Automatisierte Komposition von wissensvermittelnden Dokumenten für das World Wide Web

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
(Dr. rer. nat.)

genehmigte Dissertation

vorgelegt von

Diplom-Informatiker

Jörg Caumanns

geboren am 13. Mai 1968 in Dinslaken

Gutachter: Prof. Dr. Bernhard Thalheim

Gutachter: Prof. Dr. Erich Neuhold

Gutachter: Prof. Dr. Heinz Schweppe

Tag der mündlichen Prüfung: 30. Mai 2000

Wenn ich einen Urlaub an der Südwestküste der Türkei plane, wäre es durchaus möglich, daß ich keinen Dokumentarfilm über Bodrum finde. Aber beim National Geographic, dem PBS, der BBC und Hunderten anderer Sender erhalte ich Filmausschnitte über traditionellen Schiffsbau, den Fischfang bei Nacht, über antike Unterwasserfunde, das Babaganoush und orientalische Teppiche. Diese Fragmente könnte man zu einem Gesamtbild zusammenfügen, das meinem speziellen Bedarf entspricht. Ich würde zwar keinen Oscar für den besten Dokumentarfilm gewinnen, aber darum geht es auch gar nicht.

Nicolas Negroponte, MIT Media Lab

Diese Dissertation wäre ohne die Hilfe meiner Kollegen, Freunde und Familie nie geschrieben worden. Besonderen Dank schulde ich Herrn Prof. Dr. Hans-Joachim Lenz für viele fruchtbare Gespräche über inhaltliche und formale Aspekte der Arbeit, Herrn Prof. Dr. Bernhard Thalheim für die Unterstützung beim Übergang von einem langen Papier zu einer Dissertation, Dr. Ralf-Detlef Kutsche und Prof. Dr. Heinz Schweppe für diverse Motivationsschübe und meiner Frau und meinen Kindern für die zur Verfügung gestellte Zeit.

Inhalt

Konventionen

Im vorliegenden Text werden eine Reihe von typographischen Konventionen verwendet, die helfen sollen, Definitionen, Algorithmen und Beispiele schnell als solche erkennen zu können:

Gleichungen und formale Definitionen werden links und rechts eingerückt dargestellt, um sie vom normalen Text abzuheben. Definitionen grundlegender Begriffe und Datenstrukturen sind zusätzlich durch Einrahmung hervorgehoben. Alle Gleichungen und Definitionen sind kapitelweise durchnumeriert, so daß im Text eindeutig auf sie Bezug genommen werden kann.

In allen Definitionen und Gleichungen werden Variablen, die einzelne Objekte bezeichnen, klein geschrieben. Die Namen von Mengen und Mengen bezeichnenden Variablen hingegen beginnen immer mit einem Großbuchstaben.

Projektionen werden in lateinischen Buchstaben geschrieben, wobei in den meisten Fällen aus dem ersten Buchstaben des Namens der Datentyps des zugrundeliegenden Objekts erkennbar ist. Abbildungen beginnen immer mit einem griechischen Kleinbuchstaben gefolgt von dem tiefgestellten restlichen Namen der Abbildung in lateinischen Buchstaben. Aus dem (griechischen) Anfangsbuchstaben läßt sich auch hier der Typ des Objekts ablesen, auf dem die Abbildung operiert (z.B. μ für Medienobjekte, σ für Stichworte, etc.).

Die Konventionen für die Bezeichnung von Mengen und Datentypen sind etwas komplexer: Als Beispiel mag ein abstrakter Datentyp A dienen. Die Menge aller denkbaren Objekte vom Typ A wird - von wenigen Ausnahmen abgesehen - mit $A^{\#}$ bezeichnet. Nicht näher bezeichnete Untermengen von $A^{\#}$ werden einfach als A notiert. Fall ein Algorithmus die Erstellung einer bestimmten Untermenge von $A^{\#}$ zum Ziel hat, wird diese Untermenge A^{*} genannt und eventuell zur Berechnung von A^{*} notwendige Zwischenzustände A^{+}. Zwischen dem Namen des Datentyps und dem hochgestellten Stern oder Pluszeichen können weitere hochgestellte Buchstaben stehen, die eine Menge näher bezeichnen. Die Menge M^{bn+} ist so z.B. ein Zwischenzustand ($^{+}$) bei der Berechnung von M^{b*}, der aus einer an einen Block angepaßten (^{b}), normalisierten (^{n}) Menge von Medienobjekten (M) besteht.

Wichtige Begriffe werden bei ihrer ersten Nennung durch Fettdruck hervorgehoben. Kursive Zeichen werden zur Kennzeichnung von fremdsprachigen Fachbegriffen, feststehenden Namen und nicht-übersetzbaren Fremdwörtern verwendet.

In einem auf dem Dissertationsserver der BTU Cottbus abgelegten separaten Anhang sind fast alle der in dieser Arbeit vorgestellten Algorithmen als Pseudocode angegeben. Der Aufbau dieser detaillierten Algorithmenbeschreibungen lehnt sich an [Knuth73] an, während sich ihre Syntax an Konstrukten allgemein bekannter Programmiersprachen wie z.B. *Pascal* orientiert. Jedem Algorithmus ist ein eindeutiger Name zugeordnet, über den er sowohl aus den Text als auch aus anderen Algorithmen referenziert werden kann.

1 Einleitung

Anschließend cyberten wir uns in die britische Hitparade. 40 CD-Singles waren da aufgelistet, und zwar, was keineswegs ein Hammer ist, nach Verkaufszahlen geordnet. Weiter als bis 40 ging's nicht. "In gehobenen Musikzeitschriften geht das aber bis 50 oder 75", meckerte ich und fragte die Freunde, ob wir nicht lieber eine Gastwirtschaft aufsuchen sollten.

Max Goldt[1]

Jedem, der schon einmal einen Tag im *World Wide Web* gesurft ist, dürfe klar sein, daß das Aussterben der Berliner Eckkneipe in keinem Zusammenhang mit der wachsenden Zahl der Internet-Nutzer steht. Auch Redaktionen und Verlage werden in Anbetracht der von Max Goldt beschriebenen *Web-Site* kaum um ihre Existenz bangen müssen. Nichts desto trotz prophezeien Fachleute, daß das *World Wide Web* "Bücher und Zeitungen als bevorzugtes Distributionsmittel für Informationen ablösen" wird[2].

Die meisten Personen, die Informationen im *World Wide Web* anbieten, tun dies nicht zum Selbstzweck. Die Motive reichen dabei von finanziellen Interessen über die Steigerung der eigenen Popularität bis zur Einsparung von Zeit durch den Wegfall von direktem Kontakt. Damit diese Ziele erreicht werden können, ist es jedoch zwingend erforderlich, daß das *Online*-Angebot auch als solches erkannt und wahrgenommen wird. Dies ist um so wichtiger, wenn man sich mit einem Informationsangebot in direkte Konkurrenz zu einem anderen Medium begibt.

Für das Beispiel der *Online*-Hitparade bedeutet dies, daß die angebotenen Informationen dem Nutzer mehr Wert bieten müssen, als eine gedruckte Version der Hitparade in einer Musikzeitschrift. Die Chancen hierfür sind gut, denn das *World Wide Web* bietet mit seiner Aktualität, Adaptivität, Hypermedialität und Interaktivität Leistungsmerkmale, die - bei entsprechendem Einsatz - von keinem anderen Medium in dieser Fülle geboten werden können.

Eine *Online*-Hitparade könnte so z.B.

- durch permanente Aktualisierung aktueller sein als eine gedruckte Hitparade in einer Zeitschrift,
- durch die Möglichkeit der Festlegung von Auswahlkriterien eine Anpassung an Nutzervorgaben beinhalten (z.B. alle mehr als 5000 mal verkauften CDs anzeigen) oder
- durch Vernetzung mit den *Homepages* der Interpreten oder die Einbettung von kurzen Musikclips zusätzliche Informationen bereitstellen.

Die Realisierung einer solchen *Web-Site* erfordert, daß sämtliche CD-Titel samt ihrer Verkaufszahlen und einigen Zusatzinformationen in einer Datenbank abgelegt sind. Mit Hilfe dieser Datenbasis kann für jede

[1] aus: "32/32". In *Ä*, München: Heyne, 1997.
[2] Nathan Myhrvold (Microsoft) zit. nach [Jung99]

Nutzeranfrage ein angepaßtes Dokument erstellt werden, das auf den aktuellsten Verkaufszahlen beruht. Diese Form der Datenbankanbindung und Dokumentenerstellung ist mit existierenden Werkzeugen wie z.B. *ColdFusion* [Allaire98] oder *Active Server Pages* [Denning97] relativ problemlos möglich.

1.1 Virtuelle Dokumente

Dokumente wie die oben beschriebene *Online*-Hitparade werden als **virtuell** bezeichnet [Watters+99b], da ihr konkreter Inhalt nicht statisch vom Autor festgelegt wurde, sondern in Abhängigkeit veränderlicher Einflußgrößen für jede Nutzeranfrage individuell berechnet wird. Im Beispiel der Hitparade sind diese Einflußgrößen die aktuellen Verkaufszahlen und vom Nutzer vorgegebenen Auswahl- und Sortierkriterien.

Virtuelle Dokumente können vor allem dann sinnvoll eingesetzt werden, wenn die darzustellenden Inhalte

- einen großen und vor allem heterogenen Nutzerkreis ansprechen (z.B. *Online*-Kurse und Informationssysteme zu aktuellen oder zur Allgemeinbildung gehörenden Themen),
- von Menschen abgerufen werden, deren verfügbare Hardware und Netzanbindung entweder nicht bekannt oder sehr verschieden ist (z.B. Multimedia PC vs. Palmtop).
- permanent aktualisiert werden müssen (z.B. Aktienkurse) oder neue, für den Nutzer relevante Informationen möglichst unmittelbar publiziert werden sollen (z.B. Wirtschaftsnachrichten).

Existierende Technologien zur Erstellung virtueller Dokumente basieren zumeist auf Schablonen. Das Aussehen und die Struktur des Dokuments ist dabei vom Autor fest vorgegeben, während die darzustellenden Inhalte dynamisch aus einer Datenbank oder einer anderen Informationsquelle gelesen und an dafür vorgesehenen Stellen in das Dokument eingebettet werden. Beispiele für auf dieser Vorgehensweise basierende Systeme sind *Allaire's ColdFusion* [Allaire98] und *WebObjects* von *Apple* [Plotkin+99].

Mit Hilfe von Schablonen darstellbare Informationen umfassen sämtliche kategorisierbaren oder sortierbaren Daten wie z.B. Publikationsverzeichnisse, Börsenkurse, Sportresultate und Terminlisten.

In vielen Fällen reicht jedoch diese einfache Trennung von Form und Inhalt nicht aus, um Informationen im *World Wide Web* mit Hilfe virtueller Dokumente adäquat darzustellen. Dies gilt vor allem dann, wenn nicht nur der Inhalt eines Dokuments, sondern auch seine Struktur und Kodierung dynamisch berechnet werden müssen:

- Beim Design einer web-basierten Anleitung zum Ausfüllen einer Steuererklärung müssen nicht nur die relativ häufigen Änderungen an Tarifen und Bemessungsgrundlagen, sondern auch die äußerst unterschiedlichen Fragestellungen der Benutzer berücksichtigt werden. Für einen selbständigen Handwerker sind andere Aspekte und Teilgebiete des Steuerrechts interessant als für eine Lehrerin oder einen Landwirt. Es bietet sich daher an, anhand des Berufes eines Nutzers den inhaltlichen Schwerpunkt auf die in Frage kommenden Möglichkeiten der Steuerminderung zu setzten.
- Das Aussehen eines "idealen" Handbuchs zu einem Softwaresystem ist sehr stark von den Vorkenntnissen des Nutzers und dessen konkret zu lösender Aufgabe abhängig. Es ist wenig hilfreich, wenn in einem *VisualBasic* Handbuch die *ActiveX* Technologie für alle Nutzer identisch beschrieben wird. Ein Nutzer, der Programmiererfahrung besitzt und ein eigenes *ActiveX* Steuerelement erstellen will, benötigt andere Informationen als ein Programmiemeuling, der lediglich ein vorhandenes Steuerelement in einen Dialog einbinden möchte.
- Auch auf den ersten Blick sehr statische und kompakte Themen erfordern oftmals eine Möglichkeit der Anpassung, wenn sie für einen heterogenen Nutzerkreis aufbereitet werden sollen. Ein Beispiel hierfür ist ein Lehrsystem für Studenten zur deskriptiven Statistik: In verschiedenen Lehrbüchern werden unterschiedliche Begriffe ("empirische Varianz" vs. "Varianz" für die empirische Varianz; "Variable" vs. "Merkmal"), unterschiedliche Definitionen (Vorfaktor (1/n) vs. (1/(n-1)) für empirische Varianz und Kovarianz), unterschiedliche Klassifizierungen (nominal, ordinal, kardinal skalierte Variablen vs. stetige und diskrete Variablen) und Maßzahlen (Korrelationkoeffizient,

Rangkorrelationskoeffizient, Phi-Koeffizient) verwendet. Um für möglichst viele Studenten hilfreich zu sein, muß eine dynamische Anpassung an das vom Nutzer verwendete Lehrbuch erfolgen können.

In allen drei genannten Beispielen kann die dynamische Nutzeranpassung nicht über das Ausfüllen von einfachen Schablonen erfolgen. Vielmehr werden Mechanismen benötigt, die dynamisch aus einer Menge verfügbaren Inhalte die für den aktuellen Nutzer relevanten auswählen und zu einem Dokument mit nicht statisch bestimmbarer Struktur zusammenstellen.

1.2 Dynamische Dokumente

Virtuelle Dokumente können - wie im vorangegangenen Abschnitt beschrieben - vor allem dann sinnvoll eingesetzt werden, wenn jede Nutzeranfrage potentiell anders zusammengestellte und strukturierte Inhalte erfordert. Die entscheidende Einflußgröße bei der Berechnung eines virtuellen Dokuments ist somit der **Nutzer** [Watters+99].

Neben dem Nutzer existieren jedoch noch drei weitere Faktoren, die das Aussehen, den Inhalt und die Struktur eines Dokuments bestimmen:

- der **Autor**, der formale Kriterien (Sprache, Medienkodierung, etc.) sowie die thematische Abgrenzung, den inhaltlichen Schwerpunkt, das fachliche Niveau und das zugrundeliegende didaktische Modell festlegt.
- das **Layout**, d.h. die Anordnung und Darstellung der Inhalte auf dem Bildschirm (bzw. dem Papier).
- das behandelte **Thema**, das die Grundlage der konzeptuellen und semantischen Struktur des Dokuments bildet.

Die dynamische oder statische Erstellung eines Dokuments kann somit über die vier Einflußgrößen "Nutzer", "Layout", "Thema" und "Autor" definiert werden:

Dokument ← (Nutzer, Layout, Autor, Thema)

Ein Dokument wird als **dynamisch** bezeichnet, wenn sich jede Änderung einer der vier Einflußgrößen unmittelbar in Inhalt und/oder Struktur des Dokuments widerspiegelt.

Die im vorangegangenen Abschnitt beschriebenen Dokumente sind somit Beispiele für dynamische Dokumente, da potentiell jede Änderung der Einflußgröße "Nutzer" in einem anders zusammengestellten Dokument resultiert. Dynamische Dokumente wie diese, von denen jede Instanz nur für die Dauer eines Aufrufs gültig ist, werden als **virtuell** bezeichnet.

Existierende kommerzielle oder im Rahmen von Forschungsprojekten entwickelte Werkzeuge unterstützen den Autor zumeist lediglich bei der Berücksichtigung einiger weniger veränderlicher Einflußgrößen:

- Die Trennung von Form und Inhalt, wie sie von vielen HTML Editoren (z.B. *FrontPage*) geboten wird, erlaubt die problemlose Veränderung des Layouts eines Dokuments.
- Eine zusätzliche Datenbankanbindung (z.B. *ColdFusion*, *WebObjects* und *FrontPage*) ermöglicht darüber hinaus eine rudimentäre Unterstützung für Nutzeranpassungen und Erweiterungen von stark strukturierten Teilaspekten eines Themas (z.B. Literaturlisten oder Produktkataloge).
- Die von zahlreichen Forschungsprojekten verfolgte dynamische Vernetzung existierender Seiten [Brusilovsky98] anhand einer vorgegebenen konzeptuellen Struktur bietet eine relativ komfortable Nutzeranpassung, ist jedoch sehr inflexibel, wenn der inhaltliche Schwerpunkt oder auch nur das verwendete Layout verändert werden soll.

Die zwei Einflußgrößen, die von kaum einem existierenden Werkzeug zur Dokumentenerstellung oder -generierung adäquat berücksichtigt werden, sind der Autor und das behandelte Thema. Die Veränderlichkeit dieser beiden "Variablen" sollte jedoch nicht unterschätzt werden, insbesondere wenn das behandelte Thema sehr dynamisch ist, d.h. sich die aktuell wichtigen und relevanten Informationen

sehr schnell ändern und die Halbwertszeit von "Neuigkeiten" sehr kurz ist. Diese Eigenschaften besitzen z.B. fast alle politischen und ökonomischen Themen:

- Das deutsche Steuersystem ändert sich mittlerweile nicht nur nach jedem Regierungswechsel, sondern auch nach fast jeder von der jeweiligen Bundesregierung verlorenen Landtagswahl. Darüber hinaus sind mehr oder minder ernst zu nehmende Änderungsvorschläge ein von Politikern aller Parteien vor allem vor Wahlen gern genutztes Mittel, um in den Hauptnachrichten des Fernsehens an exponierter Stelle zu erscheinen.
- Ein ähnlich dynamisches Thema ist die Europäische Wirtschafts- und Währungsunion. Eine Aufbereitung dieses Themas im *World Wide Web* wird zusätzlich noch dadurch erschwert, daß sich der Schwerpunkt der öffentlichen Diskussion von Zeit zu Zeit radikal verändert (Kommt der Euro? Wie stabil ist er? Verschiebung der Euro-Einführung? 5 = 5.0? Ist Italien dabei? Beschäftigungspakt für Europa? Agenda2000, etc.).
- Die politische Lage auf dem Balkan ist so komplex und undurchsichtig, daß eine informative und vor allem aktuelle Aufbereitung dieser Thematik mit statischen Dokumenten nur unbefriedigend möglich ist: Mitspieler und Motive der verschiedenen Konflikte wechseln kontinuierlich und aus "den Guten" werden schnell "die Bösen" und umgekehrt.

Dynamische Themen und sich verlagernde Schwerpunkte führen zwangsläufig zu sich häufig ändernden Dokumentstrukturen und sind daher mit allein auf mehr oder minder statischen Strukturen aufbauenden Mechanismen (z.B. Schablonen) nicht adäquat im *World Wide Web* abbildbar.

1.3 Das i4 Framework

Wie diese Beispiele zeigen, liegt der entscheidende Vorteil dynamischer Dokumente nicht nur in der individuellen Nutzeranpassung, sondern vor allem auch in der vereinfachten Wartbarkeit. Politische und ökonomische Themen können tagesaktuell nur mit automatisch erstellten, dynamischen Dokumenten im *World Wide Web* dargestellt werden, da hier die nutzerunabhängigen Einflußfaktoren "Autor" und "Thema" sehr häufigen Änderungen unterliegen.

Eine Möglichkeit, um die Einflußgrößen "Autor" und "Thema" bei der Dokumentenerstellung auch als variabel zu behandeln, ist, das zu erzeugende Dokument anhand einer abstrakten Beschreibung dynamisch zu generieren. Die Idee hierbei ist, daß der Autor nicht ein Dokument erstellt, sondern lediglich über Metadaten und ähnliche Mechanismen die darzustellenden Inhalte festlegt. Anhand dieser Informationen wird anschließend von einem "intelligenten" Softwaresystem das gewünschte Dokument - unter zusätzlicher Berücksichtigung der Faktoren "Layout" und "Nutzer" - generiert.

Ein solches "intelligentes" Softwaresystem ist das in dieser Arbeit beschriebene **i4 Framework**[3]. Das *i4 Framework* erlaubt die Generierung dynamischer Dokumente unter Berücksichtigung der Veränderlichkeit aller vier genannten Einflußgrößen. Grundidee des *i4 Frameworks* ist es, anhand einer vom Autor festgelegten und vom aktuellen Nutzer beeinflußbaren abstrakten Dokumentenbeschreibung aus einer Menge vordefinierter Fragmente dynamisch die geeignetsten auszuwählen und zu einem Dokument zusammenzufügen.

Motivation für die Entwicklung des *i4 Frameworks* war die bei der Durchführung mehrerer Multimedia-Projekte gewonnene Erkenntnis, daß der Aufwand für die permanente Wartung und Aktualisierung einer *Web-Site* erheblich größer ist als der Aufwand für ihre erstmalige Erstellung. Der Grund hierfür ist die Vernachlässigung der Einflußgrößen "Autor" und "Thema" durch alle momentan existierenden Werkzeuge und Autorensysteme zur Erstellung sowohl statischer als auch dynamischer Dokumente.

[3] i4 = Integration von Informationen zu individualisierten Inhalten

1.4 Übersicht

In den folgenden Kapiteln werden die Datenstrukturen und Algorithmen beschrieben, die dem *i4 Framework* zugrunde liegen. Dabei wird der gewählte Lösungsansatz von der abstrakten Motivation der zugrundeliegenden Idee über die formale Beschreibung der verwendeten Algorithmen bis zu Details der Implementierung Schritt für Schritt konkretisiert.

In Kapitel 2 "Hypermedialer Lehrsysteme" werden Bausteine hypermedialer Lehr- und Informationssysteme auf verschiedenen Abstraktionsebenen beschrieben. Die Zusammensetzung und Sequenzierung der einzelnen Bausteine wird anhand inhaltlicher, struktureller und didaktischer Abhängigkeiten und Vorgaben theoretisch motiviert und anschließend anhand eines Beispiels auch praktisch dargestellt.

In Kapitel 3 "Erstellung von Lehrsystemen" wird zunächst ein typischer, auf dem Wasserfall-Modell basierender, manueller Entwicklungszyklus für hypermediale Lehrsysteme vorgestellt und bewertet. Zur Umgehung einiger der gravierendsten Nachteile dieses Ansatzes bei der Bearbeitung dynamischer Inhalte für heterogene Nutzergruppen wird der Einsatz automatischer Verfahren zur Zusammenstellung und Strukturierung von Lehr- und Informationssystemen empfohlen. Hierbei können zwei grundsätzliche Strategien – *Top-Down* und *Bottom-Up* – unterschieden werden. Während fast alle verwandten Arbeiten einen - an den manuellen Entwicklungszyklus angelehnten - Top-Down Ansatz verfolgen, wurde im Rahmen dieser Arbeit ein Bottom-Up Entwicklungszyklus entwickelt, der insbesondere bezüglich der Erstellungskosten und den Möglichkeiten individueller Anpaßbarkeit eine Reihe von Vorteilen gegenüber anderen Systemen besitzt. In den Kapiteln 3.2 und 3.3 werden die Grundideen der zugrundeliegenden Konzeption und der implementierten Realisierung skizziert und im abschließenden Kapitel 3.4 mit anderen automatisierten Entwicklungszyklen verglichen.

Kapitel 4 "Medienobjekte" beschreibt die der dynamischen Dokumentenerzeugung zugrundeliegenden Basisobjekte: Medienobjekte und Stichworte. Hierbei wird sowohl auf die Beschreibung dieser Objekte durch Metadaten als auch auf ihre Verknüpfung durch Indizes eingegangen. Darüber hinaus wird in diesem Kapitel der im weiteren Verlauf der Arbeit verwendete Formalismus zur Beschreibung von Algorithmen und Datenstrukturen eingeführt.

Um eine Nutzeranpassung realisieren zu können, müssen Informationen über den Nutzer vorliegen. Von Interesse hierbei sind neben Lernzielen und Vorwissen auch die zur Verfügung stehende Hardware sowie diverse Präferenzen und Einschränkungen. Aber nicht nur der Nutzer ist für ein System zur dynamischen Erstellung von Dokumenten eine variable Größe, sondern auch der Autor und das behandelte Thema. Jedes Thema hat seine Besonderheiten, die sich ebenfalls in der Struktur und Zusammenstellung eines Lehr- oder Informationssystems niederschlagen. Aus diesem Grund unterstützt das *i4 Framework* das Modell der „doppelten Adaptierbarkeit“; der Anpassung an Nutzer und Thema. Ausführliche Informationen, sowohl zur doppelten Adaptierbarkeit als auch zu den modellierbaren Eigenschaften von Nutzern, bilden den Inhalt von Kapitel 5 „Nutzungsszenarien“.

Medienobjekte und Nutzungsszenarien führen in Kapitel 6 "Der Abhängigkeitsgraph" zur Beschreibung der zentralen Datenstruktur: des sog. Abhängigkeitsgraphen. Durch den Abhängigkeitsgraphen werden alle Abhängigkeiten zwischen Medienobjekten und Stichworten in Form eines gerichteten, bipartiten Graphen beschrieben. Aufbauend auf der Definition des Abhängigkeitsgraphen werden diverse Maßzahlen definiert.

Ausgehend von diesen Maßzahlen erfolgt die Auswahl und Strukturierung von Medienobjekten. Die dazu verwendeten Algorithmen bilden den Inhalt von Kapitel 7 "Auswahl und Strukturierung". Im Mittelpunkt steht dabei vor allem der sog. *Minesweeper*-Algorithmus, mit dem eine heuristische Bestensuche zur Auswahl der für ein Szenario geeignetsten Medienobjekte durchgeführt wird.

In den Kapiteln 1 bis 7 wird die Möglichkeit der Anpassung eines Dokuments an die Erfordernisse eines Nutzers und des behandelten Themas auf einer eher abstrakten Ebene behandelt. In Kapitel 8 "Implementierung" wird die konkrete Implementierung eines auf den vorgestellten Konzepten und Algorithmen basierenden Systems, des Medienobjekt-Managers, beschrieben. Im Mittelpunkt stehen dabei die Benutzerschnittstelle, die themenspezifische Konfiguration der Auswahl- und Strukturierungsalgorithmen und die Möglichkeiten der Autorenunterstützung. Eine weitere Konkretisierung bietet Kapitel

9 „Fallbeispiel“ in dem ausgehend von einem existierenden Lehrbuch der komplette Autorenzyklus von der Fragmentierung bis zur Rekomposition anhand eines Beispiels durchgespielt wird.

Das abschließende Kapitel 10 "Zusammenfassung und Ausblick" gibt nach einer kurzen Zusammenfassung einen Ausblick auf zukünftige Erweiterungen und Einsatzgebiete des vorgestellten *Bottom-Up* Algorithmus.

2 Hypermediale Lehrsysteme

Einer der Bereiche, die immer als erstes genannt werden, wenn es um die schöne, neue, vernetzte Welt geht, ist "Lernen".

Auf die Frage, wann, wie und wo Menschen neues Wissen erwerben, glauben viele Forschungsprojekte, neue Antworten gefunden zu haben; die meisten davon beinhalten das Stichwort "World Wide Web". Neue Konzepte wie *lebenslanges Lernen* [Cresson96] und *virtuelle Universitäten* [Dwyer+95, Chellappa+97] geben eine Eindruck von dem, was uns alle in naher Zukunft erwartet. Die Grundidee ist immer dieselbe: Hypermedia; die Vernetzung von statischen und kontinuierlichen Medien zu inhaltsreichen, interaktiven Lernwelten [Issing+97].

Adaptierbarkeit und Wartbarkeit sowie die damit verbundenen Möglichkeiten der Kostenreduktion spielen für die Erstellung von Lehr- und Informationssystemen eine große Rolle. Darüber hinaus sind bei dieser Art von wissensvermittelnden Dokumenten mindestens drei der im ersten Kapitel genannten Einflußgrößen - Thema, Autor und Nutzer - veränderlich, so daß Lehr- und Informationssysteme ein ideales Anwendungsgebiet für das *i4 Framework* darstellen. Im weiteren Verlauf dieser Arbeit steht daher die dynamische Erstellung wissensvermittelnder Dokumente im Mittelpunkt der Betrachtung.

Dieses Kapitel gibt einen Überblick über Probleme und Lösungsansätze der dynamischen Erstellung von wissensvermittelnden Dokumenten. Das Hauptaugenmerk liegt dabei zunächst auf der Motivation und Herleitung des im Rahmen dieser Arbeit entwickelten *Bottom-Up* Ansatzes anhand von strukturellen Merkmalen hypermedialer Dokumente.

2.1 Adaptierbare hypermediale Lehr- und Informationssysteme

Die mit Hilfe des *i4 Frameworks* generierbaren, web-basierten wissens- und informationsvermittelnden Dokumente[4] werden im Folgenden als adaptierbare hypermediale Lehr- bzw. Informationssysteme bezeichnet:

> Unter dem Begriff "**hypermediale Lehr- und Informationssysteme**" sind strukturierte Dokumente zusammengefaßt, die das Ziel verfolgen, mit Hilfe von untereinander verbundenen, verschieden kodierten Medienelementen den Wissensstand einer definierbaren Zielgruppe bezüglich eines eingrenzbaren Themengebietes zu erweitern. (Definition basierend auf [Baumgartner97]). Ein hypermediales Lehr- bzw. Informationssystem ist **adaptierbar**, wenn Inhalt, Form und/oder Struktur des Dokuments dynamisch auf Basis veränderlicher Einflußgrößen (Nutzer, Autor, Layout und Thema) berechnet werden.

[4] Der Begriff des Dokuments wird in dieser Arbeit für eine Einheit von inhaltlich und strukturell zusammengehörigen, potentiell vernetzten Informationseinheiten verwendet. Für eine ausführliche Diskussion der Charakteristika von (elektronischen) Dokumenten sowie den Problemen einer allgemeinen Definition siehe [Schamber96].

Die Unterscheidung zwischen Lehrsystemen einerseits und Informationssystemen andererseits beruht auf dem Vorhandensein bzw. Fehlen eines didaktischen Ansatzes zur Wissensvermittlung.

Die wichtigsten Aspekte dieser Definition sind, daß ein hypermediales Lehr- oder Informationssystem immer

- eine von Autor fest umrissene Thematik behandelt,
- an eine bestimmte Zielgruppe mit definierbaren Lernzielen adressiert ist,
- aus einem oder mehreren vernetzten Einzeldokumenten (Bildschirmseiten, Dateien) besteht und
- sowohl eine Binnenstruktur als auch eine (didaktisch motivierte) Lernstruktur besitzt.

Lehr- und Informationssysteme im Sinne dieser Definition sind somit vor allem Lehranwendungen zur schulischen und universitären Aus- und Weiterbildung, sowie Dokumente über einzelne Personen, Objekte, Orte und Ereignisse. Weitere Beispiele für hypermediale Informationssysteme sind technische Dokumentationen, Handbücher, Zeitungsartikel, (insbesondere fachspezifische) Lexika und wissenschaftliche Publikationen.

Keine Lehr- oder Informationssysteme sind hingegen private *Homepages* (keine fest umrissene Thematik und Zielgruppe), *Online*-Kataloge (keine Lernstruktur) und Zeitungen (keine Struktur außerhalb eines einzelnen Artikels).

In der oben angegebenen Definition von hypermedialen Lehr- und Informationssystemen wurde das Attribut "hypermedial" als die Verwendung verschieden kodierter, vernetzter Medien umschrieben. Diese Festlegung orientiert sich an dem intuitiven Verständnis von "Hypermedia" als "Multimedia mit Hyperlinks". Insbesondere der Begriff "*Media*" wird nicht allzu wörtlich genommen: Ein einzelnes Dokument kann streng genommen nie multi- oder hypermedial sein, da als einziges Medium der Computer verwendet wird[5].

Glücklicherweise kann sich jedoch mittlerweile jeder unter "Multimedia" und/oder "Hypermedia" etwas vorstellen, das normalerweise dem entspricht, was der Benutzer dieses Wortes damit im Sinn hat. Aus diesem Grund soll den ungezählten Versuchen einer korrekten und präzisen Definition der Begriffe "Hypermedia" und "Multimedia" - und auch den Versuchen einer Abgrenzung beider Begriffe - an dieser Stelle auch kein weiterer hinzugefügt werden[6].

2.2 Bausteine Hypermedialer Dokumente

Hypermediale Dokumente - z.B. für Lehr- und Informationszwecke - sind nicht atomar; sie setzen sich vielmehr aus einzelnen (Bildschirm-)Seiten zusammen, die wiederum aus verschiedenen Medien wie z.B. Texten und Bildern bestehen.

In diesem Abschnitt werden die einzelnen Bausteine hypermedialer Dokumente und ihre Verknüpfungen sowohl unter strukturellen als auch unter semantischen Aspekten betrachtet.

2.2.1 Kapitel-, Seiten- und Medienobjektstruktur

Die meisten hypermedialen Lehr- und Informationssysteme basieren auf einer buch-ähnlichen, hierarchischen Struktur. Diese logische Dokumentstruktur setzt sich im allgemeinen aus mehreren Kapiteln zusammen, die ihrerseits aus Unterkapiteln bestehen, welche wiederum in mehrere Unter-Unterkapitel untergliedert sein können. Aus diesem Grund wird die logische Struktur eines Dokuments im Folgenden als **Kapitelstruktur des Dokuments** bezeichnet.

---

[5] Es wäre hier daher korrekter, statt von "Multimedia" von "Multicodierung" zu sprechen. Wenn bei der Multicodierung auf das Ansprechen verschiedener Sinne des Benutzers abgezielt wird, ist allerdings auch der Begriff "Multimodal" sehr zutreffend [Weidenmann97].

[6] Eine ansehnliche Sammlung von Definitionsversuchen findet sich in [Schulmeister96]

Neben den - zumeist sequentiellen - Abhängigkeiten zwischen den einzelnen Kapiteln und Unterkapiteln existieren oftmals auch inhaltliche Querverbindungen zwischen einzelnen Kapiteln. Diese Querverbindungen entstehen durch Rückgriffe, Vorgriffe, Beispiele, Vergleiche und ähnliche didaktische Konstrukte (siehe auch Kapitel 2.2.2).

Die Kapitelstruktur eines Informationssystems orientiert sich zumeist an der logischen Struktur des behandelten Themengebiets. Sie wird durch Menüs, Inhaltsverzeichnisse, *Site Maps* und andere Navigationselemente für den Benutzer explizit sichtbar gemacht.

Abbildung 2-1: Kapitelstruktur eines (fiktiven) Informationssystems zum Thema "Steuern"

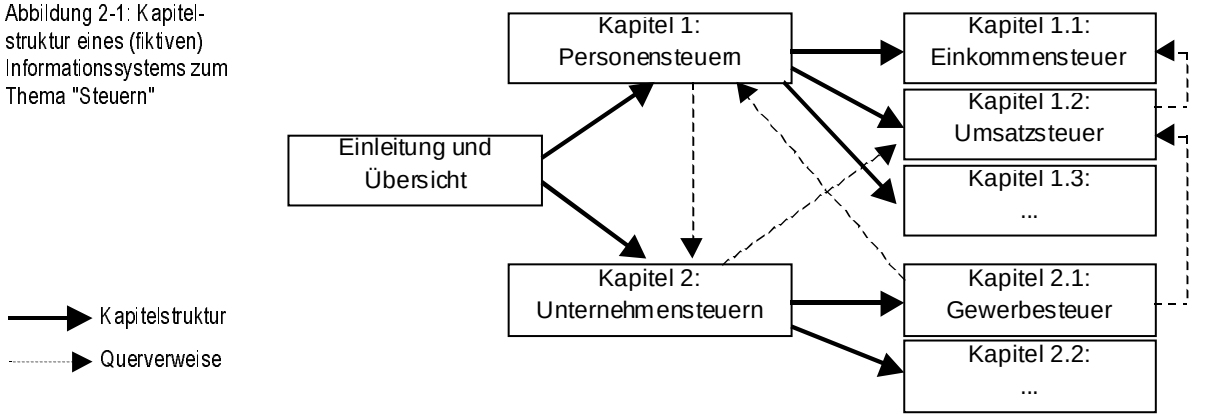


Abbildung 2.1 zeigt die Kapitelstruktur eines (fiktiven) Informationssystems zum Thema "Steuern"[7]. Auf oberster Ebene existiert ein Kapitel zur Einleitung, Motivation und Übersicht. Der inhaltliche Teil gliedert sich in zwei Kapitel ("Personensteuern" und "Unternehmensteuern"), die wiederum in verschiedene Unterkapitel aufgeteilt sind.

Jedes Kapitel und Unterkapitel besteht aus einer oder mehreren (Bildschirm)Seiten. Die einzelnen Seiten eines Kapitels sind zumeist sequentiell angeordnet, wobei jedoch auch andere, vernetzte Strukturen denkbar sind [Caumanns+99c]. Im Fall eines web-basierten Lehr- oder Informationssystems ist eine Seite identisch mit einer HTML-Datei.

Die Seitenstruktur eines Kapitels ist für den Benutzer in Form von "nächste Seite", "vor" und "zurück" *Buttons* bzw. *Hyperlinks* sichtbar. Die zusammengefaßten Seitenstrukturen der einzelnen Kapitel ergeben die **Seitenstruktur des Dokuments**. Läßt man die Kapitelstruktur außer Acht, kann man ein Lehr- oder Informationssystem anstelle einer Hierarchie aus Kapiteln auch als Hierarchie aus einzelnen Seiten ansehen.

Abbildung 2-2 stellt eine mögliche Seitenstruktur zu der in Abbildung 2-1 angegebenen Kapitelstruktur dar. Das einleitende Kapitel besteht aus einem Titelbild und einer Kapitelübersicht, von der aus der Benutzer zu den einzelnen Kapiteln (in diesem Fall "Personensteuern" und "Unternehmenssteuern") verzweigen kann. Jedes dieser beiden Kapitel besitzt eine Übersichtsseite, über die der Einstieg in die jeweiligen Unterkapitel ermöglicht wird. Die einzelnen Unterkapitel bestehen in diesem Beispiel aus zwei oder drei zumeist sequentiell aufeinander aufbauenden Seiten. Die dick gezeichneten Pfeile geben die Kanten der Seitenstruktur an. Sie definieren eine Reihe möglicher Betrachtungsfolgen der Seiten, die vom Autor anhand didaktischer Überlegungen oder aufgrund inhaltlicher Abhängigkeiten festgelegt wurden.

[7] Das in diesem Kapitel verwendete Beispieldokument orientiert sich an Inhalt und Struktur der CD-ROM "Investitionsrechnung unter Steuern" [Buchmann+98].

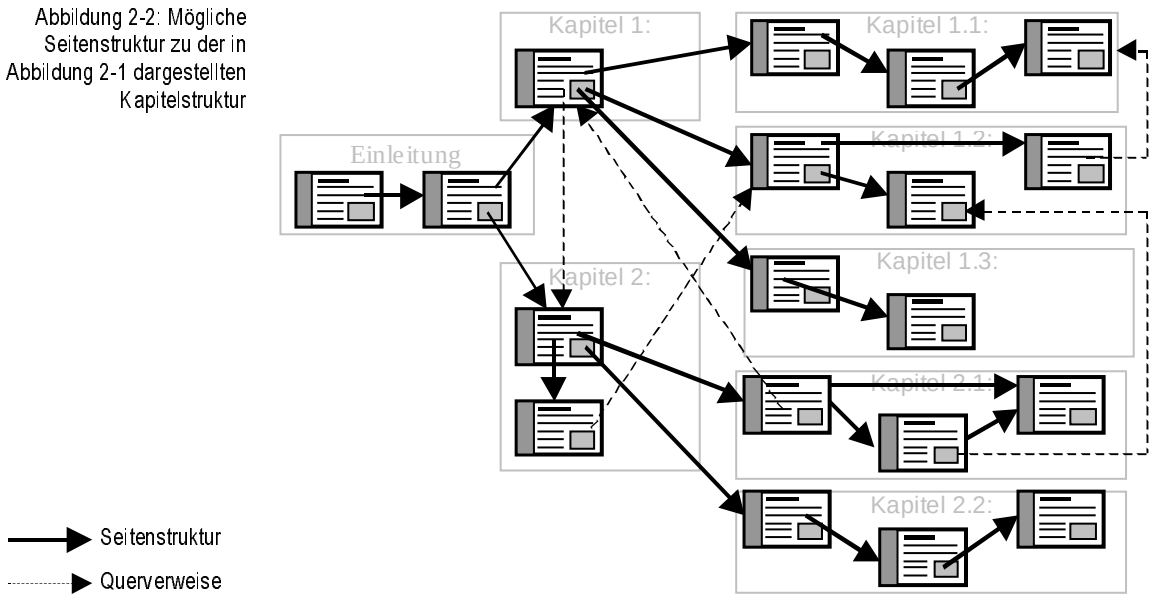


Abbildung 2-2: Mögliche Seitenstruktur zu der in Abbildung 2-1 dargestellten Kapitelstruktur

Die gestrichelten Pfeile stellen Querverbindungen zwischen Seiten dar, die über die Seitenstruktur hinweg existieren. Sie sind identisch mit den in Abbildung 2-1 dargestellten Referenzen und Verweisen innerhalb der Kapitelstruktur.

Zwischen Kapitelstruktur und Seitenstruktur besteht bei den meisten Informationssystemen eine 1:n Beziehung. Zu der gewählten Kapitelstruktur sind mehrere verschiedene Seitenstrukturen denkbar, die gewählte Seitenstruktur hingegen impliziert vor allem bei streng hierarchischen Dokumenten oftmals nur eine einzige mögliche Kapitelstruktur.

Jede Seite eines Lehr- oder Informationssystems besteht aus **Medienobjekten**; Texten, Grafiken, Fotografien, Videos, gesprochenem Text, Animationen, etc. Medienobjekte sind die kleinsten inhaltstragenden Bausteine eines hypermedialen Informationssystems. Sie können sowohl statisch (z.B. Text), als auch kontinuierlich (z.B. Videoclip) oder interaktiv (z.B. Multiple-Choice Fragen) sein.

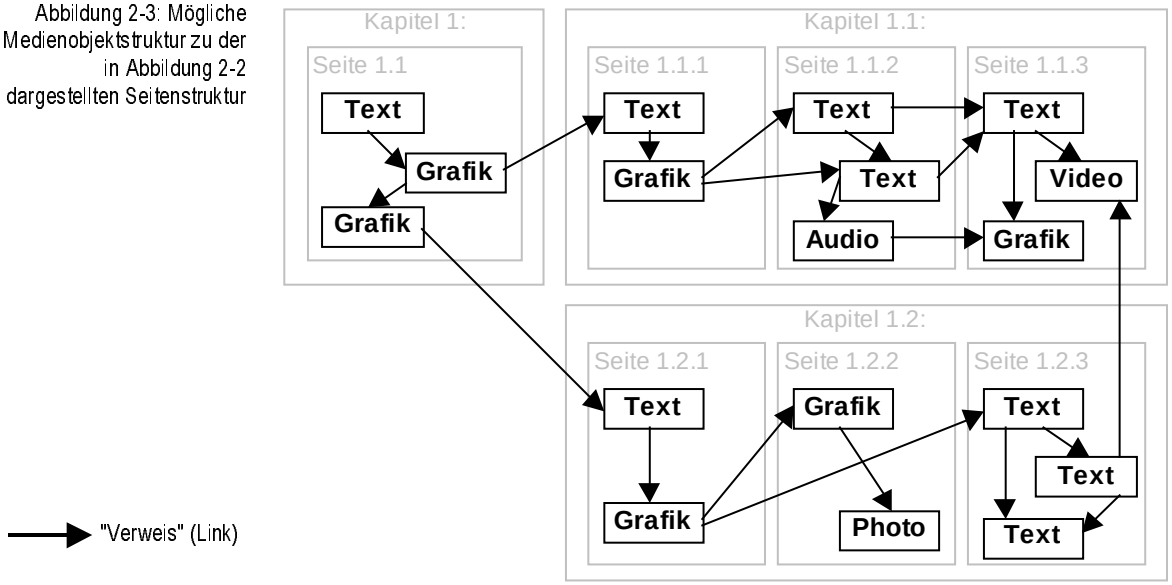
Medienobjekte bilden ebenfalls eine eigene Struktur, die im wesentlichen auf inhaltlichen Abhängigkeiten basiert. Medienobjekt A ist z.B. nur verständlich, wenn der Nutzer zuvor ein anderes Medienobjekt B betrachtet hat. In diesem Fall besteht innerhalb der **Medienobjektstruktur** eine Kante B → A.

Medienobjekte innerhalb einer Seite sind zumeist implizit sequentiell strukturiert, da davon ausgegangen wird, daß ein Benutzer die Objekte auf einer Seite von links oben nach rechts unten liest.

Kapitel- oder seiten-übergreifende Querverweise innerhalb der Kapitel- bzw. Seitenstruktur sind in der Medienobjektstruktur "normale" Verweise zwischen Medienobjekten, bei denen sich mehr oder minder zufällig Quelle und Ziel des Verweises auf verschiedenen Seiten befinden. Abbildung 2-3 zeigt einen Ausschnitt aus einer Medienobjektstruktur, wie sie als Grundlage für die in Abbildung 2-2 dargestellte Seitenstruktur dienen könnte.

Jede Seite setzt sich in diesem Beispiel aus zwei bis drei Medienobjekten zusammen. Abhängigkeiten und Verweise bestehen immer zwischen Medienobjekten und werden von der Medienobjektstruktur in die Seiten- und Kapitelstruktur übernommen. Auch die häufig anzutreffende Sequentialität der Seitenstruktur findet ihren Ursprung in einer zugrundeliegenden sequentialisierbaren Medienobjektstruktur.

Abbildung 2-3: Mögliche Medienobjektstruktur zu der in Abbildung 2-2 dargestellten Seitenstruktur

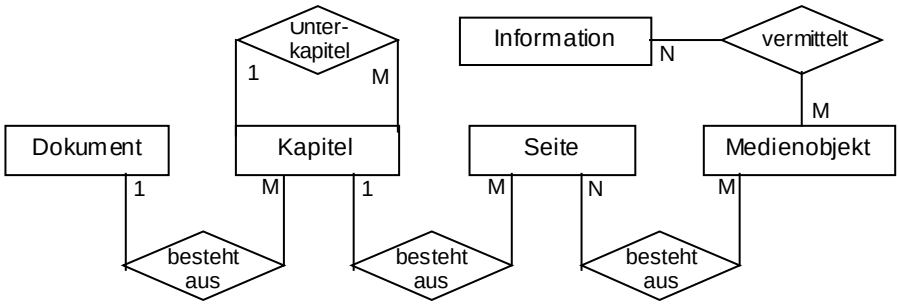


Aus der Abbildung wird deutlich, daß die Seitenstruktur eines Lehr- und Informationssystems im Grunde genommen nur ein Hilfskonstrukt zur strukturierten Zusammenfassung von Medienobjekten ist. Falls alle Medienobjekte zur Erklärung eines Sachverhaltes auf einer Seite dargestellt werden, ist die Seitenstruktur identisch mit der Kapitelstruktur. Falls eine Seite aus nur einem Medienobjekt besteht, ist die Seitenstruktur identisch mit der Medienobjektstruktur. Beide Fälle kommen in der Realität häufig vor.

Der Hauptgrund für die Existenz einer Seitenstruktur liegt vor allem in didaktischen, instruktionspsychologischen oder darstellerischen Motiven, die zum Ziel haben, den Informationsgehalt und das Layout einer jeden Seite möglichst ausgewogen zu gestalten [Nielsen97]. Im Gegensatz zu den logischen Strukturen auf Kapitel- und Medienobjekt-Ebene wird die Seitenstruktur daher zuweilen auch als Präsentationsstruktur bezeichnet [Wilkinson+96].

Das nachfolgende E-R Diagramm stellt noch einmal die Beziehungen zwischen den 3 Ebenen eines hypermedialen Dokuments im Überblick dar: Jedes Dokument besteht aus beliebig vielen Kapiteln, die ihrerseits wieder Unterkapitel besitzen können. Jedes Kapitel setzt sich aus einer Anzahl Seiten zusammen. Jede Seite besteht aus einem oder mehreren Medienobjekten, wobei einzelne Medienobjekte (z.B. Grafiken) auch mehrfach vorkommen können. Alle mit Hilfe des Dokumentes erlernbaren Informationen (Fakten Stichworte, etc.) werden durch die kleinsten atomaren Bausteine - Medienobjekte - vermittelt.

Abbildung 2-4: Beziehungen zwischen Kapiteln, Seiten und Medienobjekten



Jede Veränderung an den Einflußgrößen "Autor", "Nutzer" und "Thema" kann potentiell Auswirkungen auf die Medienobjektstruktur und damit auch auf die übergeordneten Strukturen haben. Zusätzlich kann eine andere Gewichtung von Inhalten bzw. eine veränderte thematische Abgrenzung durch den Autor auch direkt zu Veränderungen an der Kapitelstruktur führen.

Die vierte Einflußgroße, das Layout, beeinflußt lediglich die Seitenstruktur, d.h. eine Modifikation der im

Layout kodierten Ober- und Untergrenzen für Anzahl und Größe auf einer Seite darstellbarer Medienobjekte kann potentiell zu einer veränderten Seitenstruktur eines ansonsten unveränderten Dokuments führen.

2.2.2 Hyperlinks und Navigation

Kapitel-, Seiten- und Medienobjektstruktur können als gerichtete Graphen angesehen werden, deren Knoten die einzelnen Kapitel, Seiten bzw. Medienobjekte sind. Die Kanten der Graphen werden von **Hyperlinks** gebildet.

Jeder Hyperlink besitzt einen sog. **Anker**, der aktiviert werden muß, um einen Hyperlink zu verfolgen [Nielsen90]. Je nach Funktion eines Hyperlinks kann zwischen **strukturellen** (*organizational*) und **inhaltlichen** (*content-based*) Links unterschieden werden [Baron+96]. Strukturelle Links dienen dabei vor allem der Gliederung eines Dokuments während inhaltliche Links semantisch oder didaktisch begründete Referenzen und Verweise darstellen.

Das Aussehen der Anker und die Semantik der Hyperlinks ist von der Ebene abhängig, auf der sich der Link befindet:

Hyperlinks der **Kapitelstruktur** sind vor allem strukturelle Links, da sie der Navigation zwischen den einzelnen Kapiteln dienen. Ihre Anker sind in den meisten Fällen Menüs aus Grafiken und/oder Texten, über die zu einzelnen Kapitel eines Dokuments verzweigt werden kann.

Auch die Hyperlinks der **Seitenstruktur** dienen fast ausschließlich der Navigation. Ihre Anker sind üblicherweise Texte wie "nächste Seite" bzw. "vorherige Seite" oder entsprechende Grafiken.

Die Hyperlinks der **Medienobjektstruktur** sind im Gegensatz zu den Kanten der höheren Ebenen nicht auf die Beschreibung struktureller Beziehungen beschränkt. Sie orientieren sich vielmehr an inhaltlichen Aspekten, d.h. sie verbinden inhaltlich verwandte Medienobjekte unabhängig von ihrer Position in der Seiten- oder Kapitelstruktur.

Inhaltlichen Verbindungen zwischen Medienobjekte lassen sich in semantische, rhetorische und pragmatische Links unterteilen [Baron+96]:

- **Semantische Links** stellen vor allem Verbindungen zwischen ähnlichen, gegensätzlichen oder in "ist-ein" bzw. "ist-Teil-von" Beziehung zueinander stehenden Wörtern und Themen her. Semantische Links sind hauptsächlich auf der Medienobjekt-Ebene, vereinzelt aber auch auf der Seiten- oder Kapitelebene anzutreffen.
- **Rhetorische Links** dienen dazu, den Leser durch eine Folge von Informationen zu führen, um ein bestimmtes Lernziel zu erreichen. Rhetorische Links sind somit vor allem ein Werkzeug des Autors, mit dem Definitionen, Erklärungen, Illustrationen, Exkurse und ähnliche Elemente in einen Hypertext integriert werden können.
- **Pragmatische Links** stellen im Gegensatz zu rhetorischen Links keine Verbindungen zwischen Medienobjekten, sondern vielmehr zwischen dem behandelten Thema und dem vorgegebenen Lernziel bzw. der aktuellen Lernsituation des Lesers her. Beispiele für pragmatische Verbindungen sind Warnungen, Hinweise auf nutzerbezogene Beispiele und Benutzungshinweise.

Wichtig für die im Mittelpunkt dieser Arbeit stehende automatisierte Erzeugung von Lehr- und Informationssystemen ist vor allem der Spielraum, der dem dokumentenerzeugenden System bei der Verknüpfung von Medienobjekten gegeben ist: Semantische Links ergeben sich implizit aus dem Inhalt der Medienobjekte, d.h. sie sind unabhängig vom aktuellen Nutzer oder dem vom Autor zugrunde gelegten didaktisches Modell. Rhetorische und pragmatische Links hingegen bieten sehr weitreichende Möglichkeiten der Anpassung an die Einflußgrößen "Nutzer" und "Autor", da sie die Anbindung aller über die reine Wissensvermittlung hinausgehenden Elemente eines Lehr- oder Informationssystems bestimmen. Verwendung und Frequenz von rhetorischen und pragmatischen Links prägen damit entscheidend den Charakter eines wissensvermittelnden Dokuments.

Im Gegensatz zur zumeist hierarchisch aufgebauten Kapitel- und Seitenstruktur ist auf der Ebene der Medienobjekte die Struktur des behandelten Wissensgebietes nicht erkennbar. Wenn ein Dokument sehr

viele inhaltliche Hyperlinks enthält, kann es vorkommen, daß die Medienobjektstruktur die Kapitelstruktur überlagert. Die daraus resultierende Entkopplung des Dokuments von der semantischen Struktur des behandelten Themas kann dazu führen, daß der Nutzer den aktuellen Lernkontext oder gar sein Lernziel aus den Augen verliert ("lost in hyperspace").

Um diesem Effekt entgegenzuwirken, verwenden viele Lehr- und Informationssysteme verschiedene, zumeist auf oder gar oberhalb der Kapitelebene angesiedelte **Navigationstechniken**, die dem Nutzer helfen sollen, in einem Netz von Seiten und Medienobjekten nicht die Orientierung zu verlieren:

- *Site Maps* und andere Übersichten stellen entweder Kapitel- oder Seitenebene eines Dokuments ausschnittweise oder komplett grafisch dar.
- *Guided Tours* [Trigg88] bieten dem Nutzer einen oder mehrere sequentielle Pfade durch die Seitenstruktur eines Dokuments an. *Guided Tours* können dabei beliebig verlassen und wieder betreten werden.
- Wichtige und/oder markante Seiten oder Medienobjekte können als *Landmarks* verwendet werden. *Landmarks* sind von allen Seiten des Dokuments aus sichtbar und erreichbar.
- Über *Fisheye Views* können Übersichten der Dokumentstruktur so gestaltet werden, daß für die aktuelle Lernsituation relevante Seiten und Kapitel durch Positionierung in der Bildmitte grafisch hervorgehoben werden [Carlson89].
- Auch *Zoom & Roam* Techniken basieren auf einer Netzdarstellung der Dokumentstruktur. Der Nutzer hat hierbei die Möglichkeit, die organisatorische oder die semantische (konzeptuelle) Struktur in beliebiger Vergrößerung zu betrachten und per Mausklick einzelne Seiten des Dokuments aufzurufen [Beard+87].

Viele dieser Navigationstechniken sind generisch, so daß sie auf verschiedene Dokumente anwendbare Muster bilden [Garrido+97, Rossi+99] und dadurch auch für automatisch generierte Dokumente verfügbar gemacht werden können.

Neben diesen die globale Dokumentstruktur abbildenden Navigationshilfen existieren eine ganze Reihe weiterer Techniken, mit denen der Nutzer bei der Auswahl der idealerweise weiter zu verfolgenden Hyperlinks unterstützt werden soll. Beispiele hierfür finden sich in [Bieber+97, Kopetzky+99, Stanyer+99, Thüring+95 und Zellweger+98]

2.2.3 Genres von Kapiteln, Seiten und Medienobjekten

Die einzelnen Bausteine eines hypermedialen Dokuments - Kapitel, Seiten und Medienobjekte - sind nur in den seltensten Fällen uniform. Vielmehr erfüllt jeder Baustein eine bestimmte Aufgaben wie z.B. Vermittlung, Vertiefung oder Kontrolle von Wissen. Aus diesem Grund bietet es sich an, Kapitel, Seiten und Medienobjekte anhand von **Genres** zu klassifizieren.

> **Genres** dienen der Typisierung von Kommunikation anhand von Attributen wie Inhalt, Form und Verwendung und werden u.a. zur Charakterisierung von *Webseiten* und -*Sites* eingesetzt [Yates+92]. Beispiele für Genres von *Webseiten* sind Kataloge, *Homepages* und Produktpräsentationen [Watters+99a].

Genau wie sich ein Lehr- bzw. Informationssystem in seiner Struktur aus Kapiteln, Seiten und Medienobjekten zusammensetzt, so besteht es bezüglich Inhalt, Form und Verwendung aus verschiedenen Genres. Genres können jedoch nicht losgelöst von dem in Kapitel 2.1 beschriebenen Drei-Ebenen Model betrachtet werden, da z.B. die Verwendung eines Kapitels und die Verwendung eines einzelnen Medienobjektes nicht anhand der selben Kriterien beschreib- und klassifizierbar sind.

Genres, denen einzelne Kapitel der Kapitelstruktur zugeordnet werden können, sind unter anderem:

- **Wissensvermittlung**: Vermittlung von für den Betrachter relevantem, zumeist faktischem Wissen und Informationen.
- **Glossar**: Zusammenfassende Erklärung der wichtigsten Schlagworte des behandelten Themengebiets.
- **Zeitleiste**: Chronologische Auflistung von Ereignissen.
- **Spiel**: Interaktive, unmittelbare Vermittlung und Anwendung von Wissen.
- **Fallstudie**: Herstellen von Verbindungen zwischen Theorie und Praxis durch Schilderung eines authentischen Sachverhaltes.

Das Genre eines Kapitels gibt in den meisten Fällen das zu verwendende Strukturierungsschema vor. Glossar und Zeitleiste basieren so z.B. auf einer exakten, nachvollziehbaren Ordnung (alphabetisch bzw. chronologisch), während Wissensvermittlung und Fallstudie eine freie Ordnung ermöglichen, der allerdings oftmals implizite Sortierungen (Subjekte, Aufgaben, Benutzungsarten) zugrundeliegen [Thalheim98].

Welche Genres in einem Lehr- oder Informationssystem eingesetzt werden, ist dabei sowohl vom behandelten Thema als auch von der Zielgruppe und didaktischen Erwägungen abhängig. Für die automatisierte Erzeugung von Lehr- und Informationssystemen stellen die Genres der erzeugten Kapitel eines der wichtigsten Mittel der Nutzeranpassung dar, da sie den inhaltlichen Umfang und die Struktur eines Dokuments maßgeblich beeinflussen.

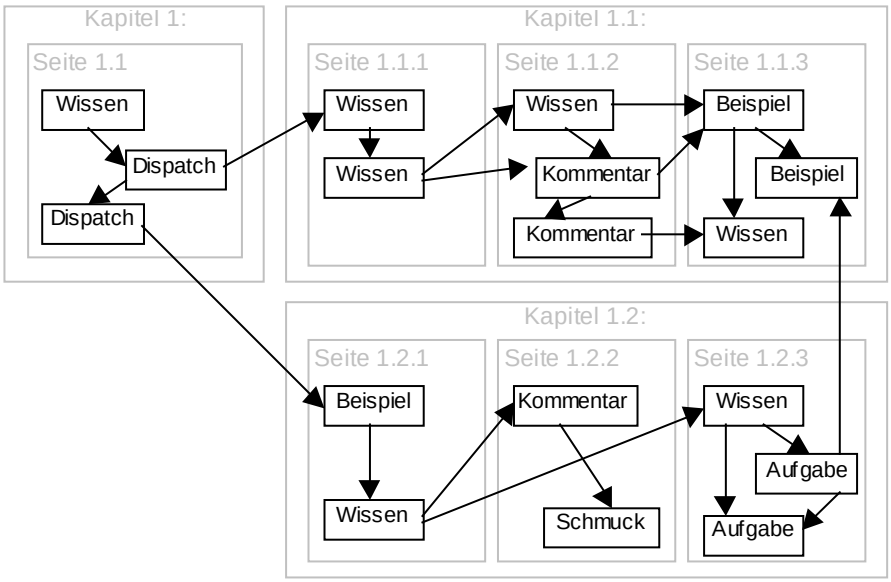
Das Genre einer Seite wird von den Genres der enthaltenen Medienobjekte geprägt. Hierbei sind vor allem die folgenden Kategorien für die Praxis von Lehr- und Informationssystemen relevant:

- **Wissensvermittlung**: möglichst objektive Vermittlung von Fakten.
- **Kommentar**: subjektive Interpretation oder Bewertung eines Sachverhaltes.
- **Beispiel**: Vertiefung oder Verdeutlichung eines Sachverhalts durch ein Beispiel.
- **Aufgabe**: Überprüfung von erlerntem Wissen.
- **Referenz**: Verweis auf verwandte Themen bzw. weiterführende Informationen oder Angabe von Quellen.
- **Glossar**: Komprimierte Erklärung eines Stichworts.
- **Dispatcher**: Übersicht über ein Thema mit der Funktion eines Ankers für weiterführende Informationen.
- **Ausschmückung**: Medienobjekt, dessen Verwendung eher dem Layout als dem Inhalt einer Seite dient.

Die Genres der in einem Lehr- oder Informationssystem enthaltenen Medienobjekten haben einen großen Einfluß auf die Medienobjektstruktur des Dokuments, da sie nicht beliebig kombinierbar sind. Eine Aufgabe zu einem Thema folgt z.B. üblicherweise nach der Wissensvermittlung zu diesem Thema, während *Dispatcher* eher vor der Wissensvermittlung angesiedelt sind.

In der nachfolgenden Variante von Abbildung 2-3 sind anstatt der Kodierungen die Genres der verwendeten Medienobjekte angegeben.

Abbildung 2-5: Medienobjektstruktur mit Angabe der Genres der verwendeten Medienobjekte



Genres von Medienobjekten spielen bei der Erzeugung von hypermedialen Dokumenten eine große Rolle, da sie Regeln für die Ordnung von Medienobjekten festlegen. Ein Beschreibung der daraus abgeleiteten Heuristiken findet sich in Kapitel 7.6.2.

Wenn Medienobjekte zu Seiten zusammengefaßt werden, sind ihre Genres verschieden dominant. Falls z.B. auf einer Seite eine Multiple-Choice Aufgabe mit 5 ausschmückenden Grafiken zusammengefaßt wird, so ist die Seite trotz der zahlenmäßigen Überlegenheit ausschmückender Objekte eine Aufgaben-Seite. Der Grund hierfür ist, daß Aufgaben den Charakter einer Seite stärker prägen als Photos oder andere ausschmückende Elemente.

Die Dominanz eines Genres ist ein wichtiger Aspekt beim Zusammenfassen von Medienobjekten zu Seiten. Medienobjekte bestimmter Genres - z.B. Aufgaben und Beispiele - lassen sich nur schlecht zu einer Seite zusammenfassen, während andere wiederum sehr gut miteinander harmonieren.

2.3 Didaktische Aspekte

Da Lehrsysteme in erster Linie der Vermittlung von Wissen dienen, spielen bei ihrer Erstellung didaktische Aspekte eine große Rolle. Das für die Wissensvermittlung anzunehmende didaktische Modell ist somit ein wesentlicher, Inhalt und Struktur des Dokuments beeinflussender, Bestandteil der Einflußgröße "Autor" (siehe Kapitel 1). Leider ist die Modellierung von didaktischen Regeln sowie die Festlegung ihrer konkreten Auswirkungen auf Inhalt und Struktur eines Dokuments so gut wie unmöglich, da nicht nur starke Wechselwirkungen mit anderen Einflußgrößen bestehen, sondern vor allem auch situative und subjektive Aspekte eine große Rolle spielen. Dennoch lassen sich einige Auswirkungen verschiedener Lehrmodelle auf Inhalte, Strukturen und Genres der drei Ebenen von Lehrsystemen zumindest teilweise beschreiben und berücksichtigen.

In diesem Abschnitt sollen einige der wichtigsten instruktionspsychologischen Modelle kurz vorgestellt werden. Ausführliche Diskussionen didaktischer Aspekte hypermedialer Lehrsysteme finden sich in [Coppola+99, Issing97, Riehm+96, Weidenmann93 und Jonassen+90].

Grundlage des **didaktischen Modells**, auf dem ein Lehrsystem beruht, ist immer eine Annahme über die Art und Weise, wie der Lernende am effektivsten neues Wissen aufnimmt und speichert. Generell werden hierbei drei Theorien unterschieden, an denen sich fast alle existierenden Lehrsysteme mehr oder weniger stark orientieren [Issing97]:

Behaviorismus: Im Zentrum der behavioristischer Lehrmodelle steht die Vermittlung von Wissen durch Instruktion [Skinner68]. Lernen wird als deterministischer Prozeß aufgefaßt, d.h. es wird impliziert,

daß bestimmte Reize zwangsläufig zu bestimmten Reaktionen führen. Beispiele für auf dem Behaviorismus basierende Lehrsysteme sind neben reinen *drill & practice* Systemen vor allem sog. *Intelligente Tutoring Systeme* (ITS) wie ELM-ART [Brusilovsky+96] und LANCA [Frasson+98].

Kognitivismus: Auf dem Kognitivismus basierende Lehrsysteme versuchen, die (vermuteten) kognitiven Prozesse des Lerners beim Erfassen und Verarbeiten von Wissen auszunutzen. Basis der meisten in diesen Bereich fallenden Lehrsysteme sind die Theorien von John R. Anderson zu kognitiven Vorgängen beim Lösen von Problemen [Anderson93].

Konstruktivismus: Konstruktivistisch ausgerichtete Lehrsysteme gehen von einer aktiven Rolle des Lerners, von einem Erarbeiten anstelle eines Vermittelns von Wissen aus. Im Mittelpunkt stehen dabei oftmals authentische, realitätsnahe Fallstudien und Szenarien, über die konkrete Situationen und Probleme wahrgenommen und bewältigt werden sollen [Gerstenmaier+94]. Medien spielen dabei die Rolle von "kognitiven Werkzeugen", die dem Lernenden Anregungen und Hilfestellungen bieten [Jonassen91].

Beispiele für aus dem Konstruktivismus abgeleitete Ansätze zu Erstellung und Einsatz von Lehrsystemen sind *Situated Learning* [Mandl+97], *Cognitive Apprenticeship* [Collins+89] und *Anchored Instruction* [Vanderbilt90].

Um ein didaktisches Modell im Rahmen eines Lehrsystems umzusetzen, existieren verschiedene Möglichkeiten, die jeweils verschiedene Ebenen des Dokuments betreffen:

- Insbesondere auf den höheren Ebenen ist durch verschiedene Strukturierungen der Kapitel und Seiten ein erheblicher Spielraum für die Umsetzung verschiedener Instruktionsmodelle gegeben. Für behavioristisch ausgerichtete Lehrsysteme bietet sich so z.B. die Verwendung einer eher sequentiellen Kapitel- und Seitenstruktur an, während stark vernetzte, zur Exploration des Wissensraums anregende Strukturen eher konstruktivistischen Ideen entsprechen [Mayes+90].
- Die Verwendung, Kombination und Gewichtung verschiedener Genres auf Kapitel- und Medienobjekt-Ebene bietet vielfältige Möglichkeiten der Anpassung an ein didaktisches Modell. Beim *Anchored Instruction* Ansatz steht z.B. eine authentische Videogeschichte im Zentrum des Lehrsystems, während *drill & practice* Ansätze fast ausschließlich auf den Genres Wissensvermittlung und Wissensabfrage aufbauen.

Neben den Genres der verwendeten Medienobjekte haben auch deren **Kodierungen** einen erheblichen Einfluß auf den Charakter und die didaktische Ausrichtung eines Lehrsystems. Die Verwendung unterschiedlicher Kodierungen ist darüber hinaus auch ein geeignetes Mittel zur Anpassung an die Zielgruppe des Dokuments. Lehrsysteme für Kinder basieren so z.B. oftmals sehr stark auf Grafiken, Photos und kontinuierlichen Medien, während sich Lehrsysteme für den universitären Bereich eher durch eine starke Verwendung von Texten und Grafiken charakterisieren lassen.

Ziel vieler Forschungsprojekte im Spannungsfeld zwischen Multimedia und Didaktik ist die Entwicklung von konkreten, schrittweisen Anleitungen zur Erstellung von multimedialen Lehrsystemen. Beispiele hierfür sind:

- die Entwicklung von Methodologien und Entwicklungszyklen auf der Basis konzeptueller Abhängigkeiten [Rossi+95],
- Checklisten für die Auswahl der geeigneten Medien-Kodierung anhand der Art des zu vermittelnden Wissens [Sutcliffe99],
- Maßgaben für die Auswahl bzw. Erstellung von Abbildungen unter Berücksichtigung ihrer Funktion [Weidenmann88].

Das große Problem mit diesen mediendidaktischen Regeln und Heuristiken ist jedoch, daß sie entweder sehr beispielbezogen und nicht generalisierbar oder aber sehr abstrakt und modellhaft sind [Riehm+96]. Daher ist besonders bei der im Mittelpunkt dieser Arbeit stehenden automatisierten Erzeugung von Lehrsystemen die Auswahl der auf das behandelte Thema anwendbaren mediendidaktischen Faustregeln ein wichtiger und für das Ergebnis kritischer Punkt.

2.4 Beispiel: Mathe Online (Universität Wien)

Die in den Abschnitten 2.2.1 bis 2.2.3 beschriebenen strukturellen Merkmale finden sich in (fast) allen hypermedialen Lehr- und Informationssystemen. Als Beispiel soll an dieser Stelle das Lehrsystem "Mathe online" der Universität Wien bezüglich der verwendeten Strukturen und Genres untersucht werden.

"Mathe online" ist ein interaktives Lehrsystem zur Oberstufen-Mathematik, das sowohl für den Einsatz in Schulen und Universitäten als auch für das Selbststudium konzipiert ist. "Mathe online" soll als CD-ROM erscheinen; momentan ist jedoch lediglich ein Ausschnitt des Systems im *World Wide Web* abrufbar (http://www.univie.ac.at/future.media/mo/).

Der von "Mathe online" abgedeckte Stoff ist in einzelne Kapitel gegliedert, wobei sich jedes Kapitel aus mehreren, inhaltlich parallele **Ebenen** zusammensetzt. Insgesamt sind 10 verschiedenen Ebenen für jedes Kapitel geplant [Embacher97a]. Die wichtigsten dieser Ebenen sind:

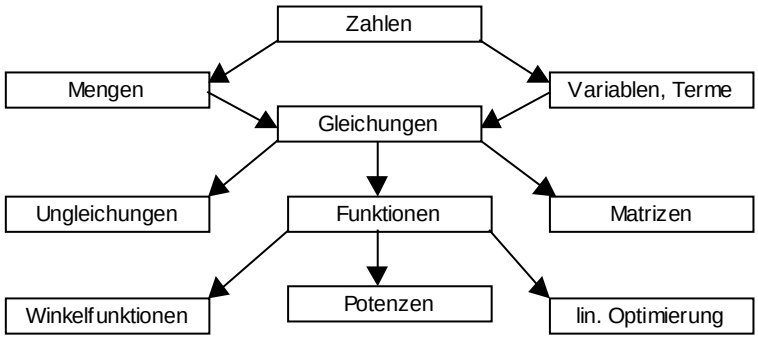
- **Text**: Darstellung des Stoffs eines Kapitels im Stile eines Lehrbuchs unter Einbeziehung interaktiver Elemente und Animationen. Geplant sind zwei Textebenen, wobei die erste Textebene Grundlagenwissen vermittelt, das von der zweiten vertieft wird.
- **Beispiele:** Beispiele zum in den Textebenen behandelten Stoff, wobei die interaktiven Möglichkeiten des Mediums CD-ROM bzw. *World Wide Web* mit Hilfe von *Java Applets* möglichst intensiv genutzt werden sollen. Zu jeder Textebene existiert eine eigene Beispielebene.
- **Wissenswertes** und **Historisches**: Darstellung alltagsrelevanter mathematischer Techniken, bzw. Anmerkungen zu wichtigen Personen und Ereignissen aus der Geschichte der Mathematik. Ziel dieser Ebenen ist die Vermittlung von Allgemeinwissen und die Motivation der Lernenden.

Jede Ebene eines Kapitels wird auf einer HTML-Seite dargestellt. Abbildung 2-6 zeigt z.B. die erste Textebene des Kapitels "Gleichungen".

Abbildung 2-6: Bildschirmseite des Lehrsystems „Mathe Online“

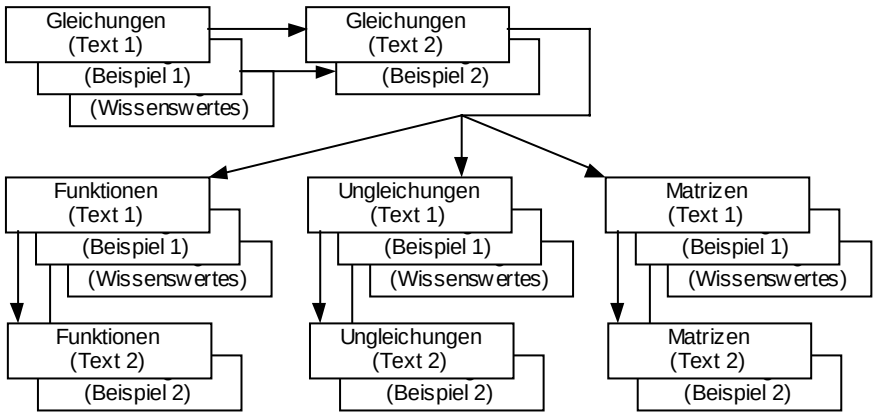
Die Kapitelstruktur von "Mathe online" besteht aus den einzelnen behandelten Themengebieten. Ihre Gliederung basiert ausschließlich auf inhaltlichen Gesichtspunkten und ist weitestgehend hierarchisch (Abbildung 2-7).

Abbildung 2-7: Ausschnitt aus der Kapitelstruktur von "Mathe online"



Die einzelnen Ebenen der Kapitel bilden die Seitenstruktur. Die Besonderheit hierbei ist, daß die Seiten eines Kapitels nicht hierarchisch oder sequentiell angeordnet sind, sondern parallel. Die Seiten eines Kapitels haben - mit Ausnahme der beiden Textebenen - keine gemeinsame Wurzel und keine Vorgänger oder Nachfolger innerhalb des selben Kapitels. Die globale (kapitelübergreifende) Seitenstruktur hingegen ist hierarchisch aufgebaut: Die Textebenen bilden die Struktur der Kapitelebene ab und stellen darüber hinaus Anknüpfpunkte für die anderen Ebenen repräsentierende, untereinander nicht verbundene Seiten dar (Abbildung 2-8). Auch bei "Mathe online" existiert somit die in Kapitel 2.2.1 angesprochene 1:n Abbildung von der Kapitel- auf die Seitenstruktur.

Abbildung 2-8: Ausschnitt aus der Seitenstruktur von "Mathe online"



Das Genre der Seiten wird durch ihre Ebene festgelegt, d.h. auf der Ebene "Beispiel" sind nur Seiten vom Genre "Beispiel" angesiedelt.

Die einzelnen Seiten von "Mathe online" bestehen aus teilweise mehr als 100 Medienobjekten. Die meisten Medienobjekte sind textuell, aber auch Grafiken und interaktive Elemente werden benutzt. Wie schon bei den Kapiteln bestimmt bei den einzelnen Seiten das Genre der Ebene das Genre der bevorzugt verwendeten Medienobjekte. Die Ebene "Beispiel" besteht so z.B. aus einer Seite vom Genre "Beispiel", die wiederum aus mehreren Medienobjekten vom Genre "Beispiel" zusammengesetzt ist.

Während die Seitenstruktur aus mehreren disjunkten, zumeist hierarchischen Einzelstrukturen der benutzten Genres besteht, bildet die Medienobjektstruktur einen Graphen, der keinerlei Besonderheiten aufweist, die auf die parallele Seitenstruktur hinweisen oder diese gar erfordern (Abbildung 2-9).

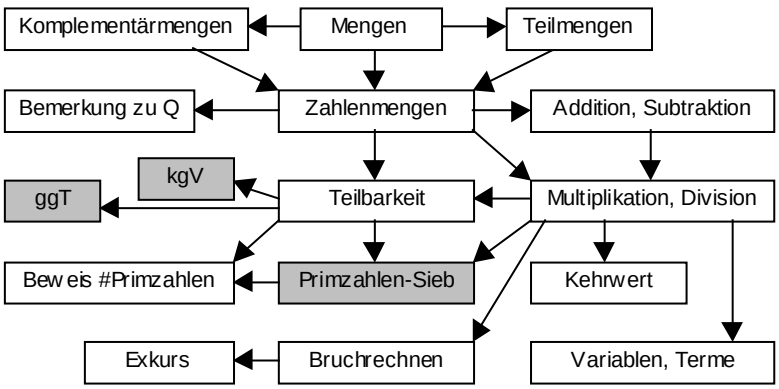


Abbildung 2-9: Ausschnitt aus der Medienobjektstruktur von "Mathe online". Die grau hinterlegten Kästchen repräsentieren interaktive Medienobjekte, alle anderen Medienobjekte sind Texte.

Eine Besonderheit von "Mathe online" ist, daß Medienobjekte ausschließlich seiteninterne, organisatorische Hyperlinks beinhalten. Alle semantischen, rhetorischen und pragmatischen Hyperlinks werden außerhalb der Medienobjekte am rechten Rand einer Seite dargestellt (siehe Abbildung 2-6). Die Autoren von "Mathe online" unterscheiden dabei lediglich die drei Link-Typen "Rückgriff", "Vorgriff" und "Querverweis". Der Linktyp jedes Hyperlinks wird durch einen entsprechenden Text und ein farbiges Symbol explizit dargestellt.

Von den in Kapitel 2.2.2 beschriebenen Navigationstechniken realisiert "Mathe online" lediglich eine statische, globale *Sitemap* der Kapitelstruktur, die gleichzeitig als Einstiegspunkt in das System dient. Da die einzelnen Seiten teilweise sehr lang sind, beginnt jede Seite mit einer Liste von Hyperlinks zu allen auf der Seite behandelten Stichworten.

Wie diese Betrachtungen zeigen, kann auch ein auf den ersten Blick ungewöhnlich strukturiertes Lehrsystem wie "Mathe online" bei genauerer Analyse auf das in Kapitel 2.2.1 beschriebene Modell aus Kapitel-, Seiten- und Medienobjektstruktur zurückgeführt werden. Viele der beschriebenen strukturellen Eigenschaften von "Mathe online" lassen sich für den überwiegenden Teil der existierenden Lehr- und Informationssysteme verallgemeinern:

- Kapitel- und Seitenstruktur sind überwiegend hierarchisch aufgebaut, wobei oftmals die Seitenstruktur lediglich eine Verfeinerung der Kapitelstruktur darstellt.
- Die Medienobjektstruktur ist stark vernetzt. Ihre Struktur und Zusammensetzung bedingt keinerlei Einschränkungen bezüglich der darüberliegenden Seitenstruktur.
- Ausnahmen bestätigen die Regel. Viele Lehr- und Informationssysteme verwenden auf der Seitenebene zumindest partiell nicht-hierarchische Strukturen. Beispiele sind die Ebenen eines Kapitels in "Mathe online" oder die Kombination aus hierarchischen und sequentiellen Seitenstrukturen in "Statistik interaktiv" [Caumanns+99c].

Diese Beobachtungen bilden zusammen mit dem 3-Strukturen-Modell die Grundlage für die im Mittelpunkt dieser Arbeit stehende automatisierte Erstellung von Lehr- und Informationssystemen.

3 Erstellung von Lehrsystemen

Authoring adaptive hypermedia remains a problem area, whether we are considering Web-based systems or not. Systems like Interbook and AHA were designed and implemented by computer scientists and the first applications of these systems were developed by (the same) computer scientists as well. Authoring is probably still too complicated for "average" authors from non-computer-related fields.

Paul De Bra [DeBra99]

Bezogen auf die in Kapitel 1.2 eingeführte Sicht eines Dokuments als Funktion aus vier weitgehend unabhängigen Einflußgrößen stellen die in Kapitel 2 dargestellten Dokumentenstrukturen jeweils einzelne Funktionspunkte dar. Bei einer Veränderung des Autors, Nutzers oder des gewünschten Layouts werden sich Abweichungen auf einer oder mehreren Ebenen ergeben, die von einfachen Umstrukturierungen bis zu kompletten inhaltlichen Neuausrichtungen reichen können.

Eines der wesentlichen Leistungsmerkmale von Autorenzyklen zur Erstellung hypermedialer Dokumente ist es daher, Autoren Werkzeuge oder zumindest Prozesse (sog. “Halbzeuge”) an die Hand zu geben, die es erlauben, notwendig werdende Veränderungen auf einer oder mehreren strukturellen Ebenen mit möglichst wenig Aufwand durchführen zu können. Da insbesondere zur Wissensvermittlung geeignete Dokumente oftmals eine stark vernetzte Struktur besitzen, sind die Abhängigkeiten zwischen den einzelnen Strukturebenen potentiell sehr komplex, wodurch zumindest eine automatisierten Verifikation strukturübergreifender Änderungen möglich sein sollte.

In diesem Kapitel wird der notwendige Übergang von komplett manuellen Autorenzyklen zu automatisierbaren Modellen motiviert.

3.1 Manuelle Entwicklungszyklen

Die Mehrzahl aller existierenden hypermedialen Lehr- und Informationssysteme wurde mehr oder weniger manuell erstellt. Im Fall von web-basierten Dokumenten beschränkt sich der Einsatz von Hilfswerkzeugen oftmals auf grafische HTML Editoren oder die Konvertierung existierender *Word* oder *TeX* Dateien.

Solche vorwiegend manuell erstellten Dokumete sind oftmals von sehr guter Qualität was die Methodik und Präsentation betrifft. Im Gegenzug führt jedoch die stark eingeschränkte Werkzeugunterstützung zu einer unzureichende Berücksichtigung der dynamischen Einflußgrößen "Nutzer", "Thema" und "Autor" , wodurch die Wartung und Aktualisierung erschwert wird.

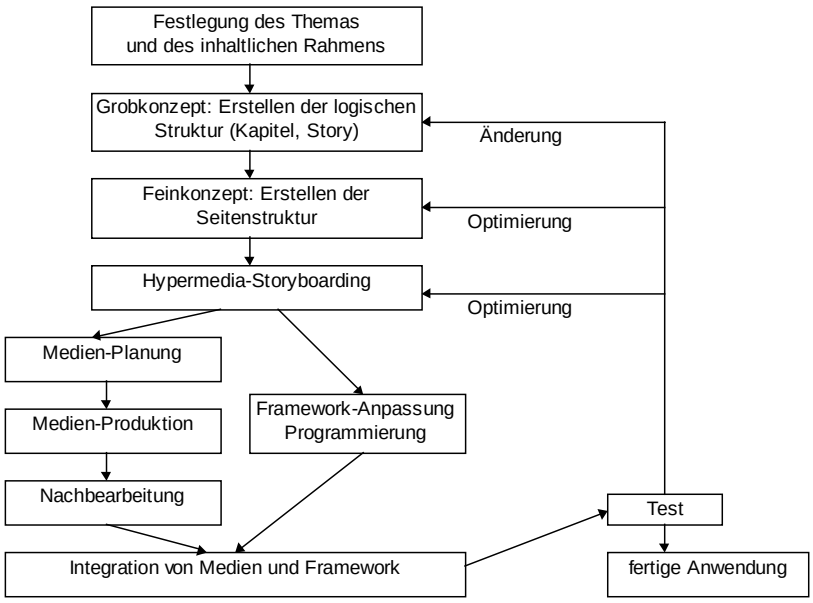
Dies soll im folgenden am Beispiel des **DIALEKT Entwicklungszyklus** [Apostolopoulos+96] verdeutlicht werden:

DIALEKT (Digitale Interaktive Lektionen) ist ein vom DFN-Verein gefördertes Projekt am *Center für digitale Systeme* (CeDiS) der FU Berlin[8], mit dem Ziel, multimediale Lehreinheiten für den Einsatz in der Hochschullehre zu entwickeln. Ein Schwerpunkt des Projekts liegt in der Einbindung von Breitbandnetzen zur Verteilung von kontinuierlichen Medien über das Internet.

Der im DIALEKT-Projekt verfolgte Entwicklungszyklus zur Erstellung hypermedialer Lehrsysteme ist nicht der einzig existierende Ansatz, aber er ist durchaus repräsentativ für Verfahren zur manuellen Erstellung von komplexen, hypermedialen Dokumenten.

Abbildung 3-1 zeigt den auf dem Wasserfall-Modell basierenden DIALEKT-Entwicklungszyklus, wie er für die Erstellung hypermedialer Lernsysteme zu den Themen "Marketing" [Apostolopoulos+96], "Steuern und Investitionsrechnung" [Buchmann+98] und "Statistik" [Caumanns+99c] verwendet wurde.

Abbildung 3-1: Manuelle Erstellung hypermedialer Lehreinheiten am Beispiel des DIALEKT-Entwicklungszyklus



Ausgangspunkt der Erstellung eines Lehrsystems nach dem DIALEKT-Ansatz ist die Festlegung des zu behandelnden Themas sowie dessen inhaltliche Eingrenzung. Ergebnis dieses ersten Arbeitsschrittes ist eine Auflistung von Themen und Stichworten, die mit Hilfe des Lehrsystems erlernbar sein sollen. Themen und Stichworte können zusätzlich gewichtet werden, um inhaltliche Prioritäten zu setzen.

Anschließend wird ein sogenanntes **Grobkonzept** erstellt, das im wesentlichen die Festlegung der Kapitelstruktur beinhaltet. Im nächsten Schritt wird basierend auf dem Grobkonzept ein **Feinkonzept** - die Seitenstruktur - erarbeitet.

Die Beschreibung der auf den einzelnen Seiten zu plazierenden Medienobjekte geschieht mittels **Hypermedia Storyboards**. Ein *Storyboard* enthält Beschreibungen aller auf einer Seite zu plazierenden Texte, Grafiken, Videos und sonstigen Medien. Darüber hinaus legt das *Storyboard* fest, wie die Interaktionen des Nutzers mit den auf der Seite plazierten Objekten aussehen können und welche Aktionen durch sie ausgelöst werden. Die *Storyboards* eines Lehrsystems beschreiben zusammen sowohl seine Seiten- als auch seine Medienobjektstruktur.

Diese Reihenfolge der Erstellung der einzelnen strukturellen Ebenen - zuerst Kapitelstruktur, dann Seitenstruktur, zuletzt Medienobjektstruktur - findet sich bei (fast) allen Verfahren zur manuellen Erstellung von Lehrsystemen. Da man sich hierbei von der obersten, abstraktesten Ebene zur niedrigsten, konkretesten Ebenen über schrittweise strukturelle Verfeinerungen durcharbeitet, werden diese Verfahren auch als analytische oder **Top-Down Ansätze** bezeichnet [Caumanns98].

Wenn alle *Hypermedia Storyboards* erstellt sind, beginnt die praktische Umsetzung des Lehrsystems. Hierbei wird versucht, die Anfertigung der benötigten Medien parallel zur Kodierung des *Frameworks*

[8] http://www.dialekt.cedis.fu-berlin.de/

durchzuführen. Unter den Begriff *Framework* fallen dabei alle zu programmierenden Funktionalitäten, vom HTML Code einer Bildschirmseite über *JavaApplets* bis zu komplexen Laufzeitumgebungen (Navigation, Mediensteuerung, etc.).

Nach der Zusammenführung von Medien und *Framework* liegt die erste Version des Lehrsystems zum Testen vor. Beim Testen eines Lehrsystems (insbesondere wenn die späteren Nutzer als Testpersonen eingesetzt werden) fallen üblicherweise eine Vielzahl von strukturellen und inhaltlichen Ungereimtheiten in jeder der drei strukturellen Ebenen auf.

Häufig anzutreffenden Probleme sind:

- Informationen werden nicht gefunden, da die Kapitelstruktur nicht identisch mit der konzeptuellen Struktur des behandelten Wissensgebiets bzw. dem mentalen Model des Nutzers über diese Struktur ist.
- Seiten sind zu überfüllt bzw. zu leer.
- Informationen werden redundant, unzureichend oder unangemessen vermittelt.

Abhängig von der Art der identifizierten Schwachpunkte müssen eine, mehrere oder manchmal auch alle strukturellen Ebenen überprüft und gegebenenfalls modifiziert werden. Dieses Wechselspiel aus Test und Optimierung wird durchgeführt, bis ein zufriedenstellendes Resultat - die fertige Anwendung - erzielt ist.

Die Stärken dieses Ansatzes liegen vor allem in seiner Einfachheit und in der strikten Trennung zwischen Konzeption und Umsetzung eines Lehrsystems:

- Der DIALEKT Entwicklungszyklus orientiert sich an der allgemein bekannten und vertrauten Erstellung von nicht-hypermedialen Dokumenten wie z.B. Büchern. Auch hier wird vom Abstrakten (der Gliederung) zum Konkreten (Text, Abbildungen) hin gearbeitet.
- Der Ansatz sieht eine Trennung zwischen fachspezifischem und umsetzungsrelevantem Wissen vor. Die Erstellung von Kapitel-, Seiten- und Medienobjektstruktur wird von Autoren aus dem behandelten Themengebiet durchgeführt, während die Programmierung und die Medienproduktion von diesbezüglichen Fachleuten geleistet wird. Einzige Schnittstelle sind die von den Autoren erstellten *Storyboards*.

Dem gegenüber stehen jedoch auch zwei gravierende Nachteile:

- Nutzeranpassung spielt in den meisten manuellen Entwicklungszyklen keine Rolle. Auch der DIALEKT Ansatz sieht dies (mit Ausnahme von rudimentärer Bandbreitenanpassung für kontinuierliche Medien) nicht vor.
- Um ein fertiges Dokument inhaltlich zu erweitern, muß unabhängig von Art und Umfang der Erweiterung der komplette Zyklus erneut durchlaufen werden. Änderungen von Texten und Animationen sind nicht nur auf das *Storyboard* beschränkt, sondern können Auswirkungen auf das Fein- und Grobkonzept des gesamten Dokuments haben. Selbiges trifft auch auf Veränderungen der inhaltlichen Schwerpunkte oder der Zielgruppe zu.

Die genannten Vor- und Nachteile sind nicht spezifisch für den DIALEKT Ansatz. Insbesondere die starke Einbindung von Fachautoren, die fehlende Adaptierbarkeit und der iterative Entwicklungszyklus sind typisch für die manuelle Erstellung von Lehr und Informationssystemen und auch bei anderen Ansätzen wie z.B. dem *Object-Oriented Hypermedia Design Model* (OOHDM) [Rossi+95]anzutreffen.

Über konzeptionellen Nachteile (fehlende Adaptierbarkeit und iterativer Entwicklungszyklus) hinaus existieren leider auch die oben genanten Vorteile größtenteils nur in der Theorie. Dadurch, daß sich der Ansatz konzeptionell an der Erstellung von Büchern orientiert, ist er den von der fachlichen Seite her kommenden Autoren vertraut. Diese Vertrautheit ist bei der Erstellung nicht-sequentieller Strukturen jedoch oftmals eher einschränkend als förderlich. Ein vorgegebener Inhalt muß für ein web-basiertes Lehrsystem anders aufbereitet werden, als für ein Buch. Vertraute Entwicklungszyklen fördern jedoch auch vertraute Ergebnisse, insbesondere wenn sie keinerlei Unterstützung bezüglich der Besonderheiten, Probleme und Möglichkeiten des verwendeten Mediums bieten.

Das Hauptproblem bei der starken Einbindung von Fachautoren in die Erstellung hypermedialer Lehr-

und Informationssysteme ist, daß diese im Normalfall nur mit der Erstellung von Büchern und anderen sequentiell strukturierten Printmedien vertraut sind. Die Probleme, die auftreten, wenn ein Buchautor ein hypermediales Dokument erstellen soll, liegen vor allem in den komplexeren Anforderungen an die Strukturierung des Inhalts sowie im angemessenen Einsatz der zur Verfügung stehenden Möglichkeiten der Wissensvermittlung:

- Informationen müssen für *Online*-Systeme anders portioniert werden als für gedruckte Dokumente. Der Inhalt eines Lehr- oder Informationssystems wird nicht durch eine Reihe kontinuierlicher Texte, sondern durch einzelne, vom Umfang her relativ kleine Medienobjekte vermittelt. Darüber hinaus erfolgt die Strukturierung des Inhalts nicht nur über logische Einheiten wie z.B. Kapitel, sondern in sehr starkem Maße auch über darstellerische Einheiten wie z.B. Bildschirmseiten.
- Der Computermonitor ist nur bedingt zum Lesen längerer Texte geeignet. Aus diesem Grund spielen alternative Formen der Wissensvermittlung wie z.B. Bilder, Grafiken oder Animationen eine herausragende Rolle. Ein vermehrter Einsatz nicht-textueller Medien führt jedoch auch dazu, daß der Charakter eines hypermedialen Lehrsystems zwangsweise ein anderer ist als der eines Buches.
- Der angemessene Einsatz von hypermedia-spezifischen Konzepten wie Hyperlinks und Interaktivität sowie die Verwendung kontinuierlicher Medien erfordern Erfahrungen und Kenntnisse, die über reines Fachwissen und "klassische" didaktische Modelle hinausgehen.

Ein DIALEKT-ähnlicher Entwicklungszyklus, der Autoren mit der multimedialen Aufbereitung der Inhalte weitestgehend alleine läßt, verlangt von diesen Kenntnisse und Fähigkeiten, die sie normalerweise nicht besitzen. Dies hat zur Folge, daß die erstellten Strukturen, Seiten und *Storyboards* oftmals von Personen überarbeitet werden müssen, die mit der multimedialen Aufbereitung von Inhalten vertraut sind (z.B. Multimedia-Autoren).

Multimedia-Autoren sind jedoch mit dem behandelten Fachgebiet nicht vertraut, so daß jede Änderung an einem Text oder einer Animation eine anschließende Kontrolle durch die Fachautoren erfordert. Da diese nun wiederum inhaltliche Änderungen vornehmen können, ist es durchaus nicht ungewöhnlich, wenn ein einziger Text oder eine einzige Animation tagelang zwischen Fach- und Multimediaautoren hin- und hergereicht wird.[9]

Neben dieses unmittelbar bei der Erstellung eines Lehrsystemes auftretenden Problemen, besitzen manuelle Entwicklungszyklen wie der DIALEKT-Ansatz eine Reihe zusätzlicher Nachteile, die erst bei Einsatz des erstellten Lehrsystems zum Tragen kommen:

- unzureichende Möglichkeiten der Erweiterung und Modifikation durch die Notwendigkeit der manuellen Überprüfung der Konsistenz der strukturellen Ebenen,
- mangelnde Möglichkeiten der Konfiguration und Anpassung aufgrund statischer Strukturen,
- erschwerte Wartung durch manuell erstellte und fest mit Medienobjekten verbundene Hyperlinks [Thistlewaite97].

Diese Nachteile schlagen sich - wie oben beschrieben - negativ in den Kosten und dem Aufwand von Entwicklung und Wartung, der Qualität und in den potentiellen Einsatzmöglichkeiten der erstellten Dokumente nieder. Darüber hinaus verhindern sie eine effektive Nutzung der durch das Medium *World Wide Web* gegebenen Möglichkeiten.

Eine der Möglichkeiten, die genannten Probleme zu mindern, ist der Einsaz automatisierter Verfahren zur Zusammensetzung und Strukturierung von hypermedialen Dokumenten. Hierdurch wird die für den Autor sichtbare Komplexität der Vernetzung von Kapiteln, Seiten und Medienobjekten größtenteils verborgen, wodurch insbesondere die Erweiterung und Anpassung von Dokumenten durch "einfache" Umstrukturierung vorhandener Bausteine vereinfacht wird. Insbesondere für sehr komplexe, einer kontinuierlichen Entwicklung unterliegende Themen und für stark heterogene Nutzergruppen ist eine

[9] Bei der Erstellung des DIALEKT Lernprogramms "Statistik Interaktiv" [DIALEKT00] hat sich z.B. herausgestellt, daß bei der Erstellung von Texten und Animationen mehr Zeit auf Nachbesserungen verwendet wurde, als auf das erstmalige Erstellen der Medien. Das Programm enthält Texte, die viermal zwischen einem Statistiker und einem Projektmitarbeiter hin- und herwanderten, bis sie von beiden Seiten akzeptiert wurden.

verstärkte Automatisierung des Entwicklungszyklus ein Muß, wenn ein halbwegs tragbares Kosten-Nutzen Verhältnis gewahrt werden soll.

3.2 Der Bottom-Up Ansatz

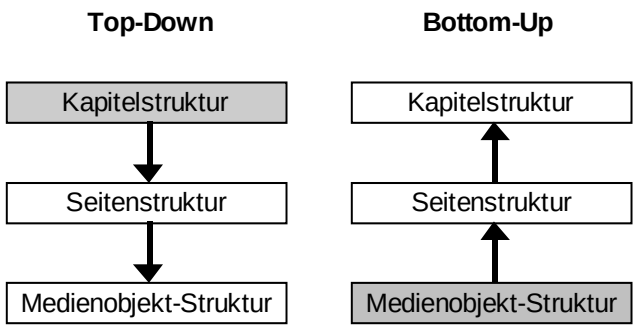
Manuelle Entwicklungszyklen werden als **Top-Down** Zyklen bezeichnet, da die strukturellen Ebenen von "oben nach unten" erstellt werden: zuerst die Kapitelstruktur, anschließend durch Verfeinerung die Seitenstruktur und abschließend durch abermalig Verfeinerung die Medienobjektstruktur [Caumanns98].

Diese Vorgehensweise ist weitestgehend resistent gegen jegliche Versuche der Automatisierung, da inhaltliche und strukturelle Verfeinerungen nur manuell durchführbar sind. Auch wenn bereits alle drei strukturellen Ebenen gegeben sind, ist eine Propagierung von Änderungen und Erweiterungen von oben nach unten aus dem selben Grund nicht möglich. Nach einem *Top-Down* Ansatz erstellte hypermediale Dokumente sind somit prinzipiell schwer wart- und erweiterbar.

Aus diesem Grund wird in dieser Arbeit ein neues Paradigma der Erstellung hypermedialer Dokumente vorgeschlagen, das von seiner Idee her als synthetisch oder *Bottom-Up* charakterisiert werden kann.

Idee des **Bottom-Up** Ansatzes ist es, bei der Erstellung eines hypermedialen Dokuments - wie z.B. eines Lehr- oder Informationssystems - mit der Medienobjektstruktur zu beginnen und daraus Seiten- und Kapitelstruktur abzuleiten (Abbildung 3-2).

Abbildung 3-2: Top-Down und Bottom-Up Ansatz zur Erstellung von Lehr- und Informationssystemen



Der Unterschied zwischen *Top-Down* und *Bottom-Up* Ansätzen läßt sich sehr gut am Beispiel eines foliengestützten Vortrags illustrieren:

> Die übliche Vorgehensweise ist *Top-Down*, d.h. zuerst wird eine Gliederung des Vortrags (Kapitelstruktur) erstellt, anschließend einzelnen Aspekte der Gliederung Folien zugeordnet (Seitenstruktur) und abschließend diese mit Texten und Bildern gefüllt (Medienobjekte).
>
> In der Praxis sind jedoch auch genau umgekehrte *Bottom-Up* ausgerichtete Verfahren der Vortragserarbeitung anzutreffen: Aus einem existierenden Satz von Folien werden diejenigen ausgewählt, die am besten zu Thema und Publikum passen. Diese Folien werden in eine sinnvolle Reihenfolge gebracht (Medienobjekt- und Seitenstruktur), woraus sich implizit die inhaltliche Gliederung des Vortrags ergibt (Kapitelstruktur).

Diese Analogie verdeutlicht die Vorteile des *Bottom-Up* Ansatzes im Vergleich zum *Top-Down* Entwicklungszyklus:

- er spart Zeit und Geld, da vorhandene Medienobjekte (Folien) wiederverwendet werden können,
- er bietet eine sehr gute Unterstützung für Modifikationen und Erweiterungen (Ändern, Entfernen, Hinzufügen von Folien), da diese nicht ein bereits strukturiertes Dokument (Vortrag) betreffen, sondern immer nur die unstrukturierte Basis der vorhandenen Medienobjekte (Folien),
- er ist eine ideale Basis für jegliche Form von Konfiguration und Anpassung, da Thema und Publikum als Parameter in die Auswahl und Strukturierung der Medienobjekte (Folien) einfließen.

Der größte Vorteil des *Bottom-Up* Ansatzes ist jedoch, daß er sehr gut automatisierbar ist. Die einzige Voraussetzung für die Anwendbarkeit dieses Ansatzes ist das Vorhandensein einer ausreichenden Menge von (teilweise redundanten) Medienobjekten.

3.3 Automatisierung des Autorenzyklus

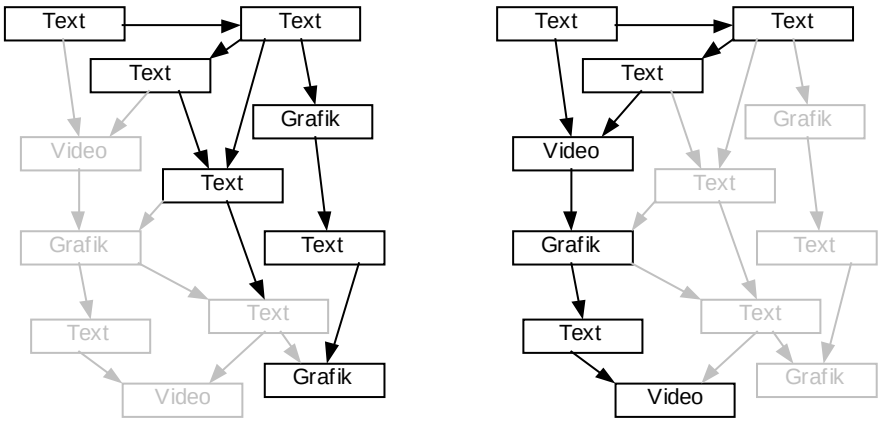
Grundidee der Automatisierung des *Bottom-Up* Ansatzes ist die Möglichkeit, aus einer gegebenen Medienobjektstruktur die übergeordneten Strukturen ableiten zu können:

- Einzelne Medienobjekte können anhand ihres Inhalts und Genres zu Seiten zusammengefaßt werden. Hierbei ist eine automatische Berücksichtigung von Aspekten der Seitengestaltung (Layout, Medienmix, Bildgrößen, Textlängen, etc.) möglich. Auch unkonventionelle Seitenstrukturen wie z.B. das Ebenen-Modell von "Mathe online" (siehe Kapitel 2.4) können durch entsprechende Parametrisierung des Strukturierungsalgorithmus erzeugt werden.
- Einzelne Seiten können auf Basis der Seitenstruktur zu Kapiteln zusammengefaßt werden. Algorithmisch kann ist dies z.B. durch eine Graphtransformation geschehen.

Durch die Möglichkeit der automatischen Ableitung übergeordneter Strukturen ist die Medienobjektstruktur die einzig wichtige Struktur bei der Erstellung eines Lehr- ober Informationssystems. Änderungen der Medienobjektstruktur erfordern keinen manuellen Abgleich mit übergeordneten Strukturen, sondern führen zu deren automatischer Neugenerierung.

Aus einer existierenden Medienobjektstruktur können verschiedene Substrukturen anhand von nutzer- oder autordefinierten Kriterien extrahiert werden, die ihrerseits wiederum Medienobjektstrukturen sind (Abbildung 3-3). Nutzeranpassung und Konfigurierbarkeit sind somit ein weiteres Leistungsmerkmal der *Bottom-Up* Generierung von Lehr- und Informationssystemen.

Abbildung 3-3: Eine Medienobjektstruktur - zwei mögliche Dokumente

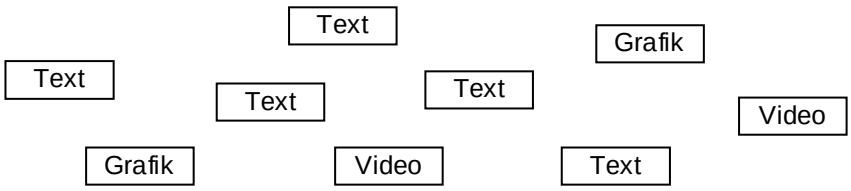


Grundlage für die genannten Möglichkeiten der Automatisierung ist das Vorhandensein der Medienobjektstruktur. Da diese Struktur im allgemeinen sehr komplex und manuell nur schwer wartbar ist, muß auch sie im Rahmen einer *Bottom-Up* ausgerichteten Autorenumgebung automatisch generierbar sein. Ziel ist dabei, daß ein Autor lediglich Medienobjekte erstellt und geeignet beschreibt, so daß die komplette Auswahl und Strukturierung der für den aktuellen Nutzer und seine Lernziele adäquatesten Medienobjekte automatisch durchgeführt werden kann.

Eine Möglichkeit zur Realisierung dieses Ziels ist der im Rahmen dieser Arbeit entwickelte automatisierte Bottom-Up Entwicklungszyklus, der in den Abbildungen 3-4 bis 3-8 leicht vereinfacht dargestellt ist:

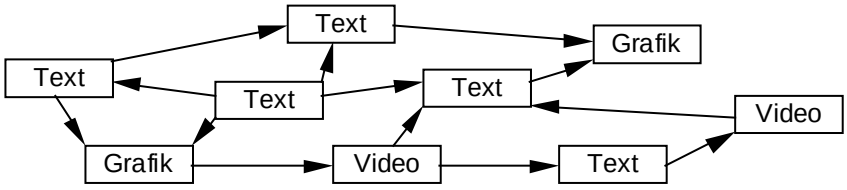
Im ersten Schritt werden die für die Generierung des gewünschten Lehr- bzw. Informationssystems verfügbaren **Medienobjekte** bestimmt. Dies sind in der Regel von Fachautoren erstellte, in Datenbanken abgelegte Texte, Grafiken, Animationen und andere Medien (Abbildung 3-4).

Abbildung 3-4: Für die Generierung eines Lehr- oder Informationssystem zur Verfügung stehende Medienobjekte



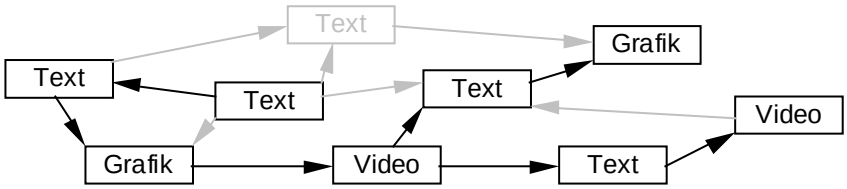
Aus den verfügbaren Medienobjekten wird ein sog. **Abhängigkeitsgraph** erstellt, in dem alle Abhängigkeiten zwischen Medienobjekten über Vorgänger-Nachfolger Beziehungen beschrieben sind (Abbildung 3-5). Durch die Berechnung von diversen Maßzahlen eines jeden Medienobjektes können zusätzliche Informationen über die Struktur des Abhängigkeitsgraphen und die Position einzelner Objekte innerhalb des Graphen gewonnen werden (siehe Kapitel 6).

Abbildung 3-5: Erstellen eines Abhängigkeitsgraphen



Anhand von Heuristiken, die auf den erwähnten Maßzahlen basieren, werden eine Reihe von Medienobjekten ausgewählt. Die ausgewählten Objekte bilden einen **Subgraphen** des Abhängigkeitsgraphen, in dem alle zu vermittelnden Stichworte erlernbar sind. Anschließend wird durch Entfernen bzw. Hinzufügen von einzelnen Medienobjekten und Kanten aus dem Abhängigkeitsgraph die **Medienobjektstruktur** des Lehrsystems berechnet (Abbildung 3-6).

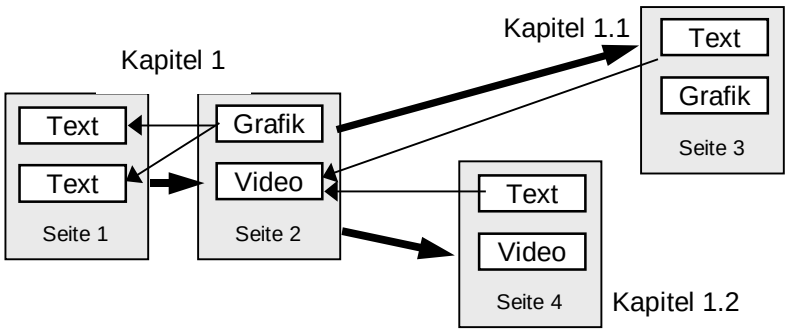
Abbildung 3-6: Auswahl und Strukturierung von Medienobjekten



Die Medienobjektstruktur kann durch Zusammenfassen von benachbarten Knoten in eine **Seitenstruktur** und anschließend durch Zusammenfassen von benachbarten Seiten in eine logische **Kapitelstruktur** überführt werden (Abbildung 3-7).

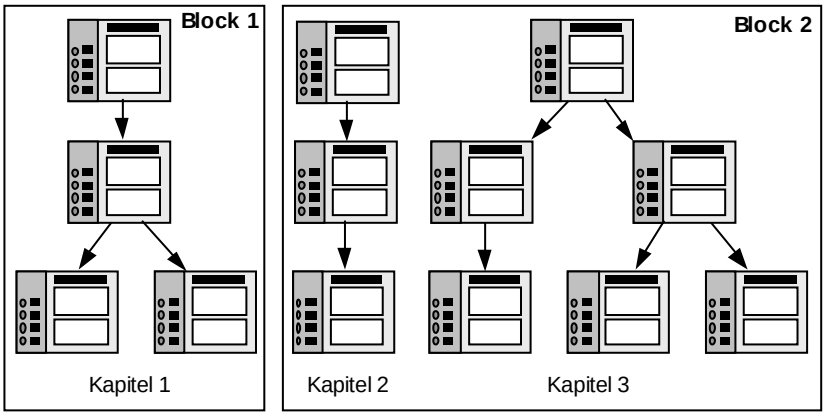
Abbildung 3-7: Erstellen von Seiten- und Kapitelstruktur

(dicke Pfeile = Navigation, dünne Pfeile = Hyperlinks)



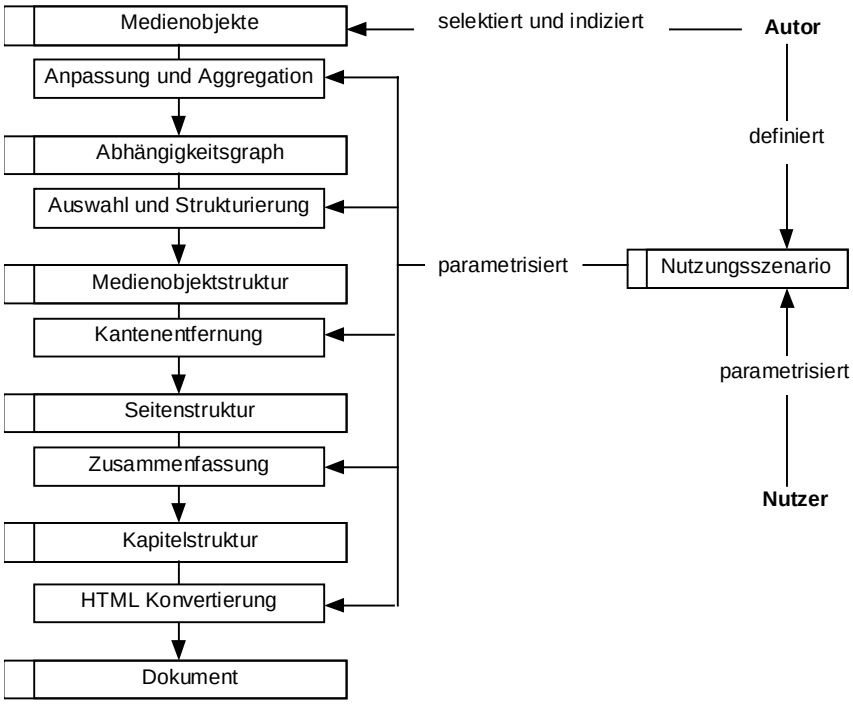
Abschließend werden die Kapitel und Unterkapitel geordnet und die einzelnen Seiten in Form von HTML Dateien auf einem Web-Server abgelegt. Hierbei werden auch alle benötigten Navigationselemente erzeugt und integriert (Abbildung 3-8).

Abbildung 3-8: Zusammenfassung der generierten Strukturen mit Navigationselemente zu HTML-Seiten



Zur Umsetzung dieses automatisierten Autorenzyklus wurde das in Abbildung 3-9 dargestellte **i4 Framework** zur Erstellung hypermedialer Lehr- und Informationssysteme implementiert. Das *i4 Framework* stellt sowohl Datenstrukturen zur Beschreibung von Medienobjekten als auch Algorithmen zu deren Auswahl und Strukturierung bereit.

Abbildung 3-9: Das *i4 Framework* im Überblick



Die im Rahmen der Realisierung des *i4 Frameworks* zu lösenden Probleme konzentrieren sich vor allem auf die Beschreibung, Auswahl und Strukturierung von Medienobjekten. Ausschlaggebend für den Erfolg einer *Bottom-Up* Generierung sind

- die Wahl geeigneter Meta-Daten zur Beschreibung von Medienobjekten, da diese sowohl einfach anzulegen, aber gleichzeitig auch aussagekräftig und für alle denkbaren Anwendungen geeignet sein müssen,
- das Design und die Implementierung eines Algorithmus, der in der Lage ist, anhand dieser Meta-Daten die für den aktuellen Nutzer geeignetsten Medienobjekte auszuwählen und in eine Medienobjektstruktur zu bringen,

- die Zusammenfassung von Medienobjekten zu Seiten, sowie die Zusammenfassung von Seiten zu Kapiteln unter Berücksichtigung von vom Autor festgelegten Vorgaben und
- die Entwicklung eines "intelligenten" *Back-Ends*, das die erzeugten Strukturen in eine Menge von durch Hyperlinks verbundene HTML-Seiten übersetzen kann.

All diese Probleme konnten zufriedenstellend gelöst werden.

3.4 Alternative Ansätze zur Generierung von Dokumenten

Das in dieser Arbeit beschriebene *i4 Framework* ist nur eines von vielen Systemen zur dynamischen Erzeugung web-basierter Lehr- und Informationssysteme. An fast jeder Universität existiert mindestens ein Forschungsprojekt, das versucht, Themen wir Adaptierbarkeit, vernetztes Lernen oder Multimedia mit dem *World Wide Web* in Einklang zu bringen. Doch auch wenn viele Projekte ein ähnliches Ziel verfolgen, so sind die gewählten Ansätze (*Top-Down*, *Bottom-Up*, *Natural Language Generation*, *Link Generation*, etc.) und Schwerpunkte (Didaktik, Medienintegration, Autorenunterstützung, Adaptierbarkeit, Adaptivität, etc.) dennoch sehr verschieden.

In diesem Abschnitt sollen daher drei Systeme vorgestellt werden, die sich bezüglich Schwerpunkt, Idee und Umsetzung nicht unbedingt mit dem *i4 Framework* überschneiden, sondern vielmehr einzelne Aspekte der automatisierten Dokumentenerzeugung grundlegend anders, teilweise vielleicht auch besser, umsetzen als das *i4 Framework*.

Bei den betrachteten Systemen handelt es sich um

- das *Konstanzer Hypertextsystem* als Beispiel für den Einsatz objekt-orientierter Techniken bei der dynamischen Dokumentenerstellung,
- *Interbook* als Beispiel für ein Autorensystem mit Schwerpunkten auf Adaptivität und Autorenunterstützung und
- *PowerTNG* als Beispiel für ein Spracherzeugungssystem, das sowohl *Top-Down* und *Bottom-Up* Verfahren verwendet

Alle drei Systeme werden im folgenden kurz beschrieben und (soweit sinnvoll) mit dem *i4 Framework* verglichen. Der Schwerpunkt der Betrachtung liegt dabei vor allem auf den Aspekten Adaptierbarkeit, Wartbarkeit und Konfigurierbarkeit.

Das Konstanzer Hypertextsystem

Das *Konstanzer Hypertextsystem* (KHS) [Hammwöhner97] ist ein an der Universität Konstanz entwickeltes Werkzeug zur Verwaltung und Präsentation von Hypertexten. Ziel des Systems ist es, Nutzern adaptierbare Pfade durch eine polyhierarchische Struktur von typisierten Hypertext Objekten aus potentiell verschiedenen, heterogenen Datenquellen anzubieten.

Abbildung 3-10: Beispiel für eine vom KHS erzeugte HTML-Seite.

Das KHS wird vom Fachbereich Informationswissenschaften der Universität Konstanz sowie einigen wenigen anderen Fakultäten deutscher Universitäten hauptsächlich zur Verwaltung von Publikationen, Forschungsberichten und Mitarbeiterinformationen eingesetzt. Abbildung 3-10 zeigt eine Seite aus der vom KHS erzeugten Publikationsliste des Fachbereichs Informationswissenschaft der Universität Konstanz.

Grundlage des KHS sind Knoten und Verknüpfungen (Links), die als Hypertext Objekte bezeichnet werden. Jedes Hypertext Objekt hat einen Typ, der Eigenschaften, Inhalt, Darstellung, Funktionalität und Struktur des Objekts festlegt. Alle Objekttypen sind in einer Vererbungshierarchie angeordnet. Neue Objekttypen - z.B. für bestimmte Kodierungen oder Verwendungszwecke - können aus existierenden Typen abgeleitet werden. Hierdurch kann das KHS relativ einfach an neue Anwendungen angepaßt werden.

Knoten repräsentieren die Inhalte des Hypertexts. Eine besondere Klasse von Knoten sind sog. Strukturknoten, die andere Knoten enthalten. Ein Strukturknoten legt Typ und Ordnung der enthaltenen Knoten fest. Da jeder Knoten in mindestens einem Strukturknoten enthalten sein muß und Strukturknoten wiederum Strukturknoten enthalten können, entsteht ein polyhierarchisch strukturierter Hypertext.

Ein Beispiel für einen durch Strukturknoten beschriebenen Hypertext ist die *Homepage* eines Instituts. Ein übergeordneter Strukturknoten für das gesamte Institut kann Strukturknoten für Mitarbeiter und Projekte enthalten. Die Bestandteile jedes einen Mitarbeiter beschreibenden Strukturknotens sind (Struktur)Knoten für Lebenslauf, Forschungsschwerpunkte und Publikationen.

Die Strukturknoten für Projekte enthalten Knoten zur Beschreibung des Projekts und Knoten für alle im Rahmen des Projekts entstandenen Publikationen. Jede Publikation ist somit sowohl in einem Strukturknoten eines Mitarbeiters als auch im Strukturknoten eines Projekts enthalten.

Im Zusammenspiel mit dem zugrundeliegende Objektmodell sind Strukturknoten ein sehr einfacher und flexibler Mechanismus zur Strukturierung von Informationen. Da jeder Strukturknoten einem Objekttyp zugeordnet ist, entstehen einheitlich strukturierte und inhaltlich ähnliche Unterbäume. Um z.B. die Daten eines neuen Mitarbeiters in die Hypertextbasis aufzunehmen, muß nur ein neuer Strukturknoten vom Typ "Mitarbeiter" erstellt und mit Knoten für Lebenslauf, Forschungsschwerpunkte und Publikationen gefüllt werden. Die Ordnung der enthaltenen Knoten sowie deren Darstellung auf einer HTML-Seite sind implizit durch den Typ des Strukturknotens festgelegt.

Durch die Definition von Filtern kann die präsentierte Hyperstruktur an den aktuellen Nutzer angepaßt werden. Die wichtigsten Filter sind Typ-Filter, die nur die Präsentation von bestimmten Typen zugehörigen Hypertext Objekten zulassen und Struktur-Filter, mit denen Unterbäume der Hyperstrukur ein- bzw. ausgeblendet werden können. Darüber hinaus sind durch an Knoten gebundene *Java Skripte* und

in *Cookies* abgelegte Zustandsbeschreibungen (besuchte Seiten, etc.) auch dynamische Anpassungen der Link-Struktur möglich.

Die komfortablen und flexiblen Möglichkeiten zur Nutzeranpassung und Konfiguration des Systems stellen sicherlich die große Stärke des KHS dar. Durch die hierarchische Typisierung von Hypertext Objekten - die sich in ähnlicher Form auch im *i4 Framework* findet (siehe Kapitel 7) - ist eine einfache Anpassung des Systems an verschiedenen Anwendungen und Themengebiete möglich.

All diese Stärken werden jedoch durch die große Schwäche des Systems, die zugrundeliegende polyhierarchische, statische Struktur, relativiert. Die hierarchische Struktur beschränkt den Einsatz des KHS auf die Präsentation eindeutig hierarchisch strukturierbarer Informationen. Auch der Einsatz von Strukturknoten macht nur Sinn, wenn das zu erstellende Informationssystem viele gleichartige Unterbäume enthält.

Dadurch, daß die Struktur der Hypertextbasis darüber hinaus auch statisch ist, wird zusätzlich die Wartbarkeit der Hypertextbasis erschwert. Der Autor des zu erzeugenden Informationssystems muß jeden Knoten einem oder mehreren Strukturknoten zuordnen; eine Aufgabe, die bei genügend großer Hypertextbasis und konzeptuell vernetzt strukturierten Themengebieten (*ill structured domains*) manuell nur mit sehr großem Aufwand zu bewältigen ist.

Diese Nachteile sind sicherlich der Grund dafür, daß das KHS vorwiegend zur Selbstdarstellung von Lehrstühlen im *World Wide Web* genutzt wird, denn hier liegt eine inhärent hierarchische Struktur zugrunde, in die neu hinzuzufügende Hypertexte leicht integriert werden können. Das KHS ist sicherlich nicht dazu geeignet, ein dynamisches, adaptierbares Lehr- oder Informationssystem zur "Europäische Währungsunion" zu erstellen, da dieses Thema nicht nur sehr schwer strukturierbar ist, sondern darüber hinaus auch nur wenig Muster enthält, die mit Strukturknoten beschrieben werden könnten.

Da die mit dem KHS präsentierbaren Themen kaum Spielraum für inhaltlich motivierte Nutzeranpassungen und Konfigurationen erlauben, führen die genannten Nachteile zusätzlich dazu, daß die Stärken des Systems in der Praxis kaum zur Geltung kommen dürften.

Adaptive Hypertext and Hypermedia: Interbook

Das *i4 Framework* ist ein Werkzeug zur dynamischen Erstellung adaptierbarer Dokumente. Eine ähnliches Ziel verfolgen dem Forschungsgebiet des "**Adaptive Hypertext and Hypermedia**" (AH&H) zuzurechnende Projekte. Der Hauptunterschied zwischen beiden Ansätzen ist dabei, daß bei der dynamischen Dokumentenerstellung Struktur und Inhalt an den aktuellen Nutzer angepaßt werden, während die meisten AH&H Systeme den Schwerpunkt auf die dynamische Erzeugung didaktisch sinnvoller Pfade innerhalb eines existierenden Dokuments legen.

Die wenigen Projekte aus dem Bereich des adaptiven Hypertexts, die auch eine Nutzeranpassung der Inhalte und der Zusammenstellung eines Dokuments verfolgen - z.B. C-Book [Kay+94] und 2L670 [Calvi+97] - resultierten zumeist in einzelnen, thematisch gebundene Anwendungen und sind von daher für einen Vergleich mit dem *i4 Framework* uninteressant.

Eines der bekanntesten AH&H Systeme ist *Interbook* [Brusilovsky+98, Eklund+97, Brusilovsky+96b], ein Werkzeug zur Erstellung und Verteilung adaptiver, elektronischer Lehrbücher im *World Wide Web*[10]. *Interbook* ist eine gemeinsame Entwicklung der *Canegie Mellon University* und der Universität Trier und wird zur Zeit u.a. als Basis für einen *Online*-Kurs über John Anderson's ACT-R Theorie eingesetzt.

Interbook wurde im Gegensatz zu den meisten anderen Systemen im Gebiet des AH&H als anwendungsunabhängige Entwicklungsumgebung konzipiert. Hierin unterscheidet es sich auch von seinem Vorgängersystem ELM-ART [Weber+97], einer web-basierten Lehranwendung zur LISP-Programmierung. Die Stärken von *Interbook* liegen vor allem in Portierung existierender Lehrbücher ins *World Wide Web*.

[10] Interbook Version 1.21 (für Macintosh) und verschiedene Publikationen zu Interbook sind im *World Wide Web* unter der Adresse http://www.contrib.andrew.cmu.edu/~plb/InterBook.html verfügbar.

Interbook verfolgt einen wissensbasierten Ansatz, d.h. zur Dokumentenerzeugung werden Informationen über das behandelte Themengebiet (Domain-Modell) und über den aktuellen Nutzer (Nutzermodell) benötigt. Das Domain-Modell, das die Basis für die an dieser Stelle interessierende Strukturierung der Inhalte bildet, besteht aus einem Glossar und einem oder mehreren sog. Textbüchern.

Das Glossar bildet die pädagogische Struktur des behandelten Themengebiets ab, d.h. es stellt die Verbindungen zwischen allen themenspezifischen Konzepten (Stichworten, Themen) dar. Zwischen Konzepten und Glossareinträgen besteht eine 1:1-Beziehung.

Das Textbuch ist die elektronische Version eines hierarchisch strukturierten Textes, dessen Elemente mit den Konzepten des Glossars indiziert sind. Ausgangspunkt der Erstellung des Textbuches ist das darzustellende Lehrbuch im *Word*-Format. Die Struktur des Textes wird dabei durch die Verwendung vordefinierter Formatvorlagen explizit gemacht. Jeder als eine Einheit zu präsentierende Textblock muß zusätzlich sowohl mit den zum Verständnis benötigten als auch mit den darin beschriebenen Konzepten indiziert werden. Diese Form der Inhaltsbeschreibung ist weitgehend identisch mit den vom *i4 Framework* zur Beschreibung der Medienobjekt-Inhalte verwendeten Meta-Daten (siehe Kapitel 4.4).

Der annotierte *Word*-Text wird auf dem Umweg über RTF in HTML übersetzt. Anmerkungen zu Vorwissen und erlernbaren Konzepten bleiben dabei als HTML Kommentare erhalten. Alle so erzeugten HTML-Dateien werden beim Start des *Interbook*-Servers eingelesen und in eine Menge einzelner Seiten übersetzt. Die zu jeder Seite verfügbaren Informationen umfassen dabei neben den referenzierten Konzepten auch die Position der Seite innerhalb der Kapitelstruktur des Originaldokuments.

Mit Hilfe von Glossar, Textbuch und Nutzermodell realisiert *Interbook* verscheidene Adaptionstechniken, die hauptsächlich auf die Unterstützung des Nutzers bei der Navigation abzielen. Einige davon sollen im folgenden anhand eines *Screenshots* aus dem eingangs erwähnten Lehrsystem zur ACT-R Theorie kurz vorgestellt werden.

Abbildung 3-11: Seite eines Interbook-Dokument zur ACT-R Theorie.

Der obere Teil des Bildschirms zeigt einen Ausschnitt aus der Kapitelstruktur des Textbuches, in dem der (hierarchische) Pfad zur angezeigten Seite sowie alle Seiten auf gleicher Ebene dargestellt sind. Im Zentrum des Bildschirms steht eine Seite des Textbuchs, neben der alle als Vorwissen verlangten oder in dem dargestellten Text erlernbaren Konzepte aufgelistet sind. Kapitel-Browser und Konzept-Leiste werden für jede Seite dynamisch erstellt.

Interbook gibt zu jedem *Hyperlink* Hinweise zum Lernzustand des referenzierten Objekts (Textbuchseite oder Glossareintrag). Der Lernzustand wird dynamisch anhand der besuchten Seiten und der darauf erlernten Konzepte bestimmt. Im Kapitel-Browser werden so zum Beispiel *Hyperlinks* zu Seiten, deren verlangtes Vorwissen beim aktuellen Nutzer vorhanden ist, mit einem grünen Punkt markiert. Seiten, die noch nicht betrachtet werden sollten, da der Nutzer nicht über ausreichendes Wissen zu den verlangten

Konzepten verfügt, sind mit einem roten Punkt markiert. Ein weißer Punkt schließlich signalisiert eine Seite, die keinerlei noch unbekannte Konzepte vermittelt.

Der Nutzer weiß somit immer, welche Seiten neue Informationen bieten und welche Seiten überhaupt besucht werden können (*adaptive navigation support* [Brusilovsky+98b]). Darüber hinaus geben die grünen Punkte Hinweise, welche Seiten idealerweise als nächste besucht werden sollten (*direct guidance*).

Über den *Button* "Teach me" kann der Benutzer adaptive Hilfe zur aktuellen Seite abrufen. Hierbei wird anhand des Nutzermodells, der bereits besuchten Seiten und des Inhalts der aktuellen Seite dynamisch eine Liste aller Seiten erstellt, die der Nutzer besuchen sollte, falls er den Inhalt der aktuellen Seite nicht versteht.

Interbook ist im Vergleich zum *i4 Framework* vor allem deshalb interessant, da es sowohl *Top-Down* als auch *Bottom-Up* Komponenten beinhaltet. Das Domain-Modell und die vorgegebene Struktur des Textbuches sind eindeutig *Top-Down* Mechanismen, da sie sich an der Struktur des Wissensgebiets orientieren. Die Art der Indizierung einzelner Seiten hingegen ist eher *Bottom-Up* orientiert, da hier im wesentlichen der Inhalt eines vorhandenen Mediums betrachtet wird. Eine interessante Frage wäre daher, inwieweit sich die adaptive Navigation auch ohne das Domain-Modell nur anhand der Indizierung von Textblöcken durchführen ließe.

Der größte Unterschied zwischen beiden Systemen besteht darin, daß *Interbook* keine Auswahl und Strukturierung vornimmt. Die fest vorgegebene Seiten- und Kapitelstruktur wird nicht verändert, sondern vielmehr dynamisch annotiert. Auch die Auswahl, welche von sich inhaltlich überschneidenden Seiten als nächste besucht wird, obliegt im Endeffekt dem Nutzer.

Da darüber hinaus kein vorgegebenes Lernziel definiert werden kann, helfen Konzepte wie *adaptive navigation support* und *direct guidance* zwar bei der Navigation innerhalb eines Wissensgebiets; sie bieten aber nur wenig Hilfestellung für ein zielgerichtetes Lernen. Hier liegt ein weiterer Unterschied zwischen *Interbook* und dem *i4 Framework*: das *i4 Framework* schränkt das Wissensgebiet ein, damit der Nutzer ein oder mehrere Lernziele möglichst gut erreichen kann, während *Interbook* darauf abzielt, dem Nutzer Hilfsmittel an die Hand zu geben, damit er sich ein Themengebiet selbständig erschließen kann. Welches dieser beiden Konzepte das bessere ist, hängt allein von der Intention des Nutzers ab.

Die große Stärke von *Interbook* ist sicherlich - neben der adaptiven Navigationsunterstützung - seine Einfachheit. Es erscheint sehr schnell und problemlos, mit Hilfe von *Interbook* ein existierendes Lehrbuch für das *World Wide Web* aufzubereiten. Hierbei wäre jedoch zu prüfen, inwieweit real existierende Lehrbücher bezüglich der Einführung neuer Konzepte auch wirklich hierarchisch strukturiert und somit für den Einsatz von *Interbook* geeignet sind. Insbesondere Lehrbücher, die mit einem kurzen Abriß des Inhalts beginnen, sind mit der von *Interbook* verwendeten, ungewichteten Indizierung sicherlich nur schwer zu erfassen[11].

Als einschränkend bezüglich Nutzeranpassung und Flexibilität muß die statisch vorgegebene Medienobjekt-, Seiten- und Kapitelstruktur angesehen werden. Hierdurch ist es nicht möglich, didaktische und/oder themenspezifische Anpassungen vorzunehmen, z.B. ein Thema wie "Statistik" je nach Nutzer entweder beispielorientiert oder formal aufzubereiten. Darüber hinaus verhindert die Notwendigkeit der hierarchischen Strukturierung des vorgegebenen Textes die Umsetzung alternativer, auf verschiedenen Konzepten und unterschiedlichem Vorwissen aufbauender Lernpfade. Ein Thema wie z.B. C-Programmierung sollte sicherlich für einen assembler-kundigen Nutzer anders strukturiert und aufbereitet werden, als für einen Programmier-Neuling. Auch die Erweiterung und Wartung eines mit *Interbook* erstellten Dokuments ist potentiell problematisch: Neue Seiten müssen kompatibel mit dem Domain-Modell sein und manuell in die hierarchische Struktur eingefügt werden.

Natural Language Generation: PowerTNG

Die dritte Architektur zur dynamischen Dokumentenerzeugung, die in diesem Abschnitt näher analysiert werden soll, ist PowerTNG [Green+99, Milosavljevic+98].

---

[11] In diese Kategorie fallen z.B. viele Bücher über Programmierung, u.a. "Klassiker" wie Stroustrup: *The C++ Programming Language* und Kernighan/Ritchie: *The C Programming Language.*

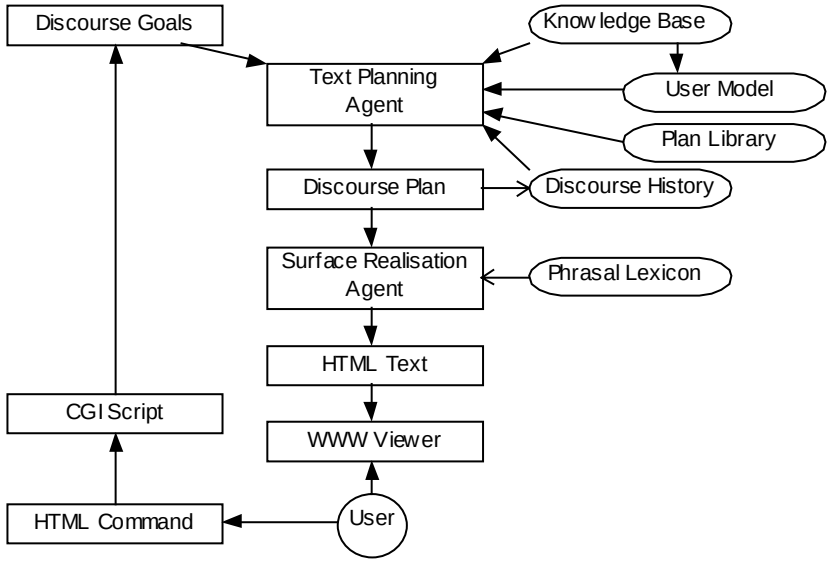
PowerTNG generiert Dokumente durch dynamische Erzeugung von adaptiven, natürlichsprachlichen Texten. Grundlage von PowerTNG ist Peba-II [Milosavljevic+96, Verspoor+98], ein System, das automatisch Beschreibungen von Tierarten erzeugen kann. Die generierten Beschreibungen werden über ein Nutzermodell an das Vorwissen des Nutzers angepaßt, so daß z.B. Vergleiche zu dem Nutzer bekannten Tierarten in die generierten Texte eingebettet werden können.

Abbildung 3-12 zeigt die Architektur von Peba-II und PowerTNG. Ausgangspunkt der Texterzeugung ist ein Lernziel (*discourse goal*), das Inhalt und Verwendung des Textes festlegt. Das Lernziel wird durch eine Nutzeranfrage bestimmt, z.B. den Wunsch nach Erzeugung eines wissenschaftlichen Textes über den Wombat.

Anhand des Lernziels wird eine Lernstrategie (*discourse plan*) aus einer vorgegebenen Bibliothek ausgewählt. Eine Lernstrategie ist eine spezielle Form eines Schemas, mit dem für Gruppen von Lernzielen die zu kodierenden Informationen sowie deren konkrete Einbettung in einen Text beschrieben sind.

Alle das behandelte Themengebiet beschreibenden Informationen sind in einer Wissensbasis abgespeichert. Aufgabe eines *Text Planning Agents* ist es, das ausgewählte Schema mit Fakten aus der Wissensbasis zu füllen. Hierbei wird auch das Profil des aktuellen Nutzers berücksichtigt, wodurch verschiedene Schwierigkeitsgrade und Darstellungsformen realisierbar sind.

Abbildung 3-12: Peba-II System Architektur (aus [Verspoor+98])



Nachdem Fakten und Schema zusammengeführt wurden, beginnt die eigendliche Texterzeugung (*surface realisation*). Das Schema wird mit Hilfe eines Phrasen-Lexikons in eine Folge von natürlichsprachlichen Sätzen übertragen und abschließend durch Einfügen von HTML-Tags formatiert.

Die *Peba-II* Architektur ist ein gutes Beispiel für eine reine *Top-Down* Architektur: Ausgehend von einer vorgegebenen logischen Struktur - dem *discourse plan* - werden Medienobjekte erzeugt und aggregiert.

Peba-II ist jedoch auch ein gutes Beispiel für die Schwächen von *Top-Down* Ansätzen. Wissensbasis und Strategie Bibliothek müssen manuell erstellt und gewartet werden. Insbesondere die feste Verknüpfung von Fakten in der Wissensbasis führt zu einer Komplexität, die auch für eng umgrenzte Themengebiete nur mit großem zeitlichen Aufwand und für große Wissensbasen gar nicht zu bewältigen ist.

Aus diesem Grund wurde für die Erstellung eines virtuellen Museums basierend auf *Peba-II* ein neues System, *PowerTNG*, entwickelt. Ziel des Projekts war es, für ca. 15000 Ausstellungsstücke eines Museums aus einer existierenden Exponat-Datenbank textuelle Beschreibungen zu generieren und im *World Wide Web* zugänglich zu machen [Milosavljevic+98, Green+99].

Die von *PowerTNG* dazu eingesetzte Strategie basiert auf einer Analyse der vorhandenen Datenbank und einer anschließenden Synthese der Beschreibungen nach der oben beschriebenen *Peba-II* Architektur. Bei der Analyse werden mit Hilfe eines Thesaurus und verschiedener *Data Mining* Techniken die vorhandenen Exponatbeschreibungen kategorisiert und strukturiert. Die Ergebnisse der Analyse ist nicht wie im *i4*

Framework ein Graph von Texten (Medienobjekten), sondern eine hierarchische Wissensbasis, an die normalisierte Kurzbeschreibungen der Exponate geknüpft sind. *Peba-II* setzt somit eine auf der Ebenen der Medienobjekte (*Bottom-Up*) angesiedelte Analyse ein, um die Grundlage für eine anschließende *Top-Down* Generierung zu schaffen.

Interessant für das *i4 Framework* ist *PowerTNG* vor allem deshalb, da dieses System mit noch weniger manuell vorgegebenen Informationen (Thesaurus und Struktur des Exponat-Datenbank) als das *i4 Framework* arbeitet. Die generierten Texte sind zwar nicht so gut wie vergleichbare, über eine manuell kodierte Wissensbasis erzeugte Beschreibungen, aber sie sind akzeptabel [Milosavljevic+98].

Darüber hinaus zeigt *PowerTNG*, daß große Informationssysteme im *World Wide Web* mit vertretbarem Aufwand nur realisierbar sind, wenn (zumindest teilweise) auf *Information Retrieval* und *Data Mining* Techniken basierende Verfahren zum Einsatz kommen.

4 Medienobjekte

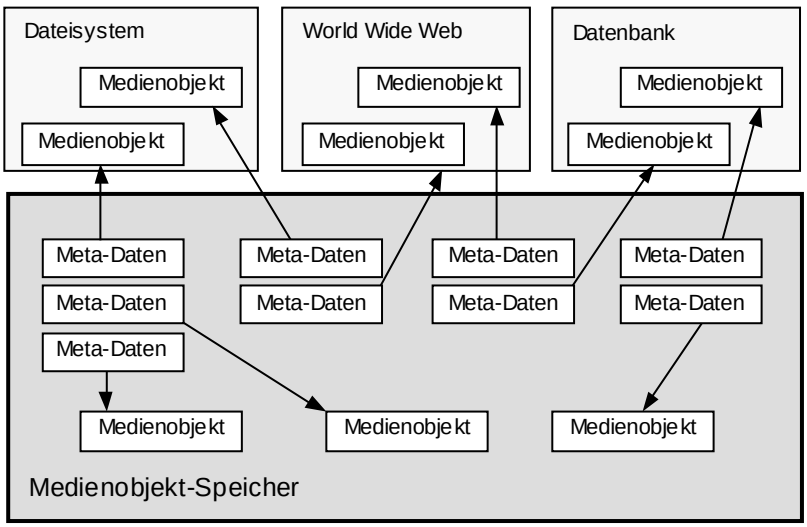
Oder nehmen wir eine Sequenz von Bildern auf einem Fernsehschirm, die eine lachende Shirley MacLaine zeigt. Wenn wir uns diese Sequenz anschauen, wissen wir, daß wir nicht wirklich eine Frau betrachten, sondern Mengen von flackernden Punkten auf einer ebenen Fläche. [...] Welche [Repräsentation] ist "wirklicher"? Es hängt davon ab, ob Sie ein Mensch, ein Hund, ein Computer oder ein Fernsehgerät sind.

Douglas R. Hofstadter, 1995

Medienobjekte sind die Grundbausteine jedes nach dem *Bottom-Up* Ansatz dynamisch generierten Dokuments. Die Verwaltung aller zur Verfügung stehenden Medienobjekte ist Aufgabe des **Medienobjektspeichers**. Die im Medienobjektspeicher abgelegten Informationen bestehen im Wesentlichen aus den **Meta-Daten** aller verfügbaren Medienobjekte. Hierzu zählen insbesondere Angaben zu Format, Verwendungsmöglichkeiten und Inhalt eines jeden Medienobjektes.

Medienobjekte selbst können intern im Medienobjektspeicher oder extern im Dateisystem, im Internet oder in einer Datenbank abgelegt sein. Sämtliche Meta-Daten hingegen müssen im Medienobjektspeicher verfügbar sein (Abbildung 4-1).

Abbildung 4-1: Logischer Aufbau des Medienobjektspeichers



Aufgabe des Autors eines dynamischen Dokuments ist es, Medienobjekte zu erstellen (bzw. als Fragmente existierender Dokumente zu benennen) und die benötigten Meta-Daten im Medienobjektspeicher abzulegen. Alles weitere – insbesondere Auswahl und Strukturierung der Medienobjekte zu einem Dokument - geschieht automatisch.

Die Meta-Daten zur Beschreibung der Medienobjekte müssen zwei - sich teilweise widersprechenden - Anforderungen genügen:

- **Mächtigkeit**: Sie sind Basis für die automatisierte Auswahl und Strukturierung von Medienobjekten. Aus diesem Grunde sollten sie eine möglichst flexible, komplexe und umfassende Beschreibung von Medienobjekten ermöglichen.
- **Einfachheit**: Sie bilden die Autorenschnittstelle des *i4 Frameworks*. Daher sollten sie möglichst einfach und offen gestaltet sein, um eine schnelle manuelle Erstellung und den unterstützenden Einsatz automatischer Verfahren zu ermöglichen.

Der Widerspruch zwischen Mächtigkeit der Beschreibungsfähigkeit auf der einen und Einfachheit auf der anderen Seite ist nicht einfach aufzulösen. Im Grunde genommen ist er eine logische Konsequenz der Idee des *Bottom-Up* Ansatzes: Die automatisierte Auswahl und Strukturierung von Medienobjekten erfordert Informationen über eben diese Medienobjekte, die von existierenden IR-Systemen gar nicht und auch von "menschlichen" Autoren nur mit großem Zeitaufwand erstellt werden können. Eines der Hauptanliegen dieser Arbeit ist es daher, zu ermitteln, wie einfach eine Beschreibung von Medienobjekten gehalten sein kann, um darauf basierend noch eine automatisierte Auswahl und Strukturierung durchführen zu können.

Nur wenn die Beschreibung der Medienobjekte über relativ rudimentäre Informationen möglich ist, macht die *Bottom-Up* Generierung von Lehr- und Informationssystemen einen Sinn. Bedenkt man, daß die Medienobjekt-Basis zur Erstellung eines Lehr- oder Informationssystems aus hunderten von Medienobjekten bestehen kann, so macht es schon einen Unterschied, ob ein Autor 5 oder 50 Minuten zur manuellen Beschreibung eines einzelnen Medienobjektes benötigt. Darüber hinaus sind die Möglichkeiten der automatischen Gewinnung oder zumindest der "intelligenten" Autorenunterstützung desto größer, je einfacher die zur Beschreibung von Medienobjekten benötigten Meta-Informationen sind.

Im *i4 Framework* wurde versucht, einen Kompromiß zwischen Einfachheit und Aussagekraft zu finden: Die Beschreibung der Medienobjekte ist sehr einfach und offen für den Einsatz von *Information Retrieval* Techniken. Dies ist eine Entscheidung zugunsten des Autors. Aufbauend auf diesen Beschreibungen wird eine komplexe Graphstruktur - der Abhängigkeitsgraph - erstellt. Durch die Verknüpfung zu einem Graph können mit Hilfe teilweise recht aufwendiger Algorithmen zusätzliche Informationen über Medienobjekte gewonnen werden, die eine automatisierte Auswahl und Strukturierung erst möglich machen.

Der Nachteile, die für diesen Kompromiß aus einfacher Beschreibung und komplexen Algorithmen in Kauf genommen werden müssen, sind:

- ein gewisser Verlust an semantischer Korrektheit der erzeugten Medienobjektstruktur,
- erhöhter Rechenaufwand und dadurch bedingt eine relativ langsame dynamische Dokumenten-erstellung.

Da jedoch die Möglichkeiten des *Information Retrieval* auch in Bezug auf semantische Aspekte immer besser und Rechner immer schneller werden, ist abzusehen, daß diese Nachteile in naher Zukunft irrelevant sein werden. Bereits zum jetzigen Zeitpunkt fallen sie in Anbetracht der in Kapitel 2 genannten Vorteile kaum ins Gewicht.

In diesem Kapitel werden verschiedene Aspekte von Medienobjekten im Detail diskutiert. Im Mittelpunkt steht dabei die Definition von Medienobjekten und deren Beschreibung durch den Autor.

4.1 Definition von Medienobjekten

Medienobjekte sind die Grundbausteine der *Bottom-Up* Generierung von Lehr- und Informationssystemen. Jede textuelle, grafische oder sonstige Kodierung von für das behandelte Wissensgebiet relevantem, faktischem Wissen ist ein Medienobjekt. Medienobjekte können Textabschnitte, Fotos, Grafiken, digitales Audio oder Video, aber auch interaktive Elemente wie z.B. *Java Applets* sein.

Die wichtigsten Aspekte bei dieser Definition von Medienobjekten ist der Kontextbezug. Medienobjekte vermitteln immer faktisches Wissen und stehen immer im Kontext eines Wissensgebietes. Für den Kontext "Fußball" wäre z.B. ein Foto von Franz Beckenbauer ein Medienobjekt (vermitteltes Wissen: Aussehen von Franz Beckenbauer), für die meisten anderen Themengebiete nicht.

Diese beiden Grundbedingungen sind sowohl für ein automatisches *Retrieval* von Medienobjekten (siehe Kapitel 9.3), als auch für deren automatische Auswahl und Strukturierung von Bedeutung:

- Durch das Vorhandensein faktischen Wissens und die Bindung an einen Kontext können zur Unterstützung des Autors bei der Medienobjekt-Beschreibung Thesauri und konzeptuelle Graphen eines Wissensgebietes eingesetzt werden. Hierdurch wird die Erkennung und Zuordnung von Medienobjekten sowie der Vergleich von Medienobjekten vereinfacht.
- Das *i4 Framework* benötigt zur Auswahl und Strukturierung von Medienobjekten keine explizit vorgegebenen Informationen über die konzeptuelle Struktur des behandelten Wissensgebiets. Dadurch, daß alles in einem Medienobjekt kodierte Wissen zu einem bestimmten Grad relevant ist und sich an Stichworte binden läßt, können Teile einer konzeptuellen Struktur konstruiert werden. Dies ist insbesondere für die Strukturierung von Medienobjekten von Bedeutung

Die Menge der für die dynamische Erstellung eines Lehr- bzw. Informationssystems zur Verfügung stehenden Medienobjekte ist nicht leer und endlich, wobei davon ausgegangen wird, daß keinerlei (relevante) Informationen existieren, die nicht in einem dieser Medienobjekte enthalten ist (*closed world assumption*).

Die Menge der zur Verfügung stehenden **Medienobjekte** ist als Indexmenge M^{avail} definiert:

$$M^{avail} := \{ m \in \mathbb{N}_0 \mid m \leq M_{max} \} \quad (4.1)$$

M_{max} gibt die Anzahl der zur Verfügung stehenden Medienobjekte an. Ein Medienobjekt steht dabei für die Dokumentenerzeugung zur Verfügung, wenn zumindest seine Metadaten im Medienobjektspeicher abgelegt sind.

4.2 Beschreibung von Medienobjekten

Eine der großen Stärken des *Bottom-Up* Ansatzes ist, daß nur lokale Informationen über Medienobjekte benötigt werden. Jedes Medienobjekt "kennt" nur sich selbst und "weiß" nichts von der Existenz weiterer Medienobjekte ("globale Ignoranz").

Dadurch, daß die Beschreibung eines Medienobjektes nur auf lokalen, von diesem Objekt abhängigen Daten beruht, können Medienobjekte sehr leicht zum Medienobjektspeicher hinzugefügt, entfernt, verändert oder ausgetauscht werden. In gewissem Sinne ist dies eine Art "*plug and play*" Mechanismus: Ein neues Medienobjekt wird samt seiner Beschreibung zum Medienobjektspeicher hinzugefügt und von diesem Moment an implizit bei der Dokumentenerstellung berücksichtigt.

Die von den in dieser Arbeit beschriebenen Algorithmen benötigten Informationen über ein Medienobjekt lassen sich zu folgenden Kategorien zusammenfassen:

- Informationen, wie ein Medienobjekt in den Medienobjektspeicher gelangt ist und wie es dort **verwaltet** wird,
- Informationen über die **Kodierung** eines Medienobjekts,
- Informationen über die **Qualität des Inhalts** eines Medienobjekts, z.B. seine Schwierigkeit,
- **Rechtliche Aspekte**, z.B. das Urheberrecht an einem Medienobjekt,
- Informationen über die **Einsatzmöglichkeiten** eines Medienobjekts,
- **Nicht-Generische Informationen**, die von dem behandelten Wissensgebiet und der mit dem *i4 Framework* erstellten Anwendung abhängig sind,
- Beschreibung des **Inhalts** eines Medienobjekts.

Um die Beschreibung der Medienobjekte möglichst einfach und erweiterbar zu gestalten, werden Eigenschaften der ersten sechs Kategorien durch einfache Attribute repräsentiert. Der Inhalt eines Medienobjekts wird über einen Index aus benötigtem und vermitteltem Wissen beschrieben.

Diese Zweiteilung der Beschreibung durch Attribute und Index wird leider von existierenden Metadaten-Standards für Lernobjekte nicht unterstützt. Existierende Metadaten-Spezifikationen wie z.B. ***Learning Object Metadata*** (LOM) vom *IEEE Learning Technology Standards Committee* [LTSC98] definieren fast ausschließlich Attribute der ersten vier Kategorien, weshalb sie für die Zielstellung dieser Arbeit nur von geringer Relevanz wären und daher nicht berücksichtigt wurden.

4.3 Attribute

Die meisten Eigenschaften eines Medienobjekts werden in Form von Attributen kodiert:

> **Attribute** sind semantische Tripel, die aus einem zugeordneten Objekt, einem Namen und einem Wert bestehen. Das Objekt gibt an, welches Objekt durch das Attribut beschrieben ist. Der Name eines Attributs ist eine Zeichenkette, über den ein Attribut eindeutig identifiziert werden kann. Der Attributwert kann - je nach Wertebereich des Attributs - eine Zeichenkette oder ein numerischer Wert sein.

Beispiel: Das Tupel zur Beschreibung des Mime-Typs eines HTML-kodierten Medienobjekts m ist z.B. (m, "MimeType", "text/html").

Die Menge aller auf Medienobjekten definierten Attribute wird im folgenden als A^M bezeichnet:

$$A^M \subseteq M^{avail} \times attr\_name \times attr\_value \tag{4.2}$$

wobei *attr_name* eine Menge beliebiger Zeichenketten bezeichnet.

Auf der Menge aller **Medienobjekt-Attribute** A^M kann eine Abbildung μ_{attr} definiert werden, die zu jedem Medienobjekt alle diesem Objekt zugeordneten Name-Wert-Paare bestimmt:

$$\mu_{attr}: M^{avail} \rightarrow \mathsf{P}(attr\_name \times attr\_value) \tag{4.3}$$

$$\mu_{attr}(m) := \{ (n, v) \mid (m, n, v) \in A^M \}$$

Aufbauend darauf kann eine weitere Abbildung μ_{val} festgelegt werden, die zu einem gegebenen Medienobjekt und einem ebenfalls gegebenen Attributnamen den zugehörigen Wert liefert. Zusätzlich erhält μ_{val} als drittes Argument einen Default-Wert:

$$\mu_{val}: M^{avail} \times attr\_name \times attr\_value \rightarrow attr\_value \tag{4.4}$$

$$\mu_{val}(m, n, d) := \begin{cases} v, & \text{falls } \exists (m, n, v) \in A^M \\ d, & \text{sonst} \end{cases}$$

Für das *i4 Framework* wurden eine Reihe von Attributen mit Namen und Wertebereichen definiert. Die meisten dieser Attribute sind optional, d.h. sie müssen vom Autor nicht für jedes Medienobjekt angegeben werden.

Die nachfolgende Tabelle listet alle in der aktuellen Implementierung definierten Attribute auf:

Attribut	**Beschreibung**	**Wertebereich**
URL	Quelle des Medienobjekts	String (gültiger URL)
UpdatePolicy	Angabe, ob und wie Medienobjekte gecacht werden können	geschlossenes Vokabular: - beliebig speicherbar - Speicherzeitraum in Sekunden - nie cachen

ValidUntil	Ende der Gültigkeit des Medienobjekts	Datum (Sekunden seit 1.1.1970), bzw. 0 für Medienobjekte mit unbegrenzter Gültigkeit
MimeType	Kodierung des Medienobjekts	Kombination aus Medientyp und Kodierung (siehe [RFC2046])
Mediasize	Gräße des Medienobjekts in Bytes	Integer
Bandwidth	ninimal erforderliche Bandbreite in Kilobit pro Sekunde (nur für kontinuierliche Medien)	Integer
Mediawidth Mediaheight	Darstellungsgröße (nur für Grafiken, Videos und Animationen)	Integer Integer
Quality	Qualität der Kodierung (Farbtiefe, Sprachqualität, Samplingrate, etc)	[0,1]
Language	Sprache des Medienobjekts	Zeichenkette nach ISO-639[12]
Thumbnail	Statischer Repräsentant für kontinuierliche Medien	URL oder Medienobjekt
AccessProtection	Erlaubte Zugriffs- und Distributionsoperationen	offenes Vokabular: - uneingeschränkt verwendbar - verwendbar nur mit Angabe des Autors bzw. Rechteinhabers - verwendbar für bestimmte Nutzer - nur zu Test- und Demozwecken verwendbar
Genre	Genre des Medienobjekts	offnes Vokabular, siehe 2.2.3
Popup	Kann das Medieniobjekt auch ohne spezifischen Kontext in einem Glossar oder als Popup angezeigt werden?	Boolean
Difficulty	Schwierigkeitsgrad	[0, 1]
Caption	Überschrift (bzw. Unterschrift bei Grafiken)	String

Für einige der auf dem Medienobjektspeicher operierenden Algorithmen sowie für das verwendete Datenbank-Schema (siehe Kapitel 8) wird eine Möglichkeit der eindeutigen Referenzierung von Medienobjekten benötigt. Aus diesem Grund besitzt jedes Medienobjekte eine eindeutige numerische ID, die vom Medienobjektspeicher vergeben und verwaltet wird[13]. Für den Autor ist die ID über ein gleichnamiges Attribut sichtbar.

Da die numerische ID bei der interaktiven Verwaltung von Medienobjekten für den menschlichen Benutzer nicht sehr sprechend ist, kann ein Medienobjekt zusätzlich mit einem Namen versehen werden. Der Name eines Medienobjektes ist eine beliebig lange - im Attribut "Name" abgelegte - Zeichenkette, die eindeutig sein sollte[14].

[12] Siehe http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt für eine Liste aller in ISO-639 definierten Sprachcodes.

[13] In der aktuellen Implementierung entspricht die ID eines Medienobjektes seinem Index in M^{avail}.

[14] Da intern immer nur mit der ID gearbeitet wird, ist der Name eines Medienobjekts nur eine Hilfe für den Autor bei der Identifikation von Medienobjekten z.B. um sie zu modifizieren oder zu löschen.

Sämtliche der oben angegebenen Attribute zur Beschreibung von Medienobjekten sind anwendungsunabhängig; sie werden zur Erstellung eines Dokuments über die Europäische Währungsunion genauso interpretiert und verwendet wie für ein Lehrsystem zur Deskriptiven Statistik.

Jedes Thema hat jedoch seine Besonderheiten, die über einen vorgegebenen, generischen Satz von Attributen nicht berücksichtigt werden können. Diese Besonderheiten erweisen sich oftmals als entscheidende Faktoren sowohl bezüglich der Auswahl als auch der Strukturierung von Medienobjekten.

Für ein Informationssystem über Wirtschaftstheorien wird so z.B. die subjektive Bewertung einzelner Theorien durch verschiedene Fachleute den Inhalt vieler Medienobjekte darstellen. Um bei der Strukturierung konträre Argumente und Meinungen gegenüberstellen zu können, muß der Strukturierungsalgorithmus jedoch "wissen", ob eine Theorie in einem bestimmten Medienobjekt eher positiv oder negativ dargestellt wird.

Dieses Kriterium ist allerdings bei einem Lehrsystem zur Deskriptiven Statistik nicht sonderlich sinnvoll. Hier wäre es z.B. wichtiger, Medienobjekte bezüglich der Verwendung einer bestimmten Definitionen der empirischen Varianz zu kennzeichnen. Diese Information könnte anschließend bei der Auswahl der Medienobjekte genutzt werden, um dem Lernenden nur Medienobjekte zu liefern, die mit den von ihm verwendeten Lehrbüchern kompatibel sind.

Aus diesem Grund besteht für den Anbieter eines dynamischen Dokuments die Möglichkeit, die im Medienobjektspeicher befindlichen Metadaten der Medienobjekte um eigene Attribute zu erweitern. Sowohl der Name eines solchen **anwendungsabhängigen Attributs** als auch sein Wertebereich können frei festgelegt werden.

Für das Beispiel der Wirtschaftstheorien wäre so z.B. ein zusätzliches Attribut "Bewertung" sinnvoll, während für die Deskriptive Statistik z.B. das zusätzliche Attribut "Varianz" mit den Werten "1/n" und "1/(n-1)" hilfreich ist.

4.4 Beschreibung des Inhalts

Aufgabe der mit einem Medienobjekt verknüpften Metadaten ist es, darauf basierend die automatische Erstellung der Medienobjektstruktur eines Lehr- oder Informationssystems zu ermöglichen.

Um eine automatische Strukturierung vornehmen zu können, reichen jedoch die bisher vorgestellten, vorwiegend formalen Attribute bei weitem nicht aus. Sie erlauben z.B. keine Aussage darüber, in welchem Verhältnis zwei Medienobjekte zueinander stehen: Ist die Kenntnis des einen Objekts Voraussetzung, um das andere zu verstehen? Vermitteln beide Medienobjekte die gleichen Informationen oder nicht? Liegen die beiden Objekte in der Medienobjektstruktur potentiell nahe beieinander oder sind sie weit voneinander entfernt?

All diese für die automatische Strukturierung wichtigen Fragen lassen sich nur beantworten, wenn neben der Form auch der **Inhalt** eines jeden Medienobjekts zumindest teilweise bekannt ist.

Die Art und Weise, wie der Inhalt von Medienobjekten beschrieben wird, ist einer der wichtigsten Aspekte eines jeden Systems zur dynamischen Erstellung von Lehr- und Informationssystemen:

- Ist die Beschreibung des Inhalts zu komplex, so ist auch in naher Zukunft keine automatisierte Unterstützung des Autors möglich. Solange keine automatisierte Unterstützung gegeben ist, muß die Inhaltsbeschreibung komplett manuell vorgenommen werden. Kann dieses nicht schnell und einfach geschehen, ist das darauf aufbauende System für den praktischen Einsatz nicht geeignet. Beispiele für komplexe, nicht automatisch erzeugbare inhaltliche Beschreibungen sind konzeptuelle und semantische Netze.
- Ist die Beschreibung des Inhalts zu einfach, kann sie zwar leicht ermittelt oder erstellt werden, aber sie bildet keine solide Grundlage für eine automatische Auswahl und Strukturierung von Medienobjekten. Dies bedeutet, daß die generierten Dokumente schlecht strukturiert und für Lehr- und Informationszwecke nur bedingt einsetzbar sind. Ein Beispiel für eine generierbare aber nicht

ausreichend aussagekräftige inhaltliche Beschreibung ist der Volltext-Index eines (textuellen) Medienobjekts.

Aus diesen Ausführungen folgt, daß eine Inhaltsbeschreibung - wie auch die komplette Medienobjekt-Beschreibung - immer ein Kompromiß zwischen Einfachheit auf der einen und Aussagekraft auf der anderen Seite ist. Je besser der Kompromiß, desto besser und praktikabler ist das darauf aufgebaute System zur automatisierten Generierung von Lehr- und Informationssystemen.

Im folgenden werden die grundlegenden Konzepte einiger in der kognitiven Psychologie und der künstlichen Intelligenz gebräuchlichen Repräsentationen zur Beschreibung von Semantik und Inhalt vorgestellt. Aus den Schwächen dieser Methoden bezüglich der genannten Kriterien wird anschließend eine einfache, auf Indizes basierende Inhaltsrepräsentation abgeleitet.

4.4.1 Propositionale und Konzeptuelle Netze

In der kognitiven Psychologie wird häufig zwischen explizitem, deklarativen Wissen (z.B. wie ein Auto funktioniert) und implizitem, prozeduralem Wissen (z.B. wie man ein Auto fährt) unterschieden [Anderson93].

Deklaratives Wissen basiert auf Konzepten und Fakten. Konzepte werden dabei als die Abbildung von Worten auf Bedeutungen gesehen und repräsentieren quasi das Vokabular des deklarativen Wissens[15]. Fakten stellen die einfachsten Sätze auf diesem Vokabular dar, über die eine Aussage bezüglich ihres Wahrheitsgehaltes gemacht werden kann.

Prozedurales Wissen hingegen ist oftmals heuristisch, individuell und kontextabhängig. Eine formale Beschreibung ist nur mit regelbasierten Produktionssystemen möglich, mit denen versucht wird, die beim Lösen eines Problems ablaufenden kognitiven Prozesse zu modellieren. Produktionssysteme sind sehr aufwendig zu erstellen und zielen eher auf automatisiertes Problemlösen als auf die Vermittlung von Wissen ab. Aus diesen Gründen sind für die Beschreibung von Medienobjekten nur auf Konzepten basierende Beschreibungen deklarativen Wissens von Interesse.

Sowohl in der künstlichen Intelligenz, als auch in der kognitiven Psychologie sind eine Vielzahl von Repräsentationen deklarativen Wissens entwickelt worden, z.B. die *Prädikatenlogik* [Nilsson98], *Schemas* [Minsky75], *Konzeptuelle Netze* [Sowa84], *Propositionale (oder strukturelle) Graphen* [Anderson+73; Kintsch74; Norman+75], und *Semantische Netze* [Quillian66]. All diese Darstellungsformen versuchen, ein Modell eines Wissensgebietes zu erstellen, indem sie die Beziehungen zwischen Konzepten durch Formeln oder Graphen explizit machen.

Obwohl fast alle der oben aufgezählten Wissensrepräsentationen ursprünglich entwickelt wurden, um aus gegebenen Fakten logische Schlüsse zu ziehen (z.B. über den Wahrheitsgehalt von Aussagen) oder um automatische Textanalysen durchzuführen, eignen sich vor allem **konzeptuelle Netze** und **propositionale Graphen** auch sehr gut für die inhaltliche Beschreibung von Texten, Grafiken, Videos, etc.

Diese aus der KI oder der kognitiven Psychologie entstandenen Wissensrepräsentationen beschreiben neben dem Inhalt eines Medienobjektes auch und vor allem die Struktur des behandelten Wissensgebietes. Diese Struktur wird bei einem *Bottom-Up* Ansatz jedoch nicht unbedingt benötigt, so daß ihre Beschreibung einen unnötigen Mehraufwand darstellt.

Dieser Mehraufwand müßte komplett vom Autor eines Medienobjekts abgefangen werden, da die semantische Beschreibung eines Medienobjektes nur ansatzweise automatisch ermittelt werden kann. Auch das Fernziel der völlig automatischen Erstellung der benötigten Meta-Informationen wäre bei Verwendung semantischer Strukturen kaum zu erreichen [Martin+99]. Aus diesen Gründen wurde für das *i4 Framework* eine einfachere, dem aktuellen Stand des *Information Retrieval* nähere Inhaltsbeschreibung von Medienobjekten gewählt.

---

[15] Ein Konzept stellt die "Intensionale Repräsentationen einer Menge von Objekten mit ähnlichen Eigenschaften" [Schmid+96] dar..

4.4.2 Indizes

Da es bei der dynamischen Erstellung von Lehr- und Informationssystemen nicht um logisches Schließen oder Textanalysen geht, stellt der Verzicht auf semantische Informationen kein allzu großes Problem dar. Das Hauptziel bei der inhaltlichen Beschreibung von Medienobjekten ist es weniger, die Semantik des Objektes als vielmehr seine Verwendungsmöglichkeiten zu erfassen. Der Strukturierungsalgorithmus muß die Medienobjekte nicht "verstehen" können; er muß lediglich in der Lage sein, auf ihnen existierende Abhängigkeiten zu erkennen.

Die Beschreibung eines Medienobjektes unter Vernachlässigung seiner Bedeutung ist vergleichbar mit der Beschreibung eines Buches durch seinen Index. Ein Index listet zwar alle benutzten Stichworte und Themen auf, sagt aber ansonsten nichts über deren Semantik und Verhältnis untereinander aus.

In Anlehnung an dieses Konzept basiert auch die für das *i4 Framework* gewählte Inhaltsbeschreibung von Medienobjekten auf **Indizes**, d.h. sie besteht aus einer Auflistung von Stichworten und Themenbezeichnungen. Im Unterschied zu Indizes von Büchern werden hierbei jedoch nicht nur die neu eingeführten Stichpunkte aufgeführt, sondern alle in dem Medienobjekt erwähnten Stichworte und Themen.

Analog zu M^{avail} werden sämtliche, von allen verfügbaren Medienobjekten referenzierten, Stichworte als Indexmenge S^{avail} definiert, wobei S_{max} die Anzahl der insgesamt referenzierten Stichworte angibt:

$$S^{avail} := \{ s \in IN_0 \mid s \leq S_{max} \} \tag{4.5}$$

Indizes können aus Gründen der Übersichtlichkeit **hierarchisch** aufgebaut werden, d.h. zu jedem Indexeintrag können beliebig viele Untereinträge gehören, die ihrerseits auch wieder Untereinträge besitzen können. Als Beispiel für die Beschreibung eines Medienobjektes durch einen Index sei folgendes textuelles Medienobjekt - ein aus dem Zusammenhang gerissener Absatz aus einem Lehrbuch zur Volkswirtschaftslehre[16] - gegeben:

> *Die Deutsche Bundesbank kauft auch an sich rediskontfähige Wechsel nicht in unbegrenztem Umfang an, sondern setzt bestimmte Obergrenzen, sog. Rediskontkontingente, fest. Der Gesamtumfang der Rediskontkontingente betrug Ende 1990 84,6 Mrd. DM. Dieses Gesamtkontingent wird nach bestimmten Kriterien ("haftendes Eigenkapital") auf die einzelnen Geschäftsbanken aufgeteilt und den einzelnen Geschäftsbanken mitgeteilt.*

Auch ohne diesen Text zu verstehen, können folgende Angaben über die verwendeten Stichworte und Themen gemacht werden:

- Der Leser erhält Informationen über "Aufteilung" und "Umfang" von "Rediskontkontingenten" und den "Ankauf rediskontfähiger Wechsel".
- Für einen Leser ohne Kenntnis der Stichworte "Deutsche Bundesbank", "rediskontfähiger Wechsel" und "Geschäftsbank" ist der Text unverständlich. Kenntnisse über "haftendes Eigenkapitel" sind nicht zwingend notwendig, aber sicherlich hilfreich.

Aus diesen leicht aus dem Text zu extrahierenden Informationen läßt sich sehr einfach ein Index des Medienobjektes erstellen:

- Deutsche Bundesbank
- Geschäftsbank
- haftendes Eigenkapital
- rediskontfähiger Wechsel
  - Ankauf
- Rediskontkontingente
  - Aufteilung
  - Gesamtumfang

Jedes der benutzten Stichworte sowie jedes behandelte Thema bildet einen Eintrag im Index des Medienobjektes. Für "Rediskontkontingente" und "rediskontfähige Wechsel" werden im Text zusätzliche,

---

[16] Baßeler, U., Heinrich, J., und W. Koch, *Grundlagen und Probleme der Volkswirtschaft (13. Auflage).* Köln: Wirtschaftsverlag Bachem, 1991.

speziellere Aspekte behandelt, die in den Index als Untereinträge des jeweiligen Stichwortes aufgenommen werden[17].

Die Erkennung der benutzten Stichworte und behandelten Themen ist manuell sehr einfach durchführbar und kann zusätzlich durch automatische Verfahren aus dem *Information Retrieval* unterstützt werden (siehe Kapitel 8.1.4 und 8.1.5).

Das einzige Problem bei der (manuellen) Erstellung der Indizes liegt darin, den richtigen Detaillierungsgrad zu finden. Das oben angegebene Medienobjekt hätte z.B. auch wie folgt indiziert werden können:

- Deutsche Bundesbank
- Geschäftsbank
- haftendes Eigenkapital
- rediskontfähiger Wechsel
- Rediskontkontingente

Bei dieser zweiten Variante wurden alle Untereinträge weggelassen und nur die wichtigsten der verwendeten Stichworte in den Index aufgenommen.

Der kleinere Index ist natürlich aus Sicht des Autors der bessere, da er schneller und einfacher zu erstellen ist. Für die automatische Generierung eines Lehr- oder Informationssystems ist jedoch die ausführlichere Variante die geeignetere, da sie mehr Informationen enthält, anhand derer die Auswahl und Strukturierung der Medienobjekte durchgeführt werden kann.

Um auch hier einen guten Kompromiß zwischen Autor und Algorithmus zu finden, ist eine gewisse Kenntnis der in Kapitel 7 beschriebenen Auswahl- und Strukturierungsalgorithmen nötig. Stark vereinfacht läßt sich der Zusammenhang zwischen Detaillierungsgrad des Index und Qualität[18] des erzeugten Dokuments wie folgt beschreiben:

- Je detaillierter der Index, desto "besser" ist die berechnete Medienobjektstruktur .
- Je weniger Medienobjekte ähnlichen Inhalts existieren, desto geringer ist die Auswirkung des Detaillierungsgrades auf die Qualität des erzeugten Dokuments.
- Die Algorithmen funktionieren auch mit knappen Indizes noch relativ gut, sind aber sensibel gegenüber verschieden detailliert beschriebenen Medienobjekten.

Aus diesen Eigenschaften der Auswahl- und Strukturierungsalgorithmen lassen sich einige Faustregeln für die Indizierung von Medienobjekten ableiten:

- Existieren nur maximal zwei bis drei Medienobjekte ähnlichen Inhalts, kann der Index relativ oberflächlich und knapp gehalten werden.
- Existieren mehr als drei Medienobjekte ähnlichen Inhalts und werden alle Medienobjekte von der gleichen Person indiziert, ist eine detailliertere Indizierung sinnvoll, da sie zu besseren Resultaten führt.
- Existieren mehr als drei Medienobjekte ähnlichen Inhalts und wird die Indizierung von mehreren Personen durchgeführt, ist aufgrund zu erwartender Schwankungen im Detaillierungsgrad zwischen den Indizierern eine eher knappe Indizierung vorzuziehen.

Eine mögliche Erweiterung des *i4 Frameworks* zur Lösung dieses Problems wäre die Integration von *Volltext Retrieval* Mechanismen als Ergänzung zur Auswertung der manuell angelegten Indizes. Dies würde unter Umständen die Abhängigkeit von Detaillierungsgrad und Qualität vermindern, was insbesondere dann hilfreich ist, wenn Medienobjekte von mehreren Autoren erstellt und indiziert werden.

---

[17] Für Untereinträge von Stichworten wird im folgenden die Notation "Stichwort::Untereintrag" verwendet.

[18] Der Begriff "Qualität" bezieht sich hier hauptsächlich auf die Nutzeranpassung und die Nähe der Medienobjekt-Struktur zur semantischen (konzeptuellen) Struktur des behandelten Wissensgebiets.

4.4.3 Gewichtung von Indizes

Ziel der automatisierten Strukturierung von Medienobjekten ist es, inhaltlich verwandte Medienobjekte zu erkennen und in eine Ordnung zu bringen.

Mit dem oben vorgestellten Index kann lediglich die erste Anforderung erfüllt werden: Durch Vergleich der Indizes können inhaltlich verwandte Objekte erkannt werden; je mehr Indexeinträge bei beiden Objekten identisch sind, desto größer ist die inhaltliche Nähe der Objekte.

Um eine Ordnung auf Medienobjekten aufprägen zu können, werden noch zusätzliche Informationen über die Verwendung der von einem Medienobjekt referenzierten Stichworte benötigt.

Die für die Erzeugung von Lehr- und Informationssystemen wichtigste Ordnung auf Medienobjekten ist die **Vorgänger-Nachfolger Beziehung**. Medienobjekt o ist dabei Vorgänger von Medienobjekt o' (und o' Nachfolger von o), wenn o Informationen vermittelt, die für das Verständnis von o' benötigt werden.

Die für die automatisierte Strukturierung benötigten Zusatzinformation sind somit vor allem das benötigte **Vorwissen** eines Medienobjektes und das neue, von ihm **vermittelte Wissen**.

In dem im vorherigen Abschnitt angegebenen Medienobjekt fallen z.B. die Stichworte "Deutsche Bundesbank", "Geschäftsbank", "haftendes Eigenkapital" und "rediskontfähiger Wechsel" in den Bereich des benötigten Vorwissens, während zu den Stichworten "Ankauf rediskontfähiger Wechsel", "Rediskontkontingente", "Aufteilung der Rediskontkontingente" und "Gesamtumfang der Rediskontkontingente" Informationen vermittelt werden.

Der Index eines Medienobjektes ist somit nicht als einfache Abbildung von einem Medienobjekt auf die Menge der darin vorkommenden Stichworte aufgebaut, sondern wird aus einzelnen - mit den benötigten Zusatzinformationen versehenen - Indexeinträgen zusammengesetzt. Diese Form der Indizierung wird auch als *role-based indexing* bezeichnet, da jedes referenzierte Stichwort über seine Rolle innerhalb des Medienobjektes beschrieben wird [Brusilovsky+96b]. Etwas vereinfacht kann man den Index somit auch als den von einem Medienobjekt abgedeckten Teil eines Thesaurus zum behandelten Themengebiet betrachten.

Jeder **Indexeintrag** besteht aus:

- einem Medienobjekt, zu dem der Indexeintrag gehört,
- einem von diesem Medienobjekt referenzierten Stichwort,
- dem Grad des verlangten Vorwissens über das Stichwort, das benötigt wird, um das Medienobjekt zu verstehen,
- dem Grad des von dem Medienobjekt über das Stichwort vermittelten Wissens und
- der Relevanz des Stichwortes für das Medienobjekt

Die nachfolgende Abbildung verdeutlicht die Stellung des Index als mit Zusatzinformationen versehene Relation zwischen Medienobjekten und Stichworten anhand eines (vereinfachten) E-R Diagramms:

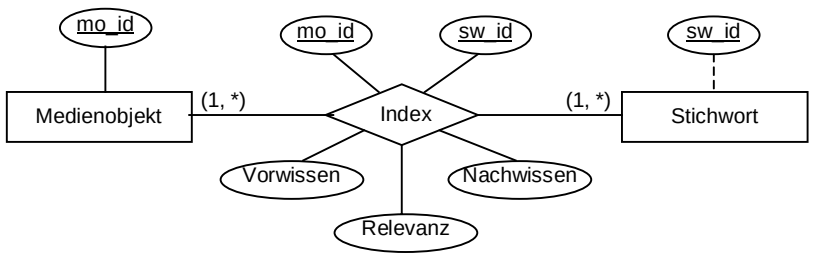


Abbildung 4-2: Verknüpfung von Medienobjekten mit Stichworten durch Indexeinträge

Der Wertebereich von verlangtem Wissen, vermitteltem Wissen und Relevanz ist jeweils

$R = \{0, 1, ..., r\}, r \in IN.$

Das **verlangte Vorwissen** gibt an, in wie weit Kenntnisse über ein Stichwort für das Verständnis eines Medienobjektes benötigt werden. Ein Vorwissen mit dem Wert 0 bedeutet, daß kein Vorwissen benötigt wird. Der Maximalwert r repräsentiert analog, daß fundiertes Wissen über das Stichwort benötigt wird, um das Medienobjekt zu verstehen.

Das **vermittelte Wissen** (Nachwissen) gibt an, in welchem Maße ein Benutzer nach Betrachten des Medienobjekts Kenntnisse über ein Stichwort besitzt. Ein Wert von 0 steht für "keine Informationen", während der Maximalwert r angibt, daß das Stichwort erschöpfend erklärt wird.

Eine **Relevanz** von 0 bedeutet, daß das Stichwort für das Medienobjekt irrelevant ist (d.h. genauso gut weggelassen werden könnte), während der Maximalwert r angibt, daß dieses Stichwort den inhaltlichen Schwerpunkt des Medienobjektes bildet.

Um einen Anhaltspunkt für die Interpretation einzelner Werte aus dem Wertebereich R zu geben, listet die folgende Tabelle die inhaltliche Bedeutung der Werte von R für r = 4 auf:

Verlangtes Wissen (required)

0: kein Vorwissen verlangt
1: ein rudimentäres Begriffsverständnis wird benötigt
2: Grundwissen wird benötigt
3: gute Kenntnisse werden benötigt
4: Detailkenntnisse sind verlangt

Vermitteltes Wissen (provided)

0: keine neuen Informationen werden vermittelt
1: Bedeutung des Stichwortes läßt sich aus seiner Verwendung ableiten
2: Grundwissen wird vermittelt
3: gute Kenntnisse werden vermittelt
4: Detailkenntnisse werden vermittelt

Relevanz (relevance)

0: Stichwort kann ignoriert werden
1: Stichwort wird eher am Rande erwähnt
2: Stichwort hat einleitenden oder weiterführenden Charakter
3: eines der im Zentrum des Medienobjekts stehenden Themen
4: zentrales Thema des Medienobjektes

Für r > 4 sollten die hier angegebenen Interpretationen in etwa auf r_0, $r_{0.25}$, $r_{0.5}$, $r_{0.75}$ und r_1 zutreffen.

Die Menge aller Indexeinträge wird als I^{avail} bezeichnet und besteht aus Tupeln der oben angeführten Elemente:

$$I^{avail} \subseteq M^{avail} \times S^{avail} \times R \times R \times R \tag{4.6}$$

mit $R := \{0, 1, ..., r\}, r \in IN^+$

Für die einzelnen Bestandteile eines Indexeintrags existieren die folgenden Projektionen:

$$mo((m, s, x, y, z)) := m \tag{4.7a}$$
$$kw((m, s, x, y, z)) := s \tag{4.7b}$$
$$req((m, s, x, y, z)) := x \tag{4.7c}$$
$$prv((m, s, x, y, z)) := y \tag{4.7d}$$
$$rel((m, s, x, y, z)) := z \tag{4.7e}$$

4.4.4 Stichworte und Konzepte

An verschiedenen Stellen dieser Arbeit wurde bereits erwähnt, daß eine der Stärken des *i4 Frameworks* darin liegt, daß Medienobjekte selbstbeschreibend sind. Hierdurch können ohne Modifikation an einer

übergeordneten Struktur neue Objekte in den Medienobjektspeicher eingefügt, bzw. veraltete Objekte daraus entfernt werden.

Aus dem gleichen Grund müssen auch **Stichworte** selbstbeschreibend sein. Sobald eine Struktur auf Stichworten existiert, muß diese gepflegt werden. Dies würde konkret bedeuten, daß jedesmal wenn ein Medienobjekt in den Medienobjektspeicher eingefügt wird, potentiell eine Aktualisierung der Stichwort-Struktur notwendig ist. Hiermit wäre jedoch die einfache Wartbarkeit und Erweiterbarkeit des Medienobjektspeichers und damit auch der erzeugten Lehr- und Informationssysteme nicht mehr gegeben.

Da das *i4 Framework* weitestgehend auf die Beachtung von semantischen Aspekten verzichtet, bei der Beschreibung von Stichworten jedoch hauptsächlich semantische Aspekte von Bedeutung wären, impliziert die Forderung nach selbstbeschreibenden Stichworten, daß Stichworte im Endeffekt ohne weitere Beschreibung existieren müssen.

Die fehlende Struktur auf Stichworten stellt für die automatische Auswahl und Strukturierung von Medienobjekten ein erheblich größeres Problem dar als der Verzicht auf eine explizite Struktur auf Medienobjekten: Stichworte sind von ihrer Granularität her noch feiner als Medienobjekte. Falls eine Struktur auf Stichworten existiert, könnte daher aus dieser Struktur relativ einfach auch eine Struktur auf Medienobjekten abgeleitet werden.

Aus diesem Grund verwenden die weitaus meisten Systeme zur automatisierten Erzeugung von Lehr- und Informationssystemen anstelle von Stichworten **Konzepte** zur inhaltlichen Beschreibung von Medienobjekten, Seiten, etc. Konzepte tragen eine Semantik und sind vor allem durch gegenseitige Beziehungen (z.B. "ist-ein" oder "ist-Teil-von") strukturiert. Eine im Bereich von Lehrsystemen weit verbreitete Struktur auf Konzepten ist das semantische Netz [Quillian66].

Im Gegensatz hierzu sind die **Stichworte** des *i4 Frameworks* nichts weiter als reine Aneinanderreihungen von Zeichen, die idealerweise aus einem indizierten Medienobjekt extrahiert wurden. Sie entsprechen somit eher den von Suchmaschinen wie z.B. *Altavista* verwendeten Phrasen als den Konzepten anderer Systeme zur Erzeugung von Lehr- und Informationssystemen.

Leider läßt sich diese restriktive Definition von Stichworten nicht immer durchhalten. Ein Indexeintrag wie z.B. "Aufteilung der Rediskontkontingente" besteht im Grunde genommen aus zwei Stichworten, nämlich "Aufteilung" und "Rediskontkontingente". Das Problem bei der ausschließlichen Verwendung von einzelnen - durch Volltextsuche ermittelbaren - Stichworten zur inhaltlichen Beschreibung von Medienobjekten ist, daß dadurch nur eine sehr grobe Beschreibung möglich ist. Zwei Texte, von denen einer die Aufteilung der Rediskontkontingente und einer den Umfang der Rediskontkontingente erklärt, wären anhand ihres Index potentiell nur noch unterscheidbar, wenn (fast) jedes in den Texten vorkommende Wort in den Index aufgenommen würde. Ein solches Verfahren der quasi Volltext-Indizierung ist bei der momentan noch notwendigen manuellen Festlegung von verlangtem und vermitteltem Wissen jedoch nicht praktikabel.

Aus diesem Grund sind als Stichworte im Sinne des *i4 Frameworks* bei einer manuellen Indizierung neben einzelnen Stichworten auch zusammengesetzte Stichworten zulässig. Wichtig hierbei ist jedoch, daß auch zusammengesetzte Stichworte keine Semantik tragen, d.h. sie sind für das *i4 Framework* nichts weiter als aus mehreren Worten bestehende Phrasen, die von einem bereits bekannten Stichwort abgeleitet sind.

Durch zusammengesetzte Stichworte entstehen Hierarchien. Ein Stichwort besteht somit immer aus einem - potentiell leeren - übergeordneten Stichwort und einer das Stichwort (bzw. den zweiten Teil des zusammengesetzten Stichwortes) repräsentierenden Phrase.

Das Stichwort "Rediskontkontingente" besteht z.B. aus einem leeren übergeordneten Stichwort und der Phrase "Rediskontkontingente". Das Stichwort "Ankauf von Rediskontkontingenten" besteht aus dem übergeordneten Stichwort "Rediskontkontingente" und der Phrase "Ankauf von". Die so entstandene Hierarchie entspricht in ihrer Bedeutung weitestgehend der Semantik von hierarchischen Buch-Indizes. Die im folgenden verwendete Schreibweise für zusammengesetzte Stichworte ist "Stichwort::Phrase".

Falls bei einem zusammengesetzten Stichwortes nicht entschieden werden kann, welcher der beiden Teile das Stichwort und welcher die Phrase ist, sollten beide Kombinationen indiziert werden (z.B. "Rediskontkontingente::Ankauf von" und "Ankauf::von Rediskontkontingenten").

Neben dem übergeordneten Stichwort und der Angabe identischer Stichworte kann für jedes Stichwort auch ein Default-Vorwissen angegeben werden. Das Default-Vorwissen legt fest, in welchem Maße bei einem Mitglied der angesprochenen Zielgruppe Vorwissen über ein Stichwort zu erwarten ist. Der Wertebereich des Default-Vorwissens ist mit dem Wertebereich des verlangten Wissens von Indexeinträgen identisch. Das Default-Vorwissen wird lediglich als Vereinfachung der Anpassung an ein Nutzungsszenario verwendet, d.h. seine Angabe ist optional (siehe auch Kapitel 5.4).

Das übergeordnete Stichwort sowie das Default-Vorwissen eines Stichwortes werden vom *i4 Framework* in Form von Attributen verwaltet. Neben diesen beiden Attributen können -analog zu Medienobjekten - auch auf Stichworten zusätzliche, anwendungsabhängige Attribute definiert werden. Die Menge aller auf Stichworten verfügbaren Attribute wird als A^S definiert:

$$A^S \subseteq S^{avail} \times attr\_name \times attr\_value, \tag{4.8}$$

Wie auf Medienobjekt-Attributen, existieren auch auf Stichwort-Attributen zwei Abbildungen, die zu einem gegebenen Stichwort alle zugehörigen Attribute, bzw. den Wert eines bestimmten Attributs liefern:

$$\sigma_{attr}: S^{avail} \rightarrow \mathsf{P}(attr\_name \times attr\_value) \tag{4.9}$$

$$\sigma_{attr}(s) := \{ (n, v) \mid (s, n, v) \in A^S \}$$

$$\sigma_{val}: S^{avail} \times attr\_name \times attr\_value \rightarrow attr\_value \tag{4.10}$$

$$\sigma_{val}(s, n, d) := \begin{cases} v, & \text{falls } \exists (s, n, v) \in A^S \\ d, & \text{sonst} \end{cases}$$

Die einzigen vordefinierten Attribute von Stichworten sind das übergeordnete Stichwort sowie das Default-Vorwissen:

$$\sigma_{parent}: S^{avail} \rightarrow S^{avail}, \quad \text{mit}$$

$$\sigma_{parent}(s) := \sigma_{val}(s, \text{"parent"}, 0) \tag{4.11a}$$

$$\sigma_{know}: S^{avail} \rightarrow R, \quad \text{mit}$$

$$\sigma_{know}(s) := \sigma_{val}(s, \text{"default"}, 0) \tag{4.11b}$$

Ist für ein Stichwort s kein übergeordnetes Stichwort angegeben, so ist der Wert von $\sigma_{parent}(s)$ gleich 0.

4.4.5 Synonyme und Homonyme

Wie bei jedem System, das auf Stichworten oder Konzepten operiert, stellt sich auch beim *i4 Framework* die Frage nach der Behandlung von Synonymen und Homonymen. Als **Synonyme** werden dabei verschiedene Stichworte für identische Konzepte bezeichnet (z.B. "Anfang" und "Beginn") während **Homonyme** genau umgekehrt identische Stichworte für verschiedenen Konzepte darstellen (z.B. "Kiefer": Knochen vs. Baum).

Homonyme stellen für das *i4 Framework* kein praktisches Problem dar, da davon ausgegangen wird, daß im Medienobjektspeicher immer nur Medienobjekte zu einem einzigen Thema verwaltet werden. Falls dennoch Homonyme vorkommen sollten, liegt es in der Verantwortung des Autors, diese - z.B. durch Annotation oder Verwendung verschiedener Phrasen - aufzulösen.

Problematisch sind jedoch Synonyme, da sie die semantische Identität von Stichworten verbergen. Dies hat zur Folge, daß ein Medienobjekt, das z.B. die Produktion von Streichhölzern erklärt, als inhaltlich verschieden von einem Objekt über die Produktion von Zündhölzern angesehen wird. Auch die Herstellung und die Produktion von Streichhölzern wären aus diesem Grunde für die Auswahl- und Strukturierungsalgorithmen zwei grundverschiedenen Dinge.

Generell lassen sich bezüglich der Auswirkungen auf die Algorithmen des *i4 Frameworks* zwei verschiedene Arten von Synonymen unterscheiden:

- synonym verwendete Ausdrücke[19] aus dem behandelten Themengebiet, bei denen nicht vorausgesetzt werden kann, daß sie vom Leser des generierten Dokuments als verschiedenen Bezeichnungen des selben Sachverhaltes erkannt werden (z.B. Abkürzungen wie "Westeuropäische Union" und "WEU" oder Umschreibungen von Orten, Zeiten oder Personen wie "Spree-Athen" und "Berlin").
- synonym verwendete Ausdrücke, die keine ausschließlich dem behandelten Wissensgebietes zuzuordnenden Begriffe sind, da sie z.B. als Unterstichworte zur genaueren Bezeichnung von Tätigkeiten und Eigenschaften verwendet werden (z.B. "Streichhölzer::Produktion" und "Streichhölzer::Herstellung").

Synonyme der ersten Kategorie sind normalerweise Bestandteil des behandelten Themengebiets, während Synonyme der zweiten Kategorie üblicherweise bei der Indizierung erzeugt werden[20] (z.B. bei der Indizierung durch verschiedene Personen).

Synonyme können im Rahmen einer automatischen Dokumentenerzeugung Ursache von unnötigen und teilweise sogar sehr störenden Redundanzen sein. Verwendet z.B. ein Medienobjekt die Begriffe "WEU" und "Westeuropäische Union" als Vorwissen, so wird der Auswahlalgorithmus versuchen, Medienobjekte in das erzeugte Dokument aufzunehmen, die diese beiden Begriffe erklären. Idealerweise sollte dabei natürlich nicht eine mehr oder minder identische Begriffserklärung zweimal - einmal für "WEU" und einmal für "Westeuropäische Union" - verwendet werden.

Synonyme der ersten Kategorie bereiten für die automatische Erzeugung von Dokumenten keine praktischen Probleme, da es in der Praxis so gut wie nie vorkommt, daß ein Medienobjekt nur einen der Begriffe erklärt. Jede Beschreibung der Westeuropäischen Union wird auch die Abkürzung WEU enthalten und jedes Medienobjekt zu "Kofi Anan" wird auch seinen momentanen Beruf nennen. Aus diesem Grund können Synonyme der ersten Kategorie dadurch aufgelöst werden, daß sie wie zwei verschiedene Stichworte behandelt und getrennt indiziert werden. Wenn z.B. in einem Medienobjekt der Begriff "Westeuropäische Union" auftaucht, wird "Westeuropäische Union" indiziert, wenn "WEU" auftaucht, dann "WEU" und wenn beide Begriffe auftauchen, dann "Westeuropäische Union" und "WEU".

Schwierigkeiten bereiten vor allem die für den Leser unsichtbaren, bei der Indizierung entstandenen Synonyme der zweiten Kategorie. Um diese Synonyme aufzulösen, existieren zwei Möglichkeiten:

- Sie werden generell vermieden, indem nur bestimmte Begriffe zur Umschreibung von Tätigkeiten, Eigenschaften, etc. verwendet werden (z.B. immer "Herstellung", nie "Produktion", immer "Anfang", nie "Beginn").
- Sie werden mit Hilfe eines Synonymwörterbuchs automatisch durch einen einheitlichen Term ersetzt.

Da das erste Verfahren fehleranfällig und bei mehreren Autoren oder automatischer Autoren-unterstützung nur bedingt einsetzbar ist, sollte jeder Medienobjekt-Index nach seiner Erstellung mit Hilfe einer Synonymliste "standardisiert" werden. Die Synonymliste fungiert dabei als Tabu-Liste, in der alle bei der Indizierung nicht zu verwendenden Phrasen samt ihrer Ersatzbezeichnungen aufgeführt sind (z.B. "Herstellung" statt "Produktion").

4.4.6 Beispiel für die Indizierung von Medienobjekten

Um einen Eindruck davon zu geben, wie die Arbeit des Autors eines Lehr- oder Informationssystems bei Verwendung einer *Bottom-Up* Generierung konkret aussieht, soll in diesem Abschnitt die Indizierung von Medienobjekten anhand eines Beispiels vorgeführt werden. Grundlage hierzu sind 5 textuelle und 1 grafisches Medienobjekt, die aus einem Buch über Typographie entnommen wurden[21]. Die Extraktion der Medienobjekte folgte dabei dem einfachen Grundsatz, daß jeder Absatz potentiell ein Medienobjekt ist.

---

[19] Hier wurde absichtliche eine etwas vage Formulierung gewählt, da auch Begriffspaare wie z.B. "Bundespräsident" und "Johannes Rau" die selben Auswirkungen wie (echte) Synonyme haben.

[20] In die zweite Kategorie fallen auch sog. *Quasi-* oder *Pseudo-Synonyme*, d.h. begriffliche Überschneidungen, die bei der Indizierung aus Unkenntnis oder Bequemlichkeit synonym verwendet werden [Meiss97].

[21] Gulbins, J. und Kahrmann, C., *Mut zur Typographie*. Berlin, Heidelberg: Springer, 1992.

Das erste Medienobjekt ist ein kurzer Absatz, der die (synonymen) Begriffe "Schriftlinie" und "Grundlinie" erklärt:

> *Die Schriftlinie oder Grundlinie ist eine gedachte Linie, an der die Schrift ausgerichtet ist. Runde Formen müssen etwas über die mathematische Grundlinie (und x-Höhe) hinausreichen, um optisch auf der Grundlinie zu stehen bzw. um optisch die gleiche Höhe wie ein glatt abschließendes Zeichen zu haben.*

Die erste Aufgabe des Autors eines dynamischen Dokuments über Typographie wäre, dieses Medienobjekt im Medienobjektspeicher abzulegen und die wichtigsten beschreibenden Attribute mit Werten zu belegen:

Name = "Typographie: Grundlinie"
Caption = "Grundlinie"
Quality = 0.8
AccessProtection = "copyrighted"
Copyright = "Springer Verlag, 1992"
Genre = "content"
Difficulty = 0.5
MimeType = "text/plain"
MediaSize = 302
MediaWidth = 360
MediaHeight = 100
Language = "de"

Die Angabe eines Namens und einer Überschrift ist optional und wurde in diesem Fall vom Autor manuell durchgeführt. Die Werte der letzten fünf Attribute - *Mime-Type*, Textlänge, Textbreite, Texthöhe und Sprache - wurden von der Autorenumgebung automatisch bestimmt (siehe Kapitel 4.5).

Die zweite Aufgabe des Autors ist das Anlegen der den Inhalt des Medienobjektes beschreibenden Indexeinträge. Hierbei sind zuerst die referenzierten Stichworte und Themen zu ermitteln:

Schriftlinie
Grundlinie
    und runde Formen
Linie
Schrift
x-Höhe
    und runde Formen
Zeichen

Die drei fett gedruckten Stichworte bilden den minimalen Index dieses Medienobjektes, die Angabe aller weiteren Stichworte fällt in den Ermessensbereich des Autors. Da in diesem Beispiel von einem Ein-Autoren Dokument ausgegangen wird, wurde eine sehr detaillierte Version des Index gewählt. Die synonymen Begriffe "Grundlinie" und "Schriftlinie" wurden getrennt in den Index aufgenommen, da es sich um themenspezifische Synonyme handelt (siehe Kapitel 4.4.5).

Abschließend muß der Autor für jeden Indexeintrag den Grad an verlangtem und vermitteltem Wissen sowie die Relevanz des Stichwortes für das Medienobjekt angeben (r=4):

Index zu Medienobjekt 1

Stichwort	required	provided	relevance
Schriftlinie	0	3	4
Grundlinie	0	3	4
und runde Formen	0	2	2

Linie	1	1	0
Schrift	1	1	2
x-Höhe	2	2	1
und runde Formen	0	2	2
Zeichen	1	1	1

Die hier dargestellte Attributierung und Indizierung muß der Autor für jedes im Medienobjektspeicher befindliche Medienobjekt durchführen. Die dazu benötigte Zeit liegt je nach Medienobjekt normalerweise zwischen zwei und fünf Minuten.

Das zweite Medienobjekt im Medienobjektspeicher des Beispieldokuments ist ein Text über die Mittellänge einer Schrift:

> *Die Mittellänge einer Schrift, auch x-Höhe genannt, gibt die Höhe der Zeichen wie a, c, e, m oder x an, wobei bei einzelnen Zeichen Kurventeile des Zeichens (z.B. beim p) geringfügig über diese Mittellänge hinausragen dürfen. Man spricht hier von einem Überhang.*

Index zu Medienobjekt 2

Stichwort	required	provided	relevance
Schrift	1	1	1
Mittellänge	0	3	4
x-Höhe	0	3	4
Zeichen	1	1	2
Zeichenhöhe	1	1	1
Kurventeile	1	1	1
und Mittellänge	0	2	2
Überhang	0	3	3

In Medienobjekt 3 werden Ober- und Unterlänge beschrieben:

> *Als Unterlänge wird die Strecke bezeichnet, um die Kleinbuchstaben wie z.B. g, j, p, q oder y über die Grundlinie nach unten ragen. [...] Die Oberlänge eines Zeichens ist das Maß, mit dem Kleinbuchstaben wie d, f, h, k, l, t über die Mittellänge hinausragen.*

Index zu Medienobjekt 3

Stichwort	required	provided	relevance
Unterlänge	0	3	4
Strecke	1	1	1
Kleinbuchstaben	1	1	1
Grundlinie	3	3	2
Oberlänge	0	3	4
Mittellänge	3	3	1

Medienobjekt vier des Beispiels ist ebenfalls ein Text:

> *Die Kegelgröße [...] bestimmt sich aus der Oberlänge plus der Unterlänge plus etwas Zuschlag und legt beim Bleisatz den Mindestzeilenabstand fest. Dieser Wert wird bei DTP-Programmen auch Schriftgrad oder Schriftgröße genannt.*

Index zu Medienobjekt 4

Stichwort	required	provided	relevance
Kegelgröße	0	3	4
Oberlänge	3	3	2
Unterlänge	3	3	2
Bleisatz	1	1	1
Mindestzeilenabstand	1	2	2
DTP-Programm	2	2	1
Schriftgrad	0	3	3
Schriftgröße	0	3	3

Medienobjekt fünf ist eine Grafik, in der die hinter den Begriffen "Oberlänge", "Unterlänge" und "Schriftlinie" stehenden Maße anhand eines Beispielwortes verdeutlicht werden:

Index zu Medienobjekt 5

Stichwort	required	provided	relevance
Schriftlinie	0	2	4
Oberlänge	0	2	4
Unterlänge	0	2	4

Medienobjekt sechs ist wieder ein Text und behandelt einige der in der Typographie verwendeten Maßeinheiten:

> *Die meisten DTP-Programme verwenden [als Einheit für die Schriftgröße] den sogenannten DTP-Punkt. Dieser leitet sich aus dem 72-sten Teil eines Inches ab [...]. Ein DTP-Punkt entspricht damit etwa 0,3528 mm. Zwölf DTP-Punkte werden [..] ein Pica genannt und entsprechen 0,4233 mm. Sechs DTP-Pica ergeben genau ein Inch.*

Index zu Medienobjekt 6

Stichwort	required	provided	relevance
DTP-Programm	2	2	2
Schriftgröße	1	1	3
DTP-Punkt	0	3	4
Inch	1	1	3
Pica	0	3	3

Jedesmal, wenn der Autor bei der Indizierung eines Medienobjektes dem Gesamtindex ein neues Stichwort hinzufügt, sollte gleichzeitig das Default-Vorwissen zu diesem Stichwortes angegeben werden. Dies geschieht durch Setzten des “default” Attributs (siehe Abschnitt 4.4.4).

Die folgende Übersicht zeigt den aggregierten Index aller sechs Medienobjekte inklusive des für die angenommene Zielgruppe vermuteten Default-Vorwissens:

Nr.	Stichwort	Vorwissen
1	Bleisatz	1
2	DTP-Programm	2
3	DTP-Punkt	0
4	Grundlinie	1
5	und runde Formen	0
6	Inch	2
7	Kegelgröße	0

8	Kleinbuchstaben	4
9	Kurventeile	1
10	und Mittellänge	0
11	Linie	3
12	Mindestzeilenabstand	1
13	Mittellänge	0
14	Oberlänge	0
15	Pica	0
16	Schrift	3
17	Schriftgrad	0
18	Schriftgröße	1
19	Schriftlinie	0
20	Strecke	3
21	Überhang	0
22	Unterlänge	0
23	x-Höhe	0
24	und runde Formen	0
25	Zeichen	3
26	Zeichenhöhe	1

Wie dieses Beispiel zeigt, eignet sich der *Bottom-Up* Ansatz gut für die Konvertierung existierender Lehrbücher in dynamische, web-basierte Dokumente. Auch die Zusammenfassung mehrerer Dokumente zu einem einzigen Dokument stellt keinen zusätzlichen Aufwand dar, da die Auswahl und Strukturierung sich inhaltlich überschneidender Medienobjekte automatisch durchgeführt wird.

4.4.7 Medienobjekte als Graphen

Ein einzelnes Medienobjekt läßt sich als gerichteter, bipartiter Graph darstellen. Die Knoten dieses Graphen sind das Medienobjekt selbst sowie die von ihm verlangten und vermittelten Stichworte. Die Menge der von einem Medienobjekt zu den Stichworten führenden Kanten sowie die von den Stichworten zum Medienobjekt führenden Kanten ergeben sich aus den verlangten und vermittelten Indexeinträgen des Medienobjektes.

Definition

Der **Graph G_m eines Medienobjektes** $m \in M^{avail}$ innerhalb einer Menge von Indexeinträgen $I \subseteq I^{avail}$ hat eine sternförmige Struktur mit dem Medienobjekt als Zentrum. Ausgehende Kanten führen zu allen vermittelten Stichworten, eingehende Kanten beginnen bei allen verlangten Stichworten:

$$G_m(m,I) = (\{m\}, \mu_{kw}(m,I), \mu_{req}(m,I), \mu_{prv}(m,I)) \quad (4.12)$$

Die nachfolgende Abbildung zeigt die Graph-Repräsentation des ersten der in Abschnitt 4.4.6 als Beispiel angegebenen Medienobjekte:

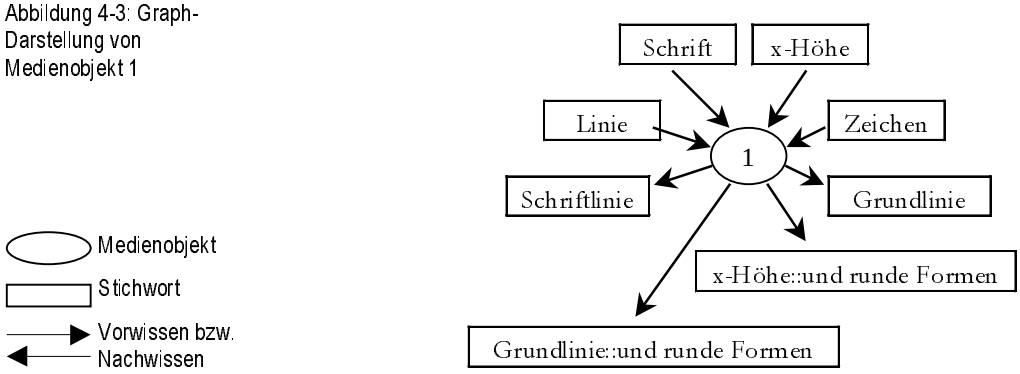


Abbildung 4-3: Graph-Darstellung von Medienobjekt 1

Abbildung 4-3 zeigt die in dieser Arbeit verwendeten Konventionen zur Darstellung von aus Medienobjekten und Stichworten bestehenden Graphen: Medienobjekte werden als Ovale dargestellt, wobei in der Mitte des Ovales der Name (oder der Index) des Medienobjektes angegeben ist. Stichworte werden durch Rechtecke repräsentiert. Auch hierbei wird der Name (oder der Index) des Stichwortes angegeben.

4.4.8 Indexeinträge

Die Indexeinträge eines Medienobjektes m bzw. eines Stichwortes s innerhalb einer vorgegebenen Menge von Indexeinträgen $I \subseteq I^{avail}$ lassen sich durch die Abbildungen μ_{ndx} und σ_{ndx} bestimmen:

$$\mu_{ndx}: M^{avail} \times \mathbf{P}(I^{avail}) \rightarrow \mathbf{P}(I^{avail}) \tag{4.13}$$

$$\text{mit } \mu_{ndx}(m, I) := \{ i \in I \mid mo(i) = m \}$$

$$\sigma_{ndx}: S^{avail} \times \mathbf{P}(I^{avail}) \rightarrow \mathbf{P}(I^{avail}) \tag{4.14}$$

$$\text{mit } \sigma_{ndx}(s, I) := \{ i \in I \mid kw(i) = s \}$$

Bezogen auf die Graphdarstellung bezeichnen μ_{ndx} bzw. σ_{ndx} die Menge aller Eingangs- und Ausgangskanten des bezeichnten Medienobjekts bzw. Stichworts.

Jedem Medienobjekt ist über seine Indexeinträge eine Menge von Stichworten zugeordnet, die von ihm entweder als Vorwissen verlangt oder (bzw. und) als neues Wissen vermittelt werden, d.h. die in der Graphdarstellung zu diesem Medienobjekt direkt benachbart sind:

$$\mu_{kw}: M^{avail} \times \mathbf{P}(I^{avail}) \rightarrow \mathbf{P}(S^{avail}) \tag{4.15}$$

$$\text{mit } \mu_{kw}(m, I) := \bigcup_{i \in \mu_{ndx}(m,I)} kw(i)$$

Anhand des verlangen und vermittelten Wissens lassen sich zu jedem Medienobjekt die Vorwissen bzw. neues Wissen repräsentierenden Indexeinträge ermitteln, wobei die Vorwissen repräsentierenden Indexeinträge die Eingangskanten des Medienobjekts darstellen und die neues Wissen repräsentierenden Einträge die Ausgangskanten:

$$\mu_{req}: M^{avail} \times \mathbf{P}(I^{avail}) \rightarrow \mathbf{P}(I^{avail}) \tag{4.16a}$$

$$\text{mit } \mu_{req}(m, I) := \{ i \in \mu_{ndx}(m, I) \mid req(i) > 0 \}$$

$$\mu_{prv}: M^{avail} \times \mathbf{P}(I^{avail}) \rightarrow \mathbf{P}(I^{avail}) \tag{4.16b}$$

$$\text{mit } \mu_{prv}(m, I) := \{ i \in \mu_{ndx}(m, I) \mid req(i) < prv(i) \}$$

Analog können auch zu jedem Stichwort die dieses Stichwort vermittelnden (Eingangskanten) bzw. als Vorwissen (Ausgangskanten) verlangenden Indexeinträge bestimmt werden:

$$\sigma_{req}: S^{avail} \times \mathbf{P}(I^{avail}) \rightarrow \mathbf{P}(I^{avail}) \tag{4.17a}$$

$$\text{mit } \sigma_{req}(s, I) := \{ i \in \sigma_{ndx}(s, I) \mid req(i) > 0 \}$$

$$\sigma_{prv}: S^{avail} \times \mathsf{P}(I^{avail}) \rightarrow \mathsf{P}(I^{avail}) \tag{4.17b}$$

$$\text{mit } \sigma_{prv}(s, I) := \{ i \in \sigma_{ndx}(s, I) \mid req(i) < prv(i) \}$$

Mit Hilfe von (4.15) bis (4.17) lassen sich zu jedem Medienobjekt m die verlangten und vermittelten Stichworte $\mu_{kwreq}(m,I)$ bzw. $\mu_{kwprv}(m,I)$ innerhalb einer Menge von Indexeinträgen I bestimmen. In der Graphdarstellung sind dies die an den Eingangskanten bzw. Ausgangskanten des Medienobjekts anliegenden Stichworte:

$$\mu_{kwreq}: M^{avail} \times \mathsf{P}(I^{avail}) \rightarrow \mathsf{P}(S^{avail}) \tag{4.18a}$$

$$\text{mit } \mu_{kwreq}(m, I) := \bigcup_{i \in \mu_{req}(m,I)} kw(i)$$

$$\mu_{kwprv}: M^{avail} \times \mathsf{P}(I^{avail}) \rightarrow \mathsf{P}(S^{avail}) \tag{4.18b}$$

$$\text{mit } \mu_{kwprv}(m, I) := \bigcup_{i \in \mu_{prv}(m,I)} kw(i)$$

Analog lassen sich auch zu jedem Stichwort s die dieses Stichwort als Vorwissen verlangenden bzw. als neues Wissen vermittelnden Medienobjekte $\sigma_{moreq}(s,I)$ bzw. $\sigma_{moprv}(s,I)$ berechnen, d.h. die Mengen der an den Eingangs- bzw. Ausgangskanten von s anliegenden Medienobjekte:

$$\sigma_{moreq}: S^{avail} \times \mathsf{P}(I^{avail}) \rightarrow \mathsf{P}(M^{avail}) \tag{4.19a}$$

$$\text{mit } \sigma_{moreq}(s, I) := \bigcup_{i \in \sigma_{req}(s,I)} mo(i)$$

$$\sigma_{moprv}: S^{avail} \times \mathsf{P}(I^{avail}) \rightarrow \mathsf{P}(M^{avail}) \tag{4.19b}$$

$$\text{mit } \sigma_{moprv}(s, I) := \bigcup_{i \in \sigma_{prv}(s,I)} mo(i)$$

Ein Stichwort kann sowohl im verlangten als auch im vermittelten Wissen eines Medienobjekts enthalten sein, d.h. mit einem Stichwort sowohl über eine eingehende als auch eine ausgehende Kante verbunden sein. Die Semantik dieses Spezialfalls ist, daß Stichworte von Medienobjekten nicht nur komplett neu vermittelt, sondern auch **vertieft** werden können, indem bereits vorhandenes Wissen über das Stichwort erweitert wird.

Vertiefte Stichworte stellen für viele der Algorithmen zur automatischen Auswahl und Strukturierung einen aufwendig zu behandelnden Sonderfall dar, da sie häufig Ausgangspunkt von Zyklen innerhalb des im übernächsten Kapitel beschriebenen Abhängigkeitsgraphen sind.

Um basierend auf Mengen gegebener Medienobjekte, Stichworte und Indexeinträge eine Struktur berechnen zu können, müssen diese bestimmte Integritätsbedingungen erfüllen.

Die zwei wichtigsten Integritätsbedingungen sind:

IG_1: Jedes Medienobjekt muß mit mindestens einem Stichwort verknüpft sein, d.h. mindestens eine eingehende oder ausgehende Kante besitzen:

$$\forall m \in M^{avail}: |\mu_{ndx}(m)| > 0$$

IG_2: Jedes Stichwort muß im Index mindestens eines Medienobjekts enthalten sein, d.h. mindestens eine eingehende oder ausgehende Kante besitzen:

$$\forall s \in S^{avail}: |\sigma_{ndx}(s)| > 0$$

Durch die zweite Einschränkungen wird sichergestellt, daß die Menge der Stichworte dynamisch aus Medienobjekten und Indexeinträgen bestimmt werden kann.

Darüber hinaus existieren noch zwei weitere, semantisch motivierte, Integritätsbedingungen für Indexeinträge:

IG$_3$: Indexeinträge sind eindeutig bezüglich der Medienobjekte und Stichworte. Für jede Zuordnung eines Stichwortes zu einem Medienobjekt existiert genau ein Indexeintrag, d.h. wenn zwei Kanten zwischen einem Medienobjekt und einem Stichwort existieren, dann müssen diese entgegengesetzt gerichtet sein:

$$\forall (m_1, s_1, x_1, y_1, z_1), (m_2, s_2, x_2, y_2, z_2) \in I^{avail}: m_1=m_2, s_1=s_2 \Rightarrow x_1=x_2, y_1=y_2, z_1=z_2$$

IG$_4$: Da davon ausgegangen wird, daß der Nutzer einmal erlerntes Wissen durch Betrachten eines Medienobjektes nicht wieder verliert, muß für jeden Indexeintrag der Grad des vermittelten Wissens mindestens so groß sein, wie der des verlangten Wissens:

$$\forall i \in I^{avail}: prv(i) \geq req(i)$$

Die Überwachung dieser vier Integritätsbedingungen wird von der in Kapitel 8 beschriebenen Implementierung des *i4 Frameworks* automatisch durchgeführt.

4.5 Automatisierte Erzeugung von Medienobjekten

Die in den Kapiteln 4.3 und 4.4 beschriebene Erstellung, Attributierung und Indizierung von Medienobjekten ist manuell einfach und schnell durchzuführen. Wenn man jedoch bedenkt, daß die Medienobjekt-Basis eines Lehr- oder Informationssystems aus hunderten von Medienobjekten besteht, so bedeutet jede Möglichkeit der Automatisierung einen enormen Zeitgewinn für den Autor.

Das hierbei angestrebte Fernziel ist eine komplett automatische Gewinnung aller benötigten Meta-Informationen. Für den Autor würde dies bedeuten, daß er nur noch Medienobjekte erstellen und im Medienobjektspeicher ablegen muß. Das manuelle Beschreiben von Form und Inhalt der Objekte würde entfallen.

In diesem Kapitel werden einige Möglichkeiten diskutiert, drei der fünf zur Zeit noch manuell vorzunehmenden Arbeitsschritte bei der Erstellung von Medienobjekten zu automatisieren:

- Bestimmen der Eigenschaften eines Medienobjektes,
- Ermitteln des Index eines Medienobjektes,
- Festlegung des durchschnittlichen Vorwissens der Zielgruppe zu den im Gesamtindex enthaltenen Stichworten.

Weiterreichende Automatisierungen - insbesondere die Erstellung von Medienobjekten sowie die Parametrisierung der Algorithmen - werden in Kapitel 8.1 bis 8.3 behandelt.

4.5.1 Retrieval von Medienobjekt-Eigenschaften

In Kapitel 4.3 wurden eine Reihe von Attributen vorgestellt, die vom Autor eines dynamisch zu erzeugenden Lehr- oder Informationssystem zur Beschreibung eines jeden Medienobjektes angegeben werden sollten.

Einige dieser Eigenschaften wie z.B. die ID, die Adresse (URL) und die Dateigröße eines Medienobjektes werden bereits automatisch mit einem Wert belegt, da sie entweder vom Framework erzeugt und verwaltet werden oder sich implizit beim Anlegen des Medienobjektes ergeben.

Im Gegensatz hierzu ist eine Eigenschaft wie "Name" nicht automatisch ermittelbar, da sie auf Informationen und Konventionen beruht, die nur dem Autor bekannt sind. Auch die Werte stark mit der Semantik eines Medienobjekts verknüpfter Attribute wie "UpdatePolicy" und "ValidUntil" sind nicht mit ausreichender Genauigkeit automatisch bestimmbar.

Alle weiteren Eigenschaften können (und sollten) jedoch zumindest näherungsweise mit Hilfe automatischer Verfahren bestimmt werden.

MimeType

Für außerhalb des Medienobjektspeichers abgelegte Medienobjekte kann die Kodierung sehr einfach über die Dateierweiterung ermittelt werden (zumindest wenn sie den Namenskonventionen entspricht).

Aber auch für nicht als Datei vorliegende, im Medienobjektspeicher direkt abgelegte Objekte ist eine Bestimmung des Mime-Typs in den meisten Fällen anhand einer Auswertung weniger Bytes vom Anfang des Objekts möglich. Alle Grafikformate besitzen z.B. ein standardisiertes Headerformat, in dem an einer festgelegten Position eine Formatkennung und teilweise auch eine Versionsnummer kodiert ist. Ähnliches gilt für die meisten Audio- und Videoformate.

Die momentan unterstützten Textformate ASCII, HTML und RTF zeichnen sich dadurch aus, daß sie auf ASCII-Kodierungen beruhen, d.h. bestimmte Zeichenkodes (z.B. 0 bis 8) nicht enthalten. Hierdurch sind sie von Grafik- und Videoformaten unterscheidbar.

Mediawidth und Mediaheight

Alle Grafik- und Videoformate enthalten in ihrem *Header* eine Angabe über die Ausgabegröße, die bei Kenntnis des Formats automatisch ausgelesen werden kann. Einzige Ausnahmen sind vektorbasierte Formate wie z.B. *Makromedia Flash*, die beliebig skalierbar sind[22].

Für Texte ist eine Bestimmung ihrer Höhe und Breite nur mit Kenntnis der verwendeten Schriftart und -größe sowie des Layouts des generierten Dokuments möglich. Aus diesem Grund wird die Berechnung der Darstellungsgröße eines Textes durch ein vom Anbieter des Dokuments zu erstellendes *Java* Skript durchgeführt. Ist kein solches *Java* Skript angegeben, wird von Default-Werten (11 Punkte Schriftgröße, 60n Zeilenbreite, normaler Zeilenabstand) ausgegangen.

Quality

Je nach Typ des Medienobjektes werden für die Festlegung der Qualität verschiedene Einflußgrößen berücksichtigt:

- Für textuelle Medienobjekte bilden den Stil beschreibende Maßzahlen die Grundlage der Qualitätsberechnung. Hierzu zählen vor allem die durchschnittliche Satzlänge, die Häufigkeit der Verwendung von Adjektiven und Fachbegriffen, sowie das Verhältnis von Haupt- und Nebensätzen.
- Die Qualität der Darstellung einer Grafik wird durch Faktoren wie Auflösung, Farbtiefe, Größe sowie Kontrast und Schärfe (bei Fotos) bestimmt.
- Für Audios existiert mit der *Samplingrate* eine sehr aussagekräftige Maßzahl, die auch relativ einfach automatisch ermittelt werden kann[23].
- Für Bewegtbilder (Animationen und Videos) setzt sich das Qualitätsmaß aus Farbtiefe, Bildwiederholrate, Darstellungsgröße und *Samplingrate* der Tonspur zusammen.

Da die zugrundegelegten Qualitätskriterien vorwiegend auf formalen Aspekten wie Größe, *Samplingrate* und Verwendung bestimmter Wortarten beruhen, sind sie größtenteils mit vertretbarem Aufwand automatisch bestimmbar. Einzige Ausnahme bilden Grafiken, da hier die Bildqualität nicht zwangsläufig mit automatisch bestimmbaren Eigenschaften wie z.B. der Farbtiefe zusammenhängt.

Language

---

[22] Aber auch das *Flash*-Format enthält zumindest eine Angabe über die Ausgabegröße, die bei der Erstellung der Animation verwendet wurde.

[23] Es wird hierbei davon ausgegangen, daß die Verwendung einer hohen *Samplingrate* vor allem durch eine gute Aufnahmequalität motiviert ist.

Für textuelle Medienobjekte kann die Sprache relativ einfach und sicher entweder mit Hilfe linguistischer Methoden (z.B. Stopwort-Listen oder Erkennung sprachtypischer Buchstabenkombinationen) oder durch statistische Verfahren (z.B. Übergangswahrscheinlichkeiten von *3-Grammen* [Dunning94]) automatisch erkannt werden.

Um diese Verfahren auch auf Videos und Audios anwenden zu können, ist eine Extraktion und Analyse des gesprochenen Textes nötig. Das Problem dabei ist neben dem Herausfiltern von Hintergrundgeräuschen die unterschiedliche Geschwindigkeit, Tonhöhe und Deutlichkeit verschiedener Sprecher. Ergebnisse aus anderen Anwendungsbereichen [Christel+99, Srinivasan+99] weisen darauf hin, daß die Fehlerrate zumindest für Nachrichten und andere professionell gesprochenen Texte tolerierbar ist.

Thumbnail

Um automatisch ein *Thumbnail* für ein Video zu erzeugen, muß ein möglichst repräsentatives und aussagekräftiges Einzelbild ermittelt werden. Dies ist mit dem aktuellen Stand der Technik zumindest teilweise möglich [Girgensohn+99].

AccessProtection und Copyright

Welcher Wert dem "AccessProtection" und dem "Copyright"-Attribut zugewiesen werden muß, ergibt sich i.A. nicht aus dem Objekt, sondern aus seiner Quelle. Über eine Abbildung von Medienobjekt-Quellen auf die möglichen Attribut-Werte kann die Urheberschaft eines Objektes halbautomatisch festgeschrieben werden.

Genre

Eine mit vertretbarem Aufwand realisierbare, automatische Bestimmung des Genres eines Medienobjektes ist nur für Grafiken und Texte und auch dort nur näherungsweise anhand von Heuristiken möglich. Die Ergebnisse aller in diesem Abschnitt beschriebenen Verfahren sind daher vom Autor auch eher als Vorschlag, denn als korrekt berechnete Werte zu interpretieren.

Die Heuristik für Grafiken ist sehr einfach und dennoch akzeptabel genau: Ausgangspunkt der Überlegung ist, daß Grafiken sich fast ausschließlich auf die Genres *Content* und *AddOn* verteilen. In das Genre *Content* fallen dabei vor allem Diagramme, während Fotografien meist eher ergänzenden oder ausschmückenden Charakter haben. Eine automatische Erkennung von Diagrammen und Fotos wiederum ist über die Farbtiefe (Diagramme: gering, Fotos: hoch) und in Zweifelsfällen zusätzlich über die Größe (Diagramme: groß, Fotos: klein) und den Datentyp (Diagramme: GIF, Fotos: JPEG) möglich.

Die halbwegs genaue Erkennung des Genres eines Textes gestaltet sich erheblich schwieriger, da potentiell alle Genres mit Ausnahme von *AddOn* in Frage kommen[24].

Hilfreich für eine zumindest näherungsweise Bestimmung des Genres sind zwei Maßzahlen:

- das Verhältnis von Indexeinträgen zu Worten im Gesamttext (Indexrate)
- der Anteil von Substantiven im Volltext, die in keinem anderen, bereits im Medienobjektspeicher befindlichen, als *content* klassifizierten Text vorkommen (Exklusivrate)

Anhand dieser Maßzahlen lassen sich die einzelnen Genres wie folgt charakterisieren:

[24] Eine der wenigen, sinnvollen Verwendungen von Texten als *AddOns* sind Zitate, wie sie zuweilen an den Anfang eines Kapitels gestellt werden.

	Indexrate	Exklusivrate
content	Normal bis hoch	niedrig
document[25]	-	-
comment	niedrig	niedrig
dispatcher	sehr hoch	sehr niedrig
example	niedrig	hoch
problem	niedrig	hoch

Die konkreten Ausprägungen der angegebenen Werte "niedrig", "normal" und "hoch" sind abhängig vom Detaillierungsgrad der Indizierung und der Anzahl bereits im Medienobjektspeicher befindlicher Objekte. Ein guter Anhaltspunkt für konkrete Werte ist jedoch, daß alle Werte außerhalb des zentralen Schwankungsintervalls potentiell niedrig oder hoch sind.

Standalone

Bei Medienobjekten, die auch ohne übergeordneten Kontext - z.B. als Bestandteil eines Glossars - präsentierbar sind, handelt es sich zumeist um Definitionen oder andere sehr präzise Erklärungen. Diese zeichnen sich in den meisten Fällen durch eine hohe Indexrate und eine sehr niedrige Exklusivrate aus.

Caption

Die automatische Erzeugung einer Überschrift zu einem gegebenen Text oder Bild ist nicht trivial. Für Online Dokumente existieren zwar auf einer Analyse von referenzierenden Links basierende Verfahren [Amitay98], diese sind jedoch nicht auf neu erstellte oder aus gedruckten Dokumenten extrahierte Medienobjekte anwendbar. In diesen Fällen ist es jedoch oftmals eine gute Wahl, das mit der höchsten Relevanz vermittelte Stichwort aus dem Index eines Medienobjektes als dessen Überschrift anzunehmen.

4.5.2 Ermittlung des Vorwissens eines Stichwortes

Da angenommen wird, daß die Zielgruppe eines Lehr- oder Informationssystems für einen Großteil der Stichworte ein nahezu identisches Vorwissen besitzt, sollten alle Stichworte mit einem Default Vorwissen versehen werden. Da die Angabe des Default Vorwissens einen nicht unerheblichen Aufwand für den Autor bedeutet, ist die automatische Ermittlung dieses Wertes erstrebenswert.

Die Idee einer automatisierten Bestimmung des durchschnittlichen Vorwissens zu einem Stichwort basiert darauf, daß dieses Vorwissen um so größer ist, desto häufiger das Stichwort im normalen Sprachgebrauch Verwendung findet. Ein guter Indikator für die Häufigkeit der Verwendung einzelner Worte ist die Anzahl der Dokumente im *World Wide Web*, die dieses Wort enthalten. Eine Möglichkeit, diesen Wert zu relativ genau zu schätzen, ist die Verwendung einer Suchmaschine wie z.B. *Altavista*.

Die nachfolgende Tabelle zeigt für die in Kapitel 4.4.6. manuell beschriebenen Stichworte jeweils den manuell festgelegten Vorwissens-Wert sowie die Anzahl der deutschsprachigen Web-Dokumente, die von *Altavista* zu diesem Wort gefunden wurden[26]. Da der Wertebereich der Anzahl der gefundenen Dokumente sehr groß ist, gibt die zweite Spalte den logarithmierten Wert der ersten Spalte an.

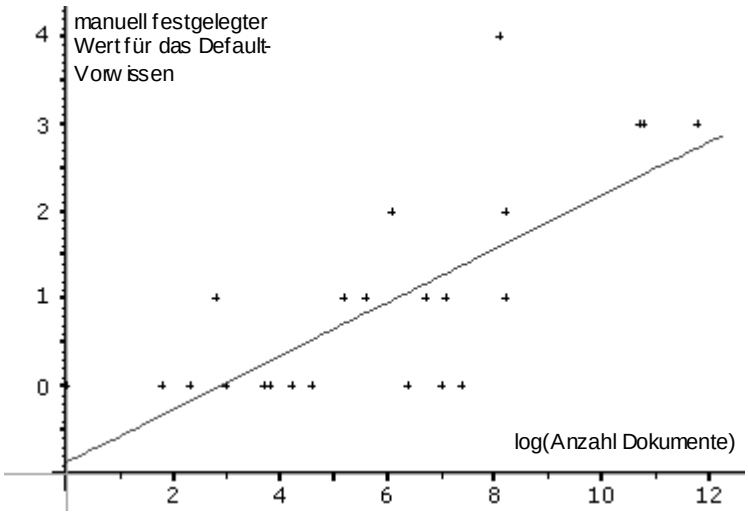
[25] Da in die Kategorie "Document" von einer Laudatio bis zu einem Vertrag jegliche Dokumente fallen können, sind die Maßzahlen Indexrate und Exklusivrate zur Erkennung von Dokumenten nicht sonderlich hilfreich. Dennoch können Dokumente mit guter Wahrscheinlichkeit dadurch erkannt werden, daß sie im Vergleich zu allen anderen Texten zumeist überdurchschnittlich lang sind.

[26] *Altavista* Deutschland (http://altavista.de/), Stand Juni 1999, nur deutschsprachige Dokumente wurden berücksichtigt.

Stichwort i	Dokumente n_i	$\log(n_i)$	manuell
Bleisatz	276	5,6	1
DTP-Programm	461	6,1	2
DTP-Punkt	10	2,3	0
Grundlinie	843	6,7	1
und runde Formen	102	4,6	0
Inch	3840	8,2	2
Kegelgröße	6	1,8	0
Kleinbuchstaben	3257	8,1	4
Kurventeile	17	2,8	1
und Mittellänge	0	0	0
Linie	440790	13,0	3
Mindestzeilenabstand	1	0	1
Mittellänge	19	3,0	0
Oberlänge	39	3,7	0
Pica	581	6,4	0
Schrift	44301	10,7	3
Schriftgrad	632	6,4	0
Schriftgröße	3550	8,2	1
Schriftlinie	45	3,8	0
Strecke	46645	10,8	3
Überhang	1086	7,0	0
Unterlänge	64	4,2	0
x-Höhe	1623	7,4	0
und runde Formen	72	4,2	0
Zeichen	129978	11,8	3
Zeichenhöhe	180	5,2	1

Abbildung 4-4 stellt zur Verdeutlichung die logarithmierten Häufigkeiten den manuell vergebenen Werten gegenüber. Die rot eingezeichnete Linie gibt die Regressionsgerade zu beiden Variablen wieder.

Abbildung 4-4: Vergleich zwischen manuell und automatisch erstellter Festlegung des Vorwissens der Stichworte aus dem Beispielindex



Auch wenn manuelle und automatische Zuweisung des Vorwissens von Stichworten nicht exakt übereinstimmen, so zeigt sich doch, daß ein Zusammenhang zwischen beiden Werten besteht (r = 0,76).

Dieser Zusammenhang kann sogar noch verstärkt werden, wenn themenbedingte Kriterien bei der Transformation der Häufigkeiten mit einfließen. Typographie ist z.B. ein auf überdurchschnittlich vielen Webseiten behandeltes Thema, so daß typographische Fachbegriffe potentiell überdurchschnittlich große Häufigkeiten haben (z.B. x-Höhe mit 1623 Nennungen). Durch eine entsprechende Normierung von Fachbegriffen können solch ungewöhnlich hohen Werte relativiert werden.

5 Nutzungsszenarien

"User" is a four-letter-word.

Sprichwort

Die Möglichkeit der Anpassung an sich dynamisch verändernde Einflußgrößen wie z.B. den aktuellen Nutzer ist einer der größten Vorteile, den die automatisierte Generierung hypermedialer Dokumente mit sich bringt. Für jeden Nutzer kann ein eigenes, individuell zusammengestelltes und strukturiertes Dokument erstellt werden.

Basierend auf der in Kapitel 4 eingeführten Notation von Medienobjekten, Stichworten und Indizes soll in diesem Kapitel dargestellt werden, wie die Anpassung an die vier in der Einleitung genannten Einflußgrößen - Nutzer, Autor, Thema und Layout - im *i4 Framework* konzeptionell realisiert wurde und welche Auswirkungen sie auf die Menge der zur Verfügung stehenden Medienobjekte hat.

5.1 Nutzungsszenarien

Bei der Erstellung von Lehr- und Informationssystemen lassen sich die konfigurierbaren Parameter der Dokumentenerstellung in vier Teilbereiche unterteilen:

1. Welche **Inhalte** sollen dargestellt werden, d.h. welche Stichworte sind mit dem generierten System erlernbar?
2. Welches **Vorwissen** bringt der Nutzer mit, d.h. welche Stichworte sind bereits bekannt und wie weitreichend sind diese Kenntnisse?
3. Welche **Einschränkungen** existieren für den Nutzer, das gewählte Layout, die darzustellende Thematik, etc.; d.h. welche Eigenschaften müssen die im generierten Dokument enthaltenen Medienobjekte zwingend erfüllen (z.B. bezüglich benötigter Bandbreite)?
4. Welche **Präferenzen** bezüglich Inhalt, Struktur und Layout haben Nutzer und Autor, d.h. welche Medienobjekte werden potentiell in welcher Zusammenstellung bevorzugt?

Die Antworten auf diese Fragen bilden ein sog. **Nutzungsszenario**, an das ein zu erstellendes Lehr- oder Informationssystem anzupassen ist. Da Präferenzen und Einschränkungen als Teil der Inhaltsbeschreibung kodiert werden, ergibt sich ein Nutzungsszenario als Tupel aus gewünschtem Inhalt (in Form von in Abschnitt 5.3 näher spezifizierten Blockbeschreibungen Blk) und über gewichtete Stichworte kodiertem, vorhandenem Vorwissen Sw:

$$U^{\#} \subseteq \mathbf{P}(\mathrm{Blk}) \times \mathbf{P}(Sw) \tag{5.1}$$

Die Menge aller möglichen Nutzungsszenarien soll im folgenden als $U^{\#}$ bezeichnet werden.

5.2 Doppelte Adaptierbarkeit

Das Nutzungsszenario stellt (neben dem Medienobjektspeicher) die Schnittstelle zwischen dem Autor und dem Benutzer auf der einen und dem dokumentenerzeugenden System auf der anderen Seite dar. Aus dieser Zwischenstellung ergeben sich eine Reihe von sich teilweise widersprechenden Anforderungen:

- Das Nutzungsszenario muß maschinell interpretierbar sein, da es vom dokumentenerzeugenden System zur Konfiguration der Auswahl- und Strukturierungsalgorithmen verwendet wird.
- Das Nutzungsszenario muß für den Autor verständlich sein, da dieser im Endeffekt für Zusammenstellung und Kodierung verantwortlich ist.
- Es muß themenunabhängig sein, d.h. sowohl sein Format, als auch die definierten Konfigurationsparameter müssen für die Erzeugung beliebiger Lehr- und Informationssysteme geeignet sein.

Die mit diesen Anforderungen verbundenen Probleme sollen im folgenden anhand eines Beispiels verdeutlicht werden:

> *Ein Autor möchte ein hypermediales Lehrsystem zur deskriptiven Statistik erstellen. Seine Zielgruppe sind Studenten der Wirtschaftswissenschaften, die an verschiedenen deutschen Universitäten die entsprechende Grundstudiumsveranstaltung besuchen. Hierbei stößt er auf das Problem, daß einige Dozenten die Varianz mit dem Vorfaktor (1/n) und andere mit (1/(n-1)) definieren. Er erweitert daher die Meta-Informationen aller Medienobjekte um ein anwendungsabhängiges Attribut "Varianz", in dem für jedes Objekt die zugrundegelegte Definition der Varianz angegeben ist.*
>
> *Damit der Bottom-Up Algorithmus bei der Auswahl nur Medienobjekte berücksichtigt, die auf einer der beiden Definitionen basieren, muß der Nutzer die gewünschte Definition angeben. Hierzu muß ein entsprechendes einschränkendes Attribut im Nutzungsszenario definiert sein, das mit dem entsprechenden Attribut in der Medienobjekt-Beschreibung verglichen werden kann*[27]*.*
>
> *Leider wissen die Studenten jedoch vor der Lektion über die Varianz gar nicht, daß es überhaupt zwei Definitionen gibt, und können somit nicht qualifiziert entscheiden, welche davon kompatibel mit der von ihnen besuchten Lehrveranstaltung ist. Aus diesem Grunde müßten die Studenten im Nutzunsszenario nicht die gewünschte Definition der Varianz, sondern vielmehr die von ihnen besuchte Universität angeben. Aufgabe des Autors wäre es dann, eine Abbildung der besuchten Universität auf die dort verwendete Definition der Varianz herzustellen.*

Dieses Beispiel zeigt, daß die beiden Einflußgrößen "Autor" und "Thema" von einem System zur Erstellung von dynamischen Dokumenten auf zwei Abstraktionsebenen berücksichtigt werden müssen:

1. Die für den Nutzer sichtbare Anpassung geschieht anhand einer Auswahl von Optionen, die von den Einflußgrößen "Thema" und/oder "Autor" vorgegeben sind (**Nutzeranpassung**). Im Beispielfall ist dies z.B. die durch das behandelte Thema bedingte Einschränkung der auswählbaren Medienobjekte auf die Verwendung einer bestimmten Definition der Varianz.
2. Die für das Autorensystem sichtbare Anpassung besteht in der Beschreibung des Einflusses der vom Nutzer gewählten, von den Charakteristika des behandelten Themas abhängigen, Optionen auf Inhalt, Struktur und Layout des zu erzeugenden Dokuments (**Thematische Anpassung**). Im Beispiel muß z.B. das Generierungssystem so konfiguriert werden können, daß es das themenabhängige, vom Autor definierte Medienobjekt-Attribut "Definiton der Varianz" bereitstellt und interpretiert.

Um die mit verschiedenen Abstraktionsebenen und thematischen Abhängigkeiten verbundenen Probleme zu lösen, wurde im Rahmen des *i4 Frameworks* das in Abbildung 5-1 dargestellte Modell der **doppelten Adaptierbarkeit** entwickelt.

[27] zur konkreten Beschreibung anwendungsabhängiger Attribute siehe Kapitel 7.

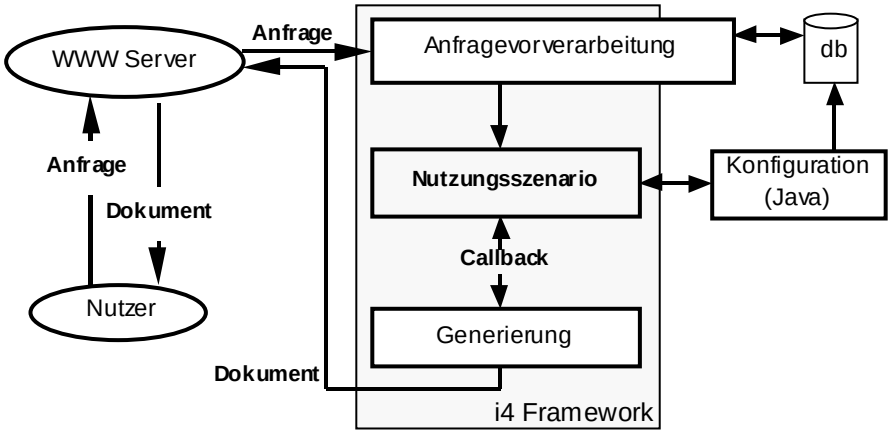


Abbildung 5-1. Übersetzung einer Anfrage in ein Nutzungsszenario

Grundidee der doppelten Adaptierbarkeit ist, die Nutzeranpassung als Teil der thematischen Anpassung zu realisieren. Die **Konfiguration** des Generierungssystems kapselt die thematische Anpassung. Teil der Konfiguration ist die **Anfragevorverarbeitung**, über die die Nutzeranpassung vorgenommen wird. Die Anfragevorverarbeitung wiederum legt die Parameter der Konfiguration fest. Die Schnittstelle zwischen beiden Anpassungsebenen bildet das Nutzungsszenario, das über *Callback*-Funktionen auch an die Algorithmen zur Dokumentengenerierung weitergereicht wird (siehe Kapitel 8).

Auf das Beispiel bezogen bedeutet dies:

> Die vom Autor erstellte, themenabhängige Konfiguration erweitert das *i4 Framework* um ein Attribut "Definition der Varianz". Darüber hinaus enthält die Konfiguration eine angepaßte Auswahlfunktion, die entsprechend dem Wert des neuen Attributs die Auswahl der geeigneten Medienobjekte steuert. Die ebenfalls vom Autor erstellte Anfragevorverarbeitung bildet den Namen einer Universität auf einen konkreten Attribut-Wert ab und reicht diesen als Teil des Nutzungsszenario an die Konfiguration weiter.

Hauptaufgabe der Anfragevorverarbeitung ist somit die Abbildung der vom Nutzer gesandten Anfrage auf ein für das Generierungssystem interpretierbares Nutzungsszenario. Die Konfiguration der Dokumentenerzeugung ist kein integraler Bestandteil dieser selbst, sondern extern durch eine Anzahl von **Java Skripten** realisiert. Diese *Java* Skripte werden vom *i4 Framework* aufgerufen und erlauben neben der Adaption von Konfigurationsparametern auch das Überschreiben kompletter Bewertungsfunktionen und Algorithmen. Über das Nutzungsszenario werden von der Anfragevorverarbeitung somit lediglich vom Autor bereitgestellte *Java* Skripte parametrisiert, die ihrerseits die Dokumentenerzeugung konfigurieren. Auch die Anfragevorverarbeitung ist nur teilweise außerhalb des *Frameworks* angesiedelt, um die Abbildung einer konkreten Anfrage auf ein Nutzungsszenario möglichst flexibel gestalten zu können.

Durch diese zweistufige Konfigurations-Hierarchie können anwendungsbezogene Parameter in der externen Konfiguration gekapselt und auf Parameter der Auswahl- und Strukturierungsalgorithmen abgebildet, bzw. durch den Java Skript Mechanismus sogar in das Framework integriert werden. Die doppelte Adaptierbarkeit ist Grundvoraussetzung für die Erstellung eines flexiblen Autorensystems. Systeme wie z.B. ELM-ART [Weber+97] bieten nur die einfache Adaptierbarkeit an den Nutzer und sind daher sehr stark an ein Thema gebunden. Aus diesem Grund sind sie eher als Anwendungen zu sehen, denn als Werkzeuge.

Weitere Vorteile der zwei-stufigen Konfigurations-Hierarchie sind:

- Die Anfragevorverarbeitung erlaubt die Abstraktion von einer konkreten Anfragesyntax. Eine Anfrage kann somit sowohl über ein CGI-Skript in Form von Name-Wert-Paaren als auch natürlich-sprachlich gestellt werden.
- Die Abbildung einer Anfrage auf ein Nutzungsszenario kann von weiteren, vom Autor des Dokuments festgelegten, Faktoren beeinflußt werden, wie z.B. der Uhrzeit (z.B. "zwischen 10 und 18 Uhr werden keine Grafiken und Videos eingebettet").
- Informationen über die für einen Nutzer bereits erzeugten Dokumente können in einer Datenbank verwaltet und bei der Generierung weiterer Dokumente berücksichtigt werden.

Die in der Anfragevorverarbeitung durchgeführten Abbildungen können beliebig komplex sein. Grundidee ist es dabei immer, abstrakte Anfrageparameter des Benutzers auf konkrete Konfigurationsparameter umzusetzen.

5.3 Blöcke

Die Kapitel der logischen Struktur eines Lehr- oder Informationssystems können anhand ihrer Genres voneinander abgegrenzt werden (siehe Kapitel 2.4). Kapitel des gleichen Genres lassen sich dabei zu sog. **Blöcken** zusammenfassen. Ein Lehrsystem, das aus drei inhaltlichen Kapiteln und einem Glossar besteht, setzt sich somit aus zwei Blöcken - Wissensvermittlung und Glossar - zusammen.

Jeder Block zeichnet sich durch die Eigenschaften

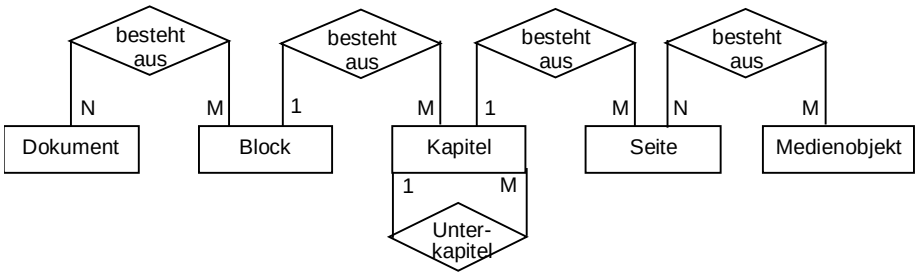
- Genre
- vermittelte Informationen
- Eigenschaften der benutzten Medienobjekte
- Präferenzen bei der Auswahl der Medienobjekte
- Struktur (sequentiell, hierarchisch, etc.) und
- Einbettung in das Gesamtdokument

aus.

Ein Block zur Wissensvermittlung ist z.B. in den meisten Fällen hierarchisch strukturiert und basiert auf Medienobjekten der Genres *Wissensvermittlung*, *Aufgabe* und *Beispiel*. Ein Glossar hingegen ist sequentiell strukturiert, besteht aus alleinstehenden Medienobjekten und ist im Idealfall in sich abgeschlossen, d.h. es kann auch ohne das restliche Dokument existieren.

Blöcke werden ausschließlich dazu verwendet, eine genre-abhängige Modellierung von Dokumentstrukturen zu ermöglichen. Im Ebenenmodell von Lehr- und Informationssystemen sind sie der Kapitelstruktur übergeordnet:

Abbildung 5-2: Blöcke als Bestandteil des Ebenen-Modells



Da jeder Block verschiedenen Dokumenten zugeordnet werden kann, sind Blöcke die größten, wiederverwendbaren Einheiten der Dokumentenerzeugung.

5.3.1 Blöcke als Inhaltsbeschreibung

Blöcke bilden innerhalb eines Nutzungsszenarios die Basis der inhaltlichen Beschreibung des zu erzeugenden Dokuments. Sie bilden das erste Element jedes ein Nutzungsszenario beschreibenden Tupels:

$u\_blocks: U^{\#} \rightarrow \mathbf{P}(Blk)$, mit

$u\_blocks((b, w)) := b$ (5.2)

Definition

Jeder **Block** b wird über ein Tupel aus Genre, zu vermittelnden Stichworten, zur Informationsvermittlung zu verwendenden Medienobjekten, zusätzlichen Medienobjekten und (potentiell themenabhängigen) frei definierbaren Attributen beschrieben. Die Menge aller Blockbeschreibungen wird mit **Blk** bezeichnet:

$$Blk \subseteq string \times \mathbf{P}(S^{avail}) \times \mathbf{P}(M^{avail}) \times \mathbf{P}(ZO) \times \mathbf{P}(string \times string) \quad (5.3)$$

Auf die einzelnen Bestandteile eines Blocks kann über die folgenden Projektionen zugegriffen werden:

$$b\_genre((g, c, b, a, f)) := g \quad (5.4a)$$

$$b\_content((g, c, b, a, f)) := c \quad (5.4b)$$

$$b\_base((g, c, b, a, f)) := b \quad (5.4c)$$

$$b\_additional((g, c, b, a, f)) := a \quad (5.4d)$$

$$b\_flags((g, c, b, a, f)) := f \quad (5.4e)$$

Das Genre eines Blocks wird über eine einfache Zeichenkette angegeben. Die aktuelle Implementierung des *i4 Frameworks* unterstützt lediglich die Genres *Wissensvermittlung* ("content") und *Glossar* ("glossary"), weitere Genres können jedoch themenabhängig vom Autor des Lehr- oder Informationssystems definiert werden.

5.3.2 Basisobjekte und Zusatzobjekte

Die nächsten drei Bestandteile der Blockbeschreibung - zu vermittelnde Stichworte, dazu zu verwendende Medienobjekte und zusätzliche Medienobjekte - kodieren Informationen über den konkreten Inhalt und die Zusammensetzung eines Blocks.

Die Menge der zu vermittelnden Stichworte - auch als **Lernziele** bezeichnet - ist eine Untermenge der im Gesamtindex enthaltenen Stichworte. Bei der Generierung eines Blocks wird versucht, diesen so zu erzeugen, daß er die gewünschten Stichworte vermittelnde Medienobjekte enthält.

Die Menge der **Basisobjekte** gibt an, welche Medienobjekte zur Vermittlung der gewünschten Inhalte verwendet werden können. Die Basisobjekte sind immer eine Teilmenge der insgesamt zur Verfügung stehenden Medienobjekte. In die Festlegung der zur Generierung eines Blockes verfügbaren Medienobjekte fließen auch die in der Einleitung von Kapitel 5 erwähnten Einschränkungen wie z.B. die verfügbare Bandbreite ein.

Theoretisch reichen diese beiden Angaben - Basisobjekte und Lernziele - aus, um Zusammensetzung und Inhalt eines Blocks zu beschreiben. In der Praxis tritt jedoch das Problem auf, daß Medienobjekte einiger Genres vom Auswahlalgorithmus so gut wie nie berücksichtigt werden und darüber hinaus ihre Einbettung in die strukturellen Ebenen nur anhand zusätzlicher Informationen möglich ist. Ein Beispiel für diese Problematik sind Aufgaben:

> Aufgaben vermitteln im Allgemeinen kein neues Wissen, d.h. sie sind für die Vermittlung der gewünschten Inhalte im Grunde genommen irrelevant. Damit sie dennoch Bestandteil des generierten Dokuments sind, muß der Auswahlalgorithmus so konfiguriert werden, daß er neben einem Medienobjekt zur Vermittlung eines gewünschten Stichwortes zusätzlich ein redundantes, aber dazu passendes Medienobjekt vom Genre "Aufgabe" auswählt. Bei der Strukturierung des Dokuments ist es sinnvoll, Aufgaben nicht an Medienobjekte zu binden, sondern sie vielmehr als Bestandteil der Seitenstruktur zu sehen (z.B. eine Aufgabenseite am Ende jedes Kapitels).

Um den Auswahl- und Strukturierungsalgorithmen die zur adäquaten Berücksichtigung nicht-wissensvermittelnder Medienobjekte benötigten Information bereitzustellen, beinhaltet die Beschreibung eines Blocks Angaben über **zusätzlich zu verwendende Medienobjekte**. Als zusätzlich zu verwendende Medienobjekte gelten dabei Objekte, die nicht primär anhand ihres vermittelten Wissens, sondern aufgrund bestimmter Quoten ausgewählt werden sollen bzw. die nicht an beliebiger Stelle in die

Medienobjektstruktur eingebettet werden können. Beispiele für Zusatzobjekte sind Fotos, Beispiele, Aufgaben, etc.

Für jeden Block können beliebig viele Genres als zusätzlich verwendbare Objekte angegeben werden. Die Angaben, die hierbei für jedes Genre gemacht werden müssen, umfassen:

- Name des Genres,
- den Namen einer strukturellen Einheit (Medienobjekt, Seite, Kapitel, Block), an der sich die Auswahl orientiert,
- die Anzahl der Objekte des angegebenen Genres, die im Verhältnis zur gebundenen Einheit ausgewählt werden sollen (**Genre-Häufigkeit**) ,
- die maximale Häufigkeit mit der ein Zusatzobjekt ausgewählt werden kann (**Objekt-Häufigkeit**) und
- die Art der strukturellen Einbettung (beim Medienobjekt, als gesonderte Seite am Anfang/Ende eines Kapitels, als gesondertes Kapitel am Anfang/Ende des Blocks).

Um zum Beispiel eine Ebenen-Struktur wie in "Mathe online" (Kapitel 2.4) zu erzeugen, müssen Beispiele, Wissenswertes und Historisches enthaltende Medienobjekte als Zusatzobjekte mit hoher Genre-Häufigkeit, einer Objekt-Häufigkeit von 1 und als eigene Seite integrierbar definiert werden.

Jede Menge zusätzlich verwendbarer Objekte wird innerhalb einer Blockbeschreibung als Tupel aus drei Zeichenketten (Genre, Bindung, Einbettung) und zwei Zahlen (Genre- und Objekt-Häufigkeit) angegeben:

$$ZO \subseteq string \times Z_{base} \times IR \times IN \times Z_{struct} \tag{5.5}$$

mit Z_{base} = { "mo", "frame", "chapter", "block" }

und Z_{struct} = { "mo", "chapStart", "chapEnd", "blockStart", "blockEnd" }

5.3.3 Eigenschaften von Blöcken

Neben den zu vermittelnden Inhalten und den verwendbaren Medienobjekten existieren noch weitere Kriterien zur Beschreibung von Blöcken, die in Form von Name-Wert-Paaren kodiert werden. Die wichtigsten dieser Kriterien sind:

- das Aussehen der Seitenstruktur des Blockes,
- seine Einbettung in das Gesamtdokument,
- seine Kompaktheit und
- Präferenzen zur Auswahl von Medienobjekten.

Die ideale Seitenstruktur eines Blockes ist sehr stark von seinem Genre und der Nutzung des Dokuments abhängig. Mit den in Kapitel 7 vorgestellten Algorithmen zur Auswahl und Strukturierung von Medienobjekten ist es möglich, eine Reihe verschiedener **Strukturen** zu erzeugen:

Sequentiell: Die Seitenstruktur ist möglichst sequentiell, d.h. bis auf die erste und letzte Seite hat jede Seite genau einen Vorgänger und einen Nachfolger. Diese Struktur ist vor allem für Genres wie Glossar und Zeitleiste geeignet, sowie für Nutzer, die das erzeugte Dokument *offline* als Ausdruck lesen wollen.

Hierarchisch: Die Seitenstruktur ist baumähnlich aufgebaut, d.h. jede Seite kann beliebig viele Nachfolger, aber nur einen Vorgänger haben. Hierarchische Seitenstrukturen sind bei fast allen existierenden Lehr- und Informationssystemen anzutreffen (siehe Kapitel 2.1).

Vernetzt: Bei einer vernetzten Seitenstruktur bestehen keinerlei Beschränkungen bezüglich der Anzahl der Vorgänger und Nachfolger einer Seite. Lediglich zyklische Strukturen sollten nach Möglichkeit vermieden werden. Vernetzte Seitenstrukturen unterstützen vor allem das explorativen Erlernen

eines Themas, werden aber auch oft als Ursache des *lost-in-hyperspace* Syndroms genannt. Einige Beispiel für den Einsatz vernetzter Strukturen sind in [Caumanns+99c] beschrieben.

Zu beachten ist, daß es sich bei der gewünschten Struktur um eine Präferenz handelt, d.h. es wird lediglich versucht, eine solche Struktur zu erzeugen. Ob dies überhaupt möglich ist, hängt ausschließlich von den zur Verfügung stehenden Medienobjekten und den zu vermittelnden Stichworten ab.

In direktem Zusammenhang mit der Struktur des zu erzeugenden Dokuments steht seine **Kompaktheit**. Ein Dokument ist um so kompakter, je kürzer die Pfade des zugrundeliegenden Medienobjekt-Graphen sind. Die für einen Block gewünschte Kompaktheit kann in einem Attribut namens "compact" als numerischer Wert zwischen 0 und 1 kodiert werden, wobei 1 für größtmögliche Kompaktheit steht.

Sobald ein Dokument aus mehr als einem Block besteht, stellt sich die Frage, wie diese zueinander in Beziehung stehen. Interessant ist hierbei vor allem, ob von einem Block vermittelte Stichworte bei der Generierung eines anderen Blocks als bekannt vorausgesetzt werden sollen und ob **Hyperlinks zwischen Blöcken** existieren können.

Diese für die automatische Generierung wichtigen Informationen werden in dem Attribut "export" als Teil der Eigenschaften eines Blockes festgelegt.

Das Attribut "export" legt fest, welche Informationen eines Blocks nach außen sichtbar und somit für andere Blöcke referenzierbar sind:

nothing: weder die enthaltenen Medienobjekte noch die vermittelten Stichworte dieses Blocks sind für andere Blöcke sichtbar.

mediaobjects: alle in dem Block enthaltenen Medienobjekte sind sichtbar und können von anderen Blöcken per Hyperlink referenziert werden.

all: alle Medienobjekte und alle vermittelten Stichworte sind sichtbar und werden bei der Generierung weiterer Blöcke als bekannt vorausgesetzt.

Die mögliche Referenzierung von Objekten und Stichworten aus anderen Blöcken läßt sich nur effektiv realisieren, wenn bei der Generierung der Blöcke eine feste Reihenfolge zugrundegelegt wird. Hierzu ein Beispiel:

> Es soll ein Informationssystem aus zwei Blöcken - Wissensvermittlung und Glossar - erstellt werden. Das Attribut "export" des Glossars wird auf den Wert "mediaobjects" gesetzt und bei der Erstellung mit dem Glossar begonnen.
>
> Die Konfiguration bewirkt, daß zuerst das Glossar und anschließend der inhaltliche Block erzeugt werden. Medienobjekte, die in der Medienobjektstruktur beider Blöcke enthalten sind, werden aus dem Inhaltsblock entfernt und durch Referenzen auf das Glossar ersetzt. Bei umgekehrter Erstellungsreihenfolge und gleicher Attributierung würde keine gegenseitige Referenzierung stattfinden. Bei umgekehrter Erstellungsreihenfolge und umgekehrter Attributierung würden doppelte Medienobjekte aus dem Glossar entfernt und durch Referenzen in den Inhaltsblock ersetzt.

Als Default ist im *i4 Framework* die Erstellungsreihenfolge "Glossar - Wissensvermittlung - Sonstige" implementiert. Der Autor des Dokuments hat jedoch die Möglichkeit, bei Bedarf auch eine andere Reihenfolge festzulegen.

Neben den hier beschriebenen Attributen "structure", "compact" und "export" können beliebige weitere Attribute an einen Block gebunden werden. Diese Attribute sind für das *i4 Framework* transparent, d.h. sie werden von den internen Algorithmen nicht interpretiert. Der Auswahlalgorithmus erlaubt jedoch die Verwendung externer, in Java kodierter Bewertungsfunktionen. Hierdurch können in weiteren Attributen eines Blocks Hinweise für bestimmte Präferenzen bei der Medienobjekt-Auswahl kodiert und durch externe, anwendungs- und nutzerbezogene Bewertungsfunktionen berücksichtigen werden.

5.4 Vorwissen

Über den Index eines Medienobjektes wird beschrieben, welche Stichworte bekannt sein müssen, um das Medienobjekt zu verstehen, bzw. welche Stichworte von diesem Objekt neu vermittelt werden.

Alle Indexeinträge sind nutzer-unabhängig, da sie das Vorwissen eines Nutzers über die indizierten Stichworte nicht in Betracht ziehen. Diese Vereinfachung ist jedoch bei der Generierung eines nutzerangepaßten Informationssystems nicht mehr zulässig. Stichworte, die dem Benutzer bereits bekannt sind, werden von dem Medienobjekt nicht neu vermittelt, d.h. sie dürfen auch nicht Bestandteil des vermittelten Wissens sein. Auch aus dem verlangten Vorwissen müssen alle dem aktuellen Nutzer bekannten Stichworte entfernt werden, da sie für die Auswahl und Strukturierung der geeignetsten Medienobjekte irrelevant sind.

Um die Anpassungen der vorhandenen Medienobjekte an den aktuellen Nutzer durchführen zu können, beinhaltet jedes Nutzungsszenario Informationen über das vorauszusetzende Vorwissen des aktuellen Nutzers.

Das **Vorwissen** eines Nutzers über ein Stichwort setzt sich aus dem Stichwort und dem Maß an Vorwissen über das Stichwort zusammen. Alle so gewichteten Stichworte bilden die Menge S^w:

$$S^w \subseteq S^{avail} \times R \qquad (5.6)$$

$sw\_kw: S^w \rightarrow S^{avail}$, mit $sw\_kw((s, x)) := s$

$sw\_val: S^w \rightarrow R$, mit $sw\_val((s, x)) := x$

Das in einem Nutzungsszenario kodierte Vorwissen wird als u_pre(u) bezeichnet und besteht aus einer Menge von gewichteten Stichworten:

$u\_pre: U^{\#} \rightarrow \mathbf{P}(S^w)$

$u\_pre((b, w)) = w$

Die Modellierung des Vorwissen anhand der im Gesamtindex vorhandenen Stichworte ist nur mit Hilfe der Anfragevorverarbeitung realisierbar. Da es nicht praktikabel ist, vom Nutzer zu verlangen, daß er zu jedem Stichwort seinen genauen Wissenstand angibt, ist eine Abbildung von abstrakten Vorgaben wie z.B. "gute mathematische Kenntnisse" auf konkrete Stichworte unabdingbar.

5.5 Anpassung an ein Nutzungsszenario

Die in den Meta-Daten einzelner Medienobjekte enthaltenen Informationen sind unabhängig vom aktuell gültigen Nutzungsszenario. Die einzige - allerdings sehr wichtige - Ausnahme bilden die einem Medienobjekt zugeordneten Indexeinträge.

Wie bereits im vorangegangenen Abschnitt beschrieben, kann ein Indexeintrag nur dann zum verlangten Vorwissen eines Medienobjektes gehören, wenn das verlangte Wissen größer ist als das vorhandene Wissen des aktuellen Nutzers. Analog darf ein Indexeintrag nur dann als vermittelt bezeichnet werden, wenn der Grad des vermittelten Wissens größer ist als das beim aktuellen Nutzer ohnehin schon vorhandene Wissen über das referenzierte Stichwort.

Um diese Eigenschaften des Index zu gewährleisten, müssen vor der eigentlichen Generierung eines Lehr- oder Informationssystems alle Indexeinträge mit dem für die Generierung gültigen Nutzungsszenario abgeglichen werden. Um diesen Abgleich durchführen zu können, muß zu jedem Stichwort s der Grad des existierenden Vorwissens $\sigma_{exist}(s, u)$ in Abhängigkeit vom aktuellen Nutzungsszenario u_c bestimmt werden:

$$\sigma_{exist}: S^{avail} \times U^{\#} \rightarrow R \qquad (5.7)$$

$$\sigma_{exist}(s, u) := \begin{cases} r, \text{wenn } (s,r) \in u\_pre(u) \\ \sigma_{know}(s) \quad , \text{sonst} \end{cases}$$

Anschließend wird ein neuer, an das aktuelle Nutzungsszenario angepaßter Index I^u erstellt, in dem nur noch die oben genannten Bedingungen erfüllende Indexeinträge enthalten sind:

$$I^u := \{ i \in I^{avail} \mid prv(i) > \sigma_{exist}(kw(i), u_c) \} \tag{5.8a}$$

Die Anpassung des Index kann zur Folge haben, daß danach einzelne Stichworte oder Medienobjekte mit keinem Indexeintrag verknüpft sind. Diese Objekte sind für die zur Auswahl und Strukturierung von Medienobjekten eingesetzten Algorithmen uninteressant, so daß auch die Mengen der verfügbaren Medienobjekte und Stichworte an das Nutzungsszenario angepaßt werden müssen[28]:

$$S^u := \{ s \in S^{avail} \mid \exists i \in I^u: kw(i) = s \} \tag{5.8b}$$

$$M^u := \{ m \in M^{avail} \mid \exists i \in I^u: mo(i) = m \} \tag{5.8c}$$

Da Blöcke die Basis der Inhaltsbeschreibung des zu erzeugenden Lehr- oder Informationssystems bilden, sind für die in den folgenden Kapiteln beschriebenen Auswahlalgorithmen jedoch weniger die insgesamt, als vielmehr die für die Erzeugung eines einzelnen Blocks zur Verfügung stehenden Medienobjekte und Stichworte von Interesse.

Die für die Generierung eines Blocks b verfügbaren Medienobjekte M^b ergeben sich aus den in den Basisobjekten von b enthaltenen insgesamt verfügbaren Medienobjekten M^u:

$$M^b := M^u \cap b\_base(b) \tag{5.9a}$$

Diese Einschränkung der Medienobjektmenge führt dazu, daß auch die Mengen der bei der Generierung eines Blocks zu berücksichtigenden Indexeinträge und Stichworte potentiell nur eine Teilmenge der insgesamt verfügbaren Indexeinträge bzw. Stichworte sind:

$$I^b := \{ i \in I^u \mid mo(i) \in M^b \} \tag{5.9b}$$

$$S^b := \{ s \in S^u \mid \exists i \in I^b: kw(i) = s \} \tag{5.10}$$

5.6 Beispiel für die Anpassung an ein Nutzungsszenario

Die Definition eines Nutzungsszenarios sowie die Berechnung der verfügbaren Medienobjekte, Stichworte und Indexeinträge soll im folgenden anhand eines Beispiels verdeutlicht werden.

Ausgangspunkt des Beispiels sind die in Kapitel 4 angegebenen sechs Medienobjekte zum Thema "Typographie". Das in diesem Beispiel zu berücksichtigende Nutzungsszenario läßt sich vereinfacht wie folgt beschreiben:

> Aufgabe des Generierungssystems ist es, aus den existierenden Medienobjekten ein Lehrsystem zu erstellen, in dem die wichtigsten Stichworte der Typographie erklärt werden. Das Lehrsystem soll aus einem einzigen Inhaltsblock bestehen und hierarchisch strukturiert sein. Das Vorwissen des aktuellen Benutzers besteht aus rudimentären Kenntnissen von DTP-Programmen.

Das zur Beschreibung dieser Informationen benötigte Nutzungsszenario u_c setzt sich wie in Kapitel 5.1 beschreiben aus den gewünschten Inhalten, sowie dem vorhandenen Vorwissen zusammen (siehe Definition 5.1):

$u_c = (Inhalt_c, Vorwissen_c)$

Die gewünschten Inhalte bestehen in diesem Fall aus einem einzigen Block b_i, in dem alle vorhandenen Informationen (mit Ausnahme des Vorwissens) kodiert werden (siehe Definition 5.3):

$b_i = (genre_i, Stichworte_i, Basisobjekte_i, Zusatzobjekte_i, Eigenschaften_i)$

[28] siehe auch Kapitel 4.4.4

Das Genre des zu erzeugenden Blocks ist wissensvermittelnd, und die einzige anzugebende Eigenschaft ist die gewünschte Struktur. Da keine Angaben bezüglich der Kompaktheit verfügbar sind, wird von einem Default-Wert ausgegangen. Zusätzlich wird festgelegt, daß keinerlei Informationen aus diesem Block für andere Blöcke sichtbar sind:

$genre_i$ = "content"

$Eigenschaften_i$ = { ("structure", "tree"), ("compact", "0.5"), ("export", none") }

Als Basisobjekte sollen alle verfügbaren Medienobjekte bereitgestellt werden. Zusätzliche Objekte sind weder vorhanden, noch gewünscht:

$Basisobjekte_i = M^{avail}$

$Zusatzobjekte_i = \varnothing$

Die von dem Block zu vermittelnden Informationen sind sehr unpäzise als "wichtigste Begriffe der Typographie" umschrieben. Es ist daher Aufgabe der Anfragevorverarbeitung, diese vage Angabe auf eine Menge konkret zu vermittelnder Stichworte umzusetzen. Unter der Annahme, daß außer den sechs beschriebenen keine weiteren Medienobjekte existieren, könnten dies z.B. die Stichworte "Grundlinie", "Kegelgröße", "Mittellänge", "Oberlänge", "Pica", "Schriftgrad", "Überhang", "Unterlänge" und "x-Höhe" sein.

In der Blockbeschreibung werden die zu vermittelnden Stichworte von der Anfragevorverarbeitung durch ihre zugehörigen Indizes in S^{avail} ersetzt:

$Stichworte_i$ = {4, 7, 13, 14, 15, 17, 21, 22, 23 }

Die komplette Beschreibung des zu erzeugenden Blocks b_i hat damit folgendes Aussehen:

b_i = ("content", {4, 7, 13, 14, 15, 17, 21, 22, 23 }, M^{avail}, ∅, { ("structure", "tree"), ("compact", "0.5"), ("export", none") })

Über das Vorwissen des aktuellen Nutzers ist ebenfalls nur eine sehr vage Angabe verfügbar. Auch hier ist es Aufgabe der Anfrage-Vorverarbeitung, eine Abbildung auf konkrete Stichworte herzustellen. Im Beispielfall würde es sich anbieten, für die beiden mit DTP-Software zusammenhängenden Stichworte "DTP-Programm" (Index=2) und "DTP-Punkt" (Index=3) das Vorwissen des Nutzers jeweils auf "Grundwissen" ($r_{0.5}$=2) zu setzen:

$Vorwissen_c$ = {(2, 2), (3, 2)}

Das komplette Nutzungsszenario sieht damit wie folgt aus:

u_c = ({("content", {4, 7, 13, 14, 15, 17, 21, 22, 23 }, M, ∅, { ("structure", "tree"), ("compact", "0.5"), ("export", none") })} , {(2, 2), (3, 2)})

Über dieses Nutzungsszenario können anschließend die Indexeinträge aller Medienobjekte an das Vorwissen des aktuellen Nutzers angeglichen werden. Für die meisten Stichworte wird dabei das Default-Vorwissen verwendet; lediglich für die Stichworte 2 und 3 kann auf ein explizit angegebenes Vorwissen zurückgegriffen werden.

Die nachfolgenden Abbildungen zeigen die Graph-Notation der ersten beiden Medienobjekte, wobei durch die Anpassung an das Nutzungsszenario entfernte Indexeinträge und Stichworte hellgrau dargestellt sind.

Abbildung 5-3: Graph-Notation von Medienobjekt 1 nach der Anpassung an das Nutzungsszenario

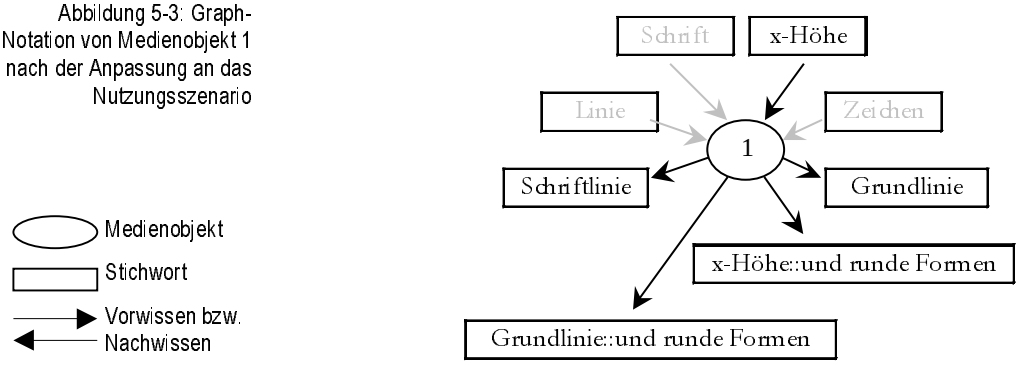
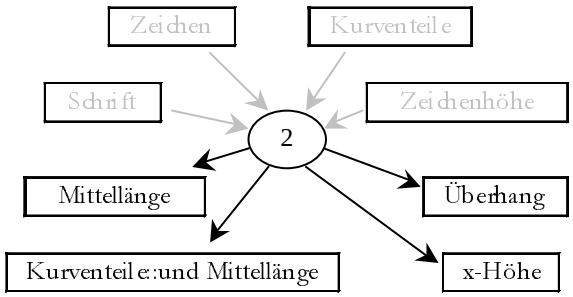


Abbildung 5-4: Graph-Notation von Medienobjekt 2 nach der Anpassung an das Nutzungsszenario



Durch die Anpassung wird die Anzahl der Indexeinträge von 38 auf 24 reduziert. Alle Medienobjekte sind jedoch noch Bestandteil mindestens eines verbleibenden Indexeintrags, so daß M^u identisch mit der Menge der vorhandenen Medienobjekte M^{avail} ist.

Im Gegensatz zu den Medienobjekten verlieren jedoch 10 der 26 Stichworte alle Referenzen zu einem Medienobjekt, so daß die Menge der an das Nutzungsszenario angepaßten Stichworte nur noch 16 Einträge umfaßt:

$$S^u = \{ 3, 4, 5, 7, 10, 12, 13, 14, 15, 17, 18, 19, 21, 22, 23, 24 \}$$

Durch die abschließende Anpassung an den zu erzeugenden Block ergeben sich keine weiteren Änderungen, da dessen Basisobjekte alle verfügbaren Medienobjekte umfassen.

6 Der Abhängigkeitsgraph

Until you've seen some of the rest, you can't make sense of any part.

Marvin Minsky, 1988

In Kapitel 3 wurden vorwiegend Eigenschaften einzelner Medienobjekte betrachtet. Erst durch gleichzeitige Betrachtung mehrerer Medienobjekte lassen sich jedoch inhaltliche und strukturelle Eigenschaften von Medienobjekten ermitteln, die über die vom Autor gemachten Angaben hinausgehen.

In diesem Kapitel werden die wichtigsten Datenstrukturen und Algorithmen beschrieben, die der automatischen Erzeugung von hypermedialen Dokumenten zugrundeliegen. Diese Datenstrukturen und Algorithmen sind immer an die inhaltliche Beschreibung eines Blocks gebunden, weswegen ihnen die im vorangegangenen Kapitel beschriebenen, an ein Nutzungsszenario angepaßten Mengen M^b, S^b und I^b zugrunde liegen.

6.1 Aggregation von Medienobjekt-Graphen

Basierend auf der Graphendarstellung eines einzelnen Medienobjektes können auch Mengen von Medienobjekten - und hierbei insbesondere die Menge aller für die Generierung eines Blocks b verfügbaren Medienobjekte M^b - durch Graphen dargestellt werden.

Definition

Der bipartite, gerichtete **Abhängigkeitsgraph G(M,I)** einer Menge von Medienobjekten $M \subseteq M^{avail}$ und einer Menge von Indexeinträgen $I \subseteq I^{avail}$ ist definiert als die Aggregation der Medienobjektgraphen aller $m \in M$ innerhalb von I:

$$G(M,I) := \left(M, \bigcup_{m \in M} \mu_{kw}(m,I), \bigcup_{m \in M} \mu_{req}(m,I), \bigcup_{m \in M} \mu_{prv}(m,I)\right) \qquad (6.1)$$

Die Menge aller in einem Abhängigkeitsgraphen G enthaltenen Medienobjekte wird als M(G) bezeichnet. Die Notationen für die weiteren Bestandteile des aggregierten Graphen sind:

- I(G) für die Menge der in G enthaltenen Indexeinträge,
- $I^{req}(G)$ bzw. $I^{prv}(G)$ für die Menge der Vorwissen bzw. Wissensvermittlung repräsentierenden Indexeinträge und
- S(G) für die in G enthaltenen Stichworte.

Aus dieser Definition ergibt sich der Abhängigkeitsgraph $G^b = G(M^b,I^b)$ der Menge aller für die Generierung eines Blocks b zur Verfügung stehenden Medienobjekte und Indexeinträge als:

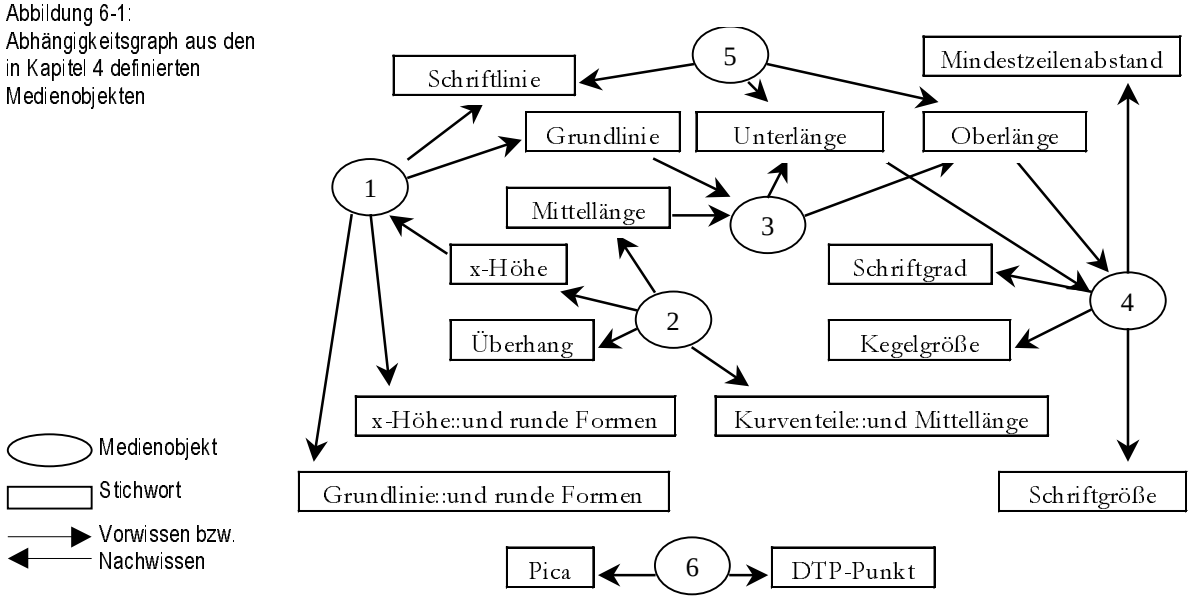
$$G^b = G(M^b,I^b) = (M^b, S^b, I^{req}, I^{prv}) \quad (6.2)$$

mit $I^{req} = \{ i \in I^b \mid req(i) > 0 \}$

und $I^{prv} = \{ i \in I^b \mid req(i) < prv(i) \}$

Abbildung 6-1 zeigt den vollständigen Abhängigkeitsgraph der sechs Medienobjekte für den wissensvermittelnden Block des in Kapitel 5 angegebenen Nutzungsszenarios:

Abbildung 6-1:
Abhängigkeitsgraph aus den in Kapitel 4 definierten Medienobjekten



Wie bereits bei der grafischen Darstellung einzelner Medienobjekte werden im folgenden auch bei der Darstellung von Abhängigkeitsgraphen Medienobjekte durch Ovale und Stichworte durch Rechtecke repräsentiert. Auch die Medienobjekte und Stichworte verbindenden Pfeile behalten ihre in Kapitel 4 beschriebene Semantik als Wissen vermittelnd (vom Medienobjekt zum Stichwort) bzw. Vorwissen verlangend (vom Stichwort zum Medienobjekt) bei.

6.1.1 Subgraphen von Abhängigkeitsgraphen

Durch die Berechnungsvorschrift für Abhängigkeitsgraphen (6.1) ist sichergestellt, daß alle Kanten eines Abhängigkeitsgraphen mit einem Medienobjekt und einem Stichwort verbundenen sind. Darüber hinaus wird jedes Stichwort von mindestens einem Medienobjekt entweder als Vorwissen verlangt oder neu vermittelt.

Diese beiden Eigenschaften finden sich auch in allen, aus Untermengen von M(G) erstellten, Subgraphen eines Abhängigkeitsgraphen G:

Lemma: G ist ein Abhängigkeitsgraph: (6.3)
($M \subseteq M(G) \Rightarrow G(M, I(G))$ ist ebenfalls ein Abhängigkeitsgraph).

Dies bedeutet, daß auch jeder **zusammenhängende Subgraph eines Abhängigkeitsgraphen** die Kriterien eines Abhängigkeitsgraphen erfüllt.

6.1.2 Beziehungen zwischen Medienobjekten

Bei der Darstellung der Eigenschaften von Medienobjektmengen sowie zur Visualisierung von auf Medienobjektmengen operierenden Algorithmen wird im folgenden bevorzugt die Repräsentation des

Abhängigkeitsgraphen verwendet. Auch die Bezeichnungen für bestimmte Eigenschaften und Maße sind größtenteils an die Graphen-Terminologie angelehnt. Für Definitionen, Darstellungen von Eigenschaften, Funktionen, sowie innerhalb des zur Beschreibung von Algorithmen verwendeten Pseudocodes wird hingegen weiterhin bevorzugt auf die Mengen- und Indexschreibweise zurückgegriffen[29].

Wurzeln und Blätter des Abhängigkeitsgraphen

Medienobjekte, die keinerlei Vorwissen benötigen, werden als **Wurzeln** des Abhängigkeitsgraphen bezeichnet. Alle Wurzeln eines Abhängigkeitsgraphen $G=(M,I)$ bilden die Menge $M_{root}(G)$:

$$m \in M(G) \text{ ist Wurzel von } G :\Leftrightarrow \mu_{req}(m,I(G))=\varnothing$$

$$M_{root}(G) := \{ m \in M(G) \mid \mu_{req}(m,I(G))=\varnothing \} \quad (6.4)$$

Stichworte, die von keinem im Abhängigkeitsgraphen enthaltenen Medienobjekt als Vorwissen verlangt werden, bilden die **Blätter** $S_{leaf}(G)$ des Abhängigkeitsgraphen.

$$s \in S(G) \text{ ist Blatt von } G :\Leftrightarrow \sigma_{req}(s,I(G)) = \varnothing$$

$$S_{leaf}(G) := \{ s \in S(G) \mid \sigma_{req}(s,I(G)) = \varnothing \} \quad (6.5)$$

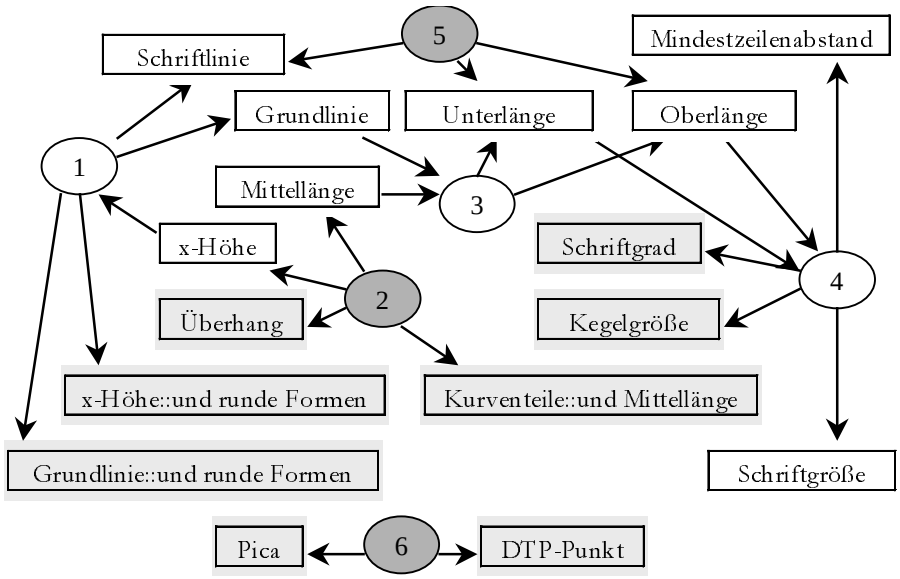
Stichworte, die von keinem der im Abhängigkeitsgraphen enthaltenen Medienobjekte vermittelt werden, werden als **offene Enden** des Abhängigkeitsgraphen bezeichnet:

$$s \in S(G) \text{ ist offenes Ende von } G :\Leftrightarrow \sigma_{prv}(s,I(G)) = \varnothing$$

$$S_{end}(G) := \{ s \in S(G) \mid \sigma_{prv}(s,I(G)) = \varnothing \} \quad (6.6)$$

Abbildung 6-2 zeigt die Wurzeln (dunkelgrau) und Blätter (hellgrau) des Beispielgraphen.

Abbildung 6-2: Wurzeln und Blätter des Abhängigkeitsgraphen



Der Beispielgraph hat drei Wurzeln - die Medienobjekte 2, 5 und 6 - sowie 8 Blätter. Offene Enden existieren nicht, da alle Stichworte von mindestens einem der in $M(G^b)$ enthaltenen Medienobjekte vermittelt werden.

[29] Die wichtigste Ausnahme von dieser Regel ist der Abhängigkeitsgraph selbst. Er wird von vielen Algorithmen als Parameter oder Rückgabewert verwendet, da er eine sehr kompakte Darstellung für die verfügbaren Medienobjekte, Stichworte und Indexeinträge erlaubt.

Vorgänger und Nachfolger

Sowohl bei den bisher vorgestellten Eigenschaften von Medienobjekten, als auch bei der Bestimmung der Wurzeln und Blätter des Abhängigkeitsgraphen wurde immer nur ein Medienobjekt isoliert betrachtet.

Der Sinn des Abhängigkeitsgraphen ist es jedoch vor allem, Aussagen über das Verhältnis zwischen Medienobjekten und zwischen Stichworten und Medienobjekten treffen zu können. Hierbei sind vor allem aus mehreren Objekten bestehende Pfade und Subgraphen innerhalb des Abhängigkeitsgraphen von Interesse.

Grundlage für Pfade und Subgraphen im Abhängigkeitsgraphen sind **Nachbarschaftsbeziehungen** zwischen Medienobjekten:

Definition

Zwei Medienobjekte m und m' werden als **benachbart** bezeichnet, wenn m Wissen über ein Stichwort vermittelt, das von m' als Vorwissen benötigt wird. Hierbei darf der Grad des von m über das verbindende Stichwort vermittelten Wissens nicht kleiner sein als der Grad des von m' verlangten Wissens.

Die verbindenden Stichworte zweier Medienobjekte m und o können über die Abbildung $\mu_{connect}$ bestimmt werden:

$$m, o \in M(G): \mu_{connect}(m, o, G) := \{ s \in S(G) \mid \exists i \in \mu_{prv}(m,I(G)), \; j \in \mu_{req}(o,I(G)) : s = kw(i) = kw(j) \text{ und } prv(i) + \varepsilon > req(j) \}$$

Die Konstante $\varepsilon \geq 0$ bezeichnet einen konfigurationsabhängigen Toleranzwert, der entweder vom Anbieter des Medienobjektspeichers oder vom Anbieter des zu erzeugenden Dokuments festgelegt wird. Falls alle Medienobjekte von einer einzigen Person indiziert wurden, hat sich in der Praxis ein Wert von $\varepsilon=r/8$ als praktikabel erwiesen. Wenn Medienobjekte von verschiedenen Personen indiziert wurden, sollte ε größer gewählt werden (z.B. $\varepsilon=r/4$), um die individuell verschiedenen Auslegungen der Indizierungskriterien auszugleichen.

Die Notation für eine Nachbarschaftsbeziehung zwischen zwei Medienobjekten m und o innerhalb eines Abhängigkeitsgraphen G ist $m \xrightarrow{G} o$:

$$\forall m, o \in M(G), m \neq o: m \xrightarrow{G} o \Leftrightarrow \mu_{connect}(m, o, G) \neq \emptyset \qquad (6.7)$$

Bei zwei benachbarten Medienobjekten $m \xrightarrow{G} o$ wird m als **Vorgänger** von o in G und o als **Nachfolger** von m in G bezeichnet.

Alle Vorgänger eines Medienobjektes m innerhalb eines Abhängigkeitsgraphen G bilden die **Menge der direkten Vorgänger** $\mu_{succ}(m,G)$ dieses Medienobjektes. Alle Nachfolger eines Medienobjektes m in einem Abhängigkeitsgraphen G bilden die **Menge der direkten Nachfolger** $\mu_{pre}(m,G)$ dieses Medienobjekts:

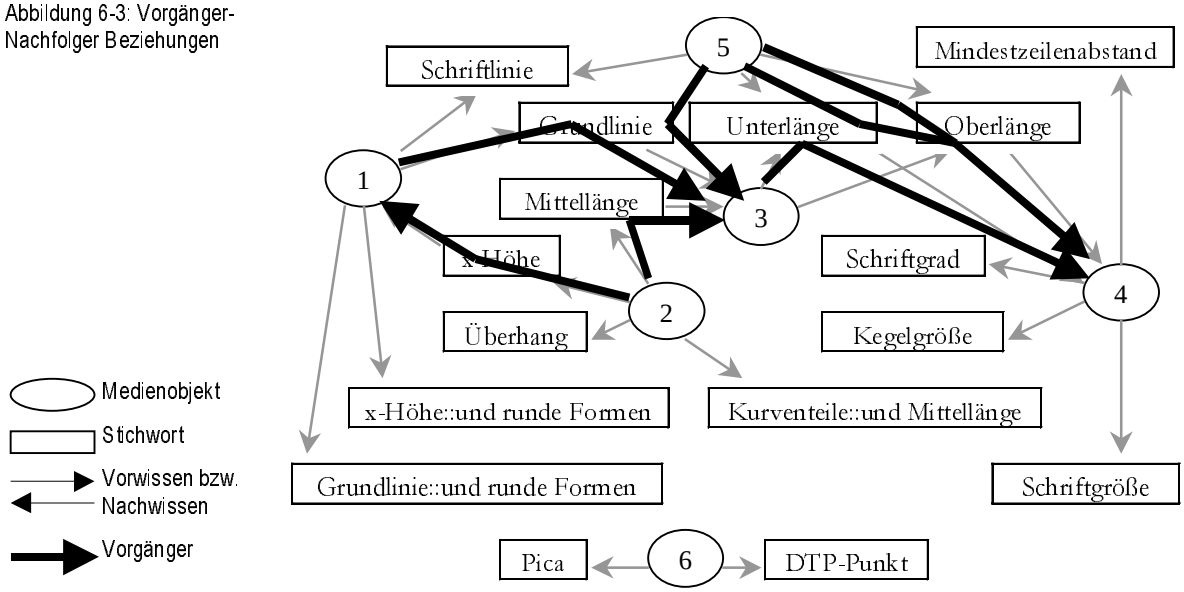
$$\mu_{succ}(m, G) := \{ o \in M(G) \mid m \xrightarrow{G} o \} \qquad (6.8a)$$

$$\mu_{pre}(m, G) := \{ o \in M(G) \mid o \xrightarrow{G} m \}. \qquad (6.8b)$$

Vorgänger-Nachfolger Beziehungen spielen vor allem für die Strukturierung von Medienobjekten eine große Rolle, da sie die ideale Betrachtungsreihenfolge repräsentieren. Falls z.B. $m \xrightarrow{G} o$ gilt, sollte im generierten Lehr- bzw. Informationssystem sichergestellt sein, daß ein Betrachter auf jeden Fall zuerst Medienobjekt m sieht, bevor o angezeigt wird.

Die nachfolgende Abbildung zeigt die Vorgänger-Nachfolger Beziehungen zwischen den Medienobjekten des Beispielgraphen.

Abbildung 6-3: Vorgänger-Nachfolger Beziehungen



Medienobjekte ohne Vorgänger oder Nachfolger in einem Abhängigkeitsgraphen werden als **isoliert** bezeichnet:

$$m \in M(G) \text{ ist isoliert in } G :\Leftrightarrow \mu_{succ}(m,G) = \mu_{pre}(m,G) = \varnothing \qquad (6.9)$$

Im Beispielgraphen ist das Medienobjekt 6 isoliert, da es weder Vorgänger noch Nachfolger besitzt.

6.1.3 Medienobjektgraphen

Ignoriert man die verbindenden Stichworte und betrachtet nur die Nachbarschaftsbeziehungen zwischen Medienobjekten, so stellen diese ebenfalls eine Graph-Struktur dar.

Definition

Der als **Medienobjektgraph** G^M eines Abhängigkeitsgraphen G bezeichnete Graph enthält als Knoten alle in G verfügbaren Medienobjekte. Die (gerichteten und gewichteten) Kanten des Medienobjektgraphen werden aus den Vorgänger-Nachfolger Beziehungen abgeleitet:

$$G^M := (M(G), E(G)) \qquad (6.10)$$

mit $E(G) := \{ (m,o,w) \in (M(G) \times S(G) \times I\!R_+) \mid m \text{--}^G\!\!\rightarrow o, w = w_{edge}(m,o,G) \}$

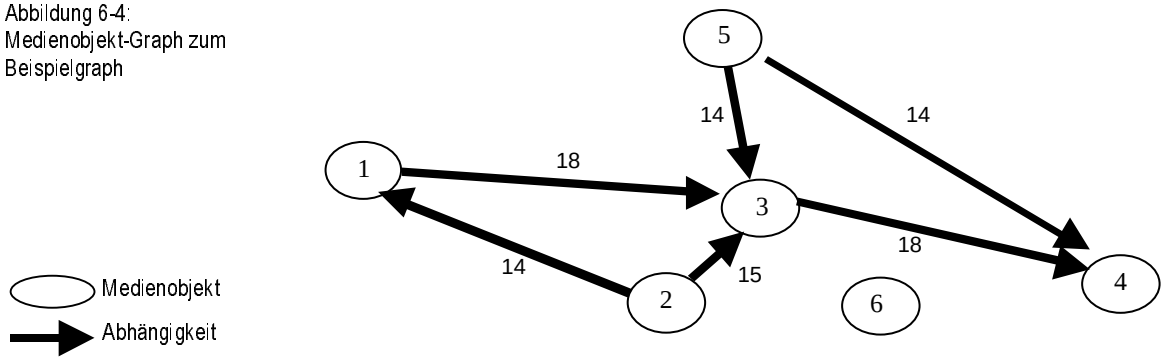
Die **Gewichtung der Kanten** repräsentiert die "Stärke" der Verbindung. Sie wird aus den gemittelten Relevanzwerten der die beiden Medienobjekte verbindenden Indexeinträge berechnet:

$$i_{connect}(m, o, G) := \{(i, j) \in (I \times I) \mid kw(i)=kw(j), kw(i) \in \mu_{connect}(m,o,G), mo(i)=m, mo(j)=o \}$$

$$w_{connect}(i,j) := (prv(i) * rel(i) + 3 * req(j) * rel(j)) / 4r^2$$

$$w_{edge}(m,o,G) := \frac{1}{|i_{connect}(m,o,G)|} \sum_{(i,j) \in i_{connect}(m,o,G)} w_{connect}(i,j), \text{ falls } m \text{--}^G\!\!\rightarrow o, \text{ ansonsten } 0. \qquad (6.11)$$

Die nachfolgende Abbildung zeigt den Medienobjektgraphen des Abhängigkeitsgraphen aus dem Typographie-Beispiel. Die Zahlen an den Kanten geben die nach (6.11) berechneten Gewichtungen an.



Im folgenden Textblock wurden die am besten zusammenpassenden Medienobjekte 1, 3 und 4 hintereinander gesetzt. Obwohl diese Medienobjekte im Original-Text nicht hintereinander standen, ergibt sich ein erstaunlich guter Textfluß.

> Die *Schriftlinie* oder *Grundlinie* ist eine gedachte Linie, an der die Schrift ausgerichtet ist. Runde Formen müssen etwas über die mathematische Grundlinie (und x-Höhe) hinausreichen, um optisch auf der Grundlinie zu stehen bzw. um optisch die gleiche Höhe wie ein glatt abschließendes Zeichen zu haben.
>
> Als *Unterlänge* wird die Strecke bezeichnet, um die Kleinbuchstaben wie z.B. g, j, p, q oder y über die Grundlinie nach unten ragen. [...] Die *Oberlänge* eines Zeichens ist das Maß, mit dem Kleinbuchstaben wie d, f, h, k, l, t über die Mittellänge hinausragen.
>
> Die *Kegelgröße* [...] bestimmt sich aus der Oberlänge plus der Unterlänge plus etwas Zuschlag und legt beim Bleisatz den Mindestzeilenabstand fest. Dieser Wert wird bei DTP-Programmen auch *Schriftgrad* oder *Schriftgröße* genannt.

6.1.4 Erreichbarkeit von Medienobjekten

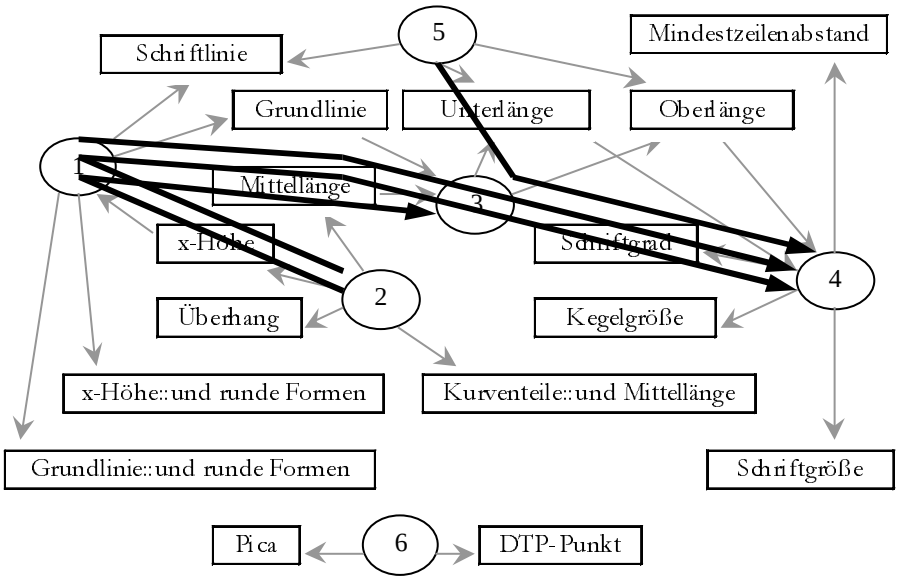
Definition

Ein **Pfad** $m \sim^{G} \sim > o$ von einem Medienobjekt m zu einem Medienobjekt o innerhalb eines Abhängigkeitsgraphen G ist eine Sequenz $<v_1, .., v_n>$ von Medienobjekten, mit $v_1 = m$, $v_n = o$ und

$v_{x-1} --^{G} \rightarrow v_x$ für alle x = 2, .., n. In diesem Fall ist o von m aus **erreichbar**. (6.12)

Abbildung 6.5 zeigt alle aus mindestens drei Medienobjekten bestehenden Pfade des Beispielgraphen.

Abbildung 6-5: Pfade innerhalb des Beispielgraphen, die aus mindestens drei Medienobjekten bestehen



Die Menge der Medienobjekte, die auf einem Pfad vom m nach o liegen, werden als **Pfadmenge $P_{m,o}$** bezeichnet. Die Pfadmenge von Medienobjekt 2 zu Medienobjekt 4 im Beispielgraphen wäre somit: $P_{2,4} = \{1, 2, 3, 4\}$

Das Kriterium der Erreichbarkeit (Def. 6.12) gilt nicht nur für zwei Medienobjekte, sondern auch für die Kombination aus einem Stichwort und einem Medienobjekt. Ein Stichwort s ist dabei von einem Medienobjekt m aus erreichbar, wenn mindestens ein s vermittelndes Medienobjekt von m aus erreichbar ist. Zur Unterscheidung wird diese Form der Erreichbarkeit durch das Symbol $\sim^{G}\sim>$ angezeigt:

$$m \in M(G), s \in S(G): m \sim^{G}\sim > s :\Leftrightarrow \exists o \in \sigma_{prv}(s,I(G)): m \sim^{G}\sim > o \qquad (6.13)$$

Pfade können **Zyklen** enthalten, d.h. für einen aus mindestens zwei Medienobjekten bestehenden Pfad kann gelten: $m \sim^{G}\sim > m$.

Da Zyklen in der Praxis sehr häufig auftreten, spielt ihre Erkennung und Eliminierung bei der automatischen Auswahl und Strukturierung von Medienobjekten eine sehr große Rolle (siehe Kapitel 6.4).

6.1.5 Erlernbarkeit von Medienobjekten

Ein wichtiges Kriterium von Medienobjekten und Stichworten ist ihre **Erlernbarkeit**:

Definition

Ein Medienobjekt ist **erlernbar** innerhalb eines Abhängigkeitsgraphen G, wenn es entweder ein Wurzelobjekt ist oder alle seine als Vorwissen verlangten Stichworte innerhalb von G erlernbar sind. Ein Stichwort ist erlernbar innerhalb eines Abhängigkeitsgraphen G, wenn mindestens ein dieses Stichwort vermittelndes Medienobjekt innerhalb von G erlernbar ist.

Die Erlernbarkeit eines Medienobjektes oder Stichwortes bedeutet somit, daß alle zum seinem Verständnis benötigten Informationen im Abhängigkeitsgraphen vermittelbar sind.

Definition

Jeder Subgraph eines Abhängigkeitsgraphen G, in dem ein Medienobjekt m erlernbar ist, wird als **vermittelnder Subgraph** von m in G bezeichnet.

Die Notation für die Erlernbarkeit eines Medienobjekts m bzw. eines Stichworts s innerhalb eines Abhängigkeitsgraphen G ist $m \lozenge G$ bzw. $s \lozenge G$, mit $m \in M(G)$ und $s \in S(G)$:

$$m \lozenge G :\Leftrightarrow (m \in M(G)) \text{ und } (\mu_{req}(m,I(G))=\varnothing \text{ oder } \forall i \in \mu_{req}(m,I(G)): kw(i) \lozenge G) \qquad (6.14a)$$

$$s \Diamond G :\Leftrightarrow \sigma_{prv}(s,I(G)) \neq \emptyset \text{ und } \exists i \in \sigma_{prv}(s,I(G)) : mo(i) \Diamond G \qquad (6.14b)$$

Abbildung 6-6: Erlernbarkeit von Medienobjekt 4 mit Hilfe von Medienobjekt 5

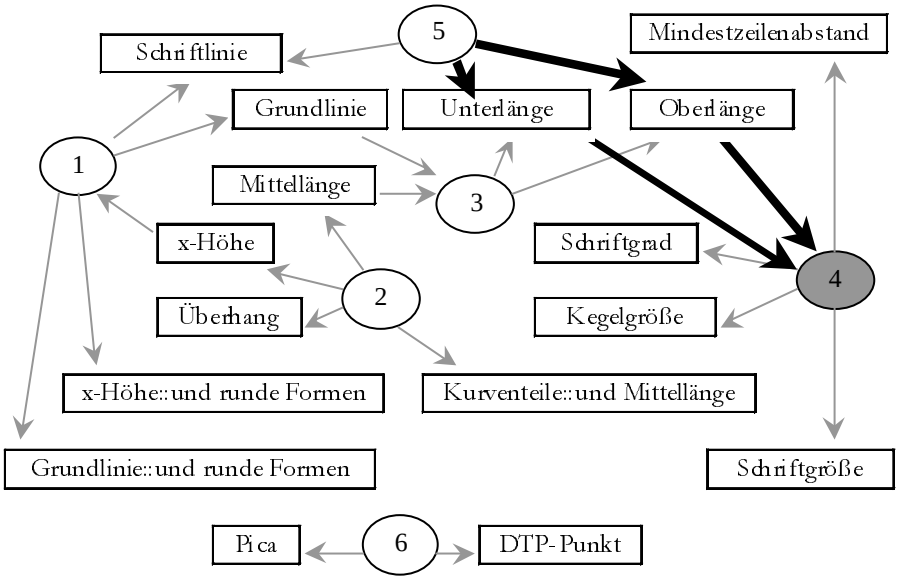


Abbildung 6-6 verdeutlicht dies am Beispiel von Medienobjekt 4:

> Medienobjekt 4 ist innerhalb des Beispielgraphen erlernbar, falls seine beiden als Vorwissen verlangten Stichworte "Unterlänge" und "Oberlänge" erlernbar sind. Das Stichwort "Unterlänge" ist erlernbar, falls mindestens eines seiner beiden vermittelnden Medienobjekte 3 oder 5 erlernbar ist. Gleiches gilt für das Stichwort "Oberlänge". Medienobjekt 5 ist erlernbar innerhalb des Beispielgraphen, da es kein Vorwissen benötigt. Da Medienobjekt 5 erlernbar ist, sind auch die Stichworte "Oberlänge" und "Unterlänge", und damit auch Medienobjekt 4 innerhalb des Beispielgraphen erlernbar. Die Medienobjekte 4 und 5 spannen somit einen vermittelnden Subgraphen für Medienobjekt 4 auf.

Da das Hinzufügen weiterer Medienobjekte zu einem vermittelnden Subgraph keinen Einfluß auf die Erlernbarkeit des vermittelten Objekts hat, ist auch jeder diesen vermittelnden Subgraph enthaltende Subgraph des Abhängigkeitsgraphen ebenfalls ein vermittelnder Subgraph:

$$m \Diamond G: (G' \supseteq G \Rightarrow m \Diamond G')$$

Von den 18 vermittelnden Subgraphen des Medienobjektes 4 zeichnen sich lediglich {4, 5} und {1, 2, 3, 4} dadurch aus, daß sie keinerlei redundanten Medienobjekte enthalten.

Definition

Ein Medienobjekt m wird als **redundant** bezüglich einer vermittelnden Untermenge M eines Medienobjekts o bezeichnet, wenn o auch in M \ {m} erlernbar ist:

$$o \Diamond G \text{ und } o \Diamond G(M(G) \setminus \{m\}, I(G)) \Leftrightarrow m \text{ ist redundant in } M \qquad (6.15)$$

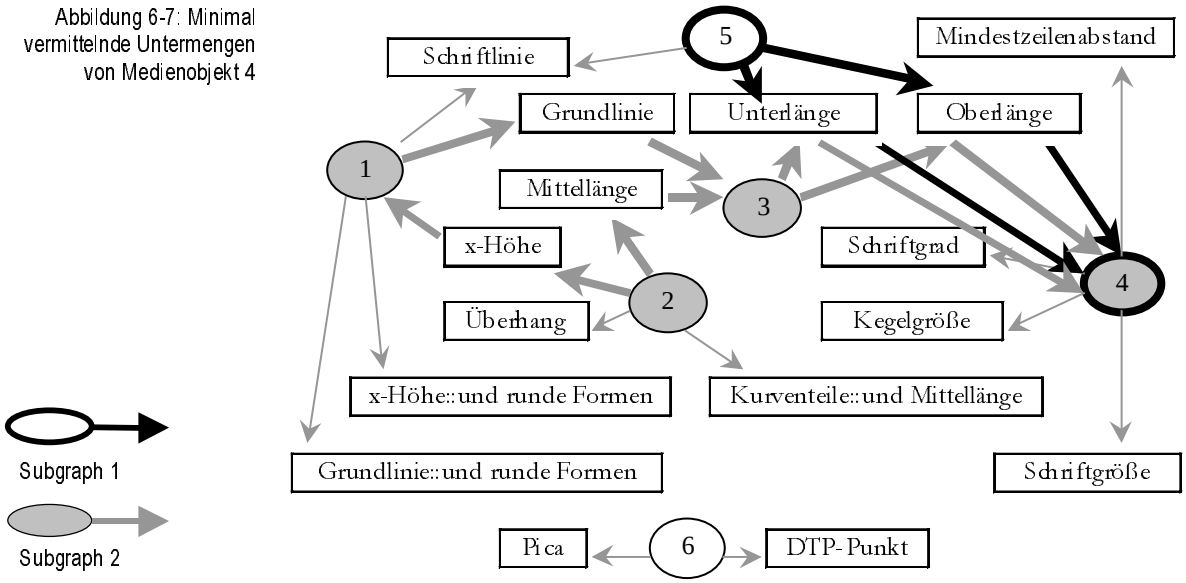
Vermittelnde Subgraphen ohne redundante Medienobjekte werden als **minimal vermittelnd** bezeichnet.

Das Symbol zur Kennzeichnung eines minimal vermittelnden Subgraphen ist ◊!:

$$m \Diamond! G :\Leftrightarrow m \Diamond G \text{ und } \forall o \in M(G): m \neg\Diamond G(M(G) \setminus \{o\}, I(G)) \qquad (6.16a)$$
$$s \Diamond! G :\Leftrightarrow s \Diamond G \text{ und } \forall o \in M(G): s \neg\Diamond G(M(G) \setminus \{o\}, I(G)) \qquad (6.16b)$$

In Abbildung 6-7 sind die beiden minimal vermittelnden Subgraphen von Medienobjekt 4 durch Einfärbung bzw. Umrandung hervorgehoben.

Abbildung 6-7: Minimal vermittelnde Untermengen von Medienobjekt 4



Definition:

Ein Abhängigkeitsgraph bzw. Subgraph eines Abhängigkeitsgraphen wird als **abgeschlossen** bezeichnet, wenn alle in ihm enthaltenen Medienobjekte auch in ihm erlernbar sind:

$$G \text{ ist abgeschlossen} :\Leftrightarrow \forall m \in M(G): m \lozenge G \qquad (6.17)$$

Analog zu den minimal vermittelnden Subgraphen wird ein Abhängigkeitsgraph als **minimal abgeschlossen** bezeichnet, wenn alle in ihm enthaltenen Medienobjekte in ihm auch minimal erlernbar sind:

$$G \text{ ist minimal abgeschlossen} :\Leftrightarrow \forall m \in M(G): m \lozenge! G \qquad (6.18)$$

Von den 25 abgeschlossenen Subgraphen des Beispielgraphen sind lediglich 7 minimal abgeschlossen:

{ 5 } { 2 } { 6 }

{ 4, 5 } { 1, 2 } { 1, 2, 3 }

{ 1, 2, 3, 4 }

Minimal abgeschlossene Abhängigkeitsgraphen sind vor allem für die Auswahl von Medienobjekten relevant, da sie eine Menge von Medienobjekten repräsentieren, mit denen Informationen ohne Redundanz vermittelt werden können.

6.1.6 Aussen- und Innengrad

Graphen werden i.A. als dicht bezeichnet, wenn die Anzahl der Kanten an das Quadrat der Knotenzahl heranreicht [Cormen+90]. Bezogen auf den Abhängigkeitsgraphen würde dies bedeuten, daß die Anzahl der Indexeinträge ca. dem Quadrat aus der Anzahl von Medienobjekten und Stichworten entspräche.

Eine solch starke Vernetzung wird i.a. jedoch weder vom Abhängigkeitsgraphen noch vom abgeleiteten Medienobjektgraphen erreicht, so daß die behandelten Graphen als "dünn besiedelt" angesehen werden können.

Anstelle der Dichte wird daher im folgenden die durchschnittliche Anzahl von Medienobjekten zur Vermittlung eines Stichwortes als wichtigstes Maß zur Beschreibung des Verhältnisses von Knoten und Kanten verwendet. Da diese Maßzahl von der Anzahl der Ausgangskanten aller Medienobjekte abhängt, wird sie auch als **Aussengrad** des Abhängigkeitsgraphen bezeichnet:

$$\text{Ausgang}(G) := \frac{|I_{prv}(G)|}{|S(G)|} \tag{6.19}$$

Ein Aussengrad von 2 bedeutet somit, daß jedes Stichwort im Durchschnitt von zwei verschiedenen Medienobjekten vermittelt wird. Da der Aussengrad implizit auch eine Aussage über die Anzahl redundanter Stichwort-Vermittlungen zuläßt, werden Abgängigkeitsgraphen mit einem niedrigen Aussengrad (Ausgang(G) ≤ 2) auch als gering redundant und Graphen mit einem großen Aussengrad (Ausgang(G) > 2) als stark redundant bezeichnet

Analog zum Aussengrad bezeichnet der **Innengrad** *Eingang(G)* des Abhängigkeitsgraphen die Anzahl der Stichworte, die durchschnittlich als Vorwissen eines Medienobjektes verlangt sind:

$$\text{Eingang}(G) := \frac{|I_{req}(G)|}{|M(G)|} \tag{6.20}$$

Aussengrad und Innengrad sind vor allem für die Berechnung des durchschnittlichen Laufzeitverhaltens vieler Algorithmen zu Auswahl und Strukturierung interessant. Darüber hinaus können sie auch als Heuristik für die Wahl der geeigneten Bewertungsfunktionen eingesetzt werden.

6.2 Erlernbarkeit der vorgegebenen Lernziele

Das Konzept der Erlernbarkeit und der abgeschlossenen Subgraphen kann von einzelnen Medienobjekten oder Stichworten auch auf Mengen von Medienobjekten bzw. Stichworten erweitert werden. Darauf aufbauend können anschließend die Anforderungen an die Auswahl und Strukturierung von Medienobjekten formalisiert werden.

Definition

Eine Menge von Medienobjekten $M \subseteq M(G)$ ist in einem Abhängigkeitsgraphen G **erlernbar** (geschrieben als $M \lozenge\!\lozenge G$), wenn alle in M enthaltenen Medienobjekte in G erlernbar sind:

$$M \lozenge\!\lozenge G :\Leftrightarrow M \subseteq M(G) \text{ und } \forall m \in M: m \lozenge G \tag{6.21a}$$

Analog ist eine Menge von Stichworten $S \subseteq S(G)$ ist in einem Abhängigkeitsgraphen G **erlernbar** (geschrieben als $S \lozenge\!\lozenge G$), wenn für jedes in S enthaltene Stichwort mindestens ein vermittelndes Medienobjekt in G erlernbar ist:

$$S \lozenge\!\lozenge G :\Leftrightarrow \forall s \in S, \exists m \in \sigma_{prv}(s, I(G)): m \lozenge G \tag{6.21b}$$

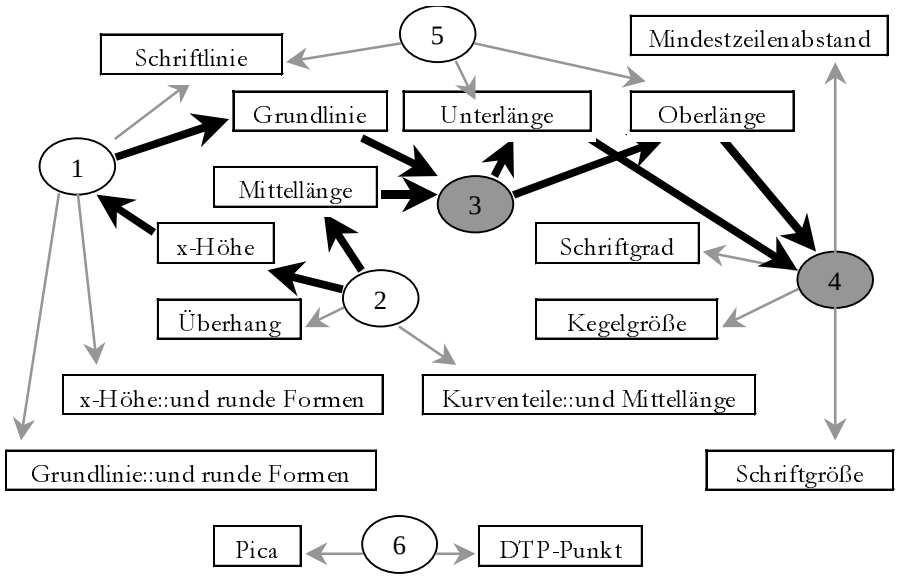
Auch hier sind für die automatische Auswahl und Strukturierung vor allem die minimal abgeschlossenen Abhängigkeitsgraphen von Interesse, d.h. die Abhängigkeitsgraphen, innerhalb deren eine vorgegebene Menge von Objekten minimal erlernbar ist. Das Symbol für die minimale Erlernbarkeit einer Menge von Objekten innerhalb eines Abhängigkeitsgraphen ist "$\lozenge\!\lozenge!$":

$$M \lozenge\!\lozenge! G :\Leftrightarrow M \lozenge\!\lozenge G \text{ und } \forall m \in M(G): M \neg\lozenge\!\lozenge G(M \setminus \{m\}, I(G)) \tag{6.22a}$$

$$S \lozenge\!\lozenge! G :\Leftrightarrow S \lozenge\!\lozenge G \text{ und } \forall m \in M(G): S \neg\lozenge\!\lozenge G(M \setminus \{m\}, I(G)) \tag{6.22b}$$

Die nachfolgende Abbildung verdeutlicht die minimale Erlernbarkeit am Beispiel der Medienobjektmenge $M = \{3, 4\}$.

Abbildung 6-8:
{3, 4} ist minimal
erlernbar in {1, 2, 3, 4}



Wie aus der Abbildung zu ersehen ist, gibt es nur einen einzigen Subgraphen des Beispielgraphen, in dem { 3, 4 } minimal erlernbar ist: $\{3, 4\} \lozenge\lozenge! G(\{1, 2, 3, 4\}, I^b)$

Etwas anderes sieht es aus, wenn es um die gleichzeitige Erlernbarkeit der Stichworte "Schriftlinie" und "Schriftgrad" geht. Hier gibt es zwei Abhängigkeitsgraphen, in denen beide Stichworte minimal erlernbar sind: $G(\{4, 5\}, I^b)$ und $G(\{ 1, 2, 3, 4\}, I^b)$.

Auf Basis der minimal vermittelnden Subgraphen können die Anforderungen an den Algorithmus zur Auswahl und Strukturierung von Medienobjekten zumindest teilweise auf formale Eigenschaften des gesuchten Subgraphen zurückgeführt werden:

> Ziel der automatischen Auswahl und Strukturierung von Medienobjekten ist es, für jeden zu generierenden Block b einen Subgraphen G^{b*} innerhalb des angepaßten Abhängigkeitsgraphen G^b zu finden, für den gilt: $b\_content(b) \lozenge\lozenge! G^{b*}$.

Darüber hinaus sollten die Struktur und die Zusammensetzung von G^{b*} an die in den Attributen des Blockes angegebenen Präferenzen angepaßt sein.

6.3 Minimale Tiefen

Ein wichtiger Bestandteil der Heuristiken zur Auswahl und Strukturierung von Medienobjekten ist die **minimale Tiefe** eines jeden Medienobjekts und Stichworts innerhalb eines Abhängigkeitsgraphen. Die minimale Tiefe eines Medienobjektes oder Stichwortes x basiert auf der Länge des kürzesten Pfades von einem beliebigen Wurzelobjekt zu den Vorgängerknoten des Objekts x. Die minimale Tiefe von x ist zugleich die minimale Anzahl der Medienobjekte, aus denen jeder x minimal vermittelnde Subgraph eines Abhängigkeitsgraphen bestehen muß.

Grundidee der minimalen Tiefen ist, daß die Anzahl der Medienobjekte in jedem ein Objekt minimal vermittelnden Subgraphen tendenziell desto größer sind, je weiter dieses Objekt von einem Wurzelobjekt entfernt ist. Da der verwendete heuristische Suchalgorithmus von den zu vermittelnden Stichworten b_content(b) aus einen Subgraphen zu einer Menge von Wurzelobjekten berechnet, geben die minimalen Tiefen der ausgewählten Medienobjekte einen Hinweis darauf, ob sich der entstehende Subgraph auch wirklich in Richtung der Wurzelknoten "bewegt". Darüber hinaus sind minimale Tiefen auch ein wichtiges Bewertungskriterium, da sie eine ungefähre Abschätzung der zu erwartenden Größe des gesuchten Subgraphen erlauben.

Definition

Die **minimale Tiefe** $\mu_{depth}(m,G)$ eines Medienobjektes m in einem Abhängigkeitsgraphen G ist eine untere Schranke für die Anzahl von Medienobjekten, die benötigt werden, um m zu erlernen. Die minimale Tiefe $\sigma_{depth}(s,G)$ eines Stichwortes s in einem Abhängigkeitsgraphen G ist eine untere Schranke für die Anzahl von Medienobjekten, die benötigt werden, um s zu erlernen:

$$\mu_{depth}(m,G) := \begin{cases} 0, \text{falls } \exists i \in \mu_{req}(m, I(G)) \text{ mit } \sigma_{depth}(kw(i), G) = 0 \\ 1, \text{falls } \mu_{req}(m, I(G)) = \varnothing \\ \max\left\{ \bigcup_{i \in \mu_{req}(m, I(G))} \{\sigma_{depth}(kw(i), G) + 1\} \right\}, \text{sonst} \end{cases} \quad (6.23a)$$

$$\sigma_{depth}(s,G) := \begin{cases} 0, \text{falls } \sigma_{prv}(s, I(G)) = \varnothing \\ 0, \text{falls } \forall i \in \sigma_{prv}(s, I(G)) : \mu_{depth}(mo(i), G) = 0 \\ \min\left\{ \bigcup_{i \in \sigma_{prv}(s, I(G)), \mu_{depth}(mo(i), G) > 0} \{\mu_{depth}(mo(i), G)\} \right\}, \text{sonst} \end{cases} \quad (6.23b)$$

Die minimale Tiefe eines einzelnen Objektes kann - analog zur Definition - rekursiv berechnet werden. Der Nachteil einer rekursiven Berechnung ist vor allem die Beschränkung auf - in der Praxis eher seltene - zyklenfreie Abhängigkeitsgraphen. Darüber hinaus bietet es sich an, die minimalen Tiefen aller Medienobjekte und Stichworte eines Abhängigkeitsgraphen G gleichzeitig zu berechnen.

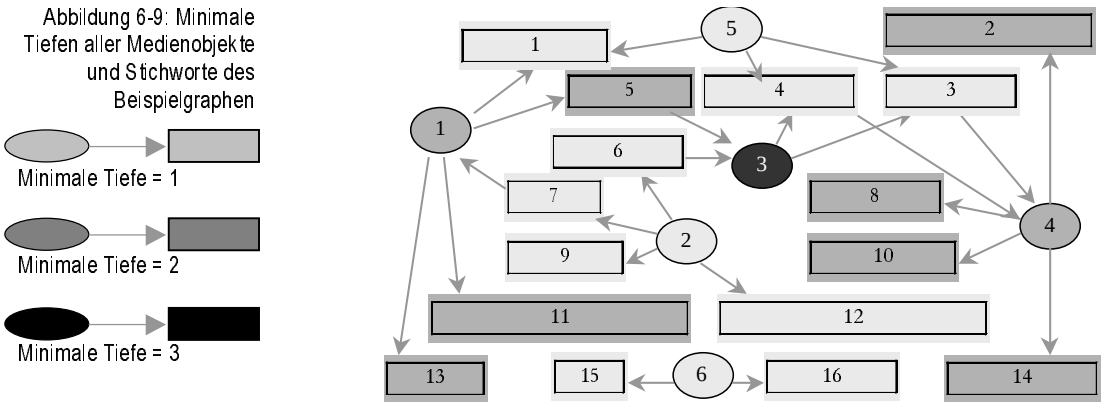
Die Berechnung der minimalen Tiefen basiert auf einem kompletten Traversieren des Abhängigkeitsgraphen mittels einer iterativen Breitensuche. Dadurch, daß beim iterativen Traversieren alle Vorgänger des aktuell besuchten Objekts bereits bekannt sind, kann der Algorithmus auch auf zyklische Abhängigkeitsgraphen angewandt werden.

Die Funktionsweise des Algorithmus ist relativ einfach:

> Es werden alle noch nicht besuchten Medienobjekte besucht, für deren sämtliche als Vorwissen verlangten Stichworte die minimale Tiefe bereits berechnet wurde. Falls keine solchen Medienobjekte gefunden werden können, terminiert der Algorithmus.
>
> Falls Medienobjekte mit dieser Eigenschaft existieren, wird ihre minimale Tiefe auf den um eins erhöhten Wert der größten minimalen Tiefe ihrer verlangten Stichworte gesetzt. Da die maximale minimale Tiefe der verlangten Stichworte identisch mit der Zahl der durchlaufenen Iterationen ist, vereinfacht sich die Berechnung. Allen von diesen Medienobjekten vermittelten Stichworten wird die gleiche minimale Tiefe zugewiesen, sofern ihre minimale Tiefe nicht bereits in einer vorherigen Iteration des Algorithmus berechnet wurde.

Abbildung 6-9 zeigt die minimalen Tiefen der Medienobjekte und Stichworte des aus den Beispieltexten erstellten Abhängigkeitsgraphen. Die Zahlen in den Ovalen bzw. Rechtecken geben die Indizes der einzelnen Medienobjekte bzw. Stichworte an. Objekte mit einer minimalen Tiefe von 1 sind hellgrau dargestellt, Objekte mit einer minimalen Tiefe von 2 dunkelgrau und Objekte mit einer minimalen Tiefe von 3 schwarz.

Abbildung 6-9: Minimale Tiefen aller Medienobjekte und Stichworte des Beispielgraphen



Der Aufwand zur Berechnung der minimalen Tiefen aller Medienobjekte und Stichworte eines Abhängigkeitsgraphen ist $O(M * (I^{prv} + S + I^{req})) = O(M * (I + S))$. Dieser relativ hohe Aufwand für eine eigentlich lineare Breitensuche kommt durch die aufwendige Ermittlung der Kandidaten für die jeweils nächste Iteration zustande.

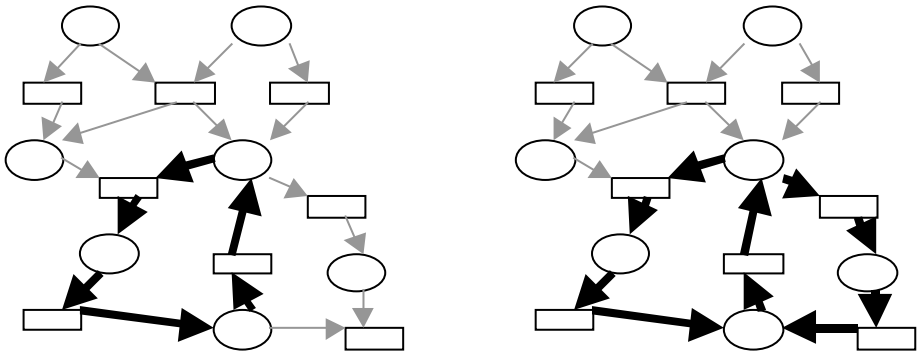
6.4 Erkennung und Bewertung von Zyklen

Wie bereits mehrfach angedeutet, enthalten Abhängigkeitsgraphen sehr häufig Zyklen. Ein **Zyklus** existiert, wenn für ein Medienobjekt m aus M(G) gilt: $m \sim^{G} \sim > m$.

Ein Pfad $<m, m_1, ..., m_n, m>$ wird als **einfacher Zyklus** bezeichnet, wenn alle m_1 bis m_n voneinander verschieden sind [Cormen+90]. Falls zwei oder mehr Zyklen gemeinsame Medienobjekte besitzen und sich somit überschneiden, bilden sie einen **komplexen Zyklus**.

Die folgende Abbildung zeigt jeweils ein Beispiel für einen einfachen (links) und einen komplexen Zyklus (rechts).

Abbildung 6-10: Einfacher (links) und komplexer Zyklus (rechts)



Zyklen entstehen, wenn sich zwei (oder mehr) Medienobjekte gegenseitig erklären. Ein Beispiel hierfür sind die beiden Sätze "Die Varianz ist das Quadrat der Standardabweichung" und "Die Standardabweichung ist die Quadratwurzel der Varianz." Sobald diese beiden Sätze in zwei verschiedenen Medienobjekten auftauchen, existiert ein Zyklus im Abhängigkeitsgraphen.

Bei der Erstellung des Zielgraphen G^{b*} nach den in Kapitel 7 beschriebenen Algorithmen ist sichergestellt, daß die resultierende Medienobjektstruktur zyklenfrei ist. Hierzu ist es jedoch notwendig, Zyklen innerhalb des Abhängigkeitsgraphen zu erkennen und bei der Auswahl und Strukturierung geeignet zu umgehen. Zumindest das Erkennen von Zyklen kann in linearer Zeit realisiert werden.

6.4.1 Starke Zusammenhangskomponenten

Grundlage der Erkennung von Zyklen innerhalb des Abhängigkeitsgraphen bilden sog. starke Zusammenhangskomponenten [Tarjan72]:

Ein Subgraph G' eines Abhängigkeitsgraphen G bildet eine **stark zusammenhängende Komponente**, wenn gilt, daß

$$\forall m, o \in M(G'), m \neq o: m \sim^{G'} \!\!> o \text{ und } o \sim^{G'} \!\!> m. \qquad (6.24a)$$

Definition:

Alle Medienobjekte eines einfachen oder komplexen Zyklus bilden eine stark zusammenhängende Komponente, denn auch in dem aus diesen Objekten gebildeten Abhängigkeitsgraphen sind je zwei Medienobjekte gegenseitig erreichbar. Die Menge aller **Zyklen eines Abhängigkeitsgraphen** G wird definiert als

$$SCC(G) = \{ M \in \mathsf{P}(M(G)) \mid \forall m, o \in M, m \neq o: m \sim^{G'} \!\!> o \text{ und } o \sim^{G'} \!\!> m \}. \qquad (6.24b)$$

Grundlage des Algorithmus zur Berechnung der starken Zusammenhangskomponenten ist eine Tiefensuche sowohl auf dem Abhängigkeitsgraph, als auch auf dessen transponierten Graphen. Der transponierte Graph G^T eines gerichteten Graphen G zeichnet sich dadurch aus, daß er die gleiche Knoten- und Kantenmenge besitzt, die Richtung aller Kanten von G^T jedoch invers zu G ist [Noltemeyer76].

Der Algorithmus ist ausführlich in [Cormen+90] beschrieben. Seine Laufzeit ist linear zur Anzahl der Knoten und Kanten des Medienobjektgraphen M^G.

6.4.2 Bewertung von Zyklen

Zyklen innerhalb des Abhängigkeitsgraphen stellen ein großes Problem dar, da sie

- potentiell zu (nicht erwünschten) Zyklen in der Seitenstruktur führen,
- die Berechnung einiger Maßzahlen erschweren (siehe Kapitel 6.5) und
- zusätzliche, zeitaufwendige Bewertungsfunktionen bei der Auswahl und Strukturierung von Medienobjekten verlangen.

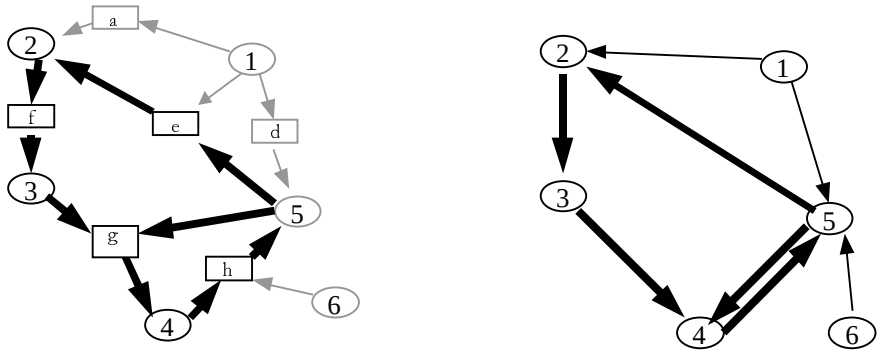
Aus diesen Gründen wird im ersten Schritt des Auswahlalgorithmus die Zyklenfreiheit des Abhängigkeitsgraphen durch Entfernen geeigneter Knoten und Kanten sichergestellt (siehe Kapitel 7.2). Alle anschließend auf dem Abhängigkeitsgraphen arbeitenden Auswahl- und Strukturierungsalgorithmen setzten einen zyklenfreien Graphen voraus.

Die einfachste Vorgehensweise zur Entfernung aller Zyklen wäre, alle auf einer starken Zusammenhangskomponente liegenden Knoten mitsamt ihrer Indexeinträge aus dem Abhängigkeitsgraphen zu löschen. Da Zyklen in der Praxis jedoch sehr häufig auftreten, besteht hierbei die Gefahr, daß sehr viele Medienobjekte gelöscht werden, was wiederum zur Folge hat, daß sehr viele Stichworte und Medienobjekte innerhalb des zyklenfreien aber stark verkleinerten Abhängigkeitsgraphen nicht mehr erlernbar sind.

Ziel der Auflösung von Zyklen muß es daher sein, sowenig Kanten und Knoten wie nur möglich aus dem Abhängigkeitsgraphen zu entfernen. Insbesondere sollten bevorzugt vermittelnde Kanten gelöscht werden, zu denen Alternativen bestehen, da hierdurch die Erlernbarkeit der Knoten des Abhängigkeitsgraphen nicht beeinflußt wird.

Abbildung 6-11 (links) zeigt einen Abhängigkeitsgraphen, der einen komplexen, aus vier Medienobjekten bestehenden Zyklus enthält. Der rechte Teil der Abbildung stellt den zugehörigen Medienobjekt-Graphen dar.

Abbildung 6-11: Beispiel für verschiedenen Knotentypen eines Zyklus



Die für die Zyklenentfernung interessanten Knoten in Abbildung 6-11 sind die Medienobjekte 4 und 5, da nur diese beiden Knoten Stichworte vermitteln, die auch von außerhalb des Zyklus liegenden Medienobjekten erklärt werden können.

Durch Entfernen der Kante von Medienobjekt 5 zu Stichwort "e" bleibt Medienobjekt 2 weiter erlernbar, da Stichwort "e" auch von Medienobjekt 1 vermittelt wird. Zusätzlich wird die Größe des Zyklus von vier auf zwei Medienobjekte (4 und 5) reduziert. Ein Knoten, der die Eigenschaft besitzt, daß die Entfernung einer oder mehrerer seiner Ausgangskanten - ohne Verlust der Erlernbarkeit - zu einer Verkleinerung des Zyklus führt, wird "**Connector**" genannt, da er eine Verbindung vom Zyklus zum Rest des Graphen herstellt.

Auch durch Entfernen der Kante von Medienobjekt 4 zu Stichwort "h" wird die Erlernbarkeit der Knoten des Abhängigkeitsgraphen nicht beeinträchtigt. Darüber hinaus wird der Zyklus durch Entfernung dieser einzigen Kante komplett aufgelöst. Knoten, die die Eigenschaft besitzen, daß die Entfernung einer oder mehrerer ihrer Ausgangskanten - ohne Verlust der Erlernbarkeit - zu einer Auflösung des Zyklus führt, werden "**Resolver**" genannt.

Insgesamt lassen sich vier verschiedene Arten von auf Zyklen liegenden Medienobjekten unterscheiden. Das entscheidende Kriterium zur Klassifizierung ist, ob ein Medienobjekt durch Entfernung von Eingangskanten weiterhin erlernbar ist, und ob es ohne Gefahr der Erzeugung eines Zyklus in das zu erzeugende Lehrsystem übernommen werden kann:

Resolver: Werden alle vermittelnden Indexeinträge dieses Medienobjekts ignoriert und die betreffenden Stichworte statt dessen ausschließlich von nicht auf dem Zyklus liegenden Medienobjekte vermittelt, ist der Zyklus aufgelöst, d.h. alle auf dem Zyklus liegenden Medienobjekte können gefahrlos verwendet werden. (Medienobjekt 4 in Abbildung 6-11)

Connector: Alle vermittelten Stichworte dieses Medienobjekts können auch durch nicht auf dem Zyklus liegende Medienobjekte vermittelt werden. Auch wenn dies geschieht, sind noch Zyklen möglich. (Medienobjekt 5 in Abbildung 6-11) *Connectoren* können nur als Teil eines komplexen Zyklus auftreten.

Bridge: Alle verlangten Stichworte dieses Medienobjekts können durch Medienobjekte vermittelt werden, die entweder nicht auf dem Zyklus liegen oder vom Typ *Bridge* sind. (Medienobjekte 2 und 3 in Abbildung 6-11)

DeadEnd: Die Verwendung dieses Medienobjektes führt unausweichlich zu einem Zyklus in der Medienobjektstruktur des zu erzeugenden Dokuments.

Für den Algorithmus zur Entfernung der Zyklen innerhalb des Abhängigkeitsgraphen steht der Zyklus-Typ eines jeden Medienobjekts als Element des Wertebereichs

$$\{ \text{"}none\text{"}, \text{"}resolver\text{"}, \text{"}connector\text{"}, \text{"}bridge\text{"}, \text{"}deadend\text{"} \} \tag{6.25}$$

zur Verfügung.

Die Werte der Abbildung werden vorberechnet und in einem Vektor abgelegt. Die Berechnung der Zyklus-Typen ist relativ einfach, so daß der Algorithmus an dieser Stelle nur grob skizziert werden soll:

Für jeden Zyklus des Abhängigkeitsgraphen werden zuerst alle Medienobjekte identifiziert, deren sämtliche vermittelten Stichworte auch durch nicht auf dem Zyklus liegende Medienobjekte vermittelbar sind. Anschließend wird für jedes dieser Medienobjekte ermittelt, ob es sich um einen *Resolver* oder einen *Connector* handelt. Hierzu wird aus den Medienobjekten des Zyklus ein Abhängigkeitsgraph erstellt, in dem alle auf dem Zyklus liegenden Ausgangskanten des betrachteten Objekts entfernt wurden. Enthält dieser Abhängigkeitsgraph keine starken Zusammenhangkomponenten, handelt es sich bei dem betrachteten Knoten um einen *Resolver*.

Nachdem alle *Resolver* und *Connectoren* identifiziert sind, können durch eine von diesen Knoten ausgehende Breitensuche innerhalb des Zyklus alle *Bridge*-Objekte ermittelt werden. Auf einem Zyklus liegende Medienobjekte, die weder *Resolver*, noch *Connector* oder *Bridge* sind, werden als *DeadEnd* markiert.

Der Algorithmus zu Klassifizierung der Knoten auf Zyklen ist sehr aufwendig, da für jeden *Resolver* und *Connector* eine Neuberechnung der starken Zusammenhangskomponenten erfolgt. Da der in Kapitel 7.2 beschriebene Algorithmus zur Entfernung von Zyklen jedoch keine komplette Klassifizierung alle Knoten benötigt, bricht der Klassifizierungsalgorithmus ab, sobald ein *Resolver* gefunden wurde. Durch Berücksichtigung relativ einfacher Heuristiken bei der Festlegung der Reihenfolge der Analyse (z.B. viele Kanten innerhalb des Zyklus), können darüber hinaus *Resolver* relativ schnell identifiziert werden.

6.5 Vorgängermengen

Die wichtigste Maßzahl für die automatische Auswahl und Strukturierung von Medienobjekten ist die Menge aller direkten und indirekten Vorgänger eines jeden Knotens des Abhängigkeitsgraphen:

Definition

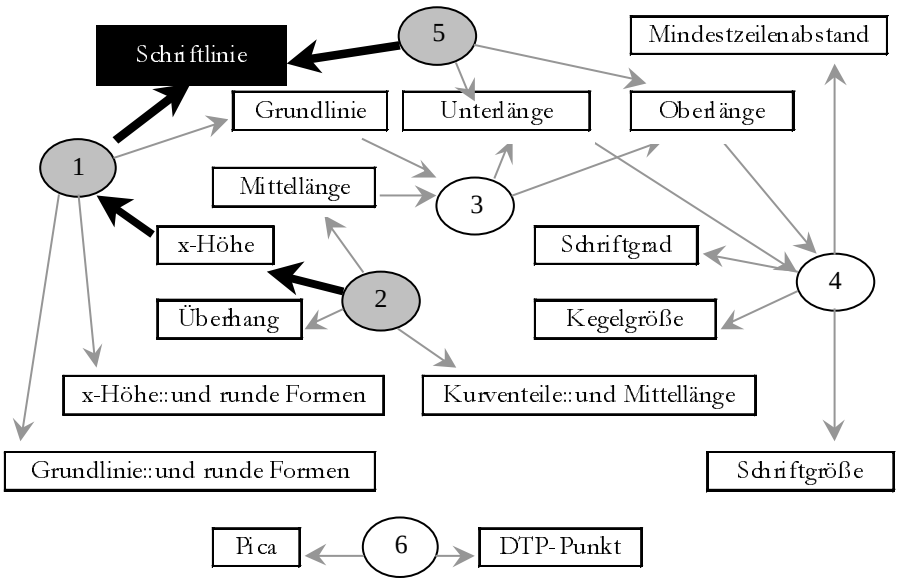
Die **Vorgängermenge** $\mu_{ps}(m,G)$ bzw. $\sigma_{ps}(s,G)$ eines Medienobjektes m bzw. Stichwortes s enthält alle Medienobjekte, von denen aus innerhalb eines gegebenen Abhängigkeitsgraphen G ein Pfad zu dem betrachteten Medienobjekt bzw. Stichwort führt:

$$\mu_{ps}(m,G) := \{ o \in M(G) \mid o \sim^{G}\!\!> m \} \qquad (6.26a)$$

$$\sigma_{ps}(s,G) := \{ m \in M(G) \mid m \sim^{G}\!\sim\!> s \} \qquad (6.26b)$$

Die nachfolgende Abbildung verdeutlicht die Idee der Vorgängermengen anhand des Stichwortes "Schriftlinie" aus dem Beispielgraphen:

Abbildung 6-12: Vorgängermenge (grau) des Stichworts „Schriftlinie“



Wie aus der Abbildung zu ersehen ist, besteht die Vorgängermenge des Stichwortes "Schriftlinie" aus den Medienobjekten 1, 2, und 5. Direkte Vorgänger des Stichwortes sind 1 und 5, während Medienobjekt 2 ein indirekter Vorgänger über Medienobjekt 1 ist.

6.5.1 Eigenschaften von Vorgängermengen

Vorgängermengen sind um so aussagekräftiger, je größer der Abhängigkeitsgraph ist. Aus diesem Grunde sollen die Eigenschaften von Vorgängermengen anhand eines etwas komplexeren als dem bisher verwendeten Graphen beschrieben werden (siehe Abbildung 6-13).

Eine der wichtigsten Eigenschaften von Vorgängermengen ist:

Lemma

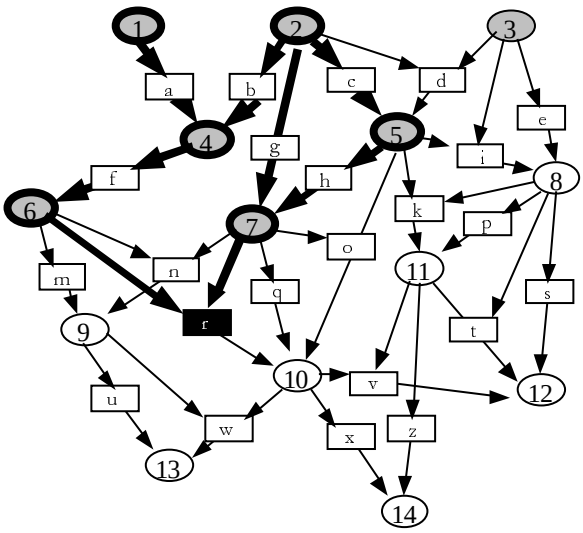
Alle Abhängigkeitsgraphen $G' \subseteq G$, in denen ein Medienobjekt m oder ein Stichwort s minimal erlernbar ist, sind Subgraphen des aus der Vorgängermenge dieses Objektes erstellten Abhängigkeitsgraphen:

$$\forall G' \subseteq G: (m \in M(G), m \Diamond! G' \Rightarrow M(G') \subseteq \mu_{ps}(m,G)) \quad (6.27a)$$

$$\forall G' \subseteq G: (s \in S(G), s \Diamond! G' \Rightarrow M(G') \subseteq \sigma_{ps}(s,G)) \quad (6.27b)$$

Abbildung 6-13 zeigt sowohl die Vorgängermenge von Stichwort "r" (grau) als auch die beiden vermittelnden Mengen {1, 2, 4, 6} und {2, 5, 7}, über die dieses Stichwort minimal erreichbar ist[30].

Abbildung 6-13: Vorgängermenge (grau) und minimal vermittelnde Mengen (umrandet) von Stichwort „r". Die hervorgehobenen Kanten bilden die Kantenmenge der "r" minimal vermittelnden Subgraphen.



Beweis:

Diese Eigenschaft von Vorgängermengen läßt sich durch einen Widerspruchsbeweis zeigen:

Annahme: $\exists m \in M(G), M' \subseteq M(G), o \in M'$: $m \Diamond! G(M', I(G))$ und $o \notin \mu_{ps}(m,G)$

$o \notin \mu_{ps}(m,G) \Leftrightarrow \mathbf{\neg(o \sim^{G} \sim > m)}$

$m \Diamond! G(M', I(G)) \Rightarrow m \neg\Diamond G(M' \setminus \{o\}, I(G))$

$\Rightarrow \exists o' \in M', s' \in \mu_{prv}(o, I(G)): s' \in \mu_{req}(o', I(G))$

$\Rightarrow \exists o'' \in M', s'' \in \mu_{prv}(o', I(G)): s'' \in \mu_{req}(o'', I(G))$

[30] Aus Gründen der besseren Unterscheidbarkeit werden bei der Darstellung dieses etwas komplexeren Graphen Stichworte durch Buchstaben indiziert.

....

$\Rightarrow \exists o^x \in M', s^x \in \mu_{prv}(o^x, I(G)): s \in \mu_{req}(m, I(G))$

$\Rightarrow o^x \sim^G \sim > m \Rightarrow ... \Rightarrow o'' \sim^G \sim > m \Rightarrow o' \sim^G \sim > m$

$\Rightarrow \mathbf{o \sim^G \sim > m}$ (Widerspruch!)

Der Beweis für die Vorgängermengen von Stichworten funktioniert analog, da er sich auf den oben angegebenen Beweis für Medienobjekte zurückführen läßt:

$\forall s \in S(G): (s \lozenge! G(M', I(G)) \Rightarrow \exists m \in M(G), m \in \sigma_{prv}(s,G) \wedge m \lozenge G(M', I(G)))$

Satz

Die Anzahl der in der Vorgängermenge eines Medienobjektes m enthaltenen Medienobjekte ist die obere Schranke für jedes M(G) mit $m \lozenge! G$. In Kombination mit der minimalen Tiefe eines Medienobjekts läßt sich für jedes $m \in M(G)$ folgende Abschätzung bezüglich der Größe seines minimal vermittelnden Subgraphen treffen:

$\forall G' \subseteq G: (m \lozenge! G' \Rightarrow \mu_{depth}(m,G) \leq |M(G')| \leq |\mu_{ps}(m,G)|)$ (6.28a)

Analog gilt für Stichworte:

$\forall G' \subseteq G: (s \lozenge! G' \Rightarrow \sigma_{depth}(s,G) \leq |M(G')| \leq |\sigma_{ps}(s,G)|)$ (6.28b)

Falls die Kardinalität der Vorgängermenge eines Objektes identisch mit seiner minimalen Tiefe ist, bedeutet dies, daß das Objekt in seiner Vorgängermenge minimal erlernbar ist:

$\forall m \in M(G): (\mu_{depth}(m,G) = |\mu_{ps}(m,G)| \Leftrightarrow m \lozenge! G(\mu_{ps}(m), I(G)))$ (6.29)

$\forall s \in S(G): (\sigma_{depth}(s,G) = |\sigma_{ps}(s,G)| \Leftrightarrow s \lozenge! G(\sigma_{ps}(s), I(G)))$

Für die Heuristiken zur Auswahl von Medienobjekten sind vor allem Eigenschaften von Vorgängermengen interessant, die zwei Medienobjekte bzw. zwei Stichworte betreffen.

Die oben bewiesene Aussage, daß die Vorgängermenge eines Medienobjektes immer eine Obermenge aller dieses Objekt minimal vermittelnden Mengen ist, läßt sich auch auf beliebige Mengen von Medienobjekten und Stichworten verallgemeinern:

$\forall G' \subseteq G, \forall M \subseteq M(G'): (M \lozenge\lozenge! G' \Rightarrow M(G') \subseteq \bigcup_{m \in M} \mu_{ps}(m,G))$ (6.30a)

$\forall G' \subseteq G, \forall S \subseteq S(G'): (S \lozenge\lozenge! G' \Rightarrow M(G') \subseteq \bigcup_{s \in S} \sigma_{ps}(s,G))$ (6.30b)

Angewandt auf den gesuchten Subgraphen G^{b*} bedeutet dies:

Satz

Der gesuchte Subgraph G^{b*} eines an eine Blockbeschreibung b angepaßten Abhängigkeitsgraphen ist in der Vereinigung der Vorgängermengen aller in b festgelegten Lernziele enthalten.

$$M(G^{b*}) \subseteq \bigcup_{s \in b\_content(b)} \sigma_{ps}(s) \quad (6.31)$$

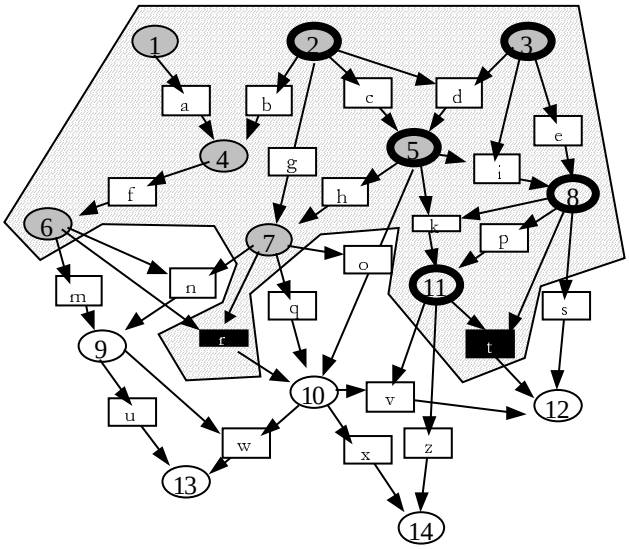
Beweis:

G^{b*} ist minimal vermittelnd bezüglich b_content(b) [siehe Kapitel 6.2]
$\Rightarrow \forall m \in M(G^{b*}): \exists s \in$ b_content(b) mit $m \sim^G \sim > s$ [siehe 6.13, 6.22]
$\Rightarrow \forall m \in M(G^{b*}): \exists s \in$ b_content(b) mit $m \in \sigma_{ps}(s)$ [siehe 6.26]
$\Rightarrow M(G^{b*}) \subseteq \bigcup_{s \in b\_content(b)} \sigma_{ps}(s)$ (q.e.d)

In der Praxis hat diese Eigenschaft zur Folge, daß zur Suche nach G^{b*} nicht der gesamte Abhängigkeitsgraph eines Blocks betrachtet werden muß, sondern nur ein aus den Vorgängermengen der gesuchten Stichworte bestehender Subgraph.

Als Beispiel sei angenommen, daß in dem gegebenen Abhängigkeitsgraphen eines Blocks b ein Subgraph gesucht ist, in dem die Stichworte b_content(b) = {r,t} minimal erlernbar sind (Abbildung 6-14). Die Vorgängermenge von r besteht aus den Medienobjekten 1, 2, 3, 4, 5, 6, 7 (grau eingefärbt), die von t aus 2, 3, 5, 8, 11 (dick umrandet). Hieraus ergibt sich für G^{b*}, daß $M(G^{b*}) \subseteq \{ 1, 2, 3, 4, 5, 6, 7, 8, 11 \}$.

Abbildung 6-14: Vereinigung der Vorgängermengen von Stichwort „r“ (grau) und Stichwort „t“ (umrandet)



Neben der Vereinigung zweier Vorgängermengen, liefert auch deren Schnitt Informationen über den gesuchten Abhängigkeitsgraphen G^{b*}. Falls es einen hierarchisch strukturierten Abhängigkeitsgraphen zur Erlernbarkeit zweier Medienobjekte (Stichworte) gibt, der nicht aus disjunkten Teilgraphen besteht, so schneiden dessen Medienobjekte die Schnittmenge der Vorgängermengen der beiden Medienobjekte (Stichworte):

$$\forall m, o \in M(G): (p \in \mu_{ps}(m,G) \cap \mu_{ps}(o,G) \Rightarrow p \sim^{G} \sim > m \wedge p \sim^{G} \sim > o) \qquad (6.32a)$$

$$\forall s, t \in S(G): (m \in \sigma_{ps}(s,G) \cap \sigma_{ps}(t,G): m \sim^{G} \sim \sim > s \wedge m \sim^{G} \sim \sim > t) \qquad (6.32b)$$

Die Wurzel eines solchen hierarchischen Subgraphen - so er denn existiert - ist Bestandteil von $\mu_{ps}(m,G) \cap \mu_{ps}(o,G) \cap M_{root}(G)$ bzw. $\sigma_{ps}(s,G) \cap \sigma_{ps}(t,G) \cap M_{root}(G)$.

Angewandt auf den Beispielgraphen bedeutet dies:

Falls ein hierarchisch strukturierter Subgraphen $G^{b*} \subseteq G$ mit {r, t} !◊◊ G^{b*} existiert, so gilt $M(G^{b*}) \cap \sigma_{ps}(r,G) \cap \sigma_{ps}(t,G) \neq \varnothing$. Darüber hinaus ist die Wurzel von G^{b*} in der Schnittmenge $\sigma_{ps}(r,G) \cap \sigma_{ps}(t,G)$ enthalten (Abbildung 6-15).

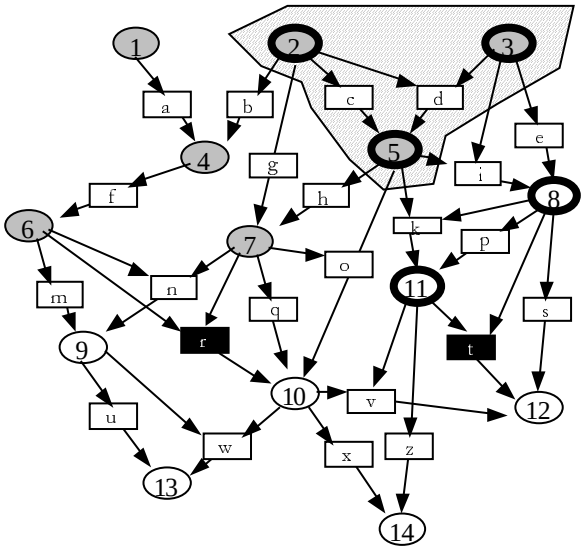


Abbildung 6-15: Schnitt der Vorgängermengen von Stichwort „r“ (grau) und Stichwort „t“ (umrandet)

Im Gegensatz zu dem vorgenannten Fall besagt eine leere Schnittmenge der Vorgängermengen zweier Medienobjekte bzw. Stichworte, daß die minimal vermittelnden Subgraphen dieser Objekte disjunkt sind:

$$\forall m, o \in M(G): (\mu_{ps}(m,G) \cap \mu_{ps}(o,G) = \varnothing \Rightarrow \neg\exists p \in M(G): p \sim^{G} \sim > m \text{ und } p \sim^{G} \sim > o) \qquad (6.33)$$

Für die automatische Auswahl und Strukturierung von Medienobjekten haben solche disjunkten Subgraphen zwei Auswirkungen:

1. G^{b*} ist kein einzelner zusammenhängender Graph, sondern vielmehr eine Menge von Graphen.
2. Die Berechnung von G^{b*} kann auf die Berechnung der Einzelgraphen reduziert werden:

$$\forall m, o \in M(G): (\mu_{ps}(m,G) \cap \mu_{ps}(o,G) = \varnothing \Rightarrow \{m,o\} \lozenge\!! G^{o} \cup G^{m} \Leftrightarrow m \lozenge\!! G^{m} \wedge o \lozenge\!! G^{o}) \qquad (6.34)$$

Neben den aufgeführten existieren noch viele weitere Eigenschaften von Vorgängermengen, die für die Auswahl und Strukturierung von Medienobjekten oder für die Seiten- und Kapitelerstellung ausgenutzt werden. Da diese Eigenschaften jedoch zumeist nur zur Lösung eines speziellen Problems herangezogen werden, findet sich ihre Beschreibung in den entsprechenden Abschnitten (7.2.3, 7.3.4, 7.4.3 und 7.6.3).

6.5.2 Berechnung von Vorgängermengen

Die Vorgängermenge eines Medienobjektes m kann innerhalb eines Abhängigkeitsgraphen G rekursiv über die Menge der direkten Vorgänger $\mu_{pre}(m,G)$ definiert werden:

$$\mu_{ps}(m,G) = \mu_{pre}(m,G) \cup \bigcup_{o \in \mu_{pre}(m,G)} \mu_{ps}(o,G) \qquad (6.35)$$

Die Vorgängermenge eines Stichwortes s ergibt sich aus der Vereinigung aller vermittelnden Medienobjekte samt deren Vorgängermengen:

$$\sigma_{ps}(s,G) = \bigcup_{m \in \sigma_{prv}(s,I(G))} (\{m\} \cup \mu_{ps}(m,G)) \qquad (6.36)$$

Ausgehend von dieser rekursiven Definition einzelner Vorgängermengen kann die Bestimmung der Vorgängermengen aller Knoten des Abhängigkeitsgraphen relativ effizient durch Zwischenspeicherung bereits berechneter Vorgängermengen durchgeführt werden. Interessant sind hierbei vor allem die Vorgängermengen der Medienobjekte, da nur zu ihrer Berechnung eine Rekursion nötig ist. Die Vorgängermengen der Stichworte können anschließend durch Vereinigung der Vorgängermengen der jeweiligen vermittelnden Medienobjekte bestimmt werden. Aus diesem Grund bietet es sich an, zur

Berechnung der Vorgängermengen anstatt des Abhängigkeitsgraphen G den zugehörigen Medienobjektgraphen G^M heranzuziehen.

Der Aufwand zur Berechnung der Vorgängermengen sämtlicher Medienobjekte und Stichworte eines Abhängigkeitsgraphen G ist quadratisch; er entspricht dem Produkt der Knoten und Kanten des zugehörigen Medienobjektgraphen G^M.

6.6 Hierarchische Subgraphen

Ein Bestandteil des Nutzungsszenarios ist die gewünschte Struktur eines jeden Blocks des zu erzeugenden Lehr- bzw. Informationssystems. Insbesondere die Erzeugung hierarchischer oder sequentieller Strukturen ist nicht trivial, da hier bei der Auswahl der geeignetsten Medienobjekte immer auch auf strukturelle Aspekte Rücksicht genommen werden muß. Diese Anforderung ist nur sehr schwer zu erfüllen, da es sich bei dem verwendeten Suchverfahren um einen lokalen Suchalgorithmus handelt, d.h. die Struktur des Informationssystems wird schrittweise entwickelt, wodurch zu keinem Zeitpunkt gesicherte Aussagen über deren endgültiges Aussehen möglich sind.

Aus diesem Grund hat es sich als zweckmäßig erwiesen, bereits vor der Auswahl und Strukturierung der Medienobjekte alle im Abhängigkeitsgraphen existierenden hierarchischen (d.h. baum-ähnlich strukturierten) Subgraphen zu berechnen und diese Information bei der Auswahl der geeignetsten Medienobjekte zu berücksichtigen.

Die einfachste Form eines hierarchischen Subgraphen ist der sog. **hierarchische Wurzelgraph**. Ein Wurzelgraph ist ein abgeschlossener Subgraph des Abhängigkeitsgraphen, der die folgenden Bedingungen erfüllt:

- er enthält genau einen Wurzelknoten,
- er besteht aus mindestens drei Medienobjekten,
- es gibt einen Pfad vom Wurzelknoten zu allen anderen enthaltenen Knoten,
- die Menge der Medienobjekte auf jedem von der Wurzel ausgehenden Pfad bildet einen abgeschlossenen Subgraphen.

Etwas formaler lassen sich diese drei Eigenschaften eines hierarchischen Wurzelgraphen wie folgt formulieren:

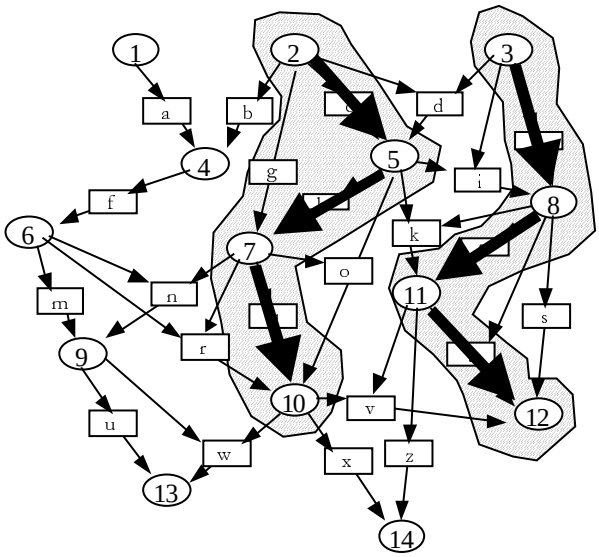
Definition

$O \subseteq M(G)$ ist ein **hierarchischer Wurzelgraph** von G, wenn

1. $|O| > 2$ und $!\exists\, w \in O, \mu_{req}(w,G) = \varnothing$ und
2. $\forall m \in O, m \neq w: \{w \sim^{G} \sim> m\} \subseteq O$ und
3. $m\ !\Diamond\ G(\{w \sim^{G} \sim> m\}, I(G))$ (6.38)

Abbildung 6-16 zeigt die beiden Wurzelgraphen des im vorherigen Kapitel verwendeten Abhängigkeitsgraphen.

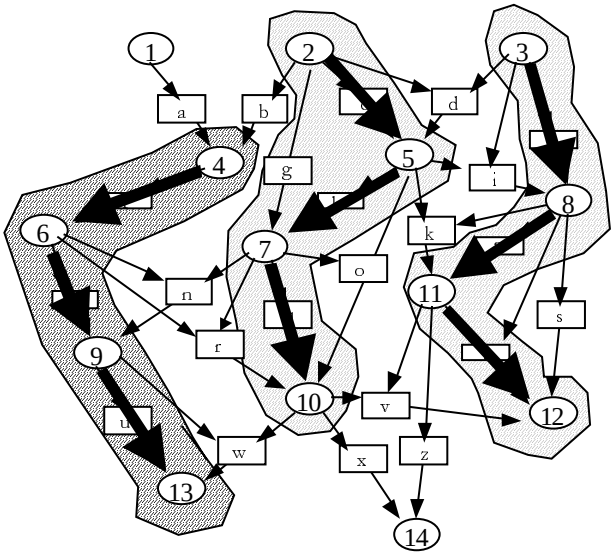
Abbildung 6-16: Hierarchische Wurzelgraphen



Hierarchische Subgraphen innerhalb eines Abhängigkeitsgraphen müssen jedoch nicht zwangsläufig einen Wurzelknoten des Abhängigkeitsgraphen als Ausgangspunkt haben. Ausschlaggebend für die Existenz eines hierarchischen Subgraphen ist vielmehr die Eigenschaft, daß ein Knoten innerhalb eines sequentiellen Pfades von Medienobjekten erlernbar ist, wenn die Wurzel des Pfades erlernbar ist. Diese Forderung ist bei hierarchischen Wurzelgraphen implizit durch die Abgeschlossenheit der Wurzelpfade erfüllt.

Abbildung 6-17 stellt alle, diese allgemeine Forderung erfüllenden, **hierarchischen Subgraphen** des Beispiel-Abhängigkeitsgraphen dar:

Abbildung 6-17: Hierarchische Subgraphen



Die Medienobjekte 4, 6, 9 und 13 bilden einen hierarchischen Subgraphen, der kein Wurzelgraph ist. Für die Auswahl und Strukturierung von Medienobjekten bedeutet dies, daß wenn Medienobjekt 4 erlernbar ist, dann kann darauf basierend eine hierarchische (hier sogar sequentielle) Struktur aufgebaut werden, so daß auch die Medienobjekte 6, 9 und 13 erlernbar sind.

Diese Einschränkung ("wenn Medienobjekt 4 erlernbar ist") führt zu folgender, allgemeinen Definition eines hierarchischen Subgraphen:

Definition

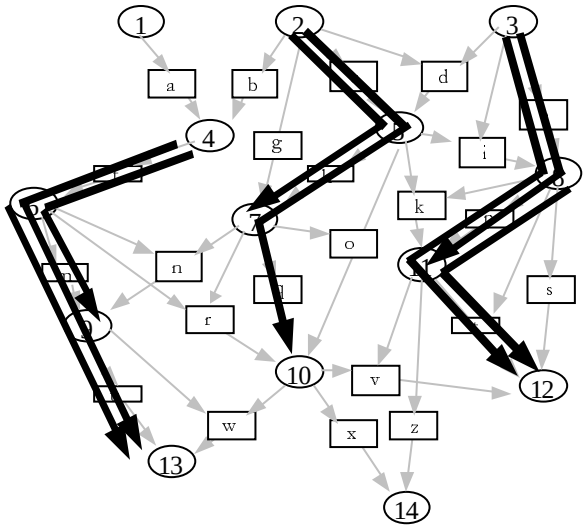
$O \subseteq M(G)$ bildet die Knotenmenge eines **hierarchischen Subgraph** von G, wenn (6.39)

$!\exists w \in O: |O| > 2$ und $\forall m \in O, m \neq w: \{w \sim^{G} \sim > m\} \subseteq O$ und

$$\forall i \in \mu_{req}(m,I(G)): \exists j \in \left(\bigcup_{o \in \{w \sim^{G} \sim > m\}} \mu_{prv}(o,I(G)) \right) \cup \mu_{req}(w,I(G)) \text{ mit } req(i) \leq prv(j)$$

Abbildung 6-18 zeigt alle im Beispielgraph enthaltenen hierarchischen Subgraphen. Hierbei fällt auf, daß hierarchische Subgraphen oftmals selbst wieder Subgraphen besitzen, die ihrerseits hierarchische Subgraphen sind.

Abbildung 6-18: Alle im Beispielgraphen enthaltenen hierarchischen Subgraphen



Die Menge aller in einem Abhängigkeitsgraphen G enthaltenen hierarchischen Subgraphen wird als HSG(G) bezeichnet:

$$HSG(G) := \{ M \subseteq M(G) \mid M \text{ ist hierarchischer Subgraph von } G \} \quad (6.40)$$

Die Grundidee des Algorithmus zur Berechnung der hierarchischen Subgraphen ist, in einem ersten Schritt alle Medienobjekte zu identifizieren, die Wurzel eines hierarchischen Subgraphen sein können und anschließend beginnend mit diesen Objekten eine Tiefensuche im Abhängigkeitsgraph durchzuführen. Ein Medienobjekt kann dabei nur dann Wurzel eines hierarchischen Subgraphen sein, wenn es mindestens einen Nachfolger besitzt, dessen verlangtes Vorwissen durch die vermittelten und verlangten Stichworte des potentiellen Wurzelknotens abgedeckt wird.

Ausgehend von den so ermittelten potentiellen Wurzeln können die kompletten hierarchischen Subgraphen durch eine rekursive Tiefensuche berechnet werden.

Der hauptsächliche Aufwand bei der Berechnung der hierarchischen Subgraphen eines Abhängigkeitsgraphen ist in der Ermittlung der Wurzeln der Subgraphen enthalten. Hierzu muß jedes Medienobjekt (O(M)) gesondert mit jedem seiner Nachfolger (maximal O(I/M)) betrachtet werden (O(Eingang(G)²). Hieraus folgt O(I * Eingang(G)²) als obere Grenze des Aufwands zur Bestimmung aller Wurzeln der in G enthaltenen hierarchischen Subgraphen. Das anschließende Verfolgen der Subgraphen hat im ungünstigsten Fall den gleichen Aufwand.

Anzahl, Größe und Struktur (Sequenz oder Baum) der in einem Abhängigkeitsgraphen enthaltenen hierarchischen Subgraphen hängt sehr stark von den Quellen der verfügbaren Medienobjekte und vom Detaillierungsgrad der Indizierung ab. Je undetaillierter der Index ist (d.h. je weniger Stichworte indiziert

wurden) und je mehr verschiedene Quellen verwendet wurden, desto weniger hierarchische Subgraphen werden potentiell im Abhängigkeitsgraphen enthalten sein.

7 Auswahl und Strukturierung

Without an intimate connection between our knowledge and our intentions, logic leads to madness, not intelligence. A logical system without a goal will merely generate an endless host of pointless truths like these: [...] If 4 is 5, then pigs can fly.

Marvin Minsky [Minsky88]

Kern der *Bottom-Up* Generierung von hypermedialen Lehr- und Informationssystemen ist die automatische Auswahl und Strukturierung von Medienobjekten. Hierdurch wird die Medienobjektstruktur des Dokuments erstellt, von der anschließend die Seitenstruktur und die Kapitelstruktur abgeleitet werden können. Auswahl und Strukturierung werden blockweise durchgeführt, d.h. für jeden im aktuellen Nutzungsszenario definierten Block wird eine eigene Medienobjekt-, Seiten- und Kapitelstruktur erstellt. Die Verknüpfung der Blöcke untereinander geschieht über exportierte Medienobjekte und/oder Stichworte (siehe Kapitel 5.3.3).

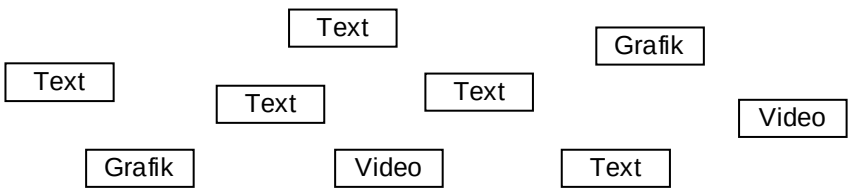
Dieses Kapitel beginnt mit einem Überblick über die notwendigen Schritte, um von einer vorgegebenen Menge von Medienobjekten zu einem Lehr- oder Informationssystem zu gelangen. Anschließend werden die einzelnen Teilalgorithmen, angefangen von der Auswahl und Strukturierung von Medienobjekten bis zur Erstellung der Kapitelstruktur, im Detail vorgestellt.

7.1 Übersicht

Die Abbildungen 7-1 bis 7-5 zeigen den Prozeß der **Bottom-Up Generierung** noch einmal im Überblick, wobei im Gegensatz zu der Darstellung in Kapitel 3 die Generierung von Blöcken als Ziel der automatisierten Auswahl und Strukturierung angenommen wird:

Für jeden zu erzeugenden **Block** werden zunächst die verfügbaren **Medienobjekte** bestimmt. Grundlage dieser Vorauswahl sind die Basisobjekte des Blocks in Abhängigkeit von den an das Vorwissen des aktuellen Nutzers angepaßten **Indexeinträgen** (Abbildung 7-1).

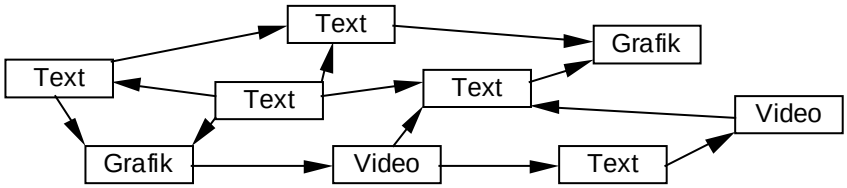
Abbildung 7-1: Für die Generierung eines Blocks eines Lehr- oder Informationssystem zur Verfügung stehende Medienobjekte



Aus den für die Erzeugung des Blocks verfügbaren Medienobjekten wird ein **Abhängigkeitsgraph** G^b erstellt, in dem alle Abhängigkeiten zwischen Medienobjekten, sowie zwischen Medienobjekten und Stichworten über Vorgänger-Nachfolger Beziehungen beschrieben sind (Abbildung 7-2).

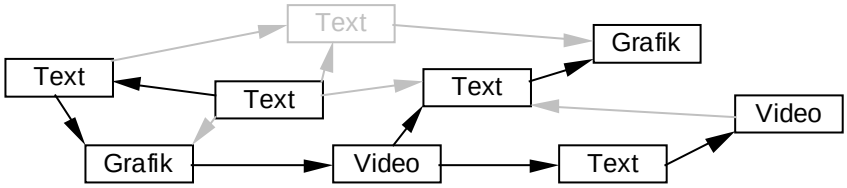
Durch die Berechnung von Maßzahlen wie z.B. der **minimalen Tiefe** oder der **Vorgängermenge** eines jeden Medienobjektes bzw. Stichwortes können zusätzliche Informationen über die Struktur des Abhängigkeitsgraphen und die Position einzelner Objekte innerhalb des Graphen gewonnen werden.

Abbildung 7-2: Erstellen eines Abhängigkeitsgraphen



Anhand von Heuristiken, die auf den im vorangegangenen Kapitel beschriebenen Maßzahlen basieren, werden eine Reihe von Medienobjekten ausgewählt. Die ausgewählten Objekte bilden einen **abgeschlossenen Subgraphen**, in dem alle zu vermittelnden Stichworte erlernbar sind. Anschließend wird durch Entfernen bzw. Hinzufügen von einzelnen Medienobjekten und Kanten aus dem zugehörigen Medienobjektgraph die **Medienobjektstruktur** des Blocks berechnet (Abbildung 7-3).

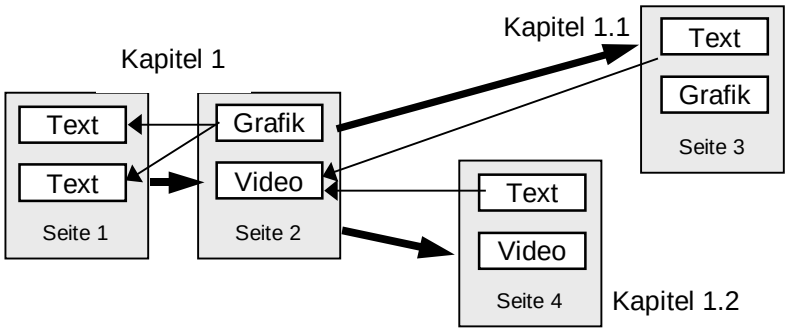
Abbildung 7-3: Auswahl und Strukturierung von Medienobjekten



Die Medienobjektstruktur kann durch Zusammenfassen von benachbarten Knoten in eine **Seitenstruktur** und anschließend durch Zusammenfassen von benachbarten Seiten in eine logische **Kapitelstruktur** überführt werden (Abbildung 7-4).

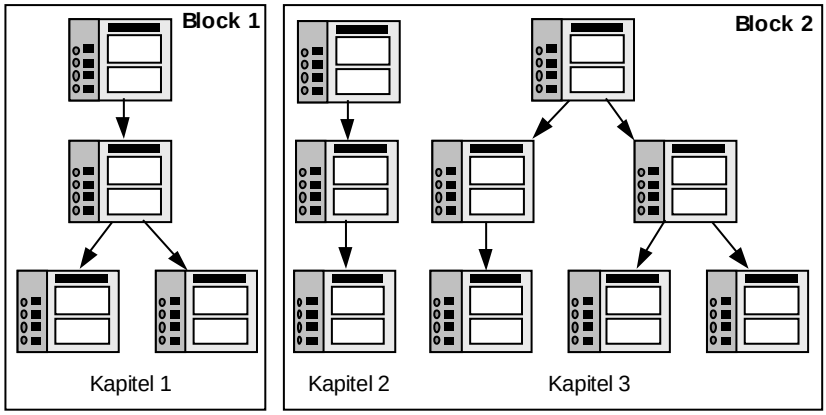
Abbildung 7-4: Erstellen von Seiten- und Kapitelstruktur

(dicke Pfeile = Navigation, dünne Pfeile = Hyperlinks)



Abschließend werden die Kapitelstrukturen aller Blöcke zu einer Gesamt-Kapitelstruktur zusammengefaßt und die einzelnen Seiten in Form von HTML Dateien auf einem Web-Server abgelegt. Hierbei werden auch alle benötigten Navigationselemente erzeugt und integriert (Abbildung 7.5).

Abbildung 7-5: Zusammenfassung der generierten Strukturen mit Navigationselemente zu HTML-Seiten



Der nachfolgend skizzierte Algorithmus „CreateDocument“ stellt die Basis der programmiersprachlichen Umsetzung dieses Modells dar. Alle benötigten Teilalgorithmen wie z.B. die Auswahl und Strukturierung von Medienobjekten sind in diesen übergeordneten Rahmen eingebettet.

Algorithmus CreateDocument

Aufgabe:	Erstellen einer Web-Site durch Auswahl und Strukturierung von Medienobjekten unter Berücksichtigung eines angegebenen Nutzungsszenarios
Eingabe:	Menge von Medienobjekten M
	Menge von Stichworten S
	Index I
	aktuelles Nutzungsszenario u^c
Ausgabe:	Hypermediales Lehr- bzw. Informationssystem (Web-Site)

[for] Für jeden Block des Nutzungsszenarios:

Aufbau eines blockspezifischen, normalisierten Abhängigkeitsgraphen [Kapitel 7.2]

Auswahl der geeignetsten Medienobjekte (Select)

Erstellen der Medienobjektstruktur (Sequence)

Erstellen der Seitenstruktur

Erstellen der Kapitelstruktur

Erstellen von HTML Seiten

[end for]

Erstellen einer übergeordneten Index-Seite

Alle für die automatisierte Auswahl und Strukturierung von Medienobjekten wichtigen Algorithmen werden in diesem Kapitel beschrieben. Dabei werden zuweilen für die Praxis kaum relevante oder zu spezielle Sonderfälle gar nicht oder nur am Rande diskutiert. Hierzu zählen z.B. die Berücksichtigung einiger Medienobjekt-Eigenschaften bei der Auswahl und Strukturierung oder die Einbindung von bestimmten Medienobjekt-Genres bei der Erstellung der Seitenstruktur.

7.2 Normalisierung des Abhängigkeitsgraphen

Der erste Schritt bei der Berechnung eines Lehr- oder Informationssystems ist die Normalisierung des zugrundeliegenden Abhängigkeitsgraphen.

Definition

Ein Abhängigkeitsgraph wird als **normalisiert** bezeichnet, wenn er **zyklenfrei** und **abgeschlossen** ist. Darüber hinaus sollte er nur Knoten enthalten, die für die Erstellung des gewünschten Dokuments benötigt werden.

Der nachfolgende Algorithmus "NormalizeDependencyGraph" normalisiert einen gegebenen Abhängigkeitsgraphen. Den ersten Schritt der Normalisierung bildet die Auflösung aller eventuell im Graph enthaltenen Zyklen (siehe Kapitel 6.4.2). Anschließend werden die minimalen Tiefen aller im Abhängigkeitsgraph enthaltenen Medienobjekte und Stichworte berechnet und alle nicht erlernbaren Knoten identifiziert und entfernt. Gleiches geschieht anhand der Vorgängermengen mit Medienobjekten, die zwar erlernbar, für die Vermittlung der gewünschten Inhalte aber nicht relevant sind. Abschließend werden die hierarchischen Subgraphen berechnet, so daß nach der Normalisierung auch alle in Kapitel 6 beschriebenen Maßzahlen des Abhängigkeitsgraphen bestimmt sind.

Algorithmus NormalizeDependencyGraph

Aufgabe:	Normalisierung eines Abhängigkeitsgraphen und Berechnung der wichtigsten Maßzahlen der enthaltenen Medienobjekte und Stichworte
Eingabe:	Abhängigkeitsgraph
	Blockbeschreibung
Ausgabe:	Normalisierter Abhängigkeitsgraph

Entfernen von Zyklen

Berechnen der Minimalen Tiefen

Entfernen nicht erlernbarer Knoten

Berechnen der starken Zusammenhangskomponenten

Entfernen nicht benötigter Medienobjekte

Berechnen der hierarchischen Subgraphen

Die Algorithmen zur Berechnung der minimalen Tiefen, der Vorgängermengen und der hierarchischen Subgraphen wurden bereits in Kapitel 6 ausführlich vorgestellt. Die drei noch fehlenden Algorithmen zur Entfernung unerreichbarer und nicht benötigter Knoten, sowie zur Vermeidung von Zyklen sind Inhalt der folgenden Abschnitte.

7.2.1 Vermeidung von Zyklen

Im Abhängigkeitsgraphen enthaltene Zyklen sind für die *Bottom-Up* Erstellung von Lehr- und Informationssystemen aus zweierlei Gründen problematisch:

1. Sie können zu Zyklen in der Medienobjektstruktur führen, was wiederum Zyklen in den übergeordneten Strukturen zur Folge haben kann. Zyklen in der Seiten- und Kapitelstruktur sind jedoch unbedingt zu vermeiden, da sie inhaltlich wenig instruktiv sind.
2. Zyklen verkomplizieren die Algorithmen zur Berechnung der benötigten Maßzahlen (z.B. Vrgängermengen) und erschweren die Auswahl und Strukturierung von Medienobjekten.

Aus diesen Gründen werden im Rahmen der Normalisierung alle Zyklen aus dem Abhängigkeitsgraphen entfernt.

Hierbei sind drei Fälle zu unterscheiden:

1. Im Fall von Sackgassen (*deadends*) müssen die betroffenen Medienobjekte samt ihrer Indexeinträge aus dem Abhängigkeitsgraphen entfernt werden.
2. Falls ein Zyklus durch einen einzelnen Knoten aufgelöst werden kann (*resolver*), entfernt der Algorithmus alle innerhalb des Zyklus liegenden Ausgangskanten des betreffenden Medienobjekts.
3. Fall ein Zyklus keinen *resolver* enthält, muß der Abhängigkeitsgraph so modifiziert werden, daß alle *Connectoren* des Zyklus ausschließlich Nachfolgerknoten besitzen, die nicht auf dem Zyklus liegen.

In den Fällen 2 (Existenz eines *Resolvers*) und 3 (Existenz von *Connectoren*) müssen aus dem Abhängigkeitsgraphen alle Kanten entfernt werden, die das betreffende Objekt mit Nachfolgerknoten innerhalb des Zyklus verbinden.

Der zur Identifizierung der *Resolver* und *Connectoren* durchschnittlich benötigte Aufwand beträgt inklusive der Entfernung von deren auf dem Zyklus liegenden Ausgangskanten O(I * Ausgang(G)). Hierin ist allerdings der Aufwand für die Berechnung der starken Zusammenhangskomponenten nicht enthalten.

7.2.2 Entfernen nicht erlernbarer Medienobjekte

Bei der Erstellung des an eine gegebene Blockbeschreibung angepaßten Abhängigkeitsgraphen wird eine Abhängigkeitsstruktur aus allen zur Verfügung stehenden Medienobjekten und den von diesen referenzierten Stichworten erstellt. Hierbei spielt die Erlernbarkeit der enthaltenen Objekte und damit auch die Abgeschlossenheit des Abhängigkeitsgraphen keine Rolle.

Falls ein Abhängigkeitsgraph G nicht abgeschlossen ist, bedeutet dies, daß zumindest ein in ihm enthaltenes Stichwort nicht mit den zur Verfügung stehenden Medienobjekten erlernbar ist:

$$G \text{ ist nicht abgeschlossen} \Leftrightarrow \exists s \in S(G): s \neg\Diamond G \tag{7.1}$$

Falls ein Stichwort nicht erlernbar ist, bedeutet dies automatisch, daß auch alle Medienobjekte, die dieses Stichwort als Vorwissen verlangen, ebenfalls nicht erlernbar sind. Nicht erlernbare Medienobjekte wiederum können eine Ursache für weitere nicht erlernbare Stichworte sein

Alle in G^b nicht erlernbaren Medienobjekte sind auch in keinem Subgraphen von G^b erlernbar, d.h. sie können auch niemals Bestandteil des gesuchten Subgraphen G^{b*} sein:

$$m \in M(G), m \neg\Diamond G \Rightarrow \forall G' \subseteq G, m \in M(G'): m \neg\Diamond G' \tag{7.2}$$

Da die bei der Auswahl und Strukturierung von Medienobjekten benutzten Algorithmen von einem abgeschlossenen Abhängigkeitsgraph ausgehen, werden bei der Normalisierung alle nicht erlernbaren Medienobjekte und Stichworte aus dem Abhängigkeitsgraphen entfernt[31].

Nicht erlernbare Medienobjekte und Stichworte können dadurch identifiziert werden, daß ihre minimale Tiefe 0 beträgt. Dies geht aus der Definition der minimalen Tiefen (Definition 6-23) hervor:

> Die minimale Tiefe eines Stichwortes ist nur dann gleich 0, wenn kein Subgraph innerhalb des Abhängigkeitsgraphen existiert, der dieses Stichwort vermittelt. Analog ist die minimale Tiefe eines Medienobjektes 0, wenn mindestens eines seiner verlangten Stichworte entweder nicht vorhanden oder nicht vermittelbar ist.

Die Abgeschlossenheit eines Subgraphen des Abhängigkeitsgraphen (bzw. des Abhängigkeitsgraphen selbst) kann somit auch über die minimalen Tiefen der in ihm enthaltenen Medienobjekte definiert werden:

$$\forall G' \subseteq G: (G' \text{ ist abgeschlossen} \Leftrightarrow \forall m \in M(G'): \mu_{depth}(m,G) > 0) \tag{7.3}$$

Der hier nicht näher ausgeführte Algorithmus "RemoveUnteachableNodes" entfernt aller Objekte mit einer minimalen Tiefe von 0 aus dem Abhängigkeitsgraphen, so daß ein abgeschlossener Abhängigkeitsgraph entsteht.

7.2.3 Äußere Hülle

Eine weitere Möglichkeit, den Abhängigkeitsgraphen zu verkleinern, besteht darin, alle Medienobjekte zu entfernen, die nicht zur Erlernbarkeit der gewünschten Inhalte eines Blocks beitragen. Die verbleibenden Medienobjekte, Stichworte und Indexeinträge werden als **äußere Hülle** des zu erstellenden Lehr- oder Informationssystems bezeichnet, da sie die kleinste im voraus berechenbare Obermenge der Medienobjektstruktur des fertigen Dokuments darstellen.

In Abschnitt 6.5.1 wurde als Obermenge für $M(G^{b*})$ eines Blocks b die Vereinigung der Vorgängermengen aller gewünschten Stichworte ermittelt:

$$M(G^{b*}) \subseteq \bigcup_{s \in \beta_{content}(b)} \sigma_{ps}(s,G)$$

[31] Auch wenn die verwendeten Algorithmen in der Lage wären, auf nicht abgeschlossenen Abhängigkeitsgraphen zu operieren, sollten vor der Auswahl und Strukturierung alle nicht-erlernbaren Medienobjekte und Stichworte entfernt werden, da diese sonst zu einer Verfälschung der inhaltlichen und strukturellen Bewertung von Medienobjekten führen können.

Alle Medienobjekte, die nicht in dieser Vereinigung enthalten sind, können auf keinen Fall Bestandteil von G^{b*} sein, da sie die Bedingung der minimalen Abgeschlossenheit von G^{b*} verletzen würden.

Die Verwendung der äußeren Hülle anstelle des kompletten Abhängigkeitsgraphen führt neben einem Geschwindigkeitsgewinn vor allem zu genaueren Bewertungen von Medienobjekten und Indizes. Insbesondere bei der vom Auswahlalgorithmus durchgeführten heuristischen Bewertung von Medienobjekten (siehe Kapitel 7.3.4) sollten nach Möglichkeit nur Indexeinträge berücksichtigt werden, die innerhalb der äußeren Hülle liegen, da nur diese für einen Vergleich von Medienobjekten relevant sind. Die Berücksichtigung außerhalb der äußeren Hülle liegender Indexeinträge würde dazu führen, daß Kriterien in die Bewertung einfließen, die für das zu erstellende Dokument nicht relevant sind[32].

7.3 Der Minesweeper Algorithmus

Ziel der Auswahl und Strukturierung ist es, in dem normalisierten Abhängigkeitsgraphen G^{bn} einen Subgraphen G^{b*} zu finden, in dem die gewünschten Inhalte minimal erlernbar sind. G^{b*} ist die Grundlage der Medienobjektstruktur des zu erstellenden Dokuments:

$$G^{b*} \subseteq G^{bn} \text{ mit } b\_content(b) \; !\lozenge\lozenge \; G^{b*} \qquad (7.4)$$

Darüber hinaus soll G^{b*} 'möglichst gut' den in der Blockbeschreibung b angegebenen Vorgaben bezüglich Medienobjekt-Auswahl, Struktur und Kompaktheit entsprechen.

Aus diesen Anforderungen läßt sich folgende, Vorgehensweise zur Ermittlung von G^{b*} ableiten:

1. Finde alle $G^{b+} \subseteq G^{bn}$ für die gilt: $b\_content(b) \; !\lozenge\lozenge \; G^{b+}$
2. Bewerte alle gefundenen G^{b+} bezüglich der in der Blockbeschreibung angegebenen Kriterien "Struktur" und "Kompaktheit"
3. Wähle den bestbewerteten Subgraphen aus.

Dieses Verfahren ist jedoch nicht praktikabel, da es in jedem Abhängigkeitsgraphen potentiell bis zu $2^{|M(G)|}$ Subgraphen geben kann, in denen die gewünschten Inhalte minimal erlernbar sind. Neben dem Aufwand, diese Subgraphen zu ermitteln, ist eine Bewertung aller gefundenen G^{b+} nicht in vertretbarer Zeit zu leisten.

Aus diesem Grund wird zur Auswahl der Medienobjekte eine Strategie verfolgt, mit der nur die Erfüllung der formalen Bedingungen (Abgeschlossenheit, alle Inhalte minimal erreichbar) gewährleistet ist. Alle anderen, nicht zwingenden Kriterien wie z.B. die gewünschte Struktur, fließen nur in Form von Bewertungsfunktionen in den Auswahlalgorithmus ein. Mit dieser Verfahrensweise wird potentiell nicht der bestmögliche Subgraph G^{b*} gefunden. Durch gute Heuristiken und geschickte Wahl der Bewertungsfunktionen ist es jedoch sehr wahrscheinlich, eine gute Annäherung an einen optimalen Subgraphen zu erzielen.

Diese Überlegungen führen zu dem folgenden, dreistufigen Auswahlverfahren:

1. Wähle einen möglichst kleinen, an die Blockbeschreibung b "angepaßten" Subgraphen $G^{b+} \subseteq G^{bn}$ aus, für den gilt: $b\_content(b) \; \lozenge\lozenge \; G^{b+}$.
2. Entferne alle redundanten Medienobjekte aus G^{b+}, so daß für den resultierenden Subgraphen G^{b*} gilt: $b\_content(b) \; !\lozenge\lozenge \; G^{b*}$.
3. Erweitere G^{b*} um möglichst viele der in der Blockbeschreibung angegebenen Zusatzobjekte.

Für den ersten Teilalgorithmus, die Ermittlung von G^{b+}, wird eine sog. *Best-First* Suche [Nilsson98] verwendet, die anhand heuristischer Bewertungsfunktionen ein Medienobjekt nach dem anderen auswählt, bis ein abgeschlossener Subgraph gefunden ist, in dem alle gewünschten Inhalte erlernbar sind. Durch das

[32] Ein Kriterium bei der Bewertung von Medienobjekten ist z.B. Anzahl und Größe von dieses Objekt enthaltenen hierarchischen Subgraphen. Relevant für die Bewertung sind jedoch im Grunde genommen nur die Subgraphen der hierarchischen Subgraphen, die auch in der Medienobjekt-Struktur des zu erzeugenden Dokuments enthalten sein können.

anschließende Entfernen redundanter Medienobjekte entsteht ein Subgraph G^{b*}, dessen Medienobjekte die gewünschten Kriterien Abgeschlossenheit und minimale Erreichbarkeit erfüllen. Das abschließende Hinzufügen von Zusatzobjekten führt zu dem Abhängigkeitsgraphen, der die Basis der Medienobjektstruktur des zu generierenden Blocks bildet.

7.3.1 Der Basisalgorithmus

Wie im vorangegangenen Abschnitt erwähnt, basiert der Algorithmus zur Auswahl der den Subgraphen G^{b+} bildenden Medienobjekte auf einer lokalen Bestensuche.

Eine **lokale Bestensuche** basiert auf einer eingeschränkten Breitensuche. Im Unterschied zur Breitensuche "bewegt" sich der Algorithmus nicht gleichmäßig in den Graph hinein, sondern besucht bevorzugt die Nachfolger (bzw. Vorgänger) von Knoten, die aufgrund von heuristischen, problemspezifischen Informationen als die besten zur Erreichung des vorgegebenen Ziels erscheinen [Nilsson98].

Übertragen auf die erste Teilaufgabe, einen aus einer Menge von Medienobjekten M^{b+} bestehenden Subgraph G^{b+} innerhalb des normalisierten Abhängigkeitsgraphen G^{bn} eines Blocks b zu finden, so daß b_content(b) $\Diamond\Diamond$ G^{b+}, bedeutet dies:

> Der Auswahlalgorithmus versucht - ausgehend von den vorgegebenen Inhalten – Schritt für Schritt einen Graph aus Medienobjekten aufzubauen, bis dieser abgeschlossen ist. Hierbei wird in jedem Schritt das Medienobjekt zu dem Graphen hinzugefügt, für das eine heuristischen Bewertungsfunktion den Schluß zuläßt, daß dieses Objekt in einem bestmöglichen Subgraphen enthalten ist.

Der **Auswahl-Algorithmus** "Select" ist im Kern sehr einfach, wird jedoch durch diverse Sonderfälle (z.B. vertiefte Stichworte) und Optimierungen (z.B. *Look-Ahead* Funktionen) erheblich kompliziert. Aus diesem Grund soll im folgenden zuerst der unoptimierte Basisalgorithmus vorgestellt werden. Anschließend wird der Basisalgorithmus sukzessiv um die Behandlung der Sonderfälle sowie einige Optimierungsmöglichkeiten erweitert.

Algorithmus Select (Basisversion)

Aufgabe: Auswahl eines Subgraphen eines gegebenen Abhängigkeitsgraphen, mit dem alle in einer gegebenen Blockbeschreibung angegeben Lernziele vermittelt werden können

Eingabe: normalisierter Abhängigkeitsgraph G^{bn}

Blockbeschreibung b

Ausgabe: Subgraph $G^{b+} \subseteq G^{bn}$ mit b_content(b) ◊◊ G^{b+}

Initialisierung der ausgewählten Medienobjekte und vermittelten Stichworte als leere Mengen

[while] solange die ausgewählten Medienobjekte keinen abgeschlossenen Subgraph bilden, in dem alle Lernziele der Blockbeschreibung erreichbar sind:

Heuristische Auswahl eines Medienobjektes (Algorithmus SelectBest)

Erweitern der Menge der ausgewählten Medienobjekte um dieses Medienobjekt

Reduzieren der Lernziele um die von dem Objekt vermittelten Stichworte

Erweitern der Lernziele um die von dem Objekt benötigten Stichworte, die nicht bereits von einem anderen ausgewählten Medienobjekt vermittelt werden

[end while]

Erstellen eines Abhängigkeitsgraphen aus den ausgewählten Medienobjekten

Das Laufzeitverhalten des Auswahlalgorithmus wird entscheidend von der Implementierung des Algorithmus *SelectBest* beeinflußt. Aus diesem Grund wird das kombinierte Laufzeitverhalten beider Algorithmen am Ende des nächsten Abschnitts diskutiert.

Falls aus anderen, bereits erstellten Blöcken exportierte Stichworte und/oder Medienobjekte bei der Erstellung eines Blocks als bekannt vorausgesetzt werden sollen, wird in Schritt 1 des Algorithmus die Menge der ausgewählten Medienonjekte mit den bekannten Medienobjekten initialisiert. Für alle weiteren Schritte des Algorithmus sind aus anderen Blöcken importierte Objekte nicht als solche erkennbar und von daher vollkommen transparent.

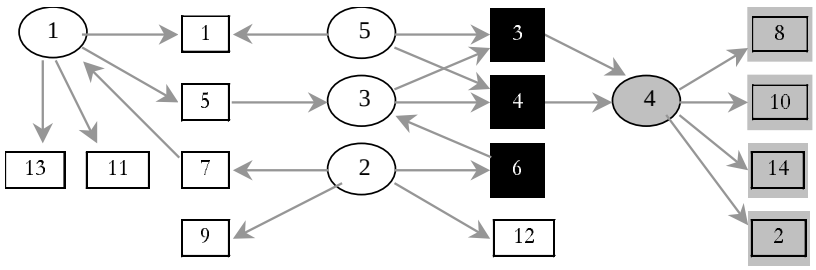
7.3.2 Auswahl anhand von Selektionswahrscheinlichkeiten

Ziel der heuristischen Bewertungsfunktion ist es, aus allen zur Verfügung stehenden Medienobjekten dasjenige auszuwählen, das am besten zum momentanen Zustand des Auswahl-Algorithmus und der Beschreibung des aktuell zu generierenden Blocks paßt. Hierzu wird jedes Medienobjekt mit einer numerischen Bewertung versehen, wobei ein größerer Wert eine bessere Bewertung repräsentiert.

Da der Subgraph G^{b+} von den vorgegebenen Inhalten in Richtung der Wurzeln des Abhängigkeitsgraphen aufgebaut wird, müssen in jeder Iteration des Algorithmus nicht alle verfügbaren Medienobjekte bewertet werden. Es reicht vielmehr, nur die Objekte zu bewerten, die sich an den Eingangskanten von G^{b+} befinden, d.h. die mindestens eines der offenen Stichworte vermitteln.

Im Abbildung 7-6 ist ein Zwischenzustand des Algorithmus "Select" dargestellt, in dem Medienobjekt 4 bereits ausgewählt wurde und die Stichworte 3, 4 und 6 die (noch) zu vermittelnden Lernziele darstellen. Die Medienobjekte 2, 3 und 5 vermitteln Informationen zu mindestens einem der offenen Stichworte, so daß eines von ihnen als das am besten passende ausgewählt werden wird. Es würde somit ausreichen, nur diese drei Objekte anhand diverser heuristischer Informationen zu bewerten und ausgehend von diesen Bewertungen eine Auswahl zu treffen.

Abbildung 7-6: Zustand von G^{b+} nach Auswahl des ersten Medienobjektes



$M^{b+} = \{ 4 \}$
$S^{req} = \{ 3, 4, 6 \}$ $S^{prv} = \{ 2, 8, 10, 14 \}$

Unter der Annahme, daß das Resultat der heuristischen Bestensuche identisch oder zumindest nahezu identisch mit der bestmögliche Lösung G^{b*} ist, kann man sogar noch einen entscheidenden Schritt weiter gehen. Die Folgerung aus dieser Annahme ist, daß jedes in M^{b+} enthaltene Medienobjekt mit hoher Wahrscheinlichkeit auch in G^{b*} enthalten ist. Hieraus wiederum folgt, daß alle zu einem beliebigen Zeitpunkt der Berechnung von M^{b+} offenen Stichworte auch in G^{b*} als verlangte Stichworte auftauchen. Dies jedoch bedeutet, daß aus jeder Gruppe von Medienobjekten, die eines der in M^{b+} offenen Stichworte vermitteln, mindestens eines in G^{b*} enthalten sein wird.

Bezogen auf den in Abbildung 7-6 dargestellten Zustand des Auswahlalgorithmus bedeutet dies:

- Medienobjekt 2 wird Element von $M(G^{b*})$ sein, da es als einziges das offene Stichwort 6 vermittelt.
- Zumindest eines der Medienobjekte 3 und 5 wird in $M(G^{b*})$ enthalten sein, da nur diese beiden Objekte das offene Stichwort 3 vermitteln.
- Zumindest eines der Medienobjekte 3 und 5 wird in $M(G^{b*})$ enthalten sein, da nur diese beiden Objekte das offene Stichwort 4 vermitteln.

Diese Überlegungen führen dazu, für jedes offene Stichwort dessen vermittelnde Medienobjekte miteinander zu vergleichen. Hierdurch kann eine Auswahlstrategie realisiert werden, die an die Gewinn-Strategie beim Spiel ***Minesweeper*** angelehnt ist:

> Bei diesem Spiel kommt es darauf an, die Position von Minen zu ermitteln, die per Zufallsgenerator in einem schachbrettartig aufgebauten Minenfeld versteckt wurden. Per Mausklick kann der Spieler beliebige Felder des Minenfelds auswählen. Enthält das gewählte Feld eine Mine, so ist das Spiel verloren. Trifft er auf ein „harmloses" Feld, so wird in diesem Feld angezeigt, wieviele der benachbarten Felder Minen enthalten.
>
> Eine mögliche Strategie ist es, anhand der angegebenen Zahlen für benachbarte Minen nach Feldern zu suchen, die entweder mit hoher Wahrscheinlichkeit eine Mine enthalten oder mit hoher Wahrscheinlichkeit keine Mine enthalten. Interessant sind dabei vor allem Felder, die mit hoher Wahrscheinlichkeit keine Minen enthalten, da sie nach Auswahl weitere Informationen über die Positionen der Minen liefern. Je mehr solcher Felder identifiziert werden können, desto mehr Informationen sind für die sichere Auswahl weiterer Felder und die Ermittlung neuer Informationen verfügbar.

Die Eigenschaft des Spiels *Minesweeper*, daß jedes ausgewählte Objekt zusätzliche Informationen liefert, findet sich auch bei der heuristische Bestensuche in Abhängigkeitsgraphen wieder:

> Je mehr offene Stichworte existieren, desto besser können Medienobjekte anhand der Vermittlung von Stichworten bewertet werden (z.B. Anzahl vermittelter, offener Stichworte). Je mehr Medienobjekte bereits ausgewählt wurden, desto erkennbarer ist die Struktur des entstehenden Subgraphen, d.h. desto besser können strukturelle Aspekte bewertet werden.

Aus diesem Grund verfolgt auch der Auswahl-Algorithmus die oben beschriebene *Minesweeper*-Strategie: es wird immer versucht, ein Medienobjekt auszuwählen, daß mit möglichst hoher Wahrscheinlichkeit in G^{b*} enthalten sein wird. Die Hoffnung dabei ist, daß durch Auswahl dieses Medienobjektes neue offene

Stichworte anfallen und Informationen über die Struktur von G^{b*} gewonnen werden, die die weitere Auswahl erleichtern.

Im Beispielfall wird somit auf jeden Fall Medienobjekt 2 als nächstes ausgewählt, da dieses als einziges das offene Stichwort 6 vermittelt. Seine Wahrscheinlichkeit, Bestandteil von G^{b*} zu sein, beträgt somit (unter der genannten Annahme) 100%.

Da von jedem offenen Stichwort mindestens eines der vermittelnden Medienobjekte Bestandteil von G^{b*} ist, versucht die verwendete Bewertungsfunktion in Analogie zur *Minesweeper*-Strategie, für jedes vermittelnde Medienobjekt die Wahrscheinlichkeit zu bestimmen, mit der es als Vermittler genau dieses Stichwortes in G^{b*} enthalten sein wird.

Formal bedeutet dies, daß die numerischen Bewertungen jedes ein Stichwort s vermittelnden Medienobjektes m in Relation zu den Bewertungen aller dieses Stichwort vermittelnden Medienobjekte gesetzt werden müssen. Den daraus resultierenden Quotienten kann man dann als **Wahrscheinlichkeit** dafür interpretieren, daß das entsprechende Medienobjekt Bestandteil von G^{b*} ist[33]:

$$p(m, s) := \frac{eval(m,s)}{\sum_{o \in \sigma_{moprv}(s)} eval(o,s)} \qquad (7.5)$$

Auf diese Weise kann stichwort-übergreifend mit Hilfe der für jede Medienobjekt-Stichwort Kombination berechneten Wahrscheinlichkeiten das global wahrscheinlichste – und somit am besten geeignete – Medienobjekt bestimmt werden.

Hierzu ein kurzes Beispiel:

> Gegeben sei ein Zustand des Auswahl-Algorithmus mit zwei offenen Stichworten a und b. Stichwort a wird von den drei Medienobjekte 1, 2 und 3 vermittelt. Die Bewertungen dieser Medienobjekte bezüglich Stichwort a sind 0.8, 0.7 und 0.5. Hieraus resultieren die Wahrscheinlichkeiten 0.4 für Medienobjekt 1, 0.35 für Medienobjekt 2 und 0.25 für Medienobjekt 3. Das offene Stichwort b wird von den beiden Medienobjekten 4 und 5 mit den Bewertungen 0.6 und 0.2 vermittelt. Die entsprechenden Wahrscheinlichkeiten sind 0.75 für Medienobjekt 4 und 0.25 für Medienobjekt 5. Obwohl die Bewertungen von Medienobjekt 1 und 2 die vom Betrag her besten sind, wird Medienobjekt 4 ausgewählt, da es am wahrscheinlichsten im gesuchten Subgraphen enthalten ist.

Um ein möglichst gleichmäßiges "Wachsen" des gesuchten Subgraphen von den offenen Stichworten zu den Wurzeln zu forcieren, werden die berechneten Selektionswahrscheinlichkeiten p(m, s) zusätzlich mit der minimalen Tiefe des betrachteten Stichwortes gewichtet:

$$q(m, s) = p(m, s) * g(s) \quad \text{mit} \quad g(s) = 1+\log 10(\sigma_{depth}(s))$$

Hierdurch werden bei ähnlichen Wahrscheinlichkeiten weiter von den Wurzeln entfernte Stichworte zuerst ausgewählt.

[33] Für den Sonderfall, daß nur zwei konkurrierende Medienobjekte betrachtet werden müssen, entsprechen die Bewertungen den Quotienten (odds) aus der subjektiven Wahrscheinlichkeitstheorie.

Algorithmus SelectBest

Aufgabe:	Auswahl des Medienobjekts mit der höchsten Selektionswahrscheinlichkeit
Eingabe:	normalisierter Abhängigkeitsgraph G^{bn}
	bereits ausgewählte Medienobjekte M^{b+}
	von diesen Medienobjekten vermittelte Stichworte S^{prv}
	Menge der offenen Stichworte S^{req}
	Blockbeschreibung b
Ausgabe:	Heuristisch best-bewertetes Medienobjekt m^{best}

[if] **wenn ein Lernziel existiert, das nur von einem Medienobjekt vermittelt wird:**
Auswählen dieses Medienobjekts mit q(m, s) = p(m, s) = 100%
[else] **ansonsten:**
[for] Für jedes noch zu vermittelnde Lernziel:
Bewertung aller dieses Lernziel vermittelnden Medienobjekte (MoEval)
Berechnung der Selektionswahrscheinlichkeit p(m, s) dieser Medienobjekte
Berechnung der normierten Selektionswahrscheinlichkeit q(m, s)
[end for]
Auswahl des Medienobjekts mit der höchsten Selektionswahrscheinlichkeit
[end if]

Im ungünstigsten Fall ist $G^{b+} = G^{bn}$, d.h. alle verfügbaren Medienobjekte wurden ausgewählt. In diesem Fall muß die Hauptschleife des Basisalgorithmus M-mal durchlaufen werden. Bei jeder Iteration wiederum müssen im schlechtesten Fall (vollständig vernetzter Graph) für jedes Stichwort alle Medienobjekte bewertet werden. Dies führt (bei Annahme einer konstanten Komplexität der Bewertungsfunktion) zu einer *worst-case* Komplexität des Auswahlalgorithmus von $O(S * M^2)$.

Eine Abschätzung der durchschnittlichen Komplexität kann mit Hilfe des Ausgangsgrades (siehe Gleichung 6.19) des Abhängigkeitsgraphen vorgenommen werden:

Für die Anzahl der durchschnittlich ausgewählten Medienobjekte wird $|M(G^{bn})|/Ausgang(G^{bn})$ angenommen. Für jedes Stichwort müssen alle vermittelnden Medienobjekte - durchschnittlich $Ausgang(G^{bn})$ - bewertet werden. Hieraus ergibt sich eine Durchschnittskomplexität von

$$O\left(\frac{M}{Redundanz} * S * Ausgang\right) = O(M * S)$$

Da ferner $|b\_content(b)|$ einen guten Durchschnittswert für die bei jeder Iteration offenen Stichworte darstellt, reduziert sich die durchschnittliche Komplexität auf $O(M * b\_content(b))$. Eine vollständige Analyse des Laufzeitverhaltens des Auswahlalgorithmus unter Einbeziehung der Bewertungsfunktion ist Thema von Abschnitt 7.3.5.

7.3.3 Berücksichtigung der Kantengewichtungen

Bei der Darstellung des Basisalgorithmus wurde die **Gewichtung (Attributierung) der Kanten** des Abhängigkeitsgraphen ignoriert. Diese Gewichtung besteht aus allen Eigenschaften eines Indexeintrags, insbesondere dem Maß an verlangtem und vermitteltem Wissen.

Eine Vernachlässigung dieser Werte ist nur dann zulässig, wenn für jedes Stichwort sämtliche Gewichtungen, mit denen dieses Stichwort als Vorwissen verlangt wird, kleiner sind als alle Gewichtungen der eingehenden Kanten des Stichwortes im normalisierten Abhängigkeitsgraph:

$$\forall s \in S(G^{bn}): (i \in \sigma_{prv}(s,I(G^{bn})), j \in \sigma_{req}(s,I(G^{bn})) \Rightarrow req(j) \leq prv(i) + \varepsilon)$$

Da dies jedoch nicht vorausgesetzt werden kann, muß zu jedem offenen Stichwort zusätzlich vermerkt werden, wie groß das gesuchte vermittelte Wissen zu diesem Stichwort ist. Analog muß zu jedem vermittelten Stichwort bekannt sein, in welchem Maß Informationen über dieses Stichwort von den bereits ausgewählten Medienobjekten vermittelt werden.

Da hierbei auch der Sonderfall von vertieften Stichworten zu berücksichtigen ist, bietet es sich an, die Mengen der offenen und vermittelten Stichworte zusammenzufassen und die einzelnen Elemente als Tripel aus Stichwort, verlangtem Wissen und vermitteltem Wissen zu kodieren.

Ein Tripel (s, x, y) in dieser vereinheitlichten Stichwortmenge S^{M+} besagt somit, daß (falls $x > \varepsilon$) ein Medienobjekt gesucht wird, mit dem das Stichwort s mit einem Maß an vermitteltem Wissen von mindestens $x - \varepsilon$ erklärt werden kann. Darüber hinaus existiert innerhalb von M^{b+} ein Medienobjekt, das s mit einem Maß an vermitteltem Wissen von y erklärt (falls $y > x$).

Die im Basisalgorithmus verwendeten Mengen S^{req} und S^{prv} lassen sich aus S^{M+} rekonstruieren, so daß die Zusammenfassung der beiden Mengen keine Einschränkung der Funktionalität bedeutet:

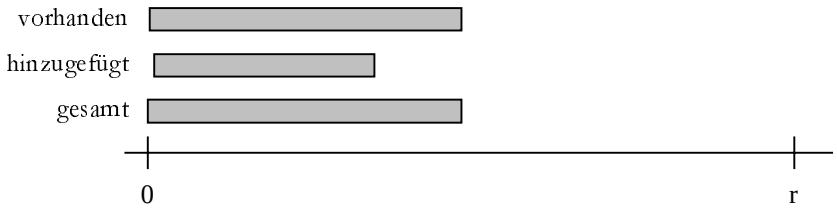
$$S^{req}(S^{M+}) := \bigcup_{(s,x,y)\in S^{M+}, x>\varepsilon} s \qquad (7.6a)$$

$$S^{prv}(S^{M+}) := \bigcup_{(s,x,y)\in S^{M+}, y>x} s \qquad (7.6b)$$

Beim Hinzufügen des von *SelectBest* ausgewählten Medienobjekts m zu M^{b+} muß für jeden Indexeintrag $i \in \mu_{ndx}(m,I(G^{bn}))$ zwischen den folgenden **fünf Fällen** unterschieden werden:

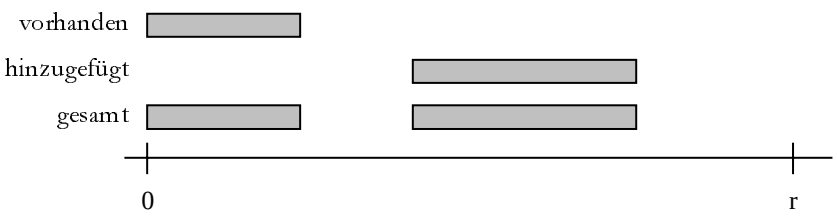
1. Es existiert noch kein Tripel (s, x, y) in S^{M+} mit s = kw(i). In diesem Fall wird das Tripel (kw(i), req(i), prv(i)) zu S^{M+} hinzugefügt.
2. Es existiert genau ein Tripel (s, x, y) in S^{M+} mit s = kw(i). Zusätzlich wird der durch Vorwissen und vermitteltes Wissen abgedeckte Bereich von i durch das vorhandene Tupel komplett abgedeckt, d.h. $x \leq req(i)$ und $y \geq prv(i)$. Der Indexeintrag i ist somit komplett in den Indexeinträgen bereits ausgewählter Medienobjekte enthalten und muß nicht zu S^{M+} hinzugefügt werden (Abbildung 7-7).

Abbildung 7-7: Komplette Überschneidung von Tripeln



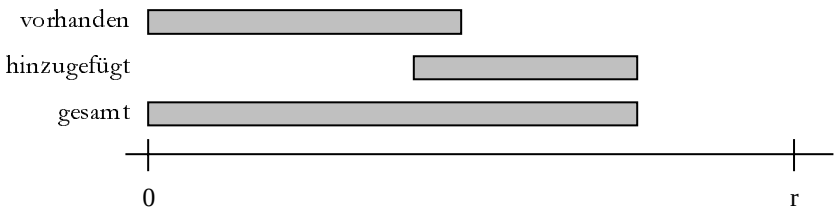
3. Es existiert genau ein Tripel (s, x, y) in S^{M+} mit s = kw(i). Die durch Vorwissen und vermitteltes Wissen aufgespannten Bereiche überschneiden sich nicht; es wird ein zweites Tripel (kw(i), req(i), prv(i)) für das gleiche Stichwort s zu S^{M+} hinzugefügt (Abbildung 7-8).

Abbildung 7-8: Keine Überschneidung der Indizes von vorhandenem und hinzugefügtem Stichwort



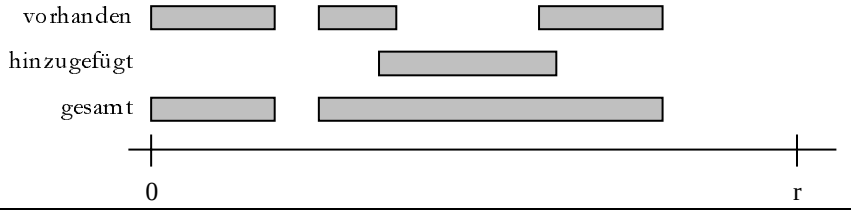
4. Es existiert genau ein Tripel (s, x, y) in S^{M+} mit s = kw(i). Der Bereich zwischen verlangtem und vermitteltem Wissen des hinzuzufügenden Medienobjektes überschneidet sich teilweise mit dem Bereich von x bis y. In diesem Fall wird das vorhandene Tripel so modifiziert, daß es den gesamten Bereich abbildet (Abbildung 7-9): $x \leftarrow \min\{ x, req(i) \}, y \leftarrow \max\{ y, prv(i) \}$

Abbildung 7-9: Überschneidung der Indizes von vorhandenem und hinzugefügtem Stichwort



5. Es existiert mehr als ein Tripel (s_i, x_i, y_i) mit s_i = kw(i). Falls sich verlangtes und vermitteltes Wissen des neuen Indexeintrages in einem der x_i, y_i Bereiche komplett enthalten ist, wird S^{M+} nicht verändert. Ansonsten werden alle x_i,y_i Bereiche, die sich mit dem Bereich von req(i) bis prv(i) überschneiden bzw. in ihm enthalten sind, zu einem neuen, den gesamten Bereich umfassenden Tripel zusammengefügt (Abbildung 7-10).

Abbildung 7-10: Hinzufügen eines Stichwortes zu mehreren, sich nicht überschneidenden Stichworten



Ein noch nicht behandeltes Problem ist, wie die Menge der offenen Tripel S^{M+} mit den als Zielknoten angegebenen Stichworten initialisiert wird. Da diese Stichworte initial offen sind, gilt für jedes ihrer in S^{M+} enthaltenen Tripel (s, x, y), daß x und y identisch sind. Der Wert von x und y wird dabei per Konvention auf 75% des maximal vermittelten Wissens zu dem jeweiligen Stichwort festgelegt. Die Sinn dieser Festlegung ist, daß idealerweise das Medienobjekt ausgewählt werden sollte, das am "meisten" Wissen zu einem verlangten Stichwort vermittelt. Es wird aber angenommen, daß ein in diesem Bewertungskriterium nur wenig schlechteres Medienobjekt durchaus die bessere Alternative sein kann, wenn es in anderen Kriterien sehr gut bewertet wird. Aus diesem Grund wird ein 25%-iger Toleranzbereich als akzeptabel angesehen.

Die Auswirkungen der Berücksichtigung von Kantengewichtungen auf den Basisalgorithmus beschränken sich im Wesentlichen auf die Einbindung der oben beschriebenen Fallunterscheidung.

7.3.4 Bewertung von Medienobjekten

Um die der Auswahl von Medienobjekten zugrundeliegenden Wahrscheinlichkeiten bestimmen zu können, wird eine Bewertung aller im aktuellen Auswahlschritt betrachteten Medienobjekte durchgeführt. Diese Bewertung stützt sich im wesentlichen auf Heuristiken, die sich in vier Kategorien untergliedern lassen:

1. **Lokale Bewertung**: Ein Aspekt bei der Bewertung eines Medienobjektes ist die Betrachtung seiner Eigenschaften unabhängig von den bisher ausgewählten Medienobjekten und den zu vermittelnden Stichworten. Standardmäßig entspricht die lokale Bewertung eines Medienobjekts dem Wert seines "Quality"-Attributs; der Autor kann jedoch auch eine eigene, anwendungsabhängige lokale Bewertung bereitstellen. Der Wert der lokalen Bewertung wird nur einmalig berechnet und anschließend als Attribut jedes Medienobjekts verwaltet.
2. **Stichwortabhängige Bewertung**: Wie in Abschnitt 7.3.2 erläutert, ist die Bewertung eines Medienobjektes immer an eines seiner vermittelten Stichworte gebunden. Je besser das Medienobjekt für die Vermittlung dieses Stichwortes geeignet ist, desto besser wird potentiell seine generelle Bewertung sein.

3. **Strukturelle Bewertung**: Ziel bei der automatisierten Erzeugung eines Lehr- bzw. Informationssystems ist es, am Ende jeden Block in der in seiner Beschreibung angegebenen Struktur generiert zu haben. Um dieses Ziel zu erreichen, müssen bei der Auswahl der Medienobjekte auch strukturelle Aspekte berücksichtigt werden. Konkret bedeutet dies, daß für jedes Medienobjekt untersucht werden muß, ob es die gewünschte Strukturierung eher begünstigt oder verhindert. Ebenfalls Teil der strukturellen Bewertung ist die Berücksichtigung bestimmter Muster in den Genres der ausgewählten Medienobjekte (siehe Kapitel 2.2.3).
4. **Anwendungsabhängige Bewertung**: Wie bei der lokalen Bewertung von Medienobjekten, kann der Autor eines Lehr- oder Informationssystems auch bei der anwendungsabhängiggen Bewertung eigene, anwendungsbezogene Kriterien einfließen lassen. Im Gegensatz zur lokalen Bewertung wird bei der anwendungsabhängigen Bewertung ein Medienobjekt jedoch nicht isoliert, sondern im Kontext der bereits ausgewählten Medienobjekte betrachtet. Bei der Erzeugung eines Informationssystems zum Thema "EURO" könnte hierdurch z.B. eine Gleichverteilung von Vor- und Nachteile der Währungsunion beschreibenden Medienobjekten hergestellt werden.

Ein Indexeintrag wird nacheinander bezüglich jedes dieser vier Kriterien bewertet. Anschließend werden die vier Einzelbewertungen gewichtet und zu einer Gesamtbewertung zusammengefaßt.

Die **Gewichtungen**, mit denen die einzelnen Bewertungen in die Gesamtbewertung einfließen, können vom Autor individuell festgelegt werden. Der entscheidende Faktor dabei ist, inwieweit die vom Autor erstellten Funktionen zur lokalen und anwendungsabhängigen Bewertung gegenüber der stichwort-bezogenen und strukturellen Bewertung gewichtet werden sollen. Bezüglich des Verhältnisses der beiden Bewertungen ist es sinnvoll, für Abhängigkeitsgraphen mit einem hohen Ausgangsgrad (Ausgang(G) > 2) die strukturelle Bewertung zu bevorzugen, während bei Graphen mit niedrigem Ausgangsgrad die stichwort-abhängige Bewertung höher gewichtet werden sollte.

Bewertung von Medienobjekten bezüglich eines Stichwortes

Die Bewertung eines Medienobjektes bezüglich eines Stichwortes geschieht anhand der vier Kriterien

- Grad des von dem Medienobjekt zu dem Stichwort vermittelten Wissens
- Relevanz des Stichworts für das Medienobjekt
- minimale Tiefe des Medienobjekts im Vergleich zur minimalen Tiefe des Stichworts
- Vorgängermengen aller das Stichwort vermittelnden Medienobjekte

Die beiden wichtigsten dieser Bewertungskriterien sind der Grad und die Relevanz, mit der das Stichwort von dem Medienobjekt vermittelt wird. Hierbei spielt auch eine Rolle, welche Position das Stichwortes im Vergleich zu allen anderen von dem Medienobjekt vermittelten Stichworten einnimmt. Falls das Stichwort bezüglich Grad und Relevanz der Vermittlung höher angesiedelt ist als alle anderen Stichworte, handelt es sich um den Hauptinhalt des Medienobjektes, was zu einer besseren Bewertung führt.

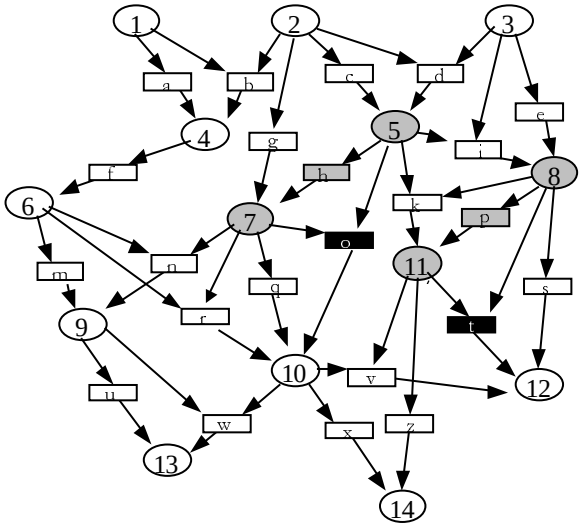
Ein Vergleich der minimalen Tiefen von Medienobjekt und Stichwort liefert zusätzliche Informationen zur Eignung des Medienobjekts für die Vermittlung dieses Stichworts:

> Falls die minimale Tiefe des Medienobjektes mit der des vermittelten Stichwortes identisch ist, so führt ein kürzester Pfad von einem Wurzelobjekt zu dem Stichwort über dieses Medienobjekt. Darüber hinaus sind minimale Tiefen eine gute heuristische Maßzahl für die Anzahl der Medienobjekte, die benötigt werden, damit ein Stichwort oder Medienobjekt erlernbar ist. Der Vergleich der minimalen Tiefen liefert somit einen Anhaltspunkt dafür, ob es sich bei dem zu bewertenden Medienobjekt um einen Knoten eines minimal vermittelnden Subgraphen des angegebenen Stichwortes handelt.

Einen weiteren Hinweis darüber, ob ein Medienobjekt zu einer einfachen und kompakten Vermittlung eines Stichworts beiträgt, liefert auch ein Vergleich der Vorgängermengen aller das Stichwort vermitttelnden Medienobjekte: Ist eines der vermittelnden Medienobjekte in der Vorgängermenge mindestens eines anderen vermittelnden Medienobjektes enthalten, kann dies bedeuten, daß dieses Objekt eine unnötig aufwendige Erklärung des Stichwortes darstellt.

In dem in Abbildung 7-11 dargestellten Abhängigkeitsgraph wird z.B. das Stichwort o von den Medienobjekten 5 und 7 vermittelt, wobei 5 in der Vorgängermenge von 7 enthalten ist. Dies bedeutet, daß potentiell ein von Medienobjekt 5 vermitteltes Stichwort benötigt wird, damit Medienobjekt 7 erlernbar ist (in diesem Fall Stichwort h). Medienobjekt 5 sollte daher besser bewertet werden als 7, da es gut möglich ist, daß es auch bei Auswahl von Medienobjekt 7 zu einem späteren Zeitpunkt zusätzlich ausgewählt wird. Gleiches gilt für die Vermittlung von Stichwort t durch die Medienobjekte 8 und 11.

Abbildung 7-11: Bewertung konkurrierender Objekte unter Berücksichtigung der Vorgängermengen



Eine Heuristik zur Berücksichtigung dieser Abhängigkeiten ist, daß ein Indexeintrag um so besser bewertet wird, je mehr alternative Medienobjekte das zum Indexeintrag zugehörige Objekt in ihrer Vorgängermenge enthalten (Inklusionshäufigkeit).

Die vier angesprochenen Kriterien – Grad der Vermittlung, relative Relevanz, minimale Tiefe und Inklusionshäufigkeit in Vorgängermengen – bilden zusammen die stichwort-abhängige Bewertung eines Medienobjekts.

Da sowohl die Bewertung der Relevanz als auch die Betrachtung der Vorgängermenge im Vergleich zu alternativen Medienobjekten geschieht, ergibt sich für die abhängige Bewertung ein konkretes Laufzeitverhalten von $O(\mu_{ndx}(mo(i)))$, und eine durchschnittliche Komplexität von $O(I / M)$.

Die Gewichtungen der einzelnen Aspekte der stichwort-bezogenen Bewertung sind abhängig vom Ausgangsgrad des Abhängigkeitsgraphen. Für Graphen hoher Redundanz (Ausgang(G) > 2) sollten vor allem die minimale Tiefen und Vorgängermengen berücksichtigenden Bewertungen stärker gewichtet werden, für weniger redundante Graphen der Grad des vermittelten Wissens und die Relevanz.

Strukturelle Bewertung

Ziel der strukturellen Bewertung ist es, die Eignung eines Medienobjektes für die gewünschte Struktur und Kompaktheit eines Blocks zu beschreiben. Die Abschätzung der zu erwartenden Kompaktheit stellt dabei kein Problem dar, da hierzu mit der minimalen Tiefe und der Größe der Vorgängermenge geeignete Heuristiken zur Verfügung stehen.

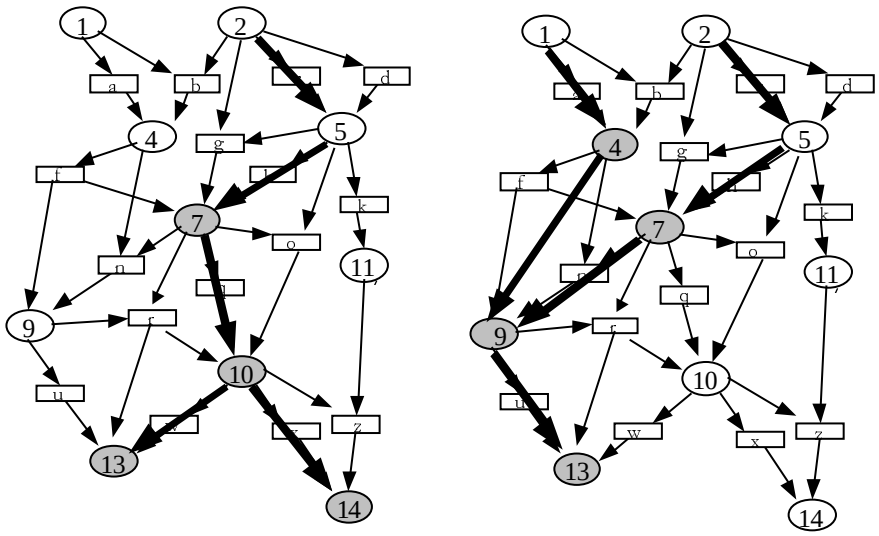
Anders hingegen sieht es mit der Bewertung bezüglich der gewünschten Struktur aus, insbesondere wenn der zu erzeugende Block eine hierarchische oder sequentielle Struktur besitzen soll.[34] Da die strukturelle Bewertung allein auf dem aktuellen Zustand von M^{b+} beruht und keinerlei Kenntnisse über die als nächstes ausgewählten Medienobjekte vorliegen, ist jede Aussage über die bei Auswahl eines bestimmten Medienobjekts zu erwartende Struktur sehr vage. Halbwegs gesicherte Aussagen zur Struktur von M^{b+} sind nur durch Analyse der M^{b+} schneidenden hierarchischen Subgraphen und durch Schnitte und Vereinigungen von Vorgängermengen möglich (siehe auch Kapitel 6.5.1).

[34] Falls als eine vernetzte Zielstruktur gewünscht ist, reduziert sich die strukturelle Bewertung auf das Kriterium „Kompaktheit“, d.h. es findet keine Analyse der Struktur von M^{b+} statt.

Je mehr hierarchische Subgraphen - und hier vor allem Wurzelgraphen – M^{b+} schneiden, desto näher ist die zu erwartende Medienobjektstruktur an einer hierarchischen Baumstruktur. Da jedoch die Anzahl der vorhandenen hierarchischen Subgraphen sehr stark von den Quellen der verfügbaren Medienobjekte abhängt, werden als zusätzliches Kriterium auch die Strukturen der den ausgewählten Subgraphen schneidenden hierarchischen Subgraphen berücksichtigt:

Hierarchische Subgraphen können Verzweigungen enthalten, bzw. zwei oder mehr hierarchische Subgraphen können gemeinsame Knoten besitzen. Verzweigungen (*branches*) werden positiv bewertet, da sie zu einer hierarchischen Medienobjektstruktur führen (Abbildung 7-12 links). Gemeinsame Knoten werden negativ bewertet, da sie potentiell zu Y-Strukturen (*merges*) führen, d.h. Strukturen, bei denen zwei oder mehr Sequenzen von Medienobjekten zusammengeführt werden (Abbildung 7-12 rechts).

Abbildung 7-12: Verzweigung in einem hierarchischen Subgraph (links) bzw. Zusammenführung zweier hierarchischer Subgraphen (rechts)



Der Vergleich der Vorgängermenge des zu bewertenden Medienobjekts mit den Vorgängermengen bereits ausgewählter Medienobjekte erlaubt ebenfalls Rückschlüsse auf die zu erwartende Struktur. Wenn M^{b+} hierarchisch strukturiert ist, ist das zu bewertende Objekt in den Vorgängermengen vieler (am besten aller) bereits ausgewählter Objekte enthalten. Wenn das zu bewertende Objekt hingegen viele bereits ausgewählte Medienobjekte in seiner Vorgängermenge enthält, deutet dies auf einen stark vernetzten Graphen hin.

Die einzige Ausnahme sind in der Vorgängermenge des zu bewertenden Objektes enthaltenen Medienobjekte, die auch in der Vorgängermenge von anderen ausgewählten Objekten enthalten sind, die ihrerseits Nachfolger des zu bewertenden Objektes sind. Diese Konstellation führt zu einer positiven Bewertung, da sie auf die Zusammenführung disjunkter Teilgraphen hinweist.

Das Laufzeitverhalten der strukturellen Bewertung wird im wesentlichen durch die diversen Vereinigungen und Schnitte von Vorgängermengen und ausgewählten Medienobjekten bestimmt und ergibt sich im schlechtesten Fall zu $O(M^{b+} * M(G^{bn}))$.

Die Gewichtung der einzelnen Teilbewertungen der strukturellen Bewertung ist vor allem davon abhängig, welcher der beiden zu berücksichtigenden Aspekte - hierarchische Seitenstruktur oder Kompaktheit - vom Autor als wichtiger angesehen wird. Generell sollte jedoch die Bewertung der hierarchischen Subgraphen mindestens 50% der strukturellen Bewertung ausmachen, da diese das verläßlichste strukturelle Bewertungskriterium darstellen.

7.3.5 Abschließende Betrachtung des Auswahlalgorithmus

Der in den vorangegangenen Abschnitten vorgestellte *Minesweeper*-Algorithmus zur Auswahl von Medienobjekten zeichnet sich durch die folgenden Eigenschaften aus:

- Es wird immer nur ein einzelnes Medienobjekt ausgewählt und zu M^{b+} hinzugefügt.

- Die Bewertung eines Medienobjektes geschieht im Vergleich zu konkurrierenden Medienobjekten.
- Bei der Bewertung werden verschiedene inhaltliche, strukturelle und anwendungsspezifische Aspekte getrennt betrachtet.

Insbesondere die beiden letzten Eigenschaften ergeben sich aus den Besonderheiten des zugrundeliegenden Anwendungsgebiets; der Erzeugung von didaktisch und strukturell „sinnvollen" Dokumenten.

In diesem Abschnitt soll versucht werden, den Auswahlalgorithmus zu bewerten, d.h. seine Stärken und Schwächen noch einmal herauszustellen, sowie Möglichkeiten zu seiner Verbesserung aufzuzeigen.

Der A* Algorithmus

Für eine Bewertung des Algorithmus liegt ein Vergleich mit einem ähnlich motivierten Graphen-Suchalgorithmus nahe. Hierbei bietet sich der **A* Algorithmus** als in der künstlichen Intelligenz und vielen anderen Gebieten weit verbreitetes Verfahren zur Suche von Pfaden in Graphen an.

Der A* Algorithmus [Pearl84] ist auf gerichteten Graphen definiert und kann sowohl zur Tiefen-, Breiten- und Bestensuche angewandt werden. Bei der im Vergleich zum *Minesweeper*-Algorithmus interessanten Bestensuche wird die Bewertung eines Knotens n durch eine Bewertungsfunktion f(n) = g(n) + h(n) vorgenommen. Die Gesamtbewertung f(n) eines Knotens ergibt sich dabei aus der Bewertung des Pfades von einem Anfangszustand zum betrachteten Knoten g(n) plus einer Abschätzung der Bewertung des Pfades vom betrachteten Knoten zu einem Endzustand h(n). h(n) ist beim A* Algorithmus immer optimistisch, d.h. die geschätzte Bewertung ist besser als die "wirkliche" Bewertung.

Wie beim Minesweeper-Algorithmus wird auch beim A* Algorithmus nur der am besten bewertete Knoten ausgewählt. Zusätzlich wird jedoch für jeden ausgewählten Knoten der komplette Pfad vom Ausgangszustand zu diesem Knoten vermerkt. Hierdurch ist ein *Backtracking*, d.h. ein Zurückspringen auf einen früheren Zustand des Suchbaumes möglich.

Um den A* Algorithmus an die Suche nach Subgraphen im Abhängigkeitsgraphen anzupassen, müssen einige Änderungen am Suchbaum und an der Bewertungsfunktion vorgenommen werden:

- Startknoten sind die Menge b_content(b) der zu vermittelnden Stichworte eines Blocks b.
- Der Zielzustand besteht nicht aus einer Menge von zu erreichenden Zielknoten, sondern aus einer minimal abgeschlossenen Menge von ausgewählten Medienobjekten.
- g(n) ist eine Bewertung der bereits ausgewählten Medienobjekte M^{b+} und h(n) eine Abschätzung der Anzahl noch auszuwählender Medienobjekte (z.B. basierend auf den minimalen Tiefen der in M^{b+} nicht erlernbaren Medienobjekte).

Die Unterschiede zwischen A* und *Minesweeper* Algorithmus liegen vor allem in der Wahl der Bewertungsfunktion und in der Verwaltung der ausgewählten Knoten. Beide Aspekte sollen im folgenden etwas genauer betrachtet werden.

Eine der großen Stärken des A* Algorithmus ist die Möglichkeit des *Backtracking*, d.h. in bestimmten Situationen können bereits gemachte Auswahlen revidiert und dadurch frühere Zustände des Suchbaumes neu bewertet werden. Diese Funktionalität ist vor allem dann wichtig, wenn bestimmte Auswahlen zwangsläufig zu Ergebnissen führen, die der Zielvorgabe nicht genügen[35].

Innerhalb des Abhängigkeitsgraphen sind zu falschen Ergebnissen führende Auswahlen nicht möglich, da durch die Abgeschlossenheit von G^{bn} sichergestellt ist, daß jeder Knoten und jede Menge von Knoten innerhalb von G^{bn} erlernbar ist[36]. Dies bedeutet, daß die Zielvorgabe eines abgeschlossenen Subgraphen für jede zwischenzeitlich aufgebaute Menge ausgewählter Medienobjekte M^{b+} erfüllbar ist.

---

[35] Viele der existierenden Varianten des A* Algorithmus – z.B. *Iterative Deepening A** (IDA*) – zielen auf eine Optimierung des *Backtrackings* durch Reduzierung des für die Verwaltung von Zwischenzuständen benötigten Speichers ab. Dies macht den A* Algorithmus vor allem für nicht-deterministischen Fragestellungen interessant.

[36] Was jedoch nicht bedeutet, daß keine schlechten Auswahlen möglich sind, die zu schlechten Ergebnissen bezüglich Inhalt und Struktur des erzeugten Dokuments führen.

Auch im Zusammenhang mit der Bewertung von Medienobjekten kann *Backtracking* bei der Suche nach einem möglichst „guten“ G^{b+} kaum sinnvoll eingesetzt werden. Eine aussagekräftige Bewertung ist nur für den Endzustand, nicht aber für die Zwischenzustände von M^{b+} möglich. Und auch hier ist normalerweise nicht feststellbar, welche Auswahl zu einem „schlechten“ G^{b+} geführt hat, und ob überhaupt ein „besserer“ Subgraph existiert, der die Vorgaben erfüllt. Ein Zurückspringen auf einen Zwischenzustand ist von daher sinnlos, denn es gibt kaum neue Informationen, die zu einer anderen Bewertung der auswählbaren Medienobjekte führen könnten.

Das andere Unterscheidungskriterium zwischen A* und *Minesweeper* Algorithmus ist die Art der heuristischen Bewertungsfunktion. Der A* Algorithmus verwendet eine kostenorientierte Bewertung einzelner Knoten, während der *Minesweeper* Algorithmus die in Frage kommenden Knoten sowohl inhaltlich als auch strukturell in Relation zu anderen, einem bestimmten Kriterium genügenden, Knoten bewertet.

Der Vorteil der relativen Bewertung ist vor allem darin zu sehen, daß dadurch den an Struktur und Inhalt des gesuchten Subgraphen gestellten Anforderungen erheblich besser Rechnung getragen werden kann. Eine Bewertung struktureller Eigenschaften eines Medienobjektes ist nur dann aussagekräftig, wenn bereits möglichst viele Medienobjekte ausgewählt wurden, so daß sich überhaupt eine Struktur abzeichnet. Hier liegen die Stärken der *Minesweeper*-Strategie: Der Abhängigkeitsgraph enthält in der Regel viele Medienobjekte, die zur Vermittlung eines Stichworts keine oder zumindest keine guten Alternativen besitzen. Durch die relative Bewertung werden diese Knoten zuerst ausgewählt, so daß in dem Moment, wenn das Kriterium „Struktur“ bei der Bewertung überhaupt eine Rolle zu spielen beginnt, bereits eine Struktur erkennbar ist.

Darüber hinaus ist es über die Bewertungsfunktion des A* Algorithmus nur schwer möglich, strukturelle Aspekte zu erfassen. Insbesondere eine Abschätzung der Struktur eines das betrachtete Medienobjekt vermittelnden Subgraphen ist nicht nur schwer möglich, sondern ohne Berücksichtigung der bereits ausgewählten Knoten auch wenig aussagekräftig.

Es ist somit vor allem die relative Bewertung der Medienobjekte, die den Minesweeper Algorithmus für die Generierung von Lehr- und Informationssystemen interessant macht. Ein A* Algorithmus wäre sicherlich in der Lage, in optimaler Zeit den kleinsten alle Lernziele vermittelnden Subgraphen zu finden. Um jedoch den bezüglich Inhalt und Struktur besten Subgraphen zu finden, wird ein etwas aufwendigeres Verfahren wie z.B. der Minesweeper Algorithmus benötigt.

Laufzeitverhalten

Leider schlägt sich die erhöhte Komplexität der relativen Bewertung negativ im Laufzeitverhalten des Algorithmus nieder. Die einzelnen Teilalgorithmen haben einen linearen bis quadratischen durchschnittlichen Aufwand:

Algorithmus	**durchschnittlicher Aufwand**
„Select“:	$O(M * \text{Eingang}(M) * o_1)$
„SelectBest“	$o_1 = O(S * \text{Ausgang}(M) * o_2)$
„Bewertung“	$o_2 = O(o_3 + o_4 + o_{\text{lokaleBewertung}} + o_{\text{Anwendungsbewertung}})$
„AbhängigeBewertung“	$o_3 = O(I/M)$
„StrukturBewertung“	$o_4 = O(M^2)$

Geht man davon aus, daß sowohl die lokale als auch die anwendungsabhängige Bewertung von konstantem Aufwand sind, und daß ferner gilt:

$$\text{Eingang}(M) \approx \text{Ausgang}(M) \approx I/S \approx I/M,$$

so ergibt sich ein Gesamtaufwand von:

$$O(M * (I/M) * S * (I/S) * (c + (I/M) + M^2)) \qquad (7.7)$$
$$= O(I^3/M + I^2 * M^2)$$

$\approx O(I^2 * M^2)$ (Annahme: $|I| < |M|^2$)

Dieser hohe Gesamtaufwand wird entscheidend von den für die Berücksichtigung struktureller Aspekte entstehenden Kosten geprägt. Ohne die Einhaltung einer vorgegebenen Zielstruktur reduziert sich der Aufwand auf O(I*M).

Optimierungen

Der Aufwand des *Minesweeper*-Algorithmus kann durch weitere Heuristiken und zusätzliche Maßzahlen in vielen Fällen erheblich reduziert werden. Hierzu vier Beispiele:

- Falls keine Medienobjekte ohne Alternative existieren, bewertet der Algorithmus „Select" die vermittelnden Medienobjekte aller offenen Stichworte. Um dieses Verfahren abzukürzen, bietet es sich an, eine Grenzwahrscheinlichkeit festzulegen, ab der ein Medienobjekt ohne Auswertung weiterer Stichworte ausgewählt wird. Wird z.B. bei der Bewertung der vermittelnden Medienobjekte des ersten Stichwortes ein Medienobjekt gefunden, das mit 80%-iger Wahrscheinlichkeit in G^{b*} enthalten ist, so wird dieses Objekt ausgewählt, ohne weitere Stichworte zu betrachten.
- Falls der Umfang der Vorgängermenge eines offenen Stichwortes s identisch mit seiner minimalen Tiefe ist (d.h. $|\sigma_{ps}(s,G)| = \sigma_{depth}(s,G)$), können alle in der Vorgängermenge enthaltenen Medienobjekte ohne Bewertung ausgewählt werden, da sie den einzigen Pfad zu dem Stichwort bilden.
- Die Bewertung eines Medienobjektes ändert sich nur dann, wenn dieses Objekt in der Vorgängermenge der Nachfolger des zuletzt ausgewählten Medienobjekts enthalten ist. Durch Berechnung einer entsprechenden Maßzahl (quadratischer Aufwand) kann sehr einfach festgestellt werden, wann ein Medienobjekt neu bewertet werden muß.
- Wenn die strukturelle Bewertung sehr stark gewichtet ist (> 30%), werden fast ausschließlich Medienobjekte ausgewählt, die Bestandteil eines hierarchischen Subgraphen sind. In diesem Fall bietet es sich an, hierarchische Subgraphen als Ganzes zu bewerten und auszuwählen.

Neben diesen den Aufwand betreffenden Optimierungen sind auch Erweiterungen des Minesweeper Algorithmus denkbar, die die Qualität der Struktur des berechneten Subgraphen verbessern. Hierzu zählt vor allem eine eingehendere Analyse der Vorgängermengen der offenen Stichworte, um Verzweigungen des Medienobjektgraphen vorherzuberechnen und disjunkte Teilgraphen zu entdecken. All diese Optimierungen sind jedoch mit zusätzlichem Rechenaufwand verbunden.

7.4 Strukturierung von Medienobjekten

Durch das in den Abschnitten 7.1 bis 7.3 beschriebene Auswahlverfahren wurde ein Subgraph G^{b+} eines an eine Blockbeschreibung b angepaßten, normalisierten Abhängigkeitsgraphen ermittelt, in dem alle gewünschten Stichworte erlernbar sind. G^{b+} bildet gleichzeitig die Grundlage der Medienobjektstruktur des Blockes b.

Die Knoten der Medienobjektstruktur T^{b*} werden als Medienobjektknoten bezeichnet und bestehen im wesentlichen aus den Knoten des Medienobjektgraphen von G^{b+}. Alle Medienobjektknoten eines Blocks sind über Hyperlinks und Vorgänger-Nachfolger Beziehungen miteinander verbunden:

Definition

Jede aus einem Abhängigkeitsgraphen abgeleitete **Medienobjektstruktur** T ist ein gerichteter Graph, der als Tripel aus Medienobjektknoten, Verbindungen zwischen Medienobjektknoten und Hyperlinks definiert ist. Die Menge aller Medienobjektstrukturen wird als $T^{\#}$ bezeichnet:

$$T^{\#} \subseteq \mathbf{P}(TM^{\#}) \times \mathbf{P}(TM^{\#} \times TM^{\#}) \times \mathbf{P}(H^{\#}) \quad (7.8)$$

Auf die einzelnen Bestandteile einer Medienobjektstruktur kann über die folgenden Projektionen zugegriffen werden:

t_nodes((nodes, edges, links)) = nodes

t_edges((nodes, edges, links)) = edges

t_links((nodes, edges, links)) = links

Definition

Jeder **Medienobjektknoten** setzt sich aus einem Medienobjekt, einer Menge von Zusatzobjekten[37] und einer Markierung (siehe Kapitel 7.5) zusammen:.

$$TM^{\#} \subseteq M^{avail} \times \mathbf{P}(AO) \times \{true, false\} \quad (7.9)$$

Der Zugriff auf die Bestandteile eines Medienobjektknotens erfolgt über die Projektionen

tm_mo((mo, addon, flag)) = mo,

tm_addon((mo, addon, flag)) = addon (siehe Kapitel 7.5),

tm_flag((mo, addon, flag)) = flag.

Die **Vorgänger** und **Nachfolger** eines jeden Medienobjektknotens t innerhalb einer Medienobjektstruktur T lassen sich direkt aus den Kanten der Medienobjektstruktur ableiten:

$$\tau_{pre}(t,T) := \{ x \mid \exists(x,t) \in t\_edges(T) \} \quad (7.10a)$$

$$\tau_{succ}(t,T) := \{ y \mid \exists(t,y) \in t\_edges(T) \} \quad (7.10b)$$

Hyperlinks der Medienobjektstruktur bestehen aus den beiden zugrundeliegenden Indexeinträgen des Abhängigkeitsgraphen, wobei der erste Indexeintrag den Ausgangsknoten mit einem verlangten Stichwort und der zweite Indexeintrag dieses Stichwort mit dem Zielknoten verbindet:

$$H^{\#} \subseteq \mathbf{P}(I^{avail} \times I^{avail}) \quad (7.11)$$

Hyperlinks bilden eine Obermenge aller Vorgänger-Nachfolger Beziehungen der Medienobjektstruktur. Die Kanten der Medienobjektstruktur sind lediglich explizit hervorgehobene Hyperlinks zur Strukturierung der Knoten. Initial bilden Hyperlinks und Vorgänger-Nachfolger Beziehungen die selben Verbindungen zwischen Medienobjekten ab. Erst durch das Entfernen von redundanten Kanten (Kapitel 7.4.3) und die Erstellung der Seitenstruktur (Kapitel 7.6) wird der semantische Unterschied zwischen beiden Konstrukten (Strukturierung vs. Referenzierung) deutlich.

Die Berechnung der Medienobjektstruktur eines Blockes b aus seinem zuvor ermittelten Abhängigkeitsgraphen G^{b+} geschieht über den Algorithmus „**Sequence**“. „Sequence“ besteht aus vier Teilalgorithmen, wobei in Schritt zwei der Übergang vom Abhängigkeitsgraphen zur Medienobjektstruktur vollzogen wird:

1. Entfernen redundanter Knoten aus G^{b+}, um zu einem Subgraphen zu gelangen, in dem alle gewünschten Stichworte minimal erlernbar sind.
2. Abbildung eines bipartiten Subgraphen des Abhängigkeitsgraphen auf einen Medienobjekt-Graphen.
3. Entfernen redundanter Kanten.
4. Hinzufügen von Zusatzobjekten, die der Blockbeschreibung entsprechen und zu den ausgewählten Medienobjekten passen.

[37] Für die formale Definition von an Medienobjektknoten gebundene Zusatzobjekte siehe Kapitel 7.5.3.

Algorithmus „Sequence“	
Aufgabe:	Berechnung von Vorgänger-Nachfolger Beziehungen und Hyperlinks innerhalb des aus den ausgewählten Medienobjekten erstellten Subgraphen des Abhängigkeitsgraphen
Eingabe:	abgeschlossener Subgraph G^{b+} des Abhängigkeitsgraphen
	Insgesamt verfügbare Medienobjekte M^b
	Blockbeschreibung b
Ausgabe:	Medienobjektstruktur T^{b*} des Blocks b aus dem Abhängigkeitsgraphen G

Entfernen redundanter Knoten

Erstellen der Medienobjektstruktur

Entfernen redundanter Kanten

Hinzufügen zusätzlicher Objekte

7.4.1 Entfernung redundanter Knoten

Der mit dem Algorithmus „Select“ berechnete Subgraph G^{b+} ist nicht zwingend minimal und abgeschlossen. Die **Redundanz** in G^{b+} kommt dabei vor allem dadurch zustande, daß

1. ein Stichwort während des Auswahlalgorithmus mehrmals mit jeweils höherem Grad „offen“ ist oder
2. ein schlechter bewertetes, alternatives Medienobjekt zu einem ausgewählten Medienobjekt in einer späteren Iteration des Algorithmus zur Vermittlung eines anderen Medienobjektes doch noch ausgewählt wird.

Die redundanten Medienobjekte in G^{b+} zeichnen sich dadurch aus, daß entweder keines ihrer vermittelten Stichworte mit ausreichendem Grad vermittelt wird (**echt redundante Medienobjekte**) oder aber alle ihre vermittelten Stichworte auch von anderen Medienobjekten in G^{b+} vermittelt werden (**abhängig redundante Medienobjekte**).

Abbildung 7-13: Beispiele für echt redundante (links) und abhängig redundante (rechts) Medienobjekte

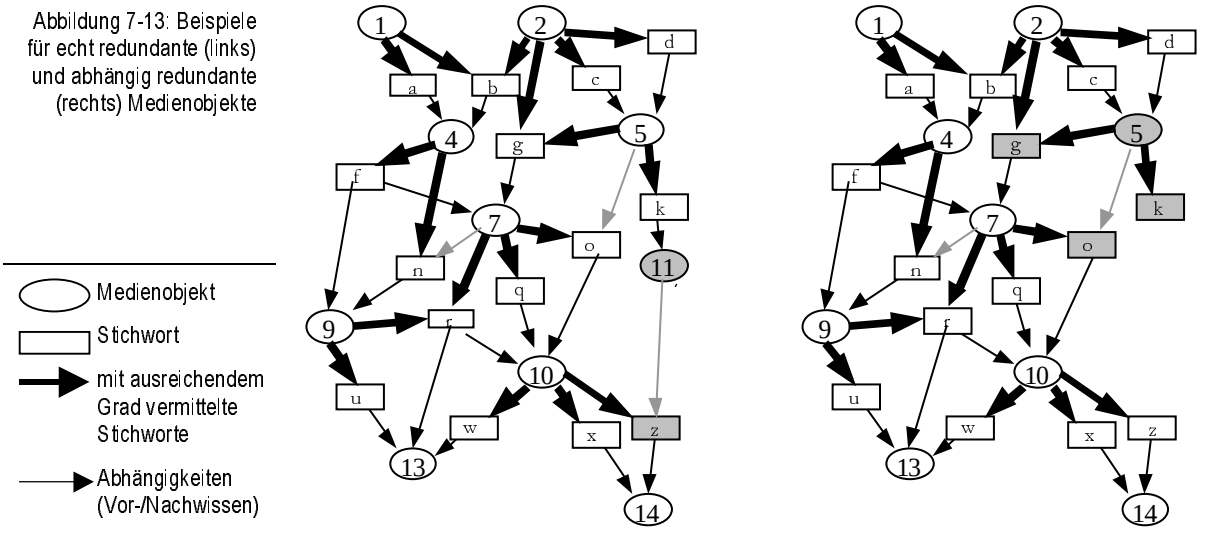


Abbildung 7-13 zeigt jeweils ein Beispiel für ein echt redundantes (links) und ein abhängig redundantes (rechts) Medienobjekt. Die hervorgehobenen Kanten repräsentieren von Medienobjekten mit ausreichendem Grad vermittelte Stichworte, d.h. Medienobjekt 5 vermittelt Stichwort k mit dem von Medienobjekt 11 verlangten Grad und Stichwort o mit einem geringeren als von Medienobjekt 10 verlangten Grad.

In Abbildung 7-13 ist Medienobjekt 11 echt redundant, da es keines der Stichworte des Abhängigkeitsgraphen mit ausreichendem Grad vermittelt. Medienobjekt 11 muß daher auf jeden Fall aus dem Abhängigkeitsgraphen entfernt werden, um zu einem minimal abgeschlossenen Graphen zu gelangen. Hierdurch entsteht der in Abbildung 7-13 (rechts) dargestellte Abhängigkeitsgraph.

Unter der Voraussetzung, daß Stichwort k nicht in b_content(b) enthalten war, ist in diesem Abhängigkeitsgraphen Medienobjekt 5 abhängig redundant. Alle von Medienobjekt 5 mit ausreichendem Grad vermittelten Stichworte innerhalb des Abhängigkeitsgraphen (in diesem Fall nur Stichwort g), werden auch von mindestens einem anderen Medienobjekt mit ausreichendem Grad vermittelt. Durch Entfernen von Medienobjekt 5 wird daher die Erreichbarkeit der Knoten des Abhängigkeitsgraphen nicht beeinträchtigt.

Um festzustellen, ob ein Medienobjekt echt oder abhängig redundant ist, muß zuerst zu jedem Stichwort der **maximal benötigte Grad der Vermittlung** berechnet werden:

$$sreqmax(s, G^{b+}) := \max\{ \bigcup_{i \in \sigma_{req}(s, I(G^{b+}))} req(i) \} \qquad (7.12)$$

Anschließend kann zu jedem Stichwort die Menge der ausgewählten Medienobjekte ermittelt werden, die dieses Stichwort mit ausreichendem Grad vermitteln:

$$sprv(s, G^{b+}) := \{ mo(i) \mid i \in \sigma_{prv}(s, I(G^{b+})), prv(i) \geq sreqmax(s, G^{b+}) - \varepsilon \} \qquad (7.13)$$

Mit Hilfe dieser Informationen können redundante Medienobjekte relativ einfach identifiziert und entfernt werden. Hierbei sind jedoch drei Sonderfälle zu beachten:

- Blätter des Subgraphen sind für die Identifizierung redundanter Knoten nur relevant, wenn es sich bei ihnen um in b_content(b) enthaltene Stichworte handelt.
- Abhängig redundante Medienobjekte können sich überschneiden, d.h. anhand der berechneten Eigenschaften erscheinen zwei Medienobjekte m und o redundant, obwohl nur eines von beiden aus dem Graphen entfernt werden kann, ohne die Abgeschlossenheit zu zerstören.
- Durch Entfernen eines redundanten Medienobjekts aus G^{b+} können weitere Medienobjekte redundant werden, bzw. abhängig redundante Medienobjekte nun alleinige Vermittler eines benötigten Stichwortes sein.

Um diesen Eigenschaften redundanter Medienobjekte gerecht zu werden, können abhängig redundante Medienobjekte nur einzeln aus G^{b+} entfernt werden. Falls mehrere abhängig redundante Medienobjekte in G^{b+} existieren, wird mittels einer Bewertungsfunktion das "schlechteste" dieser Objekte ermittelt und als erstes aus dem Abhängigkeitsgraphen entfernt. Zusätzlich wird anschließend eine Neuberechnung aller redundanten Medienobjekte durchgeführt.

Der für die Bewertung der abhängig redundanten Medienobjekte benötigte Algorithmus ist eine spezielle Adaption des in Kapitel 7.3 beschriebenen Algorithmus zur stichwortbezogenen Bewertung von Medienobjekten, wobei jedoch strukturelle Aspekte stärker gewichtet werden.

Bedingt durch die aufwendige strukturelle Bewertung der abhängig redundanten Medienobjekte ergibt sich für die Entfernung redundanter Knoten im schlechtesten Fall ein Aufwand von $O(I * M^2 + M^3)$. Da normalerweise jedoch nur wenig redundante Medienobjekte existieren, kann als durchschnittlicher Aufwand $O(I^2)$ angenommen werden. Falls die benötigten Maßzahlen bereits während des Auswahlalgorithmus berechnet werden, reduziert sich der Aufwand auf $O(M^2)$.

7.4.2 Abbildung von G^{b*} auf eine Medienobjekt-Struktur T^{b+}

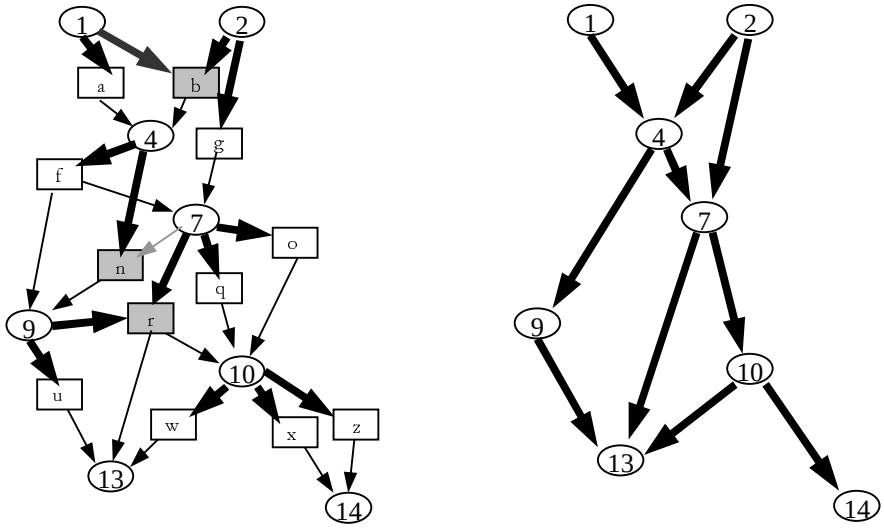
Durch die Entfernung aller redundanten Medienobjekte aus G^{b+} ist ein Abhängigkeitsgraph G^{b*} entstanden, in dem alle in der Blockbeschreibung enthaltenen Medienobjekte minimal erlernbar sind. G^{b*} bildet damit die Grundlage für die Medienobjektstruktur des Blockes b.

Um den minimal abgeschlossenen Abhängigkeitsgraphen G^{b*} in eine Medienobjektstruktur zu überführen, muß jedes in G^{b*} enthaltene Medienobjekt auf einen Medienobjektknoten abgebildet werden.

Die Kanten der Medienobjektstruktur bilden eine Untermenge der Kanten des Medienobjekt-Graph des Abhängigkeitsgraphen, da für jedes Stichwort nur eine Eingangskante in die Medienobjektstruktur übernommen wird. Existieren mehrere Medienobjekte, die ein Stichwort mit ausreichendem Grad vermitteln, wird mit Hilfe einer (heuristischen) Bewertungsfunktion das am geeignetsten erscheinende ausgewählt. Die Kriterien hierbei sind wiederum inhaltliche, stichwort-abhängige und strukturelle Aspekte.

Die nachfolgende Abbildung zeigt auf der linken Seite den von allen redundanten Medienobjekten befreiten Abhängigkeitsgraph aus Abbildung 7-13.

Abbildung 7-14: Abhängigkeitsgraph aus Abbildung 7-13 nach Entfernung aller redundanten Medienobjekte



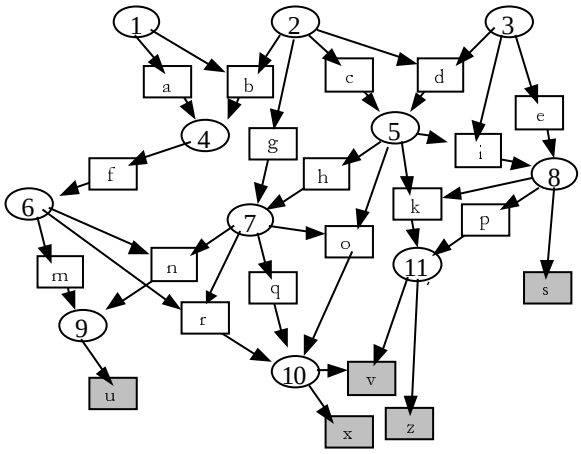
In dem dargestellten Abhängigkeitsgraph werden die Stichworte b, n und r von jeweils zwei Medienobjekten vermittelt. Hiervon muß jeweils eins für die Vermittlung des Stichworts ausgewählt werden. Bei Stichwort n fällt diese Wahl leicht, da dieses Stichwort nur von Medienobjekt 4 mit ausreichendem Grad vermittelt wird. Die Stichworte b und r werden hingegen von beiden vermittelnden Medienobjekten mit ausreichendem Grad vermittelt. Hier muß die Auswahl somit über eine Bewertung der konkurrierenden Objekte erfolgen.

Abbildung 7-14 (rechts) zeigt eine mögliche Medienobjektstruktur für den Beispielgraphen.

Der Algorithmus mit dem ein Abhängigkeitsgraphe G^{b*} auf eine Medienobjektstruktur abgebildet wird, erstellt zuerst aus jedem in G^{b*} enthaltenen Medienobjekt einen Medienobjektknoten. Anschließend werden die Vorgänger-Nachfolger Beziehungen des Abhängigkeitsgraphen als Kanten und Hyperlinks der Medienobjektstruktur repräsentiert.

Die nachfolgenden Abbildungen verdeutlichen noch einmal den Übergang vom Abhängigkeitsgraphen zur Medienobjektstruktur anhand eines weiteren Beispielgraphen.

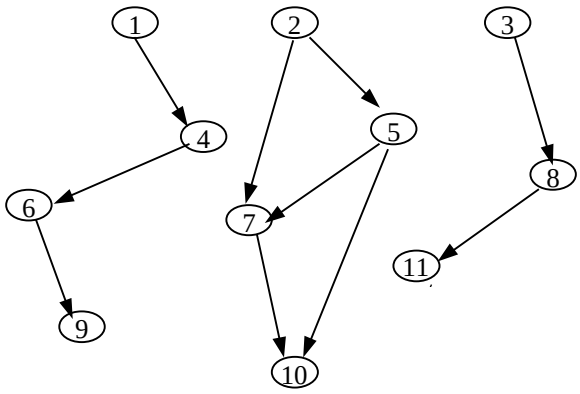
Abbildung 7-15: Abhängigkeitsgraph G^{b*}



Es sei angenommen, daß der Graph in Abbildung 7-15 das Resultat einer Medienobjektauswahl zur Vermittlung der Stichworte s, u, v, x und z ist. Redundante Knoten wurden bereits entfernt.

Der Strukturierungsalgorithmus kapselt zunächst jedes Medienobjekt des Abhängigkeitsgraphen in einem Medienobjektknoten. Anschließend werden die Verknüpfungen zwischen den Medienobjektknoten berechnet. Dies geschieht, indem jeder Indexeintrag von einem Stichwort zu einem Medienobjekt genau einem umgekehrt gerichteten Indexeintrag zugeordnet wird (z.B. die Kante von Medienobjekt 1 zu Stichwort a der Kante von Stichwort a zu Medienobjekt 4) . Die so ermittelten Paare von Indexeinträgen bilden die Hyperlinks und implizit auch die Kanten der initialen Medienobjektstruktur. Abbildung 7-16 zeigt die Medienobjektstruktur des Abhängigkeitsgraphen aus Abbildung 7-15, wobei alle Indexeinträge durch Vorgänger-Nachfolger Beziehungen innerhalb der Medienobjektstruktur ersetzt wurden.

Abbildung 7-16: Medienobjektstruktur zu Abbildung 7-15



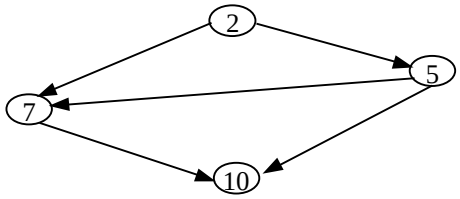
Die Abbildung auf eine Medienobjektstruktur ist die letzte Transformation des Abhängigkeitsgraphen, bei der inhaltliche Aspekte von Medienobjekten im Vordergrund stehen. Alle weiteren Algorithmen zur Optimierung der Medienobjektstruktur und zur Erstellung der Seiten- und Kapitelstruktur - mit Ausnahme der in Kapitel 7.5 beschriebenen Einbindung von Zusatzobjekten - zielen auf eine Verbesserung der Struktur ab.

7.4.3 Entfernung von Abkürzungen

Die berechnete Medienobjektstruktur basiert auf zwei Arten von Verbindungen zwischen Medienobjekten: Hyperlinks und Vorgänger-Nachfolger Beziehungen. Initial sind beide Arten von Verbindungen identisch, da alle Hyperlinks auch Vorgänger-Nachfolger Beziehungen repräsentieren (siehe Abbildung 7-16).

Die Trennung von Hyperlinks und Kanten ergibt sich durch redundante Kanten der Medienobjektstruktur. Als Beispiel soll hierzu der in Abbildung 7-17 dargestellte „mittlere“ Graph aus Abbildung 7-16 dienen.

Abbildung 7-17: Medienobjekt-Graph

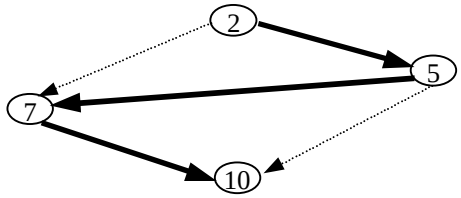


Redundante Kanten ergeben sich durch Transitivität aus der Semantik von Vorgänger-Nachfolger Beziehungen innerhalb der Medienobjektstruktur.

Betrachtet man Abbildung 7-17, wird deutlich, daß die Kante von Medienobjekt 2 zu Medienobjekt 7 redundant ist: Um Medienobjekt 7 zu verstehen, muß Objekt 5 bekannt sein, was wiederum erfordert, daß Medienobjekt 2 bekannt ist. Hierdurch ist Medienobjekt 2 auch implizit zum Verständnis von Medienobjekt 7 erforderlich.

Explizite Kanten, deren Semantik bereits durch Pfade – sog. **Umwege** – abgedeckt ist, werden als **Abkürzungen** bezeichnet. Der Graph in Abbildung 7-18 enthält neben der Abkürzung 2 $\dashrightarrow^{G}$ 7 eine weitere Abkürzung von Medienobjekt 5 zu Medienobjekt 10. Abkürzungen sind generell redundant, weswegen sie in der Medienobjektstruktur zwar als Hyperlinks, nicht aber als Vorgänger-Nachfolger Beziehungen berücksichtigt werden.

Abbildung 7-18: Vorgänger-Nachfolger Beziehungen (fett) und einfache Hyperlinks (dünn)



Für den Nutzer des generierten Dokuments ist die Entfernung von Kanten aus der Medienobjektstruktur insoweit sichtbar, als daß die betroffenen Seiten in der Seitenstruktur nicht durch explizite "nächste Seite" bzw. "vorherige Seite" Verknüpfungen verbunden sind, sondern lediglich durch von Stichworten ausgehende Referenzen (Anker).

Abkürzungen kommen in der Praxis der *Bottom-Up* Erstellung von Lehr- und Informationssystemen sehr häufig vor. Sie entstehen durch die gehäufte Verwendung von für das behandelte Themengebiet grundlegenden Stichworten. In einem Lehrsystem zum Thema „Statistik“ werden so z.B. Begriffe wie „Merkmal“, „Variable“ oder „Skalierung“ von sehr vielen Medienobjekten als Vorwissen benötigt, wobei die meisten dieser Referenzierungen Abkürzungen im oben beschriebenen Sinne darstellen.

Abkürzungen innerhalb der Medienobjektstruktur können mit Hilfe von **Vorgängermengen** erkannt werden. Die Vorgängermenge $\tau_{ps}(t,T)$ eines Medienobjektknotens t innerhalb einer Medienobjektstruktur T besteht aus allen direkten und indirekten Vorgängern von t:

$$\tau_{ps}: TM^{\#} \times \mathbf{P}(T^{\#}) \rightarrow \mathbf{P}(TM^{\#}) \qquad (7.14)$$

$$\tau_{ps}(t,T) := \tau_{pre}(t,T) \cup \bigcup_{u \in \tau_{pre}(t,T)} \tau_{ps}(u,T)$$

Die Berechnung der Vorgängermengen aller Medienobjektknoten erfolgt analog zur in Kapitel 5.5 beschriebenen Berechnung der Vorgängermengen von Medienobjekten.

Satz

Eine Kante zwischen einem Medienobjektknoten t und einem seiner Vorgänger v ist **redundant**, wenn v in der Vorgängermenge eines weiteren Vorgängers von t enthalten ist:

$$\forall (v, t) \in t\_edges(T): ((v,t) \text{ ist redundant} \Leftrightarrow \exists w \in \tau_{pre}(t,T): v \in \tau_{ps}(w,T)) \qquad (7.15)$$

7.5 Zusätzliche Medienobjekte

Die mit den bisher beschriebenen Algorithmen aufgebaute Medienobjektstruktur T^{b+} eines Blockes b besteht ausschließlich aus inhaltsvermittelnden Medienobjekten, mit denen die in der Inhaltsbeschreibung b_content(b) des Blockes festgelegten Stichworte erlernbar sind.

Um zu der endgültigen Medienobjektstruktur T^{b*} zu gelangen, muß diese „inhaltliche“ Struktur um die vom Autor festgelegten **Zusatzobjekte** – z.B. Grafiken, Beispiele und Aufgaben – erweitert werden.

Jede Gruppe von zusätzlich verwendbaren Objekten wird vom Autor (oder Nutzer) des Lehr- bzw. Informationssystems als Tupel aus Genre, Basis, Genrehäufigkeit, Objekthäufigkeit und der Position innerhalb der Ebenenhierarchie festgelegt (siehe Definition 5.5 in Kapitel 5.3.2):

$$ZO \subseteq string \times Z_{base} \times IR \times IN \times Z_{struct}, \qquad (7.16)$$

$$Z_{base} = \{ \text{"mo"}, \text{"frame"}, \text{"chapter"}, \text{"block"} \}$$

Z_{struct} = { "mo", "chapStart", "chapEnd", "blockStart", "blockEnd" }

z_genre((g, b, f1, f2, p)) := g

z_base((g, b, f1, f2, p)) := b

z_freq_{genre}((g, b, f1, f2, p)) := f1

z_freq_{obj}((g, b, f1, f2, p)) := f2

z_pos((g, b, f1, f2, p)) := p

Die Integration der Zusatzobjekte in die Medienobjektstruktur ist ein dreistufiger Prozeß:

1. Zuerst müssen die verwendbaren Zusatzobjekte ermittelt werden, d.h. die zusätzlichen Medienobjekte, deren benötigtes Vorwissen innerhalb der Medienobjektstruktur überhaupt vermittelbar ist.
2. Anschließend werden die Zusatzobjekte an Medienobjekte gebunden, wobei jedes Zusatzobjekt den Medienobjekten zugewiesen wird, zu denen es aufgrund bestimmter inhaltlicher Kriterien „paßt".
3. Nach der Zuordnung wird eine initiale Quotierung vorgenommen, um die Anzahl der jedem Medienobjekt zugewiesenen Zusatzobjekte zu minimieren.

Um den Aufwand der Integration von Zusatzobjekten in die Medienobjektstruktur so gering wie möglich zu halten, werden auch bei der Integration von Zusatzobjekten Heuristiken verwendet, die im wesentlichen auf den in Kapitel 6 beschriebenen Maßzahlen basieren.

7.5.1 Ermitteln der verwendbaren Zusatzobjekte

Jedes Zusatzobjekt, das zu einer existierenden Medienobjektstruktur G^{b*} eines Blockes b hinzugefügt werden kann, muß zwei Bedingungen erfüllen:

1. Es muß den in der Blockbeschreibung angegebenen Kriterien für zusätzlich verwendbare Objekte genügen.
2. Sein komplettes, verlangtes Vorwissen muß durch die in G^{b*} enthaltenen Medienobjekte, bzw. durch andere auswählbare Zusatzobjekte abgedeckt sein.

Der Algorithmus zur Ermittlung der verwendbaren Zusatzobjekte prüft die beiden genannten Bedingungen nacheinander ab und liefert als Resultat alle Zusatzobjekte zurück, die in die vorgegebene Medienobjektstruktur eingebettet werden können. Zusätzlich wird ein aggregrierter Abhängigkeitsgraph aus den Medienobjekten der Medienobjektstruktur und den verwendbaren Zusatzobjekten zurückgegeben, der später die Bindung der Zusatzobjekte erleichtert.

Bei der im ersten Schritt durchgeführten Auswahl anhand der Blockbeschreibung werden aus den insgesamt verfügbaren Medienobjekten diejenigen ausgewählt, deren Genre in der Liste der zusätzlich verfügbaren Objekte angegeben ist. Da jedoch auch diese Medienobjekt über verlangte und vermittelte Stichworte verfügen, kann erst durch eine Analyse ihrer Indexeinträge ermittelt werden, welche von ihnen für die Integration in die vorgegebene Medienobjektstruktur geeignet sind. Hierbei müssen zwei Bedingungen erfüllt sein:

1. Das gesamte verlangte Vorwissen des Zusatzobjektes muß abgedeckt werden können.
2. Das Zusatzobjekt sollte nicht ausschließlich Stichworte vermitteln, die im Rahmen der bestehenden Medienobjektstruktur noch nicht erwähnt wurden.

Zur Überprüfung des ersten Kriteriums wird ein Abhängigkeitsgraph aus bereits ausgewählten Medienobjekten und in Frage kommenden Zusatzobjekten erstellt. Der Vorteil dieser Vorgehensweise ist, daß nicht nur auf bereits vorhandene Algorithmen wie z.B. die Normalisierung zurückgegriffen werden kann, sondern daß insbesondere auch die dabei berechneten Maßzahlen des Abhängigkeitsgraphen – minimale Tiefen und Vorgängermengen – für die nachfolgende Bindung der Zusatzobjekte verfügbar sind.

7.5.2 Auswahl von Zusatzobjekten

Nachdem alle verwendbaren Zusatzobjekte ermittelt sind, muß im nächsten Schritt eine Verbindung zwischen Basis- und Zusatzobjekten hergestellt werden. Die zentrale Frage hierbei ist, zu welchen Medienobjekten jedes der verfügbaren Zusatzobjekte „paßt". Bei der anschließenden ersten Quotierung und der Erstellung der Seitenstruktur werden – abhängig von den Vorgaben des Autors - ausschließlich aus den „passenden" Zusatzobjekten die Objekte ausgewählt, die im erzeugten Dokument Verwendung finden. Wichtig bei der endgültigen Auswahl ist vor allem die gewünschte Anordnung der Zusatzobjekte in der Ebenenhierarchie:

Position	Verwendung eines zu einem Medienobjekt m passenden Zusatzobjektes z
beim Medienobjekt	z wird in de Seitenstruktur auf der selben Seite wie m positioniert.
gesonderte Seite	Aus den „passenden" Zusatzobjekten der Medienobjekte eines Kapitels werden einige ausgewählt und als eigene Seite innerhalb der Seitenstruktur des betrachteten Kapitels eingefügt.
gesondertes Kapitel	Aus den „passenden" Zusatzobjekten der Medienobjekte eines Blocks werden einige ausgewählt und als eigenes Kapitel innerhalb der Kapitelstruktur des betrachteten Blocks eingefügt.

Um zu einem späteren Zeitpunkt die endgültige Auswahl und Anordnung der Zusatzobjekte berechnen zu können, muß bereits bei der Strukturierung der Medienobjekte eine Vorauswahl getroffen werden, da viele der benötigten Information (z.B. der aggregierte Abhängigkeitsgraph) nach der Strukturierung nicht mehr verfügbar sind. Aus diesem Grund wird als Teil der Strukturierung jedem Medienobjektknoten eine Liste all der Zusatzobjekte angefügt, die später als Ergänzung zu diesem Basisobjekt auf der selben Seite, im selben Kapitel oder im selben Block eingefügt werden können.

Die Bindung von Zusatzobjekten an Knoten der Medienobjektstruktur ist ein zweiteiliger Prozeß:

> Zuerst werden anhand von inhaltlichen Gesichtspunkten zu jedem Medienobjekt die Zusatzobjekte ausgewählt, die überhaupt zu diesem Objekt „passen". Anschließend werden aufgrund der Vorgaben des Autors bezüglich der Frequenz des Vorkommens von Zusatzobjekten, nur die besten der für ein Medienobjekt in Frage kommenden Zusatzobjekte an den entsprechenden Medienobjektknoten gebunden. Die Vorauswahl ist Thema dieses Abschnitts, während die die endgültige Bindung in Kapitel 7.5.3 behandelt wird.

Um die „passenden" Paare von Basis- und Zusatzobjekten bestimmen zu können, wird eine ausreichend formale Festlegung benötigt, wann zwei Objekte zueinander passen.

Ein zwingendes Kriterium ist dabei, daß alle von dem Zusatzobjekt benötigten Stichworte dem Leser bei Erreichen des Basisobjekts bereits vermittelt wurden. Dies bedeutet, daß das Vorwissen des Zusatzobjektes implizit von einem „passenden" Medienobjekt abgedeckt werden muß, was wiederum nichts anderes heißt, als daß alle direkten Vorgänger des Zusatzobjektes in der Vorgängermenge des Basisobjektes enthalten sein müssen.

Die einzige Ausnahme von dieser Regel bilden Zusatzobjekte, die Vorwissen enthalten, das exklusiv von einem anderen Zusatzobjekt vermittelt wird. Hier müssen anstatt dieses Zusatzobjekts dessen Vorgänger in der Vorgängermenge des Basisobjektes enthalten sein.

Neben diesem zwingenden Kriterium gibt es ein zweites, inhaltliches Kriterium, dem zufolge das Medienobjekt entweder Vorgänger des „passenden" Zusatzobjektes sein muß oder aber die Schnittmenge der vermittelten Stichworte nicht leer ist. Aufgrund dieser zusätzlichen Einschränkungen werden nur Objekte aneinander gebunden, die eine gewisse inhaltliche Nähe besitzen, z.B. eine Aufgabe zur Varianz an ein Medienobjekt zu Varianz oder eine Grafik zum Euro-Zeitplan an ein Medienobjekt zum selben Thema.

Satz

Ein Zusatzobjekt z "paßt" genau dann zu einem Medienobjekt m in einem Abhängigkeitsgraphen G, wenn:

1. $(m \in \mu_{pre}(z,G)$ oder $(\mu_{kwprv}(m,I(G)) \cap \mu_{kwprv}(z,I(G)) \neq \emptyset))$ und
2. $((\mu_{pre}(z,G) \setminus M^a) \subseteq \mu_{ps}(m,G)$ und
3. $\forall o \in (\mu_{pre}(z,G) \cap M_a): \mu_{pre}(o,G) \subseteq \mu_{ps}(m,G))$ (7.17)

Die Berechnung der für jedes in der Medienobjektstruktur enthaltene Medienobjekt „passenden" Zusatzobjekte liefert drei Vektoren B, A und F:

- Der Vektor $B = <B_1, B_2, ..., B_{Mmax}>$ enthält für jedes in der Medienobjektstruktur enthaltene Medienobjekt m die Menge aller zu m „passenden" Zusatzobjekte: $B_m = \{ z \mid z$ „paßt" zu m $\}$
- Der Vektor $A = <A_1, A_2, ..., A_{Mmax}>$ enthält umgekehrt für jedes Zusatzobjekt z die zu diesem Objekt „passenden" Medienobjekte der Medienobjektstruktur: $A_z = \{ m \mid z$ „paßt" zu m $\}$
- Der Vektor $F = <F_1, F_2, ..., F_{Mmax}>$ enthält für jedes Vorwissen vermittelnde Zusatzobjekt z die Menge der Zusatzobjekte, die genau dieses Vorwissen benötigen: $F_z = \{m \in M^{ab} \mid m \in \mu_{succ}(z, G^{a*})\}$

Die gegenseitige Referenzierung von Basis- und Zusatzobjekten durch die Vektoren A und B wird von der im nächsten Abschnitt beschriebenen Quotierung benötigt, um die vorgegebenen Genre- und Objekthäufigkeiten der einzelnen Klassen von Zusatzobjekten berücksichtigen zu können.

7.5.3 Initiale Quotierung

Den Abschluß der Integration von Zusatzobjekten in die Medienobjektstruktur bildet die **initiale Quotierung**. Hierbei werden vor allem zwei Ziele verfolgt:

- Bindung der Zusatzobjekte an Medienobjektknoten.
- Berücksichtigung der in der Blockbeschreibung angegebenen Werte bezüglich der Häufigkeit der Verwendung von Zusatzobjekten.

Die Häufigkeit des Auftretens von Zusatzobjekten im fertigen Lehr- oder Informationssystem kann vom Autor durch zwei Parameter der Blockbeschreibung gesteuert werden[38]:

- **z_freq**$_{genre}$(z) Häufigkeit mit der Medienobjekte eines Genres im fertigen Dokument maximal vorkommen dürfen (**Genre-Häufigkeit**). Der Wert bezieht sich normalerweise auf Objekte der Ebenenhierarchie, z.B. "eine Grafik pro Medienobjekt" oder "10 Aufgaben pro Kapitel".
- **z_freq**$_{obj}$(z) Häufigkeit mit der jedes einzelne Zusatzobjekt im fertigen Dokument maximal verwendet werden darf (**Objekt-Häufigkeit**). Der Wert der individuellen Häufigkeit ist immer absolut und wir pro Genre festgelegt, z.B. "jede Aufgabe darf nur einmal vorkommen" oder "Grafiken dürfen bis zu viermal verwendet werden".

Genre- und Objekt-Häufigkeit haben Auswirkungen auf die Bindungen zwischen Medienobjektknoten und Zusatzobjekten. Die Genre-Häufigkeit gibt an, wie viele Zusatzobjekte eines Genres maximal an einen Knoten gebunden werden können, während die individuelle Häufigkeit festlegt, an wieviele Medienobjektknoten ein Zusatzobjekt maximal gebunden werden darf:

$$\forall v \in M^a, \forall z \in b\_additional(b): |\{ m \mid genre(v)=z\_genre(z)\}| \leq z\_freq_{genre}(z) \quad (7.18a)$$

$$\forall v \in M^a, \forall z \in b\_additional(b): (genre(v)=z\_genre(z) \Rightarrow |A_v| \leq z\_freq_{obj}(z)) \quad (7.18b)$$

Die endgültige Zuweisung von Zusatzobjekten zu Medienobjektknoten geschieht erst bei der Erstellung der Seiten- bzw. der Kapitelstruktur. Hierbei tritt jedoch das Problem auf, daß die konkreten Bindungen

[38] siehe auch Kapitel 5.3.2

nur mit exponentiellem Aufwand optimal berechnet werden können. Die Ursache für diesen unvertretbar hohen Aufwand liegt in möglichen Widersprüchen in den Rangfolgen der am besten zueinander passenden Objekte begründet. Zur Verdeutlichung ein Beispiel:

> *Medienobjektknoten m1 paßt am besten zu Zusatzobjekt z1. Zusatzobjekt z1 paßt jedoch am besten zu Medienobjektknoten m2, welcher wiederum am besten zu Zusatzobjekt z3 paßt, das seinerseits optimal zu Medienobjektknoten m1 paßt.*

Um diesem Optimierungsproblem aus dem Wege zu gehen, wird bei der Bindung von Zusatzobjekten an Medienobjekte darauf geachtet, daß jede Bindung gut genug ist, um bei der weiteren Strukturierung ohne erneute Überprüfung übernommen werden zu können.

Hierzu werden alle passenden Paare gewichtet und anschließend solange die am besten bewerteten Paare aneinander gebunden, bis die oben angegebenen Maximalwerte erreicht bzw. alle Paare gebunden sind. Der Nachteil dieses Verfahrens ist, daß bei der Erstellung der Seiten- und Kapitelstruktur herausfallende Bindungen nicht mehr ersetzt werden können, wodurch die Maximalwerte unterschritten werden. Dies bedeutet, daß die Bindungen nur qualitativ, aber nicht quantitativ optimiert werden.

Die Integration der Zusatzobjekte in die Medienobjektstruktur geschieht über die Medienobjektknoten der passenden Medienobjekte. Ein Medienobjektknoten t wurde in Kapitel 7.4 definiert als

$$t \in M^{avail} \times \mathbf{P}(AO^{\#}) \times \{true, false\},$$

wobei das zweite Element alle an diesen Knoten gebundenen Zusatzobjekte enthält.

Jedes **gebundene Zusatzobjekt** enthält das Zusatzobjekt selbst als Medienobjektknoten, die zur Klasse des Zusatzobjekts gehörenden Angaben aus der Blockbeschreibung, sowie die Menge der eventuell zur Vermittlung benötigten Zusatzobjekte:

$$AO^{\#} \subseteq TM^{\#} \times ZO \times \mathbf{P}(M^{avail}) \qquad (7.19)$$

Die Markierung f eines Medienobjektknotens (siehe Kapitel 7.4) dient der Kennzeichnung von ungebundenen Knoten. Bei der Strukturierung fallen ausschließlich als Vorwissen benötigte Zusatzobjekte in diese Kategorie. Nachfolgende Algorithmen erkennen diese Objekte daran, daß *tm_flag(t)* des zugehörigen Medienobjektknotens t den Wert "false" besitzt.

Das Ergebnis der initialen Quotierung ist die endgültige Medienobjektstruktur des zu erzeugenden Lehr- bzw. Informationssystems.

7.6 Erstellung der Seitenstruktur

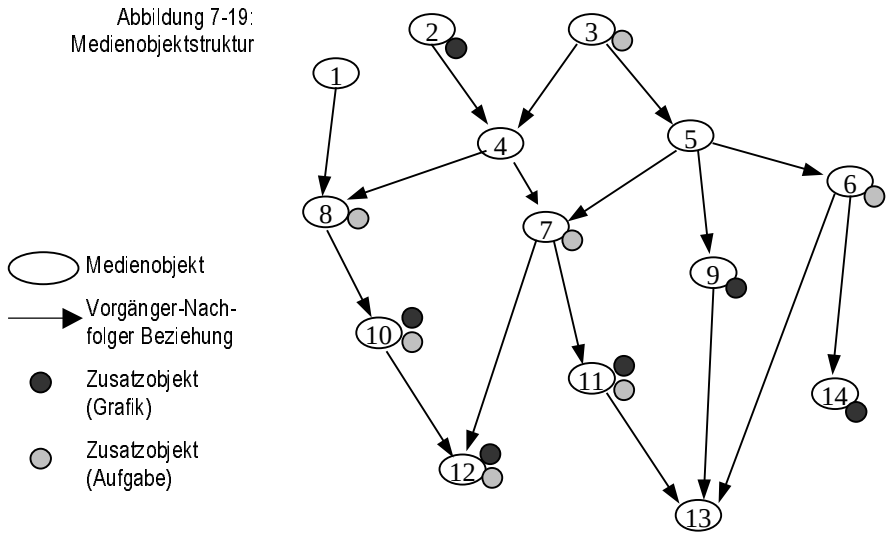
Nachdem die unterste Ebene des zu generierenden Lehr- oder Informationssystems – die Medienobjektstruktur – erstellt ist, müssen daraus die nächsthöheren Ebenen abgeleitet werden. Jede dieser Ableitung verfolgt dabei zwei Ziele:

- Zusammenfassung von Objekten der niedrigeren Ebene zu Objekten der höheren Ebene (Medienobjekte zu Seiten, Seiten zu Kapiteln)
- Annäherung der Struktur an die für Lehr- und Informationssysteme typische hierarchische Baumstruktur

Da diese beiden Ziele oftmals im Konflikt zueinander stehen, ist eine Ableitung von einer niedrigeren auf eine höhere Ebene in vertretbarer Zeit nur mit Hilfe von Heuristiken möglich.

Abbildung 7-19 zeigt eine typische Medienobjektstruktur, wie sie durch die beschriebenen Auswahl- und Strukturierungsalgorithmen erstellt worden sein könnte. Die numerierten Ovale repräsentieren Medienobjektknoten, die Pfeile Vorgänger-Nachfolger Beziehungen. Die grauen Kreise stellen zwei beliebige Klassen von an die Medienobjektknoten gebundenen Zusatzobjekten dar (z.B. Aufgaben und Grafiken).

Abbildung 7-19: Medienobjektstruktur



Das Hauptproblem beim Übergang von Medienobjekten auf **Seiten** stellt die Überführung des potentiell stark vernetzten Graphen der Medienobjektstruktur in eine hierarchische **Seitenstruktur** dar. Ziel ist es, eine Seitenstruktur zu berechnen, in der jede Seite maximal eine Vorgängerseite besitzt.

Um dieses Ziel zu erreichen, wird auch bei der Seitenstruktur zwischen Vorgänger-Nachfolger Beziehungen und einfachen Referenzen (Hyperlinks) unterschieden. Jede Seite darf somit zwar nur einen direkten Vorgänger besitzen, aber zusätzlich mit beliebig vielen anderen Seiten über Hyperlinks verbunden sein.

In Abbildung 7-19 verhindern z.B. die Medienobjektknoten 4, 7, 12 und 13 die Erstellung einer hierarchischen Seitenstruktur. Jeder dieser Medienobjektknoten hat mehr als einen Vorgänger, was bedeutet, daß vor der Zusammenfassung von Medienobjektknoten zu Seiten eine Reihe von Vorgänger-Nachfolger Beziehungen durch Hyperlinks ersetzt werden müssen.

Um eine möglichst gute Zusammenfassung von Medienobjektknoten und eine hierarchische Seitenstruktur zu erzielen, verwendet der Strukturierungsalgorithmus des *i4 Frameworks* die folgende Strategie:

1. Jeder Medienobjektknoten wird auf eine einzelne Seite abgebildet. Vorgänger-Nachfolger Beziehungen und Hyperlinks der Medienobjektstruktur werden dabei unverändert übernommen.
2. Sequentiell strukturierte Subgraphen der Seitenstruktur werden - falls möglich - zusammengefaßt oder durch Hyperlinks vom Rest des Graphen isoliert.
3. Anschließend werden nach und nach Vorgänger-Nachfolger Beziehungen durch Hyperlinks ersetzt, bis keine Seite mehr als eine Vorgängerseite hat. Nach jeder Ersetzung wird zu Schritt 2 zurückgesprungen, um eventuell neu entstandene Sequenzen zusammenzufassen.

Das Ergebnis ist die Seitenstruktur des zu erstellenden Lehr- oder Informationssystems[39].

[39] Zu dieser Seitenstruktur können bei der logischen Strukturierung unter Umständen noch zusätzliche Seiten für auf Kapitel- oder Blockebene angesiedelte Zusatzobjekte hinzugefügt werden, so daß es sich streng genommen nicht zwangsläufig um die endgültige Seitenstruktur handelt.

Algorithmus CreateFrames

Aufgabe: Abbildung einer gegebenen Medienobjekt-Struktur auf eine hierarchische Seitenstruktur

Eingabe: Medienobjektstruktur T^{b*}

Blockbeschreibung b

Ausgabe: Aus der Medienobjektstruktur T^{b*} abgeleitete Seitenstruktur F^{b*}

Durchführen einer 1:1 Abbildung von Medienobjekten auf Seiten

Zusammenfassen sequentiell strukturierter Seiten zu einer einzelnen Seite

[while] Solange Seiten mit mehr als einer Vorgänger-Seite existieren

Umwandlung einer Eingangskante einer Seite in einen Hyperlink

Zusammenfassen sequentiell strukturierter Seiten zu einer einzelnen Seite

[end while]

Der Algorithmus "CreateFrames" liefert neben der Seitenstruktur noch sog. *Ordering Informationen* für die Kapitelerstellung zurück. Diese werden in Kapitel 7.6.3 ausführlicher behandelt.

7.6.1 Seiten

Die Seitenstruktur eines Lehr- oder Informationssystems ist weitestgehend identisch mit seiner Medienobjektstruktur. Der einzige Unterschied ist, daß Seiten aus mehreren Medienobjekten bestehen können und daß nicht alle Vorgänger-Nachfolger Beziehungen der Medienobjektstruktur in die Seitenstruktur übernommen werden.

Definition

Die **Seitenstruktur** F eines Lehr- oder Informationssystems ist ein gerichteter, azyklischer, mit Kantengewichtungen versehener Graph aus einzelnen Seiten, der zusätzlich die aus der Medienobjektstruktur übernommenen Hyperlinks enthält. Alle Seitenstrukturen zusammen bilden die Menge $F^{\#}$:

$$F^{\#} \subseteq P(FT^{\#}) \times \mathbf{P}(FT^{\#} \times FT^{\#} \times IR) \times \mathbf{P}(H^{\#}) \qquad (7.20)$$

f_nodes((n, e, l)) := n

f_edges((n, e, l)) := e

f_links((n, e, l)) := l

Die Gewichtung der Kanten ist identisch mit der Bewertung der entsprechenden Kanten des zugrundeliegenden Abhängigkeitsgraphen (siehe Kapitel 6.1.3).

Definition

Jede **Seite** der Seitenstruktur setzt sich aus einem oder mehreren geordneten Medienobjektknoten und den von diesen Knoten ausgehenden Hyperlinks zusammen[40]. Eine zusätzliche Markierung gibt an, ob die Seite Teil der hierarchischen Seitenstruktur, oder nur über Hyperlinks mit dieser verbunden ist:

$$FT^{\#} \subseteq \mathbf{P}(TM^{\#}) \times \mathbf{P}(H^{\#}) \times \{\mathit{Tree}, \mathit{Linked}\} \qquad (7.21)$$

Wie bei allen anderen Datenstrukturen, ist auch hier der Zugriff auf einzelne Bestandteile einer Seite über Projektionen definiert:

ft_nodes((n,l,f)) = n

ft_links((n,l,f)) = l

ft_tree((n,l,f)) = f

Alle Medienobjektknoten der Medienobjektstruktur T^{b*} eines Blocks b sind Bestandteil genau einer Seite der Seitenstruktur F^{b*} dieses Blocks:

$$\forall f, g \in FT^{\#}: (t \in \text{ft\_nodes}(f) \text{ und } t \in \text{ft\_nodes}(g) \Rightarrow f = g) \qquad (7.22)$$

Wie auf den Knoten der Medienobjektstruktur so sind auch auf den Seiten f einer Seitenstruktur F **Vorgänger**, **Nachfolger** und **Vorgängermengen** berechenbar:

$$\phi_{pre}(f,F) := \{ g \mid \exists(g,f,z) \in \text{f\_edges}(F) \} \qquad (7.23)$$

$$\phi_{succ}(f,F) := \{ g \mid \exists(f,g,z) \in \text{f\_edges}(F) \} \qquad (7.24)$$

$$\phi_{ps}(f,F) := \phi_{pre}(f,F) \cup \bigcup_{g \in \phi_{pre}(f,F)} \phi_{ps}(g,F) \qquad (7.25)$$

Im Gegensatz zur Erstellung der Medienobjektstruktur werden für die Seitenstrukturierung auch die **Nachfolgermengen** aller Seiten benötigt. Diese ergeben sich aus einer Umkehrung der Vorgängermengen:

$$\phi_{ss}(f,F) := \{ g \mid f \in \phi_{ps}(g,F) \} \qquad (7.26)$$

Darüber hinaus werden noch zwei weitere Abbildungen ϕ_{in}: $FT^{\#} \times F^{\#} \rightarrow P((FT^{\#} \times FT^{\#} \times \mathit{IR}))$ und ϕ_{out}: $FT^{\#} \times F^{\#} \rightarrow P((FT^{\#} \times FT^{\#} \times \mathit{IR}))$ definiert, die zu einer Seite f alle Eingangs- bzw. Ausgangskanten innerhalb einer gegebenen Seitenstruktur F liefern:

$$\phi_{in}(f,F) := \{ (x,y,z) \in F \mid y = f \} \qquad (7.27)$$

$$\phi_{out}(f,F) := \{ (x,y,z) \in F \mid x = f \} \qquad (7.28)$$

Da bei der Erstellung der Seitenstruktur keine neuen Hyperlinks erzeugt werden (Hyperlinks existieren nur zwischen Medienobjekten, nicht zwischen Seiten!), können die Hyperlinks der Medienobjektstruktur unverändert übernommen werden. Durch die Zuordnung eines Medienobjektknotens zu einer Seite werden auch alle von diesem Knoten ausgehenden Hyperlinks an diese Seite gebunden und damit aus den global verwalteten Links der Seitenstruktur enfernt. Dieses etwas umständlich erscheinende Verfahren ist notwendig, um Vorwissen vermittelnde Zusatzobjekte bei der Strukturierung erst so spät wie möglich in Seiten oder Kapitel einbinden zu können.

Vorgänger-Nachfolger Beziehungen der Seitenstruktur überlagern die Hyperlinks der Medienobjektstruktur. Durch das Entfernen von die hierarchische Struktur verletzenden Vorgänger-Nachfolger Beziehungen werden die unterliegenden Hyperlinks der Medienobjektstruktur wieder sichtbar; die direkte

[40] Der Einfachheit halber wird auch für die geordnete Menge der Medienobjektknoten einer Seite die "normale" Mengen-Notation verwendet. Es wird im Folgenden jedoch implizit davon ausgegangen, daß die Vereinigung der Medienobjektknoten zweier Seiten nicht kommutativ ist, d.h. die Reihenfolge der Operanden bei der Vereinigung beeinflußt die Ordnung der Medienobjekte einer Seite.

Beziehung wird somit quasi durch einen Hyperlink ersetzt. Hierdurch ist es möglich, aus einer nicht-hierarchischen Medienobjektstruktur eine hierarchische Seitenstruktur abzuleiten.

Bevor jedoch die eigentliche Strukturierung und Zusammenfassung möglich ist, muß eine initiale Seitenstruktur vorhanden sein. Diese wird erstellt, indem jeder Medienobjektknoten in eine Seite umgewandelt wird. Die Vorgänger-Nachfolger Beziehungen der Medienobjektstruktur werden dabei in die initiale Seitenstruktur übernommen. Wie die Berechnung des Abhängigkeitsgraphen und der Medienobjektstruktur geschieht auch die Berechnung der Seitenstruktur separat für jeden zu erzeugenden Block.

Die Erstellung der initialen Seitenstruktur ist in $O(M(G^{b*}) + I(G^{b*}))$ möglich. Da der Übergang von Medienobjekten zu aus einem Medienobjekt bestehenden Seiten bereits während der Berechnung der Medienobjektstruktur vollzogen werden kann, ist sogar ein linearer Aufwand $O(I(G^{b*}))$ zu erzielen.

7.6.2 Zusammenfassung und Strukturierung

Um von der initialen, vernetzten Seitenstruktur zur endgültigen, hierarchischen Seitenstruktur zu gelangen, müssen Seiten zusammengefaßt und Kanten entfernt werden.

Hierbei wird analog zur Auswahl der Medienobjekte zwischen **einfachen und komplexen Szenarien** unterschieden. Wie beim *Minesweeper* Algorithmus werden auch bei der Seitenstrukturierung zuerst die einfachen Szenarien betrachtet. Als weitere Analogie wird auch bei der Seitenstrukturierung versucht, durch die Auflösung eines komplexen Entscheidungsproblems weitere eventuell bestehende komplexe Probleme auf einfache Szenarien zurückzuführen.

Abbildung 7-20 zeigt die initiale Seitenstruktur der in Abbildung 7-19 dargestellten Medienobjektstruktur. Seiten sind als mit Großbuchstaben gekennzeichnete Rechtecke um die enthaltenen Medienobjektknoten (samt Zusatzobjekten) dargestellt. Die Kanten des Graphen repräsentieren die existierenden Vorgänger-Nachfolger Beziehungen der Seitenstruktur.

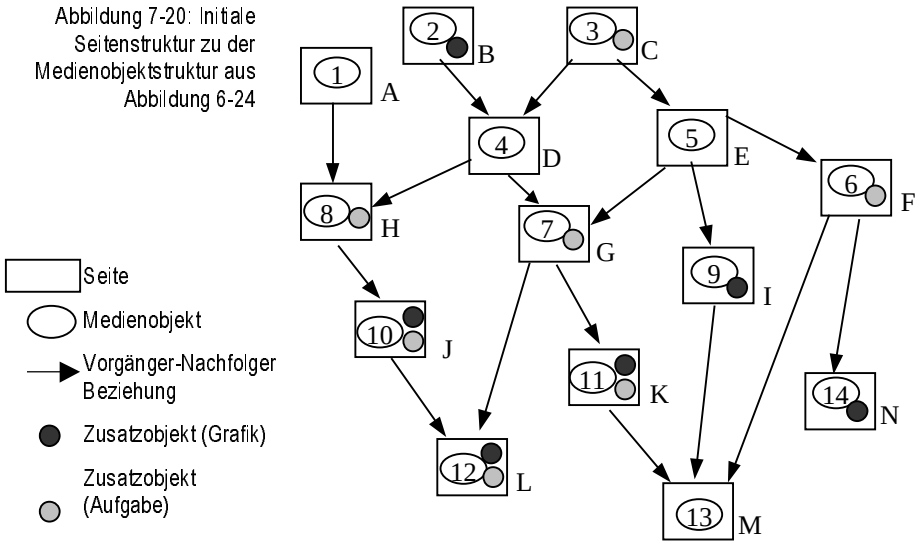


Abbildung 7-20: Initiale Seitenstruktur zu der Medienobjektstruktur aus Abbildung 6-24

Anhand dieser Beispielstruktur sollen in diesem Kapitel zuerst die einfachen Szenarien und anschließend in Abschnitt 6.6.3 die komplexen Szenarien beschrieben werden.

Szenario 1: Wurzelseiten

Das erste der einfachen Szenarien betrifft die **Wurzeln** der Seitenstruktur, d.h. alle Seiten ohne Vorgänger. Die Seitenstruktur in Abbildung 7-20 enthält z.B. die Wurzelseiten A, B und C.

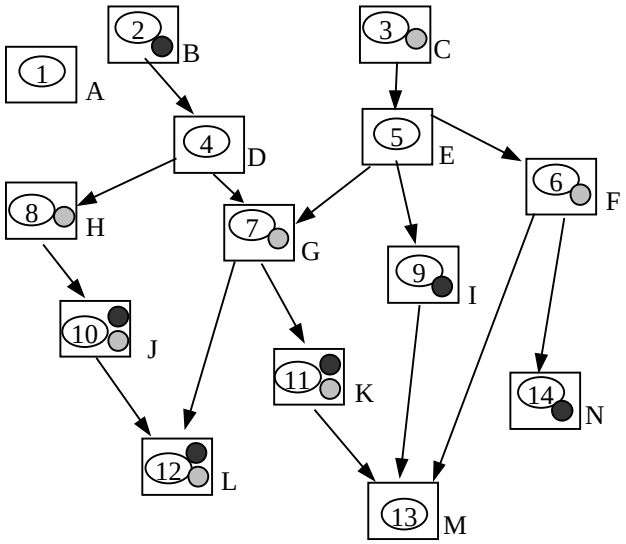
Eine Kante von einer Wurzelseite f zu einer nachfolgenden Seite g (f→g) kann ausschließlich duch den zugrundeliegenden Hyperlink der Medienobjektstruktur repräsentiert werden, wenn die folgenden Kriterien erfüllt sind:

- Die Wurzelseite f enthält nur ein einziges Medienobjekt, d.h. $|ft\_nodes(f)|=1$
- Das in der Wurzelseite f enthaltene Medienobjekt m ist als "Popup" gekennzeichnet (siehe Kapitel 4.3)
- Falls die Nachfolgerseite g mehr als einen Vorgänger besitzt, muß die Kante f→g von allen Eingangskanten von g diejenige mit der niedrigsten Bindung sein: $(f,g,e) \in f\_edges(F^{b+})$ kann entfernt werden $\Rightarrow \forall(x,g,y) \in f\_edges(F^{b+}): y \geq e$

Wenn alle drei Bedingungen erfüllt sind, kann die Kante f→g aus der Seitenstruktur entfernt und die Seite f als "Linked" markiert werden.

Für die Beispielstruktur sei angenommen, daß die Kanten A→H und C→D diese Bedingungen erfüllen. Nach Entfernung dieser Kanten ergibt sich die in Abbildung 7-21 dargestellte Seitenstruktur.

Abbildung 7-21: Seitenstruktur aus Abbildung 6-25 nach Entfernung der Kanten A→H und C→D



Die Seite A der Beispielstruktur ist nach der Entfernung der Kante zu Seite H isoliert, d.h. sie besitzt weder Vorgänger noch Nachfolger. Isolierte Seiten können im fertigen Lehr- oder Informationssystems über sog. *Popups*, d.h. die aktuell dargestellte Seite teilweise überlagernde Fenster, dargestellt werden.

Szenario 2: Einfache Sequenzen

Zwei Seiten f und g bilden eine einfache **Sequenz** innerhalb einer Seitenstruktur F, wenn g der einzige Nachfolger von f und f der einzige Vorgänger von g ist:

$$\forall f, g \in f\_nodes(F): (f \text{ und } g \text{ bilden eine Sequenz} :\Leftrightarrow \phi_{succ}(f,F)=\{g\} \text{ und } \phi_{pre}(g,F)=\{f\} \quad (7.29)$$

In der Beispielstruktur in Abbildung 6-26 bilden die Seiten H und J eine einfache Sequenz.

Die Seiten einer einfachen Sequenz können zu einer einzigen Seite zusammengefaßt werden, wenn die folgenden vier Bedingungen erfüllt sind:

- Der Grad der Bindung zwischen den beiden Seiten muß einen bestimmten Grenzwert ε^b (z.B. 0,5) überschreiten.
- Falls die Seite g eine Nachfolgerseite h besitzt, muß die Bindung zwischen f und g stärker sein als die Bindung zwischen g und h.
- Die Genres der in den Seiten enthaltenen Medienobjekte müssen zueinander passen.

- Die zusammengefaßte Seite muß mit dem vom Autor gewählten Seitenspiegel darstellbar sein.

Falls innerhalb einer Seitenstruktur mehrere einfache Sequenzen existieren, werden sie in absteigender Reihenfolge ihres Bindungsgrades analysiert und gegebenenfalls zusammengefaßt. Hierdurch werden komplexe Sequenzen aus mehr als zwei Seiten auf geschachtelte einfache Sequenzen zurückgeführt.

Die Überprüfung, ob die **Genres** der auf den zusammenzufassenden Seiten liegenden Medienobjekte harmonieren, geschieht anhand einer einfachen Tabelle. Hierbei wird für jede mögliche Kombination der unterstützten Genres vermerkt, ob zwei Medienobjekte dieser Genres in der vorliegenden Reihenfolge auf einer Seite zusammengefaßt werden können:

Miteinander harmonierende Seiten-Genres

2. Objekt / 1. Objekt	*content*	*document*	*comment*	*dispatcher*	*example*	*problem*
content	ok	ok	ok	ok	ok	ok
document	ok	ok	ok	ok	-	-
comment	ok	ok	ok	ok	ok	ok
dispatcher	-	-	-	-	-	-
example	ok	ok	ok	ok	-	ok
problem	-	ok	ok	ok	-	ok

Für die Beispielstruktur sei angenommen, daß die einfache Sequenz aus den Seiten H und J die zweite Bedingung (Bindungsstärke) verletzt und die beiden Seiten somit nicht zusammengefaßt werden können.

Abtrennung von Wurzelseiten und Zusammenfassung von Sequenzen

Die Abtrennung schwach angebundener Wurzelseiten sowie die Zusammenfassung von Sequenzen wird iterativ solange ausgeführt, bis die Seitenstruktur keine Knoten und Kanten mehr enthält, die auf die beiden genannten einfachen Szenarien zurückgeführt werden können. Die Iteration ist notwendig, da vor allem das Loslösen von Wurzelseiten zu neuen Wurzelseiten oder Sequenzen führen kann. Darüber hinaus können auch Sequenzen aus mehr als zwei Seiten durch die Iteration sehr einfach zusammengefaßt werden.

Das durchschnittliche Laufzeitverhalten dieses iterativen Algorithmus ist sehr stark von der Vernetzung der Seitenstruktur abhängig. Da die in der ersten Iteration optimierbaren Knoten und Kanten bereits bei der initialen Strukturierung identifizierbar sind und weitere Iterationen auf einem schwächer vernetzten Graphen operieren, kann jedoch von einer Komplexität nahe $O(f\_edges(F^{b*}))$ ausgegangen werden.

7.6.3 Seiten mit mehreren Vorgängern

Auch nach der Entkopplung schwach angebundener Wurzelseiten und der Zusammenfassung von Sequenzen kann die Seitenstruktur Seiten mit mehr als einer Eingangskante enthalten. Das Problem bei diesen Seiten ist, daß sie in der fertigen Seitenstruktur nur einen Vorgänger besitzen dürfen, damit das endgültige Dokument eine Baumstruktur besitzt. Es muß somit entschieden werden, welche der Eingangskanten bestehen bleibt, bzw. welche der Eingangskanten am besten durch Hyperlinks ersetzbar sind.

Szenario 3: Durch Vorgängermengen abgesicherte Hyperlinks

Ähnlich wie bei der Entfernung von Abkürzungen im Zuge der Erstellung der Medienobjektstruktur sollte bei der Seitenstrukturierung sichergestellt sein, daß trotz Entfernen von Kanten das von jedem Knoten benötigte Vorwissen – zumindest zu einem Großteil - innerhalb seiner Vorgängermenge vermittelt wird.

In der Praxis ergibt sich oftmals die Konstellation, daß eine Kante gefahrlos aus der Seitenstruktur entfernt werden kann, da der dadurch entstehende Hyperlink durch die Vorgängermenge des Zielknotens der Kante abgesichert ist.

Definition

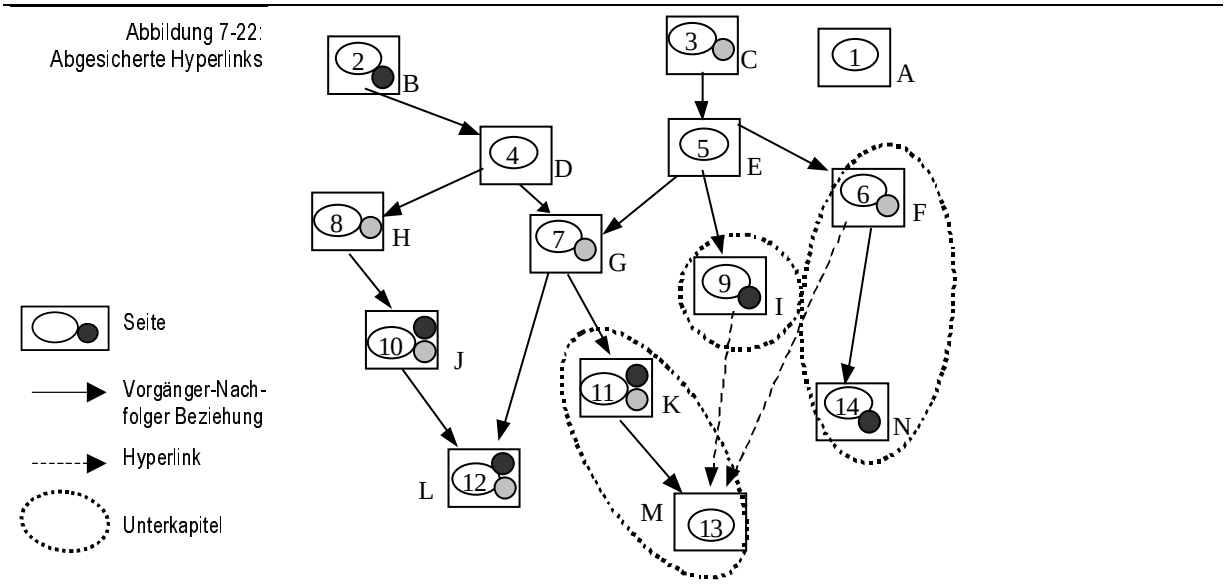
Ein Hyperlink, der innerhalb einer Seitenstruktur F eine Seite f mit einer Seite g verbindet, gilt als **abgesichert**, wenn die komplette Vorgängermenge von f in der Vorgängermenge mindestens eines anderen Vorgängers von g enthalten ist:

$$\forall (f,g,e) \in f\_edges(F): ((f,g,e) \text{ ist abgesichert} :\Leftrightarrow \exists h \in \phi_{pre}(g): \phi_{ps}(f) \subseteq \phi_{ps}(h)) \quad (7.30)$$

Für den Benutzer des generierten Lehr- oder Informationssystems bedeutet dies, daß beim Betrachten einer Seite zwar Teile des benötigten Vorwissens noch nicht vermittelt, aber über Hyperlinks verfügbar sind. Wichtig dabei ist, daß die durch Hyperlinks vermittelten Informationen verständlich sind, da ihr verlangtes Vorwissen bereits komplett über den Pfad zur Startseite des Links vermittelt wurde.

Abgesicherte Hyperlinks können durch die abschließenden Erstellung der Kapitelstruktur sogar komplett redundant werden, wenn die Strukturierung so erfolgt, daß die am Ende des Hyperlinks befindliche Seite vor der Ausgangsseite liegt und somit vom Benutzer zuerst betrachtet wird.

Im Beispielgraphen in Abbildung 7-21 sind die Kanten I→M und F→M durch abgesicherte Hyperlinks ersetzbar. Durch Entfernen beider Kanten aus der Beispielstruktur entsteht die in Abbildung 7-22 dargestellte Seitenstruktur. Die abgesicherten Hyperlinks sind als gepunktete Pfeile eingezeichnet.



Die in Abbildung 7-22 eingezeichneten Ovale stellen eine mögliche logische Zusammenfassung von Seiten zu logischen Einheiten (z.B. Unterkapiteln) dar. Falls bei der Kapitelstrukturierung die aus den Seiten K und M bestehende Einheit nach den beiden anderen Einheiten angeordnet wird, sind zu dem Zeitpunkt, an dem der Leser Seite M erreicht, alle Vorwissen vermittelnden Seiten bereits betrachtet worden. Dies bedeutet, daß abgesicherte Hyperlinks durch geschickte Kapitelstrukturierung von "normalen" Referenzen kaum unterscheidbar sind.

Szenario 4: Nicht-Abgesicherte Hyperlinks

Leider lassen sich nicht alle Knoten und Kanten innerhalb nicht-baumstrukturierter Seitenstrukturen auf die drei bisher angeführten Szenarien zurückführen. Um dennoch zu einer hierarchischen Seitenstruktur zu gelangen, müssen oftmals Kanten zu Hyperlinks reduziert werden, die weder redundant noch abgesichert sind.

In der Beispielstruktur betrifft dies die Knoten G und L. Keine der Kanten D→G, E→G, G→L oder J→L kann aus der Seitenstruktur entfernt werden, ohne daß potentiell eine Verletzung der in der Medienobjektstruktur implizit festgelegten Betrachtungsreihenfolge auftritt.

Um zu einer hierarchischen Seitenstruktur zu gelangen, müssen somit unter Umständen auch nicht abgesicherte Kanten entfernt werden. Aufgabe des Strukturierungsalgorithmus ist es dabei, die Kanten zu identifizieren, die am besten durch einfache Hyperlinks ersetzt werden können. Die wichtigsten Kriterien hierbei sind, daß der Fluß des Dokuments möglichst nicht unterbrochen wird und durch geeignete Kapitelstrukturierung eine nachträgliche Absicherung des Hyperlinks möglich ist.

Diese genannten Kriterien spiegeln sich in zwei Maßzahlen der Seitenstruktur wieder:

- Bindungsstärke
- Direkte Nachfolger einer Seite, die über Hyperlinks Vorgänger dieser Seite referenzieren

Das zweite Kriterium berücksichtigt bereits die nachfolgende Kapitelstrukturierung, indem der Verbleib von Kanten unterstützt wird, die zu kompakten Kapitelstrukturen - d.h. möglichst viele Hyperlinks sind kapitelintern und nicht kapitelübergreifend - führen.

Das Ersetzen einer Kante durch einen nicht abgesicherten Hyperlink kann potentiell zu einer Seitenstruktur führen, in der beim Erreichen einer Seite nicht das komplette benötigte Vorwissen bereits bekannt ist oder durch einen einzelnen Hyperlink vermittelt werden kann. Um die damit verbundenen Probleme zu minimieren, muß bei der Kapitelstrukturierung darauf geachtet werden, daß über nicht abgesicherte Hyperlinks angebundene Seiten vor ihren referenzierenden Seiten liegen. Hierdurch wird der Hyperlink quasi nachträglich abgesichert.

Ordering Informationen

Um die nachträgliche Absicherung von Hyperlinks vornehmen zu können, müssen entsprechende Informationen an die Algorithmen zur Kapitelstrukturierung weitergereicht werden. Diese Informationen werden **Ordering Informationen** genannt:

Definition:

Ordering Informationen legen die Reihenfolge der auf einer Hierarchiestufe liegenden Kapitel bzw. Unterkapitel fest. Jede Ordering Information o setzt sich aus zwei in struktureller Abhängigkeit zueinander stehenden Seiten und einer Markierung zusammen. Die Menge aller Ordering Informationen wird mit $O^{\#}$ bezeichnet:

$$O^{\#} \subseteq FT^{\#} \times FT^{\#} \times \{True, False\} \tag{7.31}$$

Die beiden in einer Ordering Information o = (*frame1*, *frame2*, f) enthaltenen Seiten geben die durch einen Hyperlink verursachte strukturelle Abhängigkeit an, d.h. *frame1* vermittelt Informationen, die von *frame2* über einen Hyperlink referenziert werden. Dies bedeutet, daß *frame1* in der Kapitelstruktur vor *frame2* plaziert werden sollte. Das zusätzliche Flag zeigt an, ob der der Ordering Information zugrundeliegende Hyperlink abgesichert ist (f = True) oder nicht (f = False).

Für nicht abgesicherte Hyperlinks ist die in der Ordering Information angegebene Kapitelstrukturierung zwingend, während sie für abgesicherte Hyperlinks optional ist.

Wichtig für die Erstellung der Seitenstruktur ist, daß Ordering Informationen unter Umständen widersprüchlich sein können. Aus diesem Grunde fließen bereits vorhandenen Ordering Informationen in die Auswahl der am besten durch einen Hyperlink zu ersetzenden Kanten ein.

Ersetzen von Kanten durch abgesicherte und absicherbare Hyperlinks

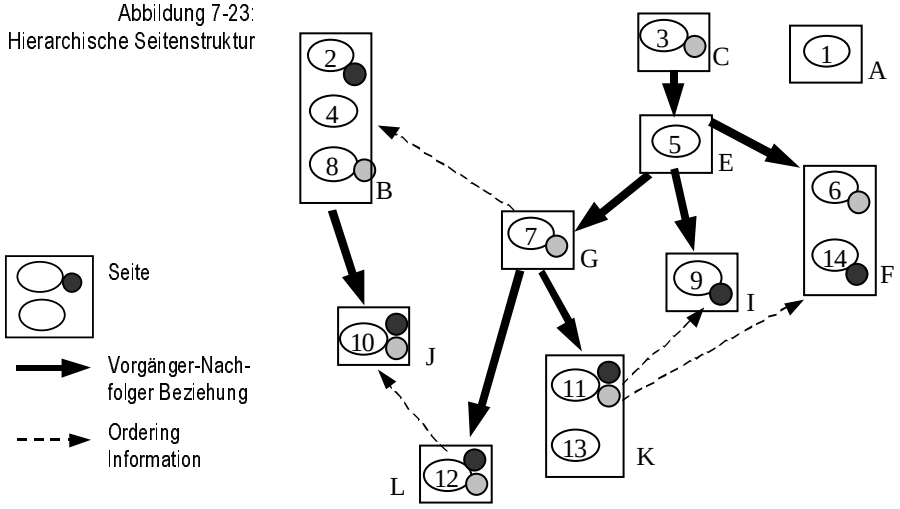
Die beiden Szenarien zur Entfernung nicht-hierarchischer Kanten können zu einem Algorithmus zusammengefaßt werden, der Eingangskanten von Seiten mit mehr als einer Vorgängerseite durch Hyperlinks ersetzt und die für die Kapitelstrukturierung benötigten Ordering Informationen erstellt.

Der Algorithmus bricht nach jeder Entfernung einer abgesicherten Kante ab, so daß der übergeordnete Algorithmus zur Seitenstrukturierung erst eventuell entstandene Seiten-Sequenzen zusammenfaßt, bevor weitere Kanten entfernt werden. Dieses Wechselspiel zwischen Kantenentfernung und Seitenzusammenfassung ist notwendig, da durch die Einführung eines abgesicherten Hyperlinks und eine anschließende Zusammenfassung eventuell entstandener Sequenzen eine völlig andere Struktur entsteht, die neu bewertet werden muß.

Wie schon bei den einfachen Szenarien so ist auch das Laufzeitverhalten der Kantenersetzung sehr stark vom Aussehen der initialen Seitenstruktur abhängig. Die größte Komplexität verbirgt sich dabei in der Analyse der Hyperlinks der Nachfolgerseiten. Da die hier berechneten Informationen jedoch statisch sind, kann die Komplexität des Algorithmus durch eine bidirektionale Verkettung von Hyperlinks und deren Zielen erheblich reduziert werden.

Nach Beendigung der Seitenstrukturierung liegt eine hierarchische Seitenstruktur vor. Abbildung 7-23 zeigt eine mögliche Seitenstruktur des Beispielgraphen. Durch Ordering Informationen beschriebene Abhängigkeiten sind dabei durch gestrichelte Pfeile dargestellt.

Abbildung 7-23:
Hierarchische Seitenstruktur



7.7 Erstellung der Kapitelstruktur

Nachdem die Seitenstruktur des gewünschten Dokuments erstellt ist, muß im letzten Schritt daraus eine Kapitelstruktur abgeleitet werden. Die **Kapitelstruktur** C^{b*} eines Blocks b entsteht durch einfache Zusammenfassung und Ordnung von Seiten aus der Seitenstruktur F^{b*}.

Definition:

Jedes **Kapitel der Kapitelstruktur** setzt sich aus den darin enthaltenen Seiten, Verweisen auf das übergeordnete und die untergeordneten Kapitel, der Position innerhalb des übergeordneten Kapitels, sowie einer Startseite zusammen. Die Menge aller denkbaren Kapitel wird als $C^{\#}$ definiert:

$$C^{\#} \subseteq \mathbf{P}(FL^{\#}) \times C^{\#} \times \mathbf{P}(C^{\#}) \times IN \times FL^{\#} \qquad (7.32)$$

c_frames((f, p, q, o, s) := f

c_parent((f, p, q, o, s) := p

c_children((f, p, q, o, s) := q

c_order((f, p, q, o, s) := o

c_start((f, p, q, o, s) := s

Kapitel höchster Hierarchiestufe besitzen kein übergeordnetes Kapitel, d.h. für sie gilt: c_parent(c) = *nil*.

Die **Binnenstruktur eines Kapitels** besteht aus einer Menge von sequentiell geordneten Seiten. Jede Seite eines Kapitels besteht aus einem Knoten der Seitenstruktur sowie Verweisen auf die jeweils vorherige und nachfolgende Seite der Seitenstruktur:

$$FL^{\#} \subseteq FT^{\#} \times FT^{\#} \times FT^{\#} \quad (7.33)$$

Da die zugrundeliegende Seitenstruktur bereits hierarchisch aufgebaut ist, besteht der Übergang zur Kapitelstruktur aus drei relativ einfachen Schritten:

1. Zusammenfassen von benachbarten Seiten zu Kapiteln und Unterkapiteln.
2. Ordnen von Unterkapiteln der gleichen Hierarchiestufe unter Berücksichtigung der bei der Seitenstrukturierung angefallenen Ordering Informationen.
3. Einbettung der an Seiten, Kapitel und Blöcke gebundenen Medienobjekte in Form von zusätzlichen Seiten oder Kapiteln.

Der Algorithmus "CreateChapters" zur Kapitelstrukturierung liefert neben der endgültigen Kapitelstruktur C^{b*} noch die Menge aller nicht in die Kapitelstruktur eingebundenen Seiten - z.B. bei der Seitenstrukturierung ausgelagerte Wurzelseiten - zurück:

Algorithmus CreateChapters

Aufgabe:	Ableiten einer Kapitelstruktur aus einer gegebenen Seitenstruktur unter Berücksichtigung vorgegebener Ordering Informationen
Eingabe:	Seitenstruktur F^{b*}
	Ordering Informationen O^{b*}
Ausgabe:	Aus der Seitenstruktur F^{b*} angeleitete Kapitelstruktur C^{b*}
	Nur über Hyperlinks eingebundene Seiten F^{free}

Zusammenfassen von benachbarten Seiten zu Kapiteln

Ordnen der Unterkapitel anhand der vorgegebenen Ordering Informationen

Einbetten der an Seiten, Kapitel oder Blöcke gebundenen Zusatzobjekte

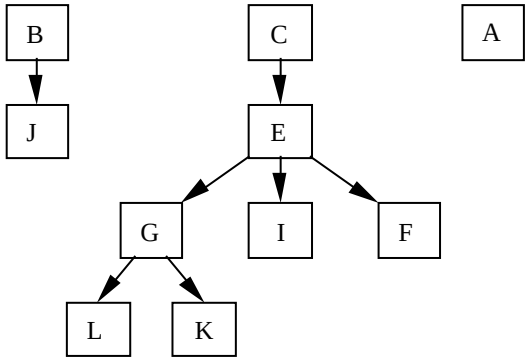
Zusammenfassen der nur über Hyperlinks angebundenen Seiten zu der Menge F^{free}

Die ersten beiden Schritte der Kapitelstrukturierung werden im Folgenden ausführlich beschrieben, Schritt 3 - die Einbindung von Zusatzobjekten - in Grundzügen skizziert. Als Ausgangspunkt dient dabei die im vorherigen Kapitel beispielhaft erstellte Seitenstruktur.

7.7.1 Initialisierung der Kapitelstruktur

Abbildung 7-24 zeigt eine auf das wesentliche reduzierte Darstellung der im vorherigen Kapitel erstellten Seitenstruktur. Die einzelnen Seiten sind als Quadrate eingezeichnet und mit Großbuchstaben gekennzeichnet. Pfeile zwischen den Seiten geben die gerichteten Kanten der Seitenstruktur wieder. Die Inhalte der Seiten und die durch Ordering Informationen beschriebenen impliziten Ordnungen sind in der Abbildung nicht dargestellt.

Abbildung 7-24: Vereinfachte Seitenstruktur aus Kapitel 6.6



Die einzelnen Kapitel der Kapitelstruktur ergeben sich direkt aus der Seitenstruktur:

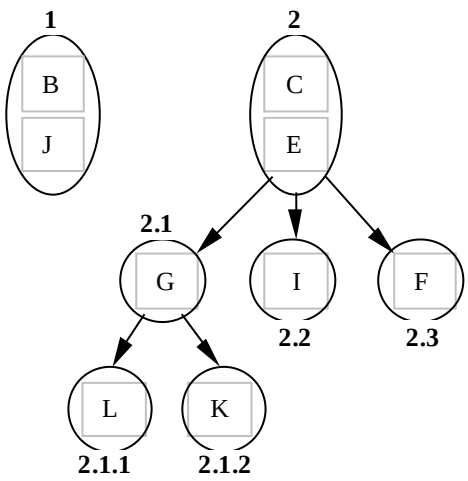
- Jede eingebundene Wurzelseite bildet die Anfangsseite eines Kapitel höchster Hierarchiestufe. Im Beispiel sind dies die Seiten B und C.
- Seiten mit mehr als einem Nachfolger bilden die letzten Seiten eines Kapitels bzw. Unterkapitels. Im Beispiel sind dies die Seiten E und G. Ihre Nachfolger sind die Anfangsseiten eines neuen Unterkapitels der nächstniedrigeren Hierarchiestufe.
- Sequenzen von Seiten werden zusammengefaßt. Im Beispiel betrifft dies die Seiten B und J, sowie C und E.

Die Umsetzung dieser einfachen Regeln geschieht durch eine rekursive, von den Wurzelseiten ausgehende Tiefensuche auf der Seitenstruktur. Hierbei wird die von der Startseite ausgehende Sequenz von Seiten zu einem Kapitel zusammengefaßt. Sobald der Algorithmus auf eine Verzweigung trifft, wird ein Unterkapitel angelegt und die Teifensuche rekursiv zu dessen Berechnung aufgerufen.

Unter der Annahme, daß die Erstellung der Kapitel gleicher Hierarchiestufe von links nach rechts erfolgt (maßgeblich ist die Darstellung in Abbildung 7-24), so werden die einzelnen Kapitel in folgender Reihenfolge (*Pre-Order*) angelegt: 1 2 2.1 2.1.1 2.1.2 2.2 2.3

Hieraus ergibt sich die in Abbildung 7-25 dargestellte initiale Kapitelstruktur[41].

Abbildung 7-25: Initiale Kapitelstruktur



Wie aus der Abbildung zu ersehen ist, stellt die initiale Kapitelstruktur lediglich eine strukturerhaltende Transformation der Seitenstruktur dar. Alle Knoten und Kanten der Seitenstruktur bleiben erhalten,

[41] Die isolierte Seite A (siehe Abbildung 7-25) wird bei den folgenden Betrachtungen ignoriert, da sie lediglich über einen Hyperlink angebunden und somit für die Erstellung der Kapitelstruktur irrelevant ist.

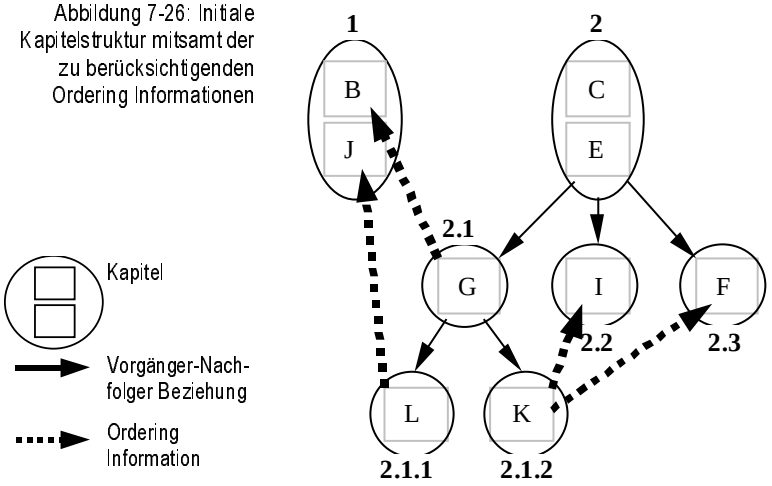
entweder explizit als Teil eines Kapitels (Seiten, Beziehungen zwischen Seiten) oder implizit durch die Kapitelhierarchie.

7.7.2 Sortieren der Unterkapitel

Die Reihenfolge der Kapitel sowie der Unterkapitel einer Hierarchiestufe ist innerhalb der initialen Seitenstruktur rein willkürlich festgelegt, da die Seitenstruktur keine Ordnung auf Ausgangskanten definiert. Im zweiten Schritt der Kapitelstrukturierung muß daher eine **Ordnung auf Kapitel und Unterkapiteln** berechnet werden, die nach Möglichkeit alle Abhängigkeiten zwischen Seiten berücksichtigt. Diese Abhängigkeiten sind in den Ordering Informationen der Seitenstrukturierung kodiert.

Abbildung 7-26 zeigt die Kapitelstruktur aus Abbildung 7-25, wobei jedoch die durch Ordering Informationen beschriebenen Abhängigkeiten als gestrichelte Pfeile eingezeichnet sind.

Abbildung 7-26: Initiale Kapitelstruktur mitsamt der zu berücksichtigenden Ordering Informationen



Ordering Informationen schreiben eine nach Möglichkeit einzuhaltende Seitenreihenfolge vor. Um diese Vorgabe für die Sortierung der Kapitel nutzbar zu machen, muß in einem ersten Schritt berechnet werden, welche der vorhandenen Ordering Informationen überhaupt für die Sortierung der Unterkapitel eines Kapitels relevant sind. Relevant für die Ordnung von Kapiteln bzw. Unterkapiteln c_1, c_2, .., c_n sind dabei alle Ordering Informationen die von einer Seite des Kapitels c_x oder eines seiner Unterkapitel auf eine Seite eines anderen Kapitels c_y bzw. eines seiner Unterkapitel verweisen.

Um die relevanten Ordering Informationen zu ermitteln, muß daher bekannt sein, welche Seiten überhaupt zu einem Kapitel mitsamt seiner Unterkapitel gehören. Hierzu wird eine Abbildung $\chi_{frames}(c)$ definiert, die rekursiv alle Seiten eines Kapitels beschreibt:

$$\chi_{frames}: C^{\#} \rightarrow \mathbf{P}(FT^{\#})$$

$$\chi_{frames}(c) = \left(\bigcup_{fl \in c\_frames(c)} fl\_frame(fl) \right) \cup \left(\bigcup_{d \in c\_children(c)} \chi_{frames}(d) \right) \qquad (7.34)$$

Die relevanten Ordering Informationen für die Sortierung der Kapitel 1 und 2 der Beispielstruktur sind die Abhängigkeiten B ← G und J ← L, während für die Sortierung der Unterkapitel 2.1, 2.2 und 2.3 die Ordering Informationen I ← K und F ← K berücksichtigt werden müssen. Für die Kapitel 2.1.1 und 2.1.2 existieren keine relevanten Ordering Informationen, so daß diese beiden Kapitel in einer beliebigen Reihenfolge angeordnet werden können.

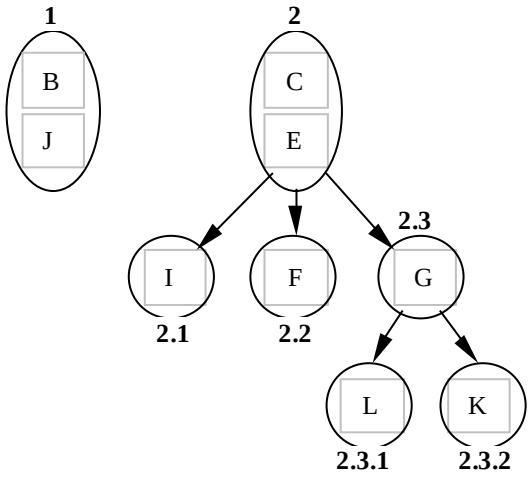
Der Algorithmus zur Umsortierung der Kapitel berücksichtigt zuerst die aus abgesicherten und anschließend die aus nicht abgesicherten Hyperlinks resultierenden Ordering Informationen. Ordering Informationen werden nicht auf ihre Konsistenz hin überprüft, d.h. jede durch eine Ordering Information

beschriebene Umsortierung wird durchgeführt, auch wenn eine widersprüchliche Ordering Information existiert. Hierdurch ist sichergestellt, daß in der endgültigen Ordnung die zwingenden Abhängigkeiten stärker berücksichtigt sind, als die aus abgesicherten Hyperlinks entstandenen, optionalen Vorgaben.

Um für eine Menge von Kapiteln, die den gleichen Vorgänger besitzten, eine Ordnung bestimmen zu können, müssen zuerst die für diese Kapitel relevanten Ordering Informationen ermittelt werden. Dies geschieht durch einen Vergleich der in den Ordering Informationen kodierten Seiten mit den Seiten der einzelnen Kapitel (inklusive ihrer Unterkapitel). Abschließend wird die durch den paarweisen Vergleich ermittelte Reihenfolge der Unterkapitel durch eine Umsortierung der Kapitel hergestellt.

Für das Beispiel ergibt sich nach der Kapitelsortierung die in der nachfolgenden Abbildung dargestellte Kapitelstruktur

Abbildung 7-27: Geordnete Kapitelstruktur



8 Implementierung

Die aktuelle Implementierung des *i4 Frameworks* - der **Medienobjekt-Manager** - besteht im wesentlichen aus einem Medienobjektspeicher, einer interaktiven Benutzeroberfläche zur Verwaltung von Medienobjekten und Dokumenten, sowie Implementierungen von Auswahl- und Strukturierungsalgorithmen. Darüber hinaus enthält er einen Java Interpreter, über den eine themenspezifische Konfiguration der Auswahl- und Strukturierungsalgorithmen möglich ist.

In diesem Kapitel soll ein kurzer Überblick über die Implementierung des Medienobjekt-Managers gegeben werden, wobei neben der Umsetzung des Medienobjektspeichers vor allem die Benutzeroberfläche und die dynamische Konfiguration der Algorithmen im Mittelpunkt steht.

8.1 Benutzeroberfläche

Der Medienobjekt-Manager ist ein interaktives Werkzeug für die *Bottom-Up* Erstellung von Lehr- und Informationssystemen. Er unterstützt den Autor bei

- der Erstellung und Verwaltung von Medienobjekten und Indizes
- der Erstellung und Konfiguration von Dokumenten und Blöcken
- der Konfiguration der Auswahl- und Strukturierungsalgorithmen

Der Medienobjekt-Manager wurde komplett in *VisualBasic* 6.0 implementiert, einer insbesondere für die Erstellung interaktiver Benutzeroberflächen gut geeigneten Programmiersprache.

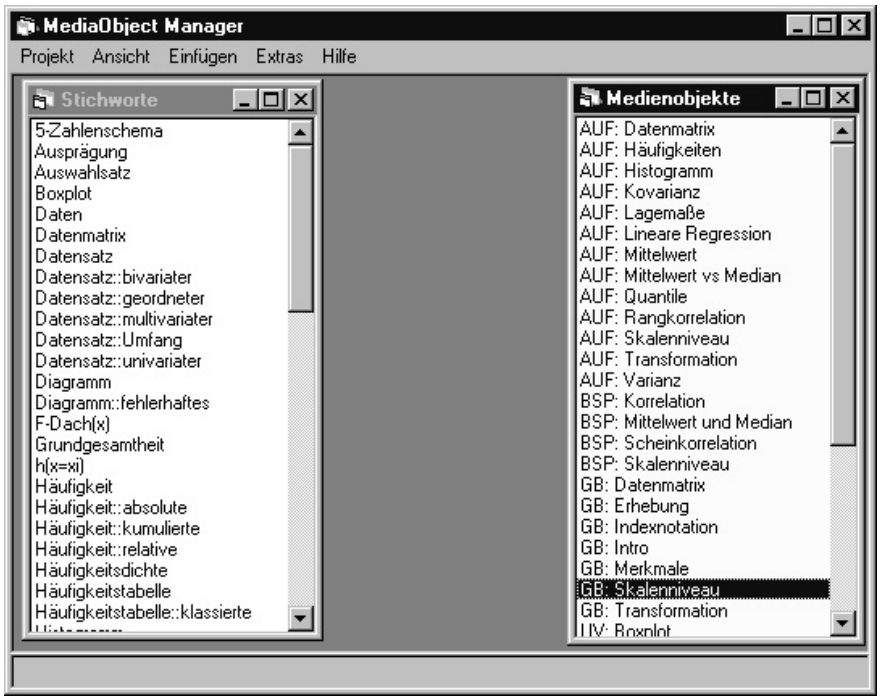
Die Generierung von Lehr- und Informationssystemen wird vom Medienobjekt-Manager über sog. **Projekte** realisiert. Zu einem Projekt gehört dabei immer eine Datenbank - der Medienobjektspeicher - und eine oder mehrere Dokumentendefinitionen. Über ein Projekt können somit zu jedem Thema auf den dazu vorhandenen Medienobjekten verschiedene Lehr- oder Informationssysteme definiert werden.

8.1.1 Verwaltung von Medienobjekten und Indizes

Der wichtigste Bestandteil eines Projektes ist der zugehörige **Medienobjektspeicher**, in dem alle für die Dokumentenerzeugung verfügbaren Medienobjekte, Stichworte und Indizes abgelegt sind.

Die nachfolgende Abbildung zeigt die Benutzeroberfläche des Medienobjekt-Managers nach dem Öffnen eines Projektes zum Thema "Deskriptive Statistik".

Abbildung 8-1: Dialoge zur Verwaltung von Medienobjekten und Stichworten



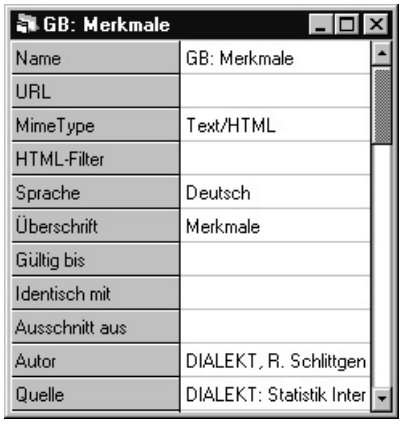
Das Fenster auf der rechten Seite des Bildschirms zeigt die Namen aller im Medienobjektspeicher abgelegten Medienobjekte an. Im Fenster auf der linken Seite sind alle in diesen Medienobjekten vorkommenden Stichworte in alphabetischer Reihenfolge aufgeführt.

Über diverse Menüs hat der Autor die Möglichkeit,

- neue Medienobjekte zu erstellen,
- Attribute vorhandener Medienobjekte zu bearbeiten und
- Indizes zu erstellen und zu modifizieren.

Wie in Kapitel 4 erwähnt, werden die Eigenschaften von Medienobjekten - mit Ausnahme ihres Index - über Name-Wert-Paare festgelegt. Durch Auswahl einer entsprechenden Option im Kontextmenü eines Medienobjektes können die Name-Wert-Paare dieses Objektes vom Autor erstellt und modifiziert werden (Abbildung 8-2).

Abbildung 8-2: Dialog zur Anzeige und Bearbeitung der Eigenschaften eines Medienobjekts (Ausschnitt)



Für viele Attributwerte stehen anstelle von Eingabefeldern **Auswahllisten** zur Verfügung, die entweder statisch festgelegt sind oder dynamisch gefüllt werden. Ein Beispiel für die Verwendung dynamischer Auswahllisten ist das Attribut "Autor": Beim Anzeigen des Konfigurationsdialogs werden dynamisch alle verschiedenen Autoren aller vorhandenen Medienobjekte zu einer Auswahlliste zusammengefaßt.

Ähnlich komfortabel ist die Erstellung der **Indizes**. Durch die in Abschnitt 8.3 beschriebene automatische Unterstützung sowie Möglichkeiten der direkten Manipulation (*drag & drop*) kann die Erstellung eines Index mit wenigen Mausklicks durchgeführt werden:

Ist ein referenziertes Stichwort bereits in einem anderen Medienobjekt enthalten, kann es durch *drag & drop* (Ziehen mit der Maus) aus der Liste aller Stichworte in den Index eingefügt werden. Die Festlegung des Grades an verlangtem und vermitteltem Wissen sowie der Relevanz geschieht anschließend über einfache Mausklicks (Abbildung 8-3).

Abbildung 8-3: Index eines Medienobjekts zum Boxplot. Im oberen Teil des Fensters sind die Stichworte der Indexeinträge des Medienobjekts aufgelistet, im unteren Teil erfolgt die Konfiguration des ausgewählten Eintrags.

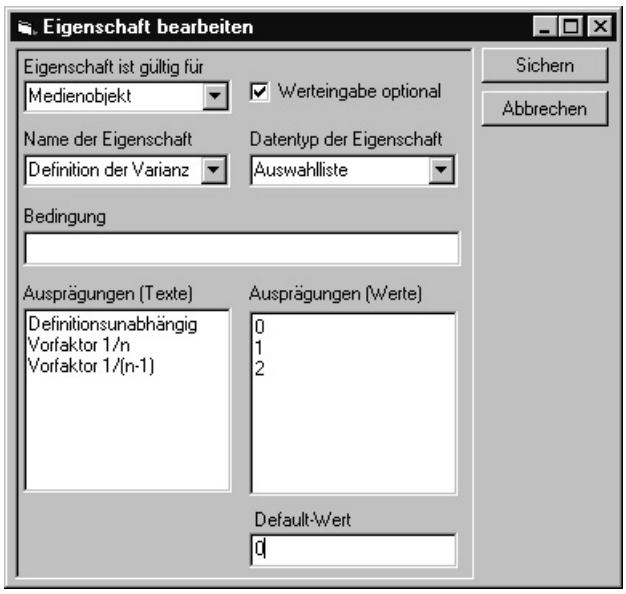
8.1.2 Erstellung anwendungsabhängiger Attribute

Jedem Medienobjekt, Stichwort und Indexeintrag ist eine Menge von Name-Wert-Paaren zugeordnet. Die Wertemengen dieser Attribute können für jedes Objekt individuell über Dialoge festgelegt werden.

Welche **Attribute** für ein Objekt verfügbar sind, hängt vom Typ des Objekts ab. Beispiele für Medienobjekt-Attribute sind *Mime-Type*, Sprache und Genre; Beispiele für Stichwort-Attribute sind das übergeordnete Stichwort und das *Default*-Vorwissen.

Um die in Kapitel 5 beschriebene doppelte Adaptierbarkeit zu ermöglichen, muß der Autor in der Lage sein, zu den vorhandenen Attributen weitere, **anwendungsabhängige Attribute** hinzuzufügen. Zur Erstellung einer Zeitleiste wird z.B. ein zusätzliches Attribut "Datum" benötigt, um Medienobjekte mit einem Datum verknüpfen zu können.

Abbildung 8-4: Definition einer neuen Medienobjekt-Eigenschaft "Definition der Varianz".



Anwendungsabhängige Attribute können mit Hilfe des Medienobjekt-Managers erstellt und in den Medienobjektspeicher integriert werden. Darüber hinaus können auch die Wertebereiche vorhandener, anwendungsunabhängiger Attribute erweitert oder eingeschränkt werden.

Abbildung 8-4 zeigt den für Erstellung anwendungsabhängiger Attribute zuständigen Attribut-Dialog.

Im oberen Teil des Dialogs kann der Autor den Namen des neuen Attributs festlegen, sowie die Objekttypen für die es gültig ist. Der Datentyp des Attributs gibt seinen Wertebereich an.

In der Zeile "Bedingung" kann ein optionaler logischer Ausdruck angegeben werden, von dessen Wert abhängt, ob das Attribut im Konfigurationsdialog angezeigt wird oder nicht. Da dieser Ausdruck nach jeder Konfigurationsänderung ausgewertet wird, sind über Bedingungen auch Abhängigkeiten zwischen Attributen darstellbar. Ist z.B. ein Attribut "Farbtiefe" nur für Grafiken und Videos gültig, so kann dies über die Bedingung `(MimeType contains "image") or (MimeType contains "text")` ausgedrückt werden.

8.1.3 Dokumente und Blöcke

Die **Konfiguration von Dokumenten und Blöcken** geschieht normalerweise über die Ableitung von *Java Wrapper Klassen* (siehe Kapitel 8.3). Da dieses Verfahren für den Autor sehr arbeitsaufwendig und unkomfortabel sein kann, erlaubt der Medienobjekt Manager die Konfiguration des zu erstellenden Lehr- oder Informationssystems mit Hilfe einer grafischen Benutzeroberfläche.

Abbildung 8-5: Zusammenstellung eines Lehr- oder Informationssystems aus drei Blöcken. Die Konfiguration der Blöcke erfolgt durch die Schaltflächen am unteren Rand des Fensters.

Jedes Lehr- oder Informationssystem wird aus einer Menge von Blöcken zusammengestellt. Lernziele, Basisobjekte und Zusatzobjekte eines Blocks können über einen Dialog sehr einfach erstellt und verwaltet werden (Abbildung 8-6).

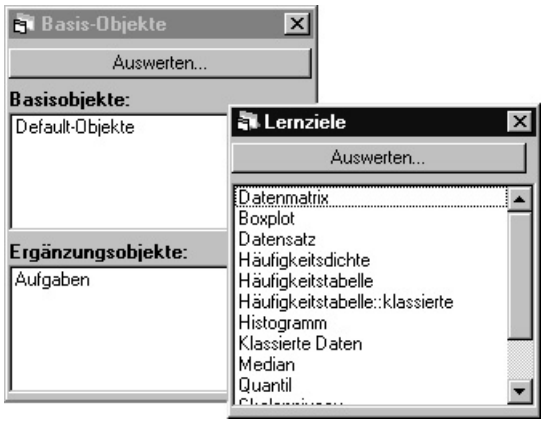


Abbildung 8-6: Festlegung der Basis- und Zusatzobjekte sowie Lernziele eines Blocks.

Für Lernziele und Basisobjekten können neben einzelnen Stichworten bzw. Medienobjekten auch sog. **Schablonen** angegeben werden. Schablonen sind Stichworte oder Medienobjekte, deren Attributwerte über *Wildcards* und logische Ausdrücke angegeben werden können. Eine Schablone repräsentiert alle Objekte, die dem Muster der Schablone entsprechen.

Eine Medienobjekt-Schablone, die für alle Attribute den Wert "*" (= beliebiger Wert) besitzt, repräsentiert somit alle verfügbaren Medienobjekte. Sind alle Werte der Schablone "*", mit Ausnahme der Bandbreite, für die der Ausdruck "< 50000" angegeben ist, so repräsentiert die Schablone alle Medienobjekte, deren Bandbreitenanforderung kleiner als 50 KByte/sec ist.

Schablonen auf Medienobjekten und Stichworten können mit dem Medienobjekt-Manager sehr einfach mit Hilfe des Konfigurationsdialogs des zugrundeliegenden Objekttyps erstellt werden (Abbildung 8-7).

Abbildung 8-7: Schablone zur Auswahl von Medienobjekten anhand von Eigenschaftswerten

In Abweichung von der in Kapitel 5 gegebenen Definition werden Zusatzobjekte im Medienobjekt-Manager ebenfalls über Schablonen beschrieben.

8.1.4 Automatisierter Import von Medienobjekten

Zur Unterstützung von Autoren beim Import von mehreren Medienobjekten aus einer Quelle besitzt der Medienobjekt-Manager eine spezielle Import-Schnittstelle, die nicht nur die Eingabe vieler Metadaten vereinfacht sondern auch über eine Volltextanalyse eine schnelle Initialisierung der Stichwortmenge durchführt.

Abbildung 8-8 zeigt den Importdialog, der als Schablone für die allen Medienobjekten gemeinsamen Metadaten dient. Für Attribute, die für alle zu importierenden Medienobjekte den gleichen Wert besitzen sollen, wird dieser Wert einfach in die entsprechende Zeile des Dialogs eingegeben. Einzige Ausnahme bildet das Feld „Name“, in dem ein Präfix für aus einer generierten ID besehende Default-Namen angegeben werden kann. Für Attribute mit Werten, die von der konkreten Ausprägung des Medienobjekts abhängig sind, werden die entsprechenden Felder freigelassen.

Im Eingabefeld im oberen Teil des Dialogs wird angegeben, welche Dateien importiert werden sollen. Dies geschieht über eine Schablone, die aus einem Pfad und einem Wildcard-Zeichen („*“ bzw. „?“) enthaltenden Dateinamen besteht. Im Beispiel sollen so z.B. alle Dateien mit der Endung „.asc“ als HTML-kodierte Medienobjekte importiert werden, die in einem bestimmten Verzeichnis abelegt sind.

Abbildung 8-8: Importfilter des Medienobjekt-Managers

Importierte Textdateien werden als Datenfelder im Medienobjektspeicher abgelegt. Hierbei werden automatisch die Pfade aller über das HTML „img“-Tag referenzierte Grafiken in absolute Pfade umgewandelt. Beim Import von Grafiken und Animationen verbleiben diese als Dateien im Dateisystem. Es wird lediglich der Pfad der Grafkdatei in das URL-Attribut des zugehörigen Metadatensatzes eingetragen.

Beim Import textueller Medienobjekte wird parallel ein Volltextindex aller in den Texten vorkommenden Substantive erstellt. Häufig vorkommende, themenunabhängige Worte werden über eine Stopwort-Liste herausgefiltert. Die verbleibenden Substantive werden auf ihre Wortstämme reduziert um verschiedene Deklinationen ein und desselben Wortes zu einem Stichwort zusammenzufassen.

Nach dem Einlesen der Medienobjekte in den Medienobjektspeicher wird dem Benutzer eine Liste aller identifizierten Wortstämme angezeigt., die im Idealfall charakteristisch für das behandelte Themengebiet sind. Aus diesen Wortstämmen kann nun eine initiale Menge von Stichworten[42] ausgewählt werden (Abbildung 8-9). Zu jedem ausgewählten Stichwort wird dabei die kürzeste in einem der importierten Texte vorkommende Form des Wortes in die Stichwort-Tabelle des Medienobjektspeichers aufgenommen. Zusätzlich wird der Stamm des Stichworts im zugehörigen "query"-Feld abgelegt.

[42] Durch diese Vorgehensweise wird die zweite der in Kapitel 4.4.4 genannten Integritätsbedingungen verletzt, da die initiale Stichwortmenge S^{avail} (ausschließlich) nicht mit Medienobjekten verknüpfte Stichworte enthält. Diese Verletzung der Integrität kann jedoch toleriert werden, da sie keinerlei Auswirkungen auf die Funktionsweise der in Kapitel 7 beschriebenen Auswahl- und Strukturierungsalgorithmen hat.

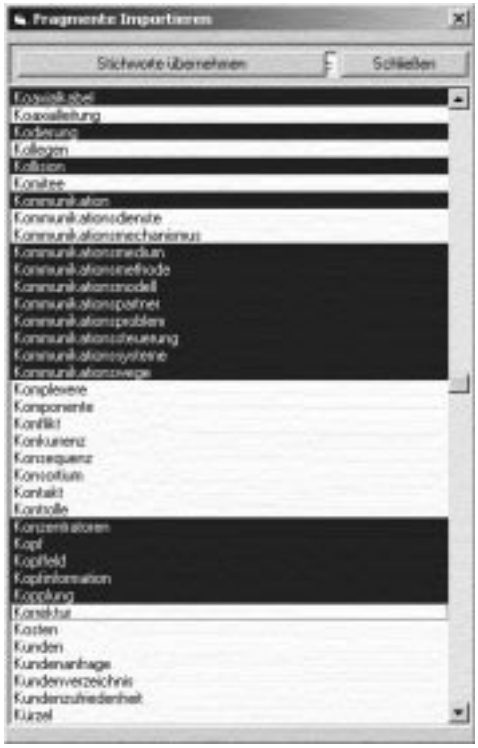


Abbildung 8-9: Auswahl der initial in die Stichwort-Tabelle aufzunehmenden Stichworte

8.1.5 Der Index Wizard

Der bei weitem komplexeste und zeitaufwendigste Teil der Erstellung von Medienobjekten ist deren Indizierung. In Anbetracht der Tatsache, daß ein Lehr- oder Informationssystem potentiell aus hunderten von Medienobjekten besteht, kann die Dokumentenerstellung durch eine Unterstützung des Autors bei der Indizierung erheblich effektiver und schneller durchgeführt werden.

Die **Indizierung eines Medienobjektes** läßt sich in drei aufeinander aufbauende Teilprobleme unterteilen:

1. Ermittlung der von einem Medienobjekt referenzierten Stichworte und Themen,
2. Aufteilung dieser Stichworte in verlangtes Vorwissen und vermittelte Informationen,
3. Festlegung der konkreten Maßzahlen für Vorwissen, vermitteltes Wissen und Relevanz.

Keines dieser Teilprobleme kann ohne Verwendung von aufwendigen IR und KI Methoden bewältigt werden. Um dem Autor dennoch eine möglichst gute Unterstützung zu bieten, enthält der Medienobjekt-Manager einen sog. **Index Wizard**, der zumindest einige einfache, in diesem Kapitel beschriebene, IR Algorithmen zur automatisierten Indizierung von textuellen Medienobjekten zur Anwendung bringt.

Das Ergebnis der automatisierten Indizierung ist nur in sehr seltenen Fällen so gut, daß es ohne manuelle Änderung durch den Autor in den Medienobjektspeicher aufgenommen werden könnte. Dennoch reduzieren bereits die implementierten, einfachen Algorithmen den manuellen Zeitaufwand zur Indizierung eines Medienobjektes auf weniger als die Hälfte der für eine komplett manuelle Indizierung zu veranschlagenden Zeit. Dies gilt jedoch wie erwähnt nur für rein textuelle Medienobjekte; die Indizierung von Grafiken und Videos muß komplett manuell durchgeführt werden.

Das erste Teilproblem der Indizierung - das Auffinden der in einem Medienobjekt referenzierten Stichworte - läßt sich zumindest näherungsweise durch eine Volltextsuche lösen. Die Idee hierbei ist, den kompletten Text des Medienobjekts nach den in der Sichworttabelle enthaltenen Stichworten zu durchsuchen und diese in den Index aufzunehmen.

Um die in einem Medienobjekt referenzierten Stichworte zu ermitteln, erstellt der *Index Wizard* einen Volltextindex des Medienobjekts und vergleicht diesen mit den Einträgen in den query-Feldern der Stichwort-Tabelle:

Aus dem Text des Medienobjekts wird eine Liste der vorkommenden Worte samt Häufigkeit ihres Vorkommens erstellt. Alle Worte werden auf eine dem Wortstamm vergleichbare, eindeutige Repräsentation reduziert [Caumanns99d] und Stop-Worte (Artikel, etc.) entfernt.

Anschließend geht der *Index Wizard* alle in der Stichwort Tabelle abgelegten Stichworte durch und überprüft, ob der im "query"-Eintrag des Stichworts vermerkte Stamm in der Wortliste enthalten ist. Ist dies der Fall, wird das Stichwort als vom Medienobjekt referenziert vermerkt. Zusätzlich werden der Anker und die Relevanz des Stichworts innerhalb des Medienobjekts bestimmt[43].

Der einzige Nachteil dieses auf der Indizierung von Substantiven basierenden Verfahrens ist, daß Unterstichworte selten und aus mehreren Phrasen bestehende Stichworte so gut wie nie in der Liste der relevanten Stichworte enthalten sind und daher auch nicht automatisch erkannt werden.

Die Durchführung aller Suchanfragen aller Stichworte ist sehr aufwendig, in den meisten Fällen aber dennoch erheblich schneller als ein manuelles Eintippen oder Heraussuchen aller in einem Medienobjekt referenzierten Stichworte.

Nachdem die von einem Medienobjekt referenzierten Stichworte ermittelt sind, gilt es, diese in Vorwissen und vermitteltes Wissen zu unterteilen und die jeweiligen Werte festzulegen.

Da der *Index Wizard* keinerlei semantische Analyse durchführt, sondern allein auf statistischen Verfahren basiert, wird die Unterscheidung in Vorwissen und vermitteltes Wissen implizit durch die Berechnung der entsprechenden Attribute "verlangtes Wissen" und "vermitteltes Wissen" durchgeführt.

Die Festlegung ungefährer Werte für verlangtes und vermitteltes Wissen basiert auf Vergleichen mit bereits indizierten Medienobjekten. Die Idee hierbei ist, daß eine relativ statische semantische Abhängigkeit zwischen Stichworten existiert, die in den Indizes zum Ausdruck kommt. D.h. wenn zwei Stichworte s und t in einem Index als verlangt (s) bzw. vermittelt (t) deklariert sind, so wird angenommen, daß dieses Verhältnis auch für alle anderen Medienobjekte gilt, in denen s und t gemeinsam auftreten.

Der *Index Wizard* führt daher für jedes in einem Medienobjekt gefundene Stichwort einen Vergleich des betreffenden Medienobjekts mit allen dieses Stichwort ebenfalls referenzierenden Medienobjekte durch. Hierbei wird jeweils ermittelt, wieviele Überschneidungen der referenzierten Stichworte existieren und deren (quadrierte) Anzahl als Gewichtung für die Ermittlung der gesuchten Werte verwendet:

$$req(i) = \sum_{j=\sigma_{ndx}(kw(i),I)} req(j) * |\mu_{ndx}(mo(i),I) \cap \mu_{ndx}(mo(j),I)|^2$$

$$prv(i) = \sum_{j=\sigma_{ndx}(kw(i),I)} prv(j) * |\mu_{ndx}(mo(i),I) \cap \mu_{ndx}(mo(j),I)|^2$$

Hierzu ein Beispiel:

Das zu indizierende Medienobjekt m referenziert 4 Stichworte "a", "b", "c" und "d". Das Stichwort "a" wird ebenfalls vom Medienobjekt n mit den Werten 20 (Vorwissen) und 60 (vermitteltes Wissen) und vom Medienobjekt o mit den Werten 0 und 40 referenziert. Medienobjekt n enthält neben "a" auch die Stichworte "b" und "c", während o von den in m referenzierten Stichworten nur "a" enthält.

Aus diesen Angabe berechnet der *Index Wizard* die folgenden Werte für verlangtes und vermitteltes Wissen des Stichworts "a":

verlangtes Wissen = (20 * 9 + 0 * 1) / (9 + 1) = 18

vermitteltes Wissen = (60 * 9 + 40 * 1) / (9 + 1) = 58

[43] Die Bestimmung der Relevanz geschieht allein anhand der Frequenz und der Positionen des Vorkommens eines Stichwortes. Anhand im Rahmen des *Index Wizard* nicht verifizierter linguistischer oder statistischer Verfahren [Hearst+93, Salton+96], die zum Teil jedoch zusätzliche kontextuelle Informationen benötigen, erscheint eine Verbesserung der momentan manchmal sehr willkürlich erscheinenden Relevanzwerte möglich.

Das angegebene Verfahren liefert um so bessere Werte, je mehr Medienobjekte bereits indiziert sind. In der Praxis hat sich jedoch vor allem die Annahme einer statischen semantischen Stichwort-Struktur als Illusion herausgestellt. Aus Medienobjekten abgeleitete semantische Netze enthalten fast immer Zyklen und nur selten läßt sich zu einem Stichwort die Menge der zugrundeliegenden Stichworte eindeutig benennen. Dies führt dazu, daß die angegebene Formel für lediglich 50 - 70% aller Stichworte die Werte für verlangtes und vermitteltes Wissen - innerhalb eines akzeptablen Tolerenzbereichs - richtig berechnet.

8.2 Tabellen und Relationen

Alle zu einem Themengebiet verfügbaren Medienobjekte werden mitsamt ihrer Indizes und Stichworte in einer Datenbank - dem **Medienobjektspeicher** - verwaltet. Schablonen, Dokumente und Blöcke bilden den Inhalt einer weiteren Datenbank, des **Dokumentenspeichers**. Während jedoch der Medienobjekt-speicher autonom ist, ist ein Dokumentenspeicher immer an einen Medienobjektspeicher gebunden. Alle Referenzen auf Medienobjekte und Stichworte, die innerhalb von im Dokumentenspeicher abgelegten Objekten enthalten sind, beziehen sich auf einen konkreten Medienobjektspeicher.

Durch die Aufteilung der zu speichernden Objekte auf zwei Datenbanken können Medienobjekte und Dokumente weitgehend unabhängig voneinander von verschiedenen Anbietern erstellt und verwaltet werden. Diese Unabhängigkeit wird durch die Verwendung von Schablonen noch verstärkt, da hierdurch im Dokumentenspeicher als Basis- oder Zusatzobjekte benötigte Medienobjekte nicht über eine ID oder einen Namen, sondern über Attributwerte identifiziert und referenziert werden. Hierdurch werden alle Änderungen des Medienobjektspeichers implizit für die Dokumentenerzeugung sichtbar.

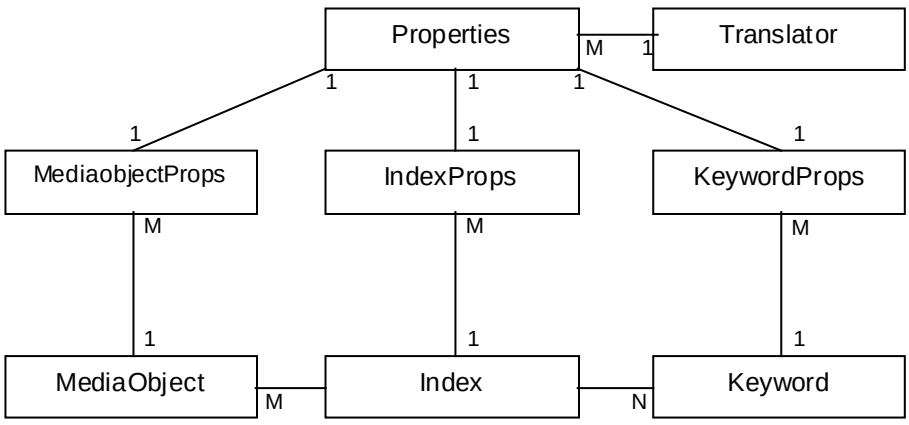
Ein denkbares Szenario wäre z.B., daß eine Universität oder ein Wirtschaftsforschungsinstitut einen Medienobjektspeicher zum Thema "Statistik" anbietet, in dem Folien, Skripte, Aufgaben und Datensätze enthalten sind. Auf diesem Medienobjektspeicher können andere Universitäten, aber auch Schulen und Unternehmen, vollkommen autonom und sicher für ihren Bedarf angepaßte Lehr- und Informationssysteme definieren. Alle Aktualisierungen und Erweiterungen der vorhandenen Medienobjekte werden implizit bei der dynamischen Erzeugung neuer Dokumente berücksichtigt.

8.2.1 Der Medienobjektspeicher

Die aktuelle Implementierung des Medienobjekt-Managers kann Medienobjektspeicher ausschließlich in Form von *Microsoft Access* Datenbanken erstellen und verwalten, die einem festen Schema entsprechen.

Die nachfolgende Abbildung stellt den für die Implementierung des Medienobjekt-Managers gewählten Entwurf des Medienobjektspeichers dar. Sämtliche Attribute werden dabei in eigenen Tabellen abgelegt und nicht als Felder der Objekttabellen modelliert, um möglichst flexible Erweiterungen der Attributmengen der einzelnen Objekttypen zuzulassen.

Abbildung 8-10: Medienobjektspeicher



Medienobjekte und Stichworte werden jeweils in einer eigenen Tabelle abgelegt und über Indizes miteinander verknüpft (siehe auch Abbildung 2-4). Attribute von Medienobjekten, Stichworten und Indexeinträgen sind in jeweils einem eigenen Attributschema abgelegt, wobei Namen und Datentypen der Attribute nicht statisch sind, sondern vielmehr durch Meta-Informationen ("Properties") beschrieben werden.

Der Medienobjektspeicher besteht somit aus 8 Tabellen, von denen 3 zur Speicherung der Medienobjekte, Stichworte und Indizes und 5 zur Verwaltung von Attributen dienen:

Mediaobject(mo_id, name, source, object): Medienobjekte, bestehend aus einem eindeutigen Schlüssel (mo_id), einem Namen (name) und der URL, unter der sich das Objekt befindet (source). Im Medienobjektspeicher befindliche Objekte sind als Speicher-Image im Feld *object* abgelegt.

MediaobjectProps(mo_id, attribute, value): Eigenschaften von Medienobjekten in Form von Name-Wert-Paaren.

Keyword(kw_id, baseId, name, query): Stichworte, bestehend aus einem eindeutigen Schlüssel, dem Schlüssel des übergeordneten Stichworts und dem Stichwort selbst. Das letzte Feld "query" enthält Informationen zur automatisierten Indizierung (siehe Kapitel 8.3).

KeywordProps(kw_id, attribute, value): Eigenschaften von Stichworten in Form von Name-Wert-Paaren.

Index(mo_id, kw_id, required, provided, relevance): Indexeinträge, bestehend aus den verknüpften Objekten und den Eigenschaften dieser Verknüpfung (Vorwissen, vermitteltes Wissen, Relevanz).

IndexProps(mo_id, kw_id, attribute, value): Eigenschaften von Indexeinträgen in Form von Name-Wert-Paaren.

Die aktuelle Implementierung des Medienobjektspeichers unterscheidet nicht zwischen allgemeinen und anwendungsabhängigen Attributen. Sämtliche Attribute aller Klassen werden in einer Tabelle "Properties" beschrieben und zur Laufzeit mit den die konkreten Werte enthaltenden Name-Wert-Paaren verknüpft:

Properties(objectType, name, dataType, default, condition, texts, values, flags)

Die Felder von "Properties" entsprechen den in Abschnitt 8.1.2 beschriebenen Feldern des Dialogs zur Erstellung anwendungsabhängiger Attribute. Das zusätzliche (boolesche) Feld "flags" gibt an, ob die Eigenschaft vom Autor modifiziert werden darf oder nicht.

8.2.2 Dokumentenspeicher

Die Tabellen des Dokumentenspeichers beinhalten neben den in Kapitel 5 beschriebenen Nutzungsszenarien auch alle auf Medienobjekten und Stichworten definierten Schablonen. Wie schon im Medienobjektspeicher werden auch im Dokumentenspeicher alle Eigenschaften von Objekten in zusätzlichen Tabellen als Name-Wert-Paare abgespeichert:

Document(doc_id, name, targetDir, targetURL, javaClass): Dokumente, bestehend aus einem eindeutigen Schlüssel (doc_id), dem Namen des zu erstellenden Lehr- oder Informationssystems (name), dem Pfad und der URL des Ausgabeverzeichnisses (targetDir bzw. targetURL), sowie dem Namen einer eventuell für das Dokument generierten Java Klasse (javaClass, siehe Kapitel 8.3).

DocumentProps(doc_id, attribute, value): Eigenschaften von Dokumenten in Form von Name-Wert-Paaren.

Block(block_id, genre, name, javaClass): Block, bestehend aus einem eindeutigen Schlüssel (block_id), dem Genre des Blocks (genre), seinem Namen (name) und dem Namen einer evtl. für den Block generierten Java Klasse (javaClass, siehe Kapitel 8.3).

BlockProps(block_id, attribute, value): Eigenschaften von Blöcken in Form von Name-Wert-Paaren.

DocBlock(doc_id, block_id): Zuordnung von Blöcken zu Dokumenten. Durch die Verwendung dieser Brückentabelle kann ein Block verschiedenen Dokumenten zugeordnet werden.

Template(tpl_id, name, objecttype, javaClass): Schablone, bestehend aus einem eindeutigen Schlüssel (tpl_id), einem Namen (name), dem Namen der zugrundeliegenden Objektklasse (objekttype mit den möglichen Werten "mediaobject" und "keyword"), sowie dem Namen der evtl. für die Schablone generierten Java Klasse (siehe Kapitel 8.3).

TemplateProps(tpl_id, attribute, operator, value): Ausdrücke zur Beschreibung von Eigenschaften der zu einer Schablone passenden Objekte. Die Felder "attribute" und "value" geben Name und verlangten Wert einer Eigenschaft an. Als Wert sind alle für das Attribut zulässigen Werte und "*" erlaubt. Über Operatoren können auch Vergleiche mit vorgegebenen Wertebereichen definiert werden.

BaseObject(block_id, objectType, objectId): Basisobjekte eines Blocks (block_id). Das Feld "objectId" enthält den Schlüssel des Basisobjekts, wobei das Feld "objectType" festlegt, ob es sich dabei um ein Medienobjekt oder eine Medienobjekt-Schablone handelt.

Lernziel(block_id, objectType, objectID): Lernziele eines Blocks (block_id). Das Feld "objectId" enthält den Schlüssel des Lernziels, wobei das Feld "objectType" festlegt, ob es sich dabei um ein Stichwort oder eine Stichwort-Schablone handelt.

AddOn(block_id, tpl_id, rel, count, abs, position): Zusatzobjekte eines Blocks (blk_id), spezifiziert durch den Schlüssel der beschreibenden Schablone, Klassenhäufigkeit (rel und count), Objekthäufigkeit (abs) und gewünschte Position der Einbindung (position).

Abbildung 8-11: E-R Diagramm des Dokumentenspeichers

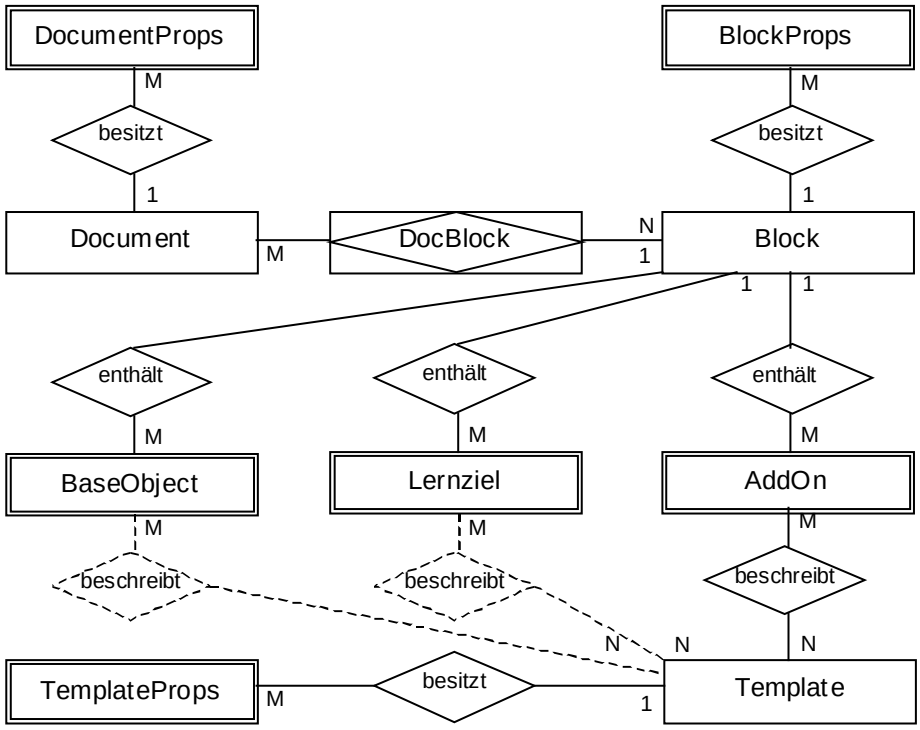


Abbildung 8-11 stellt den Dokumentenspeicher als das E-R Diagramm dar. Die Relationen zwischen Schablonen und Lernzielen bzw. Schablonen und Basisobjekten sind gestrichelt eingezeichnet, da es sich hierbei um keine statische, relationale Verbindung handelt.

Neben den aufgeführten, dokumentenbeschreibenden Tabellen und Relationen enthält der Dokumentenspeicher eine weitere Tabelle mit Name-Wert-Paaren zur Verwaltung dokumentenübergreifender Informationen:

Settings(attribute, value)

Beispiele für in "settings" abgelegte, dokumentenübergreifende Informationen sind die URL des zugehörigen Medienobjektspeichers und Angaben zu Aufbau und Aussehen der Nutzeroberfläche des Medienobjekt-Managers.

8.3 Konfiguration des i4 Frameworks

Bei der bisherigen - auf die verwendeten Datenstrukturen und Algorithmen konzentrierten - Darstellung der *Bottom-Up* Generierung von Lehr- und Informationssystemen sind eine Reihe von Fragen offen geblieben, die vor allem die Umsetzung der in Kapitel 5 beschriebenen doppelten Adaptierbarkeit betreffen:

- Wie können anwendungsabhängige Eigenschaften von Medienobjekten und zusätzliche Block-Genres definiert und in die vorhandenen Algorithmen integriert werden?
- Wie werden anwendungsabhängige Bewertungsfunktionen kodiert, bzw. generische Bewertungsfunktionen adaptiert?
- Wie werden die für den Benutzer sichtbaren HTML-Seiten unter Berücksichtigung der Seiten- und Kapitelstruktur erzeugt?

Alle mit diesen Fragen verbundenen Probleme lassen sich sehr einfach durch einen einzigen Mechanismus - die **externe Überlagerung interner Objekte** - lösen.

8.3.1 Externe Überlagerung interner Objekte

Alle in der Einleitung zu diesem Abschnitt genannten Probleme lassen sich auf die Frage nach der **Konfigurierbarkeit** eines Systems zur dynamischen *Bottom-Up* Erstellung hypermedialer Lehr- und Informationssysteme zurückführen. Die offene Frage ist dabei nicht, was konfiguriert werden muß, sondern vielmehr, wie die Konfiguration durch Anwender und Autor ermöglicht wird.

Das Hauptproblem ist, wie die extern festgelegte, anwendungs- und nutzerabhängige Konfiguration an die internen Algorithmen und Datenstrukturen übergeben wird. Hierzu gibt es eine Reihe von hinlänglich bekannten Mechanismen, wie z.B. Kommandozeilenparameter, Konfigurationsdateien und Schablonen.

All diese Mechanismen sind für die automatisierte Generierung von Lehr- und Informationssystemen nicht mächtig genug, da sie z.B. weder für die Beschreibung von neuen Medientypen noch für die Kodierung komplexer Bewertungsfunktionen geeignet sind.[44]

Dies sind jedoch unabdingbare Voraussetzungen für jede Art von dynamischer Dokumentenerstellung. Nur wenn der Autor in der Lage ist, Bewertungsfunktionen, Objektbeschreibungen, Auswahlkriterien und Strukturierungsmuster zu beeinflussen, ist das Generierungssystem auch wirklich für (fast) alle denkbaren Themen und Anwendungsgebiete geeignet. Systeme wie *Interbook*, *ELM-ART* und *Peba-II* (siehe Kapitel 3.4) sind daher im Grunde genommen keine Autorenwerkzeuge, da sie sich nur für eine statisch festgelegte Art der Darstellung einer stark eingeschränkten Menge von Themen eignen. Um die genannten Anforderungen zu erfüllen, wurde für das *i4 Framework* ein vollkommen neuer Weg der Parametrisierung gewählt, der beliebig flexibel und erweiterbar ist: die externe Überlagerung interner Objekte.

Ausgangspunkt der Überlegungen, die zum Mechanismus der Überlagerung interner Objekte führten, war die Frage, wie ein bezüglich Mächtigkeit und Flexibilität idealer Konfigurationsmechanismus aussehen könnte. Die meisten - im Grunde genommen sogar uneingeschränkte - Möglichkeiten der Anpassung sind sicherlich gegeben, wenn dem Autor eines Informationssystems der Sourcecode des Generierungssystems zur Verfügung steht:

Möchte der Autor z.B., daß alle von ihm erstellten Medienobjekte bevorzugt verwendet werden, so ändert er einfach den Code der lokalen Bewertungsfunktion auf Medienobjekten (localEval(), siehe Kapitel 7.3.4) und übersetzt das Generierungsprogramm neu. Ebenso ließen sich auch neue Medientypen und Kapitelgenres in den vorhandenen Code einbauen.

---

[44] Auch HTML Schablonen sind mittlerweile nicht mehr das, was sie einmal waren. Die von Systemen wie *Microsoft's Active Server Pages* [Weissinger99] oder *Allaire's Cold Fusion* [Allaire98] eingesetzten Mechanismen zur Beschreibung von zu erzeugendem HTML Code sind im Grunde genommen eher Programme als Schablonen. Sie stellen nicht nur Kontrollstrukturen wie Schleifen und Bedingungen zur Verfügung, sondern enthalten auch zusätzliche Funktionalitäten wie Datenbankzugriffe und Dateioperationen.

Da die Implementierung des *i4 Frameworks* jedoch relativ umfangreich und komplex ist und viele Funktionen Seiteneffekte besitzen, sollten Änderungen des Sourcecodes nach Möglichkeit nur an sinnvollen Stellen und über sichere Mechanismen möglich sein.

Hier bietet es sich an, auf objekt-orientierte Konstrukte wie Einkapselung und Vererbung zurückzugreifen. Anstatt den vorhandenen Sourcecode zu verändern, ist es einfacher und sicherer, ihn punktuell durch Ableitung der vorhandenen Implementierung zu erweitern bzw. zu überlagern. Diese Vorgehensweise hat ferner den Vorteil, daß sie auf der existierenden Klassenhierarchie des Frameworks aufbaut:

Anstelle die Implementierung der vorhandenen Methode *localEval()* innerhalb der *Framework*-Klasse *CMediaobjekt* zu modifizieren, würde der Autor eine neue, von *CMediaobject* abgeleitete Klasse erstellen und in dieser Klasse eine eigene, angepaßte Implementierung von *localEval()* bereitstellen.

Der nachfolgend angegebene Java Code würde z.B. die gewünschte Änderung der Bewertung von Medienobjekten bewirken:

```
public class MyMediaobject extends CMediaobject {

  public double localEval() {
       double le = super.localEval();
       if( Author().indexOf("MyName") > -1 ) le = le * 2;

       return le;
  }
}
```

Im nächsten Schritt müßte der Autor dem System noch mitteilen, daß anstelle der Klasse *CMediaobject* nun die Klasse *MyMediaobject* zur Beschreibung von Medienobjekten verwendet werden soll, d.h. in der Implementierung des Frameworks sind Medienobjekte nicht mehr von Typ *CMediaobject*, sondern Instanzen der Klasse *MyMediaobject*.

Um dieses Problem zu lösen, bietet es sich an, auf das Design-Muster der **Factory Methoden** zurückzugreifen [Gamma+95]. Die Idee der *Factory Methoden* ist, daß Instanzen ableitbarer Klassen nur von wenigen (im Idealfall einer einzigen), parametrisierbaren Funktionen oder Methoden erzeugt werden. Hierdurch sind die Datentypen von Objekten sehr leicht veränderbar und konfigurierbar, da erst zur Laufzeit eine Verknüpfung von Schnittstelle und Implementierung hergestellt wird.

Innerhalb der Implementierung des *i4 Frameworks* ist die *Factory Methode* zur Erzeugung von Medienobjekten die Methode *docMediaobjects()* innerhalb der Klasse *CDocument*. Durch Anpassung einer Zeile in dieser Methode kann der Medienobjekt-Datentyp von *CMediaobject* auf *MyMediaobject* geändert werden:

```
public class CDocument {

  ...

  public Vector docMediaobjects() {
       Vector vmo = new Vector();
       Cmediaobject mo;

            ...
            mo = new MyMediaobject();
            vmo.addObject( mo );
            ...

       return vmo;
  }
}
```

Diese Kombination aus Vererbung und *Factory Methoden* läßt sich mit dem Begriff "**interne Überlagerung interner Objekte**" umschreiben. Intern (d.h. im Sourcecode des *Frameworks*) enthaltene Implementierungen werden durch andere interne Implementierungen ersetzt, indem die Datentypen von Objekten erst zur Laufzeit festgelegt werden.

Die interne Überlagerung besitzt jedoch drei Nachteile, die ihre Verwendung als Konfigurationsmechanismus nicht sinnvoll erscheinen lassen:

- Die interne Überlagerung erfordert, daß dem Autor eines Lehr- oder Informationssystems der Sourcecode des *Frameworks* zur Verfügung steht. Dies wird der Anbieter einer Implementierung des *Frameworks* aus kommerziellen Überlegungen aber sicherlich kaum zulassen.
- Auch durch Rückgriff auf die Klassenhierarchie des *Frameworks* läßt sich nicht sicherstellen, daß die vom Autor vorgenommenen Änderungen nur an den dafür vorgesehenen Stellen durchgeführt wurden und keinerlei unerwünschten Seiteneffekte mit sich bringen.
- Nach jeder Änderung muß das *Framework* neu übersetzt werden. Dies bedeutet neben dem nicht zu unterschätzenden zusätzlichen Aufwand, daß dem Autor eine komplette Entwicklungsumgebung zur Verfügung stehen muß.

All diese Nachteile können dadurch beseitigt werden, daß die Überlagerung nicht intern zur Übersetzungszeit, sondern extern zur Laufzeit vorgenommen wird. Dies bedeutet, daß die selbst erstellten, von internen Klassen abgeleiteten Klassen nicht integraler Bestandteil des Frameworks sind, sondern dynamisch hinzugebunden werden.

Um eine **externe Überlagerungen interner Objekte** zu ermöglichen, bieten sich mindestens 3 grundverschiedene Techniken an, die jedoch zum Teil nicht alle der genannten Probleme der internen Überlagerung befriedigend lösen:

- Das *Framework* steht als Java *Bytecode* zur Verfügung. Der Autor kann beliebige neue Klassen erstellen, zu Bytecode übersetzen und mit den vorhandenen Klassen zusammenbinden. Der Nachteil dieser Vorgehensweise ist jedoch, daß das *Framework* komplett in Java programmiert werden muß, was zumindest bei der aktuellen Implementierung nicht der Fall ist. Darüber hinaus muß dem Autor eine Java Entwicklungsumgebung zur Verfügung stehen.
- Die abgeleiteten Klassen werden als *Dynamic Link Libraries* (DLLs) realisiert und mit Hilfe einer Konfigurationsdatei zur Laufzeit dynamisch von *Framework* geladen. Für die Umsetzung dieser Technik werden jedoch ebenfalls einen Compiler und ein Linker benötigt. Darüber hinaus erfordern *Dynamic Link Libraries* ein bestimmtes Betriebssystem und die Verwendung der Implementierungssprache des *Frameworks*, da nur so komplexe Objekte zwischen DLL und *Framework* ausgetauscht werden können[45].
- Die neu erstellten Klassen werden nicht übersetzt, sondern zur Laufzeit vom *Framework* interpretiert. Dieses Verfahren erfordert zwar die Implementierung eines (Java) Interpreters, birgt aber im Gegensatz zu den beiden anderen Varianten keinerlei technischen Probleme für den Autor. Der einzige Nachteil des Interpreters ist, daß die Ausführung der überlagernden Methoden erheblich langsamer ist, als dies bei Java *Bytecode* und vor allem bei DLLs der Fall wäre.

Da Komfort für den Autor als wichtigstes Leistungsmerkmal eines Autorensystems angesehen wird, realisiert die aktuelle Implementierung des *i4 Frameworks* die Interpreter-Variante der externen Überlagerung interner Objekte[46]. Als Konfigurationssprache wird Java verwendet, da diese Programmiersprache relativ einfach zu erlernen, plattformunabhängig und sehr weit verbreitet ist. Darüber hinaus ist Java einfacher und schneller zu interpretieren als C++ und in Bezug auf objektorientierte Programmierung leistungsfähiger als *VisualBasic*.

---

[45] Bei Verwendung von C++ müssen die *Dynamic Link Libraries* sogar mit dem gleichen Compiler erstellt werden wie das Framework, da das Speicherlayout von Objekten nicht standardisiert ist.

[46] Kernstück des Interpreters ist in der aktuellen Implementierung ein handcodierter, *recursive descent* LR Parser [Aho+86]. Für nachfolgende Versionen ist jedoch - vor allem aus Gründen der Geschwindigkeit und Wartbarkeit - der Einsatz eines generierten LALR Parsers vorgesehen.

8.3.2 Java Wrapper Klassen

Die Realisierung der externen Überlagerung interner Klassen soll im Folgenden anhand des komplett in *VisualBasic* implementierten Medienobjekt-Managers dargestellt werden. Bei der Implementierung wurde intensiver Gebrauch von objekt-orientierten Konstrukten wie Einkapselung und Polymorphie gemacht.

Jede der in den vorangegangenen Kapiteln beschriebenen Datenstrukturen ist als eine **VisualBasic Klasse** realisiert. Die wichtigsten dieser Klassen sind:

interne VisualBasic Klassen

Datenstruktur	Klassenname	Beschreibung
Medienobjekt	CMediaobject	Kapitel 4.1 - 4.4
Stichwort	CKeyword	Kapitel 4.4
Dokument	CDocument	Kapitel 2.2
Block	CBlock	Kapitel 5.3
Kapitel	CChapter	Kapitel 7.7
Seite	CFrame	Kapitel 7.6
Nutzer	CUser	Kapitel 5.1 - 5.4

Alle Abildungen und Projektionen auf den genannten Datenstrukturen sowie sämtliche Maßzahlen und Bewertungsfunktionen sind über *Properties* (Eigenschaften) und Methoden der *VisualBasic* Klassen zugänglich. Darüber hinaus wurde versucht, sämtliche Klassen möglichst selbstbeschreibend zu gestalten, d.h. auch Verwendung, Aussehen und Kodierung von Objekten werden über *Properties* und Methoden festgelegt. So enthält z.B. die Klasse *CMediaobject* eine Methode *GetLinkedCode*(), die den HTML Code zur Darstellung eines einzelnen Medienobjekts inklusive der darin enthaltenen Hyperlinks zurückliefert.

Zu jeder internen *VisualBasic* Klasse existiert eine externe **Java Wrapper Klasse.** Die *Java Wrapper Klasse* ist eine virtuelle - d.h. eine nicht implementierte - Java Klasse, deren Schnittstelle weitgehend dem sichtbaren Teil der Schnittstelle der internen Klasse entspricht. Jeder Aufruf einer Methode der *Wrapper Klasse* wird auf die Implementierung der Methode in der zugehörigen internen Klasse umgeleitet.

Das Besondere an den *Wrapper Klassen* ist, daß sie extern, d.h. außerhalb des *Frameworks* sichtbar sind. Alle Methoden einer *Java Wrapper Klasse* können sowohl intern, d.h. aus dem *VisualBasic* Code des *Frameworks* heraus, als auch extern, d.h. durch eine vom Autor erstellte Funktion oder Methode aufgerufen werden. Hierdurch ist es möglich, externe Methodenaufrufe auf interne Objekte abzubilden. *Java Wrapper Klassen* bilden somit die externen Basisklassen für den Konfigurationsmechanismus.

Die nachfolgende Übersicht zeigt die wichtigsten Datenstrukturen und internen Klassen samt deren zugehöriger *Java Wrapper Klassen*:

interne und externe Klassen

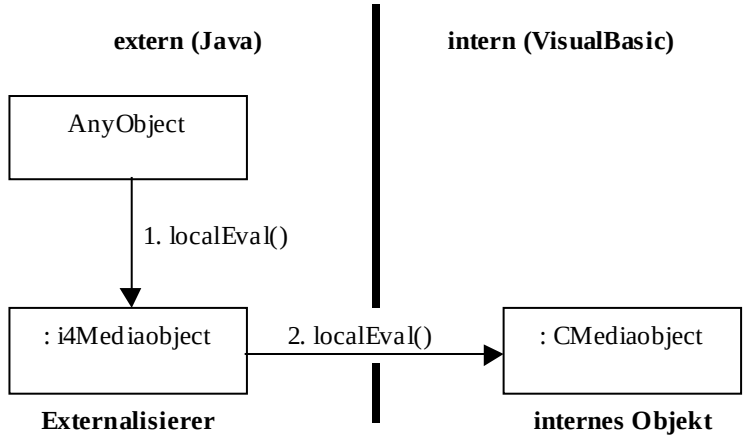
Datenstruktur	Klassenname	*Java Wrapper Klasse*
Medienobjekt	CMediaobject	i4Mediaobject
Stichwort	CKeyword	i4Keyword
Dokument	CDocument	i4Document
Block	CBlock	i4Block
Kapitel	CChapter	i4Chapter
Seite	CFrame	i4Frame
Nutzer	CUser	i4User

Für jedes interne Objekt kann eine Instanz der entsprechenden *Wrapper Klasse* - ein sog. **Externalisierer** - erstellt werden. Ein Externalisierer ist somit eine externe Schnittstelle eines internen Objektes. Abbildung

8-12 verdeutlicht das Zusammenspiel zwischen internen Objekten und Klassen auf der einen Seite und *Java Wrapper Klassen* und Externalisierern auf der anderen Seite:

Zu einem internen Objekt vom Typ *CMediaobject* wurde ein Externalisierer von Typ *i4Mediaobject* erstellt. Auf dem Externalisierer wird von einem weiteren (vom Autor erstellten) externen Objekt die Methode *localEval()* aufgerufen. Dieser Aufruf wird vom Externalisierer an das interne Objekt weitergeleitet und dort ausgeführt. Anschließend wird das Ergebnis des Methodenaufrufs vom internen Objekt über den Externalisierer an die ursprünglich aufrufende Methode zurückgeliefert. Die Umwandlung von eventuell als Parameter übergebenen externen (Java-)Objekten in interne (*VisualBasic*-)Objekte geschieht durch den Java Interpreter.

Abbildung 8-12: Interne Objekte und Externalisierer

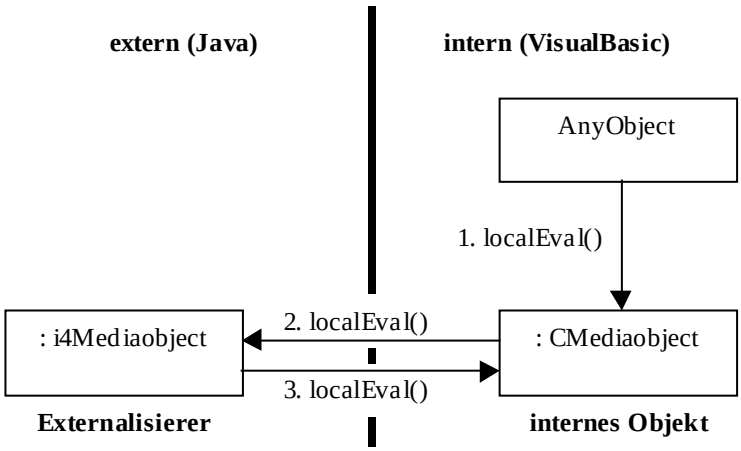


Genau wie jeder Externalisierer einen Verweis auf sein zugehöriges internes Objekt enthält, verwaltet auch jedes interne Objekt einen Verweis auf seinen zugehörigen Externalisierer. Die Beziehung zwischen internen Objekten und Externalisierern ist immer 1 : 1, d.h. jedem Externalisierer ist genau ein internes Objekt zugeordnet und umgekehrt.

Der Zugriff auf jedes interne Objekt kann sowohl "von innen" durch ein anderes internes Objekt als auch "von außen" durch den Externalisierer des Objektes geschehen. Um die bereits angedeutete Möglichkeit der dynamischen Typisierung realisieren zu können, leitet jedes interne Objekt alle "von innen" kommenden Methodenaufrufe auf seinen Externalisierer um.

In Abbildung 8-13 wird im Gegensatz zu Abbildung 8-11 die Methode *localEval()* von einem beliebigen internen Objekt auf einem internen Medienobjekt aufgerufen. Da es sich um einen internen Aufruf handelt, wird dieser mit Hilfe des Java Interpreters an den Externalisierer des Medienobjekts weitergereicht. Der Externalisierer macht wiederum nichts weiter, als den Aufruf umgehend zum internen Objekt zurückzuleiten. Dieser zweite Aufruf von *localEval()* ist für das Medienobjekt extern, so daß er ausgeführt werden kann. Das Ergebnis des Aufrufs wird anschließend über den *Wrapper* an das aufrufende Objekt zurückgegeben. Das dargestellte Szenario enthält 4 Übergänge von internen zu externen Objekten und umgekehrt, die jeweils mit Hilfe des Java Interpreters durchgeführt werden.

Abbildung 8-13: Umleitung interner Methodenaufrufe



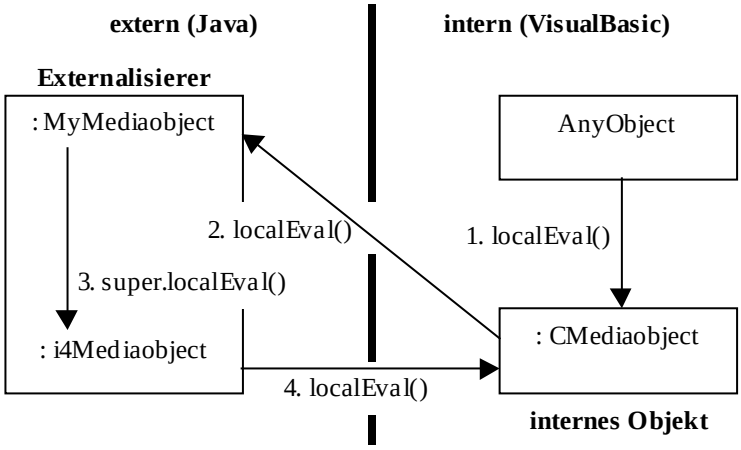
Als Externalisierer eines internen Objektes kann jedes Objekt verwendet werden, dessen Datentyp der *Wrapper Klasse* des internen Objektes oder einer davon abgeleiteten Klasse entspricht. Da darüber hinaus die Verbindung zwischen einem internen Objekt und seinem Externalisierer zur Laufzeit erstellt wird, ist es möglich, eigene Implementierungen für fast alle internen Methoden bereitzustellen. Die externen Implementierungen können dabei jedoch jederzeit über den Externalisierer auf die interne Implementierung zurückgreifen.

Mit Hilfe einer von *i4Mediaobject* abgeleiteten Klasse kann somit das am Anfang dieses Kapitels vorgestellte Szenario der Überlagerung einer lokalen Bewertungsfunktion realisiert werden:

```
public class MyMediaobject extends i4Mediaobject {

  public MyMediaobject(Object obj) { super(obj); }

  public double localEval() {
       double le = super.localEval();
       if( Author().indexOf("MyName") > -1 ) le = le * 2;

       return le;
  }
}
```

Abbildung 8-12 stellt die Beziehungen der beteiligten Klassen und Objekte untereinander sowie die Reihenfolge der verschiedenen internen und externen Aufrufe von *localEval()* im Überblick dar.

Abbildung 8-14: Überschreiben einer internen Methode



Wie in Abbildung 8-14 zu ersehen ist, wurde ein internes Medienobjekt mit einem Externalisierer verknüpft, dessen Datentyp eine von *i4Mediaobject* abgeleitete Klasse ist[47]. Jeder interne Aufruf der Methode *localEval()* wird an den Externalisierer weitergeleitet. Der Externalisierer greift auf die Methode *localEval()* seiner Basisklasse zurück, woraus ein über die Klasse *i4Mediaobject* und den Java Interpreter an das interne Objekt weitergeleiteter externer Methodenaufruf resultiert. Das Ergebnis dieses externen Aufrufs wird über den Java Interpreter und die Klasse *i4Mediaobject* an den Externalisierer weitergeleitet. Nach Ausführung des vom Autor erstellten Java Codes wird der endgültige Wert der lokalen Bewertung über die Klasse *i4Mediaobject*, den Java Interpreter und das interne Objekt an den Aufrufer übergeben.

Die Verknüpfung zwischen einem internen Objekt und einem Externalisierer wird durch den Aufruf des **Konstruktors** des Externalisierers hergestellt. In jeder *Java Wrapper Klasse* ist ein Konstruktor der Form *Wrapperclass(Object obj)* definiert. Das Argument des Konstruktors ist dabei ein Verweis auf das interne, durch den Externalisierer zu kapselnde Objekt.

Beispiel: Von der Klasse *i4Mediaobject* wurde eine Klasse *MyMediaobject* abgeleitet, von der wiederum die Klasse *MyImage* abgeleitet ist. Die Konstruktoren der abgeleiteten Klassen müssen so definiert sein, daß sie letztendlich in einem Aufruf des Konstruktors der Klasse *i4Mediaobject* münden, dem das zu kapselnde Objekt übergeben wird:

```
public class MyImage extends MyMediaobject {

  MyImage( Object objInternal ) {
       super( objInternal );
       ...
  }
  ...
}

public class MyMediaobject extends i4Mediaobject {

  MyMediaobject( Object objInternal ) {
       super( objInternal );
       ...
  }
  ...
}
```

Die Verknüpfung eines internen (Medien)Objekts mit der Implementierung der externen Klasse *MyImage* geschieht implizit bei der Erzeugung des Externalisierers:

```
  ...
  Vector wrapped = new Vector();
  Vector mos = doc().docMediaobjects();
  Enumeration enumMos = mos.elements();

  while( enumMos.hasMoreElements() ) {
       i4Mediaobject mo = enumMos.nextElement();
       if( mo.isImage() ) {
            wrapped.addObject( new MyImage( mo ));
       } else {
            wrapped.addObject( mo );
       }
  }
  ...
```

Der angegebene Beispielcode bewirkt, daß alle Medienobjekte, die Grafiken repräsentieren, mit einem Externalisierer vom Typ *MyImage* verknüpft werden. Durch Überschreiben von Methoden der internen

[47] Die Realisierung der dazu benötigte *Factory* Methode ist Thema des nächsten Kapitels.

Klasse *CMediaobject* in der externen Klasse *MyImage* können nun spezielle, nur für Grafiken gültige Eigenschaften, Bewertungsfunktionen oder Einbettungsmechanismen definiert werden.

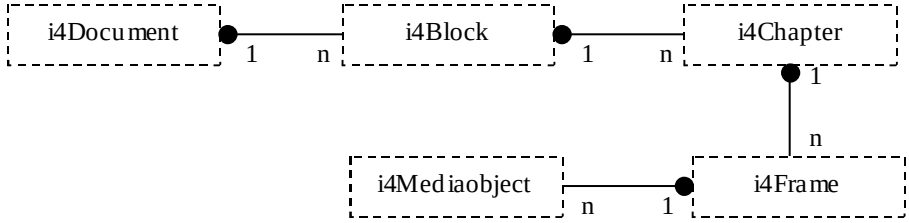
8.3.3 Benutzung der Java Wrapper Klassen

Bildet man die für die automatisierte *Bottom-Up* Erzeugung von Lehr- und Informationssystemen relevanten Konstrukte wie Blöcke, Kapitel und Medienobjekte auf eine Klassenhierarchie ab, so fällt auf, daß fast jede Klasse ein Aggregat aus anderen Klassen ist[48]:

Ein Dokument (Lehr- bzw. Informationssystem) besteht aus einer Menge von Blöcken. Jeder Block setzt sich aus einer Reihe von Kapiteln zusammen, die wiederum jeweils aus einer Menge von Seiten bestehen. Jede Seite letztlich beinhaltet ein oder mehrere Medienobjekte.

Aus dieser Schachtelung ergibt sich ein sequentielles **Klassendiagramm**, das für interne und *Wrapper* Klassen mit Ausnahme der Klassennamen identisch ist:

Abbildung 8-15: Klassendiagramm



Diese Struktur findet sich auch in den verfügbaren *Factory* Methoden: Dokumente erzeugen Externalisierer für Blöcke, Blöcke Externalisierer für Kapitel und so weiter. Ausnahmen existieren lediglich für Medienobjekte und Stichworte, die als Basisobjekte aus Dokumenten heraus erzeugbar sind, sowie für Seiten, die aufgrund der *Bottom-Up* Generierungsreihenfolge nicht aus Kapiteln, sondern aus Blöcken heraus erstellt werden.

Die nachfolgende Tabelle zeigt alle verfügbaren *Factory* Methoden im Überblick:

Klasse	***Factory Methode***	**Beschreibung**
i4Document	Vector docBlocks()	Liefert alle für das Dokument definierten Blöcke.
i4Document	Vector docMediaobjects()	Liefert alle für die Erstellung des Dokuments verfügbaren Medienobjekte.
i4Document	Vector docKeywords()	Liefert eine Liste aller referenzierten Stichworte.
i4Block	Vector blockChapters()	Liefert alle Kapitel des Blocks.
i4Block	i4Frame makeFrame(i4Mediaobject mo)	Erstellt eine Seite aus einem einzelnen Medienobjekt.
i4Block	Vector blockBaseObjects()	Liefert alle für die Erstellung des Blocks verwendbaren Medienobjekte.
i4Chapter	Vector chapFrames()	Liefert alle Seiten eines Kapitels.
i4Frame	Vector frameContent()	Liefert alle auf einer Seite enthaltenen Medienobjekte.

Wie die Tabelle der *Factory* Methoden zeigt, können mit Ausnahme von Nutzer-Objekten und Dokumenten alle bisher beschriebenen internen Objekte durch Methodenaufrufe mit Externalisierern versehen und somit in ihrer Implementierung beeinflußt werden.

[48] siehe auch Abbildung 2-4

Um **Dokumente** kapseln zu können, wird eine zusätzliche Klasse *i4System* verwendet. *i4System* ist im Gegensatz zu allen bisher vorgestellten externen Klassen kein echter *Java Wrapper*, da kein internes Pendant exisitiert.

Die Klasse *i4System* muß vom Autor des Lehr- oder Informationssystems mit einer eigenen Klasse abgeleitet werden. Wichtigster Bestandteil der abgeleiteten Klasse ist eine Implementierung der Methode

i4Document **getDocument**(Object objInternalDocument, i4User usr),

die einen Externalisierer für ein das komplette Dokument repräsentierende interne Objekt erzeugt.

Der Pfad der Implementierung der von *i4System* abgeleiteten Klasse wird dem Medienobjekt-Manager als Kommandozeilen-Argument übergeben. Eine Instanz dieser Klasse wird vom Medienobjekt-Manager erzeugt und unter dem Namen *"theSystem"* als globale Variable zur Verfügung gestellt.

In den meisten Fällen besteht die Ableitung der Klasse *i4System* nur aus dem Aufruf des Konstruktors der externen Dokument-Klasse:

```
public class MySystem extends i4System {

  i4Document getDocument( Object objDoc, i4User usr ) {
       return new MyDocument( objDoc, i4User );
  }
}
```

Alle anderen externen Objekte können anschließend mit Hilfe der in *getDocument()* erzeugten Instanz des externen Dokument-Objektes erstellt und mit internen Objekten verknüpft werden.

Die bisher beschriebenen Datenstrukturen und Objekte sind rein virtuell, d.h. sie existieren nur im Speicher des Computers. Das mit Hilfe der in Kapitel 7 beschriebenen Algorithmen erstellte und über die in diesem Kapitel beschriebenen Objekte konfigurierte Lehr- oder Informationssystem ist zwar von seiner Struktur und Zusammenstellung her fertig, aber für den Benutzer nicht sichtbar.

Um das Dokument für den Benutzer zugänglich zu machen, müssen die einzelnen Seiten (und die darin enthaltenen Medienobjekte) in der berechneten Struktur als darstellbare Dateien im Dateisystem abgelegt werden. Grundlage der Konvertierung von Seiten in ein lesbares Format ist die *Java Wrapper Klasse i4Frame*. Objekte dieser oder einer abgeleiteten Klasse externalisieren interne Objekte vom Typ *CFrame*, welche wiederum die einzelnen **Seiten** des Dokuments repräsentieren.

Die beiden wichtigsten Methoden der *Wrapper Klasse i4Frame* sind

boolean **canBeJoined**(i4Frame frm2) und

void **print**(PrintWriter pw).

Die Funktion *canBeJoined()* implementiert die in Kapitel 7.6.2 beschriebene Funktion, die überprüft, ob das Hinzufügen eines weiteren Medienobjekts zu einer Seite mit dem vom Autor festgelegten Seiten-Layout in Einklang gebracht werden kann. Der Rückgabewert der Funktion ist "true", wenn die angegebene Seite zu der aktuellen Seite hinzugefügt werden kann und "false", wenn dies nicht möglich ist. Durch Überschreiben von *canBeJoined()* können jedoch nicht nur Aspekte des Layouts, sondern auch der Strukturierung berücksichtigt werden: Um z.B. ein Ebenen-Modell wie in "Mathe online" (siehe Kapitel 2.2.5) umzusetzen, dürfen nur Seiten eines Genres zusammengefaßt werden.

Die Methode *print()*dient dem Herausschreiben einer Seite in das Dateisystem, z.B. als HTML Datei. Das übergebene Argument ist der Datenstrom, in den die Ausgabe geschrieben wird.

Anzumerken ist, daß das Ausgabeformat nicht zwangsläufig HTML sein muß. Ebenso ist es durch entsprechende Implementierung der Methode *print()* möglich, XML-Dateien oder *VisualBasic* Formulare zu erzeugen.

Dadurch, daß das Aussehen von Seiten nicht durch Schablonen beschrieben, sondern in Java programmiert wird, existieren so gut wie keinerlei Beschränkungen hinsichtlich Aufbau und Layout. Da neben den Ableitungen der *Wrapper Klassen* auch eigene - nicht an interne Objekte gebundene - Klassen erstellt werden können, ist es sogar möglich, komplette Layout Bibliotheken aufzubauen. Hierdurch

werden gestalterische Elemente wiederverwendbar, was zu einer erheblichen Verminderung des Programmieraufwands führt.

Der Preis für die Flexibilität der Ableitung interner Klassen und die Programmierung von Layouts ist - neben dem erhöhten Aufwand - vor allem Geschwindigkeit. Da der Java Code vom Medienobjekt Manager interpretiert wird, ist die Konfiguration einzelner Objekte erheblich langsamer, als wenn hierfür ein einfacherer Mechanismus wie z.B. Beschreibungsdateien oder Schablonen verwendet worden wäre.

Ohne einen mächtigen und vor allem flexiblen Konfigurationsmechanismus ist jedoch die in Kapitel 5 beschriebene doppelte Adaptierbarkeit nicht realisierbar. Da jedoch die doppelte Adaptierbarkeit eine der wichtigsten Voraussetzungen für die Erstellung eines Werkzeugs zur automatisierten Erstellung von Dokumenten ist, muß an dieser Stelle der Komplexität der Aufgabe Tribut gezollt werden.

9 Fallbeispiel

In den vorangegangenen Kapiteln wurden die einzelnen Algorithmen und die diese implementierenden Werkzeuge eher abstrakt vorgestellt. Um ein besseres Gefühl für das Arbeiten mit einem auf dem Bottom-Up Ansatz basierenden Autorenwerkzeug zu vermitteln, wird in diesem Kapitel der komplette Prozeß von der Modularisierung existierender Ausgangsmaterialien bis zur Generierung einer Web-Site anhand eines konkreten Fallbeispiels nachvollzogen.

Ziel der Fallstudie war es Dokumente zum Thema „Network Computing“ zu generieren. Im einzelnen sollten von den zu generierenden Lernmaterialien die folgenden Themenbereiche behandelt werden können:

- Lokale und Globale Netze
- Peer-to-Peer und Client/Server-Netze
- Netzwerk Referenzmodelle (OSI/ISO und IEEE 802)
- Signalübertragung und Protokolle
- Topologien und Zugriffsmethoden

Als Ausgangsmaterialien standen ein Skript der Telekom AG[49], Foliensätze verschiedener Dozenten sowie diverse Einzeldokumente und Grafiken aus dem Internet zur Verfügung. Die Einzeldokumente und ausgewählte Grafiken und Abbildungen aus den Foliensätzen wurden über den in Kapitel 8 beschriebenen Medienobjekt-Dialog manuell in den Medienobjekt-Manager eingegeben. Für den Import und die Indizierung des Skriptes wurde der in Kapitel 8.1 beschriebene Import-Mechanismus verwendet. Durchschnittlich wurden für die komplette Attributierung eines Medienobjekts weniger als 5 Minuten benötigt, was jedoch zuweilen zu Lasten der Genauigkeit ging (siehe Kapitel 9.3).

Als sehr lästig bei der Indizierung erwiesen sich die vielen zusammengesetzten Fachbegriffe, die zu sehr unterschiedlichen Schreibweisen ein und desselben Wortes führten (z.B. Mischtopologie vs. Misch-Topologie), was wiederum eine aufwendige Überprüfung der automatisch eingelesenen initialen Stichwortmenge notwendig machte.

Im Ergebnis standen jedoch am Ende mehr als 100 Medienobjekte zur Verfügung, die über mehr als 600 Indexeinträge mit knapp 200 Stichworten verbunden waren.

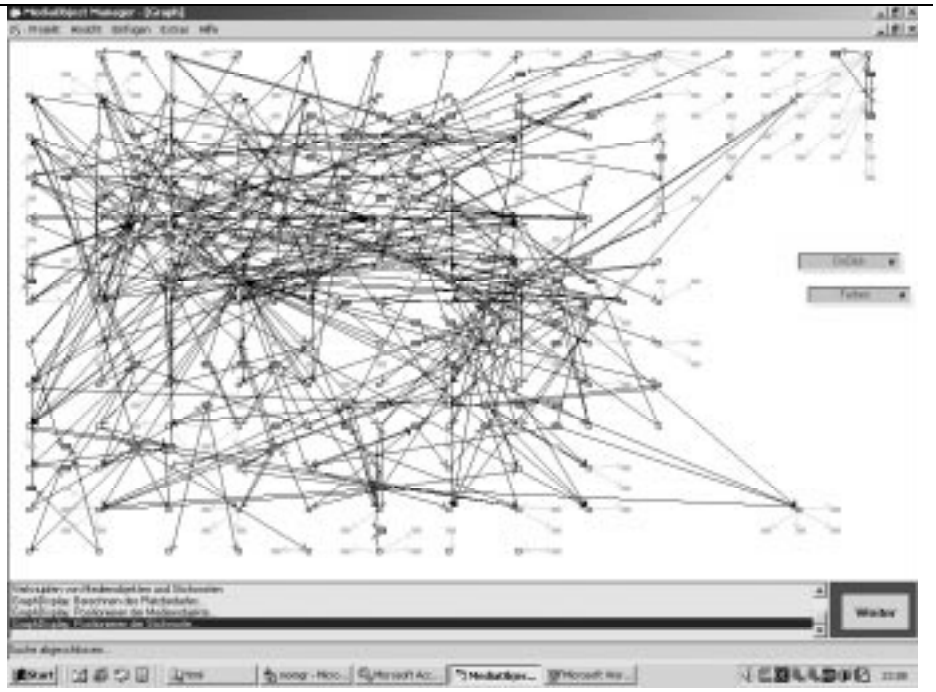
9.1 Eigenschaften des Abhängigkeitsgraphen

Unabhängig von einem konkreten Nutzungsszenario bildet der Medienobjekt-Manager aus den zur Verfügung stehenden Medienobjekten zunächst einen Abhängigkeitsgraphen und berechnet die in Kapitel

[49] Prof. Dr. Ralf Kories. „Vernetzte Rechnersysteme“. Leipzig, Dieburg, 1999.

6 definierten Maßzahlen. Um einen Eindruck von der Vernetzung eines Abhängigkeitsgraphen zu geben, ist in Abbildung 9-1 ein Screenshot des Abhängigkeitsgraphen der Beispielanwendung dargestellt. Die kleinen gelben Quadrate stellen die Medienobjekte dar, Stichworte sind durch graue Rechtecke repräsentiert.

Abbildung 9-1: Abhängigkeitsgraph



Von den 107 Medienobjekten sind ohne Angabe individuellen Vorwissens 34, d.h. knapp ein Drittel, nicht erlernbar. Dies liegt im wesentlichen daran, daß bei der Zielgruppe des eingespeisten Skriptes der Telekom AG von elektrotechnischen Grundkenntnissen ausgegangen wird, d.h. Stichworte wie z.B. „Spannungswellenamplitude“ oder „Wellenwiderstand“ ohne weitere Erklärung verwendet werden.

Im Abhängigkeitsgraph sind drei disjunkte Zyklen enthalten, von denen der größte 34 Medienobjekte aus mehreren kleineren Zyklen zu einem komplexen Zyklus verbindet. Dies bedeutet, daß ca. die Hälfte aller erlernbaren Objekte auf einem Zyklus liegt.

Insgesamt sind in dem Abhängigkeitsgraphen 10 wohlstrukturierte Subgraphen enthalten, die ihrerseits teilweise weitere Subgraphen enthalten, die ebenfalls wohlstrukturiert sind. Der längste der wohlstrukturierten Subgraphen besteht aus 5 Medienobjekten. Der längste sequentielle Subgraph umfaßt 4 Objekte und behandelt nacheinander die Themen „Rechnernetze“, „Topologien“, „Stern-Topologie“ und „Misch-Topologien“. Die einzelnen Texte passen nach meinem subjektiven Empfinden gut zueinander, auch wenn sie im Ausgangsdokument zwar in dieser Reihenfolge aber nicht unmittelbar aufeinanderfolgend angeordnet waren. Anzumerken ist noch, daß keine hierarchischen Subgraphen existieren, die Medienobjekte aus verschiedenen Quellen aufspannen.

Zur Evaluation des Medienobjekt-Managers wurden zwei Szenarien durchgespielt:

- Generierung eines Dokuments zur Vermittlung eines einzelnen Stichworts
- Generierung eines Dokuments in dem möglichst viele der vermittelbaren Stichworte enthalten sind

Als Zielformat wurde HTML gewählt, d.h. im Ergebnis entstand für jedes Testszenario eine Web-Site aus über Hyperlinks miteinander verbundenen Seiten. Hierzu wurde neben der Wrapper-Klasse „i4Application“ vor allem die Klasse „i4Application“ überschrieben, die für die Kodierung einer einzelnen Bildschirmseite zuständig ist. Abbildung 9-2 stellt exemplarisch das Layout der Seiten dar:

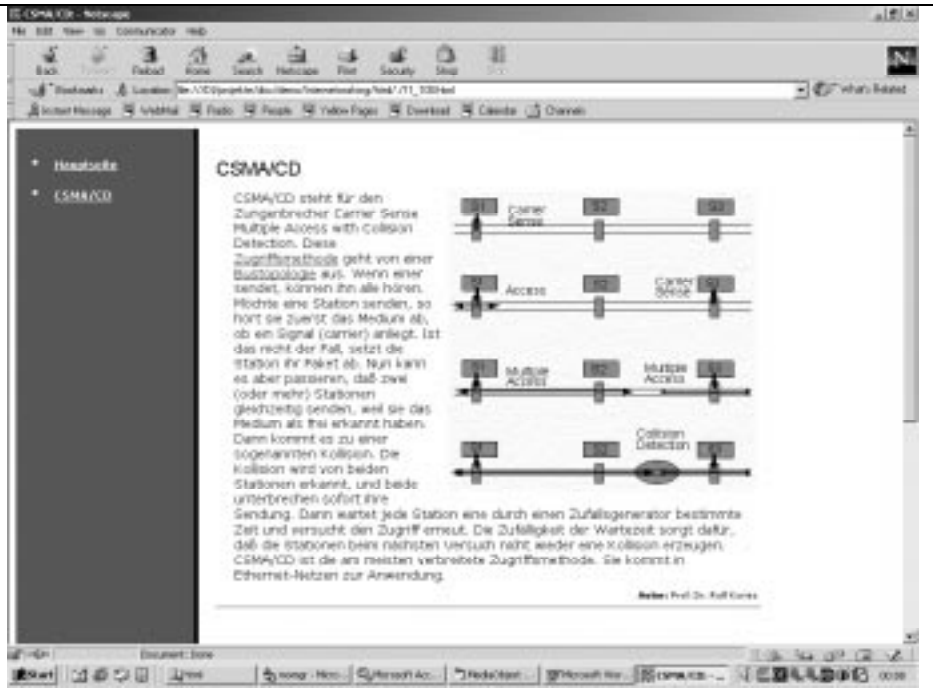


Abbildung 9-2: Bildschirm des zur Vermittlung des Stichwortes "CSMA/CD" generierten Lernsystems

Jede Bildschirmseite enthält auf der rechten Seite eine Menüleiste, über die zu den Startseiten der einzelnen Böcke gesprungen werden kann. Der Darstellungsteil besteht aus einer Seitenüberschrift, der Darstellung der auf der Seite plazierten Medienobjekten und Verweisen auf die in erzeugten Seitenstruktur nachfolgenden und vorangehenden Seiten.

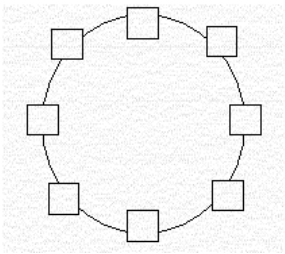
9.2 Vermittlung eines einzelnen Stichworts

Im ersten Testszenario war es das Ziel, ein Lernsystem zur Vermittlung des Stichwortes "CSMA/CD" zu generieren. Der Dokumentengenerator wählte hierzu 6 Medienobjekte (4 Texte und 2 Grafiken) aus, die in einer sequentiellen Struktur gegliedert wurden. Das gesamte Dokument liest sich sehr flüssig und enthält keine inhaltlichen Brüche:

Rechnernetze

Ein Rechnernetz besteht aus mindestens zwei Rechnern und einem Kommunikationsmedium, das sie miteinander verbindet. Dennoch gibt es natürlich einen Unterschied zwischen zwei Rechnern, die einen gemeinsamen Drucker benutzen und den Millionen von verbundenen Rechnern im Internet. Eine grobe Klassifizierung von Rechnern erfolgt in lokale Netze und Weitverkehrsnetze.

Topologien



Damit Rechner gemeinsam ein Netz bilden können, müssen sie offenbar miteinander durch ein Übertragungsmedium verbunden sein. Charakteristisch für ein Netz ist, wie die Rechner untereinander den Zugriff auf das gemeinsame Medium regeln, und das hängt wieder von der geometrischen Struktur des Netzes, der sogenannten Topologie ab. Man unterscheidet drei Topologien:

- *Bus*
- *Stern*
- *Ring*

Stationen in Bus-Topologie sind untereinander mit einem durchgehenden Kabel verbunden. Von Stern-Topologie spricht man, wenn jede Station einzeln mit einem zentralen Gerät verbunden ist. Stationen in einer

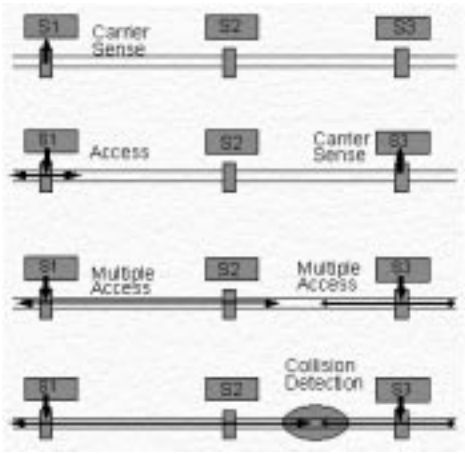
Ring-Topologie sind jeweils mit ihrem rechten und linken Nachbar verbunden, so daß insgesamt ein Ring gebildet wird.

Zugriffsmethoden

Zugriffsmethoden nennt man die Spielregeln, die innerhalb eines Protokolls gelten, um dem einzelnen Gerät den Zugriff auf das Medium zuzuteilen. Zugriffsmethoden sind eng mit der jeweiligen Topologie verknüpft. Ihr Ziel ist es, in geordneter Weise die Übertragung von Daten auch in einer Situation zu sichern, wo mehrere Stationen darum konkurrieren, Zugriff auf das Netz zu erhalten. Das kann sehr geordnet oder auch chaotisch erfolgen, Hauptsache alle Stationen halten sich an die Spielregeln und es gehen keine Daten bei der Übertragung verloren. Die bekanntesten Zugriffsmethoden sind:

- *CSMA/CD*
- *CSMA/CA*
- *Token Passing*
- *Demand Priority*

CSMA/CD



CSMA/CD steht für den Zungenbrecher Carrier Sense Multiple Access with Collision Detection. Diese Zugriffsmethode geht von einer Bustopologie aus. Wenn einer sendet, können ihn alle hören. Möchte eine Station senden, so hört sie zuerst das Medium ab, ob ein Signal (carrier) anliegt. Ist das nicht der Fall, setzt die Station ihr Paket ab. Nun kann es aber passieren, daß zwei (oder mehr) Stationen gleichzeitig senden, weil sie das Medium als frei erkannt haben. Dann kommt es zu einer sogenannten Kollision. Die Kollision wird von beiden Stationen erkannt, und beide unterbrechen sofort ihre Sendung. Dann wartet jede Station eine durch einen Zufallsgenerator bestimmte Zeit und versucht den Zugriff erneut. Die Zufälligkeit der Wartezeit sorgt dafür, daß die Stationen beim nächsten Versuch nicht wieder eine Kollision erzeugen. CSMA/CD ist die am meisten verbreitete Zugriffsmethode. Sie kommt in Ethernet-Netzen zur Anwendung.

Sämtliche Texte stammen aus dem Skript von Prof. Kories, wobei jedoch im Originaltext z.B. zwischen dem ersten und zweiten Text über 100 Absätze und zwischen dem zweiten und dritten Text mehr als 10 Absätze lagen. Die zugeordneten Grafiken entstammen verschiedenen Powerpoint-Folien der Telekom AG.

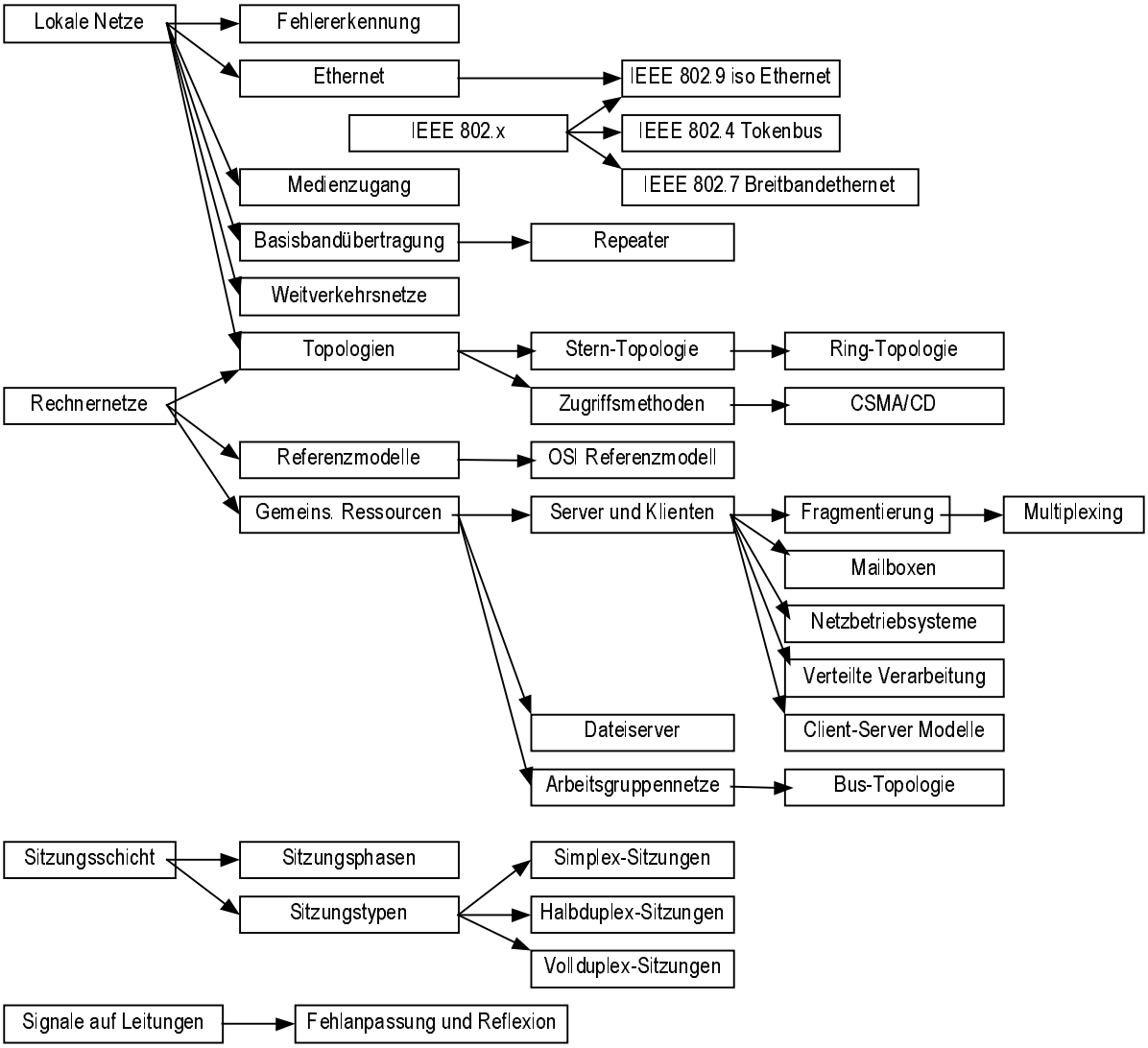
9.3 Vermittlung von umfangreichen Stichwort-Mengen

Falls die Menge der zu vermittelnden Stichworte klein war (weniger als 5 Stichworte) wurden in der überwiegenden Zahl der durchgeführten Tests gute Resultate erzielt. Die meisten der erzeugten Dokumente bestanden aus zumeist stark sequentiellen Medienobjekt- und Seitenstrukturen und enthielten nur sehr wenig inhaltliche Redundanzen. Das Ergebnis des oben beschriebenen Testfalls zur Vermittlung eines einzigen Stichworts ist in diesem Sinne durchaus repräsentativ für Struktur und Qualität der erzeugten Dokumente.

Den genau entgegengesetzten Grenzfall stellen Szenarien dar, in denen sehr viele oder gar alle vermittelbaren Stichworte von dem erzeugten Dokument abgedeckt werden sollen. Zur Simulation dieses Szenarios wurde ein Test durchgeführt, in dem aus den Medienobjekten des Telekom-Skript ein Dokument erzeugt werden sollte, das alle vermittelbaren Inhalte des Skriptes abdeckt.

Das Ergebnis dieses Test war ein Dokument, das mit ca. der Hälfte der verfügbaren Medienobjekte alle vermittelbaren Inhalte abdeckte. Die nachfolgende Grafik gibt die Seitenstruktur des generierten

Dokumentes wieder, wobei alle zusätzlich zwischen den Seiten verlaufenden Hyperlinks aus Gründen der Übersichtlichkeit nicht dargestellt wurden:



Wie aus der Abbildung zu ersehen ist, wurden 4 Kapitel erzeugt, die mit den Themen „Lokale Netze“, „Rechnernetze“, „Sitzungsschicht“ und „Signale auf Leitungen“ beginnen. Die Struktur ist weitgehend hierarchisch; lediglich zwei Knoten der Seitenstruktur besitzen mehr als einen Vorgänger („Topologien“ und „IEEE 802.9 iso Ethernet“).

Einige der erzeugten Substrukturen erscheinen etwas ungewöhnlich, da sie augenscheinlich recht weit von der Struktur des Themas entfernt sind. Einige dieser Substrukturen sind jedoch bei näherer Betrachtung durchaus schlüssig, während andere auf schlechte Indizierung oder gar Schwächen der Strukturierungsalgorithmen zurückzuführen sind. Im folgenden sollen daher einige der in Lehrbüchern normalerweise anders gewählten Strukturierungen näher betrachtet werden.

Die auf der obersten Hierarchieebene vorgenommene Aufspaltung des Inhaltes in „Lokale Netze“ und „Rechnernetze“ hat ihren Ursprung in einer ungenauen Wortwahl im Ausgangsmaterial, die in die Indizierung übernommen wurde. Das Skript beginnt mit einem groben Überblick über das Thema „Netzwerke“ wobei oftmals sehr allgemein von Netzen gesprochen wird. In den anschließenden Kapiteln werden die im Eingangskapitel angesprochenen Themen weiter vertieft, wobei hier der Fokus sehr stark auf lokalen Netzen liegt. Die Verwendung des Index-Wizard hat dazu geführt, daß Medienobjekte aus dem ersten Kapitel oftmals „Netz“ als Teil des benötigten Vorwissens enthalten, während fast alle Fragmente aus den nachfolgenden Kapiteln das Stichwort „lokales Netz“ referenzieren. Die didaktisch

motivierte und durchaus sinnvolle Verwendung des allgemeinen Begriffs „Netz“ im Skript auch für Fälle, in denen ausschließlich lokale Netze gemeint sind, führt zur impliziten Modellierung einer „is-a“-Beziehung als Synonym. Bei der Indizierung hätte jedoch viel genauer geprüft werden müssen, ob der Inhalt eines Medienobjekts für alle Arten von Netzen oder nur für lokale Netze gültig ist.

Im ersten Kapitel schließt sich an das einführende Medienobjekt zu Grundlagen lokaler Netze als erstes ein Medienobjekt zur Fehlererkennung an. Dies ist bei näherer Betrachtung der beiden Objekte akzeptabel, da in beiden Medienobjekten Informationen zu Protokollen vermittelt werden, wobei bezogen auf dieses Stichwort das zweite Medienobjekt den Inhalt des ersten weiter vertieft. Aus dem selben rund ist auch die Sequenz aus „Basisbandübertragung“ und „Repeater“ legitim; beide Objekte haben die Abschwächung und Wiederauffrischung von Signalen zum zentralen Thema.

Ungewöhnlich ist sicherlich auch die Zuordnung des Medienobjekts mit dem Titel „Bus-Topologie“ zum Objekt über Arbeitsgruppennetze. Betrachtet man sich die Texte genauer, zeigt sich jedoch ein guter Textfluß:

> *In Arbeitsgruppennetzen (peer-to-peer networks) ist jeder beteiligte Rechner gleichberechtigt. Die Arbeitsgruppe sollte nicht mehr als 10 Rechner umfassen. Die Mitglieder der Arbeitsgruppe können anderen die Ressourcen ihres Rechners zur Verfügung stellen. Die Verantwortung für Datensicherheit und Datenschutz liegt bei jedem einzelnen Teilnehmer der Arbeitsgruppe.*
>
> *In den meisten kleinen Arbeitsgruppennetzen sind die Rechner gemäß der Bus-Topologie miteinander verbunden. Eine einfache Koaxialleitung wird von Rechner zu Rechner gezogen und so die Verbindung untereinander hergestellt. Da jede Station am Buskabel wie die Rippen an der Wirbelsäule hängen, wird eine solche Anordnung im Englischen oft plastisch als backbone bezeichnet. Im Deutschen ist der Name Segment geläufig.*

Das Problem hier ist somit nicht der Inhalt, sondern die Überschrift des zweiten Medienobjekts.

Eine andere unerwartete Strukturierung hingegen hat ihrer Ursprung in den verwendeten Algorithmen: Die Medienobjekte zur Sitzungsschicht bilden zusammen mit denen zu Sitzungsphasen und Sitzungstypen ein eigenes Kapitel, da die in diesem Block enthaltenen wohlsstrukturierten Subgraphen zu einem so starken “Zusammenhalt” der Objekte untereinander führen, daß alle Verbindungen zu anderen Medienobjekten dagegen sehr schwach erscheinen. Für den Strukturierungsalgorithmus ist so ein hochgradig integriertes “Cluster” entstanden, das aufgrund seiner relativ schwachen Außenbindung als seperates Kapitel abgebildet wurde.

Die ungewöhnlichste Substruktur ist die Sequenz von “Server und Klienten”, “Fragmentierung” und “Multiplexing”. Nachfolgend sind die Inhalte dieser drei Medienobjekte als ein Text mit drei Absätzen dargestellt:

> *Ausgehend von einem kleinen Netz bestehend aus zwei oder drei Rechnern wird man schnell bemerken, daß es unwirtschaftlich ist, für jeden Rechner einen Drucker, ein Fax-Modem oder einen Plotter zu beschaffen. Abgesehen von den höheren Kosten führt diese Vorgehensweise auch zu uneinheitlichen Ausdrucken. Glücklicherweise erlauben es heutige Betriebsysteme solche Ressourcen zu teilen. Der Rechner, der die Ressource zur Verfügung stellt, heißt Server, der Rechner oder besser die Anwendung (das Programm), das sich der Ressource bedient, heißt Client.*
>
> *Sie wollen eine Datei vom Server in Ihre Textverarbeitung auf dem Clientenrechner übertragen. Die Datenmenge ist viel zu groß, um sie in einem Rutsch über das Netz zu übertragen. Folglich muß die Nachricht in kleine handhabbare Pakete aufgeteilt werden. Diesen Vorgang bezeichnet man als Fragmentierung. Das bedeutet natürlich, daß der Empfänger diese Daten wieder zusammensetzen muß. Dabei muß sichergestellt sein, daß er das in der richtigen Reihenfolge tut. Wie wir noch sehen werden, garantieren Rechnernetze nicht ohne weiteres, daß die Pakete in der Reihenfolge eintreffen, in der sie ausgesandt wurden. Also müssen die Pakete geeignet numeriert sein, schon deshalb, damit der Empfänger feststellen kann, daß eines verloren gegangen ist. Dieser Vorgang heißt Sequenzierung.*
>
> *Die gebräuchlichste Art der Transportverbindung ist die einer virtuellen, fehlerfreien Punkt-zu-Punkt-Verbindung, auf der die Nachrichten in ihrer Sendereihenfolge übertragen werden. Weitere Varianten sind Rundsendungen mit ausgesuchten Zielen (multicast) oder an alle im Netz (broadcast). Die Transportschicht erfüllt alle Funktionen, die für den Aufbau, die Kontrolle und den Abbruch einer Verbindung zwischen zwei Teilnehmern notwendig sind. In der Transportschicht können mehrere Teilnehmerverbindungen mit dem gleichem Ziel zusammengefaßt werden (Multiplexen), oder einzelne Verbindungen auf mehrere Verbindungen aufgeteilt werden (inverses Multiplexen). In der Transportschicht erfolgt die Wiederherstellung der korrekten Ordnung der Pakete, falls sie auf dem*

Übertragungswege ihre Reihenfolge geändert haben sollten. Zudem fordert die Transportschicht die Transportschicht der anderen Seite auf, die Sendung eines Paketes zu wiederholen, falls es verlorengegangen ist.

Während die Verbindung der beiden ersten Objekte durchaus akzeptabel ist, passen die Objekte zur Fragmentierung und zum Multiplexing nicht sonderlich gut zueinander. Der Grund, daß sie vom System dennoch in dieser Form strukturiert wurden, ist ein minimaler Fehler bei der Indizierung des dritten Medienobjekts: Dem Stichwort "Sequenzierung" wurde die gleiche Relevanz zugewiesen wie den Stichworten "Transportschicht" und "Multiplexing". Dies ist so nicht korrekt, denn der Text handelt im wesentlichen von den Aufgaben der Transportschicht, alle anderen Stichworte besitzen eine deutlich geringere Relevanz. Durch den Fehler in der Gewichtung wird dieses Medienobjekt vom System jedoch als Text zur Sequenzierung interpretiert, der einen bereits existierenden Text zu diesem Thema wunderbar um die Aspekte "Transportschicht" und "Sequenzierung" zu erweitern scheint.

Diese zum Teil fehlerhaften Strukturierungen sollen jedoch nicht den Eindruck entstehen lassen, als ob das erzeugte Dokument vollständig willkürlich zusammengesetzt und damit unbrauchbar wäre. Im Gegenteil: Der überwiegende Teil der Subgraphen paßt inhaltlich gut zueinander, obwohl die dort zusammengeführten Medienobjekte oftmals aus weit auseinander liegenden Abschnitten des Ausgangsmaterials extrahiert wurden. Bemerkenswert gut ist auch die Struktur des Dokuments: aus einem stark vernetzten, hochgradig mit Zyklen durchsetzten Graphen wurde eine Struktur berechnet, die mit zwei Ausnahmen komplett hierarchisch aufgebaut ist. Ähnliche Resultate – kleine Schwächen in der inhaltlichen Anordnung bei ausgezeichneter Strukturierung – ließen sich auch in den meisten anderen Test mit umfangreichen Lernzielen feststellen. Einer der Gründe hierfür ist sicherlich, daß die Bewertungsfunktionen und Algorithmen, die mit der reinen Strukturierung zusammenhängen, bereits sehr komplex und ausgereift sind, während die Modellierung und adäquate Berücksichtigung inhaltlicher Aspekte zuweilen etwas zu generisch behandelt wurde.

10 Zusammenfassung und Ausblick

Wenn ich einen Urlaub an der Südwestküste der Türkei plane, wäre es durchaus möglich, daß ich keinen Dokumentarfilm über Bodrum finde. Aber beim National Geographic, dem PBS, der BBC und Hunderten anderer Sender erhalte ich Filmausschnitte über traditionellen Schiffsbau, den Fischfang bei Nacht, über antike Unterwasserfunde, das Babaganoush und orientalische Teppiche. Diese Fragmente könnte man zu einem Gesamtbild zusammenfügen, das meinem speziellen Bedarf entspricht. Ich würde zwar keinen Oscar für den besten Dokumentarfilm gewinnen, aber darum geht es auch gar nicht.

Nicolas Negroponte, MIT Media Lab

In dieser Arbeit wurden Motivation, Design und Implementierung des *i4 Frameworks*, eines Systems zur automatisierten Erstellung web-basierter, wissensvermittelnder Dokumente, beschrieben. Die zugrundeliegende Aufgabenstellung - die dynamische Generierung von Dokumenten - ist nicht neu und wird von einer ganzen Reihe von Forschungsprojekten bearbeitet.

Neu an dem in dieser Arbeit beschriebenen System ist, daß das gewünschte Dokument nicht ausgehend von einer übergeordneten, semantischen Struktur erstellt wird (*Top-Down*), sondern auf Basis existierender Fragmente (*Bottom-Up*). Die Vorteile dieser Vorgehensweise sind vielfältig:

- schnellere und damit auch billigere Erstellung von Dokumenten durch Verwendung existierender Fragmente (Medienobjekte),
- bessere Adaptierbarkeit durch Verzicht auf statische und semantische Strukturen,
- einfachere Wartbarkeit durch Verzicht auf statische Verbindungen zwischen den Fragmenten.

Durch die in Kapitel 5 beschriebene doppelte Adaptierbarkeit ist das *i4 Framework* darüber hinaus eines der wenigen Systeme zur automatisierten Erstellung von Lehr- und Informationssystemen, das adaptierbare Dokumente unter Berücksichtigung thematisch bedingter Besonderheiten erstellen kann. Hierzu zählen vor allem die Möglichkeit, unterschiedliche Dokumentstrukturen zu erzeugen, sowie die Einbindung und Berücksichtigung themenbezogener Attribute, Heuristiken und Bewertungen.

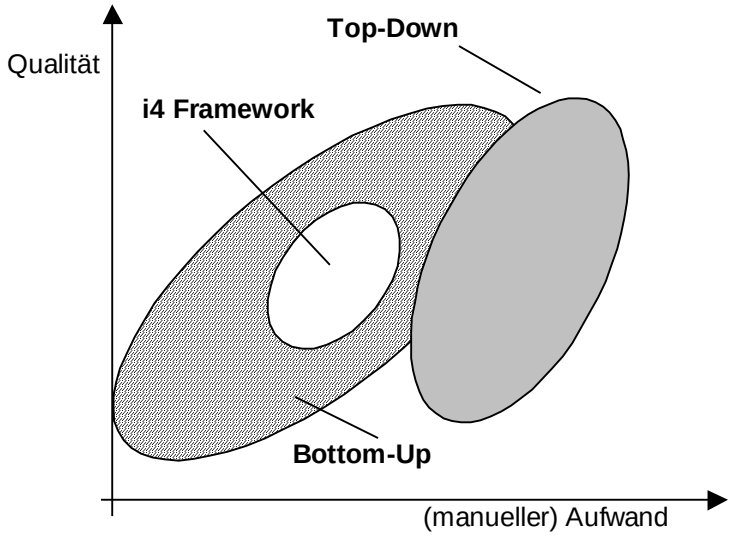
Eine der herausragenden Besonderheiten des verwendeten *Bottom-Up* Ansatzes im Vergleich zu den weit verbreiteten *Top-Down* Systemen ist die Gewichtung von Algorithmen und Datenstrukturen. *Top-Down* orientierte Systeme setzen hauptsächlich auf starke, expressive Datenstrukturen (z.B. semantische Netze oder hierarchische Wissensbasen), die eine Vielzahl umfassender, semantischer Informationen kodieren. Die Algorithmen zur *Top-Down* Erzeugung von Dokumenten sind hingegen eher einfach, da alle benötigten Informationen über Struktur und Inhalt vom Autor bereitgestellt werden (müssen).

Im Gegensatz dazu liegt der Schwerpunkt der *Bottom-Up* Generierung auf der Berücksichtigung komplexer Heuristiken und Algorithmen. Hiermit - und durch den Einsatz von *Information Retrieval* und *Data Mining* Techniken - wird versucht, den Mangel an in den verwendeten Datenstrukturen enthaltenen semantischen Informationen auszugleichen.

Die Auswirkungen der unterschiedlichen Gewichtung von (manuell) vorgegebenen semantischen Informationen liegen auf der Hand: Je mehr gesicherte semantische Informationen zur Beschreibung des zu erzeugenden Dokuments vorhanden sind, um so besser wird das generierte Lehr- oder Informationssystem bezüglich seiner Gliederung und inhaltlichen Konsistenz sein. Auf der anderen Seite gilt jedoch, daß die Dokumentenerstellung um so schneller, einfacher und dadurch automatisierbarer und billiger ist, je weniger handcodierte, semantische Informationen benötigt werden.

Diese Überlegungen führen zu einem weiteren Vorteil von *Bottom-Up* Ansätzen: Skalierbarkeit. Durch die Notwendigkeit einer semantischen und strukturellen Beschreibung sowohl des behandelten Themas wie auch des zu erzeugenden Dokuments ist der manuelle Aufwand für Erstellung und Wartung bei *Top-Down* Verfahren immer relativ hoch. Bei *Bottom-Up* basierten Ansätzen hingegen können für die Beschreibung der Medienobjekte sowohl manuelle als auch automatische IR- oder KI-Verfahren eingesetzt werden. Hierdurch ist die Qualität des erzeugten Dokuments in einem sehr weiten Bereich über den vom Autor zu leistenden Aufwand skalierbar (Abbildung 10-1).

Abbildung 10-1: Skalierbarkeit der Qualität bei Top-Down und Bottom-Up basierten Generierungssystemen. Die Variable "Aufwand" gibt die Kosten für Erstellung UND Wartung eines Dokuments wieder.



Beim Vergleich der beiden Ansätze *Top-Down* und *Bottom-Up* sollte jedoch nicht vergessen werden, daß prinzipiell *Top-Down* Verfahren zu besser strukturierten Dokumenten führen. Der Grund hierfür ist, daß die *Bottom-Up* Auswahl- und Strukturierungsalgorithmen immer (zumindest teilweise) auf Heuristiken beruhen, die eine weitgehend manuellen Strukturierung durch ein vorgegebenes semantisches Netz nur annähernd gleichwertig ersetzen können.

Das in dieser Arbeit vorgestellte *i4 Framework* ist bezüglich Aufwand und Qualität der erzeugten Dokumente ungefähr in der Mitte des von *Bottom-Up* Systemen besetzten Bereichs angesiedelt. Je nachdem, welche der in den Kapiteln 4.5 und 8.3 beschriebenen Verfahren der künstlichen Intelligenz und des *Information Retrieval* zur Autorenunterstützung genutzt werden, variieren Aufwand und Qualität in starkem Maße.

In diesem Kapitel soll nach einer kurzen Zusammenfassung des automatisierten *Bottom-Up* Entwicklungszyklus untersucht werden, welche Erweiterungen des *i4 Frameworks* nötig sind, um seine Position innerhalb des in Abbildung 10-1 angegebenen Koordinatennetzes zu verändern. Für ein reines Autorensystem wäre es z.B. durchaus akzeptabel (eventuell sogar wünschenswert), dem Autor ein Mehr an manuellem Aufwand abzuverlangen, wenn dafür im Gegenzug die Qualität der generierten Dokumente steigt. Im Gegensatz dazu ist für viele andere Anwendungsbereiche virtueller Dokumente - insbesondere in Verbindung mit Suchmaschinen - eine weitergehende Automatisierung zwingend erforderlich, auch wenn dies zu Lasten der Qualität gehen würde.

10.1 Zusammenfassung

Das in dieser Arbeit beschriebene *i4 Framework* zur automatisierten Erstellung hypermedialer Dokumente geht von einer drei-schichtigen Architektur von Lehr- und Informationssystemen aus:

- Auf der obersten Ebene besteht jedes Lehr- oder Informationssystem aus Blöcken verschiedener Genres,
- Jeder Block bildet eine Hierarchie von Kapiteln ab.
- Jedes Kapitel besteht aus einer Folge von Bildschirmseiten,
- die ihrerseits aus einer (geordneten) Menge von Medienobjekten - Texten, Grafiken, Videos, etc. - zusammengesetzt sind.

Jede dieser drei Ebenen ist eine Repräsentation der Struktur des Lehr- oder Informationssystems, wobei diese Struktur auf der untersten - der Medienobjektebene - am detailliertesten und auf der oberen - der Kapitelebene - am abstraktesten erkennbar ist.

Grundidee des *i4 Frameworks* ist, bei der Erstellung eines Dokuments nicht mit der Kapitelstruktur zu beginnen (*Top-Down*), sondern mit der Medienobjektstruktur. Dieses Verfahren wird als *Bottom-Up* Ansatz bezeichnet, da ausgehend von der niedrigsten, detailliertesten Struktur Schritt für Schritt die höheren Strukturen durch Zusammenfassung benachbarter Objekte abgeleitet werden.

Ausgangspunkt der *Bottom-Up* Generierung ist eine Menge von Medienobjekten und ein Nutzungsszenario, in dem die zu vermittelnden Inhalte und vom Nutzer festgelegte Einschränkungen und Präferenzen kodiert sind. Anhand des Nutzungsszenarios werden die geeignetsten Medienobjekte ausgewählt und strukturiert, wodurch die Medienobjektstruktur des zu erzeugenden Dokuments entsteht. Durch Zusammenfassen von Medienobjekten zu Seiten wird aus der Medienobjektstruktur die Seitenstruktur des Dokument abgeleitet, aus der anschließend durch Gliederung und Zusammenfassung von Seiten die Kapitelstruktur ensteht. Abschließend werden die einzelnen Seiten - unter Berücksichtigung der Kapitelstruktur - als HTML Dateien im Dateisystem abgelegt.

10.1.1 Medienobjekte

Medienobjekte sind die kleinsten Bausteine von Lehr- und Informationssystemen. Es kann sich bei ihnen um Textfragmente, Grafiken, Videos, *Java Applets* und beliebige andere statische, kontinuierliche oder interaktive Medien handeln.

Jedes Medienobjekt wird über Attribute und einen Index aus Stichworten beschrieben. Während Attribute hauptsächlich Verwendbarkeit, Format und Quelle von Medienobjekten in Form von Name-Wert-Paaren kodieren, enthält der Index eines Medienobjekts eine Liste aller von dem Objekt referenzierten Stichworte. Stichworte werden über gewichtete Attribute in als Vorwissen benötigte und vermittelte Stichworte unterteilt.

Alle Medienobjekte sind selbstbeschreibend, d.h. weder in Attributen noch im Index wird Bezug auf andere Medienobjekte oder eine übergeordnete (semantische) Struktur genommen. Da keinerlei globale Strukturen wie z.B. semantische Netze existieren, kann die Menge der für die Dokumentenerzeugung zur Verfügung stehenden Medienobjekte sehr einfach erweitert und modifiziert werden.

Alle Attribute und Indizes der für die Dokumentengenerierung verfügbaren Medienobjekte sind in einer Datenbank, dem sog. Medienobjektspeicher, abgelegt.

10.1.2 Der Abhängigkeitsgraph

Zur dynamischen Erzeugung des vom Benutzer gewünschten Lehr- oder Informationssystems werden die im Medienobjektspeicher abgelegten Objekte über einen sog. Abhängigkeitsgraphen zueinander in Beziehung gesetzt. Über eine Verknüpfung der verlangten und vermittelten Stichworte entstehen dynamisch implizite "wird benötigt"-Beziehungen zwischen Medienobjekten.

Diese Beziehungen zwischen Medienobjekten allein reichen jedoch noch nicht aus, um eine effektive Auswahl und Strukturierung vorzunehmen. Da der Auswahlalgorithmus auf einer lokalen Bestensuche basiert, werden zusätzlich eine Reihe, die Struktur des Abhängigkeitsgraphen beschreibende Maßzahlen benötigt:

1. Die **Minimale Tiefe** eines Medienobjektes gibt die Länge des kürzesten Pfades von einem Wurzelobjekt zu diesem Medienobjekt an.
2. **Starke Zusammenhangskomponenten** beschreiben die im Abhängigkeitsgraphen enthaltenen Zyklen.
3. Die **Vorgängermenge** eines Medienobjekte bzw. Stichwortes enthält alle Medienobjekte, über die das Objekt erreichbar ist.
4. **Hierarchische Subgraphen** beschreiben sämtliche im Abhängigkeitsgraphen enthaltenen, baumstrukturierten Subgraphen.

Mit Hilfe dieser Maßzahlen können alle im Abhängigkeitsgraphen enthaltenen Zyklen aufgelöst und nicht erreichbare Knoten entfernt werden.

10.1.3 Auswahl und Strukturierung

Wichtigstes Ziel der Auswahl und Strukturierung von Medienobjekten ist es, innerhalb des Abhängigkeitsgraphen einen Subgraph zu finden, der sämtliche Vorgaben des Nutzers (bzw. des Autors) bezüglich Inhalt, Struktur, Kodierung, etc. erfüllt. Die Auswahl der den gesuchten Subgraphen bildenden Medienobjekte geschieht über eine lokale Bestensuche. Die dabei verwendete heuristische Bewertungsfunktion basiert auf der Wahrscheinlichkeit eines Medienobjektes, in dem gesuchten Subgraph enthalten zu sein. Bei jeder Iteration des Algorithmus wird das wahrscheinlichste Objekt ausgewählt und anschließend eine Neuberechnung aller Wahrscheinlichkeiten durchgeführt. In Analogie zu dem bekannten Computerspiel wird der Algorithmus als *Minesweeper*-Algorithmus bezeichnet.

Bei der heuristischen Bewertung der Knoten des Abhängigkeitsgraphen werden vier Kriterien berücksichtigt:

- Eigenschaften von Medienobjekten unabhängig von Inhalt und Struktur des gesuchten Subgraphen,
- Eignung eines Medienobjekts zur Vermittlung bestimmter, im gesuchten Subgraphen enthaltener Stichworte,
- Eignung des Medienobjekts hinsichtlich der vom Nutzer oder Autor gewünschten Struktur,
- Qualität des Medienobjekts in Bezug auf vom Autor festgelegte, themenabhängige Kriterien.

Der Auswahlalgorithmus ist mit einem Laufzeitverhalten von $O(\text{Medienobjekte} * \text{Indexeinträge})^2$ sehr aufwendig. Dieser hohe Aufwand ist jedoch vor allem für verhältnismäßig dichte, viele Redundanzen enthaltende Abhängigkeitsgraphen notwendig, um eine gute Grundlage für die anschließende Strukturierung zu schaffen.

Nachdem ein Subgraph gefunden ist, der die geforderten Bedingungen erfüllt, werden die in ihm enthaltenen Medienobjekte strukturiert. Hierbei wird im wesentlichen von den verbindenden Stichworten abstrahiert, wodurch der bipartite Abhängigkeitsgraph in einen "normalen", gerichteten Graph von Medienobjekten überführt wird.

Durch das anschließende Entfernen redundanter Knoten und Kanten, sowie die Einbettung von nicht-inhaltsvermittelnden Objekten wie z.B. Bildern, Aufgaben und Beispielen, entsteht die Medienobjekt-struktur des zu erzeugenden Dokuments. Aus der Medienobjektstruktur werden Seiten- und Kapitelstruktur abgeleitet. Hierbei werden abermals redundante Kanten entfernt bzw. durch Referenzen (Hyperlinks) ersetzt, so daß - falls gewünscht - ein hierarchisch strukturiertes Dokument entsteht.

10.1.4 Doppelte Adaptierbarkeit

Ein wichtiger Aspekt bei der automatisierten Generierung von Informationssystemen ist die Möglichkeit sowohl des Autors als auch des Benutzers, einzelne Objekte, Algorithmen und Funktionalitäten möglichst flexibel konfigurieren zu können. Beispiele für nutzerabhängige Einstellungen sind die gewünschten Inhalte, bevorzugte Medienkodierungen, die zur Verfügung stehende Bandbreite, etc. Beispiele für vom Autor festzulegende Einstellungen sind das Layout der erzeugten HTML-Seiten, Parametrisierungen einzelner Algorithmen, Beschreibungen neuer Medientypen und Genres, Default-Werte für nutzerabhängige Einstellungen, etc.

Aus diesen Anforderungen resultieren zwei Ebenen der Adaptierbarkeit: die Anpassung des Systems an ein Thema durch den Autor und die Anpassung des erzeugten Dokuments an den aktuellen Nutzer durch das System.

Das *i4 Framework* berücksichtigt diese Forderung nach doppelter Adaptierbarkeit durch die Möglichkeit, interne Implementierungen der verwendeten Objekte zu modifizieren oder komplett zu ersetzen. Hierdurch kann das *i4 Framework* an die Besonderheiten beliebiger Themen angepaßt werden. Eine zusätzliche Anfrage-Vorverarbeitung erlaubt darüber hinaus eine komplett vom Autor gesteuerte Abbildung von Nutzerparametern auf Systemparameter.

10.2 Offene Probleme

In den vorangegangenen Kapiteln wurden die wichtigsten Leistungsmerkmale der *Bottom-Up* Generierung hypermedialer Dokumente - Adaptierbarkeit, einfache Erstellung und problemlose Wartbarkeit der erzeugten Dokumente - mehrfach hervorgehoben. Es soll jedoch nicht verschwiegen werden, daß der *Bottom-Up* Ansatz in der hier beschriebenen Form durchaus auch Probleme mit sich bringt.

Strukturierung von Stichworten

Die Schwachstelle des *Bottom-Up* Entwicklungszyklus ist die fehlende semantische Struktur auf Stichworten. Einerseits ist der Verzicht auf jegliche übergeordnete, globale Struktur einer der großen Vorteile des *Bottom-Up* Ansatzes, insbesondere wenn es um die Wartbarkeit von Dokumenten geht. Andererseits führt jedoch insbesondere das Fehlen einer hierarchischen oder vernetzten Konzept-Struktur in einigen Fällen zu Problemen bei der automatisierten Auswahl und Strukturierung von Medienobjekten:

- Insbesondere wenn nur wenig Medienobjekte zur Verfügung stehen, ist eine automatische Bestimmung der Relevanz eines Medienobjektes und der von ihm vermittelten Stichworte für das behandelte Thema und die konkrete Nutzeranfrage nur schwer möglich. Da dieser Aspekt daher bei der Bewertung von Medienobjekten nur unzureichend berücksichtigt wird, kann es vorkommen, daß in einem generierten Dokument nebensächliche Details ausführlich behandelt werden, während wichtige Stichworte nur am Rande erklärt werden.
- Sammelbegriffe und andere "aggregierte" Konzepte müssen von der Anfragevorverarbeitung explizit in ihre Bestandteile aufgeschlüsselt werden. Fordert z.B. ein Benutzer ein Dokument zum Thema "Statistik" mit dem Lernziel "Lagemaße" an, muß die Anfragevorverarbeitung dieses Lernziel in eine Menge von Stichworten übertragen ("Lagemaße", "Mittelwert", "Modus", "Median", etc."). Dies bedeutet, daß zwar keine festen Strukturen innerhalb des Medienobjektspeichers existieren, dafür aber statische, semantische Informationen in den für die Dokumentenerzeugung benötigten *Java Wrappern* kodiert sind. Das Problem der unflexiblen, statischen Strukturen wäre somit lediglich verlagert, aber nicht endgültig gelöst.
- Ein weiteres Problem aggregierter Konzepte ist, daß sie zuweilen zu inhaltlich unausgewogenen oder gar unvollständigen Dokumenten führen. Im oben angegebenen Beispiel der Lagemaße könnte es durchaus passieren, daß zwar alle drei genannten Lagemaße eingeführt werden, aufgrund inhaltlicher oder struktureller Bewertungen jedoch nur der Modus als eigentlich eher unwichtiges Lagemaß im Detail erklärt wird.

- Oft dienen (vor allem textuelle) Medienobjekte der Hinführung zu einem Thema. Bei einem Medienobjekt wie "Lagemaße beschreiben das Niveau eines Datensatzes. Die wichtigsten Lagemaße sind das arithmetische Mittel (oder Mittelwert) und der Median." erwartet der Leser, daß als nächstes nähere Informationen zu den Stichworten "Mittelwert" und "Median" folgen. Leider kann diese Erwartungshaltung jedoch mit den verwendeten Mechanismen zur Beschreibung von Medienobjekten und Indizes nur ungenügend ausgedrückt werden[50].

Um diese Probleme zu lösen, müssen den Auswahl- und Strukturierungsalgorithmen zusätzliche Informationen über die semantischen Abhängigkeiten zwischen Stichworten zur Verfügung gestellt werden.

Die offensichtlichste Lösung der genannten Probleme wäre die (manuelle) Strukturierung der in den Medienobjekten referenzierten Stichworte über ein semantisches Netz. Da jedoch insbesondere hierarchische Stichwort-Strukturen sowie "ist-ein" und "ist-Teil-von" Beziehungen zwischen Stichworten interessieren, würde wahrscheinlich auch ein semantisch stark eingeschränktes propositionales Netz oder ein systematischer Thesaurus [Gastmeyer92] genügen.

Dies würde jedoch zu einer globalen, semantischen Struktur führen, die nicht nur manuell erstellt, sondern auch gewartet werden müßte. Ferner wären Medienobjekte nicht mehr selbstbeschreibend, wodurch das einfache Hinzufügen von Medienobjekten zum Medienobjektspeicher in der beschriebenen Form nicht mehr möglich wäre.

Insbesondere um den selbstbeschreibenden Charakter von Medienobjekten zu erhalten, ist geplant, daß zukünftige Weiterentwicklungen des Medienobjekt-Managers keine vorgegebenen semantischen Netze enthalten, sondern statt dessen zusätzliche Attribute auf Index-Einträgen einführen.

Mit Hilfe solcher attributierten Indizes können viele der genannten Probleme gelöst werden. Der wichtigste Aspekt hierbei ist, daß die zusätzlichen Informationen nicht - wie z.B. bei einem Semantischen Netz - semantische, sondern verwendungstechnische Aspekte von Indizes beschreiben. Dies entspricht somit auch der dem *Bottom-Up* Ansatz zugrundeliegenden Philosophie.

Behandlung von Redundanzen

Ein weiteres Problem der *Bottom-Up* Generierung betrifft die Forderung nach minimaler Abgeschlossenheit des der Medienobjektstruktur zugrundeliegenden Abhängigkeitsgraphen G^{b*}. Die minimale Abgeschlossenheit führt zu weitgehend redundanzfreien Dokumenten; eine Eigenschaft, die jedoch oftmals nicht erwünscht ist.

Insbesondere für Lehrsysteme kann die redundante Vermittlung von Informationen durchaus sinnvoll und instruktiv sein:

- Wichtige Informationen werden an verschiedenen Stellen des Dokuments erklärt, um einen verstärkenden Effekt zu erzielen.
- Komplizierte Sachverhalte werden auf verschiedene Weise und eventuell mit verschiedenen Medien vermittelt. Die Intention dabei ist, daß sich die bei jeder Erklärung verstandenen Teilinformationen zu einem Gesamtbild zusammenfügen.

Um diese Formen der Redundanz zuzulassen, muß nicht nur die Umwandlung des Abhängigkeitsgraphen in eine Medienobjektstruktur, sondern der komplette Auswahlalgorithmus modifiziert werden. Ein offenes Problem ist hierbei vor allem, wie bei der Auswahl der geeignetsten Medienobjekte zwischen didaktisch sinnvollen Redundanzen und störenden Wiederholungen unterschieden werden kann.

[50] Um dieses relativ häufig auftretende Problem zu lösen, wurde das Medienobjekt-Genre "Dispatcher" (siehe Kapitel 3.3.4) eingeführt. Die Auswahl- und Strukturierungsalgorithmen fügen implizit alle vermittelten Stichworte eines "Dispatchers" zu den offenen Stichworten hinzu.

Vorgabe einer Kapitelstruktur

Das *i4 Framework* in der beschriebenen Form kann nicht nur zur *on-th-fly* Generierung von adaptierten Dokumenten, sondern auch als *offline*-Autorensystem eingesetzt werden. In diesem Fall werden alle bei der Generierung zu beachteten Vorgaben vom Autor festgelegt und das so erzeugte Dokument ohne weitere Möglichkeit der dynamischen Anpassung im *World Wide Web* bereitgestellt.

Bei einem solchen Einsatz des *i4 Frameworks* als reines Autorenwerkzeug werden viele Leistungsmerkmale der *Bottom-Up* Generierung nicht benötigt bzw. können mit geringen Aufwand auf den Autor übertragen werden. Hierzu zählen z.B. die Kapitelstrukturierung und eine Vorauswahl der zu verwendenden Medienobjekte:

- Beim Einsatz des *i4 Frameworks* als Autorensystem zur Generierung nicht dynamisch adaptierbarer Dokumente sollte der Autor die Möglichkeit haben, eine Kapitelstruktur vorzugeben, an der sich Auswahl und Strukturierung von Medienobjekten orientieren. Die einzige hierzu notwendige Änderung des *i4 Frameworks* ist, daß ein Dokument mehrere Blöcke gleichen Genres enthalten können muß. Anpassungen an den verwendeten Algorithmen sind nicht nötig.
- Wenn vom Autor selbst erstellte Medienobjekte die Basis der Dokumentenerzeugung bilden und Adaptierbarkeit kein zu berücksichtigender Aspekt ist, kann die Auswahl der zu verwendenden Medienobjekte zumindest zum Teil durch den Autor vorgenommen werden. Hierdurch wird dem Autor die Sicherheit gegeben, daß bestimmte, seiner Meinung nach wichtige Medienobjekte auf jeden Fall Bestandteil des erzeugten Dokuments sind. Auch diese Erweiterung ist ohne Modifikation der Auswahl- und Strukturierungsalgorithmen durch die Möglichkeit einer Vorinitialisierung von M^{b+} zu realisieren.

Beide Erweiterungen - Vorgabe einer Kapitelstruktur und Vorauswahl von Medienobjekten - sind im aktuellen Prototyp des *i4 Frameworks* testweise realisiert. Die bisherigen Ergebnisse lassen den Schluß zu, daß der damit verbundene Mehraufwand für den Autor durch eine Steigerung der Qualität der erzeugten Dokumente zu rechtfertigen ist. Hierzu tragen vor allem die manuell vorgegebenen Kapitelstrukturen bei, mit denen eine erheblich bessere Gliederung des behandelten Themas möglich ist, als mit automatisch erstellten logischen Strukturen.

10.3 Ausblick

Nachdem im vorherigen Abschnitt Nachteile und Probleme des *Bottom-Up* Ansatzes thematisiert wurden, sollen zum Ende der Arbeit wieder die Vorteile im Mittelpunkt stehen. Aus dieser Kategorie wurden bisher vor allem die Aspekte "Wartbarkeit" und "Adaptierbarkeit" hervorgehoben, während andere Vorteile, wie z.B. die Möglichkeit einer weitgehenden Automatisierung der Dokumentenerstellung, eher am Rande erwähnt wurden.

In diesem Abschnitt steht der letztgenannte Aspekt, die Automatisierung, im Vordergrund. Es soll der Frage nachgegangen werden, wie mit wenig (im Endeffekt sogar ohne) manuellen Aufwand immer noch eine akzeptable Qualität der generierten Dokumente erzielt werden kann[51].

Alle vom *i4 Framework* benötigten Meta Daten über Medienobjekte, Stichworte und Dokumente verzichten weitgehend auf die Beschreibung semantischer Aspekte. Der Hauptgrund für diese Beschränkung ist, daß nur so die Hoffnung besteht, daß alle für die Dokumentenerzeugung benötigten Informationen irgendwann einmal automatisch ermittelt werden können.

Die nachfolgende Grafik stellt die aktuelle Architektur des *i4 Frameworks* (links) einer imaginären - auf dem *i4 Framework* basierenden - Architektur zur vollautomatischen Generierung dynamischer Dokumente (rechts) gegenüber.

[51] Eine vollkommen automatische Erzeugung von Hypertexten ist momentan nur für die Vernetzung von Nachrichten realisierbar [Dalamagas+97, Green98, Södergård+99]. Hierbei werden jedoch Lernziele sowie didaktische und strukturelle Aspekte nicht berücksichtigt. Darüber hinaus wird von einer weitgehend redundanz-freien Datenbasis ausgegangen.

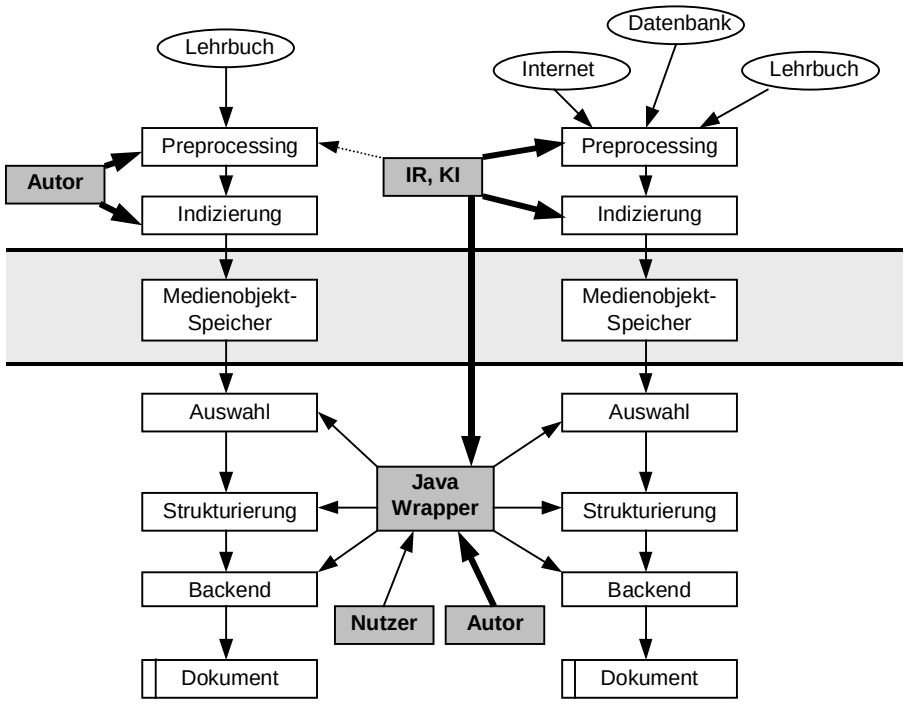


Abbildung 10-2: Ersetzen des Autors durch eine Retrieval-Komponente

Beide Architekturen erstellen Dokumente nach dem in dieser Arbeit beschriebenen *Bottom-Up* Ansatz und lassen sich in drei Komponenten unterteilen:

1. **Medienobjekt Retrieval**: Ermittlung, Extraktion, Beschreibung und Indizierung der zur Verfügung stehenden Medienobjekte.
2. **Medienobjektspeicher**: Speicherung von Medienobjekten und Meta Daten in einer Datenbank.
3. **Medienobjekt Integration**: Auswahl und Strukturierung von Medienobjekten mit Hilfe der in dieser Arbeit beschriebenen Algorithmen.

Mit dem aktuellen Stand der Technik kann der komplette Teilbereich der Medienobjekt Integration vollständig automatisch durchgeführt werden. Auch für die zusammenfassende Darstellung von Medienobjekten auf HTML-Seiten existieren bereits verschiedene Algorithmen [Graf97, Veijonen+94], so daß auch auf die manuelle Erstellung von diesbezüglichen Java-Beschreibungen verzichtet werden kann. Alle zwingend manuell vorzugebenden Informationen konzentrieren sich somit im Wesentlichen auf das Medienobjekt Retrieval.

Eine besondere Rolle kommt in dieser Architektur dem Medienobjektspeicher zu, da er die Schnittstelle zwischen der vollständig automatisierbaren Medienobjekt Integration und dem zur Zeit noch nicht automatiserbaren Medienobjekt Retrieval darstellt:

- Alle manuellen Eingriffe des Autors in das Framework finden im Medienobjektspeicher statt, um sie vor der Medienobjekt Integration zu verbergen. Hierdurch ist ein transparenter Austausch von manuellen Konfigurationen durch IR-Algorithmen möglich [Caumanns99a].
- Der Medienobjektspeicher bildet den Anknüpfpunkt für alle automatischen Verfahren zur Beschreibung und Indizierung von Medienobjekten, d.h. seine Schnittstellen und Datenstrukturen sowie deren Semantik müssen von allen verwendeten IR- und KI-Mechanismen berücksichtigt werden.

Durch die Trennung von Analyse (Medienobjekt Retrieval) und Synthese (Medienobjekt Integration) bildet ein *Bottom-Up* Verfahren, wie es in dieser Arbeit beschrieben wurde, eine gute Basis für zukünftige, vollständig automatisierte dokumentenerzeugende Systeme. Die noch zu lösende Aufgabe besteht allein darin, für eine vorgegebene Anfrage oder ein vorgegebenes Thema alle relevanten Dokumente zu ermitteln, in Medienobjekte zu zerlegen und diese zu indizieren. Und auch diese Aufgabe läßt sich bereits teilweise mit existierenden Verfahren lösen:

- Über Suchmaschinen, automatisch erstellte Kataloge [Chakrabarti+98] und ähnliche Mechanismen können ausreichend Texte, Grafiken und sonstige Medien ermittelt werden, die für ein vorgegebenes Thema relevant sind. Über eine Kontextanalyse ist dabei insbesondere für hochgradig vernetzte Dokumente eine akzeptable Präzision erreichbar [Brandman+99]. Auch das Retrieval von Grafiken und Fotos anhand vorgegebener Stichworte steht an der Schwelle zur Nutzbarkeit [Swain99].
- Über (automatische) *Clustering* Verfahren und Satzanalysen können für ein Thema relevante Stichworte ermittelt werden [Grefenstette94]. Zusätzlich ist auch eine Klassifizierung und Vorstrukturierung der gefundenen Dokumente möglich [Fürnkranz+98, Fürnkranz99, Sahami+98].
- Regelmäßig strukturierte Dokumente können automatisch in kleinere Einheiten (Medienobjekte) zerlegt werden. Darüber hinaus existieren auch im Bereich des *Information Retrieval* Projekte, die versuchen, aus vorhandenen Dokumenten lediglich die für eine Anfrage relevanten Textblöcke, Grafiken, etc. zu extrahieren (z.B. [Rölleke+97]).
- Videos und Animationen können automatisch in Sequenzen unterteilt werden [Boykin+99, Pong+99]. Hierdurch ist auch eine Extraktion von Medienobjekten aus kontinuierlichen Medien möglich.

Das einzige wirklich harte Problem ist somit die automatische Indizierung von Medienobjekten. Ist es möglich, auch dieses Problem zu lösen, sind komplett dynamisch erzeugte Dokuments, wie sie im Eingangszitat dieses Kapitels beschrieben sind, nicht länger eine Vision.

10.4 Was bleibt?

Das *i4 Framework* ist natürlich nur eines von vielen existierenden und noch kommenden Systemen zur Erstellung dynamischer Dokumente. Es wäre vermessen und unrealistisch zu glauben, daß die in dieser Arbeit beschriebenen Algorithmen und Datenstrukturen für längere Zeit den aktuellen Stand der Technik repräsentieren.

Im Grunde genommen ist das auch gut so, denn nur durch neue Ideen und die Abkehr von existierenden Lösungen ist eine so komplexe Aufgabe wie die automatische Generierung adaptierbarer, hypermedialer Dokumente zu bewältigen. Insbesondere in den Gebieten "Automatisierung", "Autorenunterstützung" und "Geschwindigkeit" läßt auch das *i4 Framework* noch reichlich Spielraum für Verbesserungen. In naher Zukunft werden sicherlich Systeme verfügbar sein, die sowohl für das *Authoring* hypermedialer Lehr- und Infromationssysteme als auch für die vollautomatische Generierung dynamischer Dokumente Lösungen anbieten, gegen die alle aktuellen Systeme primitiv und spartanisch erscheinen.

Diese Perspektive vor Augen stellt sich natürlich die Frage: Was bleibt?

Was bleibt, sind hoffentlich einige der zugrundeliegenden Ideen, z.B. die Automatiserung eines *Bottom-Up* Entwicklungszyklus und das Model der doppelten Adaptierbarkeit. Auch die verwendete Beschreibung von Medienobjekten durch Attribute und Indizes ist ein guter Ausgangspunkt für bessere, noch weniger Semantik enthaltene und damit schneller erstellbare und einfacher automatisch ermittelbare Beschreibungen hypermedialer Objekte.

Das *i4 Framework* ist eine Umsetzung einer stark automatisierten *Bottom-Up* Generierung. Die in dieser Arbeit im Detail angegebenen Algorithmen und Datenstrukturen zeigen, daß eine *Bottom-Up* Generierung hypermedialer Dokumente unter völligem Verzicht auf übergeordnete, statische Strukturen und unter Verwendung eines Minimums an semantischen Informationen möglich ist. Diese Aspekte sind vor allem für die in Kapitel 10.3 wiedergegebene Vision einer „Suchmaschine der Zukunft“ von Bedeutung.

Das *World Wide Web* ist voll von Medienobjekten, aber arm an semantischen Netzen; der Ausgangspunkt jeder Dokumentenerzeugung muß daher die Medienobjektstruktur sein. Berücksichtigt man zusätzlich noch Flexibilität, Skalierbarkeit und Adaptierbarkeit von *Bottom-Up* Verfahren, so kann das Fazit dieser Arbeit nur lauten:

Die Zukunft der automatisierten Erstellung adaptierbarer, hypermedialer Dokumente liegt in einer weiteren Automatisierung des *Bottom-Up* Entwicklungszyklus durch noch stärkere Einbindungen von Techniken aus dem *Information Retrieval*, dem *Data Mining* und der künstlichen Intelligenz. Alle anderen -

insbesondere *Top-Down* ausgerichteten - Ansätze laborieren lediglich an einzelnen Symptomen, nicht aber an den Ursachen der enormen Kosten für Entwicklung und Wartung wissensvermittelnder Dokumente.

Literatur

[Aho+86] Aho, A.V., Sethi, R. und Ullman, J. D., *Compilers - Principles, Techniques, and Tools*. Reading: Addison-Wesley, 1986

[Allaire98] Allaire Corp., *ColdFusion 4.0 White Paper*. 1998. http://www2.allaire.com/products/coldfusion/whitepapers/CF4Whitepaper_5.pdf

[Amitay98] Amitay, E., "Using Common Hypertext Links to Identify the best Phrasal Description of Target Web Documents". In *Proc. Workshop on Hypertext Information Retrieval for the Web*. Melbourne, August 1998. http://www.mri.mq.edu.au/~einat/sigir/

[Anderson+73] Anderson, J. R. und Bower, G. H., *Human Associative Memory*. Washington: Winston, 1973.

[Anderson93] Anderson, J. R., *Rules of the Mind*. Hillsdale: Erlbaum, 1993

[Anderson95] Anderson, J. R., *Cognitive Psychology and its Implications*[2]. New York: Freeman, 1995.

[Apostolopoulos+96] Apostolopoulos, N., Geukes, A., und Zimmermann, S., "Digital Interactive Lectures in Higher Education". In *Proc. Educational Multimedia and Hypermedia, ED-Media'96*. Boston, Juni 1996.

[Baron+96] Baron, L., Tague-Sutcliffe, J., Kinnucan, M. und Carey, T., "Labeled, Typed Linkes as Cues when Reading Hypertext Documents". In *Journal of the American Society for Information Science*. 47(12), pp. 896-908, Dezember 1996.

[Baumgartner97] Baumgartner, P., "Didaktische Anforderungen an multimediale Lernsoftware". In Issing, L. J. und Klimsa, P., (Eds.), *Information und Lernen mit Multimedia*. Weinheim: Beltz, 1997.

[Bieber+97] Bieber, M., Vitali, F., Ashman, H., Balasubramanian, V. und Oinas-Kukkonen, H., "Fourth Generation Hypermedia: Some Missing Links for the World Wide Web". In *Int. Journal of Human-Computer Studies*. 47(1), 31-66, Juli 1997. http://ijhcs.open.ac.uk/bieber/bieber-01.html

[Boykin+99] Boykin, S. M. und Merlino, A. E., "Improving Broadcast News Segmentation Processing". In *Proc. IEEE International Conference on Multimedia Computing and Systems, ICMCS99*. Florenz, Juni 1999.

[Brandman+99] Brandman, O., Elkan, C. P. und Paturi, M., "Improved Web Searching Using Neighborhood Ranking". In *Poster Proc. 8th International World Wide Web Conference*. Toronto, Mai 1999.

[Brusilovsky+96] Brusilovsky, P., Schwarz, E. und Weber, G., "ELM-ART: An intelligent tutoring system on World Wide Web". In *Proc. 3rd International Conference on Intelligent Tutoring Systems, ITS-96*. Montreal, Juni 1996. http://www.contrib.andrew.cmu.edu/~plb/ITS96.html

[Brusilovsky+96b] Brusilovsky, P., Schwarz, E. und Weber, G., "A Tool for Developing Adaptive Electronic textbooks on WWW". In *Proc. World Conference of the WWW, Internet & Intranet, WebNet 96*. San Francisco, Oktober 1996. http://www.contrib.andrew.cmu.edu/~plb/WebNet96.html

[Brusilovsky98] Brusilovsky, P., "Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies". In *Proc. Workshop on World Wide Web Based Tutoring*. San Antonio, August 1998. http://www-aml.cs.umass.edu/~stern/webits/itsworkshop/brusilovsky.html

[Brusilovsky+98] Brusilovsky, P., Eklund, J. und Schwarz, E., "Web-Based Education for All: A Tool for Developing Adaptive Courseware". In *Proc. 7th International World Wide Web Conference*. Brisbane, April 1998. http://decweb.ethz.ch/WWW7/1893/com1893.htm

[Brusilovsky+98b] Brusilovsky, P. und Eklund, J., "A Study of User Model Based Link Annotation in Educational Hypermedia". In *Journal of Universal Computer Science*. 4(4), Springer Science Online. http://www.iicm.edu/jucs_4_4/a_study_of_user/paper.html

[Buchmann+98] Buchmann, P. und Geukes, A., "Reproduction of Hypermedia Lectures". In *Proc. World Conference of the WWW, Internet & Intranet, WebNet 98*. Miami, März 1998.

[Carlson89] Carlson, P. A., "Hypertext and Intelligent Interfaces for Text Retrieval". In Barrett, E. (Ed.), *The Society of Text*. Cambridge, MA: MIT Press, 1989.

[Calvi+97] Calvi, L. und De Bra, P., "Using Dynamic Hypertext to Create Multi-Purpose Textbooks". In *Proc. Educational Multimedia and Hypermedia, ED-Media'97*. Calgary, Juni 1997.

[Caumanns98] Caumanns, J., "A Bottom-Up Approach to Multimedia Teachware". In *Proc 4th International Conference on Intelligent Tutoring Systems, ITS-98*. San Antonio, August 1998.

[Caumanns99a] Caumanns, J., "A Modular Framework for the Creation of Dynamic Documents". In *Proc. Workshop on Virtual Documents, Hypermedia Functionality and the Web*, Toronto, Mai 1999. http://www.cs.unibo.it/~fabio/VD99/caumanns/caumanns.html

[Caumanns+99b] Caumanns, J. and Lenz, H.-J., "Hypermedia Fusion - A Document Generator for the World Wide Web". In *Proc. IEEE International Conference on Multimedia Computing and Systems, ICMCS99*. Florenz, Juni 1999. http://www.wiwss.fu-berlin.de/~caumanns/i4/papers/icmcs99.zip

[Caumanns+99c] Caumanns, J., Apostolopoulos, N., Buchmann, P., and Geukes, A., "Computers are not Books". In *Proc. Educational Multimedia and Hypermedia, ED-Media'99*. Seattle, Juni 1999. http://www.wiwiss.fu-berlin.de/~caumanns/i4/papers/edmedia99.rtf

[Caumanns99d] Caumanns, J., "A Graph-Based Information Integration Architecture". In *Proc. 3rd Workshop on Intelligent Information Integration*. Stockholm, Juli 1999. http://www.aifb.uni-karlsruhe.de/WBS/dfe/iii99/caumanns-ijcai99-iii.ps

[Caumanns99e] Caumanns, J., *A Fast and Simple Stemming Algorithm for German Words*. Technical Report Nr. tr-b-99-16 des Fachbereichs Informatik der Freien Universität Berlin. Oktober 1999. ftp://ftp.inf.fu-berlin.de/pub/reports/tr-b-99-16.ps.gz

[Chakrabarti+98] Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D. und Kleinberg, J., "Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text". In *Proc. 7th International World Wide Web Conference*. Brisbane, April 1998. http://decweb.ethz.ch/WWW7/1898/com1898.htm

[Chellappa+97] Chelappa, R., Barua, A. und Whinston, A. B., "An Electronic Infrastructure for a Virtual University". In *Communications of the ACM*. 40(9), 56-58. September 1997.

[Christel+99] Christel, M. G. und Olligschlaeger, A. M., "Interactive Maps for Digital Video Library". In *Proc. IEEE International Conference on Multimedia Computing and Systems, ICMCS99*. Florenz, Juni 1999.

[Collins+89] Collins, A., Brown, J. S. und Newman, S. E., "Cognitive Apprenticeship: Teaching the Crafts of Reading, Writing and Mathematics". In Resnick, L. B. (Ed.), *Knowing, Learning and Instruction*. Hillsdale, NJ: Erlbaum, 1989.

[Coppola+99] Coppola, N., Rana, A. und Bieber, M., "Collaborative Hypermedia Educational Framework (CHEF): Instantiation and Assessment of an Instructional Model". *In Proc. 5th Australian Conference on the World Wide Web, AusWeb'99*. Ballina, April 1999, http://ausweb.scu.edu.au/aw99_archive/aw99/proceedings/bieber/paper.html

[Cormen+90] Cormen, T.H., Leiserson, C.E. und Rivest, R. L., *Introduction to Algorithms*, Cambridge: MIT Press, 1990.

[Cresson96] Cresson, E., *Presentation of the European Year of Lifelong Learning*. Rede in Edinburgh, 22. Februar 1995. http://europa.eu.int/en/comm/dg22/news/sp47.html

[Dalamagas+97] Dalamagas, T. und Dunlop, M., "Automatic Construction of News Hypertext". In Fuhr, N., Dittrich, G. und Tochtermann, K., *Hypertext - Information Retrieval - Multimedia '97* (Schriften zur Informationswissenschaft, Band 30). Konstanz: UVK, 1997.

[DeBra99] De Bra, P., "Design Issues in Adaptive Hypermedia Application Development". In *Proc. 2nd Workshop on Adaptive Systems and User Modeling on the World Wide Web*. Toronto, Mai 1999.

[Denning97] Denning, A., *ActiveX Controls Inside Out*, Redmond: Microsoft Press, 1997.

[DIALEKT00] Dialekt Projektteam, *Statistik interaktiv!* Berlin, Heidelberg: Springer (erscheint im Frühjahr 2000).

[Diestel96] Diestel, R., *Graphentheorie*, Berlin: Springer, 1996.

[Dunning94] Dunning, T., *Statistical Identification of Language*. TR CRL MCCS-94-273, Computing Research Lab, New Mexico State University, Las Cruces, 1994.

[Dwyer+95] Dwyer, D., Barbieri, K. und Doerr, H. M., "Creating a Virtual Classroom for Interactive Education on the Web". In *Proc. 3rd Wirld Wide Web Conference*. Darmstadt, April 1995. http://www.igd.fhg.de/www/www95/proceedings/papers/62/ctc.virtual.class/ctc.virtual.class.html

[Eklund+97] Eklund, J., Brusilovsky, P. und Schwarz, E., "Adaptive Textbooks on the WWW". In *Proc. 3rd Australian Conference on the World Wide Web, AusWeb97*. Queensland, Juli 1997. http://ausweb.scu.edu.au/proceedings/eklund/paper.html

[Embacher97a] Embacher, F., *Projektbeschreibung CD-ROM Mathematik, April 1997*. http://merlin.mpi.univie.ac.at/~fe/beschr97.htm

[Embacher97b] Embacher, F., *Hypertextstruktur CD-ROM Mathematik*. http://merlin.mpi.univie.ac.at/~fe/nb.htm

[Foshay97] Foshay, R., *CBT's first Generation: If we're so smart, why ain't we rich?* ITForum listserv. http://itech1.coe.uga.edu/itforum/paper19/paper19.html

[Frasson+98] Frasson, C., Martin, L., Gouardères, G. und Aïmeur, E., "LANCA: A Distance Learning Architecture Based on Networked Cognitive Agents". In *Proc 4th International Conference on Intelligent Tutoring Systems, ITS-98*. San Antonio, August 1998.

[Fürnkranz+98] Fürnkranz, J., Mitchell, T. und Riloff, E., „A Case Study in Using Linguistic Phrases for Text Categorization on the WWW." In Sahami, M. (Ed.) *Learning for Text Categorization: Papers from the 1998 AAAI/ICML Workshop*. pp 5-13, Madison: AAAI Press, 1998. http://www.ai.univie.ac.at/~juffi/publications/aaai-98-text.ps.gz

[Fürnkranz99] Fürnkranz, J., "Exploiting Structural Information for Text Classification on the WWW". In *Proc. 3rd Symposium on Intelligent Data Analysis IDA-99* (LNCS1642). Amsterdam, August 1999. http://www.ai.univie.ac.at/~juffi/publications/ida-99.ps.gz

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

[Garrido+97] Garrido, A, Rossi, G. und Schwabe, D., "Pattern systems for Hypermedia". In *Proc. Pattern Language of Program, PLoP 97*. Monticello, IL, 1997. http://st-www.cs.uiuc.edu/users/hammer/PLoP-97.html

[Gastmeyer92] Gastmeyer, M., *Formale und Inhaltliche Regeln zur Erstellung des Thesaurus Wirtschaft* (HWWA-Report Nr. 107). Hamburg: HWWA-Institut für Wirtschaftsforschung, 1992.

[Gerstenmaier+94] Gerstenmaier, J. und Mandl, H., *Wissenserwerb unter konstruktivistischer Perspektive*. Forschungsbericht Nr. 33, Institut für Empirische Pädagogik und Pädagogische Psychologie, LMU München, 1994. http://infix.emp.paed.uni-muenchen.de/lsmandl/forschbe/lit33.html

[Girgensohn+99] Girgensohn, A. und Boreczky, J., "Time-Contrained Keyframe Selection Technique". In *Proc. IEEE International Conference on Multimedia Computing and Systems, ICMCS99*. Florenz, Juni 1999.

[Gosling+96] Gosling, J. und McGilton, H., *The Java Language Environment*, Mai 1996, http://www.javasoft.com/docs/white/langenv/

[Graf97] Graf, W., "Constraint-based Graphical Layout of Multimodal Presentations". In Maybury, M. und Wahlster, W. (Eds.), *Readings in Intelligent User Interfaces*. Los Altos: Morgan Kaufman, 1997.

[Green98] Green, S., "Automatted Link Generation: Can we do Better than Term Repetition?" In *Proc. 7th International World Wide Web Conference*. Brisbane, April 1998. http://decweb.ethz.ch/WWW7/1834/com1834.htm

[Green+99] Green, S., Milosavljevic, M., Dale, R. und Paris, C., "When Virtual Documents Meet the Real World". In *Proc. Workshop on Virtual Documents, Hypermedia Functionality and the Web*. Toronto, Mai 1999. http://www.cs.unibo.it/~fabio/VD99/green/green.html

[Grefenstette94] Grefenstette, G., *Explorartions in Automatic Thesaurus Discovery*. Boston, Dordrecht: Kluwer Academic Publishers, 1994.

[Hammwöhner97] Hammwöhner, R., "Komplexe Hypertextmodelle im World Wide Web durch dynamische Dokumente". In: Fuhr, N., Dittrich, G. und Tochtermann, K., *Hypertext - Information Retrieval - Multimedia '97* (Schriften zur Informationswissenschaft, Band 30). Konstanz: UVK, 1997.

[Hearst+93] Hearst, M. und Plaunt, C., "Subtopic Structuring for Full-Length Document Access". In *Proc. 16th ACM-SIGIR Conference*. Pittsburgh, Juni/Juli 1993.

[Hofstadter85] Hofstadter, D. R., *Gödel, Escher, Bach - ein Endloses Geflochtenes Band*. Stuttgart: Klett-Cotta, 1985.

[HTML4.0] Raggett, D., Le Hors, A. und Jacobs, I. (Eds.), HTML 4.0 Specification (W3C Recommendation), April 1998. http://www.w3.org/TR/REC-html40/

[Issing+97] Issing, L. J. und Klimsa, P. (Eds.), *Information und Lernen mit Multimedia*. Weinheim: Beltz, 1997

[Jank+94] Jank, W. und Meyer, H., *Didaktische Modelle*, Frankfurt (Main): Cornelsen Scriptor, 1994.

[Jonassen+90] Jonassen, D. H. und Mandl, H. (Eds.), *Designing Hypermedia for Learning*. Berlin: Springer, 1990.

[Jonassen91] Jonassen, D. H., "What are Cognitive Tools?", In Kommers, P. A. M., Jonassen D. H. und Mayes, J. T., *Cognitive Tools for Learning*. Berlin: Springer, 1991.

[Jones+99] Jones, M., Marsden, G., Mohd-Nasir, N., Boone, K. und Buchanan, G., "Improving Web Interaction on Small Displays". In *Proc. 8th International World Wide Web Conference*. Toronto, Mai 1999.

[Jung99] Jung, A., "Rilke, elektronisch." In *SPIEGEL Spezial 10/99: Die Zukunft des Lesens*. Hamburg: Spiegel-Verlag, 1999.

[Kay+94] Kay, J. und Kummerfeld, R. J., "An Individualised Course for the C Programming Language". In: *Proc. 2nd International World Wide Web Conference*. Boston, Oktober 1994. http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Educ/kummerfeld/kummerfeld.html

[Kintsch74] Kintsch, W., *The Representation of Meaning in Memory*. Hillsdale: Erlbaum, 1974.

[Knuth73] Knuth, D. E., *The Art of Computer Programming, Volume 1: Fundamental Algorithms* (2nd Edition), Reading, MA: Addison-Wesley, 1973.

[Kopetzky+99] Kopetzky, T. und Mühlhäuser, M., "Visual Preview for Link Traversal on the World Wide Web". In *Proc. 8th International World Wide Web Conference*. Toronto, Mai 1999.

[LTSC98] IEEE Leraning Technology Standards Committee (LTSC), Learning Objekt Metadata (LOM), Draft Document v2.1, Juni 1998. http://ltsc.ieee.org/doc/wg12/LOMdoc2_1.html

[Leutner92] Leutner, D., *Adaptive Lehrsysteme*. Weinheim: Beltz, 1992.

[Mandl+93] Mandl, H., Friedrich, H. F., und A. Horn, "Psychologie des Wissenserwerbs". In Weidenmann, B., Krapp, A., Hofer, M., Huber, G. L., und Mandl, H. (Eds.), *Pädagogische Psychologie*, Weinheim: Beltz, 1993.

[Mandl+97] Mandl, H., Gruber, H. und Renkl, A., "Situiertes Lernen in multimedialen Lernumgebungen". In Issing, L. J. und Klimsa, P. (Eds.), *Information und Lernen mit Multimedia*. Weinheim: Beltz, 1997

[Martin+99] Martin, P. und Eklund, P., "Embedding Knowledge in Web Documents". In *Proc. 8th International World Wide Web Conference*. Toronto, Mai 1999.

[Mayes+90] Mayes, T., Kibby, M. und Anderson, T., *Learning about Learning from Hypertext*. NATO ASI Series, Volume F67, Heidelberg: Springer, 1990.

[Meiss97] Meiss, B., *Information Retrieval und Dokumentenmanagement im Multimedia-Zeitalter*. Frankfurt/Main: Deutsche Gesellschaft für Dokumentation, 1997.

[Milosavljevic+96] Milosavljevoc, M., Tulloch, A. und Dale, R., "Text Generation in a Dynamic Hypertext Environment". In *Proc. 19th Australasian Computer Science Conference*. Melbourne, Januar/Februar 1996. http://www.comp.mq.edu.au/~mariam/papers/acsc96/

[Milosavljevic+98] Milosavljevic, M., Dale, R., Green, S., Paris, C. und Williams, S., "Virtual Museums on the Information Superhighway: Prospects and Potholes". In *Proc. Annual Conf. of the International Committee for Documentation of the International Council of Museums, CIDIC'98*. Melbourne, Oktober 1998. http://www.dynamicmultimedia.com.au/papers/cidoc98/

[Minsky75] M. Minsky, "A framework for representing knowledge". In Winston, P. H. (Ed.), *The Psychology of Computer Vision*. New York: McGraw-Hill, 1975.

[Minsky88] M. Minsky, *The Society of Mind*, New York: Touchstone, 1988.

[NCSA96] The National Computational Science Alliance, *The Common Gateway Interface*, 1996. http://hoohoo.ncsa.uiuc.edu/docs/cgi/overview.html

[Negroponte95] N. Negroponte, *Total Digital*. München: Bertelsmann, 1995.

[Nilsson98] N. J. Nilsson, *Artificial Intelligence - A New Synthesis*, Morgan Kaufmann, San Francisco, 1998.

[Nielsen97] J. Nielsen, *Interface Design for Sun's WWW Site*. Palo Alto: Sun Microsystems, 1997. http://www.sun.com/sun-on-net/uidesign/

[Noltemeyer76] H. Noltemeyer, *Graphentheorie mit Algorithmen und Anwendungen*, Berlin: deGruyter, 1976.

[Norman+75] Norman, D.A. und Rumelhart, D.E., *Explorations in Cognition*. San Francisco: Freeman, 1975.

[Pearl84] Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading: Addison-Wesley, 1984.

[Perez+95] Pérez, T., Gutiérrez, P. und Lopistéguy, P., "An adaptive hypermedia system". In *Proc. 7th World Conference on Artificial Inteligence in Education, AI-ED'95*. Washington, August 1995. http://www.ji.si.ehu.es/hyper/Publicaciones/AIED95/AIED95.html

[Plotkin+99] Plotkin, B, und Garone, S., *Apple's WebObjects. An IDC White Paper*. http://www.apple.com/webobjects/whitepaper/index.html

[Pong+99] Pong, T., Chin, R. und Ngo, C.-W., "Camera Breaks Detection by Partitioning of 2-D Spatio-Temporal Images in the MPEG Domain". In *Proc. IEEE International Conference on Multimedia Computing and Systems, ICMCS99*. Florenz, Juni 1999.

[Quillian66] M. R. Quillian, *Semantic Memory*. Cambridge: Bolt, Beranak and Newman: 1966.

[RFC2046] Freed, N. and N. Borenstein, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types (Draft Standatd)*, November 1996. http://info.internet.isi.edu/in-notes/rfc/files/rfc2046.txt

[RFC2068] Fielding, R., Gettys, J., Mogul, J., Nielsen, H. F., and T. Berners-Lee, *Hypertext Transfer Protocol - HTTP 1.1 (Proposed Standard)*, Januar 1997. http://info.internet.isi.edu/in-notes/rfc/files/rfc2068.txt

[Riehm+96] Riehm, U. und Wingert, B., *Multimedia - Mythen, Chancen und Herausforderungen*. Mannheim: Bollmann, 1996.

[Rölleke+96] Rölleke, T. und Fuhr, N., "Retrieval of Complex Objects using a Four-Valued Logic". In *Proc. 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Zürich, August 1996. http://ls6-www.cs.uni-dortmund.de/ir/publications/1996/Roelleke_Fuhr:96.html

[Rölleke+97] Rölleke, T. und Fuhr, N., "Retrieving Complex Objects with HySpirit". In *Proc. 19th Annual BCS-IRSG Colloquium on IR Research*. Aberdeen, April 1997. http://ls6-www.cs.uni-dortmund.de/ir/publications/1997/Roelleke_Fuhr:97c.html

[Rossi+95] Rossi, G, und Schwabe, D., "Building hypermedia applications as navigational views of information models". In *Proc. 28th Annual Hawaii International Conference on System Science*. Hawaii, Januar 1995. http://ftp.inf.puc-rio.br/pub/docs/techreports/94_41_schwabe.ps.gz

[Rossi+99] Rossi, G., Schwabe, D. und Lyardet, F., "Improving Web information systems with navigational patterns". In *Proc. 8th International World Wide Web Conference*. Toronto, Mai 1999.

[Sahami+98] Sahami, M., Yusufali, S. und Baldonado, M., "SONIA: A Service for Organizing Networked Information Autonomously". In *Proc. 3rd ACM Conference on Digital Libraries, DL98*. Pittsburgh, Juni 1998. http://www.acm.org/pubs/articles/proceedings/dl/276675/p200-sahami/p200-sahami.pdf

[Salton+96] Salton, G., Allan, J., Buckley, C. und Singhal, A., "Automatic Analysis, Theme Generation and Summarization of Machine-Readable Texts." In Agosti, M. und Smeaton, A. (Eds.), *Information Retrieval and Hypertext*. Boston, Dordrecht: Kluwer Academic Publishers, 1996.

[Schamber96] Schamber, L., "What is a Document? Rethinking the Concept in Uneasy Times". In *Journal of the American Society for Information Science*. 47(9), pp. 669-672, September 1996.

[Schmid+96] Schmid, U. und Kindsmüller, M.C., *Kognitive Modellierung*, Heidelberg: Spektrum, 1996.

[Schulmeister96] Schulmeister, R. *Grundlagen hypermedialer Lernsysteme*. Addison-Wesley, 1996.

[Södergård+99] Södergård, C., Aaltonen, M., Hagman, S., Hiirsalmi, M., Järvinen, T., Kaasinen, E., Kinnunen, T., Kolari, J., Kunnas, J. und Tammela, A., "Integrated Multimedia Publishing: Combining TV and Newspaper Content on Personal Channels". In *Proc. 8th International World Wide Web Conference*. Toronto, Mai 1999.

[Sowa84] J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*. Reading: Addison Wesley, 1984.

[Sowa92] J. F. Sowa, "Conceptual Graphs Summary". In Nagle, T. E., Nagle, J. A., Gerholz L. L. und Eklund, P. W. (Ed.), *Conceptual Structures: Current Research and Practice*. New York, London: Ellis Horwood, 1992.

[Specht+97] Specht, M., Weber, G., Heitmeyer, S. und Schöch, V., "AST: Adaptive WWW-Courseware for Statistics". In *Proc. UM97 Workshop on Adaptive Systems and User Modeling on the World Wide Web*. Chia Laguna, Sardinien, Juni 1997.
http://www.contrib.andrew.cmu.edu/~plb/UM97_workshop/Specht.html

[Srinivasan+99] Srinivasan, S., Ponceleon, D., Amir, A. und Petkovic, D., "What is in that Video anyway?: In Search of Better Browsing". In *Proc. IEEE International Conference on Multimedia Computing and Systems, ICMCS99*. Florenz, Juni 1999.

[Stanyer+99] Stanyer, D. und Procter, R., "Improving Web Usability with the Link Lens". In *Proc. 8th International World Wide Web Conference*. Toronto, Mai 1999.

[Stillings+95] Stillings, N. A., Weisler, S. E., Chase, C. H., Feinstein, M. H., Garfield, J. L. und Rissland, E. L., *Cognitive Science - An Introduction*, Cambridge: MIT Press, 1995.

[Strömer97] T. Strömer, *Online-Recht: Rechtsfragen im Internet und in Mailboxen*. Heidelberg: dpunkt-Verlag, 1997.

[Sutcliffe99] Sutcliffe, A., "User-Centered Design for Multimedia Applications". In *Proc. IEEE International Conference on Multimedia Computing and Systems, ICMCS99*. Florenz, Juni 1999.

[Swain99] Swain, M., "Multimedia Search on the World Wide Web". In *Proc. IEEE International Conference on Multimedia Computing and Systems, ICMCS99*. Florenz, Juni 1999.

[Tarjan72] Tarjan, R. E., „Depth First Search and Linear Graph Algorithms“. In *SIAM Journal of Computing*. 1(2), 146-160, 1972.

[Thalheim98] Thalheim, B., *Folien zur Vorlesung "Informationsdienste"*. BTU Cottbus, 1998.

[Thistlewaite97] Thistlewaite, P., "Automatic Construction and Management of Large Open Webs". In Agosti, M. und Allan, J. (Eds.), *Information Processing and Management, 33(2), Special Issue on Methods and Tools for the Automatic Construction of Hypermedia*. pp. 161-173, März 1997.

[Thüring+95] Thüring, M., Hannemann, J. und Haake, J., "Hypermedia and Cognition: Designing for Comprehension". *Communications of the ACM*. 38(8), pp. 57-66, 1995.

[Trigg88] Trigg, R. "Guided Tours and Tabletops: Tools for Communicating in a Hypertext Environment". In *ACM Transactions on Office Information Systems*. Oktober 1988.

[Vanderbilt90] Cognition and Technology Group at Vanderbilt, "Anchored Instruction and its Relationship to Situated Cognition". *Educational Researcher*. 19(6), pp. 2-10, 1990.

[Vassileva97] Vassileva, J., "Dynamic Course Generation on the WWW". In *Proc. 8th World Conference on Artificial Intelligence in Education, AI-ED97*. Kobe, Japan, August 1997. http://www.contrib.andrew.cmu.edu/~plb/AIED97_workshop/Vassileva/Vassileva.html

[Veijonen+94] Veijonen, T., Ahonen, H., Leikola, V., Niku-Paavola, J. und Ylä-Jääski, J. , "Automatted Page Layout for Electronic News Distribution". In *Proc. 1st IASTED/ISMM Int. Conference on Distributed Multimedia Systems and Applications*. Honolulu, August 1994.

[Verspoor+98] Verspoor, C., Dale, R., Green, S., Milosvljevic, M., Paris, C. und Williams, S., "Intelligent Agents for Information Presentation: Dynamic Description of Knowledge Base Objects". In *Proc. International Workshop on Intelligent Agents on the Internet and Web*. Mexico City, März 1998. http://www.mri.mq.edu.au/~kversp/iaiw_paper.ps

[Watters+99a] Watters, C. und Shepherd, M., "Cypergenre and Web Site Design". In *Poster Proc. 8th International World Wide Web Conference*. Toronto, Mai 1999.

[Watters+99b] Watters, C. und Shepherd, M., "Research Issues for Virtual Documents". In *Proc. Workshop on Virtual Documents, Hypermedia Functionality and the Web*. Toronto, Mai 1999. http://www.cs.unibo.it/~fabio/VD99/shepherd/shepherd.html

[Weber+97] Weber, G. und Specht, M., "User Modeling and Adaptive Navigation Support in WWW-based Tutoring Systems". In *Proc. 6th International Conference on User Modeling*. Cagliari, Juni 1997. http://www.psychologie.uni-trier.de:8000/projects/ELM/Papers/UM97-WEBER.html

[Weidenmann93] Weidenmann, B., "Psychologie des Lernens mit Medien", In Weidenmann, B., Krapp, A., Hofer, M., Huber, G. L. und Mandl, H. (Eds.), *Pädagogische Psychologie*. Weinheim: Beltz, 1993.

[Weidenmann88] Weidenmann, B., "When good pictures fail. An information processing approach to the effects of illustrations". In Mandl, H. und Levin, R.. (Eds.), *Knowledge Acquisition from Text and Pictures*. Amsterdam: Elsevier, 1988.

[Weidenmann97] Weidenmann, B., "Multicodierung und Multimodalität im Lernprozeß". In Issing, L. J. und Klimsa, P. (Eds.), *Information und Lernen mit Multimedia*. Weinheim: Beltz, 1997

[Weissinger99] Weissinger, K. A., *ASP in a Nutshell: A Desktop Quick Reference*. O'Reilly, 1999.

[Wilkinson+96] Wilkinson, R. und Fuller, M., "Integration of Information Retrieval and Hypertext via Structure". In Agosti, M. und Smeaton, A. (Eds.), *Information Retrieval and Hypertext*. Boston, Dordrecht: Kluwer Academic Publishers, 1996.

[Wirth83] N. Wirth, *Algorithmen und Datenstrukturen*. Stuttgart: Teubner, 1983.

[Yates+92] Yates, J. und Orlikowski, W. J., "Genres of organizational communication: A structurational approach to studying communications and media". *Academy of Management Review*, 17(2), pp 299-326, 1992.

[Zellweger+98] Zellweger, P., Chang, B.-W. und Mackinlay, J., "Fluid Links for Informed and Incremental Link Transitions". In *Proc. ACM Hypertext'98*. Pittsburgh, Juni 1998.

Anhang A: Verzeichnis der Abkürzungen

$\beta_{base}(b)$	$Blk \rightarrow \mathbf{P}(M^{avail})$	Basisobjekte eines Blocks b	5.3.1
$\phi_{in}(ft,F)$	$FT^{\#} \times F^{\#} \rightarrow \mathbf{P}(FT^{\#} \times FT^{\#} \times IR)$	Menge der Eingangskanten einer Seite ft	7.6.1
$\phi_{out}(ft,F)$	$FT^{\#} \times F^{\#} \rightarrow \mathbf{P}(FT^{\#} \times FT^{\#} \times IR)$	Menge der Ausgangskanten einer Seite ft	7.6.1
$\phi_{pre}(ft,F)$	$FT^{\#} \times F^{\#} \rightarrow \mathbf{P}(FT^{\#})$	Menge der direkten Vorgänger einer Seite ft	7.6.1
$\phi_{ps}(ft,F)$	$FT^{\#} \times F^{\#} \rightarrow \mathbf{P}(FT^{\#})$	Vorgängermenge einer Seite ft	7.6.1
$\phi_{ss}(ft,F)$	$FT^{\#} \times F^{\#} \rightarrow \mathbf{P}(FT^{\#})$	Nachfolgermenge einer Seite ft	7.6.1
$\phi_{succ}(ft,F)$	$FT^{\#} \times F^{\#} \rightarrow \mathbf{P}(FT^{\#})$	Menge der direkten Nachfolger einer Seite ft	7.6.1
$\mu_{attr}(m)$	$M^{avail} \rightarrow \mathbf{P}(string \times string)$	Attribute des Medienobjektes m	4.3
$\mu_{connect}(m,o,G)$	$M^{avail} \times M^{avail} \times G^{\#} \rightarrow \mathbf{P}(S^{avail})$	Die Medienobjekte m und o im Abhängigkeitsgraph G verbindende Stichworte	6.1.2
$\mu_{cycle}(m,G)$	$M^{avail} \times G^{\#} \rightarrow \{none, resolver, ...\}$	Zyklus-Typ eines Medienobjektes m	6.4.2
$\mu_{depth}(m,G)$	$M^{avail} \times G^{\#} \rightarrow IN$	Minimale Tiefe eines Medienobjekts m	6.3
$\mu_{kw}(m,I)$	$M^{avail} \times I^{avail} \rightarrow \mathbf{P}(S^{avail})$	Von einem Medienobjekt m referenzierte Stichworte	4.4.8
$\mu_{ndx}(m,I)$	$M^{avail} \times I^{avail} \rightarrow \mathbf{P}(I^{avail})$	Indexeinträge eines Medienobjekts m	4.4.8
$\mu_{pre}(m,G)$	$M^{avail} \times G^{\#} \rightarrow \mathbf{P}(M^{avail})$	Direkte Vorgänger eines Medienobjekts m in einem Abhängigkeitsgraphen G	6.1.2
$\mu_{prv}(m,I)$	$M^{avail} \times I^{avail} \rightarrow \mathbf{P}(I^{avail})$	Vermitteltes Wissen repräsent. Indexeinträge eines Mo. m	4.4.8
$\mu_{ps}(m,G)$	$M^{avail} \times G^{\#} \rightarrow \mathbf{P}(M^{avail})$	Vorgängermenge eines Medienobjekts m in einem Abhängigkeitsgraphen G	6.5
$\mu_{req}(m,I)$	$M^{avail} \times I^{avail} \rightarrow \mathbf{P}(I^{avail})$	Vorwissen repräsent. Indexeinträge eines Medienobjekts m	4.4.8
$\mu_{succ}(m,G)$	$M^{avail} \times G^{\#} \rightarrow \mathbf{P}(M^{avail})$	Direkte Nachfolger eines Medienobjekts m in einem Abhängigkeitsgraphen G	6.1.2
$\mu_{val}(m,n,d)$	$(M^{avail} \times string \times string) \rightarrow string$	Wert des Attributs n eines Medienobjekts m	4.3
$\sigma_{attr}(s)$	$S^{avail} \rightarrow \mathbf{P}(string \times string)$	Attribute des Stichworts s	4.4.4
$\sigma_{depth}(s,G)$	$S^{avail} \times G^{\#} \rightarrow IN$	Minimale Tiefe eines Stichworts s im Abhängigkeitsgraph G	6.3
$\sigma_{exist}(s,u)$	$S^{avail} \times U^{\#} \rightarrow R$	Zu Stichwort s vorhandenes Vorwissen	5.5
$\sigma_{know}(s)$	$S^{avail} \rightarrow R$	Default-Vorwissen eines Stichworts s	4.4.4
$\sigma_{ndx}(s,I)$	$S^{avail} \times I^{avail} \rightarrow \mathbf{P}(I^{avail})$	Indexeinträge eines Stichworts s	4.4.8
$\sigma_{parent}(s)$	$S^{avail} \rightarrow S^{avail}$	Übergeordnetes Stichwort eines Stichworts s	4.4.4
$\sigma_{prv}(s,I)$	$S^{avail} \times I^{avail} \rightarrow \mathbf{P}(I^{avail})$	Vermitteltes Wissen repräsent. Indexeinträge des	4.4.8

		Stichworts s	
$\sigma_{ps}(s,G)$	$S^{avail} \times G^{\#} \rightarrow \mathbf{P}(M^{avail})$	Vorgängermenge des Stichworts s im Abhängigkeitsgraphen G	6.5
$\sigma_{req}(s,I)$	$S^{avail} \times I^{avail} \rightarrow \mathbf{P}(I^{avail})$	Vorwissen repräsent. Indexeinträge eines Stichworts s	4.4.8
$\sigma_{val}(s,n,d)$	$(S^{avail} \times string \times string) \rightarrow string$	Wert des Attributs n eines Stichworts s	4.4.4
$\tau_{pre}(tm,T)$	$TM^{\#} \times T^{\#} \rightarrow \mathbf{P}(TM^{\#})$	Direkte Vorgänger eines Medienobjektknotens tm in einer Medienobjektstruktur T	7.4
$\tau_{ps}(tm,T)$	$TM^{\#} \times T^{\#} \rightarrow \mathbf{P}(TM^{\#})$	Vorgängermenge eines Medienobjektknotens tm in einer Medienobjektstruktur T	7.4.3
$\tau_{succ}(tm,T)$	$TM^{\#} \times T^{\#} \rightarrow \mathbf{P}(TM^{\#})$	Direkte Nachfolger eines Medienobjektknotens tm in einer Medienobjektstruktur T	7.4
A^M	$M^{avail} \times string \times string$	Attribute von Medienobjekten	4.3
$AO^{\#}$	$TM^{\#} \times ZO^{\#} \times \mathbf{P}(M^{avail})$	Menge der Zusatzobjekte	7.5.3
A^S	$S^{avail} \times string \times string$	Attribute von Stichworten	4.4.4
b_addon(b)	$Blk \rightarrow \mathbf{P}(ZO)$	Zusatzobjekte eines Blocks b	5.3.1
b_content(b)	$Blk \rightarrow \mathbf{P}(S^{avail})$	Im Block b zu vermittelnde Stichworte	5.3.1
b_flags(b)	$Blk \rightarrow \mathbf{P}(string \times string)$	Attribute eines Blocks b	5.3.1
b_genre(b)	$Blk \rightarrow string$	Genre eines Blocks b	5.3.1
Blk		Menge der Blockbeschreibungen	5.3.1
$C^{\#}$	$\mathbf{P}(FL^{\#}) \times C^{\#} \times \mathbf{P}(C^{\#}) \times IN \times FL^{\#}$	Menge der Kapitel	7.7
c_children(c)	$C^{\#} \rightarrow \mathbf{P}(C^{\#})$	Untergeordnete Kapitel eines Kapitels c	7.7
c_frames(c)	$C^{\#} \rightarrow \mathbf{P}(FL^{\#})$	Seiten eines Kapitels c	7.7
c_order(c)	$C^{\#} \rightarrow IN$	Position des Kapitels c innerhalb seines übergeordneten Kapitels	7.7
c_parent(c)	$C^{\#} \rightarrow C^{\#}$	Übergeordnetes Kapitel eines Kapitels c	7.7
c_start(c)	$C^{\#} \rightarrow FL^{\#}$	Anfangsseite des Kapitels c	7.7
C^{b*}	$C^{b*} \in C^{\#}$	Kapitelstruktur des Blocks b	7.7
$F^{\#}$	$P(FT^{\#}) \times \mathbf{P}(FT^{\#} \times FT^{\#} \times IR) \times \mathbf{P}(H^{\#})$	Menge der Seitenstrukturen	7.6.1
f_edges(f)	$F^{\#} \rightarrow \mathbf{P}(FT^{\#} \times FT^{\#} \times IR)$	Kanten der Seitenstruktur f	7.6.1
f_links(f)	$F^{\#} \rightarrow \mathbf{P}(H^{\#})$	Hyperlinks der Seitenstruktur f	7.6.1
f_nodes(f)	$F^{\#} \rightarrow \mathbf{P}(FT^{\#})$	Seiten der Seitenstruktur f	7.6.1
F^{b*}	$F^{b*} \in F^{\#}$	Seitenstruktur eines Blocks b	7.6.1
$FL^{\#}$	$FT^{\#} \times FT^{\#} \times FT^{\#}$	Verkettete Seite innerhalb eines Kapitels	7.7
$FT^{\#}$	$\mathbf{P}(TM^{\#}) \times \mathbf{P}(H^{\#}) \times bool$	Menge der Seiten	7.6.1
ft_links(ft)	$FT^{\#} \rightarrow \mathbf{P}(H^{\#})$	Von einer Seite ft ausgehende Hyperlinks	7.6.1
ft_nodes(ft)	$FT^{\#} \rightarrow \mathbf{P}(TM^{\#})$	Zu einer Seite ft zusammengefaßte Medienobjektknoten	7.6.1
ft_tree(ft)	$FT^{\#} \rightarrow bool$	Markierung, ob die Seite ft Teil der Seitenstruktur ist	7.6.1
G(M,I)	$\mathbf{P}(M^{avail}) \times \mathbf{P}(I^{avail}) \rightarrow G^{\#}$	Erzeugung eines Abhängigkeitsgraphen aus einer Menge von medienobjekten M und einer Menge von Indexeinträgen I	
$G^{\#}$	$\mathbf{P}(M^{avail}) \times \mathbf{P}(S^{avail}) \times \mathbf{P}(I^{avail}) \times \mathbf{P}(I^{avail})$	Menge der Abhängigkeitsgraphen	6.1
G^{b*}	$G^{b*} \in G^{\#}$	Abhängigkeitsgraph der Medienobjektstruktur eines Blocks b	6.2

G_m	$M^{avail} \times \mathbf{P}(S^{avail}) \times \mathbf{P}(I^{avail}) \times \mathbf{P}(I^{avail})$	Medienobjektgraph eines Medienobjekts m	4.4.7
G^M	$G^{\#} \rightarrow \mathbf{P}(M^{avail}) \times \mathbf{P}(M^{avail} \times M^{avail} \times IR)$	Medienobjektgraph eines Abhängigkeitsgraphen G	6.1.3
$H^{\#}$	$I^{avail} \times I^{avail}$	Menge der Hyperlinks	7.4
HSG(G)	$G^{\#} \rightarrow \mathbf{P}(\mathbf{P}(M^{avail}))$	Hierarchische Subgraphen im Abhängigkeitsgraphen G	6.6
I^{avail}	$M^{avail} \times S^{avail} \times R \times R \times R$	Menge der Indexeinträge	4.4.3
I_b	$I_b \subseteq I^{avail}$	Für die Generierung des Blocks b verfügbare Indexeinträge	5.5
$i_{connect}(m,o,G)$	$M \times M \times G^{\#} \rightarrow I^{avail} \times I^{avail}$	Die Medienobjekte m und o verbindende Indexeinträge	6.1.3
kw(i)	$I^{avail} \rightarrow S^{avail}$	Mit Indexeitrag i verknüpftes Stichwort	4.4.3
kw(sw)	$S_w \rightarrow S^{avail}$	Stichwort eines gewichteten Stichwortes sw	5.4
M^{avail}		Menge der Medienobjekte	4.1
M(G)	$G^{\#} \rightarrow \mathbf{P}(M^{avail})$	Im Abhängigkeitsgraphen G enthaltene Medienobjekte	6.1
M^b	$M_b \subseteq M^{avail}$	Für die Generierung des Blocks b verfügbare Medienobjekte	5.5
M^{b*}	$M_b{}^* \subseteq M^{avail}$	Medienobjekte der Medienobjektstruktur eines Blocks b	5.2.1
mo(i)	$I^{avail} \rightarrow M^{avail}$	Mit Indexeitrag i verknüpftes Medienobjekt	4.4.3
$M_{root}(G)$	$G^{\#} \rightarrow \mathbf{P}(M^{avail})$	Wurzeln des Abhängigkeitsgraphen G	6.1.2
$O^{\#}$	$\{ (FT^{\#} \times FT^{\#} \times bool) \}$	Menge der Ordering Informationen	7.6.3
prv(i)	$I^{avail} \rightarrow R$	Von mo(i) vermitteltes Wissen über kw(i)	4.4.3
R	$R \subseteq IN$	Wertebereich für diverse Attribute	4.4.3
rel(i)	$I^{avail} \rightarrow R$	Relevanz von kw(i) für mo(i)	4.4.3
req(i)	$I^{avail} \rightarrow R$	Von mo(i) verlangtes Vorwissen über kw(i)	4.4.3
S^{avail}		Menge der Stichworte	4.4.2
S(G)	$G^{\#} \rightarrow \mathbf{P}(S^{avail})$	Im Abhängigkeitsgraphen G enthaltene Stichworte	6.1
S^b	$S_b \subseteq S^{avail}$	Für die Generierung des Blocks b verfügbare Stichworte	5.5
SCC(G)	$G^{\#} \rightarrow \mathbf{P}(\mathbf{P}(M^{avail}))$	Starke Zusammenhangskomponenten im Abhängigkeits-graphen G	6.4
$S_{end}(G)$	$G^{\#} \rightarrow \mathbf{P}(S^{avail})$	Offene Enden des Abhängigkeitsgraphen G	6.1.2
$S_{leaf}(G)$	$G^{\#} \rightarrow \mathbf{P}(S^{avail})$	Blätter des Abhängigkeitsgraphen G	6.1.2
sprv(s,G)	$S^{avail} \times G^{\#} \rightarrow \mathbf{P}(M^{avail})$	Das in einem Abhängigkietsgraphen G maximal zu einem Stichwort s benötigte Vorwissen vermittelnde Medienobjekte	7.4.1
sreqmax(s,G)	$S^{avail} \times G^{\#} \rightarrow R$	Im Abhängigkeitsgraphen G maximal zum Stichwort s benötigtes Vorwissen	7.4.1
S^w	$string \times R$	Menge der gewichteten Stichworte	5.4
$T^{\#}$	$\mathbf{P}(TM^{\#}) \times \mathbf{P}(TM^{\#} \times TM^{\#}) \times \mathbf{P}(H^{\#})$	Menge der Medienobjektstrukturen	7.4
t_edges(t)	$T^{\#} \rightarrow \mathbf{P}(TM^{\#} \times TM^{\#})$	Kanten der Medienobjektstruktur t	7.4
t_links(t)	$T^{\#} \rightarrow \mathbf{P}(H^{\#})$	Hyperlinks innerhalb der Medienobjektstruktur t	7.4
t_nodes(t)	$T^{\#} \rightarrow \mathbf{P}(TM^{\#})$	Knoten der Medienobjektstruktur t	7.4
T^{b*}	$T^{b*} \subseteq T^{\#}$	Medienobjektstruktur des Blockes b	7.4
$TM^{\#}$	$M^{avail} \times \mathbf{P}(AO^{\#}) \times bool$	Menge der Medienobjektknoten	7.4
tm_addon(tm)	$TM \rightarrow \mathbf{P}(AO^{\#})$	Zusatzobjekte eines Medienobjektknotens tm	7.4
tm_flag(tm)	$TM^{\#} \rightarrow bool$	Marke, ob ein Medienobjektknoten gebunden ist oder	7.4

		nicht	
tm_mo(tm)	$TM^{\#} \rightarrow M^{avail}$	Medienobjekt eines Medienobjektknoten tm	7.4
$U^{\#}$	$\mathbf{P}(Blk) \times \mathbf{P}(S_w)$	Menge der Nutzungsszenarien	5.3.1
u_blocks(u)	$U^{\#} \rightarrow \mathbf{P}(Blk)$	Im Nutzungszenario u definierte Blöcke	5.3.1
u_pre(u)	$U^{\#} \rightarrow \mathbf{P}(S_w)$	Im Nutzungsszenario u kodiertes Vorwissen	5.4
$w_{edge}(m,o,G)$	$M^{avail} \times M^{avail} \times G^{\#} \rightarrow IR$	Gewichtung der Kante zwischen den Medienobjekten m und o im Abhängigkeitsgraph G	6.1.3
z_base(zo)	$ZO^{\#} \rightarrow$ string	Basis der Verwendungshäufigkeit einer Zusatzobjekt-Gruppe zo	7.5
$z\_freq_{genre}(zo)$	$ZO^{\#} \rightarrow IR$	Verwendunghäufigkeit einer Zusatzobjekt-Gruppe zo	7.5
$z\_freq_{obj}(zo)$	$ZO^{\#} \rightarrow IN$	Verwendunghäufigkeit eines einzelnen Zusatzobjekts einer Zusatzobjekt-Gruppe zo	7.5
z_genre(zo)	$ZO^{\#} \rightarrow$ string	Genre einer Zusatzobjekt-Gruppe zo	7.5
z_pos(zo)	$ZO^{\#} \rightarrow$ string	Positionierung einer Zusatzobjekt-Gruppe zo	7.5
$ZO^{\#}$	string × string × IR × IN × string	Menge der Zusatzobjekt-Gruppen	5.3.2

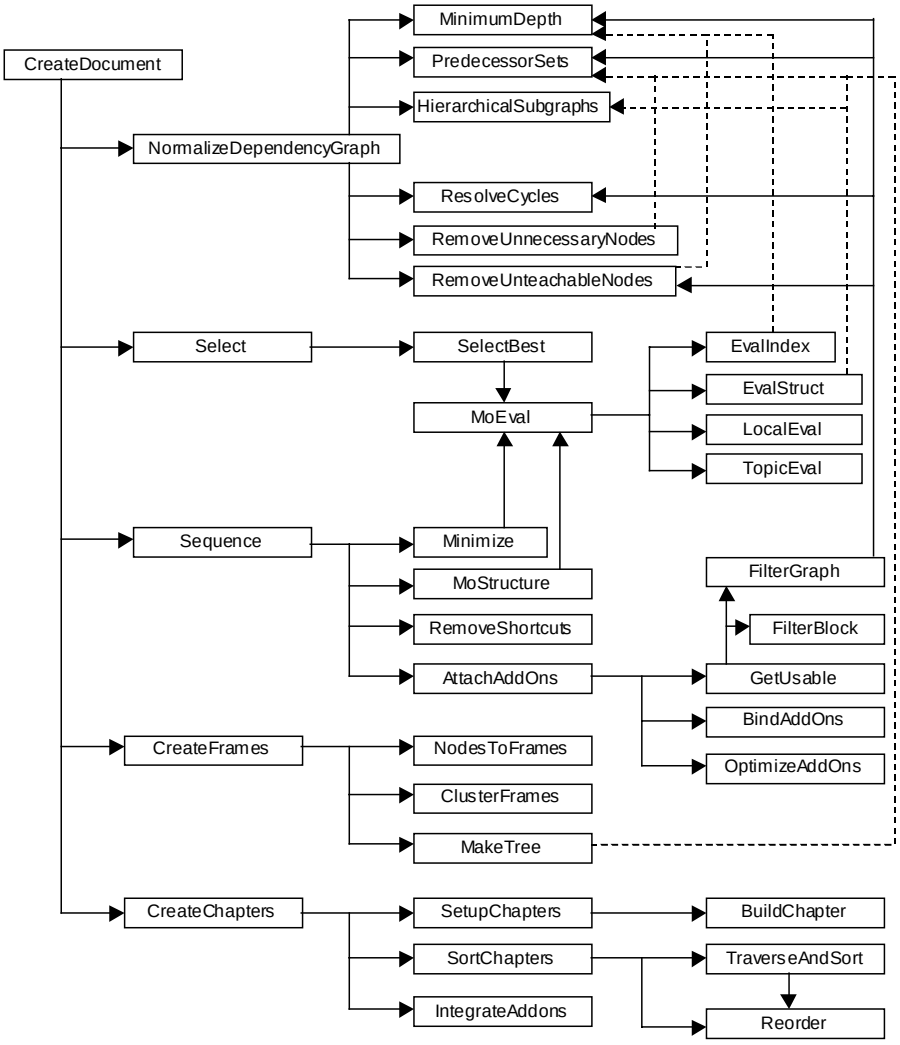
Symbole

$m \text{ --}^{G}\rightarrow o$	$m \in M^{avail}, o \in M^{avail}$	o ist direkter Nachfolger von m im Abhängigkietsgraph G	6.1.2
$m \sim^{G}\sim> o$	$m \in M^{avail}, o \in M^{avail}$	Es existiert im Abhängigkeitsgraph G ein Pfad von m zu o	6.1.4
$\{m\sim^{G}\sim>o\}$	$m \in M^{avail}, o \in M^{avail}$	Menge aller Pfade von m zu o im Abhängigkeitsgraph G	6.1.4
$m \sim^{G}\sim\sim> s$	$m \in M^{avail}, s \in S^{avail}$	Es existiert im Abhängigkeitsgraph G ein Pfad von m zu s	6.1.4
m ◊ G	$m \in M^{avail}, G \in G^{\#}$	Medienobjekt m ist im Abhängigkeitsgraph G erlernbar	6.1.5
m ◊! G	$m \in M^{avail}, G \in G^{\#}$	Medienobjekt m ist im Abhängigkeitsgraph G minimal erlernbar	6.1.5
s ◊ G	$s \in S^{avail}, G \in G^{\#}$	Stichwort s ist im Abhängigkeitsgraph G erlernbar	6.1.5
s ◊! G	$s \in S^{avail}, G \in G^{\#}$	Stichwort s ist im Abhängigkeitsgraph G minimal erlernbar	6.1.5
M' ◊◊ G	$M' \subseteq M^{avail}, G \in G^{\#}$	M' ist erlernbar in G"	6.2
M' ◊◊! G	$M' \subseteq M^{avail}, G \in G^{\#}$	M' ist minimal erlernbar in G	6.2
S' ◊◊ G	$S' \subseteq S^{avail}, G \in G^{\#}$	S' ist erlernbar in G	6.2
S' ◊◊! G	$S' \subseteq S^{avail}, G \in G^{\#}$	S' ist minimal erlernbar in G	6.2

Anhang B: Verzeichnis der Algorithmen

Das nachfolgende Diagramm stellt die Abhängigkeiten zwischen den einzelnen Algorithmen zur Auswahl und Strukturierung von Medienobjekten dar. Durchgezogene Linien zeigen dabei direkte Aufrufe an ("CreateDocument" ruft den Algorithmus "NormalizeDependencyGraph" auf), während gestrichelte Linien auf eine starke Abhängigkeit hinweisen ("EvalStruct" benötigt das Ergebnis des Algorithmus "HierarchicalSubgraphs", ohne diesen Algorithmus selbst explizit aufzurufen).

Der Pseudocode zu allen Algorithmen ist als gesondertes Dokument über den Dissertationsserver der Brandenburgischen Technischen Universität Cottbus verfügbar.



Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

Ü

V

W

Z