UNIVERSITÄT PADERBORN

Die Universität der Informationsgesellschaft

Fakultät für Elektrotechnik, Mathematik und Informatik
Institut für Informatik

Spezifikation von Information Retrieval Prozessen und Realisierung eines Datenvisualisierungsmoduls

Diplomarbeit

Frank Benteler

Paderborn, im Juli 2006

Gutachter:
Prof. Dr. Benno Stein
Prof. Dr. Hans Kleine Büning

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Paderborn, im Juli 2006 Frank Benteler

Inhaltsverzeichnis

1. Einleitung

Menschliches Wissen ist hauptsächlich als Text gespeichert und wird meistens auch als Text weitergegeben. Die Techniken zum Speichern und Durchsuchen von Textdokumenten sind beinahe so alt wie die geschriebene Sprache selbst. In traditionellen Bibliotheksindexen kann auf Dokumente nur durch eine kleine Anzahl von Indexbedingungen wie Titel, Autor und einigen Themenüberschriften zugegriffen werden. Mit automatisierten Systemen ist die Anzahl der Indizierungsbedingungen, die für ein Dokument verwendet werden können, praktisch grenzenlos.

Das Teilgebiet der Informatik, das sich mit der automatisierten Lagerung und Analyse von Dokumenten befasst, wird Information Retrieval (IR) genannt. Automatisierte Information Retrieval- (IR) Systeme wurden ursprünglich entwickelt, um die riesige Menge an wissenschaftlicher Literatur zu verwalten, die sich seit den 1940ern entwickelt hat. Viele Universitäts und öffentliche Bibliotheken verwenden heute IR Systeme, um Zugriff auf Bücher, Journale und andere Dokumente zu liefern. Kommerzielle IR Systeme bieten Datenbanken an, die Millionen von Dokumenten in unzähligen Sachgebieten enthalten. Wörterverzeichnisse und Enzyklopädien sind allgemein verfügbar.

Die Informationen in diesen Datenbanken werden normalerweise in einem vordefinierten Format gespeichert, um das Durchsuchen und die Analyse der Informationen zu erleichtern. Im wirklichen Leben jedoch kommt auf jede Quelle strukturierter Information, wie z.B. einer Datenbank von Einkaufsaufzeichnungen, eine Vielzahl von Quellen unstrukturierter Information, wie z.B. natürlich sprachliche Dokumente, Bilder und Videodaten von den verkauften Produkten. Es wird geschätzt, dass 80 bis 85 Prozent aller Informationen unstrukturiert sind, und somit zunächst ungeeignet für automatische Auswertungen. Eine stärkere Strukturierung ist Angesichts des rasanten Wachstum des Internets und der firmen internen Netzen (Intranets) nicht zu erwarten.

Diese enorme Menge an Informationen verfügbar zu machen ist Aufgabe des Information Retrieval (IR), dessen Bedeutung durch das Internet für jeden Nutzer begreifbar geworden ist: Die prinzipielle Verfügbarkeit von Informationen ist keineswegs gleichbedeutend damit ist, dass sie auch mit vertretbarem Aufwand gefunden werden kann. Das Problem, relevante Dokumente aus einer unübersehbaren Zahl ähnlichem, aber aus Sicht des Suchenden nutzlosem „Datenmüll“ herauszufiltern, ist vielschichtig und wird von den klassischen Werkzeugen (Suchmaschinen, Katalogen) nur mit Einschränkungen gemeistert.

Ziel dieser Diplomarbeit ist die Spezifikation eines Prozesses zur Lösung von Information Retrieval Problemen. Der Prozess soll die Möglichkeit bieten die ständig wiederkehrenden Aufgabenstellungen schnell lösen zu können.

Dazu sollen zunächst vorhandene Spezifikationen und Implementierungen im Bereich IR heraus gesucht, kurz vorgestellt und in ein noch zu entwickeltes Schema eingeordnet werden. Darauf aufbauend soll ein eigener Prozess entwickelt und dessen Realisierbarkeit an Hand eines Datenvisualisierungsmoduls aufgezeigt werden.

Die vorliegende Arbeit gliedert sich in sechs Kapitel auf, von denen das erste Kapitel diese Einleitung bildet und die Kapitel zwei und drei die vorhanden Spezifikationen bzw. Implementierungen vorstellen. Das vierte Kapitel erläutert den entwickelten Prozess, und die Realisierung der Baumvisualisierungen stellt das Kapitel fünf dar. Die anschließende Diskussion bildet das Kapitel sechs.

2. Spezifikationen für Information Retrieval Aufgaben

Information Retrieval kommt auf vielen verschiedenen Arten und Weisen und in Kombination mit vielen verschiedenen Systemen zum Einsatz. Alle diese Standards wurden entwickelt, um einer speziellen Anforderung gerecht zu werden; dies ist der Grund, warum so viele verschiedene Standards existieren.

Eine kleine Auswahl dieser Standards wird im folgenden näher betrachtet. CRISP-DM und PMML stellen Spezifikationenen von Data Mining Prozessen dar und CWM, SQL/MM, OLE DB, MDIS und OIM bestimmen (Meta-) Datentypen die zum Austausch von Modellen und Daten zwischen verschiedenen Systemen benutzt werden können. Die Betrachtung einer Spezifikation umfasst jeweils die folgenden Teile:

- Vollständiger Name; Herausgeber bzw. veröffentlichende Organisation; Geschichtliche Betrachtung bzw. besondere Zeitpunkte in der Entwicklung
- Zielrichtung (im Sinne von Domäne, Aufgabe)
- Komponenten
- Professionalität bzw. Größenordnung

Bei einigen Standards liefern weitere Teile Besonderheiten der jeweiligen Standards.

2.1. Prozesse

Unter einem Prozess (lat. procedere = voranschreiten) versteht man allgemein [1] eine dynamische Aufeinanderfolge von verschiedenen Zuständen eines Systems. Nach ISO 8402 besteht der Prozess aus einer Menge von Mitteln und Tätigkeiten. Zu den Mitteln können Personal, Geldmittel, Anlagen, Einrichtungen, Techniken und Methoden gehören. Ein Prozess erfordert Eingaben und gibt Ergebnisse aus. Bezogen auf die Informatik, beschreibt ein Prozess die Methoden und Zustände eines Softwaresystems. Die zwei Prozess Spezifikationen PMML und CRISP-DM sind im folgenden einmal dargestellt.

[1] siehe auch `http://de.wikipedia.org/wiki/Prozess`

2.1.1. PMML

Die *Predictive Model Markup Language (PMML)* wird seit 1998 von der Data Mining Group (DMG) veröffentlicht und hat seit dem ständige Verfeinerungen erfahren; Eine Version 0.7 wurde von dem National Center for Data Mining (NCDM) an der University of Illinois in Chicago entwickelt und im Juli 1997 veröffentlicht. Auf der Supercomputing 1998 wurde eine Vielzahl von Anwendungen demonstriert, die PMML Version 0.9 einsetzen. Die Version 1.0 entwickelten Angoss, Magnify, NCR, SPSS und NCDM. IBM trat der DMG im Jahr 1999 bei, gefolgt von Microsoft und Oracle im Jahr 2000. Die aktuelle Version 3.1[2] ist seit Dezember 2005 verfügbar. Im April 2005 fanden sich unter den Full Members der DMG unter anderem Firmen wie Corp; KXEN; Magnify Inc; Microsoft; MicroStrategy Inc.; National Center for Data Mining, University of Illinois at Chicago; Oracle Corporation; Prudential Systems Software; Salford Systems; SAS Inc; SPSS Inc und StatSoft, Inc. sowie die Associate Members Angoss Software Corp; Insightful Corp; NCR Corp; Quadstone; Urban Science und SAP.

Zielrichtung

Wie in [Raspl04] beschrieben ist PMML ein anwendungs- und systemunabhängiges XML-basierendes Austauschformat für statistische und Data Mining Modelle. Genauer gesagt, kapselt PMML ein Modell in einem anwendungs- und systemunabhängigen Format, so dass zwei verschiedene Anwendungen (der PMML Erzeuger und Abnehmer) es benutzen können. Ein Hauptziel von PMML ist es, Anwendungen und Online-Analysten mit Modellen von verschiedenen Quellen arbeiten zu lassen, ohne sich um die einzelnen Unterschiede zwischen jenen Quellen zu beschäftigen. Durch die PMML-Schnittstelle ist das Analyseprojekt nicht mehr an ein Data Mining Tool gebunden.

Komponenten

PMML besteht aus den folgenden Komponenten:

1. **Data Dictionary**. Das Data Dictionary definiert die Eingabefelder für Modelle und spezifiziert den Typ und den Wertebereich für jedes Feld.

2. **Mining Schema**. Jedes Modell besitzt ein Mining Schema, welches die im Modell benutzten Felder auflistet. Diese Felder sind eine Teilmenge der Felder des Data Dictionary. Das Mining Schema beinhaltet modellspezifische Informationen, während das Data Dictionary Datendefinitionen beinhaltet, die sich nicht mit dem Modell ändern. Zum Beispiel kann ein Mining Schema den Benutzungstyp eines eines Attributes festlegen, der activ (eine Eingabe des Modells), predicted (eine Ausgabe des Modells) oder supplementary (enthält beschreibende Informationen und werden vom Modell ignoriert).

[2] `http://www.dmg.org/pmml-v3-1.html`

3. **Transformation Dictionary**. Ein Transformation Dictionary definiert abgeleitete Felder. Abgeleitete Felder können definiert werden durch Normalisierung (Abbildung von kontinuierlichen oder diskreten Werten in ganze Zahlen), Diskretisierung (Abbildung von kontinuierlichen in diskrete Werte), Werteabbildung (Abbildung von diskreten in diskrete Werte) oder durch Aggregation (Summenbildung oder andere Gruppierung von Werten, wie z.B. Mittelwerte).

4. **Statistics**. Die statistische Modellkomponente beinhaltet grundsätzliche Statistiken über das Modell, wie das Minimum, Maximum, Durchschnitt, Standardabweichung, Median, etc. von numerischen Werten.

5. **Modell Parameter**. PMML spezifiziert Parameter, die das statistische und das Data Mining Modell definieren. In PMML Version 3.1 werden die folgenden Klassen von Modellen abgedeckt:

   - regression models,
   - cluster models,
   - trees,
   - neural networks,
   - Bayesian models,
   - association rules,
   - sequence models,
   - support vector machines,
   - rule sets, and
   - text models.

Seit PMML Version 3.0 besteht die Möglichkeit verschiedene Data Mining Operationen zusammenzusetzen. Zum Beispiel kann die Ausgabe eines Regressionsmodells als Eingabe für ein anderes Modell herangezogen werden (sequence models) und ein Entscheidungsbaum oder Regressionsmodell kann die Ausgaben von anderen eingebetteten Modellen kombinieren (model selection).

Das PMML Version 3 Text Modell besteht aus folgenden Komponenten:

1. Wörterbuch von Bedingungen oder Textwörterbuch, das die Bedingungen des Modells enthält.

2. Korpus von Textdokumenten: Dieses Element identifiziert die tatsächlichen Texte, die von diesem Modell abgedeckt werden. Nur Verweise werden gegeben, nicht die tatsächlichen Texte.

3. Dokument-Bedingungs-Matrix: Dieses Element gibt an, welche Bedingungen bei welchem Dokument verwendet werden.

4. Textmodellnormalisierung: Dieses Element definiert eine von mehreren möglichen Normalisierungen der Dokument-Bedingungs-Matrix.

5. Textmodellähnlichkeit: Dieses Element definiert die Ähnlichkeit, die verwendet wird, um zwei Dokuments in Vektordarstellung zu vergleichen.

PMML ist formal als ein W3C XML Schema definiert. XML Schemata definieren die Struktur der XML Dokumente, in denen ein oder mehrere Modell(e) enthalten seinen können. Ein PMML Dokument ist ein XML Dokument, mit einem Wurzelelement vom Type PMML. Es beschreibt die Eingabe in ein Data Mining Modell, die Transformationen die Daten für das Data Mining vorbereiten und die Parameter die das Modell selbst beschreiben. Programme die PMML benutzen stammen aus Bereichen wie Finanzen, e-Business, Direkt-Marketing, Produktion und Militär.

Professionalität

Die KDD-2004 Online-Proceedings berichten das der im August 2004 stattgefundene DM-SSP Workshop „ markiert das vierte Jahr, in dem es einen KDD Workshop über Predictive Model Markup Language (PMML) und angrenzende Bereiche gegeben hat und das zweite Jahr einer breiteren Konferenz über das Thema von Data-Miningstandards, Services und Plattformen. Eines der Ziele von PMML war, eine Standardschnittstelle zwischen Erzeugern von Modellen zu schaffen, wie statistische oder Data-Mining Systeme, und Verbrauchern von Modellen, wie Bewertungssysteme, Anwendungen mit eingebetteten Modellen und andere operative Systeme. Es gibt viele Hersteller, die Bewertungssysteme anbieten, welches ist ein wichtiges Maß des Erfolgs in diesem Bereich ist. In den vergangenen Jahren haben die Entwickler von PMML an einem ähnlichen Mechanismus gearbeitet, so dass die in der Datenverarbeitung erforderlichen Transformationen und Kompositionen, die so wesentlich für das Data-Mining sind, ebenso gekapselt werden können. “

Seit der PMML Version 3.0 wird die Spezifikation als reifer Standard bezeichnet, so dass der Einsatz durch die Erzeugung von PMML Bewertungssystemen problemlos ist. Für folgende Versionen wird das Entwicklungsteam damit fortfahren neue statistische und Data-Mining Modelle hinzuzufügen. Sie planen die Unterstützung der Datenaufbereitung zu erweitern, was immer noch für einige Aufgaben eine mühselige Angelegenheit ist.

Softwarepakete die PMML einsetzen[3] stammen unter anderem von IBM, Salford Systems, SAS und SPSS.

2.1.2. CRISP-DM

Das Projekt *CRoss-Industry Standard Process for Data Mining*[4] wurde im Juli 1997 offiziell mit der Bildung eines Konsortiums initiiert. Der Großteil der Entwicklung wurde

[3]von `http://xml.coverpages.org/ni2005-05-27-a.html`
[4]CRISP-DM Referenz Modell: `http://www.crisp-dm.org`

von Daimler Crysler, NCR, SPSS und der niederländischen Versicherung OHRA geleistet. Das Konsortium umfasst weltweit ca. 200 Mitglieder aus verschiedenen Bereichen. Bis Mitte 1999 wurde das Projekt von der Europäischen Kommission im Rahmen des ESPRIT-Programms zur Förderung von technologischen Entwicklungen in Europa teilweise subventioniert. Das CRISP-DM Referenzmodell 1.0 wurde im August 2000 fertig gestellt.

Zielrichtung

Das aktuelle Prozessmodell für Data Mining bietet eine Übersicht über den Lebenszyklus eines Data Mining Projektes. Es beinhaltet die entsprechenden Phasen eines Projektes, ihrer entsprechenden Aufgaben und Beziehungen zwischen diesen Aufgaben. Auf diesem Beschreibungslevel ist es nicht mögliche alle Beziehungen zu erkennen. Möglicherweise existieren Beziehungen zwischen allen Data Mining Aufgaben, abhängig von Zielen, Hintergrund und Interesse des Benutzers und von den Daten selbst.

Der Fokus dieses Data Mining-Prozessmodells liegt weniger auf dem technischen, sondern mehr auf dem wirtschaftlichen Bereich.

Komponenten

Ein Data Mining-Prozess ist (nahezu) nie ein linearer Prozess, in dem die Schritte in der dargestellten Reihenfolge abgearbeitet werden, sondern stellt sich in der Praxis als ein dynamischer Prozess dar, bei dem Befunde in einer bestimmten Phase oder Aufgabe den Analytiker dazu zwingen können, zu vorherigen Phasen oder Aufgaben zurückzukehren. Die Abbildung 2.1 stellt diesen Zusammenhang graphisch dar, wobei die Pfeile die wichtigsten und häufigsten Abhängigkeiten zwischen den Phasen abbilden.

Ein Data Mining Prozess geht weiter, nachdem eine Lösung entwickelt wurde. Das Wissen, dass während des Prozesses erlernt wurde, kann neue, oft genauere Wirtschaftsfragen auswerfen. Zyklische Data Mining Prozesse profitieren von den Erfahrungen der vorhergehende Durchläufe. Der äußere Kreis in der Abbildung 2.1 symbolisiert die zyklische Natur des Data Mining selbst.

Nun folgt eine kurze Übersicht über die verschiedenen Phasen:

- **Business Understanding**. Diese initiale Phase fokussiert auf dem Verstehen und Erkennen der Anforderungen und Ziele des Projekts von einer gewerblichen Perspektive aus. Dieses Wissen wird in eine Data Mining Problemdefinition umgesetzt und ein vorläufiger Plan zur Erfüllung der Ziele erstellt.

- **Data Understanding**. Diese Phase soll ein Verständnis für die verwendeten Daten aufbauen und startet mit einer initialen Datensammlung. Daran schließen sich Aktivitäten an, die die Qualität der Daten identifizieren, erste Einsichten für die Daten erschließen und die Teilmengen der Daten ausfindig machen sollen. Hieraus werden dann Vermutungen über versteckte Informationen gewonnen.

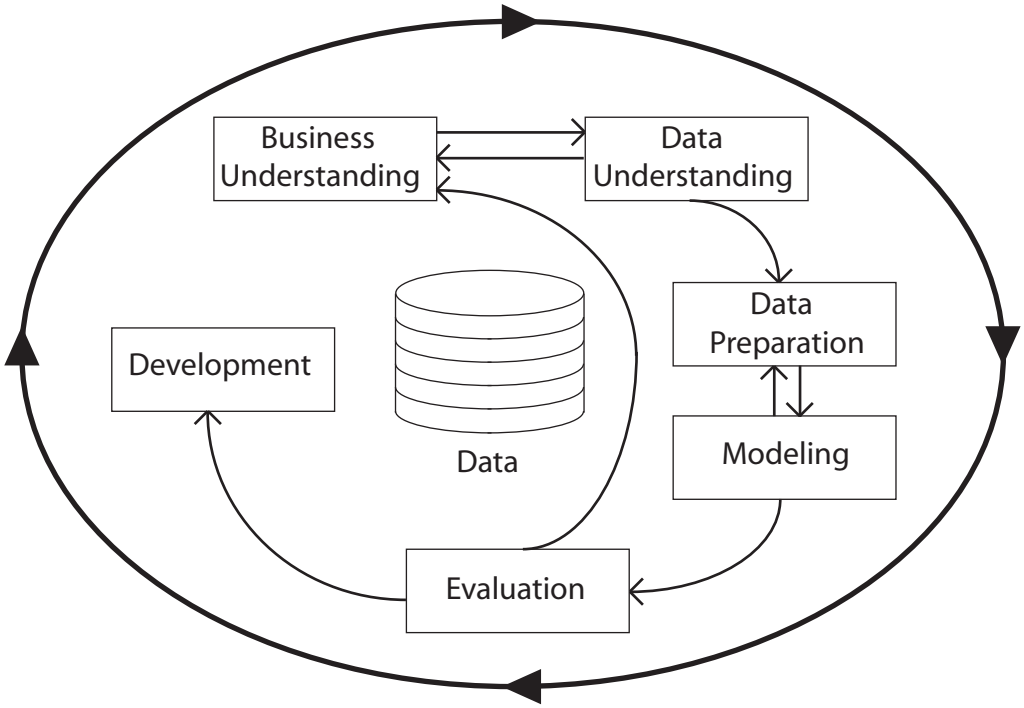

Abbildung 2.1.: Phasen des CRISP-DM Prozess Modell

- **Data Preparation**. Die Datenvorbereitungsphase umschließt alle Aktivitäten zur Erstellung der entgültigen Datenmenge (Daten, die den Modellierungstools zugeführt werden) aus den initialen Rohdaten. Datenvorbereitungsaufgaben werden meist oft durchgeführt und nicht unbedingt in einer vorgeschriebenen Reihenfolge. Aufgaben umfassen die Selektion von Tabellen, Attribute und Datensätzen, so wie die Transformation und Säuberung der Daten für Modellierungstools.

- **Modeling**. In der Modellierungsphase werden verschiedene Modellierungstechniken ausgewählt und angewendet, und deren Parameter optimal kalibriert. Typischer Weise gibt es mehrere Techniken für ein und den selben Data Mining Problemtyp. Einige Techniken haben spezielle Anforderungen an die Form der Daten. Ein Rückschritt in die Datenvorbereitungsphase findet häufig statt.

- **Evaluation**. In dieser Phase des Projektes existiert ein Modell (oder mehrere), das eine hohe Qualität zu haben scheint, jedenfalls von der Datenanalyse Perspektive aus. Bevor die nächste Phase beginnt, ist es wichtig, das Modell genauer zu evaluieren und die Schritte zur Erstellung des Modells zu überprüfen, um sicher sein zu können, dass es tatsächlich die Ziele erfüllt. Das Hauptziel ist die Entscheidung, ob es ein wichtiges Geschäftsereignis gibt, dass nicht genügend berücksichtigt wurde. Am Ende dieser Phase steht die Entscheidung, ob die Ergebnisse des Data Minings weiter benutzt werden sollen.

- **Deployment**. Die Erzeugung des Modells ist im allgemeinen nicht das Ende des Projektes. Sogar wenn das Modell dazu dient das Wissen über die Daten zu vergrößern, muss das Wissen aufbereitet und präsentiert werden, damit der Kunde es verwenden kann. Abhängig von den Anforderungen kann diese Phase einfach sein, wie die Erstellung eines Reports, oder komplex, wie das Implementieren eines sich wiederholenden Data Mining Prozesses. In vielen Fällen wird es der Kunde sein (nicht derjenige, der die Daten analysiert) der diese Phase ausführt. Wie auch immer, selbst wenn der Analyst nichts mit dieser Phase zu tun hat, ist es dennoch wichtig für den Kunden was in erster Linie durchgeführt werden muss, um tatsächlichen Nutzen von den erstellen Modellen zu erzielen.

Professionalität

Das Prozessmodell umfasst den ganzen Zyklus eines Data Mining-Prozesses. Jede der sechs Phasen besteht aus Aufgaben und Outputs und ist vollständig und generisch (d.h. sie sind unabhängig vom spezifischen Business- und Data Mining Problem). Die verwendeten Data Mining Techniken sind zudem stabil in Bezug auf zukünftige Entwicklungen im Data Mining (wie etwa neue Modellierungstechniken).

2.2. Austauschformate für Modelle und Daten

Ein Austauschformat bezeichnet meist ein Dateiformat, welches mit vielen Anwendungen auf verschiedenen Betriebssystemen kompatibel ist. Die ersten Austauschformate wurden sehr einfach und primitiv gehalten. Zum Beispiel ist das *.txt-Format mit reinen ASCII-Zeichen für Texte auch heute noch ein wichtiges Format, will man sicher gehen, das der Empfänger eines Textes diesen auch lesen kann.

Die im folgenden gezeigten Datenaustauschformate sind aus der Notwendigkeit heraus entwickelt worden Modelle und Daten über verschiedene heterogene Plattformen hinweg übertragen zu können. Sie sind nicht an eine spezielle Programmiersprache oder Rechnerplattform gebunden.

2.2.1. CWM

Das *Common Warehouse Metamodel (CWM)*[5] der Object Management Group (OMG) wurde im April 2000 in der Version 1.0 vorgestellt.

Die Object Management Group ist ein 1989 gegründetes Konsortium, das sich mit der Entwicklung von Standards für die herstellerunabhängige systemübergreifende objektorientierte Programmierung beschäftigt. Der OMG gehörten zur Gründung 11 Firmen, darunter IBM, Apple und Sun an, mittlerweile hat sie über 800 Mitglieder und die geschaffenen Standards sind international anerkannt.

Die bekanntesten Entwicklungen der OMG sind die Common Object Request Broker Architecture (CORBA), die das Erstellen von Verteilten Anwendungen in heterogenen

[5] `http://www.omg.org/cwm/`

Umgebungen vereinfacht, sowie die Unified Modeling Language (UML), die die Modellierung und Dokumentation von Objektorientierten Systemen in einer normierten Syntax erlaubt. Weitere OMG Standards sind unter anderem die Model Driven Architecture (MDA), XML Metadata Interchange (XMI) und die Meta Object Facility (MOF).

Zielrichtung

Die angesammelten Daten eines Unternehmens sind eine seiner bedeutsamsten Güter geworden. Die Analyse dieser Daten ermöglicht nicht nur, dass Verkäufe und Produktion für maximale Rentabilität eingestellt werden, sondern ermöglicht auch, dass ganz neue und profitable Produkte entdeckt und genutzt werden können. Aber es ist schwierig, die Daten in ein einzelne Datenbank zu bringen, wenn die Originaldaten über eine Vielzahl von verschiedenen Datenbanken verteilt sind, die nicht nur verschiedene Datenmodelle, sondern auch andere Metamodel

Die Metadaten Verwaltung und die Abstimmung von widersprüchlichen Metadaten, wenn Daten aus verschiedenen Quellen stammen, sind die größten Probleme denen Unternehmen gegenüberstehen, die heute mit Data Warehousing arbeiten.

OMG's Common Warehouse Metamodel liefert eine Standardlösung für dieses Problem. Es ist eine Erweiterung der Metamodellierungs Architektur der OMG für den Metadatenaustausch im Bereich Data Warehouse. Ein Data Warehouse dient dem Sammeln, Aufbereiten und Abfragen von Daten zur Unterstützung von Entscheidungen. Die beschriebenen Daten und Operationen sind Grundlage der Data Mining Verfahren und erkunden große Datenmengen explorativ.

Aufbauend auf den vorhandenen Industriestandards XML, XMI und UML beginnt das CWM durch den Aufbau eines gemeinsamen Metamodel

Entworfen um mit objektorientierten, relationalen, multidimensionalen und XML-basierten Datenbanken zu arbeiten, unterstützt das CWM Data-Mining, Transformation, OLAP, Informationsvisualisierung und andere Endbenutzerprozesse. Die Metamodel Unterstützung umfasst Data Warehouse Verwaltung, Prozess und Betrieb.

Die CWM Spezifikation wird um APIs (application programming interfaces), Austauschformate und Services erweitert, die den ganzen Lebenszyklus der Metadatenverwaltung unterstützen, einschließlich Extraktion, Transformation, Transporte, Einlagerung, Integration und Analyse. Und, Benutzer können bestimmte Integrationen durch die eingebauten Erweiterbarkeit des CWM Metamodells erreichen.

Komponenten

Das CWM ist in 5 Schichten mit insgesamt 21 Metamodellpackages eingeteilt (Abbildung 2.2), wobei die oberen Schichten auf den unteren aufbauen und diese gegebenenfalls erweitern. Jedes Package korrespondiert dabei mit einem Funktionsbereich einer typischen Data-Warehouse-Anwendung. Dabei verwendet das CWM die grundlegendsten Modellelemente von UML. Kern des Modells ist das Core-Package. Alle anderen Packages bauen auf das Core-Package auf oder erweitern es. Das CWM ist geeignet um Daten in relationalen, recordorientierten und objektorientierten Datenbanken zu modellieren,

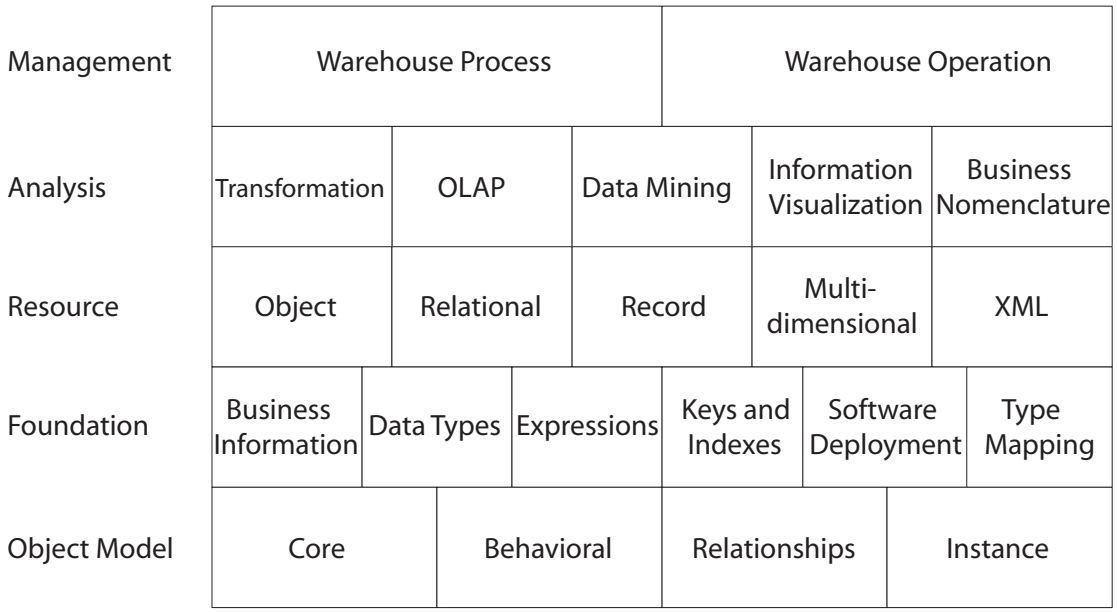

Abbildung 2.2.: Das CWM Schichtenmodell

bietet aber auch Möglichkeiten für OLAP (Online Analytical Processing), Data Mining und Informationsvisualisierung.

Zusätzlich zur Spezifikation von CWM liefert die OMG vier weitere Komponenten:

1. Das komplette Common Warehouse Metamodell in einem Rational-Rose-Modell (also in Form von UML-Diagrammen die das CWM graphisch darstellen)
2. Eine XML-Datei (die eine MOF 1.3 konforme Version des Common Warehouse Metamodells in einem XML-Dokument darstellt)
3. Eine CWM DTD-Datei, mit der Anwender die Richtigkeit eines ausgetauschten XML-Dokuments prüfen können
4. Eine IDL-Repräsentation (CORBA Interface Definition Language) des Common Warehouse Metamodells

Stellvertretend für alle anderen Packages werden hier im weiteren nur das Core-Package des Object Model und die Packages Data Mining und Information Visualization aus Analysis genauer beschrieben. Für eine genauere Betrachtung aller Packages sei auf [CWMBook2] verweisen.

CWM Core Package

Das Core-Package enthält grundlegende Klassen und Assoziationen die von allen anderen CWM-Packages verwendet werden. Es hängt nicht von anderen Packages ab. Core beinhaltet die grundlegende UML Infrastruktur, die benötigt wird um nicht-objektorientierte

Datenspeicher zu definieren, wie relationale Datenbanken und recordorientierte Dateien, ohne exklusive objektorientierte Konzepte einzuführen. Das Package enthält ebenfalls Klassen und Datentypen, die oft von anderen Packages benutzten werden.

In CWM erbt jede Klasse jedes Packages von der Klasse Element. Stellt man sich CWM in einer baumartigen Struktur von Klassen vor, so ist die Klasse Element das Wurzelelement des Baumes. Element hat keine Attribute und bietet keine Operationen an.

CWM Data Mining Package

Die Klassen des Data Mining Metamodell sind in drei Hauptbereiche gruppiert: das Core Modell, Settings und Attribute.

1. Das Core Modell repräsentiert die Ausgaben einer Data Mining Operation und beinhaltet die Klassen MiningModel, SupervisedMiningModel, MiningModelResult, MiningSetting, Application-InputSpecification und ApplicationAttribute.

2. Der Bereich Settings beschreibt Eingabeparameter und -werte von Modellattributen, die zur Konstruktion eines Mining Modells benutzt werden. MiningSettings sind eine Sammlung von Mining Modell Attributen; Fünf spezielle Typen von MiningSettings sind definiert: Statistics, Clustering, AssociationRule, Classification and Regression. Die Klasse AttributeUsageRelation bietet zusätzliche Informationen darüber wie Mining Attribute von Settings benutzt werden.

3. Der Bereich Attribute ist die Klasse MiningAttribute sorgfältig ausgearbeitet um numerische und kategorische Attribute zu ermöglichen. Kategorische Attribute sind in einer Hierarchie angeordnet und könne eine Sammlung von Eigenschaften haben, die diese Kategorie beschreiben. Die Klasse OriginalAttribute wird zum Beschreiben von Kategorien benutzt, in der die Reihenfolge der Kategorieleveln wichtig ist.

CWM Information Visualization Package

Auf Grund der großen Menge an Daten (und deren Komplexität) die in einem Data Warehouse liegen können sind die Möglichkeiten der graphischen Präsentation und Zusammenfassung essentiell für die Effektivität der Analysen über Informationen die aus dem Warehouse gesammelt wurden. In einem Warehouse gespeicherte Informationen müssen auf einer ständig steigenden Anzahl von Präsentationsstilen (Renderings) und auf sich dynamisch entwickelnde Sammlungen von traditionellen Printmedien (wie Papier und Folien) angezeigt werden können. Die Informationen müssen ebenfalls auf moderneren Medien dargestellt werden können, wie Web-Browser oder XSL-Renderings. Wegen dieser enormen Breite und Entwicklungsgeschwindigkeit in diesem Problembereich bietet das CWM ein sehr generisches Visualisierungspackage. Dessen primärer Zweck besteht darin, eine Möglichkeit zu schaffen, die Darstellung von Informationen eines jeden Typs auszutauschen und zwar für jedes Objekt in CWM. Die Klassen des Metamodells die hier definiert sind, bieten ein Framework, in dem sorgfältig ausgearbeitete

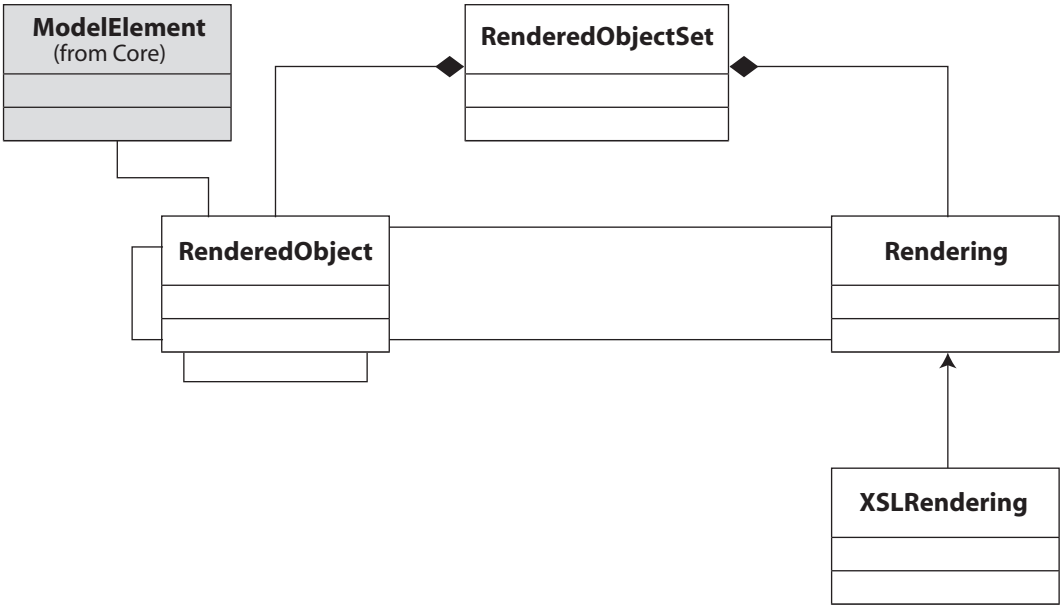

Abbildung 2.3.: Klassen des Information Visualization Metamodell von CWM

Modelle definiert werden können, um komplexe M1[6] Level Modelle auszutauschen, die auf spezielle Wiedergabeumgebungen und -tools angepasst sind. Ein Klassendiagramm des Information Visualization Metamodell ist in Abbildung 2.3 dargestellt.

Die Klasse RenderedObject fungiert als Einstig für jeder ModelElement in CWM und beinhaltet spezielle Informationen über Darstellung des Objekts, wie seine Beziehungen zu Nachbarobjekten oder seine Position auf einem Anzeigefenster. RenderedObjekts kann verschiedene Renderings referenzieren, die angeben wie die Objekte tatsächlich angezeigt werden. Renderings können als eine Art Transformation angenommen werden, die ein RenderedObjekt in ein darstellbares Objekt überführt. RenderedObjekts beschreiben die logischen Darstellungsaspekte von ModelElements, während Renderings den Darstellungtyp beschreiben, die die logischen Strukturinformationen nicht verändern.

RenderedObjets können aus einer willkürlichen Anzahl von anderen RenderedObjects zusammengesetzt werden und können andere RenderedObjects als Nachbarn referenzieren. Auf diese Weise ist es möglich komplexere RenderedObjects aus einfachen zu konstruieren. In der Tat können RenderedObjects sogar selber das Ergebnis einer komplexen Transformation sein, die mit Hilfe des Transformationspackage beschreiben werden. Die Klasse XSLRendering ist eine nützliche Unterklasse von Renderings, die XSL benutzt um HyperText Markup Language (HTML) Dokumente für Webbrowser zu erzeugen.

Professionalität

Obwohl CWM einige Kompatibilitäten mit verschiedenen anderen Standards hat, sollten diese als Ansatzpunkte für Abbildungen oder Integrationen betrachtet werden; Das Mo-

[6]Die MOF Level werden in Kapitel A.1.1 erläutert

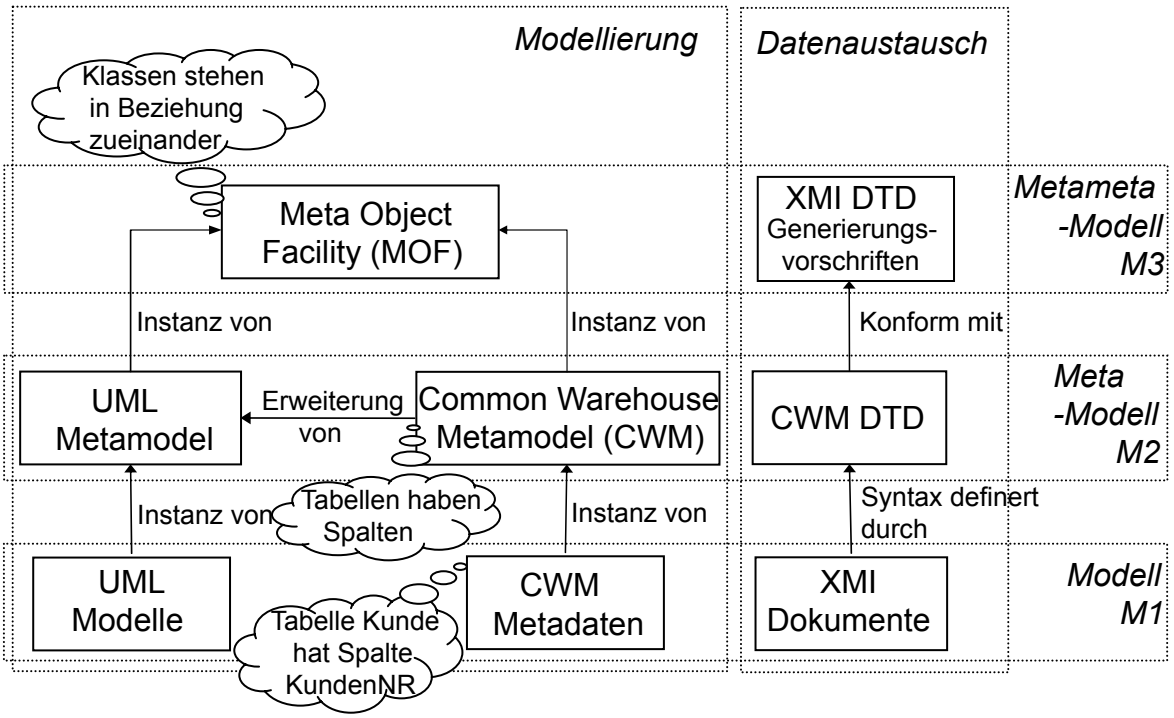

Abbildung 2.4.: Metamodellarchitektur der OMG

dell hängt nicht von anderen Standards ab, außer denen der OMG Metamodellierungs Architektur. CWM definiert weder typische Muster zum Metadatenaustausch, noch ein Protokoll (Request-Response) zum Metadatenaustausch. Im speziellen definiert CWM keine Architektur, die einen losen Austausch von DW/BI Metadaten über das Internet erlaubt.

CWM bedient sich folgender Standards der OMG:

- MOF, die Meta Object Facility, ist ein OMG Metadaten Standard, der zur Beschreibung und Veränderung von Mengen von interoperierenden Metamodellen und deren Instanzen (Modelle) benutzt werden kann. Die MOF definiert auch ein einfaches Meta-Metamodell (basierend auf UML (siehe nächsten Punkt)) mit ausreichend Semantik um Metamodelle in verschiedenen Bereichen zu beschreiben, die mit Objektanalyse und -design beginnen. CWM benutzt MOF als sein Meta-Metamodell.

- UML, die Unified Modeling Language, ist eine OMG Modellierungssprache zur Spezifikation, Entwicklung, Visualisierung und Dokumentation von Teilen eines Softwaresystems. CWM benutzt UML als seine graphische Notation und definiert sein grundlegendes Metamodell; d.h., das CWM Objektmodell ist konsistent mit dem Kern des UML Metamodells.

- XMI, der XML Metadata Interchange, ist ein OMG Mechanismus zum Stream-

basierten Austausch von MOF-kompatiblen Metamodellen. XMI ist im wesentlichen eine Abbildung der W3C's eXtensible Markup Language (XML) in die MOF. Durch die implizite MOF-Kompatibilität kann jede CWM Modellinstanz durch entsprechende Tools ausgetauscht werden, die XMI verwenden.

Zusammenfassend ist CWM ein MOF-basierendes UML-Modell, dass mit Hilfe von XMI in XML serialisiert werden kann.

Softwarepakete die CWM unterstützen stammen unter anderem von Adaptive, IBM, Hyerion, Oracle, SAS, Meta Integration, MetaMatrix und Informatica.

2.2.2. SQL/MM Part 6: Data Mining

Ein internationales ISO/IEC Standardisierungsprojekt zur Entwicklung einer SQL Klassenbibliothek für Multimediaprogramme wurde 1993 ins Leben gerufen. Der Standard mit Namen *SQL Multimedia and Application Packages (ISO 13249 - SQL/MM)* definiert Packages von SQL abstrakten Datentypen (ADT) die auf der SQL3 Spezifikation beruhen.

Der Standard ist in mehrere Teile aufgeteilt, wobei der erste Teil ein Framework beschreibt wie die restlichen Teile zu erstellen sind. Alle anderen Teile sind einem speziellen SQL Anwendungsbereich gewidmet.

1. Framework - Eine nicht-technische Beschreibung wie das Dokument strukturiert ist.

2. Volltextmethoden und abstrakte Datentypen (ADTs) für Textverarbeitung (ca. 45 Seiten).

3. Räumliche (spatial) Methoden und ADTs für räumliches Datenmanagement (ca. 200 Seiten).

4. (abandoned several years ago) Allgemeine Methoden und ADTs für komplexe Zahlen, trigonometrische und exponentielle Funktionen, Vektoren, Mengen, usw. (ca. 90 Seiten).

5. Still Image - Methoden und ADTs zur Speicherung und Bearbeitung von Bildern, wie z.B. Fotos

6. (in Bearbeitung) Data Mining - abstrakte Methoden und ADTs zur Informationsgewinnung aus großen Datenbeständen

Zielrichtung

Der sechste Teil von SQL/MM (Part 6: Data Mining) definiert benutzerdefinierte SQL-Datentypen und Methoden auf diesen Datentypen, um einen wichtigen Aspekt des modernen Datenmanagements zu adressieren: die Entdeckung von vorher unbekannten, aber wichtigen, Informationen, die in großen Datenmengen vergraben sind.

Es versucht eine standardisierte Schnittstelle für Data Mining Algorithmen bereit zu stellen, die auf einem objektrelationalen Datenbanksystem geschichtet sein können oder sich als Middleware einsetzen lässt.

Komponenten

Voraussichtlich wird es vier unterschiedliche von diesem Standard unterstützte Data Mining Techniken geben:

1. rule model
2. clustering model
3. regression model
4. classification model

Professionalität

Vier Teile von SQL/MM haben den finalen Status erreicht:

- Teil 1: SQL/MM Framework (Nov. 2002)
- Teil 2: SQL/MM Full Text (Okt. 2000)
- Teil 3: SQL/MM Spatial(Dez. 1999)
- Teil 5: SQL/MM Still Image(Mai 2001)

Der Teil 6: Data Mining ist noch nicht zu einem offiziellen Standard geworden und es scheint nicht weiter entwickelt zu werden. Eine Entwicklung auf Basis dieses Standards ist daher nicht anzuraten.

2.2.3. MDIS

Die *Meta Data Interchange Specification*(MDIS) von der Meta Data Coalition (MDC) ist ein nicht-proprietärer und erweiterbarer Mechanismus zum Datenaustausch zwischen MDIS-kompatiblen Tools. MDIS Version 1.1 besteht aus einem Metamodel Das MDIS Metamodel ist ein hierarchisch strukturiertes, semantisches Datenbank Model, dass durch eine TAG-Sprache definiert ist. Das Metamodel besteht aus einer Anzahl von generischen, semantischen Konstrukten, wie Element, Record, View, Dimension, Level und Subschema, und einer zusätzlichen Relationship entity, die benutzt werden kann, um Beziehungen zwischen willkürlichen Quellen und Zielen zu spezifizieren. Das MDIS Metamodel kann durch die Benutzung von benannten Eigenschaften erweitert werden, die Tool-spezifisch, bzw. und nicht innerhalb des MDIS definiert sind. Der Austausch wird durch eine ASCII Datei Repräsentation einer Instance des Metamodel Obwohl die Unterstützung einer API in der Spezifikation angedacht ist, wird keine API Definition geliefert. Das MDIS Access Framework spezifiziert einige sehr generelle Mechanismen, die

den Austausch von Metamodel Instanzen erlauben. Die Import und Export Funktionen, die durch das Framework beschrieben werden, sind die hauptsächlichen, dateibasierten Austauschmechanismen, die für Tools zur Verfügung stehen.

In aller Fairness sollte erwähnt werden, das MDIS ein relativ alter Standard ist, der entstanden ist bevor Technologien wie UML und XML weite Verbreitung fanden. In so fern konnte der Standard nichts von diesen Technologien benutzen als er publiziert wurde. MDIS ist ein nobler, früher Versuch einen Metadaten Austausch Standard zu definieren und es ist eine Grundlage, gegen den nachfolgende Standards verglichen werden müssen. Die MDC hat sich 1998 dazu entschieden, dass MDIS durch das Open Information Model ersetzt wird, welches als nächstes beschrieben ist.

2.2.4. OIM

Das *Open Information Model (OIM)* wurde ursprünglich und hauptsächlich von der Microsoft Corporation und Platinum Technology entwickelt. Erst später wurde das OIM an die *Meta Data Coalition (MDC)* übergeben, wo der Standard als public-domain geführt wird.

Die in 1995 gegründete Meta Data Coalition (MDC) ist ein not-for-profit Konsortium von 50 Firmen und Benutzern, unter anderem BMC, Informatica, CA, NCR, Microsoft und SAS, deren Ziel es ist eine taktische Lösung für den Metadaten Austausch zu liefern. Die Koalition brachte den Standard MDC-OIM hervor, der eine Weiterentwicklung des ursprünglich von Microsoft vorgeschlagenem Standard OIM und der Eigenentwicklung MDIS darstellt.

Die MDC-OIM Version 1.0 wurde im Juli 1999 released und benutzt UML als seine formale Spezifikationssprache und definiert allgemeine Repräsentationen von verschiedenen Typen von Datenquellen und -zielen (record, relational, OLAP) und Transformationen zwischen den Quellen und Zielen.

Zielrichtung

Das Open Information Model ist eine Zusammenstellung von Metadatenspezifikationen, die gemeinsame Nutzung und Wiederverwendung in der Anwendungsentwicklung und in Daten Warehousing Umgebungen erleichtern soll. OIM ist in der Unified Modeling Language (UML) beschrieben und benutzt weitere Industriestandards wie XML und SQL.

Komponenten

Das Open Information Model ist folgende Bereiche unterteilt

1. Analysis and Design Models
2. Objects and Components Models
3. Database and Data Warehousing Models

4. Knowledge Management Models

5. Business Engineering Models

und ist ein Kernmodel

Professionalität

Dadurch, dass zuerst ein domainspezifisches Model

Das OIM Metamodel ist vom UML Metamodel abgeleitet und die OIM Spezifikation beruft sich darauf Repository orientiert zu sein; jedoch anders als CWM ist es nicht MOF kompatible. OIM benutzt nicht XMI als Austausch Mechanismus, sondern eine eigene OIM zu XML Kodierung um Austauschdateien zu generieren. Die Extensible Markup Language (XML) ist das Standardformat für den Austausch von Metadaten, jedoch XML (an sich) definiert nicht, wie Modelle auf TAG's oder Strukturen abgebildet werden, die als XML Vokabular bekannt sind. Daher sind mit der Zeit technologieabhängige und herstellerabhängige Abbildungsvorschriften entstanden oder wurden auf dem Markt benutzt, von denen viele Industriestandards sind (BizTalk, EDI, STEP), komponentenbasierte Technologien (COM, CORBA, EJB) oder einfach nur von einem Hersteller propagiert wurde. Um diesen Problemen zu begegnen hat Microsoft zusammen mit der Koalition einen Vorschlag für eine OIM XML Kodierungsspezifikation und eine Zusammenstellung von Document Type Definitions (DTDs) herausgegeben, die es erleichtern standardisierte Import und Export zu implementieren.

Im September 2000 gaben die Meta Data Coalition (MDC) und die Object Management Group (OMG) bekannt, dass sich die MDC in die OMG eingliedert, und dass somit die beiden Standards zusammengeführt werden. Dabei herausgekommen ist das im Kapitel 2.2.1 beschriebene Common Warehouse Metamodel (CWM).

2.3. Zwischenbetrachtung

CRISP-DM ist sehr verbreitet und der Fokus dieses Data Mining-Prozessmodells liegt mehr auf dem wirtschaftlichen, als auf dem technischen Bereich. Die Modellbeschreibungssprache PMML wird von vielen führenden Unternehmen (wie IBM, Oracle, NCR, Microsoft, SAS und SPSS) in ihre Produkte integriert und bietet einen einheitlichen Standard zur einfachen Weitergabe und Nutzung der Modelle.

Die OMG propagiert mit dem Common Warehouse Metamodel und entsprechenden Vorschlägen für die Systemarchitektur einen potentiell mächtigen Standard für das Metadatenmanagement. Dabei legt die OMG explizit wert auf die Vollständigkeit der Modellierung. Dieses Kriterium erscheint deshalb so wichtig, da ein einheitlicher Standard zum Metadatenaustausch erst dann interessant wird, wenn auch möglichst viele Konzepte durch diesen Standard beschrieben werden können. Eine vollständige und ausgewogene Berücksichtigung aller Konzepte im Data Warehousing ist jedoch äußerst schwierig,

da die Anforderungen in diesem Bereich sehr stark variieren und sich ständig weiterentwickeln. Von Experten wird beispielsweise bemängelt, dass wichtige Bereiche wie Berechtigungen, Semantik von Inhalten, Qualitätsmanagement, Berichtswesen und Unterstützung organisatorischer Prozesse ausspart werden. Auch im Hinblick auf das Kriterium der Erweiterbarkeit ergeben sich Probleme. Der einfache Erweiterungsmechanismus basierend auf Stereotypes und TaggedValues ist nur sehr eingeschränkt einsetzbar. Die objektorientierte Erweiterung dagegen muss von allen an dem entsprechenden Metadatenaustausch beteiligten Anwendungen implementiert werden.

Der im Rahmen dieser Diplomarbeit entwickelte Prozess baut daher nicht direkt auf den Standards auf, sondern bedient sich einiger ihrer Konzepte.

3. Frameworks und Bibliotheken mit IR-Funktionalität

Dieses Kapitel stellt einige ausgewählte Implementierungen vor, die teilweise auf den im vorherigen Kapitel aufgeführten Spezifikationen aufsetzen bzw. diese verwenden.

Im Einzelnen sind dies die Bibliotheken WEKA und XELOPES, die Softwarepakete MetaMatrix MetaBase, SPSS Clementine, SAS Enterprise Miner, IBM Intelligent Miner und IBM WebSphere Information Integrator, sowie die Softwarearchitektur UIMA und die Java API JDM. Diese Auswahl erhebt natürlich keinerlei Anspruch auf Vollständigkeit. Es gibt viele weitere Implementierungen, auf die hier aber nicht weiter eingegangen wird.

Für jedes dieser Implementierungen wird zuerst ihr vollständiger Name und der Entwickler bzw. verbreitende Organisation genannt, danach folgt eine kurze geschichtliche Betrachtung bzw. besondere Zeitpunkte der Entwicklung. Die Zielrichtung und die einzelnen Komponenten der Implementierung bilden den Hauptteil jeder Beschreibung. Zum Abschluss wird eine kurze Betrachtung der Realisierung und der Größenordnung gegeben.

3.1. Freie Bibliotheken

3.1.1. WEKA

Die *Waikato Environment for Knowledge Analysis (WEKA)* -Bibliothek[1] ist eine populäre Data-Mining-Bibliothek im Hochschulbereich und wird von dem Machine Learning Project des Fachbereichs Informatik der Universität Waikato entwickelt.

Die erste veröffentlichte Version 3.0 ist 1999 ins Netz gestellt worden. Die aktuelle Version 3.4 ist seit Anfang 2004 verfügbar und wird ständig weiterentwickelt.

Zielrichtung

Das allgemeine Ziel des Projektes ist die Erstellung einer hochmodern Softwarebibliothek zur Entwicklung von Techniken des Maschinellen Lernens und deren Anwendung auf praktische Probleme im Bereich Data Mining.

Die allgemeinen Ziele, die die Entwickler erreichen wollen, umfassen:

- Die Techniken des maschinellen Lernens allgemein verfügbar zu machen

[1] `http://www.cs.waikato.ac.nz/~ml/weka/`

- Das Anwenden dieser Techniken auf praktische Probleme, die für die Industrie von Interesse sind
- Das Entwickeln von neuen Algorithmen für maschinelles Lernen und sie der Welt zur Verfügung zu stellen
- Einen Beitrag zu einem theoretischen Framework in diesem Gebiet liefern.

Komponenten

WEKA ist eine Sammlung von Machine Learning Algorithmen für Data Mining Aufgaben. Die Algorithmen können direkt auf Datensätze angewendet, oder aus anderem Java-Code angesprochen werden. WEKA beinhaltet Tools für

- (Daten-) Vor- bzw. Aufbereitung,
- Klassifizierung,
- Regression,
- Gruppierung,
- Assoziationsregeln und
- zur Visualisierung.

Die Software-Bibliothek WEKA stellt lokalen Programmen ihre Funktionalität zur Verfügung. Seit Weka Version 3 ist die Data Mining Software komplett in Java geschrieben. WEKA ist Open Source Software und steht unter der GNU General Public License.

Implementierung (Realisierung)

Die Software ist öffentlich verfügbar und beinhaltet eine Sammlung von Algorithmen zur Lösung von realen Datamining Problemen, die über ein einheitliches Interface zugreifbar sind.

Größenordnung Professionalität

Viele Standardtechniken sind in einer Software Workbench zusammengefasst. Mit dieser ist ein Spezialist in der Lage Machine Lerning anzuwenden, um nützliches Wissen von Datenbanken abzuleiten, die für eine manuelle Analyse zu groß sind. Benutzer von WEKA sind ML-Entwickler und industrielle Wissenschaftler, aber es wird auch in der Lehre eingesetzt.

3.1.2. JSR-73 : JDM

Der *Java Specification Request No. 73 (JSR-73)*[2] beschreibt die *Java Data Mining API Version 1.0 (JDM)* (vormals JDMAPI). Sie ist eine allgemeine Schnittstelle zwischen Data Mining Algorithmen und deren Anwendung in Programmen und entkoppelt die Algorithmen von den aufrufenden Anwendungen.

JDM wurde von der Java Community innerhalb eines *Java Community Process (JCP)* entwickelt, der im Juni 2000 mit der Erstellung des Java Specification Requests begonnen hat und im August 2004 endete. JDM ist nun ein offizieller Teil des Java Standards.

Zielrichtung

Die JDM Spezifikation entspricht dem Bedarf nach einer reinen Java API, die die Erstellung von Data-Miningmodellen, das Bewerten von Daten an Hand von Modellen unterstützt, sowie auch die Erzeugung, die Speicherung, der Zugriff und die Wartung von Daten und Metadaten von Data-Miningergebnissen und ausgewählten Datentransformationen ermöglicht.

Sie ist gegenwärtig die Standard-API für Data-Mining auf den sich die Java Community geeinigt hat.

Durch das Verwenden von JDM können Entwickler von Data-Mininganwendungen ein einzelnes Standard-API voraussetzen, durch die ihre Algorithmen angesprochen werden können. Ebenso können Programme gegen eine einzige Data-Mining-API codiert werden, das vom zugrunde liegenden Data-Miningsystem unabhängig ist.

Das ultimative Ziel ist, dass JDM für Data-Miningsysteme das liefern soll, was JDBC für relationale Datenbanken geschaffen hat: eine Standard-API.

Komponenten

JDM benutzt Teile des CWM Data-Mining Metamodells und der Java Metadata Interface (JMI; JSR-40). CWM Data-Mining erleichtert die Konstruktion und den Einsatz von Data Warehouse und Business Intelligence Anwendungen, Hilfsprogrammen und Plattformen basierend auf OMG offenen Standards für Metadaten und Systemspezifikation (MOF, UML, XMI, CWM, ...). Das Java Metadata Interface bietet eine allgemeine Namenskonvention von Methoden. Die folgenden Spezifikationen dienten beim Entwurf als Referenzen für JDM:

- DMG PMML 2.0, [PMML]
- ISO SQL/MM Part 6. Data Mining [SQL/MM-DM]
- Common Warehouse Metamodel [CWM] Volume 1, Data Mining [CWM-DM]

JDM beinhaltet Interfaces zur Unterstützung von Miningfunktionen wie Klassifikation, Regession, Gruppierung und Assoziation; zusammen mit speziellen Miningalgorithmen wie Naïve Bayes, Support Vector Machines, Entscheidungsbäume, Neuronale Netze

[2] `http://www.jcp.org/en/jsr/detail?id=73`

9/2004	Expert Group formation
1/2006	Early Draft Review
geplant 5/2006	Public Draft
geplant 10/2006	Proposed Final Draft
geplant 2/2007	Final Release

Tabelle 3.1.: Entwicklungszeitpunkte von JDM 2.0

und k-means. Diese Funktionen werden synchron oder asynchron durch Mining Tasks ausgeführt. Tasks umfassen build, apply for batch und real-time, test, Import und Export. Import und Export können mehrfache Modellrepräsentationen unterstützen und schließen PMML und native Formate ein. JDM umfasst auch Interfaces für confusion Matrizen, Lift und ROC Resultate, Taxonomy und Regel Repräsentationen und Statistiken.

JDM beinhaltet ebenfalls eine Spezifikation eines Web Service Interface, dass auf dem JDM UML Modell basiert, um eine Service Orientierte Architektur (SOA) zu ermöglichen. Obwohl JDM Web Service eng verbunden mit der Java Schnittstelle ist, bedient JDM Web Service Anforderungen weit über die Java Gemeinschaft hinaus, da es auf WSDL und XML basiert.

Implementierung (Realisierung)

JDM steht als eine Java API von der Java 2 Enterprise Edition (J2EE) zur Verfügung. Die Referenzimplementierung, die der finalen Version beiliegt, beinhaltet nur wenige realisierte Methoden und ist nur für Kompatibilitätstests geeignet.

Größenordnung Professionalität

Die JDM Experten Gruppe entschied sich früh dazu einige Features nicht in JDM einfließen zu lassen, um es besser verwalten zu können. Daher wurden Funktionen wie die Datentransformation, die Visualisierung und das Verarbeiten von unstrukturierten Daten (z.B. Text) aus der ersten Version der API weggelassen. Es ist zu bemerken, dass JDM hinsichtlich der Visualisierung viele wichtige Datenobjekte hat, die zur Visualisierung benötigt werden.

3.1.3. JSR-247 : JDM 2.0

Für die *Java Data Mining API Version 2.0 (JSR-247)*[3] hat der Entwicklungsprozess im Juni 2004 begonnen und sollte im Juni 2006 enden. Tatsächlich hat sich die Erscheinung des Early Draft Review um acht Monate verzögert, weshalb sich die entgültige Version wahrscheinlich erst Anfang 2007 feststehen wird. Eckdaten der zeitlichen Entwicklung sind der Tabelle 3.1 zu entnehmen.

[3] `http://www.jcp.org/en/jsr/detail?id=247`

Zielrichtung

JDM entspricht dem Bedarf an einer reinen Java API, das Data-Miningvorgänge und Aktivitäten unterstützt. JDM 2.0 erweitert JDM mit benötigter Funktionalität für neue Miningfunktionen und -algorithmen und entsprechender Webservice Spezifikation. Features, die in JDM 2.0 bedacht werden sind unter anderem die folgenden:

- Sequential Patterns / Time Series - Mining Funktionen zur Vorhersage und Modellierung von saisonalen oder periodischen Schwankungen der Daten.
- Transformations interface - Datenaufbereitung ist ein Schlüsselaspekt für jede Datamining Lösung. Ein separates JSR für Transformationen ist wahrscheinlich garantiert. Eine enge Integration in solch ein JSR zu haben, hat hohe Priorität.
- Ensemble models - definierte zusammengesetzte Muster, die mit Logik strukturiert sind.
- Anwendung von Assoziazionen - erweiterte Spezifikation, um Voraussage basierend auf Assoziationsregeln zu ermöglichen.
- Text Mining - ermöglicht die Verarbeitung von unstrukturiertem Textdaten.
- Model Comparison - führt die Möglichkeit ein Modelle an Hand verschiedener Qualitätsmetriken zu vergleichen.

Das Ziel der JDM 2.0 Expert Group wird sein, diese Merkmale zu untersuchen und andere zu identifizieren, die für das Data-Mining und die Java Gemeinschaft notwendig sind.

Komponenten

JDM 1.0 liefert das begriffliche Gerüst und die notwendige Infrastruktur für JDM 2.0. Die Standards PMML und SQL/MM fahren mit der Entwicklung von Data-Miningmodellspezifikationen und Funktionalität fort, die für Data-Miningbenutzer wertvoll sein werden. In JDM 2.0 werden die letzten PMML und SQL/MM Spezifikationen einfließen.

Eine **Implementierung (Realisierung)** existiert noch nicht, da der Standard noch in der Entwicklung ist. Eine besondere Implementierung dieser Spezifikation muss nicht unbedingt alle Schnittstellen und Services unterstützen, die durch JDM definiert sind. Jedoch liefert JDM einen Mechanismus um die unterstützten Schnittstellen und Fähigkeiten abzufragen.

3.1.4. prudsys XELOPES

XELOPES basiert auf dem Common Warehouse Metamodel (CWM) Standard der OMG zum Austausch kompletter Unternehmensdaten und stellt zugleich dessen erste Implementierung für Data Mining dar.

Die im Juli 1998 gegründete Prudential Systems Software GmbH wurde durch eine AG-Umwandlung im Dezember 2001 zur prudsys AG mit enviaM AG (RWE Group) als Gesellschafter. Im August 2003 stellte prudsys die XELOPES-Bibliothek vor und überzeugte damit beim KDD-Auswahlverfahren. Seit Juni 2000 veranstaltet die prudsys AG den jährlich stattfindenden DATA-MINING-CUP.

Zielrichtung

Die prudsys XELOPES Bibliothek ist eine offene, plattform- und datenquellenunabhängige Bibliothek für Embedded Data Mining. Die komplett in UML modellierte Bibliothek erlaubt die Integration von Data-Mining-Komponenten. Mit Hilfe der Bibliothek können z.B. vollautomatisch Kundenprofile generiert, Warenkorbregeln und Sequenzen aus Transaktionen extrahiert oder Prognosen verschiedenster Geschäftsdaten erstellt werden. Darüber hinaus unterstützt XELOPES alle relevanten Data-Mining-Standards, darunter das PMML-Format zum XML-basierten Austausch von Data-Mining-Modellen, den OLE DB for Data Mining Standard für Datenbanken sowie die WEKA-Bibliothek mit einer Vielzahl von Algorithmen.

Komponenten

Über das so genannte Mining Input Stream Konzept arbeitet XELOPES auf abstrakten Datenmatrizen, welche die Grundlage von Data-Mining-Analysen bilden. Damit sind alle Data-Mining-Verfahren in XELOPES Datenquellenunabhängig. Implementierungen der MiningInputStream-Klasse existieren für Arbeitsspeicher, Dateiformate (z. B. Logfiles) sowie Datenbanken. Darüber hinaus existieren Konnektoren für SAP, SAS und INTERSHOP enfinity Systeme.

Die XELOPES Bibliothek wurde in voller Übereinstimmung mit dem Model Driven Architecture (MDA-) Standard der OMG entwickelt. Der XELOPES-Kern wurde über UML als CWM-Erweiterung definiert und umfassend dokumentiert. Er ist derzeit für die Programmiersprachen Java, C++ und C# sowie für CORBA und Web Services implementiert. Über das XMI-Format kann er jederzeit auf weitere Plattformen portiert werden.

Implementierung (Realisierung)

XELOPES enthält ein GUI, welches es dem Nutzer erlaubt, sich mit der Bibliothek vertraut zu machen. Das GUI erlaubt den Zugriff auf Dateien und Datenbanken, die Anwendung aller Algorithmen der Bibliothek sowie die Visualisierung der Data-Mining-Modelle und deren Umwandlung in das PMML-Format.

Die Steuerparameter der Data-Mining-Verfahren können, sofern sie nicht automatisch reguliert werden, auf 2 Levels vorgegeben werden: Auf allgemeinem Level für Nicht-Data-Mining-Spezialisten und auf Expertenlevel.

Größenordnung Professionalität

Die Algorithmen der XELOPES-Bibliothek können über ein umfangreiches Automatisierungs-Framework selbständig ihre Steuerparameter einstellen, um nutzerdefinierte Zielkriterien zu erreichen.

Dank konsequenter Standardisierung und Einsatz von PMML (XML-Format für Data-Mining-Modelle) können die von XELOPES generierten Modelle leicht in andere Data-Mining-Systeme exportiert werden. Umgekehrt kann XELOPES unterschiedlichste Data-Mining-Formate einlesen.

3.2. Softwarearchitekturen

Eine Softwarearchitektur ist die grundlegende Organisation eines Systems, dargestellt durch dessen Komponenten, denen Beziehungen untereinander und zur Umgebung sowie den Prinzipien, die den Entwurf und die Evolution des Systems bestimmen.

3.2.1. UIMA

Das *Unstructured Information Management Architecture (UIMA)* Projekt[4] von IBM Research ist ein Framework zur Analyse unstrukturierter Daten und findet seinen Ursprung in der Tatsache, dass die Suche in unstrukturierten Daten unergiebig ist. Zur Lösung sollen innerhalb der Daten, per Analyse, Dinge bestimmt werden, die von Interesse sind, damit Suchmaschinen dann relevante Resultate erzielen können.

Entwickelt wurde UIMA in den letzten vier Jahren von IBM Research mit Unterstützung der Defense Advanced Research Projects Agency (DARPA), der zentralen Forschungseinreichtung des US-Verteidigungsministerium und den Universitäten Carnegie Mellon, Columbia, Stanford und Massachusetts Amherst. Aber auch einige weitere Unternehmen waren an der Entwicklung beteiligt.

IBM setzt UIMA bereits im WebSphere Information Integrator OmniFind Edition[5] ein, aber auch der WebSphere Portal Server und Lotus Work Place nutzen UIMA. Seit Ende letzten Jahres steht die Software auf SourceForge.net[6] als Open Source zur Verfügung.

Zielrichtung

Mit der Unstructured Information Management Architecture (UIMA) bietet IBM eine Open-Source-Technik an, die Entwicklern helfen soll, Applikationen zu entwickeln, die verstehen, welche Passagen in einem Text wichtig und relevant sind und was mit ihnen gemeint ist. So sollen sich unstrukturierte Daten besser erfassen lassen. UIMA stellt dazu ein offenes Software-Framework mit einer Standardschnittstelle bereit, über die sich Applikationen um die Analyse unstrukturierter Informationen erweitern lassen. So

[4] `http://www.research.ibm.com/UIMA/` March 2005
[5] siehe Kapitel 3.3.5
[6] `http://uima-framework.sourceforge.net/`

soll sich UIMA leicht in bestehende Systeme integrieren lassen. Zugleich soll UIMA aber auch die Basis darstellen, um neue, wiederverwendbare Analyse-Komponenten für unstrukturierte Daten zu erstellen.

Komponenten

UIMA ist eine komponentenbasierte Softwarearchitektur für die Entwicklung und Verwendung multimodaler Analytik für die Analyse unstrukturierter Information und seine Integration mit Such- und Wissensverwaltungstechniken.

Die UIMA Verarbeitung ist durch eine Serie von Modulen realisiert, die analysis engines genannt werden. Das Ergebnis der Analyse ist eine Zuordnung der Semantik zu den Elementen von unstrukturierten Daten, zum Beispiel der Hinweis, darauf daß sich das Wort Washington auf den Namen einer Person oder darauf daß es sich auf eine Stelle bezieht.

UIMA ist eine Architektur, die Komponentenschnittstellen, Designmuster, Datendarstellungen und Verwendungsaufgabenbereiche angibt. Der UIMA Software Development Kit (SDK) ist ein Softwaresystem, das ein Laufzeitframework, APIs und Hilfsprogramme umfasst, um UIMA Komponenten zu implementieren, auszuliefern und einzusetzen. Es beinhaltet eine semantische Suchmaschine zum Indexerstellen und Abfragen von Ergebnissen der Analyse.

Die Common Analysis Structure (CAS) bietet zusammenarbeitenden UIMA Komponenten eine gemeinsame Darstellung und einen Mechanismus zum gemeinsamen Zugriff auf die zu analysierenden Daten (z.B. Dokument, Audiodaten, Videostreams usw.) und auf die aktuellen Ergebnisse der Analyse. CAS ist eine Datenstruktur, für die UIMA mehrere Schnittstellen zur Bearbeitung und Auswertung bereitstellt.

Implementierung (Realisierung)

IBM's Unstructured Information Management Architecture (UIMA) ist eine Software Architektur zum Entwerfen und Entwicklen von Unstructured Information Management (UIM) Anwendungen. Eine UMI Anwendung kann als zweiphasig aufgefasst werden: Analysis und Delivery (vergleiche Abbildung 3.1). In der Analysephase werden Sammlungen von Dokumenten betrachtet und analysiert. Die Ergebnisse werden, wie von der Deliveryphase benötigt, in einer oder mehreren Formen gespeichert. In der Deliveryphase werden die Ergebnisse der Analyse, möglicherweise zusammen mit den originalen Dokumenten oder anderen strukturierten Informationen, dem Benutzer der Anwendung durch anwendungsangepasste Methoden und Schnittstellen zur Verfügung gestellt. Im Wesentlichen werden unstrukturierte Daten durch Hilfsmittel, wie Indizes, Taxonomien usw. strukturiert bzw. semantisch angereichert, um einen gezielten Zugriff zu ermöglichen.

UIMA ist eine Analyse-Software die Text in Dokumenten und anderen Inhaltsquellen verarbeitet. Die Software soll die versteckte Bedeutung, Beziehungen und zugehörige Fakten zu einem Suchbegriff „verstehen“. UIMA arbeitet mit einer Vielfalt an Analysetechniken wie statistische und rollenbasierte Verarbeitung natürlicher Sprache (Natural Language Processing - NLP), Information Retrieval (IR), maschinenbasiertes Lernen,

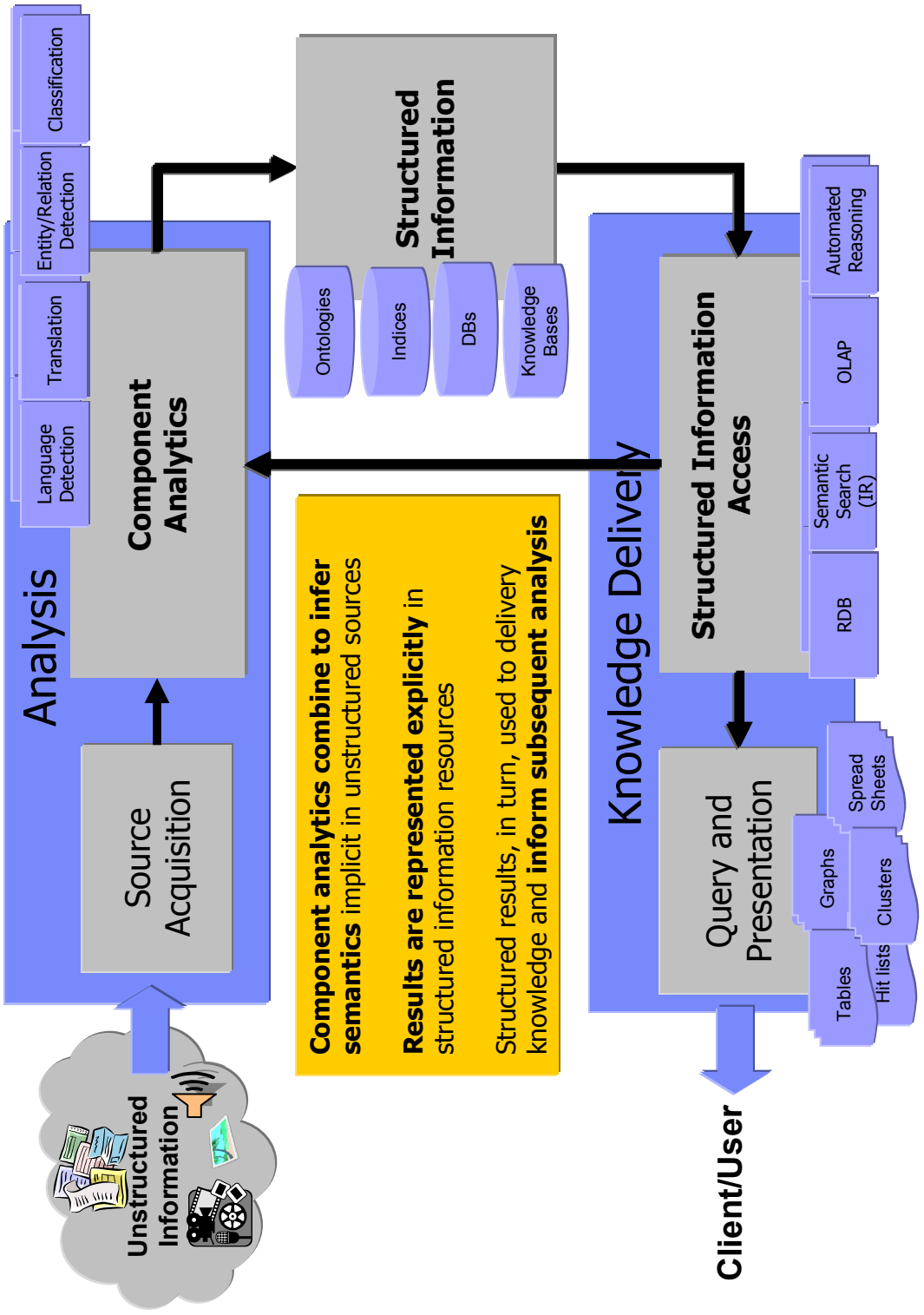

Abbildung 3.1.: UIMA Übersicht

Ontologien und die Verlinkung dieser Prozesse zu strukturierten Informationsservices wie Datenbanken und Suchmaschinen.

UIMA ist keine Fertiglösung. Als Rahmenlösung (Framework) wird UIMA in andere Anwendungen integriert.

Größenordnung Professionalität

Die Grenze zwischen den strukturierten und den unstrukturierten 'Welten' ist noch immer eine der großen Herausforderungen der IT-Industrie. Business Intelligence und Wissensmanagement haben sich jeweils beinahe isoliert von einander entwickelt, und doch haben sie die gleiche Aufgabe - eine klare und verwertbare Sicht auf Datenbestände. In IBMs Technologie sehen Analysten eine Basis, auf welcher der Brückenschlag zwischen den beiden Welten gelingen könnte.

Mit der Offenlegung des Codes will IBM eine breite Akzeptanz auf dem Markt schaffen. Unternehmen, Regierungen und Universitäten sowie unabhängige Softwarehersteller können dadurch einfacher Anwendungen oder Erweiterungen rund um UIMA entwickeln.

3.3. Kommerzielle Softwarepakete

3.3.1. SPSS Clementine

Clementine ist eine Data Mining Workbench von SPSS. SPSS entwickelt seit 1975 Software im Bereich Analyse von Daten zur Unterstützung von Entscheidungen. Nach der Übernahme von Integral Solutions Ltd. und dessen Produkt Clementine im Dezember 1998 erschien im Januar 1999 die Version 5.0 von Clementiene unter Führung von SPSS. Die aktuelle Version ist 9.0 ist seit November 2004 erhältlich.

Zielrichtung

Die Clementine Workbench ist eine umfassende Lösung zur Untersuchung von großen Datenmengen. Zu den unterstützten Analyse-Verfahren zählen beispielsweise Assoziationsanalysen, Regressionsanalysen, Clusteranalysen, neuronale Netze sowie Entscheidungsbäume.

Besonderes Kennzeichen des Data-Mining mit Clementine ist zum einen die Fähigkeit, sehr große Datenmengen sehr effizient und schnell verarbeiten zu können, zum anderen die Vielzahl von Schnittstellen zum Import von Datensätzen. So können beispielsweise Daten aus Datenbanken (via ODBC), SPSS-, SAS- oder .csv bzw. Textdateien importiert werden.

Komponenten

Mit einer Reihe von verschiedenen Funktionsknoten lassen sich Analysen grafisch darstellen und durchführen. In Hinblick auf die Knoten kann man sieben grundsätzliche

Funktionen unterscheiden: Datenquellen, Datensatzoperationen, Feldoperationen, Modelle (bzw. Verfahren), Diagramme, Auswertungen sowie geschätzte Modellparameter. Clementine unterstützt CRISP-DM und PMML per Design.

Implementierung (Realisierung)

Positiv fällt die einfache Nachvollziehbarkeit der einzelnen Auswertungsschritte - die auch insbesondere für Dritte, nicht direkt mit der Auswertung befasste Personen gegeben ist - sowie die vielseitigen Änderungsmöglichkeiten auf. Gerade nicht direkt mit der Analyse befasste, kundige Personen können sich anhand des graphisch abgebildeten Ablaufs einer vorzunehmenden Datenanalyse leicht einen Überblick über das Vorgehen bei der Analyse verschaffen. Weiterhin stehen umfangreiche Möglichkeiten zur Manipulation von Datensätzen sowie der Zusammenführung von Daten aus verschiedenen Quellen zur Verfügung.

Größenordnung Professionalität

Grundsätzlich scheint sich das Programm, mit einigen Ausnahmen, eher für explorative Analysen zu eignen. Hypothesenprüfende Verfahren sind nur wenige implementiert. Insgesamt richtet sich das Programm sowohl an den Wissenschaftler wie auch den Anwender in der Praxis. Durch die Möglichkeit, eigene Algorithmen zu implementieren kann der Funktionsumfang des Programms massiv erweitert werden. Anderseits eröffnen die strukturierte Vorgehensweise und die graphische Oberfläche auch Möglichkeiten zur Datenanalyse bei geringerer Methodenkenntnis.

3.3.2. MetaMatrix MetaBase

MetaMatrix ist ein 1997 gegründetes Technikunternehmen mit Hauptsitz in New York und Büros in St. Louis, Boston, Dallas, San Francisco und London und liefert die skalierbare, betriebssystemunabhängige Unternehmenssoftware für Enterprise Information Integration: MetaBase[7], das Data Management und Metadata Repository System.

Zielrichtung

MetaBase liefert Unternehmen eine verteilte, unternehmensweite semantische Schicht, die Daten von physischen Quellen abstrahiert, die semantische Unterschiede von verschiedenen Quellen festlegt. Diese semantische Abstimmung führt zu einer gesteigerten Sichtbarkeit von Information in einem Unternehmen, verkürzten Entwicklungszeiten und reduzierten Betriebskosten, z.B. durch die Entdeckung und Eliminierung von überflüssigen Informationsquellen.

[7] `http://www.metamatrix.com/l3_metabase.html`

Komponenten

Jede Komponente von MetaBase wurde für ein Optimum an Flexibilität und Benutzerfunktionalität entwickelt. Im einzelnen umfasst MetaBase die Produkte:

MetaBase Modeler, ein graphisches Modellierungstool, der Metadaten modellieren und von verschiedenen Quellen einlesen kann. Metabase Modeler kann auch Metadaten in ein Modell verpacken und diese Modelle als eine Hierarchie oder ein UML Diagramm darstellen. Weitere Features sind unter anderem die Modellierung von Transformationsanfragen, Modellvalidierung, Versionskontrolle und die Unterstützung von XMI Modell Import/Export von anderen Systemen.

MetaBase Server, der XMI-basierte Server, der Modelle per XMI zwischen den verschiedenen Anwendungen austauschen kann und Import/Export durch ein Pluginbasiertes Framework unterstützt. Er bietet eine Sicht auf alle Informationsquellen, die mit dem Server verbunden sind, sowie die Möglichkeit JDBC-kompatible Quellen zu verändern. Die Sammlung von verschiedenen Quellen erscheint als eine einzelne Quelle gegen die ein Entwickler sein Programm erstellt. Der Server bietet einen Echtzeitzugriff auf verschiedene Resourcen durch sein verteiltes Abfragesystem.

MetaBase Repository, ein MOF-basiertes Metadaten Repository, dass Metadaten mit entsprechenden Versionsinformationen speichern kann.

MetaBase Reporter, ein Browser-basiertes Metadaten Such- und Analysetool. Es durchsucht und analysiert Metadaten Beschreibungen des MetaBase Repository.

MetaBase basiert auf den Metadatenstandards der Object Management Group (OMG): Meta-Object Facility (MOF), Common Warehouse Metamodel (CWM), XML Metadata Interchange (XMI) und Unified Modeling Language (UML).

Implementierung (Realisierung)

MetaMatrix MetaBase ist die erste Infrastruktursoftware zur Integration von allen verschiedenen Informationsquellen eines Unternehmens.

Der MetaMatrix Server benutzt eine SQL-basierte Programmiertechnik, die einfach von Millionen von existierenden Programmieren benutzt werden kann, ohne zunächst eine neue Technik zu erlernen. Der MetaMatrix Server erweitert den Standard Application Server durch einen gleichförmigen Datenzugriff auf verschiedene Datenbanken in einem sicheren und skalierbaren Anwendungsumfeld. Die Reduzierung auf genau ein Interface bedeutet eine viel schnellere Auslieferung von Projekten, ohne die Erstellung von kostspieligen und ineffizienten Data Warehouse Lösungen.

Benutzer können Abfragen in SQL an den MetaBase Server senden und erhalten die Daten der verschiedenen Datenquellen als eine Antwort zurück. Alternative können Abfragen auch als XML Dokument an den Server gesendet werden, der darauf mit einem Schema-konformen XML Dokument antwortet. Der Server arbeitet in Echtzeit, die

Informationen werden nicht von den ursprünglichen Datenquellen verschoben, so dass Änderungen an den Quellen sofort zur Verfügung stehen. Das Model-Driven Information Integration white paper[8] erklärt im Detail, wie dass MetaMatrix System eine Infrastruktur des Model-Driven Architecture bereitstellen kann.

Größenordnung Professionalität

MetaMatrix publiziert Software für den Enterprise Software Markt und ist sehr erfolgreich damit.

MetaMatrix MetaBase modelliert und verwaltet Metadaten über verschiedene Datenquellen und stellt diese Informationen über eine einheitliche Schnittstelle zur Verfügung. Es ist möglich, die Form und die Struktur jeder Datenquelle zu modellieren, ihnen logische Namen zu geben und die Daten unabhängig von der physischen Datenbank zu organisieren. Die erstellten und gespeicherten Modelle in MetaBase beinhalten die Details der Struktur und der Beziehungen der einzelnen Objekte innerhalb des Modells. Diese Details werden dann den Wirtschaftsanalysten, Anwendungsentwicklern und anderen MetaBase Tools Benutzern zur Verfügung gestellt. Die gesteigerte Sichtbarkeit der Information ermöglicht es redundante Datenquellen zu erkennen und zu beseitigen. MetaBase Modeler bietet zusätzlich die Möglichkeit verschiedene strukturierte und unstrukturierte Informationsquellen in ein XML-Dokument abzubilden. Der Austausch über XMI ermöglicht eine einfache Anwendungsintegration und die Verwendung von Web Services.

3.3.3. SAS SAS-9, Enterprise Miner und Text Miner

SAS wurde 1976 gegründet und ist mittlerweile das größte privat gehaltene Softwareunternehmen der Welt mit fast 10.000 Angestellten rund um den Globus. Im Jahr 2004 begann SAS die Einführung von SAS®9[9], einer integrierten, offenen und erweiterbaren Enterprise Intelligence Plattform. Die Produkte Enterprise Miner und Text Miner sind eigenständige Komponenten dieser Plattform im Bereich des Analytic Intelligence.

Zielrichtung

Die SAS Enterprise Intelligence Platform ist eine integrierte Softwaresuite mit verschiedenen Aufgabenbereichen zur Nutzung von Enterprise Intelligence.

Komponenten

Die Plattform ermöglicht die optimale Konfiguration der Komponenten um sie innerhalb einer bestehenden Architektur einsetzen zu können. Die Komponenten umfassen die Bereiche:

[8] `http://www.metamatrix.com/resources/white.jsp`
[9] `http://www.sas.com/software/sas9`

- SAS Data Integration bietet vorinstallierte und hochperformante Datenverbindungen, Datenqualität, ETL (extract, transform and load), Datenmigration und Datensynchronisation. SAS bietet eine ETL-Lösung, die unabhängig von jeder Plattform arbeitet. Sie umfasst die Datenextraktion aus vielen Datenquellen, integriertes Metadaten-Management, Datenbereinigung und eine Administrations-Oberfläche, mit der der Prozess der Datenintegration in Gang gesetzt und durchgeführt werden kann.

- SAS Scalable Intelligence Server ist eine dedizierte Lösung zur Speicherung und Verteilung von Daten. SAS bietet relationale, multidimensionale und parallele Speicheroptionen, die von Beginn an dafür entwickelt wurde, Daten im Bereich Business Intelligence und Analyse-Anwendungen effizient verfügbar zu machen, ganz gleich, ob von SAS oder von Dritten.

- SAS Analytic Intelligence ist eine integrierte Umgebung für vorhersagende und beschreibende Modellierung, Vorhersage, Optimierung, Simulation, experimentellen Entwurf und mehr. SAS bietet das größte Lösungs-Portfolio für Analytik der gesamten Branche - von der einfachen Datenabfrage bis hin zu anspruchsvollen Möglichkeiten, Modelle zu entwickeln. SAS beseitigt den Mythos rund um die High-end Analyse-Technologie durch ihre Koppelung an eine breite Palette von Benutzeroberflächen und Analysewerkzeugen.

- SAS Business Intelligence liefert eine Menge von BI Fähigkeiten, die verschiedenen Benutzern die Erstellung von bedeutungsvollen Darstellungen der firmen weiten Daten ermöglichen. SAS unterstützt die Anwender durch schnellen Datenzugriff im gewünschten Format und nach Bedarf. SAS hat die passenden Oberflächen für verschiedene Benutzergruppen und -bedürfnisse. Mit SAS finden Anwender ihre Antworten selbst, während die IT die Kontrolle über Qualität und Konsistenz der Daten behält.

Als Teil einer integrierten Plattform werden alle Komponenten und Dienste von einem einzelnen Punkt verwaltet, was den Verwaltungsaufwand für Wartung von Anwendungen, Benutzern und Sicherheit reduziert. Datenkonsistenz ist gesichert weil Metadaten in einem allein stehenden Metadaten Repository gespeichert werden und dass von allen SAS Techniken und Lösungen gemeinsam benutzt wird.

Implementierung (Realisierung)

Rechen- und I/O-Durchsatz der Version 9.1 werden durch die Scalable Architecture deutlich verbessert. SAS bewältigt die zunehmende Zahl von Nutzern, CPUs, Unternehmensdaten und komplexen Analysen mit Multithreading bei Rechenverfahren, Servern

und I/O im Austausch mit Drittdatenspeichern. Diese erweiterte Skalierbarkeit fördert die optimale Nutzung der bestehenden Rechenressourcen und kommt in heterogenen Hardware- und Betriebssystemumgebungen zum Tragen.

SAS knüpft an die langjährige Praxis von SAS an und erweitert die Funktion offener Standards. Der neue Open Metadata Server ist mit dem OMG-CWM-Metadatenmodell und dem XML-basierten Austauschformat kompatibel und bietet so die Möglichkeit offenen Zugangs und offenen Austauschs. Zusätzlich zu den leistungsstarken XML-Fähigkeiten werden jetzt auch XSL, XSLT, PMML und CDISC unterstützt. Entwickler, die bereits die Stärken für die Entwicklung von Java-, C++- oder VB-Anwendungen nutzen, können die IOM-Schnittstellen von SAS Open Metadata Server verwenden. Dadurch wird sichergestellt, dass die Anwendungen zentrale Metadaten interoperabel nutzen können.

Größenordnung Professionalität

Die SAS Enterprise Intelligence Plattform bietet eine Grundlage für maßgeschneiderte Lösungen zur Unterstützung Ihrer Enterprise Intelligence, Customer Intelligence, Financial Intelligence, Supply Chain Intelligence und mehr. Ebenso bietet SAS sofort einsatzbereite Lösungen für zahlreiche vertikale Märkte wie z.B. Finanzdienstleister, Biowissenschaften, Gesundheitswesen, Handel und produzierende Branchen.

3.3.4. IBM DB2 Intelligent Miner

Gegründet wurde die International Business Machines Corporation (IBM)[10] am 14 Februar 1924. IBM ist mit einem Umsatz von 96,5 Milliarden US-Dollar im Jahr 2004 der weltweit größte Anbieter im Bereich Informationstechnologie (Hardware, Software und Services). Das Unternehmen beschäftigt weltweit rund 319.000 Mitarbeiter und ist in über 170 Ländern aktiv. Die IBM Deutschland GmbH mit Sitz in Stuttgart-Vaihingen beschäftigt derzeit etwa 25.000 Mitarbeiter an rund 40 Standorten. Die IBM Deutschland Entwicklung GmbH mit Sitz in Böblingen ist mit rund 1.700 Mitarbeitern - Informatiker, Ingenieure, Techniker - das größte Entwicklungszentrum außerhalb der USA mit globaler Entwicklungskompetenz.

Die Intelligent Miner Software gibt es in verschiedenen Varianten seit Anfang 1998. Die aktuelle Version 8.2 ist seit 2004 verfügbar.

Zielrichtung

IBM Intelligent Miner ist eine Data-Mining-Plattform für unternehmensweite Analysen. Die Software stellt DB2-Erweiterungen für die Erstellung von Data-Mining-Modellen zur Verfügung, verschiedene Scoring-Technologien (DB2 Extender und Oracle Cartridges), sowie ein Visualisierungsmodul für die Interaktion und die grafische Darstellung von Daten.

[10] `http://www.ibm.com`

Komponenten

Intelligent Miner besteht aus vier Software Produkten:

1. Intelligent Miner for Data

   - Umfassende Palette von Data Mining Tools.
   - Unterstützung von iterativen Prozessen, Datenvorverarbeitung, statistischen Analyseverfahren und Ergebnisanzeige plus einer Vielzahl von Mining-Algorithmen.
   - Verwendung erprobter individueller oder kombinierter Mining-Algorithmen für ein breites Spektrum von Geschäftsanforderungen.
   - Skalierbare Lösung mit besonderer Berücksichtigung der technischen Probleme bei umfangreichen Mining-Operationen wie großen Datenvolumina, parallelem Data Mining, langlaufenden Mining-Operationen und der Optimierung der Mining-Algorithmen.
   - API für die Entwicklung von individuellen, branchenspezifischen Mining-Anwendungen.

2. Intelligent Miner Modeling

   - Optimierte SQL-Erweiterung der DB2-Datenbank und Einbettung von Modellierungsfunktionen in Geschäftsanwendungen.
   - Unterstützung der Entwicklung von Data-Mining-Modellen in einem Format kompatibel mit PMML (Predictive Model Markup Language) V2.0, dem neuen Branchenstandard für analytische Modelle.
   - Möglichkeit der Anwendung neuer Beziehungen in Daten auf neue Daten seitens des Benutzers in Echtzeit.
   - Data-Mining-Modellanalyse anhand von DB2 Intelligent Miner Visualizer, einem Java-basierten Ergebnisbrowser. Anzeige und Evaluierung von Ergebnissen des Data-Mining-Modellierungsprozesses durch die Benutzer, auch ohne Fachkenntnisse.

3. Intelligent Miner Scoring

   - Unübertroffene Scoring-Technologie und Datenbankerweiterungen: DB2 Extender und Oracle Cartridges.
   - Skalierbarkeit und Leistungsstärke durch einfache SQL-Programmierschnittstellen und Standardschnittstellen für die Anwendungsentwicklung.
   - Speicherung von Mining-Modellen als XML-Objekte in relationalen Datenbanken Implementierung basierend auf neuen Industriestandards für Data Mining.

- Scoring von Datensätzen - Segmentierung, Klassifizierung und Ranking von Dateninhalten - basierend auf einem Satz vordefinierter Kriterien im Mining-Modell.
- Unterstützung aller Scoring-Funktionen von DB2 Intelligent Miner for Data, wie Entscheidungsbäume, neuronales und demografisches Clustering, Regression anhand von Polynomen und neuronale Netzwerke.
- Trennung des Mining-Modells und des Scoring-Algorithmus von der Anwendung: Diese Struktur ermöglicht eine kontinuierliche Modellverbesserung bei Tendenzänderungen oder beim Eingang von Zusatzinformationen, ohne dass die Anwendung unterbrochen werden muss.

4. Intelligent Miner Visualization

   - Java-Visualisierungsmodule für die Interaktion und grafische Darstellung von Ergebnissen von Verknüpfungen, demografischem Clustering und Modellierungsvorgängen für die Baumklassifizierung. Die Eingabe in die Visualisierungsmodule erfolgt in Form von PMML-2.0-kompatiblen Modellen.
   - Die Visualisierungsmodule können integriert als Unterstützung von Transaktionsverarbeitungs-Anwendungen oder als Applet in einem Webbrowser arbeiten. Benutzer haben die Möglichkeit, Modellergebnisse zu analysieren und so neue Einblicke in Geschäftsvorgänge zu gewinnen.

Implementierung (Realisierung)

DB2 Intelligent Miner for Data beinhaltet Funktionen für die Analyse, Datenaufbereitung und Skalierbarkeit, die die Data-Mining-Hauptfunktionen erweitern und wichtige Daten für die entsprechenden Benutzer leichter zugänglich machen. Das Mining von Daten aus großvolumigen transaktionsverarbeitenden Anwendungen, die beispielsweise in Verkaufsfilialen, an Bankautomaten, über Kreditkarten, Call Center oder E-Commerce-Aktivitäten gewonnen wurden, wird ebenfalls unterstützt. Zu den unterstützten Mining-Verfahren gehören die Assoziationsanalyse, die Analyse sequenzieller Muster, die Segmentierung und Klassifikation von Daten, die Zeitreihenanalyse sowie Verfahren zur Wertevorhersage.

Mit DB2 Intelligent Miner Modelling können Benutzer verborgene Beziehungen in Daten aufdecken, ohne die Daten in ein spezielles Data-Mining-System zu exportieren oder sich auf kleine Datenstichproben beschränken zu müssen. DB2 Intelligent Miner Modeling liefert DB2-Erweiterungen für die Modellierungsvorgänge „Aufdeckung von Assoziationen“, „Demografisches Clustering“ und „Baum-Klassifizierung“.

DB2 Intelligent Miner Scoring erweitert die Funktionen der Datenbank und ermöglicht es Benutzern, Data-Mining-Analysen in Echtzeit auszuführen. DB2 Intelligent Miner Scoring kann von Anwendungsprogrammen zum Klassifizieren, Sortieren und Vorhersagen der wahrscheinlichsten Geschäftsergebnisse für alle Sätze einer großen Datenbanktabelle bzw. eine Teilmenge oder einen einzelnen Datensatz eingesetzt werden. Die Ausführung ist in einer Reihe von Umgebungen möglich, einschließlich DB2 Extender

und Oracle Cartridges, DB2-Zugriff auf verteilte heterogene Datenquellen via DB2 Information Integrator und diverse Server. Durch die Verwendung von Java-Applets sind außerdem Funktionen für das Scoring von Arbeitsspeichern verfügbar.

DB2 Intelligent Miner Visualization bietet Funktionen für die Analyse von Data-Mining-Modellen über einen Java-basierten Ergebnisbrowser. Das Modul ermöglicht Benutzern die Anzeige und Evaluierung von Ergebnissen des Data-Mining-Modellierungsprozesses, auch ohne Fachkenntnisse.

Intelligent Miner Software läuft auf einer Vielzahl von Unix und Windows Betriebssystemen wie AIX, Sun Solaris, Windows NT/XP und Windows 200x.

Größenordnung Professionalität

Die Intelligent Miner Software wird in Böblingen entwickelt. Der Mining-Prozess kann weitgehend vordefiniert werden, wodurch Mining auch für Benutzer ohne spezifischen Data Mining Skill zugänglich wird, da nur begrenzte externe Unterstützung für die Erstellung und Wartung der Mining-Szenarien notwendig ist. Die Integration von Data Mining-Algorithmen direkt in die Datenbank erlaubt die Erstellung und Anwendung von Modellen direkt bei Daten.

Die Verwendung von standardisierten Techniken wie XML (PMML), SQL und JAVA ist eine einfache Anbindung von weiteren Gesamtlösungen wie SAP, SAS, SPSS und Oracle leicht möglich.

3.3.5. IBM WebSphere Information Integrator OmniFind Edition

IBM WebSphere[11] steht für eine Reihe e-business Lösungen in einer Plattform - vom Web Application Server über Entwicklungstools bis hin zu Portal-, E-Commerce- und Integration-Software.

Nach der Übernahme von Ascential Software (vormals Informix) im März 2005 wurde dessen Softwareangebot in die WebSphere Plattform integriert und steht nun unter dem Namen IBM WebSphere Information Integration zur Verfügung.

Ascential Software war der führende Anbieter von Enterprise Data Integration-Lösungen. Kunden nutzen die Integrations Suite für die effektive Integration und Bereitstellung von Daten über alle transaktionalen, operationellen und analytischen Applikationen im Unternehmen hinweg. Abgedeckt werden dabei die Bereiche Datenprofiling, Qualitätsmanagement und Bereinigung von Daten, End-to-End Metadaten-Management sowie das Extrahieren, Transformieren und Laden von Daten. Dank ihrer linearen Parallel-Processing-Architektur bietet sie höchste Performance im gesamten Integrationszyklus.

Fünf fundamentale Fähigkeiten werden durch die IBM WebSphere Information Integration Plattform zur Verfügung gestellt und jedem dieser fünf Fähigkeiten sind ein oder mehrere Produkte zugeordnet:

1. Connect - die Fähigkeit, sich an relevante Quellen anzuschließen - ob strukturiert oder unstrukturiert, Großrechner oder verteilt, intern oder extern.

[11] `http://www-306.ibm.com/software/de/websphere/`

- IBM WebSphere Information Integration Portfolio
- IBM WebSphere Information Integrator Replication Editions
- IBM WebSphere Information Integrator Event Publisher Editions

2. Understandig - die Fähigkeit, einen Einblick in den Inhalt, die Qualität und die Struktur von den Daten zu gewinnen, um ein Verständnis für die Daten zu erhalten, bevor sie überall in dem Unternehmen integriert und verbreitet wird.

   - IBM WebSphere ProfileStage

3. Cleanse - die Fähigkeit, die Daten zu normieren und aufzubereiten, so daß konsistente Sichten auf jede Person oder Geschäftssubjekte und seine Beziehungen im Unternehmensumfeld ersichtlich werden.

   - IBM WebSphere QualityStage

4. Transform - die Fähigkeit, wirksam und effizient große Mengen von Daten aus der Originaldatenquelle erfassen, umzuwandeln und weiter anzureichern.

   - IBM WebSphere DataStage
   - IBM WebSphere DataStage TX

5. Federate - die Fähigkeit, Information zu föderalisieren, die Anwendungen ermöglicht auf unterschiedliche Daten und Inhalt zuzugreifen und zu integrieren, als ob es ein einzelnes Betriebsmittel wäre, ohne Rücksicht darauf auf wo sich die Information befindet.

   - IBM WebSphere Information Integrator Standard Edition
   - IBM WebSphere Information Integrator Classic Federation for z/OS
   - IBM WebSphere Information Integrator Content Edition

WebSphere Information Integrator OmniFind Edition ist das erste Produkt, dass die UIMA Schnittstelle verwendet und soll im weiteren betrachtet werden.

Zielrichtung

WebSphere Information Integrator ist der Grundstein für ein strategisches Informationsintegrationssystem, mit dem man auf unterschiedliche, verteilte Daten und Echtzeit-Daten zugreifen und diese bearbeiten und integrieren kann, als würden diese Daten aus einer einzigen Quelle stammen.

Die OmniFind Edition ist darauf ausgelegt, die richtige Information für ein Frage schnell zu finden und bietet eine qualitativ hochwertige, skalierbare und sichere Suchfunktionalität für Unternehmen.

IBM WebSphere Information Integrator OmniFind Edition stellt Middleware für unternehmensweite Suchvorgänge zur Unterstützung von Intranets, Extranets und öffentlich zugänglichen Unternehmenswebsites zur Verfügung.

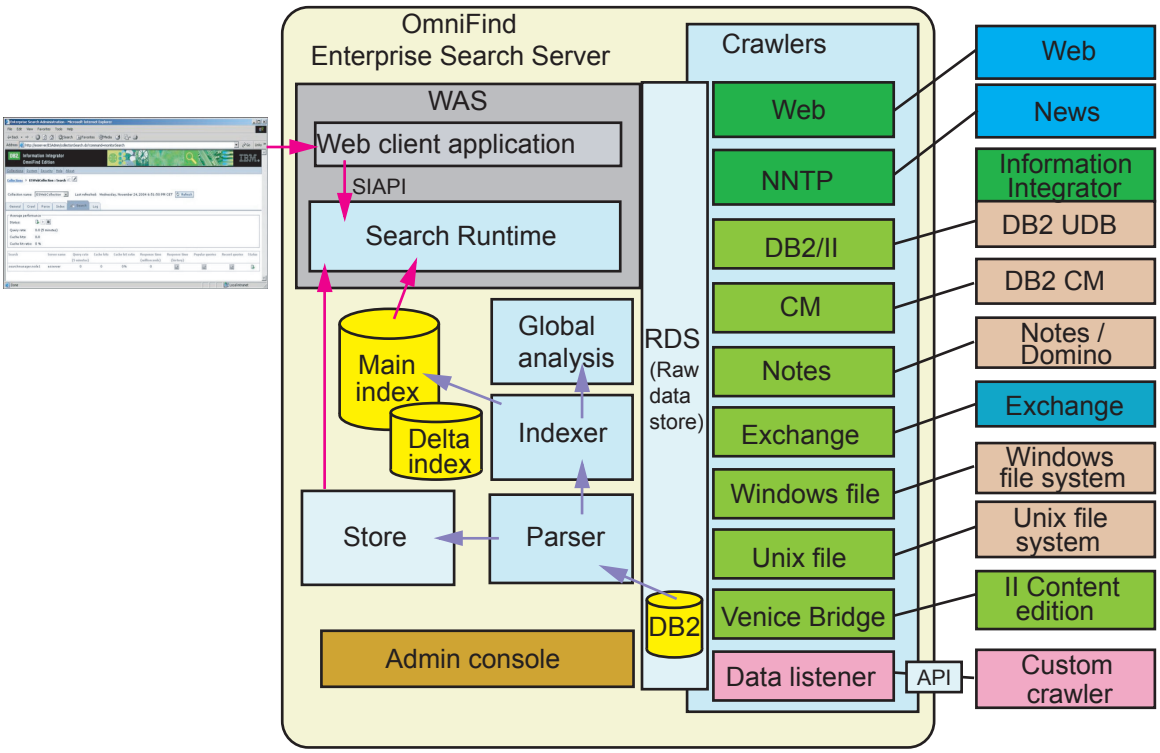

Abbildung 3.2.: IBM WebSphere Information Integrator OmniFind Search Server

Komponenten

Die Abbildung 3.2 zeigt die verschiedenen Komponenten der OmniFind Suchmaschine und den Datenfluss von den Datenquellen zu einer Suchanwendung, die Abfragen an eine Datensammlung stellt[12].

Die Crawler werden benutzt, um den Inhalt von bestimmten Datenquellen aufzufinden und einzusammeln. Die gesammelten Daten werden dann im DB2 Rohdatenspeicher abgelegt, der als Zwischenspeicherbereich dient. Der Parser analysiert die Dokumente, die von den Crawlern erfaßt wurden, aus dem Zwischenspeicherbereich und bereitet sie auf die Indizierung vor.

Der Indexer nimmt eine Tokenlistendarstellung eines Dokuments und schreibt es auf einen Zwischenspeicher, bevor die globale Analyse und die tatsächliche Indizierung ausgeführt werden. Das Hauptziel der globalen Analyse ist die Berechnung eines statischen Rangs für jedes Dokument. Beendet wird die Indexierungsphase durch das Aufbauen des Hauptindex oder des Deltaindex. Die Laufzeitumgebung des Suchprogrammes ist ein WebSphere Application Server (WAS); es stellt den Suchanfrageprozessor dar. Aufgerufen wird es von einer Webclientanwendung durch das Search and Index API (SIAPI). Nach der Abfrageanalyse wertet es die Abfrage mit dem Index aus und die übereinstim-

[12]die Abbildung 3.2 stammt aus *Intro to WebSphere Information Integrator OmniFind Edition* `http://www-136.ibm.com/developerworks/db2`

menden Dokumente werden nach Relevanz geordnet und zurückgegeben.

Implementierung (Realisierung)

WebSphere Information Integrator OmniFind Edition liefert eine Enterprise Suchlösung, die auf einem Server oder vier Servern (andere Konfigurationen werden nicht unterstützt) installieren werden kann. Der Installationsumfang umfasst die Programme

1. IBM DB2 Universal Database Server Edition, Version 8.2
2. IBM WebSphere Application Server, Version 5.1, mit integriertem IBM HTTP Server, Version 1.3.28
3. IBM WebSphere Application Server Network Deployment, Version 5.1

Um allen Benutzeranforderungen gerecht zu werden, sind folgende fünf Editionen erhältlich:

1. Replication Edition liefert Replikationsserverfunktionen für heterogene relationale Datenbanken.
2. Standard Edition liefert die Funktionen des Replikationsservers und Funktionen für den transparenten Zugriff auf verteilte heterogene Datenquellen.
3. Advanced Edition beinhaltet eine uneingeschränkte Lizenz für DB2 Universal Database Enterprise Server Edition V8.2 und die Funktionalität von DB2 Information Integrator Standard Edition. Zusätzliche Leistung und Flexibilität wird durch die Verwendung der lokalen DB2 UDB als lokalem Datenbankserver erzielt.
4. Advanced Edition Unlimited bietet alle Funktionen von WebSphere Information Integrator Advanced Edition und berechtigt darüber hinaus zu Verbindungen mit einer unbegrenzten Anzahl Datenquellen.
5. Developer Edition ist ein kostengünstiges Paket zum Entwerfen, Kompilieren oder Testen von modularen Anwendungen, die unterschiedliche Datenquellen integrieren.

Größenordnung Professionalität

- Lässt sich auf Millionen von Dokumenten und Tausenden von Benutzern skalieren.
- Integrierter Zugriff in Echtzeit auf diverse Daten anhand einer simulierten Datenbankstruktur und unabhängig vom Datenstandort.
- Liefert in Sekundenbruchteilen Ergebnisse aus Unternehmensinhalten, wie z. B. Intranets, Extranets, allgemein zugänglichen Unternehmenswebsites, relationalen Datenbanksystemen, Dateisystemen und anderen Inhaltsrepositories.

- Fügt sich problemlos in Java-Unternehmensanwendungen mit entsprechender Sicherheit ein, so dass vertrauliche Informationen nicht gefährdet werden.
- Replikationsserver zum Verwalten von Datenverschiebungsstrategien (einschließlich Verteilungs- und Konsolidierungsmodellen) und zum Überwachen des Synchronisierungsvorgangs.
- Unterstützung mehrerer Datenquellen einschließlich DB2 Universal Database, Informix, MS SQL Server, Oracle, Sybase, Teradata, ODBC und anderen.
- Kompatibel mit vielen Content-Quellen, einschließlich WebSphere MQ Message Queues, Lotus Notes, Documentum Enterprise Content Management System, Websuchmodulen und Webservices, MS Excel-Spreadsheets, XML-Dokumenten und anderen.
- Die Produktfamilie basiert auf offenen Standards wie SOA, Java, XML und J2EE und ist dadurch flexibel, skalierbar und leicht integrierbar.
- Erprobte Technik. OmniFind wird seit seit dem September 2003 erfolgreich als Suchtechnik auf dem IBM Intranet angewendet.

3.4. Zwischenbetrachtung

WEKA ist im Hochschulbereich eine populäre Data-Mining-Bibliothek, unterstützt allerdings keine öffentlichen Standards, die eine Integration in andere Programme erleichtern. Dagegen unterstützt die XELOPS-Bibliothek per Design CWM und PMML und ist derzeit für die Programmiersprachen Java, C++ und C# sowie für CORBA und Web Services implementiert.

JDM ist die Standard-Java-API für Data-Mining und ist offizieller Teil der Java 2 Enterprise Edition (J2EE). Für eine Java Data Mining Anwendung ist die Benutzung dieser API selbstverständlich.

Die UIMA von IBM ist ein Framework für Textanalyse und bietet eine Vielfalt an Analysetechniken wie statistische und rollenbasierte Verarbeitung natürlicher Sprache (NLP), Information Retrieval (IR), maschinelles Lernen, Ontologien und die Verlinkung dieser Prozesse zu strukturierten Informationsdiensten. In Zukunft wird auch die Einbindung von Sprach- und Videodaten möglich sein.

Die freien Bibliotheken und die UIMA können kostenlos eingesetzt werden und die Preise für die hier vorgestellten kommerziellen Softwarepakete sind vier bis sechsstellig.

Die beiden Tabellen 3.2 und 3.3 vergleichen die vorgestellten Implementierungen.

	WEKA	**JDM 1.0**	**JDM 2.0**	**XELOPS**	**UIMA**
Kapitel	3.1.1	3.1.2	3.1.3	3.1.4	3.2.1
Herausgeber/ Firma	Univer. Waikato	JCP	JCP	prudsys	IBM
aktuelle Ver.	3.4	1.0	2.0	1.3.1	1.0
Programmier-sprache	Java	Java	Java	Java, C++, C#, CORBA	Java, C++
Text-Mining	Nein	Nein	Ja	Nein	Ja

Tabelle 3.2.: Vergleich der freien Bibliotheken und Softwarearchitekturen

	Clementine	**MetaBase**	**SAS-9**	**Intellig. Miner**	**WebSphere OmniFind**
Kapitel	3.3.1	3.3.2	3.3.3	3.3.4	3.3.5
Firma	SPSS	MetaMatrix	SAS	IBM	IBM
aktuelle Ver.	9.0	3.0	9.0	8.2	8.2
Programmier-sprache	Java / C++	Java	SAS-spez.	Java	Java
Text-Mining	Ja	Nein	Ja	Nein	Ja

Tabelle 3.3.: Vergleich der kommerziellen Softwarepakete

4. Konzeptionierung eines IR-Frameworks

Dieses Kapitel beschreibt die Grundgedanken, die Entwicklung und Spezifikation des IR-Prozesses. Generell sind am Information Retrieval zwei (sich unter Umständen überschneidende) Personenkreise beteiligt.

Die erste Benutzergruppe, die Anwender, haben bestimmte, zum Zeitpunkt der Arbeit am IR-System akute Ziele oder Aufgaben, für deren Lösung ihnen Informationen fehlen. Diesen Informationsbedarf möchten Anwender mit Hilfe des Systems decken. Dafür müssen sie ihren Informationsbedarf in einer adäquaten Form als Anfrage formulieren.

Der zweite Personenkreis sind die Autoren, die Dokumente in einem IR-System zur Verfügung stellen. Dieses kann sowohl aktiv geschehen, indem die Autoren die Dokumente selber in das System einstellen, oder auch passiv geschehen, indem das System die Dokumente aus anderen verfügbaren Informationssystemen ausliest (wie es z. B. die Internet-Suchmaschinen praktizieren).

Dabei entsteht das Problem dass für jede Anfrage alle Dokumente durchsucht und bewertet werden müssen. Sind nur einige Dokumente zu berücksichtigen, kann das IR-System relativ schnell eine Antwort liefern. Bei vielen Dokumenten ist diese Vorgehensweise auf Grund der langen Antwortzeiten nicht immer sinnvoll. Die in das System eingestellten Dokumente werden vom IR-System in eine für die Verarbeitung günstige Form (Dokumentenrepräsentation) umgewandelt und zwischengespeichert. Eine Anfrage wird dann auf dem System-internen Modell der Repräsentation von Dokumenten ausgeführt. Ist die Aktualität der Suche sehr wichtig, oder eine schnelle Antwort des Systems ist nicht notwendig, dann kann für jede Suchanfrage der gesamte Datenbestand durchsucht werden. Die beiden Vorgehensweisen schließen sich nicht gegenseitig aus, sie ergänzen einander.

In Anlehnung an das EVA-Prinzip[1] haben sich drei hauptsächliche Phasen herausgestellt:

1. Die gewünschten Informationen müssen vom Benutzter in eine entsprechende Anfrage (Query) an das IR-System formuliert werden.
2. Das IR-System verarbeitet die Anfrage mit Hilfe von verschiedenen Methoden des Information Retrieval und des Data Mining (Analysis).
3. Die Präsentation der Analyseergebnisse (Results) müssen dem Benutzer angezeigt werden.

[1] EVA: Eingabe, Verarbeitung, Ausgabe

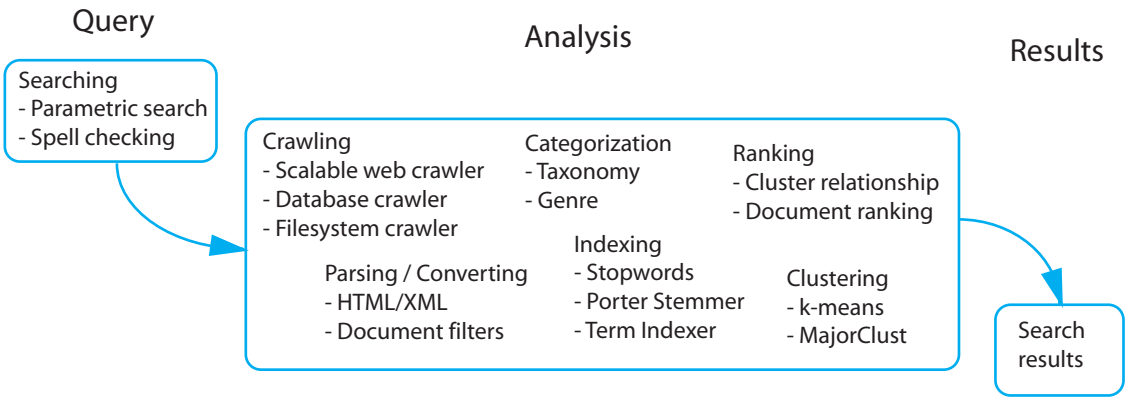

Abbildung 4.1.: Schlüsseltechnologien des IR-Prozesses

4.1. Komponenten

Der entwickelte IR-Prozess besteht aus verschiedenen Komponenten zum Einlesen von Daten (Crawling), Konvertierung, Indizierung, Kategorisierung und Relevanzbestimmung der Daten, sowie Komponenten zur Überprüfung der Suchanfrage. Eine Übersicht über die Komponenten ist in Abbildung 4.1 gezeigt.

1. *Searching* Die Suchanfrage beschreibt mit Hilfe von Schlüsselwörtern die Informationen die der Benutzer benötigt. Es werden diverse Typen von Anfragen unterstützt, wie Volltextsuche, Rechtschreibprüfung oder eigene Suchanfragen.

2. *Crawling* Crawler sammeln Daten aus unterschiedlichen Datenquellen, die durchsucht werden sollen. Eine einzelne Sammlung kann aus beliebig vielen Crawlern bestehen, und jeder Crawler sammelt Information aus einer bestimmten Datenquelle.

3. *Parsing / Converting* Der Parser analysiert Dokumente, die von einem Crawler gesammelt wurden, und bereitet sie auf die Indexierung und die relevanzbasierte Suche vor. Dabei verwendet der Parser eine Anzahl von sprachwissenschaftlichen Werkzeugen, um den Dokumenteninhalt und dessen Metadaten zu analysieren.

4. *Categorization* Kategorisierung ist der Prozess, einem gegebenen Dokument eine oder mehrere Kategorien zuzuweisen. Dabei werden regelbasierte Kategorisierungsmechanismen ebenso unterstützt wie Taxonomien.

5. *Ranking* Dokumente sind entsprechend ihrer Relevanz zu den Schlüsselwörtern eingeordnet, die ein Benutzer als Suchanfrage eintippt.

6. *Indexing* Indexer erzeugen Wortvektoren aus den Dokumenten, die dann von den Gruppierungsalgorithmen verarbeitet werden können. Dabei sollte die Größe der Wortvektoren möglichst klein gehalten werden.

7. *Clustering* Zur Gruppierung der Dokumente können verschiedene Verfahren benutzt werden wie k-means oder MajorClust.

Prozessmodule dienen der Verarbeitung von Informationen und Module zur Datenspeicherung werden Datenmodule genannt.

4.1.1. Prozessmodule

Die IR-Prozessmodule kapseln einzelne IR-Funktionalitäten und können einem komplexen IR-Prozess verschaltet werden. Die einzelnen Komponenten werden in Folgendem kurz vorgestellt:

1. *Websearch* Metainternetsuche nach einem durch den Benutzer spezifizierten Suchbegriff (Query). Die Suche liefert eine Ergebnisliste von im Internet zugänglichen Dokumenten, welche den Suchbegriff enthalten. Diese Ergebnisliste dient als Ausgangspunkt für die weitere Verarbeitung innerhalb des IR-Prozesses.

2. *File Upload* Neben den Ergebnissen der Internetsuche können auch lokale Dateien als Ausgangspunkt für den IR-Prozess dienen. Dafür müssen diese allerdings zuvor auf den Server hoch geladen werden. Dies leistet der File Upload.

3. *File Converter* Nachdem die Dateien auf den Server geladen wurden müssen diese für die weitere Verarbeitung vorbereitet werden indem sie in ein Einheitsformat überführt werden. Diese Funktionalität wird durch den File Converter zur Verfügung gestellt.

4. *Stopwords Remover* Diese Prozesskomponente entfernt so genannte Stopwörter aus den Dokumenten, welche die Internetsucheanfrage als Ergebnis geliefert hat. Stopwörter sind solche Wörter, welche im Prinzip keine wirkliche Information über den Inhalt eines Dokumentes liefern, da sie grundlegende Bestandteil der jeweiligen Sprache sind und somit gehäuft in verschieden Dokumenten vorkommen, unabhängig von deren thematischen Inhalt. Im deutschen sind z.B. „ist“, „und“, „oder“ solche Stopwörter.

5. *Porter Stemmer* Der Porter Stemmer führt die einzelnen Wörter der Dokumente auf ihrem Wortstamm zurück. Dieser Vorgehensweise geht die Annahme voraus, dass Wörter mit demselben Wortstamm mit demselben Sachverhalt in Zusammenhang stehen. Somit wird die Menge der zu verarbeitenden Wörter innerhalb des Dokumentes verringert, ohne dabei wesentliche inhaltliche Information zu verlieren.

6. *Indexer* Dokumente können als Wortvektoren repräsentiert werden. Dabei steht jede Koordinate des Vektors für ein Wort des Dokumentes. Taucht ein Wort mehrmals in einem Dokument auf, kann dies auch dargestellt werden indem die entsprechende Koordinate mit einem Gewicht (z.B. Anzahl des Vorkommens) multipliziert wird. Hat man nun mehrere Dokumente kann man, natürlich bei festgelegter Reihenfolge der Worte, anhand der Wortvektoren Ähnlichkeiten zwischen

den Dokumenten berechnen. Der Indexer erstellt für jedes Dokument innerhalb der Ergebnisliste den entsprechenden Wortvektor.

7. *Clustering* Clustern bedeutet aus einer gegebenen Grundmengen Untermengen zu bilden, wobei die Untermengen untereinander möglichst heterogen und die Elemente innerhalb der Untermenge möglichst homogen sein sollen. Außerdem wird gefordert, dass die Vereinigung der gebildeten Untermengen wieder zur Grundmenge führt. Für das Clustering stehen verschiedene Verfahren zur Verfügung z.B. kMeans, MajorClust etc.

8. *Labeler* Dieses Prozessmodul bestimmt repräsentative Namen für die gebildeten Cluster. Dies geschieht indem zuerst das Dokument bestimmt wird, welches repräsentativ für das Cluster ist und zwar ist dies das Dokument, welches am ähnlichsten zu allen anderen Dokumenten des Clusters ist. Aus diesem Dokument wird dann der am häufigsten auftretende Term extrahiert und zum Clusternamen bestimmt.

9. *Ranking* Beim Ranking geht es darum Dokumente, welche eine höhere Relevanz bezüglich des eingegebenen Suchbegriffs haben weiter oben in der Ergebnisliste zu präsentieren. Es wird also eine Rangfolge der Dokumente erstellt. Es kann auch sinnvoll sein eine Rangfolge der Dokumente innerhalb eines Clusters bezüglich des Clusternamens zu erstellen.

10. *Cluster Structure Generator* Die Cluster Structure Generator fügt die Dokumente, die generierten Cluster und Clusternamen zu einem gesamt Konstrukt zusammen, welches sich dann als Graph Visualisieren lässt. Dabei stellen die Cluster die Knoten des Graphen dar und die Kanten symbolisieren Ähnlichkeitsbeziehungen zwischen den Clustern. Die Idee dabei ist je heterogener zwei Cluster sind, desto länger ist der Weg entlang der Kanten den man innerhalb des Graphen gehen müsste um von einem Cluster (Knoten) zum anderen zu gelangen.

4.1.2. Datenmodule

Zur Speicherung von Ergebnissen und anderen Informationen gibt es Datenmodule, die an verschiedene Datenspeicher angepasst sind.

1. *XML-File* Dient der Speicherung der XML Daten in Dateien.

2. *XML-DB* Stellt eine Verbindung zu einer Datenbank her.

4.2. Realisierung

Eine Client-Server-Architektur ist für das IR-System sinnvoll, bei dem der Server den Zugriff der Clients auf die Dokumente ermöglicht. Die Analyse der Dokumente sollte direkt auf dem Server möglich sein, oder aus Performancegründen auf andere (dem Server

nahe stehende) Systeme ausgelagert werden können. In einem speziellen Sicherheitskontext kann dann verschiedenen Benutzern der Zugriff auf bestimmte Dokumente gewährt werden.

Als Architekturmuster bietet sich daher der *Rich Thin Client* an, bei dem die Applikationslogik sowohl im Client als auch im Server liegt. Da möglichst viele Benutzer mit dem System arbeiten können sollen, ist der Client möglichst plattformunabhängig zu realisieren. Der Server darf nicht monolithisch aufgebaut sein, sondern aus Modulen die voneinander unabhängig sind und sollte dem Client eine Schnittstelle bieten, mit dessen Hilfe die Funktionalität des Servers abgefragt werden kann.

Zur losen Koppelung der einzelnen Module bietet sich die Verwendung von Web Services[2] an. Es bietet die Möglichkeit der plattformunabhängigen Kommunikation zwischen den Modulen, durch die Verwendung der Protokolle HTTP, SOAP und WSDL (sowie XML zur Datenübertragung), ohne eine spezielle Programmiersprache vorzugeben. Die Registrierung der Module erfolgt dann mit Hilfe eines Verzeichnisdienstes auf Basis von UDDI.

Client

Der Client ist ein plattformunabhängiges Java Applet, das in einer Java-Laufzeitumgebung eines Webbrowsers (wie in Abbildung 4.2 [3]) ausgeführt werden kann. Die Aufgaben, die der Client erledigen muss sind:

- Abfragen der Module, die im Verzeichnis des Servers registriert sind
- Unterstützung beim Erstellen des Ablaufplans eines IR-Prozesses unter Berücksichtigung der möglichen Ein-/Ausgaben der Module
- Übertragen des Ablaufplans an den Server
- Überwachen und Anzeigen des Prozessstatus
- Darstellung der Ergebnisse ausgewählter Module

Für viele Ergebnisse reichen die Darstellungsfähigkeiten eines Standard-Browers völlig aus: beliebige Texte, Tabellen und Bilder können durch HTML im Browser angezeigt werden. Anders hingegen ist das komplexen Datenstrukturen, wie Graphen, objektorientierten Daten, mehr-dimensionalen Daten usw. bei denen erst die Interaktion mit den Daten eine Einsicht ermöglicht. Eine häufig verwendete Teilmenge von Graphen sind hierarchische Strukturen in Form von Bäumen, auf die im Kapitel 5 gesondert eingegangen wird.

[2] siehe Kapitel A.1.3; Web Service = HTTP + XML + SOAP + WSDL (+ UDDI + WSFL)

[3] aus der Vorlesung Web-Technologie von Benno Stein im Sommersemester 2005
`http://www.uni-weimar.de/cms/Lecture_Notes.550.0.html`

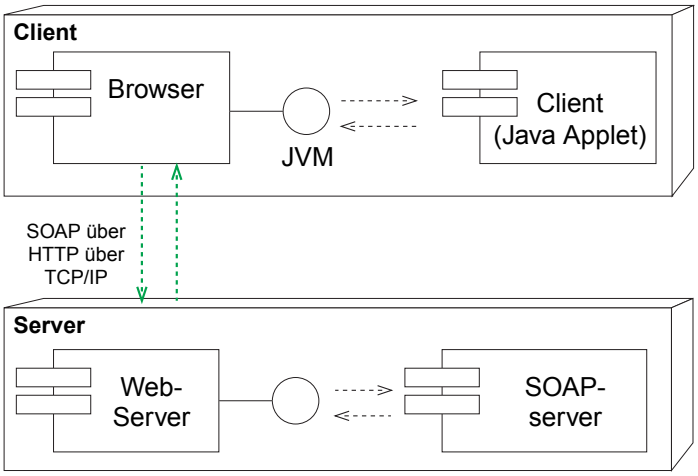

Abbildung 4.2.: Client-Server Kommunikation: HTTP + SOAP

Server

Der Server hat die grundlegenden Funktionen:

1. Bereitstellung der Web Services zur Ausführung von IR Prozessen.
2. Bereitstellung des Verzeichnisdienstes für Prozessor- und Datenmodule.
3. Auslieferung des Applets an den Browser des Benutzers.

Weiterhin stellt er die Skalierbarkeit sicher: er ermöglicht es, die Verarbeitung von Prozessmodulen an weitere Server zu delegieren.

Als Serverplattform bietet sich das Apache <Web Services/> Projekt[4] an, dass mit den Unterprojekten Axis und jUDDI eine Basis für die eigenen Implementierungen liefert. Axis ist eine Implementierung des SOAP Protokolls und schirmt den Entwickler von den Einzelheiten im Umgang mit SOAP und WSDL ab. Axis wird auf der Serverseite verwendet, um die Erstellung von Web Services zu erleichtern. Es bietet z.B. Tools zur Erstellung von Code Vorlagen (templates) für Web Services aus dem JDM Schema und dem JDM WSDL-Dokument.

jUDDI (ausgesprochen wie „Judy“) ist eine Open Source Java Implementierung der Universal Description, Discovery, and Integration (UDDI) Spezifikation und setzt genau wie Axis einen Applikationsserver mit Servlet 2.3 Unterstützung voraus wie Jakarta Tomcat, WebSphere, etc.

[4] `http://ws.apache.org`

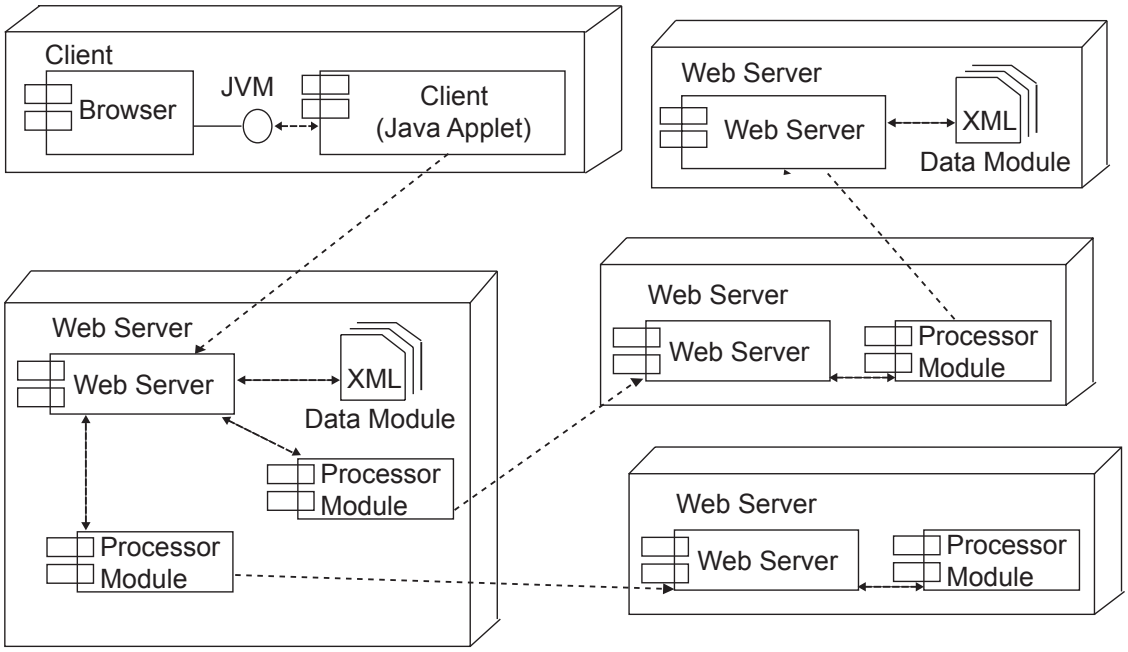

Abbildung 4.3.: Client-Server-Module

4.3. Datenformate

Die Eingabe- und Ausgabeformate der einzelnen Web Services werden in ihren WSDL-Dateien als XML-Schema definiert. Einfache Datentypen (double, int, string, ...) sowie zusammengesetzte Datentypen (complexType) lassen sich genau so gut für die Datenübertragung zwischen den Web Services benutzen, wie bereits vorhandene Schemata.

Die Datentypen der Datamining Algorithmen ist durch die Verwendung der JDM als Web Service vorgegeben. Für alle anderen sind entsprechende Schemata zu erstellen. Als konkretes Beispiel ist im weiteren das XML-Schema der Ergebnisse des Cluster Structure Generator erläutert. Dieses Beispiel ist ausgewählt worden, da genau diese Informationen im nächsten Kapitel 5 für die Visualisierung benötigt werden.

Die Ausgabe des Cluster Structure Generators ist ein Graph. Jeder Cluster wird durch genau einen Knoten repräsentiert und Kanten zwischen den Knoten stehen für die Ähnlichkeit von verschiedenen Clustern. Für die Kodierung des Graphen in XML wird GraphML[5] benutzt. GraphML kann Graphen strukturell beschreiben und bietet die Möglichkeit, an die Elemente des Graphen jede nur denkbare Art von Daten oder Attributen anzubringen. Für die Einbettung der zusätzlichen Informationen wurde das Schema[6] um spezielle Elemente und Attribute erweitert.

GraphML hat eine einfache und intuitive Struktur. Die Abbildung 4.4 zeigt einen Ausschnitt, in dem zwei Knoten durch eine Kante miteinander verbunden ist.

[5] http://graphml.graphdrawing.org/
[6] http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd

```
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
  <graph xmlns:ai="http://www.uni-paderborn.de/kbs/ai">
    ...
    <node id="n0">
      <data key="ai:text">debian&#xA;ubuntu</data></node>
    <node id="n1">
      <data key="ai:figure">gd06-logo.png</data></node>
    <edge source="n0" target="n1"/>
    ...
  </graph>
</graphml>
```

Abbildung 4.4.: Einfacher Graph mit zwei Knoten

In diesem Kapitel wurde eine Übersicht über die verwendeten Komponenten und deren Realisierung im IR-Framework gegeben, welches im Rahmen dieser Diplomarbeit entstanden ist. Auf Grund der enormen Komplexität der Prozesse wurde von Beginn an darauf verzichtet, das Framework bis in alle Details auszuarbeiten, was innerhalb des Zeitrahmens auch nicht möglich gewesen währe.

Das nächste Kapitel beschreibt die Realisierbarkeit des Frameworks an Hand von einem Datenvisualisierungsmoduls.

5. Modul zur Visualisierung von hierarchischen Strukturen

Zur Visualisierung der entstehenden (Zwischen-) Ergebnisse müssen Komponenten entwickelt werden, die die Ergebnisse optimal darstellen können. Da die Entwicklung aller benötigen Visualisierungskomponenten des Frameworks zu viel Zeit für eine Diplomarbeit in Anspruch nehmen würde, werden hier nur Komponenten zur Darstellung von einem speziellen Typ erarbeitet. Dies sind Ergebnisse die in Form von Bäumen vorliegen.

Dazu wird zunächst eine Übersicht von bekannten hierarchischen Darstellungen gegeben. Die im Rahmen dieser Diplomarbeit entstandene Verallgemeinerung bildet den Mittelteil dieses Kapitels und dessen Realisierung schließt das Kapitel ab.

Die Bäume, auf die sich im folgenden beschränkt wird, sind ungerichtete, zusammenhängende, azyklischen Graphen mit ausgezeichneter Wurzel(-knoten), so dass von diesem Knoten zu jedem weiteren Knoten genau ein Pfad existiert. Ist v ein Knoten in einem Baum, so ist das *Level* von v die Länge des (eindeutigen) Pfades von der Wurzel zu v. Jeder von v verschiedene Knoten w auf diesem Pfad heißt *Vorgänger* von v. In diesem Zusammenhang heißt v *Nachfolger* von w. Ist (v, w) eine Kante in einem Baum, so nennen wir v den *Vater(-knoten)* von w und w das *Kind* von v. Sind w_1 und w_2 zwei verschiedene Kinder von v, so bezeichnen wir sie als *Geschwister*. Ein Baum, dessen Knoten jeweils höchstens zwei Kinder besitzen, heißt *Binärbaum*. Ein *geordneter* Baum ist ein Baum, welcher die Eigenschaft besitzt, dass die Kinder eines jeden Knoten v eine festgelegte Reihenfolge haben. Ein Knoten im Baum, der keine Kinder besitzt, heißt *Blatt*. *Spiegelung* eines Baumes bedeutet, dass für jeden Knoten v im Baum die Reihenfolge der Kinder invertiert wird.

Einen Baum in einer Ebene zu zeichnen, bedeutet jedem Knoten v ein Koordinatenpaar (x_v, y_v) und jeder Kante zwischen zwei Knoten v und w eine Strecke zuzuweisen, die die Punkte (x_v, y_v) und (x_w, y_w) in der Ebene verbindet.

Es ist nun zu erwarten, dass, wenn ein Baum in eine Ebene gezeichnet wird, gewisse ästhetische Forderungen an das Layout gestellt werden. Sonst wäre es möglich, alle Knoten an der selben Stelle (x, y) durch Punkte darzustellen. Der gezeichnete Baum wäre dann ein Punkt. In der Literatur (z.B.:[GDAlgo99]) finden sich verschiedene Eigenschaften von (Graph-)Zeichnungen:

- *planar* Knoten haben verschiedene Positionen und Kanten kreuzen sich nicht.

- *straight-line drawing* Kanten sind gerade Linien.
- *polyline drawing* Kanten sind Polylinien (stückweise gerade Linien).
- *orthogonal drawing* Kanten sind eine Folge von horizontalen und vertikalen Linien.
- *grid drawing* Knoten und Ecken von Kanten haben ganzzahlige Koordinaten.
- *strong visibility representation* Knoten sind horizontale Linien und Kanten sind vertikale Linien.
- *upward drawing* Alle Kanten „zeigen“ in die gleiche Richtung (z.B. von oben nach unten, oder von links nach rechts).

Schon 1970 fragte sich D.E. Knuth „How shall we draw a tree?“ Er entwickelte einen Algorithmus, der Bäume top-down zeichnet. Die Möglichkeiten der graphischen Darstellung am Computer genügten seinen Ansprüchen jedoch nicht, da damals nur (Text-) Sonderzeichen zur Verfügung standen. Seitdem hat sich viel geändert: sowohl die Computer haben bessere, graphische Möglichkeiten, als auch die Algorithmen haben sich stark verbessert.

Alle hier vorgestellten Algorithmen liefern planare, straight-line Zeichnungen, sofern nicht anders angegeben. Sie gehen rekursiv nach dem „teile und herrsche“-Prinzip vor. Zunächst werden Layoutalgorithmen vorgestellt, die ein radiales Layout liefern, danach ein Verfahren, dass den Baum von links nach rechts zeichnet und zuletzt wird eine Explorer-artige Darstellung vorgestellt.

5.1. Hierarchische Darstellungen

Radiale Zeichnungen kennzeichnet die Eigenschaft, dass Knoten auf Kreisen um den Wurzelknoten herum angeordnet sind. Dabei gilt der Level eines Knotens entspricht dem radialen Abstand des Kreises von der Wurzel.

5.1.1. Radiales Layout

Ein einfacher Algorithmus zur Erstellung eines radialen Layouts für einen Binärbaum, unter Benutzung von kartesischen Koordinaten (ϕ, r), ist folgender:

1. Der Radius r eines Knotens v entspricht dessen Level $l(v)$.
2. Der Winkel ϕ wird für die
   - Blätter einfach gleichmäßig aufgeteilt, und die
   - restlichen Knoten werden zwischen ihre Kinder gesetzt.

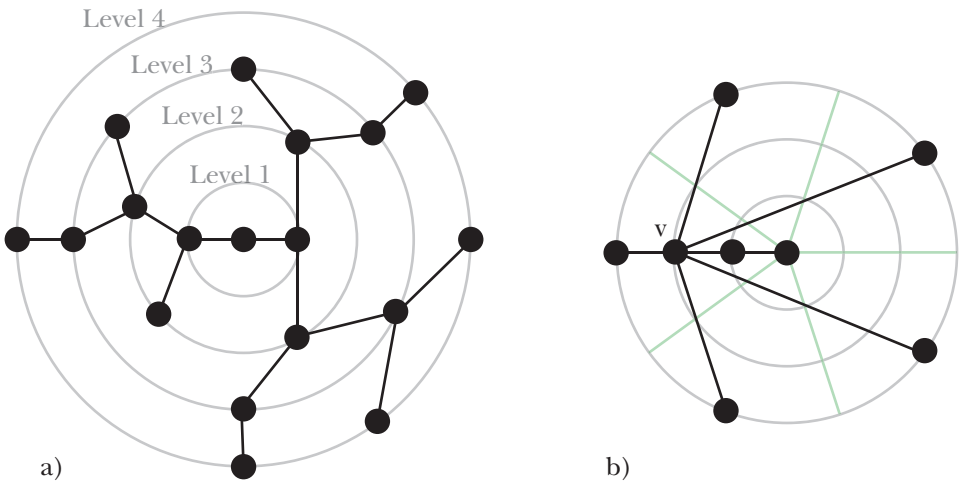

Abbildung 5.1.: Beispiele des einfachen radialen Algorithmus

In Abbildung 5.1 sind zwei Beispiele abgebildet. In *a*) sind zur Verdeutlichung die Levelkreise eingezeichnet und in *b*) zusätzlich die Sektoren, in denen die Blätter untergebracht sind. Für vollständige Bäume liefert dieser Algorithmus eine optimale Aufteilung. Ein großer Nachteil dieser Vorgehensweise ist, dass für allgemeine Bäume nicht immer die Hierarchie deutlich zu erkennen ist. Dies ist der Fall, wenn der Winkel zwischen den Kindern zu groß wird. Zu sehen ist das zum Beispiel in Abbildung 5.1 b) am Knoten v. Der Knoten hat 5 Kinder, die gleichzeitig auch die einzigen Blätter sind.

P. Eades entwickelte im Jahre 1992 ein Verfahren dass die oben genannten Schwächen nicht hat[1]. Er stellte die zusätzliche Forderung, dass Kanten die Levelkreise nur in Knoten schneiden. Dadurch verringert sich der Winkel in dem die Nachfolger eines Knotens gezeichnet werden können. In Abbildung 5.2 a) verdeutlicht die rote Linie für den Knoten v diesen Zusammenhang. Die Hierarchie ist deutlich zu erkennen. Es gibt jetzt zwei Bedingungen die eingehalten werden müssen, von denen die jeweils engeren für den Teilbaum benutzt werden. Im Beispiel a) greift die Level-Beschränkung und in b) die Sektor-Beschränkung für den Knoten v.

Bezeichnet $l(v)$ die Anzahl der Blätter unterhalb des Knotens v und $r(v)$ den Radius des Levelkreises des Knotens v, so ist der für jedes Kind u von v zur Verfügung stehende Winkel β_u definiert als

$$\beta_u = \min\left(\frac{l(u)\beta_v}{l(v)}, 2\cos^{-1}\left(\frac{r(v)}{r(u)}\right)\right)$$

Damit steht ein lineares Verfahren zur Verfügung, dass es erlaubt für Bäume von unbeschränktem Grad und Tiefe ein planares Layout zu erzeugen. Der Abstand der Levelkreise ist so zu wählen, dass die Knoten sich nicht überlappen und der Zeichenfläche optimal ausgenutzt wird. Das Problem dabei ist, es gibt keine Zeichenfläche von unbeschränkter Größe. Sind pro Knoten viele Informationen darzustellen, so belegen die

[1] P. Eades: drawing free trees, [Eades92]

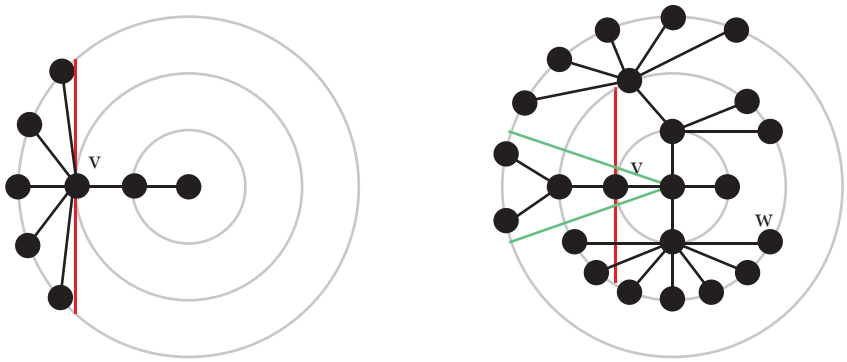

Abbildung 5.2.: Weitere Beschränkung des einfachen radialen Algorithmus

Knoten eine entsprechende Fläche und die Abstände der Levelkreise sind anzupassen. Schon bei kleinen Graphen kann es daher vorkommen, das die zur Verfügung stehende Zeichenfläche nicht ausreicht, um alle Knoten gleichzeitig darzustellen.

Es gibt verschiedene Möglichkeiten diesem Problem zu begegnen und eine davon wird im nächsten Unterkapitel erläutert.

5.1.2. Verzerrungen der Ebene

Das Grundproblem bei der Gestaltung graphischer Darstellungen besteht darin, dass große Datenmengen auf kleinem Raum dargestellt werden müssen. Die Darstellung wird daher oft sehr komplex und damit unübersichtlich. Sie kann von den Benutzern nicht bzw. nur mit großem Aufwand kognitiv verarbeitet werden.

Es gibt verschiedene Visualisierungstechniken, mit denen versucht wird, diesem Problem zu begegnen. Zum Beispiel könnte den Benutzern immer nur ein Ausschnitt aus dem Gesamtgraphen gezeigt werden. Dieser Ansatz ist allerdings problematisch, da der Gesamtkontext verloren geht: Die Benutzer erkennen zwar Details ihrer unmittelbaren Umgebung, können jedoch aus der Übersicht nicht ableiten, an welcher Stelle des Graphen sie sich befinden. Wenn Wert auf den Gesamtkontext gelegt wird, sollte das gesamte Netz dargestellt werden. Allerdings muss dann in der Regel auf Details verzichtet werden.

Besonders häufig werden die Benutzer an Detailinformationen zu ihrer unmittelbaren Umgebung interessiert sein. Daher kann es sinnvoll sein, diese Informationen direkt in die graphische Übersicht zu integrieren.

Einen Schritt weiter gehen Ansätze, bei denen wichtige Knoten groß im Vordergrund und unwichtige Knoten klein im Hintergrund dargestellt werden. Dies ist eine sehr gute Methode, um diejenigen Teile des Graphen, die im Mittelpunkt des Interesses (Focus) stehen, detailliert anzuzeigen, ohne dass der Kontext verloren geht. Zur Verdeutlichung dieses Vorgehen werden im folgenden einige Beispiele angegeben. Dabei erhält die gra-

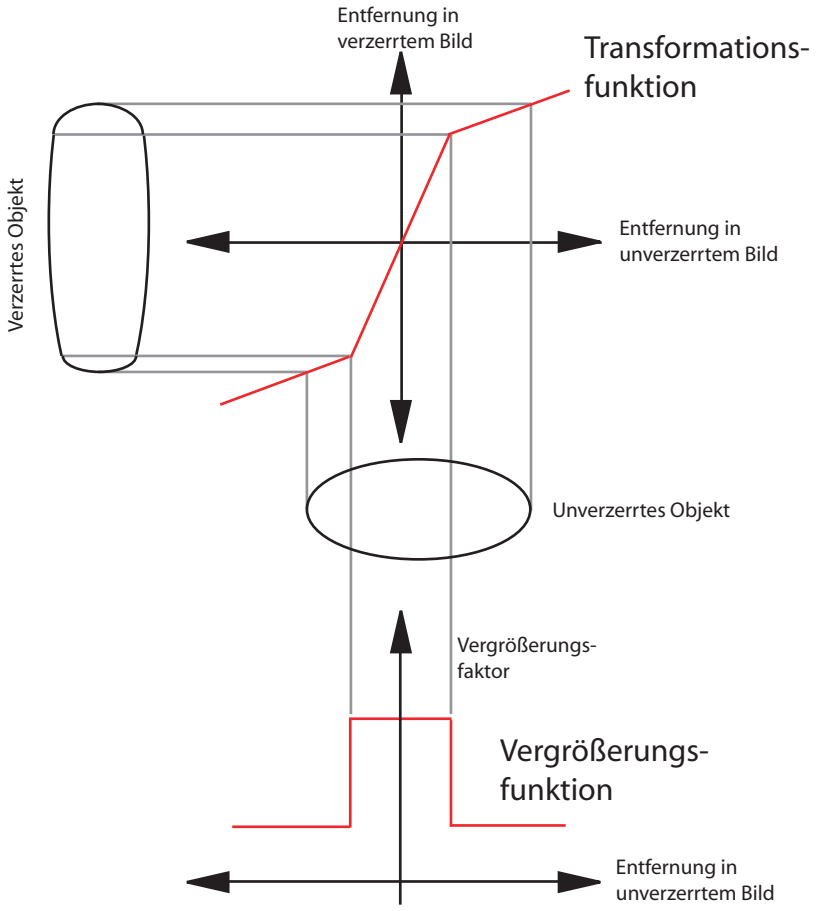

Abbildung 5.3.: Transformations- und Vergrößerungsfunktion

phische Darstellung zunächst ein herkömmliches zweidimensionales Layout. Danach wird durch eine geeignete Funktion die Ebene verzerrt und der das Layout der Verzerrung angepasst. Wie eine solche Transformation graphisch dargestellt werden kann ist in Abbildung 5.3 zu sehen. Das unverzerrte Objekt wird dabei durch eine eindimensionale Verzerrung so verändert, dass das Mittelteil größer und die Ränder kleiner erscheinen als im Original.

Die Abbildung 5.4 zeigt oben die Transformationsfunktion $T(x) = x$ und die Vergrößerungsfunktion $M(x) = 1$, die sich ergibt, wenn keine Verzerrung stattfindet. Sowie im unteren Teil ein Gitter und ein Graph, an denen später die Verzerrungen der Ebene illustriert werden.

Die Transformation kann sich entweder nur auf die Position der Knoten auswirken, oder auch gleichzeitig auf die Größe der Knoten, wie hier im weiteren angenommen.

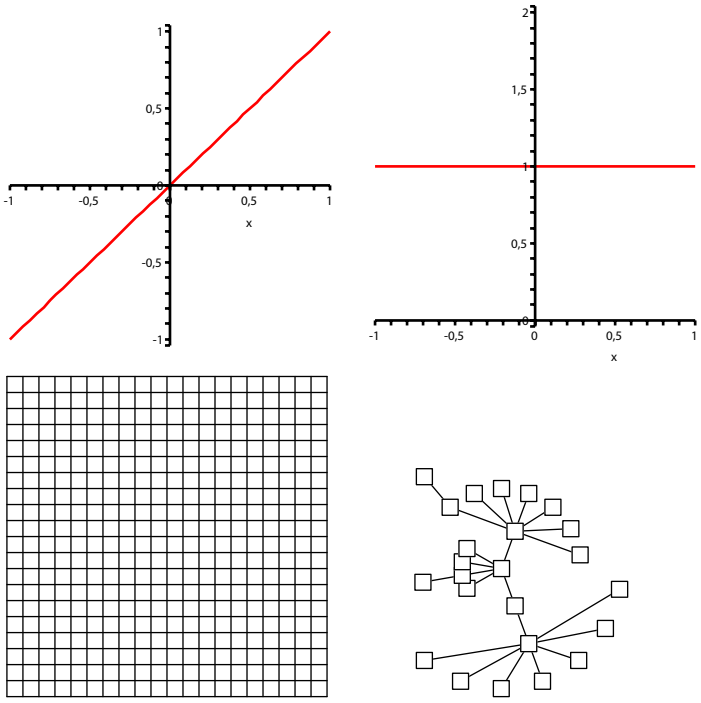

Abbildung 5.4.: keine Verzerrung

Perspective Wall

Zunächst wird nun ein Bereich bestimmt, der als zentral gilt und die meisten Detailinformationen zeigen soll. Dieser Bereich bildet den Vordergrund bzw. die Frontwand der Perspective Wall. Die anderen beiden Bereiche, die links bzw. rechts an den zentralen Bereich angrenzen, werden gedanklich nach hinten geklappt und nur noch verzerrt dargestellt (siehe Abbildung 5.5 c). Sie bilden die Seitenwände der Perspective Wall und haben die Funktion, die Detailinformationen des zentralen Bereiches in einen Kontext einzubetten.

Die Transformationsfunktion $T(x)$ der Perspective Wall ist in zwei Bereiche aufgeteilt und lautet

$$T(x) = \quad x\frac{b}{a} \quad \text{für} \quad x \leq a, \text{bzw.} \qquad \frac{b + (x-a)\cos\theta}{1 - (\frac{1-b}{1-a} - \cos\theta)(x-a)} \quad \text{für} \quad x > a,$$

wobei für die Darstellung in Abbildung 5.5 die Werte $a = 0.2$, $b = 0.6$ und $\theta = \Pi/6 = 30°$ (also $\cos\theta = 0.866$) gewählt wurde. a) zeigt die Transformations- und Vergrößerungsfunktion, b) veranschaulicht die Anwendung der Transformation auf die x-Achse des Gitters und des Graphen, c) zeigt die Transformation in x- und y-Richtung und d) illustriert die Anwendung der Transformation auf den Abstand zum Mittelpunkt (Radius in Poolarkoordinaten).

Fisheye View

Graphische Fisheye Views wurden zur Darstellung von topologischen Netzwerken entwickelt und basieren auf kontinuierlichen Skalierungsfunktionen. Die Transformationsfunktion bezieht sich nur auf die Knoten des Netzwerks und ist sowohl für kartesische als auch für Polarkoordinaten definiert.

In den ursprünglich von Furnas entwickelten Fisheye Views wurde für jeden Knoten sein Interessantheitsgrad ermittelt und in der Zeichnung weggelassen, falls dieser einen Schwellenwert nicht überschritt. Die Graphical Fisheye Views unterscheiden sich darin, dass unwichtige Knoten nicht ganz weggelassen, sondern nur kleiner dargestellt werden.

Die Transformationsfunktion $T(x)$ und die Vergrößerungsfunktion $M(x)$ sind:

$$T(x) = \frac{(d+1)x}{(dx+1)} \qquad M(x) = \frac{d+1}{(dx+1)^2}.$$

Dabei ist d der Verzerrungsfaktor, d.h., je größer d, desto größer ist der Skalierungsfaktor in der Fokusregion. In Abbildung 5.6 a) sind die Funktionen mit einem Wert von $d = 4$ gezeichnet.

Bei stetiger Vergrößerungsfunktion ist die Transformationsfunktion „glatt“, d.h. ohne Kanten.

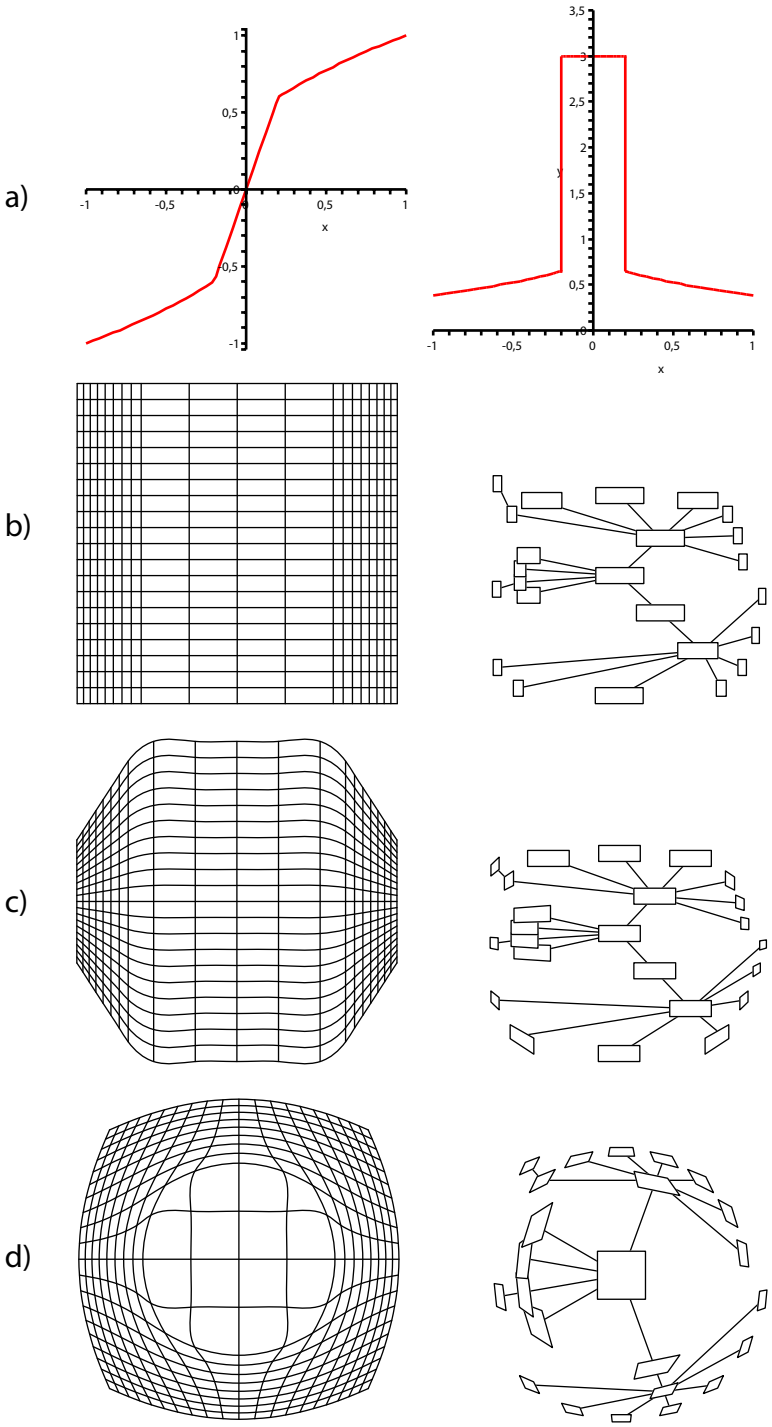

Abbildung 5.5.: Perspective Wall

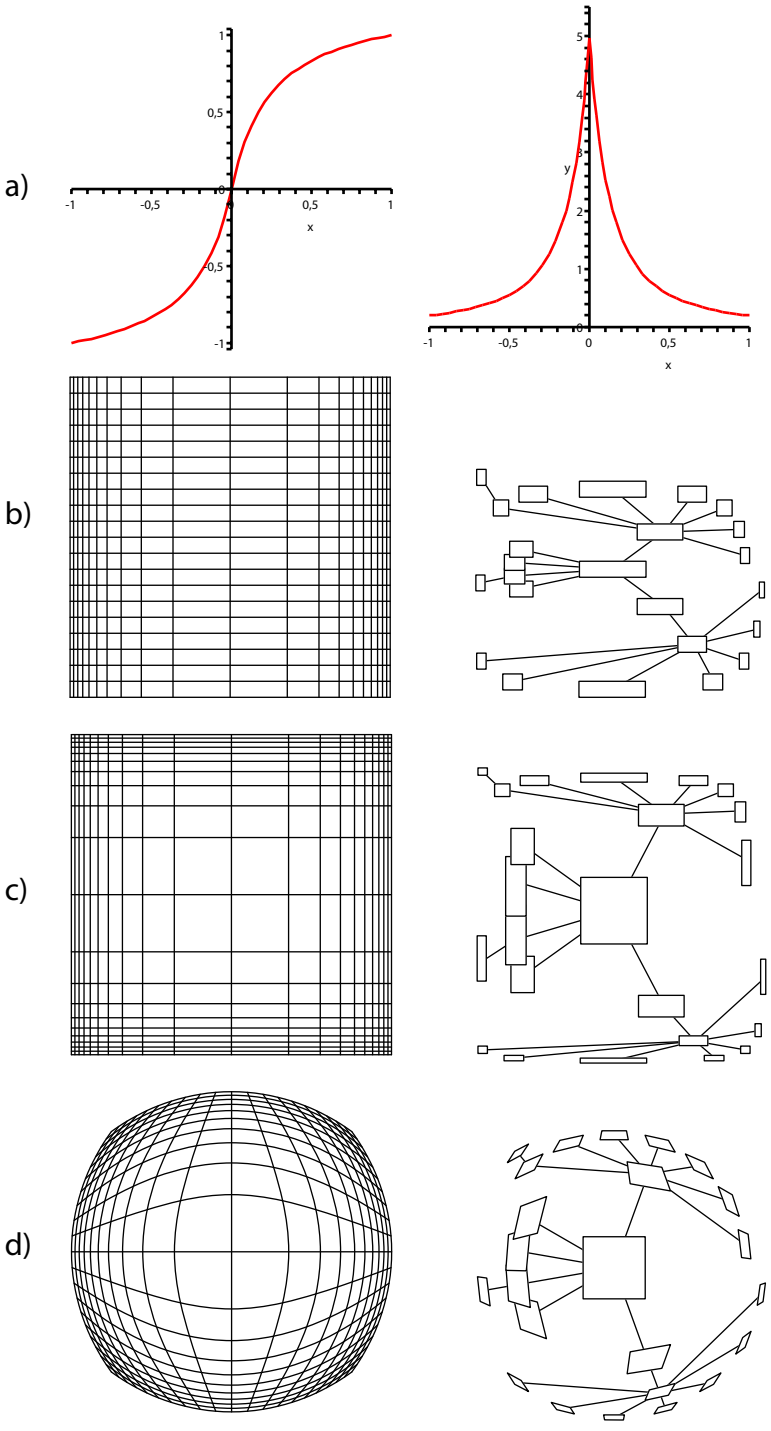

Abbildung 5.6.: Fisheye View

Polyfocal Projection

Im Allgemeinen ist die Vergrößerungsfunktion immer monoton, nur beim Polyfocal Display wird der hohe Platzverbrauch im Zentrum durch vorübergehende Verkleinerung ausgeglichen. Dadurch erreicht man ein weitgehend unverzerrtes Bild im Randbereich. Weitere positive Eigenschaften polyfokaler Darstellungen sind die glatten Übergänge und dass multiple Foki möglich sind. Dabei entsteht allerdings auch ein hoher Rechenaufwand.

Die Transformationsfunktion $T(x)$ und die Vergrößerungsfunktion $M(x)$ sind in diesem Fall:

$$T(x) = x + \frac{Ax}{1 + Cx^2} \qquad \text{und}$$

$$M(x) = 1 + \frac{A(1 - Cx^2)}{(1 + Cx^2)^2},$$

wobei A und C Konstanten sind, die für die Darstellungen in Abbildung 5.7 zu $A = 2$ und $C = 10$ gesetzt wurden.

Bifocal Display

Die 1-dimensionale Variante des Bifocal Displays kombiniert einen Fokusbereich mit 2 verzerrten Seitensichten, wobei für jede Seitensicht ein einheitlicher Skalierungsfaktor gewählt wird (vgl. Abb. 5.8 b).

Bei der 2-dimensionalen Variante (vgl. Abb. 5.8 c) wird das Bild in 9 Bereiche unterteilt: 8 Randbereiche und eine zentrale Fokusregion. Die Variante wurde für den Londoner U-Bahn-Plan entwickelt.

Zu den Vorteilen der Bifokalen Technik zählen eine einfache Implementierbarkeit und schnelle Berechenbarkeit, sowie kontinuierliche Verläufe innerhalb eines Bereiches.

$$T(x) = \quad x\frac{b}{a} \quad \text{für} \quad x \leq a, \quad \text{bzw.} \qquad b + (x - a)\frac{1 - b}{1 - a} \quad \text{für} \quad x > a$$

$$M(x) = \quad \frac{b}{a} \quad \text{für} \quad x \leq a, \quad \text{bzw.} \qquad \frac{1 - b}{1 - a} \quad \text{für} \quad x > a$$

Die Haupteigenschaft dieser Techniken ist es, dem Benutzer zu erlauben einen lokalen Bereich zu untersuchen, während zur gleichen Zeit der globale Kontext sichtbar ist.

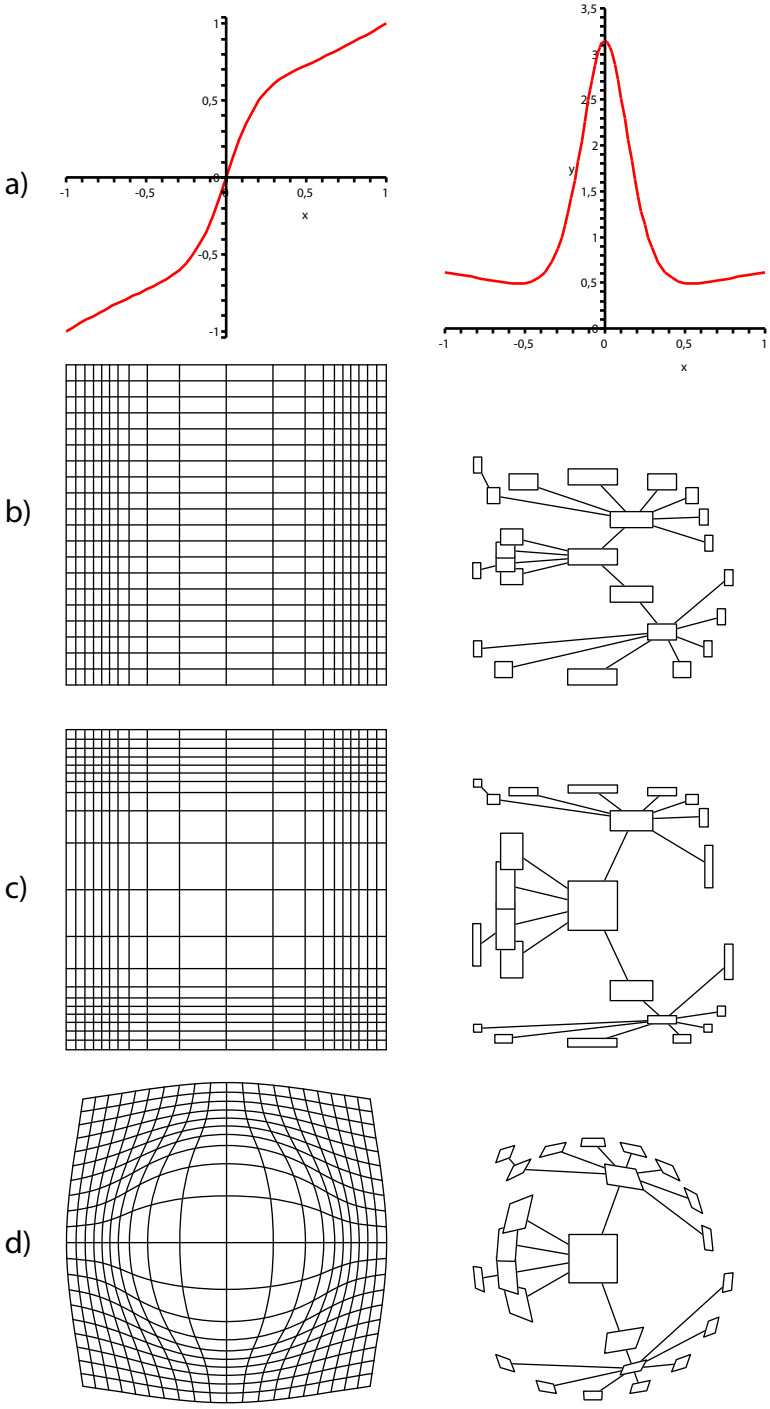

Abbildung 5.7.: Ployfocal Projection

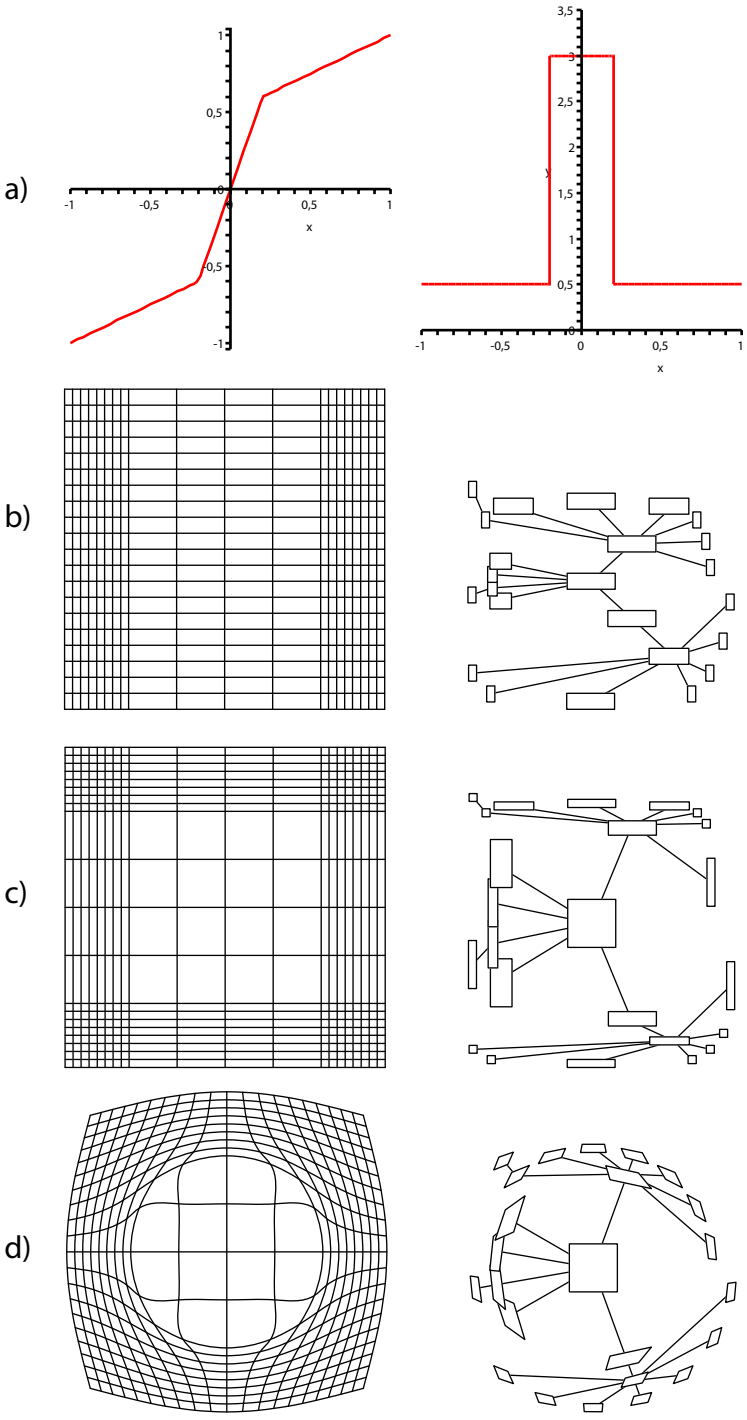

Abbildung 5.8.: Bifocal Display

5.1.3. Eigenschaften von hierarchischen Baumzeichnungen

Im folgenden sind die wichtigsten und am häufigsten benutzten Layoutbedingungen für Baumzeichnungen aufgeführt. Es sind Bool'sche Eigenschaften, die ein bestimmter Zeichnungsalgorithmus entweder erfüllt oder nicht.

Bool'sche Bedingungen für binäre Bäume mit gleichgroßen Knoten:

(1) Die y-Koordinate eines Knotens entspricht dem Level des Knotens.

(2) Ein linker Kindknoten ist links von seinem Vaterknoten positioniert und ein rechter Kindknoten rechts von seinem Vaterknoten.

(3) Ein Vaterknoten ist zentriert über seinen Kindern.

(4) Die Kanten kreuzen sich nicht und Knoten auf dem selben Level haben einen vorgegebenen horizontalen Minimalabstand.

(5) Unterbäume, die strukturgleich sind, werden bis auf Verschiebung gleich gezeichnet.

(6) Die Reihenfolge der Kinder eines Knotens wird in der Zeichnung dargestellt.

(7) Der Zeichenalgorithmus arbeitet symmetrisch, das heißt, dass die Zeichnung der Spiegelung des Baumes gleich der Spiegelung des gezeichneten Baumes ist.

Zusätzliche Bool'sche Bedingungen für n-äre Baume mit gleichgroßen Knoten:

(8) Ein Vaterknoten ist zentriert über den Zentren seines linkesten und rechtesten Sohnknotens.

(9) Horizontale Abstände zwischen Knoten des gleichen Levels haben minimale Varianz.

Zusätzliche Bool'sche Bedingungen für n-äre Bäume mit unterschiedlichen Knotengrößen

(10) Jeder Konten hat einen vorgegebenen, vertikalen Minimalabstand zu seinem Vater.

(11) Ein Vaterknoten ist zentriert über der linken Begrenzung seines linkesten Sohnes und der rechten Begrenzung seines rechtesten Sohnes.

(12) Die obere Begrenzung aller Nachfolger eines Knotens liegen auf einer horizontalen Linie.

(13) Knoten und Kanten überschneiden sich nicht.

Eine einfache Methode jedem Knoten eines binären Baumes eine Position zuzuweisen besteht darin, das Level des Knotens als eine y-Position zu verwenden und als x-Position den Rank aus einem inorder Durchlauf des Baumes. Ist ein Knoten der i-te entdeckte Knoten des Durchlaufs, so ist sein Rank gleich i. Ein solches Layout hat die Eigenschaften (1), (2), (A4) und (6). Ein Beispiel eines sich so ergebenden Darstellung ist in Abbildung

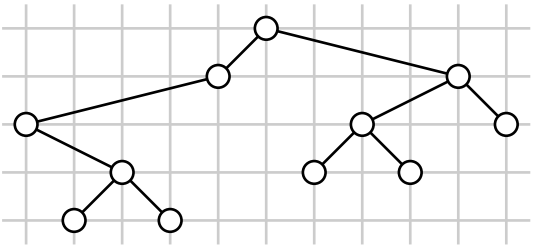

Abbildung 5.9.: Beispiel eines einfachen levelbasierten Algorithmus

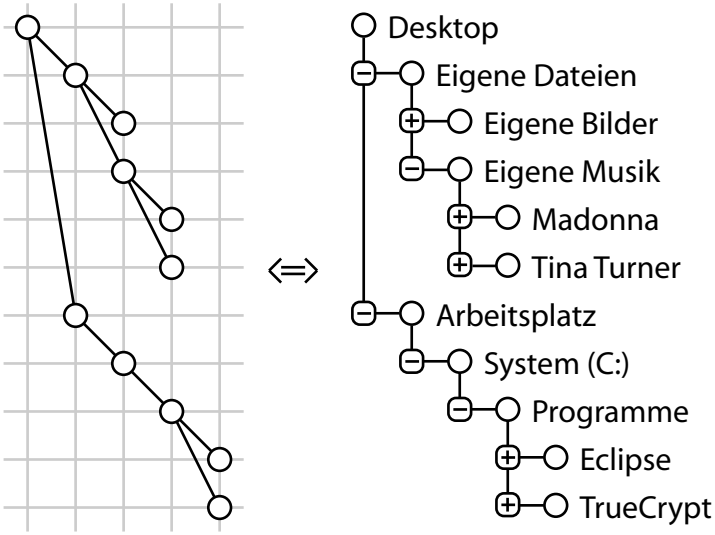

Abbildung 5.10.: Einfacher vertikaler Algorithmus

5.9 zu sehen. Wie jedoch deutlich zu erkennen, hat die Zeichnung zwar optimale Höhe, ist jedoch viel zu Breit. Die Breite entspricht der Anzahl der Knoten des Baumes.

Eine interessante Variation, die jedoch einige Bedingungen nicht erfüllt, ist in Abbildung 5.10 gezeigt. Trotz des Platzanspruches ist diese Variante in vielen Softwaresystemen implementiert worden.

5.1.4. Reingold Tilford Algorithmus

Der im Jahre 1990 vorgestellte Algorithmus von Walker wird in vielen Bereichen genutzt, um Bäume von unbeschränktem Grad zu zeichnen. Bis zu diesem Zeitpunkt waren nur Algorithmen bekannt, die geordnete Binärbäume in Linearzeit zeichnen. Zu nennen ist dabei der Algorithmus von Wetherell und Shannon, der 1979 präsentiert wurde und bereits in der Lage war, die Ordnung und Hierarchie der Knoten zu visualisieren. Einen deutlich besseren Linearzeitalgorithmus präsentierten Reingold und Tilford 1981. Dieser Algorithmus basiert auf dem Algorithmus von Wetherell and Shannon, verbessert diesen

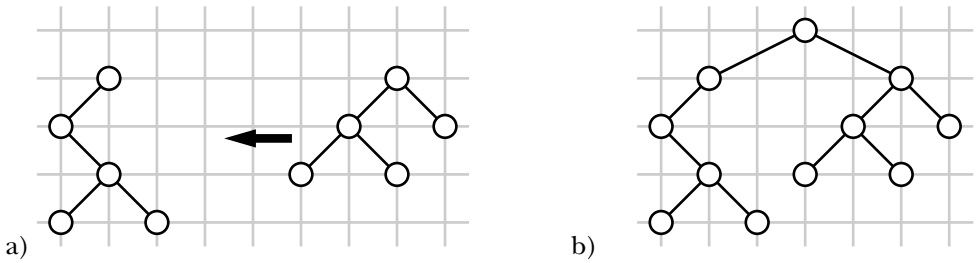

Abbildung 5.11.: Vorgehensweise des Reingold Tilford Algorithmus

jedoch in der Weise, dass strukturgleiche Unterbäume bis auf Verschiebung gleich gezeichnet werden und das Vorgehen des Algorithmus symmetrisch ist. Jedoch beschränkt sich auch die Funktionalität dieses Algorithmus auf Binärbäume. Erst neun Jahre später stellte Walker einen Algorithmus vor, der Bäume von unbeschränktem Grad zeichnen konnte. In seiner Arbeit „A node-positioning algorithm for general trees“ [[Walker90]] behauptet Walker, dass dieser Algorithmus linear in der Anzahl der Knoten des zu zeichnenden Baumes arbeite. Es stellte sich jedoch heraus, dass der Algorithmus nicht in Linearzeit arbeitet. In der Arbeit „Improving Walker's Algorithm to Run in Linear Time“ [[BuchLeip02]] werden entsprechende Korrekturvorschläge gegeben, um eine lineare Laufzeit zu gewährleisten.

Alle Arbeiten gehen davon aus, dass alle Knoten eine feste Größe besitzen, bzw. nur aus einem Punkt bestehen. Im letzten Teil dieses Abschnittes wird darauf eingegangen, wie der Walker-Algorithmus zu erweitern ist, damit auch Bäume mit unterschiedlichen Knotengrößen verarbeitet werden können.

Der erste Linearzeitalgorithmus, welcher geordnete Binärbäume unter Berücksichtigung von (1) - (7) zeichnete, wurde 1981 von Reingold und Tilford [ReinTil81] vorgestellt. Zunächst wird die rekursive Vorgehensweise des Algorithmus beschreiben, dann wird auf die Besonderheiten in Bezug auf Laufzeit hinweisen und schließlich wird Vorgehensweise des Algorithmus anhand eines Beispiels veranschaulicht.

Ein Blatt wird in der Weise gezeichnet, dass ihm eine vorläufige x-Koordinate und eine dem Level entsprechende y-Koordinate zugewiesen wird. Nachdem die Unterbäume, die durch die Kinder eines Knotens v induziert werden, gezeichnet wurden, wird der rechte Unterbaum so verschoben, dass er so nahe wie möglich auf der rechten Seite des linken Unterbaumes platziert ist. Daraufhin wird v über seinen Kindern zentriert (vgl. Abbildung 5.11 a) und b)). Existiert nur der linke (rechte) Unterbaum, so wird der Vaterknoten um eins weiter rechts (links) verschoben. Offensichtlich erfüllt dieser Algorithmus (1) - (7). Weniger offensichtlich ist aber, dass dieses Vorgehen tatsächlich in Linearzeit möglich ist. Die in Bezug auf Laufzeit kritischen Stellen im Algorithmus sind:

1. Das Berechnen der neuen Position des rechten Unterbaumes der aktuellen Wurzel.

2. Das Verschieben des rechten Unterbaumes der aktuellen Wurzel.

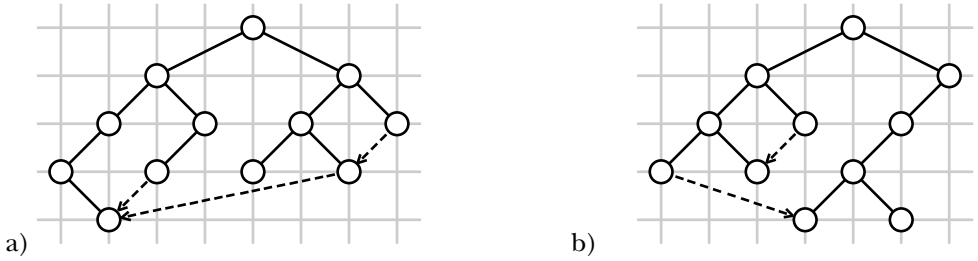

Abbildung 5.12.: Threads des Reingold Tilford Algorithmus

Zur Lösung des Problems führten Reingold und Tilford Threads und ein Offset pro Knoten ein.

Ein Thread (engl. für Faden) bezeichnet einen Pointer auf den nächsten Knoten in der aktuellen Kontur eines Unterbaumes. Vergleicht der Algorithmus eine linke mit einer rechten Kontur, so muss nicht zunächst der jeweils ganz rechts liegende Knoten des linken Teilbaums und der ganz links liegende Knoten des rechten Teilbaums gesucht werden. Wird ein längerer linker Teilbaum mit einem kürzeren rechten Teilbaum zusammengesetzt, so wird dessen rechte Kontur am untersten Blatt des rechten Teilbaums mit Hilfe eines Threads im linken Teilbaum fortgesetzt. Die Abbildung 5.12 verdeutlicht diesen Zusammenhang an zwei verschiedenen Bäumen. Die gestrichelten Linien entsprechen den Threads, die zum einfachen Durchlaufen der Kontur hinzugefügt werden. Wird ein längerer linker Unterbaum zu einem kürzeren rechten Unterbaum hinzugefügt, so wird ein Thread hinzugefügt, der die rechte Kontur des gesamten Teilbaums vervollständigt. Hierbei zeigt der Thread von dem untersten Knoten der rechten Kontur des rechten Unterbaumes auf den Knoten der rechten Kontur des linken Unterbaums dessen Level um eins größer ist als der unterste Knoten des rechten Unterbaums. Dieser Fall tritt in Abb. 5.12 a) drei mal auf und in b) genau ein mal. Der umgekehrte Fall, dass ein kürzerer linker Unterbaum und ein längerer rechter Unterbaum zusammengeführt werden, verläuft analog dazu. Hier wird die linke Kontur mit Hilfe des Threads fortgesetzt und ist in b) einmal aufgezeigt.

Der Algorithmus benötigt zwei Durchläufe durch den Baum um die tatsächlichen Positionen zu bestimmen. Im ersten Durchlauf werden die Offsets der Knoten berechnet und im zweiten Durchlauf werden die Offsets aufsummiert, was dann die entgültigen Positionen ergibt. Der Offset eines Knotens beschreibt um wie viel der linke Unterbaum nach links und der rechte Unterbaum nach rechts verschoben werden muss. Offsets wirken sich also nicht auf den Knoten selber aus, sondern nur auf dessen Unterbäume. Das gilt auch für Blätter des Baumes. Ihr Offset ist null, es sei denn sie erhalten durch Threads einen Nachfolger. Dann ist auch bei einem Blatt der Offset ungleich null.

Anhand der Abbildungen 5.13 und 5.14 wird nun die Vorgehensweise des Algorithmus verdeutlicht.

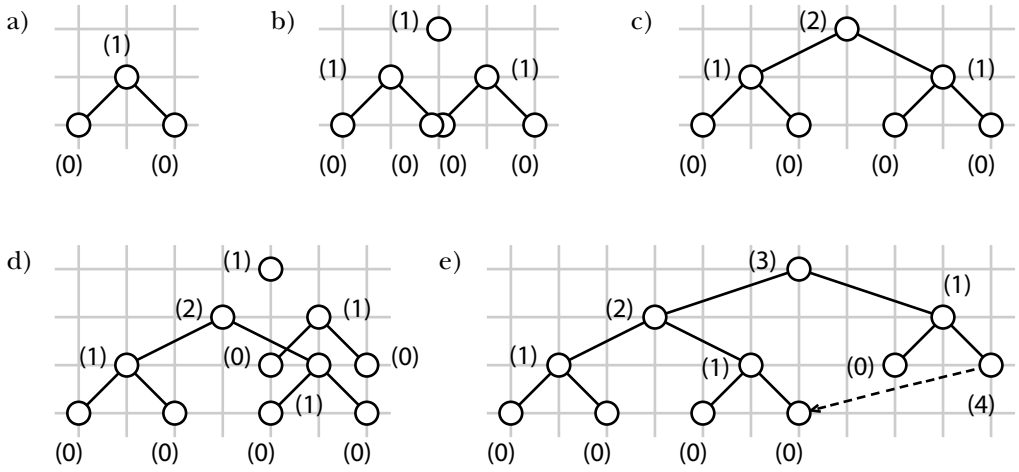

Abbildung 5.13.: Offsets des Reingold Tilford Algorithmus Teil 1

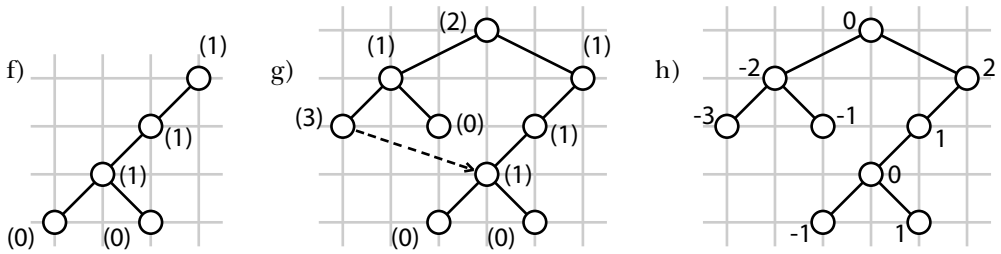

Abbildung 5.14.: Offsets des Reingold Tilford Algorithmus Teil 2

a) Die Positionierung der zwei Blätter und deren Vaterknoten ist abgeschlossen. Der Offset eins beschreibt, dass das linke Blatt eine Position weiter links als der Wurzelknoten und das rechte Blatt eine Position weiter rechts zu zeichnen ist.

b) Nun sollen zwei Unterbäume wie in a) unter einer neuen Wurzel zusammengebracht werden. Dazu wird der rechte Unterbaum zunächst mit Abstand zwei zum den linken Unterbaum positioniert. Genau diese Situation ist in b) dargestellt. Dann wird die Kontur der Unterbäume durchlaufen, d.h. es wird das nächste Level betrachtet und die Offsets der Konturen bestimmt. Im linken Unterbaum ist das rechte Blatt um eins weiter rechts, das aufsummierte Offset ist also eins ($0+1=1$). Im rechten Unterbaum dagegen ist das Blatt um eins weiter links, somit ist das Offset der Kontur eins ($2-1=1$). Beim Vergleich der beiden Offsets stellt sich heraus, dass sie keinen Abstand von mindestens zwei haben ($1-1=0$), somit muss der rechte Unterbaum um zwei nach rechts verschoben werden ($2-(1-1)=2$). Dieses geschieht allein durch eine Veränderung des Offsets der Wurzel und der Offsetsumme des rechten Unterbaums. Das Durchlaufen der Kontur ist damit abgeschlossen.

c) Die sich ergebende Situation ist in c) dargestellt. Der Offset des Wurzelknotens setzt sich zusammen aus der minimalen Distanz zweier Nachbarknoten (2) und der Verschiebung des rechten Unterbaums (2). Da sich der Offset auf beide Kinder auswirkt, ist der Offset nur Halb so groß wie die Summe ($\frac{2+2}{2}=2$). Dabei ist zu beachten, dass das Ergebnis der Division nach oben aufgerundet wird

$$ceil(\frac{2+2}{2}) = 2.$$

d) Im nächsten Schritt wird der eben erstellt Unterbaum um einen Unterbaum wie unter a) erweitert. Zu sehen ist hier wieder die Situation, dass die Wurzel des rechten Unterbaum gerade mit Abstand 2 zur Wurzel des linken Unterbaum platziert wurde. Das Durchlaufen des nächsten Konturlevels ergibt eine Verschiebung von drei

$$(2-1)-(0+2)=-1 \quad -1<2 \quad \Rightarrow \quad 2-(-1)=3.$$

e) Der Abstand zwischen den Wurzeln der Unterbäume beträgt damit $2+3=5$. Der Offset der Wurzel entspricht hier $ceil(\frac{2+3}{2}) = ceil(2.5) = 3$. Der rechte Unterbaum verschiebt sich um einen weiteren Punkt nach rechts, damit die Wurzel zentriert über Ihren Kindern steht. Da der rechte Unterbaum kürzer ist als der linke, wird die rechte Kontur mit Hilfe eines Threads fortgesetzt. Der Offset des Blattes ist die tatsächliche Verschiebung der Kontur (4).

f) Da dieser Baum nur linke Kinder besitzt sind seine Knoten in einer Linie angeordnet, wobei ein Kindsknoten immer den Abstand eins von seinem Vaterknoten hat. Dieser Baum wird der rechte Unterbaum im nächsten Beispiel.

g) Der hier gezeigte Zustand ist das Ergebnis der Zusammenführung der Unterbäume aus den vorherigen zwei Beispielen. Zunächst werden die beiden Unterbäume wieder mit Abstand zwei zwischen ihren Wurzeln platziert. Im ersten Schritt des

Konturvergleichs ergibt sich eine Verschiebung von zwei

$$(2-1)-(0+1)=0 \quad 0<2 \quad \Rightarrow \quad 2-0=2.$$

Da der linke Unterbaum zu Ende ist, muss mit Hilfe von einem Thread die linke Kontur fortgesetzt werden. Der Offset des Blattes ändert sich damit zu 3. Und der Rest des rechten Unterbaums braucht nicht mehr betrachtet zu werden.

h) Die tatsächlichen Positionen der Knoten werden rekursiv durch Summierung der Offsets bestimmt. Ist die Wurzel auf Position 0 und hatte einen Offset von 2, so erhält das linke Kind die Position $0-2=-2$ und das rechte Kind die Position $0+2=2$.

Die Beispiele verdeutlichen die hervorragende Eignung des Algorithmus für Binärbäume. Eine Erweiterung des Algorithmus auf Bäume mit unbeschränktem Grad ist leicht möglich, da nur jeweils die angrenzenden Konturen der Unterbäume miteinander verglichen werden. Die schon platzierten $i-1$ Unterbäume werden dann einfach als linker Unterbaum und der i-te Unterbaum als rechter Unterbaum aufgefasst. Der Vaterknoten wird dann wieder über allen Unterbäumen zentriert. Es gibt jedoch einen entscheidenden Nachteil dieser Vorgehensweise: Die Eigenschaft (A5) (Spiegelsymmetrie) ist nicht mehr gewährleistet. Kleinere Unterbäume zwischen zwei größeren Unterbäumen werden direkt am linken Unterbaum platziert (Abbildung 5.15 a). Zur Behebung dieses Problems wird der Baum, ähnlich wie oben, ein zweites Mal durchlaufen, nur werden in diesem Durchlauf die Unterbäume eines Knotens von rechts nach links durchlaufen und platziert (Abbildung 5.15 b). Berechnet man nun Durchschnittswerte der horizontalen Koordinaten, welche bei den beiden Durchläufen errechnet wurden, so wird (A5) erfüllt. Damit sind nun zwar die bisher geforderten ästhetischen Eigenschaften gegeben, jedoch werden bei dieser Vorgehensweise die kleineren Unterbäume zusammen gestaucht (Abb. 5.15 c). Der Algorithmus von Walker behebt nun auch dieses Problem (Abb. 5.15 d).

5.1.5. Walker's Algorithmus

Der Algorithmus von Walker [Walker90] ist so ausgelegt, dass Bäume von unbeschränktem Grad unter Berücksichtigung von (1)-(9) gezeichnet werden können. Kleinere Unterbäume verteilen sich dabei zwischen größeren Unterbäumen (vgl. Abbildung 5.15 d).

Wie sich allerdings herausstellte arbeitet der ursprüngliche Algorithmus nicht in linearer Zeit, obwohl Walker dies in seiner Arbeit behauptete. Diese Tatsache animierte Buchheim, Jünger und Leipert dazu einen überarbeiteten Algorithmus vorzustellen, der die gleichen Layouts wie Walker Algorithmus liefert, jedoch in linearer Zeit arbeitet [BuchLeip02][2]. Der im weiteren vorgestellte Algorithmus ist der von Buchheim, Jünger und Leipert, wird jedoch weiterhin als Walker Algorithmus bezeichnet.

Wie der Algorithmus von Reingold und Tilford geht auch Walker's Algorithmus rekursiv vor. Die Unterbäume der aktuellen Wurzel werden nacheinander von links nach rechts

[2]Der Technical-Report zaik2002-431 beinhaltet im Anhang A den Pseudocode des Algorithmus `http://www.zaik.uni-koeln.de/~paper/index.html?show=zaik2002-431`

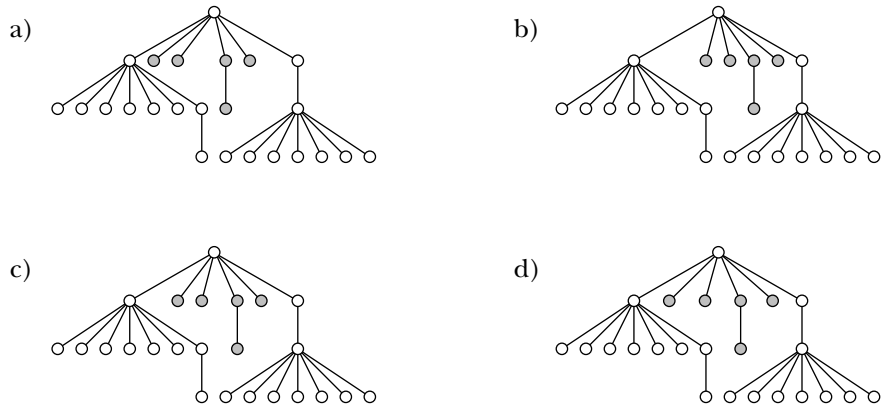

Abbildung 5.15.: Erweiterung des Reingold-Tilford Algorithmus auf Bäume mit unbeschränktem Grad.

abgearbeitet. Wird ein Unterbaum an den linken Teilwald angefügt, so wird die Wurzel dieses Unterbaumes zunächst so nahe wie möglich rechts neben seinem linken Geschwisterknoten platziert. Dann wird wie beim Algorithmus von Reingold und Tilford die linke Kontur des anzufügenden Baume durchlaufen und die Positionen der einzelnen Knoten mit den Positionen der linken Nachbarknoten verglichen. Werden nun zwei in Konflikt stehende Knoten v_- und v_+ gefunden, welche dazu führen, dass der anzufügende Unterbaum um s verschoben werden muss, so werden alle kleineren Unterbäume deren Wurzel zwischen den größten getrennten Vorgängern von v_- und v_+ liegen, ebenfalls verschoben. Es sind ein paar kleine Tricks nötig, damit auch hier die Linearität gewahrt bleibt. Die Tricks bestehen aus zusätzlichen Attributen pro Knoten. Jeder Knoten speichert Werte für seine vorläufige Position (*prelim*), eine Modifizierung der Position (*mod*), einem Zeiger auf den nächsten Knoten der Kontur (*thread*) und drei Werte für die Platzierung von kleineren Unterbäumen zwischen größeren Unterbäumen (*change*, $shift$ und *ancestor*).

Auf die Verwendung von *ancestor* wird hier nur kurz eingegangen, es sei schon einmal erwähnt, dass dieser Wert für die Bestimmung des Knotens benutzt wird, ab dem kleinere Unterbäume verschoben werden[3]. Das hieran anschließende Beispiel veranschaulicht die Verwendung der Werte *change* und $shift$.

Die Unterbäume werden von links nach rechts durchlaufen, wobei kleinere Unterbäume möglichst gleichmäßig auf den zur Verfügung stehenden Platz aufgeteilt werden sollen. Doch wie werden die kleineren Unterbäume bestimmt, die verschoben werden sollen? Und wie deren Verschiebung? Die Abbildung 5.16 demonstriert an einem Beispiel die von Buchheim, Jünger und Leipert gefundene Lösung dieser Probleme. Es zeigt vier Zwischenschritte des Algorithmus beim Berechnen der Positionen von Knoten des Baumes, der schon in Abbildung 5.15 gezeigt wurde. Die obere Zahl steht für den Wert von $shift(v_i)$ des sich darunter befindlichen Knotens v_i und die untere Zahl ist der Wert von $change(v_i)$ des entsprechenden Knotens. Die kleineren Un-

[3]Die genaue Verwendung kann in [BuchLeip02] nachgelesen werden

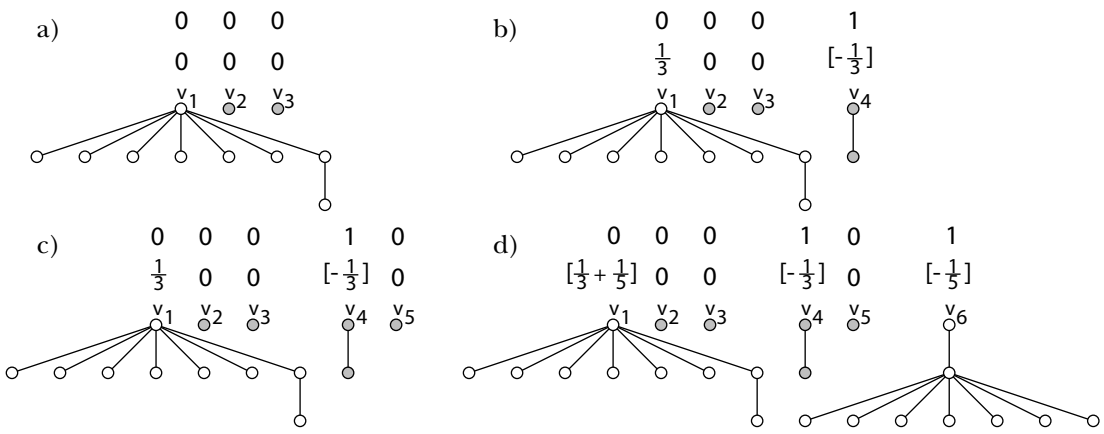

Abbildung 5.16.: Platzierung von kleineren Unterbäumen: $shift(v_i)$ und $change(v_i)$

terbäume wurden zur besseren Unterscheidung grau eingefärbt. In Abb. 5.16 a) ist die Situation dargestellt in der nur weitere Blätter an die (nicht dargestellte) Wurzel des Baumes angefügt wurden. Für jeden Knoten $v_i, i = 1, 2, 3$ sind die Werte $shift(v_i)$ und $change(v_i)$ identisch Null. Im nächsten aufgezeigten Schritt (b) wurde ein Teilbaum angefügt, der auf Grund seiner Größe nicht direkt an den schon vorhandenen Teilwald angefügt werden kann. Um Überschneidungsfreiheit zu gewährleisten muss der Unterbaum mit Wurzel v_4 um einen Wert $s = 1$ nach rechts verschoben werden. Dadurch werden die beiden grau eingefärbten Blätter zu kleineren Unterbäumen zwischen zwei Größeren. Es sei *subtrees* die Zahl der Unterbäume zwischen v_1 und v_4 plus eins ($subtrees = 3$). Nach Walker's Idee muss der i-te dieser Unterbäume um den Wert $i \cdot s/subtrees$ nach rechts verschoben werden. Gespeichert wird dies, indem $shift(v_4) = shift(v_4) + s = 1$, $change(v_4) = change(v_4) - s/subtrees = -\frac{1}{3}$ und $change(v_1) = change(v_1) + s/subtrees = \frac{1}{3}$ gesetzt wird. Die Werte $change(v_i)$ werden, nachdem alle Unterbäume abgearbeitet sind, von rechts nach links aufsummiert (Variable *change*) und die Summe wird für die Verschiebung des jeweiligen Unterbaums verwendet. Deshalb wird an dieser Stelle $change(v_4)$ um $\frac{1}{3}$ verringert und $change(v_1)$ um $\frac{1}{3}$ erhöht. In Abb. 5.16 c) ist ein weiteres Blatt v_5 an den Teilwald angefügt, wie schon im ersten Schritt werden die Werte $shift(v_5)$ und $change(v_5)$ zu Null gesetzt. Mit dem Anfügen des letzten Teilbaums in d) ist wieder die Situation wie in b) entstanden. v_5 muss einen zusätzlichen Abstand $s = 1$ einhalten, um sich nicht mit dem linken Unterbaum, dessen Wurzel v_1 ist, zu überschneiden ($shift(v_6) = 1$). Desweiteren sind dadurch wieder kleinere Unterbäume zwischen zwei Größeren entstanden, die entsprechend verschoben werden müssen. *subtrees* ist hier gleich fünf. Daher wird $change(v_6)$ um $\frac{1}{5}$ verringert und $change(v_1)$ um $\frac{1}{5}$ erhöht.

Damit können nun sämtliche Verschiebungen in einem abschließenden Durchlauf der Kinder der aktuellen Wurzel von rechts nach links ausgeführt werden. Dazu werden zwei Hilfsvariablen $shift$ und *change* zunächst mit 0 initialisiert. Wird hierbei etwa das Kind v besucht, so wird v um $shift$ nach rechts verschoben. Daraufhin wird $change =$

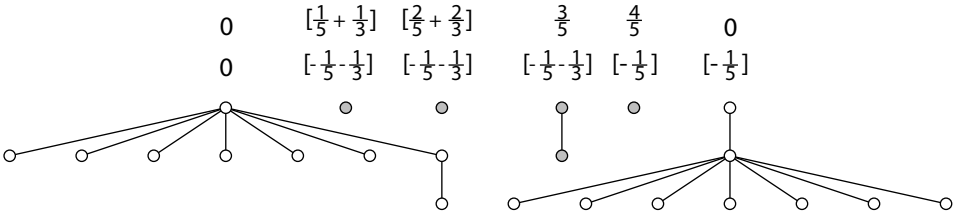

Abbildung 5.17.: Platzierung von kleineren Unterbäumen: $shift$ und $change$

$change + change(v)$ und $shift = shift + shift(v) + change$ gesetzt und mit dem linken Geschwisterknoten gleich verfahren. Die sich ergebenden Werte für $shift$ und $change$ sind in der Abbildung 5.17 dargestellt.

Der Walker Algorithmus berechnet Positionen von Knoten aus Bäumen mit beliebig vielen Nachfolgern pro Knoten in linearer Zeit und hält dabei die Eigenschaften (1) bis (9) ein. In der hier vorgestellten Form kann er nur mit Knoten gleicher Größe umgehen, bzw. kleinere Knoten nehmen den Platz wie der größte Knoten ein. Im Rahmen dieser Diplomarbeit ist eine Verfahren entwickelt worden, um auch Knoten mit unterschiedlichen Knotengrößen behandeln zu können. Das nun folgende Kapitel behandelt diese Vorgehensweise.

5.2. Verallgemeinerung der Darstellung

Die bisher vorgestellten Algorithmen setzen konstante Knotengrößen voraus. Das allgemeinere Problem, Positionen von Knoten aus Bäumen mit verschiedenen Knotengrößen zu berechnen, ist hier noch nicht betrachtet worden. Dabei haben nahezu alle realen Bäume verschiedene Knotengrößen. Eine weit verbreitete Ad-hoc Lösung ist, die maximale Höhe und Breite der Knoten zu bestimmen und dann allen Knoten so viel Platz einzuräumen, als währen diese so groß wie die maximale Ausdehnung (Abbildung 5.18). Eine andere Ad-hoc Lösung ist den Knoten immer verschiedene vertikale Positionen zuzuweisen (z.B. Abbildung 5.9 oder Abbildung 5.10). Beide Vorgehensweisen verbrauchen im Allgemeinen unnötig viel Platz und liefern nur in Ausnahmefällen ein gutes Layout. Die Verallgemeinerung, die im Rahmen dieser Diplomarbeit entstand, behebt diese Nachteile.

Eine weitere Anforderung an Baumzeichnungs Algorithmen ist die Unterstützung von interaktiver Bedienung. Der Benutzer will oft nicht nur den Baum angezeigt bekommen, sondern er will durch den Baum navigieren, Unterbäume ein- und ausblenden, nur Ausschnitte des Baumes betrachten oder zwischen Darstellungen wählen wie 'von oben nach unten' oder 'von links nach rechts'. Ebenso sind Kanten nicht immer gerade Linien und nicht immer ist die Position der ein- und ausgehenden Kanten gleich. Der Algorithmus sollte dies berücksichtigen ohne für jeden Fall besondere Ausnahmen zu haben. Statt den besten bekannten Algorithmus (zur Zeit Walker) so abzuändern, dass er alle Layoutaufgaben bewältigen kann, wird eine Zwischenschicht eingeführt, die die Layou-

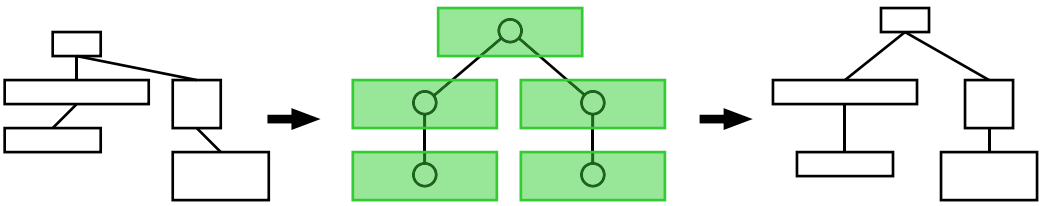

Abbildung 5.18.: weit verbreitete Ad-hoc Lösung

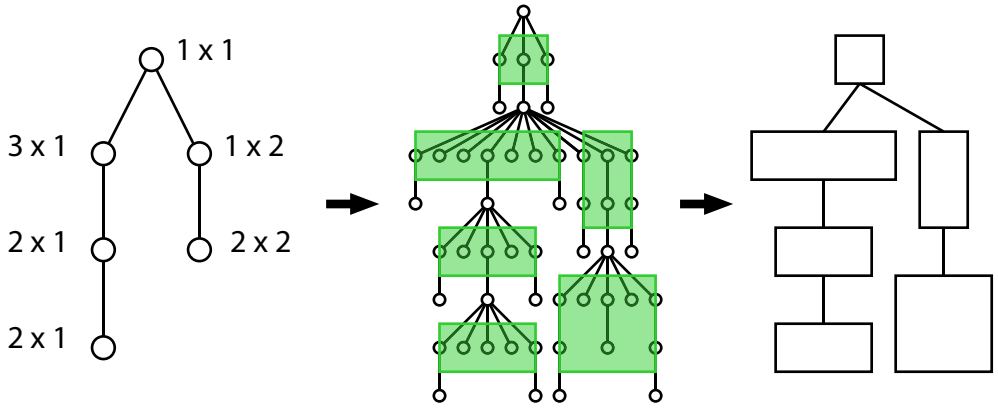

Abbildung 5.19.: BoxLayout

tanforderungen von dem Algorithmus trennen. Die Layoutanforderungen werden in eine topologisch äquivalente Struktur transformiert, für die der Walker Algorithmus eine optimale Lösung liefert. Nach Ablauf des Algorithmus wird durch Rücktransformation das eigentliche Layout erzeugt.

Die Transformation in die topologisch äquivalente Struktur ist hier eine Graphtransformation, die den gegebenen Baum in einen Baum transformiert, in dem die Layoutbedingungen enthalten sind. Die Rücktransformation besteht im wesentlichen darin, die Position der Knoten in das Koordinatensystem des ursprünglichen Baumes zu übersetzen.

```
Pseudocode:
calcPositions(Tree t) \{
 Graph g = createLayoutgraph(t); //erstelle Layoutgraph aus Anzeige-Baum
 calcPositions(g); //berechne Positionen mit Walker-Algorithmus
 transferPos(g, t); //übertrage Positionen in Anzeige-Baum
\}
```

Ein Label im Anzeige-Baum entspricht nicht mehr genau einem Knoten im Layoutgraph, sondern wird auf eine Menge von Knoten im Layoutgraph abgebildet. Diese Abbildung ist topologieerhaltend und bildet jeden Knoten des Anzeige-Baums auf eine

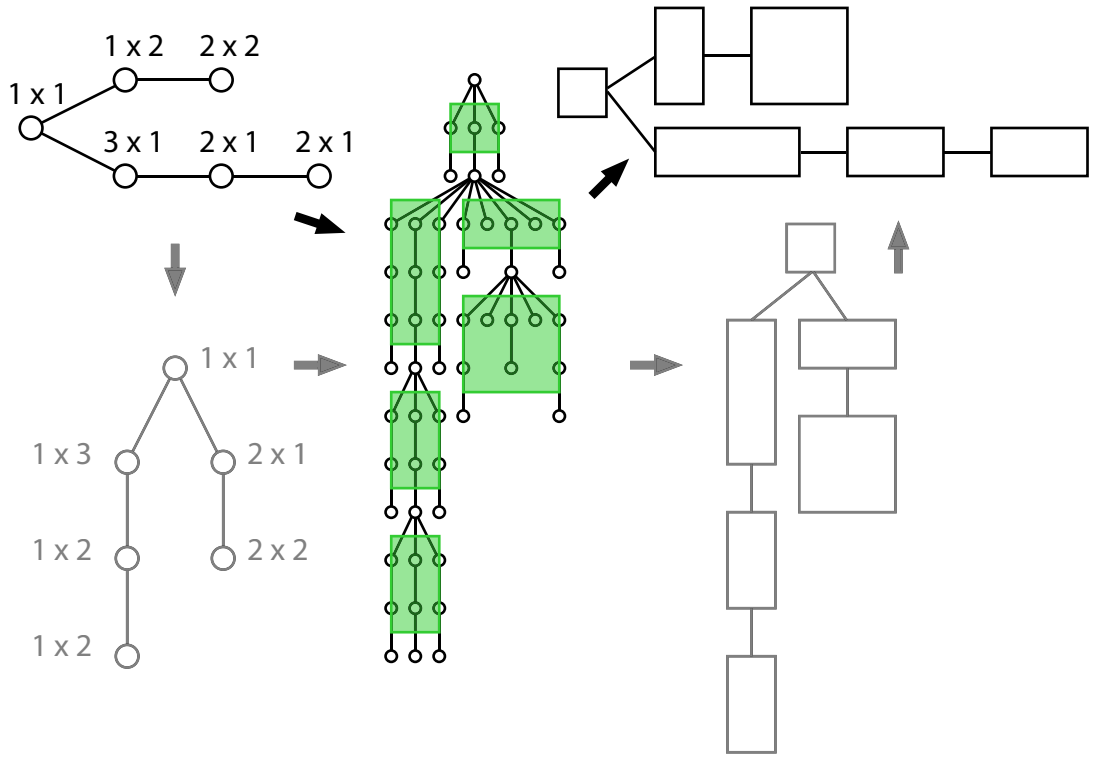

Abbildung 5.20.: BoxLayout Links nach Rechts

symmetrische Anordnung von Knoten ab, die wiederum die Form eines Baumes haben (vgl. Abbildung 5.19).

Sei v ein Knoten im Anzeigebaum der Größe[4] bxh. v wird zunächst in einen Teilbaum $T(v)$ transformiert, der l_b Blätter hat, von denen $l-3$ auf Level 1 sind und drei auf Level l_h, wobei $l_b = 2 * b + 1$ und $l_h = h + 1$ entspricht. Diese Teilbäume werden wiederum zum Layoutgraph zusammengesetzt, indem die Wurzeln der Nachfolger eines Knotens v mit dem untersten Knoten des Vaters vereint werden.

Alle Knoten im Layoutgraph hat die gleiche Größe, daher kann mit dem besten bekannten Algorithmus für diesen Graphen ein optimales Layout berechnet werden. Danach können die Positionen an den eigentlichen Anzeigebaum übertragen werden.

Dieser Zwischenschritt ermöglicht es, das Layout eines Baumes mit unterschiedlichen Knotengrößen zu berechnen mit Hilfe des besten Layoutalgorithmus für konstante Knotengrößen. Des weiteren ist es ohne großen Aufwand möglich weitere Varianten zu integrieren, ohne den darunterliegenden Algorithmus anpassen zu müssen:

1. verschiedene Orientierungen des Layouts:„Oben nach Unten“, „Unten nach Oben“, „Links nach Rechts“ oder „Rechts nach Links, wie in Abbildung 5.20 gezeigt.

2. variable Abstände zwischen den Knoten (vertikaler als auch horizontaler Abstand)

3. verschiedene Abstände für verschiedene Knoten

4. verschiedene Kantenführungen (Knicke von Kanten sind Knoten im Layoutgraph)

Dabei bleiben die Eigenschaften des darunterliegenden Algorithmus erhalten. Wird zum Beispiel der Walker-Algorithmus verwendet, der ja wie oben ausgeführt die Bedingungen[5] (1)-(9) erfüllt, dann erfüllt die hier entwickelte Vorgehensweise die Bedingungen (2), (4)-(7), (9)-(13).

Die Benutzung des Zwischenschrittes hat viele Vorteile in der Programmierung (Einfachheit, Trennung von unabhängigen Teilschritten, Wartbarkeit, Erweiterbarkeit, ...), hat aber auch den Nachteil eines etwas gestiegenen Speicherverbrauchs (um konstanten Faktor größer).

In diesem Unterkapitel wurde die Verallgemeinerung allgemein vermittelt und im anschließenden Unterkapitel findet sich eine Übersicht über die Realisierung in Java und deren Ergebnisse.

[4] beliebige Größeneinheit, wie z.B. Millimeter oder Pixel

[5] Die Layoutbedingungen wurden in Kapitel 5.1.3 aufgelistet.

5.3. Realisierung

Die Realisierung des Datenvisualisierungsmoduls erfolgte in Java mit Hilfe der freien Entwicklungsumgebung Eclipse[6] und dem Plug-In EclipseUML[7]. Omondo's EclipseUML ist ein visuelles UML Modellierungs Werkzeug, das komplett auf dem Plug-In Konzept von Eclipse aufbaut und somit eine nahtlose Integration in Eclipse darstellt. Die Visualisierungen sind aus Java Swing Komponenten zusammengesetzt, wodurch eine performante und plattformunabhängige Realisierung möglich wurde.

Vererbungshierarchie der Visualisierungen

Die Oberklasse aller Visualisierungen heißt *Visualization*, welche wiederum von *JLayeredPane* aus dem Package javax.swing abgeleitet ist. Das UML-Diagramm 5.21 veranschaulicht diesen Zusammenhang. Dadurch ist es möglich, jede Visualisierung in jedem beliebigen *javax.swing.JComponent* einzusetzen und zu verwenden. Implementiert wurden zwei verschiedene Darstellungen:

1. Die Klasse *DistortionVisualization* liefert ein radiales Layout mit Verzerrung darstellt (siehe Kapitel 5.1.1).

2. Die Klasse *NativeTreeVisualization* erstellt Layout gemäß des Kapitels 5.1.5.

Im Laufe der Implementierungen ist die zusätzliche *MessageVisualization* entstanden, die nur einen einfachen Text zentriert anzeigt und hier nicht näher erläutert wird.

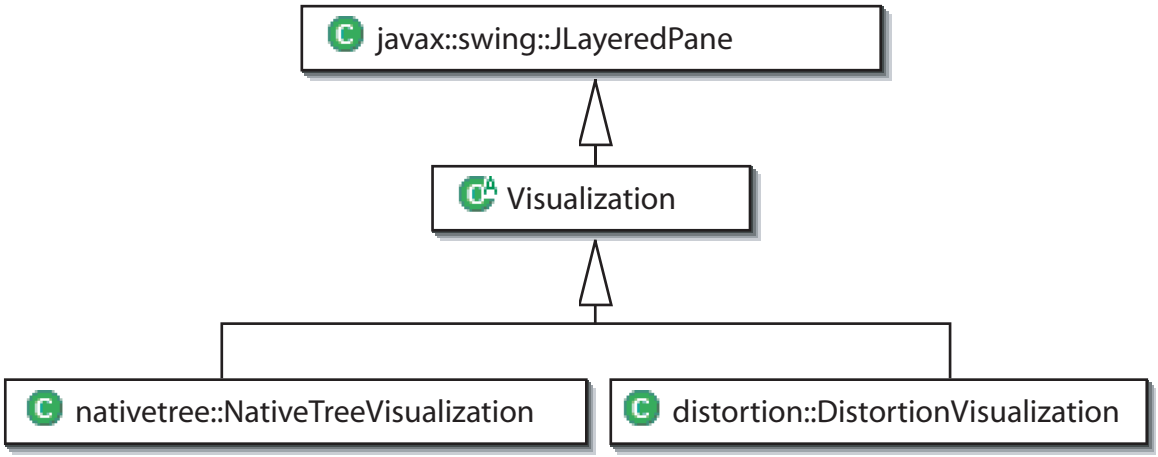

Abbildung 5.21.: Vererbungshierarchie der Visualisierungen

[6] http://www.eclipse.org
[7] http:www.omondo.de

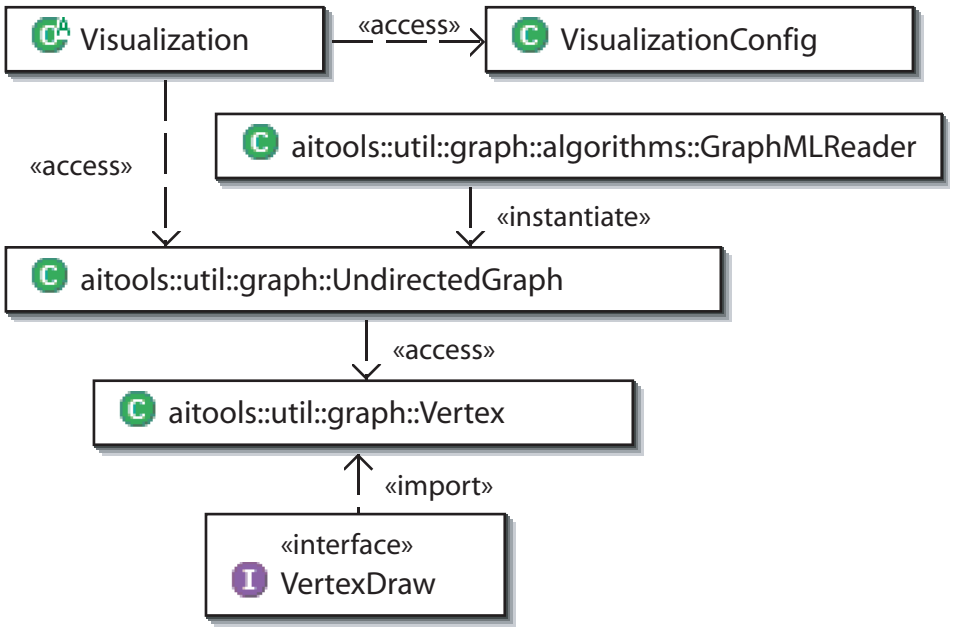

Abbildung 5.22.: Allgemeine Klassen der Visualisierungen

Einbindung des Graphen

Die Abbildung 5.22 zeigt die wesentlichen Komponenten einer Visualisierung. Die gemeinsamen Einstellungen (Schriftart, Titel, ...) werden in der Klasse *Visualization-Config* vorgehalten. Sie dient als Oberklasse für weitere Einstellungen der einzelnen Visualisierungen, wie z.B. der verwendete Layout- oder Verzerrungsalgorithmus. Die Klasse *UndirectedGraph* aus dem Package aitools.util.graph ist die vom *GraphMLReader* erzeugte Datenstruktur, die aus Knoten (*Vertex*) und Kanten (*Edge*) besteht. Zur Darstellung auf der Zeichenfläche werden *VertexDraw*s benutzt, die für jede Visualisierung verschieden sind. Sie speichern Informationen wie die aktuelle Position des Knoten und seine Sichtbarkeit (Nachfolger eines Knotens können ausgeblendet werden).

5.3.1. Ergebnisse

Dieses Unterkapitel präsentiert die Ergebnisse der Realisierung, bezüglich der praktischen Verwendbarkeit der Vorgehensweise. Dabei wird sowohl die Qualität des Layouts als auch die Laufzeit betrachtet. Die Eckdaten der zur Analyse herangezogenen Graphen ist in der folgenden Tabelle zu entnehmen.

Anzahl der Knoten im Anzeigebaum	1000 - 30,000
Grad eines Knotens	0-10, gleichverteilt
Breite der Knoten	1-100 Einheiten, gleichverteilt
Höhe der Knoten	1-100 Einheiten, gleichverteilt

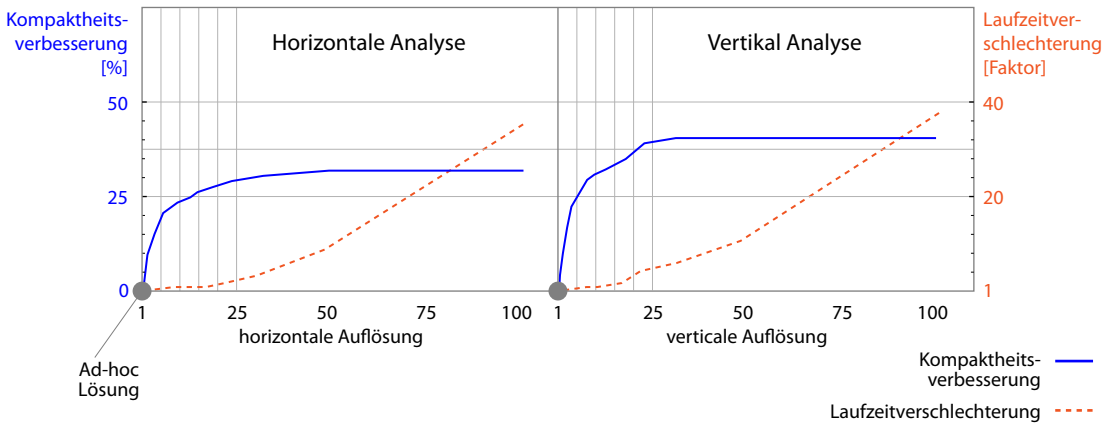

Abbildung 5.23.: Ergebnisse der Realisierung
Verhältnis zwischen der Kompaktheit des Layouts und der Laufzeit des Algorithmus. Auf Grund der Natur des Layoutalgorithmus ist in horizontaler Richtung (durchgezogene Kurve auf der linken Seite) weniger Optimierung möglich, als in vertikaler Richtung (durchgezogene Kurve auf der rechten Seite).

Abbildung 5.23 fast die verschiedenen Ergebnisse der Experimente zusammen. Die Kurven mitteln über verschiedene Graphen und zeigen die Kompaktheitsverbesserung (durchgezogene Linie) sowie die Laufzeitverschlechterung (gestrichelte Linie). Die vertikale Achse beschreibt dabei, durch wie viele Größeneinheiten der größte Knoten aufgelöst wurde. Man beachte dabei die verschiedenen Skalen: Die linke Skala zeigt die Verbesserung der Kompaktheit des Layouts in Prozent der eingesparten Fläche gegenüber der Ad-hoc Lösung; die rechte Skala zeigt die Verschlechterung der Laufzeit als Faktor gegenüber der Laufzeit der Ad-hoc Lösung. Die Ergebnisse machen deutlich, dass für eine wesentliche Verbesserung der Kompaktheit, der zusätzliche Laufzeitaufwand sehr gering ist.

Während die Abbildung 5.23 hauptsächlich relative Vergleiche bietet, ist die Angabe einer tatsächlichen Laufzeit ebenso interessant. Auf einem Standard-PC (2GHz) benötigt der Algorithmus für große Bäume ca. 100msec pro 10.000 Knoten. Der Layoutalgorithmus ist also, wie erwartet, nicht ein Flaschenhals in der Verarbeitung von großen Bäumen, sondern eher die Anzeige dieser riesigen Menge an Knoten.

6. Zusammenfassung und Diskussion

Das Analysieren unstrukturierter Informationen ist so etwas wie der Heilige Gral der Suchtechnologien[1]. Etwa 85 Prozent aller Unternehmensdaten schwirren unstrukturiert außerhalb von Datenbanken herum, sodass Angestellte etwa ein Drittel ihrer Zeit damit verschwenden, relevante Informationen in der immer größer werdenden Datenflut zu suchen.

Das Ziel dieser Diplomarbeit war die Spezifikation eines Prozesses zur Lösung von Information Retrieval Problemen. Dieses Ziel ist erreicht worden; der entwickelte Prozess ist unabhängig von Rechnerplattformen, sowie stabil in Bezug auf künftige Erweiterungen (auf Grund der Verwendung von Web Services). Da eine Realisierung des gesamten IR-Frameworks aus Zeitgründen nicht Bestandteil dieser Arbeit sein kann, wurde nur ein Teil des Frameworks tatsächlich realisiert. Für dieses Modul wurde ein eigenes Verfahren zur Darstellung von hierarchischen Strukturen entwickelt, welches als eine Verallgemeinerung der besten, bekannten Algorithmen zu sehen ist. Die Realisierung erfolgte in Java um das Modul auch für andere Projekte zur Verfügung stellen zu können.

Was würde ich beim nächsten mal anders machen?
Das Heraussuchen und Vorstellen der vorhandenen Spezifikationen, Standards und Implementierungen im Bereich des Information Retrievals hat am Anfang sehr viel Zeit benötigt. Das hatte zur Folge, dass das entwickelte Framework nur grob umrissen werden konnte. Die Realisierung des Datenvisualisierungsmoduls dagegen hat auf Grund der Vorkenntnisse des Autors nur wenig Zeit in Anspruch genommen.

Bei allen, die mir durch ihre Tipps und Hilfe bei der Durchführung und Erstellung der Diplomarbeit geholfen haben, bedanke ich mich herzlichst. Mein besonderer Dank geht an Prof. Benno Stein, für seine konstruktive und freundliche Unterstützung in unseren zahlreichen Gesprächen.

[1] `http://www.silicon.de/cpo/ts-busisoft/detail.php?nr=22887`

A. Anhang

A.1. Informationen zu verwendeten Standards

A.1.1. MOF

Gemeinsam mit der Sprache UML wurde der Standard *Meta Object Facility (MOF)* von der OMG publiziert und in den Folgejahren weiterentwickelt. Die Spezifikation der MOF beschreibt eine abstrakte Sprache und ein Grundgerüst, das MOF Modell, zur Erzeugung und Spezifikation von plattformunabhängigen Meta-Modellen. Zusätzlich definiert die Spezifikation ein Grundgerüst, mit dem ein Repository (engl. f. Lager) implementiert werden kann, das Meta- Daten(d.h. Modelle) speichert, die durch Meta- Modelle beschrieben werden. Insgesamt besteht die Spezifikation aus:

- der formalen Definition des MOF Meta-Meta-Modells, also der Sprache, mit der MOF- Meta-Modelle definiert werden,
- einem Mapping, das beliebige MOF Meta-Modelle in die Corba IDL abbildet, mit dessen Hilfe IDL Interfaces erzeugt werden, um Meta-Daten zu verwalten,
- -reflective CORBA IDL Interfaces, mit deren Hilfe es möglich ist, Meta- Daten unabhängig vom Meta- Modell zu verwalten,
- CORBA IDL Interfaces, mit denen MOF Meta-Modelle repräsentiert und verwaltet werden können,
- dem Extensible Markup Language (XML) Meta-Data-Interchange (XMI) Format, mit dem Meta- Modelle ausgetauscht werden können.

Im weiteren Verlauf soll die Ebenen-Architektur und die Kernkonzepte von MOF betrachtet werden.

Der zentrale Begriff des MOF Ansatzes zum Metadatenmanagement ist Erweiterbarkeit. Das Ziel ist ein Framework zu liefern, dass alle Arten von Metadaten unterstützt und dass es erlaubt neue Arten hinzuzufügen. Um dieses zu erreichen hat die MOF eine ebenenbasierte Metadatenarchitektur die auf der klassischen vier Ebenen Metamodellierungsarchitektur basiert die wiederum eine weite Verbreitung in den Standardisierungsgemeinschaften[1] gefunden hat. Das Schlüsselkonzept von beiden, der klassischen und der MOF Metadatenarchitektur, ist die Meta-Metamodellierungs Ebene, die die Metamodelle und die Modelle vereint.

Die MOF Metadatenarchitektur, illustriert im Beispiel in Abbildung A.1, basiert auf der traditionellen vier Ebenen Metadatenarchitektur wie oben beschrieben. Dieses Beispiel zeigt eine typische Instanzierung der MOF Metadatenarchitektur mit Metamodellen die UML Diagramme und OMG IDL repräsentieren.

Die MOF Metadatenarchitektur hat einige wichtige Features, die sie von früheren Metadatenarchitekturen unterscheidet:

[1] wie zum Beispiel ISO und CDIF

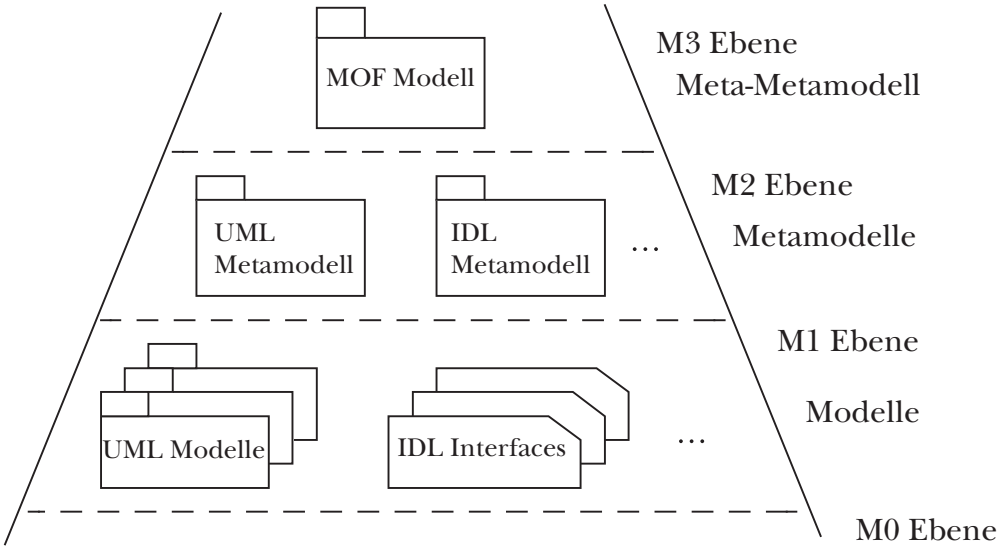

Abbildung A.1.: MOF Metadaten Architektur

- Das MOF Model (der Kern des MOF Meta-Metamodell) ist objektorientiert und benutzt metamodeling Konstrukte die sich an die Objektmodellierungskonstrukte von UML anlehnen. Daher werden in den Bespielen UML Package-Icons für MOF-basierte Metamodelle und für UML Modelle benutzt.

- Die Metalevels in der MOF Metadatenarchitektur sind nicht vorgeschrieben. Obwohl es typischerweise 4 Metalevels gibt, können es auch mehr oder weniger sein. MOF Metalevels sind eine reine Konvention zum besseren Verständnis der Beziehungen zwischen den Daten und Metadaten.

- Ein Modell (im Sinne einer Zusammenstellung von Metadaten) ist nicht notwendigerweise auf ein Metalevel beschränkt.

- Das MOF Modell beschreibt sich selbst. Dies funktioniert ähnlich wie im Duden: die deutsche Sprache wird mit Hilfe der deutschen Sprache erläutert. Mit anderen Worten, das MOF Metamodell ist formal definiert durch seine eigenen Metamodellierungskonstrukte. Das MOF Modell wird ebenso mit einem UML Package-Icon gekennzeichnet.

Die Fähigkeit des MOF Modells sich selber zu beschreiben hat wichtige Konsequenzen:

- Es zeigt dass das MOF Modell genügende Ausdrucksstärke für praktische Metamodellierung besitzt.

- Es erlaubt die Definition von MOF-Schnittstellen und Verhalten durch eine Abbildung von MOF IDL zum MOF Modell. Dieses stellt eine uniforme Semantik

zwischen Objekten bereit, die Modelle und Metamodelle repräsentieren. Das bedeutet auch, falls eine neue Technologieabbildung definiert wurde, dann sind auch die API's implizit definiert, die die Metamodelle in dessen Kontext bereitstellen.

- Es bietet eine Basis für Erweiterungen und Modifikationen an dem MOF Modell. Auftretende Probleme werden durch aufeinander folgende MOF RTFs (Revision Task Force) aufgefangen, die inkrementelle Veränderungen an dem MOF Modell vornehmen.

A.1.2. MDA

[2] Bei der *Model Driven Architecture (MDA)* von der Object Management Group (OMG) bilden Modelle die zentralen Elemente des Softwareentwicklungsprozesses. Ziel ist es, plattformspezifische Modelle möglichst automatisiert aus plattformabhängigen Modellen abzuleiten. Dadurch soll der Aufwand der Softwareentwicklung verringert und die Adaption an neue Technologien erleichtert werden.and der Softwareentwicklung verringert und die Adaption an neue Technologien erleichtert werden.

Einführung Programmcode als textuelle Repräsentation von Software ist die Grundlage, um lauffähige Programme zu erzeugen. Soll die Funktionalität einer Softwarekomponente erweitert werden, so geschieht dies in der Regel durch die Änderung des Quellcodes. Dem gegenüber dienen Modelle als graphische Repräsentation, wie sie beispielsweise in der UML notiert werden, im Softwareentwicklungsprozess hauptsächlich zur Spezifikation und Dokumentation. Nur partiell erfolgt eine Generierung von Programmcode aus solchen Modellen, der dann noch um wesentliche Teile manuell zu ergänzen ist. In seltenen Fällen werden Änderungen auf der Quellcodeebene auch auf der Modellebene durchgeführt.

Mit der Model Driven Architecture liegt nun ein Standardisierungsvorschlag der Object Management Group vor, der die Repräsentation von Software von der Programmcodeebene auf die Modellebene heben möchte[3]. Um die Komplexität auf Modellebene zu reduzieren, werden Modelle und Plattformen verschiedener Abstraktionsebenen unterschieden, die aufeinander aufbauen. Die Grundidee dieses Ansatzes ist nicht neu, sondern folgt dem Grundsatz, dass die Spezifikation einer Komponente unabhängig von der technischen Umsetzung zu beschreiben ist. Neu ist, dass mit MDA aufeinander abgestimmte Spezifikationen und ein Konzept für die modellgetriebene Softwareentwicklung existieren. Dabei kann die Realisierung eines plattformabhängigen Modells durch Wahl einer Plattform, also der konkreten technischen Umsetzung, teilweise oder vollständig automatisiert erfolgen. Modelle werden durch die Auswahl von Plattformen in weniger abstrakte Modelle transformiert bis letzten endlich ausführbarer Programmcode entsteht. Änderungen an der Software erfolgen nun nicht mehr im Programmcode, sondern, so ist das Ziel von MDA, in einem der Modelle. MDA soll die Portierbarkeit und Wiederverwendung von Modellen und dadurch der Software verbessern. Erhofft wird so eine

[2] Dieser Abschnitt basiert auf dem im *Informatik Spektrum*(Band 28, Heft 4, August 2005) erschienen Artikel *Model Driven Architecture* von Martin Kempa und Zoltán Ádám Mann (Seite 298)

[3] Miller, J., Mukerji, J.: MDA Guide Version 1.0.1, OMG (2003)

Beschleunigung und damit eine Kostensenkung in der Entwicklung von Software.

Modelle Mit der MDA wird das Modell zum zentralen Element des Softwareentwicklungsprozesses. Neben der Spezifikation und Dokumentation von Software werden formale Modelle nun auch zur Definition der Architektur, des Designs und der Implementierung genutzt. Sie beschreiben nicht nur den statischen Teil der Softwarekomponenten auf Modellebene, sondern auch den dynamischen Teil, beispielsweise in Form von Zustandsmaschinen. Drei Arten von Modellen werden unterschieden: das *Computation Independent Model*, *Platform Independent Model* und *Platform Specific Models*.

Das *Computation Independent Model (CIM)* beschreibt ein Softwaresystem auf fachlicher Ebene. Es liegt in einer Sprache vor, die für die fachlichen Anwender des Systems verständlich ist, und dient zum Diskurs zwischen Softwarearchitekten und Anwendern über Leistungsumfang und Anforderungen. Das CIM legt fest, was ein System leistet. Es definiert nicht, wie es dies leistet oder wie das System strukturiert ist.

MDA kennt als weiteren wichtigen Begriff, das Konzept der Plattform. Mit einer *Plattform* werden abgeschlossene Softwarekomponenten oder -technologien bezeichnet, die über spezifizierte Schnittstellen verwendet werden können, ohne dass die benutzende Komponente die Implementierung der von der Plattform bereitgestellten Funktionalität kennen muss. Die Plattform stellt technische Dienste bereit, ohne die die Software nicht funktionieren kann. Ein *Plattform-Modell* ist die Repräsentation einer Plattform auf Modellebene.

Das *Platform Independent Model (PIM)* modelliert die Funktionalität einer Komponente unabhängig von der gegebenen Plattform. Damit enthält ein PIM also den Teil eines Systems, der sich beschreiben lässt, ohne die entgültige Zielplattform zu kennen.

Das *Platform Specific Model (PSM)* kennt eine spezielle Plattform und realisiert ein PIM, wofür die von der Plattform bereitgestellten Schnittstellen genutzt werden.

Beispiel: Wenn das System basierend auf der Java-2-Enterprise-Edition (J2EE) Plattform zu realisieren ist, ist das Plattform-Modell die Beschreibung von J2EE, das PIM die Beschreibung des Systems ohne J2EE-spezifische Details, während das PSM ein mit J2EE-spezifischen Details angereichertes Modell ist, aus dem schon der Programmcode generiert werden kann.

Da in MDA Plattformen auf unterschiedlichen Abstraktionsebenen vorkommen können, ist es möglich, dass ein PSM der einen Ebene ein PIM der anderen Ebene ist. In solch einem Fall beziehen sich die beiden Ebenen auf verschiedene Plattformen. Die Unterscheidung der verschiedenen Modellebenen hat zur Folge, dass Änderungen an der Software immer auf der Ebene vorgenommen werden müssen, auf der die relevante Information definiert wurde.

Versprechungen und Probleme Befürworter sehen in MDA den nächsten großen Fortschritt der Softwareindustrie mit folgenden Vorteilen:

- Modelle auf einer hohen Abstraktionsebene erlauben die kompakte Darstellung komplexer Systeme. So hilft ein PIM dabei, die ohnehin komplizierten fachlichen Abläufe eines großen Softwaresystems zu verstehen und zu organisieren, ohne technische Details mit betrachten zu müssen.

- Die Technik ändert sich rasant, was viele Softwaresysteme vorzeitig veralten lässt.

MDA dagegen erlaubt eine rapide Adaptierung an neue Technologien, indem aus dem PIM ein neues, der neuen Technologie entsprechendes PSM abgeleitet wird.

- Die Standardisierung der benutzten Modellierungssprachen und Metamodellierungsformate ermöglichen eine nahtlose Integration zwischen Werkzeugen unterschiedlicher Hersteller.
- Das CIM und gewissermaßen auch das PIM ermöglichen eine effiziente Kommunikation mit dem Kunden über das zu erstellende Softwaresystem.
- Die (zumindest größtenteils) automatische Generierung des PSMs und schließlich des Programmcodes beschleunigen den Entwicklungsprozess deutlich. Dabei werden gerade die stereotypen Programmieraufgaben durch Wiederverwendung eliminiert, was auch die Wahrscheinlichkeit von Programmierfehlern minimiert.

Neben diesen Vorteilen trifft MDA auch auf Skepsis, insbesondere bei folgenden Punkten:

- Wenn nicht das gesamte Programm generiert werden kann (was vermutlich nur in speziellen Fällen möglich ist), müssen nachträglich Änderungen von Hand gemacht werden, und es ist ein schwieriges Unterfangen, ein nicht selbst geschriebenes großes Programm zu ändern. In dieser Hinsicht taucht die Frage auf, welche Qualität das generierte Programm bezüglich Strukturierung, Lesbarkeit und Wartbarkeit aufweist.
- Es ist kompliziert, manuelle Erweiterungen und Änderungen in generierten Artefakten konsistent mit den Änderungen auf Modellebene zu halten.
- Die Performanz des generierten Codes sowie der Generierung selbst könnte ein Problem sein, da die Optimierungsmöglichkeiten am Code durch das Verfahren der Generierung eingeschränkt sind.
- Die OMG arbeitet zwar schon seit Jahren an den grundlegenden Spezifikationen, aber die Interoperabilität von verschiedenen Werkzeugen funktioniert nur bedingt, zumal verschiedene Hersteller grundsätzlich verschiedene Vorstellungen von MDA haben.
- Während die Modellierung der *Struktur* eines Systems weit verbreitet ist, ist die Modellierung des *dynamischen* Verhaltens eher problematisch. Für die erfolgreiche Generierung der Anwendungslogik wäre dies aber nötig. UML 2.0 enthält zwar viele Konstrukte dazu, diese sind aber auf der gleichen Abstraktionsebene wie eine normale Programmiersprache, so dass das Vorgehen eigentlich gar nicht mehr Modellieren, sondern Programmieren ist, nur graphisch statt textuell. Noch dazu ist diese Art von „graphischer Programmierung“ nicht einmal überschaubarer als die herkömmliche, textuelle Programmierung.

Wer in dieser Debatte Recht behalten wird, lässt sich heute noch nicht sagen. Jedenfalls ist klar, dass die Grenze der Generierbarkeit sowie die am besten geeignete Modellierungsebene noch erforscht werden müssen. Zu den Herausforderungen der nächsten

Jahre zählen die Überprüfung, ob die gewählten Modellebenen für die Repräsentation komplexer Softwaresysteme angemessen sind, die Definition genügend allgemeiner Transformationsregeln in einer formalen Sprache, die nicht nur effektive Software erzeugen, sondern auch wiederverwendbar sind, und die Etablierung geeigneter Tools, die für eine Verbreitung von MDA in der industriellen Softwareentwicklung sorgen.

A.1.3. Web Services

Web Services gelten als die Technologie, mit der in Zukunft Softwarekomponenten innerhalb einer Organisation und zwischen Organisationen integriert werden.

Unter „Web Services“ versteht man hierbei ein Bündel von Technologien zur Beschreibung von Schnittstellen und Eigenschaften von Implementierungen der Schnittstellen, Beschreibung von Datenaustauschformaten und Qualitätseigenschaften des Austauschs, Registrierung von Komponenten, Komposition von Komponenten und Sicherheit im Austausch mit Komponenten. Darüber hinaus versprechen „Web Services“ die Unterstützung eines neuen Programmier- und Systemparadigmas für die Entwicklung verteilter Systeme, in dem alle Teilnehmer autonom agieren.

Es gibt keine anerkannte Definition des Begriffs Web Service. Das W3C definiert in Web Services wie folgt:

A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Sehr wichtig sind uns die folgenden beiden, zusätzlichen Eigenschaften von Web Services: Identifizierbarkeit durch URIs und Autonomie.

Zusammenfassend sind also die Schlüsseleigenschaften eines Web Services:

- Ein Web Service wird durch einen URI identifiziert[4].
- Die Schnittstelle eines Web Services ist maschinenlesbar und wird durch WSDL (s. nächsten Abschnitt) beschrieben.
- Ein Web Service kommuniziert mit anderen Softwarekomponenten durch XML-Nachrichten. Der Nachrichtenaustausch kann insbesondere mit Hilfe von Internetprotokollen (z.B. HTTP oder SMTP) stattfinden.
- Web Services sind autonom. Das heißt man kann nicht beeinflussen, ob und wie eine Nachricht von einem Web Service verarbeitet wird. Qualitätseigenschaften wie z.B. Antwortzeitgarantien müssen durch zusätzliche Vereinbarungen geregelt werden.

[4]URI: Uniform Resource Identifier. URIs sind der Standard, um Objekte im Internet eindeutig zu identifizieren. URLs wie sie in HTML zur Referenzierung von Webseiten verwendet werden sind besondere URIs.

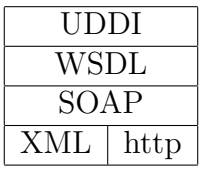

Abbildung A.2.: stack

Gemäß der obigen Definition von Web Services sind die Beschreibung von Schnittstellen und der Nachrichtenaustausch wesentliche Aufgaben. Hierzu wurden SOAP, WSDL und UDDI als drei Kerntechnologien entwickelt. Diese drei Technologien basieren alle auf XML und können Internetprotokolle wie http oder smtp verwenden (Abb. A.2). XML verstehen wir hierbei nicht nur als reines Datenformat sondern als Familie von Standards mit Datenmodell, Schema, Verschlüsselung und Signatur. Im Folgenden wird kurz SOAP, WSDL und UDDI vorgestellt.

SOAP

SOAP[5] legt die Grobstruktur und die Verarbeitungsvorschriften von Nachrichten fest. Es basiert auf XML und ist wie alle Standards des W3C plattform- und programmiersprachenunabhängig. Eine SOAP-Nachricht hat folgende Struktur:

```
<?xml version="1.0">
<env:Envelope xmlns:env = http://www.w3c.org/soap-envelope>
  <env:Header> ... </env:Header>*
  <env:Body>
    ...
    <env:Fault> ... </env:Fault>*
    ...
  </env:Body>
</env:Envelope>
```

Das heißt, jede Nachricht ist ein XML-Dokument mit `Envelope` als Wurzelelement. Jede Nachricht hat optional einen oder mehrere Header. Ein Header enthält Kontrollinformation wie z.B. eine Nachrichtennummer oder den Hinweis, dass die Nachricht eine Antwort auf eine andere Nachricht ist. Auf jeden Fall muss eine Nachricht einen Body haben. Der Body enthält die Nutzinformation, z.B. eine PurchaseOrder. Optional kann der Body auch Fehlermeldungen enthalten. Somit regelt SOAP auch, wie Fehler als Nachrichten versendet werden und gibt das Format (Fehlercode und Fehlerbeschreibung) für Fehlermeldungen vor. Neben dem Typ einer Nachricht definiert SOAP ein Verarbeitungsmodell. Gemäß dieses Modells gibt es neben dem Absender und Empfänger einer Nachricht noch sogenannte Intermediaries, die die Nachricht auf dem Weg vom Absender zum Empfänger bearbeiten können. Diese Intermediaries führen ihre Aktionen (z.B.

[5]In der Version 1.0 von Microsoft und IBM stand SOAP für „simple object access protocoll“. Seit der Version 1.2, die vom W3C herausgegeben wurde, wird auf die Verwendung dieses Akronyms verzichtet.

Protokollierung,Abrechnung) typischerweise anhand der Information aus dem Header der Nachricht aus. SOAP ist sehr flexibel einsetzbar. Zum Einen kann es dazu verwendet werden im Body XML Daten und auch Nicht-XML-Daten (z.B. ein JPEG Bild) zu versenden. Zum Zweiten kann es mit vielen unterschiedlichen Netzwerkprotokollen (http, smtp, beep usw.) verwendet werden, aber auch mit anderen Technologien wie etwa IIOP oder Message Queuing. Zum Dritten unterstützt es viele unterschiedliche Interaktionsmuster: neben dem klassischen Request/Response-Muster wie man es für RPC benötigt, kann man auch einzelne Nachrichten verschicken oder beliebige andere Muster mit Hilfe von SOAP implementieren.

WSDL

WSDL[6] dient dazu, die Schnittstelle eines Web Services zu beschreiben. Das heißt,WSDL erlaubt die Deklaration von Operation und die Definition der Nachrichten, die ein Web Service empfangen kann und die er verschickt. Das Format, in dem Web Services beschrieben werden, ist XML. Derzeit arbeitet das W3C an WSDL Version 2.0. Der derzeit gültige Standard ist WSDL 1.1. Die verwendete Terminologie dieser beiden Versionen unterscheidet sich erheblich, doch die Prinzipien sind die gleichen. Da Version 2.0 voraussichtlich erst im Sommer 2006 formal verabschiedet wird und es dann noch einige Zeit dauern wird, bis Werkzeuge hierfür entwickelt worden sind und diese Version praktisch eingesetzt wird, beschreiben wir die Konzepte von WSDL anhand der derzeit eingesetzten Version 1.1.

Eine WSDL-Beschreibung eines Web Services beantwortet drei Fragen:

1. Was: Was bietet der Web Service? D.h.welche eingehenden Nachrichten versteht der Web Service, welche ausgehenden Nachrichten („output“, “faults“) erzeugt der Web Service und welche Operationen bietet der Web Service seinen Klienten zum Aufruf an? Ein Port Type bündelt eine Menge von Operationen, die mit denselben Mechanismen (i.e., Binding) aufgerufen werden können, d.h ein Port Type definiert die Schnittstelle eines Web Services.

2. Wie: Welche Protokolle werden zum Nachrichtenaustausch verwendet und wie werden die Nachrichten kodiert? Diese Information werden in so genannten Bindings spezifiziert.

3. Wo: Wie heißt der Web Service und unter welcher Internet-Adresse kann man Nachrichten an den Web Service schicken.

Wie am Anfang dieses Abschnittes erwähnt baut WSDL auf XML auf und alle Informationen werden in XML Dokumenten gehalten. Nach Version 1.1 sieht die Grobstruktur von WSDL-Beschreibungen wie folgt aus:

```
<wsdl:definitions xmlns:wsdl = "http://w3.org/...">
  <wsdl:documentation ... />
```

[6]Web Services Description Language

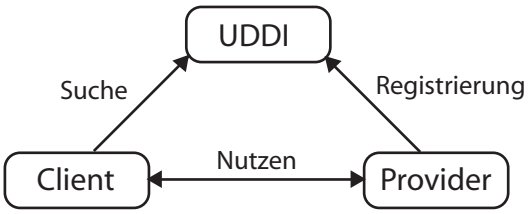

Abbildung A.3.: Verwendung von UDDI

```
  <wsdl:types> Schema Imports </wsdl:types>
  <wsdl:message> Nachrichten </wsdl:message>
  <wsdl:portType> Operationen </wsdl:portType>
  <wsdl:binding> Protokolle und Formate </wsdl:binding>
  <wsdl:service> Service Definition </wsdl:service>
</wsdl:definitions>
```

Hier ein simples Beispiel, das zeigt, wie die Nachrichten und Operationen eines Web Services, der die Operation „add“ als einzige Operation unterstützt, mit Hilfe von WSDL beschrieben werden können.

```
<wsdl:message name="addRequest">
  <wsdl:part name="term1" type="xs:double"/>
  <wsdl:part name="term2" type="xs:double"/>
</wsdl:message>

<wsdl:message name="addResponse">
  <wsdl:part name="value" type="xs:double"/>
</wsdl:message>

<wsdl:portType name="arithmetics">
  <wsdl:operation name="add">
    <wsdl:input message="addRequest"/>
    <wsdl:output message="addResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

UDDI

UDDI[7] dient zur Katalogisierung von Web Services. Es liefert somit einen wesentlichen Beitrag zum Management von Web Services und zur Virtualisierung von Diensten und Ressourcen. Abbildung A.3 zeigt die Rolle von UDDI bei der Nutzung von Web Services. Ein Provider (Anbieter eines Web Services) registriert seinen Web Service in

[7]Universal Description Discovery and Integration (OASIS)

einem UDDI-Server. Ein Klient (Nutzer) sucht nach geeignetem Web Services mit Hilfe eines UDDI Servers. Nach erfolgreicher Suche, kommuniziert der Klient direkt mit dem Provider, um den Web Service zu nutzen. Informationen über Web Services und Provider werden vom UDDI Server in sogenannten White, Yellow und Green Pages gehalten. Die Green Pages enthalten im Wesentlichen die WSDL-Schnittstellen der angebotenen Web Services. White und Yellow Pages enthalten Informationen zum Provider wie z.B.Adresse, Branche, Ansprechpartner.

Mit WSDL werden „lediglich“ die grundlegenden Eigenschaften eines Web Services definiert, nämlich dessen Schnittstelle und die Art und Weise seiner Erreichbarkeit (Formate, Protokolle,Aufenthaltsort). Weitere Eigenschaften wie etwa Qualitätseigenschaften (z.B. Transaktionsunterstützung, Sicherheit) oder geschäftlich relevante Informationen (z.B. Kosten, Zahlungsart) können durch Policies spezifiziert werden.

Typischerweise werden Policies beim Deployment (Bereitstellung des Web Services) spezifiziert. Nachdem die Implementierung eines Web Services installiert wurde, werden die entsprechenden Bindings, über die der Web Service erreichbar ist, definiert und anschließend die entsprechenden WSDL-Services samt ihrer Adressen spezifiziert. Dann wird das Verhalten, welches der Provider dieses Web Services garantieren will, in Form von Policies hinzugefügt. Die Laufzeitumgebung des Web Services stellt dann dieses Verhalten sicher.

Abgrenzung

Im Wesentlichen ist keines der Konzepte, die hinter Web Services stecken, neu: Es geht um bestimmte Varianten von distributed computing, die in der Vergangenheit bereits eingesetzt wurden. Zum Beispiel wurden Architekturen zur Enterprise Application Integration (EAI) bereits aufbauend auf CORBA und J2EE entwickelt. XML, das verwendete Datenformat, basiert auf SGML, das in den 70er-Jahren entwickelt wurde, und das XML-Datenmodell beruht auf Entwicklungen über semistrukturierte Datenmodelle. Für Web Services wurden die entsprechenden Konzepte und Technologien dem Problembereich angepasst und als XML-basierte Standards formuliert. Entscheidend für den gegenwärtigen Erfolg von Web Services ist, dass die diesen Konzepten entsprechenden Technologien nun aufeinander abgestimmt sind und von allen namhaften Herstellern unterstützt werden. Die immensen Investitionen aller Beteiligten in diesem Bereich resultieren in einer weit verbreiteten Unterstützung der Webservices-Standards und Standard-Vorschläge. Von großer Bedeutung ist auch der modularisierte Aufbau der Webservices-Technologie und -Standards: Web Services ist keine „Alles-oder-nichts-Entscheidung“, sondern man setzt die Technologien ein, die man zur Lösung eines Problembereichs benötigt.

Web-Services-Technologien sind Bausteine zur Lösung von Software-Integrationsproblemen. Sie setzen auf bestehenden Ansätzen in den Bereichen verteilte Systeme, Informationssysteme und Programmiersprachen auf und versuchen, diese Ansätze in ein einheitliches Bild zusammenzufügen und zu standardisieren. Web-Services-Technologien sind derzeit sehr populär sowohl im akademischen Umfeld als auch in der Industrie. Enor-

me Investitionen werden von Anwendern und Herstellern in diese Technologien getätigt. Fast schon aus diesem Grund sind Web Services zum Erfolg verurteilt.

A.2. Abkürzungen

In diesem Kapitel werden einige Abkürzungen aufgelistet, die in dem vorhergehenden Text verwendet wurden.

Spezifikationen

Kürzel	-	Ausgeschriebener Name (entwickelt von [siehe Abschnitt A.2])
CDIF	-	CASE Data Interchange Format (EIG)
CORBA	-	Common Object Request Broker Architecture (OMG)
CWM	-	Common Warehouse Metamodel (OMG)
HTML	-	Hypertext Markup Language (W3C)
HTTP	-	Hypertext Transfer Protocol
IDL	-	Interface Definition Language (OMG/ISO)
MDA	-	Model Driven Architecture (OMG)
PIM	-	Plattform Independend Model
PSM	-	Plattform Specific Model
MDIS	-	Meta Data Interchange Specification (MDC)
MOF	-	Meta Object Facility (OMG), oder auch Meta Object Framework (OMG)
OLAP	-	On Line Analytical Processing
OIM	-	Open Information Model (MS)
PMML	-	Predictive Model Markup Language (DMG)
SOAP	-	Simple Object Access Protocol (W3C)
UML	-	Unified Modeling Language (OMG)
URL	-	Uniform Resource Locator
XMI	-	XML Meta Data Interchange (OMG)
XML	-	eXtensible Markup Language (W3C)
DTD	-	Document Type Definition
XSLT	-	Extensible Stylesheet Language Transformations

Firmen und Vereinigungen

CA	-	Computer Associates
DARPA	-	Defence Advanced Research Projects Agency
EIG	-	Electronics Information Group
IBM	-	International Business Maschines
ISO	-	International Standards Organization
MDC	-	Meta Data Coalition
MS	-	Microsoft
NCR	-	National Cash Register
OASIS	-	Organization for the Advancement of Structured Information Standards
OEM	-	Original Equipment Manufacturer

OMG	-	Object Management Group
UBS AG	-	Union Bank of Switzerland
W3C	-	World Wide Web Consortium

Sonstiges

API	-	Application Programming Interface
CASE	-	Computer Aided Software Engineering
ETL	-	Data Extraction, Transformation and Loading
ISC	-	Information Supply Chain
IT	-	Information Technology
MbSE	-	Model-based Software Engineering Modellbasierte Software Entwicklung
OMA	-	Object Management Architecture
RFP	-	Request for Proposals
ROI	-	Return on Investment
RTF	-	Revision Task Force

Literaturverzeichnis

[CWMSpec1.0] Object Management Group (OMG): *Common Warehouse Metamodel (CWM) Specification Version 1.0* (October 2001)
`http://www.omg.org/cgi-bin/doc?formal/2001-10-01`

[CWMSpec1.1] Object Management Group (OMG): *Common Warehouse Metamodel (CWM) Specification Version 1.1* (March 2003)
`http://www.omg.org/cgi-bin/doc?formal/2003-03-02`

[MIPS] Object Management Group (OMG): *Common Warehouse Metamodel (CWM) Metadata Interchange Patterns (MIPS) 1.0* (March 2004)
`http://www.omg.org/cgi-bin/doc?formal/2004-03-25`

[CWMBook1] John Poole; Dan Chang; Douglas Tolbert; David Mellor:
Common Warehouse Metamodel: An Introduction.
John Wiley & Sons, Inc., New York (2002).
ISBN 0-471-20052-2

[CWMBook2] John Poole; Dan Chang; Douglas Tolbert; David Mellor:
Common Warehouse Metamodel Developer's Guide.
John Wiley & Sons, Inc., New York (2003).
ISBN 0-471-20243-6

[Cormen90] Thomas H. Corman; Charles E. Leiserson; Ronald L. Rivest:
Introduction to Algorithms.
The MIT Press, Cambridge, Messachusetts (1990).
ISBN 0-262-03141-8 (Hardcover)
ISBN 0-262-53091-0 (Paperback)

[UML2] Michael Jesse Chonoles and James A. Schardt:
UML 2 for Dummies.
Wiley Publishing, Inc., New York (2003).
ISBN: 0-7645-2614-6

[UMLDist] Martin Fowler:
UML Distilled - A Brief Guide to the Standard Object Modeling Language, 3rd ed. .
Addison Wesley (2004).
ISBN: 0-321-19368-7

[Raspl04] Stefan Raspl (IBM): *Overview of PMML Version 3.0* (August 2004)
http://www.google.de/?search="Overview PMML Stefan Raspl"

[WethShan79] Charles Wetherell and Alfred Shannon: *Tidy drawings of trees* (1979)
IEEE Trans. Software Engineering, 5, (5), 514-520

[Vaucher80] Jean G. Vaucher: *Pretty-printing of trees* (1980)
Software-Practice and Experience, 10, 553-561

[ReinTil81] Edward M. Reingold and John S. Tilford: *Tidier drawings of trees* (1981)
IEEE Transactions Software Engineering, 7, (2), 223-228

[SupRein83] Kenneth J. Supowit and Edward M. Reingold: *The complexity of drawing trees nicely* (1983)
Acta Informatica, 18, 377-392

[BruegWood89] Anne Brüggemann-Klein and Derick Wood: *Drawing Trees Nicely with TEX* (December 1989)
Electronic Publishing, Vol. 2(2), 101-115

[Walker90] John Q. Walker II.: *A node-positioning algorithm for general trees* (1990)
Software - Practice and Experience, 20(7):685-705

[Eades92] P.D. Eades: *Drawing Free Trees* (1992)
Bulletin of the Institute for Combinatorics and its Applications 5, 10-36

[Bloesch93] Anthony Bloesch: *Aesthetic Layout of Generalized Trees* (August 1993)
Software-Practice and Experience, Vol. 23(8), 817-827

[DGBib94] Battista, G., Eades, P., Tamassia, R. and Tollis, I.G.: *Algorithms for drawing graphs: An annotated bibliography* (1994)
Computational Geometry: Theory and Applications, 4 (5). 235-282

[Kennedy96] Andrew J. Kennedy: *Functional pearls: Drawing trees* (May 1996)
Journal of Functional Programming 6 (3):527–534

[GDTuto98] Isabel F. Cruz, Roberto Tamassia: *Graph Drawing Tutorial* (1998)
`http://www.cs.brown.edu/people/rt/gd.html`

[GDAlgo99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, Ioannis G. Tollis: *Graph Drawing - Algorithms for the Visualization of Graphs* (1999)
Prentice-Hall, 432 pp., ISBN: 0-13-301615-3

[BuchLeip02] Buchheim, Jünger, Leipert: *Improving Walker's Algorithm to Run in Linear Time* (2002)
Proc. GD'02, Springer LNCS 2528, pp. 344-353

[HasRahNis02] Masud Hasan, Md. Saidur Rahman, Takao Nishizeki: *A Linear Algorithm for Compact Box-Drawings of Trees* (2002)
Computational Geometry 2002

[TIRA05] Ai, Gerling, Neumann, Nitschke, Riehmann: *TIRA: Text based Information Retrieval Architecture* (September 11, 2005)
Proc. of the Workshop on Text-Based Information Retrieval (TIR-05)

[WebS04] Donald Kossmann, Frank Leymann: *Web Services* (April 2004)
Informatik Spektrum Band 27, Heft 2, Seite 117 - 128

[MDA05] Martin Kempa, Zoltán Mann: *Model Driven Architecture* (August 2005)
Informatik Spektrum Band 28, Heft 4, Seite 298 - 302

[ModernIR] Ricardo Baeza-Yates and Berthier Ribeiro-Neto:
Modern information retrieval.
ACM Press books (1999).
ISBN: 0-201-39829-X