Bachelor Thesis

Kontextbasierte Navigation in digitalen Archiven und Visualisierung multimedialer Inhalte am Beispiel von Museumsinformationssystemen

vorgelegt von
Michael Witt

Betreuer:

Prof. Dr. Jürgen Sieck
Dipl. Inf. Michael A. Herzog

Studiengang Angewandte Informatik (Bachelor)
Fachbereich Wirtschaftswissenschaften II
Fachhochschule für Technik und Wirtschaft Berlin

Berlin, im Juni 2009

Dankeschön an alle :)

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Der Besuch in einem Museum hat sich heute im Vergleich zu einem Besuch vor 20 Jahren stark verändert. Neben der eigentlichen Ausstellung wird dem Besucher häufig ein Multimedia-Guide angeboten, mit dem er zusätzliche Informationen zu ausgestellten Objekten abrufen kann.

Neben dem Angebot von Geräten, die dem Besucher persönlich zur Verfügung stehen, sind häufig Multimedia-Stationen zu finden, die mittels z.B. Film oder Ton Informationen bereitstellen. Verlässt der Besucher das Museum, wird der Media-Guide zurückgeben und der Zugriff auf digitale Informationen zu Exponaten wird außerhalb stark eingeschränkt, denn nur wenige Museen bieten über ihre Homepage Zugang zu Datenbeständen oder zu einer Benutzerschnittstelle zum Museums-Informationssystem an.

Die Navigation in diesen digitalen Archiven unterscheidet sich sehr von der traditionellen physischen Bewegung in einer Ausstellung. In deren Räumen befinden sich unter einem bestimmten Aspekt gruppierte Exponate, die der Besucher auf einen Blick erfassen kann. In der digitalen Repräsentation dagegen sind häufig lediglich mittels primitiver Methoden wie der Benutzung von Schlagworten Querverweise zu anderen Objekten angelegt; die Verbindungen zwischen Exponaten gehen i. d. R. verloren.

In dieser Arbeit soll, im Rahmen des *HardMut II*-Projektes [16], ein Prototyp entwickelt werden, der den Zugang zu digitalen Archiven von Museen dahingehend erweitert, dass diese Defizite beseitigt werden und ein virtueller Rundgang durch die digitale Repräsentation des Museumsbestandes um den Erforschungs- und Erkundungscharakter erweitert wird.

1.2 HardMut II-Projekt

Das *Hardmut II*-Projekt ist ein vom Europäischen Fonds für regionale Entwicklung (EFRE) gefördertes Gemeinschaftsprojekt zwischen der Hochschule für Technik und Wirtschaft Berlin und dem Jüdischen Museum Berlin.

Aufbauend auf Konzepten und entwickelten Technologien der Vorgängerprojekte *AMMELY* [14] und *HardMut I* [15] soll am Ende des Projektes eine

> „(...) Infrastruktur zur Entwicklung von mobilen Museen mit innovativen zielgruppengerechten Vermittlungsstrategien (...) “

entstehen.

Das mobile Museum, mit welchem das Jüdische Museum durch verschiedene Bundesländer Deutschlands reist und an Schulen über jüdische Kultur und Geschichte informiert, besteht aus einem Bestand von Exponaten die mit an die Schulen gebracht werden, sowie einer Sammlung von verschiedenen digitalen Medien anderer bspw. nicht transportabler Exponate oder jüdischer Kultur [16].

1.3 Zielsetzung

Ziel dieser Arbeit ist es, einen Prototypen zu entwickeln, welcher im Umfeld des HardMut II-Projektes einsetzbar ist und einen kontextsensitiven Zugang zum digitalen Archiv des mobilen Museums ermöglicht. Dabei soll bei der Entwicklung auch die Verwendung über die Grenzen des mobilen Museums hinaus berücksichtigt werden.

Besonders wichtig ist es bei der Gestaltung des Zuganges und der Visualisierung des Datenarchivs, die „innovativen und zielgruppengerechten Vermittlungsstrategien“[16] zu berücksichtigen und in den Entwurf der Anwendung einfließen zu lassen

1.4 Aufbau der Arbeit

Zuerst werden die Grundlagen, die notwendig sind, um einerseits die zur Verfügung stehenden Daten aus dem HardMut-Projekt zu verwenden und eine Anwendung zu entwickeln, die dem Benutzer optimalen Zugang zu den Datenbeständen des Museums gewährt, besprochen.

Anschließend werden verschiedene Anwendungen betrachtet, die Zugang zu digitalen Datensammlungen bieten. Dies geschieht einerseits um festzustellen, ob evtl. Referenzim-

plementierungen vorhanden sind, die in dieser Arbeit benutzt werden können, und andererseits um bewährte Ansätze zu identifizieren, wie der Zugang gestaltet werden kann. Zusätzlich werden verschiedene aktuelle Programmiersprachen und Frameworks betrachtet, die für die Implementierung einer solchen Anwendung benutzt werden können.

Danach sollen die Anforderungen, die an die Anwendung gestellt werden, betrachtet werden. Dazu gehören die Rahmenbedingungen, die durch das HardMut-Projekt gegeben sind und eingehalten werden müssen, sowie die Ermittlung aller Anwendungsfälle, die durch die zu entwickelnde Applikation berücksichtigt werden müssen.

Nachdem die Anforderungen an die Anwendung spezifiziert worden sind, sollen im nachfolgenden Kapitel der Systementwurf erstellt und erläutert werden. Die Umsetzung dieses Systementwurfes in den Prototypen wird im anschließenden Kapitel über die Implementierung beschrieben.

Abschließend soll der Prototyp und dessen Funktionalität gemäß der Anforderungen beurteilt und dessen Funktionalität demonstriert werden.

Kapitel 2

Grundlagen

In diesem Kapitel sollen die verschiedenen Bereiche, in denen sich diese Arbeit bewegt, vorgestellt werden. Diese Grundlagen stellen die Voraussetzung zur Entwicklung eines Prototypen zur kontextsensitiven Navigation in multimedialen Archiven dar.

Im ersten Abschnitt werden Technologien und Verfahren vorgestellt, die zur Gewinnung von Informationen aus unstrukturierten Texten dienen. Außerdem werden Techniken beschrieben, wie mit den gewonnenen Daten Ähnlichkeiten und Zusammenhänge zwischen Informationen identifiziert werden können. Diese Technologien werden zur Verarbeitung von Textdaten und Extraktion von semantischen Daten im HardMut-Projekt verwendet. Deshalb ist es für die Entwicklung essentiell, die Technologien und deren Funktionsweisen zu kennen, mit denen die vorliegenden semantischen Daten gesammelt wurden.

Anschließend werden zuerst Grundlagen zur Gewinnung von Kontextinformationen besprochen, danach verschiedene Definitionen und Arten von Kontext vorgestellt und abschließend, auf welche Weise mit diesen Daten zusätzliche Informationen über den Benutzer bzw. dessen Verhalten gewonnen werden können.

Zuletzt sollen unterschiedliche Ansätze zur Darstellung von Informationssammlungen diskutiert werden und wie damit die Navigation in diesen Datenbeständen realisierbar ist. Dabei wird besonders auf die Vorgehensweisen von Benutzern bei der Sammlung von Informationen und um die Unterstützung dieser Prinzipien durch Visualisierung eingegangen.

2.1 Text-Mining und semantische Analyse

Großteile von Informationen sind heutzutage digital als Textform verfügbar. Selten ist es dabei der Fall, dass dieser Text mittels z.B. Markup-Sprachen ausgezeichnet wurde, um

so eine automatische maschinelle Verarbeitung zu ermöglichen.

Text-Mining bezeichnet verschiedene Methoden, welche aus unstrukturierten Texten Informationen zu extrahieren. Zu Beginn dieses Abschnitts soll der Prozess des Text-Minings genauer betrachtet und die Bestandteile eines Text-Mining-Systems gezeigt werden.

Anschließend stehen Verfahren im Mittelpunkt, die einerseits beim Text-Mining angewandt werden, um Strukturen zu identifizieren, und so zusätzliche Informationen zu gewinnen und andererseits für die Suche von Dokumenten und die Erkennung von Zusammenhängen zwischen diesen von Nutzen sind.

2.1.1 Text-Mining

Beim Text-Mining handelt es sich um einen Prozess, in dem voll oder teilweise automatisiert aus einem unstrukturierten Textdokument enthaltene Informationen extrahiert werden. Im Idealfall handelt es sich bei diese Informationen um implizite, neue und relevante Aspekte des betrachteten Sachverhaltes [HQW06].

Text-Mining stellt eine interdisziplinäre Technologie dar, in der mathematische, statistische und linguistische Verfahren und Erkenntnisse kombiniert werden. Die Aufgabe des Text-Minings besteht darin, für den Benutzer brauchbare Informationen, welche aus dem Text extrahiert wurden, zu transformieren und dieses Resultat mittels Visualisierungstechniken zugänglich zu machen.

Wenn es dabei gelingt, dem Benutzer Kontext zugänglich zu machen und ihn zu befähigen, Rückschlüsse zu ziehen, die vorher nicht erkennbar waren, kann man von *Wissensextraktion* sprechen.

Neben der Wissensextraktion für den Nutzer kann Text-Mining auch in anderen Bereichen eingesetzt werden, um die Interaktion zwischen Mensch und Computer (*HCI*) zu vereinfachen. Dazu zählen unter anderem die Informationssuche mit natürlichsprachigen Anfragen oder Frage/Antwort-Systeme wie z.B. das *Wolfram—Alpha*-Projekt ([5]).

Ein *Text-Mining*-System besteht in der Regel aus vier Komponenten (Abbildung 2.1), die im folgenden näher erläutert werden sollen.

Datenbeschaffung In diesem ersten Schritt werden Textdokumente aus einer oder mehreren Quellen gesammelt und in einer Textdatenbank (*TDB*) gespeichert. Die zu speichernden Daten werden dabei aus ihrer ursprünglichen Form (z.B. einem HTML-Dokument) extrahiert und dekodiert (z.B. vom ASCII- zum UTF-8-Zeichensatz).

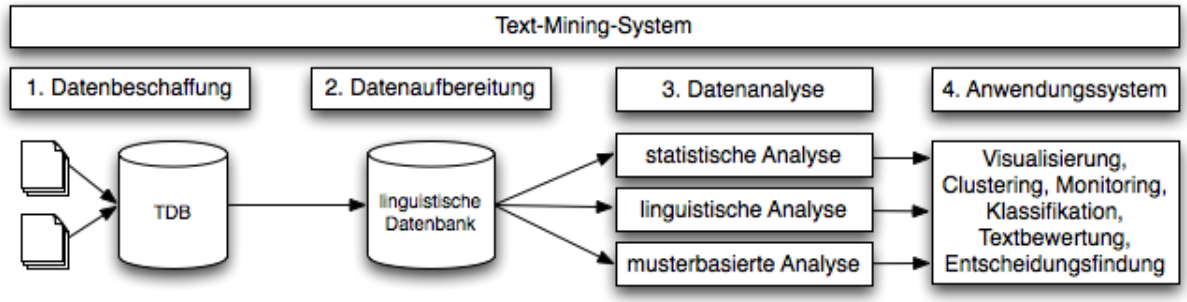

Abbildung 2.1: Schema eines Text-Mining-System nach [HQW06]

Datenaufbereitung Nach der Datenbeschaffung werden die Texte aus der Datenbank in eine oder mehrere linguistische Textdatenbanken unterteilt. Diese werden in domänenspezifische (Fachbegriffe u.a.) und allgemeinsprachliche Datenbanken unterteilt.

Die Dokumente werden ebenfalls in diesem Schritt in kleinere Einheiten wie Absätze, Sätze, Wortgruppen und Wörter unterteilt und indiziert. Über diese Einheiten wird eine künstliche Struktur gebildet, die, mit weiteren Informationen (Anzahl der Vorkommen im Dokument, relative Häufigkeit, etc.) angereichert, zur Datenanalyse benutzt werden kann.

Datenanalyse Die Datenanalyse stellt den komplexesten und zeitintensivsten Prozess im *Text-Mining-System* dar. Hier werden mit Hilfe der in der Datenaufbereitung gebildeten Strukturen versucht, unter Anwendung von musterbasierten, statistischen und linguistischen Verfahren, neues Wissen aus den Textdaten zu extrahieren [HQW06].

Anwengungssystem Im letzten Teil des Text-Mining-Systems werden die gewonnenen Informationen dem Nutzer verfügbar gemacht. Die Verfahren, die für die Visualisierung verwendet werden sollen, haben direkt Einfluss darauf, welche verschiedenen Daten im Prozess der Datenanalyse zu sammeln sind.

In den folgenden Abschnitten sollen die komplexeren Arbeitsschritte der Datenaufbereitung (2.1.2) und Datenanalyse (2.1.3) genauer betrachtet werden. Eine umfangreichere Betrachtung zur Gestaltung von Anwendungssystemen und Datenvisualisierung soll in Abschnitt 2.3.2 gegeben werden.

2.1.2 Verfahren zur Datenaufbereitung

Wie bereits beschrieben, durchläuft ein Dokument in einem Text-Mining-System vier Phasen, in denen verschiedene Anwendungsschritte ausgeführt werden, um Wissen zu extrahieren und dem Nutzer zugänglich zu machen. In diesem Abschnitt sollen Methoden

der Datenverarbeitungs-Phase betrachtet werden.

Oft wird im Zusammenhang mit Text-Mining auch das *Natural Language Processing* (NLP) angeführt. Hier werden Techniken verwendet, die ursprünglich zur automatischen Verarbeitung natürlichsprachiger Texte entwickelt wurden und das vollständige semantische Verständnis dieser zum Ziel hatten. Dabei wurden mittels linguistischer Konzepte (Grammatik, Wortarten, Koreferenzen) Informationen aus dem Text gewonnen; Verfahren, die in Text-Mining-Systemen bei der Wort- und Satzsegmentierung oder der Wortarterkennung ebenfalls eingesetzt werden.

Zusammensetzung von Dokumenten Für die in der Datenverarbeitungsphase durchgeführte Segmentierung von Text ist es wichtig, zuerst zu definieren, aus welchen Bestandteilen ein Dokument bzw. ein Abschnitt besteht. Zunächst stellen Texte eine Folge von Zeichen dar, wobei die Zeichen aus dem Alphabet der spezifischen Sprache, aus der Menge an Satz- und Sonderzeichen oder Ziffern stammen kann. Dabei werden mehrere Zeichen zu Wörter bzw. *Token* zusammengefasst [HQW06].

Aus diesen Token ergeben sich anschließend komplexere Strukturen wie Sätze, die als Sequenz von Wörtern aufgefasst werden können, und schließlich der Text als Ergebnis der Aneinanderreihung von Sätzen. Aus linguistischer Sicht betrachtet stellt das Wort nicht die kleinste Einheit in Satz dar, da es noch in weitere Bestandteile wie einzelne Silben oder Morpheme aufteilbar ist.

Betrachtet man Wörter informationstheoretisch als Daten, entsteht mittels Interpretation dieser Daten Information. Diese Informationen können entsprechend des Diskusbereiches des Text-Mining-Systems genutzt werden, um Wissen zu extrahieren. Zusätzlich gibt es *interne* und *externe* Merkmale, die auf zusätzliche Informationen des Textes schließen lassen [HQW06]. Zu den *internen* Strukturen zählen z.B. Absätze und Kapitel, während Merkmale wie der Name des Autoren, der Dokumententitel und das Verfassungsdatum *externe* Merkmale sind.

Wort- und Satzsegmentierung Um aus einem Dokument mit einem Textkörper eine Menge von Wörtern und Sätzen zu generieren, müssen Token im Text identifiziert und herausgelöst werden. Dieser als *Tokenisierung* bezeichnete Vorgang entscheidet maßgeblich über die Qualität folgender Arbeitsschritte, da bei fehlerhafter Tokenisierung nachgelagerte Prozesse nicht korrekt ablaufen und so im ungünstigsten Fall Informationen unerkannt bleiben können.

Für die Tokenisierung kommen u.a. *Finite State Transducer* oder einfache *trennzeichenbasierte Tokenizer* zum Einsatz. Bei ersteren ist das Ergebnis des Tokiniserungs-Prozesses

besser, da anstatt eines einfachen Trennzeichens zur Separation zu verwenden, ein endlicher Automat, der mit Regeln und einer Liste von gängigen Abkürzungen optimiert wurde, diesen Arbeitsschritt ausführt.

Part Of Speach Tagging Nachdem die einzelnen Wörter aus dem Text extrahiert wurden, werden mittels des Part Of Speach Taggings deren Wortarten bestimmt. Mit Hilfe dieser Technik lassen sich Bedeutungen eines Wortes im Text herausfinden sowie Token in Inhaltswörten (*content words*, Adjektive, Nomen, Verben und Adverben) und Funktionswörter (*functinal words*, Konjunktionen, Artikel, Pronomen und Präpositionen) unterteilen. Diese Unterteilung spielt gerade für die Erkennung musterbasierter Strukturen eine wichtige Rolle.

Im Zusammenhang mit dem Part Of Speach Tagging und der Bestimmung einzelner Wortarten wird für das Tagging von deutschen Texten häufig auf das *Stuttgart Tübingen Tag Set* ([12]) zurückgegriffen. In dieser Liste sind alle Wortarten der deutschen Sprache verzeichnet und mit eindeutigen Abkürzungen versehen.

Ein Problem bei der Bestimmung von Wortarten ist die Mehrdeutigkeit von Wörtern. Folgendes Bespiel soll dies verdeutlichen:

Die Verhandlung verlief für beide Seiten positiv.
Ich verlief mich letzte Nacht im Wald.

Hier hat das Wort *verlief*, welches in beiden Sätzen vorkommt, eine jeweils andere Bedeutung. Um die tatsächliche Wortart bei solchen Fällen zu ermitteln, wird der Kontext des Wortes im Satz betrachtet. *Statistisches Tagging* mittels Zustandsmodellen wie dem *Hidden Markov Modell* oder der *Maximum-Entropie-Methode* wird dafür verwendet, den korrekten POS-Tag für solche Wörter zu ermitteln. Zu erwähnen sind in diesem Zusammenhang auch lexikalische Tagger, die mit vorverarbeiteten Listen, in denen Wörter bereits entsprechenden POS-Tags zugeordnet sind, arbeiten.

Disambigiuerung von Wörtern Neben der Unterscheidung gleicher Wörter von unterschiedlichen Wortarten wie es beim Part Of Speach Tagging notwendig ist, geht es bei der Disambigiuerung von Wörtern um die Bedeutungsextraktion von mehrdeutigen Wörtern gleicher Wortart.

Als Beispiel sei hier auf das einfache Token der *„Bank“* verwiesen. Die unterschiedlichen Bedeutungen des Wortes, einerseits die Bezeichnung eines Geldinstituts und andererseits eine Sitzgelegenheit, haben erheblichen Einfluss auf die Kernaussage des Satzes, dessen

Extraktion somit erschwert wird. Das Ziel der Disambigiuerung ist es in diesem Fall, die korrekte Bedeutung des Tokens zu ermitteln und die Information des Satzes zu erhalten.

2.1.3 Bedeutungsanalyse von Textdaten

Nachdem das Textdokument verarbeitet, Token erkannt und dessen Wortarten und Bedeutungen ermittelt wurden, müssen nun Strukturen und Relationen identifiziert werden, um Wissen aus dem Dokument zu extrahieren.

Bei der Bedeutungsanalyse wird versucht, aus Inhalt und Zusammenhängen von Textmaterial Wissen zu gewinnen. Dabei stützt sich das Verfahren auf Erkenntnisse aus dem Forschungsbereich des *Linguistischen Strukturalismus*. Hier werden für eine Sprache $L = (W, S)$ mit der Menge alle Wortformen W und der Menge aller gültige Sätze S zwei Arten von Relation definiert.

Syntagmatische Relationen bestehen zwischen Wörtern, die im gleichen lokalen Kontext (z.B. Satz) auftreten. Dabei wird der lokale Kontext (K_l) eines Wortes $w_i \in W$ eines Satzes wie folgt definiert:

$$K_l(w_i) = \{w_1, ..., w_n\} \setminus \{w_i\}, \{w_1, ..., w_n\} \in S$$

Das als Kookkurrenz bezeichnete gemeinsame Auftreten zweier Wörter ($w_i, w_j \in W$) im gleichen lokalen Kontext wird durch die syntagmatische Relation SYN wie folgt beschrieben:

$$SYN(w_i, w_j) \leftrightarrow \exists K_s(w_i), w_j \in K_s(w_i)$$

Dieser Wert ist, selbstständig betrachtet, wenig aussagekräftig, da in einer natürlichen Sprache jedes Wort mit jedem anderen in einem Satz auftreten kann. Hier werden statistische Methoden zur Hilfe genommen, um auffällig häufige Kookkurenten zu erkennen und somit u.a. feste Wendungen oder Aufzählungen zu identifizieren.

Damit das Kookkurenzverhältnis zwischen Wörtern vergleichbar wird, wird ein *Signifikanzmaß* ($sig(w_i, w_j)$) eingeführt, dessen Berechnung an die *Poisson Verteilung* angelehnt ist [HQW06].

Bei *paradigmatischen Relationen* wird die Bedeutungsrelation zweier Wörter betrachtet (z.B. Synonymie). Dabei wird im Gegensatz zur syntagmatischen Relation nicht der lokale, sondern der globale Kontext eines Wortes ($K_g(w_i), w_i \in W$) verwendet. Dieser wird aus allen lokalen Kontexten von w_i gebildet, wobei nur Wörter ($w \in W$) berücksichtigt

werden, die eine statistisch hohe *syntagmatische Relation* zu w_i haben (SYN). Es gilt für $K_g(w_i)$ also formal:

$$K_g(w_i) = \{w|w_i, w \in W \wedge SYNS(w_i, w)\}$$

Mittels der paradigmatischen Ähnlichkeit zweier Wörter, welche sich aus dem Ähnlichkeitsprädikat $SIM(K_g(w_i), K_g(w_j))$ berechnet, können verschiedene logische Ausprägungen der paradigmatische Bedeutungsrelationen wie Synonomie, Antonymität, Inkompatibilität, Komplementarität und Hyperonomie ermittelt werden.

2.1.4 Ähnlichkeit zwischen Termen und Dokumenten

Die Berechnung von Ähnlichkeit zwischen einzelnen Wörtern, Termen, Dokumenten oder anderen Textkorpora ist essentiell für die Bedeutungsanalyse im Sinne der paradigmatischen Relation oder etwa bei der Suche von Mustern. Häufig wird für die Ähnlichkeitsberechnung auf eine mathematische Darstellung des Textes in einem Vektorraum zurückgegriffen.

Transformation von Text in eine vektorielle Darstellung Bei der Umwandlung von Texten in einen Vektorraum (*semantic Space* [Leo05]) steht als Ergebnis eine Matrix mit Vektoren, welche die linguistischen Einheiten mit ihren Zusammenhängen und Wertigkeiten beschreiben.

Beispielsweise sei hier der Dokumentenraum M_{doc} genannt:

$$M_{doc} = \begin{pmatrix} a_{11} & \cdots & a_{1Y} \\ \vdots & \ddots & \vdots \\ a_{X1} & \cdots & a_{XY} \end{pmatrix}, a_{i,k} = f(w_i, d_k)$$

In dieser Matrix wird der Zusammenhang zwischen einem Wort $w_i \in W$ und einem Korpus d_k, aus der Menge aller Korpora, als Element $a_{i,k}$ der Matrix M_{doc} ausgedrückt. Dieser Zusammenhang kann beispielsweise eine Aussage über das Vorkommen oder die Häufigkeit des Wortes im Korpus sein.

Berechnung der Ähnlichkeit Durch die Transformation in die vektorielle Darstellung wird die algorithmische Berechnung der semantischen Ähnlichkeit zwischen Texten möglich. Dabei gibt es einerseits Verfahren, die mittels binärer Operatoren arbeiten und andererseits Berechnungen, welche den Grad der Ähnlichkeit oder die Distanz zweier semantischer Konstrukte zueinander beschreiben.

Näher beschrieben werden soll hier die Methode der Ähnlichlichkeitsberechnung mittels Kosinusmaß, da dieses Verfahren bei der Suche von Exponaten im HardMut-Projekt eingesetzt wird.

Durch die Verwendung des Kosinus als Maßzahl ist der Wertebereich für Ergebnisse gleich dem Wertebereich der Kosinus-Funktion und somit beschränkt. Bei der Berechnung von Ähnlichkeit zwischen Dokumenten kommt weiterhin hinzu, dass bedingt durch die Unmöglichkeit negativer Werte in Termvektoren dieser Wertebereich weiter verkleinert wird ($SIM_{cos} : [0^o, 90^o] \longrightarrow [0, 1]$).

Dieser Wert wird berechnet, indem das Skalarprodukt zweier Vektoren A und B gebildet und durch das Produkt der Längen dieser Vektoren dividiert wird. Dabei ist A eine bestimmte Menge zusammengehöriger Terme wie z.B. ein Dokument oder ein globaler Kontext und B der Suchvektor.

$$SIM_{cos}(A, B) = \frac{A \bullet B}{|A| \times |B|}$$

Da A und B häufig Vektoren unterschiedlich hoher Dimensionen sind, wird der Vektor mit der höheren Dimension dem mit der niedrigeren angepasst, indem Werte, die im höher-dimensioniertem Vektor enthalten sind, nicht aber im anderen vorkommen, entfernt werden.

Der maximale Winkel zwischen den Term-Vektoren beträgt 90^o genau dann, wenn sie keine gemeinsamen Terme haben (Abbildung 2.2). Je kleiner der Winkel, desto ähnlicher sind sich die Vektoren. Aus diesem Distanzmaß (kleiner Wert $\widehat{=}$ große Ähnlichkeit) kann mittels der Formel $1/SIM_{cos}(A, B)$ das Ähnlichkeitsmaß (großer Wert $\widehat{=}$ große Ähnlichkeit) berechnet werden.

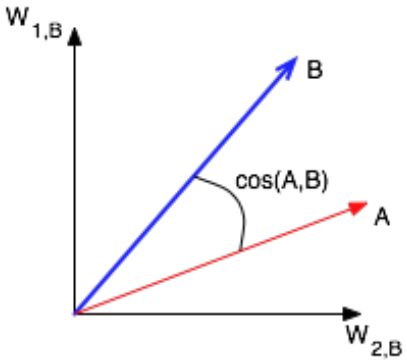

Abbildung 2.2: Kosinusberechnung zwischen Termvektoren

2.2 Kontext und kontextsensitive Applikationen

Die Kommunikation zwischen zwei Menschen ist der Kommunikation zwischen Mensch und Maschine weit überlegen, da der Mensch bei der Nachricht implizit Sachinformation, Beziehung zu anderen Person, Selbstoffenbarung und Appell übermittelt. Zusätzlich kann der Mensch auf allgemein bekanntes Wissen (Weltwissen) und die Situation, in der er sich befindet, zurückgreifen, um Kommunikation zu betreiben. So ist es zum Beispiel ausreichend zu sagen: *„Heute ist ein schöner Tag.“*, anstatt *„Heute ist ein schöner Tag, weil die Sonne scheint.“*, da durch die Anwendung bekannten Wissens („Ein sonniger Tag ist schön“) und die Einbeziehung der Situation (Sonne steht am Himmel) die Aussage der kurzen Nachricht klar ist [DA08][18].

Menschen benutzen Kontext, um effektiver miteinander zu kommunizieren. Folglich kann Kontext dazu benutzt werden, auch die Kommunikation zwischen Mensch und Computer effizienter zu gestalten. Um dies zu erreichen, muss jedoch der Computer bzw. die Applikation dazu befähigt werden, Kontext zu erkennen und zu verarbeiten. Man spricht in diesem Zusammenhang von *context-awareness*.

In folgenden Abschnitten soll zuerst eine Definition von Kontext gegeben werden, mit der Kontext in dieser Arbeit verwendet werden soll. Anschließend wird eine Übersicht über die Möglichkeiten der Gewinnung von Kontext gegeben und abschließend mögliche Anwendungsgebiete mit Fokus auf *Recommender-Systeme* vorgestellt.

2.2.1 Begriffsdefinition

In vielen Publikationen wird der Begriff des Kontexts verwendet, jedoch wird er immer wieder unterschiedlich interpretiert. So definieren Schilit und Theimer ([ST97]) Kontext als Position, Personen und Objekt in der Umgebung sowie deren Veränderung über einen gewissen Zeitraum. In der Beschreibung von Brown u. a. ([BBC97]) kommen weitere Aspekte wie Tageszeit, Jahreszeit und Umgebungstemperatur hinzu.

Andere Definitionen weichen in einigen Punkten geringfügig ab oder fassen einzelne Aspekte wie Temperatur und Tageszeit zu „Umgebung/ Umwelt des Benutzers“ zusammen. Schilit et al. hält ebenfalls fest, dass sich Kontext aus dem Ort, an dem sich der Benutzer aufhält („Where you are“), den Personen, die ihn umgeben („who you are with“) und den Ressourcen der Umgebung („resources nearby“) zusammensetzt als eine Anwendungsumgebung, die im ständigen Wandel begriffen ist ([SAW94]).

Diese Arbeit wird den Begriff des Kontexts nach der Definition von Dey und Abowd benutzen.

> „Context is any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves. “[DA08]

Diese sehr abstrakte Auffassung von Kontext ermöglicht es, für den spezifischen Diskursbereich einer Anwendung die Arten von Kontext zu identifizieren, die verarbeitet werden sollen; als Beispiel sei eine Office-Anwendung genannt. Die Interaktion findet zwischen Mensch und Anwendung statt. Da umgebende Personen für den Verlauf der Anwendung irrelevant sind, sind sie in diesem Sinne nicht Kontext, andererseits kann die Anwendung beispielsweise eingegebenen Text dazu verwenden, um auf den Zweck des Dokumentes zu schließen (z.B. einen Brief) und den Benutzer in dessen Erstellung unterstützen.

Die verschiedenen Arten von Kontext werden in zwei Kategorien unterschieden. Als *primärer Kontext* werden alle Eigenschaften bezeichnet, welche die Situation, in der sich eine Entität befindet, beschreiben. Dazu zählen z.B. Position, Identität, Zeit und Aktivität. *Sekundärer Kontext* hingegen fasst Attribute einer Entität zusammen, die mittels einer Eigenschaft des primären Kontextes ermittelt werden können. Beispielsweise ist es möglich, mittels der Identität (z.B. Benutzername) eines Benutzers dessen E-Mail-Adresse zu ermitteln.

Der Begriff der *context awareness* wird von Day und Abowd folgendermaßen definiert und zeigt, dass ein System schon dann als kontextsensitv gilt, wenn es beliebigen Kontext dazu nutzt, relevante Dienste und/oder Informationen dem Benutzer bereitzustellen [DA08].

2.2.2 Gewinnung von Kontext

Wie im oberen Bespiel gezeigt, kann im einfachsten Fall Text, den der Benutzer eingibt, als Kontext verwendet werden. In allen Fällen ist es jedoch notwendig, die eingehenden Daten zu bearbeiten, um relevante Informationen zu filtern, bevor die Ausgabe an den Benutzer erfolgt. Dies beschreiben die drei Hauptarbeitsschritte einer kontextsensitiven Applikation (Abbildung 2.3).

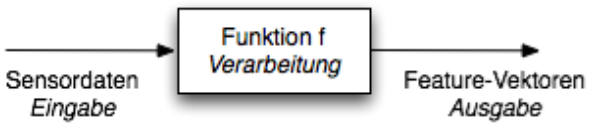

Abbildung 2.3: Schematischer Prozess der Kontextgewinnung

Dieses abgeschlossene Eingabe-Verarbeitung-Ausgabe-System kann auf komplexere Systeme bezogen werden und ermöglicht so die Einteilung des Kontextgewinnungs-Prozesses

in Teilschritte. Abbildung 2.4 zeigt ein solches System und soll als Referenz für die Anschließende Erläuterung der einzelnen Phasen der Kontextgewinnung dienen.

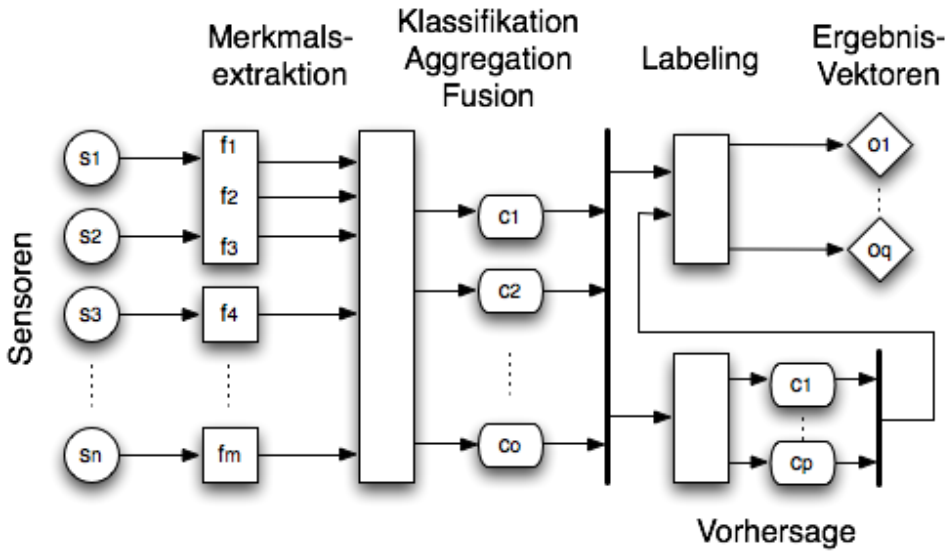

Abbildung 2.4: Prozesskette der Kontextgewinnung nach [May04]

Sensoren Sensoren verarbeiten Daten aus der Umwelt des Benutzers. Bei diesen Daten handelt es sich entweder um analoge oder digitale Signale, die in einem bestimmten Zeitintervall und/oder Wertebereich aufgezeichnet werden.

Dabei werden Sensoren in *physisch* und *logisch* unterteilt. Während physische Sensoren mittels Messverfahren Umgebungsdaten wie Temperatur, Umgebungshelligkeit und Beschleunigung ermitteln, können logische Sensoren beispielsweise die Verfügbarkeit von Internetdiensten oder eingegebenen Text widerspiegeln.

Der von den Sensoren für die Datenverarbeitung zur Verfügung gestellte Eingabe-Vektor enthält häufig neben numerischen Werten (Helligkeit, Geschwindigkeit, Zeit) auch nominelle (Benutzername) oder ordinale Daten (Rangfolgen), deren Verarbeitung deutlich schwieriger ist. Diese Daten des Eingabe-Vektors werden als *low-level Sensordaten* bezeichnet.

Klassifikation, Aggregation und Fusion Die *Klassikfikation* stellt den komplexesten Schritt bei der Verarbeitung von Signaldaten dar und wird daher oft mit der Kontextgewinnung gleichgesetzt. Bei der Klassifikation wird versucht, Muster in den Eingabedaten zu identifizieren und diese einer bestimmten Klassen von Objekten zuzuordnen. Dabei ist es möglich, das ein Merkmal mit einer bestimmten Warscheinlichkeit zu mehreren Klassen gehört.

Hierbei wird zwischen *kontrollierter* und *unkontrollierter* Klassifikation unterschieden. Bei der kontrollierten Klassifikation sind die Objektklassen bereits bekannt und vorgegeben, während bei der unkontrollierten Klassifikation mittels Algorithmen wie Hidden Markov Modellen, neuronalen Netzen oder unscharfer Logik zur Laufzeit Objektklassen identifiziert und Merkmale diesen Klassen zugeordnet werden.

Bei der *Aggregation* werden verarbeitete Daten, die bereits Rückschlüsse auf Kontext ermöglichen, miteinander verbunden, um weitere Informationen zu erhalten.

Im *Fusions*-Arbeitsschritt werden gleichartige Daten miteinander verbunden, um Fehler und Abweichungen zu minimieren und so die Genauigkeit und Zuverlässigkeit der Ergebnisse zu erhöhen.

Nachdem diese drei Arbeitsschritte durchgeführt wurden, ist es möglich, den extrahierten Daten Erkenntnissen zuzuordnen (*Labeling*) oder den zukünftigen Verlauf der Anwendung und dessen Umgebung vorherzusagen (*Vorhersage*).

2.2.3 Kontextsensitive Applikationen

Es existieren verschiedene Möglichkeiten, wie Anwendungen Kontext benutzen können um bsw. dem Benutzer zusätzliche Informationen bereitzustellen oder die Benutzung der Anwendung zu vereinfachen. Zwei Arten, die hier näher beschrieben werden sollen, sind Systeme zur *kontextsensitiven Informationssuche* und *Recommender-Systeme.*

Kontextsensitive Informationssuche Anders als bei Applikationen zur Informationssuche wie Suchmaschinen, die Kontext nicht für ihre Suche berücksichtigen und jede Suchanfrage losgelöst von der vorhergegangenen ausgeführt wird, werden Suchbegriffe bei kontextsensitiven Diensten miteinander verbunden. Dies ist besonders von Nutzen, wenn der Benutzer nicht genau beschreiben kann, wonach er sucht (bsp. ein „schönes Ferienhaus“). Zhou u. a. zeigen in einer Applikation zur Suche von Immobilien, das gerade in Bereichen, in denen nach einem am Anfang stehenden allgemeinen Suchbegriff (z. B. „Eigenheim 500000 Euro“) die Suche iterativ verfeinert wird, um die Ergebnisanzahl einzuschränken, kontextsensitve Applikationen den Prozess beschleunigen [ZHP+06].

Die Schwierigkeit bei solchen System liegt darin, dass die Suchbegriffe im korrekten Bezug zueinander gesetzt werden müssen. Dadurch, dass Kontextinformationen jedoch oftmals mehrdeutig sind und die Verknüpfung falscher Sachverhalte den Benutzer in seiner Suche mehr beeinträchtigt als unterstützt, werden besonders bei der Kontextanalyse hohe Anforderungen an solche Anwendungen gestellt.

Recommender-Systeme Eine weitere Art Anwendung sind Recommender-Systeme, die basierend auf Kontextinformationen, die sie gesammelt haben, dem Benutzer Empfehlungen beliebiger Art geben und so den Zugang zu großen Informationsdatenbanken erleichtern ([CSS07]).

Die Intention eines Recommender-Systems ist es, den Benutzer in der Entscheidungsfindung bei der Suche von Informationen mittels Kontextinformationen zu unterstützen (vgl. 2.3.1.1). Diese Informationen werden aus den Eingaben des aktuellen Benutzers, aus Eingaben vorheriger Benutzer und dem Datenbestand, in dem gesucht werden soll, gesammelt und vom Recommender-Algorithmus verarbeitet. Das Ergebnis wird anschließend an den Benutzer weitergegeben (Abbildung 2.5).

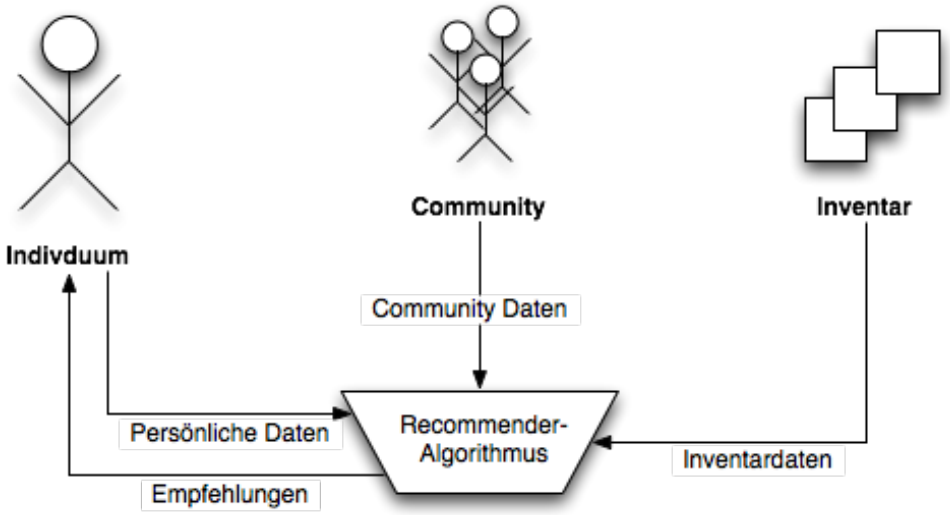

Abbildung 2.5: Datenquellen eines Recommender-Systems (Quelle: nach [CSS07])

Online-Versandhäuser bedienen sich beispielsweise Daten zu Einkäufen ihrer Kunden und können so anderen Kunden Informationen über Produkte geben, die häufig mit den Waren gekauft werden, die sie bereits besitzen oder im Begriff sind zu erwerben.

2.3 Informationszugang und -visualisierung

Je umfangreicher digitale Archive werden, umso wichtiger wird es, dem Benutzer einen einfachen und umfangreichen Zugang zu diesen Datensammlungen zu ermöglichen. Andernfalls bleibt er erfolglos bei der Suche nach spezifischen Daten und verliert das Interesse am digitalen Archiv.

Zu Beginn dieses Abschnitts sollen einige Ansätze vorgestellt werden, wie der Benutzer vorgeht, um an Informationen zu gelangen. Dabei soll die Betrachtung nicht nur auf die

Navigation auf Webseiten oder die Navigation mittels grafischer Benutzerschnittstelle beschränkt bleiben; vielmehr sollen Prinzipen aufgezeigt werden, die psychologische Aspekte beim Sammeln von Informationen behandeln.

Anschießend werden verschiedene Methoden der Visualisierung von Datenbeständen aufgezeigt, welche den Zugang zu Informationen erleichtern. Zuletzt sollen Ansätze diskutiert werden, die dazu dienen können, um die Navigation und die Orientierung in den virtuellen Umgebungen zu vereinfachen.

2.3.1 Informationszugang

In Zeiten von kostengünstigem Speicherplatz und somit immer mehr gespeicherten Daten wird es zunehmend wichtiger, auch in großen Datenbeständen einfach und gezielt auf Informationen zugreifen zu können. Dabei kommt es darauf an, wie es dem Benutzer ermöglicht wird, an die für ihn interessanten Daten zu gelangen.

Während Suchfunktionen die naheliegendste Möglichkeit darstellen, zu großen Datenbeständen einen Zugang zu finden, sollen in diesem Abschnitt vorrangig Prinzipien vorgestellt werden, wie der Benutzer intuitiv versucht, sich zu einem ihm unbekannten System Zugang zu verschaffen. Mit Hilfe dieser Erkenntnisse können dann Prinzipen für die Visualisierung und die Navigation in digitalen Archiven gewonnen werden, um diese auf die Gestaltung des Prototypen anzuwenden.

Ebenfalls wichtig für den Zugang zu digitalen Informationssammlungen ist die Benutzerschnittstelle (Browser, GUI). Da diese jedoch auch von großer Bedeutung bei der Navigation ist, soll dieser Aspekt im Abschnitt zur Navigation in digitalen Archiven (2.3.3) behandelt werden.

2.3.1.1 Information foraging

Das *Information foraging*, welches frei übersetzt werden kann mit *Horten von Informationen*, bezeichnet das Verhalten einer Person beim Suchen und Sammeln von Informationen [Pir07]; dabei stützt sich diese Theorie auf Erkenntnisse aus dem Bereich der Biologie und Anthropologie. Hier wird beobachtet, wie Tiere Vorräte sammeln, um z.B. den Winter zu überleben. Diese Beobachtungen werden auf das Beschaffen von Informationen übertragen. Dabei muss stets zwischen dem Aufwand der für die Beschaffung des Wissens und dem Wert der Information im Moment abgewogen werden. Dafür muss auf gesammelte Erfahrung zurückgegriffen werden, um diesen Wert abzuschätzen, noch bevor genauere Details bekannt sind.

Pirolli wendet diese Erkenntnisse in seinem *Scatter and Gather Browser* an [Pir07]. Dabei werden Dokumente hierarchisch in Clustern angeordnet (2.3.2.1). Der Benutzer identifiziert einen Cluster durch eine automatisch erzeugte Zusammenfassung der enthaltenen Dokumente. Entspricht dieser Cluster seinem Interesse, kann er tiefer in die Hierarchie vordringen und findet so weitere kleinere Cluster oder einzelne Dokumente.

Speziell das Internet bzw. eine Internetpräsens ist ein gutes Beispiel für die vielen Entscheidungen, die ein Nutzer bei der Sammlung von Informationen treffen muss. Oftmals verzweigt der Weg, auf dem sich der Besucher bewegt, und er kann nur im optimalen Fall einschätzen, welcher Zweig ihn dem Ziel ein Stück näher bringen wird. Im ungünstigsten Fall wird er keine Anhaltspunkte finden, die ihn bei der Navigation unterstützen und so eine zufällige Entscheidung treffen.

Deswegen stellt Pirolli fest, dass es von Bedeutung ist, dem Benutzer zu ermöglichen, den beschrittenen Pfad zurückzugehen und ggf. einen anderen Weg einzuschlagen. Auch können Darstellungen des Gesamtsachverhalts (z.B. Sitemaps, Navigationsbäume, usw.) den Zugang zu digitalen Archiven erleichtern.

2.3.1.2 Affordence-Theorie

Die *Affordence-Theorie* ist ein wahrnehmungspsychologischer Ansatz von J. J. Gibson. Dabei wird angenommen, das die Wahrnehmung eines Objektes direkt mit möglichen Handlungen assoziiert wird, welche mittels dieses Objektes ausgeführt werden können [Gib79].

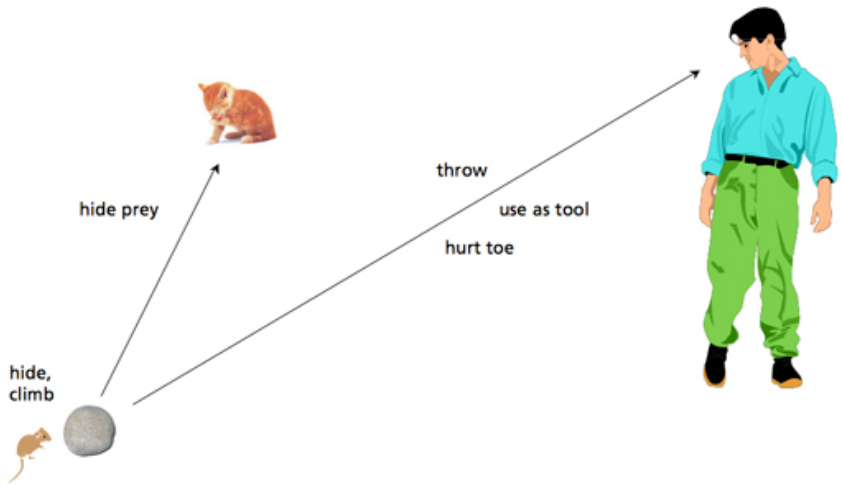

Abbildung 2.6: Wahrnehmung von Objekten nach Gibson (Quelle: [10])

Gibsons Theorie ist dabei eng verbunden mit der These der direkten Wahrnehmung (*direct perception*). Es wird davon ausgegangen, dass Handlungsmöglichkeiten (*affordences*) di-

rekt wahrgenommen und nicht erst durch die Kombination verschiedener Indizien erkannt werden. In Abbildung 2.6 wird diese Art der Wahrnehmung grafisch verdeutlicht. Es zeigt, dass der Mensch den Stein als benutzbares Werkzeug identifiziert sowie die Möglichkeiten erkennt, ihn zu werfen oder sich daran den Fuß zu verletzen.

Der Ansatz von Gibson ([Gib79]) gibt wichtige Hinweise darauf, wie der Zugang zu digitalen Archiven gestaltet werden sollte. Aktionen, die der Benutzer auslösen kann, müssen deutlich erkennbar sein. Ein Beispiel dafür zeigt der Entwurf in Abbildung 2.7. Hier werden Kontrollelemente zur Manipulation des Objekte im Raum (drehen, bewegen, usw.) in Form von Händen angezeigt, und so kann der Benutzer gemäß der Gibson-Theorie direkt auf seine Optionen schließen. So gestaltet sich der Zugang zu zwei- oder dreidimensional visualisierten Informationssystem wesentlich einfacher.

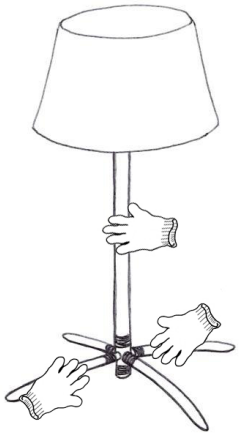

Abbildung 2.7: Kontrollelemente zur Objektmanipulation

2.3.2 Visualisierung

Bei der Darstellung von großen Informationssammlungen gibt es mehrere Aspekte, die berücksichtigt werden müssen. Einerseits ist es essentiell, die Daten und deren Zusammenhänge visuell zu transportieren und darzustellen, andererseits muss der Benutzer einen Zugang zu der Darstellung haben.

Gerade bei großen Datensammlungen wird dieser Zugang erschwert, wenn dem Benutzer zu viele Daten zur gleichen Zeit präsentiert oder Zusammenhangslos dargestellt werden. Deshalb sollen nachfolgend einige Methoden vorgestellt werden, wie Daten übersichtlich und ansprechend präsentiert werden können.

2.3.2.1 Clustering

Das Clustering-Verfahren zielt darauf ab, die unübersichtliche Datenmenge nach definierten Kriterien in Teilmengen zu zerlegen und so dem Benutzer den Zugang zu erleichtern. Die Auswahl der Kriterien, nach denen die Cluster gebildet werden sollen, wird dabei nach dem Diskursbereich der Anwendung bestimmt wie z.B. Größe, Form oder abstraktere Eigenschaften wie semantische Ähnlichkeit.

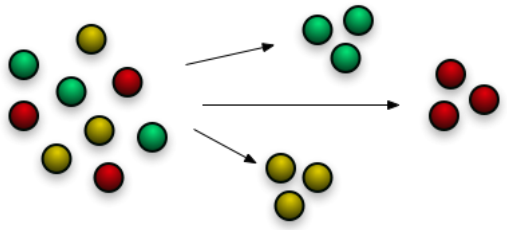

Abbildung 2.8: Bildung von Clustern anhand von Farbeigenschaften

Algorithmen zum Erzeugen von Clustern aus einem Daten-Pool basieren auf der Ähnlichkeit dessen Elemente und errechnen damit deren Nähe bzw. Entfernung zueinander. Mittels des Minkowski-Modells ist eine allgemeine Definition für die Entfernung gegeben, welche durch folgende Formel ausgedrückt werden kann:

$$d_{ij} = \left[\sum_{a=1}^{r} |x_{ia} - x_{ja}|^p \right]^{1/p} \quad (p \geq 1), x_i \neq x_j$$

Hier bezeichnet d_{ij} den Abstand zwischen den Elementen x_i und x_j. Mittels des Parameters p lassen sich die verschiedenen Distanzarten generieren. Beispielsweise liefert das Minkowski-Modell für $p = 2$ den euklidischen Abstand der Elemente als Ergebnis.

Es werden drei verschiedene Arten unterschieden, wie eine Datenmenge in Cluster unterteilt werden kann:

Graphentheoretische Methode Hierbei werden im ersten Schritt mit Hilfe des Minkowski-Modells, angewandt auf ausgewählte Eigenschaften, die Ähnlichkeit der Elemente identifiziert und so Cluster gebildet. Dabei bildet jeder Cluster einen Graphen, bei dem alle Elemente miteinander verbunden sind. (Verweis van Rijsbergen)

Bei diesem Verfahren gibt es drei Möglichkeiten, wie sich diese Graphen bilden können (*complete link*, *single link*, *average link*), auf die kurz eingegangen werden soll.

Beim *complete link*-Verfahren werden um ein Element alle weiteren Elemente gruppiert, deren Abstand ein festgelegtes Maximum nicht überschreitet. In den folgenden n Schritten

wird der neu gebildete Cluster als Element behandelt und erneut alle Elemente, die das für diesen Rechenschritt festgelegte Maximum nicht überschreiten, gruppiert. Es bildet sich auf diese Weise eine hierarchische Struktur heraus.

Im Gegensatz zum complete link-Verfahren entsteht beim *single link*-Verfahren ein minimal spannender Baum, da, anstatt alle Elemente auf Basis eines Maximalwertes zu einem Cluster zuzuordnen, im single link-Verfahren jeweils nur die Elemente mit der geringsten Distanz zueinander verbunden werden.

Beim *average link*-Verfahren werden complete link- und single link-Methoden kombiniert, um einen Kompromiss zwischen dem Rechenzeitbedarf bei der Clusterbildung und Übersichtlichkeit bei der Graphenbildung zu finden.

Single-pass-Verfahren Beim *Single-pass*-Verfahren werden die Cluster in einem einzigen Rechenschritt berechnet. Ein Beispiel für eine solche Technik ist das *seed-oriented clustering*. Hierfür werden im ersten Schritt n Startelemente (*seeds*) aus der Datenmenge ausgewählt. Das bedeutet, dass die Anzahl der zu bildender Cluster zu Beginn bekannt sein muss. Anschließend werden Elemente, die einem *seed* ähnlich sind, um diesen gruppiert.

Iterative Verfahren *Iterative Verfahren* benutzen z.B. *graphentheoretische Methoden* oder *single-pass*-Verfahren, um Cluster zu berechnen. Allerdings werden diese Methoden wiederholt in mehreren Rechenschritten angewendet und Parameter verändert, um so Ergebnisse auf Basis von heuristischen und statistischen Verfahren zu verbessern.

2.3.2.2 Hyperbolische Repräsentationen und Baumansichten

Bei der Visualisierung von Hierarchien sind Baumansichten sehr gebräuchlich wie z.B. bei der Anzeige von Verzeichnisstrukturen. Eine Möglichkeit Baumstrukturen auf einem zweidimensionalen Feld zu visualisieren, sind *TreeMaps*. Hiermit kann neben der hierarchischen Struktur auch die Größe der Elemente dargestellt werden (Abbildung 2.9).

Die Darstellung von großen Strukturen macht es zunehmend schwieriger, den Zusammenhang zwischen dem fokussierten und übergeordneten Elementen sowie anderen Knoten zu erkennen. Dieses als *focus versus context* bekannte Problem tritt besonders bei zweidimensionalen Visualisierungsmethoden wie *TreeMaps* auf.

Aus diesem Grund werden für die Darstellung von großen Hierarchien oft dreidimensionale Visualisierungen wie *Cone Trees* eingesetzt, da sie mehr Elemente derart im dreidimensionalen Raum anzeigen können, dass einerseits eine Fokussierung möglich ist, um Daten

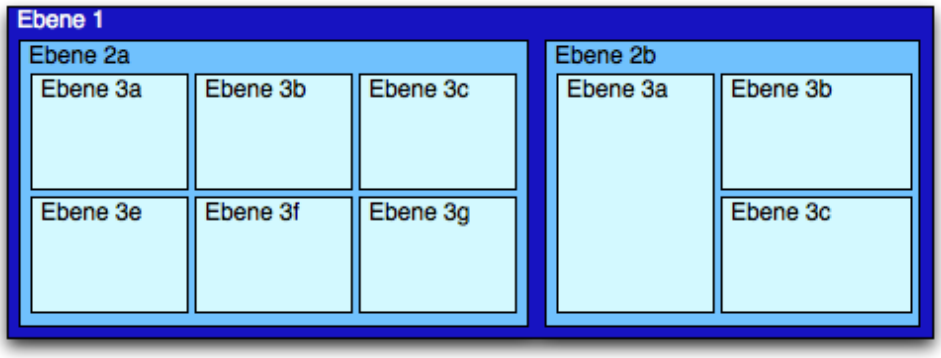

Abbildung 2.9: TreeMap-Visualisierung mit 3 Ebenen

eines Elements abzurufen, jedoch der Kontext dieses Elements weiterhin erkennbar bleibt.

Bei *Cone Trees* werden Strukturen mittels zylindrischer Objekte visualisiert. Abbildung 2.10 veranschaulicht wie von einem Wurzelelement ausgehend Unterpunkte angeordnet werden, so dass ein zylindrisches Darstellung entsteht.

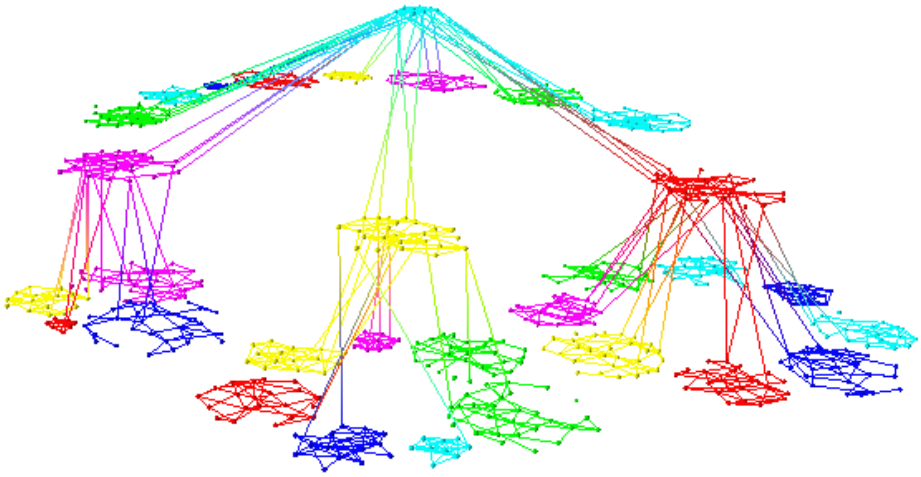

Abbildung 2.10: ConeTree-Visualisierung einer Hierarchie (Quelle: [9])

Eine Kombination aus *TreeMaps* und *Cone Trees* bilden hyperbolische Ansichten. Dabei werden Strukturen in einem dreidimensionalem euklidischen Raum dargestellt.

2.3.2.3 Ungerichtete Graphen

Graphen stellen ein vielseitiges und im Allgemeinen bei dem Benutzer bekanntes Visualisierungskonzept dar. Beispiele für das Auftreten von Graphen zur Visualisierung sind u.a. der Verlauf von Aktienkursen, historische Zeitlinien oder, als Spezialform des Graphen, hierarchische Baumdarstellungen wie etwa Verzeichnisstrukturen. Als Spezialfälle neben

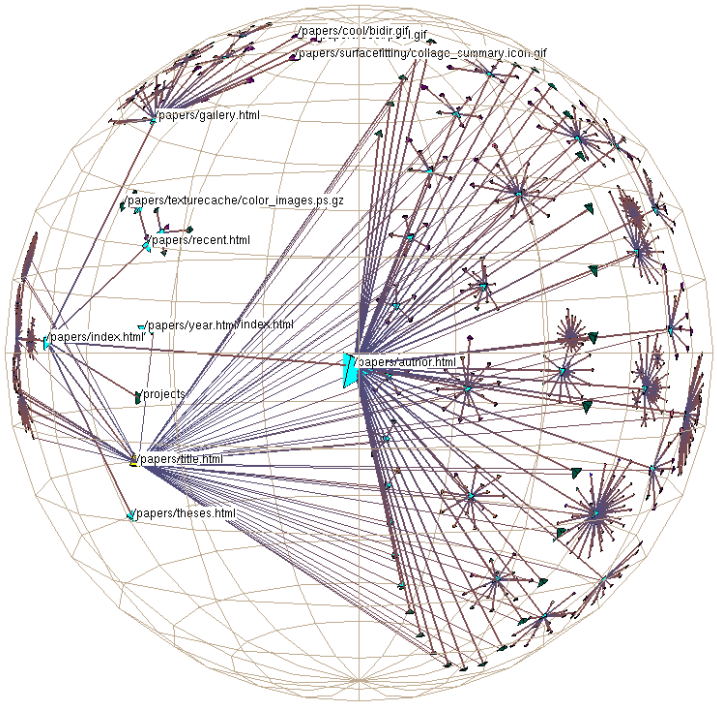

Abbildung 2.11: Hyperbolische Hierarchiedarstellung (Quelle: [17])

Bäumen sind ebenfalls zyklische Graphen oder Netze zu nennen, bei denen Schleifen, Mehrfachverbindungen und Mehrfachkanten es möglich machen, komplexere Strukturen abzubilden.

Für die Visualisierung mittels Graphen gelten (wie für andere Visualisierungstechniken auch) Mindestanforderungen an Schnelligkeit und Übersichtlichkeit. Das Problem bei der Darstellung eines Datenbestandes als Graph besteht darin, dass verschiedene Kriterien beachtet werden müssen, um dieses Mindestmaß an Übersichtlichkeit zu gewährleisten. Dazu zählen die Symmetrie eines Graphen, die gleichmäßige Verteilung von Knoten sowie einheitliche Kantenlängen und möglichst wenige Überschneidungen von Kanten.

Um diese Kriterien zu erfüllen, wurden in der Vergangenheit verschiedene Algorithmen entwickelt. Diese erfüllen jedoch jeweils nur eine Auswahl der Anforderungen, um Ansprüchen an die Geschwindigkeit und Einfachheit gerecht werden zu können. Folgend sollen einige dieser Algorithmen vorgestellt werden.

Spring-Embedder-Modell Das von Eades 1984 vorgestellte Spring-Embedder-Modell ist ein populäres Verfahren zum Erzeugen von ungerichteten Graphen und bietet bei der Darstellung die Vereinheitlichung der Kantenlängen und erzeugt soweit möglich einen symmetrischen Graphen. Der Algorithmus wird im Bereich der Visualisierung wegen seiner Einfachheit häufig benutzt.

Beim Spring-Embedder-Modell werden Knoten als Ringe angesehen, die mittels Federn miteinander verbunden sind. Jeder Knoten bzw. Ring ist, wie in Abbildung 2.12 zu sehen, gekennzeichnet durch zwei Arten von Kräften, die auf ihn wirken: Anziehungskräfte (f_a) und Abstoßungskräfte (f_r).

Abbildung 2.12: Anziehungs- und Abstoßungskräfte eines Knotens

Die Erzeugung des Graphen ist abgeschlossen, wenn die Spannungen (erzeugt durch die Wirkung der Kräfte zwischen den Ringen) im System minimal sind. Dabei berechnen sich die auftretenden Kräfte nach folgenden Formeln:

$$f_a(d) = k_a * log(d)$$

$$f_r(d) = k_r/d^2$$

Hierbei ist d die aktuelle Distanz zwischen den Knoten, k_a und k_r sind Konstanten, welche eine Konfiguration der Darstellung ermöglichen. Die Kräfte werden für jeden Knoten im Graphen berechnet und anschließend so bewegt, dass sich die Spannung im System reduziert. Dieser Vorgang wird iterativ fortgeführt, bis ein als optimal eingestufter Zustand herbeigeführt ist.

Lokales Minimum Der von Kamada und Kawai vorgestellte Algorithmus des lokalen Minimums ist eine Weiterentwicklung des Spring-Embedder-Modells. Während bei der Methode von Eades die Anziehungskräfte lediglich zwischen einem Knoten und seinen Nachbarn berechnet werden, werden bei diesem Verfahren zusätzlich die Abstoßungskräfte zwischen dem Knoten und allen anderen Knoten im Graphen berechnet. Daraus folgt, das bei steigender Knotenanzahl der Rechenaufwand quadratisch ansteigt ($\Theta(|V|^2)$).

Das Verfahren des lokalen Minimums berechnet hingegen zu einem gegebenen Graphen den Grad der Unausgewogenheit E. Das Modell impliziert, dass bei minimalem E die beste Darstellungsweise für den Graphen gefunden wurde. Dabei erzeugt das Verfahren einen soweit wie möglich symmetrischen Graphen mit möglichst wenigen Kantenüberschneidungen. Adaptiert wurde das Verfahren von Kamada und Kawai 1994 durch Kumar und Fowler, die das Modell für die Benutzung im dreidimensionalen Raum erweiterten.

Force Directed Eine signifikante Verbesserung des Spring-Embedder-Modell stellt der 1991 von Fruchterman und Reingold entwickelte Force Directed-Algorithmus dar. Er

ermöglicht es, Graphen zu erzeugen, die mehrere Übersichtlichkeitskriteren erfüllen, wie einheitliche Kantenlängen, wenige Kantenüberschneidungen und die gleichmäßige Verteilung von Knoten im Graphen.

Wie beim Spring-Embedder-Modell werden hier Anziehungskräfte zwischen einem Knoten und seinen Nachbarn sowie Abstoßungskräfte zwischen einem Knoten und allen anderen im Graphen berechnet. Jedoch wurden beim Verfahren von Fruchterman und Reingold die Formeln zur Berechnung dieser Kräfte verändert. Die Anziehungs- (f_a) bzw. Abstoßungskräfte (f_r) zwischen Knoten mit einer Distanz d berechnen sich hier folgend:

$$f_a(d) = d^2/k$$
$$f_r(d) = -d^2/k$$

Der Algorithmus berechnet anhand dieser Formeln alle Kräfte und bewegt danach alle Knoten gleichzeitig. Außerdem benutzt er einen „Temperaturfaktor“, welcher (in Anlehnung an das Simulated Annealing-Verfahren) reduziert wird, wenn das System einen spannungsärmeren Zustand erreicht hat.

Simulated Annealing Das Simulated Annealing-Verfahren, welches übersetzt werden kann mit „simulierter Abkühlung“, basiert ebenfalls auf des Spring-Embedder-Modell. Es handelt sich hierbei um ein Optimierungsverfahren, mit dem in einem System das globale Minimum gefunden werden kann, auch wenn ein lokales Minimum entdeckt worden ist und jeder neu berechnete Zustand eine Verschlechterung der Gesamtsituation darstellt. Abbildung 2.13 verdeutlicht das Problem.

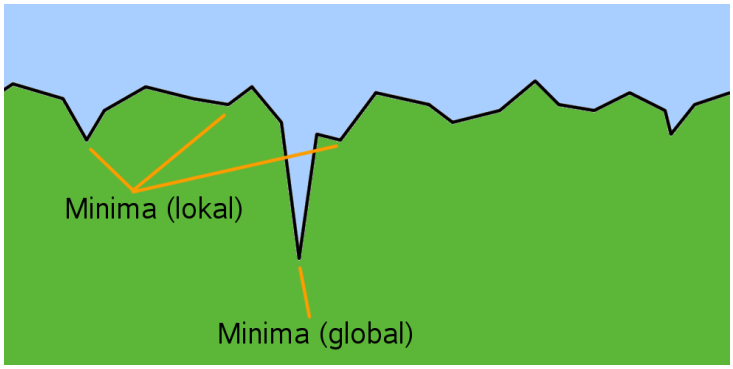

Abbildung 2.13: Lokale Minima und globales Minimum

Andere Algorithmen, wie das Spring-Embedder-Modell oder das Prinzip der Findung des Lokalen Minimum, basieren darauf, den errechneten Zustand stetig zu verbessern. Finden sie ein lokales Minimum, können sie dieses, auch wenn ein besserer Zustand existiert,

nicht mehr verlassen, da jede Konfiguration, die sie zur Annäherung an das globale Minimum testen, eine Erhöhung der Spannung des Systems bewirkt. Das tatsächliche globale Minimum wird auf diese Weise nicht gefunden.

Ein großer Nachteil des *Simulated Annealing*-Verfahrens ist die hohe Komplexität des Algorithmus und die damit verbundene hohe Rechnenzeit. Daher wird es für nicht interaktive Systeme wie etwa der Berechnung von optimalen Elementanordung auf Platinen in der Halbleiterindustrie angewandt.

2.3.3 Navigation in virtuellen Umgebungen

Neben der übersichtlichen Darstellung von Datensammlungen in einer virtuellen Umgebung ist es ebenso wichtig, dem Benutzer die Navigation in dieser Darstellung und somit im digitalen Archiv so einfach wie möglich zu gestalten. In Datenbeständen zu navigieren ist dann erforderlich, wenn die gesamte Datenmenge nicht mehr übersichtlich dargestellt werden kann und dann z.B. mittels *Clustering* (2.3.2.1) unterteilt wird.

In diesem Abschnitt sollen die verschiedenen Aspekte betrachtet werden, die für die Navigation von Bedeutung sind. Dazu zählt zuerst die Benutzerschnittstelle, mit der der Benutzer direkt auf die Anwendung einwirken kann. Weiterhin wird analysiert, wie sich der Benutzer in virtuellen Umgebungen orientiert. Abschließend sollen verschiedene Metaphern vorgestellt werden, die benutzt werden können, um diesen Orientierungsprozess zu verkürzen und somit den Zugang zum digitalen Archiv erleichtern.

2.3.3.1 Benutzerschnittstelle

Die Benutzerschnittstelle erfüllt bei einer Anwendung zwei Arten von Aufgaben. Einerseits stellt sie den Zugang zur Anwendung her. Das bedeutet, dass mittels verschiedener Visualisierungstechniken Daten, die die Anwendung verarbeitet, für den Benutzer verfügbar gemacht werden. Andererseits besitzt sie auch eine Steuerungsfunktion, mit deren Hilfe der Benutzer die Anwendung beeinflussen kann.

In diesem Abschnitt soll die Kombination dieser beiden Aufgabenbereiche veranschaulicht werden und zwar die Steuerung der Visualisierung. Einfachstes Beispiel dafür ist die Betrachtung einer umfangreichen Hierarchie wie z.B. einer Verzeichnisstruktur. Es ist nicht möglich, sämtliche Daten unter Erhalt ihrer hierarchischen Darstellung simultan dem Benutzer zu präsentieren. Deswegen wird die Hierarchie nur ausschnittsweise dargestellt, und der Benutzer kann über Steuerungsfunktionen der Benutzerschnittstelle wie z.B. durch Klicken mit der Maus oder Tastaur-Eingaben den Ausschnitt verändern.

Dieses Prinzip soll weiterhin am Beispiel von *Zooming User Interfaces* geschildert werden. Abbildung 2.14 zeigt das Programm *Pad++* ([6]), welches dieses Navigations-Paradigma implementiert. Dabei werden mittels *Clustering* Dokumentengruppen identifiziert; diese werden dann in einer Hierarchie geordnet. Bei der Darstellung werden zuerst die in der Hierarchie höchsten Knoten angezeigt (Abbildung 2.14 - links). Zusätzlich sind in der Darstellung eines Clusters mit dem zugehörigen Sammelbegriff Anhaltspunkte dafür gegeben, dass weitere Unterknoten in diesem Cluster zu finden sind. Am Beispiel von *Pad++* geschieht dies durch kleinere geometrische Objekte, die in der übergeordneten Form angezeigt werden.

Wählt der Benutzer einen Knoten aus, wird an diesen herangezoomt (Abbildung 2.14 - mitte). Das Prinzip der Darstellung enthaltener Cluster setzt sich anschließend fort (Abbildung 2.14 - rechts) bis zur letzten Ebene der Hierarchie.

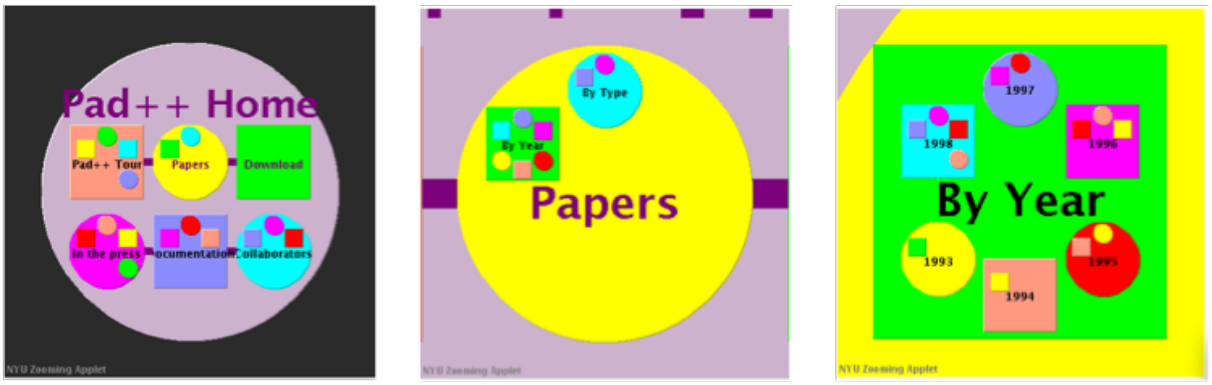

Abbildung 2.14: Pad++ - Zooming User Interface (Quelle: [6])

2.3.3.2 Orientierung

Wie im oberen Abschnitt beschrieben, dient die Benutzerschnittstelle der Steuerung der Anwendung und der Visualisierung. Mit ihr kann der Benutzer sich in der Visualisierung bewegen. Dabei ist es wichtig, dass er, während er sich in der visualisierten Datenstruktur bewegt, die Orientierung nicht verliert. Bei der Navigation in Hierarchien spielt dies eine untergeordnete Rolle, da der Benutzer weiß, dass jedes Element (bis auf das Wurzelelement) genau einen übergeordneten Knoten besitzt, zu dem er zurückkehren kann. Doch selbst in solchen einfachen Datenstrukturen zeigen Untersuchungen, dass der Benutzer, wenn er die Orientierung verloren hat, stets direkt zum Wurzelelement zurückkehrt und von vorn mit der Erschließung der Hierarchie beginnt ([Che99]).

Muss nun durch komplexere Strukturen als Hierarchien navigiert werden, die nicht im zwei- sondern im dreidimensionalen Raum dargestellt sind, fällt die Orientierung zuneh-

mend schwerer und somit wird der Zugang zur gesuchten Information ebenfalls erschwert oder im ungünstigsten Fall unmöglich.

Abbildung 2.15 zeigt eine schematische Darstellung der Interaktion eines Menschen mit einer Anwendung. Dort wird dargestellt, wie der Mensch die Visualisierung der Anwendung steuert und das Resultat im Kurzzeitgedächtnis verarbeitet. In dieser Schleife aus Aktion des Benutzers und Reaktion der Anwendung bildet sich im Kurzzeitgedächtnis eine kognitive Karte (*Cognitive Map*), mit der sich der Benutzer zu orientieren versucht.

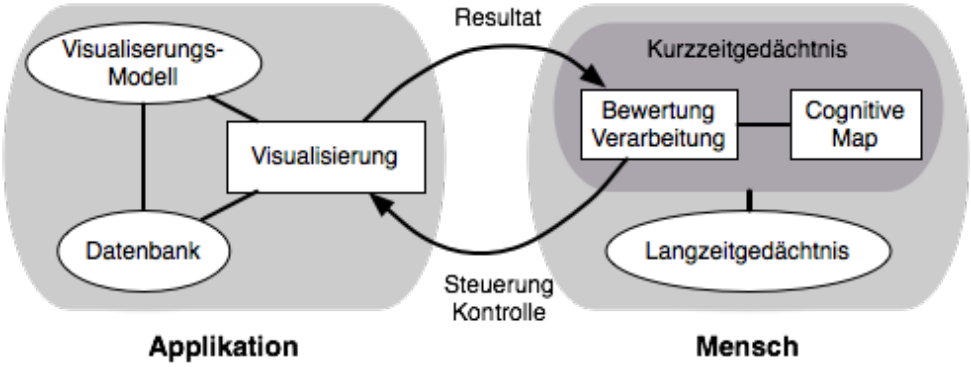

Abbildung 2.15: Navigations-Kontroll-Schleife (Quelle: nach [War00])

Dieses „Navigations-Wissen“ bildet sich in drei Phasen. Je mehr Informationen über die virtuelle Umgebung der Benutzer ansammelt, desto besser kann er sich in ihr bewegen.

Zu Beginn identifiziert der Mensch in einer unbekannten Umgebung Orientierungspunkte (*Landmarks*), wie ungewöhnliche und herausragende Strukturen die bspw. farblich abgesetzt sind, zu denen er zurückkehrt, wenn er die Orientierung verloren hat. Außerdem werden neue Objekte in Relation zu diesen Fixpunkten in der kognitiven Karte angeordnet.

Nach einer intensiveren Beschäftigung mit der virtuellen Umgebung bildet sich Wissen über Wege zwischen den Orientierungspunkten heraus (*Route Knowledge*). Damit ist der Benutzer in der Lage, sich von einem Punkt A zu einem Punkt B zu bewegen; jedoch ist nicht festgelegt, dass es sich bei diesem Weg um den bestmöglichen handelt.

Das Finden des bestmöglichen Weges zwischen zwei Punkten ist dem Benutzer erst nach weiterer Auseinandersetzung mit der virtuellen Welt möglich. Er gewinnt dann Wissen über die gesamte Umgebung (*Survey Knowledge*) und ist damit in der Lage, sich frei zu bewegen, ohne die Orientierung zu verlieren.

2.3.3.3 Metaphern

Die Dauer der Orientierungsphase variiert von Benutzer zu Benutzer. Während dieser Zeit trifft er unter Umständen falsche Entscheidungen und verliert die Orientierung, und die Frustration gegenüber der Anwendung steigt. Mit Hilfe von Metaphern kann die Orientierungsphase verkürzt werden, da dem Benutzer bekannte Paradigmen auf die virtuelle Umgebung übertragen werden und so nicht erst erlernt werden müssen.

„Reale Welt"-Metaphern Bei dieser einfachen Art werden Zusammenhänge in der realen Welt auf die virtuelle übertragen. Am Bespiel eines Ordnungssystems lässt sich das wie folgt verdeutlichen: Eine hierarchische Datenstruktur wird in der Visualisierung bspw. als Aktenschrank mit Schubladen, Aktenmappen und darin enthaltenen Dokumenten visualisiert. Unter der Annahme, das der Benutzer bereit mit dem Paradigma Aktenschrank vertraut ist, findet er einfachen Zugang zu der Visualisierung.

Räumliche Metaphern Ware nennt vier Metaphern für den Zugang zu dreidimensionalen Umgebungen, die hier kurz vorgestellt werden sollen ([War00], S. 346):

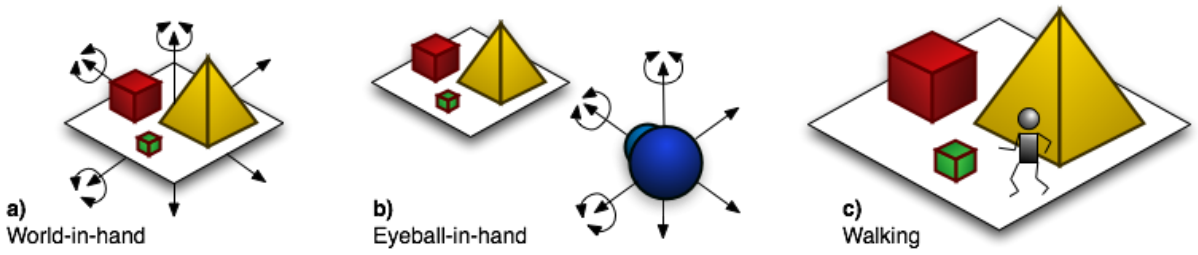

Abbildung 2.16: Räumliche Metaphern (Quelle: nach [War00])

World-in-hand Bei dieser Metapher (Abbildung 2.16.a) kann der Benutzer die virtuelle Welt als Ganzes bewegen. Diese Darstellung bietet sich für kompakte Objekte an, die als Ganzes auf dem Bildschirm angezeigt werden können.

Eyeball-in-hand Hiebei kann der Benutzer sein Blickpunkt auf die Szene verändern, ähnlich der Positionierung einer Kamera in einer Landschaft. Obwohl dieses Verfahren verständlich und greifbar erscheint, kommt es leicht zu Verwirrungen, wenn die Szene das Sichtfeld des Benutzers verlässt und somit jegliche Orientierung verloren geht (Abbildung 2.16.b).

Walking Anders als bei den beiden vorherigen Metaphern betrachtet der Benutzer hier die Welt nicht von einem entfernten Punkt aus, sondern bewegt sich in der virtuellen Umgebung (Abbildung 2.16.c).

Flying Ähnlich der *Walking*-Metapher bewegt sich der Benutzer direkt in der virtuellen Welt, jedoch ohne Restriktionen bzgl. der Höhe seines Standpunktes. So ist es ihm möglich, bspw. durch Häuserschluchten zu gehen oder, um eine Übersicht über das Terrain zu gewinnen, die gesamte Szenerie aus der Luft zu betrachten. Diese Darstellungsart kombiniert somit Aspekte aus *Walking*- und *Eyeball-in-hand*-Metapher.

Kapitel 3

Digitale Archive und multimediale Datensammlungen

In diesem Kapitel sollen verschiedene bereits existierende Anwendungen vorgestellt werden, die Zugang zu einem digitalen Archiv ermöglichen. Diese sollen dann unter Berücksichtigung der Zielsetzung dieser Arbeit bewertet werden, um bereits vorhandene Programmbibliotheken und/oder Algorithmen zu finden und für die Entwicklung des Prototypen zu benutzen oder Verfahren zu identifizieren, welche dessen Benutzung vereinfachen können.

Anschließend werden verschiedene zur Zeit häufig verwendete Programmiersprachen und Programmierumgebungen vorgestellt, mit denen Anwendungen zum Zugang zu digitalen Archiven entwickelt werden können. Diese Gegenüberstellung ist für die Auswahl von Programmiersprache und Framework bei der Implementierung des Prototypen von großer Bedeutung.

3.1 Ausgewählte Anwendungen

In diesem Abschnitt sollen exemplarisch vier Anwendungen, die in Lage sind, dem Benutzer Zugang zu großen Datensammlungen zu ermöglichen oder mittels Visualisierungstechniken eine ansprechende Aufbereitung der zu Grunde liegenden Daten bereitstellen, betrachtet werden. Um ggf. Aspekte der vorgestellten Lösung zu identifizieren, die in dieser Arbeit verwendbar sind, sollen zuerst Beurteilungskriterien ermittelt werden, anhand derer es möglich ist, die einzelnen Komponenten der Anwendungen zu beurteilen.

3.1.1 Beurteilungskriterien

Hinsichtlich folgender Gesichtspunkte sollen die vorgestellten Anwendungen betrachtet werden:

Datenquellen Unter diesem Punkt soll beurteilt werden, mit Hilfe welcher Daten die Anwendung arbeitet. Auch die Möglichkeit, Daten aus anderen Datenquellen, wie den im HardMut-Projekt vorhandenem Webservice, zu benutzen soll untersucht werden, da, sollte die Anwendung auf eine grundlegend verschiedene Art der Datenbeschaffung beschränkt sein, eine Verwendung dieser Komponenten im Prototypen nicht möglich ist.

Laufzeitumgebung Hier wird bewertet, in welchen verschiedenen Laufzeitumgebungen die Anwendung funktionsfähig ist, da es für den zu entwickelnden Prototypen von Bedeutung ist, auf möglichst vielen Geräten zu arbeiten.

Visualisierung Die Visualisierung von Daten spielt im Prototypen eine große Rolle, deshalb sollen unter diesem Aspekt besonders die verschiedenen Methoden, die zur Anzeige der Datenbestände dienen, betrachtet werden, um so Algorithmen und Verfahren zu identifizieren, die im Prototypen ebenfalls verwendbar sind.

Benutzerschnittstelle Bei diesem Punkt soll die Gestaltung der Benutzerschnittstelle dahingehend beurteilt werden, ob diese einfach zu benutzen ist und trotzdem alle Steuerungs- und Navigationsmöglichkeiten bereitstellt, die der Benutzer für diese Anwendung benötigt. Außerdem wird darauf geachtet werden, ob diese Bedienelemente so angeordnet bzw. dargestellt sind, dass Sinn und Zweck sich dem Benutzer direkt erschließen (vgl. 2.3.1).

Rechtliche Beschränkungen und Verfügbarkeit Sollen einzelne Bestandteile der vorgestellten Applikationen wiederverwendet werden, so ist unbedingt zu berücksichtigen, ob rechtliche Restriktionen die Verfügbarkeit und Verwendung von Programmquellen unter bestimmte Bedingungen stellen oder ganz verhindern.

3.1.2 Commetrix

Bei Commetrix handelt es sich um eine Applikation zur Analyse und Visualisierung von Kommunikations-Netzwerken. Dabei wird der Fokus auf die Darstellung der Entwicklung dieser Netzwerke gelegt. Commetrix ermöglicht es damit, bspw. Personen in einem sozialen Netzwerk zu identifizieren, die eine Vermittler-Rolle übernehmen und so einzelne Cluster miteinander verbinden [1].

Datenquellen Commetrix benutzt für die Datensammlung so genannte *Konnektoren*. Mit Hilfe dieser Konnektoren ist es möglich, Kommunikationsbeziehungen aus NNTP-Newsgroups, HTML-Diskussionen und aus E-Mail-Verkehr zu extrahieren. Aus diesen Daten werden Knoten im Kommunikations-Netzwerk (Personen) und deren Beziehungen untereinander identifiziert und unter Erhaltung zeitlicher Aspekte vorverarbeitet und anschließend in einer MySQL-Datenbank gespeichert.

Nachdem diese Datengewinnung abgeschlossen ist, arbeitet die Anwendung ausschließlich mit den Daten in der MySQL-Datenbank. Es ist nicht möglich, eine andere Art von Datenquelle wie z.B. einen Webservice (wie der, der im HardMut-Projekt benutzt wird) zu benutzen. Auch ist der Diskursbereich von Commetrix (Visualisierung sozialer Netzwerke) zu verschieden vom Anwendungsgebiet des hier zu entwickelnden Prototypen, als dass für die Datengewinnung Algorithmen übernommen werden könnten.

Jedoch ist der Ansatz, verschiedene Konnektoren zu benutzen, um Daten aus unterschiedlichen Quellen zu extrahieren, ein interessanter und für den Entwurf des Prototypen zu berücksichtigender Ansatz.

Laufzeitumgebung Für die Programmierung von Commetrix wurde Java verwendet. Somit ist es möglich, die Anwendung mit allen Betriebssystemen zu benutzen, die eine *Java Runtime Environment* bereitstellen. In der derzeitigen Version von Commetrix ist es nicht möglich die Anwendung als Applet in einem Browser auszuführen; eine Installation ist zwingend erforderlich.

Visualisierung Das in der Datensammlungs-Phase extrahierte soziale Netz wird als ungerichteter Graph wahlweise im zwei- oder drei-dimensionalen Raum dargestellt. Dabei werden Personen im Netzwerk als Knoten und Verbindungen zwischen den Personen (in diesem Fall Kommunikation) durch Linien dargestellt. Die Relevanz einer Person wird durch die Größe des Knoten visualisiert. Zusätzlich ist es möglich, Cluster (welche Commetrix im Netzwerk automatisch identifizieren kann) unterschiedlich farblich anzuzeigen (Abbildung 3.1).

Um den komplexen Graphen unter Berücksichtigung von bereits erläuterten Übersichtlichkeitskriterien darzustellen, wird eine Implementierung des *Force Directed*-Algorithmus von Fruchterman und Reingold verwendet (vgl. 2.3.2.3).

Dem Benutzer ist es ermöglicht worden, die Darstellung mittels Zoom- und Rotations-Funktionen zu manipulieren, um für ihn relevante Aspekte des Graphen zu fokussieren.

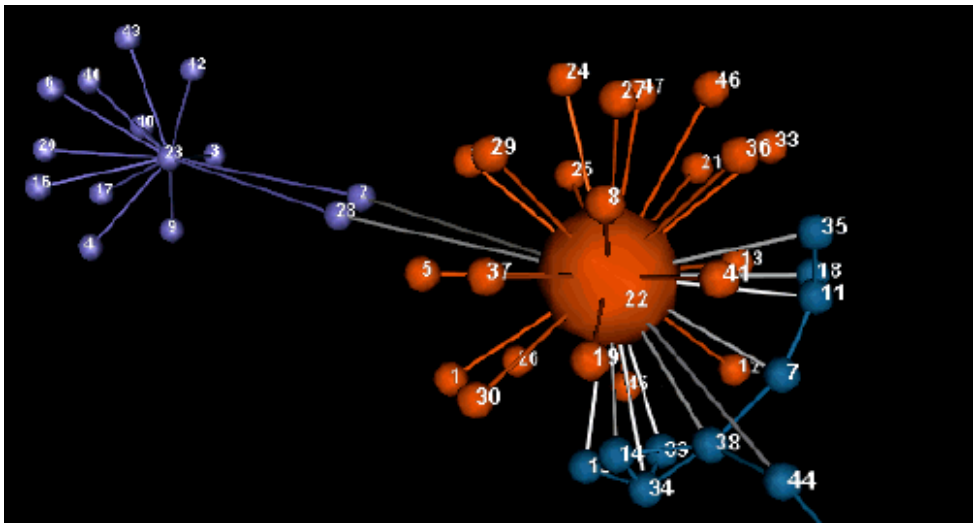

Abbildung 3.1: 3D Visualisierung eines Graphen mit drei Clustern

Benutzerschnittstelle Bedingt durch den Einsatz von Commetrix als Analyse-Software zur Auswertung und Veranschaulichung unterschiedlicher Aspekte in sozialen Netzwerken, sind die Konfigurationsmöglichkeiten der Anwendung sehr komplex. Diese Komplexität spiegelt sich in der Benutzeroberfläche wieder.

Die vielfältigen Möglichkeiten bei der Konfiguration von bspw. Datenquellen und Graphen-Visualisierung machen es erforderlich, dass der Benutzer sich einige Zeit mit der Schnittstelle auseinander setzen muss, um sie vollständig bedienen zu können. Auch ist ein bestimmtes Maß an Vorwissen über Zweck und Arbeitsweise der Anwendung erforderlich, um alle Funktionen zu erschließen.

Dies ist im Rahmen der Benutzung als Analyse-Software durchaus berechtigt, zeigt jedoch auch, dass die Benutzerschnittstelle - soll das Programm von einem breiten Publikum benutzt werden - deutlich einfacher gestaltet werden muss (Abbildung 3.2).

Rechtliche Beschränkungen Im Rahmen der Anfertigung dieser Arbeit besteht Zugriff auf die Quellen der Commetrix-Anwendung in der Version 1.11. Dieser kann als Vorlage für die Implementierung ausgewählter Algorithmen (Force Directed-Methode zur Optimierung von Graphendarstellungen) benutzt werden.

3.1.3 Viewzi Photo Tag Cloud

Bei Viewzi handlet es sich um eine Sammlung von Anwendungen, die von der Firma Viewzi Inc. entwickelt wurden, um die Suche nach Informationen einfacher und für den Benutzer interessanter zu gestalten [3].

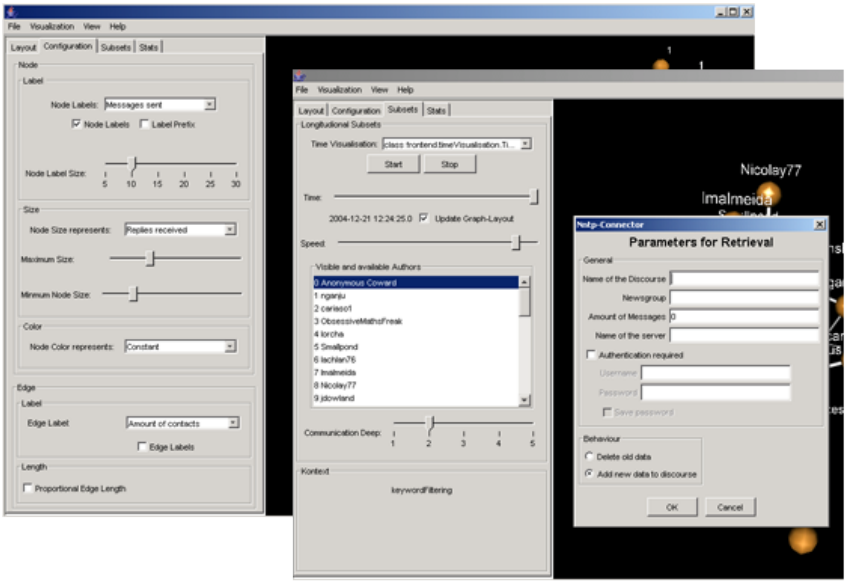

Abbildung 3.2: Komplexe Bedienelemete der Commetrix-Applikation

Anstatt eine einfache Liste von Suchergebnisse (im Stil von Suchmaschinen wie bspw. Google [2]) zu einem Suchwort auszugeben, sollen ansprechende *Views* das Ergebnis aufbereiten und dem Ergebnis der Suche angepasst (bspw. Ergebnisse einer Bildersuche als Album dargestellt) präsentieren.

Hier soll exemplarisch die Visualisierungsmethode „Photo Tag Cloud“ vorgestellt werden.

Datenquellen Als Datenquelle benutzt die Anwendung den Flickr Fotodatenbank-Webservice ([4]), um nach Bilder und damit verknüpften Schlagworten und Beschreibungen zu suchen. Eine Möglichkeit zur Benutzung von anderen Datenquellen wird nicht unterstützt.

Algorithmen, die für die Auswahl von Bildern und Schlagworten aus der Datenbank verwendet werden, konnten nicht nachvollzogen und dem entsprechend keine Erkenntnisse für den Entwurf des Prototypen gewonnen werden, ebenfalls ist es unter den genannten Umständen nicht möglich, eine Modifizierung der Anwendung anzustreben, um daraus einen Prototypen zu entwickeln.

Trotzdem sollen die anderen Aspekte der Anwendung beurteilt werden, um Konzepte zu identifizieren die in den Prototypen übernommen werden können.

Laufzeitumgebung Die Viewzi Photo Tag Cloud ist eine Flash-Anwendung und wird mittels Flash 9-Plugin in unterstützten Browsern ausgeführt. Daraus folgt, dass es möglich ist, das Programm auf allen Rechnern und Betriebssystemen auszuführen, die einen ent-

sprechenden Browser mit Flash-Plugin installiert haben.

Da für die Wiedergabe ein Plugin benötigt wird, der den vollen Umfang der Flash 9-API bereitstellt, ist eine Benutzung auf mobilen Geräten, die nur eine Umgebung mit eingeschränktem Funktionsumfang besitzen, wie z.B. Handys oder PDAs, nicht möglich.

Visualisierung Zur Anzeige der Suchergebnisse wird ein dreidimensionaler Raum verwendet, in dem Suchworte und Bilder dargestellt werden. Dabei werden zu einem Suchwort relevante Bilder und verwandte Begriffe um dieses gruppiert dargestellt (Abbildung 3.3)

Besteht zwischen zwei Suchbegriffe eine Beziehung, wird diese mittels einer Linie visualisiert. Werden Bilder und andere Schlagworte, die beide Suchworte betreffen gefunden, werden diese um den Mittelpunkt der Linie gruppiert.

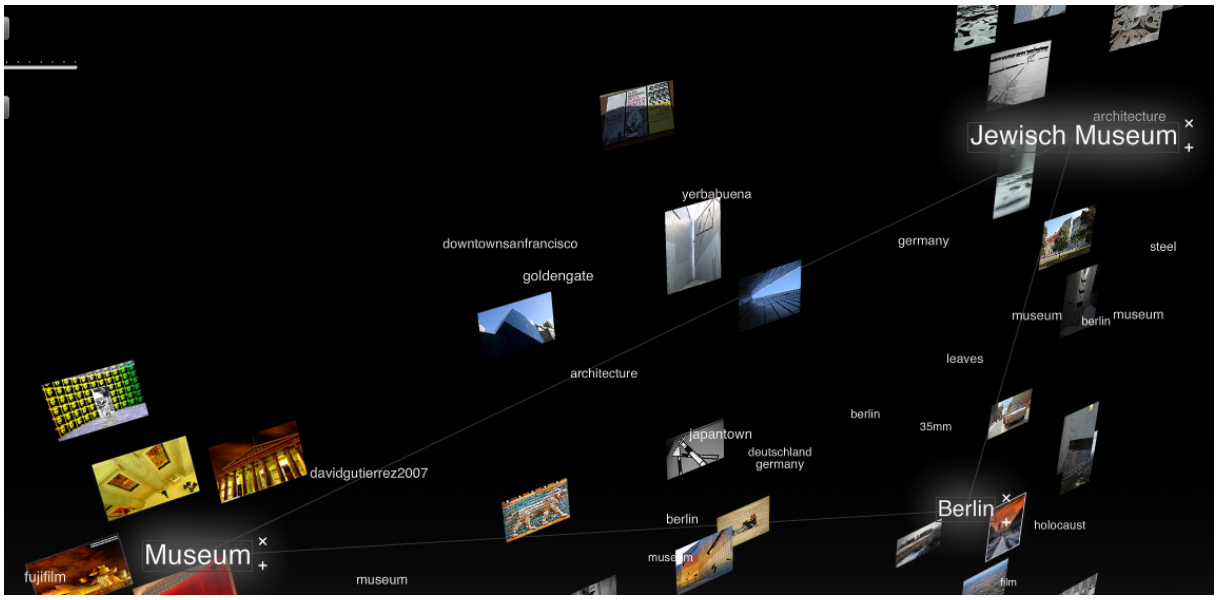

Abbildung 3.3: Drei Suchbegriffe mit Verbindungen, Suchergebnissen und Steuerelementen visualisiert in der Viewzi Photo Tag Cloud

Diese Darstellungsweise ermöglicht es dem Benutzer, direkt Zusammenhänge zwischen Suchwort und Ergebnis zu erkennen. Zusammenhänge zwischen den Suchbegriffen werden durch Linien verdeutlicht. Der explorative Aspekt, der auch im Prototypen umgesetzt werden soll (Vgl. 1.3), wird hier dadurch unterstützt, dass weitere artverwandte Begriffe und somit potenzielle Suchbegriffe im unmittelbaren Umfeld des eigentlichen Suchwortes angezeigt werden.

Benutzerschnittstelle Die Benutzerschnittstelle ist bei der Viewzi Photo Tag Cloud sehr spartanisch und funktional gestaltet.

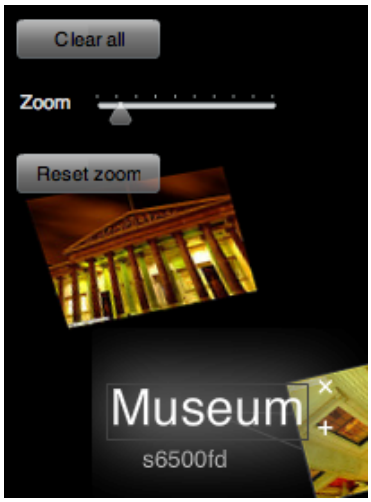

Abbildung 3.4: Viewzi Kontrollelemente

Dauerhaft im Blickfeld des Benutzers sichtbare Bedienelemente befinden sich im rechten oberen Bereich der Anwendung. Abbildung 3.4 zeigt diese statischen Elemente, mit denen der Benutzer einerseits alle Elemente die er im Sichtvolumen angesammelt hat entfernen (*Clear all*) und andererseits den Zoomfaktor per Schieberegler einstellen kann. Wird die Schaltfläche „*Reset zoom*“ betätigt, wird der Zoomfaktor auf den Standardwert zurückgesetzt.

Zusätzlich zu diesem statischen Kontrollelementen verfügt jeder Suchbegriff über zwei Schaltflächen, mit denen entweder der Suchbegriff und mit ihm verbundene Elemente aus der Darstellung entfernt werden („x“-Symbol) oder weitere Bilder und verwandte Suchbegriffe abgerufen werden („+“-Symbol). Suchworte selbst können per Drag and Drop im Sichtvolumen bewegt werden.

Diese Gestaltung der Benutzerschnittstelle, bei der alle benötigten Steuerelemente entweder dauerhaft sichtbar sind, oder Funktionen zur Objektmanipulation sich direkt am Objekt befinden und sich somit deren Zweck ebenfalls direkt erschließen lässt, erfüllt die Kriterien von Gibson und Pirolli (Vgl. 2.3.1) zur Vereinfachung des Informationszugangs und sollte im Prototypen berücksichtigt werden.

Rechtliche Beschränkungen Bei der Applikation handelt es sich um eine kommerziell vermarktete Anwendung, von der die Quelltexte von den Rechte-Inhabern nicht veröffentlicht wurden. Es ist also nicht möglich, einzelne algorithmische Bestandteile in den Prototypen zu übernehmen oder mittels Modifikation der Applikation den Prototypen zu erstellen.

Jedoch sollen die guten Ansätze in der Visualisierung und Oberflächengestaltung in die Gestaltung des Prototypen einfließen.

3.1.4 MoMA Online Archiv

Da es sich bei dem zu entwickelnden Prototypen um eine Anwendung handelt, die im Umfeld des Jüdischen Museums eingesetzt werden wird, sollen hier auch zwei Online-Anwendungen vorgestellt werden, mit denen Museen Zugang zu deren digitalen Archiven bieten.

Begonnen werden soll hier mit dem Online-Archiv des *Museum of Modern Art* (MoMA).

Hierbei handelt es sich um eine Sammlung von Audio- und Videomaterial sowie verschiedenen Flash-Anwendungen [8].

Datenquellen Da es sich bei dem Zugang zum Archiv des MoMA um eine dynamisch generierte HTML-Seite handelt, kann vermutet werden, dass als Datenquelle ein Content-Management-System (CMS) verwendet wird und der Inhalt aus der angeschlossenen Datenbank generiert wird.

Die genaue Funktionsweise der Technologien zum Abruf von Daten aus dem CMS des Museums kann nicht genau nachvollzogen werden, womit deren Verwendung im Prototypen ausgeschlossen ist.

Laufzeitumgebung Durch die Verwendung von HTML-Seiten, sowie darin eingebetteten Multimediadaten kann nur mittels eines Browsers, jedoch unabhängig von Betriebssystem, der Zugang zum Archiv des Museums genutzt werden. Auch eine Verwendung von mobilen Geräten (die einen Browser und Internetzugang besitzen) ist möglich.

Visualisierung Im Gegensatz zu den Anwendungen von Commetrix und Viewzi, ist die Präsentation der Medien zu den Exponaten des MoMA eher funktional. Video- und Audio-Dateien werden in einem kleinen Fenster abgespielt, und dort sind auch die Steuerelemente für diese Medien dargestellt. Neben diesem Bereich werden zusätzliche Textinformationen zu den Exponaten angezeigt (Abbildung 3.5).

Abbildung 3.5: Darstellung von Audio- und Videomaterial sowie Textinformationen zu Exponaten

Außerdem ist eine Wiedergabeliste mit ausgewählten Medien im rechten Bereich angesiedelt. Hier kann der Benutzer entweder selbst eine Liste erstellen, in der er von ihm präferierte Medien einfügt, oder vom Museum vorbereitete Listen benutzt, die für einen bestimmten Diskursbereich eine Exponatsübersicht zu geben versuchen.

Benutzerschnittstelle Um ein Medium auszuwählen und anzeigen zu lassen, muss der Benutzer zuerst eine Kategorie auswählen (Abbildung 3.6, links). So wird der große Datenbestand in kleinere Einheiten unterteilt. In manchen Kategorien finden noch weitere Unterteilungen statt (bspw. Videos ⟼ Current Exhibitions/Past Exhibitions etc.).

Im rechten Bereich sind sind verschiedene Elemente zu sehen, die zur ausgewählten Kategorie gehören. Mittels der Schaltflächen in oberen rechten Bereich dieser Darstellung können weitere Suchergebniss angezeigt werden.

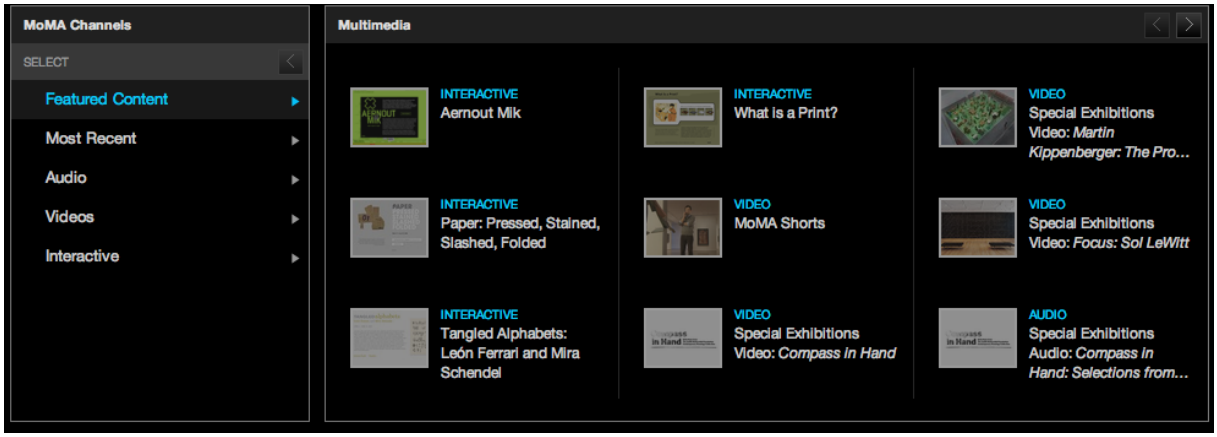

Abbildung 3.6: Auswahl von Medien des Museum of Modern Art

Rechtliche Beschränkungen Laut Impressum des MoMA sind alle Programme, Medien und Inhalte Eigentum des Museums und dürfen nicht verwendet werden.

Da es sich bei diesem Informationsportal lediglich um eine HTML-Seite mit dynamisch eingefügten Inhalten, welche aus einem CMS unter nicht vollständig nachzuvollziehenden Parametern ausgewählt werden, handelt, konnten hier keine wichtigen Aspekte identifiziert werden die bei der Erstellung des Prototypen berücksichtigt werden sollten.

3.1.5 Guggenheim Bilbao Online Rundgang

Das Guggenheim Museum im spanischen Bilbao bietet auf seiner Webpräsenz einen virtuellen Rundgang durch das Museum und die Besichtigung ausgewählter Exponate an [7].

In dieser Abschnitt soll dieser Rundgang vorgestellt werden, da er auf innovative Weise Museumsrundgang und Erkundung eines (wenn in diesem Fall auch sehr kleinen) digitalen Archivs verknüpft.

Datenquellen Die Anwendung scheint eine statische Auswahl von Exponaten zur Auswahl anzubieten. Eine dynamische Erzeugung von Inhalten oder eine Anpassung des Exponats-Angebots in Abhängigkeit zum Benutzerverhalten findet nicht statt.

Dem entsprechend ist anzunehmen, dass die Anwendung, mit allen zu ihr gehörenden Medien, nach ihrer Fertigstellung nicht mehr erweitert wird.

Laufzeitumgebung Als Laufzeitumgebung dient der Flash-Plugin des Browsers. Daraus folgt, dass das Angebot des Guggenheim-Museums, wie die Viewzi Tag Cloud und das Multimedia-Archiv des MoMA, überall dort verfügbar ist, wo ein vollwertiger Flash-Plugin zur Verfügung steht.

Visualisierung und Benutzerschnittstelle Zentraler Punkt bei der Darstellung des Museums ist dessen Außenansicht, welche ebenfalls als Ausgangspunkt bei der Navigation zu den Exponaten fungiert (Abbildung 3.7 - oben). Mittels zwei Schaltflächen ist es dem Benutzer möglich, die Kameraposition in der Panoramaaufnahme des Museums und in den Aufnahmen der einzelnen Exponate, die an *QTVR*-Rundpanoramen erinnern, zu bewegen.

Abbildung 3.7: Navigation von der Außenansicht zum Exponat *Puppy*

Neben dem Sichtausschnitt, in dem die Exponate angezeigt werden, sind zusätzliche Informationen eingeblendet wie erläuternder Text oder Audio- und Videokommentare. Zur Steuerung dieser Multimedia-Elemente sind ebenfalls verschiedene Standard-Schaltflächen

(Play, Pause, Scroll-Leisten) verfügbar.

Rechtliche Beschränkungen Da alle Rechte an der Anwendung über die Architektur des Museumsgebäude und den darin ausgestellten Exponaten beim Guggenheim Museum liegen und eine Verwendung ausdrücklich untersagt ist, können für die Entwicklung des Prototypen keine verwendeten Algorithmen oder Methoden berücksichtigt werden.

Das Visualisierungs-Paradigma des Rundgangs, welche insbesondere den „Erkundungsgedanken“ aufgreift, ist für eine Verwendung im Prototypen nicht geeignet, da es sich bei der Visualisierung von Exponaten im Prototypen weder ausschließlich um architektonische Objekte handelt, noch diese Objekte in einem virtuellen Gebäude angeordnet werden sollen.

3.2 Programmierumgebungen

Die Betrachtung der verschiedenen digitalen Archive und Anwendungen zur Visualisierung von Datenbeständen hat gezeigt, dass verschiedene Technologien und Frameworks verwendet werden, um diese Anwendungen zu realisieren.

Im folgenden Abschnitt sollen verschiedene aktuelle Programmiersprachen und Frameworks beschrieben werden, die in der Lage sind, Anwendungen für verschiedenen Laufzeitumgebungen und Plattformen (Desktop-Rechner, mobile Geräte, Browser, installierbare Anwendung) aus einer Code-Basis zu erzeugen.

Die hier vorgestellten Programmiersprachen sollen auch als Auswahlmöglichkeiten für die Entwicklung des Prototypen berücksichtigt werden. Dem entsprechend ist eine genauere Betrachtung, auf die in späteren Kapiteln beim Entwurf zurückgegriffen wird, sinnvoll.

3.2.1 Java FX

Die Programmiersprache *Java*, entwickelt von Sun Microsystems [11], ermöglicht, im Gegensatz zu bspw. *C* pder *C++*, plattformunabhängiges Programmieren. Dies wird dadurch erreicht, indem plattformunabhängiger Bytecode durch die Kompilierung generiert wird und erst bei der Ausführung in einer betriebssystemabhängigen *Java Virtual Machine* (JVM) abgearbeitet wird.

Mittels Plugins ist es ebenfalls möglich, Java-Anwendungen für verschiedene Browser auszuführen; so genannte *Applets*. Auch auf mobilen Geräten wie Handys und PDAs existiert eine im Funktionsumfang reduzierte Version von Java (*Java ME*).

Durch die Veröffentlichung von Java FX versucht Sun Microsystems nun zusammen mit der NetBeans Entwicklungsumgebung in den Bereich der Erstellung von *Rich Desktop Applikations* und *Rich Internet Applikations* vorzudringen.

In Java FX können alle Java-Programmbibliotheken (JARs) benutzt werden, jedoch ist Java FX nicht so streng typisiert wie Java. Beim Kompilierungsvorgang werden Java FX-Quelldateien in einem ersten Schritt in Java-Quelltexte übersetzt, und diese anschließend mit einem gewöhnlichen Java-Kompiler in Bytecode übersetzt. Damit ist es möglich, Java FX-Programme auf allen Plattformen auszuführen, die über die eine JVM verfügen.

Zusätzlich beschleunigt der GUI-Builder in der Programmierumgebung NetBeans die Entwicklung von Benutzeroberflächen. Die Erweiterung Java FX Mobile erlaubt es, ebenfalls aus Java FX-Quellcode, Java ME-konforme, und somit auf mobilen Geräten funktionsfähige Programme zu erstellen.

3.2.2 OpenLaszlo

OpenLaszlo ist ein von der Firma Laszlo Systems entwickeltes System und bietet, wie Java FX, eine Plattform für die Entwicklung von Rich Internet Applications. Seit Oktober 2004 steht es unter der *Common Public Licence* und ist als *Freie Software* verfügbar.

Entwickelt werden Anwendungen mittels *LZX*-Dateien. Dieses XML-ähnliche Dateiformat ermöglicht die Kombination von deklarativen Anweisungen wie z.B. die Positionierung eines Steuerelement und prozeduralen oder funktionalen Komponenten, die mittels *ECMA-Scripten* realisiert werden können.

Diese LZX-Quellcode-Dateien werden zur Laufzeit von einem Tomcat-Server in wahlweise DHTML- oder Flash-Bytecode übersetzt. Damit ist es möglich, einerseits die Anwendung in Browsern, die keinen Flash-Plugin besitzen (was häufig auf mobilen Geräten der Fall ist) als DHTML oder die komplette Anwendungsumgebung im Flash-Plugin auszuführen. Mittels so genannter *SOLO*-Applikationen lassen sich Flash-Anwendungen exportieren und können somit, unabhängig vom Browser, lokal auf einem Rechner ausgeführt werden.

Das System ermöglicht es mit seiner deklarativen Entwicklungsumgebung und der Trennung von Anwendungsdesign und -funktionalität nach dem *MVC*-Entwurfsmuster, einfach komplexe und ansprechende Frontend-Anwendungen zu entwickeln.

3.2.3 Flex 2 und ActionScript 2

Flash-Anwendungen sind eine oft verwendete Technologie, um komplexere Anwendungen im Internet zugänglich zu machen, wenn mittels statischen (HTML, CSS) oder vorverarbeiteten Inhalten (PHP) nicht die angestrebte Interaktivität mit dem Benutzer realisiert werden kann.

Flash-Player, die entweder als Browser-Plugin Inhalte wiedergeben oder auf dem Rechner installierten Flash-Content auch ohne Browser darstellen können, sind wegen der hohen Verbreitung von Flash-Anwendungen auf vielen Rechnern installiert. Auch auf mobilen Endgeräten sind Player mit abgerüsteter Funktionalität vorhanden.

Mittels der von Adobe entwickelter Programmiersprache ActionScript und der Programmierumgebung Flex lassen sich umfangreiche Rich Internet Applications entwickeln. Die Verwendung von ActionScript 2 eignet sich besonders für die Entwicklung von Anwendungen für mobile Geräte, da moderne Versionen von abgerüsteten mobilen Flash-Playern ActionScript 2 vollständig unterstützen. Aus diesem Grund wird diese Version, trotz der Verfügbarkeit von ActionScript Version 3 weiterhin verwendet.

3.2.4 Flex 3, ActionScript 3 und Adobe AIR

Als nächsten Schritt in der Entwicklung von Flash-Anwendungen und der Erstellung von Rich Internet- und Rich Desktop Applications veröffentlichte Adobe ActionScript in der Version 3 mit der dazugehörigen Entwicklungsumgebung Flex, die nun auf der freien IDE *Eclipse* basiert.

Im Gegensatz zu ActionScript 2 bietet die neue Version die Typ-Überprüfung zur Kompilierzeit und implementiert alle Aspekte des objektorientierten Programmier-Paradigmas. Diese signifikanten Veränderungen bedeuten jedoch, das ActionScript 3-Quellcode nicht länger kompatibel zur Version 2 ist.

Die Entwicklungsumgebung Flex verfügt über einen GUI-Designer, mit dem *MXML*-Dateien erstellt werden können. Dieses XML-ähnliche Format ermöglicht es in deklarativer Form die Komponenten der Benutzerschnittstelle zu beschreiben. Diese Dateien werden zur Kompilierzeit in ActionScript-Klassen übersetzt und in das Programm eingebunden. Es ist also möglich, Komponenten, die in diesen MXML-Dateien beschrieben sind, in anderen ActionScript-Klassen zu benutzen. Diese Kombinationsmöglichkeiten ermöglichen es bspw. den Entwicklungsprozess einer Anwendung dahingehend aufzuteilen, dass der Designer mittels von der Flex-Entwicklungsumgebung bereitgestellter Werkzeuge die Oberfläche gestaltet, während losgelöst davon die Geschäftslogik als ActionScript-

Komponenten implementiert werden.

Der Nachteil, dass bedingt durch das Sicherheitskonzept von Flash-Anwendungen keine Daten lokal gespeichert oder gelesen werden können, wurde durch die Entwicklung von Adobe AIR versucht zu beseitigen. Damit ist es möglich, Desktop-Anwendungen zu erstellen, die, wie andere Desktop-Applikationen auch, eine Interaktion mit dem Host-Betriebssystem ermöglichen. Auch ist eine Installation eines Flash-Players für die Ausführung solcher Anwendungen mit Adobe AIR nicht nötig, da ein eingebetteter *Projektor* diese Aufgabe übernimmt.

Kapitel 4

Anforderungsanalyse

Die in den vorherigen Kapitels beschriebenen Grundlagen und die vorgestellten Beispielanwendungen zum Zugang zu digitalen Archiven sollen in diesem Kapitel dazu dienen, die Anforderungen, welche an den zu entwicklenden Prototypen gestellt werden, zu formulieren.

Hierzu ist es ebenfalls notwendig, zu Beginn die genauen Rahmenbedingungen zu betrachten, die später bei der Konzeption der Anwendung berücksichtigt werden müssen. Diese Betrachtung soll im ersten Abschnitt dieses Kapitels durchgeführt werden, indem die Anforderungen des *HardMut II*-Projektes (vgl. 1.2) genauer beschrieben werden.

Im folgenden Abschnitt sollen die Anwendungsumgebung des Prototypen genauer spezifiziert und anschließend die verschiedenen Anwendungsfälle identifiziert und beschrieben werden.

4.1 Rahmenbedingungen

Wie bereits beschrieben ist es das Ziel des HardMut II-Projektes, die Infrastruktur zur Entwicklung von mobilen Museen zu schaffen. In diesem Zusammenhang ist auch das mobile Museum des Jüdischen Museums zu nennen, in dessen Rahmen diese Entwicklungen eingesetzt und evaluiert werden sollen. Ebenfalls soll hier noch einmal erwähnt werden, dass in diesem mobilen Museum physische Exponate und ein digitales Archiv mit virtuellen Exponaten und Medien vorhanden sind.

Physische Exponate sind mit Barcodes ausgezeichnet, die eine elektronische Identifizierung ermöglichen. Diese Strichcodes sollen im Laufe des Projektes durch RFID-Chips ersetzt werden. Dadurch ist es möglich, mittels eines Lesegerätes ein Exponat zu identifizieren und somit bspw. mittels eines PDAs zusätzliche Informationen zu diesem anzuzeigen.

Diese Funktion sollte im Idealfall im Prototypen implementiert werden.

Zur Verwaltung von virtuellen Exponaten, deren Medien und anderer Informationen wie bspw. zu Touren durch das digitale Archiv, wurde eine Webapplikation entwickelt, die Daten zu Exponaten semantisch analysiert und vorverarbeitet in einer Datenbank ablegt. Zugang zu dieser Datenbank wird über einen via *SOAP*-Protokoll zugänglichen Webservice gewährleistet. Im zu entwickelnden Prototypen soll dieser als Datenquelle für Exponats- und Medien-Daten dienen. Eine genaue Funktionsbeschreibung dieses Dienstes soll im folgenden Punkt gegeben werden.

4.2 Anwendungsumgebung

In diesem Abschnitt soll genauer untersucht werden, in welcher Umgebung die Anwendung eingesetzt wird. Dazu zählt, welche Datenquellen benutzt werden können, wie die Laufzeitumgebung gestaltet ist, auf welchen Plattformen die Anwendung benutzt wird und für welche Zielgruppe die Anwendung letztendlich konzipiert werden soll.

4.2.1 Datenquellen

Wie bereits beschrieben, steht im HardMut II-Projekt ein Webservice zur Verfügung, der über einen Zugang via SOAP-Technologie verfügt, um so Informationen über Exponate und deren Medien abzurufen.

Jedoch verfügt dieser Dienst noch über weitere Schnittstellen, die hier kurz beschrieben werden sollen, da diese Funktionen auch im Prototypen benutzt werden können.

Exponate Mittels vier verschiedener Funktionen ist es möglich, Daten zu Exponaten abzurufen. Einerseits können Ausstellungsstücke direkt via ihrer ID, mit der sie in der Datenbank abgelegt und indiziert sind, oder eines anderen identifizierenden Attribut (z.B. Barcode oder RFID-Tag) abgerufen werden.

Die anderen Funktionen dienen der Suche von Exponaten. Eine ermöglicht es mittels einem oder mehrerer Suchworte in dem Beschreibungstexten der Ausstellungsstücke zu suchen, und liefert als Ergebnis eine Menge von Exponaten mit einem Relevanzwert, der die Ähnlichkeit zum Suchwort-Vektor anzeigt. Bei der Suche im Datenbestand auf dem Webserver wird das Kosinus-Ähnlichkeitsmaß (vgl. 2.1.4 - Berechnung der Ähnlichkeit) verwendet, um diesen Wert zu bestimmen. Die zweite Suchfunktion implementiert die Suche von Ausstellungsstücken, die einem gegebenen Exponat ähnlich sind.

Mit Hilfe dieser Funktionen ist es möglich, auf Exponate direkt zuzugreifen, indem sie mittels eines identifizierenden Attributs abgerufen werden (ID, RFID-Tag), oder mittels eines Suchwortes oder eines anderen Exponats nach ähnlichen Objekten zu suchen.

Medien Medien können entweder via ihrer ID, oder - da eine Verbindung zwischen dem Exponat und den zugehörigen Medien besteht - alle Medien, die zu einem spezifizierten Ausstellungsstück gehören, abgerufen werden.

Virtuelle Touren In der Webanwendung können ebenfalls virtuelle Touren durch das digitale Archiv angelegt werden. Dabei beginnt der Benutzer bei einem Startelement (Exponat), welches einen oder mehrere Nachfolger hat. Diese haben wiederum ebenfalls einen oder mehrere Folgeelemente, so dass sich der Weg beliebig stark verzweigen kann. Diese Tour-Daten können ebenfalls vom Webservice abgerufen werden und ermöglichen es somit auch im Prototypen, die Touren durch den Datenbestand anzubieten.

4.2.2 Laufzeitumgebung

Die bereits vorgestellten Anwendungen (vgl. 3) können entweder in einem Browser via eines Plugins ausgeführt werden oder müssen auf dem Zielrechner installiert sein, um zu funktionieren. Ebenfalls lässt sich feststellen, dass keine der Anwendungen Betriebssystem-spezifisch ist.

Der Prototyp sollte ebenfalls in verschiedenen Umgebungen funktionsfähig sein. Eine Verwendung in einem Browser, so dass auf der Website des Museums Zugang zum digitalen Archiv ermöglicht werden kann; oder aber als installierbare Applikation, so dass der Benutzer, im Gegensatz zur Browseranwendung, persönliche Einstellungen dauerhaft speichern und wieder verwenden kann.

Da es sich beim HardMut-Projekt um die Entwicklung eines mobilen Museums handelt, sollte die Verwendung auf einem mobilen Gerät nicht ausgeschlossen werden. Auch die Unterstützung externer Lesegeräte oder die Verarbeitung von solchen Umgebungsdaten sollte im Prototypen ermöglicht werden.

Wenn der Prototyp eine solche Vielfalt von Laufzeitumgebungen unterstützen soll, werden aufgrund der verschiedenen Geräteeigenschaften (bspw. Rechenleistung, Auflösung) hohe Ansprüche an die Skalierbarkeit der Anwendung gestellt.

4.2.3 Zielgruppe

Da Schulen die Stationen einer Tour des Jüdischen Museums mit dem mobilen Museum darstellen, ist vorrangig junges Publikum als Zielgruppe für die Anwendung zu erkennen. Soll der Prototyp zusätzlich auch öffentlich und frei zugänglich auf der Homepage des Museums angeboten werden, so ist diese Zielgruppen-Beschränkung nicht länger gültig.

Jedoch sollte auch die Gestaltung der Anwendung hinsichtlich Benutzerschnittstelle und Usability den Zugang für ein möglichst breites Spektrum an Benutzern ermöglichen.

4.3 Anwendungs-Gestaltung

Der zu entwickelnde Prototyp soll, wie bereits erwähnt, für ein breites Publikum zugänglich sein und für den Zugang zum digitalen Archiv des (mobilen) Museums eingesetzt werden. Ähnlich wie bei einem tatsächlichen Besuch im Museum soll der Erkundungsgedanke sich ebenfalls im Prototypen wieder finden, in dem sich der Benutzer ausgehend von einem (selbstgewählten oder vordefinierten) Startpunkt aus durch das Archiv bewegt.

Nachdem verschiedene theoretische Verfahren zur Visualisierung von großen Datenmengen (vgl. 2.3.2) und verschiedene Anwendungen, die auf unterschiedliche Art diese Daten darstellen (vgl. 3), betrachtet wurden, soll hier nun das Grundkonzept für die Datendarstellung im Prototypen festgelegt und erläutert werden.

Ähnlich der Viewzi Photo Tag Cloud-Anwendung soll auch beim Prototypen der Benutzer vor einer (fast) leeren Fläche beginnen, auf der er, an beliebiger Position, ein Suchwort eingeben kann. Zu diesem Suchbegriff relevante Exponate des Museums werden anschließend mit ihren Namen und einem Vorschaubild eines ihrer Medien um diesen Suchbegriff angeordnet und mittels einer Linie mit diesem verbunden. Damit kennzeichnet die Linienstärke die Relevanz des Exponats zum Suchwort (Abbildung 4.1).

Anschließen kann der Benutzer, ebenso wie bei der Viewzi-Applikation, ein Knoten dieses entstandenen Graphen auswählen, welcher dann ein eigenständiges Element in der Darstellung wird und ebenfalls Knoten um sich herum anordnet. Anhang A.1 zeigt diese Entwicklung anhand der Auswahl eines Exponat-Vorschau-Knotens. Nach der Auswahl setzt sich dieses Element vom ursprünglichen Suchelement ab und gruppiert (da es sich um einen Exponats-Knoten handelt) Vorschau-Knoten zu verknüpften Medien und Elemente mit Verweisen zu ähnlichen Exponaten um sich, anhand dieser der Benutzer weiter im digitalen Archiv „stöbern“ kann. Dieses Vorgehen entspricht dem zu realisierenden Ansatz des explorativen Zugangs.

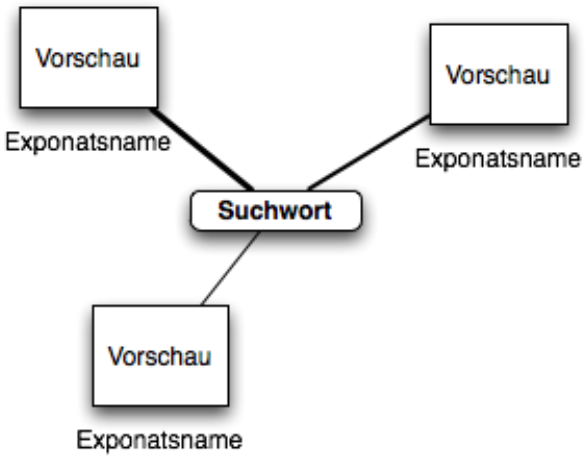

Abbildung 4.1: Suchwort mit relevanten Exponaten

4.4 Use-Case-Analyse

Mit der Identifizierung der einzelnen Anwendungsfälle sollen grundlegende Erkenntnisse zum Funktionsumfang der Anwendung gewonnen werden, die für einen folgenden Systementwurf unbedingt notwendig sind.

In Anhang A.2 sind alle Anwendungsfälle in einem Use-Case-Diagramm dargestellt. An dieser Stelle sollen die einzelnen Elemente genauer betrachtet und damit verbundene Arbeitsabläufe aufgezeigt werden.

4.4.1 Datenquellen verwalten und Anwendungskonfiguration

Analog zur Anwendung des Commetrix-Projektes sollte auch der zu entwickelnde Prototyp nicht auf eine Art von Datenquelle festgelegt sein (SOAP-Webservice), sondern mittels verschiedener *Konnektoren* auch Verbindungen zu anderen Diensten ermöglichen. Da die zu benutzenden Datenquellen nicht statisch im Programmcode eingetragen werden sollen, um ein gewisses Maß an Flexibilität zu gewährleisten, ist es notwendig, dass dem Benutzer die Möglichkeit gegeben wird, zu benutzende Datenquellen zu konfigurieren.

Zu diesen Verwaltungsaufgaben zählen das Hinzufügen, Löschen und Bearbeiten von Verbindungseinstellungen zu einer Datenquelle. Beim Hinzufügen einer neuen Verbindung muss diese einmalig konfiguriert werden. Um die Anwendung benutzen zu können, muss mindestens eine Verbindung eingerichtet werden, da sonst keine Daten abgerufen werden können.

Um diese Einstellungen nicht bei jeder erneuten Programmausführung neu eingeben zu müssen bzw. einem neuen Anwender das Programm in einem vorkonfigurierten Zustand

zur Verfügung zu stellen, ist es ebenfalls erforderlich, dass der Prototyp die Möglichkeit besitzt, mittels einer Konfigurationsdatei einen einmal eingerichteten Zustand wiederherzustellen.

Die Möglichkeit, eine Konfigurationsdatei zu benutzen um einen definierten Zustand wiederherzustellen setzt voraus, dass Einstellungen und Zustand der Anwendung gespeichert werden können, um eine solche Konfigurationsdatei zu erzeugen. Das Speichern der Konfiguration ist jedoch nur auf Plattformen möglich, bei denen die Anwendung Zugriff auf dessen Dateisystem hat.

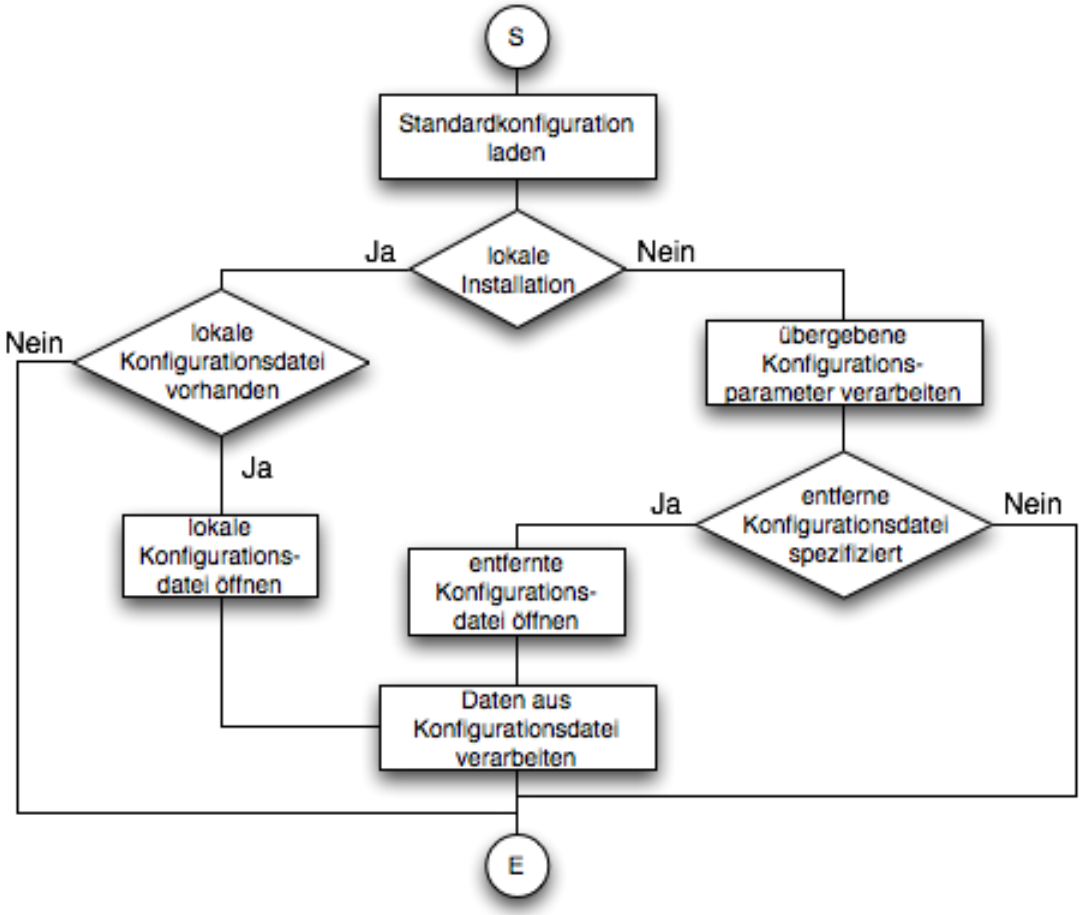

Abbildung 4.2: Einstellen von Standardparametern mittels Konfigurationsdatei

Auch die automatische Konfiguration der Anwendung stellt, bedingt durch die Vielzahl an zu unterstützenden Plattformen und Funktionsmodi, einen komplexeren Arbeitsablauf dar. Abbildung 4.2 stellt diesen Vorgang als Flussdiagramm dar. Der einfachere Fall, bei dem die Anwendung auf einem Rechner installiert ist, prüft lediglich nach dem Laden der Standardeinstellung, ob eine Konfigurationsdatei vorhanden ist, um diese zu verarbeiten. Bei der Ausführung z.B. im Browser müssen dagegen zuerst mittels Parameter, die direkt von der Anwendung verarbeitet werden, eine entfernte Konfigurationsdatei spezifiziert

werden, die dann zur Einstellung von Parametern verwendet wird.

4.4.2 Exponate und Medien betrachten

Die Betrachtung von Exponaten und damit verknüpften Medien ist der am häufigsten auftretende Anwendungsfall des Prototypen. Neben der Auswahl eines konkreten Objektes bzw. Mediums muss es dem Benutzer ebenfalls ermöglicht werden, nach Objekten zu suchen oder ähnliche Exponate zu einem Gegebenen anzuzeigen.

Durch die bereits vorgestellte Gestaltung der Anwendung ist der Ablauf bei der Auswahl eines Exponats oder Mediums festgelegt. Im darzustellenden Graphen mit Exponats- und Medien-Knoten kann der Benutzer einen dieser auswählen, um mehr Information dazu zu erhalten. Wählt er dabei einen Exponats-Knoten, handelt es sich um nähere Informationen zum Objekt, während bei der Auswahl eines Mediums dessen Daten (Bild, Modell, Ton) abgerufen und angezeigt werden.

Die Suche ist dem Benutzer möglich, indem er Suchknoten im Graphen anlegt, während die Suche nach verwandten Objekten von jedem Exponats-Knoten aus möglich ist.

4.4.3 Touren

Ein weiterer Anwendungsfall stellt die Möglichkeit, eine vordefinierte Tour durch das digitale Archiv zu machen, dar. Dabei muss zuerst die entsprechende Tour ausgewählt werden können. Welche Touren dabei zur Verfügung stehen, muss von der Datenquelle ermittelt werden, weshalb deren Verbindungskonfiguration auch hier zwingend erforderlich ist. Die Durchführung einer Tour ähnelt dabei der freien Erkundung des digitalen Archivs, nur das anstatt bei einem Exponats-Knoten im Graphen nicht ähnliche Exponate angezeigt werden, sondern lediglich die nachfolgenden Tourstation zur Auswahl verfügbar sind. Somit ist es dem Benutzer nicht möglich, vom definierten Weg abzuweichen.

Kapitel 5

Systementwurf

Nachdem die Anforderungen an den Prototypen spezifiziert worden sind, soll in diesem Kapitel mittels dieser Vorgaben ein Systementwurf erstellt und beschrieben werden.

Zuerst muss dafür eine System-Architektur gewählt werden. Anschließend werden die einzelnen Komponenten dieser Architektur entwickelt und beschrieben werden.

5.1 Architektur

Komplexe Software-Projekte erfordern es aufgrund ihrer umfangreichen Funktionalität in eine Struktur gebracht zu werden, um Wartbarkeit und Erweiterbarkeit zu gewährleisten.

Eines dieser Architekturmuster ist das *Model-View-Controller*-Prinzip, bei dem die Funktionalität einer Anwendung in die drei Bereiche Modell bzw. Datenhaltung (*Model*), Präsentation (*View*) und Steuerung (*Controller*) unterteilt wird. Diese drei Komponenten werden miteinander assoziiert. Das Standardmodell sieht dabei direkte Verbindungen von Controller zu Model und View sowie von der View-Komponente zum Model und indirekte Verbindungen, welche bspw. über Nachrichten (*Events*) realisiert werden, vom Model zur View-Komponente und von dort zum Controller. Abbildung 5.1 zeigt diesen Aufbau als Diagramm.

Auch wenn viele Anwendungen nach dem MVC-Muster entwickelt werden, wird dieses Prinzip häufig in abgewandelter Form verwendet. So bietet das Entwurfsmuster bspw. keine festen Richtlinien, wo die Geschäftslogik unterzubringen ist. Auch besitzen Komponenten, die für die Oberflächengestaltung einer Anwendung verwendet werden, also Teil der View-Komponete sind, häufig einen eigenen Controller für die Verarbeitung von Nachrichten und zum Teil sogar eigene Modell-Daten, die mit der Anwendung synchronisiert werden müssen.

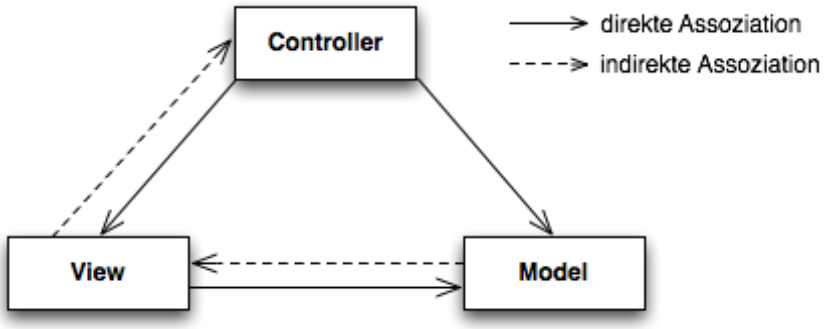

Abbildung 5.1: Model-View-Controller-Konzept mit Komponenten-Assoziationen

Das MVC-Entwurfmuster soll auch bei der Entwicklung des Prototypen zum Einsatz kommen. In folgenden Abschnitten wird dargelegt, welchen Komponenten dabei welche Funktionen der Anwendung zugeordnet werden.

5.2 Modell-Komponenten

Nach der Anforderungsanalyse für den Prototypen können drei Bestandteile der Modell-Komponente identifiziert werden.

Die zu realisierende Unterstützung von verschiedenen Möglichkeiten zur Datenbeschaffung, die im Prototypen beispielhaft durch die Implementierung einer Verbindung zu einem SOAP-Webservice realisiert werden soll, macht es notwendig, einheitliche Datentypen für die verschiedenen Modell-Daten zu definieren und eine Architektur zu entwickeln, welche die geforderte Erweiterbarkeit gewährleistet.

Der Graph, welcher durch die Anwendungsbenutzung und die Navigation im digitalen Archiv entsteht und aus verschiedenen Arten von Knoten und mit Eigenschaften versehenen Kanten besteht, stellt einen weiteren Bestandteil der Modell-Komponente dar.

Da die Anwendung ebenfalls in der Lage sein soll, auf externe Dateneingabe wie das Einlesen eines Barcodes/ RFID-Tags zu reagieren, werden Programmbestandteile in der Modell-Komponente benötigt, die diese eingehenden Daten verarbeiten und weitere Aktionen auslösen können.

5.2.1 Modelldaten und Datentypen

Wenn die Daten, welche die Applikation verarbeiten und visualisieren soll, aus verschiedenen Datenquellen stammen sollen, ist es zwingend notwendig einheitliche Datentypen zu definieren, in welche die empfangenen Daten umgewandelt werden, um eine Abstrakti-

onsschicht zu schaffen, mit deren Hilfe eine standardisierte Verarbeitung unabhängig von der Datenquelle möglich wird.

Folgende Datentypen werden benötigt:

Exponat Zu den Daten, welche zu einem Exponat übermittelt werden, gehören dessen ID, Name, Beschreibungstext, Sprache, Erstellungsdatum sowie verschiedene Identifikatoren des Autors, der dieses Objekt angelegt hat, sowie Status und Sichtbarkeit des Exponats. Außerdem verfügt jedes Exponat über eine bestimmt Menge von Attributen wie Barcode oder RFID-Tag, mittels denen (neben der ID) eine Identifizierung möglich ist.

Medium Ein mit einem Exponat verknüpftes Medium besitzt ebenfalls eine ID, mit der es bei der Datenquelle identifiziert werden kann, sowie die ID des Exponats, zu dem es gehört. Außerdem gehören Informationen zu Name, MIME-Type, Erstellungsdatum sowie Verweise auf Vorschau- und die Original-Datei zu einem Medien-Datensatz.

Benutzer/Autor Da es möglich ist, Informationen über Autoren von der Datenquelle abzurufen, wird dafür ebenfalls ein Datentyp benötigt. Benutzerinformationen bestehen aus ID, Benutzernamen und realem Name (soweit angegeben).

Tour Eine Tour besteht aus mehreren Stationen und definiert eine davon als Startpunkt. Wie Exponat und Medium auch, besitzen Tour-Datensätze einen Identifikator und Namen.

Station einer Tour Die Station einer Tour enthält Verweise zur Tour zu der sie gehört, sowie zum Exponat, welches an dieser Station vom Benutzer betrachten werden soll. Außerdem sind Informationen über direkte Vorgänger und Nachfolger dieser Station mit ihr verknüpft.

Das Entity-Relationship-Diagramm in Anhang A.3 soll die Zusammenhänge der einzelnen Datentypen untereinander verdeutlichen. Mittels der Konnektoren werden die empfangenen Daten in dieses Schema transformiert.

5.2.2 Datenquellen und Konnektoren

Ein Konnektor soll in der Anwendung dazu benutzt werden eine Verbindung zu einer Datenquelle herzustellen, um von dieser dann Informationen über Exponate, Medien usw. zu beziehen.

Für den Prototypen soll ein Konnektor entwickelt werden, der eine Verbindung zu einem SOAP-Webservice herstellen und Daten von dort abrufen kann. Da jedoch berücksichtigt werden muss, dass später weitere Konnektoren für andere Datenquellen hinzugefügt werden, muss eine Struktur zur Verfügung stehen, die dies ermöglicht.

Um dies zu realisieren, muss der Funktionsumfang eines Konnektors definiert werden, um eine einheitliche Schnittstelle für andere Funktionen und Komponenten der Anwendung bereit zu stellen. Mittels objektorientierter Programmierkonzepte wie *Vererbung* können spezifische Konnektoren diese Schnittstelle dann implementieren.

Bei der Analyse der Anwendungsumgebung wurden folgende Funktionen ermittelt, die ein Konnektor zur Verfügung stellen muss:

- Exponate anhand deren ID abrufen
- Exponate anhand eines Identifikor-Attributs abrufen
- Exponate suchen
- Zu einem Exponat ähnliche Objekte suchen
- Autor-Informationen abrufen
- Medien anhand deren ID abrufen
- Alle Medien eines Exponats abrufen
- Verfügbare Touren abrufen
- Tour anhand deren ID abrufen
- Alle Stationen einer Tour abrufen
- Station anhand deren ID abrufen

Zusätzlich sind weitere Funktionen nötig, die nicht direkt dem Datenempfang dienen, aber für eine einheitlich zu verwendende Schnittstelle notwenig sind.

- Konnektor konfigurieren
- Verbindung aufbauen/abbauen

5.2.3 Graph

Während der Benutzer die Applikation verwendet und sich durch das digitale Archiv bewegt, entsteht ein ungerichteter Graph, in dem Suchwort-, Exponats- und Medien-Knoten enthalten sind. Diese Knoten sind über gewichtete Kanten miteinander verbunden.

Diese Struktur soll in folgendem Klassendiagramm dargestellt werden, wobei bewusst auf Methoden- und Attributsnamen verzichtet wird, da zu diesem Zeitpunkt eine solch konkrete Formulierung noch nicht möglich ist.

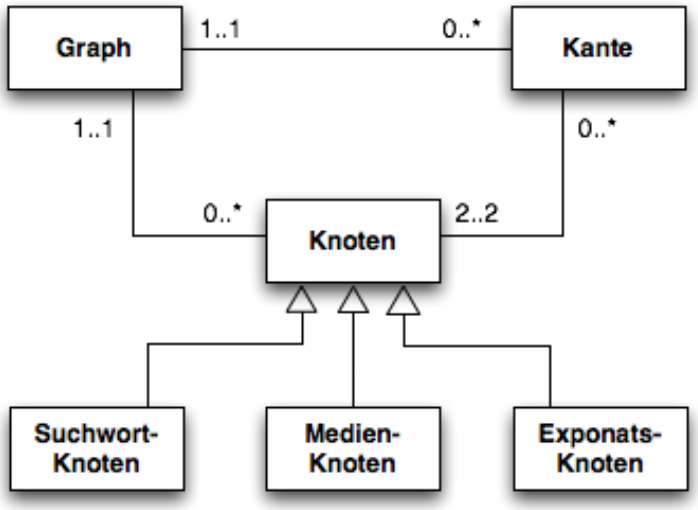

Abbildung 5.2: Klassen-Beziehungen und Vererbungen der Graphenkomponenten

5.2.4 Externe Dateneingabe

Um die externe Dateneingabe zu ermöglichen, können verschiedene Konzepte verwendet werden. Es kann dabei bspw. eine standardisierte Schnittstelle spezifiziert werden, mittels der (ähnlich dem Prinzip der Konnektoren) verschiedene Implementierung von solchen *externen Sensoren* möglich wären.

Ein anderer Ansatz, welcher einen höheren Grad der Abstraktion benötigt, jedoch somit eine Verringerung der Komplexität erzielt, ist die Benutzung einer standardisierten Netzwerkschnittstelle. Dabei wird der Prototyp in die Lage versetzt, sich mit einem spezifizierten Dienst mittels etablierter Standard-Technologien zu verbinden und über diesen Daten eines externen Sensors zu empfangen.

Die Vorteile dieser Technik werden offensichtlich, wenn bspw. betrachtet wird, welche Vorraussetzungen für das Einlesen und Verarbeiten eines RFID-Tags eines Exponats notwendig sind, welches mittels eines mobilen Gerätes erfasst werden soll.

In diesem Fall muss das Lesegerät direkt angesteuert werden. Mittels der in Abschnitt 3.2 vorgestellten Programmiersprachen und Frameworks ist dies, gerade wegen der von ihnen ermöglichten hohen Plattformunabhängigkeit, oft nicht möglich. Verwendet man nun jedoch ein separates Programm zum Ansteuern der Hardware, welches dann über Netzwerkprotokolle mit dem Prototypen kommuniziert und die eingelesenen Daten übermittelt, ist eine Entkopplung von hardwarespezifischen und plattformunabhängigen Komponenten gelungen.

Aufgrund dieser Vorteile soll dieses Verfahren für den Empfang externer Dateneingaben verwendet werden.

5.3 Anwendungssteuerung

Die Anwendungssteuerung ist für die Überwachung des Anwendungsverlaufs zuständig. Sie kontrolliert die Erzeugung von Komponenten der Präsentations- und Datenhaltungsschicht und sorgt für die Einrichtung aller benötigten Programmbibliotheken zum Anwendungsstart.

Im Idealfall kann die Anwendung dann ohne weitere Konfiguration durch den Benutzer sofort benutzt werden; dieser Fall ist bei der Verwendung mittels bspw. Browser angestrebt, um sofortigen Zugang zum digitalen Archiv zu gewähren. Anhang A.4 zeigt die Prozessschritte, die vom Controller beim Anwendungsstart ausgeführt werden müssen.

In dieser Komponente müssen ebenfalls Anpassungen vorgenommen werden, die notwendig sind, um die Anwendung für den Ablauf für die jeweiligen Plattform und deren Anwendungsumgebung einzurichten.

5.4 Präsentationsschicht

Die Programmstruktur der Präsentationsschicht muss vor allem im Hinblick auf die angestebte Unterstützung verschiedener Plattformen und der damit verbundenen Unterstützung verschiedener Auflösungen und Rechenkapazitäten so gewählt werden, dass es möglich ist, die Benutzerschnittstelle und die Datenvisualisierung zur Programmlaufzeit an diese Gegebenheiten anzupassen.

Dabei bezeichnet in diesem Zusammenhang die Benutzerschnittstelle alle Steuerelemente, mit denen der Benutzer einerseits die Anwendung steuern kann, wie bspw. Funktionselemente zum Speichern/Laden von Konfigurationen, und andererseits Komponenten, mit denen die Datenvisualisierung beeinflusst werden kann, wie z.B. Einstellungsmöglichkeiten

zum Zoomfaktor.

Die Gründe zur Aufteilung der Präsentation in Benutzerschnittstelle und Visualisierung soll Abbildung 5.3 darstellen. Sie zeigt, dass während die Visualisierung stets den Großteil der Bildschirmfläche, den die Anwendung benutzt, belegt und lediglich in Ihrer Größe variiert, die Steuerelemente je nach Plattform völlig unterschiedlich dargestellt werden können.

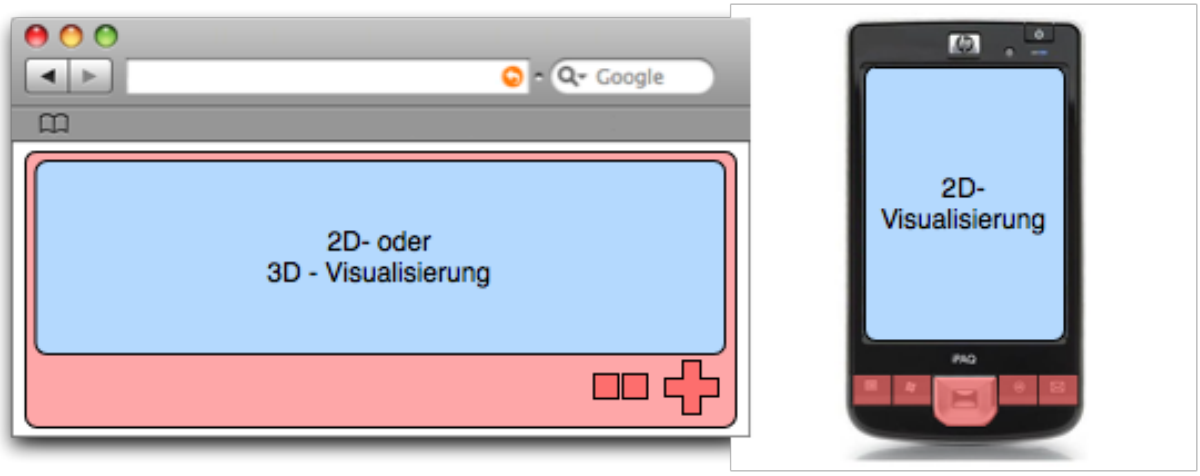

Abbildung 5.3: Komposition von Benutzerschnittstelle und Visualisierung

Um größtmögliche Flexibilität zu erreichen und Doppelentwicklungen von Code zu vermeiden, werden diese beiden Komponenten von einander getrennt und erst bei der Erstellung der Benutzeroberfläche durch den Controller wieder kombiniert. Dazu ist es notwendig, Schnittstellen und Nachrichten zu spezifizieren, welche die Kommunikation zwischen diesen Komponenten ermöglichen.

Damit jedoch die Benutzeroberfläche als ein kompaktes Subsystem von anderen Programmstrukturen steuerbar ist, kann eine *Fassade*-Klasse benutzt werden, welche Methoden zur Oberflächen-Manipulation bereitstellt und entsprechend an Schnittstellen- und Visualisierungs-Komponenten weitergibt (Abbildung 5.4). Da es jeweils nur eine Benutzeroberfläche pro Anwendungsinstanz gibt, ist es sinnvoll, diese Fassade-Klasse nach dem *Singleton*-Entwurfsmuster zu implementieren.

5.4.1 Benutzerschnittstelle

Im vorherigen Abschnitt wurde bereits erläutert, dass die Benutzerschnittstelle mit ihren Steuerelementen am stärksten von der Anpassung an die Laufzeitumgebung betroffen ist. Aus diesem Grund sollte die Gestaltung und das Hinzufügen neuer Benutzeroberflächen möglichst einfach und im Idealfall auch für ungeschultes Personal möglich sein.

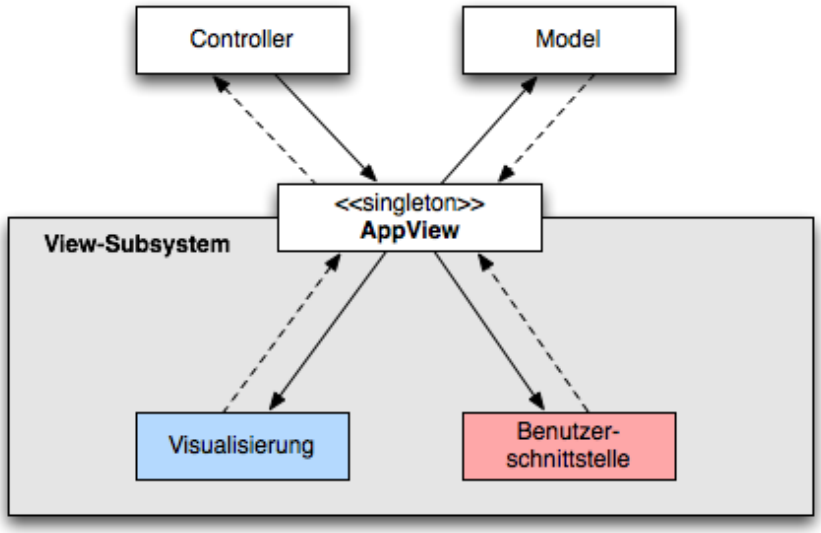

Abbildung 5.4: Fassade-Klasse zur Verwendung der Benutzeroberflächen-Kapselung

Die Benutzerschnittstelle beinhaltet, trotz ihrer variierenden Darstellung, stets die gleichen Steuerungselemente. Das Vorhandensein einzelner Elemente kann jedoch von Laufzeitumgebung zu Laufzeitumgebung verschieden sein. So wird bspw. eine Komponente zum Öffnen und Einlesen einer Konfigurationsdatei nicht notwendig sein, wenn die Anwendung mittels eines Plugins im Browser ausgeführt wird.

Folgende Funktionen soll eine Benutzerschnittstelle des Prototypen bereitstellen können:

Konfiguration laden/speichern Ist die Anwendung auf einer Plattform installiert, muss es mit der Benutzerschnittstelle möglich sein, Konfigurationsdateien zu speichern oder zu laden. Dabei übernimmt die Benutzerschnittstelle lediglich die Aufgabe, den Pfad, an dem die Datei gespeichert bzw. von dem sie geladen werden soll zu bestimmen, während der Speicher-/Lade-Vorgang dann von Model- und Controller-Funktionen ausgeführt wird.

Anwendung beenden Ebenfalls nur dann verfügbar, wenn die Applikation lokal installiert und als eigenständige Anwendung ausgeführt wird, sind Steuerelemente zum Beenden des Programms.

Manipulation der Visualisierung Zu diesem Punkt zählen alle Steuerfunktionen, die zur Veränderung der Visualisierung dienen. Dies sind Zoom- und Translations-Funktionen, die den Sichtausschnitt des Benutzers auf die Visualisierung verändern sowie Funktionen um dessen bisherigen Pfad durch das digitale Archiv zu zeigen und die Darstellung auf den Ausgangspunkt zurückzusetzen.

Spezifikation des Visualisierungs-Bereichs Da erst zum Anwendungsstart Visuali-

sierung und Benutzerschnittstelle kombiniert werden, muss die Benutzerschnittstelle einen Bereich spezifizieren, in dem die Visualisierung-Komponente dargestellt wird.

Da die Visualisierung von der Benutzerschnittstelle getrennt ist, jedoch dessen Kontrollelemente die Darstellung manipulieren können müssen, ist es notwendig, dass beide Komponenten in der Lage sind, miteinander zu kommunizieren. Dies kann realisiert werden, indem sie entweder eng gekoppelt werden, d.h. jede Instanz eine Referenz der anderen besitzt und so jeweils Methoden der anderen aufrufen kann, oder sie nur lose gekoppelt werden und so lediglich mittels Nachrichten kommunizieren.

Da die zweite Variante die flexiblere darstellt und auch für die spätere Erweiterung des Prototypen besser geeignet ist als die erste, soll diese in der Implementierungs-Phase verwendet werden.

5.4.2 Visualisierung

Die Darstellung der Exponate des Museums und die Visualisierung des Graphen, mit dessen Hilfe der Benutzer Zugang zum digitalen Archiv erhält und in diesem navigiert, stellt eine komplexe Aufgabe dar. Diese Aufgabe lässt sich in drei Teilbereiche untergliedern: die Darstellung des Graphen im einem virtuellem Raum, die Optimierung dieser Darstellung durch Positionierung der Knoten nach in Abschnitt 2.3.2.3 beschrieben Kriterien und die Visualisierung verschiedener Medientypen.

Einige dieser Aufgaben erfordern zusätzlich eine Anpassung an die Laufzeitumgebung. So ist es bspw. nicht möglich, auf einer abgerüsteten mobilen Plattform komplexe texturierte 3D-Modelle darzustellen, ohne einen unvertretbaren Performance-Verlust der Anwendung hinzunehmen. Dementsprechend kann auf solchen mobilen Geräten der Graph lediglich als zwei-dimensionales Objekt dargestellt werden, und muss demzufolge nicht in der Z-Achse, sondern nur in X- und Y-Richtung optimiert werden; ferner ist eine Visualisierung von drei-dimensionalen Objekten nicht möglich.

Abbildung 5.5 verdeutlich den Zusammenhang zwischen den Komponenten zur Darstellung des virtuellen Raumes (*SpaceRenderer*), zur Graphen-Optimierung (*GraphOptimizer*) und zur Darstellung von verschiedenen Medientypen (*MediumRenderer*).

Virtueller Raum und Graphen-Darstellung Die Komponente, welche den virtuellen Raum und den darin enthaltenen Graphen visualisiert, muss verschiedene Schnittstellen besitzen, um auf Benutzereingaben zu reagieren. Dazu zählen Nachrichten, die versandt werden, wenn die Darstellung des virtuellen Raum manipuliert, ein Knoten aus

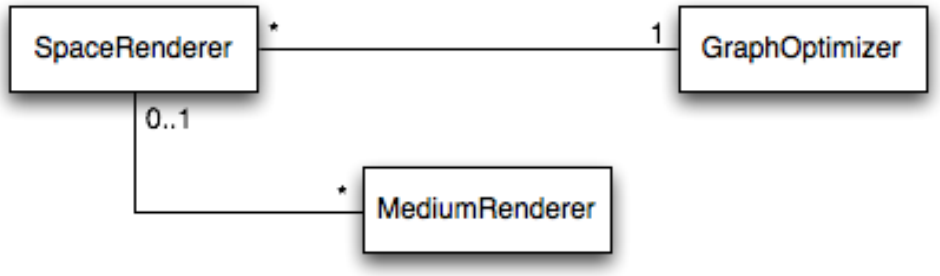

Abbildung 5.5: Komposition der Visualisierungskomponenten

dem Graphen entfernt oder hinzugefügt wird oder ein spezifischer Knoten angewählt oder dessen Inhalt fokussiert dargestellt werden soll.

Außerdem werden, da der Graph verschiedene Arten von Knoten besitzt, unterschiedliche Methoden benötigt, um diese im virtuellen Raum anzuzeigen. Auch bewirkt eine Auswahl eines dieser Knoten durch den Benutzer, abhängig vom Knoten-Typ, eine andere Reaktion der Anwendung. Eine Auswahl eines Exponats-Knoten in der Vorschau bewirkt, dass der Graph sich erweitert (Anhang A.1),während bei der Auswahl eines Medien-Knotens die Originaldatei des Mediums abgerufen und im Vollbild mit zugehörigen Informationen dargestellt wird.

Medien-Visualisierung Unterschiedliche Medien-Typen benötigen verschiedene Methoden der Darstellung. Ein Bild muss anders visualisiert werden als eine Audio-Aufzeichnung oder ein 3D-Modell. Auch lassen sich, wie bereits erwähnt, nicht alle Medien in allen Laufzeitumgebungen darstellen oder erfordern abhängig von dieser eine differenzierte Darstellungsform. Bspw. muss ein Bild auf einer zwei-dimensionalen Fläche lediglich als Bildelement mit X- und Y-Koordinate dargestellt werden, während zur Visualisierung im 3D-Raum meistens eine texturierte Fläche mit X-, Y- und Z-Koordinaten benötigt wird.

Eine Komponente zur Darstellung von Medien im Prototypen muss über sich selbst Auskunft über unterstützte Medientypen geben können. Damit ist es möglich, dass ein SpaceRenderer beliebig viele MediumRenderer besitzt und anhand des benötigten Medientyps die passende Komponente auswählt. Diese muss zusätzlich Informationen über vom Medium unterstütze Manipulationsmöglichkeiten bereitstellen, damit Kontrollelemente der Benutzerschnittstelle an diese angepasst werden können.

Graphen-Optimierung Die Darstellung des Graphen und dessen unterschiedlichen Knoten erfordert besondere Algorithmen, um ein Mindestmaß an Ästhetik und Übersichtlichkeit zu garantieren. Zu diesem Zweck wird der Force-Directed-Algorithmus

von Fruchterman und Reingold zum Einsatz kommen, da eine Referenz-Implementierung im Commetrix-Projekt vorliegt und so die Entwicklungszeit verkürzt werden kann.

Um die Erweiterbarkeit des Programmes zu gewährleisten und Doppelentwicklung von Programmcode zu vermeiden, soll für alle drei Aufgabenbereiche der Visualisierung Basisklassen erstellt werden, welche grundlegende Funktionen bereits enthalten und konkrete Implementierungen einzelner Sachverhalte in abgeleiteten Klassen untergebracht werden.

Im Rahmen der Prototyp-Entwicklung sollen zwei verschiedene SpaceRenderer und GraphOptimizer implementiert werden, die die Darstellung der Graphen und dessen Optimierung im zwei- und drei-dimensionalen Raum ermöglichen. Jeder dieser SpaceRenderer muss dementsprechend, aus oben beschriebenen Gründen, mehrere MediumRenderer besitzen, um unterschiedliche Medien-Typen darstellen zu können.

Kapitel 6

Implementierung

Nach dem Systementwurf soll in diesem Kapitel die Umsetzung der verschiedenen Programm-Kompontenen beschrieben werden. Dabei soll im ersten Abschnitt die Wahl der Programmiersprache und des verwendeten Frameworks begründet werden, anschließend wird die Implementierung der Model-, Controller- und View-Komponenten näher erläutert werden.

6.1 Programmiersprache

Für die Umsetzung der Anforderungen in den Prototypen gab es keine Limitierungen bei der Auswahl der Programmiersprache und des Frameworks. Lediglich die Verwendung von Standard-Technologien zur Kommunikation mit anderen Diensten und Anforderungen an Erweiterbarkeit und Vielseitigkeit der unterstützten Laufzeitumgebungen mussten berücksichtigt werden.

Im Idealfall ist es mit der gewählten Programmierumgebung möglich, eine Anwendung zur Benutzung im Browser und zur lokalen Installation sowie eine Verwendung auf rechenstarken Desktop- und Notebook-PCs und eingeschränkten Mobilplattformen zu erstellen. Zusätzlich sollte die Erzeugung all dieser Varianten aus der gleichen Codebasis ohne oder nur mit geringfügigen Änderungen möglich sein.

Dazu sollen die bereits in Abschnitt 3.2 vorgestellten Frameworks und Programmiersprachen erneut gegenüber gestellt und hinsichtlich der identifizierten Anforderungen untersucht werden.

Java FX und Java FX Mobile Mittels Java FX und der Erweiterung Java FX Mobile ist es möglich, Anwendungen für Desktop-, Notebook und Mobil-Geräte bis hin zu Han-

dys zu erstellen. Auch eine Verwendung als Web-Anwendung im Browser ist mittels der Applet-Technologie möglich. Da es sich bei Java FX und der Basis Java um eine objektorientierte Sprache handelt, lassen sich die gewählten Programmstrukturen wie geplant implementieren; durch den verfügbaren GUI-Builder ist es möglich, auch für ungeschultes Personal Benutzeroberflächen zu erstellen, die dann in das Programm integriert werden können.

Allerdings ist es mittels Java FX Mobile nicht möglich, ohne umfangreiche externe Bibliotheken auf den HardMut-Webservice zuzugreifen. Auch die Erstellung von Applikationen aus der gleichen Code-Basis ist wegen der starken Einschränkung der Mobil-Variante kaum möglich. Hinzu kommt, das Bibliotheken wie Java 3D zur Darstellung vom dreidimensionalen Objekten (noch) nicht unter Java FX funktionieren.

OpenLaszlo Die verschiedenen Ausgabeformate DHTML, Flash 7 und Flash 8 machen eine Verwendung im Browser und als Stand-Alone-Anwendung im Flash-Player auf Desktop- oder Mobil-Plattform möglich. Die Kombination von deklarativer Beschreibungs- und funktionaler Programmiersprache ermöglicht die strikte Trennung von Visualisierungs- und Datenverarbeitungs-Komponenten.

Jedoch ist es mit OpenLaszlo allein nicht möglich, 3D-Modelle darzustellen. Auch der Verbindungsaufbau zum Webservice, um Daten zu empfangen war nicht möglich. Die Echtzeit-Verarbeitung des Quellcodes wurde stets mit einem internen Fehler abgebrochen, so dass weiterführende Tests nicht möglich waren.

ActionScript 2 und Adobe Flex 2 Das Ergebnis einer Anwendung, die mittels ActionScript 2 und Flex-Entwicklungsumgebung erstellt wurde, ist eine Flash 9-Datei, die mittels Flash 9- und Flash Lite-Player wiedergegeben werden kann. Damit ist eine Verwendung in verschiedenen Umgebungen möglich. Es existiert ebenfalls mit *Papervision3D* [13] ein 3D-Framework, welches die Darstellung von dreidimensionalen Objekten ermöglicht.

Der Zugriff und die Manipulation lokaler Daten ist aufgrund des Flash-Sicherheitskonzepts, trotz Ausführung im lokalen Flash-Player, nicht möglich. Auch sollte bedacht werden, dass es sich bei ActionScript 2 um ein veraltete Version handelt, die zwar noch häufig eingesetzt, langfristig aber durch die neuere Version ersetzt werden wird.

ActionScript 3, Adobe Flex 3 und Adobe AIR Wie bei ActionScript 2 erzeugt Flex 3 Flash 9-kompatiblen Code, der aufgrund der grundlegend anderen Struktur von ActionScript 3-Klassen nicht in einem Flash-Lite-Player ausgeführt werden kann. Ledig-

lich mobile Geräte, die einen vollwertigen Flash-Player besitzen sind somit in der Lage, solche Anwendungen auszuführen.

Mittels Adobe AIR ist eine Ausführung als native Anwendung mit allen Zugriffsmöglichkeiten auf das lokale Dateisystem auf einem Desktop-Rechner, selbst ohne installierten Flash-Player, möglich. Das Papervision3D-Framework zur Darstellung von 3D-Objekten ist ebenfalls für ActionScript 3 verfügbar. Der in der Flex-Entwicklungsumgebung integrierte GUI-Builder ermöglicht ein schnelles und einfaches Gestalten von Benutzerschnittellen, welche, durch die Übersetzung in ActionScript-Klassen, in anderen Programmstrukturen verwendet werden können.

Zusammenfassung Nachdem die einzelnen Vor- und Nachteilen der Programmierumgebungen verdeutlicht werden, sollen diese in folgender Tabelle gegenübergestellt werden und anhand dieser eine Entscheidung für die Entwicklung des Prototypen getroffen werden.

	Desktop/ Mobil	Browser/ Stand-Alone	Einheitliche Code-Basis	2D- und 3D-Darstellung	HardMut-Umgebung
Java FX	+	o	-	o	+
OpenLaszlo	+	o	+	o	-
AS 2 und Flex 2	+	o	+	+	+
AS 3, Flex 3 und Adobe AIR	o	+	+	+	+

Tabelle 6.1: Gegenüberstellung ausgewählter Programmiersprachen und Frameworks

Anhand dieser Gegenüberstellung soll ActionScript 3 mit Verwendung der Flex 3-Entwicklungsumgebung und des Adobe AIR-Toolkit benutzt werden, um den Prototypen zu entwickeln. Trotz Abstrichen bei der Verwendbarkeit auf mobilen Geräten wird diese Variante gewählt, da es sich bei ActionScript Version 2 um eine veraltete und deshalb weniger nachhaltige Technologie handelt.

6.2 Datenhaltungs-Schicht

6.2.1 Konnektoren

6.2.2 Graph

6.2.3 Konnektoren

6.2.4 Externe Dateneingabe

6.3 Anwendungssteuerung

6.3.1 Konfiguration

6.4 Visualisierung

6.4.1 Fassade-Klasse

6.4.2 Benutzerschnittstelle

6.4.3 Visualisierung

Virtueller Raum und Graphen-Darstellung

Graphen-Optimierung

Medien-Visualisierung

Kapitel 7

Evaluation und Demonstration

7.1 Testkriterien

7.2 Demonstration der Funktionalität

7.3 Auswertung der Ergebnisse

Kapitel 8

Zusammenfassung

Literaturverzeichnis

[BBC97] P. J. Brown, J. D. Bovey, and X. Chen. Context-aware applications: From the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64, 1997.

[Che99] Chomei Chen. *Information Visualisation and Virtual Environments*. Springer-Verlag London Limited, London, UK, 1999.

[CSS07] Ronald Chung, David Sundaram, and Ananth Srinivasan. Integrated personal recommender systems. In *ICEC '07: Proceedings of the ninth international conference on Electronic commerce*, pages 65–74, New York, NY, USA, 2007. ACM.

[DA08] Anind K. Dey and Gregory D. Abowd. Toward a better understanding of context and context-awareness. Technical report, Graphics, Visualization and Usability Center and College of Computing Georgia Institute of Technology, Atlanta, GA, USA, 2008.

[Gib79] J. J. Gibson. *The ecological approach to visual perception*. Lawrence Erlbaum, Boston, MA, USA, 1979.

[HQW06] Gerhard Heyer, Uwe Quasthoff, and Thomas Wittig. *Text mining Wissensrohstoff Text*. W3L-Verlag, Herdecke, 2006.

[Leo05] Edda Leopold. On semantic spaces. *Zeitschrift für Computerlinguistik und Sprachtechnologie*, 20(1):63–86, 2005.

[May04] Rene Mayrhofer. *An architecture for context prediction*. Trauner, 2004.

[Pir07] Peter Pirolli. *Information foraging*. Oxford University Press, New York, NY, USA, 2007.

[SAW94] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. *First International Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994.

[ST97] B. Schilit and M. Theimer. Disseminating active map information to mibile hosts. *IEEE Network*, 8(5):22–32, 1997.

[War00] Colin Ware. *Information Visualization.* Academic Press, San Diego, CA, USA, 2000.

[ZHP+06] Michelle X. Zhou, Keith Houck, Shimei Pan, James Shaw, Vikram Aggarwal, and Zhen Wen. Enabling context-sensitive information seeking. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 116–123, New York, NY, USA, 2006. ACM.

Internet-Quellen

[1] IKM/sysedv Technical University Berlin. Commetrix - dynamic network visualization software. http://www.commetrix.de/, 2006. Zugriffsdatum: 07.05.2009.

[2] Google Inc. Google deutschland. http://www.google.de, 2009. Zugriffsdatum: 07.05.2009.

[3] Viewzi Inc. Viewzi - photo tag cloud. http://www.viewzi.com/search/whitevoid-photocloud/viewzi, 2009. Zugriffsdatum: 07.05.2009.

[4] Yahoo! Inc. Flickr fotosharing. http://www.flickr.com, 2009. Zugriffsdatum: 07.05.2009.

[5] Wolfram Alpha LLC. Wolfram—alpha. http://www.wolframalpha.com/, 2009. Zugriffsdatum: 20.04.2009.

[6] University Of Maryland. Pad++: Zoomable user interfaces (zuis). http://www.cs.umd.edu/hcil/pad++/, 1998. Zugriffsdatum: 16.04.2009.

[7] FMGB Guggenheim Bilbao Museoa. Guggenheim bilbao - virtual tour. http://www.guggenheim-bilbao.es/visita_virtual/visita_virtual.php?idioma=en, 2007. Zugriffsdatum: 08.05.2009.

[8] The Museum of Modern Art. Moma — introduction to the exhibition. http://www.moma.org/explore/multimedia/interactives/, 2009. Zugriffsdatum: 07.05.2009.

[9] University of Sydney. Gallery - visualization of clustered graph in 3d. http://www.cs.usyd.edu.au/%7Evisual/valacon/gallery/C3D/, 2004. Zugriffsdatum: 04.05.2009.

[10] MACS project. Multi-sensory autonomous cognitive systems. http://www.macs-eu.org/index.html, March 2009. Zugriffsdatum: 04.04.2009.

[11] Inc. Sun Microsystems. Java. http://www.java.com/de/, 2009. Zugriffsdatum: 16.05.2009.

[12] Universität Tübingen. Stts tagset. `http://www.sfs.uni-tuebingen.de/Elwis/stts/stts.html`, 1995. Zugriffsdatum: 21.04.2009.

[13] Carlos Ulloa, John Grden, Ralph Hauwert, Tim Knip, and Andy Zupko. Papervision3d. `http://blog.papervision3d.org/`, 2009. Zugriffsdatum: 16.05.2009.

[14] Forschungsgruppe Informations und Kommunikationsanwendungen. Aufbau eines mobilen jüdischen museums und entwicklung mobiler, multimedialer museumsinformationssysteme. `http://inka.fhtw-berlin.de/ammely/`, 2008. Zugriffsdatum: 15.05.2009.

[15] Forschungsgruppe Informations und Kommunikationsanwendungen. Hardmut - hardware und multimediatechnik zur entwicklung eines mobilen museums. `http://inka.fhtw-berlin.de/hardmut/archiv`, 2008. Zugriffsdatum: 15.05.2009.

[16] Forschungsgruppe Informations und Kommunikationsanwendungen. Hardmut ii - hardware und multimediatechnik zur entwicklung eines mobilen museums. `http://inka.fhtw-berlin.de/hardmut/index`, 2008. Zugriffsdatum: 09.05.2009.

[17] Stanford University. H3: Laying out large directed graphs in 3d hyperbolic space. `http://graphics.stanford.edu/papers/h3/`, 2004. Zugriffsdatum: 24.03.2009.

[18] Friedemann Schulz von Thun. Das kommunikationsquadrat. `http://www.schulz-von-thun.de/mod-komquad.html`, 2004. Zugriffsdatum: 17.05.2009.

Abbildungsverzeichnis

Listings

Anhang A

Diagramme

A.1 Graphen-Layout

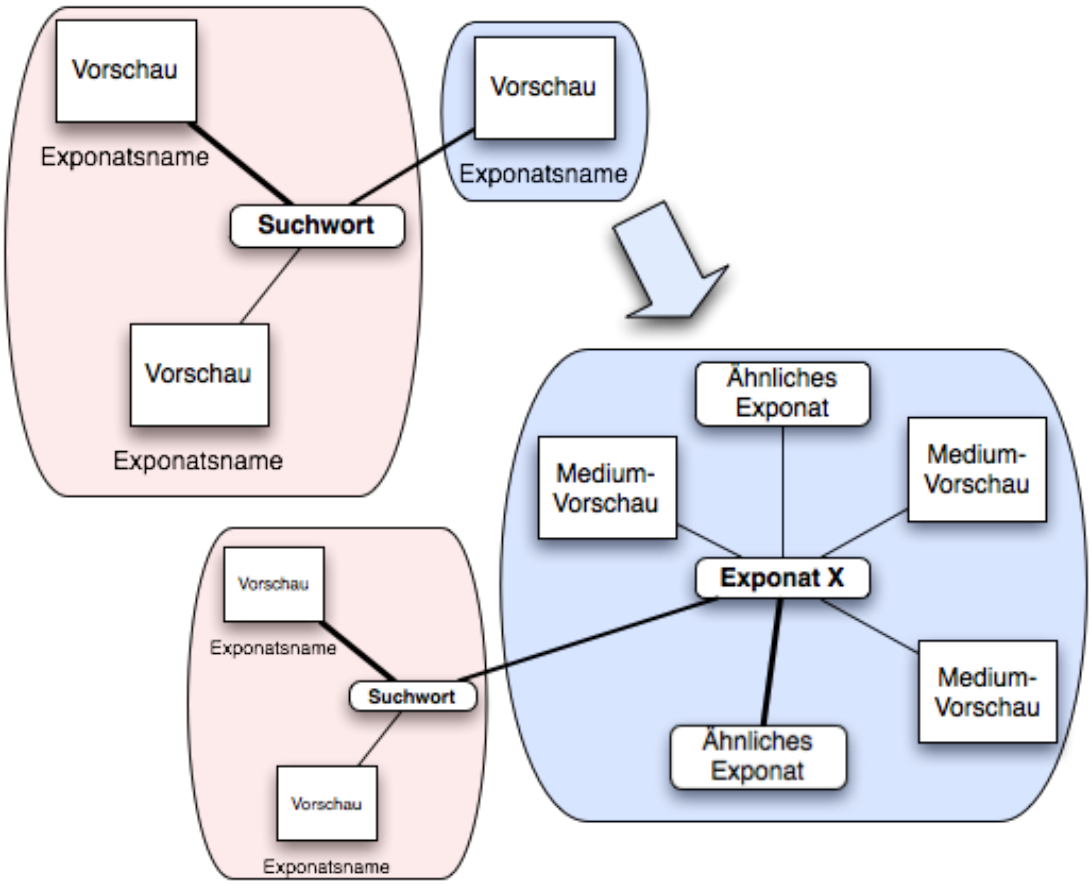

Abbildung A.1: Arten von Knoten im Graphen und deren Anordnung

A.2 Use-Case-Diagramm

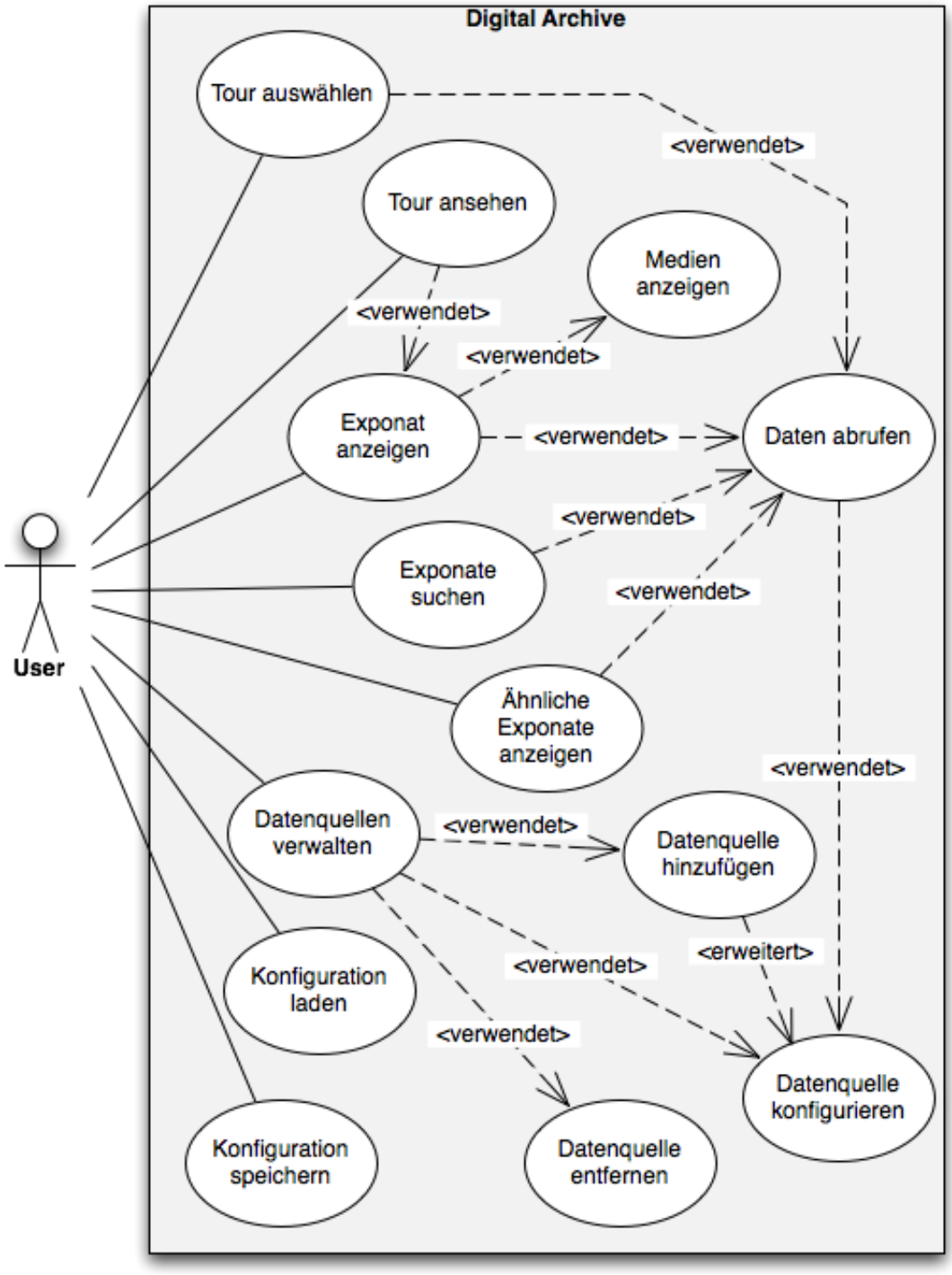

Abbildung A.2: Anwendungsfälle des Prototypen

A.3 Datentypen-ERM

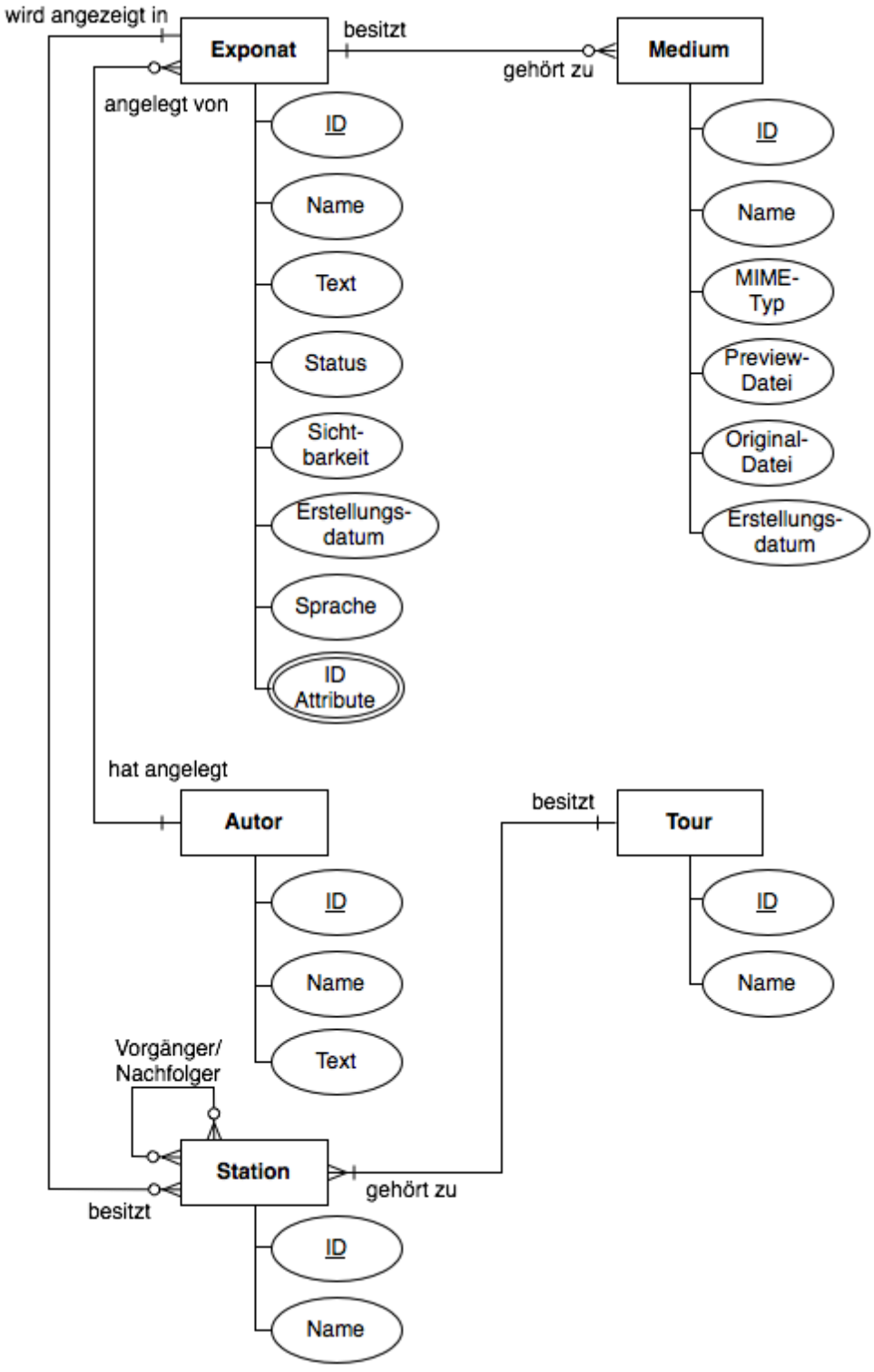

Abbildung A.3: Relationen und Attribute der verwendeten Datentypen

A.4 Flussdiagramm Anwendungsstart

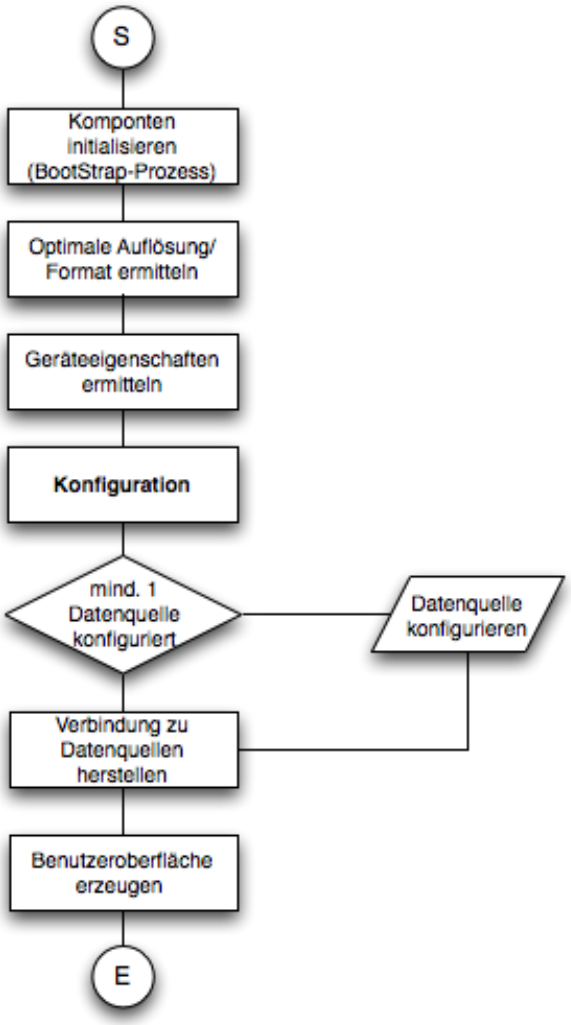

Abbildung A.4: Vom Controller gesteuerte Prozesskette beim Anwendungsstart

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, der 17. Mai 2009

Michael Witt