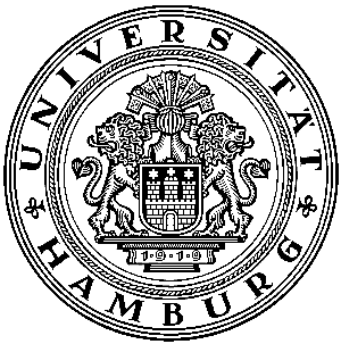

Universität Hamburg
FB Informatik

Generische Dienste zur Präsentation, Nutzung und Verwaltung von Nachschlagewerken in Rechnernetzen

Diplomarbeit

eingereicht bei

Prof. Dr. Joachim W. Schmidt
Arbeitsbereich Softwaresysteme
Technische Universität Harburg

und

Prof. Dr. Christopher Habel
Arbeitsbereich Wissens- und
Sprachverarbeitung
Universität Hamburg

von
Ilona Hachmann

10. Dezember 1997

Danksagung

Viele Menschen standen mir bei der Erstellung dieser Arbeit zur Seite. Ihnen gebührt mein Dank an dieser Stelle.

Mein spezieller Dank gilt den Herren Prof. Dr. Joachim W. Schmidt und Prof. Dr. Christopher Habel, für die Unterstützung bei der Durchführung dieser Arbeit.

Frau Claudia Niederée danke ich für ihre Anregungen zu diesem Thema und stete Geduld bei der Beantwortung meiner Fragen. Viele Gespräche, auch über die Informatik hinaus, waren förderlich für ein gutes Arbeitsklima.

Herrn Gerald Schröder danke ich für seine Hinweise zu meinem Thema, die, bedingt durch sein Forschungsfeld, aus einer ganz anderen Sichtweise entstanden sind.

Besonderer Dank gilt meiner Mutter, die mir meinen Lebensweg ermöglicht und mich zu jeder Zeit unterstützt hat.

Hamburg, 10. Dezember 1997 Ilona Hachmann

Inhaltsverzeichnis

1 Einleitung und Motivation **1**
- 1.1 Einordnung der Arbeit . . . 2
- 1.2 Ziel der Arbeit . . . 2
- 1.3 Struktur der Arbeit . . . 3

2 Konventionelle und digitale Nachschlagewerke **5**
- 2.1 Einführung in die Terminologie . . . 5
- 2.2 Medien im Vergleich . . . 7
  - 2.2.1 Das Buch . . . 9
  - 2.2.2 Die CD-ROM . . . 9
  - 2.2.3 Das Internet . . . 10
  - 2.2.4 Fazit . . . 10
- 2.3 Konventionelle Nachschlagewerke . . . 10
- 2.4 Digitale Nachschlagewerke . . . 13
  - 2.4.1 Nachschlagewerke auf CD-ROM . . . 13
  - 2.4.2 Nachschlagewerke im Internet . . . 15
- 2.5 Ein Klassifizierungsschema für Nachschlagewerke . . . 16

3 Entwicklungsumgebungen und Sprachen **23**
- 3.1 Java . . . 23
  - 3.1.1 Ein Überblick . . . 23
  - 3.1.2 Einbinden von Java-Programmen . . . 26
- 3.2 Relevante Programmiersprachenkonzepte . . . 27
  - 3.2.1 Verteilte Objekte . . . 27
  - 3.2.2 Innere Klassen . . . 29
  - 3.2.3 Ereignisbehandlung . . . 29
- 3.3 Konzepte zur Entwicklung . . . 31
  - 3.3.1 Wiederverwendbarer objektorientierter Entwurf . . . 31
  - 3.3.2 Verwendung von Entwurfsmustern . . . 32
  - 3.3.3 Client/ Server-Technologie . . . 34

4 Generische Dienste für Nachschlagewerke **37**
- 4.1 Lebenszyklus digitaler Nachschlagewerke . . . 37
- 4.2 Entwurf einer Architektur . . . 39
- 4.3 Anforderungen an eine generische Dienstbibliothek . . . 39
  - 4.3.1 Generische Unterstützung der Datenakquisition . . . 39

4.3.2 Generische Dienste zur Repräsentation und Zugriff 41
4.3.3 Generische Präsentation- und Visualisierungsdienste 44
4.3.4 Generische Speicherungsdienste 49
4.3.5 Generische Zugriffsunterstützung und Nutzung 50
4.3.6 Generische Integrationsdienste 52
4.4 Mehrwertbildende Dienste 53

5 Realisierung konkreter Nachschlagewerke 55
5.1 Beschreibung der Anwendung 55
5.2 Realisierung 55
5.2.1 Datenakquisition 56
5.2.2 Repräsentation und Zugriff 60
5.2.3 Präsentation und Visualisierung 64
5.2.4 Speicherungsdienste 68
5.2.5 Zugriffsunterstützung und Nutzung 71
5.2.6 Integration 72
5.3 Verteilung der Anwendung 75
5.4 Beschreibung der prototypischen Realisierung 77

6 Zusammenfassung und Ausblick 79
6.1 Zusammenfassung der Ergebnisse 79
6.2 Erfahrungen mit Java 80
6.2.1 Positive Erfahrungen 80
6.2.2 Einschränkungen 81
6.3 Kommerzielle Relevanz 81
6.4 Ausblick 82

A Verwendete Notation 85

B Ausgewählte Schnittstellen 87

Literaturverzeichnis 93

Abbildungsverzeichnis

2.1 Fazit des Fragenkatalogs 11
2.2 Japanisches Zeichen für *nihon* 12
2.3 Bertelsmann Universal-Lexikon Oberfläche 14
2.4 Startseite des Meyers Lexikon – Das Wissen A-Z 15
2.5 Suchformular des Meyers Lexikon – Das Wissen A-Z 16
2.6 Startseite des Beilstein *Dictionary* 17
2.7 Suchformular des Beilstein *Dictionary* 18
2.8 Ausschnitt der Startseite des BerlinINFO 18
2.9 Folgeseite des Berlin-Info 19
2.10 Mediale Ausrichtung der Zugriffsstruktur eines Wörterbuchs 20
2.11 Mediale Ausrichtung der Zugriffsstruktur eines Bestimmungsbuches der Biologie 20
2.12 Mediale Ausrichtung der Zugriffsstruktur des Stadtplanes BerlinINFO 20
2.13 Mehrstufige Zugriffsstruktur eines Bestimmungsbuches der Biologie 22

3.1 Erstellung und Interpretation von Java Programmen (angelehnt an [LePe 96]) 24
3.2 Formen von Polymorphismus (nach [CaWe 85]) 25
3.3 Kommunikation zwischen Proxyobjekt und realen Objekt mittels ORB (nach [Hran 97]) 28
3.4 Beispielcode für innere Klassen und Ereignisbehandlung 30
3.5 Demonstration des Beispielcodes 31
3.6 Kapselung von ähnlichen Algorithmen: Strategiemuster 33
3.7 Förderung der losen Kopplung: Vermittlermuster 34
3.8 Client/ Server Interaktion nach [Dadam 96] 35
3.9 Komponenten einer Applikation nach [Dadam 96] 35
3.10 Möglichkeiten der Zerlegung einer Applikation nach [Dadam 96] 36

4.1 Lebenszyklus von Nachschlagewerken 38
4.2 Die Dienstgruppen 38
4.3 Benutzung eines Nachschlagewerks via Internet 40
4.4 Architektur der Dienste 41
4.5 Datenakquisition 42
4.6 Interne und konzeptuelle Ebene 43
4.7 Zugriff auf ein Nachschlagewerk 43
4.8 Grundelemente eines Bildschirm(ausschnitt)s in OMT-Notation 44
4.9 Grundelemente eines Bildschirm(ausschnitt)s nach [Stary 96] 45
4.10 Grundelemente eines Bildschirm(ausschnitt)s nach [Bons 96] 45
4.11 Grundelemente eines Bildschirm(ausschnitt)s nach [Bons 96] 46

4.12 Ablauf der Benutzung eines Nachschlagewerks 47
4.13 Ablauf bei Benutzung eines Nachschlagewerks mit mehrstufiger Zugriffsstruktur 48
4.14 Graphische Aufbereitung eines Alphabets 48
4.15 Interaktiver Prozeß der Nutzung eines Nachschlagewerks 50
4.16 Integration eines Nachschlagewerks in eine HTML-Seite 53
4.17 Personalisierter Index 54

5.1 Architektur der Dienste aus Sicht des Systems 56
5.2 Anforderungen der Datenakquisition 57
5.3 Schnittstelle und Klassen zum Dateneinlesen 58
5.4 Verbindende, generische Klasse `Acquisition` 59
5.5 Graphische Unterstützung bei der Datenakquisition 60
5.6 Die Schnittstellen `Datastructure` 61
5.7 Eine Indexstruktur aus Indexknoten 61
5.8 Die Klassen `IndexStructure` und `IndexNode` 62
5.9 Nicht gültige Indexstruktur 63
5.10 Ungültige Indexstruktur 63
5.11 Nachschlagewerk mit den drei Grundelementen 64
5.12 Die abstrakte Klasse `NavigationDescriptorLayout` 65
5.13 Die Klasse `Layout` 66
5.14 Die Schnittstelle `OrientationArea` 67
5.15 Die Klasse `Arrange` 68
5.16 Graphische Unterstützung bei der Positionsfestlegung 69
5.17 Die Schnittstelle `Store` 70
5.18 Baukastenprinzip 71
5.19 Orte der Integration der administrativen Dienste in einem Nachschlagewerk . 72
5.20 Die Schnittstelle `PayService` 73
5.21 Mögliche implementierende Klassen der Schnittstelle `PayService` 73
5.22 CGI-Skript zur Erzeugung eines Applets mit zuvor ausgelesen Parametern . . 75
5.23 Drei Ebenen der Anwendung 76
5.24 Entstehungszyklus eines Nachschlagewerks 77
5.25 Graphische Unterstützung bei der Layoutauswahl 78

6.1 Kommunikation zwischen JDBC und Datenbank 83

A.1 Abstrakte und konkrete Klasse sowie eine Schnittstelle 85
A.2 Beziehungen zwischen Klassen 86

Kapitel 1

Einleitung und Motivation

Durch den Verfall der Preise in der IT-Industrie erfolgt ein zunehmender Einsatz von Rechnern in allen Bereichen. Immer mehr Arbeitsprozesse finden teilweise oder vollständig am Computer statt [Böhle 97]. Als derzeit stetig wachsende Informationsquelle bietet das Internet für den wissenschaftlichen als auch für den kommerziellen Bereich zahlreiche Möglichkeiten aus einem breiten Spektrum von Themen Informationen zu sammeln und individuell zusammenzustellen. In einem Umfeld, das von verstärkter Nutzung digitaler Möglichkeiten geprägt ist, besteht nicht nur der Bedarf digitale Dokumente zu bearbeiten und zu verwalten, sondern auch Hilfsmittel für den jeweiligen Arbeitsprozeß ohne Medienwechsel, z.B. zu einem Buch, verfügbar zu haben.

Nachschlagewerke sind wichtige Hilfsmittel für das wissenschaftliche und kommerzielle Arbeiten: Sei es, um ein Synonym, ein Akronym, eine Translation oder bestimmte Einzelheiten über Personen und Gegenstände herauszufinden. Dies zeigt einen Bedarf an Nachschlagewerken in Rechnernetzen und damit ein Bedarf an Mitteln zur Realisierung dieser Aufgabe. Ein Nachschlagewerk in elektronischer Form reduziert den Aufwand beim Suchen eines bestimmten Wortes oder Sachverhalts und steigert den Informationsgehalt durch die Möglichkeit der Verknüpfung mit angrenzenden oder verwandten Themen [MüNe 96]. Einige Nachschlagewerke sind bereits in elektronischer Form (CD-ROM, Internet) realisiert (z.B. [BerlinINFO], [Meyers Lexikon]). Immer mehr Nachschlagewerke-Anbieter (bzw. die Verlage, z.B. Saur-Verlag) denken über eine Internetpräsenz nach, die nicht nur einen Index zur Verfügung stellt, sondern ganze Werke. Ein hemmender Grund sind unzulängliche bzw. noch nicht ausgereifte Abrechnungsmechanismen.

Wie kann nun ein Anbieter eines Nachschlagewerks die Inhalte im Netz bereitstellen? Wie können Benutzer auf die Nachschlagewerke zugreifen, um ihre Informationsbedarfe zu befriedigen? Für diese Zwecke werden Dienste benötigt, die einem Nachschlagewerke-Anbieter Möglichkeiten bieten seine Nachschlagewerkinhalte in geeigneter Weise zu erfassen und zu präsentieren. Darüber hinaus muß ein Anbieter verschiedene Optionen zur Nutzung seines Nachschlagewerks bereitstellen. Durch die Verwendung des Internets als Publikationsform entsteht ein Wunsch nach Mehrwertdiensten, z.B. Integration in andere Programme und andere Nachschlagewerke und Integration von anderen entwickelte Dienste (z.B. Abrechnungsmechanismen).

1.1 Einordnung der Arbeit

Ein neues Forschungsfeld der Datenbanken und Informationssyteme sind digitale Bibliotheken (*digital libraries*). Nachschlagewerke bilden einen Teilaspekt dieses großen Forschungsfeldes. Digitale Bibliotheken bestehen, wie konventionelle Bibliotheken, aus einer Dokumentensammlung und einer Technologie zur Verwaltung dieser Sammlung. Diese Dokumentensammlung beinhaltet multimediale Information in Form von Bildern, Daten, Texten, Bildfolgen und gesprochene Sprache [Red 95]. Eine digitale Bibliothek ist in der Lage, umfangreiche multimediale Dokumentensammlungen zu repräsentieren und systematisch zu organisieren. Weiter ist es möglich, diese Dokumentensammlung einer großen und eventuell örtlich verteilten Benutzergruppe kostengünstig und effizient anzubieten [Gershon et al. 95]. In einer Arbeitsumgebung, in der eine digitale Bibliothek benutzt wird, bieten Nachschlagewerke in digitaler Form eine effektive Unterstützung typischer Arbeitsprozesse.
Im September 1996 startete das interdisziplinäre Projekt REGIERUNGSBANK mit der Forschungsstelle Politische Ikonographie des Kunstgeschichtlichen Seminars und dem Arbeitsbereich Softwaresysteme (AB STS) der TUHH[1]. Die digitale Bibliothek REGIERUNGSBANK verfolgt das Ziel, digitale Bild-, Text- und Tondokumente zu Protagonisten der Politik aus Vergangenheit und Gegenwart nach Personen, Kontexten und politischen Darstellungs- und Handlungsmustern zugänglich zu machen [SSNM 97]. Dabei ist eine Forderung an das Projekt die Erschließung von Nachschlagewerken. Ein wichtiger Aspekt dabei sind die unterstützten Zugriffsmechanismen. Wie in einem Papiermedium auch, soll eine gezielte Suche über einen bestimmten Suchbegriff ebenso möglich sein, wie ungezieltes Durchblättern.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es, Konzepte zur Realisierung generischer Dienste zur Präsentation, Nutzung und Verwaltung von Nachschlagewerken in Rechnernetzen zu entwickeln. Eine Untersuchung von verschiedenen Werken (konventioneller und digitaler Art) unterstützt die Erkennung und Bildung einer Abstraktion. Darauf aufbauend wird eine generische Dienstbibliothek entworfen und ausgewählte Dienste exemplarisch realisiert. Die Generizität der Dienste gewährleistet breite Einsatzmöglichkeiten und eine hohe Wiederverwendbarkeit der Dienstbibliothek. Nicht Ziel dieser Arbeit ist es, anspruchsvoll gestaltete Oberflächen für elektronische Nachschlagewerke zu entwerfen und zu realisieren. Diese Aufgabe ist von kommerziellen Produkten bereits hinreichend realisiert. Die Erstellung eines Nachschlagewerks mit der in dieser Arbeit entwickelten prototypischen Bibliothek kann in Bezug auf die Punkte graphisches Erscheinungsbild und Suchmöglichkeiten mit kommerziellen Produkten nicht konkurrieren.

Die Ziele dieser Arbeit sind also:

- Erkennung und Darstellung der Strukturen von Nachschlagewerken aus verschiedenen Themenbereichen und zugrundeliegenden Medien.
- Entwurf und Vorstellung einer generischen Dienstbibliothek.
- Realisierung und Bewertung exemplarisch ausgewählter Dienste der prototypischen generischen Dienstbibliothek.

[1] der AB STS der TUHH ging 1996 aus dem AB Datenbanken und Informationssysteme (DBIS) des Fachbereichs Informatik der Uni HH hervor.

1.3 Struktur der Arbeit

Diese Arbeit gliedert sich in die im folgenden übersichtsartig vorgestellten Kapitel:

Konventionelle und digitale Nachschlagewerke: Das Kapitel 2 nimmt eine Bestimmung der Begriffe vor. Insbesondere wird der für diese Arbeit zentrale Begriff des Nachschlagewerks definiert. So steht z.B. am Anfang eines Artikels über Nachschlagewerke auf CD-ROM: „In einem Nachschlagewerk will man meist nicht schmökern, sondern gezielt suchen" [Schu 97].
Weiter wird untersucht, welche Charakteristiken die Informationsaustauschmedien Buch, CD und Internet aufweisen. Dazu werden verschiedene Werke aus diesen verschiedenen Mediengattungen betrachtet. Das Kapitel schließt mit einer klassifizierenden Zusammenfassung der untersuchten Werke.

Entwicklungsumgebungen und Sprachen: Kapitel 3 führt die für diese Arbeit relevanten Aspekte der hier verwendeten Programmiersprache Java ein. Dies betrifft insbesondere die neuen Konzepte der Version 1.1.

Weiter werden in diesem Kapitel Konzepte vorgestellt, die bei dem Entwurf der Dienstbibliothek hilfreich sind. Hierzu zählen generelle Prinzipien der objektorientierten Programmierung sowie verschiedene Entwurfsmuster, die im Implementationsteil Verwendung finden und eine Einführung in relevante Aspekte der Client/Server-Technologie.

Generische Dienste für Nachschlagewerke: Das Kapitel 4 betrachtet die Anforderungen, die sich ergeben, wenn Nachschlagewerke in Rechnernetze gestellt werden. Aus der Untersuchung des Lebenszyklus eines Nachschlagewerks ergeben sich eine Reihe von Dienstgruppen. Diese Dienstgruppen werden im Rest des Kapitels ausgearbeitet und bilden eine Grundlage für die zu entwickelnde Dienstbibliothek.

Realisierung konkreter Nachschlagewerke: Das 5. Kapitel greift die in Kapitel 4 formulierten Anforderungen auf und stellt den Entwurf einer generischen Dienstbibliothek sowie die exemplarische Realisierung ausgewählter Dienste vor. Die in Kapitel 2 vorgestellten Werke werden zur Demonstration und zum Testen der Dienste herangezogen.

Zusammenfassung und Ausblick: Das Kapitel 6 bildet den Schluß dieser Arbeit. Hier werden die wichtigsten Eigenschaften der vorgestellten Dienste und Konzepte zusammengefaßt sowie Erfahrungen mit der Programmiersprache Java ausgewertet. Es wird die kommerzielle Relevanz dieser Arbeit diskutiert. Ein Ausblick auf mögliche Anschlußarbeiten (und offene Punkte) beschließt dieses Kapitel und damit diese Arbeit.

Kapitel 2

Konventionelle und digitale Nachschlagewerke

Dieses Kapitel stellt eine grundsätzliche Einführung über Nachschlagewerke dar. Es dient dieser Arbeit als Ansatzpunkt, um Anforderungen und Strukturen von Nachschlagewerken zu identifizieren und aufzustellen. Dazu wird in Absatz 2.1 eine Einführung in die Terminologie und eine Vorstellung relevanter Nachschlagewerkearten vorgenommen. In Absatz 2.2 werden die Medien Buch, CD und Internet in Bezug auf verschiedene Merkmale hin analysiert. Die darauf folgenden Absätze 2.3 und 2.4 betrachten Nachschlagewerke, die auf diesen Medien realisiert sind. Dieses Kapitel schließt mit einer klassifizierenden Zusammenfassung der untersuchten Werke (Absatz 2.5).

2.1 Einführung in die Terminologie

Dieser Abschnitt führt in die Terminologie ein, die für die gesamte Arbeit von Bedeutung ist. Begonnen wird mit der Klärung und Abgrenzung des Begriffs Nachschlagewerk und den damit zusammenhängenden Begriffen Stichwort und Eintrag. Anschließend wird eine Kurzcharakteristika verschiedener Werke vorgestellt.

Was ist ein Nachschlagewerk?

Die Definitionen in der Literatur sind sich sehr ähnlich. Es seien hier exemplarisch zwei vorgestellt:

> Schriftwerk, das in übersichtlicher, meist alphabetischer Anordnung eine schnelle Orientierung über einen Gegenstand ermöglicht (Lexikon, Wörterbuch) [Wahr].

> Buch, das in übersichtlicher, meist alphabetischer Anordnung der schnellen Orientierung dient. [DUDEN].

Ein Nachschlagewerk besteht aus vielen Stichwörtern. Ein *Stichwort* ist ein "Wort, das erklärt wird [...]" [Wahr, S.3424]. Die Erklärung des Stichwortes erfolgt in einem *Eintrag*. Wichtig bei einem Nachschlagewerk sind die übersichtliche Anordnung und die Möglichkeit der schnellen Orientierung beim Zugriff.

Was ist kein Nachschlagewerk?

Eine eindeutige Trennung ist nicht möglich. Eine Abgrenzung läßt sich jedoch durch die Art der Benutzung eines Nachschlagewerks festlegen. So zählen z.B. Werke, die ein sequentielles Lesen der Kapitel voraussetzen, in dieser Arbeit nicht zu den Nachschlagewerken. Hierzu gehören z.B. *Einführungen in bestimmte Themen.* Solche Werke können jedoch Nachschlagewerkstrukturen besitzen, z.B. ein Register am Ende des Buches, über welches leicht zum Stichwort bzw. Eintrag zu gelangen ist.

Welche Arten gibt es?

Es gibt eine Vielzahl von Arten von Nachschlagewerken. Folgende Liste enthält die Nachschlagewerke, die für diese Arbeit relevant sind:

- Adreßbücher
- Bild- und Tafelwerke
- Enzyklopädien und Lexika
- Karten und Kartenwerke
- Kursbücher und Fahrpläne
- Wörterbücher
- Referenzbücher

Im folgenden wird für die relevanten Nachschlagewerke dieser Arbeit eine Kurzcharakteristika vorgenommen.

Adreßbücher

Ein Adreßbuch ist ein "alphabetisches Verzeichnis der Bewohner eines örtlich begrenzten Kreises (Stadt, Land) oder der Angehörigen eines bestimmten Personenkreises, z.B. Berufs- oder Gewerbezweiges" [HeLe 82]. Diese Definition umfaßt eine Reihe von Nachschlagewerken dieses Typs, z.B. Personenverzeichnisse, Telefonbücher.

Bild- und Tafelwerke

In ihnen sind Abbildungen jeder Art vorhanden: Fotographien von Personen, Landschaften, Reproduktion von Kunstwerken, schematische Zeichnungen von Abläufen, zeitgeschichtliche Ereignisse, usw.

Enzyklopädien und Lexika

In der Umgangssprache werden die Termini "Enzyklopädie" und "Lexika" synonym verwendet. Enzyklopädien und Lexika enthalten Informationen über Sachen, Personen, Räumen und Epochen. Im Gegensatz zum Wörterbuch (siehe unten) überwiegt die inhaltlich-sachliche Erläuterung. Zu den Textartikeln kommen noch bibliographische Teile, graphische Abbildungen (z.B. Tabellen, Übersichten), Bilder, Pläne und Karten.

Karten und Kartenwerke

Karten und Kartenwerke vermitteln Informationen durch graphische Darstellung, die vielfach von Texten (Beschriftung, Unterschriften, erklärende oder zusammenfassende Begleitartikel) untersützt und ergänzt wird. Zu Karten und Kartenwerke zählen Atlanten (z.B. Welt-, National- oder Heimatatlanten), Stadtpläne, Wanderkarten, Spezialatlanten (z.B. Geschichts-, geologischer- oder meereskundlicher Atlas) als auch Abbildungswerke aus der Anatomie, Botanik und Zoologie.

Kursbücher und Fahrpläne

Kursbücher und Fahrpläne geben Auskunft über Streckennetze, Abfahrt- und Ankunftszeiten, usw. Sie sind sofort als Nachschlagewerk zu erkennen.

Wörterbücher

Ein Wörterbücher ist ein in der Regel alphabetisch geordnetes Werk, eine Sammlung von Wörtern mit sprachlichen oder auch sachlichen Erläuterungen. Es umfaßt den Wortschatz einer oder mehrerer Sprachen bzw. die Begriffe eines Fach- oder Sachgebiets. In Wörterbüchern überwiegt die sprachliche Erläuterung. Neben Wörterbüchern, in denen der gesamte Wortschatz einer Sprache dokumentiert ist, gibt es spezifische Wörterbücher wie Rechtschreibe- und Aussprachewörterbücher, Synonymwörterbücher, Fremdwörterbücher, Abkürzungs- und Symbolwörterbücher, etc.

Referenzbücher

Referenzbücher bezeichnen Werke, die einen schnellen Überblick über bestimmte, gut aufzulistende Themen geben. Eine Klassenhierarchie einer objektorientierten Sprache läßt sich z.B. gut als Referenz realisieren: Alle Klassen sind alphabetisch nacheinander aufgeführt.

2.2 Medien im Vergleich

Nachschlagewerke sind längst nicht mehr nur in Buchform zu finden, wie in den obigen Definitionen angegeben. So liegen bereits viele Versionen in digitaler Form vor. An dieser Stelle wird der Begriff des *Mediums* wichtig. Ein Nachschlagewerk kann auf verschiedenen Medien realisiert sein und verschiedene Medien enthalten. Bei genauerer Betrachtung des vorigen Satzes ist die Mehrdeutigkeit des Begriffs Medium festzustellen. Daher folgt eine erste Definition des Begriffs *Medium* nach [Stein 93]:

> "Als Medium bezeichnet man im allgemeinen ein Mittel zur Verbreitung und Darstellung von Informationen. Beispiele eines Mediums sind Text, Graphik, Bild, Sprache, Geräusch und Musik."

Nach diesem Verständnis von Medium lassen sich auch Wasser und Atmosphäre dazuzählen. [Stein 93] verweist deshalb auf eine differenziertere Bestimmung des Begriffs nach verschiedenen Kriterien von [MHEG 93]. Dabei wird zwischen Perzeptions-, Repräsentation-, Präsentations-, Speicher-, Übertragungs- und Informationsaustauschmedien unterschieden. Für diese Arbeit sind die Perzeptions- und Informationsaustauschmedien interessant.

Perzeptionsmedien zielen auf die humanen Sinnesorgane ab. Als Frage formuliert: Wie nimmt der Mensch die Information auf? Primär wird hier zwischen Sehen und Hören unterschieden. Visuelle Medien umfassen Text, Einzelbild, Animation. Zu den auditiven Medien zählen Musik, Geräusch und Sprache.

Informationsaustauschmedien beinhalten alle Datenträger, die zur Übertragung von Informationen verwendet werden. Als Frage formuliert: Welcher Informationsträger wird für den Austausch von Informationen zwischen verschiedenen Orten verwendet? In diese Gruppe gehören alle Speicher- und Übertragungsmedien. Medien zum Informationsaustausch können z.B. sein: Mikrofilm, Papier, Diskette, CD-ROM, Koaxialkabel, Glasfaser, Rechnernetze[1].

Perzeptionsmedien werden bezüglich der Dimension Zeit in zwei Gruppen aufgeteilt:

- zeitunabhängige (oder diskrete) Medien und
- zeitabhängige (oder kontinuierliche) Medien.

Medien wie Text und Graphik sind diskrete Medien. Die Gültigkeit und Richtigkeit der Daten hängt nicht von einer Zeitbedingung ab. Die Werte von Medien wie Ton und Animation verändern sich über die Zeit hinweg, hier hängt die Gültigkeit und Richtigkeit der Daten von einer Zeitbedingung ab.

An dieser Stelle werden prinzipielle Ansatzpunkte für eine kritische Bewertung der Informationsaustauschmedien Papier, CD-ROM und Internet formuliert. Diese Punkte, die als Fragenkatalog formuliert sind, bilden die Grundlage für die Bewertung von Nachschlagewerken, die auf diesen Medien realisiert sind (siehe Abschnitte 2.2.1, 2.2.2 und 2.2.3).

- Orientierungslosigkeit: Wie schnell kann ein Benutzer die Übersicht in dem verwendeten Medium verlieren?
- Navigation: Welche und wieviele Bedienungsmodalitäten stehen dem Benutzer zur Verfügung?
- Bedienung: Wie kompliziert gestaltet sich die Bedienung für den Benutzer?
- Technische Hilfsmittel: Ist der Einsatz technischer Hilfsmittel erforderlich?
- Unterstützte Perzeptionsmedien: Welche Perzeptionsmedien können eingesetzt werden?
- Kosten: Wie und wann ergeben sich Kosten?
- Erweiterbarkeit: Welche Möglichkeiten gibt es, das Medium um besondere Neuerungen zu erweitern?
- Aktualität: Wie lang sind die Aktualisierungszyklen?
- Zugang für Benutzer: Wieviele Benutzer können dieses Medium gleichzeitig nutzen? Welche Voraussetzungen müssen erfüllt werden?
- Kapazität: Wieviel Speichervermögen besitzt das Medium?
- Platzbedarf: Wieviel Platz (physikalisch gesehen) benötigt dieses Medium?

[1] Ein möglicher Zusammenschluß von Rechnernetzen ist das Internet. Im folgenden wird der Begriff Internet synonym zu Rechnernetz benutzt.

2.2.1 Das Buch

In einem Buch verliert ein Benutzer die Orientierung relativ selten, da es nur zwei Möglichkeiten gibt sich durch ein Buch zu bewegen: Vor- oder Zurückblättern. Die Anzahl der Bedienungsmodalitäten ist abhängig von der Art des Nachschlagewerks. Im allgemeinen sind jedoch Inhaltsverzeichnisse und Register anzutreffen. Weiter ist einfaches Blättern ebenfalls eine Bedienungsmodalität. Durch die leichte Orientierung und recht geringe Anzahl von Navigationsmöglichkeiten gestaltet sich die Bedienung für den Benutzer recht einfach. Technische Hilfsmittel sind für die Benutzung eines Buches nicht notwendig. Dies hat den Vorteil, daß das Buch keine (evtl. schnell) veraltende Lesegräte benötigt. Der Einsatz von Perzeptionsmedien kontinuierlicher Art ist nicht möglich. Nachschlagewerke in Buchform sind in der Anschaffung häufig sehr teuer. Mit den Jahren und bei viel Benutzung relativieren sich jedoch diese Kosten. Die Kosten pro Zugriff reduzieren sich. Dieses Medium ist als gedrucktes und gebundenes Buch in sich abgeschlossen. Erweiterungen sind nur durch die Anschaffung weiterer Werke möglich. Der Aufwand für die Erstellung einer Neuauflage eines Werkes beträgt unter Umständen Jahre, der Aktualisierungszyklus ist also recht lang. Ein Buch wird in der Regel nur von einem Leser zur Zeit benutzt. Die Kapazität eines Buches ist durch physikalische Eigenschaften dahingehend beschränkt, nur bis zu einer bestimmten Größe und Gewicht für einen Benutzer akzeptabel zu sein. Aufgrund dieser Eigenschaften benötigt ein eventuell mehrbändiges Nachschlagewerk viel Regalplatz.

2.2.2 Die CD-ROM

Bei einer CD-ROM hängt der Grad der Orientierungslosigkeit von der Art der Realisierung bzw. dem Aufbau der Navigationshilfen ab. Navigationshilfen können auf der CD-ROM vielfältig angeboten werden. Die tatsächlich vorhandenen Bedienungsmodalitäten hängen vom Hersteller ab. Da eine CD-ROM von der Speicherkapazität her begrenzt ist, ist theoretisch in einem gewissem Rahmen auch die Orientierungslosigkeit begrenzt. Auch die Komplexität der Bedienung hängt weitgehendst von den Bedienungsmodalitäten ab. Die meisten Werke sind per Mausklick zu durchwandern. Die Verwendung einer CD-ROM erfordert technische Hilfsmittel wie einen Rechner, ein CD-ROM Laufwerk und überwiegend eine Maus. Der Nachteil hierbei ist, daß dieses Speichermedium mittelfristig nicht mehr von Lesegeräten neuerer Generationen unterstützt wird. Da die CD-ROM ein elektronisches Medium ist, können mit ihr mulitmediale Anwendungen realisiert werden. Nachschlagewerke auf CD-ROM sind nicht billig, jedoch relativiert sich auch hier der Preis der Anschaffung bei viel Benutzung. Eine CD um weitere Eigenschaften zu erweitern ist so nicht möglich. Nachdem ein System auf eine CD gebrannt wurde, sind Erweiterungen an diesem System nicht mehr durchführbar. Da die Aktualisierungszyklen bei einer CD im Verhältnis zum Buch viel kürzer und günstiger sind, können Erweiterungen relativ schnell eingebaut werden. Ist das CD-Laufwerk mit einem Netzwerk verbunden, können mehrere Benutzer auf eine CD zugreifen[2]. Eine CD vereinigt mehrere Bücher und führt somit zu mehr Platz in den Regalen, als wenn sie dort in Buchform stehen würden.

[2]Bei dem Abruf von Echtzeitdaten kann es zu Engpässen bei den Netzverbindungen kommen. Andere Benutzer müssen in diesem Moment warten, oder der Transport von Echtzeitdaten wird unterbrochen.

2.2.3 Das Internet

Das Internet, bzw. der meistgenutzte Dienst des Internets, das WWW bietet eine Fülle an Informationen[3]. Leicht kann der Benutzer nach dem Öffnen mehrerer Dokumente die Übersicht verlieren. Die Navigation im Internet erfolgt über *hyperlinks*, die durch einen Mausklick aktiviert werden. Hier liegt es beim Autor der Seiten, diese so zu gestalten, daß sich der Benutzer seines Standortes permanent bewußt ist. Die Bedienung der *hyperlinks* zeichnet sich durch Einfachheit aus. Den Zugang zum Internet erhält nur wer einen Rechner mit den entsprechenden Einrichtungen (Modem, Internetsoftware) besitzt. Auch in diesem Medium können alle Perzeptionsmedien eingesetzt werden. Die Kosten für Informationen aus dem Internet ergeben sich aus verschiedenen Komponenten: Zum einen muß eventuell der Zugang zum Internet bei einem *Provider* bezahlt werden und zum anderen kann für einige Informationen eine Gebühr vom Anbieter erhoben werden. Dieses Medium ist sehr gut mit anderen Diensten aus dem Internet verknüpfbar. Weiter bietet das Internet Nachschlagewerke (und sehr viel weitere Information), die sonst nicht im eigenen Besitz wären. Die meisten Angebote im Netz sind zudem (noch) kostenfrei zu benutzen. Ein großer Vorteil des Internets sind seine kurzen Aktualisierungszyklen, die noch kürzer als bei der CD sind. Änderungen können sofort eingefügt werden und sind ad-hoc sichtbar für die Benutzer. Die Kapazität der einzelnen Server, die an dem Internet mitwirken, ist beschränkt. Da das Internet jedoch aus einem Zusammenschluß mehrerer Server besteht ist die Kapazität des Netzes, theoretisch gesehen, unendlich. Außer dem Platz für den Rechner wird keiner benötigt.

2.2.4 Fazit

Die Betrachtung der Medien Buch, CD-ROM und Internet in Bezug auf den obigen Fragenkatalog zeigt, daß das Internet die meisten Vorzüge aufweist. Gerade in den Punkten Aktualität, unterstützte Medien und Erweiterbarkeit bietet das Internet die größte Flexibilität. Die Abbildung 2.1 zeigt dies graphisch aufbereitet. Auf der x-Achse sind die unterstützten Medien aufgeführt, wobei diskrete Medien Text und Graphik beinhalten und kontinuierliche Audio und Video (s.o.). An der y-Achse ist die Aktualität abgetragen, eingeteilt in gering, mittel und hoch. Auf der z-Achse ist die Erweiterbarkeit, die in Pfeilrichtung zunimmt, angegeben.

2.3 Konventionelle Nachschlagewerke

Der folgende Abschnitt nimmt exemplarisch für einige Arten von Nachschlagewerken eine Untersuchung einzelner Werke vor. Die vorgestellten Werke werden in Abschnitt 2.5 auf bestimmte, dort aufgestellte Kriterien hin untersucht.

Telefonbuch für Hamburg

Das Telefonbuch für Hamburg, sowie alle anderen Telefonbücher für Deutschland, führen die eingetragenen Einwohner der Stadt in alphabetischer Reihenfolge auf. Zu einem Namen gibt es immer mindestens eine Telefonnummer, teilweise sind Adressen mit angegeben.

[3] Eine umfangreiche Betrachtung über das WWW als Informationssystem liefert [Bösch 97].

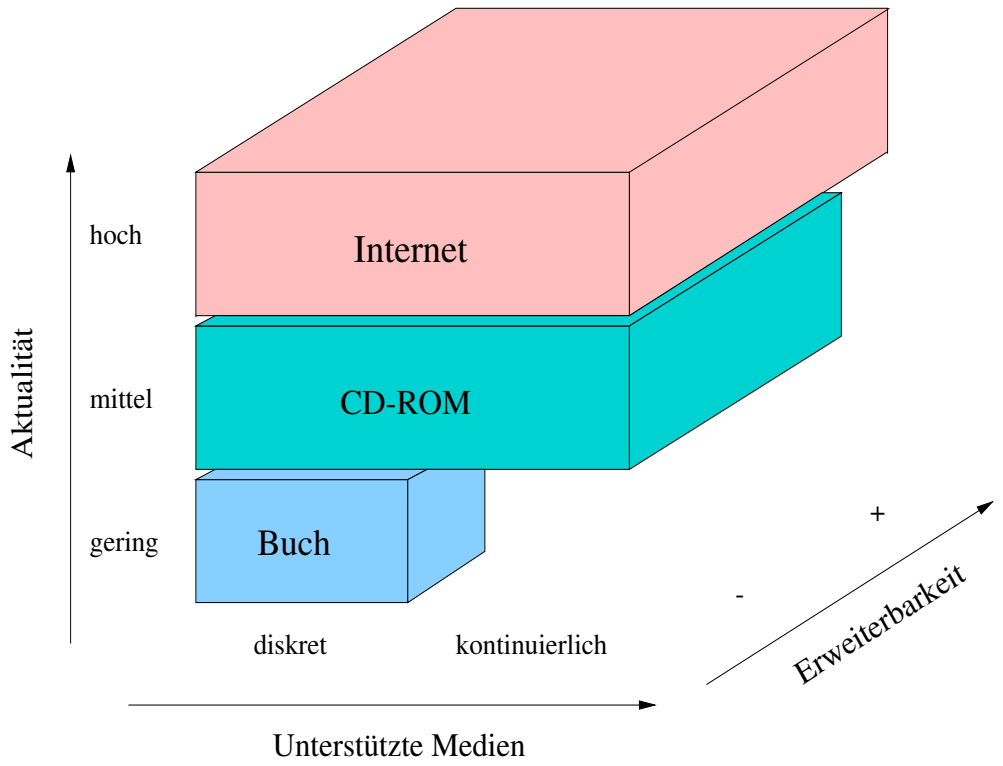

Abbildung 2.1: Fazit des Fragenkatalogs

Deutsche Biographische Enzyklopädie

Die Deutsche Biographische Enzyklopädie (DBE, siehe [KiVi 95]) gibt Auskunft über 60.000 Personen aus allen Bereichen des öffentlichen und politischen Lebens. Die Artikel setzen sich zusammen aus Name und Lebensdaten sowie Biographie und Literaturhinweisen. Die Stichwörter sind alphabetisch angeordnet. Es sind keine Bilder oder Übersichten vorhanden. Am Ende eines Artikels stehen optional Literaturangaben. Verweise zu anderen Artikeln werden durch einen Pfeil (→) angezeigt.

Kosmos Naturführer, Was blüht denn da?

Dieses Nachschlagewerk (siehe [AiGo 86]) gehört zu Karten und Kartenwerke aus dem Bereich der Biologie. Es gibt dem Verwender die Möglichkeit auf Wanderungen eine einfache Bestimmung der Pflanzen vorzunehmen. Die Bestimmung der Pflanzen erfolgt auf drei aufeinanderfolgenden Schritten, die als Frage formuliert sind.

- Welche Blütenfarbe hat die fragliche Pflanze?
- Zu welcher Standortgruppe gehört der Fundort?
- Hat die Pflanze radiär-symmetrische Blüten?
  Wenn ja, wie viele Blütenblätter oder -zipfel besitzt die Blüte? Oder ist die Blüte zweiseitig-symmetrisch?

Mit Hilfe der Antworten bewegt sich der Verwender durch den farbigen Bestimmungsteil. Die Farbe der Blüte wird durch die entsprechende Farbe am Buchseitenrand dargestellt. Der Standort ist als Überschrift auf jeder Seite und die Anzahl der Blütenblätter als Symbol daneben dargestellt.
Zu jeder Abbildung gibt es einen erläuternden Textteil. Anhand dieser Beschreibungen muß die fragliche Pflanze identifiziert werden.

DUDEN, Deutsches Universal Wörterbuch

Dieses Wörterbuch (siehe [Duden 83]) erfaßt den Wortschatz der deutschen Gegenwartssprache. Die Stichwörter sind alphabetisch geordnet, der Bedeutungsteil ausführlich mit Anwendungsbeispielen versehen. Der Bedeutungsteil enthält Bestimmungen von Wortinhalten und Beschreibungen von Bedeutungsstrukturen.

Japanese-English Character Dictionary

Dieses Werk [Nels 90] dient dazu, japanische Zeichen ins Englische übersetzen zu können. Es beinhaltet ca. 5000 ausgewählte Zeichen sowie 70.000 zusammengesetzte, die übersetzt werden. Auf diese Zeichen wird nach einem bestimmten System zugegriffen. In diesen Zeichen gibt es Muster, die immer wieder vorkommen. Diese Muster werden in 214 Basiselemente (sogenannte *Radicals*) klassifiziert. In einer Tabelle am Anfang des Buches sind die Radicals nach ihrer Strichanzahl geordnet aufgelistet. Der (vereinfachte) Benutzungsablauf ist wie folgt:

- Entscheide, welches das Radical in dem Zeichen ist und ermittel die Anzahl der Striche, aus denen es besteht.
- Ermittle in der Tabelle nach der Nummer des Radicals
- Unter dieser Nummer ist das Radical mit seinen Kombinationen zu finden. Die Zeichen sind nach der Strichanzahl der restlichen Striche angeordnet.

Die Abbildung 2.2 zeigt das Zeichen für *nihon* (Land der aufgehenden Sonne, Japan) sowie das Radical und die Zählung der Striche. Verweise zu anderen ähnlichen Zeichen sind im Text

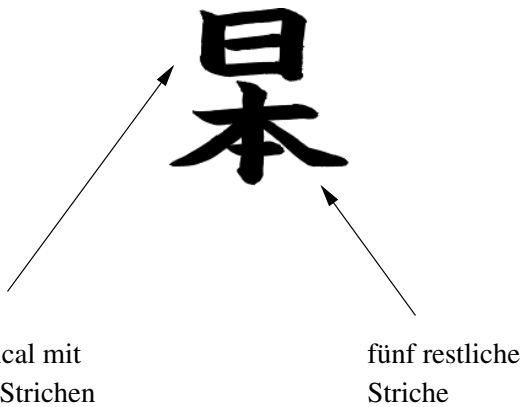

Abbildung 2.2: Japanisches Zeichen für *nihon*

durch Bemerkungen angegeben.

2.4 Digitale Nachschlagewerke

Häufig liegen digitalen Nachschlagewerken konventionelle Nachschlagewerke zugrunde, die um mediale Elemente wie Bild und Ton erweitert worden sind [Göd 94]. Diese Produkte sind textorientiert und sollen weiterhin gelesen werden. Mediale Elemente werden zum besseren Verständnis und zur Illustration eingesetzt. In den folgenden Abschnitten werden digitale Nachschlagewerke vorgestellt, die auf CD-ROM (Abschnitt 2.4.1) erhältlich sind oder im Internet (Abschnitt 2.4.2) realisiert sind.

2.4.1 Nachschlagewerke auf CD-ROM

Digitale Nachschlagewerke auf dem Medium CD-ROM haben gegenüber herkömmlichen Nachschlagewerken den Vorteil auf "kleinem Raum" mehrbändige Nachschlagewerke zu vereinigen, einen schnelleren Zugang auf diese zu gewähren und vor allem andere Zugriffsmöglichkeiten zu bieten. Eine CD-ROM wird in der Regel von einem Benutzer verwendet. Ein Vorteil gegenüber dem Buch liegt in dem kürzeren Aktualisierungszyklus. Eine Neuauflage ist günstiger als beim Buch. Dieser Abschnitt stellt ausgewählte Produkte vor.

Bertelsmann Universal-Lexikon

Dieses Universal-Lexikon (siehe [Bert 96] und Abbildung 2.3) bietet eine Sammlung an Dokumenten, die überwiegend in Textformat vorhanden sind. Vereinzelt gibt es zum Text erläuternde Graphiken, Bilder, Tabellen oder Animationen und Videos. Auf die Dokumente wird über eine alphabetisch sortierte Liste mit Stichwörtern zugegriffen. Über diese Stichwörter kann ein Filter gelegt werden, damit z.B. nur Stichwörter angezeigt werden, die Einträge mit Tabellen, Musik, Video oder Fotos referenzieren. Innerhalb der Texte sind Verbindungen zu anderen Einträgen farbig markiert und mit Mausklick anwählbar. Weiter gibt es die Möglichkeit über eine *History*-Liste oder einen *Go Back-Button* ehemals besuchte Einträge erneut anzuzeigen. Zwei Pfeile ermöglichen ein sequentielles vor- und zurückblättern in der Stichwortliste. Über das Sinnbild eines Lesezeichens gelangt der Benutzer zu der Möglichkeit Lesezeichen zu setzen und zu kommentieren oder zu löschen. Über das Drucksinnbild können alle Dokumente zum Drucker geschickt werden. Eine Volltextsuche ermöglicht die Suche nach dem angegebenen Wort in den vorhandenen Texten.

Microsoft Encarta 95

Diese allgemeine Enyklopädie auf CD-ROM [Enca 95] umfaßt diverse Dokumente, die aus diskreten als auch kontinuierlichen Medien bestehen. Die Encarta bietet eine Reihe von Möglichkeiten, auf die in ihr enthaltene Information zuzugreifen. Über aufklappbare Menüs (*popupmenu*) ist es möglich eine Kategorie (*Area of Interest*, systematisch geordnet) und daraus wiederum eine Subkategorie (*Category*) auszuwählen. Ein weiteres Fenster, in dem alle zugehörigen Artikel aufgelistet sind, erscheint. Über diese Artikel können wiederum Filter gelegt werden. Filter sind z.B. Medienart (z.B. Video, Graphik, Audio, ...), ein bestimmter geschichtlicher Zeitabschnitt oder geographische Angaben. Als Orientierunghilfe bietet die Encarta einen *Go Back-Button* und eine Liste der besuchten Einträge, auf die der *Go Back-Button* zurückgreift. Gesuchte Stichwörter können direkt in ein Suchformular eingegeben und mit booleschen Operatoren verknüpft werden. Weiter gibt es die Möglichkeit über "A-Z"-Knöpfe *Buttons* in der Kategorie oder der Subkategorie sequentiell alphabetisch zu blättern.

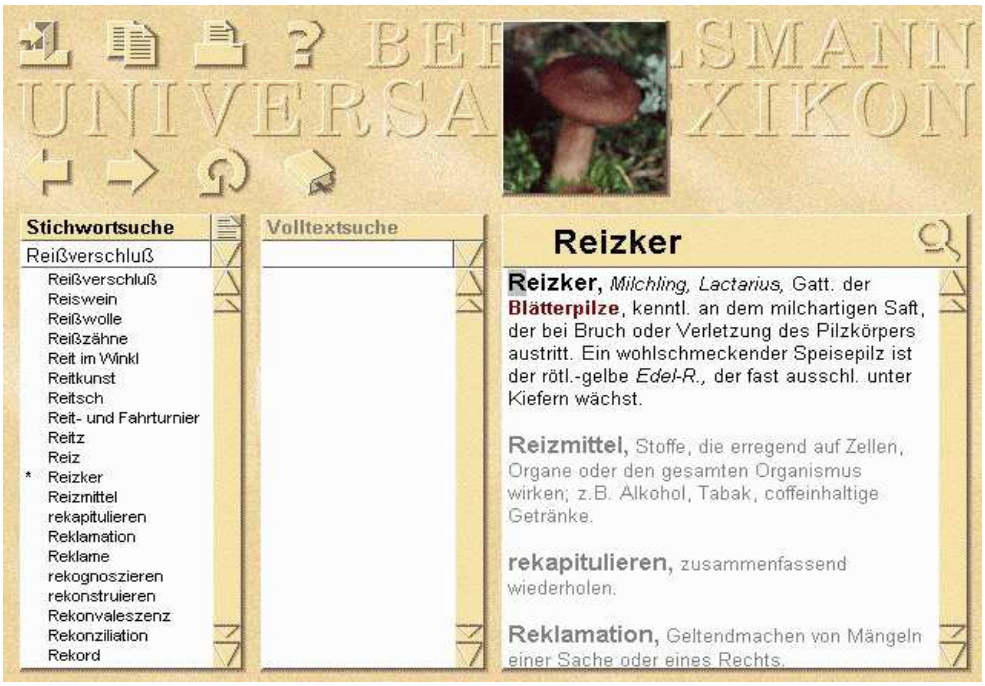

Abbildung 2.3: Bertelsmann Universal-Lexikon Oberfläche

Auf vorhandene Multimediaelemente wird durch kleine Sinnbilder hingewiesen. Gibt es zu einem Text Bilder, werden sie nicht nur durch das Sinnbild angekündigt, sondern sind bereits in einem kleinen Fenster parallel zum Text dargestellt. Alle medialen Dokumente sind auch direkt abfragbar. Der integrierte Atlas erlaubt eine direkte Anwahl von geographischen Punkten, sofern Artikel dazu vorhanden sind.

Infopedia 2.0

Infopedia ist eine CD-ROM, die eine 29-bändige Enzyklopädie enthält, einen Atlas, einen Weltalmanach, ein Synonymwörterbuch und vier Bände des Merriam Webster. Alle Werke sind in eine graphische Benutzeroberfläche integriert. Über einen allgemeinen Index läßt sich direkt auf alle Einträge zugreifen. Durch die Wahl von Themenbereichen und Medientypen reduzieren sich die Einträge auf eine Teilmenge. Jedes Buch kann auch linear gelesen werden, vom Titelbild bis zum Index. Zum Eintrag gehörende Medien werden als Sinnbild (*Icon*) angeboten. Zu den Texteinträgen gibt es auf verschiedene Detailgrade einstellbare Inhaltsverzeichnisse. Die Stichwortsuche sollte aufgrund der umfangreichen Ergebnismenge durch boolesche Operatoren ergänzt werden. Als Orientierungshilfe wird eine Liste angeboten, die die besuchten Einträge aufzählt. Durch Auswahl eines Eintrags kann wieder dorthin gesprungen werden. Einen *Go Back-Button* gibt es nicht. Zwei Pfeile-*Buttons* bieten die Möglichkeit innerhalb des gerade aktuellen Werkes sequentiell vor- und zurückzublättern. In den Karten des Atlas ist es möglich angrenzende Gebiete anzuwählen und die dazugehörige Karte zu erhalten. Artikel über Auswahl einer Bildbeschriftung sind so nicht zu erreichen. Diese müssen über das Sinnbild selektiert werden.

2.4.2 Nachschlagewerke im Internet

Das Internet bietet bereits eine Vielzahl von Nachschlagewerken an. Am häufigsten sind Wörterbücher zu finden. Auf Informationen im Internet können viele Benutzer gleichzeitig zugreifen. Neue Informationen und Änderungen können sofort durchgeführt und dem Benutzer zur Verfügung gestellt werden. Im folgenden werden Nachschlagewerke aus dem Internet vorgestellt.

Meyers Lexikon – Das Wissen A-Z

Diesem Nachschlagewerk (siehe Abbildung 2.4) [Meyers Lexikon] liegt die gleichnamige Papierausgabe zugrunde. Über den Anfangsbuchstaben des Stichwortes wird auf eine Liste zugegriffen, die den betreffenden Buchstaben weiter unterteilt. Nachdem die entsprechende

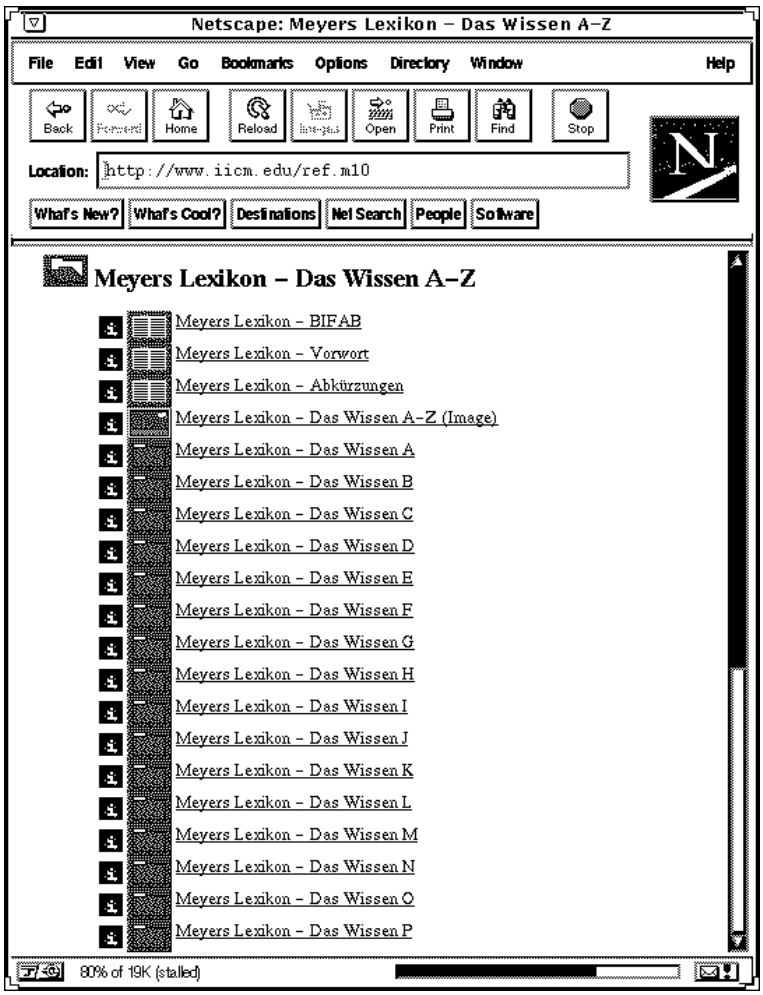

Abbildung 2.4: Startseite des Meyers Lexikon – Das Wissen A-Z

Unterteilung ausgewählt wurde, gelangt der Benutzer zu einer Liste mit Wörtern. Aus diesen Wörtern ist das Gesuchte auszuwählen. Ergebnis ist ein kurzer beschreibender Text. Inner-

halb dieses Textes gibt es keine Verweise auf andere Artikel.
Mit einer Suchfunktion (siehe Abbildung 2.5) werden die Artikel in durch Optionen steuerbarer Weise (z.B. Wort im Titel) nach dem angegebenen Wort durchsucht.

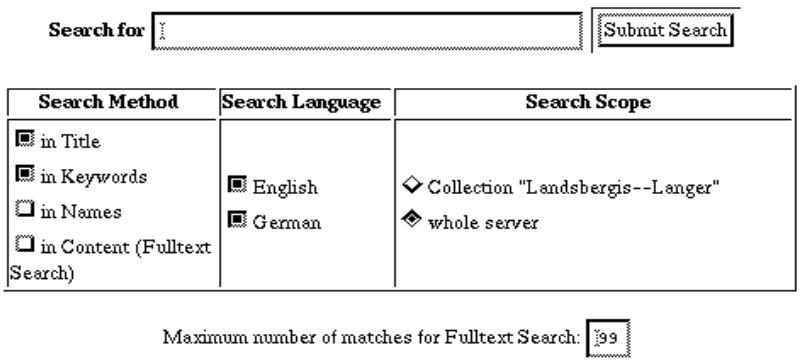

Abbildung 2.5: Suchformular des Meyers Lexikon – Das Wissen A-Z

Beilstein *Dictionary*

Das Beilstein *Dictionary* [Beilstein] ist ein Übersetzungswörterbuch (Englisch/Deutsch) für die organische Chemie. Über ein horizontal aufgelistetes Alphabet (siehe Abbildung 2.6) wird auf eine Liste mit Wörtern und deren Übersetzung zugegriffen.

Über ein Suchformular (siehe Abbildung 2.7) lassen sich ebenfalls Wörter im Bestand selektieren.

BerlinINFO – Stadtplandienst im WWW

BerlinINFO ist ein Online-Stadtplan (siehe Abbildung 2.8) [BerlinINFO] von Berlin, der für das WWW realisiert wurde. In dieses Formular kann ein Straßenname (oder dessen erste Buchstaben) und optional eine Hausnummer eingegeben werden. Bei erfolgreicher Suche gelangt der Benutzer zu einem Ausschnitt eines Berlinstadtplanes (siehe Abbildung 2.9). Diese Ausschnitte wurden in Planquadrate eingeteilt. Es wird immer das vollständige Planquadrat angezeigt, in der die gesuchte Straße liegt. Weiter werden auch die anliegenden Planquadrate teilweise mit angezeigt. Wurde eine Hausnummer angegeben, erscheint ein Kreis als Markierung an dem entsprechenden Ort. In der Abbildung 2.9 erscheint der Kreis in einer kleinen Zusatzkarte. Bei der Java-Variante dieses Plans erscheint der Kreis direkt im Planquadrat.

Der Benutzer hat jetzt die Möglichkeit sich durch Berlin zu bewegen, indem die anliegenden Planquadrate ausgewählt werden oder in der Umrisskarte von Berlin ein Planquadrat direkt ausgewählt wird.

2.5 Ein Klassifizierungsschema für Nachschlagewerke

In den vorigen Abschnitten werden Nachschlagewerke untersucht, die konventioneller und digitaler Art sind. Im folgenden werden Kriterien zur Klassifizierung von Nachschlagewerk aufgestellt, die für diese Arbeit relevant sind. Kriterien für Rezensionen von Nachschlagewerken wie z.B. Anzahl der Stichwörter oder zugrundeliegende Werke bei CD-ROMs sind für

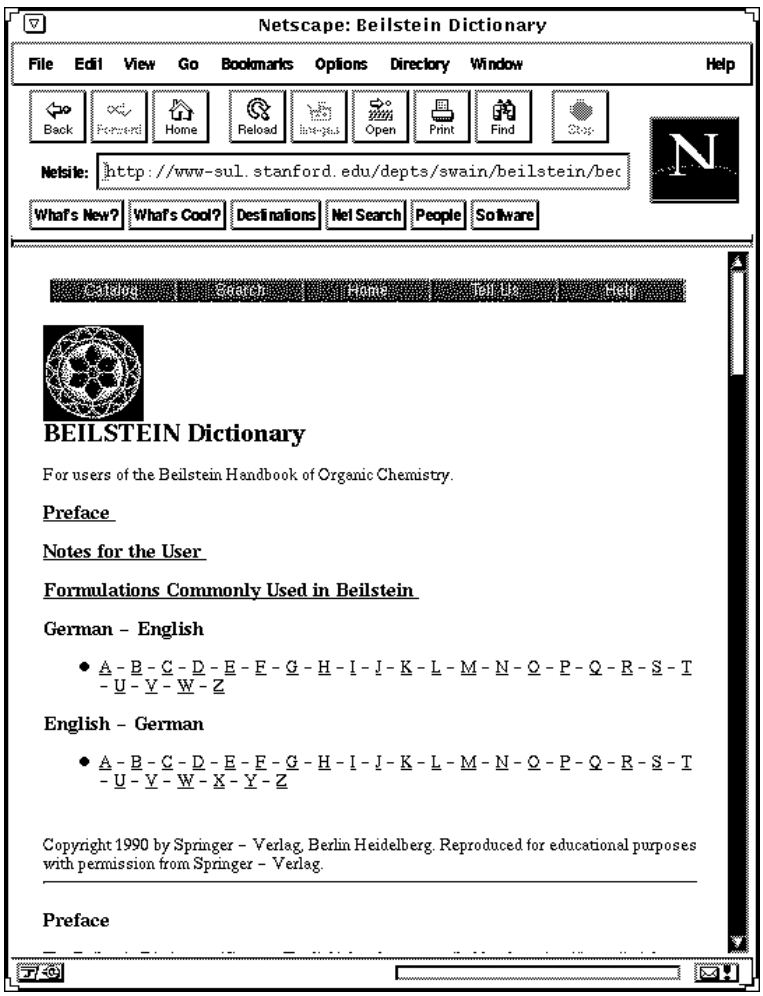

Abbildung 2.6: Startseite des Beilstein *Dictionary*

die Dienstaufstellung dieser Arbeit nicht von Bedeutung[4]. Vielmehr interessieren die Zugriffsstrukturen und deren mediale Ausrichtung sowie die verschiedenen Möglichkeiten der Zugriffe. Weiter sind die Dokumente mit ihren verschiedenen medialen Elementen und Verknüpfungen untereinander interessant. Daraus ergeben sich Kriterien, die wie folgt benannt werden:

- Mediale Ausrichtung der Zugriffsstrukturen
- Einstufige und mehrstufige Zugriffsstrukturen
- Zugriffsmodalitäten
- Mediale Ausrichtung der Dokumente
- Verknüpfung von Dokumenten

Mediale Ausrichtung meint die Perzeptionsmedien.

[4]Kriterien für die Rezension von Nachschlagewerken auf CD-ROM finden sich in [Göd 94]

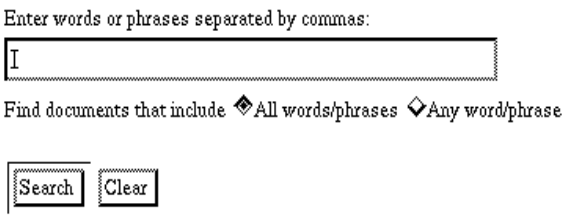

Abbildung 2.7: Suchformular des Beilstein *Dictionary*

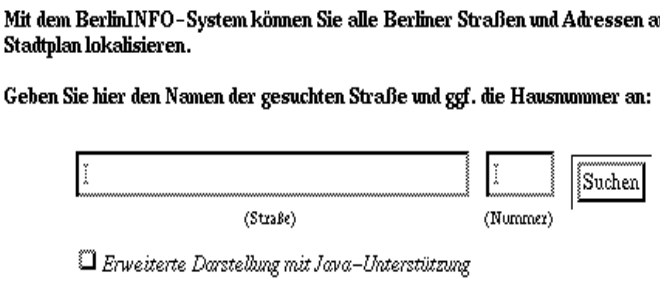

Abbildung 2.8: Ausschnitt der Startseite des BerlinINFO

Mediale Ausrichtung der Zugriffsstrukturen

Die mediale Ausrichtung der Zugriffsstruktur eines Wörterbuchs z.B. ist in Abbildung 2.10 dargestellt. Über ein Wort wird auf Dokumente zugegriffen, die aus Text oder Wort bestehen. Die mediale Ausrichtung der Zugriffsstruktur aus dem Bestimmungsbuch der Biologie und des Stadtplanes BerlinINFO sind in den Abbildungen 2.11 und 2.12 dargestellt. In dem Bestimmungsbuch der Biologie wird über eine Farbe (darstellbar als Text oder Abbildung) auf Textabschnitte zugegriffen. In dem Stadtplan BerlinINFO gibt es auch die Möglichkeit von einer Karte aus wiederum auf eine Karte zuzugreifen.

Einstufige und mehrstufige Zugriffsstrukturen

Einstufig und mehrstufig bezeichnet die Ebenen, über die ein Nachschlagewerk benutzt wird, um zum endgültigen Eintrag zu gelangen. Das Wörterbuch besitzt eine einstufige Zugriffsstruktur (siehe Abbildung 2.10), das Bestimmungsbuch der Biologie eine mehrstufige. Bei einer mehstufigen Zugriffsstruktur wird über mehrere Ebenen vom Eintrag aus auf den Inhalt zugegriffen (siehe Abbildung 2.13).

Zugriffsmodalitäten

Unter Zugriffsmodalität soll hier die Art und Weise des Zugriffs verstanden werden. Ein Nachschlagewerk unterstützt mehrere, wenn auf unterschiedlichen Wegen auf den Inhalt des Nachschlagewerks zugegriffen werden kann. Im Beispiel des Bestimmungsbuches der Biologie wäre das der Zugriff

- über die mehrstufige Zugriffsstruktur,

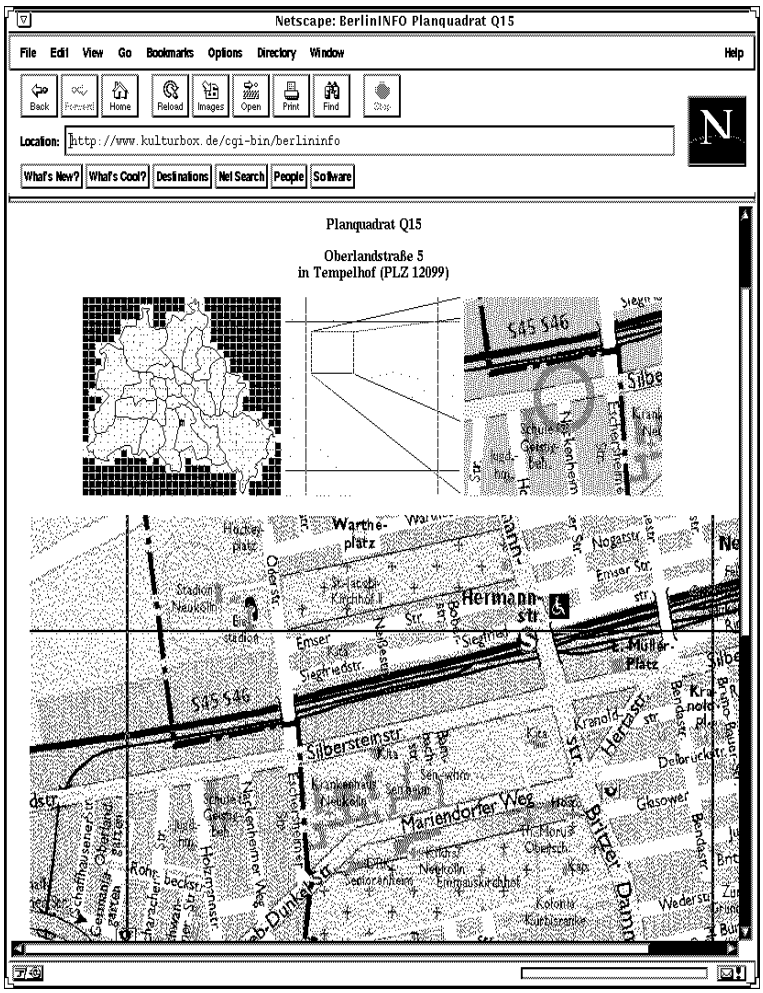

Abbildung 2.9: Folgeseite des Berlin-Info

- über eine Übersicht der Tabellen,
- über den alphabetischen Index am Ende des Buches.

Ein weiterer Aspekt, der alle Werke (konventionelle als auch digitale) betrifft, ist das Blättern. Durch die zufällige Selektion wird ebenfalls auf Inhalte zugegriffen.

Mediale Ausrichtung der Dokumente

Die Dokumente können die gleichen medialen Ausrichtungen wie Zugriffsstrukturen besitzen. Beispiele dazu finden sich in den bereits gezeigten Abbildungen 2.10, 2.11 und 2.12. Die mediale Ausrichtung der Dokumente in diesen Beispielen ist Text, Wort und Karte.

Verknüpfung von Dokumenten

Dieses Kriterium bezieht sich besonders auf die elektronischen Ausgaben. Während bei herkömmlichen, papierbasierten Nachschlagewerken Verknüpfungen nur durch kleine graphi-

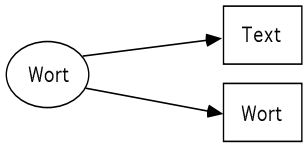

Abbildung 2.10: Mediale Ausrichtung der Zugriffsstruktur eines Wörterbuchs

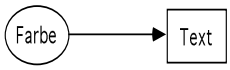

Abbildung 2.11: Mediale Ausrichtung der Zugriffsstruktur eines Bestimmungsbuches der Biologie

sche Symbole (z.B. $\uparrow$ oder $\rightarrow$) möglich sind, wird von den elektronischen Werken mehr erwartet, teilweise aber nicht erfüllt.

Zusammenfassung

Die Tabelle 2.5 gibt eine Übersicht über die untersuchten Produkte in Zusammenhang mit den zuvor aufgestellten Kriterien. Auffällig dabei ist, daß bei allen Werken die Zugriffsstruktur aus medialen Elementen diskreter Art besteht. Interessant ist dabei der Stadtplan BerlinINFO der eine Auswahl über Bilder zuläßt. Dieses ist bei der Encarta95 oder Infopedia2.0 zwar auch möglich, jedoch sehr eingeschränkt. Die Nachschlagewerke mit einer einstufigen Zugriffsstruktur überwiegen. Für jedes Werk gilt der Aspekt des Blätterns in der Spalte Zugriffsmodalitäten. Bei der medialen Ausrichtung der Dokumente sind bei den digitalen CD-Versionen alle Perzeptionsmedien vertreten. Im Internet wird (noch), aufgrund der fehlenden Protokolle und geringen Bandbreite, auf allzuviel Aufwand verzichtet. Bei den papiergebundenen Nachschlagewerken liegt es in der Natur der Sache keine kontinuierlichen Medien anzubieten. Die Verknüpfung von Dokumenten erfolgt durch diskrete Medien. Um eine feinere Einteilung in der Tabelle angeben zu können, sind diskrete Medien in Wort, Text, Bild und Graphik unterteilt und diskrete Medien in Audio, Video und Animation.

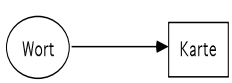

Abbildung 2.12: Mediale Ausrichtung der Zugriffsstruktur des Stadtplanes BerlinINFO

	Mediale Ausrichtung d. Zugriffsstruktur	**Ein- u. mehrstufige Zugriffsstruktur**	**Zugriffsmodalitäten**	**Mediale Ausrichtung d. Dokumente**	**Verknüpfung von Dokumenten**
Telefonbuch	Wort	einstf.	nur über die (alphabetischen) Einträge	Wort	keine
DBE	Wort	einstf.	-"-	Wort	Pfeil mit Wort
Naturführer	Farbe, Text	mehrstf.	über mehrstf. Zugriffsstruktur; über Übersicht von Tabellen; über alphabetischen Index	Text, Graphik	durch Text
Duden	Wort	einstf.	nur über die (alphabetischen) Einträge	Wort,Text	Pfeil mit Wort
Character Dict.	Bild (Zeichen)	mehrstf.	nur über vorgegebenes Verfahren	Wort	durch Text
Bertelsmann	Wort	einstf.	Stichwortliste, Volltextsuche	Wort,Text,Graphik, Audio,Animation	Text (farbig markiert)
Encarta 95	Wort	einstf.	über alphabetischen Index; über Suchformular; über Kategorien mit Filtern	-"-	Graphik,Wort(Hyperlink)
Infopedia 2.0	Wort	einstf.	über alphabetischen Index; über Stichwortsuche	-"-	Graphik,Wort(Hyperlink)
Meyers Lexikon	Wort	einstf.	über alphabetischen Index; über Suchformular	Wort,Text	keine
Beilstein Dict.	Wort	einstf.	nur über Anfangsbuchstaben d. betr. Wortes	Wort	keine
BerlinINFO	Bild, Wort	einstf.	über Formular mit Straßen-/ Ort-Angabe; über Berlinbild	Bild	Bild

Tabelle 2.1: Zusammenfassung der Produkte in Bezug auf die Kriterien

Abbildung 2.13: Mehrstufige Zugriffsstruktur eines Bestimmungsbuches der Biologie

Kapitel 3

Entwicklungsumgebungen und Sprachen

Dieses Kapitel stellt die in dieser Arbeit verwendete Entwicklungsumgebung und Sprachen vor. Die ersten Abschnitte stellen Java-Eigenschaften im Überblick vor und gehen anschließend auf einige verwendete Konzepte ein. In der vorliegenden Arbeit wird viel mit den Neuerungen der Java-Version 1.1 gearbeitet. Die Informationen zu diesen neuen Konzepten wird der JDK1.1.1 Dokumentation von Sun Microsystems entnommen (siehe [Sun 97]). Am Ende des Kapitels werden Konzepte vorgestellt, die bei dem Entwurf und der Entwicklung der Implementation von Bedeutung sind.

3.1 Java

Dieser Abschnitt stellt, ausgehend von Schlagworten, die Programmiersprache Java vor und skizziert die Einbindung in statische Dokumente, wie HTML-Seiten.

3.1.1 Ein Überblick

Java ist eine von Sun Microsystems entwickelte, objektorientierte, C++-ähnliche Programmiersprache [MSS 96]. Sie eignet sich sowohl zut Entwicklung von eigenständigen Applikationen als auch von Internetanwendungen. Durch die Möglichkeit, Java-Programme über das Internet zu laden, können statische HTML-Seiten mit einem dynamischen Aspekt versehen werden. Es können aber auch eigenständige verteilte Anwendungen entwickelt werden. Folgende Abschnitte erläutern ausgewählte Eigenschaften von Java.

Portabel

Java ist portabel. Java-Programme werden von einem Übersetzer (*compiler*) in *Bytecode* für die virtuelle Maschine übersetzt. Der Code für die virtuelle Maschine wird von einem Interpreter ausgeführt. Der Code kann somit auf jeder Plattform verwendet werden, für die eine Portierung des Interpreters existiert. Abbildung 3.1 stellt diesen Vorgang dar.

Weiter sind alle Datentypen exakt festgelegt, unabhängig von der jeweiligen Implememtierung. Z.B. wird der Typ int immer als 32-Bit-Zahl mit Zweierkomplement-Schreibweise für negative Zahlen dargestellt. Der Preis der Plattformunabhängigkeit ist der Performanzverlust durch die Verwendung des Interpreters (Faktor 10-20 gegenüber C).

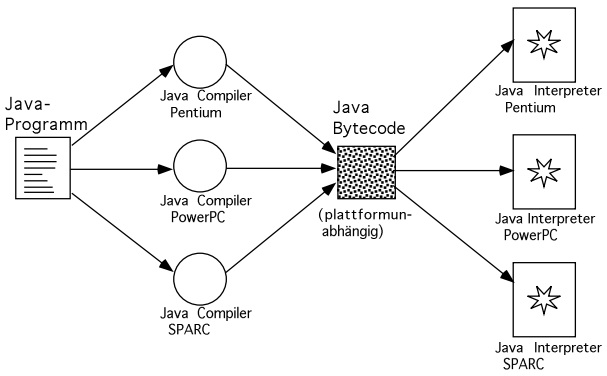

Abbildung 3.1: Erstellung und Interpretation von Java Programmen (angelehnt an [LePe 96])

Objektorientiert

Java ist vollständig objektorientiert aufgebaut. In weiten Zügen ähnelt es C++. Java erlaubt jedoch keine direkten Zugriffe auf Strukturen oder Zeigermanipulationen. So werden Instabilitäten und Zugriffe über "Hintertüren" auf unerlaubte Bereiche unterbunden. Java stellt eine Menge von vordefinierten Klassen, die in Paketen (*packages*) zusammengefaßt sind, bereit [Flan 96]. Es gibt Klassen für die Erstellung graphischer Benutzeroberflächen, für die Netzwerkkommunikation, für Ein- und Ausgabeströme, uvm. Java erfüllt die Grundanforderungen an eine objektorientierten Sprache[1]:

Kapselung: In objektorientierten Sprachen bilden die Daten zusammen mit ihren Methoden eine Einheit. Diese Einheit wird Objekt genannt. Die Daten eines Objekts werden in lokalen Variablen des Objekts gespeichert. Der Inhalt der lokalen Variablen bestimmt den Zustand des Objekts. Der Zustand des Objekts kann nur durch Methoden verändert werden. Der Anwender hat somit keinen Zugriff auf die interne Struktur des Objekts. Die Daten sind in den Objekten gekapselt (*information hiding*).

Vererbung: Zwischen Klassen können Vererbungsbeziehungen existieren [MoSch 92]. Subklassen werden aus Superklassen abgeleitet. Subklassen erben alle Eigenschaften (Variablen und Methoden) der Superklasse. Die Subklassen können in den Eigenschaften modifiziert bzw. erweitert werden. Java bietet nicht die Möglichkeit der Mehrfachvererbung auf Klassen, sondern nur eingeschränkt auf Schnittstellen *Interfaces*.

Polymorphie: In polymorphen Sprachen können Werte mehreren Typen angehören. Es gibt mehrere Formen von Polymorphismus in Programmiersprachen (siehe Abbildung 3.2). Java unterstützt beide Formen des adhoc-Polymorphismus und den Inklusionspolymorphismus. Zur genauen Erläuterung der Polymorphismusarten siehe [CaWe 85]. Zur Unterstützung des parametrischen Polymorphismus ist ein Einsatz von Pizza [OdWa 97] möglich.

[1] weitere Konzepte und Prinzipien objektorientierter Sprachen finden sich in [Meyer 97]

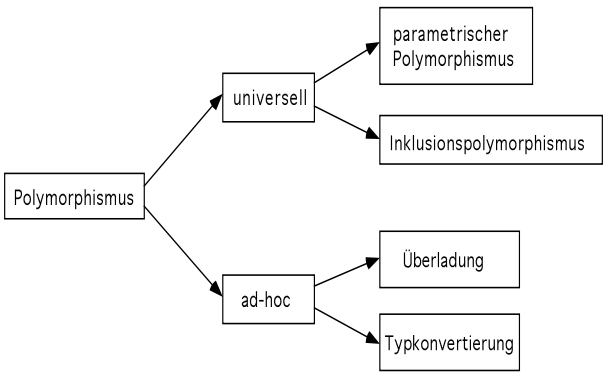

Abbildung 3.2: Formen von Polymorphismus (nach [CaWe 85])

Multithreaded

In einer graphischen Anwendungsumgebung wie einem WWW-Browser ist es sinnvoll Anwendungen (in Java *Applets* genannt) in mehrere einzelne Prozesse (*Threads*) aufzuteilen. Während sich ein *Thread* um die Aktualisierung der Bildschirmausgabe kümmert, kommuniziert ein anderer mit einem Server und ein weiterer *Thread* spielt einen Audioclip. Java unterstützt die gleichzeitige Ausführung mehrerer *Threads*, die unterschiedliche Aufgaben erledigen. Java bietet weiter Methoden zur Interprozeß-Kommunikation und Synchronisation (Monitore und *Condition Variables* nach C.A.R. Hoare).

Verteilt und Netzwerkfähig

Java ist dahin konzipiert, Programme über das Internet zu laden und lokal auszuführen. Die Java Klassenbibliothek enthält Methoden, die die Verwendung von TCP/IP-Protokollen wie HTTP und FTP und den direkten Zugriff auf *Sockets* erleichtern. Mit der Version 1.1 erweitern sich die Netzwerkfähigkeiten um eine Schnittstelle zu SQL-Datenbanken (*JDBC-Package*), die Übertragung von Javaobjekten inklusive aller Referenzen (*OSS-Package*) und die Möglichkeit Objekte über ein Netzwerk zu referenzieren und deren Methoden aufzurufen (*RMI-Package*) [Hend 97].

Robust

Durch den Verzicht auf Zeiger (*Pointer*) ist eine große Fehlerquelle vermieden worden. Java besitzt einen eingebauten *Garbage Collector*, der sich um die Speicherverwaltung kümmert. Der Programmierer braucht keine Speicherverwaltungsfunktionen wie *malloc* und *free* zu kennen, das System übernimmt die Anforderung und Freigabe von Speicherplatz.
Java ist streng typisiert und erlaubt eine Überprüfung der Typkompatibilität zur Übersetzungszeit (*compile time*). Die automatische Typumwandlung in Java ist sehr eingeschränkt. Weiter unterstützt Java die Ausnahmebehandlung (*Exception Handling*).

Sicher

Da bei einer Java-Anwendung über das Internet Code von beliebigen Servern geladen wird und lokal zur Ausführung gebracht wird, führt Java ein Sicherheitskonzept ein. So wird z.B.

durch einen sogenannten *Bytecodeverifizierer* das geladene Programm dahin überprüft, keine unerlaubten Dateimanipulationen vorzunehmen. Zu diesem Sicherheitskonzept gehört ebenfalls das schon erwähnte Vermeiden von Zeigermanipulationen. Neuere Überlegungen gehen dahin, mit ausgewählten Servern ein "Vertrauensverhältnis" einzugehen oder eine Authentifizierung vorzunehmen (siehe auch [Templ 96]).

3.1.2 Einbinden von Java-Programmen

HTML

Mit der Seitenbeschreibungssprache HTML (*Hypertext Markup Language*) werden Seiten für das WWW (*World Wide Web*) erstellt [Tol 96]. Die Dokumente werden in ihrer konzeptuellen Struktur beschrieben [Koch 95]. Es wird lediglich markiert, wie bestimmte Textpassagen aussehen sollen. Die konkrete Darstellung der Seite hängt vom verwendeten Browser ab. Die Markierungen, die das Aussehen der Textpassagen bestimmen, heißen *Tags*. Aufgrund der Verschiedenen Versionen von HTML und unterschiedliche Browser gibt es eine Vielzahl von *Tags*, die nicht von jedem Browser interpretiert werden können. Grundsätzlich werden *Tags* vom Browser ignoriert, wenn dieser sie nicht interpretieren kann. Einige *Tags* erlauben die Angabe von zusätzlichen Informationen, Attribute genannt. Einem Browser unbekannte Attribute werden ebenfalls ignoriert [MuKe 96].

Aufbau einer HTML-Seite

Die Grundstruktur jeder HTML-Seite sieht wie folgt aus:

```
<HTML>[2]
<HEAD>
  Kopfinformationen
</HEAD>
<BODY>
  Seiteninhalt
</BODY>
</HTML>
```

Applets

Anwendungen, die für das Internet in Java geschrieben werden, heißen *Applets*. Diese Applets werden mit einem *Applet-Tag* in eine HTML-Seite integriert:

```
<APPLET CODE=Say.class WIDTH=150 HEIGHT=200>
</APPLET>
```

Die Attribute `CODE, WIDTH` und `HEIGHT` sind notwendig. Das Attribut `CODE` bestimmt die Klasse, die ausgeführt wird. `WIDTH` und `HEIGHT` geben die Größe des Ausschnitts in dem Browser an, in dem das Applet schließlich angezeigt wird. Die Attribute `CODEBASE, ALT, NAME` und `ALIGN` sind optional. Zusätzlich gibt es die Möglichkeit Applets zu parametrisieren. Hierfür gibt es den *Param-Tag*. Das *Param-Tag* wird in der Applet-Umgebung plaziert:

[2] Dieser *Tag* ist technisch nicht notwendig

```
<APPLET CODE=Say.class WIDTH=150 HEIGHT=200>
<PARAM NAME=text VALUE="Hello World">
</APPLET>
```

Das *Param-Tag* besitzt die Attribute `NAME` und `VALUE`, die vom Applet ausgelesen werden. Bei Aufruf einer HTML-Seite mit eingebundenem Applet, wird der Code, der im Applet referenziert ist, geladen (im obigen Beispiel der Code der Klasse *Say*) und lokal ausgeführt. Dazu ist keine weitere Installation von Software nötig. Der Java-Interpreter ist in einigen Browsern inzwischen integriert. Ist der Browser nicht Javafähig, so wird der *Applet-Tag* ignoriert.

3.2 Relevante Programmiersprachenkonzepte

Nachfolgend werden einige ausgewählte Programmiersprachenkonzepte vorgestellt, die in dieser Arbeit verwendet werden. Alle vorgestellten Konzepte gehören zu den neuen Eigenschaften der Javaversion 1.1. Eine Beschreibung aller Konzepte der Programmiersprache Java geht über den Rahmen dieser Arbeit hinaus. Zum Verständnis der in dieser Arbeit verwendeten Programmbeispiele sei auf [Flan 96] und [MSS 96] verwiesen.

3.2.1 Verteilte Objekte

In dieser Arbeit wird eine Trennung zwischen dem Nachschlagewerk und den Benutzern vorgenommen. Zur Realisierung dieser Trennung dient das verteilte Objektmodell für Java (*Java Distributed Object Model*, DOM).

Ziel einer verteilten Objektarchitketur ist es, entfernte Objekte als lokale Objekte darzustellen. In dem Benutzer (Client) wird die Illusion geweckt mit einem lokalen Objekt zu arbeiten, obwohl sich das reale Objekt in einem anderen Adreßraum befindet. Zur Realisierung dieser Illusion wird eine eindeutige Referenz auf das entfernte Objekt benötigt. Die Objektreferenz besteht u.a. aus einer Rechneradresse, einem Kommunikationsendpunkt und einer Nummer [Hran 97]. Diese Art von Objektreferenz ist sehr unhandlich. Zu diesem Zweck gibt es einen Namensdienst, der die Zuordnung zwischen Namen und Objektreferenz verwaltet (ähnlich einem Telefonbuch). Der Benutzer braucht nur den Namen zu kennen und sich beim Namensdienst nach der Objektreferenz erkundigen. Anhand der Objektreferenz erzeugt das Laufzeitsystem (*Object Request Broker*, ORB) ein Stellvertreterobjekt (Proxyobjekt), welches das reale Objekt im Adressraum des Benutzers vertritt. Jede Nachricht an das Proxyobjekt wird von diesem mit Hilfe des ORB an das reale Objekt weitergeleitet. Damit die Kommunikation zum realen Objekt und zurück funktioniert, müssen die Argumente der Methodenaufrufe und die Rückgabewerte in einen zusammenhängenden Speicherbereich geschrieben werden. Diese Vorgänge werden als *Marshaling* (oder Serialisierung) und *Unmarshaling* (Deserialisierung) bezeichnet und werden vom Stub und Skeleton durchgeführt. Der Stub und der Skeleton werden aus der Implementation der Klasse mit einem speziellen Compiler generiert, die das verteilte Objekt repräsentiert. Die Abbildung 3.3 zeigt die Zusammenhänge der oben benutzten Begriffe auf.

Mit *Remote Method Invocation* (RMI) bietet Java eine verteilte Objektarchitektur, die auf einer homogenen Java-Umgebung basiert. Das bedeutet, bei der Realisierung verteilter Objekte in Java wird sowohl der Server als auch der Client in Java implementiert. In diesem

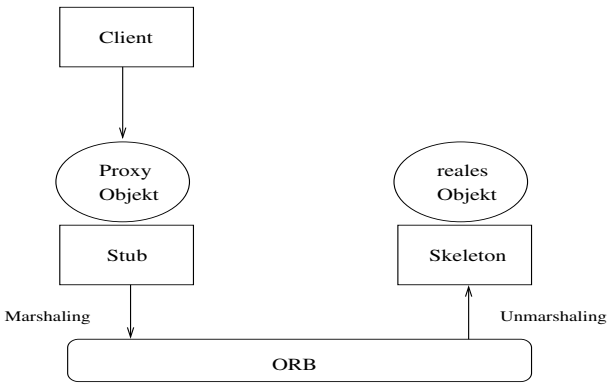

Abbildung 3.3: Kommunikation zwischen Proxyobjekt und realen Objekt mittels ORB (nach [Hran 97])

Punkt unterscheidet sich RMI von dem verteilten Objektmodell von CORBA[3]. CORBA ist eine sprachunabhängige Architektur [OrHa 97]. CORBA dient zur Spezifikation von objektorientierten Applikationen in heterogenen Umgebungen. Die Beschreibung der Schnittstellen erfolgt in einer Metasprache (*Interface Description Language*, IDL). Aus dieser Beschreibung wird der Stub und der Skeleton für die verwendete Programmiersprache erzeugt. Der Client bzw. Server kann jeweils in einer anderen Programmiersprachen implementiert sein. Durch die nahtlose Integration von RMI in die Sprache Java entsteht eine sehr enge Bindung an die Sprache, die jedoch durch die große Java-Präsenz auf vielen Plattformen ihre Existenzberechtigung besitzt.
Im folgenden sind einige Vorteile von RMI aufgeführt:

- Keine Änderung der Sprache Java durch die Einführung von RMI.
- RMI bietet Referenzen auf entfernte Objekte beliebigen Typs.
- Keine neue Schnittstellenbeschreibungssprache nötig, da Java *Interfaces* definiert.
- Volle Unterstützung des Java Typkonzepts inklusive Vererbung und Schnittstellen.
- Unterstützung der automatischen Speicherbereinigung von nicht-referenzierten entfernten Objekten.
- Unterstützung des Sicherheitskonzeptes.

Es gibt jedoch auch Nachteile:

- hohe Sprachabhängigkeit.
- Teilweise Aufhebung der Illusion durch den Zwang, auf der Client-Seite evtl. auftretende Kommunikationsfehler abzufangen.

Bei der Arbeit mit verteilten Objekten sind einige Punkte zu beachten. Die Unterschiede und Ähnlichkeiten des verteilten und des nichtverteilten Objektmodells werden im folgenden

[3] **C**ommon **O**bject **R**equest **B**roker **A**rchitecture

aufgelistet.

Ähnlichkeiten der Modelle:

- Referenzen auf entfernte Objekte können als Argument oder Rückgabewert jedes Methodenaufrufs (entfernt oder lokal) übergeben werden.
- Ein entferntes Objekt kann in jeden Schnittstellentyp konvertiert werden, den die Implementation des entfernten Objekts unterstützt.
- Mit dem *instanceof* Operator kann getestet werden, welche Schnittstellen das entfernte Objekt unterstützt.

Unterschiede der Modelle:

- Benutzer von entfernten Objekten interagieren nur mit den Schnittstellen entfernter Objekte, niemals mit den Implementationsklassen dieser Schnittstellen.
- Nichtentfernte Argumente und Rückgabewerte eines entfernten Methodenaufruf werden als Kopie übergeben, nicht als Referenz.
- Die Semantik einiger Methoden der Klasse *Object* ändert sich für entfernte Objekte (z.B. *equals, hashCode, toString, clone).*
- Benutzer entfernter Objekte müssen mit zusätzlichen Ausnahmebehandlungen für Kommunikationsfehler arbeiten.

3.2.2 Innere Klassen

Das Konzept der inneren Klassen erweitert die Möglichkeiten der Deklaration von Klassen in verschiedenen Sichtbarkeitsbereichen gegenüber der bisherigen Version. Klassen können innerhalb anderen Klassen oder Methoden deklariert werden. Innerhalb einer Anweisung kann eine Klasse auch anonym deklariert werden. Innere Klassen können direkt auf die Variablen der sie umgebenden Klasse zugreifen. Von außen sind innere Klassen nur durch qualifizierten Aufruf benutzbar. Innere Klassen simulieren Funktionen höherer Ordnung.

Dieses Konzept erleichtert das Schreiben von Adapterklassen, die z.B. gut im neuen Modell der Ereignisbehandlung (siehe nächsten Abschnitt) eingesetzt werden können. Die Abbildung 3.4 zeigt ein kleines Beispiel einer inneren Klasse im Kontext der Ereignisbehandlung. Nach Ausführung des Programms erscheint ein Fenster wie in Abbildung 3.5. In dieser Arbeit werden innere Klassen überwiegend in der Ereignisbehandlung eingesetzt.

3.2.3 Ereignisbehandlung

Durch das neue Modell der Version 1.1 wird die Trennung zwischen der eigentlichen Applikation und den Code für die graphische Benutzeroberfläche unterstützt. Die bisherige Behandlung von Ereignissen (Version 1.0) wurde von der komponentenzugehörigen Methode *action* übernommen. Die Ereignisse wurden durch Konstanten definiert, die in der Methode *action* abgefragt wurden. Für die schnelle Entwicklung von kleineren Anwendungen war diese Art gut geeignet, für die Entwicklung komplexer graphischer Benutzeroberflächen jedoch sehr aufwendig und unbequem [Joha 97].

Das Ereignismodell der Version 1.1 basiert auf einem Delegationsmodell, welches vorsieht, ein Ereignis von einer Quelle an einen Abnehmer (*EventListener*) zu übergeben. Der *EventListener* implementiert eine Unterschnittstelle von `java.util.EventListener`, die die Methoden enthält, die als Reaktion auf ein Ereignis ausgeführt werden. An einer Quelle können ein oder mehrere *EventListener* installiert sein. Die Reihenfolge der Abarbeitung läßt sich allerdings nicht voraussagen. In einem Programm mit graphischer Benutzerschnittstelle sind die Dialogelemente die Quellen, während die *EventListener* von der Applikation oder einer Adapterklasse realisiert werden. Ein weiterer Vorteil dieses Modells gegenüber der Vorversion ist seine erhöhte Performanz: Alle Ereignisse an denen kein *EventListener* interessiert ist verfallen.

```
import java.awt.*;
import java.awt.event.*;

public class GUI extends Frame{
  Button goButton;

  public GUI(){
    super("AWT&Inner Class Ex.");
    goButton = new Button("GO !");
    goButton.addActionListener(new GoAction());
    add(goButton,"North");
  }

  class GoAction implements ActionListener{
    public void actionPerformed(ActionEvent e){
    //code for execution
    ...
    }
  }

  public static void main(String[] argv){
    GUI gui = new GUI();
    gui.setSize(200,90);
    gui.setVisible(true);
  }

}
```

Abbildung 3.4: Beispielcode für innere Klassen und Ereignisbehandlung

Das Beispiel aus Abbildung 3.4 erzeugt einen *Frame* der einen *Button* besitzt. An diesem *Button* ist ein *EventListener* installiert, ein Objekt der inneren Klasse *GoAction*. In diesem Beispiel erfolgt bei Betätigung des *Buttons* keine weitere Reaktion.

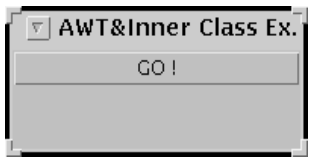

Abbildung 3.5: Demonstration des Beispielcodes

3.3 Konzepte zur Entwicklung

Die folgenden Abschnitte geben einen Einblick in Konzepte, die bei der Entwicklung und Realisierung der Dienste für diese Arbeit von Bedeutung sind. Der Abschnitt 3.3.1 erläutert Prinzipien des wiederverwendbaren objektorientierten Entwurfs. Abschnitt 3.3.2 führt kurz das Konzept Entwurfsmuster ein und stellt ausgewählte Muster, die in der Arbeit Verwendung finden, vor. Der Abschnitt 3.3.3 beschreibt im Ansatz die Client/Server-Technologie.

3.3.1 Wiederverwendbarer objektorientierter Entwurf

Durch Klassenvererbung wird die Funktionalität einer Superklasse wiederverwendet und erweitert. Neue Arten von Objekten können auf Basis existierender Objekte definiert werden [Gamma et al. 95]. Auf bestehende Implementierungen kann zurückgegeriffen werden, indem von existierenden Klassen geerbert wird.

Durch Schnittstellenvererbung wird eine Familie von Objekten mit identischen Signaturen definiert. Dies geschieht durch Erben von einer abstrakten Klasse oder durch Implementieren einer Schnittstelle und ermöglicht Subtyppolymorphie. Bei abstrakten Klassen ist zu beachten, daß sie bereits einen Teil ihrer Methoden implementiert haben können, d.h. man erbt eventuell auch Implementierungen.

Werden Objekte ausschließlich über eine Schnittstelle oder die Signatur einer abstrakten Klasse manipuliert ergeben sich zwei Vorteile:

- Benutzende Klassen wissen nichts über die Klassen der Objekte, die sie benutzen.
- Benutzende Klassen wissen nichts von den Klassen die diese Objekte implementieren. Benutzende Klassen kennen nur die abstrakte Klasse oder die Schnittstelle.

Durch Schnittstellenvererbung werden Implementationsabhängigkeiten vermieden, was die entwickelte Software robuster und leichter wart- und erweiterbar macht. Dies führt zum folgenden Prinzip des wiederverwendbaren objektorientierten Entwurfs:

> Programmiere auf eine Schnittstelle hin, nicht auf eine Implementierung.

Variablen werden nicht als Exemplare konkreter Klassen deklariert, sondern einer abstrakten Klasse oder Schnittstelle.

Eine weitere Technik zum Wiederverwenden von Funktionalität in objektorientierten Systemen ist, neben der Klassenvererbung, die Objektkomposition. Hierbei wird neue komplexe Funktionalität durch das Zusammenführen oder die Komposition von Objekten erreicht. Objektkomposition basiert auf wohldefinierten Schnittstellen der beteiligten Objekte. Diese Art

der Wiederverwendung wird *Black-Box-Wiederverwendung* genannt. Keine internen Details der Objekte sind nach außen hin sichtbar[4].

Klassenvererbung wird statisch zur Übersetzungszeit festgelegt. Die Verwendung von Klassenvererbung wird direkt von der Programmiersprache unterstützt und erleichtert so das Modifizieren von geerbten Code. Geerbter Code kann zur Laufzeit jedoch nicht geändert werden. Weiter ist bei der *White-Box-Wiederverwendung* die Kapselung der Details der Superklasse nicht mehr gegeben. Jede Änderung der Implementation der Superklasse wirkt sich auf dessen Subklassen aus. Diese Abhängigkeit schränkt die Flexibilität und Wiederverwendbarkeit der Subklassen ein. Eine Lösung wäre, von abstrakten Klassen zu erben, da diese wenig oder gar keinen Code enthalten.

Objektkomposition wird dynamisch zur Laufzeit festgelegt. Objekte erhalten Referenzen auf andere Objekte. Bei der Komposition von Objekten wird auf umsichtig definierte Schnittstellen der Objekte geachtet, da die Objekte ausschließlich über diese angesprochen und verwendet werden. Die Kapselung wird nicht aufgebrochen. Objekte können zur Laufzeit ausgetauscht werden, wenn sie denselben Typ besitzen. Objektkomposition führt zu kleinen Klassen und Klassenhierachien aber auch zu mehr Objekten. Das Systemverhalten hängt von den Beziehungen der Objekte untereinander ab und ist nicht in einer einzelnen Klasse definiert. Dies führt zu einem weiteren Prinzip des wiederverwendbaren objektorientierten Entwurfs:

> Ziehe Objektkomposition der Klassenvererbung vor.

Vererbung und Objektkomposition schließen sich nicht gegenseitig aus. Vererbung und Objektkomposition arbeiten zusammen.

Diese beiden Prinzipien finden in den Entwurfsmustern von [Gamma et al. 95] vielfach Verwendung.

3.3.2 Verwendung von Entwurfsmustern

Entwurfsmuster sind z.Zt. viel diskutiert in der Welt des objektorientierten Softwareentwurfs. Bekannteste Vertreter der Entwurfsmuster und immer wieder zitiert sind [Gamma et al. 95] und [Pree 95]. Entwurfsmuster helfen wiederverwendbare objektorientierte Software zu schaffen. Sie entstanden durch die Idee nicht jedes Problem von Grund auf neu zu durchdenken, sondern zuvor erfolgreich angewendete Lösungen wiederzuverwenden. In [Gamma et al. 95] wird ein Katalog mit Entwurfsmustern aufgestellt, der Muster enthält, die häufig beim objektorientierten Entwurf auftretende Probleme lösen und sich bewährt haben. Der Einsatz von Entwurfsmustern verlangt gelegentlich mehr Aufwand beim Programmieren, als es eine näher liegende Lösung tun würde, zahlt sich jedoch durch eine erhöhte Flexibilität und Wiederverwendbarkeit aus.

Allgemein besteht ein Entwurfsmuster aus vier grundlegenden Elementen:

Musternname: Dieser Abschnitt enthält den Namen sowie eine Beschreibung der Lösungen und Auswirkungen des Entwurfsproblems. Durch die Benennung ist ein Entwurf auf einer höheren Abstraktionsebene möglich.

[4] Wiederverwendung durch Subklassenbildung wird auch oft *White-Box-Wiederverwendung* genannt. Die internen Datails der Superklassen sind für die Subklassen sichtbar.

Problemabschnitt: Der Problemabschnitt beschreibt, welches Problem gelöst werden soll und die Situation, in der das Muster anzuwenden ist. Es werden typische Entwurfsprobleme oder Klassen- und Objektstrukturen beschrieben, die zu einem unflexiblen Entwurf führen. Unter Umständen wird eine Liste von Bedingungen angegeben, die erfüllt sein müssen, um eine sinnvolle Anwendung des Musters zu erreichen.

Lösungsabschnitt: Der Lösungsabschnitt formuliert die Elemente und deren Beziehungen, Zuständigkeiten und Interaktionen, aus denen der Entwurf besteht. Das vorgestellte Muster ist eine Schablone, die eine abstrakte Beschreibung eines Entwurfsproblems bietet, keine konkrete Implementierung. Es wird eine Anordnung von Klassen und Objekten zur Problemlösung angegeben.

Konsequenzenabschnitt: Im Konsequenzenabschnitt werden die Konsequenzen der Musteranwendung erörtert (z.B. Speicherplatzverbrauch, Ausführungszeit, Einfluß auf Flexibilität, Erweiterbarkeit und Portabilität des Systems). Es werden die Vor- und Nachteile des resultierende Entwurfs beschrieben sowie Entwurfsalternativen angegeben.

Strategie

Das Entwurfsmuster *strategy* oder Strategie [Gamma et al. 95] beschreibt eine Technik, um Algorithmen, die zu einer Familie gehören, zu kapseln und somit austauschbar zu machen. In dieser Arbeit findet dieses Muster bei der Implementierung der graphischen Benutzungsoberflächenelemente Verwendung. Das Strategiemuster erlaubt es, den Algorithmus unabhängig von den ihn nutzenden Klassen zu variieren. Das Strategiemuster wird u.a. eingesetzt, wenn sich viele ähnliche Klassen nur in ihrem Verhalten unterscheiden. Strategieobjekte bieten eine Möglichkeit, Klassen mit einer von den möglichen Verhaltensweisen[5] zu konfigurieren. Die Struktur des Strategiemusters ist in Abbildung 3.6 (nach [Gamma et al. 95]) dargestellt.

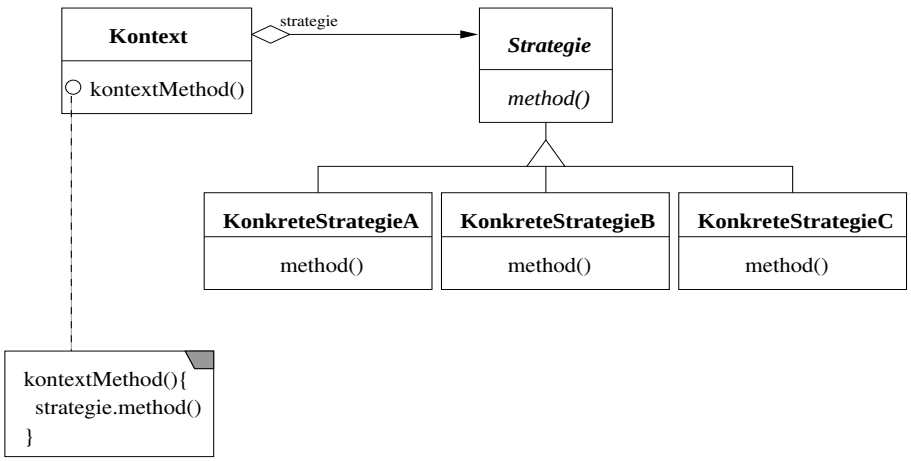

Abbildung 3.6: Kapselung von ähnlichen Algorithmen: Strategiemuster

Strategie deklariert eine Schnittstelle, die von allen Algorithmen unterstützt wird. Eine konkrete Strategie implementiert den Algorithmus. Kontext hält eine Referenz auf ein Strategieobjekt, das mit einem konkreten Strategieobjekt belegt ist. Konsequenzen dieses Musters

[5] Weitere Gründe der Anwendung des Strategiemusters siehe [Gamma et al. 95].

sind u.a. eine Auswahlmöglichkeit zwischen verschiedenen Implementierungen als auch eine erhöhte Anzahl von Objekten.

Vermittler

Das Entwurfsmuster *mediator* oder Vermittler [Gamma et al. 95] definiert ein Objekt, welches das Zusammenspiel einer Menge von Objekten in sich kapselt. In dieser Arbeit wird dieses objektbasierte Verhaltensmuster bei der Implementierung der Klassen benutzt, die zwischen Klassen vermitteln, die sonst direkt aufeinander Bezug nehmen würden. Vermittlerobjekte verhindern die explizite Bezugnahme von Objekten aufeinander. Dies fördert die lose Kopplung. Ein Vermittler reduziert die Anzahl der Verbindungen der Objekte. Die Struktur des Vermittlermusters ist in Abbildung 3.7 (nach [Gamma et al. 95]) dargestellt.

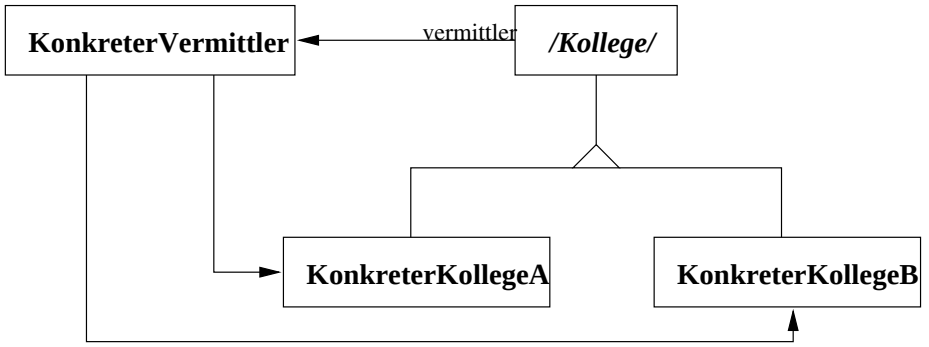

Abbildung 3.7: Förderung der losen Kopplung: Vermittlermuster

Konsequenzen dieses Musters sind unter anderem die Ersetzung von n-zu-m Beziehungen durch 1-zu-n Beziehungen sowie die Zentralisierung der Steuerung im Vermittlerobjekt, welches dadurch zunehmend kompliziert und schwer wartbar werden kann.

3.3.3 Client/ Server-Technologie

Folgender Abschnitt führt kurz und übersichtsartig in die Client/ Server-Technologie ein. Dieses Thema ist interessant im Zusammenhang mit verteilten Objekten. Die Client/ Server-Idee wird in dieser Arbeit in Abschnitt 5.3 aufgegriffen.

Charakteristisch für Client/ Server-Datenbank-Anwendungen ist die Funktionsaufteilung zwischen einem Rechner, der die eigentliche Anwendung (Client) ausführt und einem anderen Rechner, der die Datenbank hält (Datenbank-Server) [Dadam 96]. Vielfach ist mit Client/ Server-Anwendungen die Vorstellung verbunden, daß eine Ablösung von Zentralrechnern und den darauf laufenden Anwendungen stattfindet und durch kleinere, dezentral organisierte und dezidierte für bestimmte Aufgabengebiete zuständige (Server-) Systeme ersetzt werden.
Das Client/ Server Modell ist ein Software-Architektur-Modell. Das Modell beschreibt folgendes Grundschema der Kooperation:

- Der Client ergreift stets die Initiative zu einer Interaktion mit dem Server.
- Aufträge, vom Client formuliert, werden zur Ausführung an einen Server geschickt.
- Jedem Server ist bekannt, welche Dienste er anbietet.

In der Abbildung 3.8 werden die (asymmetrischen) Rollen des Client und Servers sowie die zeitliche Abfolge der Interaktionsschritte verdeutlicht.

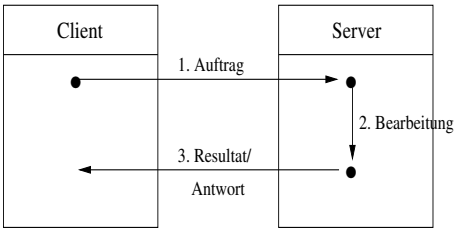

Abbildung 3.8: Client/ Server Interaktion nach [Dadam 96]

Verschiedene realisierte Client/ Server-Systeme unterscheiden sich in der Aufteilung der Funktionen der gesamten Anwendung (Applikation) auf das Client- und Serversystem. Die Komponenten einer Applikation können in drei Kategorien aufgeteilt werden:

- Präsentationsfunktionen,
- Applikationsfunktionen und
- Datenverwaltungsfunktionen.

Die Abbildung 3.9 illustriert diese Komponenten.

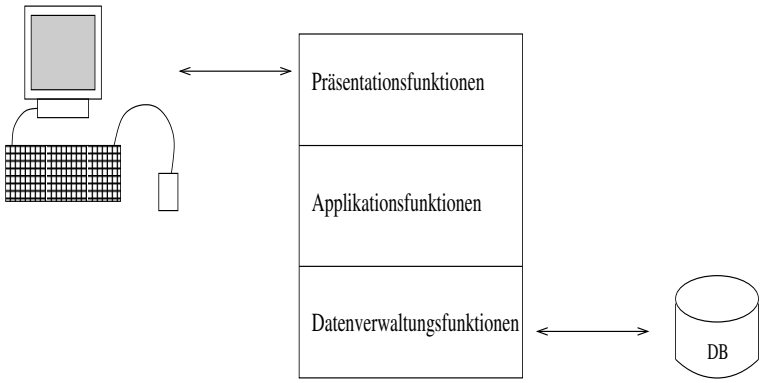

Abbildung 3.9: Komponenten einer Applikation nach [Dadam 96]

Präsentationsfunktionen sind all die Aufgaben, die mit der Eingabe von Informationen über Tastatur und Maus sowie mit deren Ausgabe auf Bildschirm und Drucker in Zusammenhang stehen. Applikationsfunktionen repräsentieren die gesamte anwendungsspezifische Programm- und Ablauflogik. Datenverwaltungsfunktionen realisieren den physischen Zugriff auf die Datenbank oder das Dateisystem mit den gespeicherten Daten. Eine klare Trennung zwischen den Komponenten ist in der Realität schwer zu vollziehen, in einigen Bereichen ist der Übergang durchaus fließend.

Eine Applikation kann nun an fünf Stellen in eine Client- und in eine Serverkomponente zerlegt werden (siehe Abbildung 3.10).

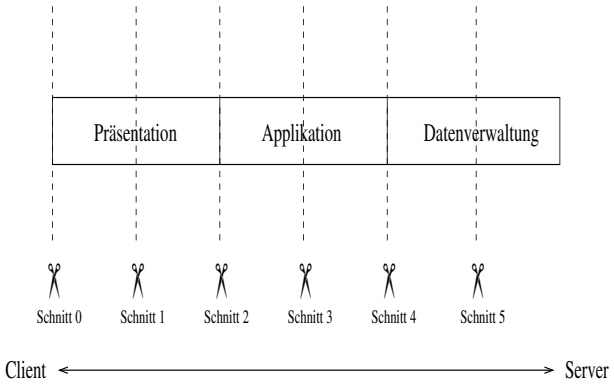

Abbildung 3.10: Möglichkeiten der Zerlegung einer Applikation nach [Dadam 96]

Schnitt 0 (keine Zerlegung) charakterisiert die Zentralrechneranwendung. Das Zentralsystem erledigt alles vom Datenzugriff über die Anwendungslogik bis zum Bildschirmaufbau.

Schnitt 1 liefert eine verteilte Präsentation. Die Präsentationfunktionen werden zwischen Client und Server aufgeteilt. Beispiel hierfür ist das X-Window-System [ScGe 90].

Schnitt 2 führt zu einer entfernten Präsentation. Die Applikation steuert den Bildschirm durch entsprechende Kommandosequenzen. Beispiel hierfür ist ein Terminalemulationsprogramm auf PC, welches mit einem Hostsystem verbunden ist.

Schnitt 3 teilt die Anwendung in zwei Komponenten und realisiert somit eine verteilte Applikation. Ein Teil der Applikation kommt auf der Client- der andere Teil auf der Serverseite zur Ausführung. Beispiel: Verlagerung der Anwendungslogik auf den Server um (entfernte) Datenbankenaufrufe zu reduzieren.

Schnitt 4 realisiert den entfernten Datenbankzugriff, der über eine semantisch hohe Schnittstelle (z.B. SQL) realisiert wird.

Schnitt 5 führt zu einem Server, der auf einen reinen Dateiserver reduziert worden ist. Beispiel: LAN-Netzbetriebssysteme, bei denen ein Dateiserver gemeinsame Daten speichert.

Diese sechs Formen der Architekturen haben sich durchgesetzt. Auf Misch- und Pseudoformen wie in [KaMu 94] erwähnt, wird hier nicht eingegangen.

Kapitel 4

Generische Dienste für Nachschlagewerke

In diesem Kapitel werden Konzepte für die zu entwickelnden Dienste für Nachschlagewerke beschrieben. Die Realisierung der Konzepte wird in Kapitel 5 vorgestellt. Die Kriterien, die in Abschnitt 2.5 (auf Seite 16) formuliert werden, fließen hier in die Anforderungen mit hinein. Das Kapitel ist wie folgt strukturiert: Der erste Teil dieses Kapitels beschreibt den Lebenszyklus digitaler Nachschlagewerke. Dieser Zyklus dient als erste Orientierung, welche Dienste benötigt werden und welche Dienstgruppen daraus entstehen. Aufbauend auf diesem Zyklus wird eine Architektur entworfen, die die herausgearbeiteten Dienstgruppen aus Nutzungssicht präsentiert. In den darauf folgenden Abschnitten wird auf die einzelnen Dienste, die in Dienstgruppen zusammengefaßt sind, näher eingegangen.

4.1 Lebenszyklus digitaler Nachschlagewerke

Zur Definition der Anforderungen an generische Dienste für Nachschlagewerke wird der Lebenszyklus von Nachschlagewerken betrachtet. Dieser Zyklus (siehe Abbildung 4.1) wird eingeleitet von einer Initialphase in der die Daten erfaßt werden (Datenakquisition), gefolgt von einem Gestaltungsvorgang (Layout), in dem die Benutzeroberfläche des Nachschlagewerks bestimmt wird. Es folgt eine meist längere Phase der Nutzung. In regelmäßigen Abständen oder bei Bedarf erfolgt eine Aktualindierung des Nachschlagewerks. Dies entspricht einer Neuauflage des Nachschlagewerks. Nach der Aktualisierung beginnt wieder die Phase der Nutzung. Aus diesem Lebenszyklus lassen sich bereits verschiedene Dienstgruppen ableiten. Jeweils eine Dienstgruppe für

- die Datenakquisition,
- die Visualisierung (das Layout) und
- die Nutzung.

Der Vorgang der Aktualisierung ist dem der Datenakquisition ähnlich. Auf eine eigene Dienstgruppe wird daher verzichtet. Die Dienste zur Aktualisierung werden in die Gruppe Datenakquisition mit aufgenommen. Die Kriterien aus Abschnitt 2.5 müssen bei der Erkennung der Dienstgruppen berücksichtigt werden. Dies gilt insbesondere für die Dienstgruppen Visualisierung und Nutzung.

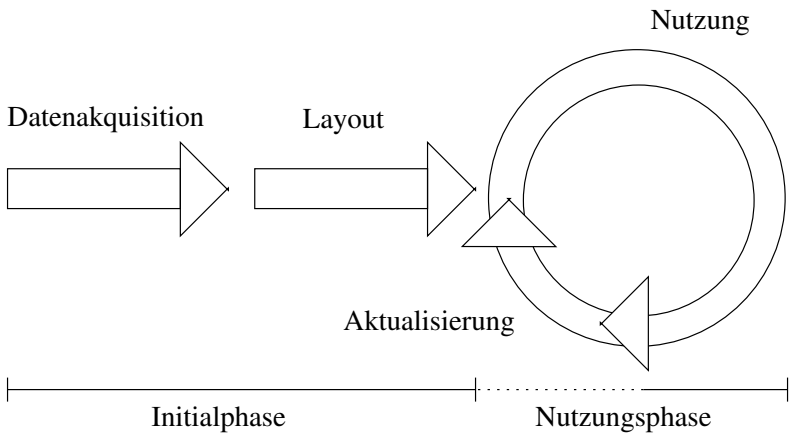

Abbildung 4.1: Lebenszyklus von Nachschlagewerken

Die Datenakquisition transformiert die Daten in eine geeignete Repräsentation[1]. Im nächsten Schritt wird das Layout für das enstehende Nachschlagewerk angelegt. Weiter werden verschiedene Möglichkeiten zur Nutzung eines Nachschlagewerks zur Auswahl angeboten. Die eingelesenen Daten sowie die Layoutangaben und die Auswahl der Nutzungsangebote werden persistent abgelegt. Dies erfordert Dienste zur Speicherung. Nun ist das Nachschlagewerk bereit benutzt zu werden. Um die Möglichkeiten des Internets zu nutzen und Eigenschaften zu bieten, die über die normalen eines Nachschlagewerks hinausgehen (*value adding*), werden Dienste erforderlich, die z.B. eine Integration in schon bestehende Anwendungen ermöglichen und schon bestehende Anwendungen und Dienste integrieren. Die Abbildung 4.2 illustriert die eben aufgeführten Dienstgruppen. Die Anordnung der Dienstgruppen in der Abbildung gibt keine Hierarchie wieder, vielmehr stehen diese Gruppen orthogonal zueinander. Sie sind weitgehend frei miteinander kombinierbar.

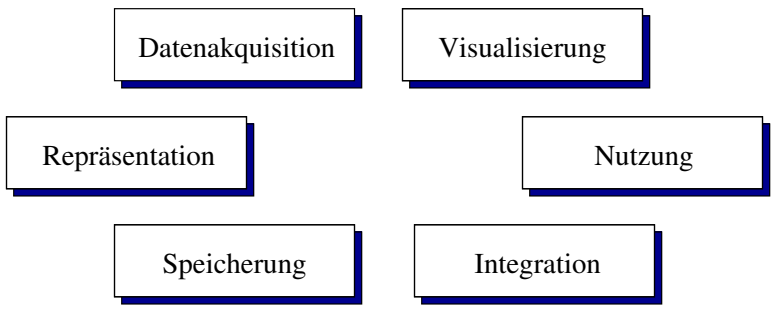

Abbildung 4.2: Die Dienstgruppen

[1] Dieser Prozess ist nicht zwingend, da es z.B. Daten in einer Datenbank gibt, welche nicht extra herausgelesen werden sollen, siehe Absatz 5.2.1.

4.2 Entwurf einer Architektur

Auf eine Architektur gibt es, bedingt durch verschiedene Blickwinkel, mehrere Sichten. Aus der Nutzungssicht ist interessant, welche Komponenten das System besitzt, welche Schnittstellen vorhanden sind und wie der Fluß der Abarbeitung ist. Bei einer Systemsicht ist der Aufbau und das Zusammenspiel der einzelnen Klassen relevant.
An dieser Stelle wird die Architektur aus der Nutzungssicht betrachtet. Eine Systemsicht der Architektur erfolgt in Kapitel 5. Grundsätzlich stellt sich die Benutzung eines Nachschlagewerks im Internet so dar, daß mehrere Benutzer von verschiedenen Rechnern an verschiedenen Orten auf ein persistentes Nachschlagewerk, auf einem Server abgegelgt, zugreifen (siehe Abbildung 4.3). Damit dieses Verfahren funktionieren kann, kommen die Dienste der in Absatz 4.1 genannten Dienstgruppen zum Einsatz. Die Architektur der Dienste stellt sich wie in Abbildung 4.4 dar. Die Pfeile geben den Datenfluß zwischen den einzelnen Komponenten an. Die Dicke der Pfeilspitze gibt die Intensität an, mit der Daten in die entsprechende Richtung fließen. Aus einer Datenquelle werden mittels Diensten aus der Dienstgruppe Datenakquisition die Daten gewonnen und in eine Repräsentation transformiert. Um ein Layout angereichert (erfolgt durch die Dienste der Gruppe Visualisierung), werden die Repräsentation und das Layout persistent abgelegt. Durch eine geeignete Visualisierung ist eine Benutzung via Internet möglich. Integrationsdienste unterstützen das Zusammenwirken mit anderen Anwendungen und Diensten. Teil der Integrationsdienste ist es, wie in der Abbildung zu erkennen, daß eine Nutzung eines Nachschlagewerks auch durch Programme erfolgen kann. Diese hier vorgestellte Architektur ist offen für:

- verschiedene Einsatzgebiete: Die Dienste lassen sich für vielerlei Arten von Nachschlagewerken verwenden.
- Erweiterungen: Keiner der Dienste ist eingebaut oder mit anderen Diensten eng verwoben. Sie sind austausch- und/oder erweiterbar.
- externe Dienste: Es besteht die Möglichkeit, die Funktionalität von Schnittstellen durch externe Dienste wie z.B. Suchfunktionen, Information Retrieval Algorithmen oder Abrechnungsmechanismen zu realisieren.

4.3 Anforderungen an eine generische Dienstbibliothek

In diesem Abschnitt werden die Dienstgruppen aus Abbildung 4.2 im einzelnen vorgestellt. Es wird erläutert, welche Aufgaben die einzelnen Dienstgruppen zu erledigen haben, bzw. welche Vorgänge sie unterstützen müssen. Die Realisierung der Dienste, bzw. Details zur Implementation finden sich in Kapitel 5.

4.3.1 Generische Unterstützung der Datenakquisition

Um die Inhalte eines Nachschlagewerks zur Verfügung stellen zu können, müssen diese erfaßt werden (siehe z.B. auch [HRRS 95]). Ein Nachschlagewerk wartet mit einer großen Quantität von Daten auf. Die Erfaßung der Daten aus einer Datenquelle und ihre Überführung in eine für die Weiterverarbeitung geeignete Repräsentation wird hier als *Datenakquisition* bezeichnet. Der Vorgang der Datenakquisition ist zu Beginn vor der Benutzung des Nachschlagewerks

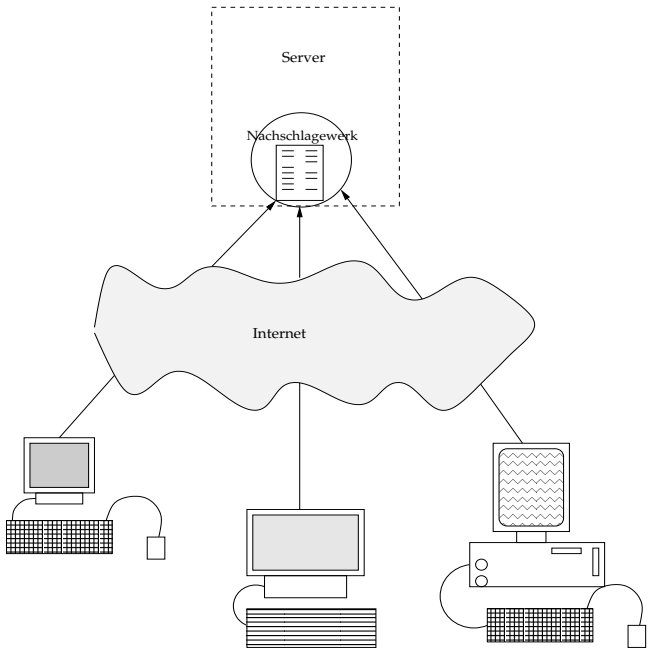

Abbildung 4.3: Benutzung eines Nachschlagewerks via Internet

durchzuführen. Bei einer Neuauflage des Werkes ist dieser Vorgang erneut durchzuführen (während bei Aktualisierungen dies nicht zwingend notwendig ist).
Es wird hier davon ausgegangen, daß die Datenquelle, bzw. der Inhalt eines Nachschlagewerks in einer Form vorliegt, die eine elektronische Weiterverarbeitung erlaubt. Es sind Datenquellen in Form von Dateien, Datenbanken oder persistente Objektspeichersysteme denkbar. Das Format der einzulesenden Daten kann also ganz unterschiedlichen Typs sein. Abbildung 4.5 stellt dies graphisch dar. Die Daten die eingelesen werden, werden in einer Repräsentation abgelegt. Das Einlesen der Daten aus der Datenquelle übernimmt ein *Parser*. Um den Implementierungsaufwand bei der Verwendung neuer Formate als Datenquellen zu verringern, wird an der Schnittstelle zur Repräsentation lediglich ein Satz Daten übergeben. Die Implementierung dieser Schnittstelle übernimmt das Einfügen in eine Repräsentation. So ist es möglich, den Vorgang des Einlesens, bzw. die Implementation, gegen eine andere Realisierung auszutauschen, ohne den Aufbau der Repräsentation anzufassen. Der Austausch der Daten erfolgt über die Methoden der Schnittstelle. Allen Datenquellen gemein ist der Vorgang der Transformation in die Repräsentation. Bei ähnlichen Datenquellen kann dieser Vorgang durch Parametrisierung der Einlesefunktion generalisiert werden. Dienste zur Speicherung der eingelesenen Daten gehören in die Dienstgruppe Speicherung (siehe Abschnitt 4.3.4).

Nach einer bestimmten Zeit der Nutzung eines Nachschlagewerks ergibt sich der Bedarf nach einer Aktualisierung. Wie in Kapitel 2 auf Seite 10 festgestellt wird, ist eine hohe Aktualität ein wesentlicher Pluspunkt des Internets. Daher sollte der Vorgang der Aktualisierung möglichst einfach und effizient gehalten werden. In Abhängigkeit von der Art des Nachschlagewerks können Stichwörter und Einträge eingefügt werden. Aktualisierungen von einzelnen Stichwörtern und Einträgen können z.B. über das Internet in einem Formular vorgenommen werden. Änderungen in der Datenbank bewirken direkt eine Änderung im Netz. Aktualisie-

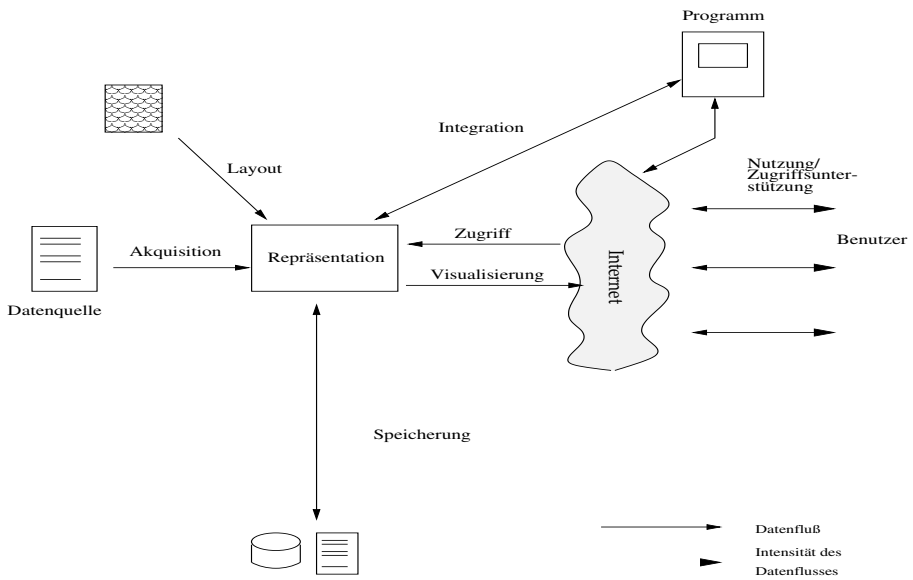

Abbildung 4.4: Architektur der Dienste

rungen werden in das bestehende Layout übernommen. Eine Aktualisierung, die das Layout betrifft, wird in der Gruppe Visualisierung abgehandelt (siehe Abschnitt 4.3.3).
Sind die Aktualisierungen so umfangreich geworden, so daß ein manuelles Einfügen oder Ändern durch einen Administrator nicht mehr effizient erscheint, so ist eine Neuauflage des Werkes ratsam. Dies bedeutet, die gesamten Daten einzulesen.

Zusammenfassend werden folgende Dienste benötigt:

Datenakquisition: Bei der Datenakquisition gibt es zwei Ansätze Generizität zu erreichen:

- Es werden Dienste entwickelt, die eine klare Trennung zwischen den reinen Einlesefunktionen und den Funktionen zum Aufbau der Repräsentation zulassen. Die Einlesefunktionen behandeln die verschiedenen Formate der Datenquellen zum Herauslesen der Daten. Die Dienste der Repräsentation übernehmen die Plazierung der Daten an der richtigen Stelle in der Repräsentation.
- Durch Parametrisierung der Dienste sind diese für Datenquellen mit ähnlichem Format einsetzbar.

Aktualisierung: Für die Aktualisierung gibt es Dienste, die zwischen dem Format der Datenquelle unterscheiden und entsprechende Inhalte in die Repräsentation einfügen. Die Neuerungen werden dem bestehendem Layout zugeordnet.

4.3.2 Generische Dienste zur Repräsentation und Zugriff

Der Vorgang der Datenakquisition transformiert die Daten in eine Repräsentation. Hierbei gibt es zwei Abstraktionsebenen. Zum einen die konkrete, physische Repräsentation, die durch eine Datenstruktur realisiert wird (interne Ebene) und zum anderen eine konzeptuelle Ebene die auf dieser Datenstruktur aufsetzt. Eine dritte Abstraktionsebene ergibt sich, wenn

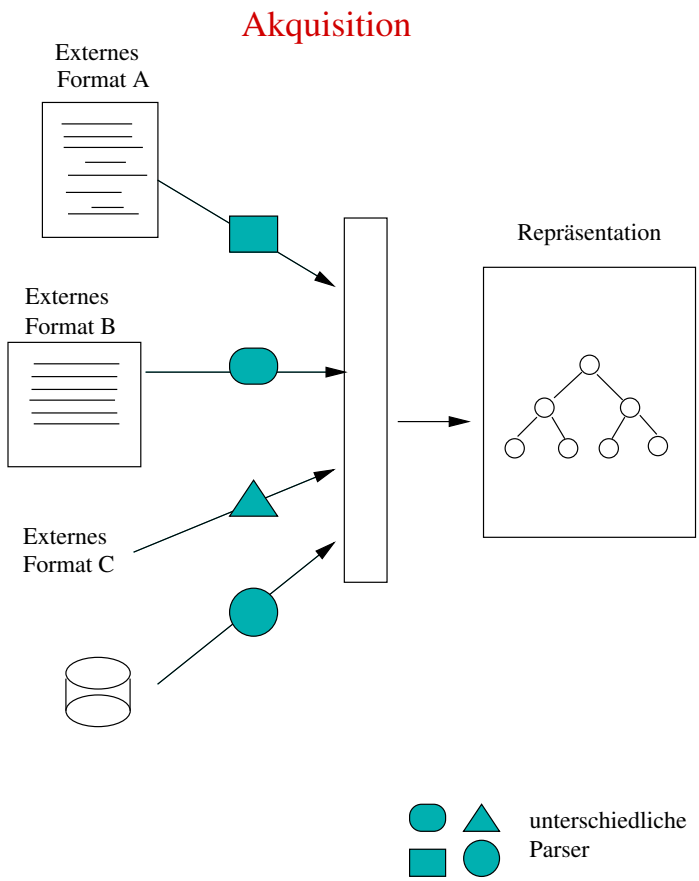

Abbildung 4.5: Datenakquisition

die Benutzergruppe unterschieden wird in Administrator und Benutzer und sich daraus verschiedene externe Sichten auf die konzeptuelle Ebene ergeben[2]. In der konzeptuelle Ebene sind Integritätsbedingungen formuliert. Bei Integritätsbedingungen handelt es sich häufig um in der Außenwelt beobachtbare Bedingungen (semantischer Natur) wie z.B. "In einem Nachschlagewerk besitzt jedes Stichwort mindestens einen Eintrag". Integritätsbedingungen regeln somit die semantische Korrektheit der Daten. Abbildung 4.6 illustriert die Abstraktionsebenen interne Ebene und konzeptuelle Ebene, wobei die konzeptuelle Ebene auf die interne Ebene abgebildet wird.

Wird eine Anfrage an das System gestellt, so meint dies die Navigation in der Repräsentation (auf der konzeptuelle Ebene) durch einen Benutzer bzw. ein Programm. In der Abbildung 4.7 wird einerseits die Benutzersicht auf eine Interaktion mit dem System dargestellt (Schritt 1 und 4), und andererseits der tatsächliche Vorgang der Navigation in der Repräsentation (Schritt 2 und 3) aufgezeigt. Ein Nachschlagewerk wird für den Benutzer durch eine visualisierte Zugriffsstruktur dargestellt. In der Zugriffsstruktur trifft der Benutzer Auswahlen, die an das System übermittelt werden (Schritt 1). Durch die Dienste des Zugriffs wird in der Repräsentation das Ergebnis ermittelt (Schritt 2), zurückgegeben (Schritt 3) und

[2]Diese Abstraktionsebenen sind analog zum drei Schichtenmodell bei Datenbanksystemen, siehe auch [Voss 96]

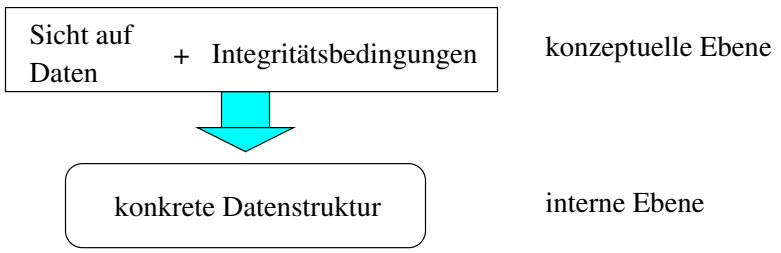

Abbildung 4.6: Interne und konzeptuelle Ebene

für den Benutzer visualisiert (Schritt 4). Die Dienste der Repräsentation sind Basisdienste, die nur für den Zugriff auf die Repräsentation zuständig sind. Dies zeigt die zentrale Bedeu-

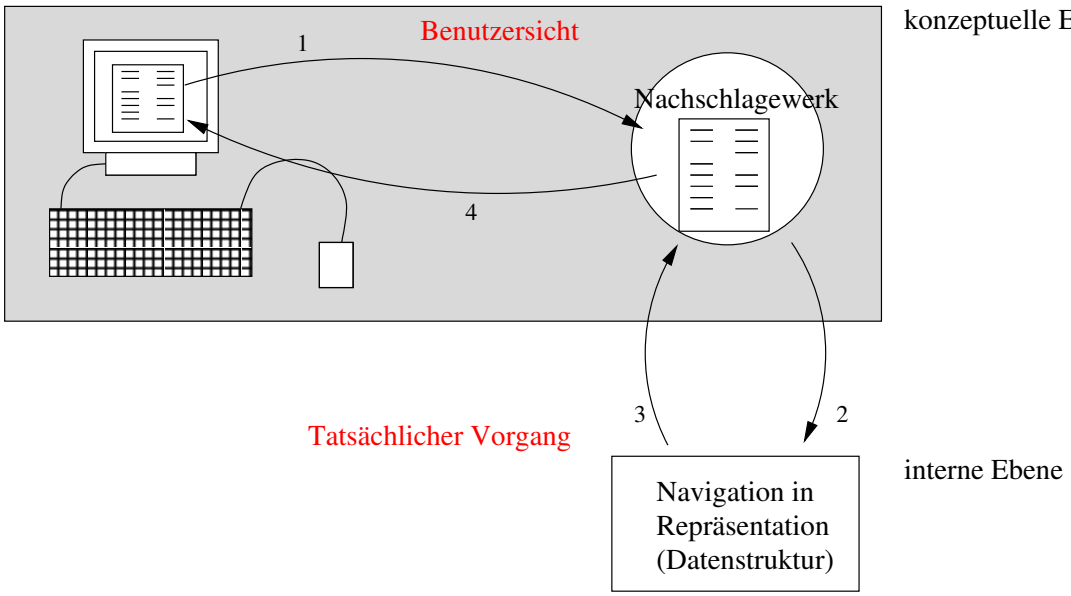

Abbildung 4.7: Zugriff auf ein Nachschlagewerk

tung der Repräsentation. Bei dem Entwurf der Repräsentation wird von der Art der Daten abstrahiert. Die interne Ebene der Repräsentation wird durch eine Datenstruktur realisiert. Diese Datenstruktur ist nur über eine Schnittstelle ansprechbar. Dies hat den Vorteil, daß eine konkrete Datenstruktur später gegen andere ausgetauscht werden kann, sofern die anderen Datenstrukturen die Schnittstelle implementieren[3]. Diese Datenstruktur ist für viele Arten von Daten verwendbar. In der Repräsentation kann auf mehreren Arten navigiert werden. Die Navigation kann z.B. von der Wurzel aus starten oder setzt unter Umständen direkt in der Datenstruktur an. Weiter gibt es neben den Benutzern auch Programme, die das Nachschlagewerk benutzen und darin navigieren.

[3]Die Konzentration auf Schnittstellen wird in [Gamma et al. 95] als ein Prinzip des wiederverwendbaren objektorientierten Entwurfs gesehen: "*Program to an interface, not an implementation*". Siehe auch Abschnitt 3.3.1.

Zusammenfassend werden folgende generische Dienste benötigt: Es werden Dienste benötigt, die den Aufbau und die Füllung der Repräsentation, neben den Diensten der Datenakquisition, dahin unterstützen, die Integritätsbedingungen zu überprüfen und einzuhalten. Weiter wird eine effiziente Unterstützung für die Navigation in der Repräsentation gefordert. Zusätzliche Dienste sind für einen Administrator anzubieten, da dieser Manipulationen in der Repräsentation vornehmen darf (auf der konzeptuellen Ebene als auch in der Datenstruktur).

4.3.3 Generische Präsentation- und Visualisierungsdienste

Als Grundelemente eines Bildschirms oder Bildschirmausschnitts lassen sich folgende Elemente feststellen [Stary 96]:

Orientierungsbereich: Der Orientierungsbereich enthält Daten über den Status der Sitzung eines Benutzers sowie den Status der Anwendung.

Arbeitsbereich: Der Arbeitsbereich beinhaltet Selektions-, Navigations- oder Anwendungsdaten.

Aktionsbereich: Der Aktionsbereich enthält eine Übersicht der verfügbaren anwendungs- und sitzungsbeszogenen Funktionen.

Die Abbildung 4.8 zeigt dies in OMT-Notation. In diese Bereiche ist die Funktionalität

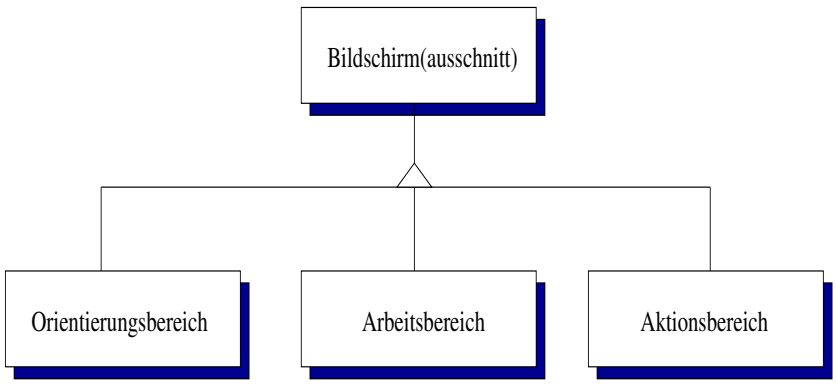

Abbildung 4.8: Grundelemente eines Bildschirm(ausschnitt)s in OMT-Notation

des Nachschlagewerks mit seinen spezifischen Funktionen bei der Visualisierung einzuordnen. Zum einen müssen die drei Bereiche angeordnet werden, und zum anderen müssen die Komponenten innerhalb der Bereiche ausgewählt, angeordnet und visualisiert werden.

In dieser Arbeit werden im Arbeitsbereich die Zugriffsstruktur und damit eng zusammenhängende Teile plaziert. In den Aktionsbereich werden grundsätzliche Funktionen wie Drucken und Beenden gestellt, als auch Navigationselemente und Mehrwertfunktionen. Der Orientierungsbereich gibt an, welches Stichwort ausgewählt wurde sowie, im Falle eines mehrstufigen Nachschlagewerks, welche Ebene gerade aktuell ist, bzw. den gesamten beschrittenen Pfad.

Anordnung der Grundelemente

Die Anordnung der Grundelemente legt ein grobes Layout für die Benutzungsschnittstelle fest. Zur Strukturierung dieser Bereiche gibt es verschiedene Vorschläge, die auf Erfahrungswissen sowie Konzepten zur Brauchbarkeit von Benutzungsschnittstellen (*styleguides*) basieren [Stary 96]. Die Abbildungen 4.9, 4.10 und 4.11 zeigen verschiedene empfohlene Arten der Strukturierung der drei Bereiche. Die Dienste unterstützen eine Standardanordnung sowie die Auswahl einer der vorgeschlagenen Anordnungen und entwicklerspezifische Angaben. Nimmt der Entwickler eigene Entscheidungen bezüglich der Anordnung vor, so wird empfohlen, sich an den Vorschlägen der *styleguides* zu orientieren. In [Red 95] wird besonders auf die sinnvolle Gestaltung der Benutzungsschnittstelle hingewiesen, da diese die Akzeptanz und den Erfolg einer Anwendung erheblich mitbeeinflußt.

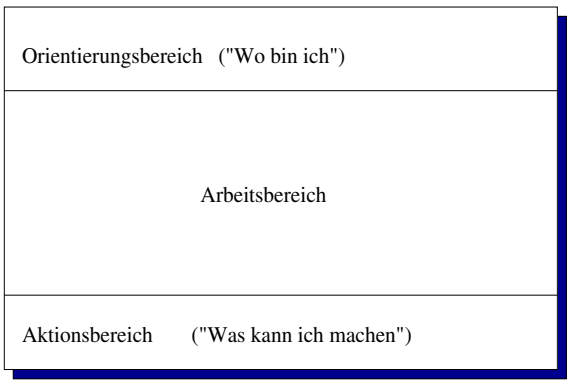

Abbildung 4.9: Grundelemente eines Bildschirm(ausschnitt)s nach [Stary 96]

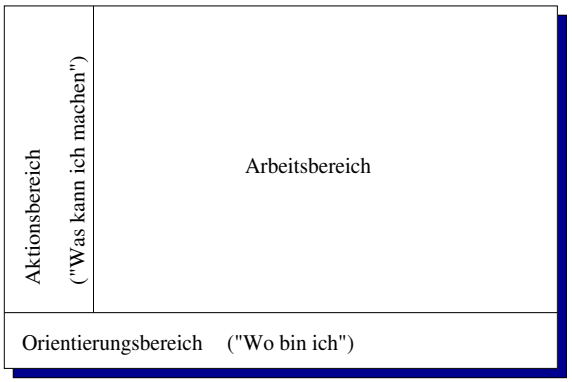

Abbildung 4.10: Grundelemente eines Bildschirm(ausschnitt)s nach [Bons 96]

Visualisierung der Zugriffsstruktur im Arbeitsbereich

Die Zugriffsstruktur, ein- oder mehrstufig, muß für den Benutzer des Nachschlagewerks visualisiert werden. Als Formen der Realisierung der Zugriffsstruktur können theoretisch alle

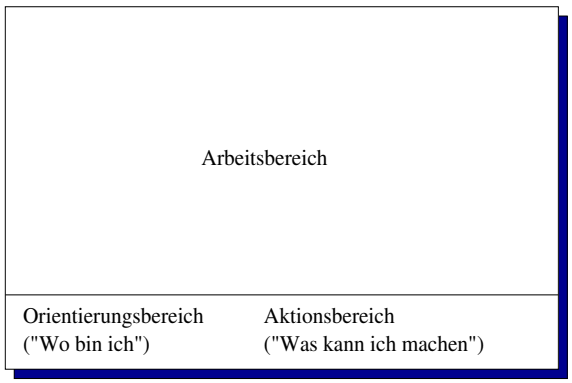

Abbildung 4.11: Grundelemente eines Bildschirm(ausschnitt)s nach [Bons 96]

Medien aus der Definition der Perzeptionsmedien von Seite 8 in Betracht kommen[4]. Am Beispiel des Wörterbuches und des Bestimmungsbuchs (siehe Kapitel 2) sei dies erläutert.

Ein Wörterbuch besitzt in der Regel eine einstufige Zugriffsstruktur.[5] Bei der Realisierung der Zugriffsstruktur mit Hilfe von diskreten Medien (z.B. Wort) bieten sich verschiedene Möglichkeiten. Diese variieren in der Verwendung von graphischer Benutzungsoberflächenelemente. Abbildung 4.12 zeigt einen Ablauf der Benuztung eines einstufigen Nachschlagewerks. Hier wurde die Zugriffsstruktur mit dem graphischen Element Schalter *Checkbox* realisiert. Weitere Elemente sind z.B. Knöpfe (*Buttons*), Listen und Auswahllisten (*Choice*).

Das Bestimmungsbuch der Biologie besitzt eine mehrstufige Zugriffsstruktur. Hier können auf den verschiedenen Ebenen jeweils andere Elemente zur Visualisierung ausgewählt werden. Die Abbildung 4.13 zeigt die Benutzung eines mehrstufigen Nachschlagewerks. Die erste Ebene in der Abbildung ist mit Schaltern (*Checkboxes*), die zweite Ebene mit einer Liste und die dritte Ebene mit Knöpfen (*Buttons*) realisiert. In der ersten Ebene des Bestimmungsbuches wird nach der Farbe der gesuchten Pflanze gefragt. Hier gibt es die Möglichkeit anstatt eines Wortes *Rot* die Farbe Rot direkt durch ein graphisches rotes Element zu präsentieren. In der Zukunft ist auch eine sprachgesteuerte Eingabe denkbar.

Zusätzlich zu der Gestaltung der Stichwörter der Zugriffsstruktur läßt sich angeben, wie die Anordnung der Zugriffsstruktur selbst sein soll. Am Beispiel einer Zugriffsstruktur alphabetischer Art wird dies erläutert. Eine alphabetische Auflistung von Wörtern kann auf mehreren Arten angeordnet werden. So gibt es die Möglichkeit in der Anzahl der Spalten zu variieren (in der Abbildung 2.4 z.B. einspaltige, vertikale Darstellung) oder wie in der Abbildung 2.6 zu sehen, ein schlichtes horizontales Auflisten des Alphabets anzugeben. Weiter kann ein Alphabet nicht nur textuell dargestellt werden, sondern auch eine graphische Aufbereitung erfahren (siehe Abbildung 4.14). Als eher sekundär kann die Farbgebung der Schrift und des Hintergrundes aufgefasst werden.

[4] Wobei der Zugriff auf Daten über Video im Moment noch im Entwicklungsstadium ist. Ebenso die Eingabe von Audiosequenzen um z.B. passende Musikstücke zu lokalisieren.

[5] Eine Ausnahme ist z.B. das Japanese-English Character Dictionary

Visualisierung des Eintrags

Ein Vorteil des Internets ist es, plattformunabhängig verschiedene Medien darstellen zu können. Diese Eigenschaft kann für die Visualisierung der Einträge genutzt werden. Ein Eintrag ist in einem oder mehreren Medientypen realisiert. Zusätzlich dazu kann angegeben werden, *wie* dieser Medientyp dargestellt wird. Ein Text muß nicht zwingenderweise als Text dargestellt werden, sondern vielleicht als Bild (Text im Bild). Die Darstellung dieser Medien übernimmt der *Browser*. Aktuelle *Browser* sind, eventuell unter Verwendung von *Plug-ins*, in der Lage alle Perzeptionsmedien wiederzugeben.

Visualisierung der Orientierungshilfe im Orientierungsbereich

Eine Orientierungshilfe zeigt den aktuellen Standort eines Benutzers in einem System an. Orientierungshilfen für Nachschlagewerke enthalten Informationen über den Weg, der zurückgelegt wurde, um dieses Dokument zu erreichen. Die Darstellung dieser besuchten Pfade ist auf verschiedenen Arten denkbar. Besuchte Pfade können als einfache Textausgaben bis hin zu komplexen graphischen Darstellungen, z.B. *Fish-eye-view* (der aktuelle Standort wird groß und zentralistisch, weiter entfernte Punkte kleiner angezeigt) oder wie in [AySt 95], wo eine 2D-Sicht auf Dokumente angeboten wird, die ein Benutzer in einer Sitzung besucht hat, visualisiert werden. Diese 2D-Sicht wird wahlweise mit dem Titel, der URL und einem verkleinertem Bild von der Seite angezeigt. Bei einer einstufigen Zugriffsstruktur ist der momentane Standort relativ klar zu bestimmen, hingegen können bei einer mehrstufigen Zugriffsstruktur durchaus Unklarheiten in bezug auf die momentan aktuelle Ebene auftreten, was eine Orientierungshilfe besonders notwendig erscheinen läßt.

Visualisierung der Navigationsfunktionen im Aktionsbereich

Zusätzlich zur Navigation über die Zugriffsstruktur gibt es weitere Formen an Bewegungsmöglichkeiten in einem Nachschlagewerk. Diese werden gesondert im Aktionsbereich angeboten. Hierzu zählen z.B. Inhaltsverzeichnisse, Register, Glossare, Dialoghistorien, retrospektive Übersichten (*Back-* und *Undo*funktionen), leserdefinierte Fixpunkte (*bookmarks*) und autorendefinierte Wegweiser (*thumb tabs*)[6]. Diese Navigationshilfen können im Aktionsbereich in geeigneter Weise graphisch aufbereitet und angezeigt werden.

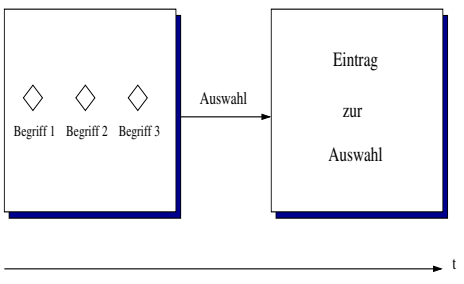

Abbildung 4.12: Ablauf der Benutzung eines Nachschlagewerks

[6]Diese Formen der Navigationsmittel werden in [Kuhlen 97] als konventionelle Metainformationsformen bezeichnet und als Orientierungshilfe angesehen. In dieser Arbeit werden sie primär als Navigationshilfen bzw. Aspekte der Nutzung verstanden.

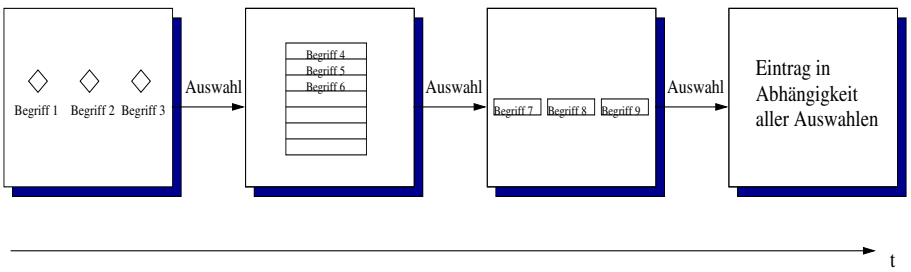

Abbildung 4.13: Ablauf bei Benutzung eines Nachschlagewerks mit mehrstufiger Zugriffsstruktur

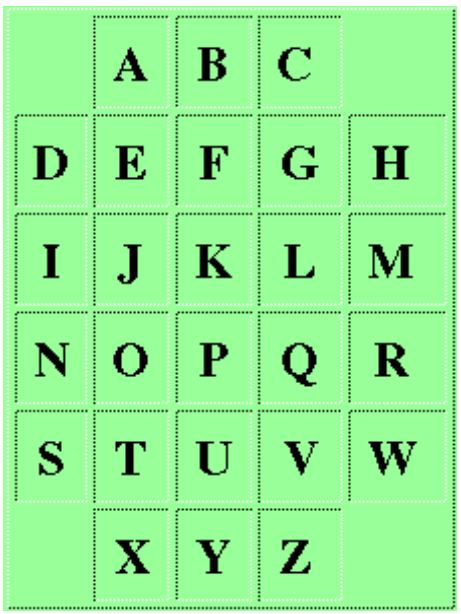

Abbildung 4.14: Graphische Aufbereitung eines Alphabets

Zusammenfassend werden folgende Dienste benötigt:

Anordnung der Grundelemente: Hier muß einerseits eine Standardanordnung möglich sein, die einfach und praktisch zu handhaben ist, und andererseits werden Dienste zur individuellen Gestaltung bereitgestellt, die einen freien Gestaltungsraum und eine erhöhte Flexibilität bieten.

Visualisierung der Zugriffsstruktur im Arbeitsbereich: Die aus den zuvor eingelesenen Daten erzeugte Repräsentation wird hier mit der Information verbunden, mit welchen Benutzungsoberflächenelementen sie dargestellt werden. Die Konstruktion der Dienste muß derart offen gelassen werden, so daß spätere, neue Visualisierungsmöglichkeiten integriert werden können.

Visualisierung der Orientierungshilfe im Orientierungsbereich: Hier werden Dienste benötigt, die es erlauben, verschiedene Arten von Orientierungshilfen zu erzeugen. Zum einen wird eine vorgefertigte Standardorientierungshilfe (z.B. eine Pfadangabe in einem

Textfeld) und zum andereren textuelle und graphische Versionen bereitgestellt. Bei der Standardorientierungshilfe sind weitere Optionen für den Aufbau möglich: Variationen in der Größe des Textfeldes und der Länge (Vollständigkeit) der Pfadangabe.

Visualisierung der Navigationsfunktionen im Aktionsbereich: Es werden Dienste benötigt, die die oben angegebenen Navigationsformen erzeugen und visualisieren. Nach dem Baukastenprinzip können die Navigationsformen ausgewählt und die Anordnung zueinander anegegeben werden.

4.3.4 Generische Speicherungsdienste

Während der Erzeugung des Nachschlagewerks entsteht eine Vielzahl von Objekten (z.B. für jedes Stichwort, jeden Eintrag, für die Information über das Layout usw.). Grundlage der Nutzung des Nachschlagewerkes ist die persistente Speicherung dieser Objekte. Für ein Nachschlagewerk werden Dienste zur Speicherung nach der Datenakquisition (und Aktualisierung) und nach der Festlegung des Layouts benötigt. Weiter ist die Verwendung von Lesezeichen (*bookmarks*) und Annotationen nur dann sinnvoll, wenn sie nach einer beendeten Sitzung persistent werden. Zur Speicherung der Objekte gibt es mehrere Mechanismen. Bei der Speicherung der Objekte (Daten) in objektorientierten Programmierumgebungen gibt es zwei Ansätze:

- Datenbanken und
- den Persistenzmechanismus der Programmiersprache.

Diese zwei Ansätze sind weiter differenzierbar. Bei den Datenbanken kann man unter anderem zwischen relationalen und objektorientierten unterscheiden. Der Persistenzmechanismus der Programmiersprache kann durch die Konzepte persistente Objektspeicher oder Objektserialisierung (Speichern in Datei) realisiert sein.

relationale und objektorientierte Datenbanken: In diesem Fall werden nicht alle Vorzüge einer Datenbank (DB) benötigt. Die DB dient lediglich als Möglichkeit das entwickelte Nachschlagewerk persistent ablegen zu können und Änderungen bequem konsistent und dauerhaftig durchzuführen. Die Benutzung eines Nachschlagewerks erfolgt, auch bei Mehrbenutzerbetrieb, nur lesend. Ein Transaktionskonzept wird erst dann benötigt, wenn mehrere Entwickler Aktualisierungen gleichzeitig durchführen dürfen. Von Vorteil ist die Fähigkeit einer DB große Datenmengen, wie sie bei einem Nachschlagewerk durchaus anzutreffen sind, effizient verwalten und zugreifbar machen zu können.

Persistenter Objektspeicher: In persistenten Objektspeichern kann jedes Objekt persistent werden. Bei einigen persistenten Objektspeichern muß während der Entwicklung der Anwendung, bzw. Erzeugung der Objekte nicht von vornherein festgelegt werden, ob diese die Laufzeit des erzeugenden Programmes einmal überleben werden. Andere wiederum verlangen eine spezielle Kennzeichnung der betreffenden Objekte. Weiter braucht der Programmierer keine expliziten Konvertierungen zwischen persistenten Ojekten und transienten Objekten vorzunehmen. Bei Objektspeichern mit orthogonaler Persistenz ist diese unabhängig von den Elementen der verwendeten Programmiersprache, ihrem Typsystem und ihren Steuerkonstrukten. Atkinson und Morrison fassen die Vorteile dieser Systeme in [AtMo 95] wie folgt zusammen:

- Steigerung der Produktivität aufgrund der einfachen Semantik,
- inkrementelle Entwicklung der Datenmodelle,
- Vermeidung von ad-hoc-Lösungen für Datenkonvertierungen und
- automatische Gewährleistung der referentiellen Integrität für das ganze System während dessen Lebensdauer.

Objektserialisierung (Speichern in Datei): Um ein Objekt in einer Datei speichern zu können, muß dieses und alle von ihm aus erreichbaren Objekte serialisiert werden. Jedes Objekt bekommt eine eindeutige Nummer (*Handle*) und wird in einen seriellen Bytestrom umgewandelt. Dieses Verfahren beansprucht bei großen Serialisierungen eine beträchtliche Zeit, die den Programmierer dazu veranlassen, die Datenstruktur passend zur Anwendung in Einheiten aufzuteilen [Atkinson et al. 96]. Diese Tatsache verletzt jedoch die Forderung von Atkinson und Morrison nach der Unabhängigkeit der Persistenz. Für einfache Standardanwendungen ist die Serialisierung jedoch bequem, ebenso für den Transport von Code zwischen zwei Adressräumen (RMI, siehe Abschnitt 3.2.1).

Welcher Mechanismus zur Speicherung eingesetzt werden kann, ist abhängig von der Anforderung an die Effizienz, der Datenmenge und von den bereitgestellten Mechanismen der verwendeten Programmiersprache.

Zusammenfassung der Dienste: Es werden Speicherdienste benötigt, die die Repräsentation und die Layoutinformation persistent machen. Die Speicherdienste sind unabhängig vom verwendeten Speichermedium.

4.3.5 Generische Zugriffsunterstützung und Nutzung

Die Nutzung eines Nachschlagewerks in einem Rechnernetz ist ein interaktiver Prozeß. Der zentrale Prozeß ist folgender: Ein Benutzer stellt Anfragen an das Nachschlagewerk. Dazu wählt er ein Stichwort aus dem Nachschlagewerk aus. Das Nachschlagewerk beantwortet diese Anfrage mit dem dazugehörigen Eintrag (siehe Abbildung 4.15). Weitere mögliche Aktionen während dieses Prozesses sind zwischen Nutzungsangeboten auszuwählen, wie z.B. Drucken, Inhaltsverzeichnis betrachten, einen Schritt zurückgehen usw.

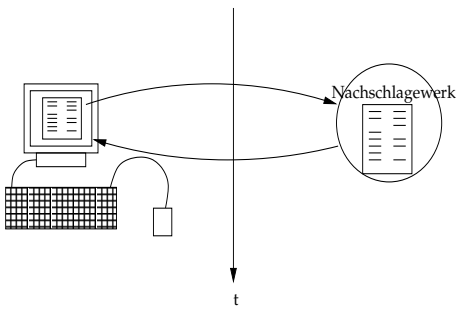

Abbildung 4.15: Interaktiver Prozeß der Nutzung eines Nachschlagewerks

Wichtig an dieser Stelle ist die Zugriffsunterstützung, die die Verwendung eines Nachschlagewerks über den Zugriff auf die Repräsentation hinaus unterstützt. Für die Zugriffsun-

terstützung werden unterschiedliche Dienste benötigt. Diese Dienste lassen sich in zwei Kategorien einteilen: Einerseits gibt es die administrativen Dienste und andererseits die Dienste, die eine grundsätzliche Benutzung eines Nachschlagewerks ermöglichen.

Zu den administrativen Diensten zählen unter anderem:

Authentisierung: Dem Entwickler eines Nachschlagewerks ist es freigestellt, ob eine Registrierung der Benutzer notwendig ist. Soll eine Abrechung erfolgen, wird eine Authentisierung benötigt. Der Vorgang der Authentisierung beeinflußt die Benutzerverwaltung. Authentisierung kann auch eingesetzt werden, um ein genaueres Benutzerprofil zu erhalten. Eine Authentisierung muß vor allem dann vorgenommen werden, wenn das Nachschlagewerk geändert werden soll (diese Person muß für Änderungsfunktionen autorisiert sein).

Benutzerverwaltung: In einer Benutzerverwaltung werden alle Benutzer eingetragen. Weiter können in einer Benutzerverwaltung die Zeiten der Nutzung festgehalten werden. Wesentlicher Zweck der Benutzerverwaltung ist das Zuordnen der Antworten auf die Anfragen der verschiedenen Benutzer. Anhand dieser Angaben kann eine Abrechnung oder auch eine Personalisierung (z.B. für Lesezeichen) erfolgen.

Abrechnung: Da das Internet nicht für kommerzielle Zwecke entwickelt wurde, wurden während der Entwicklung Punkte wie Geheimhaltung, Sicherheit, Abrechnung und Zuverlässigkeit keine Aufmerksamkeit gewidmet [FeWo 97]. Benötigen Anwendungen einige dieser Dienste, so sind diese selbst zu realisieren oder von anderen Anbietern zu importieren. In [Cous et al. 95] wird eine Architektur vorgestellt, die Transaktionen finanzieller Art für digitale Bibliotheken unterstützt. Der dort vorgestellte Ansatz berücksichtigt verschiedene Zahlungsarten (z.B. digiCash, Visa), unterschiedliche Interaktionsmodelle und Abrechnungsverfahren (z.B. feste monatliche Raten, Zeit der Benutzung oder Anzahl durchgesehener Dokumente). Innerhalb dieser Architektur werden die ökonomischen Aufgaben von denen der digitalen Bibliothek getrennt betrachtet, welches eine universalere Einsetzbarkeit der Abrechnungsmechanismen ermöglicht. Solche Mechanismen zur Abrechnung können für die Zwecke eines Nachschlagewerks im Netz ebenfalls importiert und genutzt werden.

Nachdem ein Benutzer sich angemeldet hat, startet die eigentliche Sitzung mit dem Nachschlagewerk. Innerhalb dieser Phase können verschiedene Aktivitäten sowohl im Arbeits- als auch im Aktionsbereich[7], vorgenommen werden. In den Arbeitsbereich gehören nur Elemente, die unmittelbar zum Nachschlagewerk und dessen Strukturen gehören. Im Aktionsbereich werden Navigationshilfen angeboten sowie Möglichkeiten Annotationen zu schreiben[8], Lesezeichen zu setzen, ausgewählte Inhalte zu drucken, eine Verknüpfung mit einem weiteren Nachschlagewerk zu starten, Suchen zu starten und das Nachschlagewerk verlassen zu können. Zu den Diensten einer grundsätzlichen Benutzung zählen:

Unterstützung bei der Auswahl von Stichwörtern: Nachschlagewerke können, wie oben erwähnt ein- oder mehrstufig sein. Die Abbildungen 4.12 und 4.13 zeigen dies auf der Ebene der Präsentation. Anhand eines ausgewählten Stichwortes muß die Präsentation der nächsten Ebene aufgebaut werden. Entweder ist dies eine Ebene mit neuen Stichworten oder der Eintrag.

[7] siehe Abschnitt 4.3.3 auf Seite 44 über Bildschirmelemente

[8] In [Red 95] wird angeregt, Anmerkungen auch auf Video und Audio vornehmen zu können.

Navigationshilfen: Navigationshilfen dienen der bequemen und einfacheren Bewegung in großen Mengen von Objekten[9] aller Art. Navigationshilfen wurden bereits in Abschnitt 4.3.3 im Zusammenhang mit der Visualisierung von Navigationshilfen im Aktionsbereich vorgestellt. Dort fließen ebenfalls die Mechanismen mit hinein, die in den untersuchten Werken aus dem Kapitel 2 benutzt werden, z.B. die häufig vorkommende *History*-Liste und der *GoBack-Button*. Für diese Dienste bedeutet dies, aus der vorhandenen Information Inhaltsverzeichnisse zu erzeugen oder besuchte Orte zu speichern, um eine Dialoghistorie oder retrospektive Übersichten bereitstellen zu können.

Orientierungshilfen: Als Orientierungshilfen bieten sich verschiedene Möglichkeiten an. Es gibt graphische aber auch textorientierte Varianten. Verschiedene Ideen werden bereits in Abschnitt 4.3.3 im Zusammenhang mit der Visualisierung von Orientierungshilfen im Orientierungsbereich erörtert. Die zugrundeliegenden Dienste müssen entweder den abgeschrittenen Pfad festhalten oder ermitteln den Pfad in der Repräsentation vom aktuellen Standort aus rückwärts.

Suche: Ein Vorteil der elektronischen Verarbeitung von Daten ist die Möglichkeit auf diesen mit unterschiedlichen Kriterien suchen zu können[10]. Dies eröffnet dem Benutzer neue Zugänge zu dem Datenmaterial. Die vorgegebene Zugriffsstruktur muß nicht mehr streng eingehalten werden. Über Verknüpfungen von boolschen Operatoren kann eine explizite Menge von Daten ausgewählt werden. Suchwerkzeuge werden bereits in vielen Varianten im Internet angeboten. So ein Werkzeug kann in diese Dienstarchitektur eingebunden werden.

Weiter ist es durch die elektronische Realisierung gegeben die Reihenfolge des Zugriffs zu modifizieren. So kann der herkömmliche Ablauf eines mehrstufigen Nachschlagewerks in seiner Benutzungsreihenfolge umgekehrt oder sogar permutiert werden.

All die Dienste, die zur Nutzung benötigt werden (administrative, grundsätzliche und auch die importierten) bedürfen der Dienste einer Zugriffsunterstützung. Die Zugriffsunterstützung bietet einen abtrakteren Zugriff auf die Repräsentation, als die Basisdienste der Dienstgruppe Repräsentation und Zugriff.

Zusammenfassung der benötigten Dienste: Wichtig sind hier Dienste, die für die verschiedenen Nutzungsmöglichkeiten eine Zugriffsunterstützung anbieten, die überschaubar in der Anwendung ist und dabei die vielen unterschiedlichen Anforderungen der Nutzungsmöglichkeiten erfüllen. Weiter sind die Navigations- und Orientierungshilfen auf Dienste angewiesen, die sich bestimmte Pfade explizit merken oder ad-hoc während der Benutzung die entsprechende Information aus der Repräsentation ableiten.

4.3.6 Generische Integrationsdienste

Es ist hier an zweierlei Arten von Integration gedacht:

- Auf der einen Seite ermöglichen diese Integrationsdienste eine Integration von Diensten anderer Anbieter in das Nachschlagewerk und

[9] Objekte sind hier nicht im Sinne der Programmierung gemeint. Objekte meinen hier Daten, Dokumente, etc.

[10] Dieser interessante Punkt ist nicht Hauptbestandteil dieser Arbeit. Er wird an dieser Stelle als eine wichtige Möglichkeit der Erweiterung erwähnt.

- auf der anderen Seite lassen die Integrationsdienste eine Integration des Nachschlagewerks in andere Anwendungen und Dienste zu.

Eine Integration von Diensten anderer Anbieter betrifft z.B. die Angebote für Abrechnungsmechanismen, einer Benutzerverwaltung, Authentisierungesmechanismen, Suche und Interpretation von Medientypen.

Mit einer Integration des Nachschlagewerks in andere Anwendungen und Dienste wird das Nachschlagewerk nicht als einzeln existierendes Programm eingekapselt. So gibt es die Möglichkeit auf die Einträge des Nachschlagewerks aus anderen Programmen heraus zuzugreifen. Dies sind z.B. Textverarbeitungsprogramme, die bei einer Rechtschreibprüfung auf ein entsprechendes Nachschlagewerk zugreifen (in der Abbildung 4.4 auf Seite 41 ist in der oberen rechten Ecke ein Programm symbolhaft skizziert). Weiter kann ein Ergebnis genutzt werden, um es als Eingabe in einem nächsten Nachschlagewerk zu benutzen. Ferner gibt es die Möglichkeit, daß eine HTML-Seite einen Verweis auf das Nachschlagewerk, mit dem entsprechenden Eintrag (siehe Abbildung 4.16), besitzt.

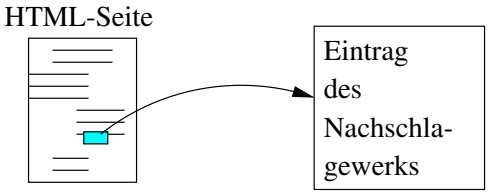

Abbildung 4.16: Integration eines Nachschlagewerks in eine HTML-Seite

Zusammenfassend werden folgende Dienste benötigt:

Dienste für die Integration externer Dienste: Durch den Aufbau der Architektur, die offen für externe Angebote ist, können Dienste anderer Hersteller sowie Inhalte anderer Nachschlagewerke eingebunden werden. Nicht unwichtig ist auch die Wahl der Programmiersprache, die weit verbreitet und plattformunabhängig sein sollte und die Anbindung externer Dienste erlaubt (z.B. C-Schnittstelle).

Dienste für die Integration mit anderen Anwendungen: Hier wird anderen Anwendungen die Möglichkeit geboten, ein bestimmtes Nachschlagewerk zu referenzieren. Weiter ist eine Spezifizierung der Ebene des Nachschlagewerks möglich. Das Nachschlagewerk verzweigt dann an die angegebene Ebene.

4.4 Mehrwertbildende Dienste

Die bisher betrachteten Dienste beziehen sich auf den Standardzugriff auf selbsterstellte oder -verwaltete Datenbestände. Neben diesen Diensten sind auch noch eine Reihe weiterer Dienste denkbar, die z.B. durch zusätzliche Zugriffsmöglichkeiten oder die Unterstützung von Benutzerpräferenzen einen Mehrwert für die Nachschlagewerke darstellen (Mehrwertdienste). In einer Zeit, in der immer mehr große Datenbestände zu den unterschiedlichsten Anwendungsdomänen im Internet verfügbar werden, spielen Mehrwertdienste, die oberhalb der Standardzugriffsdienste angesiedelt sind, eine immer wichtigere Rolle. Folgende unterschiedliche Dienste sind hier denkbar:

Permutation: Bei der Darstellung der Zugriffsstruktur gibt es, bedingt durch die Verwendung des elektronischen Mediums, Ansätze diese durch Permutationen zu erweitern. Permutationen meinen hier das Aufbrechen einer vorgegebenen Zugriffsstruktur, eventuell von einem papierbasiertem Medium herrührend, in mehrere Zugriffsstrukturen durch zyklische Vertauschung der Ebenen.

Personalisierter Index: Ein personalisierter Index bezeichnet die Zusammenstellung von Zugriffsstrukturen durch einen Benutzer aus eventuell verschiedenen Datenbeständen nach dessen Kriterien. Dieser erzeugte Index kann wiederum für andere verfügbar gemacht werden (siehe Abbildung 4.17).

Indexstrukturen für externe Bestände: Wenn die Informationsanbieter lediglich ihre Datenbestände zur Verfügung stellen, ohne Dienste zum Zugriff auf diese, ist eine Verwendung des Inhalts mit eigenen Diensten als problemlos anzusehen. Problematischer ist es, wenn (proprietäre) Dienste vorhanden sind. Hier müssen die Mehrwertdienste eine Ebene über den angebotenen Diensten bilden.

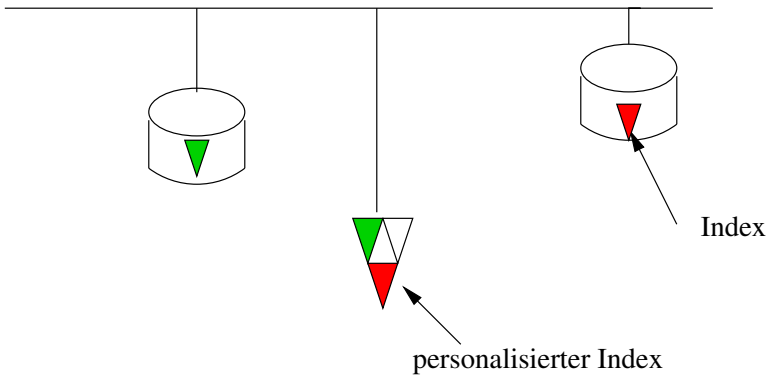

Abbildung 4.17: Personalisierter Index

Aufbruch des diskreten Zugriffs: [11] Der Zugriff auf ein Nachschlagewerk erfolgt nach einem vorgegeben Pfad. Eine Auswahl zu treffen, die zwischen zwei Stichwörtern liegt, ist in konventionellen Medien nicht möglich. Im Falle des Bestimmungsbuches der Biologie, meint dies z.B. eine Blütenfarbe anzugeben, die zwischen zwei anderen angesiedelt ist. Die Blütenfarbe wird z.B. mittels eines Schiebereglers oder einer graphischen Farbkugel eingestellt. Bei der Umsetzung der Werte des Schiebereglers in Information zur Navigation im Index kann weiterhin ein Pfad beschritten werden, der anhand des größeren Farbanteils ausgewählt wird, oder es wird kontinuierlich verfahren. Dabei werden mehrere Pfade abgeschritten und alle möglichen Ergebnisse präsentiert. In dieser Arbeit wird die Variante der diskreten Pfade betrachtet. Die Betrachtung kontinuierlicher Pfade bildet einen Ansatzpunkt der Erweiterung für diese Arbeit.

[11] Die Bezeichnungen "diskret" und "kontinuierlich" haben an dieser Stelle nicht die Bedeutung aus Kapitel 2, sondern die aus der Mathematik.

Kapitel 5

Realisierung konkreter Nachschlagewerke

Dieses Kapitel beschreibt die Realisierung einer prototypischen Dienstbibliothek für Nachschlagewerke in Rechnernetzen, wobei der Schwerpunkt auf dem Entwurf von Schnittstellen liegt. Die Anforderungen an diese Dienste sind in Kapitel 4 aufgestellt. Das Kapitel ist wie folgt aufgebaut: Im Absatz 5.1 wird erläutert, mit welchen Anwendungen die Dienste vorgestellt und getestet werden. Der darauf folgende Abschnitt 5.2 ist eingeteilt in die Dienstgruppen, die aus Kapitel 4 schon bekannt sind. Absatz 5.3 geht kurz auf die Verteilung der Andwendung mittels RMI ein. Für die Realisiserung der Dienste wird die Programmiersprache Java verwendet, deren relevante Konzepte in Kapitel 3 beschrieben werden.

5.1 Beschreibung der Anwendung

Da die Dienste für mehrere Nachschlagewerke konzipiert sind, gibt es in diesem Kapitel nicht eine konkrete Anwendung die beschrieben wird, sondern es werden, eventuell verschiedene, Anwendungen zur erklärenden Beschreibung der Dienste herangezogen. Die untersuchten Werke der vorangehenden Kapitel bieten unterschiedliche Besonderheiten, die bei der Konkretisierung, der in Kapitel 4 aufgestellten Dienstanforderungen, interessant sind. So wird z.B. des öfteren das Bestimmungsbuch der Biologie (siehe Kapitel 2 auf Seite 11) referenziert. Dieses Nachschlagewerk bietet mit seiner mehrstufigen Zugriffsstruktur und dessen unterschiedliche Ausprägungen der medialen Elemente vielfältige Ansatzpunkte.

5.2 Realisierung

In den folgenden Abschnitten werden die Anforderungen an Nachschlagewerke in Rechnernetzen aus Kapitel 4 wieder aufgegriffen und eine konkrete prototypische Realisierung vorgestellt. Die Abschnitte sind gemäß der Reihenfolge der Dienstgruppen aus Kapitel 4 eingeteilt. Die verwendete Notation wird in Anhang A beschrieben. Aus Gründen der Lesbarkeit und des Gedankens allgemeine Lösungsvorschläge vorzustellen, werden Typen und Parameter nur selten angegeben.

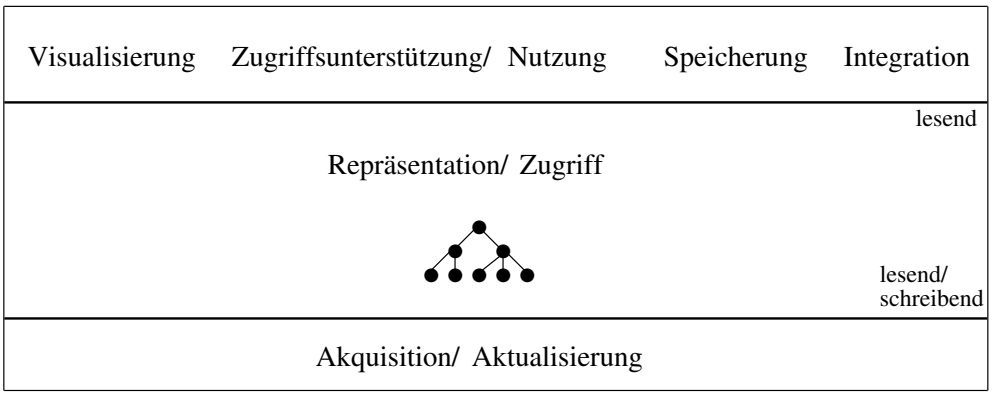

Abbildung 5.1: Architektur der Dienste aus Sicht des Systems

Abbildung 5.1 zeigt die Dienstgruppen aus Systemsicht. Auf die Repräsentation wird lesend als auch schreibend zugegriffen, wobei die lesenden Dienste überwiegen. An einigen Stellen sind die Grenzen zwischen den Dienstgruppen schwer auszumachen (z.B. Nutzung und Integration), letztendlich jedoch wird jeder Dienst einer Gruppe zugeordnet.

5.2.1 Datenakquisition

Daten (bzw. Nachschlagewerkinhalte) sind auf Speichermedien abgelegt. Als Speichermedien gibt es u.a. Dateien, persistente Objektspeicher oder Datenbanken. Diese Daten können auf verschiedenen Wegen zum Füllen des Nachschlagewerks herangezogen werden:

- Die Daten werden aus dem betreffenden Speichermedium herausgelesen und in eine interne Repräsentation überführt oder
- die Daten bleiben auf dem Speichermedium und werden über ein *Gateway* benutzt.

Die erste Variante kann auf allen oben erwähnten Speichermedien durchgeführt werden. Aus dem Grund der Vermeidung von Redundanz ist es möglich, daß die Daten eines persistenten Objektspeichers oder Datenbank in ihrer Repräsentation erhalten bleiben und von außen über ein *Gateway* auf diesen gearbeitet wird. Im Folgenden wird der Vorgang der Überführung der Daten in eine interne Repräsentation betrachtet.

Ziel ist es, die Daten, die auf unterschiedlichen Speichermedien existieren können, in eine Repräsentation zu füllen. Die Einlesefunktionen haben aus Gründen der Unhabhängigkeit keinen direkten Zugang zur Repräsentation. Daher wird eine Schnittstelle mit vordefinierten Methoden eingefügt, die von den Parsern des jeweiligen Speichermediums zu implementieren sind. Die jeweilige Realisierung der Methoden ist abhängig von dem zugrundeliegenden Speichermedium. Weiter wird eine generische, verbindende Klasse benötigt, welche die Vermittlung zwischen den Speichermedien und der Repräsentation übernimmt und den Datenfluß initiiert und kontrolliert. Die Abbildung 5.2 illustriert dies graphisch.

Die Schnittstelle `ReadInStructure` in Abbildung 5.3 gibt die benötigten Methoden sowie eine Reihe möglicher Klassen an, die diese Schnittstelle implementieren.

Die implementierenden Klassen zum Dateneinlesen erhalten keine direkte Zugriffsmöglichkeit auf die Repräsentation. Als "Vermittler" dient hier eine weitere Klasse (`Acquisition`), welche die Methoden der Schnittstelle `ReadInStructure` benutzt, um die Repräsentation

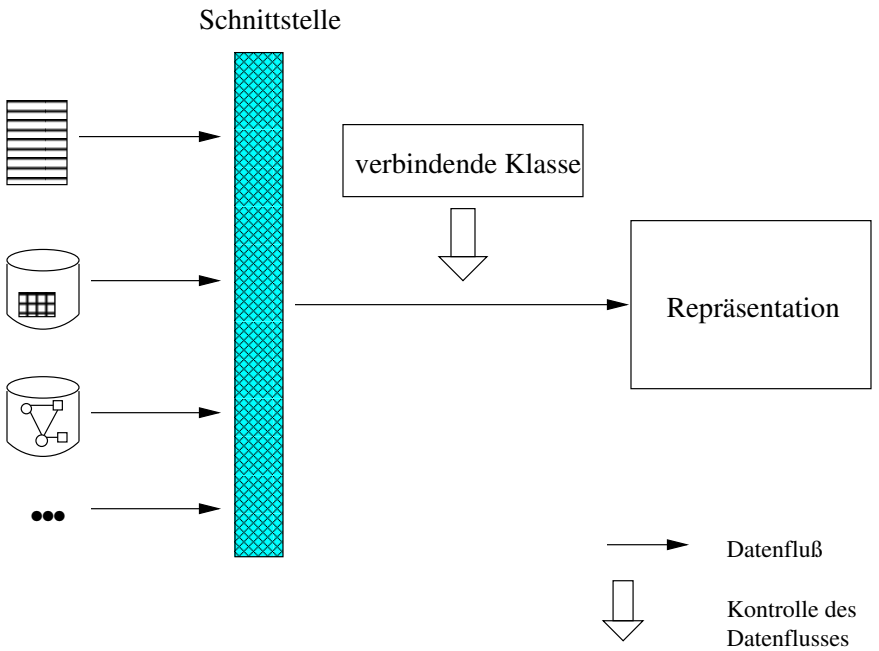

Abbildung 5.2: Anforderungen der Datenakquisition

aufzubauen und zu füllen. Diese Klasse ist unabhängig von der tatsächlichen Quelle der Daten und der konkreten Repräsentation[1]. Die vermittelnde Klasse unterscheidet bei der Füllung der Repräsentation zwischen Navigationsinformation und der eigentlichen Information, dem Dokument (oder Eintrag). Daher sind in der Schnittstelle `ReadInStructure` zwei Methoden zum Lesen der unterschiedlichen Informationen (`readInNavigation()` und `readInInformation()`) angegeben. Die beiden Methoden und die Methoden `readNext()` und `isEmpty()` ergeben den Durchlauf zum Einlesen eines Navigationsstranges sowie des dazugehörigen Eintrags. In der Abbildung 5.4 ist diese "Vermittler" Klasse (`Acquisition`) mit der Methode `fillFrom(...)`, die die Methoden der Schnittstelle `ReadInStructure` verwendet, abgebildet.

Da erst der Methode `fillFrom(...)` ein konkretes Objekt der Repräsentation einer Datenquelle (siehe Abbildung 5.3, z.B. `ReadFileInStructure` oder `ReadDBInStructure`) übergeben wird, ist die Möglichkeit gegeben, mehrere verschiedene Datenquellen in eine Repräsentation einzulesen, beispielsweise aus einer Datei, einer relationalen DB und einem Objektspeicher:

```
ReadInStructure file = new ReadFileInStructure(filename, delimiter);
ReadInStructure db = new ReadDBInStructure(dbname, tablename);
ReadInStructure os = new ReadObjStoreInStructure(osname, object_id);

Acquisition a = new Acquisition();
a.fillFrom(file);
a.fillFrom(db);
a.fillFrom(os);
```

[1] Es kann unterschiedliche Strukturen von Repräsentationen geben. Siehe Abschnitt 5.2.2.

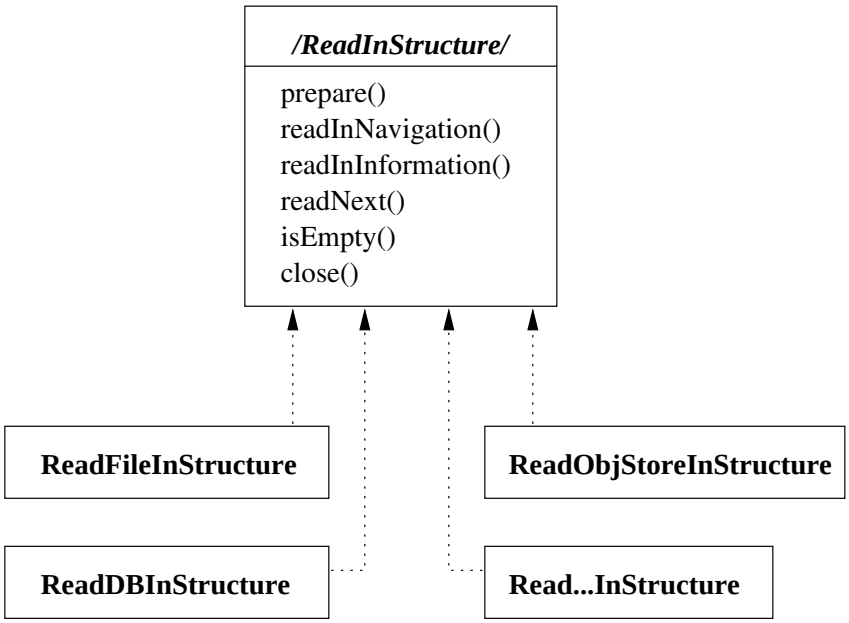

Abbildung 5.3: Schnittstelle und Klassen zum Dateneinlesen

Für jedes Speichermedien, das verwendet werden soll, muß mit einer speziellen Klasse die Schnittstelle `ReadInStructure` implementiert werden.

Es gibt jedoch auch Unterschiede im Format innerhalb eines Speichermediums. Am Beispiel der Datei sei dies verdeutlicht. In Dateien werden Wörter durch bestimmte Zeichen (Trennzeichen) getrennt. Diese Trennzeichen können entweder der Zeilenumbruch, Leerzeichen, Tabulatorzeichen oder andere Zeichen und Zeichenkombinationen sein. Eine Möglichkeit des Aufbaus einer Datei zeigt z.B. folgendes Listing, wobei $ das Trennzeichen ist:

```
$Rot$Feld$vier$Eintrag1
$Rot$Feld$fuenf$Eintrag2
$Rot$Feld$symm$Eintrag3
...
```

Dieser Aufbau der Datei simuliert die mehrstufige Zugriffsstruktur des Nachschlagewerks der Biologie. Nach dem ersten Trennzeichen steht die erste Ebene ("Rot" für die Farbe der Pflanze), nach dem zweiten Trennzeichen die Navigationsinformation für die zweite Ebene usw. Hinter dem letzten Trennzeichen ist der Eintrag vermerkt.

Die Struktur einer Datei, die eine einstufige, alphabetische Zugriffsstruktur besitzt, kann z.B. wie folgt aussehen:

```
$A$Eintrag1
$A$Eintrag2
$A$Eintrag3
$B$Eintrag4
$B$Eintrag5
$B$Eintrag6
...
```

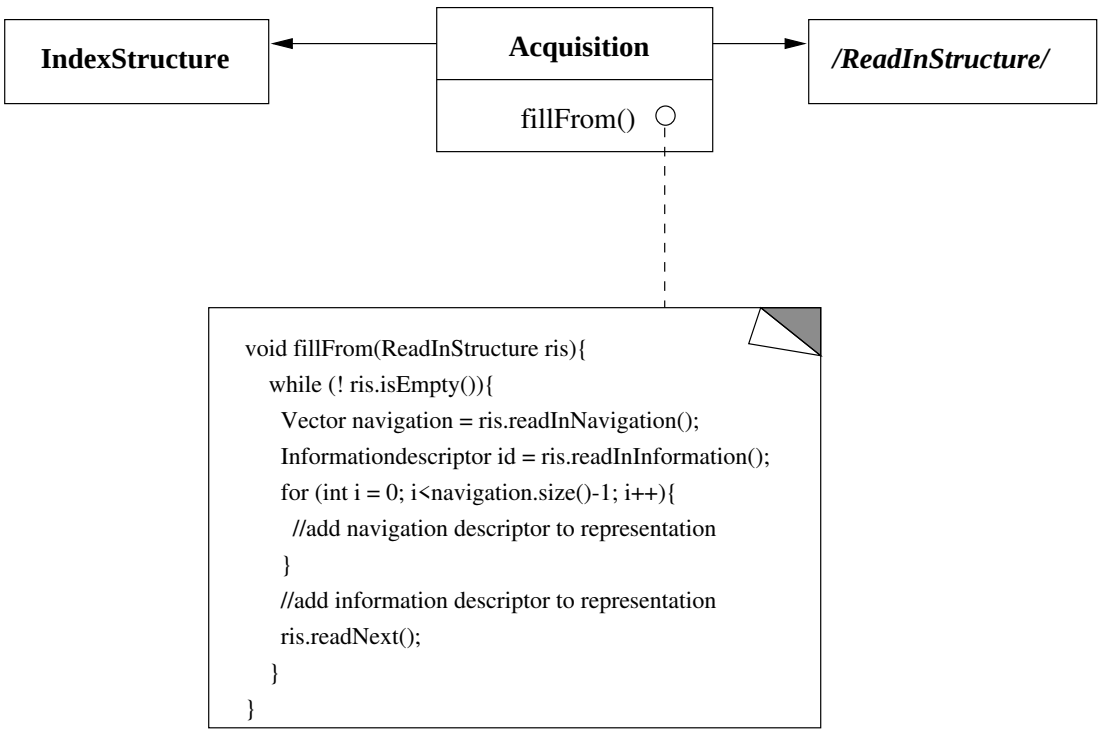

Abbildung 5.4: Verbindende, generische Klasse `Acquisition`

Eine Objektvariable vom Typ `ReadInStructure` wird zur Laufzeit mit einem konkreten Objekt einer Klasse, z.B. `ReadFileInStructure`, wie folgt erzeugt:

```
...
ReadInStructure r = new ReadFileInStructure("input.txt","$");
...
```

Der Konstruktor erwartet als Paramteter zwei Zeichenketten, wobei der erste Parameter die einzulesende Datei angibt und der zweite das Trennzeichen. Hier sind verschiedene Trennzeichen wählbar. An dieser Stelle ist eine graphische Unterstützung denkbar. Abbildung 5.5 zeigt eine graphische Benutzungsoberfläche, die eine Datei und ein Trennzeichen entgegennimmt.

Eine andere Möglichkeit die Datei aufzubauen ist, den neuen Buchstaben nur jeweils bei einem Wechsel zu notieren:

```
$A$Eintrag1
$Eintrag2
$Eintrag3
$B$Eintrag4
$Eintrag5
$Eintrag6
...
```

Wenn von einer bestimmten Normalform ausgegangen wird, wie dies bei der Klasse `ReadFileInStructure` der Fall ist, so müssen Umwandlerklassen eine Transformation in die-

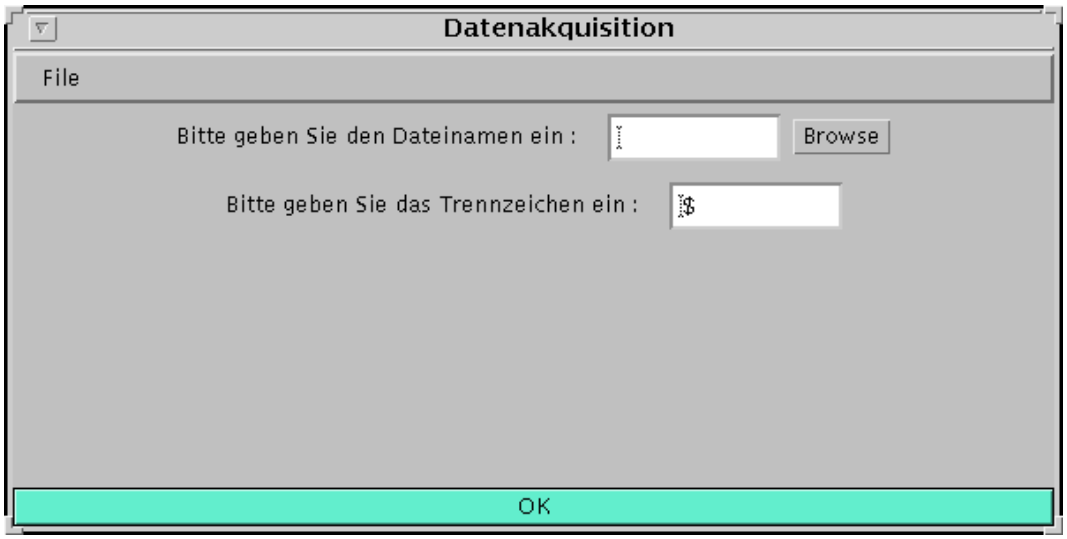

Abbildung 5.5: Graphische Unterstützung bei der Datenakquisition

ses Format vornehmen. Eine andere Lösung ist, weitere Klassen für diese Datenquelle zu implementieren und/oder über Parametrisierung größere Freiheitsgrade zu erhalten.

5.2.2 Repräsentation und Zugriff

Durch den Vorgang der Datenakquisition entsteht eine Repräsentation[2]. Hier sind die Dokumente (Einträge des Nachschlagewerks) sowie die Pfade über die diese Dokumente zu erreichen sind, abgelegt. Wobei es, wie in Abschnitt 4.3.2 angedeutet, eine interne Ebene und eine konzeptuelle Ebene der Repräsentation gibt.

Interne Ebene

Die interne Ebene dient der Speicherung der Daten in einer konkreten Datenstruktur. Die angebotenen Methoden des Zugriffs auf diese Datenstruktur beschränken sich darauf, eine Basisfunktionalität zur Verfügung zu stellen. Eine Modifikation der internen Ebene beläßt die konzeptuelle Ebene unverändert.

Als Grundlage zur Datenspeicherung wird hier eine konkrete Datenstruktur benötigt. Zur Realisierung einer Datenstruktur können unterschiedliche Implementierungen eingesetzt werden. Zur Abstraktion dieser Tatsache werden zwei Schnittstellen entworfen. Abbildung 5.6 zeigt diese Schnittstellen. Eine Datenstruktur, die die Schnittstelle `Datastructure` implementiert besteht aus mehreren Knoten, die den Typ `Datanode` besitzen. Es gibt Methoden, die nur auf die Datenstruktur anwendbar sind, wie z.B. `getRoot()`, `setRoot()` und `getDepth()`. Die Methoden beider Schnittstellen ermöglichen das Füllen der Datenstruktur sowie das Abfragen über den Inhalt der Datenstruktur. Eine Klasse vom Typ `Datanode` besitzt Methoden wie z.B. `getNavigator()`, `numberChilds()`, `getChild()`, `addChild()`, `isLeaf()`, `toString()`, `equals()`. In dieser Arbeit wird ein Baum als Datenstruktur gewählt und realisiert (Klasse `Tree`). Dieser Baum besteht aus mehreren Knoten (Klasse `TreeNode`) .

[2]Vorausgesetzt, es wird der Weg der Überführung in eine interne Repräsentation gewählt.

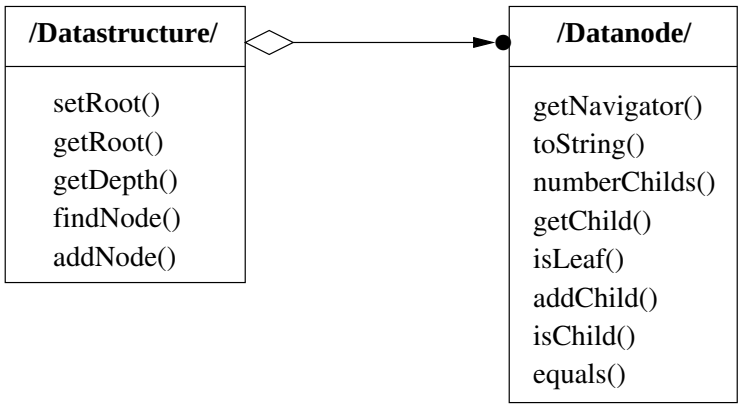

Abbildung 5.6: Die Schnittstellen `Datastructure`

Konzeptuelle Ebene

Auf der konzeptuellen Ebene wird die logische Sicht auf die Daten und ihrer Beziehungen untereinander im konzeptuellen Schema repräsentiert. Um physische Datenunabhängigkeit zu gewährleisten ist es wesentlich, daß das konzeptuelle Schema von der konkreten Datenstruktur- und Zugriffsaspekten frei ist.

In der konzeptuellen Ebene sind Integritätsbedingungen für den Aufbau einer Indexstruktur formuliert. Eine Indexstruktur ensteht durch Anwendung der Integritätsbedingungen auf die eingelesenen Daten, die anschließend in der Datenstruktur abgelegt werden. Eine Indexstruktur besteht aus Indexknoten. Abbildung 5.8 zeigt den Aufbau einer Indexstruktur aus Indexknoten. Aus der Struktur des Aufbaus der Indexstruktur lassen sich die Integritätsbedingungen erkennen.

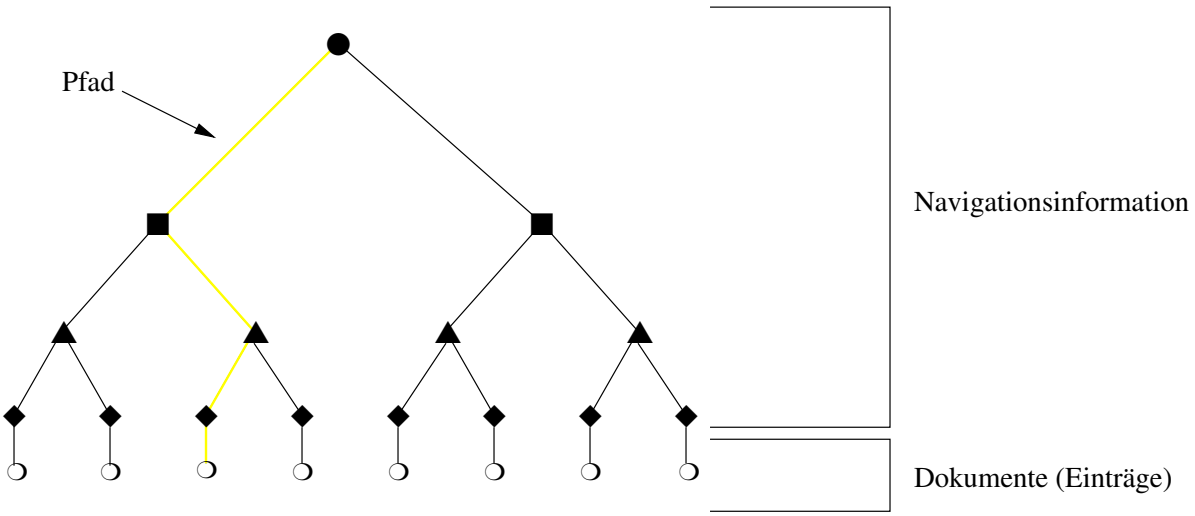

Abbildung 5.7: Eine Indexstruktur aus Indexknoten

Im Folgenden sind einige Integritätsbedingungen zur Erstellung einer Indexstruktur angegeben. Dieser Satz an Integritätsbedingungen ist als exemplarisch zu betrachten. Er kann

eventuell für verschiedene Nachschlagewerke variieren.

- Jedes Stichwort besitzt mindestens einen Eintrag.
- Nach einem Eintrag kann keine neue Ebene eingefügt werden.
- Bei einem mehrstufigen Nachschlagewerk hat jeder Knoten nur einen Elter.
- Bei einem mehrstufigen Nachschlagewerk wird keine Ebene übersprungen.
- Innerhalb einer Ebene werden nur Knoten eingordnet, die zur gleichen Kategorie gehören.

Auf der konzeptuellen Ebene gibt es eine Klasse, die dafür sorgt, daß nur solche Datenstrukturen erzeugt werden, die den Integritätsbedingungen entsprechen. Alle Modifikationsoperationen an der Datenstruktur gehen über die konzeptuelle Ebene und damit über diese Klasse. Diese Klasse heißt `IndexStructure` und hat folgenden Konstruktor:

```
IndexStructure(Datastructure)
```

Der Konstruktor `IndexStructure(Datastructure)` erzeugt eine Indexstruktur mit einer Datenstruktur vom Typ `Datastructure`.

In den Methoden `add()`, `addInfAt()` und `addNavAt()` der Klasse `IndexStructure` wird die Prüfung der semantischen Integritätsbedingungen vorgenommen und eine Indexstruktur aufgebaut, die obige Auflagen einhält.

Die Klasse `IndexNode` besitzt ähnliche Methoden wie `Datanode` jedoch mit dem Unterschied, daß diese sematisch reichhaltiger sind. Abbildung 5.8 zeigt den Zusammenhang von Indexstruktur und Indexknoten.

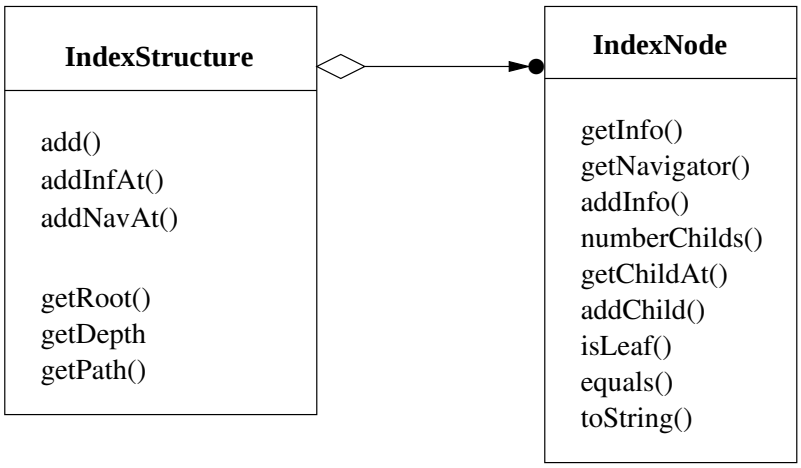

Abbildung 5.8: Die Klassen `IndexStructure` und `IndexNode`

Zusammenwirken der Ebenen

Die beiden Ebenen wirken zusammen, indem die konzeptuelle Ebene auf die interne Ebene abgebildet wird und bestimmte Konstellationen in einer Indexstruktur nicht zugelassen werden. Die Abbildungen 5.9 und 5.10 zeigen solche "verbotenen" Konstellationen. In Abbildung 5.9 besitzen nicht alle Pfade am Ende einen Eintrag. Dieses ist nicht zugelassen. In

Abbildung 5.10 sind die Stichwörter, die auf einer Ebene sind, nicht alle von einer Kategorie.

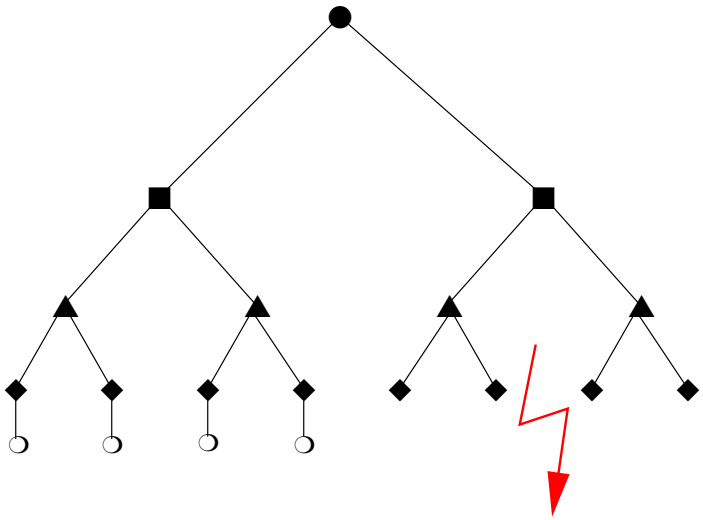

Abbildung 5.9: Nicht gültige Indexstruktur

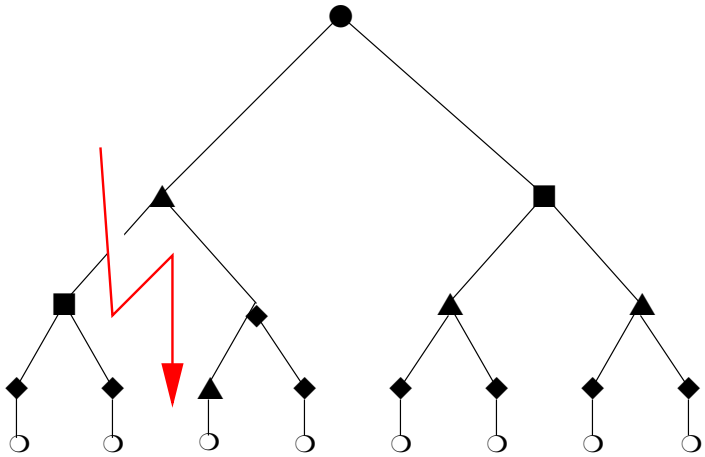

Abbildung 5.10: Ungültige Indexstruktur

Zugriff auf die Indexstruktur

Das Navigieren in der Indexstruktur erfolgt durch dIe Methoden `getRoot()`, `setRoot()`, `getChild()` und `findNode()`. Der jeweils aktuelle Knoten wird während der Navigation als neue Wurzel vermerkt. Weitere Navigation in der Indexstruktur findet von diesem Knoten aus statt. An dieser Stelle ist die Verwaltung von mehreren Benutzern interessant. Die Antworten des Nachschlagewerks müssen der entsprechenden Anfrage zugewiesen werden. Dies bedeutet, daß jeder Benutzer z.B. ein eigenes Indexobjekt erhält, über welches er mit dem Nachschlagewerk kommuniziert.

5.2.3 Präsentation und Visualisierung

In Abschnitt 4.3.3 wird angegeben, welche Grundbausteine für eine Präsentation von Nachschlagewerken in Rechnernetzen benötigt werden und wie verschiedene Anordnungen aussehen. Die Visualisierung betrifft die Anordnung der Grundelemente, die Indexstruktur und die Dokumente. Durch die Verwendung der Programmiersprache Java erfolgt eine Festlegung auf das AWT (*abstract window toolkit*), die graphische Oberflächenbibliothek. Diese Festlegung bestimmt das Layout sowie den Entwurf der Schnittstellen entscheidend. Abbildung 5.11 zeigt beispielhaft eine Realisierung eines Nachschlagewerks mit textueller Orientierungshilfe im oberen Drittel des Bildschirmausschnitts angeordnet, einem Arbeitsbereich in der Mitte und zwei Nutzungsfunktionen im unteren Bereich.

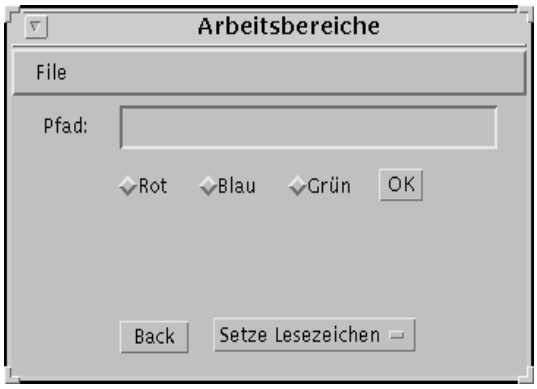

Abbildung 5.11: Nachschlagewerk mit den drei Grundelementen

Visualisierung der Zugriffsstruktur/ Indexstruktur

Die Indexstruktur, die aus dem Einlesevorgang entstanden ist (siehe Abschnitt 5.2.1), wird von einem Anwender mit einem Layout versehen. Layout meint hier die Darstellung der Navigationsinformation und der Dokumente mit den Möglichkeiten des AWT. Mit dem Layout wird eine Möglichkeit geschaffen, Navigationsinformation zu visualisieren und in der Phase der Benutzung bestimmte Navigationsinformation auszuwählen. Bei der Navigationsinformation kann bei einem mehrstufigen Nachschlagewerk das Layout jeder Ebene unterschiedlich gewählt werden (siehe Abbildung 4.13).

Abbildung 5.12 zeigt die abstrakte Klasse `NavigationDescriptorLayout`. Mit ihren Methoden realisiert sie den dynamischen Vorgang der Bindung eines Layouts an eine Ebene des Zugriffs. Die Methode `display(...)` sorgt, unter Verwendung der (abtrakten) Methode `navigatorChild(...)` für das Visualisieren der jeweiligen Indexknoten der Indexstruktur. Jede Subklasse implementiert ihre eigene Ereignisbehandlung. Bei der Ausführung einer Aktion aufgrund eines Ereignisses wird die Methode `selected(...)` aufgerufen. Diese Methode entscheidet, ob ein Eintrag angezeigt werden muß, oder die nächste Ebene mit Indexknoten visualisiert wird. Die Klassen, die von der abstrakten Klasse `NavigationDescriptorLayout` erben, müssen nach ihren Bedürfnissen die Methoden überschreiben und vor allem ihre spezielle Ereignisbehandlung hinzufügen (hier findet das Konzept der inneren Klassen praktische Verwendung, siehe Abschnitt 3.2.2) und als "Lauscher" (*Listener*) installieren[3]. Die Ereig-

[3]Dieser Aufbau der Klassen entspricht dem Entwurfsmuster Strategie, siehe Abschnitt3.3.2, Seite 33.

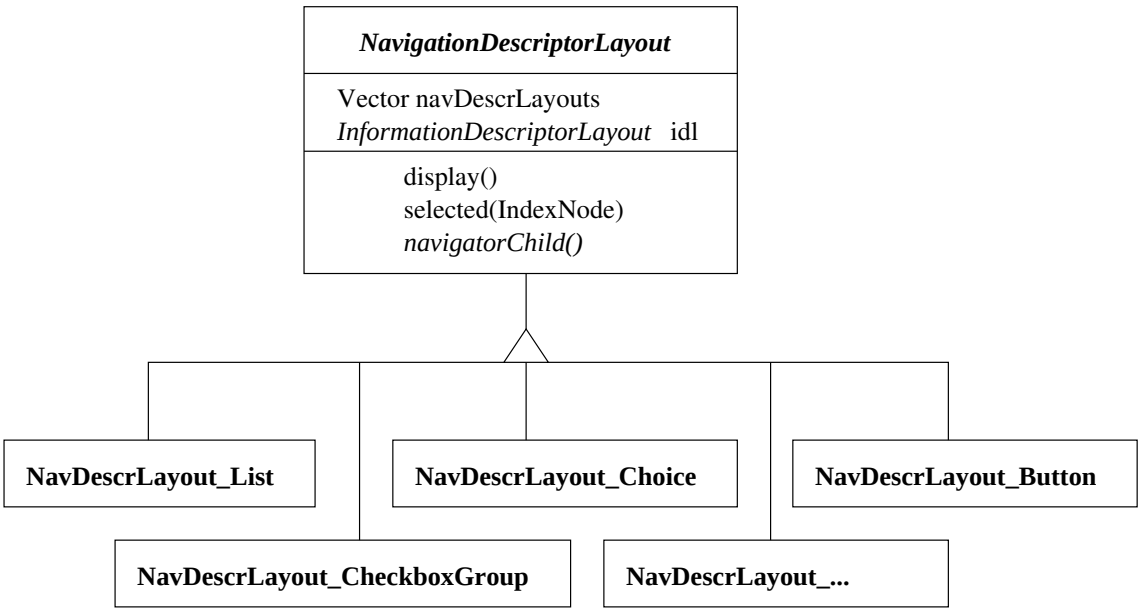

Abbildung 5.12: Die abstrakte Klasse `NavigationDescriptorLayout`

nisbehandlung sorgt für den weiteren Ablauf der Anwendung. Hier wird das Konzept der abstrakten Klasse gewählt, und keine Schnittstelle, da der Bedarf besteht, Code zu extrahieren, der allen Subklassen gemeinsam ist. Zukünftige Layouts (z.B. bei Erweiterung des AWTs) sind einzufügen, indem neue Subklassen von `NavigationDescriptorLayout` definiert werden.

Folgendes Codebeispiel zeigt die Erzeugung des Layouts der Navigationsinformation für ein mehrstufiges Nachschlagewerk:

```
...
NavigationDescriptorLayout ndl1 = new NavDescrLayout_Button();
NavigationDescriptorLayout ndl2 = new NavDescrLayout_Choice();
NavigationDescriptorLayout ndl3 = new NavDescrLayout_CheckboxGroup();
Vector descrLayouts = new Vector(3);
descrLayouts.addElement(ndl1);
descrLayouts.addElement(ndl2);
descrLayouts.addElement(ndl3);
...
```

Im obigen Beispiel handelt es sich um die Definition des Layouts für drei Ebenen eines Nachschlagewerks. Die erste Ebene wird mit Knöpfen (*Buttons*) dargestellt, die zweite mit einem aufklappbarem Auswahlmenü (*Choice*) und die dritte mit Schaltern (*CheckboxGroup*), wobei nur einer gewählt sein darf. Diese Layoutangaben, die in drei Objekten abgelegt sind, werden in einem Objekt `descrLayouts` zusammengefaßt.

Die Dokumente, die am Ende eines Navigationsstranges stehen, bestehen aus beliebigen multimedialen Daten. So gibt es Einträge, die aus Text, Bildern und/oder aus Audiodaten bestehen (siehe auch Abschnitt 2.5 auf Seite 16). Es wird hier davon ausgegangen, daß diese Dokumente im Hypertextformat vorliegen und ihre Darstellung daher dem Browser überlassen

werden kann. Die Gestaltung der Dokumente im Hypertextformat läßt die Verwendung beliebiger Perzeptionsmedien zu.

Die Layoutklassen benötigen Informationen aus der Indexstruktur. Die Layoutklassen müssen wissen, welche Navigationsinformationen und Dokumente die Struktur enthält, um diese dynamisch mit dem aktuellen Layout der Ebene zu verknüpfen und visualisieren zu können. Daher gibt es eine Klasse, die zwischen der Repräsentation und den Klassen für das Layout vermittelt: Die Klasse `Layout`. Abbildung 5.13 stellt diesen Zusammenhang dar. Die Klasse `Layout` hält Referenzen auf Objekte, die die Art der Darstellung der Navigationsinformation (`NavigationDescriptorLayout`) sowie die des Eintrags (`InformationDescriptor-Layout`), wissen. Die Klasse `Layout` benötigt weiter einen Zugriff auf die Indexstruktur. Die Indexstruktur wird der Methode `showIt(...)` als Parameter übergeben. In dieser Methode findet der Aufruf statt, die erste Ebene der Indexknoten mit der ersten Layoutangabe zu verknüpfen sowie die restlichen Layoutangaben weiterzureichen. Der Mechanismus, erst der Methode `showIt(...)` die Indexstruktur zu übergeben, hat den Vorteil, daß ein einmal entworfenes Layout für weitere (ähnlich aufgebaute) Indexstrukturen genutzt werden kann. So kann z.B. *corporate design* unterstützt werden.

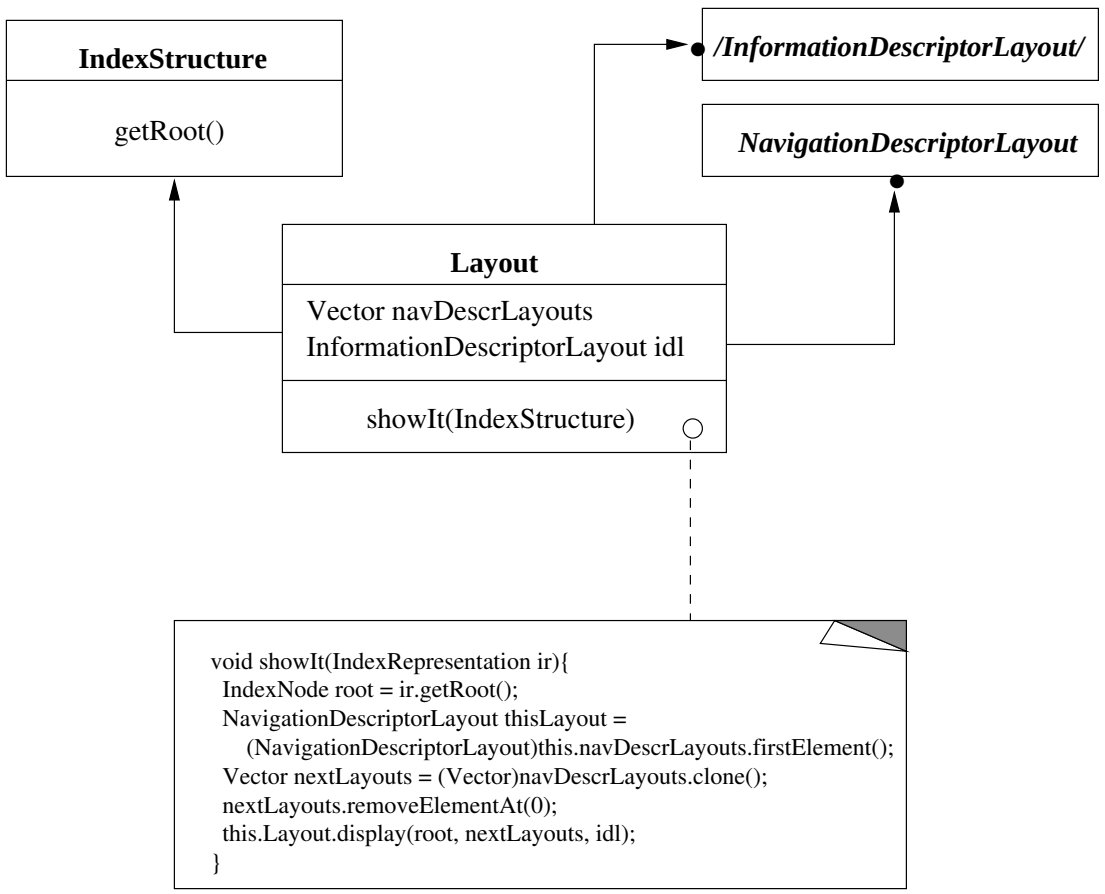

Abbildung 5.13: Die Klasse `Layout`

Orientierungshilfen im Orientierungsbereich

Eine Orientierungshilfe gibt Auskunft über die aktuelle Position eines Benutzers innerhalb eines Systems. Orientierungshilfen sind in verschiedenen Darstellungen möglich (siehe Abschnitt 4.3.3). Um verschiedene Darstellungen einbinden zu können, unabhängig von deren konkreter Gestalt, wird eine einheitliche Schnittstelle gefordert, die dies ermöglicht. Hier wird eine Schnittstelle `OrientationArea` angeboten, die von ihren implementierenden Klassen fordert, die Methode `display(String selected)` geeignet zu implementieren (siehe Abbildung 5.14). Bei jeder Wahl eines Stichwortes wird die Methode `display()` mit einer `String` Repräsentation des selektierten Stichworts an dem Objekt aufgerufen, das die Orientierungshilfe repräsentiert. Diese ist nun für die weitere Aufbereitung der übermittelten Information zuständig. Auf diese Weise ist es gegeben, daß von der Präsenation her unterschiedliche Realisierungen von Orientierungshilfen dargestellt werden können. Jede Orientierungshilfe "weiß", wie sie sich, bzw. das selektierte Stichwort, darzustellen hat. In der prototypischen Realisierung wird eine textuelle Orientierungshilfe bereitgestellt. Im Falle eines mehrstufigen Nachschlagewerks werden die besuchten Ebenen in dieser textuellen Hilfe zu einem Pfad aneinandergereiht und angezeigt.

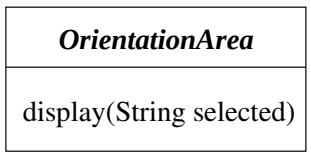

Abbildung 5.14: Die Schnittstelle `OrientationArea`

Navigations- und Nutzungsfunktionen im Aktionsbereich

Navigationsfunktionen realisieren die verschiedenen Zugriffsmodalitäten, die ein Nachschlagewerk besitzt. Nutzungsfunktionen stellen grund- und zusätzliche Bedienungsmöglichkeiten eines Nachschlagewerks dar. Navigations- und Nutzungsfunktionen werden nach dem Baukastenprinzip ausgewählt und im Aktionsbereich zusammengefügt. Die Erzeugung dieser Komponenten wird im Abschnitt 5.2.5 behandelt.

Anordnung der Grundelemente

Die Anordnung der drei Grundelemente (siehe Abschnitt 4.3.3) eines Bildschirm(ausschnitt)s wird durch den Anwender festgelegt. Dazu wird eine Klasse benötigt, die zum einen ein Standardlayout vorschlägt aber auch selbstdefinierte Layouts zuläßt. Die Klasse `Arrange` (siehe Abbildung 5.15) realisiert diese Anforderungen. Durch die Initialisierung von Exemplarvariablen wird ein Standardlayout angeboten. Die Methode `arrangeIt()` greift auf diese Variablen zurück, um die entsprechenden Bereiche in diesem Layout einzufügen. Durch das Überladen der Methode mit Angabe weiterer Parameter, welche die neuen Positionen angeben, wird eine selbstdefinierte Anordnung ermöglicht. Dem Konstruktor der Klasse werden die Objekte übergeben, die die Bereiche darstellen. Auch hier wird über initialisierte Exemplarvariablen ein Standard bereitgestellt. Folgendes Codebeispiel erzeugt zunächst die Bereiche für die Orientierung, Arbeit und Aktion, um sie dann dem Konstruktor der Klasse

`Arrange` zu übergeben. Der Aufruf der Methode `arrangeIt(...)` des `Arrange` Objekts legt die Ausrichtung der Komponenten fest.

```
...
OrientationArea oa = new OrientationTextArea();
WorkArea wa = new WorkAreaPanel();
ActionArea aa = new ActionAreaPanel();

Arrange ar = new Arrange(oa, wa, aa,...);
ar.arrangeIt("South", "Center", "North");
```

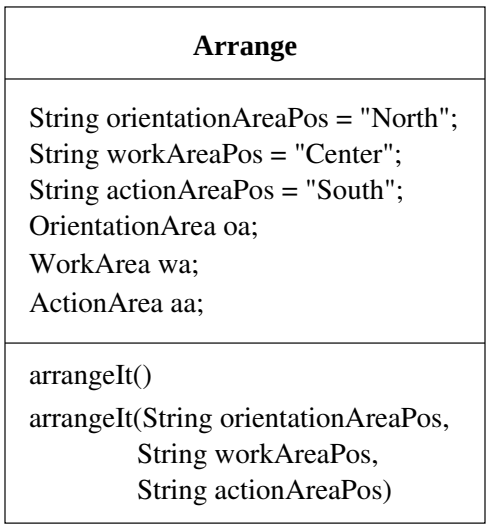

Abbildung 5.15: Die Klasse `Arrange`

Bei der Anordnung der Bereiche ist eine graphische Unterstützung für den Benutzer denkbar. Abbildung 5.16 stellt eine mögliche Benutzungsschnittstelle dar, die prototypisch implementiert ist. Diese Benutzungsschnittstelle gibt im oberen Drittel drei Vorschläge an, die vom Aufbau her empfohlen werden. Darunter ist eine selbstdefinierte Variante möglich.

5.2.4 Speicherungsdienste

Die Speicherungsdienste nehmen eine Sicherung der Darstellung der Ebenen des Nachschlagewerks sowie der dazugehörigen Indexstruktur vor. Diese können auch einzeln abgespeichert werden. Die Speicherungsdienste sind unabhängig von dem zugrundeliegenden Speichermedium. Die Schnittstelle `Store` gibt die Methoden an, die die Klassen implementieren müssen, die die Speichermedien repräsentieren (siehe Abbildung 5.17). Der Klasse `Saver` wird in der Methode `saveTo()` ein Objekt einer Klasse übergeben, welche die Schnittstelle `Store` implementiert. Über die Konstruktoren der Klasse `Saver` werden die Indexstruktur und das Layout, einzeln oder gemeinsam, übergeben. Folgendes Codebeispiel erzeugt Objekte vom Typ `Store`, `Saver` Objekte mit unterschiedlichen Konstruktoren und ruft die Methode `saveTo()` auf. Die Informationen werden hier auf unterschiedlichen Speichermedien abgelegt.

```
...
```

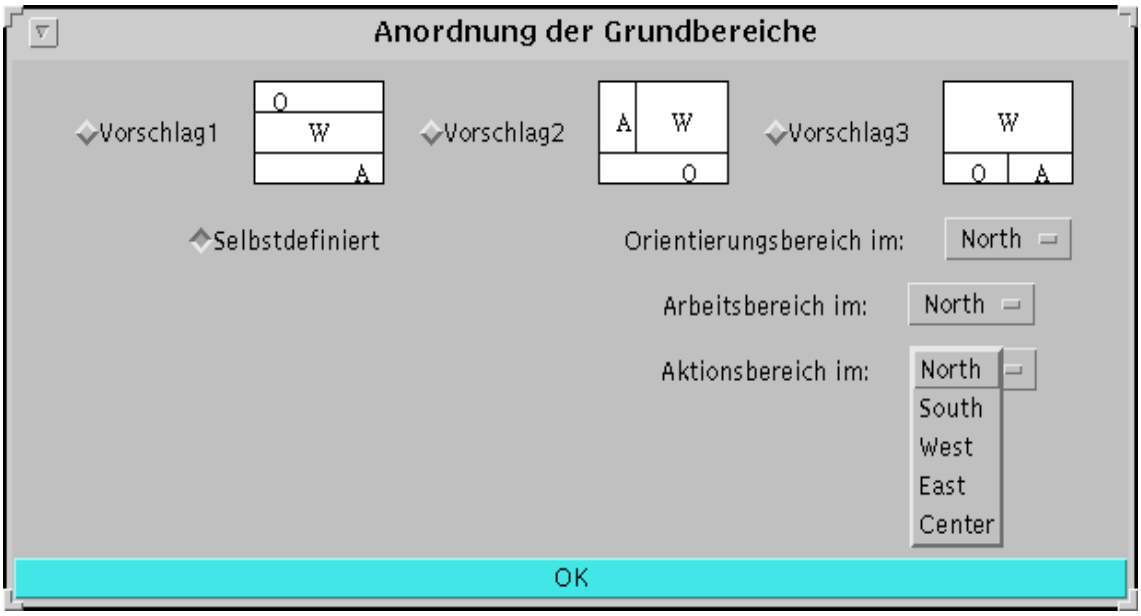

Abbildung 5.16: Graphische Unterstützung bei der Positionsfestlegung

```
Store file = new StoreInFile(filename);
Store db = new StoreInDB(dbname);
Store os = new StoreInObjStore(osname);

Saver sf = new Saver(indexstructure);
Saver sdb = new Saver(layout);
Saver sos = new Saver(indexstructure, layout);
sf.saveTo(file);
sdb.saveTo(db);
sos.saveTo(os);
...
```

Sind Indexstruktur und Layout unabhängig voneinander abgespeichert, so muß bei einer Zusammenführung einer Indexstruktur und einem Layout eine Prüfung vorgenommen werden, ob sie zusammenpassen.

Im Folgenden wird, mit Hilfe der Objektserialisierung, der Speichermechanismus der Programmiersprache Java beschrieben. Bei der Objektserialisierung wird ein Objekt inklusive aller vom Objekt aus direkt und transitiv erreichbaren Objekte gespeichert. Das Speichermedium ist die Datei. Folgendes Codebeispiel illustriert die Erzeugung eines `Store` Objektes sowie eines `Saver` Objektes. Das `Saver` Objekt erhält mit seinem Konstruktor das zu speichernde Objekt (`ar` ist ein Objekt vom Typ `Arrange`, welches zuvor erzeugt wurde. In diesem Objekt sind sowohl Repräsentation als auch Layoutinformationen enthalten. Die Klasse `Arrange` gehört zur Dienstgruppe Präsentation und Visualisierung, siehe Abschnitt 4.3.3). Das `Saver` Objekt erhält die Nachricht `saveTo()` mit dem zuvor angelegten `Store` Objekt.

```
...
Store file = new StoreInFile("Ausgabedatei");
```

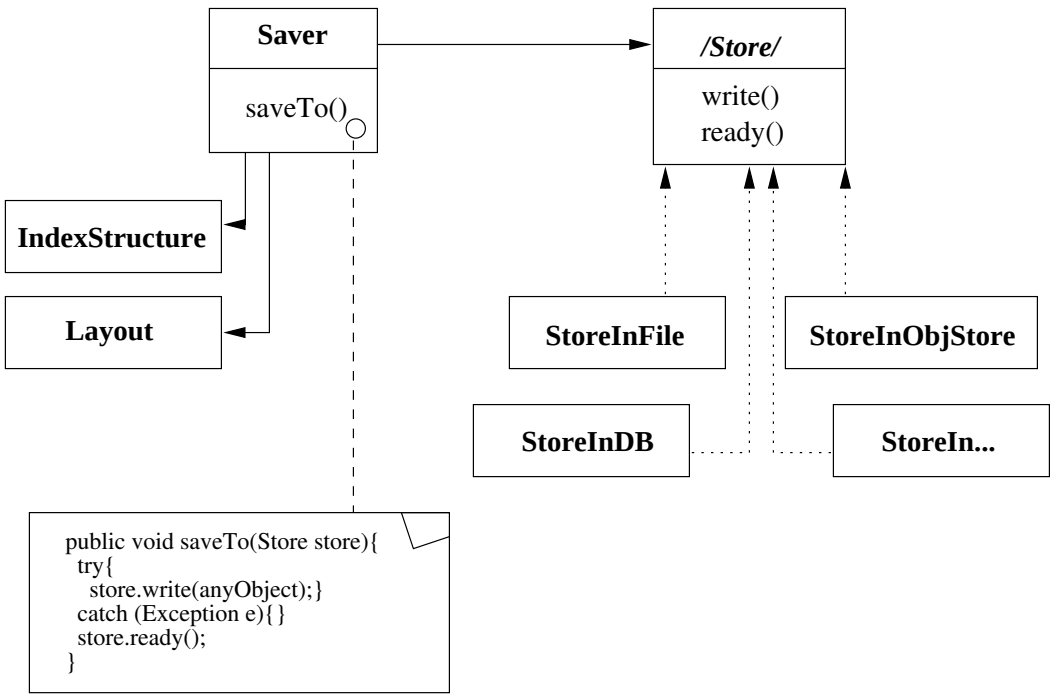

Abbildung 5.17: Die Schnittstelle `Store`

```
Saver save = new Saver(ar);
save.saveTo(file);
...
```

Die Methode `saveTo()` benutzt die Methoden `write()` und `ready()` eines Objekts der Klasse `StoreInFile`, die die Schnittstelle `Store` implementiert. Die Datei, in der das Objekt gespeichert wird, heißt "Ausgabedatei".

Bei der Speicherung der Objekte in eine relationale Datenbank tritt das Problem auf, Objekte in eine geeignete Tabellenrepräsentation überführen zu müssen. Andererseits bieten DBs nicht zu vernachlässigende Vorteile, wie sie in Abschnitt 4.3.4 aufgezählt werden. In Java existiert die Anwendungsprogrammierschnittstelle (API) JDBC, die die Speicherung von Daten in einer relationalen DB unterstützt.

Persistente Objektspeicher erleichtern die Speicherung von Objekten gegenüber relationalen DBs. Die Universität Glasgow entwickelt derzeit in Zusammenarbeit mit Sun Microsystems eine orthogonale persistente Variante von Java: das sogenannte PJava. Die Ziele für das Design von PJava sind [Atkinson et al. 96]:

- keine Sicherheitslücken durch die Verwendung der Persistenz,
- minimale Erweiterungen bzw. Änderungen der Sprache Java,
- keine negative Beeinflußung der Performanz in Abhängigkeit von der Nutzung der Persistenz,
- sowohl einfache Transaktionsmodelle (ACID, verknüpfte Transaktionen) als auch universell nutzbare Transaktionsbausteine, zur Erstellung eigener Modelle, dem Benutzer

zur Verfügung zu stellen.

Da die Entwicklung von PJava zur Zeit der Entstehung dieser Arbeit noch nicht weit genug fortgeschritten ist, wird an dieser Stelle lediglich auf diese Möglichkeit der Speicherung verwiesen.

5.2.5 Zugriffsunterstützung und Nutzung

Diese Dienstgruppe bietet Dienste, um ein Nachschlagewerk mit verschiedenen Angeboten der Nutzung auszustatten. Dieses geschieht nach dem Baukastenprinzip. Hier werden Klassen angeboten, die z.B. eine explizite Pfadverwaltung vornehmen. Diese Speicherung der Pfade wird unter anderem für Nutzungsangebote wie Lesezeichen und Zurückfunktion benötigt. Bei der Erstellung eines Nachschlagewerks darf der Erzeugende aus einem Angebot die gewünschte Funktionalität auswählen. Abbildung 5.18 zeigt einen solchen Baukasten. Einige Zugriffs-

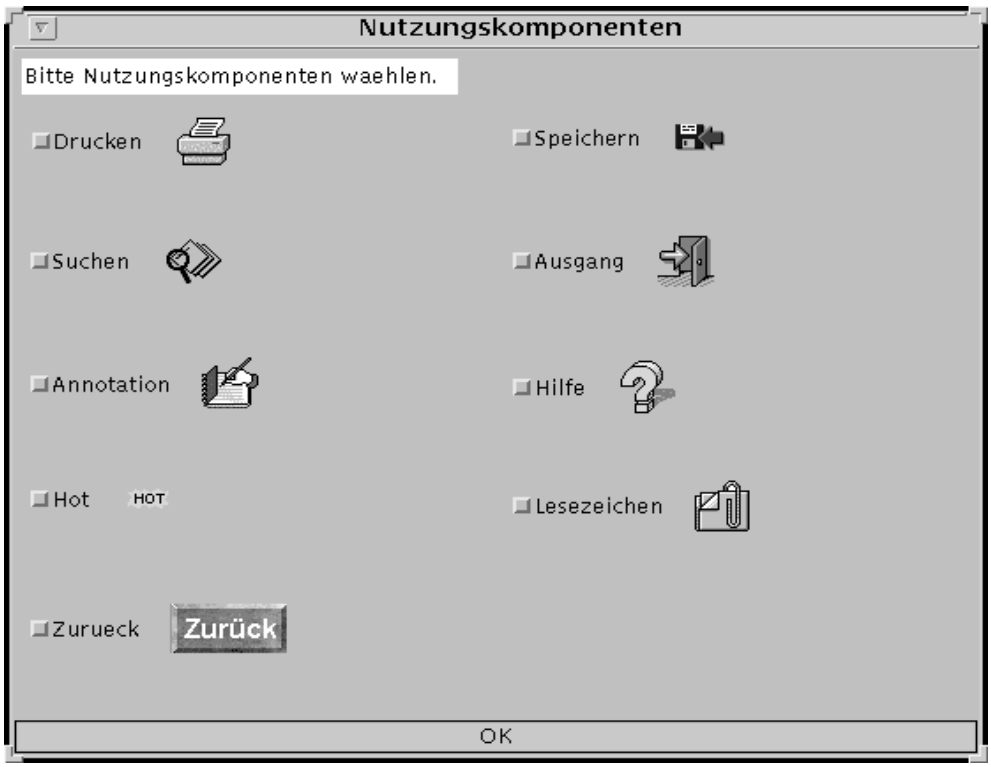

Abbildung 5.18: Baukastenprinzip

unterstützungen sind in den Abschnitten 5.2.3 (Orientierungshilfe) und 5.2.6 (Abrechnung) beschrieben. Das Einbinden eines Abrechnungsdienstes, einer Benutzerverwaltung oder eines Suchdienstes erfolgt nach dem gleichen Prinzip, wie z.B. das Festlegen der unterschiedlichen Layouts der Nachschlagewerkebenen. Es gibt Schnittstellen mit Methoden für diese Zugriffsunterstützungen, die implementiert werden müssen. Bei der Konstruktion eines Nachschlagewerks wird der entsprechende Dienste instantiiert. Die prototypische Realisierung bietet hier ein Lesezeichen und eine Zurückfunktion.

Die administrativen Dienste können z.B. an verschiedenen Stellen in das System integriert werden. So kann eine Authentisierung vor Beginn der Benutzung erfolgen oder erst bei

Zugriff auf ein Dokument. Benutzerverwaltungs- und Abrechnungsdienste können mit in den Authentisierungsdienst integriert sein oder werden explizit nacheinander eingebunden. Abbildung 5.19 illustriert diese Ansatzstellen.

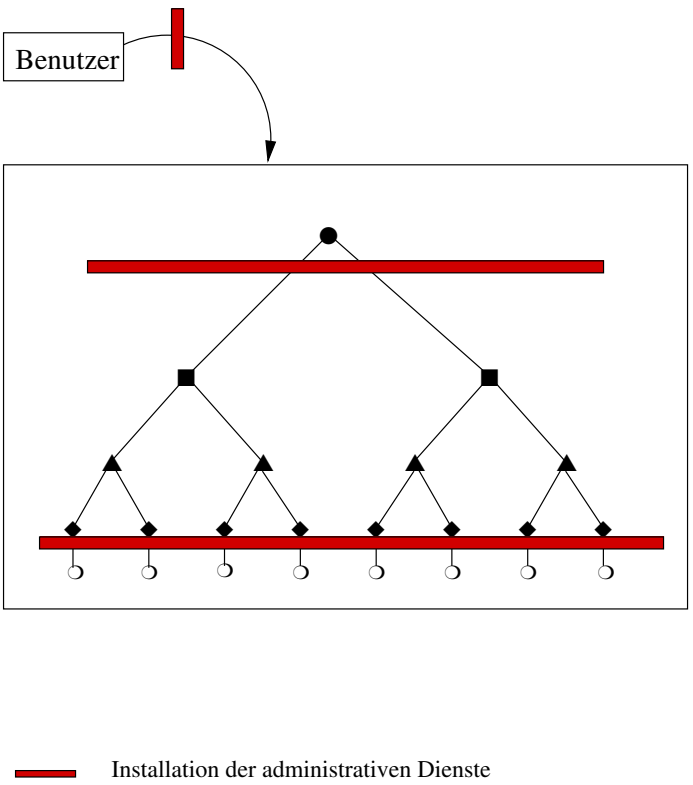

Abbildung 5.19: Orte der Integration der administrativen Dienste in einem Nachschlagewerk

5.2.6 Integration

Es ist nicht das Ziel, ein vorhandenes Nachschlagewerk als isolierte Einheit anzubieten und benutzbar zu machen. Integration bezeichnet hier den Import von Diensten anderer Anbieter (z.B. Abrechnungsdienst) sowie den Export von Nachschlagewerkinhalten in andere Dienste oder Systeme (siehe auch Abschnitt 4.3.6).

Dienste für die Integration externer Dienste

Exemplarisch sei hier der Abrechnungsdienst beschrieben. Das Vorgehen gestaltet sich ähnlich dem Entwurf der vorhergehenden Schnittstellen. Eine Schnittstelle gibt an, welche Methoden die Anbieter von Abrechnungsdiensten bereitstellen müssen. Diese Methoden werden vom System aufgerufen, um die Abrechnung einzuleiten. Da Objektvariablen mit dem Typ der Schnittstelle angegeben werden, ist bei einer konkreten Instantiierung das gewünschte Abrechnungsverfahren, das diesen Typ besitzt, wählbar. Abbildung 5.20 zeigt die Schnittstelle `PayService` mit den Methoden `register()`, `addAmount()`, `makeBill()` und `close()`. Diese Methoden nehmen eine Anmeldung der Benutzer vor sowie die Addition der zu zahlenden

Beträge. Bevor der Abrechnungsmodus geschlossen wird, kann eine Rechnung ausgestellt werden.

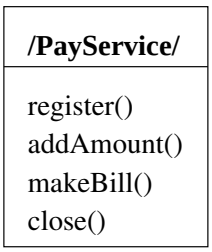

Abbildung 5.20: Die Schnittstelle `PayService`

Die Schnittstelle `PayService` kann von unterschiedlichen Abrechnungsmechanismen benutzt werden. Bei der Konfiguration des Nachschlagewerks ist die gewünschte Abrechnungsform auszuwählen und einzusetzen[4].

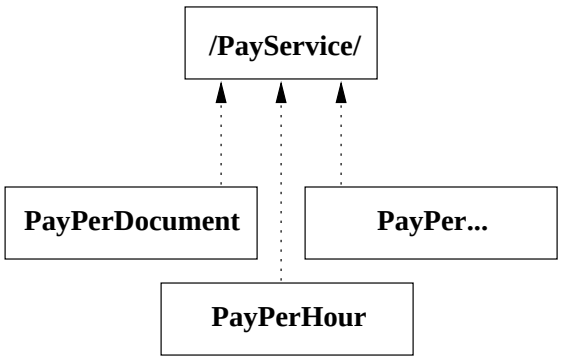

Abbildung 5.21: Mögliche implementierende Klassen der Schnittstelle `PayService`

Dienste für die Integration mit anderen Anwendungen

An dieser Stelle wird eine Möglichkeit angeboten, das Wissen und die Leistungen eines bestehenden Nachschlagewerks in andere Anwendungen zu integrieren. Zur Demonstration wird der Vorschlag aus Abschnitt 4.3.6 aufgegriffen, eine Möglichkeit anzubieten, ein Nachschlagewerk aus einer HTML-Seite heraus zu referenzieren. Eine Anwendung, die dieses Angebot nutzt, kann durch spezifizierende Angaben ein bestimmtes Nachschlagewerk und eine bestimmte Ebene dieses Nachschlagewerks festlegen. Diese Angaben werden einem Programm übermittelt, welches anhand der Angaben das entsprechende Nachschlagewerk auf der angegebenen Ebene startet.

Dieser oben geschilderte Ablauf benötigt ein Verfahren, das in der Lage ist, eine Reihe von Parametern aus einem HTML-Dokument herauszulesen um damit ein Applet zu parametrisieren. Anhand dieser Parameter initialisiert und startet das Applet das entsprechende Nachschlagewerk. Das Herauslesen von Parametern aus einem HTML-Dokument erfolgt über das *Common Gateway Interface* (CGI). Mit Hilfe des CGI kann auf einem Server ein beliebiges

[4] Konkrete Zahlungsformen und deren Sicherheitsaspekte im Internet werden in [SFE 97] besprochen.

Programm aufgerufen und diesem die Parameter übergeben werden. Das Programm berechnet dann aufgrund dieser Parameter eine Ausgabe, die dem Benutzer in seinem Browser präsentiert wird. Folgende Zeile steht in einem HTML-Dokument, um ein Nachschlagewerk zu referenzieren:

```
...
Die
<A HREF="http://www.sts.tu-harburg.de/cgi-bin/uncgi/myscript.cgi?
nachschlagewerk=Bio&ebene1=Rot&ebene2=Wiese">
roten Wiesenblumen
</A> sind sehr sch&ouml;n.<P>
...
```

Der Verweis hinter `roten Wiesenblumen` führt zu einem Nachschlagewerk, welches die Biologie (`nachschlagewerk=Bio`) betrifft und an der Ebene ansetzt, die zu `Rot` und `Wiese` gehört. Der Vorgang wird durch ein CGI-Skript (hier `myscript.cgi`) realisiert. Nach Betätigen des Verweises ruft der Server dieses Skript auf und übermittelt ihm die angegeben Variable=Wert Paare. Das Skript (siehe Abbildung 5.22) erzeugt eine Antwortseite, auf der ein Applet integriert ist, welches als Parameter die Werte der Variable=Wert Paare erhält. Das Applet startet sofort, da die HTML Seite die aktuelle im Browser ist und berechnet anhand der Applet Parameter das betreffende Nachschlagewerk und die Ebene. Weiter ist denkbar, daß fortgeschrittene Anwender über diesen Weg auch das Layout der Ebenen angeben können.

Die Möglichkeit auf einer bestimmten Ebene des Nachschlagewerks beginnen zu können, erfordert einen (lesenden) Zugriff auf die Repräsentation. Mit Hilfe der Angaben aus den Parametern erfolgt eine Navigation in der Repräsentation. Die Repräsentation stellt eine Methode `findNode()` zur Verfügung, die einen gesuchten Indexknoten zurückgibt. Zusätzlich muß das Layout der aktuellen Ebene errechnet werden, damit die Stichworte in dem festgelegtem Layout angezeigt werden. Folgendes Codesegment zeigt die Suche nach einem bestimmten Indexknoten, von dem aus die Anzeige der nächsten Ebene des Nachschlagewerks erfolgt. Die zuvor vom Applet gelesenen Parameter werden in ein Feld namens `levels` eingeordnet und so der Methode `findNode()` übergeben. Die Größe der Füllung des Feldes bestimmt die Anzahl der Aufrufe der Methode `findNode()`. Die Suche nach einem bestimmten Indexknoten erfolgt von der Wurzel der Indexstruktur.

```
...
int i = 0;
while (levels[i] != null){
  Indexnode root = ir.findNode(root, levels[i]);
  ((OrientationArea)help).display(levels[i]);
  i++;
}
...
```

Weiter muß beachtet werden, daß die Orientierungshilfe ebenfalls an den aktuellen Pfad angepasst wird. Anschließend wird das Layout ermittelt und zusammen mit dem gefundenem Indexknoten der Methode zum Visualisieren übergeben. Die weitere Benutzung des Nachschlagewerks unterscheidet sich nicht von der, als wenn am Anfang des Werkes begonnen wird.

```
#!/bin/sh

  echo "Content-type: text/html"
  echo ""
  echo "<HTML><HEAD><TITLE>Nachschlagewerk $WWW_nachschlagewerk</TITLE></HEAD>"
  echo "<BODY TEXT="000000"  LINK="#FF0000" VLINK="#006699" ALINK="#0000FF">"
  echo "<CENTER>"
  echo "<H1>Nachschlagewerk $WWW_nachschlagewerk</H1>"
  echo "<HR>"
  echo "<BR>"
  echo "<P>"
  echo "<APPLET CODEBASE="http://www.sts.tu-harburg.de/people/hachmann"
        CODE="Integration.class" WIDTH=300 HEIGHT=120>"
  echo "<PARAM NAME=which VALUE=$WWW_nachschlagewerk>"
  echo "<PARAM NAME=firstlevel VALUE=$WWW_ebene1>"
  echo "<PARAM NAME=secondlevel VALUE=$WWW_ebene2>"
  echo "<PARAM NAME=thirdlevel VALUE=$WWW_ebene3>"
  echo "<PARAM NAME=fourthlevel VALUE=$WWW_ebene4>"
  echo "<PARAM NAME=fifthlevel VALUE=$WWW_ebene5>"
  echo "<PARAM NAME=sixthlevel VALUE=$WWW_ebene6>"
  echo "<PARAM NAME=seventhlevel VALUE=$WWW_ebene7>"
  echo "<PARAM NAME=eighthlevel VALUE=$WWW_ebene8>"
  echo "</APPLET>"
  echo "</CENTER>"
  echo ""
  echo "</BODY></HTML>"
  echo ""
  exit
```

Abbildung 5.22: CGI-Skript zur Erzeugung eines Applets mit zuvor ausgelesen Parametern

5.3 Verteilung der Anwendung

Da bei einem Nachschlagewerk eine große Quantität an Daten zu erwarten ist, ist eine Verteilung der Anwendung zu empfehlen. Dieser Abschnitt überlegt mögliche Ansätze der Trennung in der Anwendung. Dazu werden in Abbildung 5.23, in Analogie zu Abbildung 3.9, die Komponenten einer Anwendung illustriert. Auf der Ebene der Präsentation werden Ereignisse von außen entgegengenommen sowie deren Ergebnisse nach außen hin dargestellt. Die Applikationsebene enthält den Code zur Ausführung. Dieser wird durch Ereignisse der Präsentationebene angestoßen und abgearbeitet. Auf der Ebene der persistenten Daten sorgen Datenverwaltungsfunktionen für den Zugriff auf die Daten.

Für eine verteilte Anwendung eines Nachschlagewerks ist zu überlegen, welche Komponenten dem Client und welche dem Server zugeordnet werden. In Abschnitt 3.3.3 werden Variationen der Trennungsmöglichkeiten vorgestellt. Bei einer Java Anwendung über das Internet wird der gesamte Bytecode einer Anwendung übertragen und auf dem anfordern-

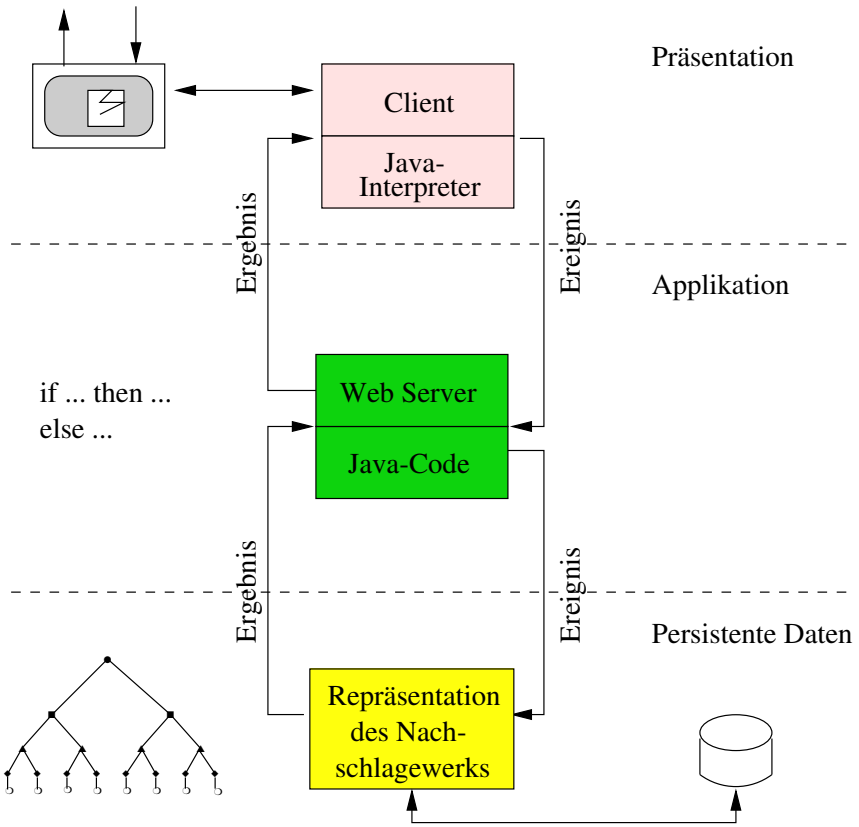

Abbildung 5.23: Drei Ebenen der Anwendung

den Rechner lokal zur Ausführung gebracht. Dies bedeutet, wenn die Daten nicht auf einer Datenbank liegen, sondern z.B. als Objekt in einer Datei, der komplette Inhalt des Nachschlagewerks übertragen wird. Bei jedem Aufruf des Nachschlagewerks dieses im Ganzen einem Benutzer zu übermitteln ist wenig effizient. In Java gibt es die Möglichkeit entfernte Objekte zu definieren (siehe Abschnitt 3.2.1). Folgender Absatz diskutiert mögliche Trennungen.

Untersuchung der Trennungsstellen

Die Varianten Schnitt 0 und Schnitt 5 (siehe Abbildung 3.10) werden in [Rubl 97] als *fat server* bzw. als *fat client* Ansatz bezeichnet. Der Nachteil an "dicken" Clients ist die Abhängigkeit zwischen der Präsentationslogik und der Anwendungslogik sowie der Bedarf an Kapazität auf dem Client. Bei "dicken" Servern ist der Server für die Präsentationslogik und die Anwendungslogik zuständig. Die Zuständigkeit des Client wird auf die Präsentation der graphischen Benutzungsschnittstelle reduziert. Ressourcen des Client werden nicht ausgenutzt.

Heute eingesetzte Client Rechner besitzen in der Regel genügend Ressourcen, um einen Teil einer Anwendung lokal ausführen zu können. Daher sind die Varianten Schnitt 0 und Schnitt 5 (ebenso wie Schnitt 1 und Schnitt 4) für die in dieser Arbeit beschriebenen Anwendung (ein Nachschlagewerk im Netz) nicht optimal geeignet.

Eine sinnvolle Trennung liegt zwischen Schnitt 2 und Schnitt 3[5]. Diese Lösung bedeuted,

[5]Eine klare Trennung ist in der Realität nicht immer möglich. Die Übergange sind durchaus fließend

daß die Funktionen zur Verwaltung der Daten sowie ein Teil der Applikationslogik (der Zugriff auf die Indexstruktur) auf dem Server angesiedelt sind. Die Präsentationsfunktionen werden dem Client zur selbständigen Ausführung übermittelt.

5.4 Beschreibung der prototypischen Realisierung

Um die in dieser Arbeit entworfenen Dienste zu testen und deren Verwendbarkeit nachzuweisen wird eine prototypische Implementierung vorgenommen. Teile der Implementierung werden bereits in den vorigen Abschnitten kurz angesprochen.

Enstehungszyklus

Die prototypische Implementierung bildet den Enstehungszyklus eines digitalen Nachschlagewerks nach. Hierbei wird ein Nachschlagewerk über die Stadien Datenakquisition, Layoutangabe, Nutzungselemente wählen, Anordnung der Grundelemente bis hin zur Speicherung dieses Nachschlagewerks, entwickelt (siehe Abbildung 5.24).

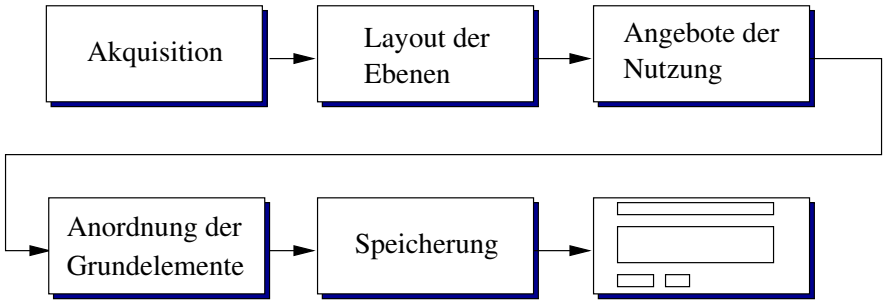

Abbildung 5.24: Entstehungszyklus eines Nachschlagewerks

Für die Datenakquisition wird eine Datei als Datenquelle angenommen und die Klasse `ReadFileInStructure` implementiert. Diese Datei ist wie folgt aufgebaut:

```
$Rot$Feld$vier$http://www.sts.tu-harburg.de/~hachmann/Bilder/rofe4.jpg
$Rot$Feld$fuenf$http://www.sts.tu-harburg.de/~hachmann/Bilder/rofe5.jpg
$Rot$Feld$>fuenf$http://www.sts.tu-harburg.de/~hachmann/Bilder/rofe6.jpg
...
```

Diese Datei und das Trennzeichen ist über die graphische Benutzungsschnittstelle aus Abbildung 5.5 einzugeben. Das Layout für die Ebenen (in diesem Beispiel handelt es sich um ein mehrstufiges Nachschlagewerk) kann aus den graphischen Oberflächenelementen, die Java anbietet, erstellt werden. Hier werden alle angebotenen Oberflächenelemente zur Verfügung gestellt und mit der entsprechenden Ereignisbehandlung versehen. Eine graphische Schnittstelle, wie in Abbildung 5.25, erleichtert die Auswahl. Zur Demonstration von Nutzungsaspekten werden eine Zurück- und eine Lesezeichenfunktion implementiert. Diese sind ebenfalls in einer graphischen Schnittstelle (siehe Abbildung 5.18) anwählbar. Als Orientierungshilfe wird hier eine textuelle Form ausgewählt und installiert. Die Anordnung der Grundelemente kann durch vordefinierte oder selbstdefinierte Vorschläge realisiert werden. Abbildung 5.16 illustriert die graphische Unterstützung für die Anordnung. Schließlich werden alle Angaben

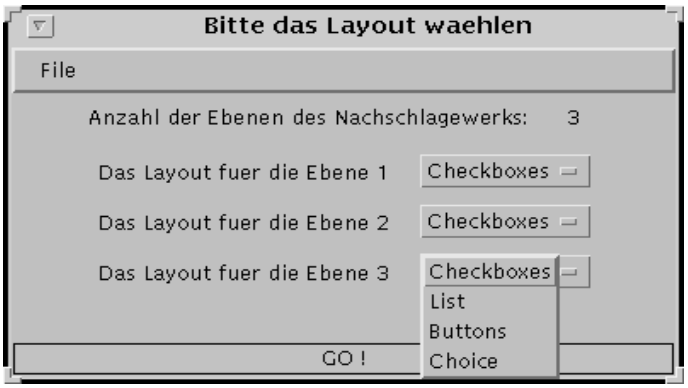

Abbildung 5.25: Graphische Unterstützung bei der Layoutauswahl

gespeichert. Hier wird das Speichermedium Datei verwendet und die Klasse `StoreInFile` implementiert. Das so erzeugte Nachschlagewerk ist bereit, benutzt zu werden.

Integration

Der Ablauf der Integration des Nachschlagewerks in andere Anwendungen wird in Abschnitt 5.2.6 ausführlich behandelt.

Kapitel 6

Zusammenfassung und Ausblick

Inhalt dieses Kapitels ist eine Zusammenfassung und Auswertung der Ergebnisse der vorliegenden Arbeit. Es ist wie folgt aufgebaut: Abschnitt 6.1 faßt die Ergebnisse der Arbeit zusammen. Im Abschnitt 6.2 wird auf die Erfahrungen eingegangen, die bei der Implementierung der Bibliothek generischer Dienste mit der Programmiersprache Java gemacht worden sind. Abschnitt 6.3 diskutiert die kommerzielle Relevanz dieser Arbeit. Mit Abschnitt 6.4, der einen Ausblick auf mögliche Erweiterungen und Ergänzungen dieser Arbeit und somit auch auf Anschlußarbeiten gibt, endet diese Arbeit.

6.1 Zusammenfassung der Ergebnisse

Ausgangspunkt für diese Arbeit ist die Idee prinzipielle Anforderungen zu formulieren, die es ermöglichen Nachschlagewerke in Rechnernetzen zu präsentieren, nutzen und verwalten. Diese Aufgabenstellung führt zunächst zu einer Untersuchung verschiedener Nachschlagewerke, um Strukturen und Eigenschaften herauszufiltern. Hierbei werden Nachschlagewerke auf den Medien Papier, CD-ROM und Internet betrachtet. Ergebnis dieser Untersuchung ist eine Klassifikation nach bestimmten Merkmalen. Aufgrund dieser Analyse und weiteren Betrachtungen von Nachschlagewerken, z.B. dem Lebenszyklus eines Nachschlagewerks, ensteht ein Anforderungskatalog an generischen Diensten. Dieser Anforderungskatalog wird mittels Schnittstellen und Klassen umgesetzt.

Diese Arbeit läßt sich somit grob in zwei Phasen aufteilen:

1. Analysephase
   (a) Untersuchung bestehender Nachschlagewerke
   (b) Aufstellung eines Klassifizierungsschemas
2. Entwurfs- und Implementierungsphase
   (a) Entwurf eines Anforderungskatalogs
   (b) Entwicklung und Implementierung einer prototypischen Dienstbibliothek

Die einzelnen Ergebnisse werden im folgenden aufgeschlüsselt.
Die Analyse bestehender Nachschlagewerke erfolgt nicht nach Kriterien, die üblicherweise von Kritikern herangezogen werden, um ein Nachschlagewerk zu bewerten. Vielmehr liegt

das Interesse bei dem Aufbau und den Strukturen eines Nachschlagewerks (1a). Eine Reihe von Nachschlagewerken, auf unterschiedlichen Medien realisiert, dienen als Grundlage der Untersuchung (siehe Abschnitte 2.3 und 2.4). Aus dieser Untersuchung enstehen die Kriterien nach denen die zu betrachtenden Nachschlagewerke in Abschnitt 2.5 eingeordnet werden (1b). Zu diesen Kriterien gehören z.B. die Art der *medialen Ausrichtung der Zugriffsstruktur* oder die Feststellung einer *ein- oder mehrstufigen Zugriffsstruktur*.

Das Ergebnis der systematischen Betrachtung von Nachschlagewerken, mit Hilfe deren Lebenszyklus, führt zu dem Entwurf eines Anforderungskatalogs (2a). Dieser Katalog (siehe Kapitel 4) ist unterteilt in mehrere Dienstgruppen, angelehnt an den Lebenszyklus. Jede Dienstgruppe faßt am Ende ihrer Beschreibung zusammen, welche Anforderungen an die Dienste zu stellen sind. Die Implementierung einer prototypischen Dienstbibliothek (siehe Kapitel 5) zeigt die Realisierbarkeit eines solchen Anforderungskatalogs (2b). Ausgewählte Schnittstellen finden sich in Anhang B.

6.2 Erfahrungen mit Java

Die Bibliothek generischer Dienste wird in der Sprache Java erstellt. Der Entwurf von Schnittstellen und die Erstellung einer prototypischen Implementation bieten einen Rahmen um die Eignung einer Programmiersprache für diese Zwecke zu untersuchen. Dieser Abschnitt beschreibt die Erfahrungen mit der Sprache Java. Es wird sowohl darauf eingegangen, welche Konzepte der Sprache sich als nützlich erwiesen haben (siehe Absatz 6.2.1), als auch Stellen, an denen konzeptuelle Schwächen oder Defizite der Sprache auftreten (siehe Absatz 6.2.2). Zusammenfassend kann gesagt werden, daß die Sprache Java, bis auf wenige Einschränkungen, zur Bewältigung der vorliegenden Aufgabe ausreichend Unterstützung anbietet.

6.2.1 Positive Erfahrungen

Als erstes ist zu erwähnen, daß Java einfach zu erlernen ist. Als besonders nützlich haben sich Schnittstellen und abstrakte Klassen erwiesen. Mit diesen Konstrukten läßt sich das Konzept gegen eine Schnittstelle zu programmieren (siehe Abschnitt 3.3.1) verfolgen. Das Konzept der Vererbung wird benötigt, wenn abstrakte Klassen verwendet werden. Vererbung ermöglicht weiter ein evolutives Wachsen der Dienste, bzw. eine Art "customizing".

Innere Klassen sind mit der Version 1.1 eingeführt worden. Sie eignen sich für den Einsatz in der Programmierung der Ereignisbehandlung. Im Grunde simulieren sie Funktionen höherer Ordnung.

Bei der Organisation des Quellcodes ist das Konzept der Pakete (*Packages*) nützlich. Pakete dienen dazu, Klassen, die zu einem gemeinsamen Aufgabenbereich gehören, zu bündeln. Die Java Klassenbibliothek ist ebenfalls aus vordefinierten Standardpaketen zusammengesetzt.

Weiter ist das Konzept der Oberflächenprogrammierung hervorzuheben. Java stellt vordefinierte grafische Komonenten zur Verfügung, aus denen Oberflächen erstellt werden können. Diese Komponenten sind im *Abstract Window Toolkit* (AWT) zusammengefaßt. Das AWT bietet dem Benutzer Klassen und Methoden, die von den Systembibliotheken abstrahieren. Aus diesem Grund ist die Benutzung des AWT in seinen Möglichkeiten eingeschränkt, gegenüber einer direkten Programmierung in Motif. Auf der anderen Seite ist das AWT dadurch leichter anzuwenden.

Hilfreich ist die Java "Core API" Dokumentation. Diese ist als Hypertext realisiert. So können während der Arbeit mittels eines Browsers Details zu Paketen, Klassen und Methoden ermittelt und nachgelesen werden. Diese Art der Dokumentation ist nicht auf vordefinierte Klassen beschränkt. So eine Dokumentation kann ebenfalls von den eigenen Klassen erzeugt werden. In dem Quellcode werden durch besondere Kommentarzeichen die Textpassagen markiert, die später in der Dokumentation zu lesen sein sollen. Durch spezielle zusätzliche *Tags* sind Verweise auf den Author, andere Klassen und Methoden möglich. Ein spezieller Compiler generiert aus dem so markierten Quellcode die Dokumentation.

Weiter gibt es inzwischen Klassenbibliotheken von anderen Anbietern, die zum eigenen Gebrauch importiert und installiert werden können (siehe z.B. [Game 97]).

6.2.2 Einschränkungen

Einschränkungen der Sprache Java treten bei dem Versuch auf, generischen Code zu schreiben, der für unterschiedliche Datentypen verwendet werden kann. Hierfür wird das Konzept des parametrischen Polymorphismus benötigt. Ein Ansatz zur Lösung ist die Verwendung der Sprache Pizza. Pizza [OdWa 97] erweitert die Programmiersprache Java um Programmiersprachenkonzepte wie parametrischen Polymorphismus, Funktionen höherer Ordnung und algebraische Datentypen. Pizza Programme werden in Java Bytecode übersetzt. Damit sind bereits vorhandene Java Programme interoperabel zu neuen Pizza Programmen und Pizza Programmierer können die umfangreiche Java Bibliothek nutzen. Nachteilig daran ist die Abhängigkeit von der Aktualität der Klassenbibliothek Pizzas.

Um Zugriff auf Objekte während der Laufzeit zu erhalten, müssen diese stets mit weitergereicht werden. Dies läßt Parameterlisten unübersichtlich groß werden.

Nachteilig an Java ist der relativ junge Entwicklungsstand der Sprache. So sind bis jetzt wenig dynamische Datenstrukturen realisiert. Dies wird sich jedoch mit den kommenden neuen Auflagen verändern.

Die Graphikbibliothek des JDK ist recht weit ausgebaut. Graphische Benutzungsschittstellen sind wichtig für interaktive Anwendungen über das Internet. Eine Oberfläche zu erstellen kann jedoch aufwendiger sein als zunächst vermutet. Hier könnten graphische Entwicklungswerkzeuge Abhilfe schaffen.

6.3 Kommerzielle Relevanz

Die Entwicklung firmen- oder gruppeneigener Informationsnetze (Intranet) bildet einen Anwendungsraum für die in dieser Arbeit formulierten generischen Dienste. Mitarbeiter, die ihr persönliches Nachschlagewerk anderen zur Verfügung stellen möchten, erzeugen mit Hilfe der Dienste ein Nachschlagewerk, daß über das Intranet für alle verfügbar ist. Durch die Konzipierung der Dienste ist weiter ein einheitliches Layout für erzeugte und noch folgende Nachschlagewerke möglich und garantiert so ein *corporate design*.

Weiter ist die Plattformunabhängigkeit Javas von Vorteil. Innerhalb vieler Firmennetze sind unterschiedliche Plattformen (Hardware als auch Software) anzutreffen. Java Bytecode abstrahiert von diesen Plattformen. Erst der Interpreter der betreffenden Maschine produziert graphische Layouts und andere maschinenspezifische Elemente in Abhängigkeit der darunterliegenden Fensterbibliothek in dem für das System typische *look & feel*. Die Entwicklungsumgebung von Java (JDK) ist zudem kostenlos von Sun Microsystems für die gängigen Hard- und Softaware Plattformen zu erhalten.

Nicht zuletzt werden in dieser Arbeit bei der Konstruktion der Dienste Mächtigkeit und Grenzen der Programmiersprache Java untersucht und dokumentiert.

6.4 Ausblick

Die hier vorgestellte Bibliothek generischer Dienste stellt einen ersten Prototyp dar. Erweiterungen und Ergänzungen dieses Ansatzes sind möglich und werden im Folgenden beschrieben:

Permutation: Es ist zu untersuchen, ob die Einführung einer permutierten Indexstruktur (siehe Absatz 4.4) für den Benutzer akzeptabel und brauchbar ist.

Personalisierung: Bei der Personalisierung werden die persönlichen Präferenzen eines Benutzers gespeichert und beim nächsten Aufruf des Programms wieder eingespielt. So wird ein Nachschlagewerk von vornherein mit allen Permutationen, mehreren Layouts und Anordnungen erzeugt, um aus diesem Angebot einen Benutzer seine Vorlieben wählen zu lassen.

Optimierung: Bei der Implementierung der Bibliothek wurde der Aspekt der Effizienz weitgehend vernachlässigt. Optimierungen sind im Bereich der Speicherungsdienste sowie bei der Datenakquisition von Interesse.

Graphische Unterstützung: Graphische Benutzungsschnittstellen tragen zur Erhöhung der Benutzerfreundlichkeit bei. Teilweise werden in dieser Arbeit solche Oberflächen implementiert und als Demonstration dieser Möglichkeit eingesetzt. Hier ist ein konsequenterer Einsatz und weitere Ausarbeitung dieser Oberflächen denkbar.

Externe Dienste: In Absatz 4.3.5 werden Dienste zur Nutzung vorgestellt. Unter anderem auch externe Dienste. Dazu gehören z.B. Abrechnungsdienste und Authentisierung. Bei einer Erweiterung dieser prototypischen Bibliothek ist die Einbindung solcher Dienste zu beachten. [Riga 98] untersucht Mechanismen um Annotationen in digitalen Bibliotheken vorzunehmen. Das Ergebnis dieser Arbeit kann zum Testen der Flexibilität dieser Dienste verwendet werden.

Datenbankanbindung: Für große Datenmengen, wie sie bei Nachschlagewerken überwiegend anzutreffen sind, ist auf jeden Fall eine Datenbankanbindung zu überlegen. Das API JDBC erlaubt Verbindungen zu externen Datenbanken aufzubauen und diese über SQL Befehle abzufragen oder zu aktualisieren. Eine Anwendung die JDBC benutzt, kommuniziert mit dem JDBC Treibermanager, welcher seinerseits mit dem installierten Datenbanktreiber der tatsächlich vorhandenen Datenbank Verbindung aufnimmt (siehe Abbildung 6.1, angelehnt an [CoHo 97]). Dieses Prinzip gestattet es, ein Programm auf allen Datenbanken zu benutzen, die auf SQL basieren. Eine Datenbankanbindung hat ebenfalls Einfluß auf den obigen Punkt Optimierung. Alternativ zu einer Datenbank ist der Einsatz eines persistenter Objektspeichers PJava von [Atkinson et al. 96] denkbar. Diese Lösung umgeht die semantische Lücke zwischen relationalen Datenbanken und objektorientierten Anfragesprachen.

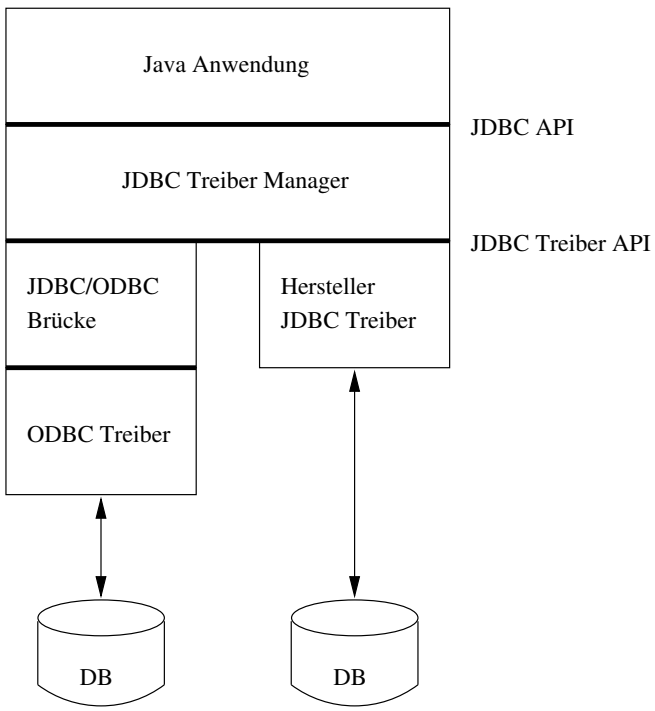

Abbildung 6.1: Kommunikation zwischen JDBC und Datenbank

Anhang A

Verwendete Notation

In dieser Arbeit wird, hauptsächlich in Kapitel 5, eine graphische Notation verwendet, um Beziehungen und Interaktionen zwischen Klassen zu beschreiben. In diesem Anhang wird die Notation im Detail beschrieben.

Die hauptsächlich verwendete Diagrammart ist das

Klassendiagramm: Diese Art Diagramm zeigt Klassen, deren Struktur sowie die statischen Beziehungen zwischen ihnen.

Die Klassendiagramme basieren auf OMT, der Object Modeling Technique [RBP et al. 91], die um weitere Elemente ergänzt wird.

Klassendiagramme

Eine Klasse wird durch ein Rechteck abgebildet. Der Klassenname wird fettgedruckt im oberen Teil des Rechtecks angegeben. Die zentralen Methoden der Klasse sind im mittleren Teil, die möglichen Exemplarvariablen im unteren Teil angeordnet.

Die Angabe des Typs ist optional. Nach Java-Konvention wird der Typname vor den Namen der Methode, vor die Exemplarvariable, den aktuellen Parameter oder den Namen der Konstante gestellt. Der Typname vor der Methode zeigt den Typ des Rückgabewertes an. Kursiv gestellte Namen zeigen an, daß die Klasse oder Methode abstrakt ist. Durch Schrägstriche eingefaßte Namen symbolisieren Schnittstellen. Die Methoden einer Schnittstelle sind immer abstrakt. Abbildung A.1 illustriert obige Angaben.

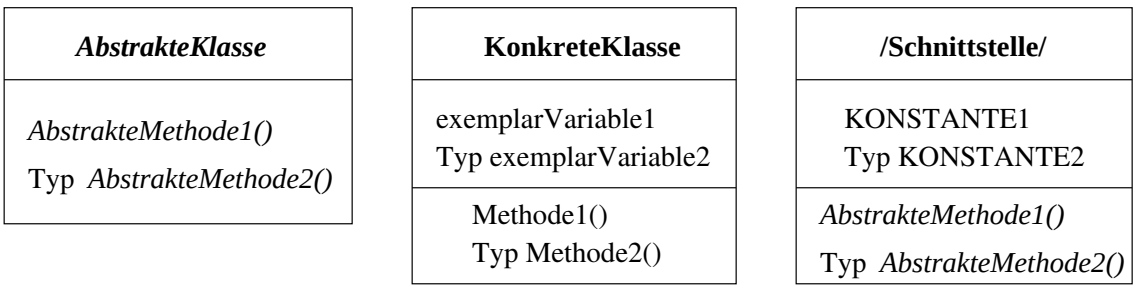

Abbildung A.1: Abstrakte und konkrete Klasse sowie eine Schnittstelle

Klassen stehen in verschiedenen Beziehungen zueinander. Abbildung A.2 zeigt diese verscheidenen Beziehungen. Klassenvererbung wird durch ein Dreieck ausgedrückt, das eine

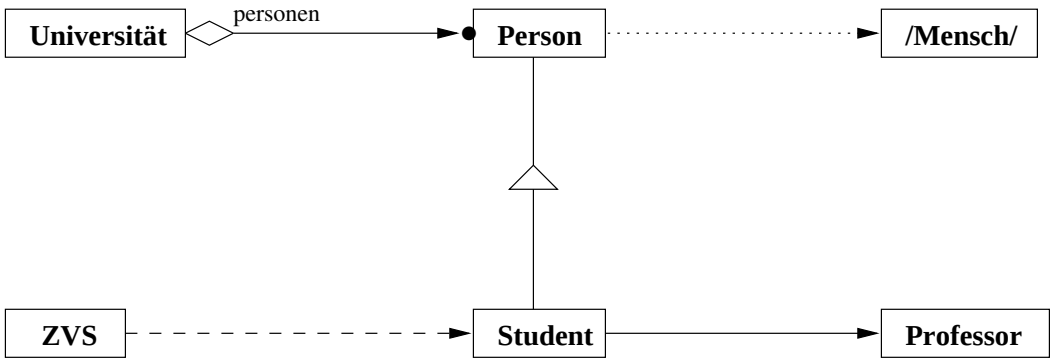

Abbildung A.2: Beziehungen zwischen Klassen

Subklasse (Student in der Abbildung) mit seiner Superklasse verbindet (Person). Eine Linie mit Pfeilkopf repräsentiert eine Bekanntschaftsbeziehung (Student hält eine Referenz auf ein Professorobjekt). Aggregationsbeziehungen werden durch Diamanten am Ausgangspunkt der Objektrefernz dargestellt. Um Referenzen voneinander zu unterscheiden, kann neben dem Ausgangspunkt ein Name angegeben werden. Eine gestrichelte Linie mit Pfeilkopf gibt eine "Erzeugt"-Beziehung an. Der Pfeil zeigt auf die Klasse, von der ein Objekt erzeugt wird. Am Ausgangspunkt der gestrichelten Linie steht die "Erzeuger"-Klasse. Der gefüllte Kreis bedeutet, daß "mehr als ein" Objekt dieser Klasse gemeint ist. Eine Universität aggregiert mehrere Personenobjekte. Eine gepunktete Linie mit Pfeilspitze gibt an, welche Klassen welche Schnittstellen implementieren.

Anhang B

Ausgewählte Schnittstellen

In diesem Anhang werden die Schnittstellen zusammengefaßt aufgeführt, die in Kapitel 5 vorgestellt werden.

ReadInStructure

```
import java.util.*;

public interface ReadInStructure{

  public void prepare() throws Exception;

  public Vector readInNavigation() throws Exception;
  //ASSERT: Vector(NavigationDescriptor)

  public InformationDescriptor readInInformation() throws Exception;

  public void readNext() throws Exception;

  public boolean isEmpty();

  public void close();

}
```

NavigationDescriptor

```
import java.io.*;

public interface NavigationDescriptor extends Serializable{

 boolean equals(Object obj);

 String toString();
```

```
}
```

Datanode

```
import java.io.*;

public interface Datanode implements Serializable{

  public void addInfo(InformationDescriptor id);

  public NavigationDescriptor getNavigator();

  public String toString();

  public int numberChilds();

  public Datanode getChild(int i);

  public boolean isLeaf();

  public void addChild(Datanode n);

  public boolean isChild(Datanode n);

  public boolean equals(Object obj);

}
```

Datastructure

```
import java.io.*;

public interface Datastructure implements Serializable{

  public void setRoot(Datanode nr);

  public Datanode getRoot();

  public int getDepth(Datanode start);

  public int getDepth();

  public Datanode findNode(Datanode start, Datanode search);

  public Datanode findNode(Datanode start, String search);

  public Datanode findParent(Datanode start, Datanode search);
```

```
  public Datanode addNode(Datanode where, Datanode n);
}
```

WorkArea

```
import java.awt.*;

public interface WorkArea{

  public void add(Layout l, IndexStructure is,
                  Container where, Container help);

  public void add(Layout l, IndexStructure is,
                  Container where, Container help,
                  String[] levels);

  public Layout getNWLayout();

  public IndexStructure getIndexStructure();

  public Container getWhere();

  public Container getHelp();

  public String[] getLevels();

}
```

OrientationArea

```
import java.util.*;

public interface OrientationArea{

  public void display(String selected);

  public String getPath();

  public void clear();

  public boolean isEmpty();

}
```

NavigationDescriptorLayout

```
import java.util.*;
```

```
import java.awt.*;
import java.io.*;

public abstract class NavigationDescriptorLayout implements Serializable{
  Vector descrLayouts;
  InformationDescriptorLayout idl;
  Container where;
  Container help,idlTarget;

  public void display(IndexNode node, Vector descrLayouts,
                      InformationDescriptorLayout idl,
                      Container idlTarget,
                      Container where, Container help){
    this.idl = idl;
    this.idlTarget = idlTarget;
    this.descrLayouts = descrLayouts;
    this.where = where;
    this.help = help;

    for (int i=0; i<node.numberChilds();i++){
        navigatorChild(node.getChild(i));
    }
  }


  public void selected(IndexNode myClickedNode, Container idlTarget,
                       Container where, Container help){
    if (myClickedNode.isLeaf()){
      idl.display(myClickedNode, idlTarget);
      ((OrientationArea)help).display(myClickedNode.toString());
    }
    else{
      NavigationDescriptorLayout nodeLayout =
        (NavigationDescriptorLayout) descrLayouts.firstElement();
      Vector nextLayouts = (Vector)descrLayouts.clone();
      nextLayouts.removeElementAt(0);
      nodeLayout.display(myClickedNode, nextLayouts,idl, idlTarget, where, help);
      ((OrientationArea)help).display(myClickedNode.toString());
    }
  }

  public abstract void navigatorChild(Datanode node);

}
```

Store

```
import java.util.*;

public interface Store{

  public void write(Object anything) throws Exception;

  public void ready();

}
```

Literaturverzeichnis

[AiGo 86] Aichele, D., Golte-Bechtle, M. *Was blüht denn da? Wildwachsende Blütenpflanzen Mitteleuropas.* 1986.

[Atkinson et al. 96] Atkinson, M., Daynés, L., Jordan, M., Printezis, T. and Spence, S. *An Orthogonally Persistent Java.* 1996.

[AtMo 95] Atkinson, M. and Morrison, R. *Orthogonally Persistent Object Systems.* VLDB Journal Vol.4 (3), 1995.

[AySt 95] Ayers, E. and Stasko, J. *Using Graphic History in Browsing the World Wide Web.* World Wide Web Journal, Issue One: Conference Proceedings: Forth International World Wide Web Conference, Boston 1995. http://www.w3j.com/1/Contents.html

[Beilstein] Beilstein *Dictionary*
http://www-sul.stanford.edu/depts/swain/beilstein/bedict1.html

[BerlinINFO] BerlinINFO *http://www.kulturbox.de/berlininfo*

[Bert 96] Bertselsmann *Bertelsmann Universal-Lexikon.* Bertelsmann Electronic Edition, Gütersloh, München 1996.

[Böhle 97] Böhle, Knud *Elektronisches Publizieren.* aus: Grundlagen der praktischen Information und Dokumentation. Ein Handbuch zur Einführung in die fachliche Informationsarbeit, Bd. I, S. 379-424, München 1997.

[Bons 96] Bonsiepe, Gui *Interface. Design neu begreifen.* Bollmann Verlag, 1996.

[Bösch 97] Bösch, Andreas *Das WWW als Informationssystem: Konzepte, Dienste, Defizite, Entwicklungen.* Diplomarbeit, Universität Hamburg, 1997.

[CaWe 85] Cardelli, L. und Wegener, P. *On Understanding Types, Data Abstraction and Polymorphism.* ACM Computing Surveys, Jg.17, Dezember 1985, Nr. 4, S. 471-552.

[CFPS 95] Corsten, S., Füssel, S., Pflug, G., Schmidt-Künsemüller, F. A. *Lexikon des gesamten Buchwesens.* Anton Hiersemann, Stuttgart, 1995.

[CoHo 97] Cornell, Gary and Horstmann, Cay *Core Java.* Second Edition, Prentice Hall, 1997. http://csdl.tamu.edu/DL95/

[Cous et al. 95] Cousins, S., Ketchpel, S., Paepcke, A., Garcia-Molina, H., Hassan, S. and Röscheisen, M. *InterPay: Managing Multiple Payment Mechanisms in Digital Libraries.* aus: Digital Libraries '95. The Second Annual Conference on the Theory and Practice of Digital Libraries. http://csdl.tamu.edu/DL95/

[Dadam 96] Dadam, Peter *Verteilte Datenbanken und Client/ Server-Systeme, Grundlagen, Konzepte und Realisierungsformen.* Springer, Berlin, Heidelberg, 1996.

[Duden 83] DUDEN *Deutsches Universal Wörterbuch.* Mannheim, 1983.

[DUDEN] DUDEN *Das große Wörterbuch der deutschen Sprache.* Bd. 4.

[Enca 95] Microsoft *Encarta 95.* 1995.

[FeWo 97] Ferguson, I.A. and Wooldridge, M.J. *Paying Their Way. Commercial Digital Libraries for the 21st Century* D-Lib Magazine, June 1997.

[Flan 96] Flanagan, D. *Java in a Nutshell. A Desktop Quick Reference for Java Programmers.* O'Reilly & Associates, Inc. 1996.

[Game 97] Gamelan *http://www.gamelan.com/*, 1997.

[Gamma et al. 95] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[Gershon et al. 95] Gershon, N., Ruh, W., Le Vasseur, J. Winstead, J. and Kleiboemer, A. *Searching and Discovery of Resources in Digital Libraries.* in ADL 95, pages 191-199, 1995.

[Göd 94] Gödert, W. *Multimedia-Enzyklopädien auf CD-ROM: eine vergleichende Analyse von Allgemeinenzyklopädien.* Deutsches Bibliotheksinstitut, Berlin, 1994.

[HeLe 82] Heinrich, G., Lewe, B. *Materialien zum Aufbau von Informationsbeständen in öffentlichen Bibliotheken.* Fachhochschule für Bibliotheks- und Dokumentationswesen in Köln, Köln, 1982.

[Hend 97] Hendrich, Norman. *Java für Fortgeschrittene.* Springer Berlin, Heidelberg, New York, 1997.

[Hran 97] Hranitzky, Norbert *Spieglein, Spieglein an der Wand..., Verteilte Objekte mit Java.* Java Spektrum, Nr.1, Januar/Februar 1997.

[HRRS 95] Hüser, R., Reichenberger, K. Rostek, L. and Streitz, N. *Knowledge-based Editing and Visualization for Hypermedia Encyclopedias.* Communications of the ACM, April 1995, No. 4, pages 49-51.

[Joha 97] Johann, M. *Neu Aufgelegt. Beta2 des Java Development Kit 1.1.* iX, Multiuser Multitasking Magazin, Nr. 2, S. 52-55, 1997.

[KaMu 94] Karer, A. und Müller, B. *Client/Server Technologie in der Unternehmenspraxis. Vision und Realität der Informationsverarbeitung im restrukturierten Unternehmen.* Springer-Verlag, Berlin Heidelberg, 1994.

[KiVi 95] Killy, W. und Vierhaus, R. Hrsg. *Deutsche Biographische Enzyklopädie.* K.G. Saur Verlag, München, 1995.

[Koch 95] Koch-Steinheimer, P. *HTML Veröffentlichen im Internet.* Thun und Frankfurt am Main, 1995.

[Kuhlen 97] Kuhlen, Rainer. *Hypertext.* aus: Grundlagen der praktischen Information und Dokumentation. Ein Handbuch zur Einführung in die fachliche Informationsarbeit, Bd. I, S. 355-369, München 1997.

[LeMa 95] Levy, D. M. and Marshall, C. C. *Going Digital: A Look at Assumptions Underlying Digital Libraries.* Communications of the ACM, April 1995, No. 4, pages 77-84.

[LePe 96] Lemay, L and Perkins, C.L. *Teach Yourself Java in 21 days.* sams.net, 1996.

[MaMa 95] Marchionini, G. and Maurer, H. *The Roles of Digital Libraries In Teaching and Learning.* Communications of the ACM, April 1995, No. 4, pages 67-75.

[Meyer 97] Meyer, B. *Object-Oriented Software Construction.* Second Edition, Prentice Hall, 1997.

[Meyers Lexikon] *Das Wissen A-Z.* http://www.iicm.edu/ref.m10.

[MHEG 93] ISO/IEC JTC/SC29/WG12 *Information Technology - Coded Representation of Multimedia and Hypermedia Information Objects (MHEG).* ISO Commitee Draft, ISO/IEC CD 13522-1, 1993.

[MoSch 92] Monjau, D. und Schulze, S. *Objektorientierte Programmierung, Ein Einführendes Lehrbuch mit Beispielen in Modula2.* Vieweg Braunschweig, Wiesbaden, 1992.

[MSS 96] Middendorf, S., Singer, R., Strobel, S. *Java Programmierhandbuch und Referenz.* , Heidelberg: dpunkt, 1996.

[MuKe 96] Musciano, C. and Kennedy, B. *HTML The Definitive Guide.* O'Reilly & Associates, Inc., 1996.

[MüNe 96] Müller, Thomas und Neth, Hansjörg *Wissenswust. Multimedia-Enzyklopädien auf CD-ROM.* c't, Magazin für Computertechnik, Nr.9 , 1996.

[Nels 90] Nelson, Andrew N. *Japanese-English Character Dictionary.* Japan, 1990.

[OdWa 97] Odersky, M. and Wadler, P. *Pizza into Java: Translating theory into practice.* Proceedings of the ACM Symposium on Priciples of Programming Languages, jan 1997, or http://www.cis.unisa.edu.au/~cismxo/papers/popl97.ps.gz

[OrHa 97] Orfali, R. and Harkey, D. *Client/Server Programming with Java and CORBA.* Wiley & Sons, 1997.

[Pree 95] Pree, Wolfgang. *Design Patterns for Object-Oriented Software Development.* Addison-Wesley Publishing Company, 1995.

[RBP et al. 91] Rumbaugh, James, Blaha, Michael, Premerlani, William, Eddy, Frederick and Lorensen, William *Object-Oriented Modeling and Design.* Prentice Hall, Englewood Cliffs, NJ, 1991.

[Red 95] Reddy, Raj *The Universal Library: Intelligent Agents and Information on Demand.* Lecture Notes in Computer Science, Digital Libraries, Research and Technology Advances, ADL '95 Forum, USA, pages 27-34, 1995.

[Riga 98] Rigamonti, Amanzio *Annotations in Digital Libraries.* Diplomarbeit, TU Hamburg-Harburg, 1998 (in Arbeit).

[Rubl 97] Ruble, David *Practical Analysis & Design for Client/Server & Gui Systems.* Prentice Hall, 1997.

[ScGe 90] Scheifler, R.W. und Gettys, J. *The X Windows System.* Software-Practice and Experience, Vol.20(S2), October 1990.

[Schu 97] Schult, Dr. T.J. *Darf's eine Scheibe mehr sein? Nachschlagen perfekt: Die CD-ROM erobert den Alltag.* c't, Magazin für Computertechnik, Nr.1, 1997.

[SFE 97] Schuster, R., Färber, J. und Eberl, M. *Digital Cash. Zahlungssysteme im Internet.* Springer-Verlag, 1997.

[SSNM 97] Schmidt, J.W., Schröder, G., Niederée, C. and Matthes, F. *Linguistic and architectural requirements for personalized digital libraries.* International Journal on Digital Libraries, Special issue on Languages for Digital Libraries, Springer Berlin Heidelberg, 1997.

[Stary 96] Stary, Christian *Interaktive Systeme. Software-Entwicklung und Software-Ergonomie* Vieweg, 1996.

[Stein 93] Steinmetz, Ralf *Multimedia-Technologie. Einführung und Grundlagen.* Springer-Verlag, 1993.

[Sun 97] Sun Microsystems *JDK 1.1.1 Documentation, JDK Realease Notes.* http://www.javasoft.com:80/products/jdk/1.1/docs/index.html, 1997.

[Templ 96] Templ, J. *Schwarzes Loch. Wie sicher ist Java wirklich?* iX, Multiuser Multitasking Magazin, Nr. 5, S. 70-73, 1996.

[Tol 96] Tolksdorf, R. *Die Sprache des Web: HTML3. Informationen aufbereiten und präsentieren im Internet.* Heidelberg: dpunkt, Verlag für digitale Technologie, 1996.

[Voss 96] Vossen, Gottfried *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme.* Addison-Wesley, 2. Auflage, 1996.

[Wahr] Wahrig *Deutsches Wörterbuch.*

Erklärung

Hiermit erkläre ich, daß ich die vorliegende Diplomarbeit selbständig durchgeführt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den 10. Dezember 1997

(Ilona Hachmann)