Framework for Information Visualization. In: PATTISON, Tim (Hrsg.) ; THOMAS, Bruce H. (Hrsg.): *InVis.au* Bd. 24, Australian Computer Society, 2003 (CRPIT). – ISBN 1–920682–03–1, S. 57–66 (Bezugnahme / zitiert auf Seite 39)

[PJ01] PEDERSEN, Torben B. ; JENSEN, Christian S.: Multidimensional Database Technology. In: *ACM Comput. Surv.* 34 (2001), S. 40–46. – ISSN 0018–9162 (Bezugnahme / zitiert auf Seite 17)

[Pri00] PRISNER, Erich: *Geordnete Mengen*. „Website“, 02 2000. – Online verfügbar auf `http://www.math.tu-cottbus.de/INSTITUT/lsgdi/DM/Ordnung.html`; abgerufen am 29.10.2009 (Bezugnahme / zitiert auf Seite 42)

[RFF+08] ROBERTSON, George ; FERNANDEZ, Roland ; FISHER, Danyel ; LEE, Bongshin ; STASKO, John: Effectiveness of Animation in Trend Visualization. In: *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), Nr. 6, S. 1325–1332. – ISSN 1077–2626 (Bezugnahme / zitiert auf den Seiten 46 und 47)

[Rhe02] RHEINGANS, Penny: Are We There Yet? Exploring With Dynamic Visualization. In: *IEEE Computer Graphics and Applications* 22 (2002), Nr. 1, S. 6–10. – ISSN 0272–1716 (Bezugnahme / zitiert auf Seite 26)

[RM90] ROTH, Steven F. ; MATTIS, Joe: Data characterization for intelligent graphics presentation. In: *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 1990. – ISBN 0–201–50932–6, S. 193–200 (Bezugnahme / zitiert auf den Seiten 39, 40, 41, und 42)

[SBC97] SHNEIDERMAN, Ben ; BYRD, Don ; CROFT, W. B.: Clarifying Search: A User-Interface Framework for Text Searches. Corporation for National Research Initiatives, 1997. – Forschungsbericht (Bezugnahme / zitiert auf Seite 39)

[SBM08a] SIMOFF, Simeon J. ; BÖHLEN, Michael H. ; MAZEIKA, Arturas: Visual Data Mining: An Introduction and Overview. In: *V*isual Data Mining[SBM08b], S. 1–12 (Bezugnahme / zitiert auf Seite 45)

[SBM08b] SIMOFF, Simeon J. (Hrsg.) ; BÖHLEN, Michael H. (Hrsg.) ; MAZEIKA, Arturas (Hrsg.): *Lecture Notes in Computer Science*. Bd. 4404: *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics (Lecture Notes in Computer Science)*. Springer, 2008. – ISBN 978–3–540–71079–0 (Bezugnahme / zitiert auf den Seiten 138 und 140)

[Sch07] SCHNEIDEWIND, Jörn: *Scalable Visual Analytics : Solutions and Techniques for Business Applications*. Universitätsstr. 10, 78457 Konstanz, Universität Konstanz, Diss., 2007 (Bezugnahme / zitiert auf Seite 1)

[Shn92] SHNEIDERMAN, Ben: Tree visualization with tree-maps: 2-d space-filling approach. In: *ACM Trans. Graph.* 11 (1992), Nr. 1, S. 92–99. – ISSN 0730–0301 (Bezugnahme / zitiert auf Seite 31)

[Shn96] SHNEIDERMAN, Ben: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In: *VL*, 1996, S. 336–343 (Bezugnahme / zitiert auf den Seiten 4, 26, und 39)

[SLA+09] SANTOS, Emanuele ; LINS, Lauro ; AHRENS, James ; FREIRE, Juliana ; SILVA, Claudio: VisMashup: Streamlining the Creation of Custom Visualization Applications. In: *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), Nr. 6, S. 1539–1546. – ISSN 1077–2626 (Bezugnahme / zitiert auf Seite 22)

[SN07] SCHULZ, Hans-Jörg ; NOCKE, Thomas: Maschinelle Datenanalyse im Informationszeitalter - Können oder müssen wir ihr vertrauen? In: LANGNER, Ronald (Hrsg.) ; LUKS, Timo (Hrsg.) ; SCHLIMM, Anette (Hrsg.) ; STRAUBE, Gregor (Hrsg.) ; THOMASCHKE, Dirk (Hrsg.): *Ordnungen des Denkens - Debatten um Wissenschaftstheorie und Erkenntniskritik*. Münster : LIT-Verlag, 2007 (Verhandlungen mit der Gegenwart). – ISBN: 3-8258-0358-2 (Bezugnahme / zitiert auf den Seiten 1 und 3)

[Sob09] SOBOLEWSKI, Meike: *Epidemiologie*. „Website“, 05 2009. – Online verfügbar auf `http://www.bfs.de/de/ion/wirkungen/epidem.html`; abgerufen am 14.12.2009 (Bezugnahme / zitiert auf Seite 3)

[Tho09] THOBEN, Wilfried: *Epidemiologisches Krebsregister Niedersachsen*. „Website“, 06 2009. – Online verfügbar auf `http://www.krebsregister-niedersachsen.de`;

abgerufen am 14.12.2009 (Bezugnahme / zitiert auf Seite 3)

[TIC09] TOBIASZ, Matthew ; ISENBERG, Petra ; CARPENDALE, Sheelagh: Lark: Coordinating Co-located Collaboration with Information Visualization. In: *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), Nr. 6, S. 1065–1072. – ISSN 1077–2626 (Bezugnahme / zitiert auf den Seiten 24 und 26)

[TK04] TOLVANEN, Juha-Pekka ; KELLY, Steven: Domänenspezifische Modellierung. In: *Objekt Spektrum* 04 (2004), S. 30–34 (Bezugnahme / zitiert auf Seite 7)

[TPM05] TORY, Melanie ; POTTS, Simeon ; MÖLLER, Torsten: A Parallel Coordinates Style Interface for Exploratory Volume Visualization. In: *IEEE Transactions on Visualization and Computer Graphics* 11 (2005), Nr. 1, S. 71–80. – ISSN 1077–2626 (Bezugnahme / zitiert auf Seite 26)

[TSB04] TANG, Diane ; STOLTE, Chris ; BOSCH, Robert: Design choices when architecting visualizations. In: *Information Visualization* 3 (2004), Nr. 2, S. 65–79. – ISSN 1473–8716 (Bezugnahme / zitiert auf den Seiten 21, 22, und 25)

[Tuk77] TUKEY, John W.: *Exploratory data analysis*. Addison-Wesley, 1977 (Bezugnahme / zitiert auf Seite 1)

[VC09] VOELTER, Markus ; CONELIUSSEN, Lars: Domänenspezifische Sprachen und Microsoft Oslo. In: *dot.net Magazin* 03 (2009) (Bezugnahme / zitiert auf den Seiten 7 und 8)

[VMT08] Department of Automation and Applied Informatics: *Department of Automation and Applied Informatics - VMTS*. „Website“, 2008. – Online verfügbar auf `http://www.aut.bme.hu/portal/Vmts.aspx`; abgerufen am 02.12.2009 (Bezugnahme / zitiert auf Seite 8)

[VPF06] VALIATI, Eliane R. A. ; PIMENTA, Marcelo S. ; FREITAS, Carla M. D. S.: A taxonomy of tasks for guiding the evaluation of multidimensional visualizations. In: *BELIV '06: Proceedings of the 2006 AVI workshop on BEyond time and errors*. New York, NY, USA : ACM, 2006. – ISBN 1–59593–562–2, S. 1–6 (Bezugnahme / zitiert auf den Seiten 39 und 42)

[WBJ03] WENZEL, Sigrid ; BERNHARD, Jochen ; JESSEN, Ulrich: Visualization for modeling and simulation: a taxonomy of visualization techniques for simulation in production and logistics. In: CHICK, Stephen E. (Hrsg.) ; SANCHEZ, Paul J. (Hrsg.) ; FERRIN, David M. (Hrsg.) ; MORRICE, Douglas J. (Hrsg.): *Winter Simulation Conference*, ACM, 2003. – ISBN 0–7803–8132–7, S. 729–736 (Bezugnahme / zitiert auf den Seiten 39, 43, und 47)

[Wie08] WIEKEN, John-Harry: *SQL - Einstieg für Anspruchsvolle*. Pearson Studium, 2008 (Bezugnahme / zitiert auf Seite 41)

[WK03] WARMER, Jos ; KLEPPE, Anneke: *The Object Constraint Language: Getting Your Models Ready for MDA*. Second Edition. Addison Wesley, 2003 (Bezugnahme / zitiert auf Seite 12)

[WL90] WEHREND, Stephen ; LEWIS, Clayton: A problem-oriented classification of visualization techniques. In: *VIS '90: Proceedings of the 1st conference on Visualization '90*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1990. – ISBN 0–8186–2083–8, S. 139–143 (Bezugnahme / zitiert auf den Seiten 26, 39, 42, und 47)

[YKSJ07] YI, Ji S. ; KANG, Youn a. ; STASKO, John ; JACKO, Julie: Toward a Deeper Understanding of the Role of Interaction in Information Visualization. In: *IEEE Transactions on Visualization and Computer Graphics* 13 (2007), Nr. 6, S. 1224–1231. – ISSN 1077–2626 (Bezugnahme / zitiert auf den Seiten 45 und 46)

Index

R

S

T

V

Z

Versicherung

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, den 19. April 2010

B.Sc. Arne Uphoff