Ein Modell zur strukturellen Beschreibung von formatierbaren Lernmodulen für die orthogonale Konstruktion konsistenter Lerneinheiten

Zur Erlangung des akademischen Grades
DOKTORINGENIEUR (Dr.-Ing.)
der Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn
vorgelegte Dissertation
von
Dipl.-Inform. Andreas Baudry
aus Ahrensburg

Referent:	Prof. Dr.-Ing. Bärbel Mertsching
Korreferent:	Prof. Dr.-Ing. Reinhard Keil-Slawig
Tag der mündlichen Prüfung:	06.04.2006

Paderborn, den 19.04.2006

Diss. 14/218

Kurzfassung

Die Entwicklung multimedialer Lernmaterialien sowie die Konstruktion technischer Lernsysteme, hat in den letzten Jahren einen zunehmenden Einfluss auf die Hochschullehre und die betriebliche Ausbildung genommen. Neben zahlreichen Learning Management Systemen (LMS) zur Verwaltung von Kursen und Studierenden sowie technischen Systemen zur Unterstützung des Konstruktionsprozesses, sind insbesondere Standardisierungen im Kontext strukturierter Kursinhalte und deren Verwaltung durch Metadaten entstanden.

Die Entwicklung fachübergreifender Lernmaterialien setzt jedoch die Modellierung abstrakter modularer Materialien voraus, die bei konventionellen Ansätzen meist wenig oder gar nicht berücksichtigt werden. Daher stellt die Integration bestehender Ressourcen in einen Entwicklungsprozess oft ein unüberwindbares und meist kostspieliges Problem dar.

Aus diesem Grund wird in dieser Arbeit ein technisches Modell vorgestellt, das die Entwicklung technischer Systeme zur Erstellung modularer und darstellungsfreier Lernobjekte ermöglicht. Das Modell ist unter anderem durch das SCORM-Referenzmodell motiviert und erweitert das dort definierte Aggregation-Model um ein Content-Model. Während des zweistufigen Entwicklungsprozesses werden zunächst ausschließlich der Inhalt sowie die Struktur der Materialien spezifiziert. Beide werden separat in formatierbaren Bausteinen organisiert, wodurch sich frei skalierbare Aggregationslevel ergeben. In einem zweiten Schritt wird anhand spezifischer Regeln ein Präsentationsformat abgeleitet und aus den abstrakten Bausteinen ein spezialisierter Kurs generiert. Des weiteren beinhaltet das Modell einen Ansatz zur Integration bestehender Lernmaterialien, die in ihre Bestandteile Kursstruktur, Inhalt, Metadaten und Ressourcen zerlegt und auf das abstrakte Kursmodell abgebildet werden.

Im Gegensatz zu herkömmlichen Systemen, unterstützt dieser Ansatz den kooperativen Entwicklungsprozess von Teams. Abstrakt formulierte Lernbausteine und Kursfragmente werden in einem zentralen Repository mit dem Ziel koordiniert, interdisziplinäre, auf das Lernszenario angepasste Lernmaterialien, zu erzeugen.

Die Leistungsfähigkeit des Modells wird durch die Referenzimplementierung im Autorenwerkzeug Lyssa vorgestellt. Es enthält neben einem Autorensystem zur Aggregation und Transformation auch ein Werkzeug zur Layoutdefinition sowie ein zentrales Repository zur Verwaltung von abstrakten Lernmaterialien.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Zielsetzung der Arbeit 2
1.2 Eingliederung der Arbeit in das Projekt mαth-kit 3
1.3 Vorgehensweise und Aufbau der Arbeit 4

I Stand der Forschung und Technik **7**

2 Grundlagen **9**
2.1 Multimediales Lernen 9
2.1.1 Multimedia 10
2.1.2 E-Learning 11
2.2 Lerntheorie 12
2.2.1 Lernprozess 13
2.2.2 Lernparadigmen 16
2.2.3 Ein Heuristisches Lernmodell 17
2.3 Standardisierung im E-Learning 18
2.3.1 ADL (SCORM) 20
2.3.2 IMS 25
2.3.3 IEEE-LTSC (LOM) 27
2.3.4 AICC 28
2.3.5 ARIADNE 30
2.3.6 Analyse 31
2.4 Lernobjekte (Learning Objects) 31
2.4.1 Metaphern 34
2.4.2 Analyse 36
2.4.3 Granularität 37

3 Technologien **39**
3.1 Wiederverwendung von Lernmaterialien 39
3.1.1 Slicing Books 39
3.1.2 ITO-Projekt 40
3.1.3 Analyse 41

3.2 Formale Beschreibung von Lernmaterialien 42
3.2.1 SGML / XML 43
3.2.2 DocBook 47
3.2.3 OMDoc / OpenMath 48
3.2.4 MathML 50
3.2.5 Learning Material Markup Language (LMML) 51
3.2.6 Educational Modelling Language (EML) 53
3.2.7 Analyse 55
3.3 Lernplattformen – Learning Management Systeme 56
3.4 Autorenwerkzeuge 58
3.4.1 Ausgewählte Systeme 60
3.4.2 Analyse 64

4 Bewertung **67**

II Modellierung und Umsetzung **69**

5 Ein Modell zur Entwicklung konsistenter Lernmaterialien **71**
5.1 Das Baukastenprinzip 71
5.1.1 Bausteine 72
5.1.2 Kurse 73
5.1.3 Prozess 74
5.2 Formatierbare Lernbausteine 76
5.2.1 Dokumentstruktur 78
5.2.2 Spezialisierte Inhalte 83
5.2.3 Dateistruktur 85
5.3 Formatierbare Kurse 87
5.3.1 Granularität 90
5.3.2 Abstrakte Kursmodell 90
5.4 Abbildung von Lernmaterialien 94
5.4.1 Analyse 94
5.4.2 Abbildung auf das Kursmodell 96
5.4.3 Nachbearbeitung 98
5.5 Transformation 99
5.5.1 Aggregation 99
5.5.2 Transformationspaket 100
5.5.3 Prozessoren 105
5.5.4 Layout 105
5.6 Kooperation 109
5.6.1 Der kooperative Entwicklungsprozess 110
5.6.2 Operationen 111

6 Umsetzung **115**
6.1 Das Autorenwerkzeug Lyssa . . . 115
6.1.1 Systembeschreibung . . . 116
6.1.2 Formatierbare Bausteine . . . 118
6.1.3 Formatierbare Kurse . . . 124
6.1.4 Das Kursmodell . . . 127
6.1.5 Abbildung bestehender Formate . . . 130
6.1.6 Transformation . . . 135
6.1.7 Framework . . . 138
6.2 Das Layoutwerkzeug Lyssa-Designer . . . 140
6.2.1 Implementierung . . . 143
6.3 Construction Kit Server . . . 145
6.3.1 Slide . . . 147

7 Beispiele **149**
7.1 Technische Informatik - Exemplarische Kursentwicklung . . . 149
7.1.1 Analyse . . . 150
7.1.2 Aufbau einer Lerneinheit . . . 150
7.1.3 Kurskonstruktion . . . 150
7.2 Wiederverwendung bestehender Skriptteile . . . 160
7.3 Exemplarische Entwicklung eines Formats zur Präsentation . . . 162
7.3.1 Festlegung des Layouts . . . 162
7.3.2 Festlegung des Formats . . . 163
7.4 Ausgewählte Bausteine . . . 166

III Analyse **173**

8 Zusammenfassung **175**
8.1 Evaluation . . . 176

9 Bewertung **179**

10 Ausblick **183**

IV Anlagen **185**

A Abkürzungsverzeichnis **187**

B Glossar **189**

C Attribute **193**

Literaturverzeichnis **197**

Abbildungsverzeichnis

2.1 Begriffsbeschreibung (vgl. [Kerres98], Seite 14) . . . 11
2.2 Lernparadigmen (Quelle: [Baumgartner97], Seite 5) . . . 18
2.3 Ein Heuristisches Lehr- und Lernmodell (Quelle: [Baumgartner97], Seite 8) 19
2.4 Kooperation internationaler Standardisierungsgremien und -Organisationen 20
2.5 Beispiel: Organisation . . . 22
2.6 Beispiel: Content Package . . . 23
2.7 SCORM Laufzeitumgebung . . . 24
2.8 IMS Digital Repository . . . 26
2.9 Variable Kursstruktur für Lernmodule zum leichten Transfer von einem CMI-System auf ein anderes System. . . . 30
2.10 Das modulare Prinzip der Reusable Learning Objects (RLO) . . . 32
2.11 Modulare Content-Hierarchie . . . 33
2.12 Aggregationslevel für Lernobjekte nach dem IEEE Standard für Metadata (LOM) . . . 37

3.1 System Modules of Slicing Books . . . 40
3.2 Import . . . 41
3.3 Export . . . 42
3.4 Syntax eines XML-Elements . . . 44
3.5 Eindeutige Zuordnung von Elementbezeichnern durch Präfixnotation . . . 45
3.6 Beispiel: XSL-Transformation einer einfachen XML-Struktur in eine FO-Repräsentation . . . 47
3.7 Teachware Modell . . . 52
3.8 XML-Schema Bindung des EML Rahmenwerkes . . . 54
3.9 Idealtypische Architektur eines Learning Mangement Systems . . . 57
3.10 Verschiedene Kategorien von Autorenwerkzeugen . . . 59
3.11 Toolbook . . . 60
3.12 Authorware . . . 62
3.13 Macromedia Dreamweaver MX . . . 63

5.1 Basisformat für Lernbausteine . . . 73
5.2 Basisformat für Kurse . . . 74
5.3 Interaktion der Rollen . . . 76

5.4 Struktur eines formatierbaren Bausteins 78
5.5 Syntaktische Struktur des LOB-Elements 79
5.6 Textgruppe 80
5.7 Strukturgruppe 82
5.8 Mdeia Gruppe 84
5.9 Lerngruppe 85
5.10 Mathematikgruppe 86
5.11 IMS Content Package Information Model 87
5.12 Formatierbarer Kurs 88
5.13 Verschachtelte Kurse 91
5.14 Kursmodell Spezifikation 92
5.15 Abbildung eines Kurses auf die Bausteinstruktur 93
5.16 Wiederverwendung von Lernmaterialien 95
5.17 Abbildung auf das Kursmodell 97
5.18 Kursaggregation 99
5.19 Transformationspaket 102
5.20 Modell des Transformationspakets 103
5.21 Abbildung 104
5.22 Prozessorpipeline 106
5.23 Rollen 107
5.24 Abbildung des Layouts auf die Transformatoren 108
5.25 Spezifikation des Layouts 110
5.26 Der Construction Kit Server 112

6.1 Das Autorenwerkzeug *Lyssa* 117
6.2 Ansicht des Bausteinfensters 118
6.3 Der Export-Dialog 119
6.4 UML-Diagramm für ein OO-Binding-Beispiel 123
6.5 UML-Diagramm einer abstrakten Fabrik für die Kursumwandlung 127
6.6 Sequenzielle Darstellung eines Objekt Diagramms zur Veranschaulichung einer *bottom-up* Kurskonstruktion 129
6.7 UML-Klassendiagramm des Kursmodells 130
6.8 OpenOffice Screenshot 132
6.9 Datenfluss eines Latex-Dokuments bei der Konvertierung in einen Formatierbaren Kurs. 135
6.10 UML-Diagramm der Transformation Package Klasse 136
6.11 Framework vs. Klassenbibliothek 139
6.12 Das Framework 140
6.13 Lyssa Designer – Konfigurationsansicht 142
6.14 Lyssa Designer – Entwicklungsansicht 143
6.15 Lyssa Designer – Grundeinstellung 144
6.16 UML-Diagramm des Designers 144

6.17 CKS Browser – Ermöglicht das Navigieren in verteilten Dateisystemen und die Überwachung nebenläufiger Aktionen 146
6.18 CKS Administration – Eine Eingabemaske, mit der Benutzer und Benutzerinnen angelegt und gelöscht werden. 146
6.19 Schichtenmodell des Construction Kit Server 147

7.1 Basisformat für Kurse 151
7.2 Kursmaterialien für die Präsenzveranstaltung *Technische Informatik* in der Grundstudiumsausbildung 152
7.3 Der Baustein *Minimierungsverfahren* zusammengestellt aus einer Content-Datei und mehreren Abbildungen 153
7.4 Das Autorenwerkzeug Lyssa mit dem geöffneten Kurs *Grafische Minimierung* 154
7.5 Der Kurs *Grafische Minimierung* übersetzt mit einem Transformationspaket für die Erstellung eines GET-Lab-Kurses. 155
7.6 Ansicht einer interaktiven Übung zur Minimierung mit KV-Diagrammen . . 156
7.7 Der Kurs Wechselstromschaltungen mit Zugriff auf den CKS 159
7.8 Präsentationsfolie *Sinusförmige Signale* aus dem Skript Systemtheorie . . . 160
7.9 Die Kursansicht von Lyssa mit dem importierten Systemtheorie-Skript . . . 161
7.10 Sinusförmige Signale als Onlinekurs. Bei dieser Präsentation wurde das mαth-kit Layout verwendet 162
7.11 Screenshot des Lyssa-Designers 166
7.12 Der Kurs *Technische Informatik* als PDF-Dokument 167
7.13 Wahrscheinlichkeitstabelle des Bausteins Kodierungsverfahren nach Huffman 168
7.14 Konstruktionsbreich des Bausteins Kondierungsverfahren nach Huffman . . 168
7.15 Auswertungsbereich des Bausteins Kondierungsverfahren nach Huffman . . 169
7.16 Baustein zur Zahlenumwandlung 170
7.17 Baustein Quiz-Umgebung Komplexe Zahlen 170
7.18 Überlagerung zweier harmonischer Schwingungen 171
7.19 Rechnen mit komplexen Zahlen (Kartesische Koordinaten) 171

Tabellenverzeichnis

2.1 Katalog mit kategorisierten Aspekten der Problembeschreibung 10
2.2 Fertigkeitsstufen 15
2.3 Neun Kategorien für Lernobjekte nach der Learning Object Metadata Spezifikation (LOM) 28

3.1 Kategorien für Elemente zur Strukturierung von DocBook Dokumenten . . 48
3.2 Gegenüberstellung der analysierten Systeme 66

5.1 Attribute des `cell`-Elements 81
5.2 Attribute des `mmo`-Elements 83
5.3 Das Item Element der IMS Content Package Spezifikation 89
5.4 Layouttypen 109

6.1 Arbeitsteilung für Systemkomponenten 116
6.2 Abbildung Strukturelemente auf XML-Elemente 120
6.3 Abbildung zwischen OpenOffice-Elementen und denen des Kursmodells . . 133
6.4 Abbildung zwischen OpenOffice-Elementen und dem Kursmodell 134

7.1 Integration der Komplexen Zahlen in unterschiedliche Fachrichtungen . . . 157

C.1 Attribute und Elemente des Transformationspakets 194
C.2 Attribute und Elemente des Kursmodells 195
C.3 Attribute und Elemente des Layouts 196

Kurzfassung

Die Entwicklung multimedialer Lernmaterialien sowie die Konstruktion technischer Lernsysteme, hat in den letzten Jahren einen zunehmenden Einfluss auf die Hochschullehre und die betriebliche Ausbildung genommen. Neben zahlreichen Learning Management Systemen (LMS) zur Verwaltung von Kursen und Studierenden sowie technischen Systemen zur Unterstützung des Konstruktionsprozesses, sind insbesondere Standardisierungen im Kontext strukturierter Kursinhalte und deren Verwaltung durch Metadaten entstanden.

Die Entwicklung fachübergreifender Lernmaterialien setzt jedoch die Modellierung abstrakter modularer Materialien voraus, die bei konventionellen Ansätzen meist wenig oder gar nicht berücksichtigt werden. Daher stellt die Integration bestehender Ressourcen in einen Entwicklungsprozess oft ein unüberwindbares und meist kostspieliges Problem dar.

Aus diesem Grund wird in dieser Arbeit ein technisches Modell vorgestellt, das die Entwicklung technischer Systeme zur Erstellung modularer und darstellungsfreier Lernobjekte ermöglicht. Das Modell ist unter anderem durch das SCORM-Referenzmodell motiviert und erweitert das dort definierte Aggregation-Model um ein Content-Model. Während des zweistufigen Entwicklungsprozesses werden zunächst ausschließlich der Inhalt sowie die Struktur der Materialien spezifiziert. Beide werden separat in formatierbaren Bausteinen organisiert, wodurch sich frei skalierbare Aggregationslevel ergeben. In einem zweiten Schritt wird anhand spezifischer Regeln ein Präsentationsformat abgeleitet und aus den abstrakten Bausteinen ein spezialisierter Kurs generiert. Des weiteren beinhaltet das Modell einen Ansatz zur Integration bestehender Lernmaterialien, die in ihre Bestandteile Kursstruktur, Inhalt, Metadaten und Ressourcen zerlegt und auf das abstrakte Kursmodell abgebildet werden.

Im Gegensatz zu herkömmlichen Systemen, unterstützt dieser Ansatz den kooperativen Entwicklungsprozess von Teams. Abstrakt formulierte Lernbausteine und Kursfragmente werden in einem zentralen Repository mit dem Ziel koordiniert, interdisziplinäre, auf das Lernszenario angepasste Lernmaterialien, zu erzeugen.

Die Leistungsfähigkeit des Modells wird durch die Referenzimplementierung im Autorenwerkzeug Lyssa vorgestellt. Es enthält neben einem Autorensystem zur Aggregation und Transformation auch ein Werkzeug zur Layoutdefinition sowie ein zentrales Repository zur Verwaltung von abstrakten Lernmaterialien.

Listings

3.1 Beispiel einer XML-Deklaration 43
3.2 Entity Notation 45
3.3 Beispiel einer Element-Definition 46
3.4 DocBook Beispiel für einen einfachen Artikel 48
3.5 OMDoc Repräsentation einer Definition 49
3.6 MathML-Beispiel einer Formel mit presentation encoding 50
3.7 MathML-Beispiel einer Formel mit content encoding 51
3.8 Ein einfaches LMML-Beispiel 53
5.1 Algorithmus Disaggregation 100
6.1 Ein einfaches BrickML-Beispiel 120
6.2 BrickML Tabellen-Definition mit XML-Schema 121
6.3 Beispiel eins Manifests für einen Grundlagenkurs Technische Informatik. . . 124
6.4 Übersetzungsregel eines Listenelements mit XSL beschrieben 137
7.1 Dieses einfache Beispiel stammt aus einem Baustein zur Erläuterung des Minimierungsverfahrens nach Quine-McCluskey und veranschaulicht ein XML-basiertes Lernobjekt 151
7.2 Definition eines Layouts 163
7.3 XSL:FO-Dokument für einen PDF-Kurs 164
7.4 XSL:FO-Liste für einen PDF-Kurs 165

Kapitel 1

Einleitung

Seit der rasanten Entwicklung des Internets hat sich der menschliche Lernprozess grundlegend verändert. Wo früher teure und langwierige Experimente durchgeführt werden mussten, können heute Methoden, Verfahren und Ergebnisse weltweit innerhalb weniger Sekunden abgerufen und evaluiert werden. Komplexe Probleme können aufgrund des großen Fundus an Wissen schnell und effektiv gelöst werden. Hierbei unterstützen Wissensdatenbanken ebenso wie Disskusionsforen, in denen Fragestellungen mit Experten und Expertinnen analysiert werden.

Das Fachgebiet, das einerseits aus der Kombination der technischen Möglichkeiten durch Computer und Netzwerke und andererseits durch das pädagogische Lernen entstanden ist, wird als *E-Learning* bezeichnet. Der Einsatz von E-Learning-Anwendungen weitet sich zunehmend auf Privathaushalte, Universitäten, Schulen und Firmen aus, die ihre Mitarbeiter und Mitarbeiterinnen bzw. Lernende effizienter schulen und ausbilden können.

Der in den letzten zehn Jahren entstandene E-Learning-*Hype* hat sich auch beträchtlich auf die Forschung ausgewirkt. Zunehmend werden nationale und internationale Forschungsprojekte ins Leben gerufen, die sich mit Fragestellungen und Problemen zur Erstellung und Anwendung von E-Learning-Inhalten befassen. In Deutschland hat das Bundesministerium für Bildung und Forschung (BMBF) im Jahr 2000 das Schwerpunktprogramm "Neue Medien in der Bildung" gestartet und fördert seither mehr als 100 Projekte, an denen Hochschulen und Unternehmen gleichermaßen beteiligt sind.

Im Rahmen von Forschungsprojekten, der Universitätslehre und der betrieblichen Fortbildung, sind in den letzten Jahren zahlreiche E-Learning-Inhalte entstanden, die die individuelle Ausbildung unterstützen und verbessert haben. Oft sind multimediale Inhalte durch den Einsatz hoher Personalkosten entwickelt worden, wodurch Problemen, wie Nachhaltigkeit und Wiederverwendbarkeit, wenig Beachtung geschenkt wurde. Obwohl es im Hochschulbereich zahlreiche Überschneidungen bei Themen in der Grundlagenausbildung gibt, sind die meisten Inhalte jedoch derart aufbereitet, dass sie nicht in verwandten Fachdisziplinen eingesetzt werden können. Daraus resultiert, dass E-Learning-Inhalte mehrfach für dasselbe Thema produziert werden und außerdem nicht in bestehende Inhalte integriert werden können. Das liegt insbesondere an der individuellen Gestaltung und Aufbereitung, der Berücksichtigung didaktischer Aspekte sowie an der Festlegung des Granularitätsle-

vels bei der Themenzusammenstellung. Obwohl es theoretische Überlegungen, wie z. B. die Formulierung der Inhalte als Lernobjekte [Wiley00b] und technische Rahmenbedingungen, wie Standardisierungen (vgl. Kapitel 2.3) im E-Learning-Bereich gibt, fehlt es dennoch an Methoden und Verfahren Kurse derart zu gestalten, dass Autoren und Autorinnen bereits bestehende Materialien nahtlos übernehmen und auf ihre Bedürfnisse anpassen können.

Aus diesem Bedarf heraus ist es Ziel dieser Arbeit, ein Modell zu entwickeln, nach dem es möglich ist, E-Learning-Inhalte semantisch zu modellieren, ohne sich auf individuelle Ausprägungen, didaktische Konzepte oder gar auf spezielle Anwendungen festzulegen.

1.1 Zielsetzung der Arbeit

Ziel dieser Arbeit ist die Entwicklung eines Modells zur formalen Beschreibung von E-Learning-Inhalten, das Autoren und Autorinnen eine nachhaltige Generierung von Inhalten ermöglicht. Der Bedarf eines solchen Modells hat sich gerade am Anfang des Projekts mαth-kit (vgl. Abschnitt 1.2) herausgestellt, da Inhalte der Mathematik fachübergreifend entwickelt werden sollten und es keinen Mechanismus zur modularen Konstruktion gab, der die Integration ermöglicht hätte. Ohne ein entsprechendes Modell, wären individuell gestaltete Einzellösungen produziert worden, die mit den Mitteln des World Wide Web (WWW) durch Hyperlinks zusammengesetzt worden wären. Aus diesem Bedarf heraus, soll in dieser Arbeit ein theoretisches Modell zur Konstruktion solcher E-Learning-Inhalte entwickelt werden und die technische Realisierung im Projekt mαth-kit prototypisch zum Einsatz kommen.

Das dargelegte Problem tritt naturgemäß nicht nur im Projekt mαth-kit auf, sondern lässt sich ebenso auf anderer Projekte übertragen. Deshalb wird die Forschungsfrage allgemein formuliert:

> "Wie lassen sich E-Learning-Inhalte unabhängig von einer individuellen Darstellung, einem didaktischen Konzept und einem Präsentationsformat formulieren, damit sie von anderen Autoren und Autorinnen nahtlos integriert und unabhängig vom ausgewählten Lernszenario eingesetzt werden können?"

Aus dieser allgemeinen Fragestellung lassen sich weitere, detaillierte Fragestellungen ableiten, die das Gesamtziel deutlicher darlegen:

- Wie können Lerneinheiten kontextunabhängig mit anderen Lerneinheiten aggregiert werden, ohne sich während des Konstruktionsprozesses auf ein konkretes Präsentationsformat festzulegen (Formatierbarkeit)?
- Wie lassen sich Struktur und Inhalt von Lerneinheiten formal von ihrer Präsentation entkoppeln (Modularität)?
- Welche Granularität von Lernmodulen sollte für eine Wiederverwendung gewählt werden?

- Wie müssen Lerneinheiten beschaffen sein, damit sie in verschiedenen Fachdisziplinen einsetzbar sind (Interdisziplinarität)?
- Durch welche Maßnahmen können Lerneinheiten nachhaltig von Bestand sein?
- Welche Mechanismen sind notwendig, um bereits existierende Lernmaterialien in den Entwicklungsprozsess mit einzubeziehen (Intergrierbarkeit)?
- Wie kann der kooperative Entwicklungsprozess für E-Learning-Inhalte in Projektteams unterstützt werden (Kooperation)?
- Auf welche technische und pädagogische Weise findet die Wissensvermittlung mit E-Learning Materialien statt und welche Methoden und Paradigmen müssen modelliert werden?
- Welchen Einfluss haben didaktische Aspekte für den Entwicklungsprozess? Hierbei ist festzustellen, wie der Dialog zwischen Lernmaterialen und Anwendern bzw. Anwenderinnen stattfindet.

Wie bereits erwähnt wurde, setzt sich der Themenbereich E-Learning nicht nur aus technischen Systemen oder Werkzeugen zur Gestaltung von Inhalten zusammen, sondern auch aus didaktischen und pädagogischen Methoden und Verfahren. Der Fokus dieser Arbeit richtet sich vor allem auf die technischen Aspekte der Modellierung und lässt pädagogische Gesichtspunkte, sofern sie nicht von zentralem Interesse sind oder notwendig für die Entwicklung des technischen Modells, weitestgehend außer acht. Ferner besteht auch kein Interesse an einem Verfahren, welches auf linguistischer Ebene Inhalte analysiert und bewertet. Vielmehr soll das entwickelte Modell bei der technischen Modellierung von Softwaresystemen und der Konstruktion von Inhalten herangezogen werden, damit sie kooperativ und nachhaltig genutzt werden können.

1.2 Eingliederung der Arbeit in das Projekt mαth-kit

Die in dieser Arbeit produzierten Ergebnisse und durchgeführten Untersuchungen sind im Rahmen des Forschungsprojekts *mαth-kit* [1] [Mat05a] entstanden, welches im März 2001 durch das Bundesministerium für Bildung und Forschung (BMBF) im Schwerpunktprogramm “Neue Medien in der Bildung” ins Leben gerufen wurde. Im Mittelpunkt des Vorhabens stand die Konzeption und Entwicklung eines Baukastensystems für multimediale Lerninhalte mit dem Schwerpunkt Mathematik. Die verfolgten Projektziele waren die Verbesserung der Grundstudiumsausbildung durch den Einsatz von Lerneinheiten zur Exploration, Übungen mit direkter Erfolgskontrolle und Lerneinheiten zur Präsentation in Lehrveranstaltungen. Die in einem virtuellen Baukasten aufbewahrten Lerneinheiten und Lernmodule sollten wie in einem richtigen Baukasten gelagert und verwaltet werden,

[1] Gefördert vom Bundesministerium für Bildung und Forschung (BMBF), Projekt math-kit (08NM084)

um sie in Lehrmaterialien verschiedener Fachdisziplinen[2], basierend auf einem Fundus mathematischer Grundlagen, einsetzen zu können. Die Inhalte sind im Rahmen eines Verbundprojekts der Universitäten Hamburg, Bayreuth, Paderborn und Hagen entstanden, die unterschiedliche Expertisen und Erfahrungen einbrachten.

Zunächst an der Universität Hamburg, später an der Universität Paderborn fortgeführt, wurde ein Baukastensystem zur allgemeinen Verwaltung modularer Lerneinheiten entwickelt und Lerneinheiten für die Grundstudiumsveranstaltung *Technische Informatik* entwickelt. Zudem wurden an der Universität Paderborn Materialien für die Veranstaltung *Mathematik für Maschinenbauer* produziert.

Darüber hinaus verbesserte das Projekt mαth-kit auch die Lehre an der Fernuniversität Hagen. Die dort ansässige Gruppe entwickelte spezielle Lernmaterialien zum Thema *Lineare Algebra*, die spezielle Aspekte des Fernstudiums berücksichtigen. Damit die entwickelten Inhalte Studierenden zugänglich waren und der Dialog zwischen Anwendern und Lernmaterialen stattfand, evaluierte die Gruppe der Universität Bayreuth die Inhalte und brachte Beiträge zur didaktischen Gestaltung ein. Neben universitären Instituten lieferte auch eine industrielle Einrichtung dem Projekt wichtige Ergebnisse. So wurde von der Firma Sci-Face, im Rahmen der Projektlaufzeit, ein Computing Server (MCS) für das Mathematikprogramm *MuPad* umgesetzt, der für die Berechnung komplexer mathematischer Formeln herangezogen werden konnte.

Das in dieser Arbeit vorgestellte Modell liefert einen wichtigen Beitrag zu dem im Projekt mαth-kit entwickelten Baukastensystem. Es beschreibt auf theoretischer und konzeptioneller Ebene einen Rahmen, der als Grundlage für die Implementierung eines Autorensystems gilt, welches von den am Projekt beteiligten Arbeitsgruppen eingesetzt und bewertet wurde.

1.3 Vorgehensweise und Aufbau der Arbeit

Diese Arbeit dokumentiert die Vorgehensmethodik und den Arbeitsprozess der zur Lösung der in Kapitel 1.1 beschriebenen Problemstellung gewählt wurde. Zu Beginn der Arbeit steht die Forschungsfrage im Mittelpunkt der Untersuchung. Es wird überprüft, welche Ergebnisse hinsichtlich der gestellten bzw. aufgeworfenen Fragen bereits erzielt worden sind und anschließend bewertet. Die gewonnen Resultate fließen in die konzeptionelle Phase und die Umsetzung des Modells mit ein. Damit das Modell evaluiert und bewertet werden kann, wird ein Prototyp entwickelt, an dem die Richtigkeit sowie die Praxistauglichkeit abgeleitet werden kann. Abschließend findet eine Bewertung der erzielten Ergebnisse statt, die in Relation zu den aufgeworfenen Fragen diskutiert wird.

In Kapitel 2 und Kapitel 3 wird der aktuelle Stand der Forschung in Zusammenhang mit der in diesem Kapitel gestellten Forschungsfrage dargelegt. Für die Analyse und die abschließende Bewertung wird ein Kriterienkatalog herangezogen, mit dem sich Grundla-

[2]Im Zeitraum der Projektlaufzeit sollen multimediale Lerneinheiten zur Evaluierung in den Veranstaltungen *Mathematik für Maschinenbauer*, *Mathematik für Informatiker*, *Digitale Systeme* und in den Fernstudiumskurs *Lineare Algebra* eingesetzt werden.

gen, Verfahren und bestehende Systeme einschätzen und bewerten lassen. Hierbei werden Forschungsergebnisse, die sich mit den Themen Formatierbarkeit, Modularität, Granularität, Interdisziplinarität und Nachhaltigkeit von Lernmaterialien befassen, ausgewertet und analysiert. Es werden sowohl technische als auch theoretische Grundlagen, die zum Verständnis der beschriebenen Systeme beitragen, vorgestellt und erörtert. Hierzu zählen neben lerntheoretischen Grundlagen die Definitionen der Begriffe *Multimedia* und *E-Learning*. Die abschließende Bewertung aktueller Arbeiten dient als Grundlage für die Modellierung und Implementierung.

Kapitel 5 beschreibt das Modell zur konsistenten Kursentwicklung von E-Learning-Inhalten. Als Basis für die Modellierung dient das Baukastenprinzip, welches aus dem Kontext und der Aufgabenstellung des Projekts mαth-kit für einen technischen Lösungsansatz abgeleitet wird. Ausgehend von diesem Prinzip wird zunächst die Konstruktion formatierbarer Lernbausteine beschrieben. An diesem Punkt fließen neben technischen auch pädagogische Überlegungen in die Modellierung mit ein. Weiterhin wird auf die Kurskonstruktion eingegangen und ein Konzept zur abstrakten Beschreibung von Inhalten vorgestellt, welches vor allem für die Integration bereits existierender digitaler Lehrmaterialien und Onlinekurse benötigt wird. Weiterhin wird gezeigt, wie Kurse von Studierenden und Lehrenden in Abhängigkeit eines präferierten Lernszenarios publiziert und angewendet werden können. Insbesondere wird das *Transformationspaket* vorgestellt, mit dessen Hilfe sich individuelle und gestalterische Elemente separieren lassen. Abschließend wird auf den Entwicklungsprozess kooperierender Projektteams eingegangen und dargelegt, wie die kooperative Entwicklung von Lerninhalten, unterstützt durch eine dezentrale Datenhaltung, mit dem Kursmodell verbessert und effizienter gestaltet werden kann.

Kapitel 6 stellt den Prototyp vor, der zur Realisierung der theoretischen Überlegungen implementiert worden ist. Der Prototyp ist ein Autorenwerkzeug zur Gestaltung von formatierbaren Lerninhalten, die adaptiert und publiziert werden können. Neben dem Autorenwerkzeug ist ein weiteres Werkzeug zur Entwicklung von Präsenationsformaten und deren Layouts entstanden sowie ein *Server* zur kooperativen Entwicklung. Zusätzlich stehen die technischen Aspekte, die bei der Umsetzung des Prototyps herausgearbeitet wurden, im Vordergrund und werden im jeweiligen Kontext erörtert.

In Kapitel 7 wird exemplarisch ein Kurs zum *Technische Informatik* vorgestellt, im Rahmen des Projekts mαth-kit erstellt wurde. Es wird der Einfluss der erzielten Ergebnisse auf den Erstellungsprozess diskutiert und der daraus entstehende Mehrwert verdeutlicht. Dabei wird der kooperative Entwicklungsprozess und die damit verbundene projektübergreifende Nutzung von Lernmaterialien verdeutlicht.

Nach der Zusammenfassung in Kapitel 8, werden in Kapitel 9 die Resultate der Arbeit in Anlehnung an die Aufgabenstellung bewertet. Es wird auf die Erfahrungen im Umgang mit dem Prototypen und auf Kooperationen mit industriellen Partnern zurückgegriffen. Insbesondere wird gezeigt, dass der Prototyp die zu Beginn der Projektlaufzeit auftretenden Probleme löst und ein effektives und effizientes Arbeiten ermöglicht.

Abschließend wird in Kapteil 10 ein Ausblick auf weitere interessante Fragestellungen gegeben, die im zeitlichen Projektrahmen nicht weiter untersucht werden konnten.

Teil I

Stand der Forschung und Technik

Kapitel 2

Grundlagen

In diesem Kapitel werden zunächst die für diese Arbeit relevanten Grundlagen und Forschungsergebnisse auf dem Gebiet E-Learning dargelegt. Zu Beginn werden die grundlegenden Begriffe *Multimedia* und *E-Learning* eingeführt. Es soll ein Grundverständnis der gängigen Lerntheorien entwickelt werden, um eine Charakterisierung und Zuordnung moderner Lernsysteme leichter durchzuführen. Zu diesen theoretischen Grundlagen wird der aktuelle Stand der Forschung im Bereich der Standardisierung von E-Learning-Inhalten vorgestellt, der sich über die Jahre hin aus den Ergebnissen zahlreicher Forschungsprojekte entwickelt hat und deren Ergebnisse sich mit der Unterstützung unterschiedlicher Gremien gerade zu etablieren beginnen. Einige dieser wissenschaftlichen Arbeiten beschäftigen sich mit Lernobjekten, die für das in dieser Arbeit vorgestellte Modell eine solide Grundlage bilden und einen wichtigen Beitrag für den Modellierungsprozess leisten.

Damit die Analyse bestehender Systeme, Plattformen, Standardisierungen, Materialien und Theorien im Kontext des in dieser Arbeit verfolgten Ziels stehen, werden zu den in Kapitel 1 aufgeworfenen Fragestellungen Bewertungskriterien definiert. Auf diese wird bei den anstehenden Analysen sowie bei der abschließenden Bewertung Bezug genommen, so dass der Zusammenhang zum eigentlichen Thema dieser Arbeit hergestellt wird. Tabelle 2.1 zeigt die aus der Zielsetzung abgeleiteten Bewertungskriterien für diesen Teil der Arbeit.

2.1 Multimediales Lernen

Bei der Analyse von Autorensystemen, Lernplattformen (vlg. Kapitel 3.3) und multimedialen Lernangeboten gibt es eine Vielzahl unterschiedlicher Systeme und Produkte. Bei genauerer Betrachtung wird schnell ersichtlich, dass sie sowohl in ihrer Anwendung als auch in ihrer Qualität erhebliche Unterschiede aufweisen. Je nach Anwendungskontext unterstützen solche Systeme den Erstellungsprozess von Lehrinhalten und deren Verwaltung. Um sie bewerten zu können und im Verlauf dieses Kapitels Position zu beziehen, sollen die Fragen “Was ist multimediales Lernen?” und “Wie lernt der Mensch?” beantwortet werden. Hierzu ist es zwingend notwendig, die Begriffe Multimedia und E-Learning [Frerich00, Lytras, Reglin03] genau zu definieren. Dies ist jedoch keineswegs einfach, da die

Bewertungskriterien	Fragestellungen
Formatierbarkeit	Besteht die Möglichkeit, die Struktur der Lerneinheit von der Präsentation zu entkoppeln?
Modularität	Können Lerneinheiten mit anderen kontextunabhängig aggregiert werden?
Granularität	Wie lässt sich die Granularität der Lernmodule justieren?
Interdisziplinarität	Ist dieselbe Lerneinheit in verschiedenen Kontexten einsetzbar?
Nachhaltigkeit	Können Lerneinheiten nachhaltig bestehen?
Integrierbarkeit	Ist es möglich, bestehende Ressourcen wiederzuverwenden?
Kooperation	Wird die kooperative Arbeit in Projektteams unterstützt?
Mathematik	Wie werden mathematische Lehrinhalte behandelt?
Didaktik	Wie ist der Dialog zwischen Lerneinheiten bzw. Plattformen und Lernenden definiert und welche Lernformen, also die Art der Wissensvermittlung, werden unterstützt ?

Tabelle 2.1: Katalog mit kategorisierten Aspekten der Problembeschreibung

Begriffe nicht immer einheitlich verwendet werden, sondern speziell formuliert sind, d. h. auf eine bestimmte Anwendungsdomäne bezogen. Aus diesem Grund soll in den folgenden Abschnitten eine einheitliche Sicht auf die grundlegenden Begriffe erarbeitet werden.

2.1.1 Multimedia

Für den Begriff Multimedia existieren abhängig vom Betrachtungswinkel unterschiedliche Definitionen. Grundsätzlich beschreibt der Begriff das kombinierte Auftreten verschiedener visueller und akustischer Medien. Aus technischer Sicht definiert Negroponte [Negroponte96] den Begriff Multimedia als die Vermischung digitaler Daten wie Bilder, Animationen oder Musik. Neben diesem technischen Ansatz gibt es auch die informationstechnische Betrachtungsweise, bei der der Begriff Multimedia nicht nur die Datentypen einschließt, sondern speziell den Informationsbegriff berücksichtigt [Schulmeister96]. Hiermit ist insbesondere die Informationsverarbeitung des Benutzers bzw. der Benutzerin gemeint, die aus den unterschiedlichen Daten erst Informationen ableiten. Ein weiterer Aspekt konzentriert sich auf die Interaktivität von Multimedia. Damit ist vor allem das interaktive Einwirken durch Anwender und Anwenderinnen auf den sequenziellen Ablauf von Multimedia gemeint. Schulmeister fasst diese unterschiedlichen Sichtweisen auf den Multimediabegriff zu einer allgemeinen Definition zusammen:

> "... dass Multimedia als eine interaktive Form des Umgangs mit symbolischem Wissen in einer computergestützten Interaktion betrachtet werden muss ([Schulmeister96], Seite 18)."

Kerres [Kerres98] betrachtet bei seiner Definition von Multimedia außerdem das Medium, welches sich nicht nur durch analoge oder digitalisierte Daten auszeichnet, sondern

ebenso durch das Übertragungsmedium, wie z. B. das Kabelnetz, Telefonnetz oder das Internet. Diese Gruppe von Medien werden als Telemedien bezeichnet und sind zum größten Teil disjunkt mit Multimedien, da sie sich schlichtweg auf Übertragungswege und Techniken reduziert.

2.1.2 E-Learning

Der Begriff E-Learning wurde Mitte der neunziger Jahre geprägt und bezeichnet im Kern die Wissensvermittlung mit Hilfe elektronischer Geräte. Hinzu kommt die Unterstützung des Lernprozesses durch unterschiedliche Medientypen, wie z. B. Multimedia oder Telemedien. Die ursprüngliche Form der computergestützten Wissensvermittlung reicht bis in die sechziger Jahre zurück, in denen Skinner und Holland Lernmaschinen entwickelt haben, die Wissen auf Textbasis vermittelten, das anschließend durch gezielte Fragen überprüft wurde [Niegemann03]. In den achtziger Jahren ist der Begriff CBT (Computer Based Training) eingeführt worden, der als allgemeiner Überbegriff für die Wissensvermittlung mit Computern verstanden wird. Erst mit der Entwicklung des World Wide Web (WWW) hat sich eine neue Dimension der Wissensvermittlung ergeben. Seither konnten computergestützte Lernprogramme über das Internet übertragen werden. Seitdem haben Lernende die Möglichkeit, mit anderen Lernenden in Kontakt zu treten, Erfahrungen auszutauschen und gemeinsame Fragestellungen zu ergründen (kooperatives Lernen). Diese Art der Wissensvermittlung wird als WBT (Web Based Training) bezeichnet.

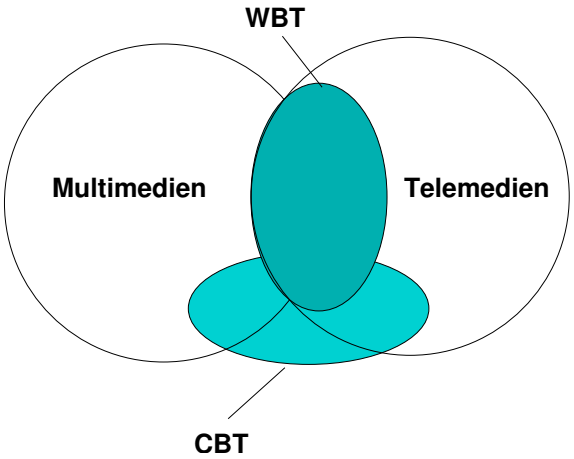


Abbildung 2.1: Begriffsbeschreibung (vgl. [Kerres98], Seite 14)

Der Begriff E-Learning wird also als Oberbegriff für alle Arten der Wissensvermittlung über das Internet verstanden [Kerres98]. In der Tat lässt sich der Begriff nur vage definieren, weil er neben pädagogischen, sozialen und didaktischen Aspekten auch technische mit einschließt. Demnach sollen zwei grundlegende Begriffsdefinitionen vorgestellt werden:

> "... e-Learning als einen übergeordneten Begriff für softwareunterstütztes Lernen verstehen ([Baumgartner02], Seite 5)".

> ”e-Learning findet statt, wenn Lernprozesse in Szenarien ablaufen, in denen gezielt multimediale und (tele-) kommunikative Technologien integriert sind [Seufert02].”

Wie bereits erwähnt wurde, hat das E-Learning gerade zu Beginn des 21. Jahrhunderts einen starken Einfluss auf die Organisation der Lehre an Hochschulen ausgeübt. Dieses zeigt sich insbesondere durch die zahlreichen Förderprogramme des Bundesministeriums für Bildung und Forschung (BMBF), wie z. B. die Programme “Neue Medien in der Bildung” und “Notebook Universities”. Hierbei steht für die Hochschulen die Effizienz- und Effektivitätssteigerung der Lehre im Blickpunkt. Simon [Simon02] leitet den Begriff Effizienz am Verhältnis der *Input-Output*-Qualität der Lehre ab. Demnach ist eine Bildungseinrichtung effizienter, wenn sie bei gleichbleibender Qualität der Lehre mehr Studierende betreut. Dieses lässt sich zweifellos durch den Einsatz von E-Learning erreichen, wodurch die dezentrale Wissensvermittlung – d. h. die zeit- und ortsunabhängige Wissensaufnahme von Studierenden – unterstützt wird. Daniel [Daniel98] weist darauf hin, dass die Ursache für eine solche Effizienzsteigerung in der strikten Trennung zwischen dem Entwicklungsprozess von Materialien und deren Inhaltsvermittlung liegt. Ein Beispiel hierfür ist die Fernuniversität Hagen. Durch den digitalen Vertrieb von Lehr- und Lernmaterialien, anstatt der herkömmlichen Zuteilung über den Postweg [Bauch03], wird eine erhebliche Reduktion des logistischen und finanziellen Aufwands seitens der Fernuniversität erreicht. Eine solche Effizienzsteigerung ist auch das Ziel dieser Arbeit. In Kapitel 1.2 wurde das Forschungsprojekt mαth-kit vorgestellt, in dessen Rahmen diese Arbeit ausgewertet wird und verschiedene Formen der Wissensvermittlung, wie z. B. Präsenzlehre oder Fernlehre mit einfließen. Es sollen also Mechanismen untersucht und entwickelt werden, mit deren Unterstützung eine Effizienzsteigerung durch Reduktion des *Inputs* erzielt werden kann.

2.2 Lerntheorie

Bei der Entwicklung von E-Learning Inhalten ist es bereits in der Konzeptionsphase wichtig, festzulegen, mit welcher Methodik Lernprogramme das Wissen an Studierende weitergeben. Die Festlegung auf eine bestimmte Lernstrategie hängt zum einen von dem zu vermittelnden Stoff ab, den Lehrende auswählen und zum anderen von den Fertigkeiten der Studierenden. Um ein grundlegendes Verständnis für den Lernprozess zu entwickeln, und diesen bei der Konstruktion zu berücksichtigen, wird zunächst das von Dreyfus [Dreyfus86] eingeführte hierarchische Lernmodell vorgestellt, bei dem Lernende von Anfängern mit einfachen Faktenwissen zu Experten mit intuitiven Fähigkeiten fortschreiten. Hierbei wird sukzessiv auf jede Entwicklungsstufe eingegangen und dabei auf mögliche Gefahren mit dem Umgang des Wissens hingewiesen. Ausgehend vom Prozess, wie der Lernende sich entwickelt, sind auch die Mechanismen und Verfahren zum Wissens- und Fähigkeitserwerb von Interesse. Aus diesem Grund werden die drei wichtigsten Lernparadigmen *Behaviorismus*, *Kognitivismus* und *Konstruktivismus* vorgestellt. Aufbauend auf diesen Grundlagen erlaubt das von Baumgartner [Baumgartner99] eingeführte *Heuristische Lernmodell*, Weiterbildungsprozesse hinsichtlich der variablen Lernziele, Lehrinhalte und Lehrstrategie zu

betrachten.

2.2.1 Lernprozess

Im Folgenden werden die einzelnen Stufen des Lernprozesses nach Baumgartner [Baumgartner97, Baumgartner99, Baumgartner95] beschrieben.

Anfänger/-in

Der bzw. die Anfänger/-in ist im Umgang mit Fachwissen noch nicht vertraut. Er bzw. sie sammelt Schritt für Schritt Wissen an, indem objektive Fakten erlernt werden, die in einem allgemeinen Kontext gültig sind. Ausgehend von den gelernten Fakten und Regeln ist der Lernende in der Lage, Merkmale und Strukturen wieder zu erkennen. Dem Lernenden ist also die Existenz bestimmter Regeln und Fakten bewusst, ohne den Hintergrund tiefgründig zu verstehen. Er bzw. sie ist daher nicht in der Lage, die Regeln ohne nachzufragen bzw. einem Einwirken von außen anzuwenden, weil er bzw. sie noch nicht selbst entscheiden kann, welche die richtige Regel ist. Als Beispiel wird eine Person betrachtet, die das Schachspielen erlernt. Zu Beginn wird versucht, jeden Figurentausch auf ein Punktesystem zurückzuführen, indem jeder Figur eine bestimmte Anzahl von Punkten zugeordnet wird. Ein Tausch mit einer gegnerischen Figur lohnt sich nach dieser Regel nur, wenn ein Punktevorsprung erzielt wird. Ihm bzw. ihr ist zu diesem Zeitpunkt nicht klar, dass es Situationen gibt, in denen ein Tausch mit der gegnerischen Figur trotz Punkteverlust einen Vorteil liefert. Ein weiteres Beispiel ist der bzw. die Fahranfängerin, die zu Beginn lernt, bei einer bestimmten Geschwindigkeit das Getriebe zu schalten.

Fortgeschrittene/-r

Die erworbenen kontextfreien Regeln, d. h. Regeln und Merkmale die isoliert betrachtet werden, müssen durch Übungen in konkreten praktischen Situationen angewendet werden. Durch diese Form des erfahrenden Lernens erwirbt der bzw. die Anfängerin die Fähigkeit, erlernte Regeln auf situationsabhängige Ereignisse anzuwenden und Situationen richtig einzuschätzen. Der bzw. die Schachspieler/-in erkennt beispielsweise nach einer gewissen Zeit, dass bestimmte situationsabhängige Stellungen ungünstig für die Deckung strategischer Figuren sind, ohne dafür auf definierte Regeln zurückzugreifen. Ebenso verändert sich das Schaltverhalten einer Autofahrerin je nach Situation und Motorengeräusch. Die Gefahr, die Lernenden in den ersten zwei Stufen droht, ist eine zu starke Generalisierung der Fakten und Regeln. Der bzw. die Lernende versucht, die sehr allgemeinen Regeln auf spezielle Situationen anzuwenden ohne sie gegebenenfalls an das fokussierte Problem anzupassen.

Kompetenz

Diese Lernstufe ist erreicht, wenn eine Person die relevanten Fakten und Regeln kennt und sie in den meisten Fällen auch anwenden kann. Hierdurch ist die kompetente Person in

der Lage, eine Vielzahl von Problemen zu lösen und selbständig zu handeln. Es wird also das Erreichen des Ziels gegenüber dem Anwenden von Regeln in den Vordergrund gestellt. Der Schachspieler wird nach dem einfachen Anwenden der Regeln an den Punkt gelangen, den gegnerischen König anzugreifen mit dem Ziel das Spiel zu gewinnen. Ebenso wird sich der oder die Autofahrer/-in nach einer gewissen Zeit mehr auf das Erreichen des Ziels konzentrieren, als auf das genaue Einhalten der Verkehrsregeln. Bei dieser Stufe kann es jedoch leicht zur Selbstüberschätzung kommen.

Gewandtheit

Haben Lernende die vierte Lernstufe erreicht, besitzen sie die Fähigkeit gewandt zu handeln. Dies bedeutet, dass in bestimmten Situationen der Entscheidungsprozess für eine anzuwendende Regel nicht mehr evaluiert werden muss, sondern aufgrund der bereits gesammelten Erfahrungen getroffen wird. Durch eine auf Erfahrungen basierte Subsummierung von Merkmalen und durch Vernachlässigung nicht relevanter Eigenschaften, gestaltet sich die Wahl der anzuwendenden Regel intuitiv. Die Eigenschaft, intuitiv Muster auf Szenen anzuwenden und nur bestimmte Merkmale zu betrachten nennt Dreyfuss “holistisches Erkennen von Ähnlichkeiten”. Ein bzw. eine Autofahrer/-in, die mit hoher Geschwindigkeit bei nasser Fahrbahn auf eine Kurve zufährt, muss sich entscheiden, ob sie bremst, das Gas wegnimmt oder den Motor zum Bremsen einsetzt. Gewandte Autofahrer und Autofahrerinnen entscheiden sich aufgrund ihrer bereits gesammelten Erfahrungen, ohne die einzelnen Möglichkeiten durchzugehen. Der bzw. die gewandte Schachspieler/-in besitzt entsprechend die Fertigkeit sich je nach der Spielsituation die Strategie auszuwählen, für die lediglich eine kleine Menge möglicher Züge in Frage kommen.

Expertentum

Eine Person, die Expertise in einem Gebiet erworben hat, vermag intuitiv zu handeln. Sie muss nicht über anzuwendende Regeln nachdenken, sondern trifft in jeder Situation engagiert und verantwortungsbewusst eine Entscheidung. Das intuitive Handeln verschmilzt mit dem Körper zu einer Einheit. So ist z. B. das Gehen einer erwachsenen Person eine intuitive Handlung eines Experten. Die Gefahr, die sich für diese und die vorherige Lernstufe ergibt, ist das Beibehalten einer bestimmten Sichtweise, ohne unnatürliche Ereignisse einzubeziehen. Hieraus ergibt sich der so genannte “Tunnelblick”. Tabelle 2.2 zeigt die einzelnen Attribute jeder Lernstufe (Tabelle aus [Baumgartner99], Seite 85).

Stufe	Lernelemente	Perspektive	Entscheidung	Einstellung	Gefahr
Anfängertum	Fakten und kontextfreie Regeln	keine	keine, passive Rezeption	distanziert	Übergeneralisierung
Fortgeschritten	Anwendung von Fakten/kontextfreien Regeln in Situationen; sammeln Erfahrungen	keine	Nachahmung und Imitation	distanziert	Übergeneralisierung einzelner Erfahrungen bzw. gelernter Regeln
Kompetenz	Anwendung von Fakten und kontextfreien Regeln; Einbeziehung eigener Erfahrungen	bewusst gewählt	analytisch	distanziertes Verstehen u. Entscheiden; an Ergebnissen gefühlsmäßig beteiligt	Überschätzung eigener Fähigkeiten, erhöhte Unfallgefahr
Gwandtheit	Gestaltwahrnehmung, holistisches Erkennen von Ähnlichkeiten	implizit durch Erfahrung vorhanden	analytisch	teilnehmendes Verstehen; distanziertes Entscheiden	Tunnelperspektive
Expertentum	Gestaltwahrnehmung, holistisches Erkennen von Ähnlichkeiten	implizit durch Erfahrungen vorhanden, im Körper integriert	intuitiv	gefühlsmäßig beteiligt, persönliche Verantwortung	Tunnelperspektive

Tabelle 2.2: Fertigkeitsstufen

2.2.2 Lernparadigmen

Nachdem die Fertigkeitsstufen des Lernens beschrieben wurden, soll nun der Frage nachgegangen werden, wie der Lernende von einer Lernstufe zur nächsten gelangt, bzw. wie der menschliche Lernprozess funktioniert. Hierzu werden im Folgenden die wichtigsten Lernparadigmen betrachtet. Eine Zusammenstellung ihrer Thesen findet sich in Werken von Baumgartner, Schulmeister, Thissen und Kerres [Baumgartner97, Baumgartner99, Kerres98, Schulmeister81, Thissen97, Thissen99a, Thissen99b].

Behaviorismus

Der Behaviorismus beschreibt die Theorie des Lernens durch Verstärkung. Der Begründer dieser Theorie war Iwan Petrowitsch Pawlow, der der Überzeugung war, dass Verhalten auf Reaktion beruht und nannte dieses Prinzip *Konditionierung*. Er bemerkte eher zufällig, dass seine Versuchshunde bereits bei dem Klingelsignal zur Fütterung Speichel absonderten ohne das Futter zu sehen. Aus diesem Versuch schloss er, dass die Hunde einen Zusammenhang zwischen dem Signal und der Fütterung erlernt hatten. Johan B. Watson hat diese Reiz-Reaktions-Steuerung auf den Menschen überführt und vertrat die Annahme, dass jedes Verhalten konditionierbar ist. Skinner erweiterte diese Theorie um das *operante Konditionieren*, das besagt, dass das Verhalten von Menschen durch Belohnung und Bestrafung beeinflusst werden kann (vgl. [Skinner78], Seite 9-11). Dabei "steht das Verhalten in Verbindung mit den Ereignissen, die ihm nachfolgen. Verhalten hat bestimmte Konsequenzen und diese entscheiden über das zukünftige Auftreten ([Edelmann96], Seite 110)."

Nach Baumgartner wird der innerhalb einer Person stattfindende Lernprozess von außen konditioniert. Lehrende kennen zu jedem Zeitpunkt den zu vermittelnden Stoff und entscheiden, was Lernende zu erlernen haben. Das Gehirn wird als Blackbox betrachtet, deren interne Vorgänge nicht interessieren. In dieser Lerntheorie wird davon ausgegangen, dass das menschliche Gehirn nur auf eine geeignete Art und Weise gereizt werden muss, um eine gewünschte Reaktion auszulösen. (vgl. Abbildung 2.2). Durch entsprechendes Feedback können Lehrende auf den Lernprozess eingehen und bestimmte Fertigkeiten verstärken oder verringern. Das Feedback kann sowohl positiv als auch negativ sein. Durch eine negative Rückmeldung oder durch Ignorieren des Verhaltens kann Verhalten langfristig vom Lernenden reduziert werde, wo hingegen eine positive Bewertung des Verhaltens - diese sollte sich jedoch im Rahmen bewegen - eine Verstärkung hervorruft. Ein großer Nachteil des Behaviorismus ist, dass der Verhalten-Feedback-Prozess nicht die geistigen Fähigkeiten des lernenden Menschen erfassen kann. Nur für solche Fertigkeiten, die sich auf körperliche Eigenschaften beschränken, lassen sich Fortschritte erzielen. Ein Beispiel für den sinnvollen Einsatz dieses Paradigmas sind Übungsaufgaben für Maschinenschreiber oder Klavierspieler. Im E-Learning-Kontext basierten die ersten computergestützen Lernprogramme (CBT) (vgl. Abschnitt 2.1.2) auf diesem Paradigma. Hierbei handelte es sich um einfache Wissensfragen mit anschließender Erfolgskontrolle.

Kognitivismus

Der Kognitivismus (Lernen durch Einsicht) stellt im Gegensatz zum Behaviorismus interne Verarbeitungsprozesse des Lernenden in den Vordergrund. Es werden die internen - also im Gehirn - ablaufenden komplexen Prozesse untersucht, jedoch nicht das Verhalten der Lernenden (vgl. Abbildung 2.2). Folglich stehen nicht die Stimuli als Reaktion auf bestimmte Verhaltensmuster im Vordergrund, sondern das Lernen von Methoden und Verfahren zur Problemlösung, um zu einem späteren Zeitpunkt zu den richtigen Antworten zu gelangen. Hierbei werden eingehende Informationen *(bottom-up)* mit bereits vorhandenen Erfahrungen *(top-down)* einer Person gepaart. Neue Informationen werden also immer auf Basis von vorhandenen Wissen interpretiert und verstanden. Als Beispiel für den Kognitivismus im Bereich von Softwaresystemen kann ein Programm betrachtet werden, welches dem Anfänger bzw. der Anfängerin, mit Hilfe einer tutoriellen Unterstützung, den Lehrstoff anhand von Beispielen veranschaulicht. Durch den Einsatz von Animationen und Begriffserklärungen kann das Wissen dem Lernenden illustrativ demonstriert werden. Treten Fragen zu der Thematik oder zu den gestellten Aufgaben auf, kann ein 'virtueller' Tutor konsultiert werden.

Konstruktivismus

Die wesentliche Eigenschaft des Konstruktivismus ist es, das Lernen als aktiven Prozess anzusehen. Fragestellungen sollen vom Lernenden durch Konstruktion verstanden werden. Hierbei stehen im Gegensatz zum Kognitivismus nicht die Problemlösungen im Vordergrund, sondern vielmehr das mit dem Konstruktionsprozess einhergehende Erzeugen neuer Probleme und Fragestellungen. Beim Konstruktivismus stellt sich der Lernprozess durch das Einbringen eigener Erfahrungen, Explorationsergebnisse, Fehlentscheidungen sowie durch das Entdecken von Zusammenhängen ein. Abbildung 2.2 zeigt schematisch die drei Lernparadigmen mit den auf den menschlichen Organismus einwirkenden externen Reizen. Der Konstruktivismus ist hierbei ein weitestgehend geschlossenes System, welches lediglich multimodal energetische Stimuli von 'außen' aufnimmt. Das bedeutet, dass es keinen informationellen Input und Ouput mit der Umwelt gibt. Der Organismus erzeugt selbständig Informationen, die im Lernprozess verarbeitet werden. Lernsysteme, die auf dem konstruktivistischem Paradigma basieren, besitzen eine explorative Umgebung, in der Lernende, durch praktisches Konstruieren, Erkenntnisse und Erfahrungen sammeln können.

2.2.3 Ein Heuristisches Lernmodell

Das Heuristische Lernmodell dient der lerntheoretischen Kategorisierung von Lehrinhalten, Lernszenarien und Lernformen. Es wird ein Würfel zur räumlichen Darstellung verwendet, um monokausale und hierarchische Zusammenhänge zu überwinden. Das Würfelmodell ermöglicht, je nach Fragestellung und Blickwinkel auf den Würfel, die Ansicht einer der drei Kategorien *Handlungsebene*, *Lehr-/ Lernebene* und *Ebene der sozialen Organisation*. Bedingt durch die dreidimensionale Darstellung wirken die anderen Kategorien auf die

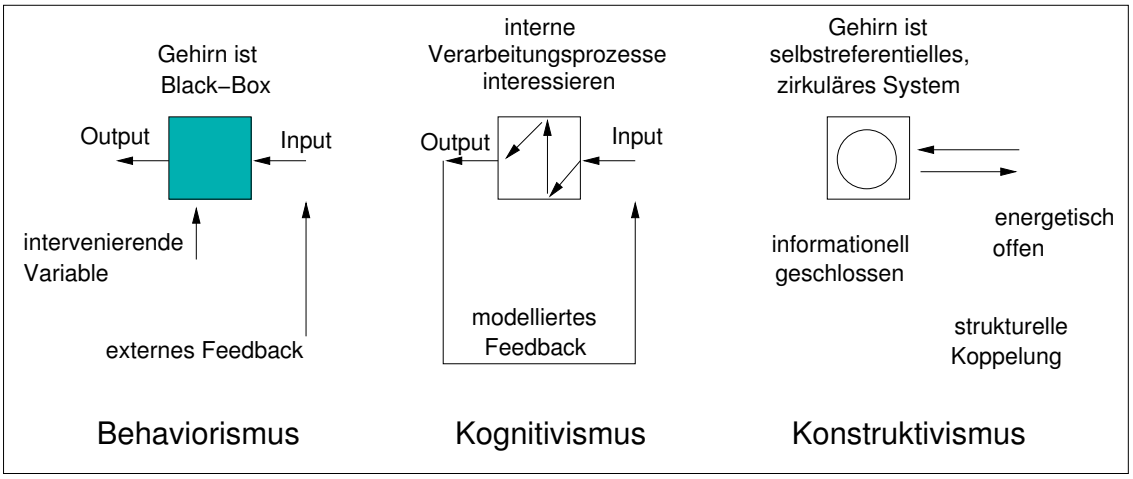


Abbildung 2.2: Lernparadigmen (Quelle: [Baumgartner97], Seite 5)

fokussierte ein. Außerdem unterstützt das Modell die Konkretisierung von Fragestellungen, die bei der Gestaltung von Aus- und Weiterbildungen auftreten. Hierbei geht es z. B. um folgende Frage: Welche Fertigkeitsstufe soll auf welcher Stufe der Handlungsfähigkeit mit welcher Lernform erworben werden? Abbildung 2.3 zeigt das von Baumgartner [Baumgartner97, Baumgartner99] beschriebene heuristische Lernmodell.

2.3 Standardisierung im E-Learning

Bereits in der Vergangenheit hat sich gezeigt, dass Standardisierungsinitiativen durch Organisationen und Gremien, wie z. B. die *Deutsche Industrie Norm (DIN)*, das *Institute of Electrical and Electronics Engineers (IEEE)* sowie die *International Organization for Standardization (ISO)*, Einzug in sämtliche Anwendungsdomänen gehalten haben. Eine Standardisierung ermöglicht es Anbietern eigene Produkte mit denen anderer Anbieter zu kombinieren. Hierdurch sind Hersteller in der Lage, sich auf die Produktion ausgewählter Teile zu spezialisieren, so dass fehlende Teile von Drittanbietern bezogen werden können. Zudem unterstützen Standards die Entwicklung von Produkten dahingehend, dass sie das Risiko und die Kosten, die während der Entwicklungszeit anfallen, erheblich reduzieren. In [Wilson99] wird hierzu ein *life-cycle* Modell illustriert. Äußerst nützliche Standardisierungen sind z. B. elektrische Stecker und das OSI-Referenzmodell für Netzwerkprotokolle. Sogar Kinder profitieren von Standardisierungen, wie es z. B. bei Lego Bausteinen der Fall ist. Diese lassen sich von Kindern aller Altersstufen zusammensetzen und auseinander bauen, ungeachtet von ihrer Farbe und ihrer Form. In der E-Learning-Domäne werden neben zahlreichen Online-Lernangeboten eine Vielzahl an Learning Management Systemen (LMS)[1] entwickelt, wodurch eine Standardisierung unumgänglich ist. Entwickelte Lerneinheiten sollen unabhängig von Lernplattformen konzipiert und vertrieben werden.

[1]Learning Management Systeme (LMS) bzw. Lernplattformen werden detailliert in Kapitel 3.3 vorgestellt. Eine Begriffsdefinition befindet sich im Glossar (vgl. Anhang B)

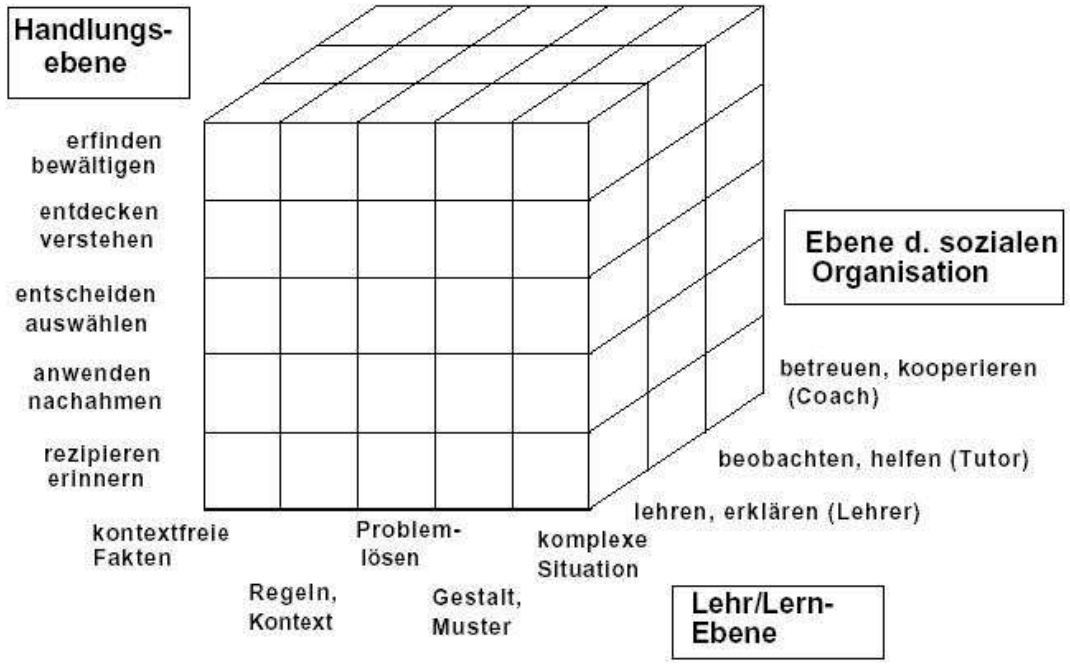


Abbildung 2.3: Ein Heuristisches Lehr- und Lernmodell (Quelle: [Baumgartner97], Seite 8)

Damit eine nahtlose Integration solcher Materialien funktioniert, sind Organisationen und Gremien für Standardisierung mit folgenden Fragestellungen beschäftigt: “Wie lassen sich Lehrinhalte einfach und sinnvoll kombinieren?”, “Wie können Lernmaterialien, basierend auf unterschiedlichen Formaten, zusammenwirken?” und “Wie ist sicherzustellen, dass entwickelte Lehrinhalte mit Lernplattformen anderer Anbieter kompatibel sind?”. Darüber hinaus werden auch technische Fragen diskutiert, wie z. B. die Definition von Metadaten oder Lernobjekte[2]. Im folgenden werden die wichtigsten und einflussreichsten Organisationen vorgestellt und deren Abhängigkeiten dargelegt:

- **ADL** Advanced Distributed Learning Initiative (vgl. Abschnitt 2.3.1)
- **IMS** Instructional Management Systems Project (vgl. Sbschnitt 2.3.2)
- **IEEE LTSC** IEEE Learning Technology Standard Committee (vgl. Sbschnitt 2.3.3)
- **ARIADNE** Aliance of Remote Instructional Authoring and Distibution Networks for Europe (vgl. Abschnitt 2.3.5)

[2]Die Begriffe Metadaten und Lernobjekte werden erst in den Abschnitten 2.3.3 und 2.4 eingeführt. Eine kurze Begrifssdefinition ist im Glossar (vgl. Anhang B) zu finden.

- **AICC** Aviation Industry Computer Based Training Committee (vgl. Abschnitt 2.3.4)

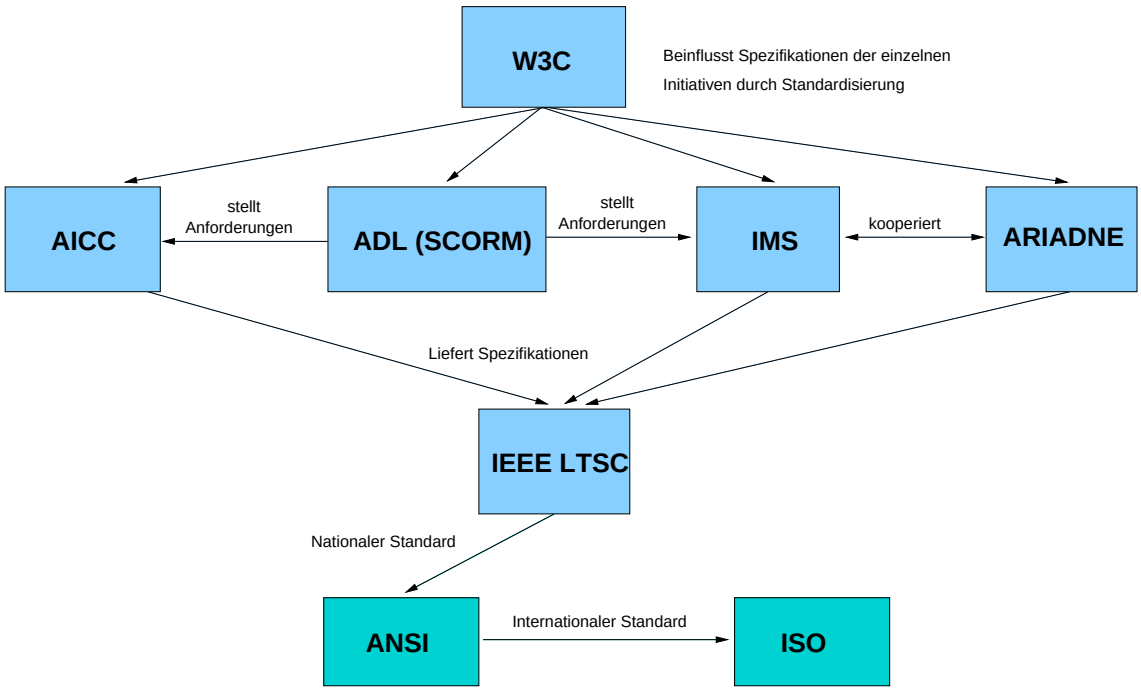


Abbildung 2.4: Kooperation internationaler Standardisierungsgremien und -Organisationen

Die vier Konsortien AICC, ADL, IMS und ARIADNE haben zu Beginn ihrer Arbeit damit angefangen, Standardisierungsvorschläge für die Domäne E-Learning zu erarbeiten. Es hat sich jedoch herausgestellt, dass Arbeitsergebnisse ausgetauscht und konsolidiert werden können. Diese Bemühungen sind unter anderen dadurch entstanden, weil nur das IEEE das Recht besitzt, Spezifikationen für die Etablierung eines Standards bei den relevanten Organisationen wie z. B. dem ANSI (Amercan National Standards Institute) einzureichen [Baumgartner02]. Abbildung 2.4 veranschaulicht diese Zusammenhänge.

2.3.1 ADL (SCORM)

Die *Advanced Distributed Learning (ADL)* Initiative wurde 1997 vom amerikanischen Verteidigungsministerium *Department of Defense (DoD)* gegründet, um Standardisierungen im Bereich E-Learning zu entwickeln, die potenzielle Kooperationen zwischen den Vorhaben der Wirtschaft, Wissenschaft und Regierung ermöglichen und erleichtern. Hierbei wurden vor allem die Themen Wiederverwendbarkeit, Langlebigkeit und Zugänglichkeit fokussiert. Als Resultat hat die ADL Initiative im Frühjahr 1999 damit begonnen, Spezifikationen und Standardisierungen anderer Organisationen im E-Learning-Sektor aufzugreifen, und zu einem generellen Referenzmodell, dem *Sharable Content Object Reference Model (SCORM)* [ADL04] zusammenzuführen. Die erste öffentlich zugängliche Version

0.7.3 des SCORM-Referenzmodells wurde Mitte 1999 verabschiedet und bis zur Version 2004 weiterentwickelt. Das SCORM-Referenzmodell fast im Wesentlichen zwei Standards für Lernobjekte zusammen: das *Content Aggregation Model* und das *Run-Time Environment*. Das Content Aggregation Model spezifiziert die Struktur von Lernobjekten, damit sie aus Sicht der Anbieter von Lehrinhalten integrierbar, wiederverwendbar und zugreifbar sind. Dagegen ermöglicht das Run-Time Environment die Auswertung von Lernobjekten und die Kommunikation mit Lernplattformen (vgl. Kapitel 3.3).

Content Aggregation Model

Das Content Aggregation Model definiert durch Mechanismen zur Entwicklung von Elementarteilen einen Rahmen für die Erstellung und den Vertrieb von Lehrinhalten. Das Elementarteil – genannt Asset – ist ein Sammelbegriff für sämtliche Dateien, die zu einer Lerneinheit gehören, wie z. B. Java Applets, Texte, Bilder oder auch Videos. Sie werden aggregiert und zu höheren, komplexeren Lerneinheiten zusammengefasst. Hierbei definiert der Standard drei wesentliche Aspekte: das *Content Model*, die *Metadaten* und das *Content Packaging*. Das Content Model ist eine Nomenklatur zur Definition der Komponenten, durch die eine Lerneinheit aufgebaut ist. Dazu gehören die bereits erwähnten *Assets*, die *Sharable Content Objects (SCOs)* sowie deren *Aggregation*. SCOs sind eine sinnvolle und inhaltlich abgeschlossene Komposition von Assets, die als eigenständige Lernmodule in ein Learning Management System (LMS) (vgl. Abschnitt 3.3) geladen werden und mit diesem kommunizieren können. Hierbei ist die Festlegung ihrer Granularität äußerst wichtig. SCOs müssen kontextuell abgeschlossen sein, so dass sie nicht auf andere Lernmodule verweisen oder gar von ihnen abhängig sind. Darüber hinaus enthält ein SCO – wenn es die Kommunikation zu einem LMS aufbauen soll – ein spezielles Asset, welches die Kommunikations-API kapselt. Die Kommunikations-API wird durch JAVA-Skript-Befehle aufgerufen, die direkt in der Einstiegsseite des SCOs verankert sind, und baut so die Verbindung zum LMS (vgl. Abschnitt 3.3) auf. Im Minimalfall wird die Auswertungsinstanz des LMS mit dem Aufruf `LMSInitialise` initialisiert und mit `LMSFinish` abgeschlossen. Sind SCOs und Assets definiert, können sie auf unterschiedliche Weise organisiert werden, so dass der Kursinhalt Lernenden mit unterschiedlichen Lernpfaden präsentiert werden kann. Dieser Vorgang wird als *Content Aggregation* bezeichnet. Ein Beispiel für unterschiedlich organisierte Inhalte ist die Bereitstellung verschiedener komplexer Lehrinhalte für entsprechend geeignete Fertigkeitsstufen (vgl. Abschnitt 2.2.1). Ein Kurs für Anfänger bzw. Anfängerinnen wird beispielsweise Grundlagen und Regeln enthalten, die für einen kompetenten Lernenden uninteressant sind und deshalb nicht in der Organisationsstruktur des Kurses vorkommen. Abbildung 2.5[3] zeigt schematisch die Struktur bzw. die Organisation von Assets und SCOs. Die Knotenpunkte der Organisationen werden *Items* genannt und verweisen auf genau eine Ressource. Ressourcen wiederum sind eine Liste eines oder mehrerer Assets bzw. SCOs. Sie gruppieren semantisch abhängige Dateien.

Der im SCORM-Standard verwendete Metadaten-Ansatz wurde aus der *IEEE LTSC*

[3] Quelle: ADL, Sharable Content Object Referenz Model (SCORM) Version 2004, http://www.adlnet.org (29.10.2005) [ADL04]

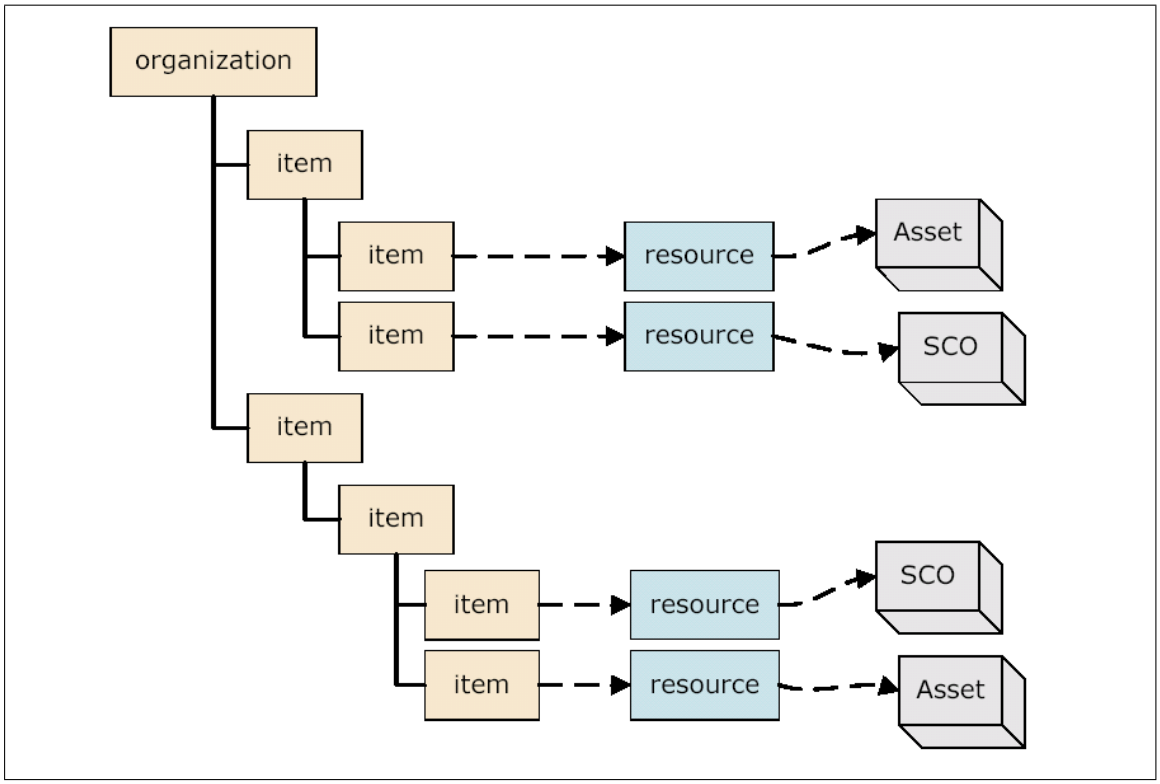


Abbildung 2.5: Beispiel: Organisation

Learning Object Metadata (LOM) Spezifikation übernommen und im Content Aggregation Model (CAM) eingebunden. Metadaten können unterschiedlichen Komponenten des Modells zugeordnet werden. Sie gehen mit dem Content Aggregation Model einher und spezifizieren Daten zur Klassifizierung von Kursstrukturen. Darüber hinaus können sie auch SCOs zugeordnet werden. Sie sind dann wichtig, wenn Lernende Informationen zu einer Lerneinheit erhalten möchten, wie z. B. den Schwierigkeitsgrad oder das Lernziel. In manchen Fällen scheint es sogar sinnvoll zu sein, Assets, wie z. B. Video-Dateien, mit Metadaten auszustatten. Der Standard sieht für jenen Fall die direkte Behaftung von Metadaten an Assets vor. An dieser Stelle ist anzumerken, dass es ebenso Standardisierungen für Assets gibt, wie sie z. B. das Videoformat MPEG-7 anbietet, die beschreiben, in welcher Weise auf Metadaten innerhalb digitaler Formate zugegriffen wird [Klamma03]. Allgemein formuliert helfen Metadaten Anwendern und Anwenderinnen bei der Suche von Kursen bzw. deren Fragmenten in so genannten *Repositories* (vgl. Abschnitt 2.3.2) mit der Absicht, Lehrinhalte durchzuarbeiten oder sie in höhere selbstdefinierte Strukturen gezielt einzuordnen.

Abschließend wird betrachtet, wie ein Content Package physikalisch organisiert ist. Hierzu führt der SCORM-Standard das *Package Interchange File (PIF)* ein. Es besteht physikalisch aus einer Manifest-Datei und beliebig vielen kursrelevanten Assets. Die Manifest-Datei, welche in XML (vgl. Abschnitt 3.2.1) definiert wird, enthält die bereits oben beschriebenen Komponenten Meta-Daten, Organisationen, Ressourcen und Manife-

ste[4]. Die folgende Abbildung 2.6[5] zeigt grafisch den Aufbau eines Content Packages.

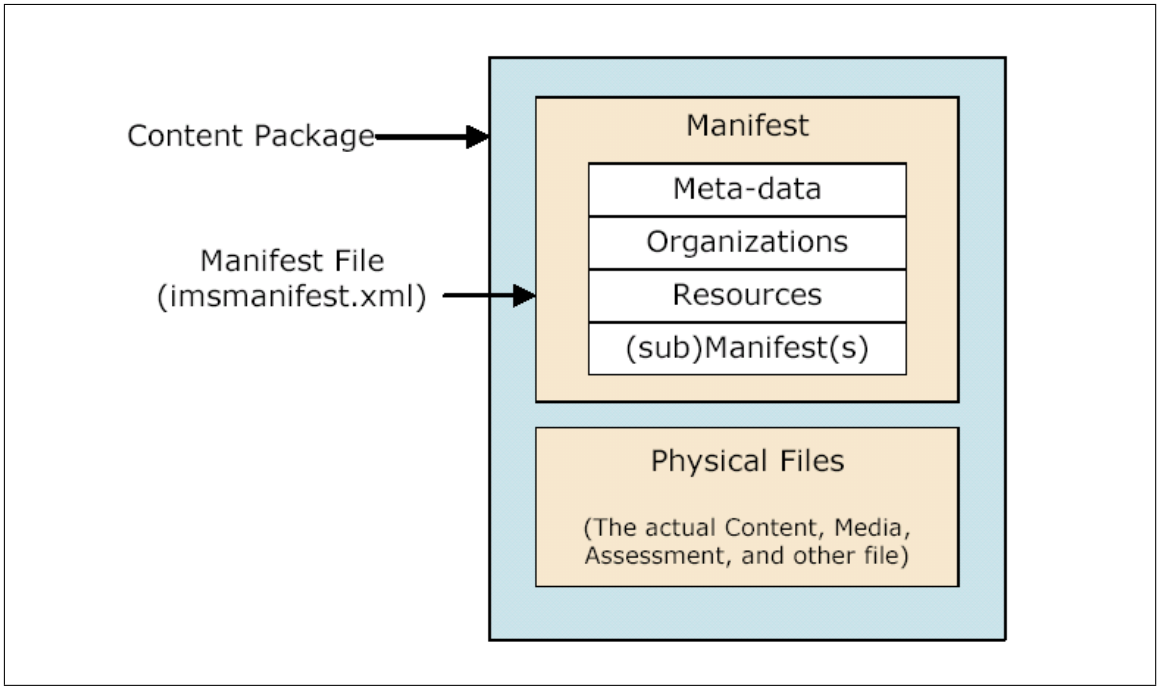


Abbildung 2.6: Beispiel: Content Package

Das Konzept der PIF-Datei wurde einerseits zum Datenaustausch zwischen verschiedenen standardkompatiblen LMS entwickelt und andererseits, um direkt auf Kurse zuzugreifen, die beispielsweise auf einer CD-ROM abgelegt sind. Für den Fall der Kursinstallation in einem LMS bietet sich der Einsatz komprimierter Archive geradezu an. Alle Kursdateien, einschließlich der Assets und Manifest-Datei, werden zu einer komprimierten PIF-Datei zusammengefasst, die sich dadurch wesentlich einfacher handhaben lässt. Der Vorteil liegt auf der Hand: Sollen Kurse, bestehend aus einer großen Anzahl von Bild-, Audio- oder Textdateien, in ein LMS geladen werden, muss jede Datei sukzessiv in das System eingestellt werden. Genau diese Unterteilung wird jedoch benötigt, damit die Kursinhalte durch einen Browser im Dateisystem geöffnet werden können. Des weiteren hat das *Canadian Department of National of Defence* eine Erweiterung des SCORM-Standards eingebracht, die sich mit der Darstellung der Kursstruktur befasst [National Defence02].

Runtime Environment

Das Konzept der im SCORM-Standard 1.2 vorgestellten Laufzeitumgebung (Runtime Environment) ermöglicht grundsätzlich die Kommunikation zwischen Lernumgebungen und den dort eingeladenen Lerneinheiten. Die ausschließlich statischen Inhalte (SCOs und As-

[4] An dieser Stelle ist anzumerken, dass Manifeste innerhalb anderer Manifeste rekursiv verschachtelt werden können.

[5] Quelle: ADL, Sharable Content Object Referenz Model (SCORM) Version 2004, www.adlnet.org (29.10.2005) [ADL04]

sets) werden in Kombination mit einem standardkonformen LMS eingelesen und gestartet *(Launch)*. So kann das LMS durch die stattfindende Dynamisierung Benutzerinformationen bzw. Lernfortschritte statistisch protokollieren und auswerten. Kernstück der Kommunikation ist ein in das Lernmodul eingebetteter *API-Adapter*, der als Bindeglied zwischen LMS Server und SCO fungiert. Über ihn werden bidirektional Daten, z. B. der Initialisierungsaufruf oder Fehlermeldungen, vom SCO zum LMS geschickt. Für den Datenaustausch existiert das sogenannte *Data Model*. Mit diesem Modell werden laufzeitbezogene Daten, wie z. B. die Bearbeitungszeit oder der aktuelle bzw. maximal zu erreichende Punktestand, übertragen. Abbildung 2.7[6] zeigt grafisch den Zusammenhang zwischen LMS-Server, API-Adapter und SCO.

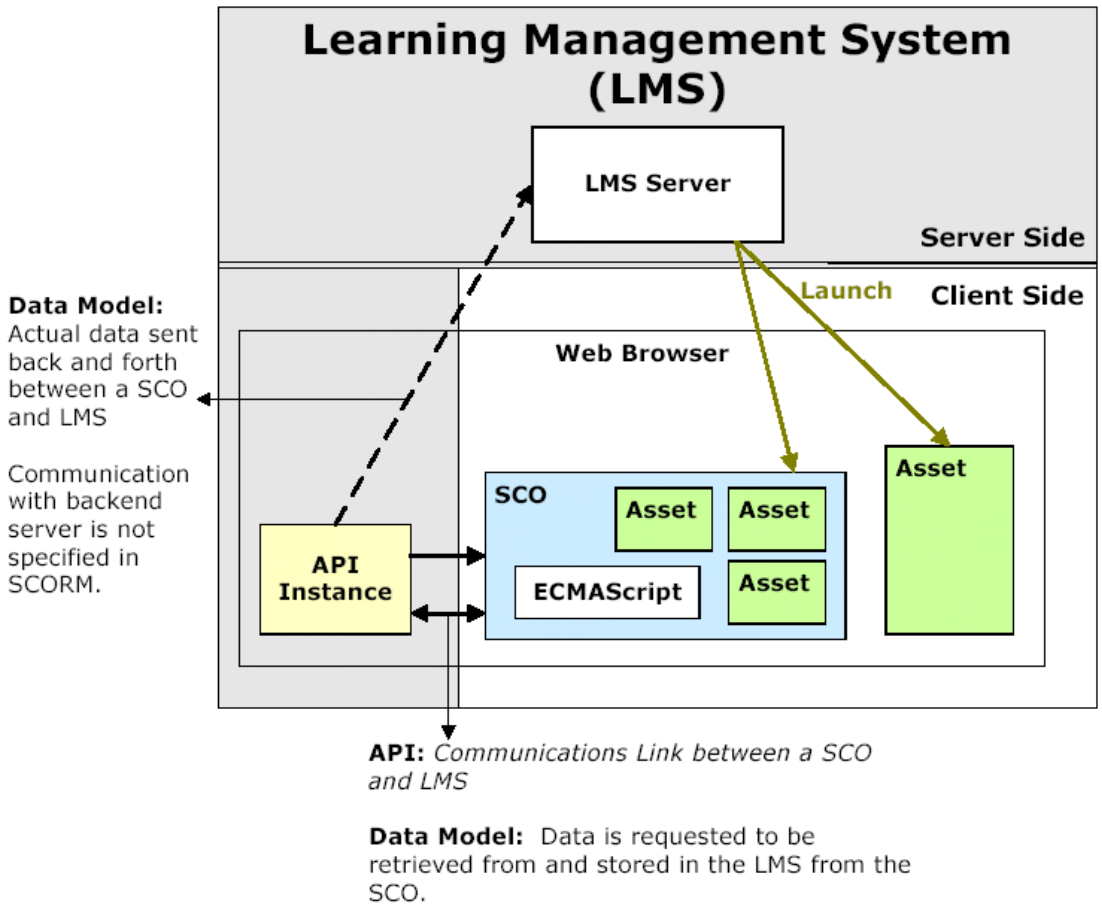


Abbildung 2.7: SCORM Laufzeitumgebung

[6]Quelle: ADL, Sharable Content Object Referenz Model (SCORM) Version 2004, www.adlnet.org (29.10.2005) [ADL04]

2.3.2 IMS

Das *Instructinal Management Systems Project (IMS)* [IMS05b] ist aus der National Learning Infrastructure Initiative der EDUCAUSE entstanden und befasst sich seit 1997 mit unterschiedlichen Fragestellungen aus dem Bereich verteilter Lerntechnologien. Hierzu zählen nicht nur Systeme oder Inhalte, die an institutionellen Einrichtungen, wie z.B Schulen oder Hochschulen, eingesetzt werden, sondern auch solche, die in Heimarbeit am privaten Computer einsetzbar sind. Das IMS hat sich mit dieser Frage eingehend beschäftigt und eine Reihe wichtiger de facto Standards durchgesetzt, die den Umgang mit statischen und dynamischen Inhalten sowie asynchronen und synchronen Lernszenarien spezifizieren. Hierzu gehört die IMS Metadaten Specification [IMS03d], die im Übrigen auch in den LOM Standard (vgl. Abschnitt 2.3.3) aufgenommen wurde und Lernenden das Auffinden von Materialien ermöglicht, gleichgültig, ob sie auf einer DVD gespeichert wurden oder im Internet zur Verfügung stehen. Zudem wird ein Austauschformat, basierend auf der IMS Content Package Specification [IMS03a] entwickelt, das den Austausch von Lernmaterialien zwischen standardkompatiblen Lernplattformen und Autorensystemen ermöglicht. Das IMS Content Package wird in seinen Grundzügen in Abschnitt 2.3.1 beschrieben, und soll an dieser Stelle nicht weiter vertieft werden.

Zudem hat das IMS ein Datenformat spezifiziert (IMS Question and Test Interoperability Specification [IMS05a]), mit dem es möglich ist, Quiz-Elemente zwischen Anwendungen auszutauschen. Sie wird hauptsächlich zur Definition von Test- und Quizumgebungen verwendet, die dann von einem LMS ausgewertet und ausgeführt werden können. Autorensysteme oder Quizeditoren können mit Hilfe dieses Standards die intern verwendete Datenstruktur auf eine XML-Instanz abbilden. Die IMS-QTI Spezifikation wurde im März 2000 verabschiedet und spezifiziert die drei Grundelemente *Item*, *Section* und *Assessment*.

- Ein **Item** ist eine elementar austauschbare Frage, die zusätzlich Instruktionen, wie Darstellungen, Antworten, Hinweise und Metadaten, mit einschließt.
- Das **Assessment** Element beschreibt eine Sequenz von Item-Elementen, die, je nach dem Wissen des Kandidaten, angepasst werden kann. Darüber hinaus existieren Instruktionen zur Auswertung der vom Lernenden bearbeiteten Fragen mit dem Resultat einer globalen Punktewertung.
- Das **Section**-Element ermöglicht die hierarchische Anordnung von Fragen. Dieses ist z. B. gewünscht, wenn Fragestellungen an die Struktur der Themenbeschreibung angeglichen werden sollen. Zudem lassen sich mit Hierarchien unterschiedliche Fragesequenzen erzeugen.

Zusätzlich zur IMS Content Package Spezifikation und den IMS Metadaten wurde die IMS Digital Repositories Interoperability Spezifikation (DRI) zur Schnittstellenbeschreibung digitaler Repositories [IMS03b] verabschiedet. Diese Spezifikation verwendet mehrere Spezifikationen, unter anderen das IMS Content Packaging und die IMS Metdaten, mit dem Ziel, eine einheitliche Schnittstelle für die Verwaltung von Ressourcen zu definieren. Hierbei können beliebige Ressourcen unter Berücksichtigung festgelegter Regeln in das

Repository eingepflegt und wiedergefunden werden. Für den Zugriff auf ein Repository spezifiziert das IMS die folgenden Kernfunktionen:

- Search/Expose
- Gather/Expose
- Submit/Store
- Request/Deliver
- (Alert/Expose)

Abbildung 2.8 veranschaulicht den Zusammenhang zwischen den Rollen, Kernfunktionen und Diensten.

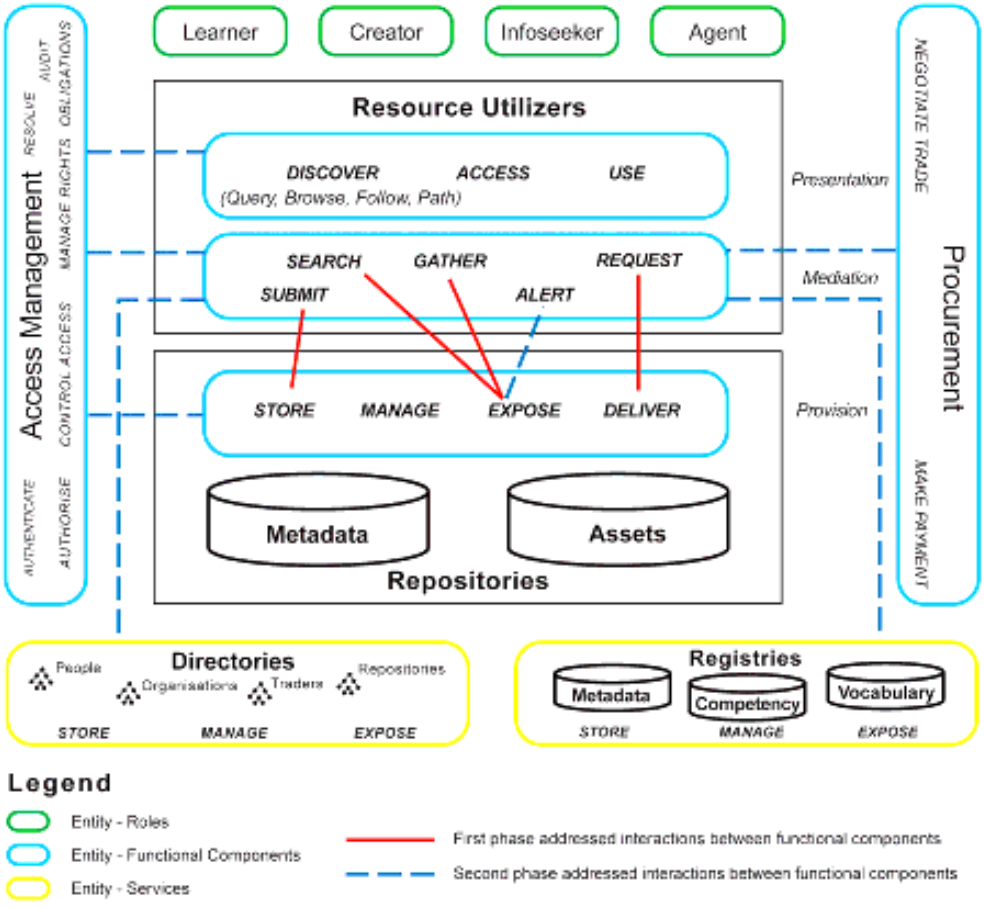


Abbildung 2.8: IMS Digital Repository

Durch die festgelegten Kernfunktionen ist es für unterschiedliche Anwendungen, wie z. B. Learning Content Management Systeme (LCMS)(vgl. Abschnitt 3.3), Learning Management Systeme (LMS) oder Portale, möglich, auf Repositories zuzugreifen. Zusätzlich wird von der einzusetzenden Repsoitory-Implementierung und somit von dem technischen Konzept abstrahiert, welches sich hinter ihnen verbirgt. Beispielsweise kann so der herkömmliche Repoitory Standard *Z39.50* mit XML-Respoitories vernetzt werden. Für den Zugriff auf XML-Datenbanken wird für die Suchoperation *Search* vom IMS eine XQuery [W3C05c] Empfehlung angegeben, mit dem Metadaten durchsucht werden können. Des weiteren spezifiziert die DRI den Zugriff auf mehrere Repositories, indem Kernfunktionen zum Zusammensammeln (Gather) verteilter Metadaten und deren Auswertung angeboten werden. Das Einstellen neuer Daten in ein Repository findet bei herkömmlichen, nicht DRI-Repositories, per FTP- oder HTTP-Zugang statt. Werden jedoch DRI-Repositries eingesetzt, so gibt der Standard ein *XML-Binding*[7] vor, das beliebigen *Clients* den Zugriff auf SOAP-Webservices [Chappell03, Snell02] ermöglicht, welche als Zugangspunkte für die DRI-Repositories dienen.

2.3.3 IEEE-LTSC (LOM)

Der *Learning Object Metadata (LOM)* Standard wurde seit 1997 durch das IEEE-Learning Technology Standards Committee (LTSC) entwickelt, und im Juni 2002 in seiner ersten Version verabschiedet. Den Grundstein der von der Metadata Working Group verfassten Spezifikation haben parallel laufende Projekte der ARIADNE [Duval02] und des IMS (vgl. Kapitel 2.3.2) gelegt. Der LOM Standard beschreibt generell die Semantik und Struktur von Daten, die Informationen über Lernobjekte (Learning Objects) (vgl. Kapitel 2.4) enthalten. Gerade aus dieser unidirektionalen Abbildung von Daten auf andere Daten wurde der Begriff Metadaten abgeleitet. Durch die Verwendung von Metadaten werden folgende Vorteile erzielt [Saddik01b, Saddik02]:

- Zusammenfassung des Inhalts und der Bedeutung von Lernobjekten,
- Suchen nach speziellen Lernobjekten wird ermöglicht,
- Lernenden die Auswahl von Lernobjekten, die für den eigenen Lernfortschritt relevant sind (Sequencing), zu ermöglichen,
- Der Zugriff auf Lernobjekte ist kontrolliert,
- Vermittlung von Informationen über Daten, wie z. B. das Format.

Der LOM-Standard vereinfacht die Beschaffung, das Suchen und die Auswertung von Lernressourcen durch Dritte, d. h. Lernende oder Lehrende. Darüber hinaus ermöglicht der auf XML (vgl. Kapitel 3.2.1) basierende Standard eine Auswertung bzw. Bearbeitung durch ein technisches System. Diese Anforderungen treten vor allem bei Sammlungen von Lernmodulen in Repositories oder Katalogen auf, bei denen Lernmodule anhand

[7] Mit dem Begriff *Binding* wird hier die technische Umsetzung durch eine bestimmte Technologie, wie in diesem Fall XML bezeichnet.

verschiedener Kategorien eingeordnet werden. Die LOM-Spezifikation sieht hierfür neun Kategorien vor, die in Tabelle 2.3 [Electrical02] aufgeführt sind.

Kategorie	Erklärung
General	Besitzt generelle Informationen über das Lernobjekt.
Lifecycle	Dokumentiert den historischen Verlauf sowie den aktuellen Status des Lernobjekts. Darüber hinaus werden Lernobjekte über den Zeitraum der Lebensdauer erfasst.
Meta-Metadata	Verwaltet Informationen über die Metadaten-Instanz.
Technical	Beschreibt technische Anforderungen und technische Eigenschaften des Lernobjekts.
Educational	Erfasst die pädagogischen und die für die Lehre relevanten Informationen über das Lernobjekt.
Rights	Verwaltet die für das Lernobjekt geltenden Nutzungsbestimmungen sowie die Urheberrechte.
Relation	Spezifiziert die Beziehung mit anderen Lernobjekten und deren Zusammenhang.
Annotation	Legt Anmerkungen über das Lernobjekt ab. Es wird das Lernszenario beschrieben und das Erstellungsdatum der Anmerkung.
Classification	Einordnung des Lernobjekts in ein Klassifizierungssystem.

Tabelle 2.3: Neun Kategorien für Lernobjekte nach der Learning Object Metadata Spezifikation (LOM)

Als Beispiel für die Umsetzung des Standards stellt Shen [Shen02] das webbasierte *Learning Resource Metadata Management System (LRMMS)* vor, welches als System bestehend aus untereinander kooperierenden Servern ein dezentrales Rahmenwerk präsentiert, mit dem Lernende Inhalte publizieren, verwalten und recherchieren können. Ebenso wurde im ARIADNE Projekt (vgl. Abschnitt 2.3.5) das *Knowledge Pool System (KPS)* entwickelt, das nicht nur den LOM-Standard umsetzt, sondern ihn zusätzlich um lernpädagogische Eigenschaften erweitert.

2.3.4 AICC

Das *Aviation Industry Computer-Based Training Committee (AICC)* [AIC03] ist eine international operierende Vereinigung und wurde 1988 mit dem Ziel gegründet, Vorschläge für die Entwicklung, Bewertung und den Vertrieb von computerunterstützten Lern- und Trainingseinheiten zu erarbeiten. Der AICC Standard ermöglicht die von einem Learning Management System (LMS) unabhängige Entwicklung von Lerneinheiten und bietet Firmen langfristig Investitionssicherheit, weil durch die Einhaltung des Standards Lerneinheiten (*Computer Based Training (CBT)*) mit Lernplattformen (*Computer Manage Instruction Systems (CMI)*) verschiedener Anbieter lauffähig sind. Der CMI-Standard verwaltet aber nicht nur Kursinhalte, sondern auch Studierende. Lehrende können Studierenden Inhalte zur Verfügung stellen und den Lernfortschritt der Studierenden auswerten. Grundsätzlich

gibt es in einem CMI-System [Bergstrom01] fünf Komponenten, die durch den Standard spezifiziert werden:

- Die **Kursstruktur**, aufgeteilt in *courses*, *blocks* und *lessons*, die hierarchisch angeordnet werden können,
- eine Komponente für **Tests**,
- eine **Verwaltungskomponente** für Studierende, die sowohl die Registrierung als auch demographische Erhebungen vorsieht,
- eine **Betreuungskomponente** für Studierende, die Aufgaben Studierenden zuweist und deren Lernfortschritt überwacht,
- die **Datenverwaltung**, die Lehrenden und Entwicklern das Auffinden und das Verwalten von Kursinhalten erleichtert.

Ein wichtiger Aspekt, der durch den AICC CMI-Standard beleuchtet wird, behandelt die Frage, wie Kurse unter verschiedenen CMI-Systemen ausgetauscht werden können und wie Kurse, die von Drittanbietern entwickelt wurden, in das eigene CMI eingebunden werden können. Zur Klärung dieser Frage untersucht der Standard drei Aspekte der Interoperabilität: Die Übertragbarkeit von Kursstrukturen, Verhalten und Inhalt auf andere Systeme, die Kommunikation zwischen Kursen und dem CMI-System sowie das Abspeichern bzw. das Verwalten von Informationen über Studierende. Im Zuge der weiteren Analyse des Standards steht vor allem der Kurs als modulare Lerneinheit im Mittelpunkt des Interesses, der wie folgt definiert wird:

> "A course may be as simple as a few lessons to be viewed sequentially, or it may be as complex as hundreds of lessons, some of which are prerequisites to others and some of which may be experienced in any order. Basically, courses have three components: instructional elements, structure, and behavioral elements ([Bergstrom01], Seite 45)".

Der Leitgedanke, der hier verfolgt wird, ist die Aufteilung von Kursen in kleine Einheiten, wie Lessons oder Tests, die ungeachtet von ihrer Reihenfolge definiert werden. Eine separate Struktur legt fest, ob und wie einzelne Kurselemente strukturell angeordnet und eingesetzt werden und hängt von der Expertise des Lernenden ab. Abhängig vom Vorwissen des Studierenden kann die Kursstruktur eher einfach oder sehr viel komplexer konsumiert werden. Diese Struktur hilft ebenso bei der Überführung von Kursinhalten auf andere Systeme, weil hiermit einzelne Kurseinheiten nicht mühevoll von Hand kopiert werden müssen. Abbildung 2.9 zeigt exemplarisch eine vom Kurs separierte Struktur, die mit mehreren Kurseinheiten assoziiert ist.

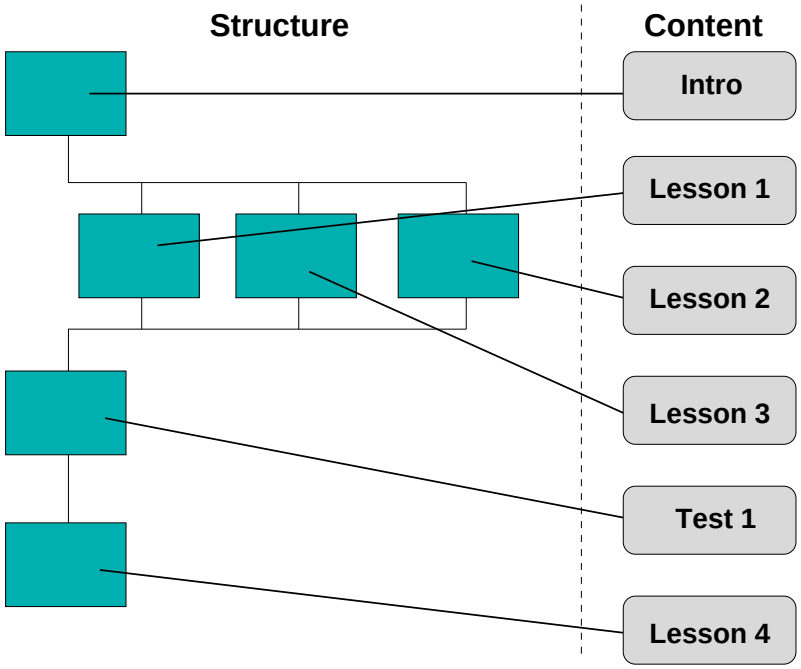


Abbildung 2.9: Variable Kursstruktur für Lernmodule zum leichten Transfer von einem CMI-System auf ein anderes System.

2.3.5 ARIADNE

Die *Aliance of Remote Instructional Authoring and Distibution Networks for Europe (ARIADNE)* [Ari05] ist ein EU-Projekt im 4. Rahmenprogramm und wurde mit der Zielsetzung gegründet, Werkzeuge und Methoden zur Produktion, Verwaltung und Wiederverwendung von rechnergestützten und pädagogischen Lerneinheiten sowie für die Erstellung von Curriculae zu entwickeln. Hierbei wird nicht nur der technische Rahmen definiert, sondern auch kulturelle und sprachliche Aspekte bei der Entwicklung berücksichtigt. Insbesondere soll die Indexierung, wie z.B die Eingabe von Metadaten, und das Auffinden von pädagogischen Inhalten erheblich vereinfacht werden.

Als Ergebnis dieses Vorhabens präsentiert ARIADNE die Ergänzung der im LOM-Standard fehlenden Einstellungen zu pädagogischen Merkmalen sowie die Entwicklung des international vernetzten **Knowledge Pools (KPS)**, der als Referenzimplementierung für die am LOM-Standard neu eingebrachten Vorschläge dient. Die entstandene Spezifikation ist ein Klassifizierungs-Schema, das aus sechs Kategorien besteht:

- generelle Informationen über Ressourcen,
- Semantik der Ressourcen,
- pädagogische Attribute,
- technische Charakterisierung,
- Nutzungsbedingungen,

- Meta-Metadaten.

Das KPS ist als verteiltes Repsoitory für Lerneinheiten realisiert, auf das mit voneinander unabhängigen Werkzeugen zugegriffen wird. Einerseits können Lehrende Inhalte einstellen und pflegen und andererseits ist es Studierenden möglich, durch Verwendung von Werkzeugen zur Indizierung und Suche, Inhalte über das Netzwerk verteilter Repositories zu beschaffen.

2.3.6 Analyse

In den letzten zehn Jahren gab es in den verschiedenen Zweigen des E-Learnings intensive Bemühungen, die auf diesem Gebiet erzielten Ergebnisse der Forschungsinstitutionen und Gruppen, wie dem IMS, durch Standardisierungsorganisationen (ISO oder IEEE) in internationale Standards umzusetzen. Diese relativ jungen Gruppen haben, im Vergleich zu anderen Domänen, den Einzug in internationale Unternehmen noch nicht abgeschlossen. Es werden voraussichtlich noch Jahre vergehen, bis sich der LOM- und der SCORM-Standard als unverzichtbar erweisen. Dieses fällt vor allem bei der Betrachtung von Autorensystemen (vgl. Abschnitt 3.4) oder Learning Management Systemen (vgl. Abschnitt 3.3) auf, die bemüht sind, junge Standards als neue Leistungsindikatoren, sowohl aus der Sicht der Technik als auch aus der des Marketings, anzubieten. Die gängigen Systeme sind jedoch noch weit von der Akzeptanz durch Endbenutzer und Endbenutzerinnen entfernt. Im Alltag ist jeder Mensch ständig mit einer Reihe von Standards konfrontiert, ohne es überhaupt zu bemerken. So wird beispielsweise von jeder Tankstelle der gleiche Kraftstoff zum Betanken für Kraftfahrzeuge angeboten, unabhängig vom Modell oder Land. Ebenso können Videogeräte, Fernseher und DVD-Player über die gleichen Kabel und Stecker problemlos verbunden werden. Im E-Learning ist dieser Grad an Integrität jedoch noch nicht erreicht. Viele Forschungseinrichtungen und industrielle Institutionen entwickeln proprietäre Lösungen, die inkompatibel zu existierenden Systemen sind und daher die Akzeptanz von E-Learning-Standards nicht gerade fördern [Saddik01a][Bohl02].

2.4 Lernobjekte (Learning Objects)

Bei der Erstellung und Entwicklung von multimedialen Inhalten kommt es häufig vor, dass Inhalte für eine spezielle Thematik unter hohem Zeit- und Kostenaufwand entwickelt und getestet werden. Gerade für den Hochschulbereich, in dem an zahlreichen Instituten und Einrichtungen dieselben Grundlagenveranstaltungen durchgeführt werden und nahezu der gleiche Stoff vermittelt wird, ist eine mehrfache Neuentwicklung lernbegleitender Inhalte ungewünscht. Aus diesem Grund werden im folgenden Ansätze vorgestellt, die Konstruktionsprinzipien für Kurse vorstellen, bei denen nicht die Entwicklung monolithischer Kurse im Vordergrund steht, sondern vielmehr auf die modulare Konstruktion mit *Learning Objects (LO)* [Hodgins02, Downes00, Downes04, Heng03, Ignatiadis03, Wiley00a] verwiesen wird. Lernobjekte werden für die Wiederverwendung von Lehrinhalten benutzt. Hierbei ist es wichtig, den Lehrinhalt derart aufzuteilen, dass modulare Einheiten entstehen, die in an-

deren Lernmaterialien eingebunden und wiederverwendet werden können. Die Aufteilung in modulare Lerneinheiten ist in seinen Grundzügen durch das Paradigma der objektorientierten Programmierung (OOP) [Wiley00b, Balzert00, Züllighoven98, Rumbaugh91, Meyer97] motiviert, bei der Software-Komponenten klassifiziert und durch Objekte repräsentiert werden. Ein wichtiger Aspekt bei der Konstruktion von Lernobjekten ist ihre Abgeschlossenheit. Das bedeutet, dass sie thematisch abgeschlossen sein müssen und nicht von anderen Inhalten abhängen. Eine solche Abhängigkeit tritt insbesondere dann auf, wenn z. B. in einer Aufgabenstellung zu dem Themenkomplex *Komplexe Zahlen* ein Hinweis auf die Definition der zugehörigen Operationen existiert. Deshalb wird grundsätzlich gefordert, die kleinste *sinnvolle* Größe bei der Entwicklung zu wählen.

Learning Objects bestehen in der Regel aus einer Menge von Informationseinheiten oder Datenobjekten, wie z.B Texte, Videos oder Programmen. Aus ihnen können durch Aggregation bzw. Komposition unterschiedliche Kurse zusammengestellt werden, die wiederum zu kompletten Lehrgängen kombiniert werden können (vgl. Abbildung 2.10, Baumgartner [Baumgartner02], Seite 33).

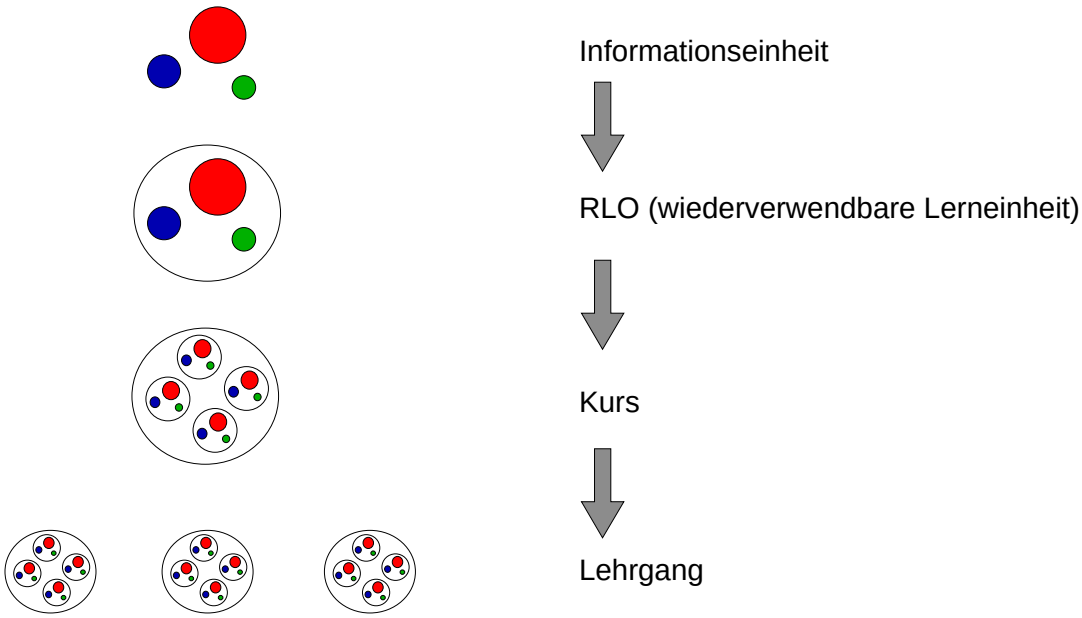


Abbildung 2.10: Das modulare Prinzip der Reusable Learning Objects (RLO)

Eine andere Darstellung von Lernobjekten wurde bei der Firma *Autodesk* eingeführt (vgl. Abbildung 2.11)[Hodgins02]. In dieser fünfstufigen Darstellung werden Informationsobjekte aus atomaren Medienelementen und Assets gebildet. Aus diesen Informationsobjekten lassen sich Lernobjekte bauen, die zu höheren Lerneinheiten oder kompletten Kapiteln zusammengesetzt werden. Werden diese ebenso kombiniert, ist das Resultat ein Kurs oder ein Buch. Je höher die Aggregationsstufe, desto schwieriger wird die Wiederverwendung der erzeugten Inhalte.

Das Learning Technology Standard Commitee (LTSC) (vgl. Abschnitt 2.3.3) hat den Begriff Learning Objekt wie folgt definiert:

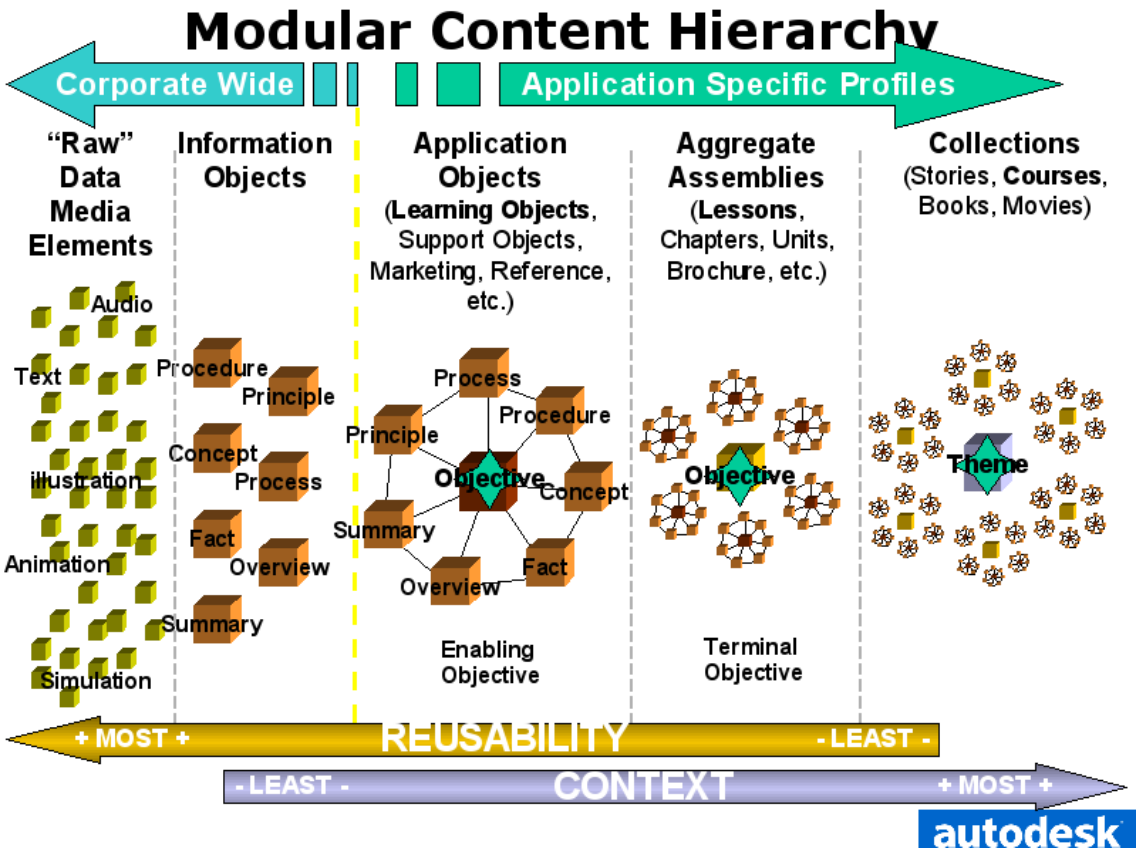


Abbildung 2.11: Modulare Content-Hierarchie

> "A learning object is defined as any entry -digital or non-digital that may be used for learning, education or training.([Electrical02], Seite 5)"

Baumgartner verwendet eine speziellere Begriffsbeschreibung für Learning Objects und definiert zusätzlich den Begriff *Reusable Learning Object (RLO)*:

> "Ein LO (Learning Object) ist die kleinste sinnvolle Lerneinheit, in die ein Online-Kurs zerlegt werden kann. Demnach kann ein LO entweder aus einem einzelnen Bild, einer Grafik, einem Text, einer Flash-Animation oder aus einer kurzen Anweisung mit einem definierten Lernziel und einem Test zur Lernerfolgskontrolle bestehen.
>
> Wenn diese LOs mit Metadaten versehen und zu größeren Online-Kurseinheiten kombiniert werden können, dann spricht man von RLOs (Reusable Learning Objects = wieder verwendbare Lernobjekte) ([Baumgartner02], Seite 31)."

Wie bereits herausgearbeitet wurde, sind Learning Objects modular und universell einsetzbar. Jedoch stellt sich die Frage, wie eine große Anzahl von ihnen moderat gehandhabt werden soll: Stellt man sich einen Pool zahlloser Lernobjekte vor und will gerade ein

bestimmtes Lernobjekt entnehmen, welches für einen zu erzeugenden Kurs benötigt wird, so wird zwangsläufig nach Mechanismen zur Selektion verlangt. Beispielsweise wäre eine Suchfunktion auf Schlagwortbasis eine plausible Lösung, wie es die Suchmaschinen beim Auffinden von Webseiten anwenden. Es gibt jedoch gravierende Unterschiede hinsichtlich Metainformationen zwischen Webseiten und Lernobjekten, die sich auf den jeweiligen Lernkontext beziehen. Die Daten, die Aufschluss über ein Lernobjekt geben, werden als *Metadaten* bezeichnet. Sie liefern beispielsweise Informationen über die maximale Lerndauer, den Grad der Schwierigkeit oder das Fachgebiet.

Metadaten gliedern sich in *objektive* und *subjektive* Metadaten. Erstere repräsentieren generelle Daten über ein Lernobjekt, die in den meisten Fällen durch ein Softwaresystem generiert werden. Hierzu zählen das Erstellungsdatum, der Autor, die Kosten oder die Lerndauer. Im Gegensatz dazu definieren subjektive Metadaten bewertende Einschätzungen über ein Lernobjekt, die für eine mögliche Identifikation eines Lernobjektes ausschlaggebend ist. Damit Metadaten von Lernplattformen und Repositories ausgewertet werden können, wurde soeben der Learning Objects Metadaten (LOM) Standard durch das IEEE verabschiedet (vgl. Abschnitt 2.3.3).

Ein zweiter wichtiger Aspekt der Lernobjekte bezieht sich auf deren Sequenzierung. In Abhängigkeit von individuellen Vorkenntnissen und Fähigkeiten kann die Reihenfolge von Lernobjekten adaptiert werden. Dieses ermöglicht Learning Management Systemen (vgl. Abschnitt 3.3) ein Profil des Lernenden zu erstellen und einen auf den Lernenden abgestimmten Lehrstoff selektiv anzubieten. Das hat den Vorteil, dass Lernende nicht Lernobjekte abarbeiten müssen, deren Inhalte bereits erlernt wurde. Darüber hinaus lassen sich Ordnungsrelationen für Lernobjekte definieren. Ein Beispiel für eine Relation auf Lernobjekten ist die chronologische Anordnung von Kursen oder der Komplexitätsgrad von Aufgaben.

2.4.1 Metaphern

Bei der Entwicklung neuer Verfahren und Systeme dienen oft Entwicklungen anderer Domänen oder die Natur als inspiratives Vorbild. Diese zumeist durch Metaphern beschriebenen Sachverhalte ermöglichen den Anwendern und Entwicklern einen möglichst raschen Zugang zu einer Thematik, in der sie nicht zwingend Fachkompetenz mitbringen müssen. Hierbei findet eine kontextuelle Abbildung eines Sachverhaltes einer Domäne ohne Fachkompetenz und Erfahrung auf den Sachverhalt einer anderen Domäne statt, in der ein zumindest substantielleres Wissen zu erwarten ist. Der Begriff der Metapher wird in Seufert [Seufert02] wie folgt definiert:

> "Metapher (griech.: *meta pherein*=anderswo hintragen) ist das sprachliche Bild, dessen Bedeutungsübertragung auf Bedeutungsvergleich beruht: das eigentlich gemeinte Wort wird durch ein anderes ersetzt, das eine sachliche oder gedankliche Ähnlichkeit oder dieselbe Bildstruktur aufweist wie z. B. Quelle für Ursache (aus [Rieser00], Seite 400)."

Der Mehrwert, der durch die Nutzung von Lernobjekten hervorgebracht wird, kann ebenso durch die Verwendung von Metaphern beschrieben werden. Hodgins [Hodgins02] gilt als Vater der Lernobjekte und hat sie anhand der LEGO-Baustein Metapher abgeleitet. Die Vision, die sich hinter dieser Metapher verbirgt, versinnbildlicht eine Welt, in der jede Lerneinheit in kleine handliche Stücke aufgeteilt ist, entsprechend der LEGO-Bausteine, die als elementare Teile zu einem komplexen Gebilde zusammengesetzt werden können, die auf unterschiedliche Arten zu einem individuellen, je nach Vorkenntnissen ausgerichteten Kurs aggregiert werden können. Hodgins hat ebenso bemerkt, dass die Verwendung von LEGO-Bausteinen unabhängig vom individuell präferierten lerntheoretischen Ansatz ist, d. h., dass neben einem konstuktivistischen (vgl. Kaptiel 2.2.2) Vorgehen ebenso ein behavioristisches Vorgehen möglich ist. Die Grundlage für die universelle Verwendung der LEGO-Bausteine bildet ihr standardisierter Aufbau. Durch die einheitliche Größe der Stecker ergeben sich folgende Eigenschaften [Wiley99]:

- jeder LEGO-Baustein ist kompatibel mit jedem anderen,
- LEGO-Bausteine können in beliebiger Weise zusammengesetzt werden,
- LEGO-Bausteine sind so einfach aufgebaut, dass sie von jedem zusammengesetzt werden können.

Diese Vorteile sollen auch für die Verwendbarkeit von Lernobjekten gelten. Standardisierungsbemühungen für Lernobjekte wurden bereits in Abschnitt 2.3.2 eingeführt und nehmen eindeutig Bezug auf die durch die LEGO-Metapher motivierten Eigenschaften.

Hodgins verfeinert den Ansatz zu der robusteren Analogie der *Bauindustrie* (Quelle: [Hodgins02], Seite 76). Laut Hodgins werden rund 85-95 Prozent der bei der Konstruktion neuer Bauwerke verwendeten Komponenten vorgefertigt und in Abhängigkeit vom Bauvorhaben ausgewählt. Als Komponenten werden unter anderen Türen, Schränke oder Fenster bezeichnet. Obwohl diese Komponenten einen immensen Spielraum an Kreativität bieten, werden globale Richtlinien bei deren Fertigung berücksichtigt. So ist z. B. zu gewährleisten, dass der Konstrukteur beim Entwurf einer Tür die vorgegebene Höhe und Breite berücksichtigt, damit der Architekt auf eine möglichst große Auswahl vorgefertigter Türen zurückgreifen kann und sich nicht von vornherein auf ein bestimmtes Modell festlegen muss.

David Willy schlägt hingegen eine andere Form der Konkretisierung trivialer LEGO-Bausteine vor [Wiley99]: Er favorisiert die *ATOM-Metapher* als abstraktes und reales Vorstellungsmodell für Lernobjekte, die eine festgelegte Aggregationsbeziehung zwischen Atomen vorsieht, aus der sich größere Kristallstrukturen ergeben. Willy hat dies mit folgendem Satz beschrieben:

> ”Rather than thinking about LEGOs or Lincoln Logs, perhaps our minds should be pointed toward something like a “learning crystal”, in which individual learning objects are combined into useful structure ([Wiley00a], Seite 20).”

Gegenüber der LEGO-Metapher zeichnet sich das Atom-Modell durch folgende Eigenschaften aus:

- Nicht jedes Atom kann mit einem beliebigen Atom kombiniert werden,
- Atome können nur auf bestimmte Weise, vorgegeben durch ihre Struktur, zusammengesetzt werden,
- für das Zusammensetzen von Atomen wird neben dem nötigen Verständnis auch die Erfahrung benötigt.

2.4.2 Analyse

Nachdem die drei wesentlichen Denkweisen auf dem Gebiet der Metaphern für Lernobjekte eingeführt wurden, soll nun in Hinblick auf die Fragestellung dieser Arbeit eine abschließende Diskussion stattfinden. Die LEGO-Metapher zeichnet sich als einfaches Mittel aus, um bei der Erstellung von Inhalten komplexe Vorgehensmodelle zu beschreiben und diese zu diskutieren. Ein solcher Ansatz bietet eine solide Basis für die Erstellung modularer Kurseinheiten, die speziell für interdisziplinär einsetzbare Kurse unbedingt notwendig ist. Das bedeutet, dass Entwickler und Entwicklerinnen von Inhalten die selbe modulare Denkweise für den Konstruktionsprozess entwickeln, damit eine mögliche Kursaggregation erst möglich wird. Bedauerlicherweise befasst sich die Metapher ausschließlich mit der Aggregation von Bausteinen und nicht mit deren Lagerung und Auswahl, wodurch weitere interessante Betrachtungsfelder und Fragen aufgeworfen werden. Die Atom-Metapher bietet im Vergleich zu der LEGO-Metapher einen stark restriktiven Ansatz, der die Handlungsfreiheit bei der Konstruktion von Kursen massiv einschränkt. Demnach lassen sich nur sinnvolle Konstrukte erzeugen, was sicherlich nicht im Sinne der Autoren und Autorinnen ist. Des weiteren lässt die auferlegte hierarchische Ordnung Kristall, Atom, Proton und Quark keine verschachtelte Struktur zu, die bei der Kombination unterschiedlich großer Kurse zwangsläufig zu großen Problemen führt. Als Beispiel stelle man sich zwei Kurse vor: Einer enthält ein umfangreiches Vorlesungsskript über Analysis und der andere kleinere einen Kurs mit dem Thema Nullstellenberechnung. Bei dieser Betrachtung ist absolut unklar, ob der *kleine* Kurs einem Kristall, einem Atom oder sogar einem Proton entspricht. Eine derart restriktive Kurshierarchie unterbindet die Aggregation mit Kursen anderer Ebenen.

Die Bauindustrie-Metapher bietet hingegen die Verwendung komplexer Komponenten an, was mit der zeitaufwändigen und langwierigen Entwicklung von Lernobjekten vergleichbar ist. Ein wichtiger, von Hodgins nicht beleuchteter Aspekt ist die Rolle des Architekten, der zwangsläufig in den Entwicklungsprozess eines Gebäudes involviert ist. Die Auswahl einer speziellen Tür, mit allen individuellen Merkmalen, wird meistens zu Beginn eines Bauvorhabens festgelegt und trifft auf alle Objekte desselben Typs zu. Würde dieses nicht berücksichtigt werden, ließe sich das Haus sicherlich schwieriger verkaufen, denn wer möchte schon in einem Haus wohnen, in dem jedes Objekt eine andere Ausprägung besitzt. Was bei der Entwicklung von Bauvorhaben jedem klar zu sein scheint, wird bei der Konstruktion von Lernobjekten völlig außer Acht gelassen.

2.4.3 Granularität

Die vorgestellten Metaphern haben bereits gezeigt, dass die Aussage, wie groß ein Lernobjekt sein sollte, nicht ohne weiteres zu beantworten ist. Die LEGO-Metapher beschreibt kleine Bausteine, von denen tausende ein komplexes Gebilde ergeben. Im Gegensatz werden sowohl bei der ATOM- als auch bei der Bauindustrie-Metapher komplexere, aus anderen elementareren Teilen zusammengesetzte Bausteine herangezogen. Ein großes Problem tritt bei der Festlegung der Granularität eines Lernobjektes auf. Je nach Einschätzung der konstruierenden Person wird sie relativ angegeben und kann dadurch zu Kompatibilitätsproblemen führen. Pauschal lässt sich jedoch sagen, dass je kleiner Lernobjekte konstruiert werden, desto leichter können sie in andere Kontexte eingebettet werden [Hodgins02, Wiley04, Weitl04]. Der Nachteil liegt auf der Hand: Bei einer enorm großen Anzahl an Lernobjekten erhöht sich der Aufwand bei der Suche nach einem speziellen Lernobjekt und es müssen wesentlich mehr Metadaten zur eindeutigen Identifikation herangezogen werden. Letzteres erfordert ein hohes Maß an Bereitschaft, wie es z. B. bei der Metadatenspezifikation von OMDoc-Inhalten (vgl. Abschnitt 3.2.3) der Fall ist und ist von den meisten Autoren und Autorinnen wohl nicht zu erbringen.

Als Lösungsvorschlag wird in der IEEE LOM Spezifikation (vgl. Abschnitt 2.3.3) der so genannte Aggregationslevel eingeführt (siehe Abb. 2.4.3), mit dem der Granularitätsgrad explizit eingestellt werden kann (LOM-Spezifikation [Electrical02], Seite 15).

Aggregation Level - The functional granularity of this learning object:

1. the smallest level of aggregation, e.g. raw media data or fragments.
2. a collection of level 1 learning objects, e.g., a lesson.
3. a collection of level 2 learning objects, e.g., a course.
4. the largest level of granularity, e.g., a set of courses that lead to a certificate.

Abbildung 2.12: Aggregationslevel für Lernobjekte nach dem IEEE Standard für Metadata (LOM)

Die vom IEEE eingeführten Aggregationsstufen widersprechen zunächst den Aussagen von Hodgins und Wiley und zwar, indem ein Lernobjekt so klein wie möglich sein soll. Ignatiadis beschreibt dieses wie folgt:

> "A Learning Object must be able to satisfy a single learning objective and should not introduce many different ideas. There is no reason why a large learning object should be designed, since a compound learning object consisting of two or more other Learning Objects can be built, which can serve the same purpose ([Ignatiadis03], Seite 10)."

Verfolgt man diesen Ansatz weiter, so würden alle Kurse oder Module aus der maximalen Anzahl von Lernobjekten bestehen, die bei großen Kursen schnell unübersichtlich werden und schlecht zu warten sind. Wird jedoch der Aspekt der *Verschachtelung* berücksichtigt, so könnten Lernobjekte aus anderen bestehen, wobei eine gewünschte Dekomposi-

tion sicherlich kein allzu großes Problem darstellt. Aufgrund der Generalität von Standards ist die Festlegung von Aggregationsstufen sinnvoll, weil die subjektive Selektion einer Granularitätsstufe unterschiedlich motiviert sein kann. Ein solches Auswahlkriterium könnte die Zeit sein, die für die Bearbeitung eines Lernobjekts benötigt wird, die Sequenzierung eines Kurses oder der Aspekt der Wiederverwendbarkeit von Lernobjekten.

Kapitel 3

Technologien

In diesem Kapitel werden technische Realisierungen von Lerninhalten analysiert und Resultate aus ausgewählten Projekten vorgestellt. Danach findet eine Auswertung kommerzieller und wissenschaftlicher Systeme statt, um einerseits deutlich zu machen, welche zum Teil großen Defizite die analysierten Systeme aufweisen und andererseits, um wichtige Erkenntnisse aus diesen Arbeiten zu gewinnen.

3.1 Wiederverwendung von Lernmaterialien

Dieser Abschnitt diskutiert den aktuellen Stand der Entwicklung von Systemen und Technologien zur modularen Kurskonstruktion mit dem Fokus einer konsistenten Kurserzeugung und Wiederverwendung bestehender Ressourcen. Hierbei soll der Aufbau und die dafür entwickelten Datenstrukturen untersucht werden, die Aufschluss über die technische Modellierung eines Systems für interdisziplinär nutzbare Lerneinheiten geben sollen. Im Folgenden werden zwei technisch verschiedene Ansätze betrachtet, die unterschiedliche Strategien für das Einlesen, Verwalten und Transformieren von Materialien verfolgen.

3.1.1 Slicing Books

Am Koblenzer Institut für Informatik wurde die Slicing-Book-Technologie [Dahn02, Dahn01] entwickelt, mit der es möglich ist, digitale Bücher, unabhängig von einer speziellen Fachrichtung, derart aufzubereiten, dass sie in kleine Teile, so genannte *Slices*, zerlegt werden, um sie dann zu neuen Büchern zusammen setzen zu können. Dahn [Dahn05] definiert den Begriff Slice wie folgt:

> "A potential slice is a connected part of a document that can be reused under well defined conditions (aus [Dahn05], Seite 3)."

Der Inhalt wird hierarchisch angeordnet, wobei Relationen zwischen Slices mit Hilfe von Metadaten definiert werden [Valerius01]. Dabei sind insbesondere Abhängigkeiten von Slices gemeint, die bei der Dekomposition von elektronischen Büchern beachtet werden müssen. Ein Beispiel hierfür ist die Abhängigkeit eines Beweises von einer zugehörenden

Definition. Ohne die Definition kann der Beweis nicht in ein anderes Buch eingebunden werden. Die Slices und die zugehörigen Metainformationen werden auf einem zentralen Server gespeichert, der, abhängig vom Benutzer- bzw. Benutzerinnenprofil, eine individuelle Zusammenstellung von Kursunterlagen erlaubt. Zusätzlich zu der Wiederverwendung von bestehenden elektronischen Büchern, wird die Neuentwicklung von Slices unterstützt.

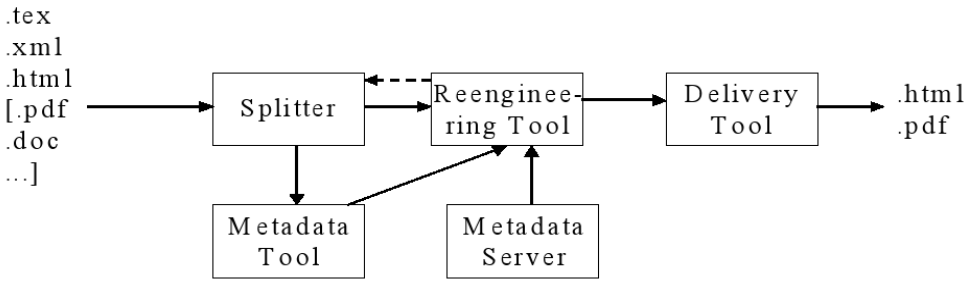


Abbildung 3.1: System Modules of Slicing Books

Abbildung 3.1 (aus [Dahn02]) zeigt die Interaktion der Module, die für die Verarbeitung eines Dokuments notwendig sind. Der erste Disaggregationsschritt wird vom *Splitter* übernommen. Zunächst werden Metadaten, wie z. B. Referenzen oder Schlüsselwörter (Keywords) durch den Splitter aus dem Dokument extrahiert. Da viele Dokumente nicht ausreichend mit Schlüsselwörtern versehen sind, werden sie in einem zweiten Schritt automatisch den extrahierten Slices zugewiesen. Das Ergebnis wird dem Reengineering Tool übergeben, das mit einem Schlüsselwort-Server verbunden ist. Dieser liefert Verweise auf Slices anderer Bücher und bietet dem Anwender bzw. der Anwenderin eine Liste möglicher Schlüsselwörter, die für die anstehende Überarbeitung des Dokuments relevant sein können.

3.1.2 ITO-Projekt

Das Ziel des ITO-Projekts [Finsterle02] der Universität Stuttgart ist die Entwicklung von modularen Lehreinheiten für die Fächer Elektrotechnik, Informatik und Informationstechnik. Die realisierten Module sollen im Rahmen des Projekts *Multimediale englischsprachige Vorlesungen* für den Studiengang *Information Technology* aufbereitet werden.

In diesem Projekt wurde ein System entwickelt, mit dem es möglich ist, bereits verfasste Skripte und Unterlagen, die speziell für die Lehre entwickelt wurden, derart zu konvertieren, dass sie für die Einbettung in eine Online-Präsentation geeignet sind. Für die Wiederverwendung von MS Word, MS Powerpoint und OpenOffice Dokumenten wurde die OpenOffice-Anwendung als Mediator verwendet. Durch die zahlreichen Import- und Export-Filter und das auf XML basierte Dokumentenformat [OOS02], bietet sich diese Anwendung sehr gut für eine Dokument Transformation an. Ein Transformator zerlegt Dokumente in die Teile *Inhalt*, *Metadaten*, *Objekte*, *Struktur* und *Layout* und speichert diese Informationen zusammen mit dem Quelldokument in einer Datenbank ab (vgl. Abbildung 3.2).

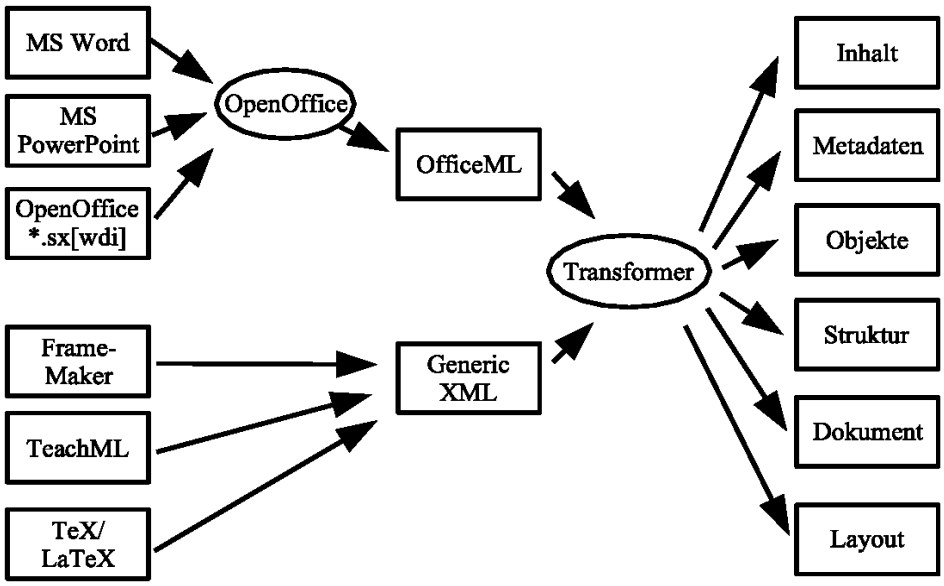


Abbildung 3.2: Import

Das analysierte Dokument kann jetzt für die Gernerierung unterschiedlicher Ausgabeformate eingesetzt werden (vgl. Abbildung 3.1.3). Soll z. B. eine WEB-Präsentation erstellt werden, wird die gespeicherte Strukturinformation in Verbindung mit den vorhandenen Metadaten für die Generierung einer Navigation verwendet. Der unterhalb der Navigation angezeigte Inhalt wird entsprechend aus der Struktur, dem Inhalt und den vorhandenen Objekten generiert. Um eine inhaltliche Bearbeitung durchzuführen, wird auf das ursprüngliche Dokument zurückgegriffen.

3.1.3 Analyse

Die in diesem Abschnitt vorgestellten Ansätze, bestehende Dokumente technisch zu verarbeiten, beziehen sich auf die beiden zu untersuchenden Themen *Integrierbarkeit* und *Formatierbarkeit* (vgl. Tabelle 2.1). Der Ansatz der bei der Slicing Book-Technik gewählt wurde, zerlegt ein bestehendes Dokument in kleine, voneinander abhängige Teile, die mit Hilfe von Metadaten vernetzt werden. Hierbei wird das Reengineering mit Metadaten als aktiver Prozess verstanden, der maßgeblich an der Gestaltung neuer Dokumente beiträgt. Dieses schließt ebenso die Definition der Metadaten mit ein, die je nach Umfang des zu transformierenden Dokuments einen nicht zu unterschätzenden Zeitaufwand mit sich bringt.

Das ITO-Projekt bietet eine technische Lösung, um unterschiedliche Formate in ein allgemeines Zwischenformat zu überführen, dass, zu einem späteren Zeitpunkt, in ein anderes Ausgabeformat konvertiert werden kann. Hierbei ist die Aufspaltung in die Teilelemente Inhalt, Metadaten, Objekte, Struktur und Layout ein guter Ansatz, der in dieser Arbeit berücksichtigt werden muss. Dennoch fehlen Mechanismen, um die Aspekte *Modularität*,

also die Aggregation mit anderen Dokumenten, und die damit verbundene *Granularität* umzusetzen.

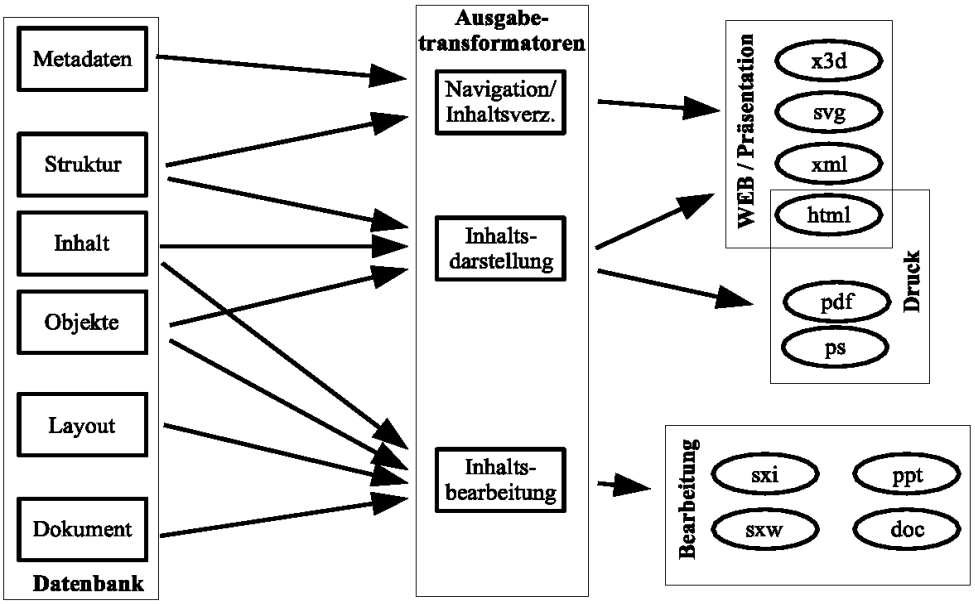


Abbildung 3.3: Export

3.2 Formale Beschreibung von Lernmaterialien

Im Zuge der anstehenden Betrachtungen soll analysiert werden, wie Lernmaterialien beschaffen sind, formuliert und konstruiert werden. Lernmaterialien werden voranging als HTML-Dokumente oder als PDF-Dokumente Studierenden angeboten. Es hat sich allerdings gezeigt, dass es auf diesem Gebiet vielseitige Bemühungen gibt, Definitionen und Spezifikationen für Lehrinhalte vorzuschlagen, die es teilweise bis zur Standardisierung geschafft haben. Die meisten in diesem Zusammenhang dargelegten Vorschläge basieren technisch auf der Auszeichnungssprache XML (eXtendable Markup Language, vgl. Abschnitt 3.2.1). In Arbeiten von [Gersdorf02, Freitag02a, Kohlhase02, Belqamsi02, Baudry02b, Baudry03, Roisin98, Net05, Verbert04] werden Ergebnisse bei der Konstruktion von E-Learning-Inhalten präsentiert, die mit XML formal beschrieben werden. Sie zeigen außerdem, dass der Einsatz von XML die Implementierung von Suchsystemen für Kursinhalte erheblich vereinfacht [Wollowski02]. Aus diesem Grund wird eine kleine Einführung in die Grundlagen der Markupsprachen gegeben, um das notwendige Verständnis für die anstehenden Ausführungen, Analysen und Bewertungen zu vermitteln. In diesem Abschnitt sollen allein die technischen Möglichkeiten aufgezeigt werden; Themen wie z. B. Adaptive Lerninhalte (vgl. [Brusilovsky01],[Brusilovsky98]) bleiben daher unberücksichtigt.

3.2.1 SGML / XML

Mitte der 80er Jahre wurde nach einer Möglichkeit gesucht, Daten auf einfache Weise abzuspeichern, zu bearbeiten und zwischen Anwendungen auszutauschen. Hierbei wurde insbesondere der Forderung nachgegangen, die inhaltliche Struktur von Dokumenten von ihrer Darstellung so zu trennen, dass sie für unterschiedliche Anwendungszwecke eingesetzt werden können. So können beispielsweise Verlage mit einem solchen Format Dokumente bzw. Artikel austauschen und abhängig vom vorgegebenen Layout Konvertierungen vornehmen. Jedoch gibt es auch Anwendungsszenarien, bei denen Daten zwischen Applikationen ausgetauscht werden müssen, die frei von Layout-Eigenschaften sind. Aus diesem Grund hat das ISO-Gremium *(International Strandard Organization)* 1986 die normierte Auszeichnungssprache *SGML* verabschiedet [Szillat95, Goldfarb90]. SGML bietet die Möglichkeit, eigene Strukturelemente, wie z. B. Absätze, Überschriften oder textuelle Hervorhebungen, zu definieren, ohne zu spezifizieren, mit welcher Schriftgröße, Schriftart oder Farbe sie gedruckt werden. Des weiteren wird durch SGML die Reihenfolge der Auszeichnungen festgelegt und durch ein spezielles Programm, den so genannten Dokumentparser, wie z. B. Xerces [Xer05] analysiert und übersetzt. SGML erwies sich jedoch auf Grund seiner Komplexität als zu unhandlich, um schnelle und preisgünstige Parser entwickeln zu können. Deshalb gibt es seit Beginn der 90er Jahre Bemühungen eine vereinfachte Form zu entwickeln. Als Ergebnis wurde Mitte der 90er Jahre die *Extensible Markup Language (XML)* vom W3C (World Wide Web Consortium) verabschiedet. Der wesentliche Unterschied zu SGML liegt in der Überprüfung der Dokumente. Zur Vereinfachung führt XML das Prinzip der Wohlgeformtheit ein, welches besagt, dass die Struktur von Dokumenten eindeutig sein muss. Dieses ermöglicht dem Parser, das Dokument einzulesen, ohne Annahmen über die Struktur treffen zu müssen, indem eine DTD (siehe Abschnitt 3.2.1) zur Überprüfung herangezogen wird. Des Weiteren wird gewährleistet, dass nachfolgende Programme fehlerfrei arbeiten, da die Integrität und Konsistenz gewährleistet wird [Ray01].

Zentrales Konstrukt ist das *Element*, das zur Textstrukturierung dient. Ein Dokument besitzt grundsätzlich ein Wurzelelement, welches mehrere Kindelemente beinhaltet. Diese Kindelemente können weitere Kindelemente enthalten. Elemente sind also Container, die genau einem übergeordneten Element unterstellt sind. Zudem können sie eine Kombination von Elementen und Textknoten enthalten, die vom Parser wie Objektknoten behandelt werden, jedoch nicht über ein syntaktisches Konstrukt verfügen. Dieses impliziert, dass aus jedem Dokument eine Repräsentation als Baumstruktur abgeleitet werden kann. Damit ein Parser die Struktur analysieren kann, werden zur Hervorhebung Sonderzeichen verwendet, welche sich vom eigentlich zu verwendeten Zeichensatz des Dokuments unterscheiden. Der XML- und SGML-Standard sehen hierfür die Schreibweise von Elementen in spitzen Klammern vor, in die der Bezeichner eingeschlossen wird. Dem XML-Standard entsprechend, muss jedes geöffnete Element auch wieder geschlossen werden. Dieses wird wiederum durch einen in spitzen Klammern eingeschlossenen Bezeichner erreicht, mit dem Unterschied, dass dieser einen Schrägstrich als Präfix führt. Das folgende Beispiel zeigt eine einfache XML-Struktur eines Buchs mit zugehörigem Titel und Autor.

Listing 3.1: Beispiel einer XML-Deklaration

```
<Buch>                                                    1
  <Title>
    Einführung in XML
  </Title>                                                4
  <Autor>
    Erik T. Ray
  </Autor>                                                7
</Buch>
```

Das *Attribut* ist ein wichtiger Zusatz zum Elementbegriff. Er fungiert als Informationsspeicher und kann dazu benutzt werden, Elementen eindeutige Zustände zuzuordnen. Abbildung 3.1 illustriert den syntaktischen Aufbau eines Elements mit zwei Attributen. Attributwerte werden in Anführungszeichen gesetzt und dürfen innerhalb eines Elements genau einmal vorkommen. In einer Element-Definition können Attribute – im Gegensatz zum Funktionsaufruf einer Prozedurdefinition – an beliebige Positionen gesetzt werden. Die Reihenfolge ist für den Parser nicht relevant.

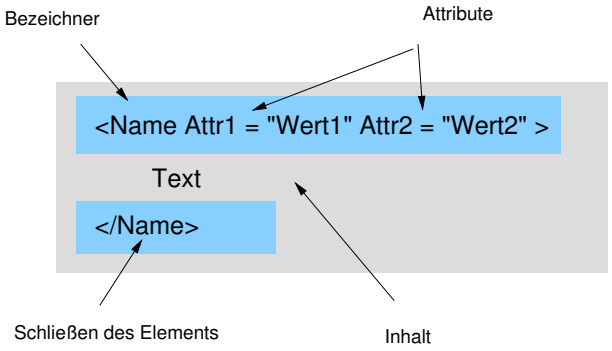


Abbildung 3.4: Syntax eines XML-Elements

Ein weiteres Leistungsmerkmal von XML ist das Konzept der *Namensräume*. Ein Namensraum beschreibt eine Menge oder Gruppe von Elementen und Attributen, die genau einem Kontext zugeordnet werden. Gäbe es kein solches Mittel, würden beim Übersetzen von Dokumenten zwangsläufig Konflikte auftreten. Als Beispiel sei hier die Analogie zum Wortumfang der deutschen Sprache genannt [Ray01]. Es gibt zahlreiche kontextabhängige Begriffe, wie z. B. das Wort *Golf*. Der Mensch ist in der Lage, kontextabhängig zu entscheiden, ob es sich um das Spiel, das Auto oder einen Einschnitt des Meeres im Festland handelt. Die Maschine handelt hingegen deterministisch und trifft keine eindeutige Zuordnung. Die XML-Spezifikation sieht hierfür eine Präfixnotation vor, die – abgetrennt durch das Doppelpunktsymbol – den Kontext des Attribut- bzw. Elementbezeichners festlegt. Ein eher technisches Beispiel ist der Einsatz von Namensräumen bei der Transformationssprache XSL (vgl. Abschnitt 3.2.1): Elemente lassen sich in die zwei Gruppen *Befehle* und *Ausgabeelemente* unterteilen. Letztere werden dem resultierenden Dokument hinzugefügt. Befehle gehören dagegen zum Sprachumfang von XSL und besitzen deshalb das

Präfix `xsl:`. Der XSLT-Prozessor erkennt während der Verarbeitung des Dokuments eine Instruktion und führt sie aus. Elemente, die zum Inhalt gehören, können entweder keinem Namensraum oder einem zu `xsl` unterschiedlichen Namensraum zugeordnet sein. Zur Deklaration eines Namensraums wird das reservierte Attribut `xmlns:` verwendet. Es signalisiert dem Parser, dass alle Nachkommen eines Elements Teil des im Attribut spezifizierten Namensraums sind.

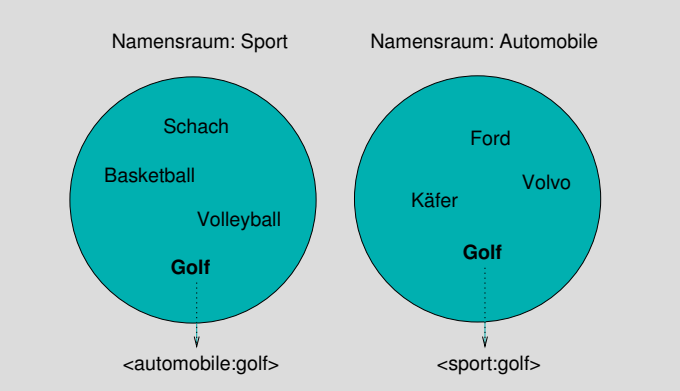


Abbildung 3.5: Eindeutige Zuordnung von Elementbezeichnern durch Präfixnotation

Als letztes wesentliches Konzept der Markupsprache XML sind *Entities* zu nennen. Sie werden als Platzhalter für Zeichen, Wörter, Wortgruppen oder XML-Fragmente benutzt und können an unterschiedlichen Stellen im Dokument vorkommen. Während der Dokument-Analyse substituiert der Parser die Entities mit den ihnen zugewiesenen Werten. Darüber hinaus können mit ihnen extern eingebundene XML-Fragmente referenziert werden. Die Entity-Notation am Anfang eines Dokuments lässt sich wie folgt formulieren:

Listing 3.2: Entity Notation

```
<!ENTITY Bezeichner "Wert">
```

Bisher wurden die Basiskonzepte von XML eingeführt, um deren syntaktische Struktur zu erläutern. Zudem wird im Folgenden die syntaktische Struktur semantischer Regeln und Beziehungen vorgestellt. Dokumentenmodelle werden formal spezifiziert und lassen sich durch eine entsprechende Grammatik formulieren. In diesem Abschnitt werden nun zwei Konzepte vorgestellt: Die *DTD (Document Type Definition)*, die grundsätzlich eine Sammlung von Regeln darstellt, sowie das *XML-Schema* [W3C05a], ein ausdrucksstarker und flexibler Mechanismus, zur datentyporientierten Definition von Dokumentenmodellen.

Validierung

Das zurzeit populärste Dokumentenmodell ist die DTD. Sie wurde bereits mit SGML vor der Verabschiedung von XML eingeführt und besteht aus einem Regelwerk, das die in einem Dokument zulässigen Elemente definiert. Ferner wird festgelegt, aus welchen Elementen ein Element besteht, deren genaue Reihenfolge und ob sie an einer bestimmten

Position auftreten dürfen. In [Ray01] wird die Typdefinition für ein Element als *Content-Model* bezeichnet. Neben der Elementstrukturierung werden außerdem deren Attribute spezifiziert. Es wird festgelegt, welche Attribute ein Element verwenden darf, dessen Datentyp und das Auftreten von Elementen an bestimmten Positionen. Im Folgenden ist eine Beispielregel zu sehen, die die Schablone für das Element `Dokument` definiert.

Listing 3.3: Beispiel einer Element-Definition

```
<!ELMENT Dokument
  (Autor*, Title, Jahr?, ( Absatz | Aufzaehlung )* )
>
```

Diese Notation ähnelt derer für reguläre Ausdrücke und bestimmt, dass jedes Dokument beliebig viele Autoren besitzt, genau ein Titel enthält, eventuell genau ein Erscheinungsjahr notiert und einer Sequenz willkürlich angeordneter Absätze und Aufzählungen folgt.

Neben dem bereits veralteten DTD-Konzept findet die Strukturdefinition mit XML-Schemata (XSD) immer größeren Zuspruch. Das XML-Schema dient – genau wie die DTD – zur Definition syntaktischer Regeln und deren Strukturierung. Es wurde vom W3C eingeführt, weil das DTD Konzept erhebliche Schwächen aufwies. In [Hansch02] sind die Vorteile des XML-Schema gegenüber dem DTD-Konzept herausgearbeitet:

- Jedes XML-Schema ist selbst ein XML-Dokument, wodurch keine spezielle Syntax definiert werden muss. Ferner kann ein XML-Schema wiederum durch ein Schema validiert werden, ohne eine weitere Metaebene einzuführen,
- komplexe Strukturen sind beschreibbar,
- Typprüfung wird durch vordefinierte und selbstdefinierte Typen ermöglicht,
- bei Datentypen wird Vererbung und Substitution unterstützt,
- Nullwerte können dargestellt werden,
- das Modularisieren und Wiederverwenden von XML-Schemata ist möglich,
- Namensräume verhindern Benennungskonflikte.

Tranformation

Nachdem die Struktur von XML-Dokumenten eingeführt wurde, stellt sich zwangsläufig die Frage, wie diese Dokumente bearbeitet werden können. Zu diesem Zweck gibt es so genannte *Transformatoren*, deren Aufgabe die Verarbeitung und Abbildung einer vorliegenden Struktur auf eine andere ist. Einerseits besteht die Möglichkeit, die den Dokumenten zugrunde liegende Baumstruktur auf die Datenstruktur einer Programmiersprache abzubilden und diese daraufhin zu modifizieren. Für die Sprache Java existiert z. B. die JDOM

[Harold02a] Bibliothek, die durch Einsatz von Entwurfsmustern, wie z. B. dem Fabrikmuster [Gamma95], eine abstrakte Schnittstelle zur Erzeugung und Manipulation von Objektbäumen anbietet. Auf der anderen Seite wurde die auf XML basierende Skriptsprache XSL [XSL99, Harold02b] (*eXtendable Stylesheet Language*) entwickelt, die ein mächtiges Mittel zur Dokumententransformation darstellt. XSL basiert auf dem Paradigma der funktionalen Programmierung und bietet daher keine Wertzuweisung an. Der Dokumentparser arbeitet das einzulesende XML-Dokument hierarchisch ab und wendet eine dem Element durch das XSL-Skript zugewiesene Regel an.

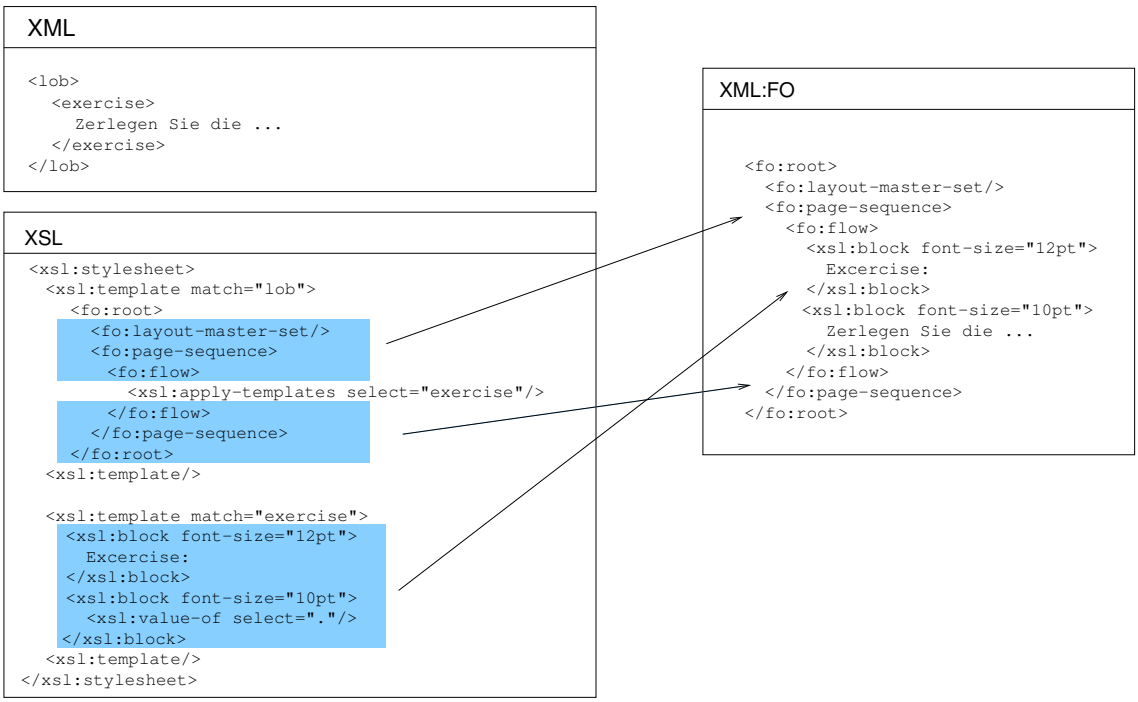


Abbildung 3.6: Beispiel: XSL-Transformation einer einfachen XML-Struktur in eine FO-Repräsentation

Die Regel beschreibt unter Verwendung von funktionalen Mitteln den zu generierenden Ausgabestrom. Mit entsprechenden XSL-Befehlen lassen sich komplex verschachtelte Strukturen mit wenig Aufwand konstruieren und warten. Zudem bieten *XPath* [W3C99] und *XQuery* [W3C05b] Mechanismen an, um auf andere Elemente an beliebigen Positionen im Dokument zuzugreifen. Abbildung 3.6 zeigt exemplarisch die für einen Transformator relevanten Dokumente. Das XSL-Dokument enthält jeweils eine Regel für die Elemente `lob` und `exercise`, die, ineinander verschachtelt, vom Transformator in die Ausgabedatei geschrieben werden.

3.2.2 DocBook

DocBook ist 1991 aus einer Kooperation des O'Reilly-Verlages und der Firma Hal. Computer Systems entstanden. Der auf XML basierende Standard zur strukturellen Beschreibung

von Büchern, Artikeln und anderen Dokumenten [Walsh02], liegt als DTD und als XML-Schema vor. Seit 1991 wurde DocBook in vielen Projekten, wie z. B. dem KDE2-Projekt, zur technischen Dokumentation eingesetzt. Die Struktur eines Reports oder eines Buchs kann mit DocBook modular und strukturiert aufgebaut werden. Hierzu wird in [Walsh02] eine kategorische Übersicht der verfügbaren Elemente angegeben (vgl. Tabelle 3.1). Listing 3.4 zeigt exemplarisch den Aufbau einer DocBook-Datei.

Kategorien	Erklärung
Set	Beschreibt eine Büchersammlung
Book	Definiert das Wurzelelement für ein Buch
Division	Gliedert ein Buch in verschiedene Teile auf
Component	Gliedert Bücher und Teile in Kapitel auf
Section	Kapitel können verschachtelte Abschnitte enthalten
Meta-information	Alle Elemente überhalb der Abschnitte können Metadaten anhaften
Block	Entsprechen einem Absatz und enthalten Listen, Tabellen, Abbildungen, usw.
Inline	Legt fest, ob ein Element innerhalb des Textflusses positioniert werden soll

Tabelle 3.1: Kategorien für Elemente zur Strukturierung von DocBook Dokumenten

Listing 3.4: DocBook Beispiel für einen einfachen Artikel

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<article>
  <artheader>                                                    3
    <title>My Article</title>
    <author>
      <honorific>Dr</honorific>                                  6
      <firstname>Emilio</firstname>
      <surname>Lizardo</surname>
    </author>                                                    9
  </artheader>
  <para> ... </para>
  <sect1>                                                        12
    <title>On the Possibility of Going Home</title>
    <para> ... </para>
  </sect1>                                                       15
  <bibliography> ... </bibliography>
</article>
```

3.2.3 OMDoc / OpenMath

Seit der Entwicklung des Hypertextes gibt es zahlreiche Bemühungen, mathematische Inhalte mit einem Webbrowser adäquat darzustellen. Neben der Formeldarstellung als Grafi-

ken, die sich im Übrigen nicht skalieren lassen und eine nicht zu unterschätzende Ladezeit seitens der Browser verursachen, wurden auch weitgehende Bemühungen zur Standardisierung von Formelbeschreibungen unternommen. Als Ergebnis entstanden die zwei miteinander konkurrierenden Standards OpenMath [Caprotti02] und MathML [Mat05b], von denen MathML bereits von den Browsern *Mozilla* und *Amaya* unterstützt werden. Mathematische Dokumente bestehen jedoch nicht nur aus Formeln, sondern auch aus Definitionen, Lemmata und Beweisen. Um diese Elemente ebenfalls spezifizieren zu können, wurde das Datenmodell *Open Mathematical Documents (OMDoc)* [Kohlhase00, Kohlhase02] entwickelt. OMDoc ist ebenfalls eine Markup Language und erweitert den OpenMath-Standard, indem Onthologien mathematischer Dokumente festlegt werden. Zu der einfachen Formelbeschreibung werden Aussagen über die Beziehungen zwischen den mathematischen Objekten getroffen. Mathematische Objekte können z. B. Theoreme, Beweise mit zugewiesenen Eigenschaften und Relationen sein. Außerdem unterstützt OMDoc den Metadaten-Standard *Dublin Core*, um Dokumente repräsentieren zu können, sowie pädagogische Elemente zur Festlegung von Lernszenarien.

Listing 3.5: OMDoc Repräsentation einer Definition

```
<definition id="def_order" for="order" type="simple">
  <metadata>
    <title xml:lang="en">
      Definition of the order of a group element
    </title>
    <difficulty level="easy"/>
    <field use="mathematics"/>
    <relation type="depends_on">
      <ref theory="Th1" name="group"/>
      <ref theory="elementary" name="positive_integer"/>
    </depends-on>
  </metadata>
  <CMP xml:lang="en" verbosity="3">
    If <OMOBJ><OMV name="G"/></OMOBJ> is a
    <ref xref="Th1_def_group"> group </ref> and <OMOBJ><OMA>
    <OMS cd="set1" name="in"/><OMV name="g"/><OMV
    name="G"/></OMA></OMOBJ>, then the order of <OMOBJ>
    <OMV name="g"/></OMOBJ> is the smallest positive
    integer <OMOBJ><OMV name="m"/> </OMOBJ> with
    <OMOBJ id="OMOBJ_o1">
      <OMA>
        <OMS cd="relation1" name="eq"/>
        <OMA>
          <OMS cd="Th1" name="power"/>
          <OMV name="g"/>
          <OMV name="m"/>
        </OMA>
        <OMS cd="Th1" name="unit"/>
```

```
    </OMA>
  </OMOBJ>.
  ...
</definition>
```

Listing 3.5 (aus [Melis03], Seite 13) zeigt exemplarisch eine OMDoc-Definition mit dem Titel *Definition of the order of a group element*. Die Definition ist mit Metadaten angereichert, welche die Definition beschreiben und deren genauen Einsatzkontext angeben. Zudem wird eine Abhängigkeitsrelation definiert, welche jene Objekte festlegt, die für das Verständnis essenziell sind. Im zweiten Abschnitt ist die Definition informell mit dem `<CMP>`-Element angegeben. Es enthält neben dem Definitionstext auch die Formelbeschreibung mit OpenMath.

Die in diesem Beispiel integrierte OpenMath Formelbeschreibungen beginnen mit dem XML-Element `<OMOBJ>` und enthalten Variablen (`<OMV>`), Symbole (`<OMS>`) und Applikationen (`<OMA>`). Die Variable g wird z. B. durch den OpenMath-Ausdruck `<OMOBJ><OMV name="g"/></OMOBJ>` definiert und die Potenz durch eine Applikation mit einer Funktion und zwei zugeordneten Paremetern: `<OMA><OMS cd=Th1name=power"/><OMV name="g"/><OMV name=m"/></OMA>`.

3.2.4 MathML

MathML [Mat05b] ist ebenso wie OpenMath eine auf XML basierende Auszeichnungssprache zur Definition von Formeln. Sie wurde zum Datenaustausch zwischen technischen Systemen, zur Anzeige im WebBrowser oder als Datenstruktur für Computer-Algebra-Systeme entwickelt. Je nach Anwendungskontext unterstützt MathML zwei Verarbeitungsmodi, das *content encoding* und das *presentaition encoding*.

Das *presentaition encoding* beschreibt den syntaktischen Aufbau einer Formel und verwendet ca. 30 XML-Elemente. Hierbei wird jedes Element der Formel durch ein eigenes XML-Element angegeben, so dass Anzeigeprogramme die Semantik der Formel nicht interpretieren müssen, sondern lediglich die grafische Darstellung zeichnen müssen. Listing 3.6 zeigt ein einfaches Beispiel dieser Kodierungsvariante für die Formel $(a + b)^2$.

Listing 3.6: MathML-Beispiel einer Formel mit presentation encoding

```
<msup>
  <mfenced>
    <mi>a</mi>
    <mo>+</mo>
    <mi>b</mi>
  </mfenced>
  <mn>2</mn>
</msup>
```

Dieses Beispiel zeigt, dass es für Variablen und Operationen XML-Elemente gibt, die eine Zeichenfolge einschließen. die Variablen a und b werden durch das `<mi>`-Element eingeschlossen und der Operator + durch das Element `<mo>`. Das Klammernpaar wird durch das Element `<mfenced>` angegeben und die Zahl 2 wird durch das `<msup>`-Element hochgestellt.

Das bloße Anzeigen von Formeln reicht jedoch für die Verarbeitung in einem Computer-Algebra-System nicht aus. Hier werden semantische Informationen über eine Formel benötigt, um sie richtig interpretieren zu können. Das *content encoding* verwendet ca. 100 XML-Elemente – also deutlich mehr als das *presentation encoding* – und definiert den strukturellen Aufbau einer Formel. Listing 3.7 zeigt ein einfaches Beispiel dieser Kodierungsvariante für die Formel $(a + b)^2$.

Listing 3.7: MathML-Beispiel einer Formel mit content encoding

```
<apply>                                          1
  <power/>
  <apply>
    <plus/>                                      4
    <ci>a</ci>
    <ci>b</ci>
  </apply>                                       7
  <cn>2</cn>
</apply>
```

Dieses Beispiel zeigt, dass die Variablen a und b nicht durch einen Infix-Operator getrennt werden, sondern die Addition durch das `<apply>`-Element definiert wird. Der erste Parameter spezifiziert die Operation oder Funktion und die folgenden Parameter die Argumente. Ebenso wird die Potenz durch das `<apply>`-Element definiert.

3.2.5 Learning Material Markup Language (LMML)

LMML wird seit 1999 an der Universität Passau entwickelt und steht für *Learning Material Markup Language*. Zentrale Idee ist die Entwicklung eines allgemeinen Modells – dem *Passauer Teachware Modell* – zur Konstruktion von E-Learning-Inhalten, das je nach Bedarf an individuelle Ansprüche adaptiert und erweitert werden kann. LMML basiert auf XML (vgl. Abschnitt 3.2.1) und ist die konkrete Umsetzung des Teachware Modells und wird zuweilen auch als LMML-Framework bezeichnet. Im Gegensatz zu anderen auf XML basierenden Sprachen definiert LMML genau genommen eine erweiterbare Familie von XML-Sprachen, deren Elemente für spezielle Anwendungsdomänen, wie z.B Informatik oder Politikwissenschaften modifiziert werden können.

Das Passauer Teachware Metamodell [Freitag02b, Freitag02a] definiert den modularen Aufbau von Lerneinheiten. Die Architektur dieses Modells (vgl. Abbildung 3.7) sieht vier Modellierungsebenen vor: Die *reale Welt*, die *hypermedia Ebene*, das *Domänenmodell* sowie

das *abstrakte Modell* [Süß99]. Diese Ebenen sind übereinander angeordnet und beschreiben aufsteigend ein immer abstrakter werdendes Modell zur Spezifikation von Lernmodulen. Auf der untersten Ebene, der realen Welt, siedeln sich je nach Anwendungsdomäne unterschiedliche Themen zur Wissensvermittlung an. Diese sind auf der nächst höheren Ebene, der hypermedialen Ebene durch verlinkte Hypertexte, unter spezieller Berücksichtigung von Navigationskonzepten realisiert. Besonderes Interesse genießt die nächst höhere Ebene, das sogenannte Domänenmodell, mit dessen Hilfe eine domänenspezifische Modellierung der Hyperdokumente ermöglicht wird. Auf dieser Ebene werden für den Autor bzw. die Autorin vordefinierte *Content Objekte*, wie z. B. *Texte* oder *unsortierte Listen*, zur Konstruktion individueller Modelle bereitgestellt. Darüber hinaus werden fachspezifische Objekte, wie z. B. *Definitionen* oder *Theoreme* aus den vorgegebenen Objekten gebildet. Das abstrakte Modell vereinigt die Gemeinsamkeiten, denen die Gesamtheit der Domänenmodelle zugrunde liegt. Dieses schließt sowohl die abstrakte Modellierung von Materialien bestehend aus Content-Objekten, als auch die abstrakte Definition der Navigationsstruktur ein.

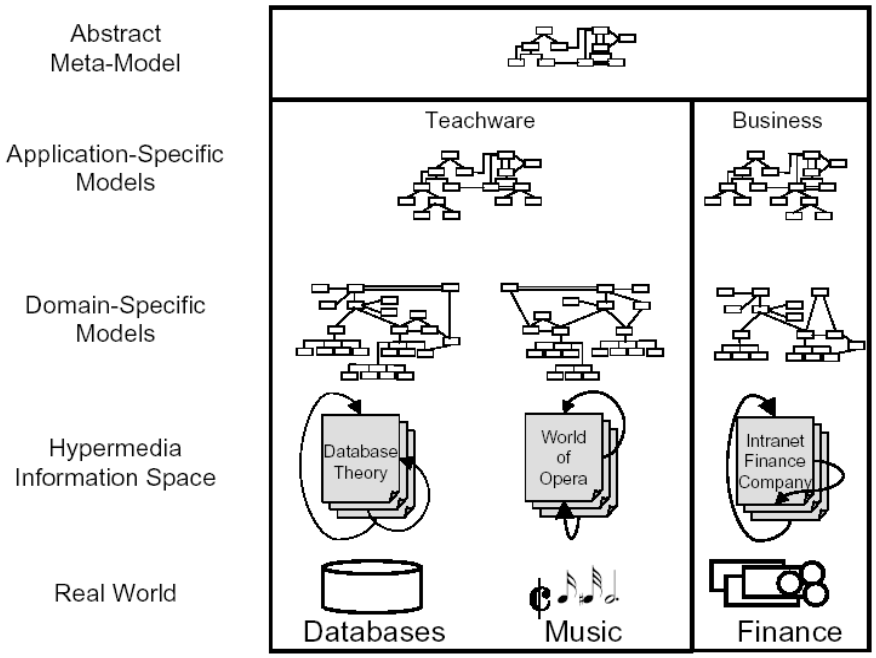


Abbildung 3.7: Teachware Modell

Das LMML-Rahmenwerk wird als XSD-Schema oder als DTD-Modul angeboten und ermöglicht die Spezifikation neuer Elemente durch die Erweiterung bestehender Modelle. Durch diesen Mechanismus wird beispielsweise eine Sprache für die Anwendungsdomäne Informatik erzeugt, die spezialisierte Elemente für Definitionen und Illustrationen charakterisiert. Der folgende LMML-Code zeigt einen Kurs zum Thema Datenbank-Theorie mit je zwei unterschiedlich schwierigen Definitionen für B-Trees:

Listing 3.8: Ein einfaches LMML-Beispiel

```
<?xml version="1,0" standalone="no"?>
<!DOCTYPE LearningMaterial ... >
<LearningMaterial>                                                    3
  <courseunit author="Hans Meier" date="99/04/10"
    discipline ="computer science" language="english" title="B-tree"
    topics="B-trees,database index,search key,records,space           6
    , efficiency ">
    <illustration  title ="Search Key" topics="search key,records"
      difficulty ="medium"/>                                          9
    <definition  title ="B-Tree" topics="B-Tree" difficulty="low">
      A B-tree is a multi level index.
      <ul>                                                            12
        <li>The file is organized as a balanced multipath tree
          width</li>
        <li>reorganization capabilities.</li>                         15
      </ul>
    </definition>
    <definition  title ="B-Tree" topics="B-Tree" difficulty="high">   18
      A B-tree of height h and fan out of 2k+1 is either empty or
      an ordered tree width the following properties:
      ...                                                             21
    </definition>
    ...
  </courseunit>                                                       24
</LearningMaterial>
```

3.2.6 Educational Modelling Language (EML)

Die Open University der Niederlande hat die *Educational Modelling Language (EML)* mit dem Ziel entwickelt, Lerneinheiten um ein umfangreiches pädagogisches Informationsmodell zu erweitern. Dieses soll die Wiederverwendbarkeit von Lerneinheiten sowie deren Zusammenwirken erheblich verbessern. Die Modellierung wird unter Einsatz der *Unified Modelling Language UML* durchgeführt, wobei die konkrete Umsetzung über XML-Schemata [Koper04, Koper01] erfolgt. Im Gegensatz zu der Content Package Spezifikation von IMS und ADL präsentiert EML eine pädagogische Komponente und kompensiert hiermit die Unzulänglichkeiten der bestehenden Standards. Die grundlegende Erstellung von Lernobjekten und die dazugehörende Anreicherung mit Metadaten, reicht für die pädagogische Sicht nicht aus. Stattdessen ist die Typisierung von Lernobjekten notwendig. Genau wie im objektorientierten Ansatz (OOA) gibt es unterschiedliche Klassen von Lernobjekten, wie der zu vermittelnde Lehrstoff oder zu lösende Aufgaben.

Abbildung 3.8 zeigt die XML-Schema-Bindung des EML-Rahmenwerkes (aus [Koper04], Seite 8). Ausgangspunkt ist die *unit of study*, die eine Lerneinheit aus verschiedenen Granularitätsstufen zusammensetzt [Koper04, Pawlowski02]. Neben *Metada-*

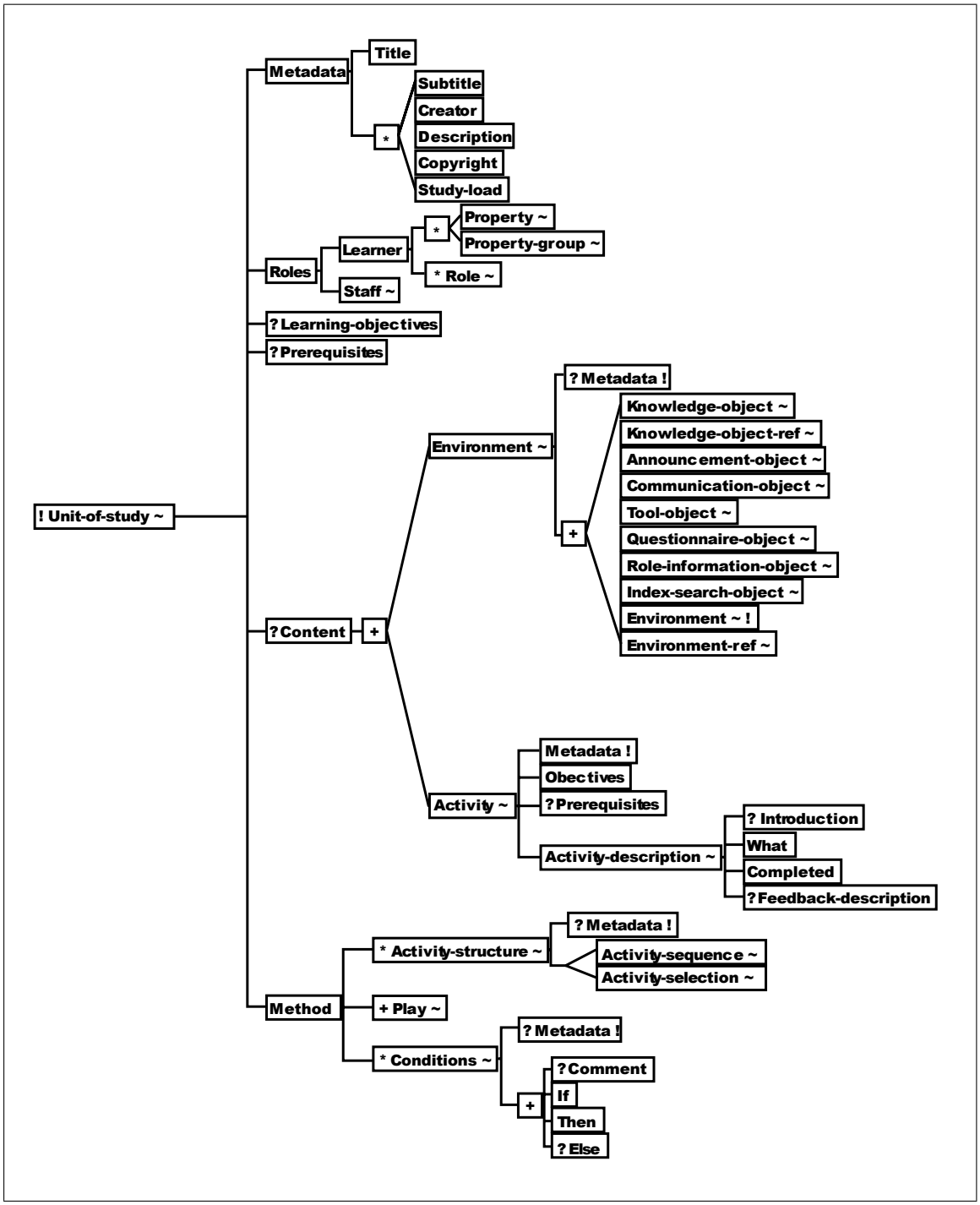


Abbildung 3.8: XML-Schema Bindung des EML Rahmenwerkes

ten muss auch die jeweilige *Rolle* des Benutzers bzw. der Benutzerin angegeben werden. Dieses können *Lernende* sein, die sich in beliebig viele Untergruppen aufteilen lassen, oder *Betreuende*, die den Lernfortschritt verfolgen. Zudem können das *Lernziel* sowie *Voraussetzungen*, wie z. B. der notwendige Wissensstand, optimal angegeben werden. Anhand

dieser *Methode* wird der Ablauf des Lernprozesses spezifiziert. Hierzu gehört beispielsweise die Zurodnung der *Lernobjekte* zu den Rollen. Im *Content*-Bereich wird der Inhalt durch Objekte definiert, wie z. B. Verweise auf externe Dateien oder intern formulierte Inhalte, die aus einer Mischung von DocBook- und HTML-Elementen bestehen. Zudem lassen sich zahlreiche Lernaktivitäten definieren, anhand derer Studierende aktiv auf die Reihenfolge des Inhalts eingehen können.

Die durch das EML-Informationsmodell erziehlten Erkenntnisse sind zu einem relevanten Teil in die *IMS Learning Design Information Model Spcification* [IMS03c] eingeflossen.

3.2.7 Analyse

Die in diesem Abschnitt vorgestellten Markupsprachen zur formalen Beschreibung von Inhalten sollen in Hinblick auf die eingeführten Bewertungskriterien untersucht und ausgewertet werden. Zu diesem Zweck werden die in dieser Arbeit geforderten Kriterien *Granularität* und *Modularität* in der anstehenden Analyse reflektiert und bewertet.

DocBook ist eine Auszeichnungssprache zum Beschreiben von Büchern und Artikeln. Für die Beschreibung von Lerninhalten ist sie jedoch nur bedingt einsetzbar, weil es vorwiegend für die Erzeugung ganzer Bücher entwickelt wurde und sich der Autor bzw. die Autorin bei der Dokumenterstellung immer auf das ganze Dokument bezieht. Zudem besitzt die DocBook DTD ca. 400 Elemente, die derart orthogonal entworfen sind, dass sie dem Anwender bzw. der Anwenderin eine möglichst flexible Kombinationsvielfalt bieten, was die Entwicklung eines Parsers erheblich erschwert. Hinzu kommt die Unterstützung des Metadatenstandards *Dublin Core*, der einzelnen Textteilen direkt zugewiesen werden kann. Für das in dieser Arbeit verfolgte Ziel ist jedoch die vom Dokument gelöste Metadatendefinition die bessere Lösung. Erstens, weil E-Learning-Metadaten berücksichtigt werden müssen und zweitens, weil die Metadaten in weiteren Verarbeitungsschritten durch eine Softwareschnittstelle vom Dokument getrennt werden müssen.

OMDoc ist ein sehr mächtiges Modell zur strukturellen Abbildung mathematischer Dokumente. Es eignet sich vor allem für den Datenaustausch zwischen Wissenssystemen wie Computer Algebrasystemen, die spezielle Relationen zwischen mathematischen Elementen auswerten können. Hinzu kommen pädagogische Elemente, die ein bestimmtes Lernszenario festlegen. OMDoc ist kein allgemeines Strukturmodell für Inhalte, das darüber hinaus eine derart komplexe Struktur besitzt, dass die Semantik nicht ohne einen gewissen Einarbeitungsaufwand seitens der Autoren und Autorinnen zu erkennen ist. OMDoc setzt zwar auf Standards wie OpenMath für die Formelrepräsentation, unterstützt jedoch keinen allgemeinen Mechanismus zur Modularisierung, wie es das IMS Content Packaging anbietet. Die logische Strukturierung, die sich auf Definitionen und Beweise bezieht, führt zu einer äußerst feingranularen Struktur. Außerdem werden Metadaten nicht vom Dokument getrennt, sondern den jeweiligen Elementen direkt zugewiesen.

Das Datenmodell der Markupsprache LMML ermöglicht die domänenspezifische Erweiterung mit einer grundlegenden Sprachsyntax zur Formulierung spezialisierter Lerninhalte. Dieses ist dann sinnvoll, wenn für einen bestimmten Anwendungsbereich spezialisierte Strukturen benötigt werden. Ferner ermöglicht das Modell die adaptive Erweiterung von

Materialien. Das bedeutet, dass ein für ihn oder sie angepasster Schwierigkeitsgrad, abhängig vom Wissensstand des Lernenden durch den Lehrenden eingestellt werden kann. Leider vermischt LMML ebenso wie OMDoc die Metadaten mit dem Inhalt und es werden keine E-Learning-Standards unterstützt. Das Prinzip der Lernobjekte bzw. der Aspekt der Modularität wird in diesem Konzept nicht weiter berücksichtigt.

EML ist ein umfangreiches Modell zur Gestaltung von Lerninhalten. Der Fokus zielt in diesem Projekt eindeutig auf die fehlenden pädagogischen Regeln gegenwärtiger Standards. Diese werden um das sogenannte Informationsmodell erweitert, was letztendlich die Grundlage für die IMS Learning Design Information Model Spezifikation gebildet hat.

Die in diesem Abschnitt eingeführten Strukturmodelle beschreiben zwar grundlegend die vom Layout getrennten Inhalte, haben jedoch ihren Fokus eindeutig auf die Implementierung pädagogischer Konzepte ausgerichtet. Kein Ansatz unterstützt die modulare Konstruktion orthogonaler Kurseinheiten, die beliebig miteinander kombiniert werden können.

3.3 Lernplattformen – Learning Management Systeme

Lernplattformen bzw. Learning Management Systeme (LMS) sind technische Systeme, die das institutionelle Lernen überhaupt erst ermöglichen. Vor allem Hochschulen, aber auch Betriebe, setzen zunehmend auf Plattformen, mit denen sie einerseits Lehrinhalte Studierenden und Mitarbeitern anbieten können und andererseits deren Lernfortschritt verfolgen und auswerten können. In diesem Sinn liegt die Hauptaufgabe eines LMS nicht in der Unterstützung der Konzeption und Entwicklung von Lehrinhalten, sondern vielmehr in der Verwaltung von Lernobjekten, Ressourcen und deren Anwendern bzw. Anwenderinnen. Verfügt eine Hochschule über ein LMS wie Loncapa [LON05], ILIAS [ILI04], WebCT [WEB05] oder Blackboard [Bla03], so können Dozenten sämtliche Unterlagen wie Skritpe, Übungen, multimediale Inhalte, etc., Studierenden unabhängig von Ort und Zeit zugänglich machen. Studierende erhalten ein persönliches Benutzerkonto, das nicht nur den Benutzerkreis für einen bestimmten Kurs einschränkt, sondern auch die Auswertung des Lernfortschritts eines jeden Studierenden ermöglicht. Ferner ist es für Studierende möglich, über unterschiedliche Kommunikationskanäle, wie Diskussionsforen, Chats oder E-Mail, mit anderen Kommilitonen und wissenschaftlichen Mitarbeitern Probleme zu diskutieren und Erfahrungen auszutauschen. Baumgartner hat die Definition einer webbasierten Lernplattform wie folgt definiert:

> "Unter einer webbasierten Lernplattform ist eine serverseitig installierte Software zu verstehen, die beliebige Lehrinhalte über das Internet zu vermitteln hilft und die Organisation der dabei notwendigen Lernprozesse unterstützt ([Baumgartner02], Seite 14)."

Der Schwerpunkt dieser Definition liegt bei der Unterstützung des Lernprozesses, wie z. B. Lernflussmanagement. Damit werden einfache Content Management Systeme (CMS), die zur reinen Erzeugung von digitalen Inhalten dienen, systematisch ausgeschlossen.

Es gibt zahlreiche Anbieter von Lernplattformen mit unterschiedlichen Schwerpunkten und Ausprägungen. In [Schulmeister03] werden im Rahmen einer Studie zur Evaluation von Lernplattformen Bewertungskriterien festgelegt, anhand derer sich die Vielzahl unterschiedlicher Systeme bewerten lässt. Dieses lässt bereits darauf schließen, dass es keine glasklare Definition für ein LMS gibt, sondern vielmehr einen Anforderungskatalog der beschreibt, welche Funktionen ein LMS besitzen sollte. Hierfür werden die folgenden fünf Kategorien festgelegt:

- Benutzerverwaltung,
- Kursverwaltung,
- Rollen und Rechte,
- Kommunikationsmethoden und Werkzeuge,
- Darstellung von Kursinhalten, Lernobjekten und Medien.

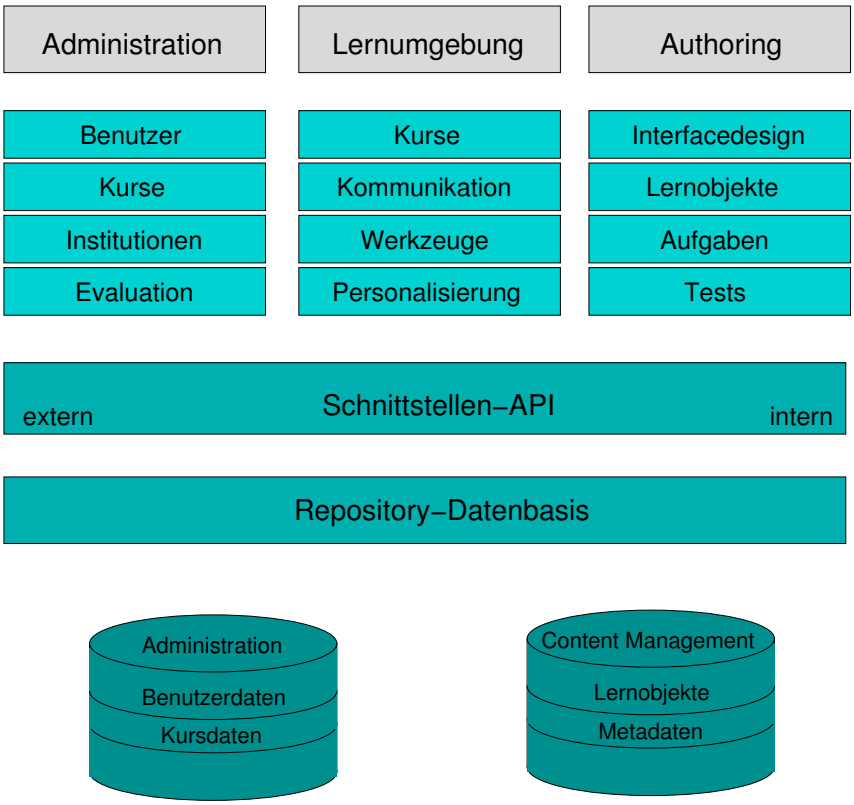


Abbildung 3.9: Idealtypische Architektur eines Learning Mangement Systems

Nach Schulmeister [Schulmeister03] besteht ein idealtypisches LMS aus genau drei Schichten (vgl. Abbildung 3.9). In der untersten Schicht, der *Datenbankschicht*, werden

Lernobjekte und die dazugehörigen Metadaten gespeichert. Dieses ermöglicht der darüberliegenden Schicht den Zugriff auf singuläre Lernobjekte und sichert deren konsistente Verwaltung.

Zusätzlich werden administrative Daten wie Benutzerobjekte und Kursinformationen gespeichert. Damit die Datenbank nicht nur vom LMS benutzt werden kann, bietet die *Schnittstellen-API* sowohl interne als auch externe Zugriffsmöglichkeiten an, durch die externe Applikationen Datenbestände auswerten können. Die oberste Schicht spaltet sich in die drei Säulen *Administration*, *Lernumgebung* und *Authoring* auf und bildet die Schnittstelle zum Anwender bzw. zur Anwenderin. Abhängig von der Rolle, mit der sich angemeldet wurde, können unterschiedliche Aspekte des LMS genutzt werden. Mit der Rolle *Administration* lassen sich neue Benutzer oder Kurse erzeugen. Der Bereich *Lernumgebung* bietet Lernenden den Zugang zu Lernobjekten und öffnet darüber hinaus Kommunikationswege zu anderen Anwendern und Anwenderinnen. Die dritte Säule repräsentiert den Bereich für kreative Aufgaben, wie z. B. das Interface-Design oder die Entwicklung neuer Lernobjekte.

3.4 Autorenwerkzeuge

Der Abschnitt über Learning Management Systeme hat gezeigt, wie Lehrinhalte Studierenden angeboten werden und wie Statistiken bzw. Analysen über den Lernerfolg eines Studierenden erstellt werden. Für die Content-Entwicklung sind diese Systeme jedoch nur unzureichend geeignet. Einige Plattformen, wie z.B das LMS Ilias, bieten eine Autorenumgebung an, die die direkte Eingabe von HTML-Seiten erlaubt. Für die Erstellung von Inhalten existieren so genannte Autorenwerkzeuge, mit deren Hilfe es möglich ist, auf einfache Weise multimediale Inhalte zu erzeugen. Bei näherer Betrachtung existierender Inhalte ist eine allgemeine Definition dieses Begriffs jedoch nicht möglich, weil es unterschiedliche Anwendungsfälle für den Einsatz von Autorensystemen gibt. Zwei Beispiele sollen den Unterschied verdeutlichen: Das erste Szenario beschreibt einen Schullehrer, der bestimmte Inhalte für seine Schüler und Schülerinnen multimedial aufbereitet hat und sie in Form von interaktiven Animationen und Videos präsentiert. Anschließend findet eine *Multiple Choice*-Auswertung am Rechner statt. Diesem behavioristischen Ansatz folgt das eher konstuktivistische (vgl. Kapitel 2.2) Beispiel eines Physikstudenten, der neben ausführlichen und hochqualitativen Texten auch virtuelle Experimente durchführt. Aus diesem Beispiel geht eindeutig hervor, dass es verschiedene Betrachtungsweisen über den zu vermittelnden Inhalt gibt. Fünfstück [Fünfstück00] spricht in beiden Fällen von multimedialen Anwendungen, teilt sie jedoch in zwei Gruppen auf: den *dokumentenbestimmten Anwendungen* und den *programmbestimmten Anwendungen*. Die bei der dokumentenbestimmten Anwendung formulierte Semantik zeichnet sich durch die zugrundeliegende Dokumentenbeschreibungssprache HTML oder XML [Farinetti00] (vgl. Abschnitt 3.2.1) aus. Interaktionsobjekte gehören nicht zwangsläufig zum Dokument und werden zumeist extern gespeichert. Bei programmbestimmten Anwendungen wird die Semantik und das zugehörige Anwendungslayout durch eine Programmiersprache festgelegt. Multimediale Daten sind

entweder elementarer Bestandteil der Anwendung und werden als Datenstruktur gespeichert oder werden in Dateien abgelegt. Fünfstück beschreibt Multimediale Autorensysteme wie folgt:

> "Autorensysteme ermöglichen die Entwicklung multimedialer Anwendungen in einer grafisch-interaktiven Form unter weitgehendem Verzicht auf die Programmierung, verwenden hierzu visuelle interface development tools sowie Werkzeuge zur Erstellung und Bearbeitung von Medien. Sie gehen vom Entwurfsprozeß der Benutzungsoberfläche aus und unterstützen die anderen Phasen des Entwicklungsprozesses häufig nur unvollständig. Die Vorgehensweisen sind weniger an herkömmlichen Programmiertechniken ausgerichtet, sondern beruhen auf einprägsamen Metaphern und einem leichten Zugang der an Gestaltung und Inhalten orientierten Entwickler (aus [Fünfstück00], Seite 16)."

Eine andere Form der Klassifizierung von Autorensystemen richtet sich nach dem notwendigen Einarbeitungsaufwand für den Benutzer bzw. für die Benutzerin. Abbildung 3.10 (aus [Heafele05], Seite 2, [Niegemann03], Seite 266) zeigt fünf verschiedene Klassen von Autorenwerkzeugen, die im Bezug auf Komplexität und Benutzbarkeit von links nach rechts sortiert sind.

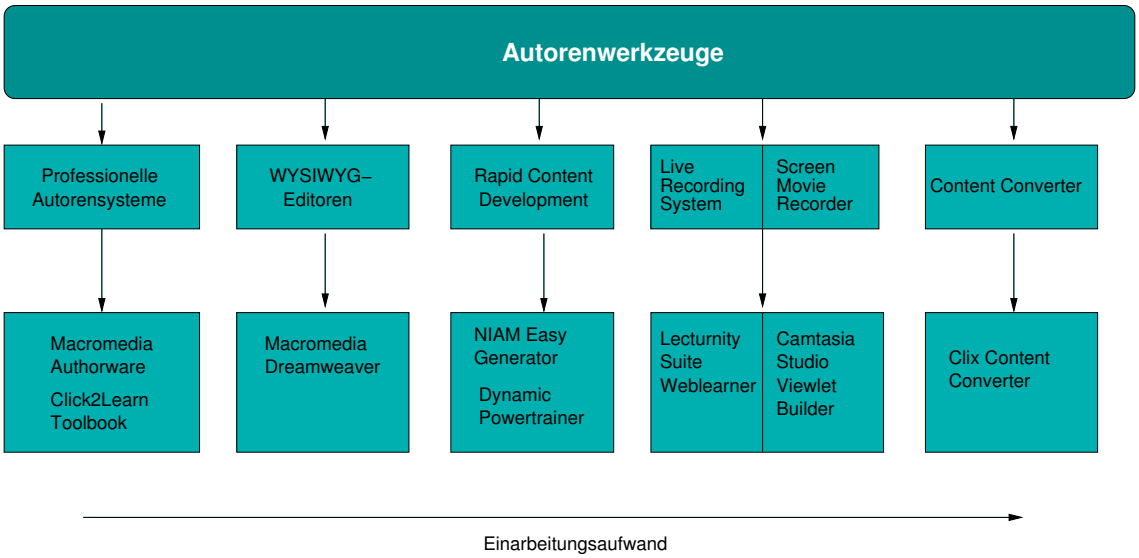


Abbildung 3.10: Verschiedene Kategorien von Autorenwerkzeugen

Zu der Gruppe professioneller Autorensysteme gehören *Macromedia Authorware* [Mac05] und *Click2Learns Toolbook*. Diese Autorensysteme zählen zu den programmbestimmten Anwendungen, weil sie als Resultat Lehrinhalte, basierend auf interaktiven Videos produzieren, die keine eindeutige Dokumentenstruktur besitzen. Die zweite Gruppe umfasst HTML-Editoren, welche überwiegend für die Produktion von WBT-Inhalten eingesetzt werden. Neben diesen recht kompliziert zu nutzenden Werkzeugen, kann jedoch auf einfache Alternativen zurückgegriffen werden. Die Gruppe der Autorenwerkzeuge, die

zu schnell entwickelten Inhalten führen, wird in dieser Darstellung mit *Rapid Content Development* bezeichnet. Darüber hinaus werden auch Aufnahmesysteme für Videoproduktionen und Bildschirmaufzeichnungswerkzeuge zum Erstellen von E-Learning-Inhalten eingesetzt, die in der Regel Zusatzwerkzeuge für den Erstellungsprozess sind. Die letzte Gruppe der Autorenwerkzeuge beschreibt sämtliche Konvertierungswerkzeuge, die z. B. zur Umwandlung spezieller Dateiformate herangezogen werden.

3.4.1 Ausgewählte Systeme

Die Auswahl eines Autorensystems richtet sich subjektiv nach der Einsatzdomäne des zu erstellenden Materials. Im Schulbereich und im Ausbildungssektor größerer Firmen werden vorzugsweise multimediale Applikationen (MMA) [Fünfstück00] mit zahlreichen Videos und Quizumgebungen benötigt. Im Hochschulbereich hingegen werden eher Dokumentbezogene Inhalte eingesetzt, die oft als Ergänzung zu vorhandenen Lehrbüchern und ausgedruckten Skripten verstanden werden. Aus diesem Grund wird nachfolgend eine Auswahl kommerzieller sowie wissenschaftlicher Systeme der Gruppen *programmbestimmte* und *dokumentbestimmte Autorensysteme* verglichen und in Relation zu der in dieser Arbeit aufgeworfenen Forschungsfrage ausgewertet.

Programmbestimmte Autorensysteme

Abbildung 3.11: Toolbook

Toolbook[1] wartet als erster Vertreter dieser Gruppe mit einer recht interaktiven Schnittstelle zum Benutzer auf. Ziel ist es, multimediale Inhalte durch technisch nicht versierte Personen zu erstellen, die über das Internet Lernenden angeboten werden können. In Abhängigkeit des präferierten Lernszenarios, lassen sich unterschiedliche Strukturen für einen Kurs in Form von Templates auswählen, die Kurseigenschaften wie Navigationen, Inhaltsverzeichnisse oder Tests bestimmen. Jeder Kurs besteht aus einer linearen Folge von Seiten, die durch den Autor bzw. durch die Autorin frei gestaltet werden. Toolbook bietet eine Reihe vordefinierter Objekte an, wie z. B. Tests oder Videos, die auf Seiten positioniert werden. Kurse können als HTML-Seiten exportiert werden und unterstützen die AICC und ADL-SCORM Standards.

Mit *Authorware* [2] lassen sich Applikationen und Quizumgebungen erstellen. Diese basieren auf *Macromedia Flash* und werden über das Internet angeboten. Die Kursstruktur wird bei *Authoreware* mit einem Zeichenbrett konstruiert und ermöglicht so einen intuitiven Kursaufbau. Um während der Bearbeitung zu einem speziellen Thema zu gelangen, kann es außerdem als Navigationshilfe für die Entwicklung herangezogen werden. Die eigentlichen Quizzes und Applikationen basieren auf *Macromedia Flash* und werden durch die Anordnung multimedialer Objekte sowie Objekte für die Ablaufsteuerung frei gestaltet. Als Resultat wird ein interaktiv zu steuernder Film generiert. Authorware unterstützt ADL-SCROM und bietet für die Generierung entsprechende Werkzeuge an.

Ein weiterer Vertreter der programmbestimmten Autorenwerkzeuge ist *DazzlerMax*[3]. Genau wie mit seinen Kontrahenten *Authorware* und *Toolbook* können Multimedia Präsentationen auf einfache Weise erstellt und bearbeitet werden. Jeder Kurs wird in einem Strukturfenster angezeigt und lässt sich spielend per Drag'n'Drop durch multimeidale Objekte erweitern. Kurse bestehen aus Sequenzen von *Aufgaben*, die sich jeweils in Aktionen und Antworten gliedern. Aktionen sind *Abbildungen*, *Klänge*, *Transitionen* oder *Texte* und für den Aufbau einer Szene verantwortlich. Antworten, die ebenso zu einer Aufgabe gehören, beschreiben das Verhalten des Lernsystems, ausgelöst durch das interaktive Einwirken des Benutzers bzw. der Benutzerin.

Dokumentbestimmte Autorensysteme

Als erster Vertreter der dokumentbestimmten Autorensysteme wird *Macromedia Dreamweaver MX*[4] vorgestellt. *Dreamweaver* wurde als klassischer HTML-WYSIWYG Editor entwickelt, dem eine umfangreiche Erweiterung für multimediale Inhalte beigesteuert wurde. Neben den üblichen Funktionen eines HTML-Editors, (Einfügen von Hyperlinks, Grafiken oder Framesets) unterstützt *Dreamweaver* ebenso das Erstellen von dynamisierten Seiten wie z. B. Active-Server-Pages (ASP). Zudem können Flashfilme in bestehende Seiten eingebunden werden, was durch das gemeinsame Ausliefern mit *Macromedia Authorware* eine gute Kombination einer dokumentbasierten und programmbasierten Entwicklungs-

[1] Quelle: http://www.sumtotalsystems.com/ (29.10.2005)
[2] http://www.macromedia.com/software/authorware/ (29.10.2005)
[3] http://www.dazzler.net/ (29.10.2005)
[4] http://www.macromedia.com/ (29.10.2005)

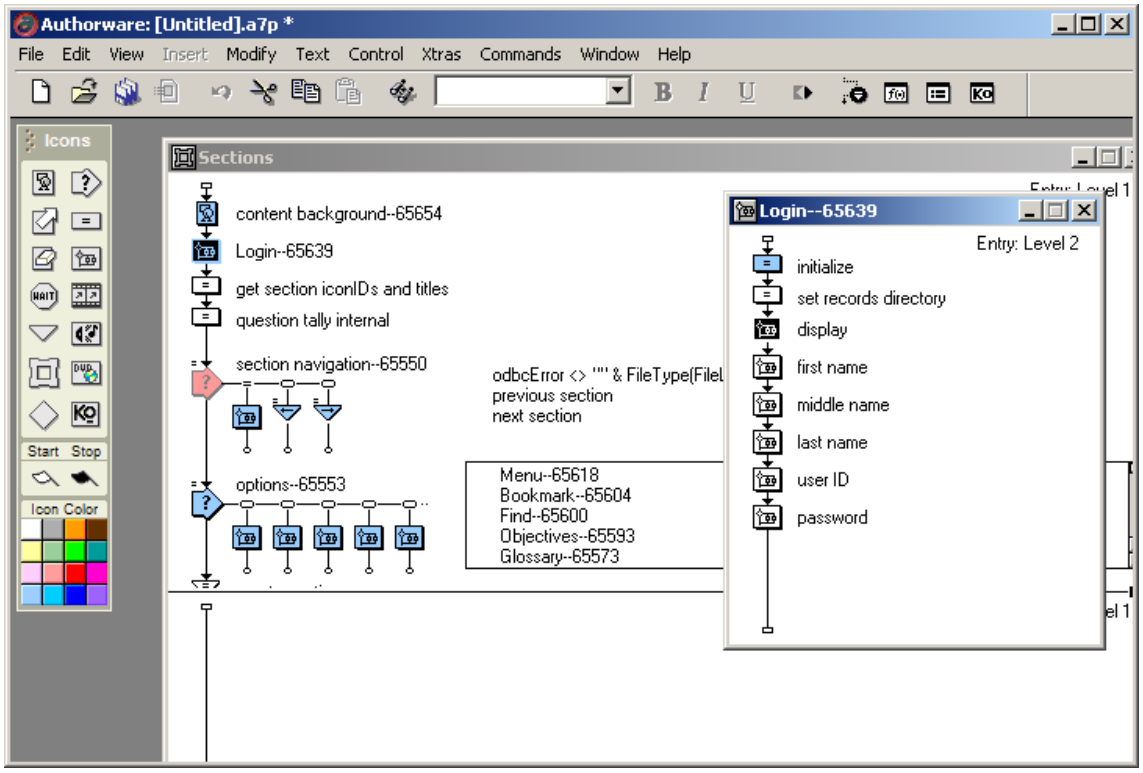


Abbildung 3.12: Authorware

umgebung ergibt. Mit *Macromedia MX* lassen sich auch Lernobjekte (vgl. Abschnitt 2.4) erzeugen. In Cuthbert [Cuthbert02] wird detailliert beschrieben, wie die Erzeugung von Lernobjekten mit Dreamweaver stattfindet. Hierbei ist die Curriculum-Entwicklung ein mehrstufiger Prozess eines Teams, bestehend aus *Curiculum Designer*, *Flash Specialist*, *Interface Designer* und *Database Specialist*. Je nach Kompetenz fallen verschieden Aufgaben, wie z. B. die Assetentwicklung oder die Gestaltung eines Designs, das auf der eben vorgestellten Rolle beruht. Abbildung 3.4.1 zeigt die Designansicht von Macromedia Dreamweaver.

Microsoft hat mit dem *LRN-Toolkit* [5] ein Werkzeug entwickelt, das die Erstellung von IMS kompatiblen Paketen ermöglicht. Hierzu wird auf der linken Seite der Programmansicht das Manifest des Dokuments als Baumstruktur angezeigt, das interaktiv mit Assets verknüpft werden kann. Hierbei können auch Untermanifeste angelegt und von einem Item eines auf höherer Ebene definierten Manifests referenziert werden. LRN unterstützt sowohl das kopierende Einbinden von Assets sowie das Einfügen referenzierter Assets, die über eine URL eingebunden oder an einer festgelegten Stelle im Dateisystem abgelegt werden. Der exportierte Kurs wird dann mit Hilfe des Internet-Explorers angezeigt, wobei ein über Stylesheets einstellbarer Rahmen generiert wird. Das LRN-Toolkit ist also ein Werkzeug zum Zusammenschnüren von IMS kompatiblen Paketen mit dem Ziel, sie reibungslos in ein LMS zu integrieren.

[5] http://www.microsoft.com/elearn/ (29.10.2005)

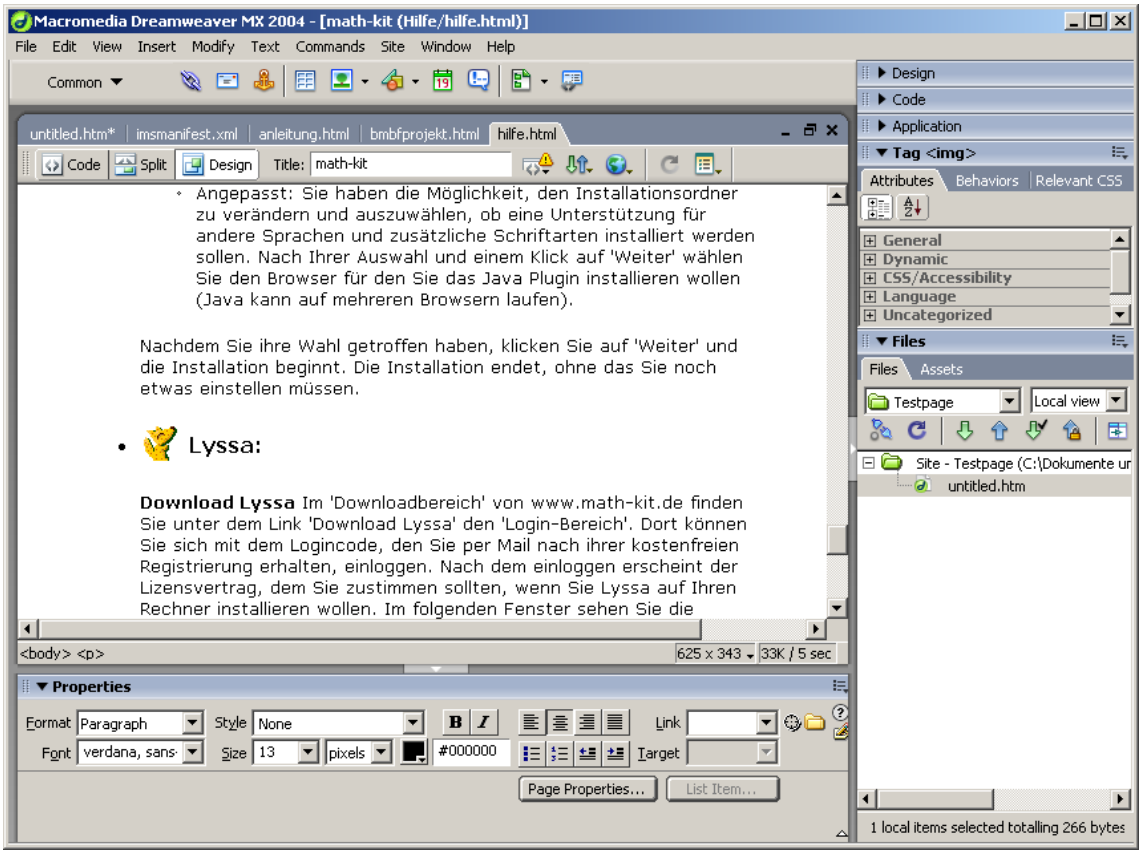


Abbildung 3.13: Macromedia Dreamweaver MX

Das Passauer Knowledge Management System (PaKMaS) [Süß00] ist zugleich ein Autorensystem und eine Benutzerumgebung, die auf der bereits vorgestellten Auszeichnungssprache *Learning Material Markup Language (LMML)* (vgl. Abschnitt 3.2.5) basiert. Es unterstützt die Erstellung von Lern- und Lehrmaterialen bestehend aus einer Menge untereinander vernetzter Wissensmodule. Das System kombiniert ein XML-Content Management System mit einem adaptiven Lernsystem. Ein wesentlicher Kernpunkt ist die individuelle Adaptierbarkeit von Wissensmodulen durch den Nutzer bzw. die Nutzerin für einen speziellen Lernzweck. Zudem werden auch Funktionalitäten von Learning Management Systemen (LMS) bereitgestellt (vgl. Abschnitt 3.3), wie z. B. die für ein LMS geforderten Kommunikationsmethoden E-mail, News und Chat. Ein wichtiges Merkmal dieser Architektur ist die Trennung zwischen Inhalt und Visualisierung. Hierbei werden Inhalte durch XSLT 3.2.1 in verschiedene Ausgabeformate übersetzt. Ein weiterer wichtiger Beitrag ist die Vernetzung der Wissenseinheiten und die daraus resultierende Orientierung für Dozenten und Lernende innerhalb des Systems. Lernpfade werden durch Dozenten vorgeschlagen, können jedoch durch Lernende dynamisch modifiziert und an ihre Bedürfnisse angepasst werden. Neben diesen sogenannten *geführten Touren* werden darüber hinaus auch herkömmliche Strukturierungen nach Kapiteln und Abschnitten angeboten.

3.4.2 Analyse

Die in diesem Abschnitt vorgestellten Autorensysteme geben gewiss keinen vollständigen Überblick über alle gängigen Systeme, die zurzeit in der Wirtschaft verbreitet sind oder an denen wissenschaftlich entwickelt wird. Allerdings werden die weitgehenden Einsatzmöglichkeiten und der damit zusammenhängende Bedarf anschaulich demonstriert.

Die erste Kategorie, die Gruppe der programmgesteuerten Systeme, ermöglicht vor allem die Entwicklung interaktiver Animationen und Sequenzen, die durch den Anwender bzw. durch die Anwenderin frei aufgebaut werden und im Regelfall ein individuelles Programm liefern, ähnlich wie bei der Erstellung eines Films. *Authorware* erzeugt z. B. einen Flash-Film, der mit Hilfe des Macromedia Flash-Plugins für nahezu jeden Browser anzeigbar ist. Die entstehenden interaktiven Videos werden durch den Benutzer bzw. durch die Benutzerin interaktiv gesteuert, wodurch ein chronologischer Ablauf entsteht, der u.a. die Visualisierung von 2-D Transitionen auf Objekten ermöglicht. Die strukturelle Kursdefinition oder sogar die formale Auszeichnung von Inhalten ist hierbei jedoch nicht gewünscht und wird auch nicht unterstützt. Diese Art von Autorensystemen sieht es grundsätzlich nicht vor, modulare Kurse zu erzeugen, die sich frei kombinieren lassen und somit eine autorenübergreifende Entwicklung ermöglichen. Auch das Einbeziehen bestehender Kursmaterialien ist in diesen Systemen nicht vorgesehen, wenn von einer Microsoft-OLE-Integration abgesehen wird. Interessanterweise unterstützen diese Werkzeuge allesamt den SCORM 1.2 Standard und scheinen dadurch strategisch am Markt positioniert zu sein. Bei näherer Betrachtung zeigt sich jedoch, was bereits aus der bloßen Strukturierung der Kurse hervorgeht: es werden lediglich die IMS-Ressourcen mit den eingebundenen Videos und Flash-Filmen versehen. Aufgrund der faktisch nicht vorhandenen Kursstruktur kann sie beim Exportieren naturgemäß nicht erzeugt werden, obwohl das Resultat in Verbindung mit Learning Management Systemen (LMS), die bemüht sind aus den SCORM-Paketen eine dem Kurs entsprechende Struktur zur Navigation aufzubauen, zwar kompatible ist, aber keineswegs im Sinn der Standards ist.

Um Antworten für die in dieser Arbeit aufgeworfenen Fragestellungen zu finden, liefert eine Betrachtung der zweiten Kategorie, der Gruppe der dokumentgetriebenen Systeme, erheblich mehr Aufschluss. Im Gegensatz zu den programmgetriebenen Systemen werden in dieser Gruppe Dokumente anstatt Videos und Animationen durch das Autorensystem produziert. Der große Unterschied beider Gruppen liegt darin, dass es keine chronologische Abhängigkeit einzelner Lernsequenzen gibt und Inhalte in der Regel eine innere Struktur aufweisen. Dieses wird bei *Macromedia Dreamweaver* ersichtlich. Der HTML-Editor mit *multimedia extension (MX)* ermöglicht die Erstellung dokumentbasierter Inhalte. Leider wird hierbei die Visualisierung nicht vom Inhalt getrennt, was das kooperative Arbeiten oder zumindest die substanzielle Wiederverwendung nicht möglich macht. Zwar können Stylesheets ausgetauscht werden, eine absolute formatunabhängige Repräsentation und eine entsprechende Modularisierung wird jedoch nicht angeboten. Zwar wird die Curriculum-Entwicklung von Teams verfolgt, allerdings handelt es sich hierbei um ein Team mit disjunkten Aufgabengebieten, wie z. B. die Layoutentwicklung oder das Entwickeln von Flash-Quizzes. Die Entwickler und Entwicklerinnen kommunizieren

über definierte Schnittstellen und kommen sich fachlich, im Gegensatz zu dem in dieser Arbeit verfolgten Ziel der gemeinsamen Entwicklung, nicht nahe.

Im Gegensatz zu diesem Ansatz konzentriert sich das LRN-Toolkit lediglich auf die Struktur von Lerninhalten. Der auf der IMS Content Package Spezifikation basierende Inhalt setzt dabei kein bestimmtes Format voraus, sondern erlaubt das Zusammenstellen beliebiger Inhalte. Das Werkzeug unterstützt zwar die modulare Kursentwicklung, verbietet jedoch die konsistente Kombination von Kursteilen verschiedener Autoren und das Zusammensetzen zu einem heterogenen Kurs. Das Passauer Knowledge Management System stellt zugleich ein Autorenwerkzeug und eine Lernumgebung für Studierende bereit. Der besondere Fokus dieses Systems liegt auf den adaptiven Wissensmodulen, die nicht nur von Lehrenden sondern auch von Studierenden in Abhängigkeit von ihrer Qualifikation und ihrem Wissensstand angepasst werden können und auf diese Weise individuelle Lernpfade generieren. Obwohl dieses auf wissenschaftlichen Arbeiten beruhende System, aufgrund seiner stark auf adaptive Lernsysteme ausgerichteten Funktionalität, ein anderes Ziel verfolgt, wurden in Bezug auf die domänenspezifische und formale Datenmodellierung interessante Erkenntnisse gewonnen, die in dieser Arbeit berücksichtigt werden müssen.

Die Ergebnisse sind in Tabelle 3.2 noch einmal übersichtlich zusammengefasst.

	System	Formatierbarkeit	Modularität	Granularität	Nachhaltigkeit	Wiederverwendbarkeit	Kooperation
Programmbestimmt	Toolbook	Freie Layoutgestaltung im Powerpoint Stil	Keine modulare Kursstruktur	Erstellung kompletter Inhalte	AICC, SCORM 1.2 (nur Ressourcen)	Inkompatible zu Dokumenten, Unterstützt aber OLE-Objekte	Kooperative Konstruktion wird nicht unterstützt
	Macromedia Authorware	Unterstützt die Erstellung von interaktiven Flash-Filmen für Quiz-Umgebungen, jedoch nicht die formale Beschreibung von Inhalten	Keine modulare Verarbeitung von Inhalten	Erstellung kompletter Inhalte	Standardkompatibel durch SCORM 1.2	Wiederverwendung bestehender Inhalte wird nicht unterstützt	Kooperative Konstruktion wird nicht unterstützt
	DazzlerMax	Strukturierung durch Platzierung von Aktionen und Antworten	Nur Entwicklung kompletter Kurse	Granularität auf Aufgaben-Ebene	IMS und SCORM 1.2 kompatibel	Bestehende Inhalte können nicht eingebunden werden	Kooperation nicht möglich
Dokumentbestimmt	Dreamweaver MX	Unterstützt die Erstellung von Webseiten und interaktiven Flash-Filmen jedoch nicht die Trennung zwischen Inhalt und Darstellung	Templatemechanismus unterstützt Entwicklerteams	Erstellung kompletter Curricula	Standardkompatibel durch SCORM 1.2	Wiederverwendung bestehender Inhalte wird nicht unterstützt	Teamarbeit wird nur durch Arbeitszuweisung unterstützt
	Microsoft LRN	Das LRN Toolkit ist ein Werkzeug zum Erstellen von Content Packages mit beliebigen Formaten. Hierdurch wird nur die Struktur festgelegt und nicht die Inhalte	Modulare Aggregation von Assets, HTML-Seiten und Applikationen	Nicht eingeschränkt	IMS Content Package kompatibel	Aggregation beliebiger Formate	Nicht unterstützt
	Passauer Knowledge Management System	Unterstützt semantische Auszeichnungen mit LMML	Speichert Wissensmodule in Datenbanken	Frei wählbare Granularität	Standards werden nicht unterstützt	Integration verschiedener Dokumenttypen auch anderer Autoren	Kooperative Konstruktion wird nicht unterstützt

Tabelle 3.2: Gegenüberstellung der analysierten Systeme

Kapitel 4

Bewertung

In diesem Abschnitt findet die abschließende und zusammenfassende Bewertung aktueller wissenschaftlicher Arbeiten und Technologien statt.

Der Abschnitt Lernparadigmen hat gezeigt, dass es drei grundlegende Paradigmen (vgl. Abschnitt 2.2.2) zur Wissensvermittlung gibt. Obwohl eine Reihe von Lernprogrammen existieren, die gerade auf eine Wissensvermittlung, basierend auf dem behavioristischen Paradigma anhand von Multiple-Choice-Tests setzen, wird in dieser Arbeit ein generellerer Ansatz präferiert, der Autoren und Autorinnen keine Einschränkungen bezüglich des Paradigmas vorgibt.

Die E-Learning-Standards bieten bereits Lösungsansätze für einige Teilprobleme dieser Arbeit an. So umfasst beispielsweise der SCORM-Standard und die IMS-Spezifikation (vgl. Abschnitt 2.3) ein Modell zur Gruppierung und Strukturierung von Kursen, inklusive der enthaltenen Assets. Durch diese generelle Struktur können beliebige Kurse in ein Format überführt werden, welches kompatibel zu anderen Systemen ist, insbesondere zu Learning Management Systemen. Dieses fördert neben der Kompatibilität auch die *Nachhaltigkeit* von Lerninhalten. Bei der Analyse der Autorensysteme hat sich herausgestellt, dass Kurse, die keine eigene Struktur aufweisen, trotzdem durch die Standards abgebildet werden können. Zwar werden die Standards in diesem Fall ausschließlich zum Datenaustausch verwendet, jedoch besitzen sie aufgrund ihrer Aggregationsfähigkeit eine Eigenschaft, die zu einer mächtigen Datenstruktur für die programminterne Repräsentation von Lerninhalten führt. Die Kriterien *Modularität* und *Granularität* können unter Berücksichtigung der Standards mit in die Modellierung einfließen. Die Begriffe *Modularität* und *Granularität* werden ebenso für Lernobjekte (vgl. Abschnitt 2.4) auf theoretische Weise beleuchtet.

Mit der Einführung von Metaphern wird versucht, ein allgemeines Verständnis für den Aufbau gut strukturierter Inhalte zu erarbeiten. Vor allem geht es – wie sich bereits in anderen Disziplinen gezeigt hat – um eine verbesserte Nutzung bestehender Ressourcen, also um deren *Wiederverwendung*. Vergleiche mit der Bauindustrie haben gezeigt, dass heute Lernmaterialien oft neu entwickelt und bestehende Inhalte nicht genügend weiter genutzt werden. In der Bauindustrie, wo bereits vor Jahren Standards und Normen festgelegt wurden, werden gleiche Hauskomponenten von verschiedenen Herstellern konstruiert und kombiniert.

Für die Konstruktion von Lerneinheiten ist die Berücksichtigung der modularen und granularen Gestaltung ein äußerst wichtiges Ziel. Denn nur so können einzelne Komponenten aus Kursen entnommen und wiederum in andere Kurse eingebettet werden. Die Analyse der Autorensysteme (vgl. Abschnitt 3.4.2) hat jedoch gezeigt, dass bis auf das LRN-Toolkit keins der vorgestellten Systeme eine wie von Hodgins und Downes (vgl. Abschnitt 2.4.1) vorgeschlagene modulare Struktur anbieten. Zwar gibt es Bemühungen, diese Theorien mit aktuellen Autorensystemen zu koppeln (vgl. Abschnitt 3.4.2 und [Cuthbert02]), wobei einem über die Jahre entstandenen System sicherlich nicht einfach postum ein neues Konzept angehaftet werden kann. Hinzu kommt, dass die bis jetzt unternommenen Bemühungen bei der Umsetzung der Metaphern lediglich die Aspekte der modularen Strukturierung aufgreifen. Das Problem, dass Inhalte ebenso thematisch zu anderen Inhalten passen müssen, wurde bis jetzt völlig außer Acht gelassen.

Die Kategorie *Wiederverwendung* bezieht sich nicht nur auf entwickelte Lernobjekte, sondern ebenso auf Ressourcen, die bereits mit anderen konventionellen Systemen, wie z. B. Latex oder Office-Produkten, studienbegleitend entwickelt wurden. Gerade die partielle Wiederverwendung solcher Inhalte ist ein wichtiger Aspekt dieser Arbeit. Die Slicing Book Technik und die im ITO-Projekt entwickelten Abbildungsverfahren (vgl. Abschnitt 3.1), liefern interessante Aspekte, die im nächsten Kapitel konzeptionell aufgegriffen werden.

Die Notwendigkeit, Inhalte von ihrer Darstellung zu trennen, wird immer häufiger bei der Entwicklung von E-Learning-Systemen berücksichtigt. Hierbei werden im Besonderen zwei Ziele verfolgt: Inhalte werden mit einem einfachen Dokumentparser maschinell eingelesen und verschiedene Ausgabeformate werden generiert. Das DocBook-Format ist beispielsweise ein Vertreter der Markupsprachen, das für die Erstellung technischer Dokumentationen eingesetzt wird, die unabhängig von einem Zielformat formuliert werden. Im Gegensatz dazu ist OMDoc ein generisches Austauschformat für *mathematische* Inhalte. Leider unterliegen die vorgestellten Sprachen, wie z. B. EML, einem zu starken pädagogischen Einfluss, so dass modulare Strukturen nur unzureichend ausgeprägt sind.

Die Begriffe *Lernplattformen* und *Autorensysteme* sind nicht eindeutig definiert, weil sie je nach Anwendungszweck unterschiedliche Dienste anbieten oder auf unterschiedlichen Konzepten basieren (vgl. Abschnitt 3.4 und 3.3). Die Auswertung dieser Systeme hat ergeben, dass kein aufgeführtes System die in Kapitel 1 geforderten Eigenschaften implementiert. Insbesondere der *kooperative* Entwicklungsprozess, der gerade bei der Entwicklung interdisziplinärer Inhalte notwendig ist, wird nicht berücksichtigt.

Abschließend ist festzuhalten, dass die theoretischen Modelle viel Potential für die Konzeption eines modularen E-Learning-Systems liefern, es bislang jedoch nicht ausgenutzt wurde. Aus diesem Grund besteht der Bedarf ein technisches Modell zu entwickeln, welches erstmals zusätzlich zu der Struktur von Kursen auch deren Inhalte im Sinne der Metaphern formal als Lernobjekte modelliert, um die zu Beginn dieser Arbeit aufgeworfenen Fragestellungen zu lösen. Folglich sollen Lernobjekte für E-Learning-Systeme ähnliche Eigenschaften wie Präsentationsfolien in PowerPoint besitzen. Folien von anderen Autoren lassen sich bequem und mit geringem Aufwand in den eigenen Vortrag einbinden, da sie an das Layout und die Struktur des neuen Vortrags durch das System angepasst werden.

Teil II

Modellierung und Umsetzung

Kapitel 5

Ein Modell zur Entwicklung konsistenter Lernmaterialien

In diesem Kapitel wird ein technisches Modell vorgestellt, das als Grundlage für die Konzeption von E-Learning-Systemen dient, bei denen Kooperation und Interdisziplinarität im Vordergrund steht. Das Modell beschreibt die hierfür notwendige Kursstrukturierung, um eine optimale Kompatibilität mit anderen Kursen zu erreichen. Zuerst wird das Baukastenprinzip erörtert, welches im Rahmen des mαth-kit Projekts entstanden ist, und dessen Komponenten vorgestellt. Nach diesem Überblick findet eine detaillierte Betrachtung der jeweiligen Komponenten statt, bei der technische Realisierungsvorschläge präsentiert werden.

Der zweite wesentliche Aspekt des Modells befasst sich mit der Verarbeitung bereits in digitaler Form vorliegender Lehrinhalte, sowie mit der Entwicklung von Lehrinhalten für unterschiedliche Lehrformen, wie z. B. dem Präsenzunterricht oder dem Selbststudium.

Im letzten Abschnitt wird gezeigt, wie das Modell für ein kooperativ arbeitendes Team, bestehend aus Entwicklern für E-Learning-Inhalte, genutzt werden kann.

5.1 Das Baukastenprinzip

Das Ziel, das im Projekt mαth-kit verfolgt wurde, war die Entwicklung eines multimedialen Baukastens für die Mathematikausbildung im Grundstudium. Der Aspekt des Baukastens wurde als Metapher für die Konzeption eines Modells gewählt [Bungenstock02], um eine allgemeine Leitidee sowohl für Entwickler und Entwicklerinnen als auch für Anwender und Anwenderinnen zu vermitteln. Das Baukastenprinzip basiert auf den in Kapitel 2.4.1 vorgestellten Metaphern und erweitert die dort vorgestellten Ideen um eine zentrale Verwaltungsstelle für Lehrinhalte, das Baukastensystem, sowie ein einheitliches Erscheinungsbild. Hierbei wird der Frage nachgegangen, was mit einer Konstruktionen nach ihrer Fertigstellung geschieht. Diese beiden Aspekte beziehen sich vor allem auf die zu Beginn dieser Arbeit aufgeworfenen Fragestellungen nach kooperativer Konstruktion und dem damit verbundenen Problem einer konsistenten Präsentation.

5.1.1 Bausteine

Zentraler Begriff des Baukastenprinzips ist der Baustein. Ein real existierender Baukasten besteht aus einer Sammlung unterschiedlicher Bauklötze. Sie unterscheiden sich in ihrer Form und Farbe und können unter Berücksichtigung physikalischer Gesetze vielseitig zusammengesetzt werden. Ein Baukasten enthält zumeist mehrere Exemplare eines Bausteins, die hinsichtlich ihrer Form gleiche Eigenschaften besitzen. Dieses können z. B. Kegel, Zylinder oder Quader sein. Bausteine identischer Form müssen jedoch nicht zwangsläufig die gleiche Farbe besitzen. Während der Konstruktion komplexer Bauwerke wird zumeist versucht, diejenigen Bausteine zusammenzusetzen, die auch optisch zueinander passen.

Entsprechend der Leitidee nach Hodgins und Willy (vgl. Abschnitt 2.4) besitzt der primitive Baustein zunächst die Eigenschaft mit anderen Bausteinen kompatibel zu sein und in beliebiger Weise miteinander kombiniert werden zu können:

- Jeder Baustein ist kompatibel mit jedem anderen,
- Bausteine können in beliebiger Weise zusammengesetzt werden,
- Bausteine sind so einfach aufgebaut, dass sie von jedem zusammengesetzt werden können.

Zu diesen drei formalen Eigenschaften müssen weitere Regeln festgelegt werden, nach denen festgestellt werden kann, ob es sinnvoll ist, Bausteine überhaupt zusammenzusetzen. Die folgenden Regeln müssen während der Konstruktion zusätzlich berücksichtigt werden:

- Jeder Baustein soll auch optisch zu anderen passen,
- Konstruktionen sollten möglichst stabil sein,
- Bausteine sollen kontextfrei sein.

Diese Regeln erscheinen auf den ersten Blick trivial und werden intuitiv angewendet. Wie verhalten sich jedoch Entwickler bzw. Entwicklerinnen von realen E-Learning-Systemen? Werden diese Regeln bei der Entwicklung komplexer E-Learning-Systeme beachtet und wie sehen Kurse strukturell aus, die aus mehreren Lernobjekten verschiedener Entwickler zusammengesetzt werden? Der SCORM und IMS Standard (vgl. Kapitel 2.3) bieten zwar eine technische Lösung zum Archivieren bestehender Inhalte an, die es jedem Kursfragment ermöglicht, ganz gleich wie es intern strukturiert ist, mit anderen Kursen kombiniert zu werden, legen damit aber eine neue Form für den Baustein fest. Lernmaterialien bestehen häufig aus unterschiedlichen Dateiformaten, wie z. B. Flash-Filmen oder PDF-Dokumenten und besitzen damit eine grundlegend unterschiedliche interne Struktur und Präsentation. Werden sie einfach zusammengeführt, fehlen gerade die in dieser Arbeit wichtigen Aspekte der konsistenten Kursstruktur und die einheitliche Präsentation. Anstatt die Lernmaterialien einfach einzupacken, sollte vielmehr ein generelles Zwischenformat entwickelt werden, das dem Kursentwickler bzw. der Kursentwicklerin die Verwendung optimal zusammenpassender Bausteine ermöglicht. Abbildung 5.1 zeigt exemplarisch einen

generischen Baustein, der unabhängig formuliert ist und je nach Bedarf des Lehrenden auf ein für ihn zutreffendes Präsentationsformat abgebildet werden kann. Das Beispiel zeigt einen Baustein, bestehend aus dem Inhalt, den zugehörigen Abbildungen und einer Animation. Dieser wird für eine Online-Präsentation, abhängig von den gestalterischen Aspekten, die von einem zugehörigen Kurs vorgegeben wird, auf zwei verschiedene HTML-Versionen abgebildet. Ebenso soll eine druckbare Version für eine Präsenzveranstaltung angefertigt werden. Hierfür muss der Baustein in ein Dokumentformat, in diesem Beispiel das Portable Document Format (PDF), überführt werden, um es ggf. in ein bereits existierendes Skript einzuflechten.

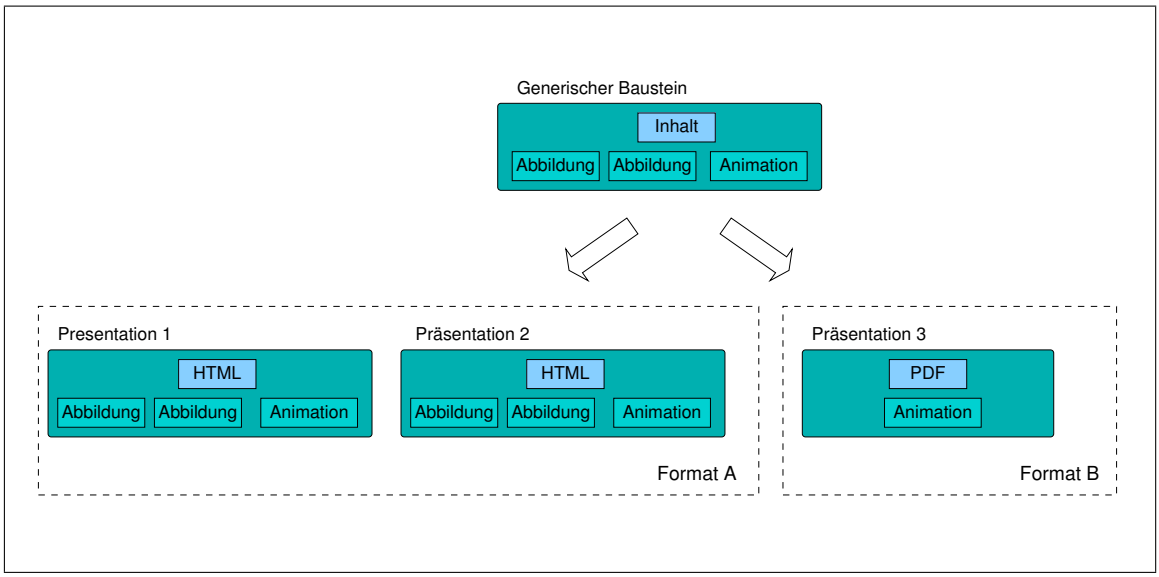


Abbildung 5.1: Basisformat für Lernbausteine

5.1.2 Kurse

Kurse entsprechen in der Analogie eines Baukastens, den entwickelten Bauwerken, oder auch Modellen bestehend aus einer Vielzahl von Bausteinen. Dem Baukasten liegt in der Regel eine Anleitung mit Konstruktionshinweisen bei. Sie kann Angaben über die Anordnung der Bausteine und deren Typ enthalten. Anhand des Bauplans wird dem Gebäude eine gewisse Struktur auferlegt, mit der der Konstrukteur bzw. die Konstrukteurin entscheiden kann, welche Bausteine dem Baukasten zu entnehmen sind. Nun kann es vorkommen, dass entwickelte Bausteingruppen – aufgrund von Symmetrieeigenschaften eines Bauwerks – mehr als einmal verwendet werden. Wenn das Bauwerk fertig konstruiert ist, kann es je nach Bedarf fixiert und ggf. lackiert werden. Dieser Zustand verhindert zwar, dass die verwendeten Bausteine für andere Konstruktionen eingesetzt werden können, fixiert jedoch das Gesamtresultat.

E-Learning-Systeme verwenden – wie bereits in Abschnitt 3.4 gezeigt – ganz unterschiedliche Kursstrukturen. Die Spanne reicht von Videosequenzen, die Studierende interaktiv steuern, über sequenzielle Abfolgen von Kursfragmenten und hierarchisch aufge-

bauten Kursen bis hin zu vernetzten personalisierten Kursstrukturen. Um für eine solche Vielfalt ein kompatibles Austauschformat zu entwickeln, das Learning Management Systemen die Verarbeitung beliebiger Kurse ermöglicht, wurde der SCORM-Standard und die IMS-Content Package Specification entwickelt (vgl. Abschnitt 2.3.1). Das dort entworfene hierarchische Modell für E-Learning-Kurse bietet sich für die Strukturierung von Kursen geradezu an, da es neben der Verwaltung ganz normaler Kursstrukturen auch das Einbinden von Subkursen unterstützt.

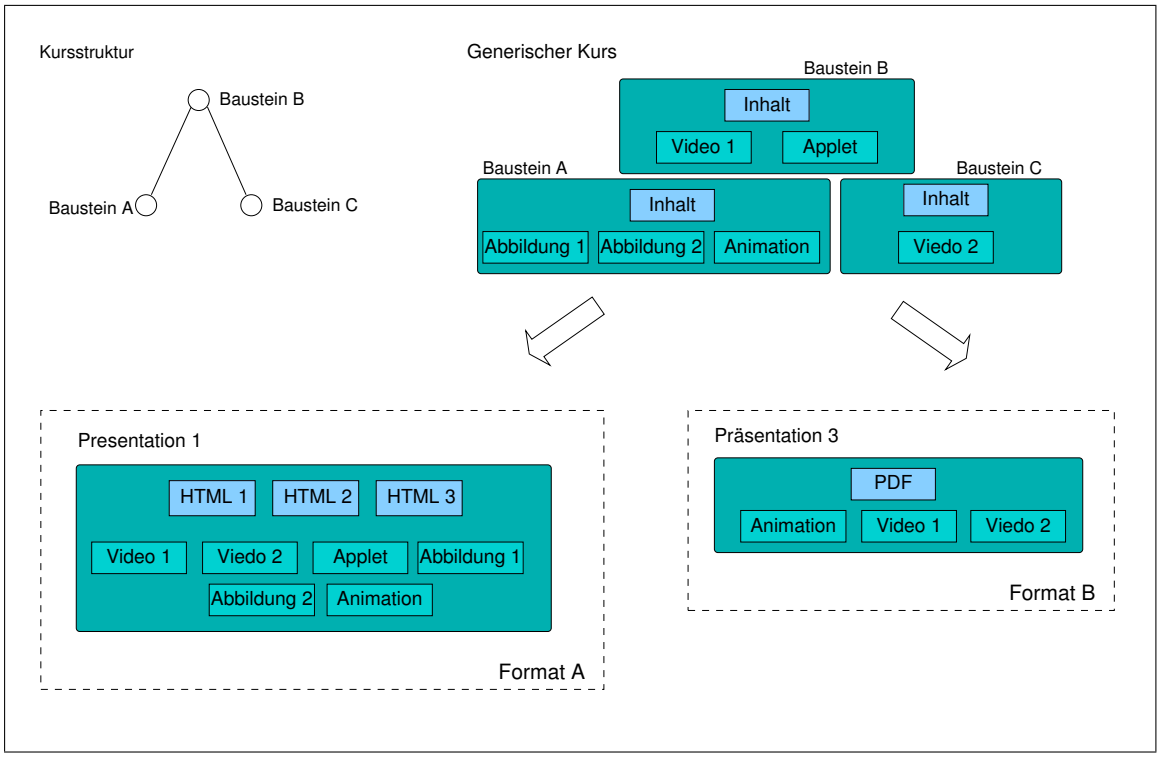


Abbildung 5.2: Basisformat für Kurse

Die in Abschnitt 5.1.1 eingeführten Lernbausteine müssen durch eine Kursstruktur zusammengesetzt werden. Abhängig vom gewünschten Ausgabeformat wird diese Struktur auf ein Format abgebildet, welches dem eigentlichen Lernmaterial entspricht. Abbildung 5.2 zeigt exemplarisch die Zusammenstellung der drei Lernbausteine A, B und C. Baustein B ist den Bausteinen A und C hierarchisch übergeordnet und befindet sich in der zu generierenden Kursstruktur auf der darüber liegenden Ebene. Aufgrund der Kursstruktur und der Inhalte der Bausteine ist es nun möglich, aus der universellen Kursstruktur einen kompletten Kurs zu generieren, mit einer konsistenten Struktur und einer einheitlichen Darstellungsform.

5.1.3 Prozess

Bei der Entwicklung von E-Learning-Inhalten werden unterschiedliche Fachkompetenzen benötigt, die in der Regel nicht nur bei einer Person liegen. So fällt beispielsweise die Zu-

sammenstellung und die Auswahl von studienbegleitenden Lernmaterialien in den Kompetenzbereich von Professoren und Professorinnen. Die eigentliche Entwicklung von Animationen, Flash-Filmen und Programmen, wird hingegen oft von studentischen Hilfskräften durchgeführt. Eine solche Rollenverteilung soll im Folgenden für das Baukastenprinzip aufgestellt werden. Anhand der festgelegten Rollen soll der Umgang mit dem Baukastensystem erläutert werden, sowie der sich daraus ergebende Entwicklungsprozess von Lernmaterialien. In [Baudry02b, Baudry02a] nehmen die bei der Entwicklung beteiligten Personen folgende Rollen an.

Teacher: Der *Teacher* verwendet einzelne Bausteine als Ergänzung zu Präsenzveranstaltungen und Fernlehre. Aufgabe dieser Rolle ist das Suchen und Archivieren von Lernbausteinen und Kursen.

Student: Der *Student* arbeitet mit einem Learning Management System (LMS) an generierten Kursen. Hierfür stehen spezielle Lernbausteine zur Exploration und Aufgabenbearbeitung zur Verfügung. Darüber hinaus können Studierende bei Bedarf Inhalte nachschlagen und vertiefen, die der Lehrende im Lehrplan nicht explizit vorgesehen hat.

Composer: Diese Rolle ist für die Kurserzeugung zuständig. Für einen Kurs müssen ausgewählte Bausteine gesucht und zu einem Kurs für Studierende zusammengesetzt werden. Gibt es zu einem Thema keinen geeigneten Baustein um die Lücke zu schließen, muss in die Rolle *Developer* gewechselt werden.

Developer: Der *Developer* erzeugt multimediale Bausteine und pflegt sie in das Baukastensystem ein. Hierbei müssen Assets, wie z. B. Applets, Flash-Filme, Texte, Grafiken und Animationen, erzeugt und zu Bausteinen kombiniert werden. Darüber hinaus wird der Inhalt in einem allgemeinen, darstellungsfreien Format definiert und kann somit anderen Autoren zur Verfügung gestellt werden.

Publisher: Diese Rolle passt einen zusammengestellten Kurs an die gewünschte Lehrveranstaltung an. Kurse können in verschiedene Darstellungen und Ausgabeformate übersetzt werden.

Administrator: Diese Rolle unterstützt während der Entwicklung die anderen Rollen und ist für den reibungslosen Funktionsablauf des Baukastensystems verantwortlich.

Als Beispiel soll im Folgenden der Entwicklungsprozess und das damit verbundene Zusammenspiel der Rollen anhand eines Beispiels, das die schrittweise Entwicklung eines Grundlagenkurses für die Technische Informatik veranschaulicht, vorgeführt werden. In Abbildung 5.3 werden zunächst die vier Bausteine *Allgemeine Aufgaben*, *Definition Komplexe Zahlen*, *Explorationswerkzeug Komplexe Zahlen* und *Elektrotechnik Aufgaben* von der Rolle Developer erzeugt [1]. Um aus ihnen einen Kurs über Schaltkreise zu entwerfen,

[1] Diese Bausteine wurden im Rahmen des mαth-kit Projekts fachübergreifend erstellt und enthalten Aufgaben, Übungen und Explorationsumgebungen für das Thema Komplexe Zahlen. In Kapitel 7.1.3 wird der Konstruktionsprozess näher beschrieben.

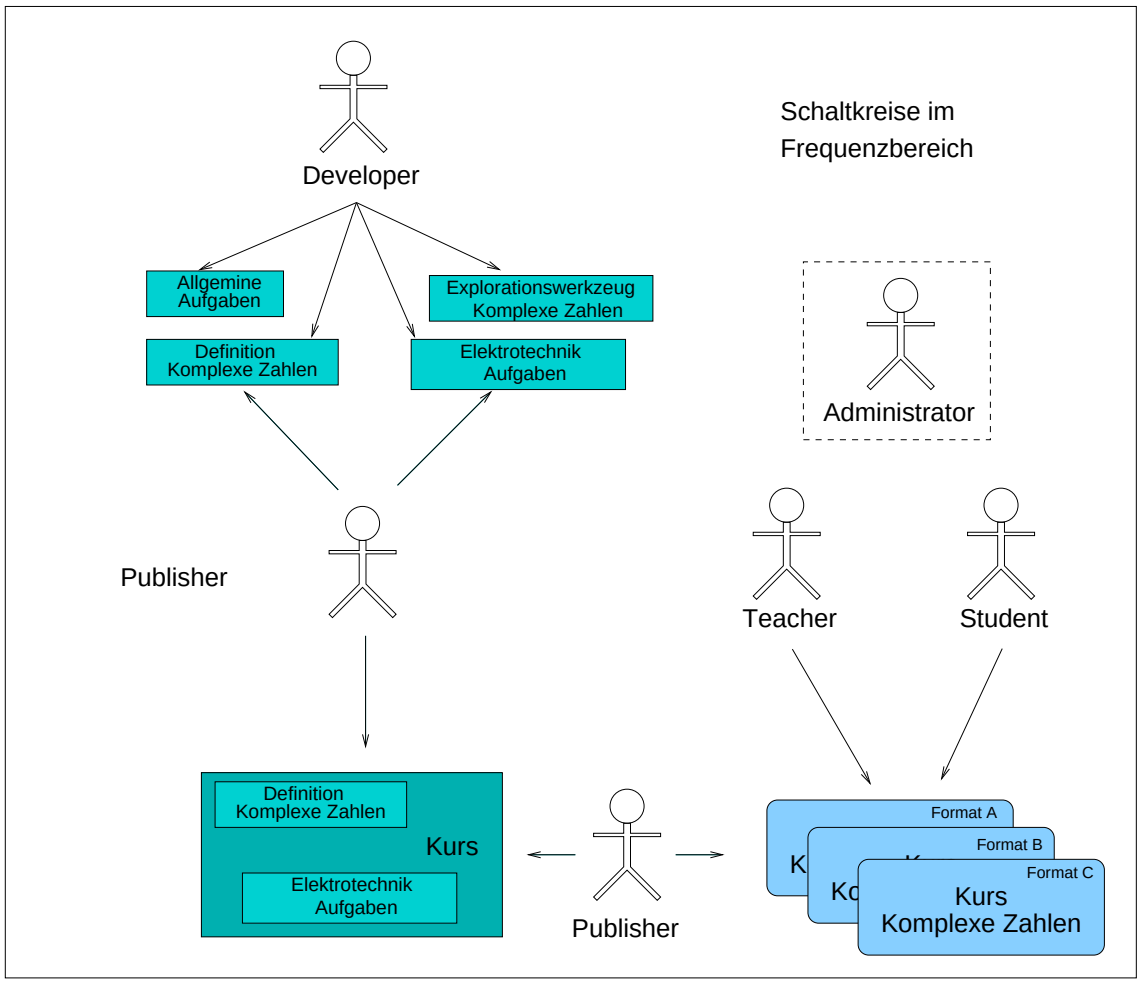


Abbildung 5.3: Interaktion der Rollen

werden aus dem Bestand der verfügbaren Bausteine diejenigen ausgewählt, die für den zu erstellenden Kurs von Interesse sind. In diesem Fall nimmt der Composer die Bausteine *Definition Komplexe Zahlen* und *Elektrotechnik Aufgaben* und kombiniert sie zu einem Kurs. Der Publisher nimmt diesen Kurs, der sich zu diesem Zeitpunkt in einem Zwischenformat befindet und nicht von Studierenden oder Lehrenden genutzt werden kann, und transformiert ihn in das gewünschte Ausgabeformat. Dieser Prozess ist vom jeweiligen Lernszenario abhängig, denn soll der Kurs Studenten studienbegleitend angeboten werden, bietet sich die Erstellung eines Online-Kurses mit zusätzlichen Aufgaben und Animationen an. Soll der Kurs jedoch eine Präsenzveranstaltung begleiten und dort in gedruckter Form verteilt werden, ist eine Transformation in das PDF-Format eine gute Lösung.

5.2 Formatierbare Lernbausteine

Nachdem der Lernbaustein als zentraler Gegenstand des Entwicklungsprozesses in einer metaphorischen Abbildung eingeführt worden ist, wird im Folgenden die Modellierung be-

schrieben. Der Frage nach den *Eigenschaften* eines *allgemeingültigen* Lernbausteins wurde bereits in der Metapher nachgegangen. Demgegenüber stehen die zu Beginn dieser Arbeit definierten Ziele *Modularität*, *Granularität*, *Kooperation* und *Wiederverwendbarkeit*.

Die Analysen aus Kapitel 2 haben ergeben, dass es keine Beschreibungssprache für Lerneinheiten gibt, die nicht nur eine modulare Aufteilung ermöglichen, sondern auch wesentliche Strukturmerkmale enthält. DocBook ist z. B. für die Definition ganzer Bücher oder Artikel konzipiert und nicht für die Beschreibung kleiner Kursfragmente. Dagegen hat die Analyse der Auszeichnungssprache OMDoc ergeben, dass sie nicht für die Eingabe großer Dokumente geeignet ist, sondern vielmehr für die Definition von Formeln und mathematischen Regeln.

Ein weiterer Aspekt betrifft die physikalische Strukturierung von Lernbausteinen, insbesondere die Verwaltung von eingebundenen Dateien, wie z. B. Videos oder Applets. Das IMS hat für dieses Problem die in Abschnitt 2.3.1 vorgestellte *Context Aggregation Model*-Spezifikation entwickelt. Sie regelt das physikalische Speichern von Assets, wie Videos oder Applets, in einem Lernobjekt. Jedes Paket enthält ein *Manifest*, das in einer XML-Kodierung die Dateien der eingebundenen Ressourcen verwaltet.

Aufbauend auf diesen Erkenntnissen soll in diesem Abschnitt ein Strukturmodell entwickelt werden, mit dem es möglich ist, Bausteine unterschiedlicher Granularität zu entwerfen, die von anderen Autoren oder Autorinnen übernommen werden können. Dieses ist jedoch nur unter Berücksichtigung der in Kapitel 5.1.1 eingeführten Regeln zu erreichen. Aus ihnen lässt sich der folgende technische Bausteinbegriff ableiten:

> Ein **formatierbarer Baustein (FB)** besteht aus einem generellen, darstellungsfreien Dokument beliebiger Granularität zur strukturierten Speicherung des Inhalts und wird erst in Verbindung mit einem Regelwerk für ein beliebiges Präsentationsformat zu einem Lernobjekt. Formatierbare Bausteine besitzen einen definierten Rahmen, der die Aggregation mit anderen formatierbaren Bausteinen ermöglicht.

Abbildung 5.4 zeigt den prinzipiellen Aufbau eines formatierbaren Lernbausteins. Auf der linken Seite befinden sich Assets, die in einer Dateistruktur verwaltet werden. Jeder Baustein enthält einen Satz zugeordneter Assets, in diesem Fall *Bild1*, *Bild2*, *Applet* und *Video*, und verfügt über exakt ein Dokument, das die interne Dokumentstruktur des Bausteins kapselt.

Die Dokumentenstruktur lässt sich als Baum mit einer Wurzel und untergeordneten Teilbäumen oder Blättern darstellen. In diesem Beispiel teilt sich das Dokument auf oberster Ebene in eine Überschrift und in einen Paragraphen auf. Letzterer enthält zwei Bilder, zwischen denen ein Textabschnitt positioniert ist. Die Überschrift besitzt ebenfalls einen Paragraphen, der neben einem Textblock noch eine Tabelle enthält, die sich über die Anzahl ihrer Zeilen definiert. Die Abbildung beschreibt die Struktur einer Tabelle, die auf der linken Seite Textblöcke enthält und auf der rechten Seite multimediale Elemente. Die Blätter des Baums, für die es im Übrigen keine textuelle Entsprechung gibt, sondern lediglich Binärdaten, referenzieren die in der externen Struktur verwalteten Dateien.

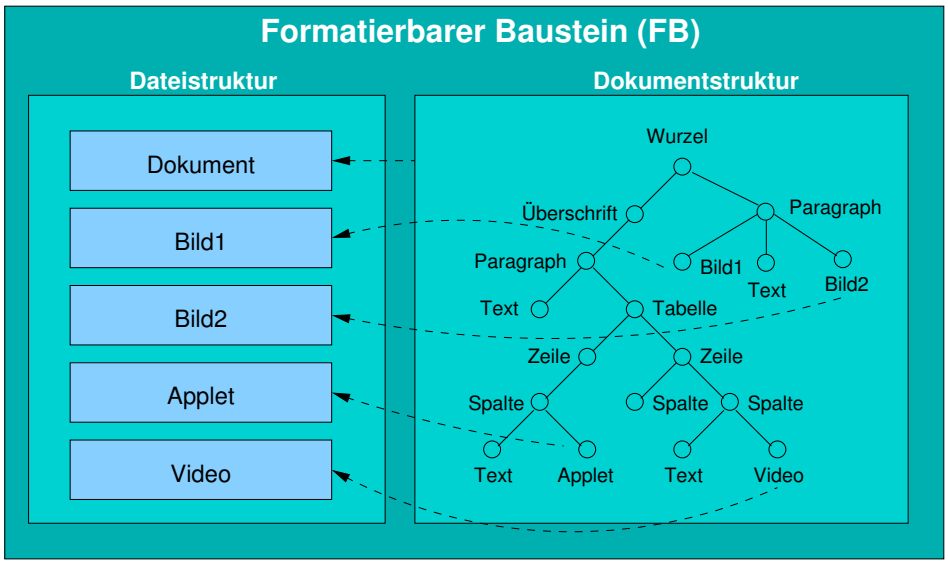


Abbildung 5.4: Struktur eines formatierbaren Bausteins

5.2.1 Dokumentstruktur

Für die Strukturierung des Dokuments wurde die Sprache BrickML entwickelt, die eine Vielzahl an Basiselementen zur formalen Beschreibung anbietet. Während der Entwicklung hat sich herausgestellt, dass die syntaktische Struktur einfach aufgebaut werden muss, damit sie sowohl leicht zu erlernen ist und sich gut zum Überprüfen von Fehlern eignet. Dieses Ziel wird erreicht, indem sämtliche Formatierungseigenschaften nicht Bestandteil der Dokumentstruktur sind, sondern in einem Regelwerk separat definiert werden.

Wurzelobjekt

Zunächst wird der Wurzelknoten durch das `lob`-Element (Learning Object) definiert. Er ist der Einstiegspunkt für einen Baustein und enthält entweder reinen Text oder komplexere Strukturen, wie Tabellen oder Listen. `lob`-Elemente besitzen jedoch keine Strukturierungselemente, wie z. B. Kapitel oder Abschnitte, weil sie ausschließlich zur Strukturierung von Kursen dienen und daher in der Regel thematisch abgeschlossen sind. Aus diesem Grund wird innerhalb der Bausteine auf solche Elemente verzichtet, damit der zu erzeugende Kurs konsistent ist und sich Kapitelinformationen innerhalb des Dokuments nicht mit denen der Kapitelstruktur vermischen.

Der Typ des `lob`-Elements enthält mehrere thematisch getrennte Elementgruppen: `textGroup`, `structureGroup`, `learnGroup`, `mathGroup` und `mediaGroup`. Jedes Element dieser Gruppen kann beliebig oft und in einer nicht spezifzierten Reihenfolge unter einem `lob`-Element auftreten. Zudem lassen sich `p`-Element (Paragraphen) für die Definition von

Absätzen unter einem `lob`-Element einfügen. Abbildung 5.5 veranschaulicht die syntaktische Struktur eines `lob`-Elements.

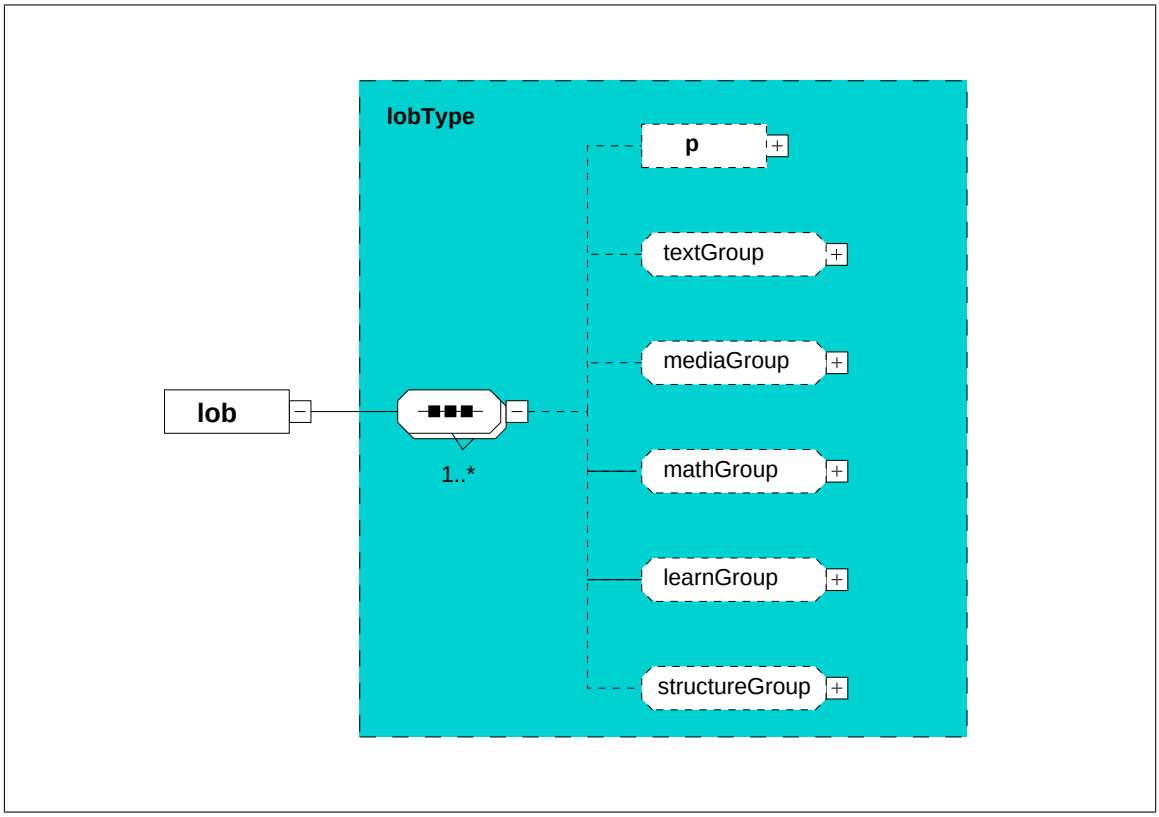


Abbildung 5.5: Syntaktische Struktur des LOB-Elements

Textblöcke

Der Typ `textGroup` enthält Elemente, die hauptsächlich zur Auszeichnung von Texten verwendet werden. Dieses sind `highlight` (zum Hervorheben von Textpassagen) und `italic` (zur kursiven Darstellung von Textabschnitten). Sollen Texte unterstrichen werden, können sie mit dem `underline`-Element definiert werden. Hin und wieder sollen Textstellen jedoch in einen Schreibmaschinen ähnlichen Schriftstil gesetzt werden. Zu diesem Zweck existiert das `typewriter`-Element. Dieses Element spezifiziert zwar die Textformatierungen, legt jedoch nicht ihre genaue Präsentation fest. Es wird z. B. keine Aussage darüber getroffen, wie oft eine Textpassage unterstrichen ist, oder wie die Hervorhebung auszusehen hat.

Der praktische Einsatz hat ergeben, dass Texte noch weitere Formatierungselemente benötigen. Dazu zählen Tabulatoren `t`, Zeilenumbrüche `br` und Leerzeichen `ws`. Auch diese Elemente dienen ausschließlich der Strukturierung und besitzen keine eigene Präsentationsform. Des weiteren muss die aus Hypertext bekannte Verknüpfung modelliert werden. Wie in HTML-Dokumenten üblich, verweisen Inhalte unter Verwendung des `link`-Elements auf andere Hypertext Dokumenten. Weiterhin werden die Elemente `anchor` und `ref` benötigt.

Sie werden für Verweise innerhalb von Bausteinen verwendet und ermöglichen darüber hinaus auch eine Referenzierung auf Inhalte anderer Bausteine. Abbildung 5.6 zeigt den strukturellen Aufbau der Textgruppe.

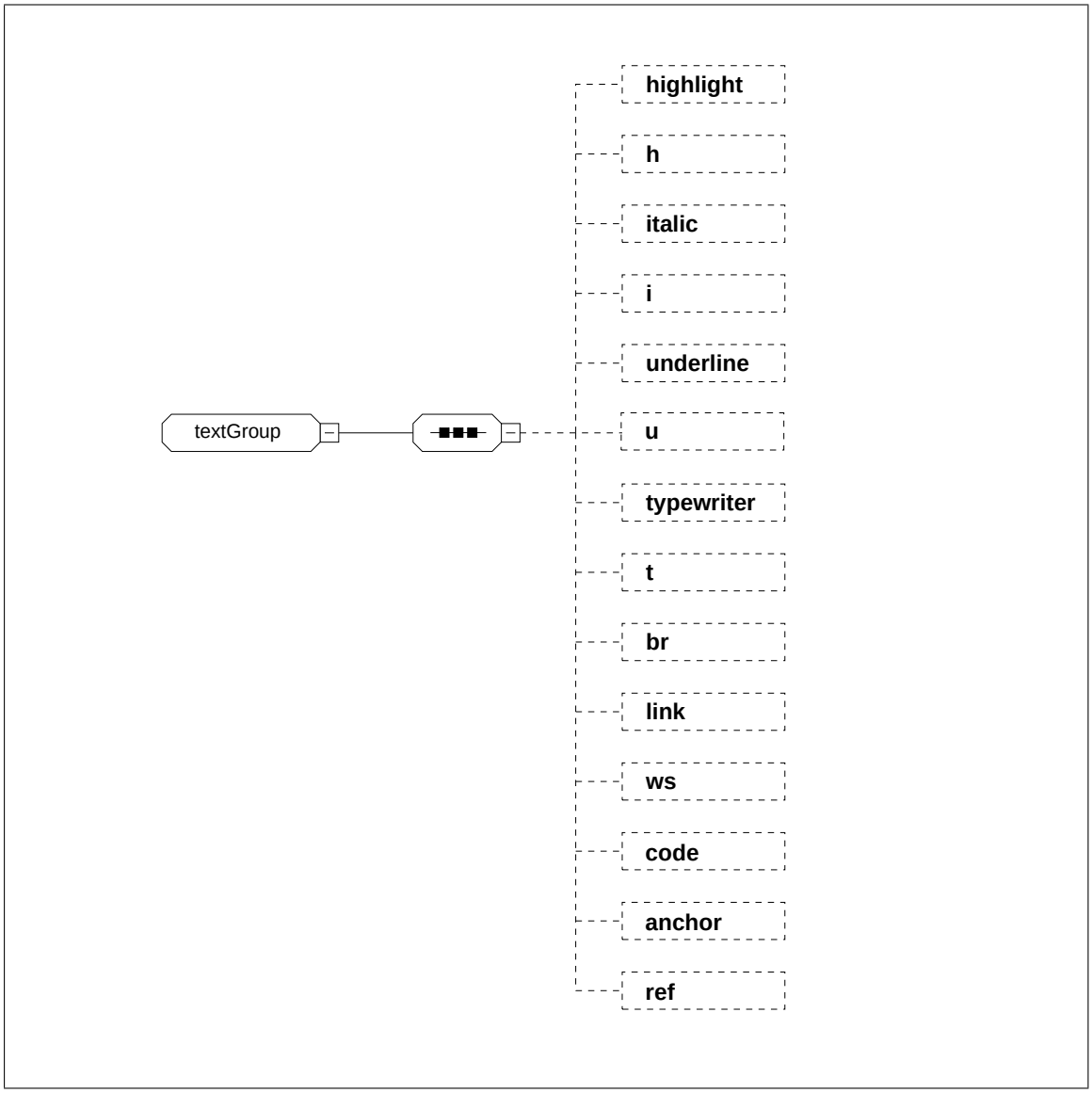


Abbildung 5.6: Textgruppe

Strukturen

Das Verfassen einfacher Texte reicht jedoch für die Gestaltung von Inhalten meist nicht aus. Vielmehr werden Elemente, wie z. B. Aufzählungen oder Tabellen, benötigt, die unerlässlich für den Aufbau von Dokumenten sind. Aus diesem Grund werden in der `structureGroup` die Typen für Tabellen, Aufzählungen, Nummerierungen und Überschriften zusammengefasst. Eine Tabelle besteht aus einer Sequenz beliebig vieler `column`-Elemente,

die zum einen die Anzahl der Spalten der Tabelle angeben und zum anderen Daten über die Spalten enthalten, wie die Zeilenbreite, gefolgt von einer beliebigen Anzahl `row`-Elemente. Letztere enthalten jeweils eine Sequenz von `cell`-Elementen, deren Eigenschaften, wie z. B. die Ausdehnung angrenzender Spalten und Zeilen zur Konfiguration dienen. Tabelle 5.1 zeigt die einstellbaren Attribute des `col`-Elements.

Innerhalb einer Tabellenzelle muss es möglich sein, Texte und andere strukturelle Elemente einzubinden. Aus diesem Grund schließt der Zellentyp neben dem `p`-Element, zur Definition weiterer Paragraphen, auch die Gruppen `textGroup`, `mediaGroup` und `structureGroup` ein. An dieser Stelle sei angemerkt, dass die Strukturgruppe rekursiv definiert ist, was eine notwendige Vorraussetzung bei der Erstellung rekursiver Strukturen, wie beispielsweise verschachtelte Tabellen oder verschachtelte Aufzählungen, ist. Aufzählungen und Nummerierungen benutzen den gemeinsamen Elementtyp `enumerationType`. Er erlaubt das Einbinden beliebiger `item`-Elemente, die die gleiche Struktur wie der Zelltyp besitzt, jedoch unterschiedliche Eigenschaften aufweist. Dadurch wird ebenfalls das rekursive Einbinden von Tabellen und Nummerierungen innerhalb einer Aufzählung ermöglicht. Zu guter Letzt enthält die Strukturgruppe Überschriften. Sie dienen lediglich zur Trennung von Textabschnitten und werden bei der Generierung von Kursstrukturen berücksichtigt.

Name	Typ	Verwendung
valign	normalizedString	optional
width	decimal	optional
height	decimal	optional
colspan	byte	optional
rowspan	byte	optional
unit	normalizedString	optional

Tabelle 5.1: Attribute des `cell`-Elements

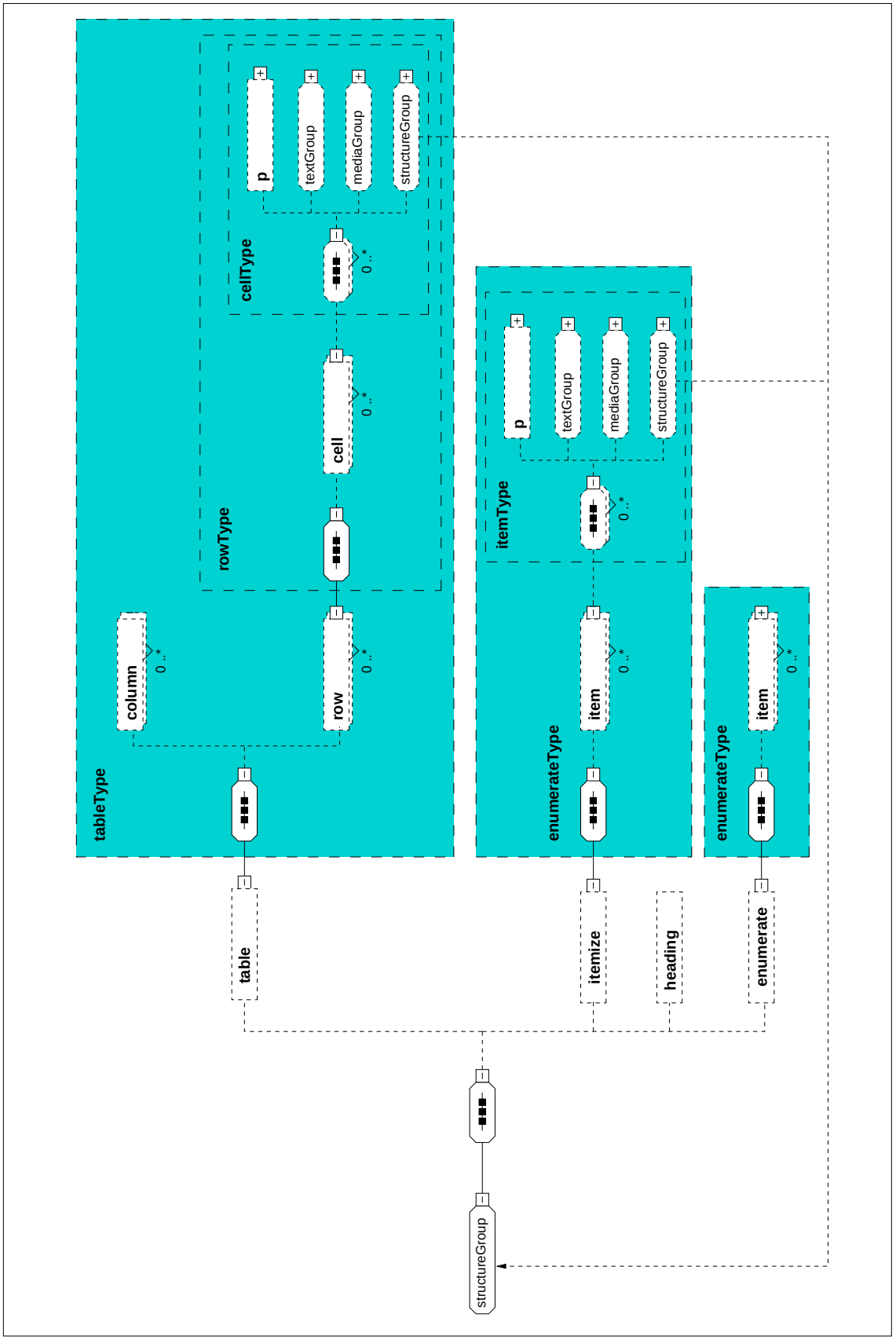


Abbildung 5.7: Strukturgruppe

Media Gruppe

Die Media Gruppe (vgl. Abbildung 5.8) verwaltet jene Elemente, die zur Darstellung multimedialer Inhalte benötigt werden. Sie besteht aus den Elementen `image`-Element und `mmo`-Element. Das `image`-Element bindet Abbildungen in sämtlichen Bildformaten, unter anderen `png`, `jpg`, `gif`, `pdf`, ein. Es wurde in Anlehnung an das HTML-Element `img` definiert, mit dem Unterschied, dass zu den Bitmap-Grafiken auch Vektorformate angeboten werden. Die Verwendung eines Bildformats ist abhängig von dem zu erzeugenden Ausgabeformat und muss ggf. konvertiert werden. Hierzu wird in Abschnitt 5.5.3 der Einsatz von Prozessoren diskutiert. Zeitabhängige Formate, wie Videos oder interaktive Flashfilme, werden mit dem `mmo`-Element eingebunden. Das `mmo`-Element verfügt über mehrere Konfigurationsparameter, die in Tabelle 5.2.1 näher erläutert werden. In Abhängigkeit vom `type`-Attribute erfolgt eine kontextsensitive Auswertung der optionalen Attribute.

Name	Typ	Verwendung
type	normalizedString	required
width	decimal	optional
height	decimal	optional
code	normailzedString	optional
archive	normailzedString	optional
unit	normalizedString	optional
data	normalizedString	optional
src	normalizedString	optional
target	normalizedString	optional

Tabelle 5.2: Attribute des `mmo`-Elements

Die in diesem Abschnitt vorgestellten Gruppen beschreiben lediglich den Aufbau eines Bausteins. Die konkrete Implementierung kann daher mit unterschiedlichen Ansätzen realisiert werden. In späteren Kapiteln wird Bezug auf die hier vorgestellte Struktur genommen und die Implementierungsvarianten *XML-Bindig* (vgl. Abschnitt 6.1.2) und *OO-Binding* (vgl. Abschnitt 6.2) vorgestellt. Die eingeführte Bausteinstruktur wird mit BrickML bezeichnet und deckt die Basisfunktionalität, die bei der Erstellung von multimedialen Kursen benötigt wird, ab. Im nächsten Abschnitt wird gezeigt, wie der Funktionsumfang für eine spezielle Anwendungsdomäne erweitert werden kann.

5.2.2 Spezialisierte Inhalte

Die Entwicklung multimedialer Inhalte und die damit verbundene Strukturierung des Lernmaterials ist abhängig vom Themengebiet, den der Kurs beschreiben soll. So werden beispielsweise in der Mathematik zahlreiche Formeln und Definitionen benötigt, die explizit von einem Dokumentenmodell unterstützt werden müssen. Im Gegensatz dazu werden in einem Fachgebiet wie der Theologie weniger Definitionen, sondern Verse und Bücher referenziert. Ein weiteres Beispiel ist das Fachgebiet Rechtswissenschaften, in dem es unumgänglich ist, Gesetzesparagraphen zu beschreiben. Dieser Gedanke wurde bereits in

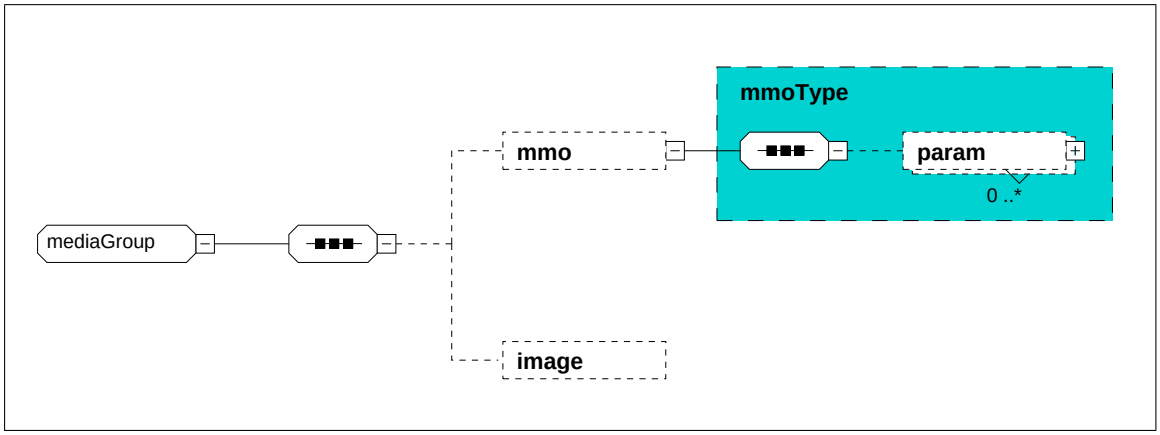


Abbildung 5.8: Mdeia Gruppe

Arbeiten von [Freitag02b] (vgl. Kapitel 3.2.5) aufgegriffen und soll in das hier vorgestellte Modell mit aufgenommen werden.

Die zwei wesentlichen Eigenschaften sind das *Erweitern* der Grundstruktur, um neue Elemente, sowie die *Spezialisierung* bestehender Elemente für dieselbe Fachdomäne mit anderen Eigenschaften. Erweiterungen sind dann notwendig, wenn fachbezogene Strukturen einem Dokument hinzugefügt werden müssen, die zur Integration domänenspezifischer Elemente, wie z. B. Paragraphen eines juristischen Textes, dienen. In diesem Fall werden Elemente mit einem neuen Namensraum (vgl. Abschnitt 3.2.1) versehen und der Grundstruktur hinzugefügt. Diese Form der Erweiterung führt zwangsläufig zu einer Einschränkung der in dieser Arbeit geforderten Kompatibilität mit Bausteinen anderer Autoren, weil eine Erweiterung des Regelwerks zur Formatierung mit einhergeht. Obwohl formatierbare Bausteine anderer Autoren mühelos in einen erweiterbaren Kurs mit aufgenommen werden können, ist jedoch deren Verwendung in einem herkömmlichen Kurs nicht möglich, weil entsprechende Formatierungsregeln fehlen. Aus diesem Grund muss eine Normalisierung durchgeführt werden, die spezialisierte Bausteine auf den Basissprachumfang zurückführt. Dabei findet eine surjektive Abbildung des neu definierten Strukturbaums auf die Standardelemente statt. Einfacher hingegen ist die Verwendung von Spezialisierungen. Hierbei werden neue Elemente von bereits bestehenden Element-Typen abgeleitet und kommen bei der Normalisierung ohne eine Neudefiniton der Formatierungsregeln aus.

Im Folgenden werden die im Projekt mαth-kit (vgl. Abschnitt 1.2) notwendigen Erweiterungen exemplarisch vorgestellt.

Spezialisierung für die Mathematik

Während der Laufzeit des mαth-kit Projekts hat sich der Bedarf einer Quizumgebung für Studierende herausgestellt. Sie soll die Überprüfung des erlernten Wissens anhand von Multiple-Choice-Tests ermöglichen. Hierbei kamen zwei Varianten einer Quizumgebung zum Einsatz. Der erste Quiz-Typ wertet ausgewählte Felder aus und präsentiert die richtige Lösung in Form einer Antwort. Der zweite Typ behandelt lediglich die Auswertung von

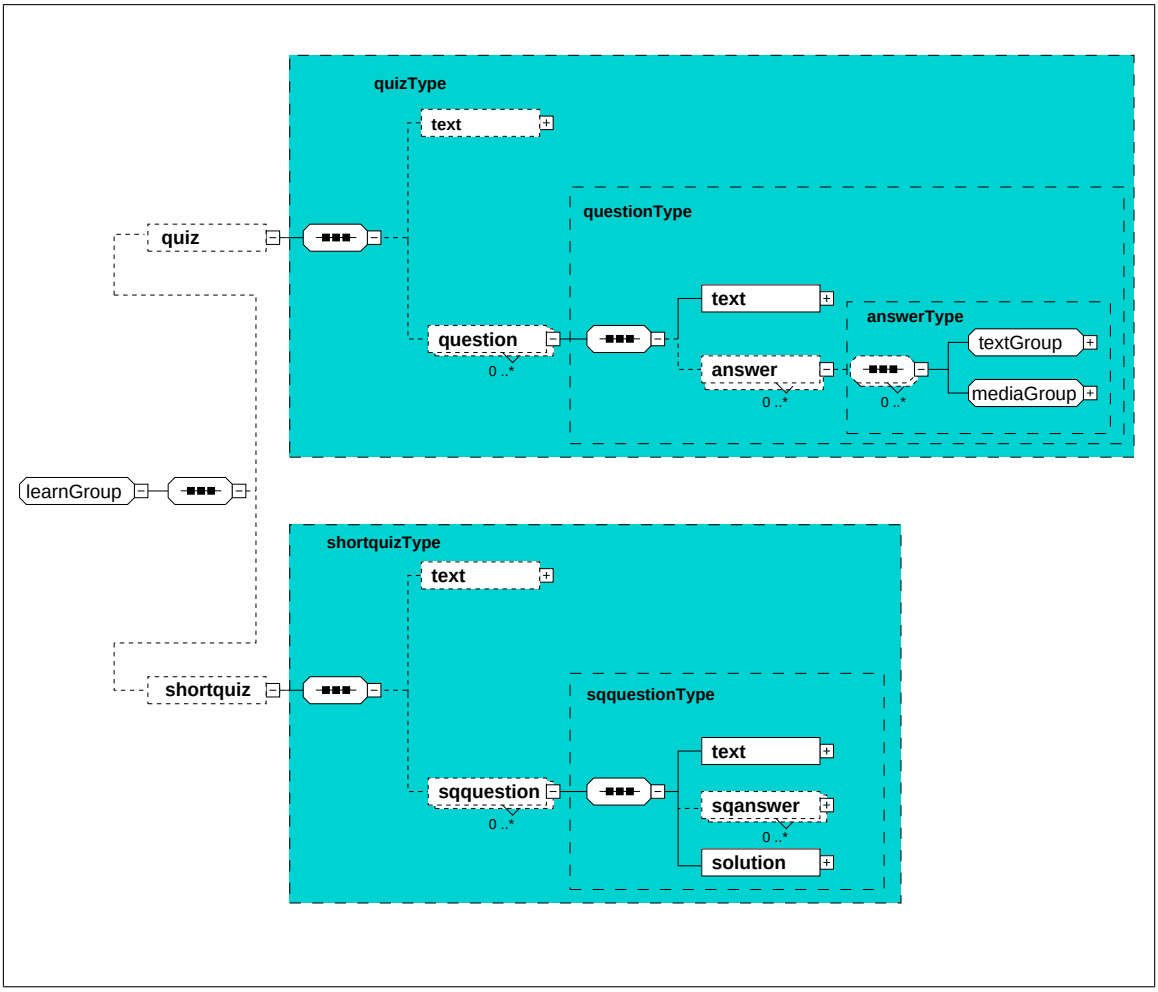


Abbildung 5.9: Lerngruppe

Fragestellungen und liefert die Anzahl richtig ausgewählter Antworten. Abbildung 5.9 stellt die Struktur der beiden Quiz-Elemente `quiz` und `shortquiz` vor. Zu beiden Quizzes gibt es einen optional einführenden Text, gefolgt von einer variablen Anzahl Fragen. Im Gegensatz zu dem `quiz`-Element enthält das `shortquiz`-Element die passende Antwort zu der Frage, die zwingend definiert werden muss.

Ein Beispiel für eine Spezialisierung ist die `mathGroup`. Da im Projekt mαth-kit überwiegend mathematische Inhalte entstanden sind, werden die benötigten Elemente `definition`, `conclusion`, `lemma`, `corollary`, `notice`, `hint` und `example` als Spezialisierung des allgemeinen `theorem`-Elements in einer eigenen Gruppe zusammengefasst.

5.2.3 Dateistruktur

Zur Strukturierung der Dateien eines Bausteins bietet sich das bereits in Kapitel 2.3.1 vorgestellte *IMS Content Package Information Model* an. Es organisiert beliebige Dateien in einem komprimierten Archiv. Kern des *Content Package* ist das Manifest, welches

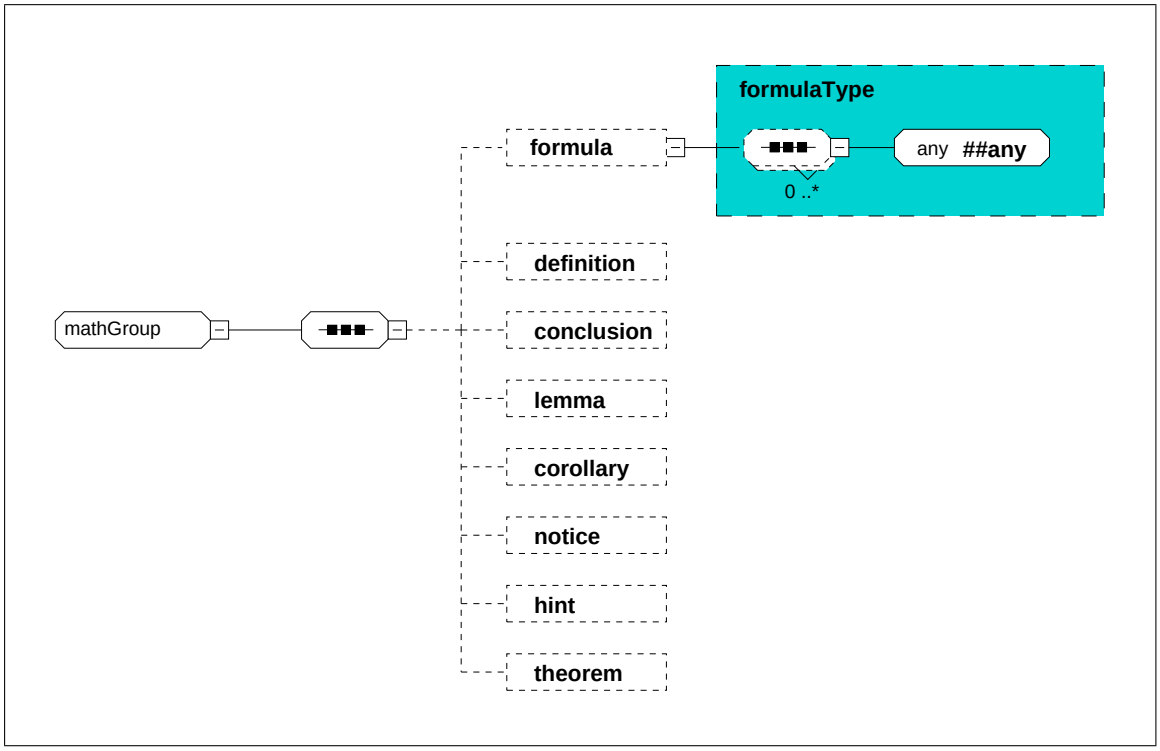


Abbildung 5.10: Mathematikgruppe

durch ein Manifest-Element als Wurzelknoten gekenntzeichnet ist[2]. Die enthaltenen Dateien werden durch `Resource`-Elemente referenziert, die mit dem `Resources`-Element gruppiert werden. Jede dieser einzelnen Ressourcen verweist jedoch nicht nur auf eine Datei, sondern verwaltet mehrere von ihnen. Eine solche Gruppierung ist vor allem dann sinnvoll, wenn mehrere Dateien, wie Abbildungen oder Applets, zu einer HTML-Seite gehören. Gekennzeichnet werden sie als `File`-Element.

Weiterhin können Metadaten zugewiesen werden, die Aufschluss über den Inhalt der Ressource geben und ggf. zur Suche herangezogen werden. Hierbei wird der LOM-Standard (vgl. Kapitel 2.3.3) mit seinen speziell auf Lerninhalte abgestimmten Gruppen verwendet.

Abschließend ist hinzuzufügen, dass jede Ressource optional auf andere Ressourcegruppen verweisen kann, die mit ihr in einer Abhängigkeitsbeziehung stehen, um so die Integrität des Bausteins zu gewährleisten. Abbildung 5.11 verdeutlich diese Zusammenhänge.

Da der technische Rahmen für die formatierbaren Bausteine steht, muss jetzt noch die Schnittstelle zu anderen Bausteinen und Kursen definiert werden. Damit Kursen der Inhalt von Bausteinen korrekt zugeordnet werden kann, reicht die Bereitstellung eines Manifests nicht aus. Vielmehr muss ein Einstiegspunkt festgelegt werden, mit dem der Übersetzer bzw. die Übersetzerin die richtige Ressource auswählen und transformieren kann. Diese Ressource wird mit dem speziellen Attribut `transformable` versehen. Der Standard bietet zwar den generellen Typ `webcontent` an, dieser signalisiert jedoch, dass es

[2] Diese Betrachtung bezieht sich auf die im *XML-Binding* vorkommenden Element-Knoten

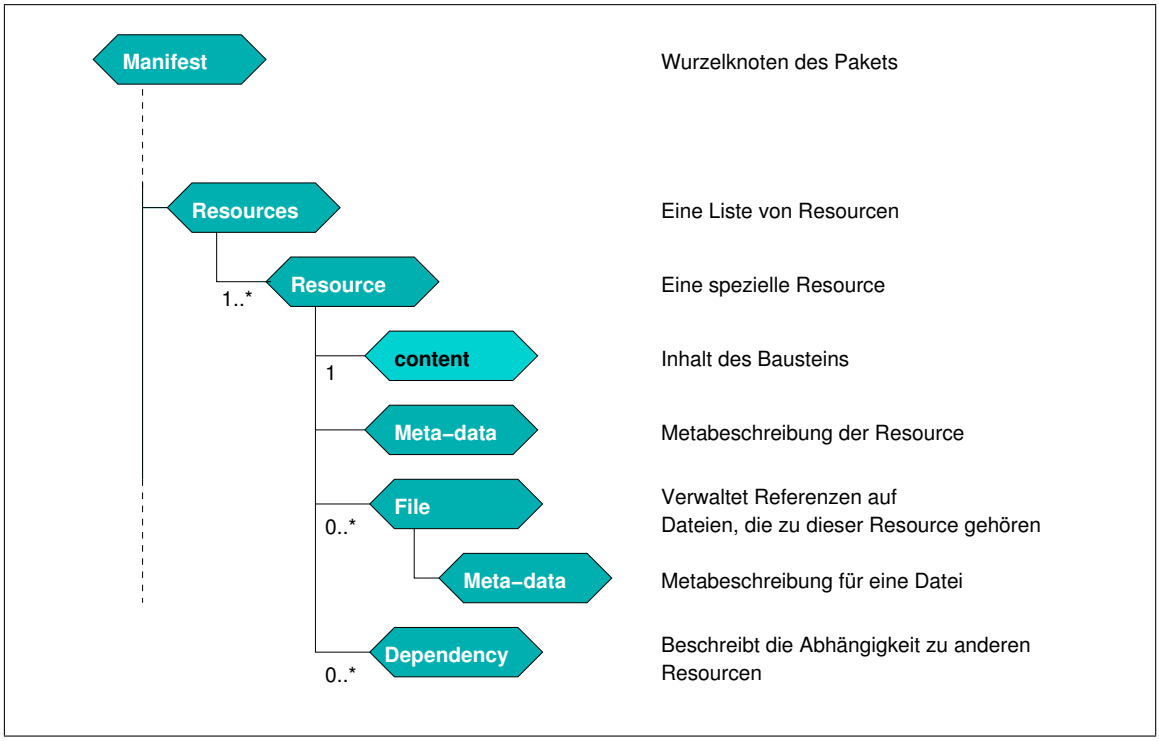


Abbildung 5.11: IMS Content Package Information Model

sich um einen Teil einer Webpräsentation handelt. Durch das Modifizieren dieses Attributes wird während der Erzeugung einer Präsentation das Dokumentmodell erkannt und mit dem zugrundeliegenden Regelwerk weiterbearbeitet.

Mit dem Einsatz des IMS Content Packages und des im vorigen Abschnitt eingeführten Strukturmodells, entsteht eine in sich geschlossene Form, die, unabhängig von anderen Bausteinen, in beliebigen Kursen genutzt werden kann. Hiermit wird der Forderung nachgekommen, Inhalte von Lernmaterialien von ihrer konkreten Darstellung zu entkoppeln und somit einen skalierbaren und orthogonalen Ansatz zu schaffen.

5.3 Formatierbare Kurse

Bisher wurde lediglich der technische Rahmen für Bausteine betrachtet, jedoch nicht deren Aggregation modelliert. Da sie keine Kursstruktur enthalten, sondern ausschließlich Textstrukturen, muss für sie ebenso ein technischer Rahmen geschaffen werden, der die Integration von Bausteinen regelt und zudem ein formatierbares Basisformat besitzt. Aus diesem Grund wir im Folgenden der Begriff *Formatierbarer Kurs* eingeführt und das zugehörige Modell näher erläutert.

> Ein **formatierbarer Kurs (FK)** besteht aus mindestens einem formatierbaren Baustein und einer beliebigen Anzahl Assets und definiert zugleich deren Organisation. Erst in Verbindung mit einem Regelwerk für ein Präsentationsformat entsteht eine nutzbare Kurseinheit. Formatierbare Kurse besitzen einen

definierten Rahmen, der die Aggregation mit anderen formatierbaren Kursen ermöglicht.

Zunächst muss entschieden werden, wie Kurse organisiert sind, da deren Ressourcen sich auf unterschiedliche Arten anordnen lassen. Möglich sind sequenzielle Abfolgen von Lerneinheiten, wie es bei einigen gängigen Autorensystemen der Fall ist (vgl. Abschnitt 3.4), hierarchische Anordnungen, die einer Baumdarstellung entsprechen sowie vernetzte Strukturen, die Querverbindungen zu anderen Lerneinheiten ermöglichen. Wie bereits in den Abschnitten 2.3.1 und 2.3.4 eingeführt wurde, beziehen sich gängige Standards auf hierarchische Kurse, deren Aufbau sich als sinnvolles Strukturierungsmittel durchgesetzt hat. Der nächste Schritt ist die Zuordnung der formatierbaren Bausteine zu der übergeordneten Kursstruktur. Abbildung 5.12 zeigt zur Verdeutlichung einen Kurs, der aus den sechs hierarchisch gegliederten Lernobjekten `A-F` besteht. Die Lernobjekte `B-D` sind direkte Nachfolger des Lernobjekts `A` und sind ihm thematisch untergeordnet. Eine solche Zuordnung wird vorgenommen, wenn z. B. die Lerneinheit *Komplexe Zahlen* in die Einheiten *Einführung*, *Definitionen* und *Exploration* unterteilt ist. Die Lerneinheiten `A-C` enthalten jeweils einen Verweis auf einen formatierbaren Baustein, der den Inhalt des entsprechenden Kursknotens kapselt. Lernobjekt `D` wiederum besteht aus Lernobjekte `E` und `F`, enthält jedoch – im Gegensatz zu den vorher beschriebenen Lernobjekten – keinen eigenen Inhalt. Dieser Kursknoten dient ausschließlich zur Kursstrukturierung und gruppiert thematisch die Bausteine `E` und `F`. Als Beispiel könnte es sich um einen Aufgabenblock handeln, der eine Liste themenbezogener Aufgaben anbietet.

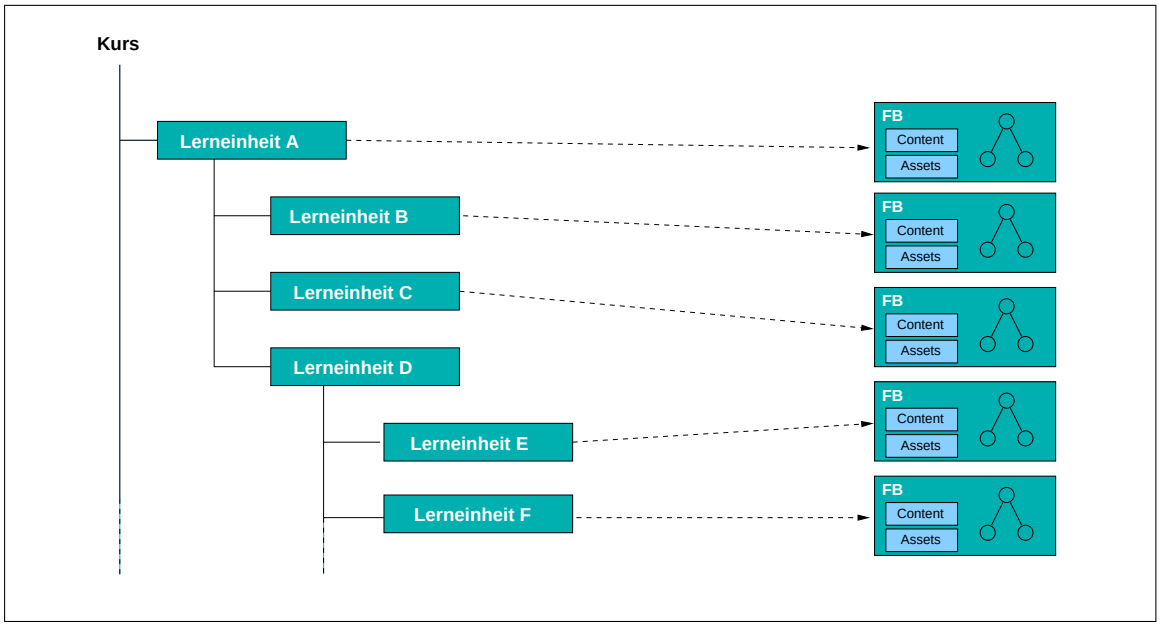


Abbildung 5.12: Formatierbarer Kurs

Die durch die IMS Content Package Spezification festgelegten Bausteine werden als komprimiertes Archiv gespeichert. Dadurch lassen sie sich ohne weiteres aus einer Kursstruktur entnehmen und in andere Strukturen einbinden, was vor allem den Einsatz von

Lernobjekten anderer Autoren begünstigt. Da Bausteine über keine Layoutinformationen verfügen, passen sie sich nahtlos an übergeordnete Kurse an.

Die hier vorgestellte Struktur soll als allgemeines Schema verstanden werden, das sich nach Bedarf mit verschiedenen Techniken implementieren lässt. Eine Variante ist die Abbildung der Kursstruktur auf eine Relationale Datenbank, bei der zu jedem Knoten ein entsprechender Tabelleneintrag vorgenommen wird. Hierarchische Strukturen können über Fremdschlüssel-Beziehungen zwischen den jeweiligen Knoten hergestellt werden. Ferner ist auch die Verwendung eines *XML-Bindings* denkbar, das die Kursstruktur in einem XML-Manifest speichert. In Abhängigkeit von der präferierten Lösung, ist eine Umsetzung mit der in Kapitel 2.3.2 vorgestellten IMS Content Package Spezification möglich, wie es in Abbildung 5.11 illustriert wird. Die eingebundenen Ressourcen sind jedoch keine Lernmaterialien, sondern formatierbare Bausteine. In einem Kurs-Manifest lassen sich mehrere Organisationsgerüste anlegen, wobei jedes einen eigenen Baum verschachtelter `item`-Elemente enthält. Diese verschachtelte Struktur entspricht der Kursstruktur und enthält Referenzen auf formatierbare Bausteine.

Name	Erklärung	Benötigt	Typ
Item	A node that describes the shape of the organization.	M	Container
dentifier	An identifier, for the Item, that is unique within the Manifest file.	M	ID
IdentifierRef	A reference to an identifier in the resources section or a (sub)Manifest	O	String (2000)
Title	Title of the item.	O	String (200)
IsVisible	Indicates whether or not this item is displayed when the structure of the Package is displayed or rendered.	O	Boolean
Parameters	Static parameters to be passed to the resource at launch time.	O	String (1000)
Item	A sub-node within this organization.	O	Container
Meta-data	Meta-data describing this item.	O	Container

Tabelle 5.3: Das Item Element der IMS Content Package Spezifikation

Tabelle 5.3 zeigt einen Auszug aus dem *Information Model* zur Charakterisierung des `Item`-Elements. `Item`-Elemente besitzen die Attribute `Identifier`, `IdentifierRef`, `IsVisible` und `Parameters`. Mit dem Attribut `IdetifierRef` wird einem Item eine Ressource zugeordnet, indem auf den Identifier einer beliebigen, im Manifest beschriebenen Ressource, verwiesen wird. Zudem dient das Item-Element als Container für weitere Item-Elemente und kann mit LOM-Metadaten (vgl. Kapitel 2.3.3) angereichert werden. Der Vorteil, der sich aus dieser Struktur ergibt, ist die gewünschte layoutfreie Beschreibung von Kursstrukturen, aber auch die erlangte Standardkompatibilität mit Learning Management Systemen (vgl. Kapitel 3.3), die immer mehr Standards unterstützen und entsprechende Mechanis-

men zur adäquaten Anzeige implementieren. Jedes LMS kann die Darstellungen seiner eingebundenen Kurse individuell auf seine Web-Darstellung anpassen und in das Präsentationskonzept einbauen.

5.3.1 Granularität

Genau wie formatierbare Bausteinen sollen auch Kurse bzw. Teile von ihnen wiederverwendet werden. Eine solche Modularität ist vor allem dann notwendig, wenn die Bausteine eines zu entnehmenden Kursabschnitts feingranular konstruiert sind. In diesem Fall müssen neben formatierbaren Bausteinen auch *formatierbare Kurse (FK)* in eine Kursstruktur eingebunden werden. Da Kurse – genau wie Bausteine – dem IMS-Standard folgen, lassen sie sich ebenso als Archiv mit zugehörigem Manifest speichern. Dieses Archiv kann dann als Ressource genutzt werden und lässt sich von `item`-Elementen eines übergeordneten Kurses referenzieren. Diese Herangehensweise ermöglicht eine flexible Festlegung der Granularitätsstufen einzelner Subkurse und erhöht damit die Modularität des Konstruktionsprozess. Abbildung 5.13 veranschaulicht diese Zusammenhänge. Der formatierbare Kurs A besteht aus den Lektionen 1 und 1.2, wobei die letztere der ersten untergeordnet ist. Beide Lektionen verweisen auf Dateien, wobei Lektion 1 auf einen Baustein und Lektion 1.2 auf einen Kurs verweist. Wird Baustein 1 geöffnet, so enthält er zwei weitere Dateien, von denen die erste den Inhalt des Bausteins in Form einer XML-Datei beschreibt und die zweite eine Abbildung. Die Archiv-Datei von Kurs B speichert hingegen einen Kurs mit einer eigenen Item-Struktur und den referenzierten Baustein 2. Dieser kapselt wieder eine XML-Datei, die den Inhalt des formatierbaren Bausteins enthält.

Aufgrund der separat gekapselten Strukturinformationen können formatierbare Kurse und deren Bausteine durch Auflösung der Referenzen aus einem Kurs seziert werden. Damit Kurse als solche erkannt werden, müssen sie, wie bereits die Bausteine, einen eigenen Typ besitzen. Die Typdefinition erfolgt in Anlehnung an die Typzuweisung der XML-Dokumente innerhalb von Bausteinen. Für Kurse und Bausteine werden die Typen `fk` und `fb` verwendet. Sie werden während der Erzeugung eines Zielformats herangezogen und legen fest, ob es sich um eine Kursstruktur handelt oder einen zu transformierenden Baustein.

5.3.2 Abstrakte Kursmodell

Nachdem die formatierbaren Bausteine und Kurse eingeführt wurden, besteht eine solide Grundlage, auf der darstellungsunabhängige und modulare Kurse entwickelt werden können. Dieses Modell eignet sich zur Entwicklung von Teilkursen, die in andere Kurse eingebettet werden können. Ein solches Maß an Flexibilität und Orthogonalität führt jedoch zwangsläufig zu einem komplexeren Benutzungsmodell. Kurse werden nicht mehr als ein gesamtes Werk erstellt, sondern in kleinen Teilstücken konzipiert und bearbeitet. Darüber hinaus sollen bereits für die Lehre eingesetzte Materialien und bestehende Dokumente wiederverwendet werden. Aufgrund der bestehenden Formatvielfalt, wie z. B. *Latex*, *Microsoft PowerPoint* oder *Microsoft Word*, ist es sinnvoll, eine Abstraktion der formatierbaren

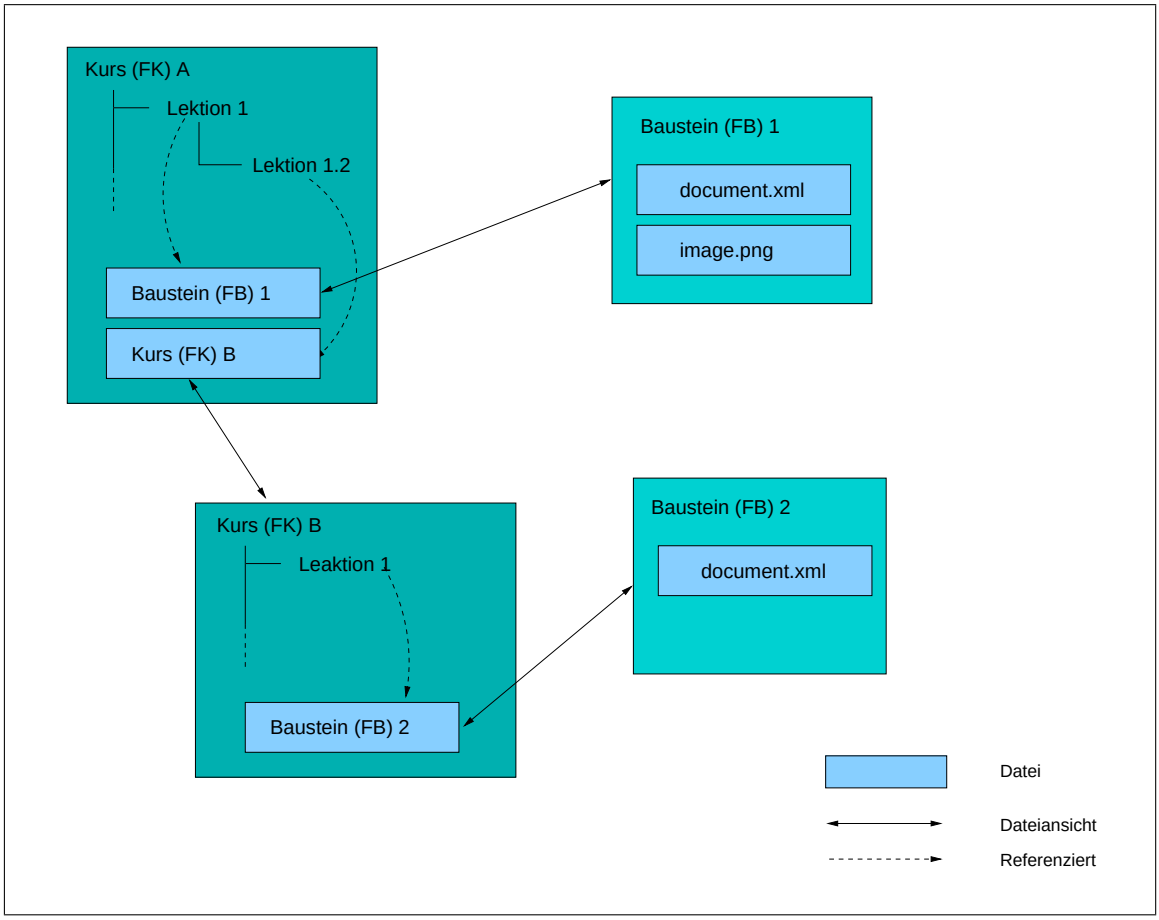


Abbildung 5.13: Verschachtelte Kurse

Kurse zu finden, die die konkrete modulare Kurskonstruktion verbirgt. Die vorliegenden Formate unterscheiden sich hinsichtlich ihrer internen technischen Struktur sowie in ihrem eigentlichen Verwendungszweck. Sollen sie auf das in dieser Arbeit definierte Kursmodell abgebildet werden, muss eine einheitliche Schnittstelle definiert werden, so dass während des Abbildungsprozesses ausschließlich die Analye des Quellformats im Vordergrund steht. Aus diesem Grund wird das *Abstrakte Kursmodell* eingeführt (vgl. [Baudry04b]), welches die Bearbeitung der vollständigen Kurssturktur, inklusive aller Texte und multimedialer Komponenten, ermöglicht und dabei kein explizites Wissen über verschachtelte Bausteine oder Standards voraussetzt. Zu diesem Zweck wird eine komplette Baumstruktur erzeugt, die sowohl die Kurshierarchie als auch den Lerninhalt enthält.

Abbildung 5.14 zeigt den schematischen Aufbau des Kursmodells[3]. Jedes Kursmodell besitzt als Wurzelknoten das `course`-Element, das sich nach Bedarf mit LOM-Metadaten anreichern lässt. Direkt unter dem Kurs-Element werden rekursiv `lesson`-Elemente definiert, die später der Kursstruktur entsprechen. Sie können entweder aus der in Abschnitt 5.2.1 vorgestellten Dokumentstruktur mit Elementen aus den Gruppen `textGroup`, `struc-`

[3]Es handelt sich bei dieser Abbildung keineswegs um die vollständige Darstellung des Modells, sondern vielmehr um das Fragment einer Organisation, das ausschließlich zur Vorstellung dient.

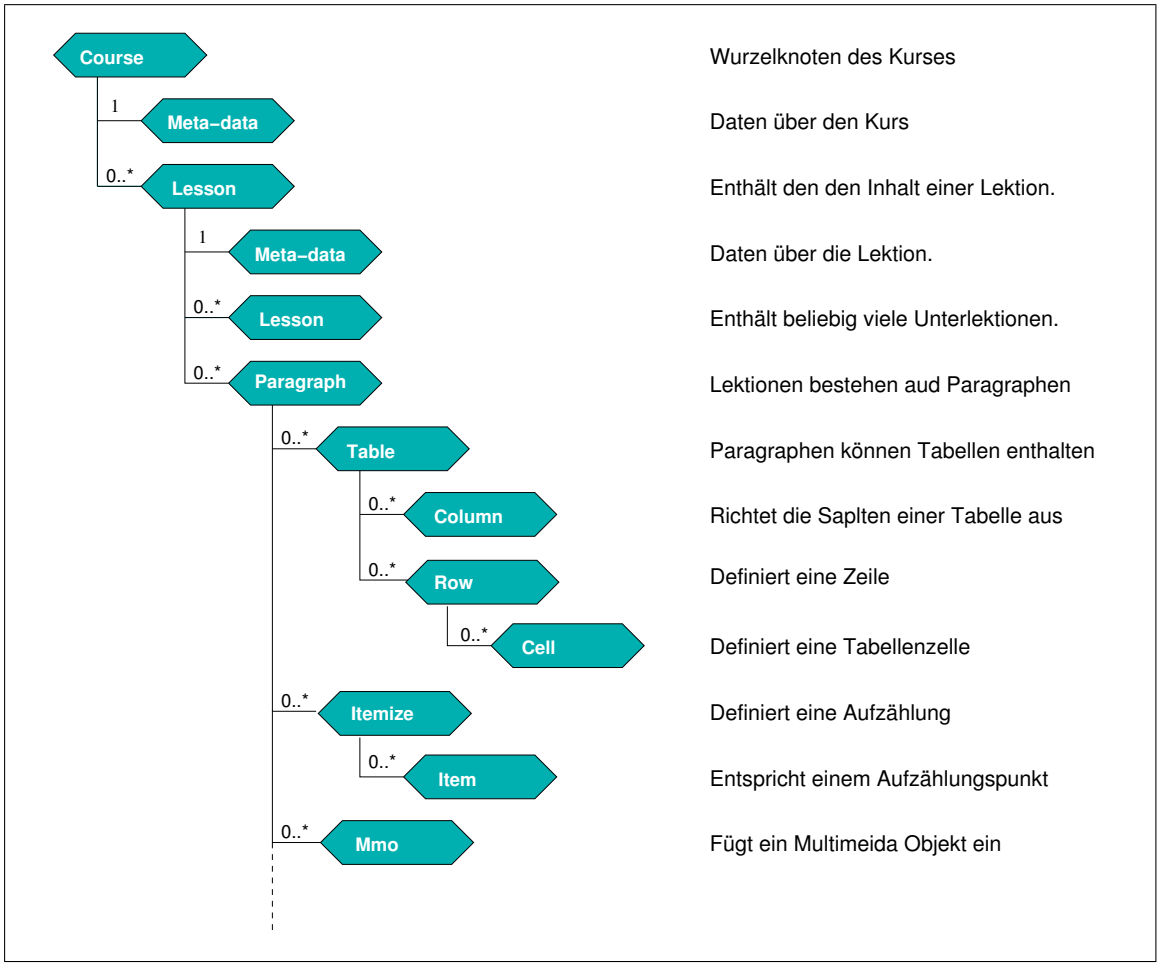


Abbildung 5.14: Kursmodell Spezifikation

`tureGroup` und `mediaGroup` bestehen oder als einfacher Knoten zur Gruppierung verwendet werden. In Abbildung 5.14 wird zur Veranschaulichung ein Abschnitt mit Tabellen, Aufzählungen und Multimedia Objekte dargestellt. Die Verschachtelungsstruktur des Kurses geht jedoch nicht verloren, weil sie aus dem Kursmodell eindeutig abgeleitet wird. In der Regel beschreibt jede Lektion den Inhalt eines Bausteins. Treten jedoch starke Abhängigkeiten zwischen Bausteinen auf oder weisen sie einen derart abgeschlossenen Inhalt auf, dass sie sich in andere Kurse einbauen lassen, kann eine Lektion als Subkurs definiert werden. Hierzu beschreibt Tabelle C.2 die bereits in Abbildung 5.14 vorgestellten Elemente. Das Attribut `courseObject` (1.2.1) entscheidet, ob es sich bei der zugeordneten Lektion tatsächlich um einen Baustein oder um einen Teilkurs handelt.

Die Abbildung auf einen formatierbaren Baustein erfolgt, indem die Unterstruktur der Lektionen übernommen wird und als neues Dokument in das Datei-Archiv des formatierbaren Bausteins kopiert wird. Des weiteren müssen die in der Lektion verwendeten Dateien in den Baustein kopiert werden. In Tabelle C.2 besitzt das `MMO`-Element (1.2.8) hierfür das `href`-Attribut, mit dem ein assoziertes Asset eingebunden werden kann.

Im Gegensatz zu einem Kurs benötigt ein Baustein immer einen festen Bezugspunkt

um seine Assets abzulegen und zu verwalten. Ein solcher Bezugspunkt ist stets ein übergeordneter formatierbarer Kurs. Diese Abhängigkeit ist deshalb wichtig, weil der fachliche Zusammenhang sich zwangsläufig auf die verwendeten Assets überträgt und sich bei einer Disaggregation entsprechend auf die beteiligten Datein der Unterlektionen auswirkt. Abbildung 5.15 veranschaulicht exemplarisch diesen Zusammenhang. Ein abstrakter Kurs wird genau einem formatierbaren Kurs zugewiesen. Wird dem Kurs dynamisch die erste Lektion hinzugefügt, kann entweder ein Subkurs oder ein Baustein erzeugt werden, der als Teil des formatierbaren Kurses angelegt wird. Lektion 1 besitzt den Parameter `courseObject` mit dem Wert `true` und wird deshalb dem Subkurs `Kurs 2 (FK)` zugeordnet. Letzterer dient als Container für weitere untergeordnete Lektionen. Lektion 1.1 und 1.2 werden deshalb jeweils auf `Baustein 1 (FB)` und `Baustein 2 (FB)` abgebildet, wobei die zu Lektion 1.2 gehörende Dokumentstruktur, bestehend aus einem Paragraphen, einem Textblock und einem Video, ebenfalls `Baustein 2 (FB)` zugeordnet werden. Im Gegensatz zu Lektion 1 ist der Parameter `courseObject` von Lektion 2 auf den Wert `false` gesetzt und bewirkt, ebenso wie bei Lektion 1.1 und Lektion 1.2, die Erzeugung eines einfachen Bausteins. Baustein 3 besitzt als Bezugspunkt wieder `Kurs 1 (FK)`. Durch die Änderung des Parameters `courseObject` können jetzt komplexe, rekursive Strukturen definiert und modifiziert werden.

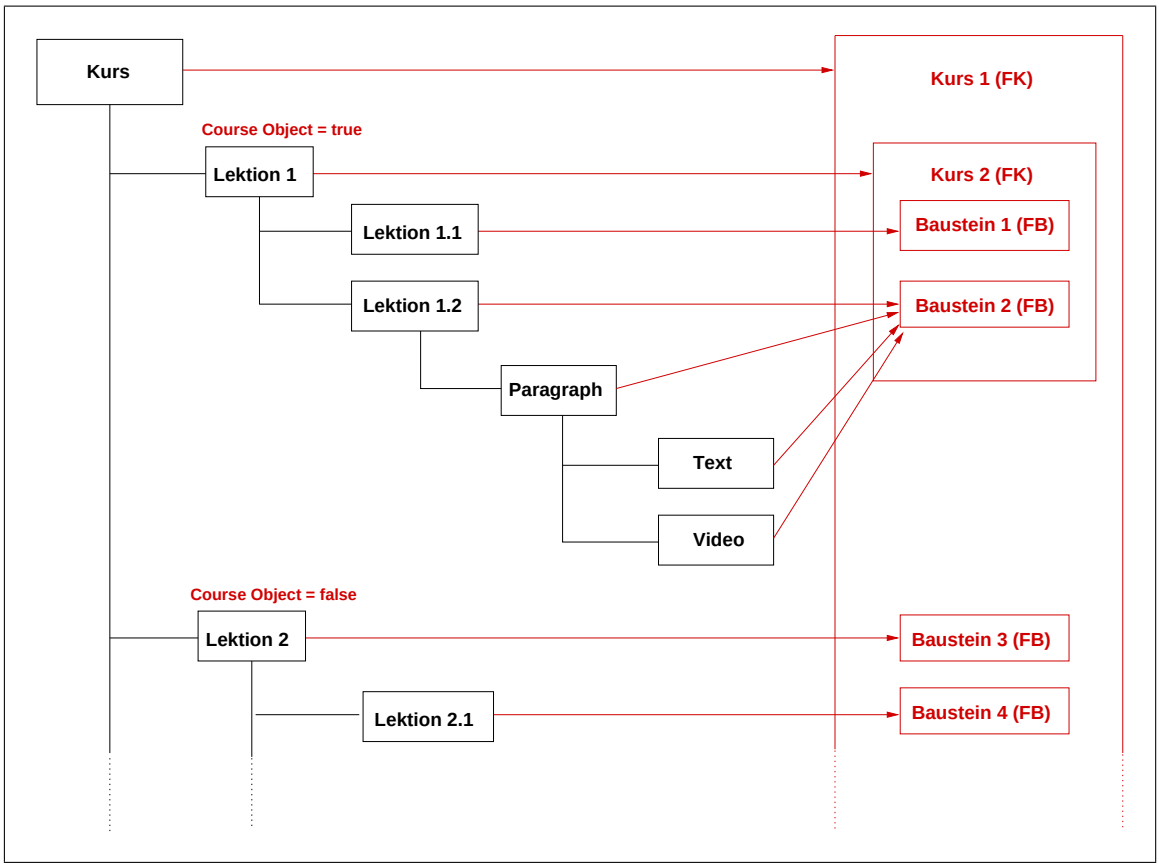


Abbildung 5.15: Abbildung eines Kurses auf die Bausteinstruktur

5.4 Abbildung von Lernmaterialien

Bei der Entwicklung von E-Learning-Inhalten werden zur Anfertigung neuer Kurse häufig bestehende Skripte, Präsentationsfolien und andere Kursmaterialien herangezogen. Jedoch weisen diese Inhalte grundlegende Unterschiede sowohl in ihren inhaltlichen Strukturen als auch in ihrem Format auf. Einerseits existieren Skripte, die in Vorlesungen studienbegleitend eingesetzt werden und Studierenden eine solide Grundlage bieten, sich in bestimmte Themen einzuarbeiten und anderseits existieren Präsentationsfolien, anhand derer ein Überblick über ein bestimmtes Themengebiet während einer Präsenzveranstaltung vermittelt wird. Der Unterschied liegt hierbei in der inhaltlichen Ausarbeitung der Folien, die bei der Verwendung in Präsenzveranstaltungen, im Gegensatz zu Skripten, häufig Aufzählungscharakter aufweisen. Für die Realisierung eines neuen Kurses werden meistens keine vollständigen Skripte oder Foliensätze benötigt, sondern vielmehr einzelne Textpassagen oder Abbildungen. Deshalb wird in diesem Abschnitt eine Lösung präsentiert, anhand der beliebige Lehrinhalte auf das in Kapitel 5.3 vorgestellte modulare Kurskonzept übertragen werden können. Ausgangspunkt ist das zuvor eingeführte Abstrakte Kursmodell, welches als allgemeine Schnittstelle für Kursmaterialien verstanden wird, mit der modulare und darstellungsfreie Kurse konstruiert werden können, ohne die zugrunde liegende technische Realisierung berücksichtigen zu müssen.

5.4.1 Analyse

Das grundlegende Problem, das bei der Konzeption auftritt, ist der Umgang mit der Formatvielfalt, die bestehende Materialien aufweisen. Mit dem Begriff Format ist hier die technische Realisierung eines Zwischenformats gemeint, mit dem Materialien abgespeichert und ausgetauscht werden. So wird in der Regel die Präsentation eines Vortags mit MS PowerPoint-Folien durchgeführt, wogegen Skripte häufig mit *MS Word*, *Latex* oder *Framemaker* erstellt werden. Jedes dieser Formate besitzt individuelle Eigenschaften und Vorzüge. Die MS Office-Formate Porwerpoint und MS Word werden in einem Binärformat kodiert, in dem alle Daten – selbst Abbildungen und Videos – eingebettet werden. Latex-Dokumente werden hingegen textuell kodiert und verweisen auf externe Daten, wie z. B. Bilder, die dem Dokument im ursprünglichen Format beiliegen. Abbildung 5.16 zeigt auf der linken Seite die Autorenwerkzeuge *MS-Word*, *OpenOffice*, *XML-Editoren* und *Lyx*. Jedes dieser Autorenwerkzeuge verwaltet seine Ressourcen auf unterschiedliche Art. *MS-Word* speichert seinen Dokumentinhalt in einem proprietären Binärformat in einer einzelnen Datei ab. *OpenOffice* verwendet eine auf mehrere Dateien ausgelegte Dateistruktur, die als komprimiertes ZIP-Archiv bearbeitet wird. *XML-Editoren* speichern Inhalte in XML-Dokumenten, wobei das jeweilige Format der eingesetzten Ressource beibehalten wird. Dieses Beispiel zeigt eine Bild-Referenz auf eine PNG-Datei. Der letzte Vertreter der Bearbeitungswerkzeuge ist der WYSIWYG-Editor *Lyx*. Er wurde speziell für Latex-Dokumente konzipiert, und abstrahiert von der textuellen Gestaltung der Latex-Dokumente. Das Abspeichern hinzugefügter Ressourcen erfolgt bei dieser Variante ebenfalls extern.

Es unterscheiden sich jedoch nicht nur die technischen Formate, sondern auch der in-

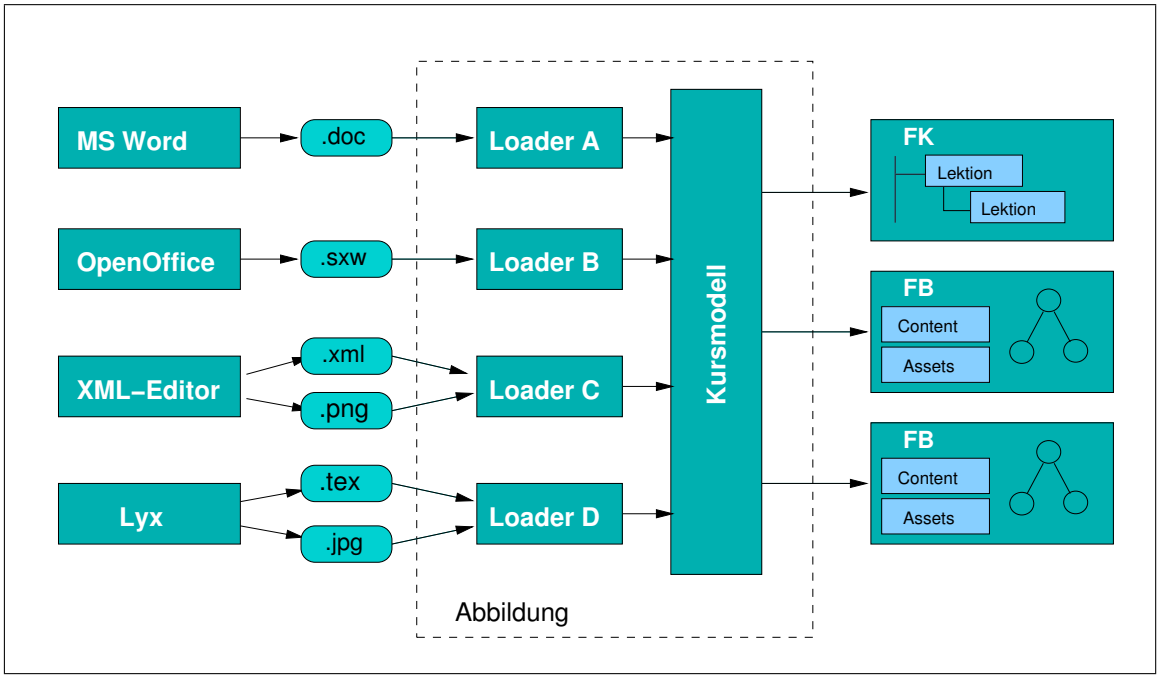


Abbildung 5.16: Wiederverwendung von Lernmaterialien

terne Dokumentaufbau. Bei *Latex* werden Dokumente hierarchisch, d. h. aus Kapiteln und Abschnitten, aufgebaut. Das Dokument wird ohne feste Bindung an ein Layout definiert und speichert es in sogenannten *Style*-Dateien. Im Gegensatz hierzu erfolgt bei MS-Word bzw. OpenOffice-Dokumenten die Definition des Layouts während der Dokumenterstellung. Hieraus folgt jedoch, dass derart gestaltete Dokumente aus einer Sequenz formatierter Texte bestehen und keineswegs hierachisch strukturiert sind. Überschriften werden als *normale* Texte gespeichert, die zur Darstellung den Typ *Überschrift* mit einer zugeordneten Verschachtelungstiefe verwenden [OOS02]. Die abzubildenden Materialien können jedoch noch andere Strukturen enthalten. Mit MS-PowerPoint werden Foliensequenzen erstellt, die Abbildungen, Videos, kleine Textabschnitte und Aufzählungen enthalten. Damit jedoch möglichst viele Formate auf formatierbare Kurse abgebildet werden können, soll das in Abschnitt 5.3.2 vorgestellte abstrakte Kursmodell eingesetzt werden. Dieses Modell erleichtert, bedingt durch seine einfache Struktur, die Konvertierung bestehender Materialien. Ohne dieses Modell müsste jedes Format, entsprechend seiner Spezifikation, verarbeitet werden und eine komplexe verschachtelte Kurstruktur erzeugt werden, die aus formatierbaren Bausteinen und Kursen besteht. Aus diesem Grund soll hier eine strikte Trennung zwischen dem *Loader*, für ein spezielles Format, und dem Kursmodell stattfinden. Diese Trennung bewirkt, dass sich die Implementierung eines *Loaders* komplett auf die technischen Gegebenheiten des zu analysierden Formats bezieht und dabei der komplexe Abbildungsprozess auf formatierbare Kurse nicht berücksichtigt werden muss. Der Vorteil liegt auf der Hand, denn die Implementierung bewirkt, bedingt durch die geringere Komplexität des *Loaders*, eine Kosteneinsparung und erhöht zugleich die Wartbarkeit. Abbildung 5.16 zeigt, dass für jedes Format ein eigener austauschbarer Loader implementiert

werden kann.

5.4.2 Abbildung auf das Kursmodell

Die Loader sind für die Abbildung eines Formats auf das Kursmodell zuständig. Wie bereits im vorigen Kapitel ausgearbeitet wurde, verwenden die abzubildenden Formate verschiedene technische Realisierungen. Wird z. B. ein *Loader* für *OpenOffice* implementiert, muss zunächst, für die Anaylse der Datenstrukur, ein XML-Parser vorgeschaltet werden. Wird hingegen ein Flash-Film abgebildet, muss die Binär-Datei Bit für Bit eingelesen werden, weil es sich hierbei um ein Format handelt, bei dem es vor allem auf eine – durch die Übertragung im Internet bedingte – komprimierte Datenhaltung ankommt. Sobald die technischen Hürden überwunden sind, muss das Dokument in seine Grundbestandteile *Struktur*, *Metdadaten*, *Inhalt*, *E-Learning Komponenten*, *Assets* und *Layout* zerlegt werden. Eine solche Separation wird in Abbildung 5.17 illustriert.

Der Begriff *Struktur* beschreibt den bereits im vorigen Abschnitt erwähnten Materialaufbau. Strukturen sind hierarchisch oder sequenziell aufgebaut und werden sukzessiv auf die Lektionen des Kursmodells übertragen. Als Beispiel dient ein Office-Dokument, in dem eine Überschrift erkannt wird und der Titel als Name für die zu erstellende Lektion fungiert. An dieser Stelle kann der Lektion mitgeteilt werden, ob es sich um ein umfangreiches und abgeschlossenes Thema handelt und es somit auf einen Subkurs abgebildet werden soll oder eher auf einen einfachen Baustein.

Materialien verfügen in der Regel über unterschiedlich viele Metadaten, die sich in die Gruppen der *objektiven* und *subjektiven* Metadaten aufteilen lassen. Objektive Metadaten sind solche, die eindeutig festgelegt werden können, wie z. B. das Erstellungsdatum und die im Anschluss protokollierten Lebenszyklen. Subjektive Metadaten werden hingegen vom Autor bzw. von der Autorin definiert. Es können anvisierte Lernziele oder definierte Nutzungsrechte an einem Material sein. In Abschnitt 2.3.3 wurde bereits das vom Kursmodell unterstützte Metadaten-Modell LOM eingeführt. Dieses Kursmodell ermöglicht die Definition der Metadaten bei Lektionen und Ressourcen, die mit externen Datenquellen verknüpft werden. In Abhängigkeit von dem jeweiligen Kontext der Lektion, sind Matadaten fester Bestandteil eines Bausteins oder Kurses.

Inhalte bestehen im Allgemeinen aus Grundelementen wie Tabellen, Aufzählungen, Nummerierungen, Abbildungen und Animationen. Sie werden durch den *Loader* erkannt und mit der Modellierungssprache BrickML einer Lektion zugeordnet. Eine allgemeingültige Schnittstelle kann jedoch nicht alle Eigenschaften eines Formats detailliert abbilden. So gibt es beispielsweise für Kopfzeilen, die in Office-Formaten üblich sind, keine eindeutige Entsprechung, da sie zwar in Office-Dokumenten sinnvoll sind, sich jedoch nicht auf beliebige Ausgabeformate übertragen lassen. Vielmehr ist die Erzeugung einer Kopfzeile, mit Seitenzahlen und Überschriften, abhängig von dem zu generierenden Format und wird im nächsten Abschnitt weiter erörtert.

Im E-Learning-Kontext werden Quiz-Umgebungen gern zur Wissensüberprüfung herangezogen. Lernobjekte können in Kombination mit einem LMS gestartet und durch das LMS ausgewertet werden. Damit lassen sich z. B. zeitliche Limitierungen für eine Ler-

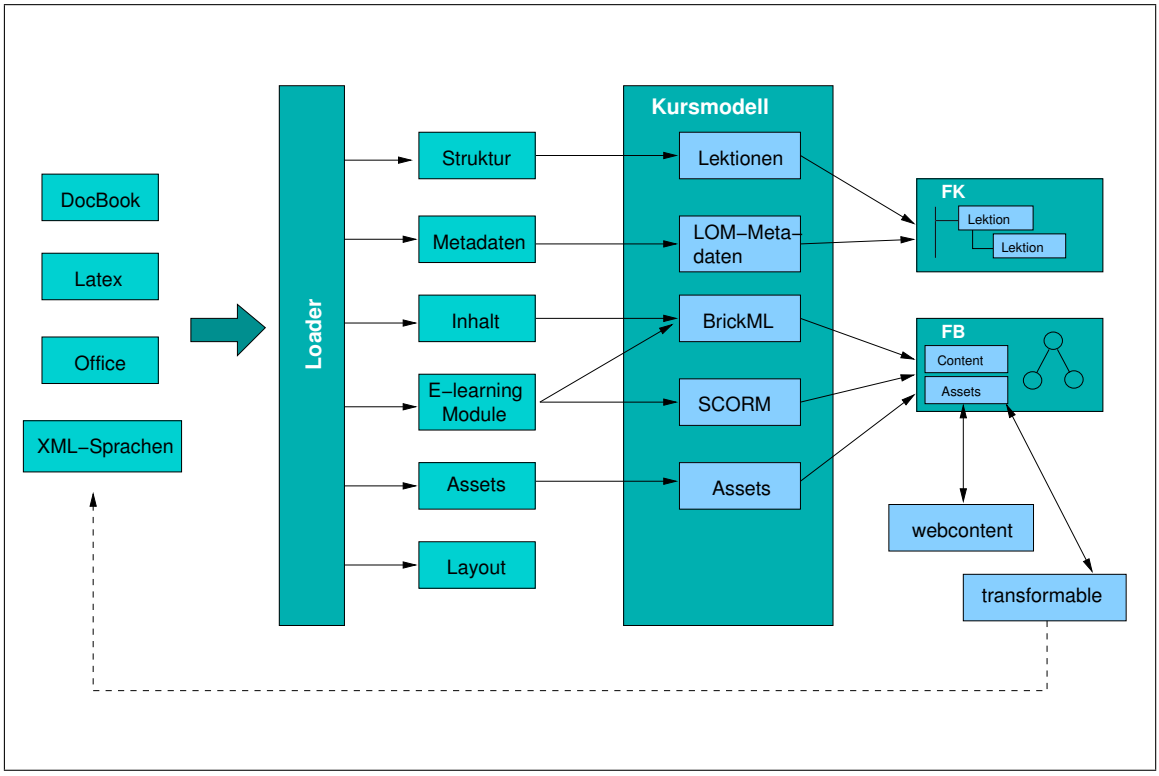


Abbildung 5.17: Abbildung auf das Kursmodell

neinheit festlegen oder Punktekontos für Studierende führen. Diese Eigenschaften sind für ein allgemeines Kursmodell unerlässlich und müssen auf dieses abbildbar sein. Quiz-Umgebungen lassen sich auf das in Abschnitt 5.2.1 beschriebene Dokuementenmodell übertragen, wobei die *Funktionalität* von Lernobjekten auf den SCORM-Standard (vgl. Abschnitt 2.3.1) zurückzuführen ist.

Die in das abzubildende Format eingebetteten Daten sowie die von ihnen referenzierten Assets, müssen vom Loader erfasst und dem Kursmodell übergeben werden. Die Zuordnung der Daten zu einem formatierbaren Baustein erfolgt auch hier kontextsensitiv, indem der einem Baustein zugeordnete Kurs wieder als Bezugspunkt für die Dateizuordnung dient. Trifft der Loader während der Verarbeitung des Quell-Dokuments jedoch auf dokumentspezifische Formatierungsregeln, so dürfen diese nicht weiterbearbeitet werden. Die vorhandenen Formatierungen müssen genau an diesen Stellen vom Inhalt getrennt werden, damit neue, separate Formatierungsregeln den zu erzeugenden formatierbaren Bausteinen zugewiesen werden können.

Darüber hinaus kann das Kursmodell zu einem anderen Zweck verwendet werden. Mit Hilfe des Kursmodells und der damit verbundenen Abbildungsmechanismen, können auch Fremdformate als integraler Bestandteil der formatierbaren Bausteine eingesetzt werden. Dieses ist vor allem dann sinnvoll, wenn ein Dokument in seinem Quell-Format weiterbearbeitet werden soll und erst bei der Generierung des Ziel-Formats zu einem Baustein oder Kurs konvertiert wird. So kann z. B. ein Office-Dokument den Inhalt eines Bausteins

repräsentieren, der fortlaufend modifiziert wird. Dieses hat zur Folge, dass der, aus dem Office-Dokument erzeugte, Subkurs an Stelle des Bausteins in den formatierbaren Kurs eingebunden und anschließend, in einem zweiten Verarbeitungsschritt, in das gewünschte Format übersetzt wird. Abbildung 5.17 zeigt die Typen `webcontent`, `transform` und `map`. Dateien, die als abbildbar gekennzeichnet sind, werden mit dem entsprechenden Loader wieder auf einen verschachtelten Kurs abgebildet.

5.4.3 Nachbearbeitung

Der Abbildungsprozess von Lernmaterialien setzt einen logisch strukturierten Kurs voraus. Dieser kann jedoch bei der Nutzung bereits erstellter Materialien nicht zwingend vorrausgesetzt werden. Handelt es sich bei der Quelle um ein Dokument, das nicht aus abgeschlossenen Kapiteln besteht und sämtliche Verweise auf andere Textstellen enthält, ist eine Segmentierung in eigenständige Objekte äußerst schwierig und Kurse werden schnell zu großen 'Klumpen', die sich nicht mehr austauschen lassen. Ist jedoch Wiederverwendung das langfristige Ziel, so müssen generierte Kurse überarbeitet werden. Der für eine Modularisierung notwendige Aufwand hängt naturgemäß vom Quellmaterial ab. Nun lässt sich einfach sagen: "Das Dokument muss überarbeitet werden!", aber wie sollte damit begonnen werden und wie können Lernbausteine überhaupt seiteneffektfrei überarbeitet werden? Ähnliche Probleme treten bei der objektorientierten Programmierung und bei der Komponenten orientierten Programmierung auf [Griffel98, Szyperski97]. Treten nach der Abbildung vom fachlichen auf das technische Modell Änderungen an den Schnittstellen der Objekte auf, oder sollen schlecht strukturierte Objekte im Sinne des Paradigmas verbessert werden, so wird im Allgemeinen von *Refactoring* [Fowler99] gesprochen. Einige Mechanismen und Maßnahmen, die bei der Software-Entwicklung [Pomberger96], zu guten und anspruchsvollen Lösungen führen, können durch leichte Modifikationen auf die Überarbeitung von Lernobjekten angewendet werden. Zunächst müssen jedoch die Unterschiede bzw. Gemeinsamkeiten zwischen Lernobjekten und Objekten der objektorientierten Programmierung (OOP) [Taylor96] herausgearbeitet werden. Ein wesentlicher Aspekt in der OOP ist die *Vererbung*. Sie ermöglicht es Objekten die Funktionalität von anderen Objekten zu übernehmen und selbst anzubieten. Des weiteren kennt das Paradigma eine zweite Objektrelation, die *Aggregation*. Sie stellt im Wesentlichen eine Benutztbeziehung zwischen zwei Objekten dar. Lernobjekte besitzen diese Formen der Relation nicht, werden aber thematisch anderen Lernobjekten untergeordnet. Die nachfolgenden Regeln werden dahingehend adaptiert, dass der Sinn und Zweck auf die hierarchische Beziehung zwischen Lernobjekten übertragen wird. Ein weiterer Unterschied bezieht sich auf die in der OOP vorkommenden Schnittstellen und der damit verbundenen Datenkapselung. Methoden erweitern das Angebot von Objekten, ähnlich wie es Lernabschnitte in einem Lernobjekt tun. Die Regeln für das *Refactoring* von Objekt-Methoden wird also auf die Bearbeitung von Textabschnitten projiziert. Im Folgenden werden wichtige Maßnahmen und Regeln vorgestellt, die speziell für die Nachbearbeitung von Lernobjekten angepasst worden sind.

Extract Block: Separiere einen Textabschnitt innerhalb des Bausteins und eleminiere alle auftretenden Bezüge.

Extract Learning Object: Verschiebe einen Textabschnitt, der einen semantisch anderen Bezug hat, in einen neuen, eigenständigen Baustein.

Move Block: Verschiebe einen zusammengehörenden Textabschnitt in einen anderen Baustein.

Collapse Hierarchy: Entnehme den Inhalt eines Bausteins und füge ihn in den Inhalt des in der Kursstruktur übergeordneten Bausteins ein.

Extract Hierarchy: Teile den Inhalt eines Bausteins auf mehrere Bausteine auf und ordne sie als Unterbausteine in die Kursstruktur ein.

Remove Middelman: Entferne einen Kursknoten, der ausschließlich zur Gruppierung anderer dient und verteile dessen Inhalt auf seine Unterbausteine.

5.5 Transformation

Bisher wurden Modelle vorgestellt, mit denen Kurse darstellungsfrei entwickelt werden können. Diese Kurse sind jedoch in der bis jetzt definierten Form nicht für einen Einsatz mit Studierenden geeignet, weil für die Übersetzung auf ein spezielles Format das Regelwerk fehlt. In diesem Abschnitt wird erläutert, wie die modulare Kursstruktur prozessiert wird und die formatierbaren Bausteine und Kurse für die Generierung eines speziellen Formats dabei herangezogen werden.

5.5.1 Aggregation

In Abbildung 5.13 wurde bereits gezeigt, dass Bausteine in Kursen abgelegt werden, die wiederum weitere Subkurse enthalten können. Abbildung 5.18 verdeutlicht die Aggregationsbeziehung zwischen Kursen und Bausteinen.

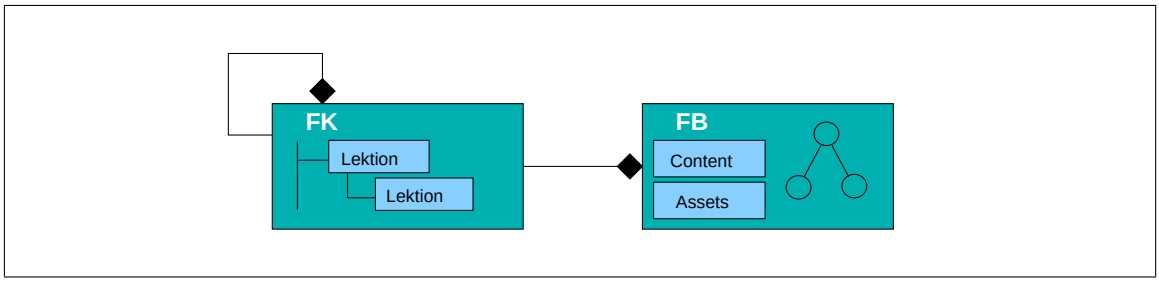


Abbildung 5.18: Kursaggregation

Damit ein solcher Kurs übersetzt werden kann, muss er in eine für Studierende lesbare Form gebracht werden. Hierbei wird die Kurs- bzw. Bausteinaggregation durchlaufen und jeder Baustein übersetzt. Anschließend muss anhand der Zwischenergebnisse das gewünschte Format erzeugt werden. Kurse verwenden die in Abschnitt 5.3.1 eingeführten Ressource-Typen `webcontent`, `fk` und `fb`. Ressourcen, die mit `webcontent` gekennzeichnet sind, werden während der Übersetzung umkopiert und nicht weiterverarbeitet. Tritt

einer der beiden Typen `fk` oder `fb` auf, so handelt es sich bei dem Asset um ein Paket, das weiteren Inhalt aggregiert. Diese Pakete müssen – je nachdem, ob es sich um einen Subkurs oder einen Baustein handelt – rekursiv weiterverarbeitet werden, wobei das Resultat in den aktuellen Kurs eingehängt wird. Demgegenüber wird der Inhalt von Bausteinen in das gewünschte Zielformat transformiert. Die Assets eines Bausteins lassen sich mit folgenden Typen assoziieren: `webcontent`, `transform` und `map` (vgl. Abschnitt 5.4.2). Assets mit dem Typ `webcontent` werden beibehalten und der Typ `transform` bewirkt die Übersetzung des Assets. Der Typ `map` bewirkt zunächst eine Abbildung der Asset-Datei auf einen formatierbaren Kurs, der im Anschluss den Typ `fk` erhält und vom Übersetzungsprozess weiterverarbeitet wird. Listing 5.1 [4] zeigt einen umgangssprachlichen Algorithmus, der das Durchwandern eines formatierbaren Kurses formuliert.

Listing 5.1: Algorithmus Disaggregation

```
Bearbeite Kurs (Disaggregation)
1. Öffne FK
2. Bearbeite Ressourcen
   Wenn Ressource vom Typ:
     webcontent: Übernehme Ressource
     fk: [Bearbeite (Sub)Kurs] und binde seine Struktur in
         diesen ein
     fb: [Bearbeite Baustein] und binde seine Assets in
         diesen ein
3. Durchlaufe Kursstruktur
   Wenn Knoten referenziert:
     fk: Setze Referenz auf neue Substruktur (Submanifest)
     fb: Referenziere die Startdatei des Bausteins

Bearbeite Baustein
1. Öffne FB
2. Durchlaufe Ressourcen
   Wenn Ressource vom Typ:
     webcontent: Übernehme Ressource
     transform: Übersetze Ressource
     map: Nutze Loader und [bearbeite Kurs]
```

5.5.2 Transformationspaket

Nachdem die Datenhaltung der modularen E-Learning-Inhalte detailiert beschrieben wurde, wird im Folgenden das zu einer Übersetzung definierte Regelwerk erläutert. Die Verwaltung der Übersetzungsregeln soll ebenso austauschbar sein, wie die Bausteine selbst. Das

[4] Die in eckigen Klammern gesetzten Textfragmente bezeichnen rekursive Aufrufe und bewirken einen Sprung zu einer anderen Zeile.

Ziel ist die Erzeugung möglichst vieler unterschiedlicher Formate, die zudem mit spezifischen Layoutmerkmalen versehen sind. Hierfür ist es zweckmäßig, alle für die Übersetzung relevanten Daten in einem Paket zusammenzufassen, das über einheitliche Schnittstellen verfügt. Ein solcher Ansatz ermöglicht, die für die Kurserzeugung relevanten Daten, auszutauschen, um unterschiedliche Repräsentationen für einen Kurs zu produzieren. Ebenso wird die Entwicklung spezieller Werkzeuge ermöglicht, die über diese einheitliche Schnittstelle Layouts anpassen oder neue erzeugen können. Hieraus ergibt sich folgende Beziehung:

$$formatierbarerKurs + Transformationspaket = Lernobjekt \tag{5.1}$$

Wird das Transformationspaket ausgetauscht, ändert sich zwangsläufig das zu erstellende Lernobjekt. Das Transformationspaket enthält die Ressourcen, die für eine Übersetzung des formatierbaren Kurses notwendig sind. Hierzu gehören neben Anwendungen, einem Schema und der Beschreibung auch die assoziierten Übersetzungsregeln. Letztere werden durch so genannte *Transformatoren* angegeben, die mit verschiedenen Techniken implementiert werden. Für die Übersetzung kann z. B. die in Abschnitt 3.2.1 vorgestellte extended Stylesheet Language (XSL) eingesetzt werden. Ferner lassen sich andere Techniken verwenden, wie z. B. die Java XML-Schnittstelle JDOM [Harold02a, JDO05]. Mit dieser Schnittstelle können XML-Dokumente spielend durchlaufen und gestaltet werden. Weiterhin werden *Anwendungen* benötigt. Sie bearbeiten Dokumente für den Fall, dass erst ein Zwischenformat generiert werden muss. Dieses wird beispielsweise bei der Erstellung von Formating Objects Dokumenten (FO) generiert, da es mit einem FO-Prozessor in ein PDF-Dokument konvertiert werden muss. Ein anderes Beispiel ist die Erzeugung von Latex-Dokumenten, aus denen erst mit einem Latex-Prozessor das PDF- oder DVI-Format erzeugt wird.

Jedes Transformationspaket enthält die Sprachdefinition für die formatierbaren Bausteine. Der Sinn und Zweck, der mit der Zuordnung eines beliebigen *Schemas* erfüllt wird, liegt in der Allgemeingültigkeit der Pakete. Somit lassen sich je nach Anwendungskontext andere Sprachen bestimmen, die mit diesem Paket übersetzt werden können. Außerdem lassen sich spezialisierte Spracherweiterungen und die damit verbundene Anreicherung der Übersetzungsregeln für einen Transformator in demselben Paket durchführen.

Bei der Übersetzung einzelner Bausteine wird in vielen Fällen eine Nachbearbeitung benötigt. Mögliche Anwendungsfälle sind die Übersetzung von Formeln in eine bestimmte Darstellung, die Anpassung der Bausteine an ein einheitliches metrisches System sowie die Konvertierung von Bildformaten. Diese Spezialaufgaben sollen möglichst flexibel einstellbar sein. Deshalb können jedem Transformator *Prozessoren* zugeordnet werden, die den Baustein entweder vor der Übersetzung oder im Anschluss bearbeiten. Prozessoren sind ebenso wie Anwendungen Programme, die sich innerhalb des Pakets befinden.

Abbildung 5.19 veranschaulicht in Anlehnung an das *IMS Content Package Information Model* den Aufbau des Transformationspakets. Es sind komprimierte Archive mit einem Manifest und einer variablen Anzahl an Dateien. Das Manifest entspricht einer Beschreibung der enthaltenen Ressourcen und ordnet ihnen gewisse Aufgaben zu. Aufgrund des

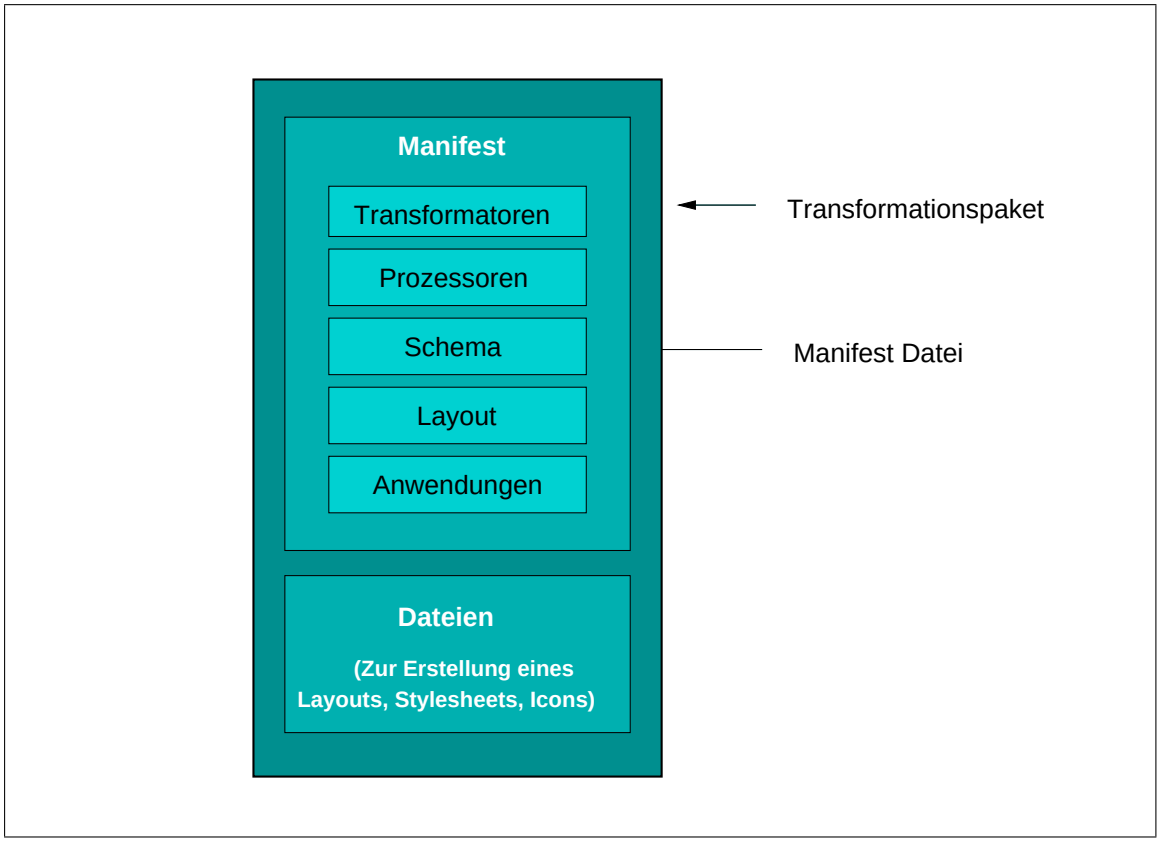


Abbildung 5.19: Transformationspaket

spezifizierten Aufbaus, kann ein beliebiges Programm das Transformationspaket verarbeiten und für die Übersetzung eines formatierbaren Kurse heranziehen.

Der strukturelle Aufbau des Manifests wird in Abbildung 5.20 illustriert. Das Manifest ist zugleich das Wurzelelement und der Container für die Elemente Transformatoren, Anwendungen, Schemata, Ressourcen und Layout. Tabelle C.2 enthält eine detaillierte Beschreibung der Elemente mit ihren Attributen. Jedes Manifest enthält eine Ressourcen-Liste, von denen mehrere Dateieinträge besitzen. Diese Ressourcen dienen zur allgemeinen Dateiverwaltung und können via Identifier referenziert werden.

In jedem Manifest müssen mindestens zwei Transformatoren definiert werden, damit den Minimalansprüchen für die Übersetzung genüge getan wird. Ein Transformator wird für die Übersetzung, der in einem Baustein definierten Dokumente, benötigt und ein weiterer für die Umwandlung der Kursstruktur in das Ausgabeformat. Jeder Transformator verfügt über einen Namen, eine `Id`, einen Typ, der den Transformator die gewünschte Aufgabe zuordnet, sowie eine Dateiendung, die für das zu erstellende Format notwendig ist. Zu einem Transformator gehören in der Regel eine oder mehrere Dateien. Wird z. B. ein XSL-Transformator eingesetzt, ist dieser auf mehrere Stylesheets angewiesen. Diese werden im Transformationspaket als *Ressourcen* deklariert und lassen sich per `identifierRef` attributieren. Ein Programm, welches das Transformationspaket verwendet, kann anhand des Identifiers den Ressourcen-Ast durchwandern, bis der referenzierte Identifer

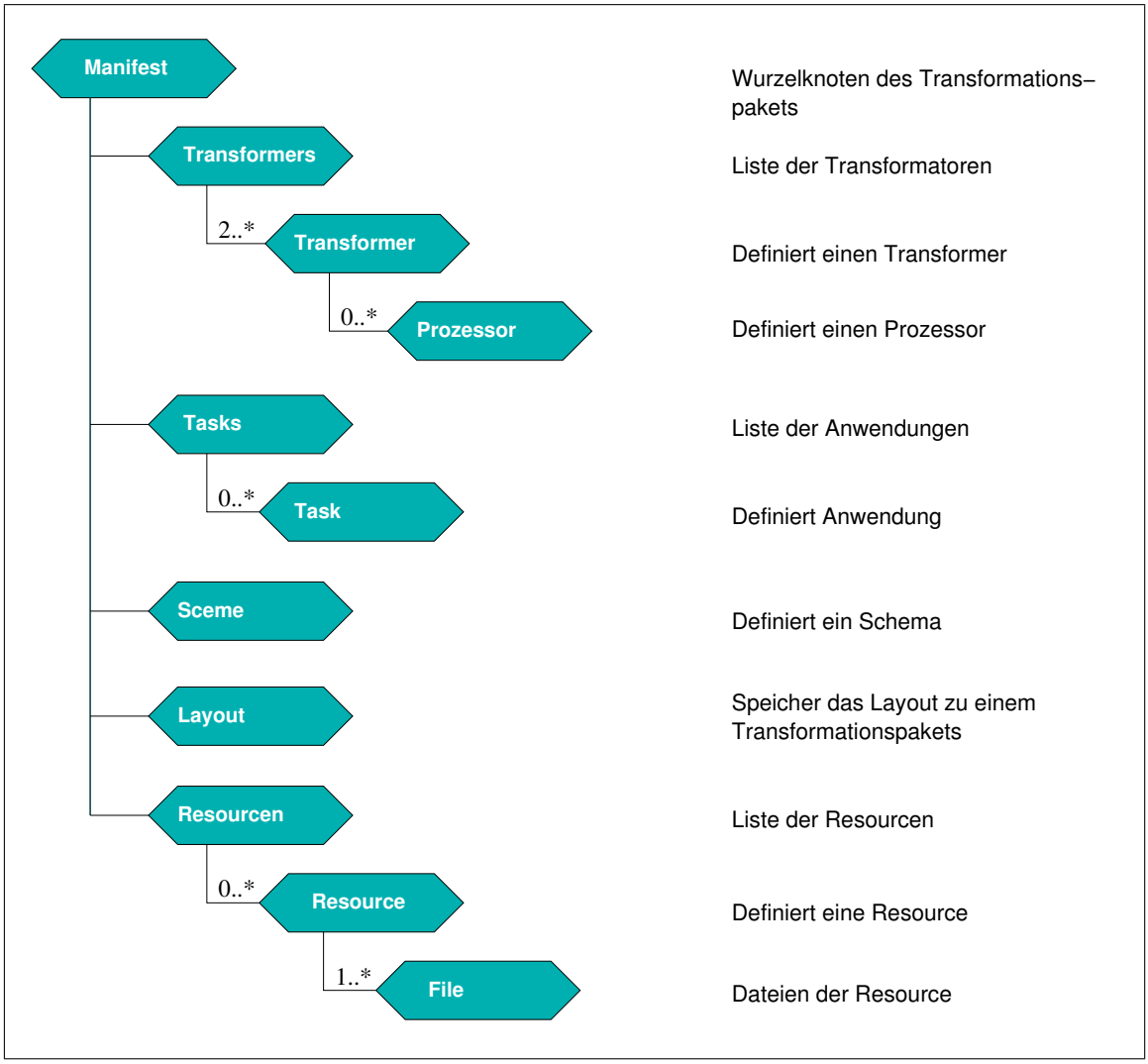


Abbildung 5.20: Modell des Transformationspakets

mit dem der Ressource übereinstimmt, und die Dateien verarbeiten. Zusätzlich können einem Transformator mehrere Prozessoren zugewiesen werden, die ebenso wie Transformatoren mit einer Ressource assoziiert sind. Für den Fall, dass sie als *aktiv* markiert sind, werden sie in der durch das Attribut `order` festgelegten Reihenfolge aufgerufen, um den Inhalt zu verarbeiten.

Die Liste der Anwendungen werden mit dem `tasks`-Element gruppiert. Ein Task enthält den ausführbaren Programmcode, der wieder mit einer Ressource assoziiert ist. Die zugrunde liegende Sprache wird über das `lang`-Attribut angegeben. Der Typ gibt Aufschluss darüber, ob die Anwendung vor der Übersetzung gestartet werden soll, oder erst nachdem der Kurs komplett übersetzt wurde. Anwendungen, die vor der Übersetzung gestartet werden, können z. B. Systemvariablen setzen, wogegen Anwendungen, die nach der Übersetzung laufen, das Resultat prozessieren, wie der bereits erwähnte FO-Prozessor.

Abschließend werden Schema und Layout des Transformationspakets definiert. Ge-

nauso wie die Transformatoren und Prozessoren, verweisen Schema- und Layout-Knoten ebenfalls auf Ressourcen und damit auf die Dateien, die die jeweiligen Sprach- und Layout-Definition enthalten.

Die Bezeichnung des Transformationspakets wird direkt im Manifest-Element mit dem Attribut `desc` hinterlegt. Durch die Referenz auf eine Ressource lässt sich außerdem eine Icon-Datei hinterlegen. Diese zusätzlichen Attribute werden zur Anzeige innerhalb von Applikationen oder Servern genutzt, um über deren Inhalt und Ausgabeziel zu informieren.

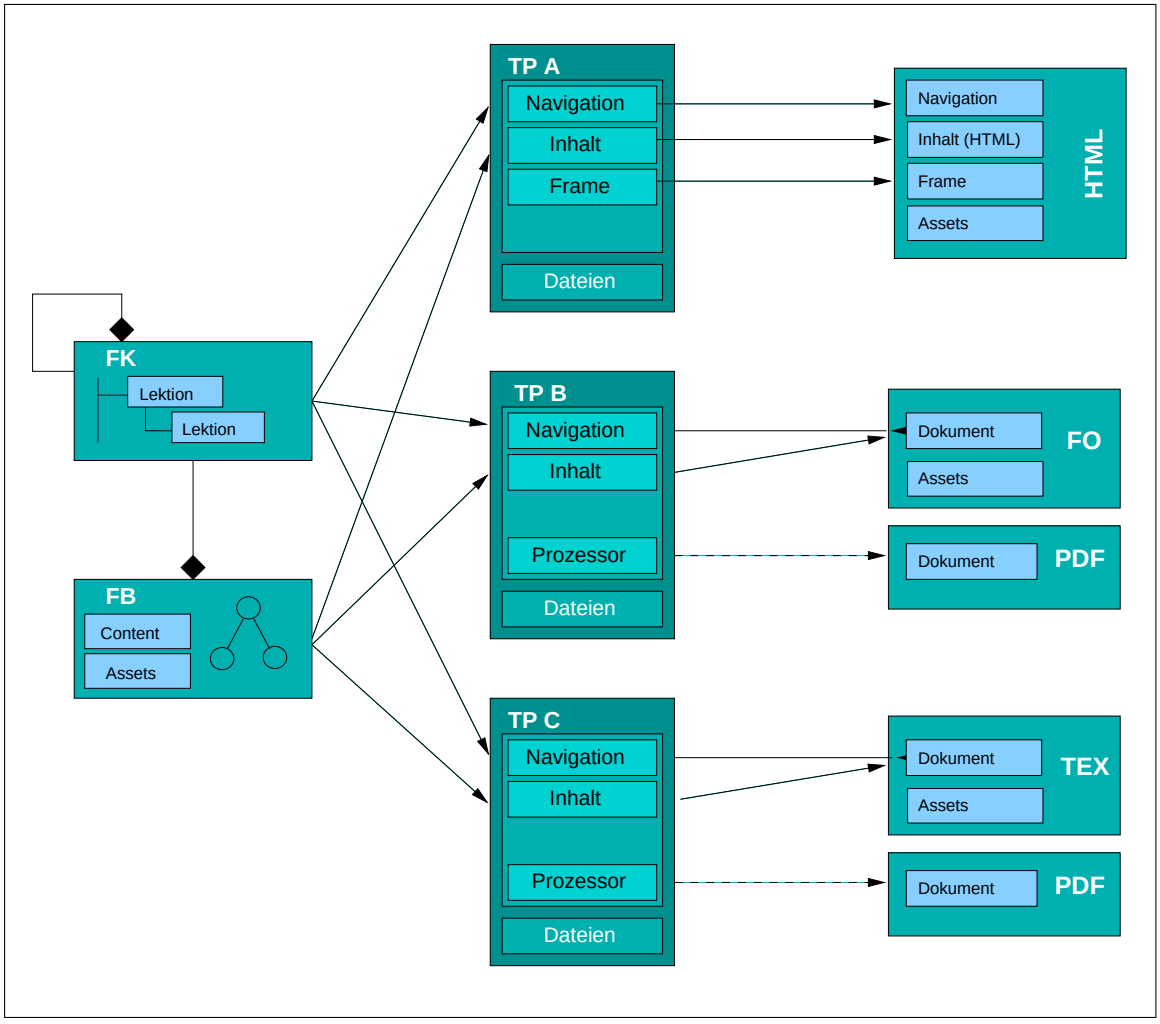


Abbildung 5.21: Abbildung

Abbildung 5.21 demonstriert die Vielseitigkeit von Transformationspaketen, indem ein formatierbarer Kurs auf die drei unterschiedlichen Formate *HTML*, *PDF (FOP)* und *PDF (Latex)* abgebildet wird. Die Kursstruktur der `FK`-Komponente wird mit dem jeweiligen Transformator `Navigation` in eine Repräsentation des Ausgabeformats übersetzt.

Die `FB`-Komponenten enthalten neben dem Inhalt auch die dazugehörenden Assets und werden mit dem Transformator `Inhalt` in das Ausgabeformat übersetzt. Diese beiden Transformatoren sind Bestandteil aller abgebildeten Transformationspakete. *TP A* ver-

fügt im Gegensatz zu *TP B* und *TP C* über einen weiteren Transformator, der für die Erzeugung eines HTML-Framesets zwingend benötigt wird. Das Frameset ist eine HTML-Seite, die die erzeugte Navigation mit den generierten HTML-Seiten verbindet und den Inhalt der Bausteine anzeigt. Dadurch ist es möglich, die Navigation während der Abarbeitung des Kurses permanent präsent zu haben. Anwendungen werden für dieses Format nicht benötigt, da bereits die Transformatoren das Endergebnis liefern. Die Gestaltung von PDF-Dokumenten benötigt hingegen einen weiteren Verarbeitungsschritt, wodurch für *TP B* und *TP C* jeweils eine Anwendung eingesetzt wird. *TP B* erzeugt, im Gegensatz zu *TP A*, nur ein FO-Dokument und wird anschließend, mit dem FO-Prozessor, in ein PDF-Dokument übersetzt. Ähnlich wie bei *TP B* wird bei *TP C* pdflatex eingesetzt, um aus den erzeugten TEX-Quellen ein PDF-Dokument zu generieren.

5.5.3 Prozessoren

Prozessoren wurden bereits im vorigen Abschnitt erwähnt und sollen hier näher erläutert werden. Bei der Übersetzung von Inhalten treten immer wieder individuelle Wünsche und Bedürfnisse auf, die nicht mit herkömmlichen Abbildungsverfahren, wie z. B. Stylesheets, zu realisieren sind. Vielmehr werden eigenständige Prozesse benötigt, die spezielle, für das Ausgabeformat angepasste, Mechanismen zur Verfügung stellen. Ein Beispiel hierfür ist die Verarbeitung von Formeln. Aus heutiger Sicht ist es durchaus sinnvoll, Formeln als Grafiken in Webseiten einzubauen, wie es bei herkömmlichen Latex zu HTML-Konvertern üblich ist. Es ist jedoch abzusehen, dass MathML oder OpenMath (vgl. Kapitel 3.2.4) von den führenden Browser-Herstellern in naher Zukunft unterstützt werden. Aus diesem Grund ist die Erzeugung verschiedener Formelrepräsentationen zweckmäßig.

Des weiteren wurde das Baukastenprinzip eingeführt, um möglichst unabhängig vom Ausgabeformat zu sein. Soll beispielsweise Latex generiert werden, so müssen Formeln in Latex gesetzt werden, wodurch eine Formelübersetzung notwendig ist. Genau für solche Anwendungsfälle muss ein allgemeiner Mechanismus gefunden werden, der sowohl die Definition individueller Lösungen unterstützt als auch flexibel einzusetzen ist. Abbildung 5.22 zeigt hierfür einen Ansatz, der Dokumente seriell durch eine Prozessor-Pipeline delegiert. Jeder Prozessor bearbeitet das Quelldokument und übergibt sein Resultat an den in der Reihenfolge als nächsten nominierten Prozessor. Sie werden entweder durch eine externe Applikation als Übersetzungsoption zur Auswahl angeboten oder direkt in das Transformationspaket verankert.

5.5.4 Layout

Das Transformationspaket ist ein allgemeiner Container für Transformatoren und Prozessoren. Zwar wurde bis jetzt noch nicht näher auf die Übersetzungsregeln eingegangen, es ist jedoch festzuhalten, dass ihre Beschreibung in Abhängigkeit von dem zu erzeugenden Format technisch formuliert werden. Die Frage, die sich jetzt stellt, ist, wie zu einem festgelegten Format ein korrespondierendes Layout erstellt werden kann, ohne dass in den technischen Abbildungprozess eingegriffen wird. Hierfür ist eine strikte Trennung zwischen

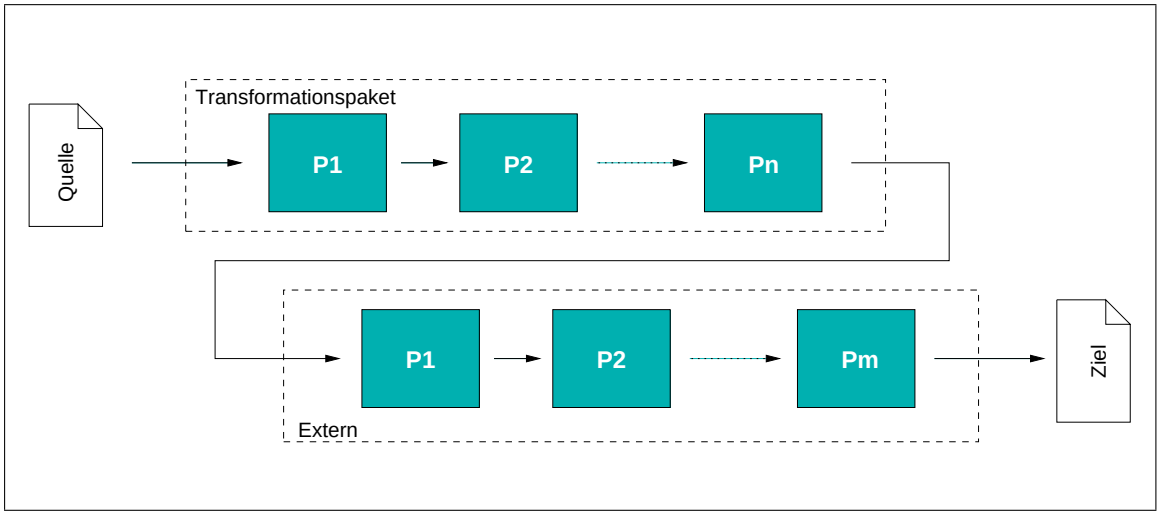


Abbildung 5.22: Prozessorpipeline

den Regeln, die für die Formaterzeugung benötigt werden, und der Präsentation bzw. dem Layout notwendig. Bei der Vielfalt der verschiedenen Ausgabeformate ist jedoch die präzise Festlegung des Layouts für das zu erzeugende Format nicht möglich. Jedes Format besitzt spezielle Eigenschaften und lässt nur einen begrenzten Spielraum für eine allgemeine Beschreibung zu. Als Beispiel dienen die bereits erwähnten Ausgabeformate HTML und PDF. Bei PDF-Dokumenten wird z. B. die Seitengrößen angegeben, wogegen HTML-Seiten dynamische Längen und Breiten benötigen. Außerdem lassen sich HTML-Seiten durch den Einsatz von *Framesets* und *Tabellen* wesentlich flexibler gestalten, als es bei PDF-Dokumenten möglich ist. Aus diesem Grund muss ein abstrakter Beschreibungsmechanismus gefunden werden, der die wesentlichen stilistischen Einstellungen speichert und diese den Transformatoren zur Verfügung stellt.

Aus dieser Separation folgt die Modifikation, ohne detaillierte Kenntnisse über das zu erzeugende Format zu besitzen. Dieses führt zu den zwei Benutzersichten bzw. Rollen *Formaterzeuger/-in* und *Layoutanpasser/-in*, die im Folgenden näher erläutert werden.

- Formatentwickler/-in: Die Rolle des Formaterzeugers benötigt ein fundiertes Wissen über das zu erstellende Format. Sie definiert das Basis-Layout und legen darüber hinaus die einstellbaren Merkmale fest, die über eine festgelegte Schnittstelle der Rolle Layoutentwickler/-in bekannt gegeben werden.

- Layoutentwickler/-in: Diese Rolle benötigt nicht das technische Wissen über das zu erzeugende Format. Sie entscheidet über die Darstellung des Ausgabeformats, indem sie ein Layout definiert und es über die von der Rolle Formatentwickler/-in bereitgestellte Schnittstelle verbindet.

Abbildung 5.23 veranschaulicht diesen Zusammenhang. Die Rolle Formatentwickler/-in erzeugt die notwendigen Transformatoren T_1 bis T_n und definiert darüber hinaus die Stellen, an denen Layoutmerkmale Bestandteil des Formats sind und später dynamisch

zugewiesen werden sollen. Sie erzeugt das Layout, indem zunächst die Maße für die Präsentation definiert werden. Des weiteren werden Layouteigenschaften wie Schriftentypen und Farben für einzelne Transformatoren spezifiziert.

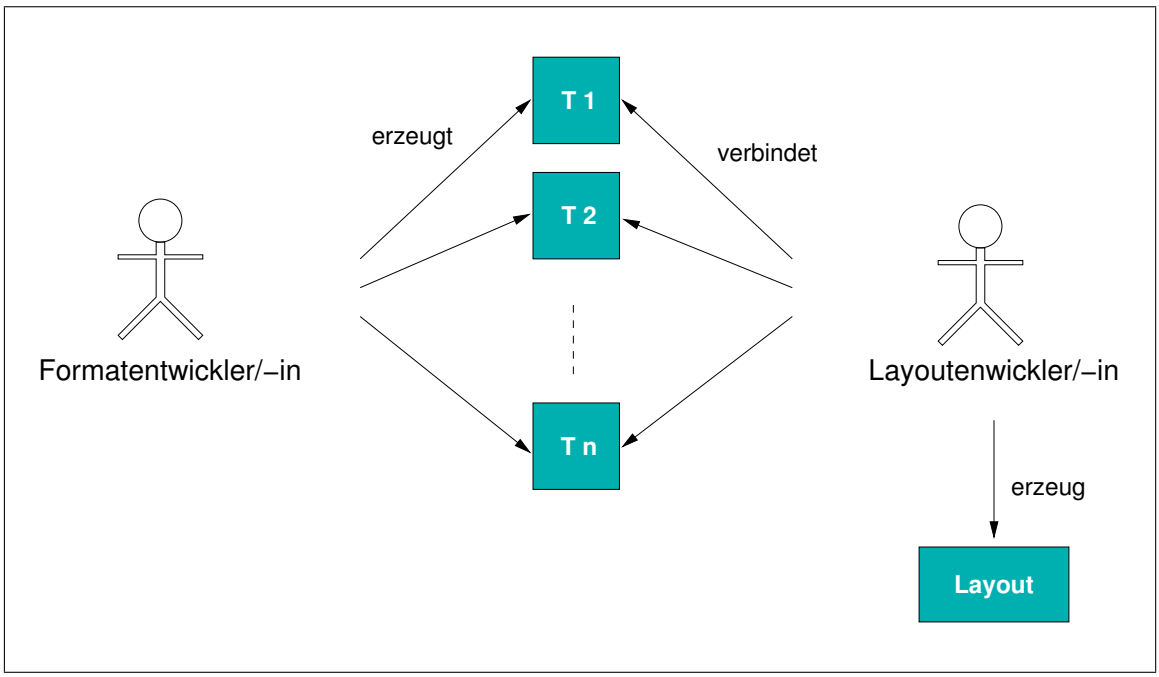


Abbildung 5.23: Rollen

Im Folgenden wird der strukturelle Aufbau des Layouts untersucht. Das Layout besteht aus zwei Teilen: dem Master-Layout, zur Definition der Seitenformatierung, und den Style-Eigenschaften, die als Auswahlkatalog für Transformatoren dienen. Mit dem Seiten-Master werden detailierte Angaben über den Aufbau einer Seite definiert. Hierzu gehören die Höhe und Breite einer Seite, genaue Angaben über den zu verwendenden Seitenausschnitt (*Body*) sowie Kopf- und Fußzeilen. Diese Angaben sind jedoch nicht zur Definition eines Layouts ausreichend. Es werden weitere Angaben über stilistische Merkmale einer Präsentation benötigt, wie Schrifttypen, Schriftgrößen oder auch Farbwerte. Liegt eine umfangreiche Liste mit Layouteigenschaften vor, werden sie auf die jeweiligen Transformatoren abgebildet. Hierbei findet die Zuordnung der im Layout festgelegten Eigenschaften zu den in den Transformatoren verwendeten Ressourcen statt. Der Vorteil, der sich aus einer solchen Abbildung ergibt, ist die Mehrfachnutzung definierter Eigenschaften. So können festgelegte Farben durch mehrere Transformatoren verwendet werden und müssen nur an einer Stelle, dem Layout, geändert werden. Abbildung 5.24 zeigt das Zusammenspiel zwischen Transformationspaket und Layout.

Das Layout lässt sich, ebenso wie das Manifest eines Transformationspakets, hierarchisch aufbauen (vgl. Abbildung 5.25 und Tabelle C.3). Der Wurzelknoten `Layout` dient als Container für `Master`, `Styles` und `Mapping` Elemente. Das Master-Element ist selbst ein Container für die Elemente, die den Aufbau einer Seite beschreiben. Es enthält das `Page`-Element, zur Speicherung der genauen metrischen Maße einer Seite, sowie das `Body`-

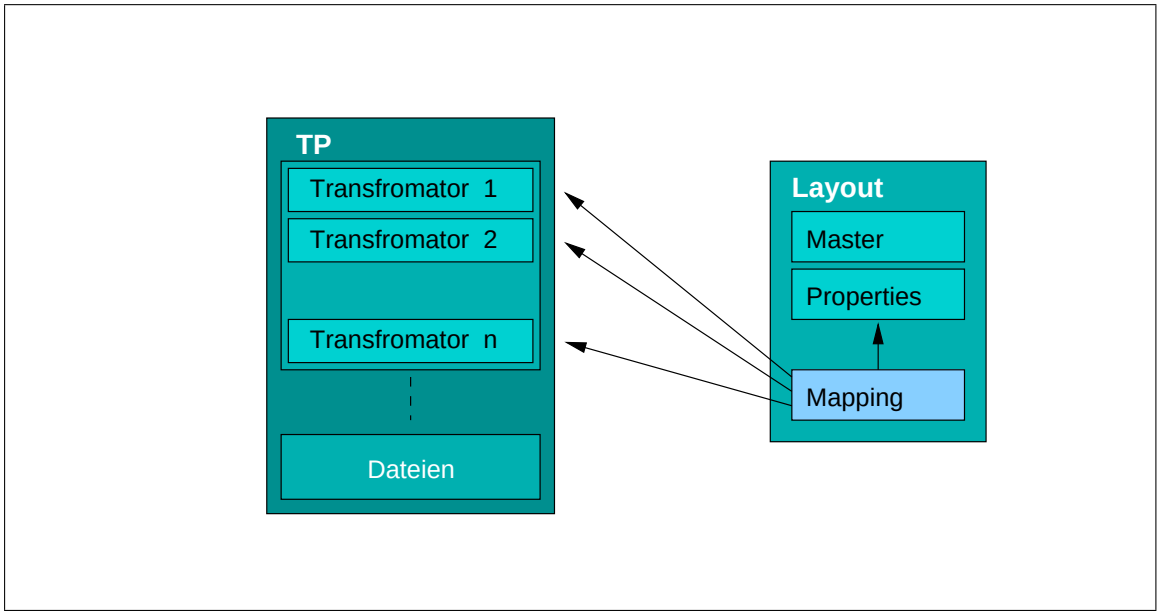


Abbildung 5.24: Abbildung des Layouts auf die Transformatoren

Element, zur Spezifikation der Seitenränder. Zusätzlich können Kopf- und Fußzeilen definiert werden, soweit sie für das zu erzeugende Ausgabeformat relevant sind.

Neben den Größenangaben enthält ein Layout `Styles`. Jeder `Style` entspricht selbst einer Aggregation aus Eigenschaften, die einen Typ besitzen, der aussagt, um welches Layoutmerkmal es sich handelt. Aufgrund der Formatvielfalt, die mit Transformationspaketen erzeugt werden kann und die damit verbundene individuelle technische Umsetzung, ist eine separate Speicherung der Layouteigenschaften notwendig, damit sie an beliebigen Stellen von Transformatoren abgerufen werden können. Diesbezüglich sollen ausschließlich die in Tabelle 5.5.4 definierten Typen eingesetzt werden. Der Vorteil, der sich aus diesem Ansatz ergibt, liegt auf der Hand. Die Struktur der Dokumente wird innerhalb der Übersetzungsregeln bearbeitet, wobei der eindeutige Bezug zur Sprachdefinition der Bausteine hergestellt wird. Genau solch ein Bezug soll in der Layout-Definition nicht vorhanden sein. Vielmehr muss eine *lose* Kopplung zwischen der Layoutbeschreibung und den Transformatoren bestehen, damit eine Veränderung des Layouts nicht zwangsläufig zu einer Modifikation der Transformatoren führt. Dieses ist vor allem dann vorteilhaft, wenn Transformatoren nicht als Skriptsprachen implementiert werden und deshalb nicht modifiziert werden können.

Um letztendlich die Verbindung zu den Transformatoren herzustellen, muss die Zuordnung der definierten Styles zu den festgelegten Layoutmerkmalen hergestellt werden. Realisiert wird diese Zuordnung durch eine Liste von `Reference`-Elementen, die einerseits definierte Styles mit Identifier referenzieren und andererseits den Namen des im Transformator festgelegten Merkmals enthalten. Hierdurch findet die eindeutige Abbildung der Style-Definitionen zu den Transformatoren statt. Das Ergebnis ist ein separiertes Layout, das sich unabhängig von Transformatoren verändern lässt. Hierbei können sowohl die Layout-Eigenschaften als auch deren Zuordnungen angepasst werden.

Typ	Verwendung
integer	Eine ganze Zahl
dezimal	Eine Zahl mit Nachkommastellen
boolean	Definiert einen boolschen Wert
align	Legt die Textausrichtung fest
percent	Speichert einen Prozentwert
string	Definiert einen variablen Text
image	Verweist auf eine austauschbare Abbildung
color	Eine Farbe in RGB-Notation
font	Legt einen Schrifttyp fest
fontsize	Schriftgröße in Punkten
fontfamily	Name einer Schriftart
chooser	Auswahlliste für eine Zeichenkette

Tabelle 5.4: Layouttypen

5.6 Kooperation

Nachdem das Modell zur Konstruktion von formatierbaren und modularen Lernmaterialien vorgestellt worden ist, wird in diesem Kapitel der Fokus auf die *koopertative* Entwicklung gerichtet. Die formatunabhängige Darstellung der Materialien ist in größeren verteilten Projekten eine wichtige Voraussetzung für die Entwicklung von E-Learning-Inhalten. Hierbei können Materialien an verschiedenen Standorten und von unterschiedlichen Entwicklern und Entwicklerinnen konstruiert, ausgetauscht und zusammengeschnürt werden. Genau diese Möglichkeit bieten die in Kapitel 2 vorgestellten Systeme nicht an. Mit ihnen lassen sich lediglich spezialisierte Kursunterlagen erzeugen, bei denen das Format und Layout fest verankert ist. Dabei wird zwangsläufig der vom Entwickler bzw. von der Entwicklerin präferierte Einsatzkontext festgelegt. Diese Kurse lassen sich aufgrund der fehlenden Orthogonalität und Granularität nicht ohne weiteres aggregieren, von den unterschiedlichen Präsentationsformaten abgesehen. Aus diesem Grund ist es sinnvoll, Kursunterlagen mit dem in diesem Kapitel vorgestellten Modell zu beschreiben, damit Materialien nicht nur formatunabhängig definiert werden, sondern sich auch durch unterschiedliche Anwender bzw. Anwenderinnen wiederverwenden lassen. Gerade im universitären Umfeld bewirkt die Wiederverwendung bereits erstellter Materialien eine erhebliche Kostenreduktion bei der Entwicklung neuer Inhalte.

Es ist keineswegs sinnvoll, Kurse per E-Mail zu verschicken und an anderen Standorten weiter zu bearbeiten. Das Problem ist, die Schwierigkeit zwischen Original und Kopie zu unterscheiden. Für einen solchen kooperativen Entwicklungsprozess ist es wesentlich effizienter, die Kurse und Bausteine dezentral zu verwalten und ein Entwicklungsrepository zur konsistenten Datenhaltung zu verwenden. In Relation zu dem in diesem Kapitel vorgestellten Baukastenprinzip entspricht das Repository dem Baukasten.

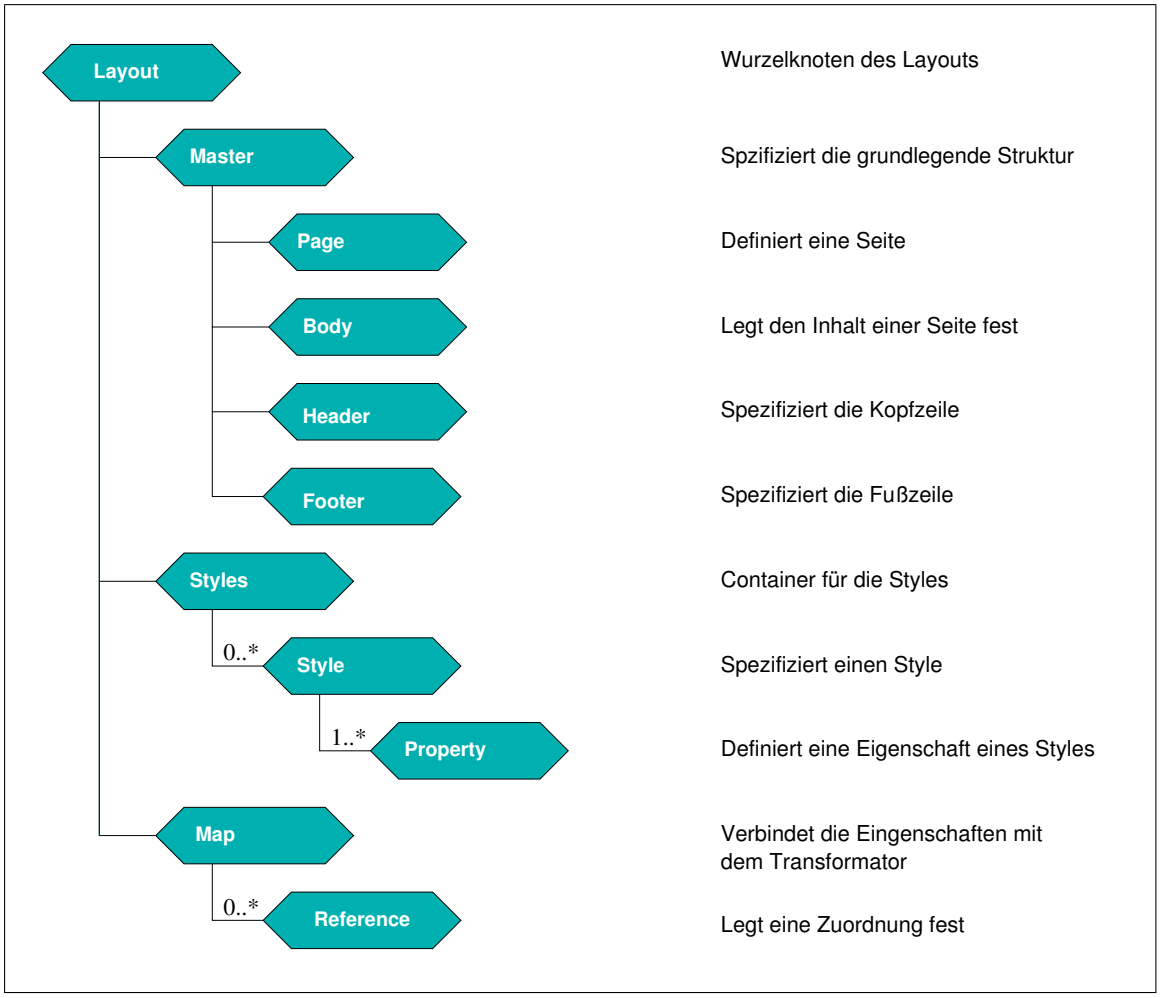


Abbildung 5.25: Spezifikation des Layouts

5.6.1 Der kooperative Entwicklungsprozess

In Kapitel 5.1.3 wurde bereits der Entwicklungsprozess für die Erstellung formatierbarer Kurse vorgestellt. An dieser Stelle soll dieser Prozess wieder aufgegriffen werden und unter besonderer Berücksichtigung des Aspekts *Kooperation* untersucht werden. Deshalb werden die Rollen *Bausteinentwickler/-in*, *Kurszusammensteller/-in* und *Umsetzer/-in* zu der Rolle *Autor/-in* vergröbert. Sie ist zwingend notwendig, weil in diesem Abschnitt ausschließlich das Zusammenspiel mehrerer Autoren von Interesse ist. Die anderen Rollen werden soweit beibehalten.

An einem kooperativen Entwicklungsprozess sind mindestens zwei Autoren bzw. Autorinnen beteiligt. Sie verfügen über ein Autorensystem, das formatierbare Bausteine, nach dem in Kapitel 5.2 vorgestellten Modell, aggregieren und transformieren kann. Hierbei findet der Erstellungsprozess der formatierbaren Bausteine dezentral mit jedem Autorensystem statt. Erst nachdem ein Baustein oder ein kompletter Kurs fertig gestellt ist, kann dieser anderen Autoren und Autorinnen angeboten werden. Arbeiten zwei Personen an

einem Kurs oder teilen sich bei der Kurserstellung bestimmte Bereiche, können sie per Dateisystem oder via E-Mail verschickt werden. Arbeiten hingegen mehrere Entwickler und Entwicklerinnen an einem Kurs, wird der Entwicklungsprozess schnell unübersichtlich und es kann zu Inkonsistenzen und Datenverlusten kommen, weil Kurs bzw. Bausteine von mehreren Personen gleichzeitig modifiziert werden. Für diesen Fall bietet sich die Verwaltung auf einem zentralen Server an, auf den die am Entwicklungsprozess beteiligten Parteien gleichermaßen zugreifen können. Abbildung 5.26 verdeutlicht diesen Zusammenhang. Bei einer Kooperation von m Teams existieren n Autorensysteme, die mit mehreren der j Repositories verbunden sein können. Entsprechend dem Baukastenprinzip werden die Repositories metaphorisch als Baukasten bezeichnet, wodurch sich der Begriff *Construktion Kit Server CKS* ableitet. Jedes Autorenwerkzeug ist in der Lage, die von ihm erstellten Bausteine und Kurse auf einem CKS abzulegen und sie dadurch anderen anzubieten. Als Beispiel kann das erste Team einen Online-Kurs für das Anwendungsfach *Technische Informatik* erstellen. Zwei studentische Mitarbeiter entwickeln hierfür einen Kurs, der später studienbegleitend eingesetzt werden soll. Jeder der Studenten setzt sein eigenes Autorensystem ein, welches jeweils mit dem ersten CKS verbunden ist. So ist es für beide möglich, an demselben, auf dem CKS gespeicherten, Dokument zu arbeiten. Einer der Autoren verfasst das Thema *Wechselstromschaltungen* und möchte einen Exkurs zum Thema *Komplexe Zahlen* anbieten. Dieser existiert bereits auf einem anderen CKS, und wurde von einem anderen Team entwickelt. Team m verwendet jedoch die *Komplexen Zahlen* in einem anderen Zusammenhang, nämlich als Skriptum in einer Präsenzveranstaltung. Aufgrund des eingeführten allgemeinen Modells, ist es ohne großen Aufwand möglich, den Kurs von Team m in den eigenen Kurs einzubauen und ihn innerhalb der Online-Präsentation zu nutzen.

Der CKS wird im Gegensatz zu einem LMS oder einem öffentlichen Respository als Entwicklungsplattform für formatierbare Kurse und Bausteine eingesetzt, die ohne eine entsprechende Transformation von Studierenden oder Lehrenden nicht genutzt werden können. Nach der Fertigstellung des Kurses kann er durch eine Transformation für das jeweilige Einsatzgebiet genutzt und mit einem individuellen Layout versehen werden. Der einfachste Anwendungsfall ist der Einsatz eines Web-Browsers, der HTML oder PDF Inhalte direkt anzeigt. Soll jedoch ein LMS eingesetzt werden, lassen sich Kurse dort direkt, durch Erzeugung eines IMS Content Package kompatiblen Pakets, importieren. Außerdem können transformierte Inhalte als Lernobjekte anderen Entwicklern und Entwicklerinnen in öffentlichen Respositories angeboten werden. Hierdurch zeichnet sich der Vorteil gegenüber allgemeinen Lernobjekte ab, die nicht modular, austauschbar und veränderbar sind.

5.6.2 Operationen

Wie bereits in Abschnitt 2.3.2 erläutert wurde, arbeitet das IMS an einer Spezifikation für die Konzeption von Lernobjekt-Repositories. Sie spezifizieren den Zugriff und das Auffinden von Lernobjekten. Im Gegensatz dazu dient der CKS als Entwicklungsplattform, auf dem die Quellen der Bausteine und Kurse abgelegt werden. Aus diesem Grund

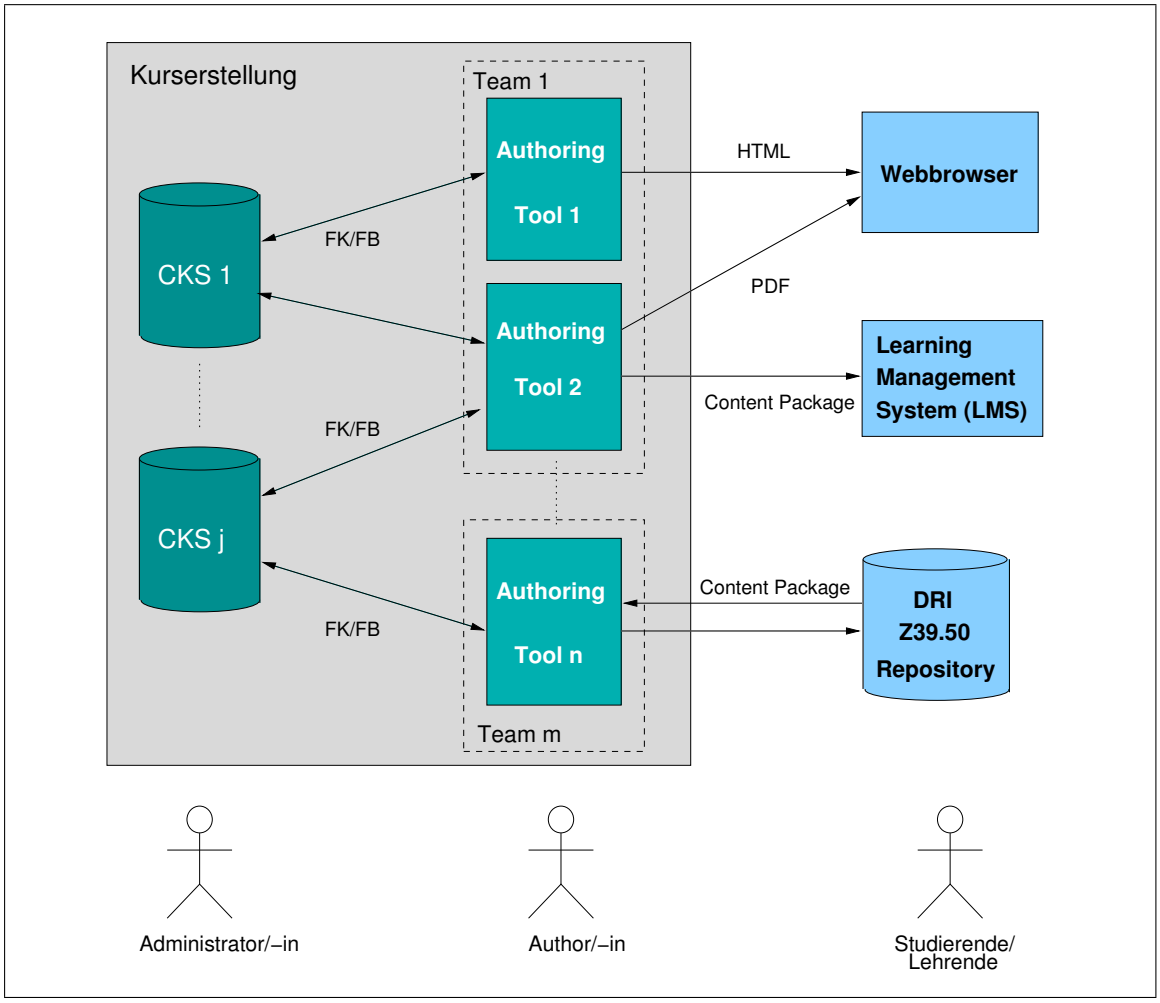


Abbildung 5.26: Der Construction Kit Server

werden für den Entwicklungsprozess, neben Suchverfahren, entsprechende Mechanismen zur kooperativen Arbeit benötigt. Diese sind das exklusive Schreibrecht auf Dateien, eine Versionskontrolle sowie Authentifizierungsmechanismen.

Wird ein Baustein von einem Autor bzw. einer Autorin bearbeitet, soll zu diesem Zeitpunkt keine Veränderung durch eine andere Person stattfinden. Hierfür muss das *exklusive Schreibrecht* erworben werden, das zwar das Öffnen eines bereits verwendeten Bausteins ermöglicht, jedoch nicht das Abspeichern. Dieses Verfahren wird bei jedem modernen Betriebssystemen eingesetzt und führt auf das Leser-Schreiber-Problem der theoretischen Informatik zurück [Valk80, Engesser93]. Es tritt dann auf, wenn mehrere Autoren bzw. Autorinnen am selben Dokument arbeiten. Ein anderes Verfahren, wie es bei CVS (Concurent Version Control) eingesetzt wird, bietet die Arbeit mit Kopien an. Das Original verbleibt auf einem Server und wird bei jedem Client als Kopie angelegt.

Soll eine bearbeitete Kopie wieder auf dem Server gespeichert werden, muss das Original ersetzt werden. Zuvor muss jedoch ein möglicher lokaler Konflikt behoben werden, der genau dann entsteht, wenn zwei Autoren bzw. Autorinnen an derselben Stelle ei-

nes Dokuments arbeiten. Um einen möglichst einfachen Zugriff auf den CKS anzubieten, wird das WebDAV-Protokoll [Goland99] angeboten, mit dem Dateisysteme über das HTTP-Protokoll [Fielding99] eingebunden werden und so einfach über einen Webserver realisiert werden können. WebDAV unterstützt das *exklusive Schreiben* und bietet somit *Lock*-Operationen auf Dateien an.

Ein weiterer wichtiger Mechanismus ist die *Versionskontrolle*. Durch sie kann der Entwicklungsprozess überwacht und im Bedarfsfall ein in der Vergangenheit liegender Entwicklungszustand wiederhergestellt werden. Eine Funktion, die sich dadurch ergibt, ist z. B. das Markieren der aktuellen Version, damit sie zu einem späteren Zeitpunkt mit einem früheren Entwicklungsstand zusammengeführt werden kann.

Die bisherigen Mechanismen kommen jedoch nicht ohne einen Authentifizierungsmechanismus aus. Werden Dateien *gelockt* oder neu erstellt, müssen individuelle Rechte an den Anwender bzw. an die Anwenderin vergeben werden. Dieses ist nur dann durchführbar, wenn er bzw. sie innerhalb des Systems bekannt ist, denn bei einer Datei, die nur im Lese-Modus geöffnet werden kann, weil sie zur selben Zeit bereits von einer anderen Person bearbeitet wird, muss die Person, die das exklusive Schreibrecht besitzt, ermittelt werden. Im Zusammenhang mit der Versionierung ist es notwendig festzustellen, welcher Autor bzw. welche Autorin die Änderung durchgeführt hat.

Kapitel 6

Umsetzung

Bisher wurde ein Modell zur Konstruktion homogener Kursmaterialien eingeführt, welches gezielt abstrakt formuliert wurde, damit unterschiedliche Technologien, wie z. B. Datenbanksysteme oder XML-Dokumente, für die Realisierung eingesetzt werden können. In diesem Kapitel wird nachfolgend eine dem Modell zugrunde liegende Referenzimplementierung vorgestellt, die die wesentlichen Eigenschaften des Konzepts verdeutlicht und dessen Realisierbarkeit nachweist.

6.1 Das Autorenwerkzeug Lyssa

Das in Kapitel 5 vorgestellte Modell ist in dem Autorenwerkzeug *Lyssa* umgesetzt worden, das somit als Referenzimplementierung gilt. Der folgende Abschnitt veranschaulicht nunmehr, wie das Modell der *Formatierbaren Bausteine* und *Kurse* sowie das *Kursmodell* und das *Transformationspaket* in einer gemeinsamen Architektur zusammenspielen. Eine zentrale Eigenschaft des Autorenwerkzeugs *Lyssa* ist die dem Baukastenprinzip (vgl. Kapitel 5.3) entsprechende Erstellung modularer Bausteine. Mit einer grafischen Benutzerschnittstelle (GUI) können Autoren und Autorinnen auf einfache Weise Bausteine erzeugen und sie mit anderen Bausteinen individuell kombinieren. Diese Komposition ergibt einen Kurs, der wieder in seine Bestandteile zerlegt werden kann. Hierdurch entsteht ein flexibles Benutzungsmodell, das die Entwicklung interdisziplinärer Lernmaterialien erst ermöglicht. Das Autorensystem verwendet das Modell der Formatierbaren Bausteine und Kurse, wodurch sichergestellt ist, dass Bausteine verschiedener Autoren und Autorinnen kompatibel zueinander sind und nahtlos zu neuen Kursen zusammengeführt werden können.

Das System wurde basierend auf mehreren wissenschaftlichen Arbeiten entwickelt, die in Tabelle 6.1 aufgeführt sind. Die Realisierung der grafischen Benutzerschnittstelle sowie der Software-API zum Verwalten von Archiven und IMS Content Packages wurde von Michael Bungenstock durchgeführt [Baudry05, Bungenstock03b]. Marc Vollmann hat den CKS um eine Search Engine auf Schlagwortbasis erweitert, die zwar wegen der Vollständigkeit erwähnt wird, jedoch in dieser Arbeit nicht weiter vertieft wird.

Die Systemkomponenten, die aus dem in dieser Arbeit entwickelten Modell resultieren, werden in den nachfolgenden Kapiteln vorgestellt.

Komponente	Verantwortung	Kontext	Kapitel
File Management	Michel Bungenstock	Dissertation	–
Content Package Engine	Michel Bungenstock	Dissertation	–
Multimedia Environment	Michel Bungenstock	Dissertation	–
Transformation Engine	Andreas Baudry	Dissertation	6.1.1
Formatierbare Bausteine	Andreas Baudry	Dissertation	6.1.2
Formatierbare Kurse	Andreas Baudry	Dissertation	6.1.3
Kursmodell Framework	Andreas Baudry	Dissertation	6.1.4
DokBook Engine	Andreas Baudry	Dissertation	6.1.5
Office Engine	Andreas Baudry	Dissertation	6.1.5
Latex Engine	Andreas Baudry	Dissertation	6.4
Metadaten Engine	Michael Bungenstock	Dissertation	–
Layout Engine	Andreas Baudry	Dissertation	6.2
Construktion Kit Server	Andreas Baudry	Dissertation	6.3
Search Engine	Marc Vollmann	Diplomarbeit	–

Tabelle 6.1: Arbeitsteilung für Systemkomponenten

6.1.1 Systembeschreibung

Dieser Abschnitt erläutert die Funktionsweise des Autorensystems *Lyssa*. Das Benutzungsmodell ist an die Arbeit mit einem Dateisystem angelehnt. Abbildung 6.1 zeigt das Autorensystem mit einem bereits geöffneten Kurs. Auf der linken Seite befindet sich die *Workbench*, die ähnlich, wie der Windows Explorer, aus Ordnern und Dateien aufgebaut ist. Sie ermöglicht das Öffnen und Verschieben von Assets, Bausteinen und Kursen. Innerhalb des MDI-Bereichs (Multiple Device Interface) werden geöffnete Bausteine und Kurse angezeigt. Abbildung 6.1 veranschaulicht einen geöffneten Kurs, der um neue Kursabschnitte erweitert werden kann, indem ein neuer Baustein oder ein Kurs von der Workbench direkt in die Kursstruktur gezogen und dort abgelegt wird. Ebenso lässt sich eine entsprechende Disaggregation durchführen, indem ein Baustein, der Bestandteil des geöffneten Kurses ist, einfach mit einer Mausbewegung aus einem Kurs heraus auf die Workbench verschoben wird. Neben gewöhnlichen Bausteinen lassen sich in gleicher Weise ganze Kurse in andere Kurse ziehen. Auch ihnen wird genau ein Thema in der Kursstruktur zugewiesen. Die Erweiterung von Kursen durch andere Kurse und Bausteine wird technisch durch die Verschiebung der Archiv-Dateien in das komprimierte Archiv des übergeordneten Kurses realisiert, wodurch beliebig tiefe Verschachtelungsstrukturen entstehen.

Neben Kursen können mit *Lyssa* auch Bausteine zum Bearbeiten geöffnet werden. Allerdings enthalten sie selbst keine Bausteine oder Kurse, sondern Assets mit multimedialem Inhalt, wie z. B. Videos, Flash-Filme oder Abbildungen. Abbildung 6.2 zeigt exemplarisch einen geöffneten Baustein, der mehrere Assets aggregiert. Zudem wird in jedem Baustein eine Datei als Einstiegspunkt definiert. Er dient während des Übersetzungsvorgangs zur Identifikation des Lernmaterials. Andere Assets lassen sich mit frei wählbaren Werkzeugen erstellen und bearbeiten, und werden nicht explizit in *Lyssa* eingebunden. Abbildungen

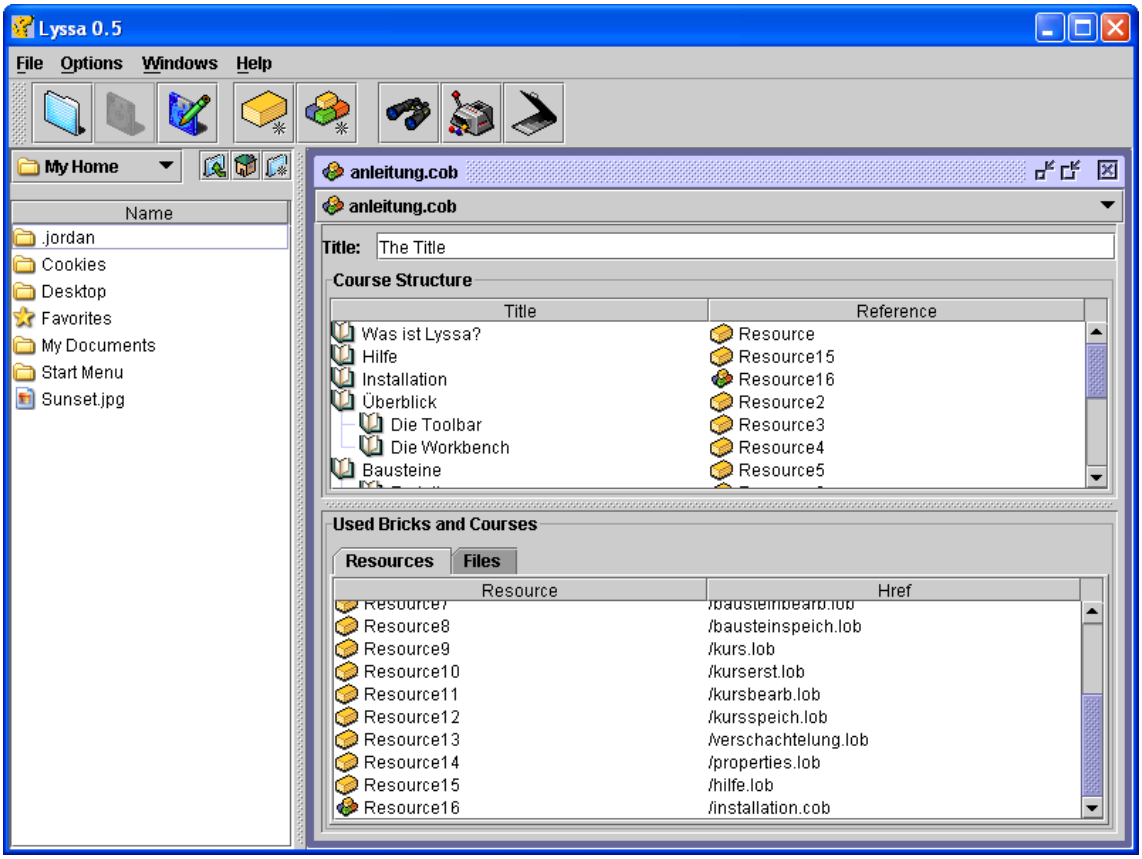


Abbildung 6.1: Das Autorenwerkzeug *Lyssa*

und Grafiken können z. B. mit *Gimp* oder *Photoshop* erstellen werden und danach in einen Baustein gezogen werden. Sind die Assets fertiggestellt und in den Baustein kopiert, können sie von dem Content-Modell des Bausteins erfasst und referenziert werden.

Übersetzungsprozess

Derart erzeugte Kurse und Bausteine sind in dieser Form nicht für die Lehre nutzbar. Das liegt an den aggregierten Kursen und Bausteinen, aber auch an der noch Layout freien Form der Bausteine. Aus diesem Grund bietet das Autorenwerkzeug *Lyssa* einen Übersetzungsmechanismus an, der es erlaubt, Kurse in verschiedene Zielformate zu übersetzen. Hierzu wird ein erweiterbares Rahmenwerk zum Einbinden neuer Formate angeboten.

Abbildung 6.3 zeigt den Export-Dialog von Lyssa, in dem sich Übersetzungsparameter einstellen lassen. Der Übersetzungsprozess ist im Wesentlichen die Umsetzung des in Kapitel 5.5.2 eingeführten Transformationspaketes mit den zugehörenden Prozessoren. Unter der Rubrik *Media Type* wird das zu verwendende Transformationspaket ausgewählt, wobei ein Icon das zu erzeugende Zielformat symbolisiert. Der Schalter *Uncompress Archive* legt fest, ob das zu generierende Format als komprimiertes Archiv abgelegt werden soll oder alternativ in das Dateisystem generiert wird. Dies ist vor allem dann relevant, wenn Pakete

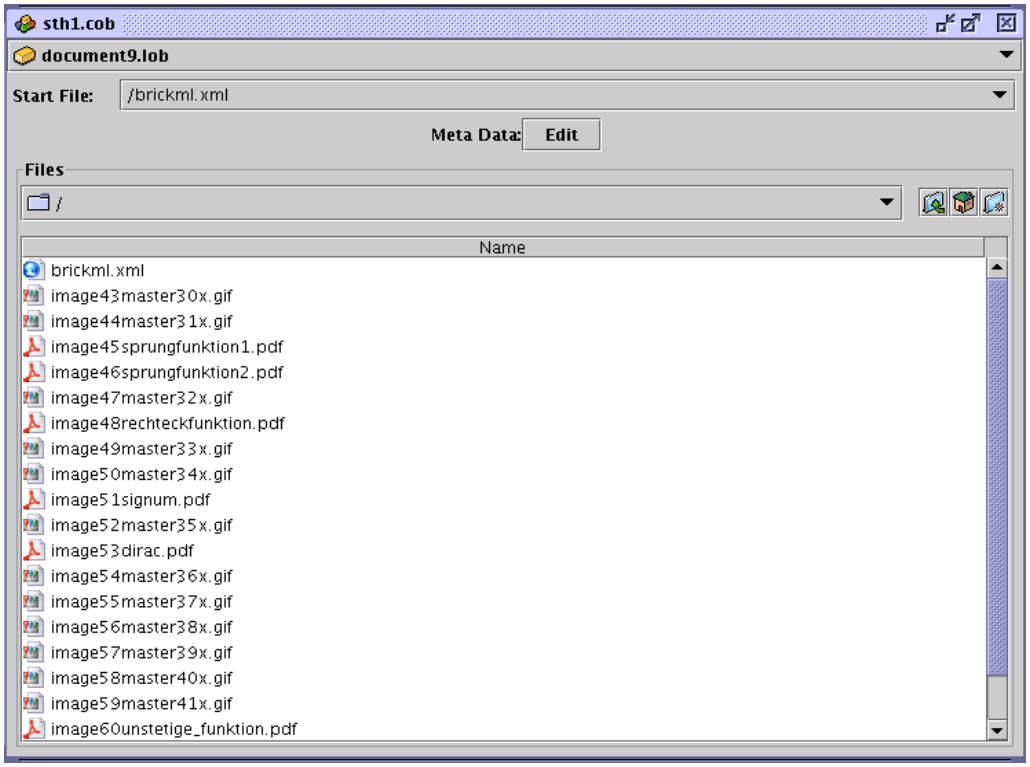


Abbildung 6.2: Ansicht des Bausteinfensters

nach dem IMS Content Package Standard erzeugt und direkt von einem LMS eingeladen werden sollen [Bungenstock03a]. Ferner existieren Formate, wie PDF-Dokumente, bei denen dieser Schalter nicht angezeigt wird, weil er eine, für dieses Format, irrelevante Option darstellt. Im mittleren Bereich werden Prozessoren ausgewählt, die zusätzliche Funktionen für den Übersetzungsprozess anbieten. Um den Übersetzungsvorgang zu überwachen, lässt sich ein Ausgabe-Fenster öffnen, das den Vorgang protokolliert.

6.1.2 Formatierbare Bausteine

Abschnitt 3.2 hat bereits gezeigt, dass gängige Modelle zur Beschreibung von Lernmaterialien mit der *Extendable Markup Language (XML)* gesetzt werden. Aufgrund der wohlgeformten Struktur und der hierarchischen Anordnung der Elemente, kann eine Realisierung des Content-Modells als bijektive Abbildung der Elementstruktur auf XML-Elemente stattfinden. XML ermöglicht neben der guten Lesbarkeit auch eine einfache und fehlertolerante Verarbeitung durch technische Systeme. Sollen formatierbare Bausteine allerdings durch ein technisches System konstruiert werden, sind andere Mechanismen notwendig. Mit dem so genannten *OO-Binding* wird das den Bausteinen zugrunde liegende Modell auf eine physikalisch im Speicher liegende dynamische Datenstruktur abgebildet. Im Folgenden werden diese beiden Varianten näher erläutert.

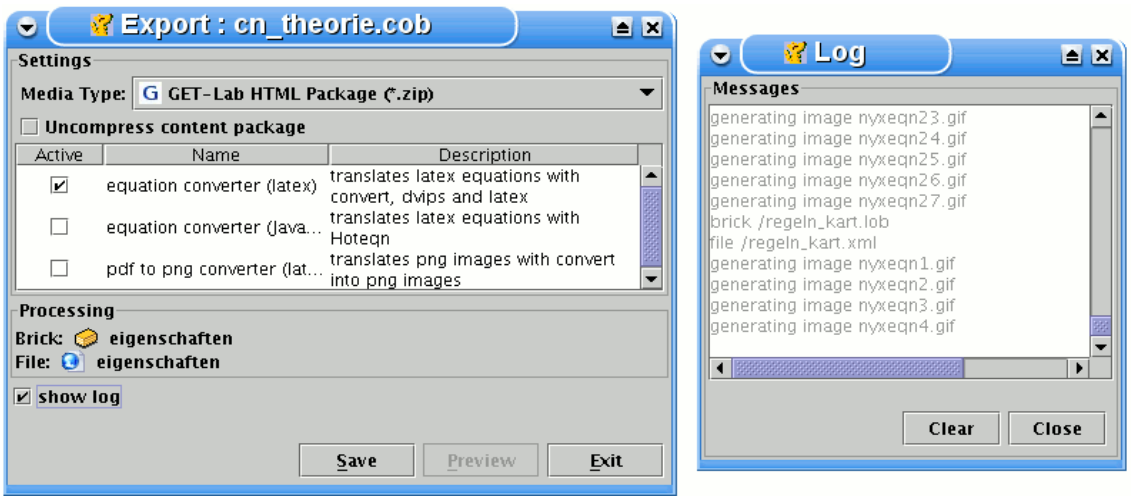


Abbildung 6.3: Der Export-Dialog

BrickML (XML-Binding)

Jeder Baustein besitzt genau eine XML-Datei, die das Content-Modell speichert. Diese Datei wird als Start-Datei definiert und vom Übersetzungsprozess verarbeitet. Jedes Element des Modells lässt sich daraufhin in eine entsprechende XML-Form bringen. In Abbildung 5.5 wurde bereits der syntaktische Aufbau des Wurzelelements *lob* vorgestellt. Das korrespondierende XML-Element wird in spitzen Klammern notiert `<lob></lob>`. Besitzen Elemente Attribute, so werden sie, entsprechend der XML-Spezifikation, hinter den Namen des Elements geschrieben und deren Werte mit Anführungsstrichen umschlossen (vgl. hierzu Abschnitt 3.2.1). Für den Titel des Bausteins ergibt sich die folgende Notation: `<lob title='Sinusförmige ...'></lob>`. Genau wie das `lob`-Element lassen sich alle auftretenden Elemente in einer solchen Repräsentation notieren. Tabelle 6.2 zeigt die Umsetzung der in Kapitel 5.2.1 definierten Strukturelemente in BrickML-Elemente.

Strukturelement	XML-Binding (Beispiel)	Beschreibung
lob	<lob> ... </lob>	Lernobjekt
highlight	<h> ... </h>	Hervorhebung
italic	<i> ... <i/>	Kursiv
typewriter	<t> ... </t>	Schreibmaschine
br	 	Zeilenumbruch
link	<link target=”any url”/>	URL-Verweis
ws	<ws/>	Leerzeichen
code	<code> ... </code>	Quellcode
anchor	<anchor name=”any name”/>	Markierung

nächste Seite

Strukturelement	XML-Binding (Beispiel)	Beschreibung
ref	<ref name=”anchor name”/>	Referenz
table	<table border=1 widht=”any size”> ... </table>	Tabelle
cell	<cell> ... </>	Tabellen-Feld
row	<row align=”center”> ... </row>	Tabellen-Zeile
column	<column> ... </column>	Tabellen-Spalte
heading	<heading> ... </heading>	Tabellen-Kopfzeile
itemize	<itemize> ... </itemize>	Nummerierung
enumerate	<enumerate> ... </enumerate>	Aufzählung
item	<item> ... </item>	Element einer Aufzählung oder Nummerierung
mmo	<mmo type=”applet” width=”any size” src=”any URL”/>	Multimedia-Element
param	<param/>	MMO-Parameter
image	<image width=”any size” src=”any URL”/>	Image-Element
quiz	<quiz> ... </quiz>	Leitet ein Quiz ein
text	<text> ... </text>	Quiz-Beschreibung
question	<question> ... </question>	Gruppiert Fragen
answer	<answer> ... </answer>	Definiert eine mögliche Antwort
shortquiz	<shortquiz> ... </shortquiz>	Leitet ein Quiz mit Lösung ein
sqquestion	<sqquestion> ... </sqquestion>	Gruppiert Fragen
sqanswer	<sqanswer> ... </sqanswer>	Definiert eine mögliche Antwort
solution	<solution> ... </solution>	Definiert die Lösung
definition	<definition title=”any Text”> ... </definition>	Definition
conclusion	<conclusion> ... </conclusion>	Zusmammenfassung
lemma	<lemma> ... </lemma>	Lemma
notice	<notice> ... </notice>	Anmerkung
hint	<hint> ... </hint>	Hinweis
example	<example> ... </example>	Beispiel

Tabelle 6.2: Abbildung Strukturelemente auf XML-Elemente

Das folgende Beispiel zeigt exemplarisch einen Auszug aus einer Lehreinheit, die mit der Bausteinbeschreibungssprache BrickML formuliert wurde.

Listing 6.1: Ein einfaches BrickML-Beispiel

```
<?xml version=”1.0” encoding=”ISO−8859−1” ?>
<lob title=”Sinusförmige Wechselgrößen” xml:lang=”de”>
```

```
<p>Betrachtung einer beliebigen sinusförmigen Wechselgröße
  <formula inline="true">x(t):</formula>

  <formula inline="false">x(t)=\hat{x}\sin(\alpha+\varphi_0)</formula>
  <formula inline="false">=\hat{x}\sin(\omega t+\varphi_0),</formula>

  wobei:
  <table border="0">
    <row>
      <col><formula inline="true">x(t)</formula></col>
      <col>Momentan– oder Augenblickswert der Wechselgröße</col>
    </row>
    <row>
      <col><formula inline="true">\hat{x}</formula></col>
      <col> Maximalwert der Wechselgröße (Scheitel–, Spitzenwert),
        (engl. maximum amplitude)</col>
    </row>
  </table>
</p>
</lob>
```

Dieses Beispiel zeigt das Element `<p>`, das einen Textabschnitt als Absatz deklariert, mehrere Formeln, die unterschiedliche Formate speichern, sowie eine Tabelle, die aus mehreren Zeilen und Spalten besteht.

Die Bestimmung der syntaktischen Struktur ist zwar die Grundvoraussetzung für den Aufbau von Bausteinen, reicht aber bei weitem nicht aus. Neben der syntaktischen Struktur ist der semantische Aufbau ebenso wichtig, weil nur sinnvoll aufgebaute Inhalte eindeutig erkannt und richtig übersetzt werden. Hierzu existiert ein XSD-Schema (vgl. Abschnitt 3.2.1), das die eindeutige Vater-Kind-Beziehung zwischen den Elementen definiert. Darüber hinaus werden die erlaubten Attribute spezifiziert, indem ihnen Basistypen, wie z. B. *Integer*, *String* oder *Boolean*, zugewiesen werden. Elemente lassen sich in einem XSD-Schema von anderen Element-Definitionen ableiten, indem, ähnlich wie bei der Objektorientierung, die Definition des abzuleitenden Elements in der neuen Definition automatisch gültig ist. Ableitbare Elemente werden in der XSD-Definition mit dem Element `complexeType` versehen. Als Beispiel für eine XML-Schema Element-Definition wird das in Listing 6.1 verwendete `table`-Element vorgestellt. Listing 6.2 zeigt das Konstrukt zur semantischen Definition einer Tabellenstruktur. Demnach besteht eine Tabelle aus einer Sequenz `column`-Elementen[1], gefolgt von mindestens einer Zeile. Zusätzlich können die optionalen Attribute *border*, *width*, *heigt*, *caption* und *unit* hinzufügt werden.

[1] Die Spalten-Elemente speichern wichtige Merkmale der Tabellen, enthalten jedoch selbst keinen Inhalt. Der Zelleninhalt wird in `cell`-Elementen innerhalb des `row`-Elements gespeichert wird.

Listing 6.2: BrickML Tabellen-Definition mit XML-Schema

```
<xsd:element name="table" minOccurs="0" maxOccurs="1"
type="tableType"/>

<xsd:complexType name="tableType">
  <xsd:sequence>
    <xsd:element name="column" type="columnType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="row" type="rowType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="border" type="xsd:boolean" use="optional"/>
  <xsd:attribute name="width" type="xsd:integer" use="optional"/>
  <xsd:attribute name="height" type="xsd:integer" use="optional"/>
  <xsd:attribute name="caption" type="xsd:string" use="optional"/>
  <xsd:attribute name="unit" type="xsd:string" use="optional"/>
</xsd:complexType>
```

OO-Binding

Das bisher beschriebene XML-Binding begünstigt die einfache Lesbarkeit des Inhalts und stellt außerdem die kontextfreie Verarbeitung sicher. Damit dieses Modell von einem Prozess verarbeitet werden kann, wird ein Speichermodell benötigt, das den aus Sicht des Programmierers praktikablen Umgang gewährleistet. Gerade für eine derart komplexe Datenstruktur bieten sich die Mittel der Objektorientierten Programmierung (OOP) an. Jedes Element wird als selbständige Klasse definiert und kapselt die relevanten Eigenschaften eines Modellknotens. Für jedes Element wird eine korrespondierende Klasse angelegt, deren Membervariablen den Element-Attributen entsprechen und ihre Unterelemente, durch öffentliche Methoden der Klasse, zugänglich sind. Auf diese Weise wird gewährleistet, dass der Zustand der Objektstruktur konsistent ist. Diese Datenstruktur wird nicht nur zum Navigieren des Modells verwendet, sondern bietet ebenso die Möglichkeit zur Konstruktion einer Objektstruktur, die zur Erstellung eines kompletten Content-Modells für Bausteine führt. Dieses ist insbesondere bei der automatischen Erzeugung von Bausteinen wichtig und wird von Lade-Routinen (vgl. Abschnitt 5.4.2) benötigt. Ein anderes Beispiel ist die Abbildung einer XML-Datei auf Objektstrukturen. Dokumente werden auf diese Weise in den Speicher geladen und lassen sich dort dynamisch modifizieren. Dieses Verfahren wird z. B. vom Übersetzungsprozess verwendet, um Elemente zu suchen und deren Zustand ggf. anzupassen.

Für die Implementierung des OO-Bindings wurde das Rahmenwerk Java Document Object Model (JDOM)[JDO05] eingesetzt. Gängige XML-Parser, wie z. B. Xerces [Xer05], konstruieren zwar eine Objektstruktur (Document Object Model DOM), jedoch auf Basis einer einzigen Element-Klasse. JDOM hingegen implementiert das in Gamma vorgestellte [Gamma95] Fabrikmuster, mit dem zu jedem eingelesenen XML-Element, anhand seines

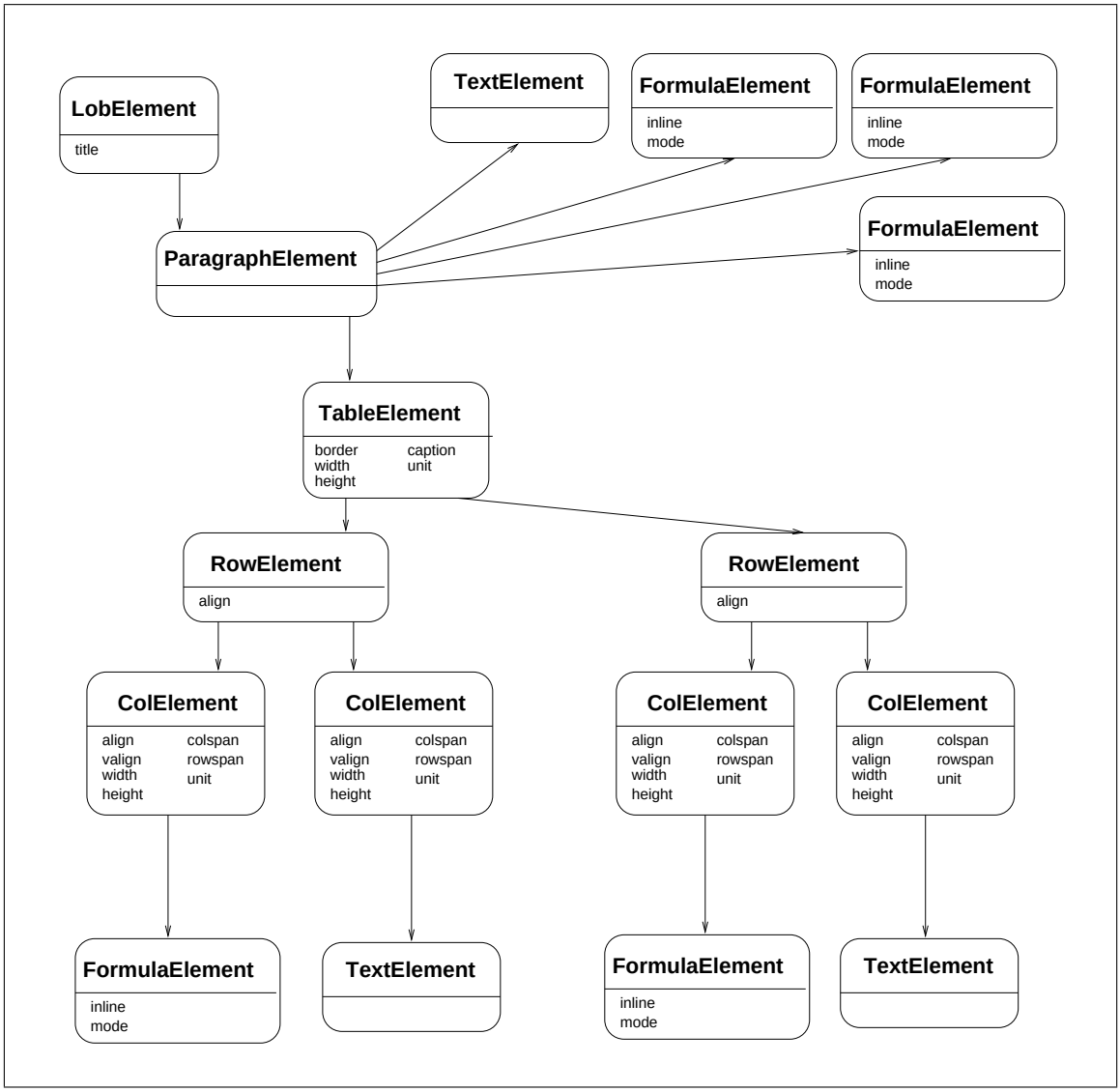


Abbildung 6.4: UML-Diagramm für ein OO-Binding-Beispiel

Namens, ein Objekt einer spezialisierten Klasse instantiiert wird. Ohne diesen Mechanismus müsste über komplizierte und fehleranfällige String-Kommandos der Objekt-Zustand erfragt werden.

Das im vorigen Abschnitt präsentierte XML-Listing wird als OO-Binding in Abbildung 6.4 gezeigt. Das Wurzelelement `<lob/>` wird auf die Klasse `LobElement` abgebildet. Des weiteren entspricht das `ParagraphElement` dem `<p/>` Element, das über eine entsprechende Methode dem `LobElement` zugeordnet wird. Eine weitere Methode weist dem `LobElement` den Titel zu, der als Attribut im XML-Element enthalten ist. Dieser Vorgang setzt sich fort, bis das gesamte Dokument im Speicher aufgebaut worden ist.

6.1.3 Formatierbare Kurse

Als Grundlage für Formatierbare Kurse dient das IMS Content Package 2.3.2, das, durch das in seiner Spezifikation festgelegte *XML-Binding*, den Aufbau von Kursstrukturen definiert. Von einer Realisierung durch ein Datenbankschema wurde an dieser Stelle abgesehen, weil Kurse – ähnlich wie PDF-Dokumente – unabhängig von Datenbanken behandelt werden sollen, damit sie während des Entwicklungsprozesses von Autoren bzw. Autorinnen per E-Mail bzw. über einen gemeinsamen Dateiserver ausgetauscht werden können. Die IMS-Kursstruktur besteht im Wesentlichen – wie bereits geschildert – aus einer verschachtelten `Item`-Struktur, die mit exakt einer Ressource verknüpft werden können. Ist eine solche Referenz vorhanden, so enthält das `Resource`-Element den Inhalt des Kursknoten. In Abschnitt 5.3 wurde bereits erörtert, dass `Item`-Elemente Formatierbare Bausteine oder auch Formatierbare Kurse referenzieren, die selbst Manifeste und Kursstrukturen enthalten können. Damit sich diese von einem technischen System erkennen und verarbeiten lassen, wird ein Mechanismus zur Identifikation benötigt. Das IMS hat für den Fall, dass `type`-Attribut für `Resource`-Elemente reserviert, welches im Standardfall auf den Wert `webcontent` gesetzt wird, um anzuzeigen, dass es sich um eine Ressource einer Web-Präsentation handelt. Im Folgenden werden die für Formatierbare Bausteine relevanten Typen vorgestellt:

- **webcontent**: Das Attribut *webcontent* kennzeichnet normale *Assets*, wie z. B. HTML, JPEG oder JAR Dateien.
- **brickml**: Bei dieser Ressource handelt es sich um die XML-Datei, die als Grundlage der Bausteine dient.
- **fb**: Anhand des *fb*-Attributs erkennt der Übersetzer, dass es sich um einen formatierbaren Baustein handelt.
- **fk**: Anhand des *fk*-Attributs erkennt der Übersetzer, dass es sich um einen formatierbaren Kurs handelt.

Listing 6.3 zeigt das einfache Manifest eines Grundlagenkurses der Technischen Informatik. Das Manifest besteht aus Organisationen mit verschachtelten `Item`-Elementen und zusätzlichen Ressourcen. Der Kursknoten *Schaltnetze* referenziert die Ressource mit dem Identifier `Resource`, die durch das `href`-Attribut auf einen Unterkurs verweist. Die anderen Kursknoten, wie der Eintrag *Wechselstromschaltungen*, sind hingegen einfache Bausteine. Sie enthalten nur den für den jeweiligen Kursknoten relevanten Lerninhalt.

Listing 6.3: Beispiel eins Manifests für einen Grundlagenkurs Technische Informatik.

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns="http://www.imsproject.org/xsd/imscp_rootv1p1p2"                2
  xmlns:adlcp="http://www.adlnet.org/xsd/adl_cp_rootv1p1"
   identifier ="Manifest3">
```

```
<organizations>                                                          5
  <organization identifier ="Organization">
    <title>Technische Informatik</title>
    <item identifier ="Item3" identifierref ="Resource1>                 8
      <title>Schaltnetze</title>
    </item>
    <item identifier ="Item8" identifierref ="Resource2">                11
      <title>Wechselstromschaltungen</title>
      <item identifier ="Item13" identifierref ="Resource3">
        <title>Sinusförmige Wechselgrößen</title>                        14
      </item>
      <item identifier ="Item14">
        <title> Mittelwerte sinusförmiger Zeitfunktionen</title>         17
      </item>
      <item identifier ="Item9" identifierref ="Resource4">
        <title>Arithmetische Operationen zweier sinusförmiger            20
            Wechselgrößen gleicher Frequenz</title>
      </item>
    </item>                                                              23
  </organization>
</organizations>
<resources>                                                              26
  <resource type="fk" identifier ="Resource1 href="/actos.cob"
      adlcp:scormtype="sco"><file href="/actos.cob"/>
  </resource>                                                            29
  <resource type="fb" identifier ="Resource2" href="/wechselstrom.lob"
      adlcp:scormtype="sco"> <file href="/wechselstrom.lob" />
  </resource>                                                            32
  <resource type="fb" identifier ="Resource3" href="/wechsel_sin.lob"
      adlcp:scormtype="sco"><file href="/wechsel_sin.lob" />
  </resource>                                                            35
  <resource type="fb" identifier ="Resource4" href="/wechsel_ao.lob"
      adlcp:scormtype="sco"> <file href="/wechsel_ao.lob" />
  </resource>                                                            38
</resources>
</manifest>
```

Verarbeitung

Diese Struktur ist jedoch noch nicht für Lehrzwecke nutzbar, da die Bausteine und Kurse erst zu einem einzigen Kurs zusammengesetzt werden müssen. Hierfür wird der in Listing 5.1 eingeführte Algorithmus implementiert. Zuerst muss der Kurs geöffnet und das enthaltene Manifest mit dem in [Bungenstock04] vorgestellten OO-Binding eingelesen werden. Es unterstützt die dynamische Verarbeitung und die Aggregation von Kursen.

Entsprechend der Algorithmus-Definition wird der geöffnete Kurs nach Bausteinen

und Subkursen durchsucht. Das Zusammenführen zweier Kurse ist erst möglich, wenn der Subkurs entfernt wird und stattdessen sein Subkurses eingesetzt wird. Zu diesem Zweck werden Submanifeste verwendet, die in einem zusätzlichen Manifest definiert werden. Submanifeste werden über Referenzen von den Kursknoten eingebunden und lassen sich daher an verschiedenen Stellen der Kursstruktur verankern. Dieses Verfahren lässt sich rekursiv fortfahren, bis der komplette Kursbaum aufgebaut ist. Trifft der Übersetzer auf einen Baustein, werden die dort enthaltenen Assets in den zu erzeugenden Kurs kopieren und als Ressourcen eingetragen.

Die Transformation besteht aus drei Schnitten: Zuerst wird der aggregierte Kurs zu einem einzigen Kurs zusammengebaut, der lediglich ein Manifest enthält. In einem zweiten Schritt wird daraufhin der Inhalt der BrickML-Dateien transformiert. Der letzte Schritt dient zur Erzeugung des Ausgabeformats, dessen Zeit- und Kostenaufwand abhängig von seiner Komplexität variiert.

Für die Realisierung der drei Verarbeitungsschritte wird das *Abstrakte Fabrik*-Modell verwendet [Gamma95]. Anhand einer konkreten Fabrik ermöglicht das Muster einem Klienten-Objekt die Konstruktion einer speziellen Datenstruktur. Um andere Konstruktionen zu erzeugen, muss lediglich die Instanz der Fabrik ausgetauscht werden. Obwohl jede Fabrik ihre individuell ausgeprägten Klassen besitzt, sind deren Schnittstellen dennoch einheitlich, um weitere Kursvarianten zu erzeugen.

Für den Übersetzungsprozess werden die zwei Fabriken `CPCSFactory` und `NaviCSFactory` benötigt. Erstere erzeugt einen Kurs auf Basis einer verschachtelten Bausteinstruktur und die zweite transformiert die Kursstruktur in eine einfache Datenstruktur, aus der die Navigation generiert wird. Diese wird insoweit benötigt, um komplexe Manifeste, bestehend aus vielen verschachtelten Submanifesten, zu einer darstellbaren Struktur zusammenzubinden. Die abstrakte Klasse `CSFactory` erzeugt entsprechend der IMS-Content Package Spezification (vgl Kapitel 2.3.2) Objekte der Klasse `Course`, `Item` und `Resource`. Da diese Klassen abstrakt sind, kann keine von ihnen zur Laufzeit instantiiert werden. Nur Objekte abgeleiteter Klassen können bei der Content Package-Erzeugung instanziiert werden, wie z. B. `CPCourse`, `CPItem` und `CPResource`.

Die Klasse `CPConverter` kann durch die polymorphe Eigenschaft der Verwebung beispielsweise ein `Course`-Objekt anfordern, erhält jedoch, ohne dessen konkrete Implementierung zu kennen, die abgeleitete Klasse `CPCourse`. Des weiteren lassen sich mit diesem Muster andere Repräsentationen, wie z. B. verwandte Standards oder eine vereinfache Textansicht, generieren.

Die Fabrik wird durch die Klasse `CPConverter` verwendet. Sie implementiert den in Listing 5.1 vorgestellten Algorithmus und nutzt während der Abarbeitung die Fabrik, um einen neuen übersetzten Kurs zu konstruieren. Jeder neu entdeckte Baustein zwingt die Klasse `CPConverter`, ihn zu öffnen und das dort enthaltene Manifest rekursiv weiter zu verarbeiten.

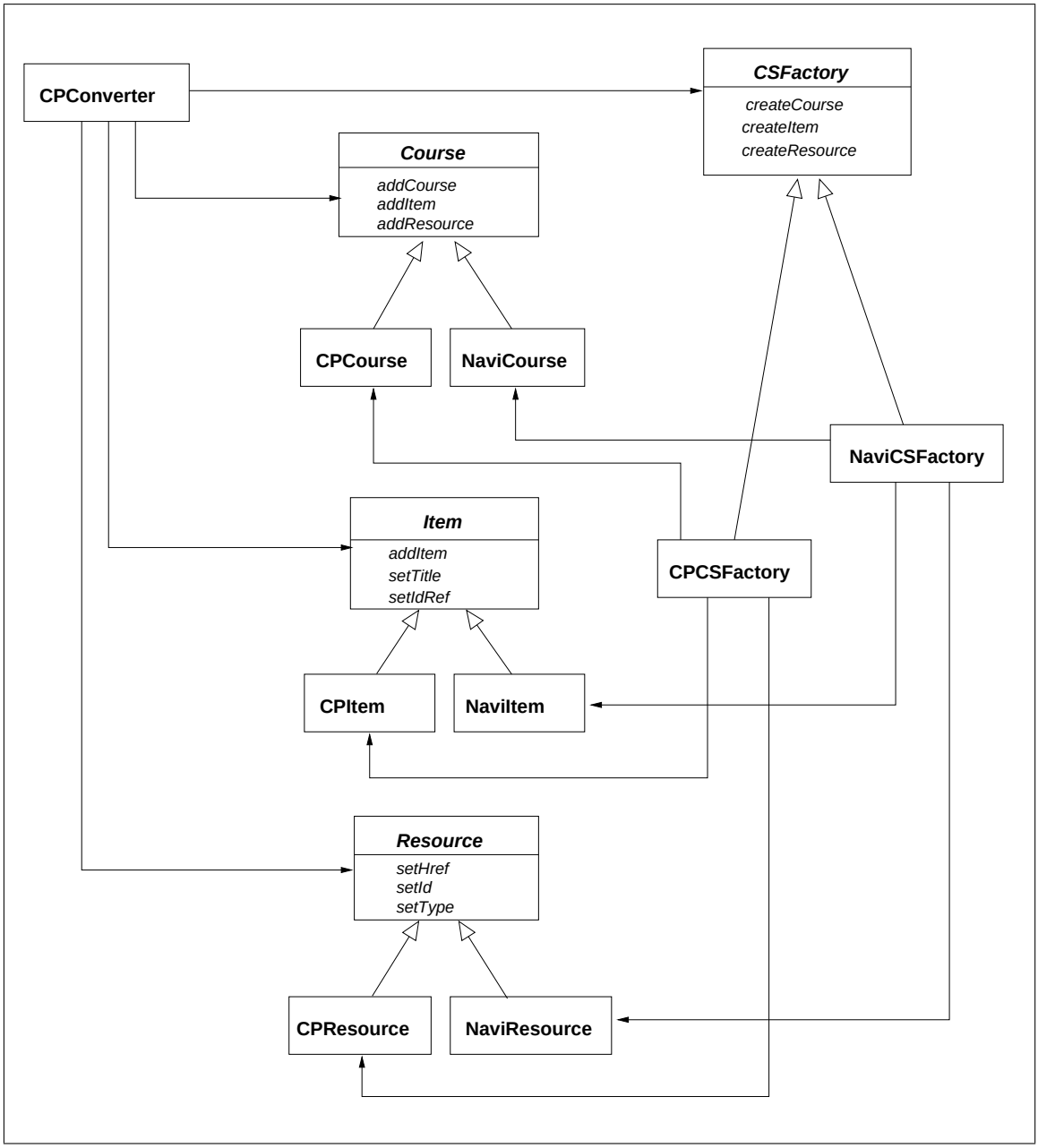


Abbildung 6.5: UML-Diagramm einer abstrakten Fabrik für die Kursumwandlung

6.1.4 Das Kursmodell

In Kapitel 5.3.2 wurde das allgemeine Kursmodell vorgestellt, mit dem die Erzeugung modularer Kursstrukturen erheblich vereinfacht wird. Dieses ist besonders wichtig, wenn bestehende Inhalte, wie z. B. Office-Dokumente, für die Entwicklung neuer Kurse wiederverwendet werden müssen und sie auf eine modulare formatierbare Kursstruktur abgebildet werden sollen. Das Kursmodell bietet Entwicklern und Entwicklerinnen eine abstrakte Schnittstelle, anhand derer sich komplett verschachtelte Kurse mit einfachen Mitteln

erzeugen lassen. Der Mehrwert folgt aus der Reduktion der Komplexität, während der Entwicklung von Laderoutinen für Fremdformate.

OO-Bindung Kursmodell

Nachdem das Kursmodell theoretisch betrachtet wurde, wird in diesem Abschnitt die konkrete Umsetzung im Autorenwerkzeug *Lyssa* vorgestellt. Bedingt durch das realisierte *OO-Binding* existiert zu jedem, in Abbildung 5.15 gezeigten Kursknoten genau eine Objektinstanz.

Ausgehend vom Kursobjekt, können Lektionen bzw. Kapitel dynamisch in die Baumstruktur eingeordnet werden. Hierbei spielt es keine Rolle, ob eine Lektion zuerst konstruiert wird und im Anschluss in die Kursstruktur eingehängt wird oder ob sie zuerst in die Kursstruktur eingebunden und dann weiter ausgebaut wird. Diese, auf den ersten Blick triviale, Forderung führt jedoch bei genauerer Betrachtung zu einigen Hindernissen. Das Kursmodell sieht vor, dass jede Lektion, festgelegt durch das Attribut `CourseObject`, entweder als Baustein realisiert wird oder dem Kursknoten eines übergeordneten Kurses zugeordnet wird. Wird ein Kursobjekt mit diesem Attribut versehen, folgt die Erstellung eines selbständigen, im Dateisystem gespeicherten IMS Content Package. Wird der Kursknoten nach seiner Erzeugung in die Kursstruktur eingepflegt, werden alle weiteren Assets und Subkurse in das Content Package kopiert. Die Lektion lässt sich zwar autark aufbauen, ihr Inhalt muss jedoch zu einem späteren Zeitpunkt in den übergeordneten Kurs kopiert werden. Problematisch ist eine nicht als Kursobjekt gekennzeichnete Lektion. In diesem Fall muss die Objektinstanz solange im Speicher gehalten werden, bis die Zuordnung zu einer überordneten, als Kursobjekt deklarierten Lektion stattfindet. Inhalte werden, genau wie Lektionen, dynamisch konstruiert und die Objektinstanzen miteinander verknüpft. Hierbei ist die Verwendung von Content Packages ebenso bedeutsam, wie bei der Aggregation von Lektionen. Einige Objektinstanzen, unter anderen Videos und Abbildungen, die im Dateisystem abgelegt werden, müssen ebenfalls Ressourcen verwalten. Werden diese Objekte, denen Dateien zugeordnet sind, vor der Zuordnung zu einer Lektion erzeugt, fehlt das zugehörige Content Package, d. h. die Dateien können nicht als Assets in das Package kopiert und in das Manifest eingetragen werden.

Durch diesen Ansatz werden zwei mögliche Erzeugungsvarianten für die Abbildung auf das Kursmodell angeboten. Einerseits kann das zu importierende Format eingelesen werden und das entsprechende Kursmodell *top-down* aufgebaut werden. Wird mit dem Kursknoten begonnen, so ist immer ein zugrunde liegendes Content Package vorhanden, in das weitere Content Packages und Assets kopiert werden können. Auf der anderen Seite kann das Kursmodell nach dem *bottom-up*-Verfahren, beginnend mit seinen Blättern, konstruiert werden. Letztere Variante ist dann notwendig, wenn das einzulesende Quellformat zuerst rekursiv traversiert werden muss, um es komplett einlesen zu können. Zusätzlich ermöglicht dieser Ansatz eine Granularitätsabschätzung, die zur Erkennung redundanter Kursfragmente, also solche, die keinen Inhalt besitzen, eingesetzt werden kann. Diese Fragmente werden nicht als Baustein modelliert, sondern lediglich als Kursknoten verankert.

Abbildung 6.6 zeigt die sequenzielle Darstellung einer *bottom-up* Kurskonstruktion und

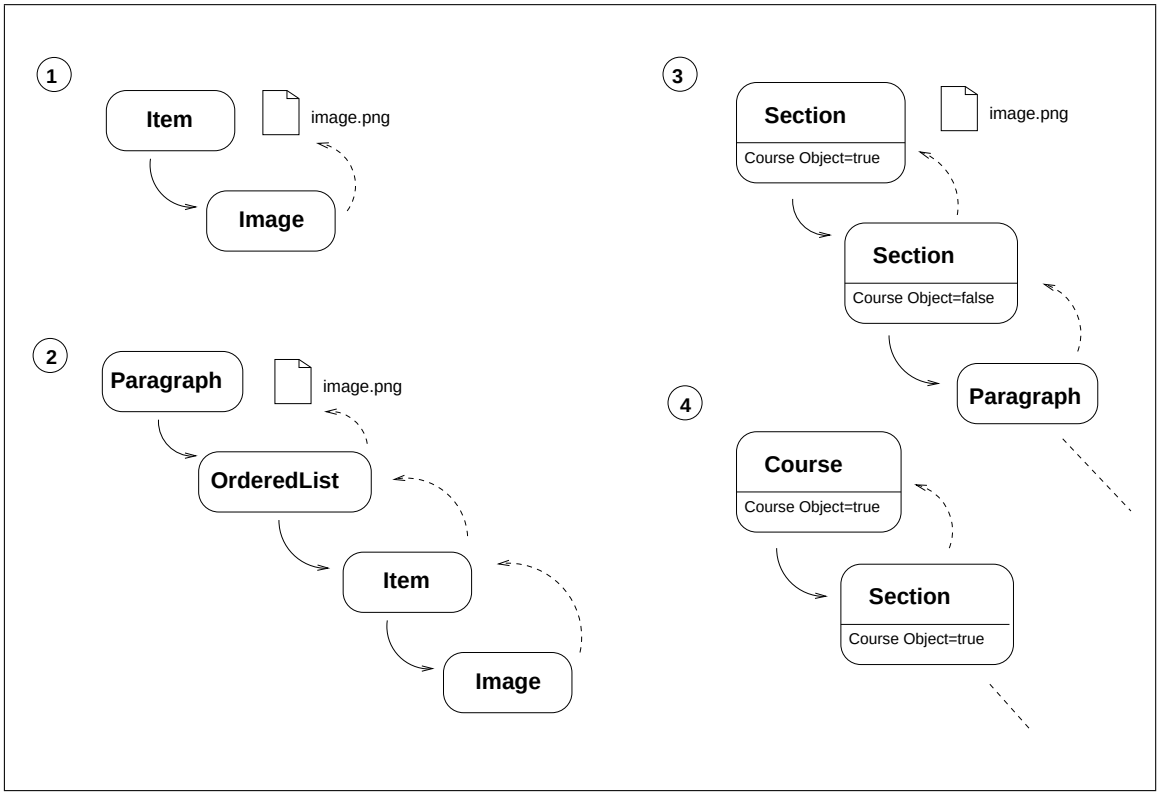


Abbildung 6.6: Sequenzielle Darstellung eines Objekt Diagramms zur Veranschaulichung einer *bottom-up* Kurskonstruktion

den Umgang mit Datei-Ressourcen. Im ersten Schritt wird ein `Image`-Objekt erzeugt, das einem `Item`-Objekt zugeordnet wird. An dieser Stelle überprüft das `Image`-Objekt, ob bereits ein übergeordnetes Content Package existiert. Ist dies nicht der Fall, so übergibt es die Datei-Information an sein `Item`-Objekt. Im zweiten Schritt wird der Kurs weiter aufgebaut und die Datei-Information der Abbildung, wie im ersten Schritt, bis zum Paragraphen weitergereicht. Der dritte Schritt verdeutlicht die Übergabe des Paragraphen an eine Lektion. Der Paragraph übergibt die Datei-Ressource nur dann, wenn eine Lektion mit dem Attribut `CourseObject=true` in der Liste seiner Vorfahren vorhanden ist. Trifft dies zu, wird die Datei in das Content Package der Lektion kopiert. In Schritt 4 wird die Lektion mit dem erzeugten Content Package einem Kursobjekt zugeordnet und als neues Asset in dessen Content Packages kopiert. Abschließend erfolgt der Eintrag in das Manifest.

Das zugehörige UML-Klassendiagramm ist in Abbildung 6.10 zu sehen. Die Klasse `CPContainer` fungiert als Verwalter für IMS Content Packages. Sie übernimmt die Erzeugung und Verwaltung der physikalisch vorhandenen Dateien und ermöglicht neben dem Abspeichern auch das Aggregieren von Lektionen. Der Wurzelknoten des Kursmodells ist der Kurs selbst, der durch die Klasse `Course` instantiiert wird. Der Kurs ist abgeleitet von der Klasse `CPContainer` und entspricht einem spezialisierten Content Package. Lektionen sind ebenfalls spezialisierte Content Packages, die sich zu Kompositionen zusammenführen lassen. Der bis an dieser Stelle eingeführte Aufbau der Kursstruktur entspricht im

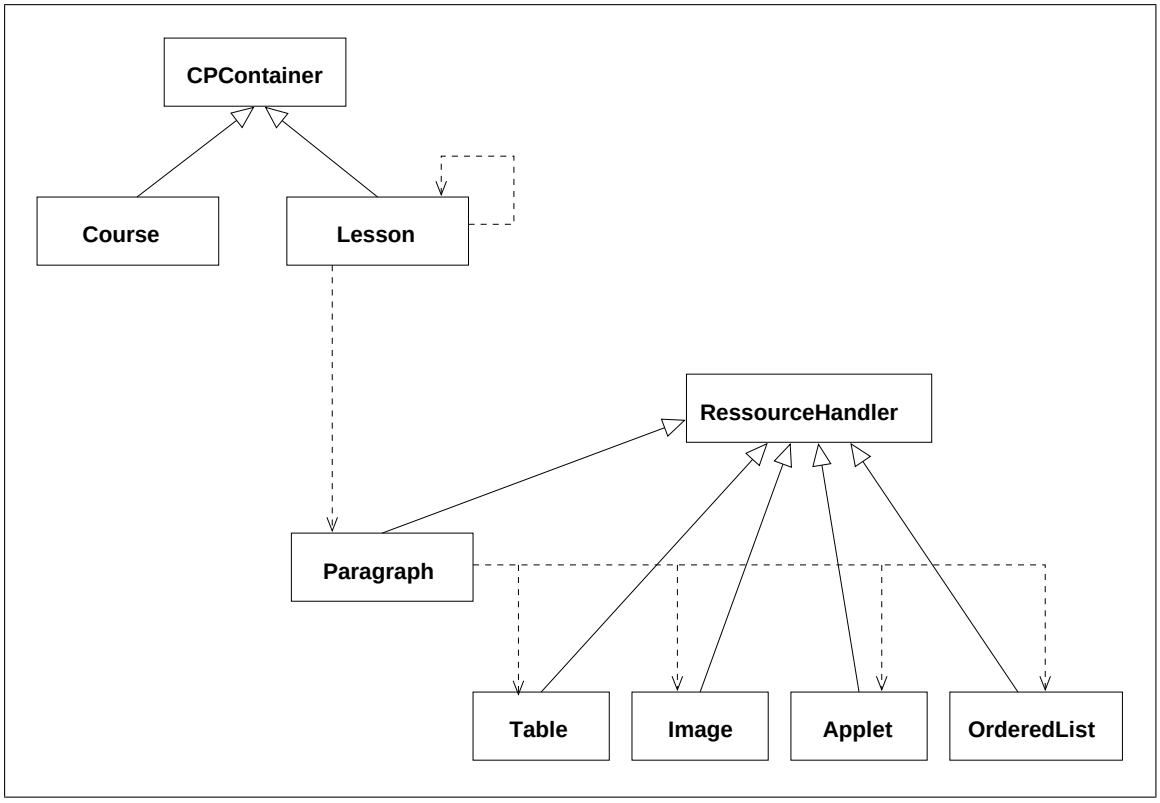


Abbildung 6.7: UML-Klassendiagramm des Kursmodells

Wesentlichen dem eines formatierbaren Kurses. Damit das Kursmodell jedoch vollständig umgesetzt werden kann, müssen Content-Objekte mit den Objekt-Exemplaren direkt verbunden werden, um die angestrebte Transparenz des Modells zu erhalten. Aus diesem Grund lassen sich z. B. `Paragraph`-Objekte direkt in Lektionen einhängen und die Datenstruktur, entsprechend der in Kapitel 5.2 eingeführten Baustein-Struktur, als vollständiger Objektbaum konstruieren. Des weiteren ist die `Lesson`-Klasse für die Anbindung des in Abschnitt 6.2 vorgestellten OO-Bindings an die Kursstruktur verantwortlich.

Es wurde bereits darauf hingewiesen, dass das Kursmodell mit externen Ressourcen hantieren muss und sie insbesondere für die *bottom-up* Konstruktion berücksichtigen muss. Der Lösungsansatz für dieses Problem sieht die Klasse `ResourceHandler` vor, von der jede Klasse des Bausteinmodells abgeleitet wird. Die Klasse `ResourceHandler` kapselt die Verwaltung der Ressourcen und übergibt sie, während der Verknüpfung zweier Objekte, an die Übergeordnete weiter. Dieser Vorgang wird vor dem Klienten bzw. der Klientin des Kursmodells verborgen, so dass der *bottom-up* Ansatz, wie er in Abbildung 6.6 illustriert wird, trivial anwendbar ist.

6.1.5 Abbildung bestehender Formate

Das Kursmodell dient in erster Linie zur vereinfachten Abbildung bestehender Lernmaterialien. Diese Vereinfachung zeichnet sich insbesondere durch die strikte Trennung zwischen

dem konkreten Kursaufbau, mit in sich verschachtelten Bausteinen sowie der Analyse des Quellformats aus. Letzterer soll in diesem Abschnitt weiter untersucht werden. Um die Leistungsfähigkeit des Modells zu demonstrieren, werden die Formate *OpenOffice*, *DocBook* und *Latex* analysiert und die im Prototypen realisierten Abbildungsroutinen vorgestellt.

OpenOffice

OpenOffice hat sich als Open-Source-Variante von dem Office Produkt *StarOffice* abgespalten und wurde seither ständig weiterentwickelt. Eine wichtige Eigenschaft dieser Office-Suite ist die Kompatibilität zu mehreren Betriebssystemen, da, neben der Windows-Version, ebenfalls Varianten für verschiedenen Linux-Derivate existieren. OpenOffice wurde in dieser Arbeit als zu analysierendes Office-Produkt untersucht, weil das zugrunde liegende Dateiformat öffentlich zugänglich ist und es zudem die Auszeichnungssprache XML einsetzt [OOS02]. Außerdem kann es, aufgrund seines großen Fundus an Import-Filtern, als Mediator für andere Formate eingesetzt werden (vgl. Abschnitt 3.1.2).

Damit das Dokument jedoch auf das Kursmodell abgebildet werden kann, muss zunächst seine interne Struktur interpretiert werden. Das OpenOffice.org-XML-File-Format besteht aus den vier Dateien `meta.xml`, `styles.xml`, `content.xml` und `settings.xml`, die zusammen das gesamte Dokument enthalten.

meta.xml enthält Metadaten über das Office Dokument. Hierzu zählen z. B. der Name des Autors und das Erstellungsdatum.

content.xml speichert den Inhalt des Dokuments. Jedes Element besitzt eine Referenz auf einen Style, der die Darstellung beschreibt. Zusätzlich enthält die Datei noch *automatic styles*. Hierbei handelt es sich um Layouteigenschaften, die während der Bearbeitung eines Dokuments Elementen zugeordnet werden.

styles.xml enthält Style-Definitionen für Elemente. Hierzu zählen z. B. Farben, Schrifttypen und Abstände.

settings.xml speichert applikationsspezifische Einstellungen, wie z. B. Druckerinformationen oder Fenstergrößen.

Ein wesentliches Problem bei der Verarbeitung von OpenOffice-Dokumenten ist die fehlende Dokumentstruktur. Kapitel und Unterkapitel werden ausschließlich linear abgespeichert und nicht, wie in diesem Ansatz gefordert, hierarchisch. Dieses ergibt sich aus der Zweckmäßigkeit des Anwendungsmodells. Mit OpenOffice werden Kapitel und Abschnitte quasi grafisch über die Schnittstelle zum Anwender bzw. zur Anwenderin festgelegt. Ein Abschnitt kann mit einem Mausklick in ein Kapitel umdeklariert werden. OpenOffice kennt also kein strukturiertes Dokument, das aus Kapiteln und Abschnitten besteht, sondern es speichert die Präsentation des Dokuments linear ab. Folglich ist eine Kapitelüberschrift eine Textzeile mit einem Hinweis, dass es sich um eine Überschrift mit einer zugehörigen Verschachtelungstiefe handelt. Die Konsequenz, die sich hieraus ergibt, ist, dass Elemente, wie Aufzählungen oder Tabellen, keinem Kapitel oder Abschnitt zugeordnet werden. Dies

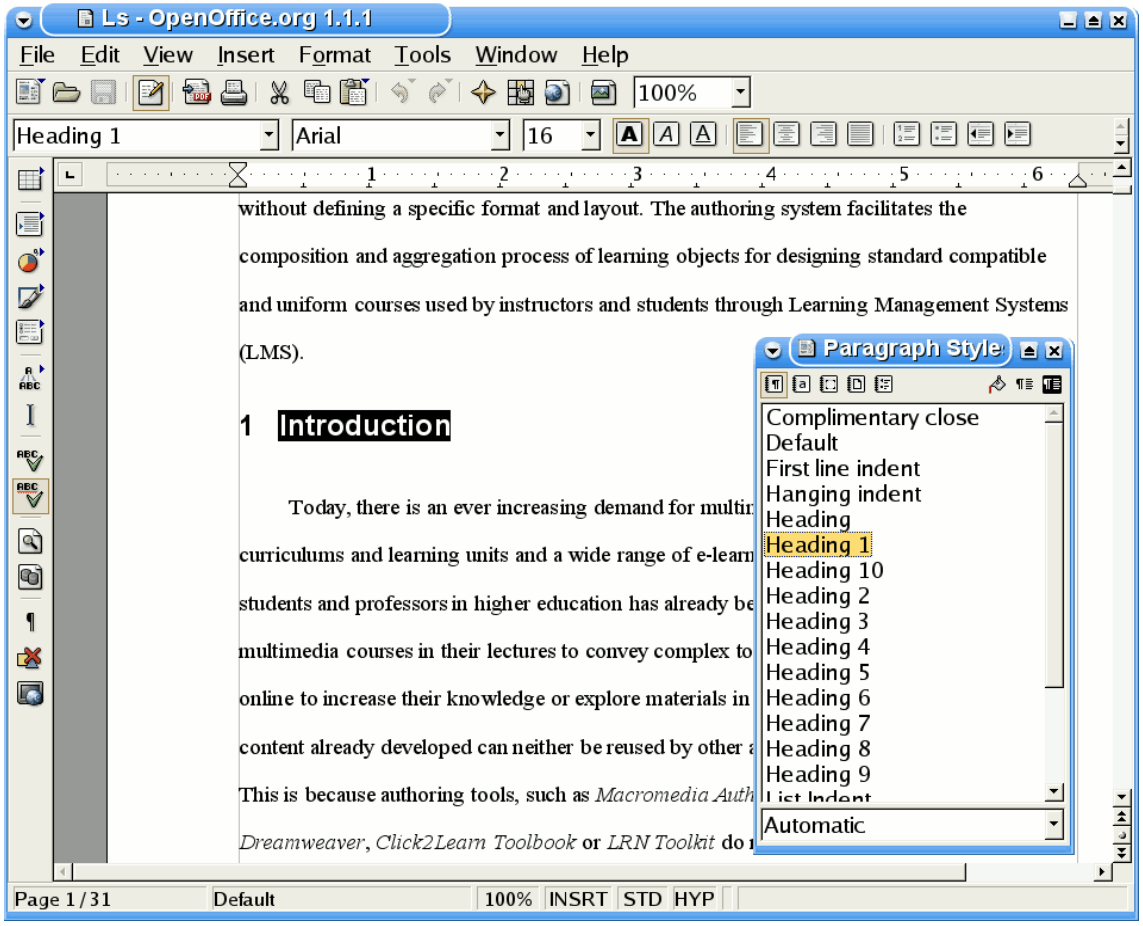


Abbildung 6.8: OpenOffice Screenshot

wirkt sich auf die Komplexität der Dokumentanalyse aus, denn zu jeder in der Analysephase neu gefundenen Überschrift muss das dazugehörige Kapitel ermittelt werden, um die Zuordnung für das Kursmodell herzustellen.

Abbildung 6.8 zeigt einen Bildschirmausschnitt der OpenOffice-Applikation. Damit eine Zeile als Überschrift erkannt werden kann und entsprechend auf einen Baustein abgebildet wird, muss ein *Absatztyp* ausgewählt werden. In der Abbildung sind einige Standardtypen aufgelistet, von denen der Typ *Heading 1* ausgewählt worden ist. Im Übrigen bezeichnet die anstehende Nummer auch gleich die Verschachtelungstiefe des Kapitels. Der *Office-Loader* erzeugt aus jeder Überschrift einen Baustein und fügt ihn der Kursstruktur hinzu. Die Zuordnung zwischen den Office-Elemente und denen die Lyssa anbietet, zeigt Tabelle 6.3.

DocBook

DocBook ist ein weit verbreiteter Standard zum Austausch digitaler Dokumente. Die Akzeptanz von DocBook wächst stetig und immer mehr Office-Produkte-Anbieter, wie StarOffice oder KOffice, stellen Export-Filter für dieses Format bereit. DocBook eignet sich

OpenOffice Element	Kursmodell Element
Image	Image
LineBreak	Break
ListItem	ListItem
OrderedList	OrderedList
Paragraph	Paragraph
Properties	
S	
Span	
Style	Underline, Italic, Emphasis, Typewriter
Styles	
TabStop	
TableCell	Cell
TableColumn	Column
TableColumns	-
TableColumnsHeader	-
Table	Table
TableHeaderRow	-
TableRow	Row
TableRowGroup	
Heading	Section
UnorderedList	UnorderedList

Tabelle 6.3: Abbildung zwischen OpenOffice-Elementen und denen des Kursmodells

zum Schreiben ganzer Skripte und Bücher. Im Rahmen der Diplomarbeit *Entwicklung einer Software-Komponente zur Integration bestehender Lehr- und Lernmaterialien und deren Metadaten in das mαth-kit System* [Sünneli04] wurde die Nutzbarkeit von DocBook untersucht, und analysiert inwiefern der Dokument-Inhalt auf das in dieser Arbeit vorgestellte Kursmodell abgebildet werden kann. Die Analyse ergab zunächst, dass DocBook mehr als 400 Elemente für den Dokumentaufbau vorsieht. Die Verarbeitung des gesamten Wortschatzes von DocBook ist jedoch sehr aufwendig, so dass der reduzierte Wortschatz *Simplified DocBook* [Sim05] ausgewählt wurden. DocBook bietet, aufgrund seiner flexiblen und orthogonalen Struktur, zahlreiche Kombinationsmöglichkeiten für seine Elemente an, wodurch die Entwicklung eines Parsers erheblich erschwert wird. Weiterhin ergab die Analyse, dass Elemente oft in mehreren Varianten vorkommen. So gibt es zu vielen Elementen ein Element mit dem Präfix *Informal*, welches lediglich das Vorhandensein eines Titels kennzeichnet. Solche Eigenschaften werden im Kursmodell zumeist über Attribute angeboten, wodurch die Abbildung mehrerer DocBook-Elemente auf ein Element des Kursmodells notwendig ist. Tabelle 6.4 zeigt in Ausschnitten die Abbildung wichtiger DocBook-Elemente auf jene des Kursmodells.

OpenOffice Element	Lyssa Element	Erklärung
Article, Book, Set	Document	Dokumenttypen
Appendix, Article, Chapter, Dedication, Index, Part, Preface, Reference, Sect1 - Sect5, Section, SimpleSect, SetIndex	Lesson	Strukturierungselemente werden auf Bausteine abgebildet
FormalPara, Para, SimPara	Paragraph	Abbildung mehrerer Paragraphen (DocBook kennt Paragraphen mit Titeln und einfache Paragraphen ohne weitere Blockelemente)
InformalTable, Table	Table	Abbildung einer Tabelle mit und ohne Titel
Figure, Graphic, ImageObject in MediaObject, InformalFigure	Image	Bilder und Grafiken
CalloutList, GlossList, ItemizedList, SegmentedList, SimpleList, VariableList	UnorderedList	Listen
OrderedList	OrderedList	Aufzählungen
Equation, InformalEquation	Equation	Mathematische Gleichungen
Example, InformalExample	Example	Beispiel
ProgramListing	Code	Zur Quellcode-Beschreibung
Emphasis	Emphasis	Hervorhebungen

Tabelle 6.4: Abbildung zwischen OpenOffice-Elementen und dem Kursmodell

Latex

In der Universitätslehre sowie in Wissenschaftsbetrieben wird Latex für die Gestaltung von Skripten, Übungsaufgaben und Veröffentlichungen eingesetzt. Ein großer Vorteil, den Latex gegenüber anderen Dokumenterstellungswerkzeugen aufweist, ist die gute Unterstützung mathematischer Formeln. Im Rahmen des Forschungsprojekts mαth-kit ist es von großem Interesse, bestehende Skripte mit Lyssa einzulesen und bisher verfasste Texte, Grafiken und Übungen, für die Anreicherung mit multimedialen Komponenten, wieder zu verwenden.

Die Implementierung eines Parsers zum Einlesen des Source-Codes und insbesondere zur Verarbeitung von Makros ist jedoch eine zeitaufwändige und fehleranfällige Aufgabe. Aus diesem Grund wird der Einsatz eines Konverters, der das Latex-Format in DocBook

umwandelt, als Lösungsansatz präferiert. Mit dem Latex-Paket *tex4ht* [Goosen99] können aus einem Latex Dokument neben HTML-Seiten auch DocBook-Dateien erstellt werden. Die Verwendung des *tex4ht*-Konverters und der DocBook-Laderoutine ermöglichen es nunmehr, bestehende Latex-Dokumente für die Weiterverarbeitung umzuwandeln. In Abbildung 6.9 ist der *tex4ht*-Konverter zu sehen, der zunächst den Latex-Text bearbeitet. Hierbei werden bereits wichtige Entscheidungen getroffen, wie z. B. die wahlweise Umkodierung der mit Latex gesetzten Formeln in *MathML* [Gurari00] oder Bitmap-Grafiken. Das Ergebnis ist eine DocBook-Datei mit den aus Formeln generierten Grafiken sowie die zu den Latex-Quellen gehörenden Abbildungen. Nun kann der *DocBook-Loader* die soeben erzeugten Dateien öffnen und, mit Hilfe der entwickelten DocBook- und Metadaten-API [Sünneli04], die Abbildung auf das Kursmodell vornehmen. Das Ergebnis ist eine Kursdatei, die den Inhalt der Latex-Datei als Bausteine enthält.

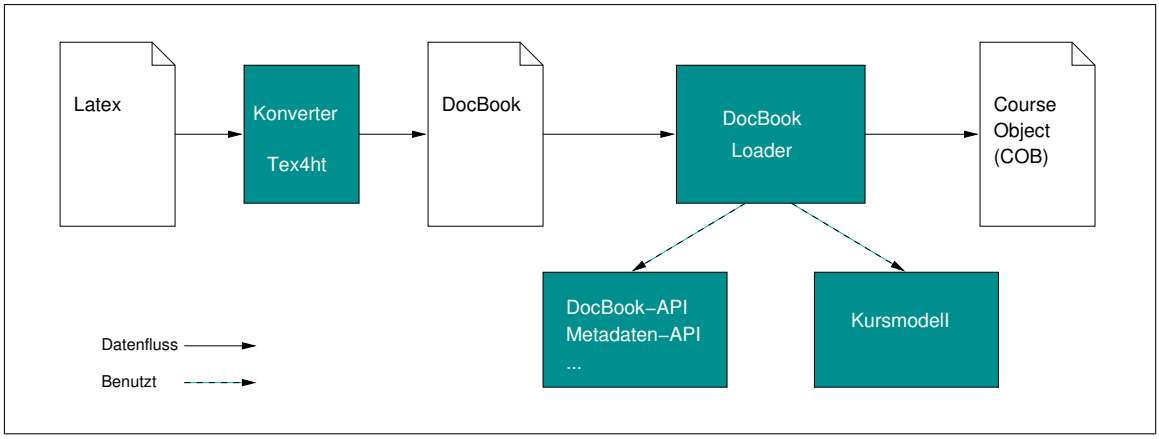


Abbildung 6.9: Datenfluss eines Latex-Dokuments bei der Konvertierung in einen Formatierbaren Kurs.

6.1.6 Transformation

Nachdem zunächst die technische Realisierung der formatierbaren Bausteine und Kurse im Autorensystem Lyssa beschrieben wurde sowie die Abbildung bestehender Inhalte auf die Kursstruktur, soll nachfolgend die Umsetzung der Transformationspakete ausführlich erläutert werden.

Transformationspakete

Das Transformationspaket ist ein komprimiertes ZIP-Archiv, indem die Datei `manifest.xml` den konkreten Aufbau des Pakets definiert. Während der Startprozedur des Autorensystems werden die vorhandenen Transformationspakete eingelesen und durch einen Pool verwaltet. Wird ein Kurs übersetzt, so können die vorhandenen Pakete zur Auswahl angeboten werden und, je nachdem welches Format erzeugt werden soll, das präferierte ausgewählt werden. Das bedeutet, dass alle notwendigen Informationen für die Erstellung des Ziel-Formats im Transformationspaket gekapselt sein müssen. Abbildung

6.10 zeigt das UML-Diagramm eines Transformationspaketes. Dieses besitzt mindestens zwei Transformatoren, einen zur Übersetzung der Bausteine und einen für die Übersetzung der Kursstruktur. Für diese beiden Varianten gibt es die vom `Transformer` abgeleiteten Klassen `NavigationTransformer` und `ContentTransformer`. Zusätzlich existiert eine allgemeine `Transformer`-Klasse, die weitere Aufgaben, wie z. B. die Erzeugung eines HTML-Framesets, übernehmen kann. Hierbei handelt es sich lediglich um die Objektrepräsentationen der Transformatoren. Sie verwenden intern einen `XMLConverter`, der auf verschiedene Arten implementiert werden kann. So ist es möglich, die Extended Stylesheet Language XSL einzusetzen oder Übersetzungsmechanismen direkt in Java zu implementieren. Eine entsprechende Umsetzung findet in der Klasse `JDOMConverter` statt. Außerdem verwendet jeder `XMLConverter` ein `Layout`-Objekt, damit die für eine Übersetzung relevanten Layouteinstellungen berücksichtigt werden.

Allerdings legt das vorgestellte Klassenkonzept nicht fest, welche Konverter von den Transformatoren eingesetzt werden. Transformationspakete sind daher in der Lage, auf unterschiedliche Techniken während der Übersetzung zurückzugreifen.

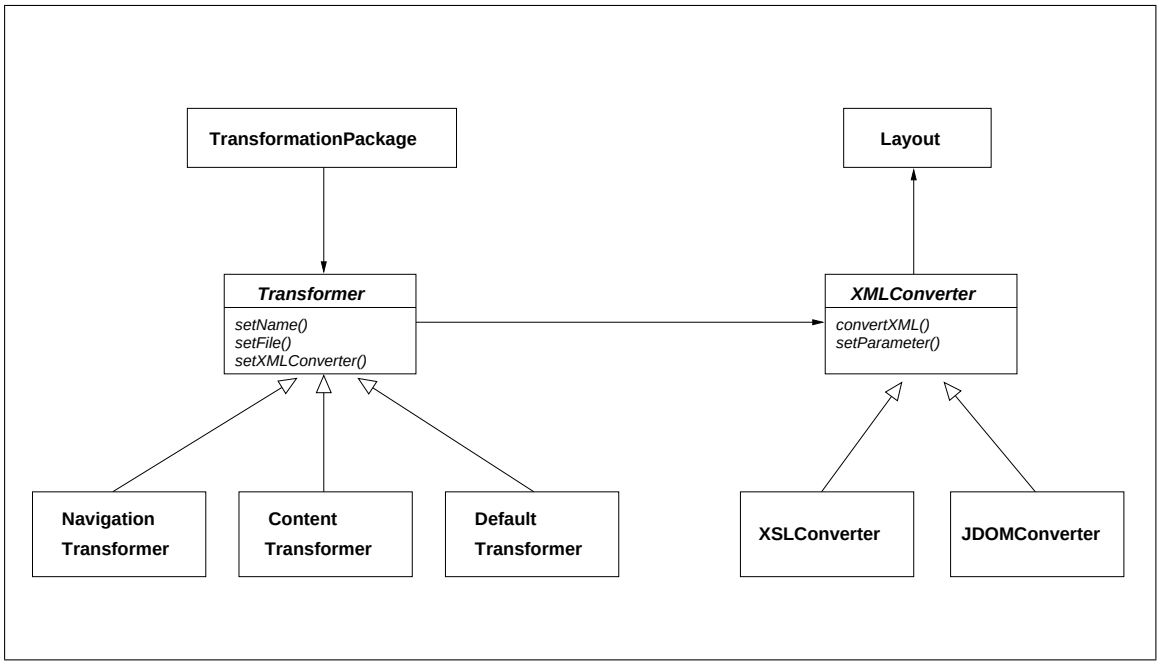


Abbildung 6.10: UML-Diagramm der Transformation Package Klasse

XSL-Konverter

Für die Umwandlung der XML-Inhalte verwendet der XSL-Konverter den XSL-Prozessor *Xalan*[2]. Hierfür wird ein so genanntes XSL-Stylesheet 3.2.1 benötigt, indem die Übersetzungsregeln aufgeführt werden. Für jedes in der Quelle auftretende Element sollte eine korrespondierende Übersetzungsregel angeboten werden, die in mehreren Dateien abgelegt

[2]http://xml.apache.org

und dem XSL-Prozessor zur Verarbeitung übergeben werden kann. Aus diesem Grund sind die Stylesheet-Dateien Bestandteil des Transformationspaketes und werden im Manifest berücksichtigt.

Im Folgenden wird nun ein Beispiel für eine einfache Übersetzungsregel gegeben und deren Abarbeitung demonstriert. Listing 6.4 zeigt eine Regel für das Element `Item`. Trifft der Übersetzer bei der Traversierung des Quelldokuments auf ein solches Element, so werden die zwischen einem `XSL`-Element auftretenden Inhalte, die nicht zum *Namensraum* (vgl. Abschnitt 3.2.1) `xsl:` gehören, in die Ausgabe umgelenkt. In diesem Beispiel wird also ein Listenelement von *BrickML* in eines für Formating Objects (FO) umgewandelt [3].

Listing 6.4: Übersetzungsregel eines Listenelements mit XSL beschrieben

```
<xsl:template match="item">
  <fo:list-item space-after.optimum="6pt">                                  2
    <fo:list-item-label>
      <fo:block start-indent="5pt">
        <xsl:if test="name(..)='itemize'">                                  5
          &#x2022;
        </xsl:if>
        <xsl:if test="not(name(..)='itemize')">                             8
          <xsl:number level="multiple" format="1"/>
        </xsl:if>
      </fo:block>                                                           11
    </fo:list-item-label>
    <fo:list-item-body start-indent="25pt">
      <fo:block>                                                            14
        <xsl:apply-templates />
      </fo:block>
    </fo:list-item-body>                                                    17
  </fo:list-item>
</xsl:template>
```

Im ersten Block wird entschieden, ob es sich bei dem Listenelement um eine Aufzählung oder eine Nummerierung handelt und die entsprechende Darstellung für eine Aufzählung oder eine Nummerierung in die Ausgabe generiert. Der zweite Block befasst sich mit der Verarbeitung des Inhalts, der innerhalb eines Elements definiert ist. Es sind entweder andere Elemente des Quelldokuments oder lediglich einfache Texte. An dieser Stelle wird der Kontrollfluss des XSL-Prozessors rekursiv an eine weitere Übersetzungsregel weitergegeben. Nach erfolgreicher Verarbeitung des Inhalts, der unter einem Item-Element steht, kann die Abarbeitung fortgeführt werden.

[3] Formating Objects Elemente verwenden den Namensraum `fo:`

Java-Konverter

Alternativ lassen sich auch andere Übersetzungstechniken für die Transformation einsetzen. So ist es möglich, selbst einen in Java geschriebenen Konverter auf Basis der JDOM-Bibliothek [JDO05] zu entwickeln. Der Java Konverter traversiert die Baumstruktur, die durch das Quelldokument aufgespannt wird und produziert zu jedem gefundenen Elementknoten eine Ausgabe, die in den Ausgabestrom gelenkt wird. Diese Technik ist dann notwendig, wenn aus den XML-Quellen kein Textformat generiert werden soll, sondern eine Binärdatei, wie z. B. ein Flash-Film.

Java Konverter werden durch mehrere Klassen repräsentiert, die wie XSL-Stylesheets im Transformationspaket gespeichert und im Manifest eingetragen werden.

6.1.7 Framework

Durch den Einsatz von Rahmenwerken wird ein höherer Grad an Wiederverwendbarkeit und Erweiterbarkeit bei der Entwicklung von Softwaresystemen erreicht. Genau dieser Vorteil soll auch beim Einsatz von Transformatoren und Laderoutinen genutzt werden, da sie individuelle Ergänzungen und Ausprägungen der Standardsoftware sein können. Anwender bzw. Anwenderinnen sollen das Autorensystem Lyssa durch ihre eigenen Softwarekomponenten erweitern können, ohne das gesamte Projekt neu übersetzen zu müssen. Balzert definiert den Begriff Rahmenwerk wie folgt:

> "Ein Rahmenwerk ist ein durch Software-Entwickler anpassbares oder erweiterbares System kooperierender Klassen, die einen wiederverwendbaren Entwurf für einen bestimmten Anwendungsbereich implementiert (vgl. [Balzert00] Seite 843)."

Jene Klassen, die entweder von anderen Klassen des Rahmenwerks abgeleitet wurden, oder die eine Implementierung einer abstrakten Klasse sind, können in das System integriert werden. Dieser Mechanismus stellt sicher, dass die erzeugten Klassen mit den bestehenden Klassen des Rahmenwerks kommunizieren können, wobei jedoch die Logik der Ablaufsteuerung durch das Rahmenwerk verborgen bleibt. Die hinzugefügten Klassen empfangen dann Nachrichten nach dem Hollywood-Prinzip: "Don't call us, we'll call you" [Gamma95].

Im Gegensatz zum Framework sind Klassenbibliotheken Sammlungen von Klassen, die in ein durch den Anwender erzeugtes System integriert werden. Abbildung 6.11 veranschaulicht den Unterschied dieser gegensätzlichen Konzepte.

Rahmenwerke können durch ihre Architektur klassifiziert werden. Für ein **Black-Box**-Rahmenwerk werden spezielle Objekte zur Parametrisierung instantiiert. Diese können dann durch andere, vom Rahmenwerk zur Verfügung gestellte Objekte, konfiguriert werden. Bei diesem Architekturdesign werden keine speziellen Kenntnisse der internen Abläufe benötigt. Die Anpassungsmöglichkeiten der Anwendung sind bei dieser Architektur begrenzt, denn die Struktur ist durch das Rahmenwerk vordefiniert.

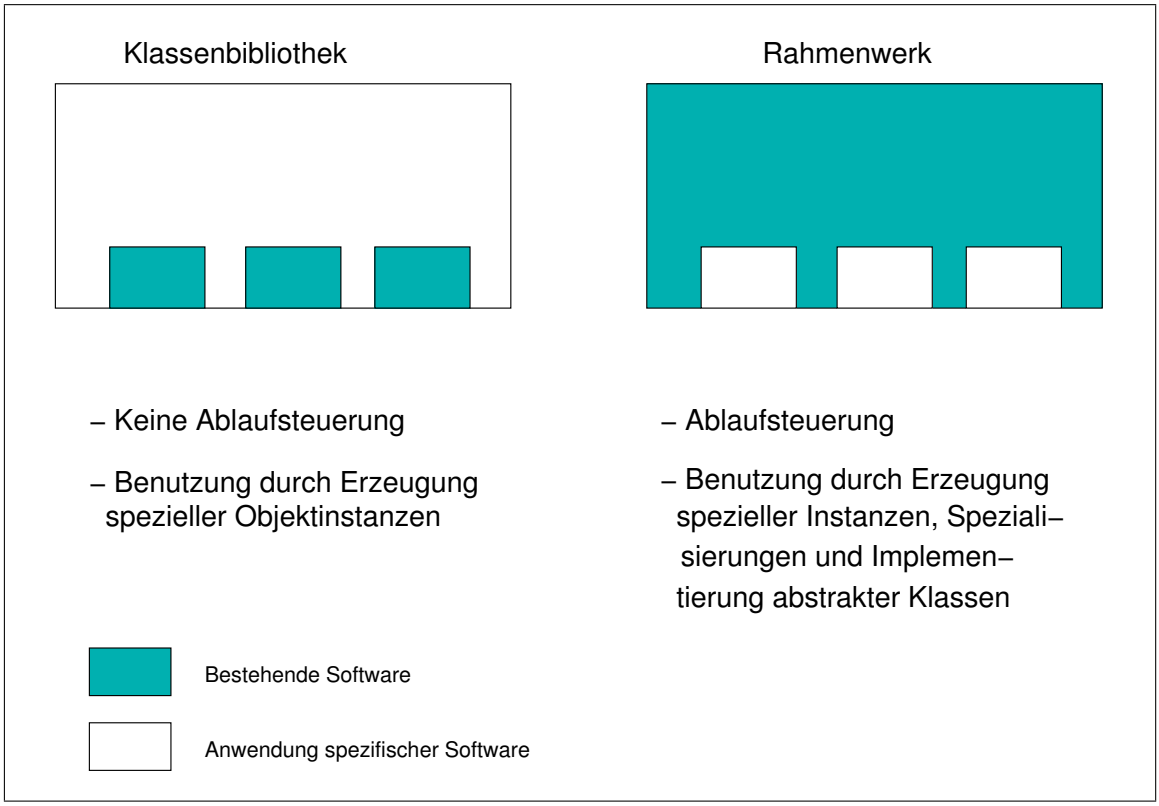


Abbildung 6.11: Framework vs. Klassenbibliothek

Die Verwendung eines **White-Box**-Rahmenwerks erfordert hingegen die Kenntnis der internen Abläufe. Der Entwickler bzw. die Entwicklerin muss die genauen Zusammenhänge kennen, da der entwickelte Code mit dem Rahmenwerk zu einer Anwendung instantiiert wird. Die Erweiterung der Applikation findet durch Unterklassenbildung, bzw. Implementierung vorgefertigter Schnittstellen statt. White-Box-Rahmenwerke bieten durch ihren erweiterbaren Ansatz eine große Flexibilität in der Anwendungsentwicklung (vgl. [Züllighoven98]).

Eine andere Möglichkeit bietet eine Kombination beider Architekturprinzipien. So ist es vorstellbar, dass White-Box-Rahmenwerke vordefinierte Standardlösungen enthalten, die ohne Kenntnis der internen Struktur verwendet werden. Andererseits lassen sich auch Black-Box Rahmenwerke derart erweitern, dass sie durch White-Box-Mechanismen bearbeitet werden können.

Das Kurs-Framework

Bei der Entwicklung großer Softwaresysteme treten in der Konzeptionsphase wiederholt Fragen nach der *Erweiterbarkeit* und der *Wiederverwendbarkeit* auf, die bei dem Entwurf berücksichtigt werden sollten. Gerade für die Implementierung einer Autorenumgebung steht zu Beginn der Entwicklung noch nicht fest, welche Formate eingelesen werden müssen und in welcher Form Kurse präsentiert werden sollen. Um an dieser Stelle eine

gewisse Unabhängigkeit zu schaffen, wurde das in Abschnitt 6.1.7 vorgestellte Framework-Konzept umgesetzt. Auf der einen Seite verwendet die Autorenumgebung Lyssa das Kurs-Framework als Klassenbibliothek, damit das Framework zugleich von anderen Applikationen eingebunden werden kann und andererseits ist sie durch den Einsatz abstrakter Klassen derart erweiterbar, dass individuell implementierte Lösungen dynamisch nachgeladen werden können, ohne die gesamte Applikation neu übersetzen zu müssen.

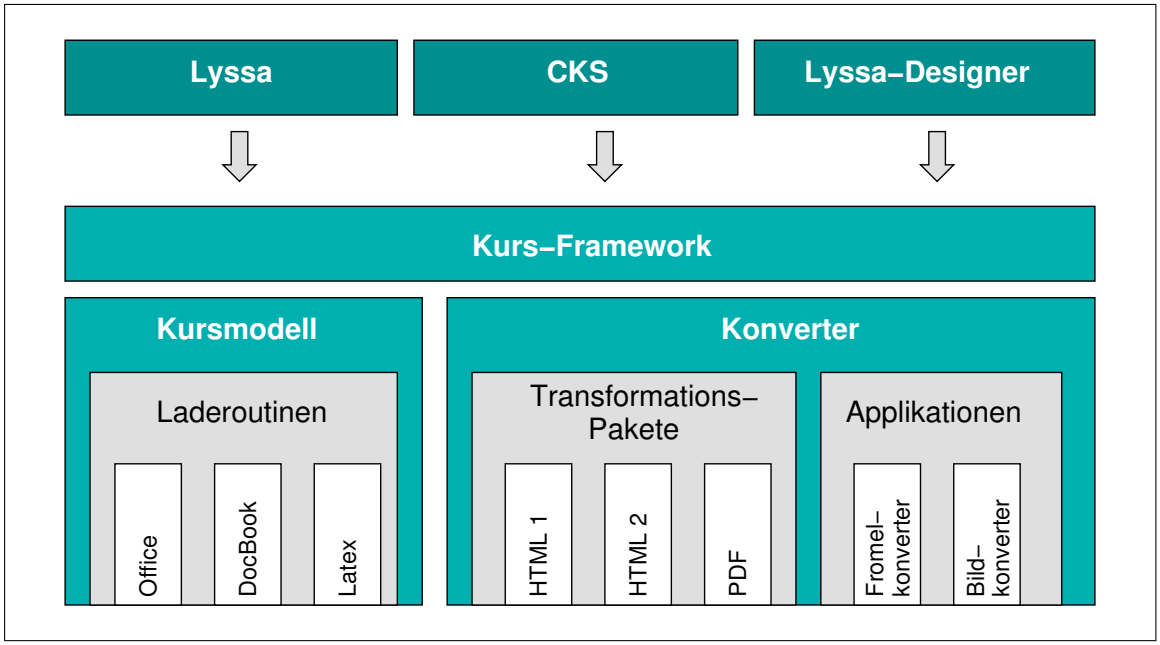


Abbildung 6.12: Das Framework

Abbildung 6.12 zeigt den modularisierten Aufbau des erweiterbaren Kurs-Frameworks. Das Framework besteht aus den beiden Komplexen Kursmodellschnittstelle und Konvertierungsschnittstelle. Die hellen Komponenten zeigen die austauschbaren bzw. erweiterbaren Module, die sich unabhängig von dem Framework entwickeln lassen. Das Kursmodell kann während seiner Initialisierung die vorhandenen Laderoutinen einlesen und sie den übergeordneten Applikationen anbieten. Zudem verfügt der Konverter über eine variable Anzahl an Transformationspaketen und Applikationen, die den Funktionsumfang der nutzenden Applikation beliebig erweitern.

6.2 Das Layoutwerkzeug Lyssa-Designer

Bisher wurde lediglich die Erstellung und Bearbeitung von formatierbaren Kursen vorgestellt. Im Folgenden wird nun ein weiteres Werkzeug, der Lyssa-Designer, präsentiert. Er unterstützt die Format- und Layoutentwicklung und wurde speziell für die in Abschnitt 5.5.4 eingeführten Rollen *Formatentwickler/-in* und *Layoutentwickler/-in* konzipiert. Mit dem Designer lassen sich Transformationspakete (TP) über eine grafische Benutzerschnittstelle öffnen, bearbeiten und speichern. Das Anwendungsmodell sieht die zwei Modi *Konfiguration* und *Entwicklung* vor, auf die im Folgenden näher eingegangen wird.

Konfiguration: Dieser Modus erlaubt es ausschließlich Einstellungen an einem TP vorzunehmen. Jeder Transformator besteht aus einer Liste assoziierter Übersetzungsregeln, die detaillierte Informationen für die Transformation liefern. Ein Beispiel für eine solche Regel ist die Übersetzungsvorschrift für eine Tabelle. Zu dieser Tabelle kann genau ein *Property*-Fenster geöffnet werden, welches die einzustellenden Layout-Merkmale der Tabelle anzeigt.

Development: Bei dem zweiten Modus steht vor allem die Formatentwicklung im Vordergrund. Hierfür wird in erster Linie das zu erzeugende Datenformat definiert, indem Überstzungsregeln formuliert werden. Für die Regeln müssen während der Formatdefinition variable Parameter, wie Abstände, Farbwerte oder Linienstärken, festgelegt werden. Sie werden derart gekennzeichnet, dass sie vom Konfigurationsmodus als einstellbare Layouteigenschaften erkannt und zur Modifikation angezeigt werden. Des weiteren werden die für ein TP relevanten Transformatoren zugewiesen und Ressourcen, Prozessoren und Schemata bearbeitet.

Abbildung 6.13 zeigt einen Screenshot des Konfigurationsmodus. Auf der linken Seite werden die im Paket konfigurierten Transformatoren aufgeführt. Jeder Transformator besitzt eine Liste von Übersetzungsregeln, die sich bei Bedarf öffnen und schließen lassen. Wird eine der Regeln durch Selektion ausgewählt, erscheint auf der rechten Seite ein MDI-Fenster, das die Eigenschaften der ausgewählten Übersetzungsregeln anzeigt. Dieses Beispiel zeigt die konfigurierbare Kopfzeile einer HTML-Seite. Die Eigenschaften bestehen aus einem *Bezeichner* und einem zugehörigen Wert. Anhand des Typs wird entschieden, wie der Wert manipuliert werde muss. So ist z. B. die Einstellung eines Farbwertes an einen Auswahldialog für Farben gekoppelt und Zahlenwerte an eine Textbox, in der sie modifiziert werden können. Für die Titelzeile lassen sich zuerst zwei Farbwerte einstellen, gefolgt von dem Logo der Titelzeile, mit weiteren Bildattributen. Das vorliegende Beispiel zeigt die Titelzeile zur Darstellung eines HTML-Layouts der GET-Arbeitsgruppe.

Der zweite Screenshot (vgl. Abb. 6.14) zeigt hingegen den Entwicklungsmodus des Lyssa-Designers. Auf der linken Seite werden die Transformatoren in Form von Karteikartenreitern angezeigt, von denen genau einer zur Zeit ausgewählt werden kann. In diesem Beispiel ist der Transformator für die Navigation selektiert und wurde der Startdatei `navigation.xsl` zugeordnet. Mit dieser Zuordnung kann der Übersetzer bzw. die Übersetzerin entscheiden, welche Datei für die Transformation eingesetzt werden soll. Unter den Transformatoren befinden sich die Dateien, die für die Übersetzung benötigt werden. In diesem Beispiel werden ausschließlich `.xsl`-Dateien angezeigt, wobei für Java-Transformatoren `.class`-Dateien ausgewählt werden müssen. Zusätzlich ist ein *File System Explorer* vorhanden, der ähnlich wie der Windows-Explorer die Arbeit mit dem Dateisystem des Arbeitsrechners ermöglicht. Hierdurch lassen sich zusätzliche Dateien per *Drag and Drop* in den darüber liegenden Bereich ziehen und stehen so für die Transformation zur Verfügung.

Auf der rechten Seite wird der Entwicklungsbereich der geöffneten Navigation angezeigt. Der obere Abschnitt entspricht der bereits bekannten Darstellung der Konfigurationsansicht. Sie dient zur Verifizierung neuer Einstellungen, die im unteren Bereich durch-

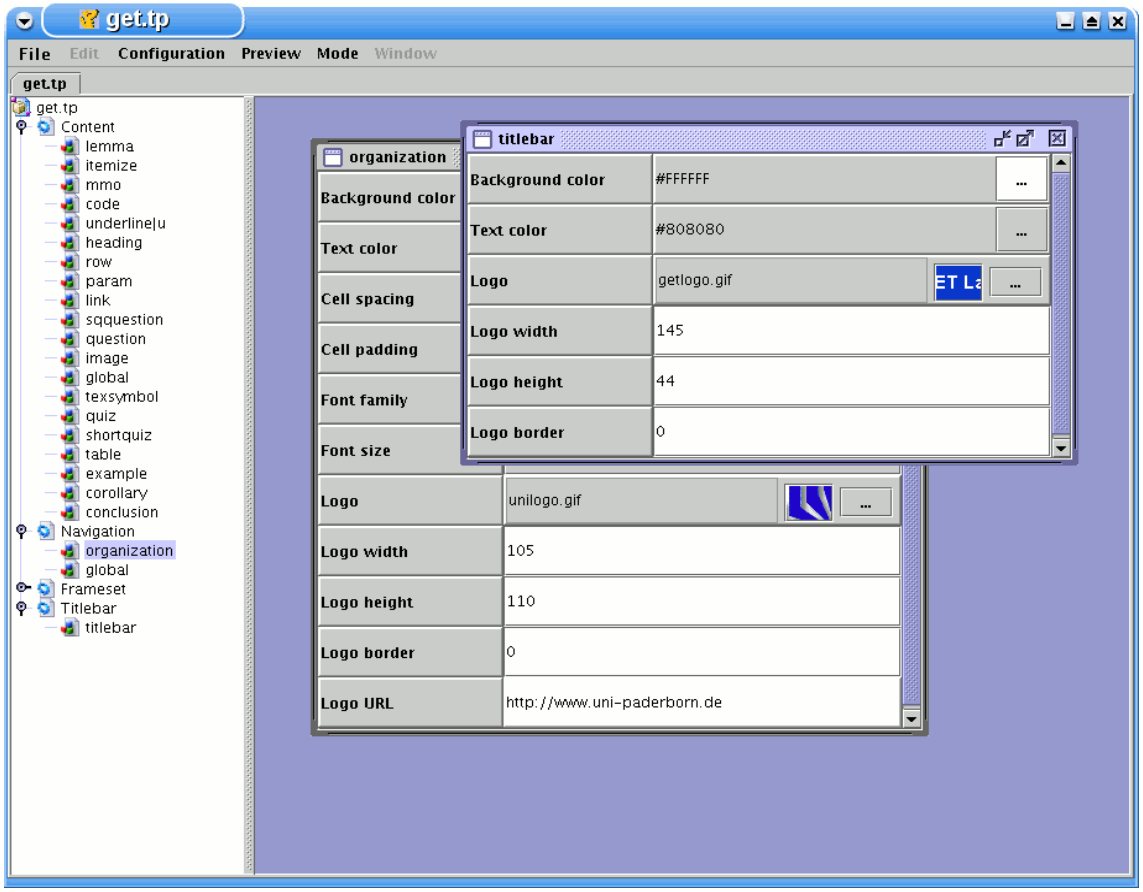


Abbildung 6.13: Lyssa Designer – Konfigurationsansicht

geführt werden. In diesem Bereich können die im Layout festgelegten Eigenschaften in die Übersetzungsregeln eingebaut werden. Dadurch kann die Rolle Formatentwickler/-in (vgl. Abb. 5.23), in der Gestaltungsphase die Layout-Eigenschaften variabel einbauen. Hierbei wird der Grad der Konfigurierbarkeit vom Layoutentwickler bzw. von der Layoutentwicklerin festgelegt. Wurde beispielsweise eine neue Eigenschaft in einer Übersetzungsregel verankert, kann durch Betätigung des Knopfes unterhalb des Entwicklungsbereichs, die Änderung direkt übernommen werden, wobei die Konfigurationsansicht um das gerade erstellte Element erweitert wird. Wurde die Eigenschaft jedoch syntaktisch falsch definiert, weist das Programm auf den Fehler hin. Dieses gewährleistet, dass alle Layouteigenschaften in der Konfigurationsansicht vorhanden sein müssen.

Des weiteren verfügt der Lyssa-Designer über eine Voransicht des zur Zeit in Bearbeitung stehenden Layouts. Dies ist hilfreich bei der Fehlersuche und ermöglicht die Ansicht der kurz zuvor durchgeführten Änderungen.

Die Einstellungen des Transformationspaketes werden in einer Manifest-Datei gespeichert. Die Parameter (vgl. Attribute 1 bis 1.5 in C.1), die nicht direkt den Übersetzungsprozess konfigurieren, sind Informationen über das Paket und werden über einen separaten Konfigurationsdialog angeboten. Abbildung 6.15 zeigt den Dialog zur Konfiguration der

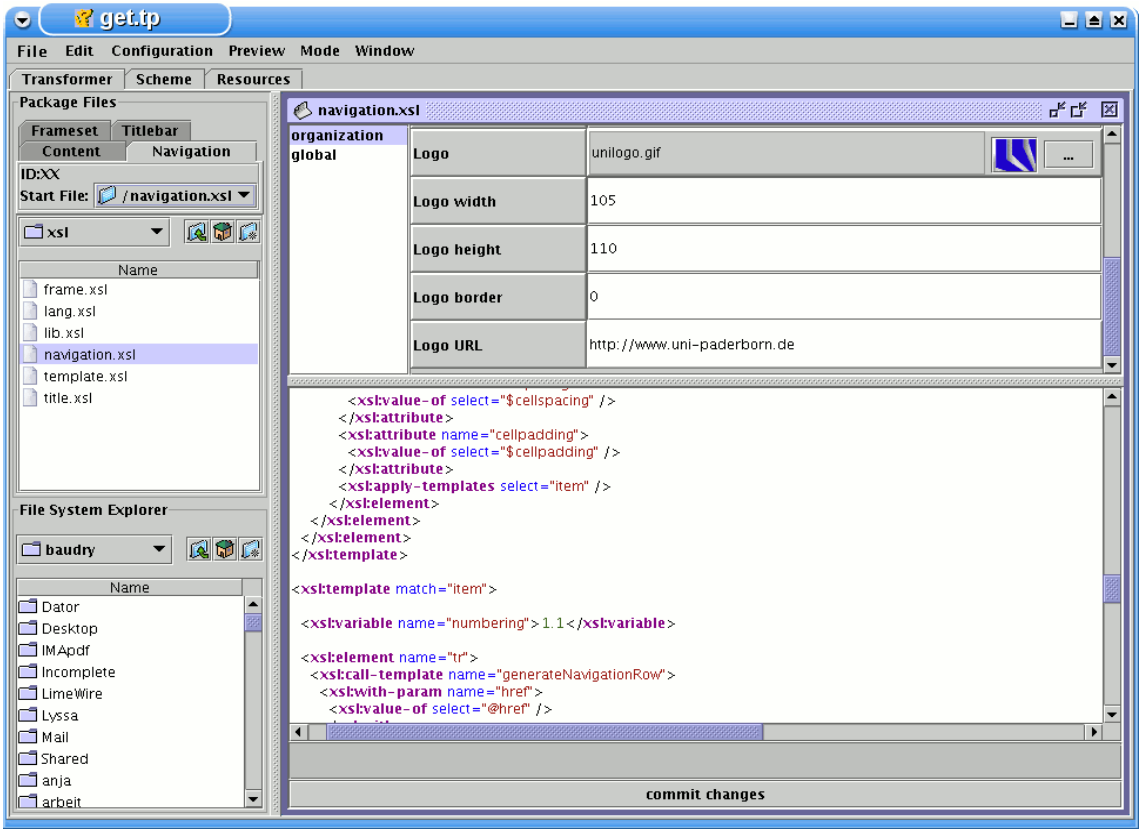


Abbildung 6.14: Lyssa Designer – Entwicklungsansicht

Grundeinstellungen eines TPs. Zunächst wird der Name des Pakets und dessen Kursbeschreibung eingegeben. Zudem kann ein beliebiges Logo festgelegt werden. Diese Informationen werden von dem Autorensystem Lyssa ausgewertet und für die Präsentation der TPs als Auswahlliste präsentiert. Der mittlere Block kennzeichnet das Ausgabeformat. Wenn das Ausgabeformat sich aus mehreren Dateien zusammensetzt, muss für jede zu erzeugende Datei ein Dateipräfix hinterlegt werden und die Startdatei ausgewählt werden. Diese Dateisammlung kann optional in einem komprimierten Archiv gespeichert werden. Dieses ist z. B. beim HTML-Format notwendig, da mehrere HTML-Dateien für eine Ausgabe benötigt werden. Soll das Paket jedoch ein PDF-Dokument erzeugen, wird nur eine Datei angelegt, die nicht als Archiv exportiert werden muss.

6.2.1 Implementierung

Für die Implementierung des Designers wurde bereits in der Entwurfsphase der Bezug von den Daten zu dem GUI hergestellt. Jeder Layouttyp wird durch eine eigene Klasse realisiert, die gemeinsam mit anderen Typen die Klasse `AbstractLayoutType` als Basis verwendet. Aufgrund der polymorphen Eigenschaften der abgeleiteten Klassen, können sie als Objekte der Klasse `AbstractLayoutType` einheitlich vom `LayoutHandler` mit Werten, wie z. B. der Kursbeschreibung oder das Daten-Objekt selbst, versehen werden. Daraus

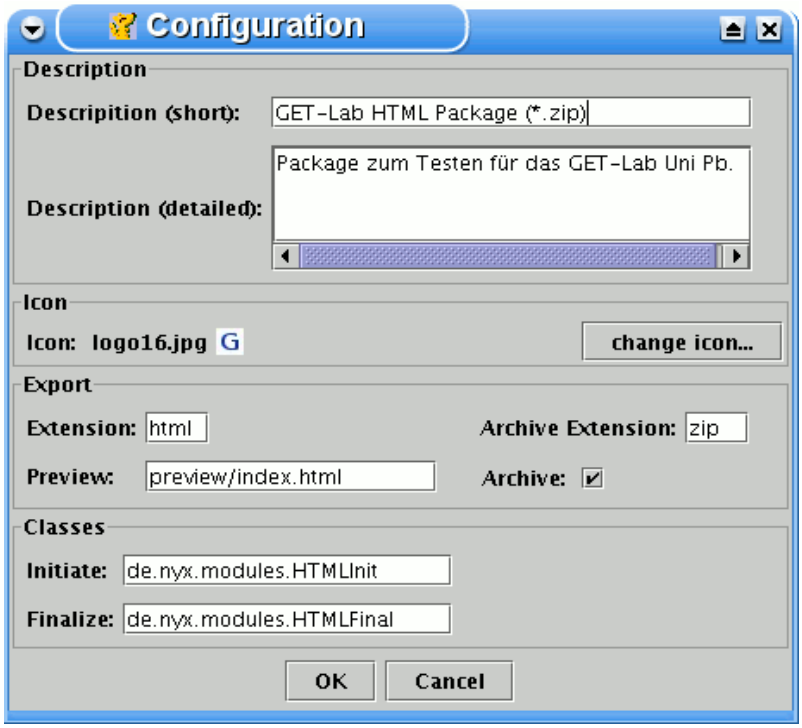


Abbildung 6.15: Lyssa Designer – Grundeinstellung

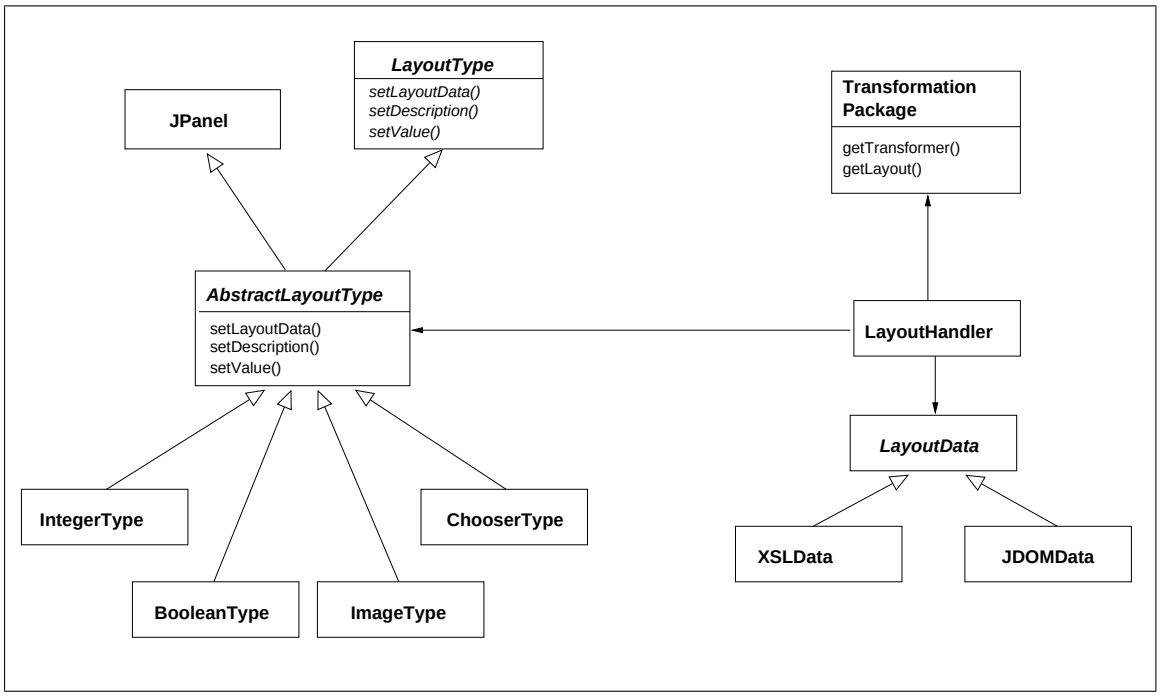


Abbildung 6.16: UML-Diagramm des Designers

resultiert, dass Layouttypen jederzeit ihr Datum verändern können. Des weiteren erbt

jeder Layouttyp von der Klasse `JPanel`[4], so dass jeder Komponententyp seinen eigenen Konfigurationsdialog anbieten kann. Wird dieser interaktiv manipuliert, kann der modifizierte Wert direkt dem Objekt der `LayoutData`-Klasse übergeben werden. Soll das TP abgespeichert werden, so müssen die Werte nicht erst aus dem GUI abgerufen werden, sondern können unter Verwendung der Datenstruktur gespeichert werden.

6.3 Construction Kit Server

Der Construction Kit Server (CKS) [Baudry04a] ist ein zentrales Repository für die Verwaltung formatierbarer Lernbausteine und Kurse und erweitert das Autorensystem um eine kooperative Entwicklungsumgebung. Der im Rahmen des mαth-kit Projekts realisierte CKS implementiert die im Modell (vgl. Kaptiel 5.6) geforderten Eigenschaften eines zentralen Repositories:

Authentifizierung: Benutzer und Benutzerinnen des Autorensystems können sich am CKS anmelden, um Zugriff auf einen persönlichen Arbeitsbereich zu erhalten. Außerdem ist die Authentifizierung notwendig, um den Zugriff auf Bausteine zu regeln.

Exklusive Schreibrechte: Das exklusive Schreibrecht regelt den parallelen Zugriff auf verteilte Dateisysteme.

Versionierung: Zwischenstände werden mit Versionsnummern versehen und ggf. zu einem späteren Zeitpunkt abgerufen.

Der CKS wurde als Webapplikation realisiert, weil die geforderten Eigenschaften mit Hilfe moderner Technologien, wie z. B. dem Jakarta-Tomcat und dem Slide Projekt, leicht umzusetzen sind. Die Authentifizierung wird durch den Authentifizierungsmechanismus des Webservers realisiert, so dass die Anmeldemaske als dynamische Website aufgebaut wird. Nach erfolgreicher Authentifizierung, wird ein Dateibaum, bestehend aus Kursen und Bausteinen, aufgebaut. Hinter der jeweiligen Datei, stehen die Attribute *Größe*, *Erstellungsdatum*, *Benutzer*, *Revisionsnummer* und *Bearbeitungszustand*. Wird ein Baustein bzw. ein Kurs selektiert, so werden weitere Daten über das Objekt angezeigt. Hierzu gehört die Revisionsnummer, die den Zugriff auf eine frühere Version erlaubt. Die zwei Schlösser auf der rechten Seite der Abbildung informieren über den aktuellen Zustand der Objekte und zeigen an, dass sie zur Zeit mit exklusiven Schreibrechten geöffnet wurden.

Abbildung 6.18 zeigt die Administrationsansicht des CKS. Mit diesem ebenfalls webbasierten Werkzeug, können neue Anwender bzw. Anwenderinnen angelegt oder gelöscht werden. Hierbei wird jedem Benutzernamen eine feste Gruppe zugeordnet, die, so wie in einem realen Dateisystem, gruppenweite Rechte festlegt.

[4] JPanel ist Bestandteil der Java-Swing-Bibliothek und wird für die grafische Gestaltung eines Fensterbereichs in einem GUI benötigt.

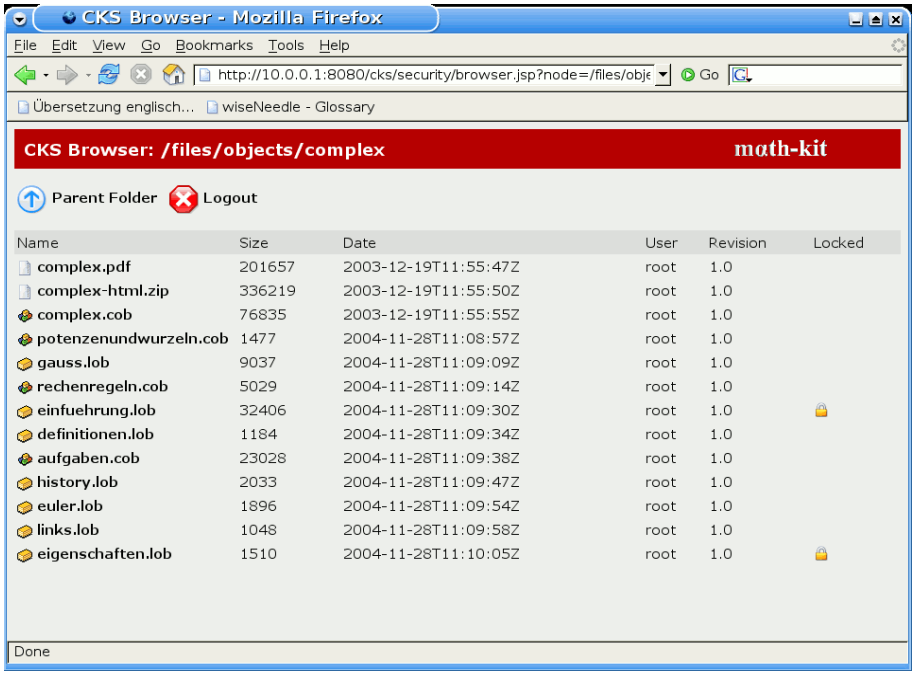


Abbildung 6.17: CKS Browser – Ermöglicht das Navigieren in verteilten Dateisystemen und die Überwachung nebenläufiger Aktionen

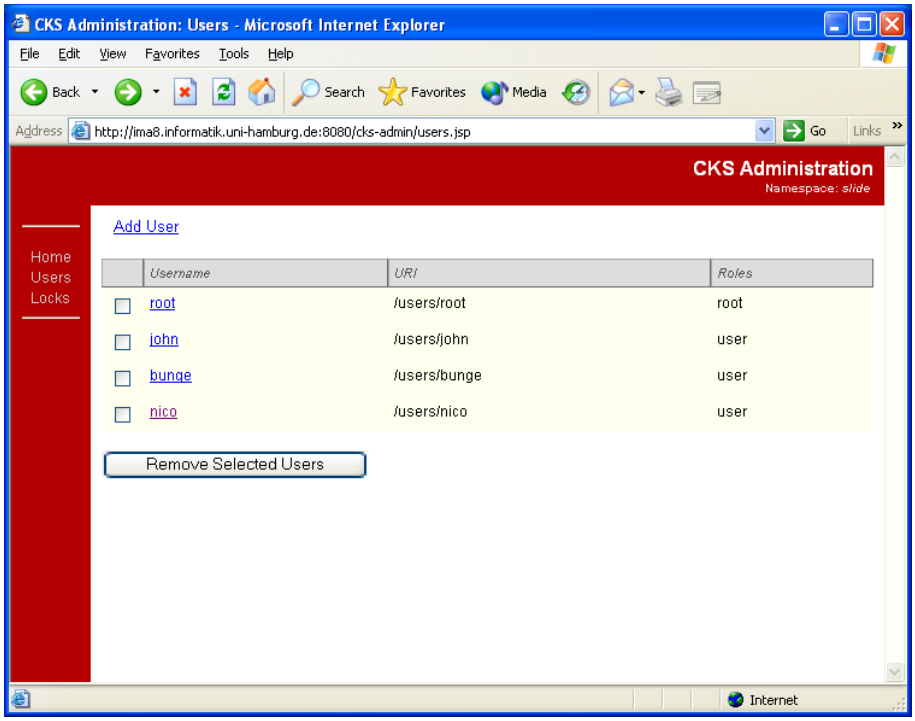


Abbildung 6.18: CKS Administration – Eine Eingabemaske, mit der Benutzer und Benutzerinnen angelegt und gelöscht werden.

6.3.1 Slide

Für die Implementierung des CKS wurde das Jakarta-Slide Projekt ausgewählt. Hierbei handelt es sich um ein technisches Rahmenwerk zur Erstellung von im Internet verteilten Dateiservern. Durch den Einsatz von Struts [Str05] und *Java Server Page* (JSP) [JSP05] werden Kommandos angeboten, die einen einfachen Zugriff auf Dateisysteme über Webseiten ermöglicht. Für die physikalische Dateiverwaltung werden neben Dateisystemen auch Datenbanken unterstützt. Je nach Bedarf können die über das WebDAV-Protokoll angebotenen Dateien entweder direkt in das Dateisystem des Servers geschrieben werden oder in einer Datenbank gespeichert werden. Eigene Applikationen nutzen die von Slide angebotenen Dienste, um Informationen über Dateien zu erhalten. So können z. B. Dateigröße, Autoren, Zugriffsrechte oder Revisionsnummern hierarchisch abgefragt und auf einer Webseite dargestellt werden.

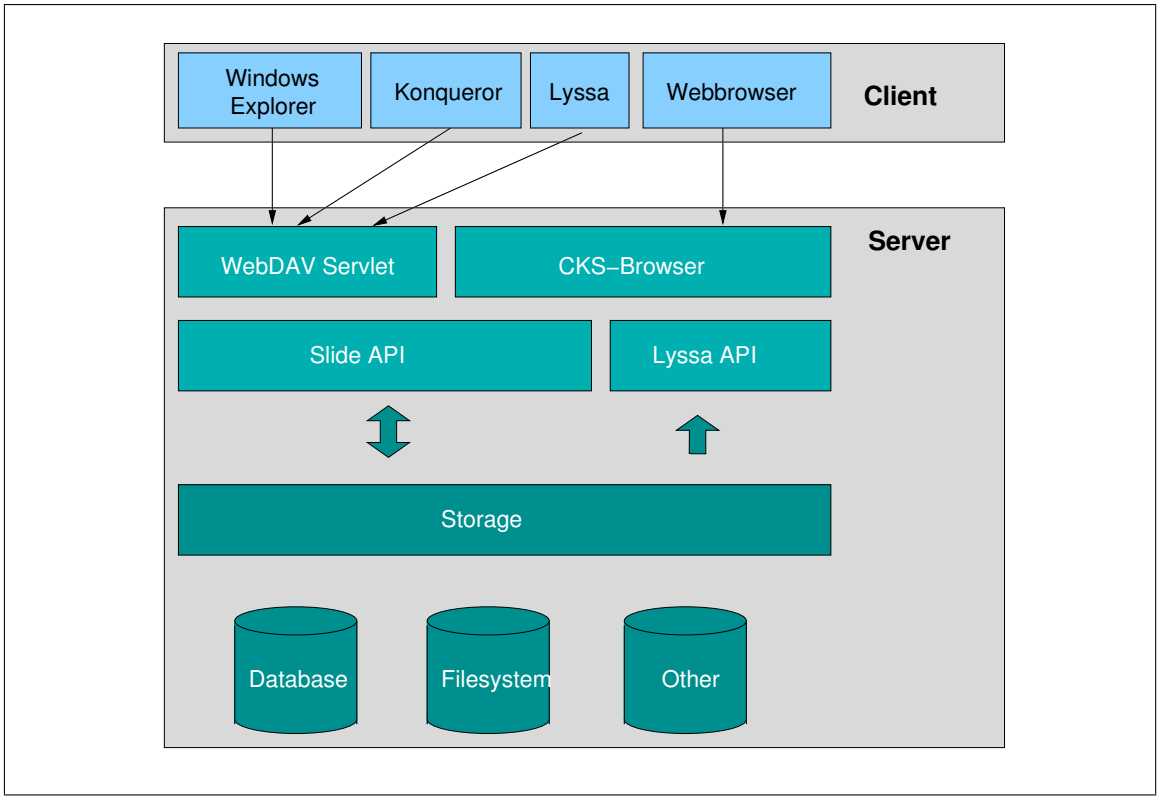


Abbildung 6.19: Schichtenmodell des Construction Kit Server

Abbildung 6.19 illustriert die Serverschnittstelle des CKS. Das *WebDAV*-Servlet erlaubt den Zugriff auf das Dateisystem über das WebDAV-Protokoll. Neben dem *Windows-Explorer* unterstützt auch der *KDE-Konqueror* den WebDAV-Zugriff auf Repositories. Zudem kann unter Verwendung eines einfachen HTML-Browsers auf den CKS zugegriffen werden. Der CKS-Browser setzt, genau wie der Lyssa-Designer, direkt auf dem Kurs-Framework auf und bietet daher die Generierung einer Voransicht für ausgewählte Kurse an.

Kapitel 7

Beispiele

Nachdem das grundlegende Modell und dessen Implementierung vorgestellt wurden, sollen nun konkrete Ergebnisse in Form von Beispielen folgen. Der Kurs, der in den folgenden Abschnitten konstruiert wird, wurde im Rahmen des mαth-kit Projekts entwickelt, das sich als hervorragendes Umfeld für die Anwendbarkeit des Modells herausgestellt hat. Hierbei geht es vor allem um die Demonstration der kooperativen und modularen Kurskonstruktion und der gezielten Adaption individueller Präsentationsformen.

Zunächst wird die für die Grundstudiumsausbildung ausgewählte Grundlagenveranstaltung *Technische Informatik* analysiert und eine mögliche Umsetzung in einen E-Learning-Kurs diskutiert. Hierzu gehört neben den technischen Rahmenbedingungen auch didaktische Merkmale, die erheblich zu der Gestaltung des Kurses beitragen. Des weiteren werden die ausgewählten Lektionen *Minimierungsverfahren* und *Komplexe Zahlen* präsentiert und deren konkrete Umsetzung durch das Autorensystem erläutert. Damit der entwickelte Kurs für verschiedene Lernszenarien eingesetzt werden kann, wird abschließend die Regeldefinition für Präsentationsformate veranschaulicht.

7.1 Technische Informatik - Exemplarische Kursentwicklung

Für die Grundstudiumsausbildung im Fach *Technische Informatik* existiert ein mit Latex gesetztes Skript [Mertsching02], welches Studierenden der Informatik vorlesungsbegleitend zur Verfügung gestellt wird, um den in den Vorlesungen erarbeiteten Stoff zu einem späteren Zeitpunkt nachbereiten zu können. Ferner dient es zur Vorbereitung auf Klausuren und Übungsgruppen, die den Studierenden zusätzlich angeboten werden. Des weiteren werden Übungsaufgaben verteilt, zu denen entsprechende Musterlösungen existieren. Dieses Kursmaterial wird im Folgenden zur Illustration der Kursentwicklung verwendet.

Ergänzend, zu dem bereits vorhandenen Angebot, soll das Projekt mαth-kit die Ausbildung durch den Einsatz von Online-Angeboten derart verbessern, dass Studierende anhand multimedialer Lernmodule komplexe Inhalte interaktiv bearbeiten können und theoretische Sachverhalte durch geeignete Abstraktionen selbst *ausprobieren* und leichter verstehen können. Ebenso wichtig ist die Anreicherung des bestehenden Angebots

durch bewerte Exkurse, die, in Abhängigkeit individueller Voraussetzungen, durchzuarbeiten sind.

Die nächsten Abschnitte stellen, angefangen bei der Analyse, den Aufbau des Onlinekurses *Technische Informatik* vor und zeigen, wie Inhalte, die durch Projektpartner und Projektpartnerinnen entstanden sind, sich nahtlos in den Kurs einflechten lassen. Zur Veranschaulichung werden die beiden Kapitel *Minimierungsverfahren* und *Wechselstromschaltungen* vorgestellt.

7.1.1 Analyse

Zunächst muss der Lehrstoff derart strukturiert werden, dass er aus logisch zusammenhängenden Teilen besteht. Hierzu ist, neben dem eigentlich zu vermittelnden Stoff, auch ein *Übungsbereich* vorhanden, in dem Studierende Aufgaben lösen können, sowie ein *Explorationsbereich*, der, entsprechend dem konstruktivistischen Lernparadigma, die Möglichkeit zum Lernen durch *Erfahren* anbietet [Unger04].

7.1.2 Aufbau einer Lerneinheit

Nachdem die grobe Struktur des Lernmoduls festgelegt worden ist, muss der Lehrstoff entsprechend dem Baukastenprinzip partitioniert werden. Es stellt sich die Frage nach der Wahl einer sinnvollen Granularität. Das Baukastenprinzip überlässt, im Gegensatz zu anderen Modellen für Lernobjekte, Autoren bzw. Autorinnen freie Hand bei der Erstellung von Bausteinen. Lediglich deren Abgeschlossenheit muss bei der Entwicklung weiter berücksichtigt werden. Insbesondere ist darauf zu achten, dass Themenbereiche nicht zwingend andere voraussetzen. Trifft dieses jedoch zu, sollten solche Bausteine zu größeren Bausteinen zusammengesetzt werden. Dabei ist grundsätzlich zu bedenken, dass die Granularität von einem einfachen Applet bis hin zu einer vollständigen Lektion, inklusive enthaltener Übungen und Explorationsbereiche, reicht.

Abbildung 7.1 zeigt auf der rechten Seite das Kapitel *Wechselstromschaltungen* mit seinen Unterkapiteln und auf der linken Seite eine Liste eigenständiger Module. In diesem Beispiel ist der Abschnitt *Analyse von Wechselstromschaltungen* in die Unterabschnitte *Komplexe Größen* und *Zeigerdarstellungen von sinusförmigen Wechselgrößen* gegliedert. Der Kurs *Komplexe Zahlen* ist als Exkurs im Grundstudium zu betrachten, der sich aufgrund seiner thematischen Unabhängigkeit gut für die Umsetzung eines Bausteins eignet. Entsprechendes gilt für das Unterkapitel *Zeigerdarstellung von sinusförmigen Wechselgrößen*, da es das Thema *Komplexen Zahlen* zwar für das Verständnis voraussetzt, jedoch nicht als integralen Bestandteil mit einbezieht.

7.1.3 Kurskonstruktion

Im Folgenden werden aus dem Themenbereich *Technische Informatik* die beiden Abschnitte *Minimierungsverfahren* und *Wechselstromschaltungen* ausgewählt und deren Umsetzung vorgestellt. Anhand des Themas *Minimierungsverfahren* soll zunächst der modulare Aufbau einer Lektion illustriert werden, der durch den oben beschriebenen Kursaufbau

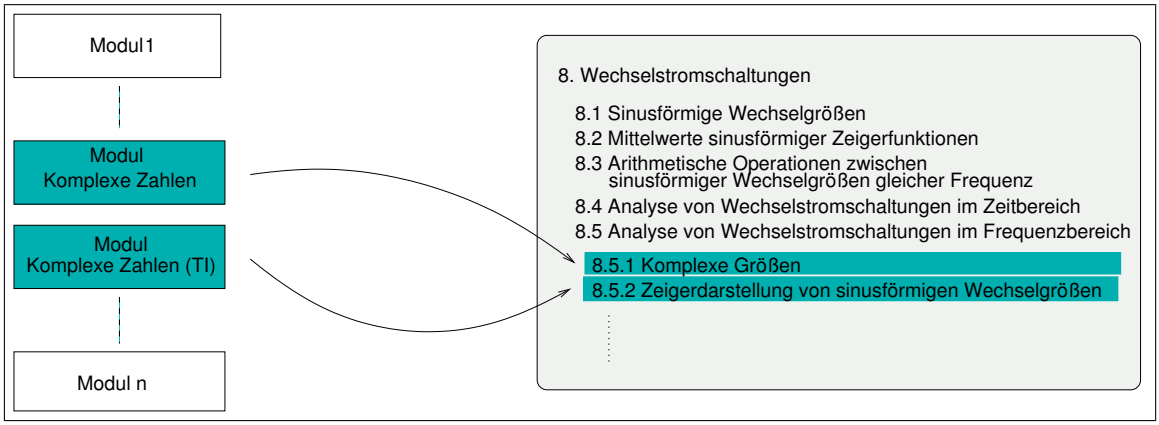


Abbildung 7.1: Basisformat für Kurse

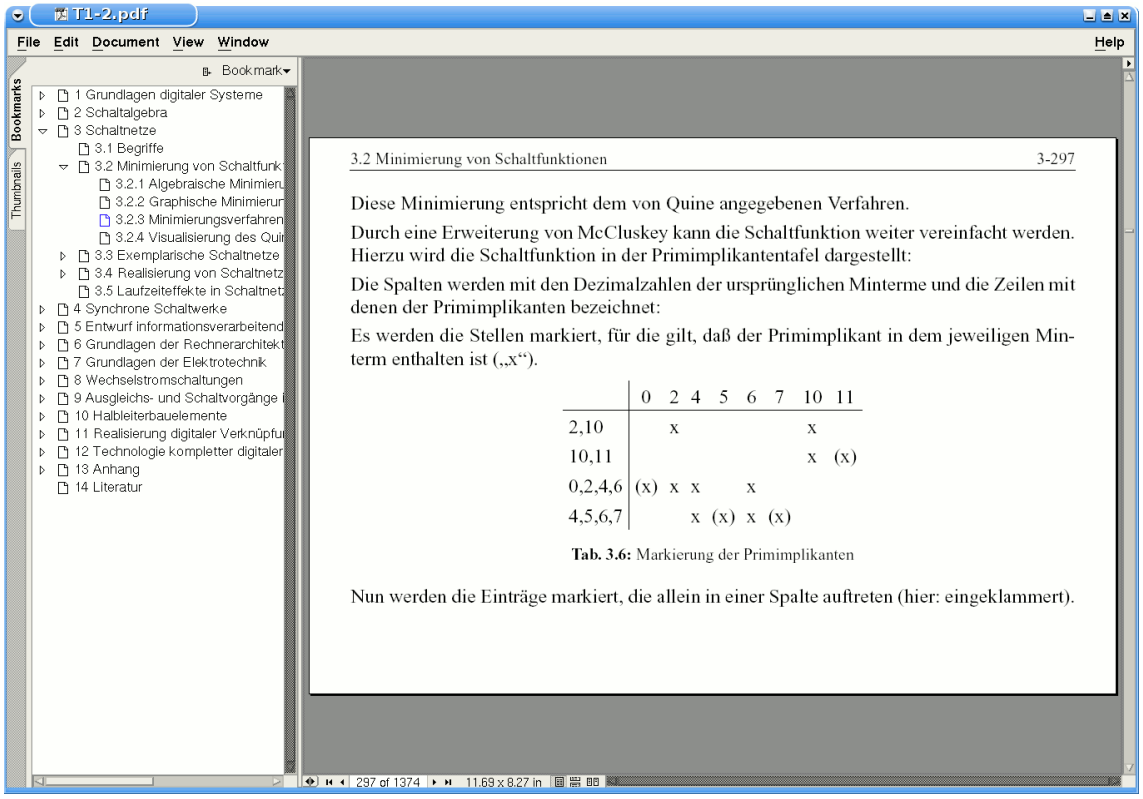
charakterisiert ist. Das zweite Thema verdeutlicht dagegen das Zusammenwirken mehrerer Autoren und zeigt, wie, unter Verwendung des in dieser Arbeit vorgestellten Modells, eine kooperierende Erzeugung fachübergreifender Lehrinhalte erstmals ermöglicht wird.

Minimierungsverfahren

Das Thema *Minimierungsverfahren* ist integraler Bestandteil einer Grundlagenveranstaltung mit dem Titel *Minimierung von Schaltfunktionen*, die für die Synthese von Schaltwerken notwendig ist. In dieser Lektion werden die Verfahren nach Quine und McCluskey und die Minimierung mit KV-Diagrammen durch interaktive Lernbausteine konstruiert und vorgestellt. Zur Veranschaulichung zeigt Abbildung 7.2 eine ausgewählte Präsentationsfolie, die in der Präsenzveranstaltung eingesetzt wird. Auf der linken Seite ist das Inhaltsverzeichnis des gesamten Skripts zu sehen, das aus mehreren hundert Folien besteht. Um Bausteine erstellen zu können, muss eine sinnvolle Granularität gewählt werden. In Anlehnung an die in Kapitel 7.1.2 dargelegte Analyse, wird das Kapitel *Minimierungsverfahren nach Quine und McCluskey* herausgegriffen und als eigenständiger Baustein modelliert. Dieser Baustein soll als Einführung in die Thematik dienen und erfüllt das Kriterium der Abgeschlossenheit, da keine Verweise auf andere, für das Verständnis wichtige, Abschnitte vorhanden sind und dessen Größe, mit dem aus didaktischer Sicht zumutbaren Umfang, einhergeht. Abbildung 7.3 zeigt einen mit dem Autorenwerkzeug Lyssa zusammengesetzten Baustein, der aus mehreren Abbildungen besteht, die als `PNG`-Dateien eingebunden sind. Zusätzlich ist eine Content-Datei enthalten, die den Inhalt des Lernobjekts anhand des in Kapitel 6.1.2 vorgestellten XML-Bindings kodiert. Das folgende Listing zeigt einen Ausschnitt aus dem Content-Modell des Bausteins, wie er in der Quelle vorliegt.

Listing 7.1: Dieses einfache Beispiel stammt aus einem Baustein zur Erläuterung des Minimierungsverfahrens nach Quine-McCluskey und veranschaulicht ein XML-basiertes Lernobjekt

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```



3.2 Minimierung von Schaltfunktionen 3-297

Diese Minimierung entspricht dem von Quine angegebenen Verfahren.

Durch eine Erweiterung von McCluskey kann die Schaltfunktion weiter vereinfacht werden. Hierzu wird die Schaltfunktion in der Primimplikantentafel dargestellt:

Die Spalten werden mit den Dezimalzahlen der ursprünglichen Minterme und die Zeilen mit denen der Primimplikanten bezeichnet:

Es werden die Stellen markiert, für die gilt, daß der Primimplikant in dem jeweiligen Minterm enthalten ist („x“).

	0	2	4	5	6	7	10	11
2,10		x					x	
10,11							x	(x)
0,2,4,6	(x)	x	x		x			
4,5,6,7			x	(x)	x	(x)		

Tab. 3.6: Markierung der Primimplikanten

Nun werden die Einträge markiert, die allein in einer Spalte auftreten (hier: eingeklammert).

Abbildung 7.2: Kursmaterialien für die Präsenzveranstaltung *Technische Informatik* in der Grundstudiumsausbildung

```
<lob title="Minimierungsverfahren nach Quine und McCluskey" xml:lang="de">
  ...
  <p> Durch eine Erweiterung von McCluskey kann die Schaltfunktion
  weiter vereinfacht werden. Hierzu wird die Schaltfunktion in der
  Primimplikantentafel dargestellt:</p>

  <p> Die Spalten werden mit den Dezimalzahlen der
  ursprünglichen Minterme und die Zeilen mit denen der
  Primimplikanten bezeichnet:</p>

  <p> Es werden die Stellen markiert, für die gilt, daß der
  Primimplikant in dem jeweiligen Minterm enthalten ist ("x").</p>

  <image src="bild_323_5.png"/>

  <p>Nun werden die Einträge markiert, die allein in einer Spalte
  auftreten (hier: eingeklammert).</p>

  <p>Die Primimplikanten der so markierten Zeilen sind wesentlich
  (<h>Kernimplikanten</h>) und müssen in der Schaltfunktion
  auftreten, die restlichen Primimplikanten werden gestrichen.</p>
```

```
                                                                          23
  <p>Es ergibt sich daher die folgende Vereinfachung:</p>

  <formula inline="false" mode="latex">                                   26
        y = (b_2 \wedge \bar{b_3} \wedge b_4) \vee (\bar{b_1} \wedge
        \bar{b_4}) \vee(b_3 \wedge \bar{b_4})
  </formula>                                                              29

  <p>(Minimalform)</p>

                                                                          32
  <p>*</p>
</lob>
```

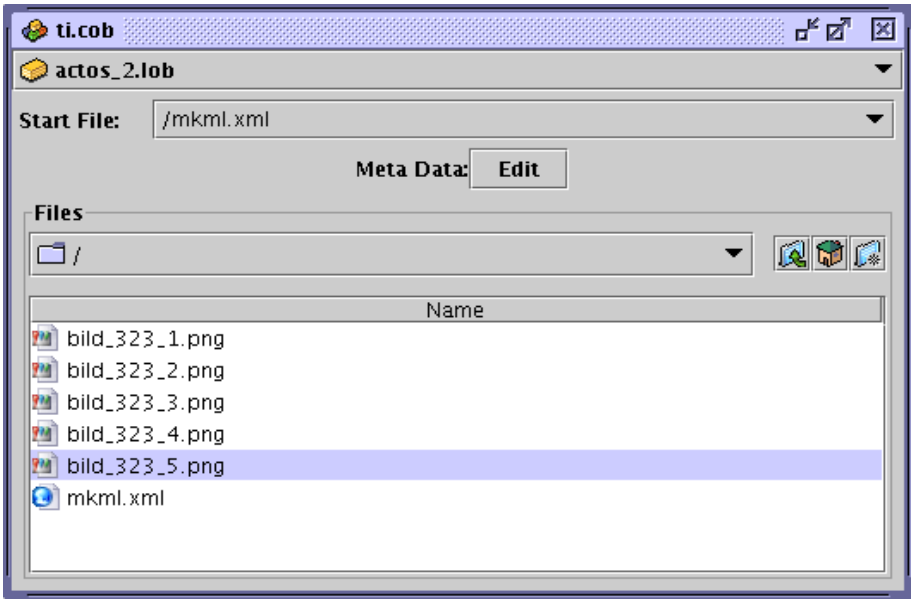


Abbildung 7.3: Der Baustein *Minimierungsverfahren* zusammengestellt aus einer Content-Datei und mehreren Abbildungen

Dieses einfache Beispiel verdeutlicht die Kombination grundlegender Elemente zum Setzen von Texten, Grafiken und Formeln. Nachdem der Baustein fertig gestellt ist, kann er in beliebige Grundlagenkurse, deren Thema die Einführungen in *Minimierungsverfahren* ist, kontextfrei eingebettet werden.

Eine andere Variante zur Minimierung von Schaltfunktionen ist die Methode mit *KV-Diagrammen*. Dieses ebenfalls eigenständige Gebiet wird in gleicher Weise als separater Baustein modelliert. Dieser Abschnitt lässt sich jedoch noch weiter unterteilen, so dass z. B. der Baustein *Minimierung bei Don't-care-Termen* als weiterführende Vertiefung angehängt werden kann. Abbildung 7.4 zeigt einen modular aus Bausteinen aufgebauten Kurs zum Thema *Grafische Minimierung*. Beide Minimierungsvarianten werden separat eingeführt und können zu jedem Zeitpunkt per Drag'n'Drop aus der Kursansicht heraus

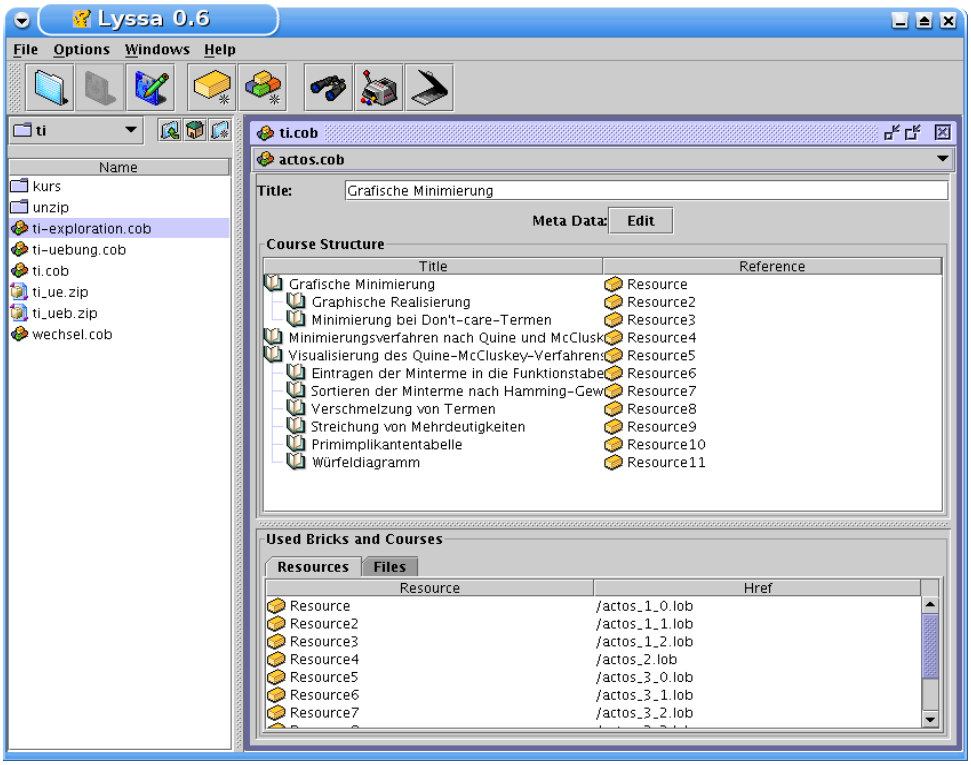


Abbildung 7.4: Das Autorenwerkzeug Lyssa mit dem geöffneten Kurs *Grafische Minimierung*

in einen anderen Kurs verschoben werden. Wurde jedoch von einem anderen Autor bzw. einer anderen Autorin eine weitere Variante implementiert, z. B. ein algebraisches Minimierungsverfahren, so kann dieses nahtlos in den bereits bestehenden Kurs mit aufgenommen werden und aufgrund der gemeinsamen Modellbasis, ein konsistenter Kurs entstehen. Die Umsetzung des soeben diskutierten Kurses ist in Abbildung 7.5 veranschaulicht. Für das Zielformat wurde eine auf der GET-Homepage basierende HTML-Variante entwickelt, die aus den drei Bereichen *Navigation*, *Titelzeile* und *Baustein* besteht. Die Kursnavigation befindet sich auf der linken Seite und ist aus der Kursstruktur von Lyssa generiert worden. Die Titelzeile enthält den Namen des Kurses und auf der rechten Seite ist der in Listing 7.2 vorgestellte Baustein zu sehen.

Der entwickelte Kurs wurde auf der Grundlage des bereits existierenden Skriptums erstellt und ist daher für Studierende rein informativ. Der Mehrwert, der gerade durch multimediale Komponenten erzielt wird, ist bisher nicht weiter genutzt worden. Deshalb soll nun der vorliegende Inhalt – entsprechend dem Resultat aus der in Abschnitt 7.1.1 vorgestellten Analyse – um eine *Explorations-* und *Übungsumgebung* erweitert werden.

Für die verschiedenen Verfahren zur grafischen Minimierung von Schaltfunktionen bietet sich die Entwicklung einer interaktiven Umgebung geradezu an, denn die Algorithmen basieren auf grafischen Verfahren, die sich durch ein Softwaremodell vollständig realisieren lassen. Übungen werden ebenso wie der bereits erzeugte Grundlagenkurs als Bausteine

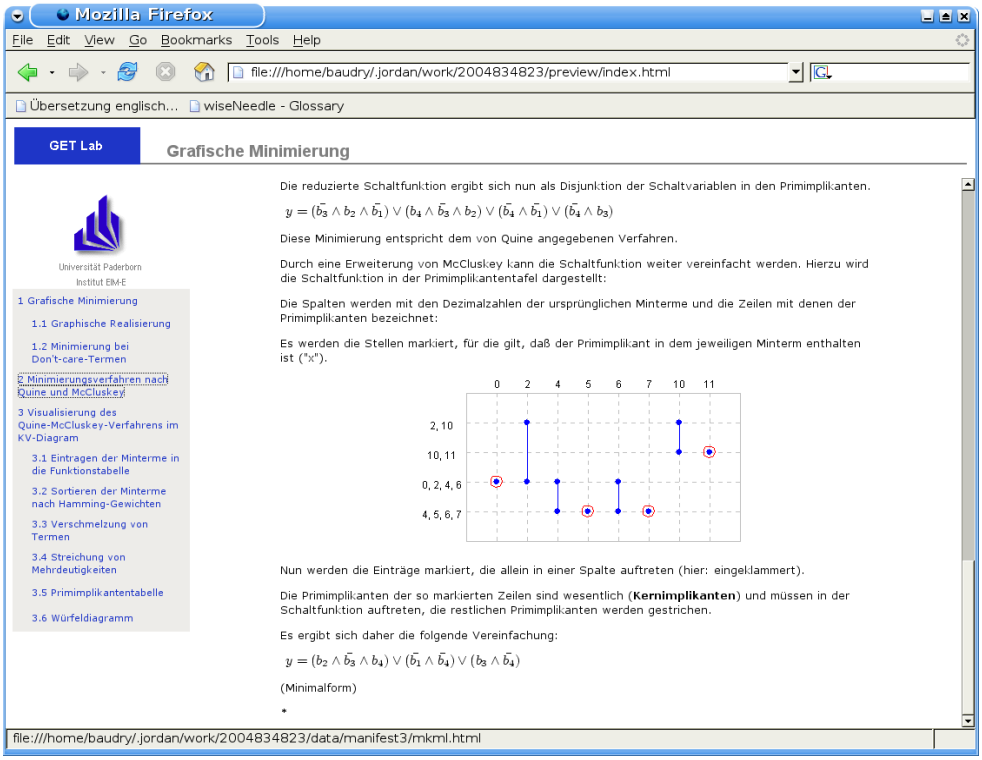


Abbildung 7.5: Der Kurs *Grafische Minimierung* übersetzt mit einem Transformationspaket für die Erstellung eines GET-Lab-Kurses.

erstellt. Das Ergebnis sind zwei Aufgabenkomplexe zu je einem Minimierungsverfahren, wobei jede Aufgabe einen eigenständigen Baustein darstellt. Die Aufgaben unterscheiden sich vor allem durch ihre ansteigende Komplexität sowie durch die Beschaltung unterschiedlich vieler Schaltvariablen. Das Ergebnis einer übersetzten Übungseinheit ist in Abbildung 7.6 zu sehen. Auf der linken Seite befindet sich ein interaktives KV-Diagramm, welches zur Lösung einer Aufgabe benutzt werden kann. Die Aufgabe, die hier gestellt wird, ist die optimale Minimierung der angegebenen Schaltfunktion zu finden. Durch geschickte Gruppierung von Nullen bzw. von Einsen können redundante Operationen erkannt und vermieden werden. Die linke Seite zeigt das korrespondierende Schaltnetz mit seinen UND- und ODER-Gattern. Hierbei können auch nicht optimale Lösungen gefunden werden. Unerfahrene Studierende können nach dem behavioristischen Lernparadigma durch 'ausprobieren' und der Analyse eigener Fehler die Grundprinzipien der grafischen Minimierung einprägsam erlernen.

Wechselstromschaltungen

Nachdem gezeigt worden ist, wie Kurse strukturiert werden und wie ein Kurs zur Minimierung von Schaltfunktionen realisiert werden kann, soll sich nun die Aufmerksamkeit auf die kooperative Konstruktion richten. Anhand des Kurses *Komplexe Zahlen* soll erstmals gezeigt werden, wie Lernressourcen von mehreren Autoren und Autorinnen erzeugt und

verwendet werden.

Eine elementare Grundvoraussetzung für die Entwicklung solcher Lernmaterialien ist die Einhaltung der in Abschnitt 7.1.1 beschriebenen Vorgehensweise während der Konzeptionsphase. Das bedeutet insbesondere, dass bereits bei der Kurskonzeption an die Wiederverwendung einzelner Subkomponenten gedacht wird. Was auf den ersten Blick einfach erscheint, ist auf den zweiten Blick jedoch keineswegs trivial. Die Schwierigkeit liegt vor allem darin, einen Baustein derart zu gestalten, dass er wiederverwendet werden kann. Dies geht nur dann, wenn der Baustein seiteneffektfrei entworfen wird. Bei der Entwicklung neuer Kurse muss der gesamte Kursaufbau strukturiert werden und nicht einfach ein großer Baustein erstellt werden. Welche Auswirkungen dies haben kann, wird im folgenden Kapitel demonstriert.

Nachdem das Paradigma modularer Bausteine diskutiert wurde, soll anhand des Kurses *Wechselstromschaltungen* gezeigt werden, wie Kurse in verteilten Teams entwickelt werden können. Der Kursteil, der im Folgenden betrachtet wird, ist die Einführung in das mathematische Gebiet der *Komplexen Zahlen*. Naturgemäß stellt sich die Frage: “Warum ausgerechnet *Komplexe Zahlen*?”. Die Antwort ist einfach, denn sie bieten sich zur Demonstration für die Wiederverwendung von Lehrinhalte an. In vielen naturwissenschaftlichen Grundlagenfächern sind sie Voraussetzung für das weitere Verständnis. Allerdings ist der Fokus oft domainenspezifisch und Einführungen in das Themengebiet werden an einem fachverwandten Anwendungsbeispiel verdeutlicht. Wird jedoch nach Gemeinsamkeiten gesucht, so wird schnell ersichtlich, dass eine allgemeine Definition gilt und Rechenoperationen, wie die Addition oder Multiplikation, in jedem Fach identisch sind.

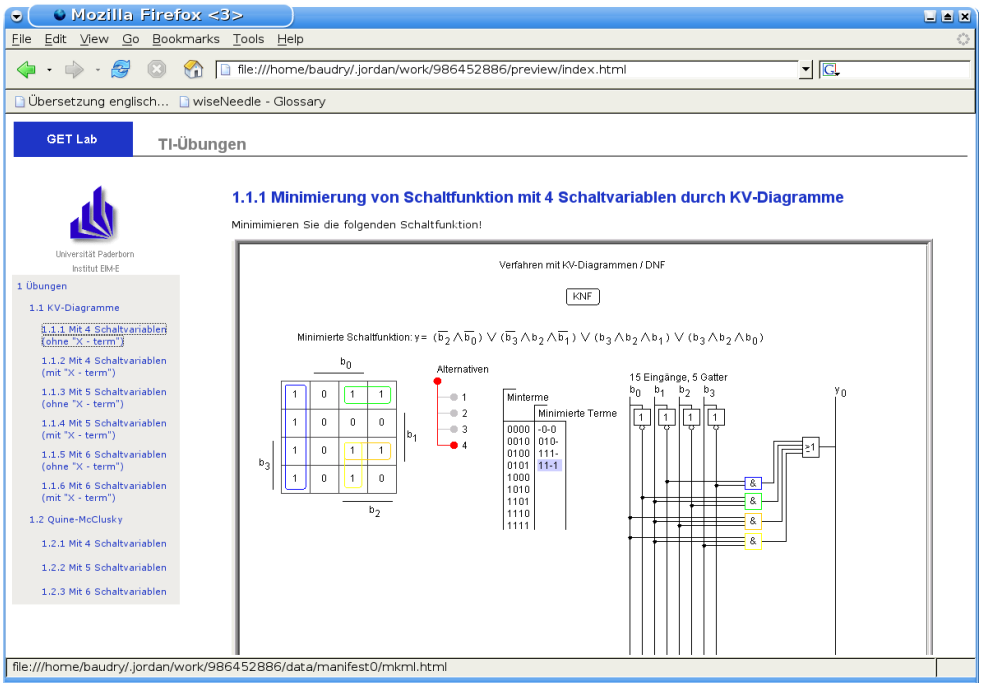


Abbildung 7.6: Ansicht einer interaktiven Übung zur Minimierung mit KV-Diagrammen

Das Forschungsprojekt mαth-kit hat sich speziell mit dieser Thematik auseinander gesetzt und die Fachrichtungen *Mathematik für Maschinenbauer*, *Maschinendynamik* und *Technische Informatik* multimedial aufbereitet. Es hat sich herausgestellt, dass die Behandlung der Komplexen Zahlen in jedem Fach unterschiedlich intensiv erfolgt, was dazu führt, dass in einem Fach mehr Unterthemen angesprochen werden, als bei einem anderen. Die Unterthemen lassen sich dennoch in markante Bausteine aufteilen, die sich zum Teil mit den anderen Fächern überdecken. Tabelle 7.1.3 zeigt eine Gegenüberstellung der Gliederungen aus den unterschiedlichen Fachgebieten.

Fach	Thema	Gliederung
Maschinendynamik	Harmonische Schwingungen	Einführung, Definition, Gauß, Darstellungsformen
Mathematik für Maschinenbauer I	Komplexe Zahlen	Einführung, Definition, Eigenschaften, Darstellungsformen, Beispiel Harmonische Schwingungen, Fundamentalsatz der Algebra
Technische Informatik	Analyse von Wechselstromspannungen im Frequenzbereich	Definition, Eigenschaften, Darstellungsformen

Tabelle 7.1: Integration der Komplexen Zahlen in unterschiedliche Fachrichtungen

Die Gliederung der Kapitel[1] zeigt die Themenüberschneidung in den Abschnitten *Definition*, *Eigenschaften*, *Darstellungsformen* und *Eigenschaften*. Die jeweiligen Ausprägungen der Abschnitte unterscheiden sich zwar voneinander, warum aber sollten drei verschiedene Varianten des gleichen Themas umgesetzt werden und mehrere Personen in einem Team die gleichen Abschnitte bearbeiten. An diesem Punkt angelangt, kann das Problem durch den Einsatz von Bausteinen gelöst werden. Jeder der oben genannten Abschnitte lässt sich als eigenständiger und unabhängiger Baustein realisieren. Bei der Konstruktion können die besten Darstellungen und Erläuterungen aus den Bausteinen herausgezogen werden, und ähnlich wie beim *Refactoring* [Fowler99] (vgl. Abschnitt 5.4.3) für objektorientierte Anwendungen, entweder in übergeordnete Bausteine verschoben werden oder in einen neuen Unterbaustein kopiert werden.

Im Projekt mαth-kit wurden die Gemeinsamkeiten herausgearbeitet und die Abschnitte in eigenständige Bausteine zerlegt, die sich nunmehr in andere Fachkurse integrieren lassen. Im Folgenden ist der Aufbau des allgemeinen Kurses zu sehen:

- Einführung
- Definition
- Eigenschaften
- Gaußsche Zahlenebene

[1] Auszüge aus den Skripten der verschiedenen Fächer sind auf der Projekt-Homepage zu finden [Mat05a]

- Eulersche Formel und Polarkoordinaten
- Rechenregeln
- Potenzen und Wurzeln
  - Beispiel 1
  - Beispiel 2
- Aufgaben
  - Grundrechenarten
  - Betragsbildung
  - Einheitswurzel
  - Darstellungsarten
- Exploration
- History
- Links

Dieser Kurs besteht aus einer allgemeinen Einführung, Grundlagen zu *Komplexen Zahlen*, Eigenschaften und Rechenregeln sowie aus Beispielen und Aufgaben. Abgerundet wird der Kurs durch einen Explorationsbereich zum Lernen durch 'ausprobieren'. Je nachdem wie ausführlich der Abschnitt zu *Komplexen Zahlen* für die jeweilige Fachdisziplin werden soll, können Bausteine, wie die Aufgaben, weggelassen werden oder spezialisierte Bausteine mit aufgenommen werden, wie z. B. die anwendungsorientierte Einführung.

Durch solche Maßnahmen werden die Kosten für die Erzeugung von Kursen erheblich gesenkt, weil sich der Aufwand für Neuentwicklungen beträchtlich reduziert, denn nicht jeder Kurs muss von Grund auf neu geplant und umgesetzt werden. Das schließt die Erstellung von Materialien für unterschiedliche Lernformen mit ein, denn sie können als einzelne Kurse im Netz, mit einem LMS (vgl. Kapitel 3.3) oder für den Präsenzunterricht angeboten werden[2]. Außerdem ist es möglich, Hardcopy-Varianten an Studierende zu verteilen.

In diesem Beispiel sind die an der Kurserzeugung beteiligten Partner projektbedingt über mehrere Standorte verteilt, was in der Regel zu Schwierigkeiten in der Absprache während der Entwicklung führt. Es handelt sich nämlich um autonome Arbeitsgruppen, die in ihren konkreten Implementierungsschritten keiner übergeordneten Instanz unterstehen. Damit jedoch eine gemeinsame Entwicklung möglich wird, wurde im mathkit-Projekt der CKS (vgl. Abschnitt 6.3) entwickelt, der kooperative Arbeitsprozesse sinnvoll unterstützt. Aufgrund seiner standardkompatiblen Schnittstellen, wurde die Integration in Lyssa umgesetzt, die es den kooperierenden Partnern und Partnerinnen ermöglicht, wie gewohnt

[2]Hierfür eigenen sich insbesondere die Bausteine zur Exploration, mit denen sich spezielle Verfahren interaktiv vorstellen lassen

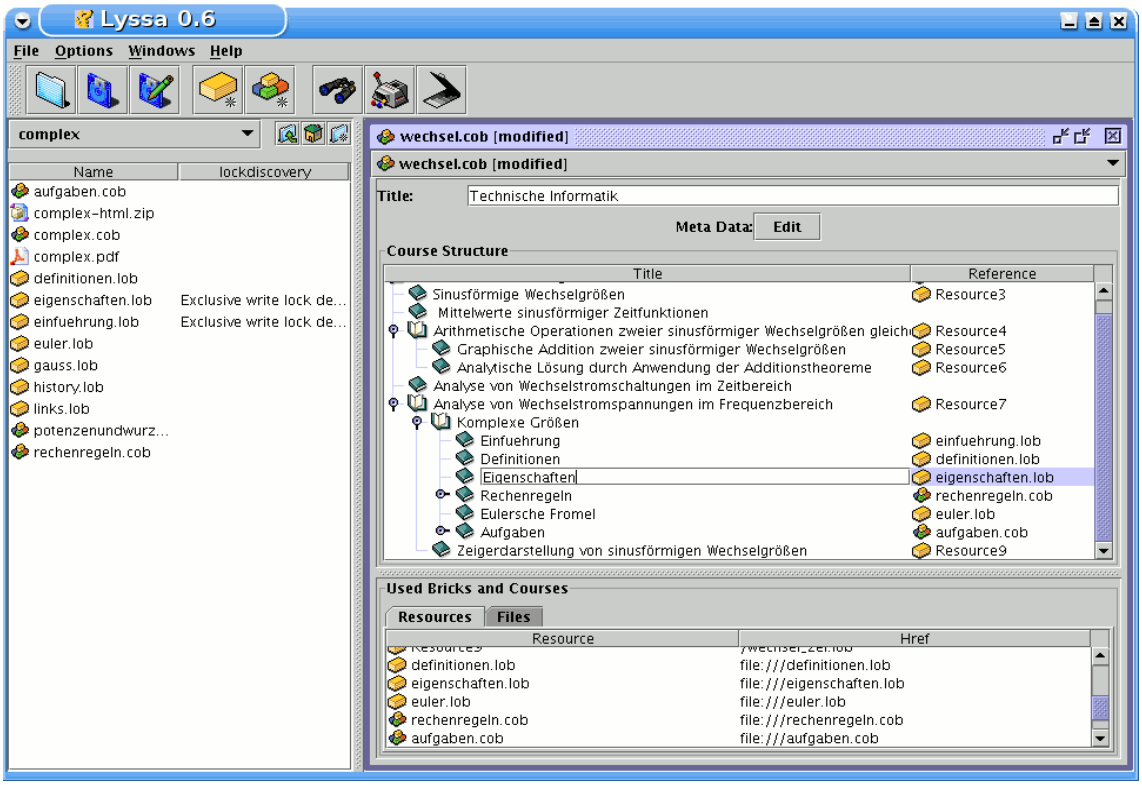


Abbildung 7.7: Der Kurs Wechselstromschaltungen mit Zugriff auf den CKS

Bausteine zu öffnen, anzulegen oder zu verschieben, ohne dass der Zugriff über das Netzwerk überhaupt bemerkt wird. Lediglich der exklusive Zugriff auf einen im Netz gespeicherten Baustein wird durch ein kleines Schloss symbolisiert, welches das Überschreiben des Bausteins durch andere Autoren verhindert.

Das zugrunde liegende Anwendungsmodell sieht zunächst die Speicherung eines neuen Kurses durch Team A vor. Hierzu werden die Subkurse *Definition* und *Rechenregeln* produziert und im eigenen Arbeitsbereich abgelegt. Daraufhin werden die Bausteine nach der Bearbeitung in den öffentlichen Bereich kopiert, der auch anderen Personen zugänglich ist. Diese sind nun in der Lage, diese Bausteine gezielt zu verwenden und ggf. den gesammelten Fundus zu ergänzen. Ebenso können Verbesserungen bereits bestehender Kurse und Bausteine eingepflegt werden, wobei der Zugriff auf einen früheren Datenstand jederzeit möglich ist.

Abbildung 7.7 zeigt das Autorenwerkzeug Lyssa mit dem geöffneten Kurs *Technische Informatik*. Auf der linken Seite wird ein Ausschnitt des Construction Kit Servers gezeigt, der aus Bausteinen und Kursen zum Thema *Komplexe Zahlen* besteht. Der gleiche Inhalt wurde bereits in Abbildung 6.17 gezeigt, jedoch mit dem Unterschied, dass es sich dort um den Zugang mit einem Web-Browser handelt. Bei der Entwicklung des Kurses *Technische Informatik* können dadurch Kosten eingespart werden, indem Bausteine von anderen Autoren wiederverwendet werden. Der Abschnitt *Wechselstromschaltungen* setzt bei Studierenden die Kenntnis über das Thema *Komplexe Zahlen* für die Analyse

im Frequenzbereich voraus. Die bereits erzeugten Basisbausteine *Einführung*, *Definition*, *Eigenschaften*, *Rechenregeln* und *Eulersche Formel* lassen sich von dem CKS in den neu erstellten Kursknoten *Komplexe Größen* verschieben. Zusätzlich wurde ein weiterer spezialisierter Baustein entwickelt, der fachbezogene Aufgaben enthält. Durch eine geschickte Aufteilung der *Komplexe Zahlen* kann der Kurs *Technische Informatik* an der bereits geleisteten Arbeit partizipieren.

7.2 Wiederverwendung bestehender Skriptteile

Die Entwicklung neuer E-Learning-Inhalte erfordert häufig die Wiederverwendung bereits bestehender Materialien. Dieses können, neben informativen Büchern, auch digitale Dokumente sein, deren Texte und Abbildungen in interaktiven Onlinekursen nicht fehlen dürfen. Dieser Abschnitt zeigt im Folgenden, wie das digitale Skript mit dem Titel *System Theorie* [Mertsching01] vom Prototypen eingelesen und mit Hilfe des Kursmodells in einen formatierbaren Kurs transformiert wird. Dazu muss in den Latex-Quellen nach Dokumentstrukturen gesucht werden und extrahiert werden, Assets und Metadaten erkannt werden und aus den Daten Bausteine erstellt werden.

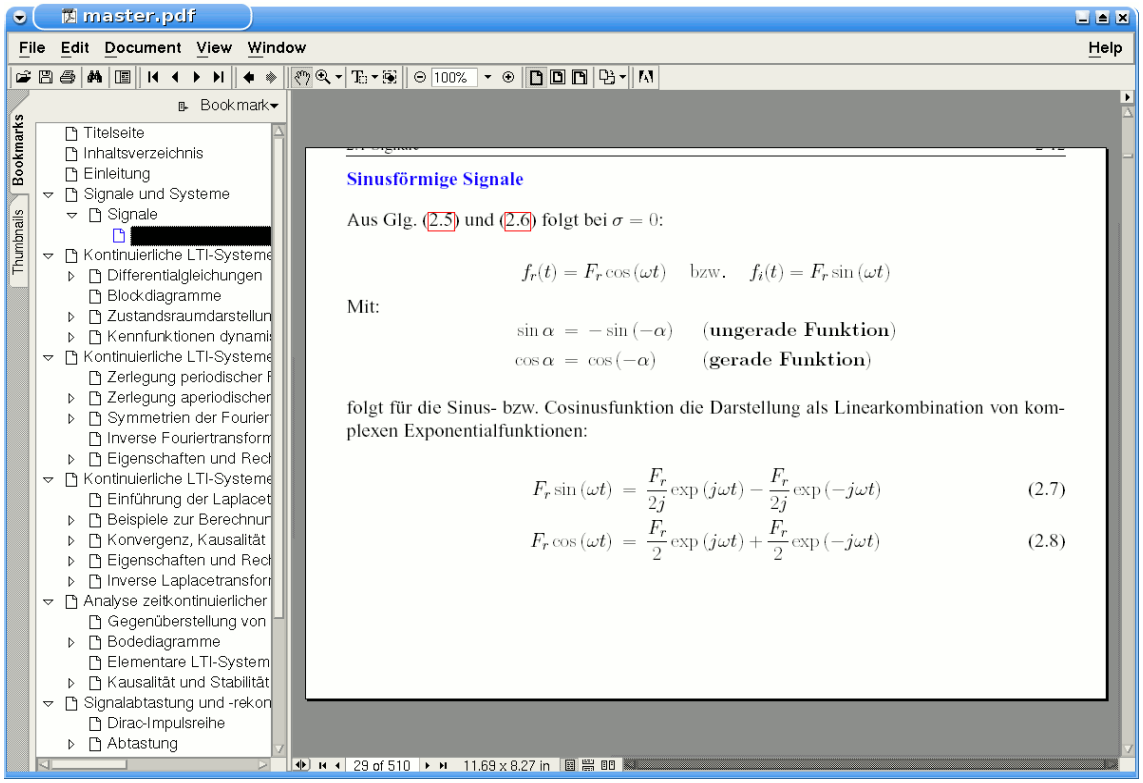


Abbildung 7.8: Präsentationsfolie *Sinusförmige Signale* aus dem Skript Systemtheorie

Zuerst wird das Skript entsprechend der Vorgehensregel aus Abschnitt 6.4 von Latex nach DocBook konvertiert, damit die Quellen über den DocBook-Loader eingelesen werden können. Dieses geschieht mit Hilfe des *tex4ht*-Konverters, der die Latex-Strukturen

untersucht und daraus eine neue DocBook-Struktur aufbaut. Tex4ht setzt bereits bei der Übersetzung der Latex-Quellen das Einbinden des *tex4ht*-Paketes voraus, wodurch Zusatzinformationen in die Ausgabe generiert werden. Aus diesem Grund müssen dem zu übersetzenden Latex-Dokument folgende Zeilen hinzugefügt werden:

```
\pdfoutput=0
\usepackage[xhtml,docbook]{tex4ht}
\let\pdfoutput\undefined
```

Nach dem Übersetzungsvorgang wird der Prozess `t4ht` angestoßen, um das endgültige DocBook-Dokument zu erhalten. Der tex4ht Konverter hat hierbei die zum Basisdokument gehörenden Dateien eingelesen und daraus ein einziges – in Anbetracht des STH-Skripts – großes XML-Dokument erzeugt. Verweise auf externe Dateien werden übernommen und in das DocBook-Dokument eingebunden, so dass zusätzliche Dateien nicht in den Verarbeitungsprozess mit einbezogen werden. Formatierungen, soweit diese durch Latex zugelassen werden, sind ebenfalls entfernt worden, da sie nicht durch das DocBook-Format unterstützt werden und außerdem vom Transformationspaket redefiniert werden.

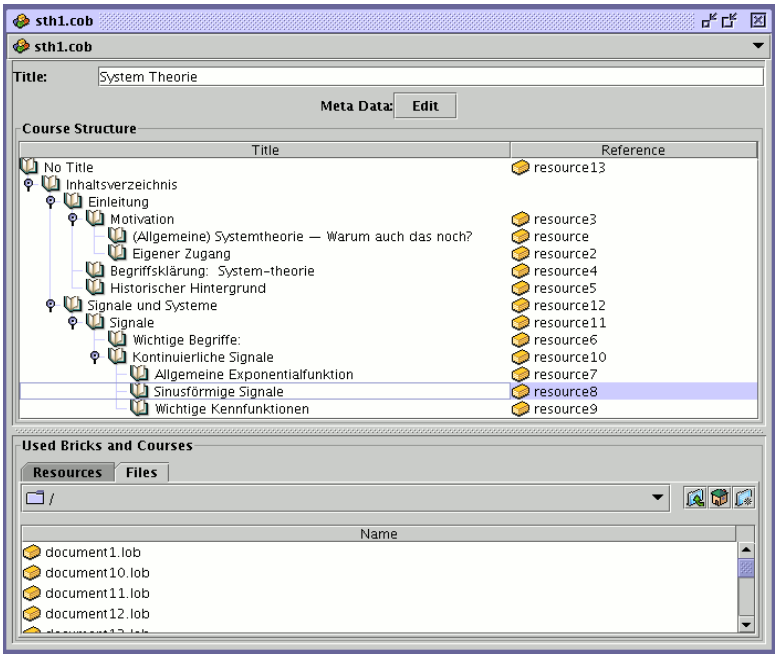


Abbildung 7.9: Die Kursansicht von Lyssa mit dem importierten Systemtheorie-Skript

Nachdem der Konvertierungsprozess abgeschlossen ist, kann die DocBook-Datei mit dem entsprechendem *Loader* eingelesen und auf das Kursmodell abgebildet werden. Um die Bausteine und Kurse zu erzeugen, wird die in Abschnitt 6.1.4 vorgestellte Methode des OO-Bindings verwendet. Hierbei werden nicht nur Bausteine erstellt, sondern auch vorhandene Ressourcen in die erstellten Bausteine kopiert. Abbildung 7.12 zeigt das mit Lyssa eingelesene Skript als modularen Kurs, der nun für den nächsten Bearbeitungsschritt bereit

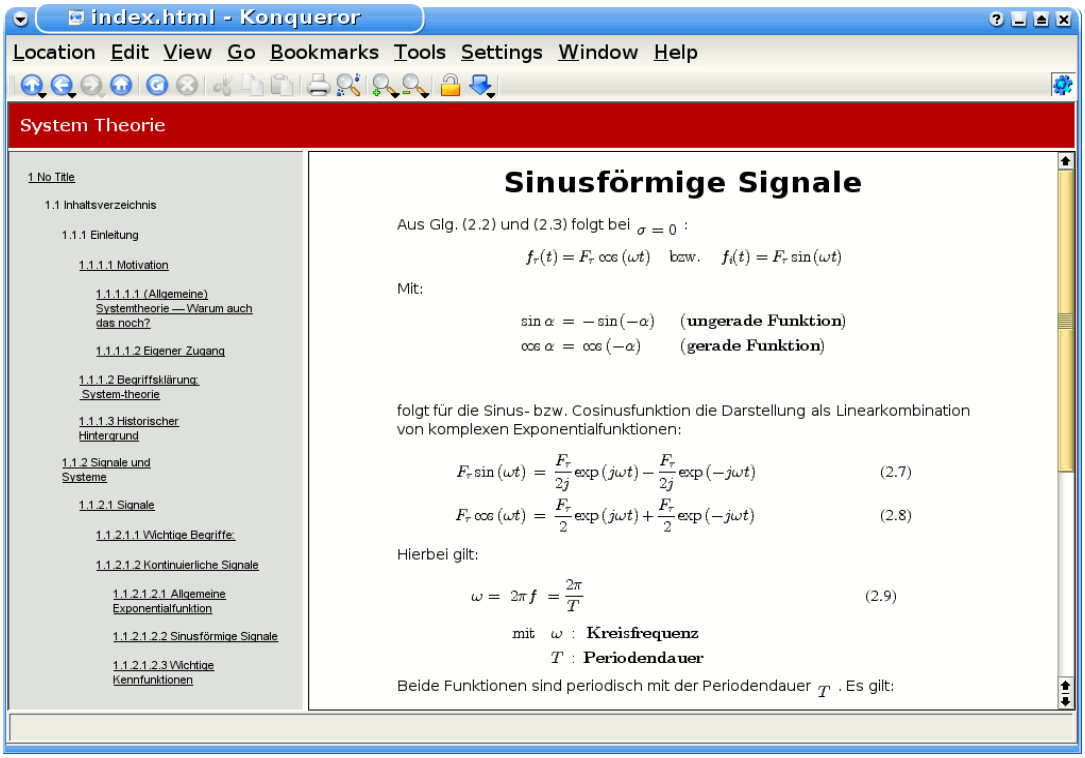


Abbildung 7.10: Sinusförmige Signale als Onlinekurs. Bei dieser Präsentation wurde das mαth-kit Layout verwendet

steht. Bausteine lassen sich jetzt in andere Kurse ziehen, durch interaktive Komponenten erweitern oder mit den Nachbearbeitungsmethoden aus Abschnitt 5.4.3 modularisieren.

Das Ergebnis einer beispielhaften Transformation zeigt Abbildung 7.10. Es handelt sich um einen in HTML gesetzten Kurs über das Fachgebiet Systemtheorie, der mit den Formatierungsregeln des mαth-kit Projekts erzeugt worden ist.

7.3 Exemplarische Entwicklung eines Formats zur Präsentation

Dieser Abschnitt stellt die Entwicklung eines Präsentationsformats vor und zeigt, wie Layout-Attribute festgelegt und später durch den Prototypen einzustellen sind. Das ausgewählte Format wird für die Erstellung studienbegleitender Unterlagen eingesetzt, die als DIN A4-Skript benötigt werden. Betrachtungsgegenstand ist erneut das TI-Skript aus Kapitel 7.1.

7.3.1 Festlegung des Layouts

Das Ziel, das mit der Layout-Vorgabe erreicht werden soll, ist eine strikte Trennung zwischen Einstellungen, die das Aussehen der Präsentation beeinflussen und denen, die für die

Generierung des Formats notwendig sind. Ferner sollen nur relevante Änderungsmerkmale, also jene, die zu einem späteren Zeitpunkt durch die Rolle Layoutentwickler/-in angepasst werden soll, separat definiert werden.

Listing 7.2: Definition eines Layouts

```
<pub:layout>
  <pub:master unit="cm">                                          2
    <pub:page width="21" background="#FFFFFF"/>
    <pub:body margintop="1">
    ...                                                           5
  </pub:master>
  <pub:styles>
    <pub:style id=" liststyle ">                                  8
      <pub:property id="prop01" typ="integer"
        desc="List item space"/>
      <pub:property id="prop02" typ="integer"                     11
        desc="Item indent" />
    </pub:style>
    ...                                                           14
  </pub:styles>
  <pub:map>
    <pub:resource ref="listitemspace" id="prop01" />              17
    <pub:resource ref="itemlabelindent" id="prop02" />
    ...
  </pub:map>                                                      20
```

Das Layout wird zunächst gemäß der Layoutspezifikation (vgl. Abschnitt 5.25) als XML-Dokument formuliert. Es enthält die für das Seitenformat notwendigen Einstellungen und wird vom Übersetzungsprozess zur Laufzeit herangezogen. In diesem Beispiel beträgt die Breite einer Seite 21 cm und ihre Hintergundfarbe wird auf weiß gesetzt. Neben diesen globalen Eigenschaften werden für die individuelle Gruppierung von Merkmalen *Styles* eingesetzt. Zur Verdeutlichung zeigt Listing 7.2 zwei Style-Attribute für die Abstandsformatierung von Aufzählungen.

7.3.2 Festlegung des Formats

Nachdem das Layout feststeht, muss das Format erzeugt werden. Für die Generierung eines druckbaren DIN A4 Formats bietet sich entweder der Einsatz von *Latex* oder *Formating Objects* an. Für dieses Beispiel fiele die Wahl auf die *Formating Objects*, weil es sich hierbei um ein modernes Verfahren zur Texterzeugung handelt.

Formatierbare Kurse bestehen in der Regel aus Bausteinen oder anderen verschachtelten Kursen (vgl. Abschnitt 5.3). Damit Kurse auf das Ausgabeformat abgebildet werden können, muss deren interne Struktur, bestehend aus Organisationen und verschachtelten Item-Elementen, rekursiv traversiert werden. Dieses Beispiel stellt einen Transformer

vor, der auf der Extended Stylesheet Language XSL basiert. Für den Kurs und den Baustein existieren separate Übersetzungsregeln, die im Folgenden genauer erörtert werden. Der grundlegende Aufbau des Zielformats wird mit dem Auftreten des `organization`-Elements in der ersten Zeile ausgeführt. Dies hat zur Folge, dass die eingeschlossenen Elemente mit dem Namensraum `fo:` in die Ausgabe geschrieben werden. Die Variablen für die Gestaltung des Dokuments werden durch das Layout-Dokument definiert. Der zweite Teil beschreibt den Aufbau einer Seite und verarbeitet rekursiv die auftretenden `item`-Elemente (vgl. Zeile 33).

Listing 7.3: XSL:FO-Dokument für einen PDF-Kurs

```
<xsl:template match="organization">                                    1
  <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master master-name="simple">                     4
        <xsl:attribute name="page-height">
          <xsl:value-of select="$pageheight" /></xsl:attribute>
        <xsl:attribute name="page-width">                              7
          <xsl:value-of select="$pagewidth" /></xsl:attribute>
        <xsl:attribute name="margin-top">
          <xsl:value-of select="$margintop" /></xsl:attribute>         10
        ...
        <fo:region-body margin-top="1.5cm" margin-bottom="1.5cm"
          column-count="2" column-gap="0.9cm">                         13
          <xsl:attribute name="background-color">
            <xsl:value-of select="$background" />
          </xsl:attribute>                                             16
        </fo:region-body>
        <fo:region-before extent="1.5cm"/>
        <fo:region-after extent="1.5cm"/>                              19
        <fo:region-start/>
        <fo:region-end/>
      </fo:simple-page-master>                                         22
    </fo:layout-master-set>

    <fo:page-sequence master-reference="simple">                       25
      <fo:static-content flow-name="xsl-region-before">
        <fo:block text-align="end" font-size="8pt"
          font-family="serif" color="red">                             28
          Seite <fo:page-number/> </fo:block>
      </fo:static-content>
      <fo:flow flow-name="xsl-region-body">                            31
        <xsl:apply-templates select="item" />
      </fo:flow>
    </fo:page-sequence>                                                34
  </fo:root>
</xsl:template>
```

Bei der Verarbeitung von `item`-Elementen trifft der Übersetzer auf Referenzen, die auf Dokumente innerhalb der Bausteine verweisen. Für deren Übersetzung wird ein weiterer Transformer benötigt, der die Regeln für die Dokumentstruktur definiert. Listing 7.4 zeigt die Regel für die Übersetzung einer Liste. Auch hier werden wieder die mit `fo:` gekennzeichneten Elemente in die Ausgabe geschrieben und die mit `xsl:` definierten Elemente ausgeführt. Zeile drei und vier zeigen die Wertzuweisungen zweier Variablen, die innerhalb der Übersetzungsregel für Aufzählungen ausgeführt werden. Als Vorlage für diese Einstellungen dient die Layout-Datei, in der die Bezeichner dieser beiden Variablen referenziert werden. Hierdurch wird während der Übersetzung die eindeutige Zuordnung hergestellt.

Listing 7.4: XSL:FO-Liste für einen PDF-Kurs

```
<xsl:template match="item">

 <xsl:variable  name="listitemspace">6</xsl:variable>                    3
 <xsl:variable  name="itemlabelindent">5</xsl:variable>

 <fo:list-item>                                                          6
     <xsl:attribute  name="space-after.optimum">
        <xsl:value-of select="concat($listitemspace,'pt')"/>
     </xsl:attribute>                                                    9
      <fo:list-item-label>
          <fo:block>
            <xsl:attribute  name=" start-indent">                        12
              <xsl:value-of select="concat($itemlabelindent,'pt')"/>
            </xsl:attribute>
              <xsl:if  test="name(..)='itemize'">&#x2022;</xsl:if>       15
              <xsl:if  test="not(name(..)='itemize')">
                   <xsl:number level="multiple" format="1"/>
              </xsl:if>                                                  18
          </fo:block>
      </fo:list-item-label>
      <fo:list-item-body start-indent="25pt">                            21
          <fo:block><xsl:apply-templates /></fo:block>
     </fo:list-item-body>
  </fo:list-item>                                                        24
</xsl:template>
```

Wird nun der Wert der Layout-Datei mit dem *Lyssa-Designer* modifiziert, entsteht eine neue Layout-Vorlage und das Ergebnis wird, entsprechend der vollzogenen Änderung, angepasst. Abbildung 7.11 zeigt einen Dialog zum Einstellen der Dokumenteigenschaften, in der die drei Variablen *pageheight*, *pagewidth* und *margintop* wiederzufinden sind. Das zweite Einstellungsfenster zeigt variable Regeln für die Generierung der Aufzählung.

Mit diesem Verfahren lassen sich jetzt, aus Sicht der Rolle Formatentwickler/-in, verschiedene Formate definieren, die sich unterschiedlich komplex konfigurieren lassen. Durch das Hinzufügen weiterer Variablen können z. B. Grafiken, Abstände und Farben frei skaliert werden.

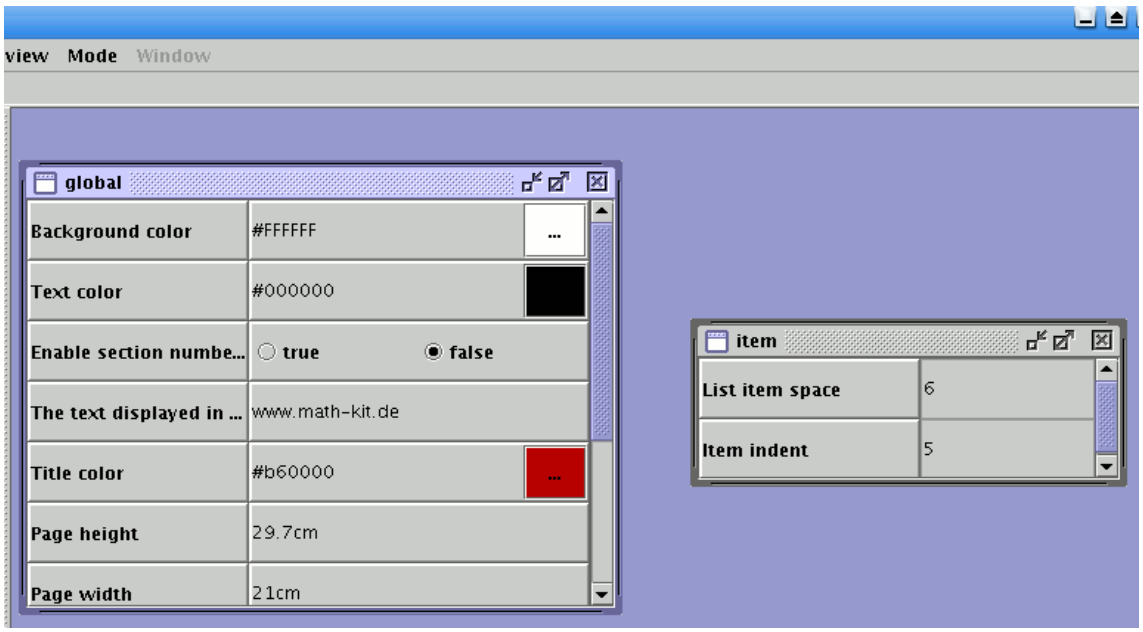


Abbildung 7.11: Screenshot des Lyssa-Designers

Abbildung 7.12 zeigt das Ergebnis als PDF-Dokument, dass aus dem Formating-Objects-Dokument generiert wurde. Es handelt sich hierbei um ein zweispaltiges Skript für die *Technische Informatik*.

7.4 Ausgewählte Bausteine

Das letze Beispiel zeigt Bausteine, die im Rahmen des mαth-kit Projekts entwickelt wurden und fachübergreifend in Kursen eingesetzt werden können. Auch der in diesem Kapitel entwickelte Kurs *Technische Informatik* (vgl. Abschnitt 7.1.3) kann mit diesen Bausteinen angereichert werden, um verschiedenen Lernparadigmen (vgl. Abschnitt 2.2.2) gerecht zu werden [Bauch04a, Bauch04b]. Die hier vorgestellten Bausteine folgen dem behaviouristischem oder konstruktivistischem Lernparadigma, indem sie entweder Aufgaben nach dem Multiple-Choice-Verfahren abfragen und Lernende durch direktes Feedback die Anzahl gelöster Aufgaben mitteilen oder Aufgaben zur individuellen Konstruktion anbieten, bei dem Lernende durch Ausprobieren Erfahrungen sammeln und dadurch Verfahren verstehen lernen.

Für die Explorationsumgebung *Technische Informatik* wurden zwei Applets im Kontext Kodierungsverfahren angefertigt. Das Applet ermöglicht Studierenden, das Verfahren der Huffman-Kodierung zu erlernen, indem sie zunächst für einen frei wählbaren Eingabetext die Wahrscheinlichkeitstabelle aufstellen müssen. Das Applet überprüft die Eingabe

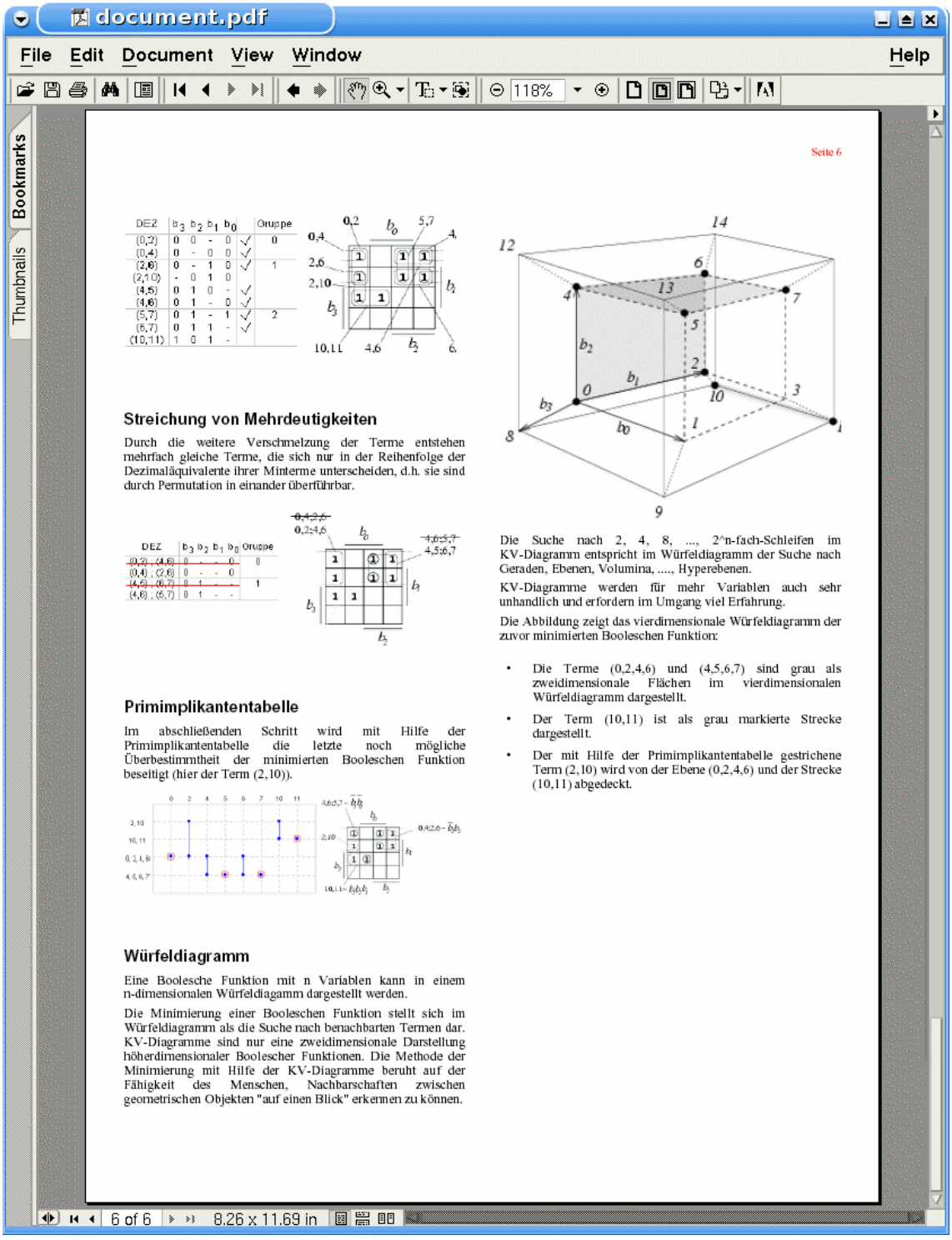


Abbildung 7.12: Der Kurs *Technische Informatik* als PDF-Dokument

der Wahrscheinlichkeitswerte und weist auf falsche Werte hin. Die Wahrscheinlichkeitstabelle ist in Abbildung 7.13 zu sehen. Wurde die Eingabe optimal durchgeführt, gelangt

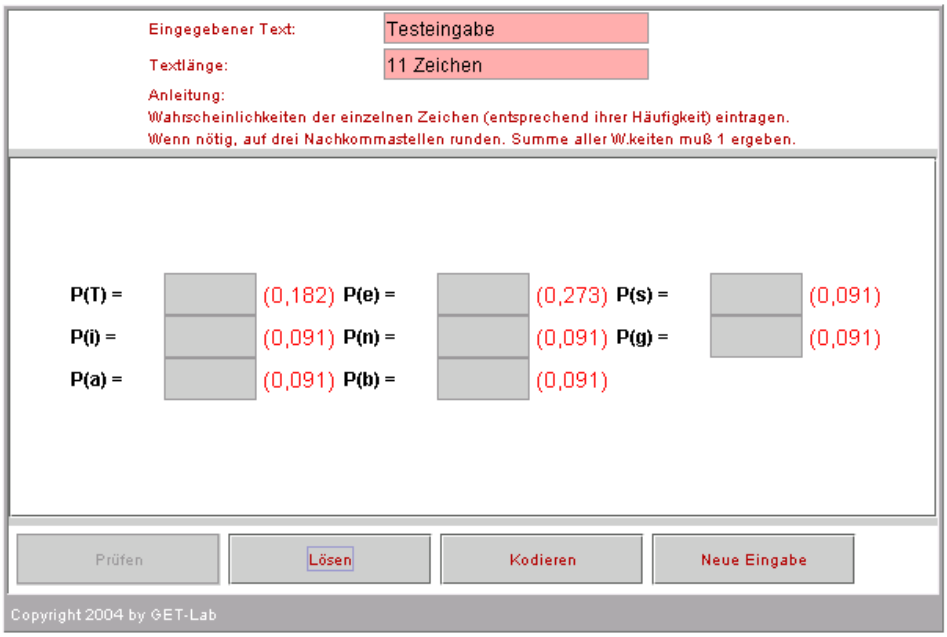


Abbildung 7.13: Wahrscheinlichkeitstabelle des Bausteins Kodierungsverfahren nach Huffman

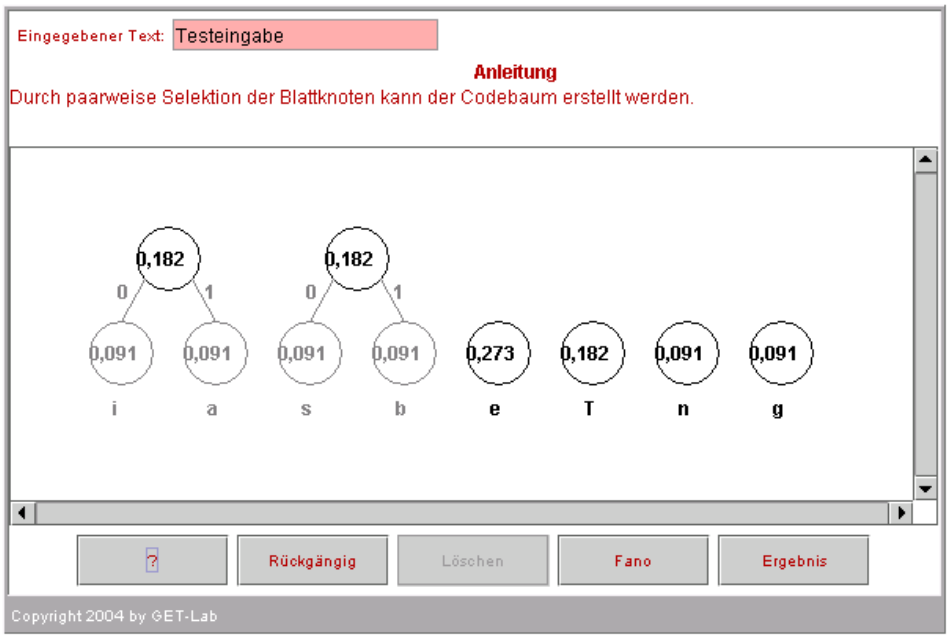


Abbildung 7.14: Konstruktionsbreich des Bausteins Kondierungsverfahren nach Huffman

der bzw. die Lernende in den Konstruktionsbereich, in dem der Huffman-Baum individuell aufgebaut werden kann. Freilich läßt sich nicht immer die optimale Lösung finden, das Applet weist jedoch auf eine solche hin und ermöglicht danach einen neuen Versuch (vgl.

Abbildung 7.14). Nachdem der Baum erstellt wurde, findet eine Auswertung statt, bei der die konstruierte Lösung in Relation zu der Eingabe ausgewertet wird (vgl. Abbildung 7.15).

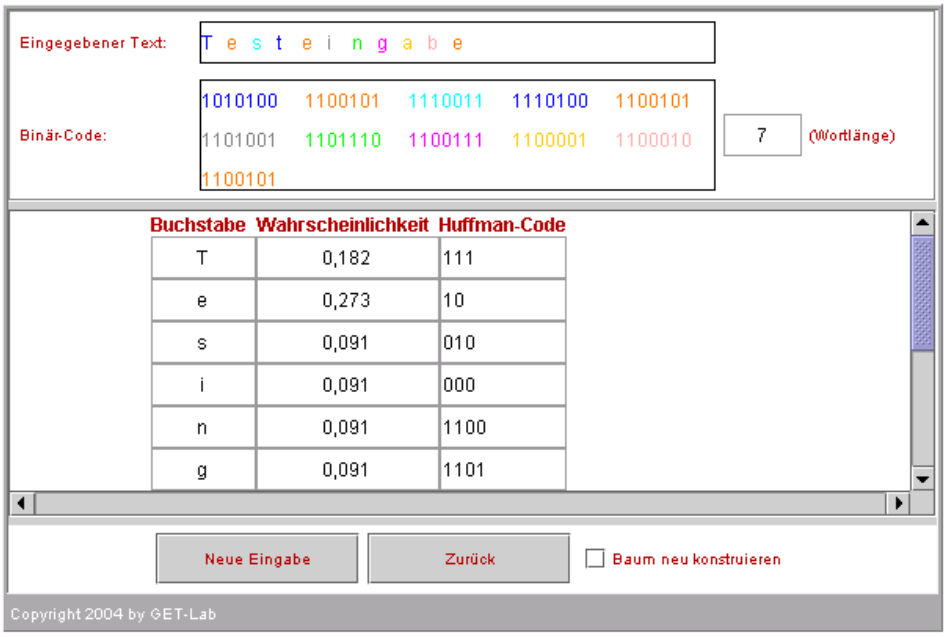


Abbildung 7.15: Auswertungsbereich des Bausteins Kondierungsverfahren nach Huffman

Ein weiterer Baustein ermöglicht die Umwandlung von Zahlen unterschiedlicher Basen. Studierende denken sich eine Zahl aus, die sie mit den Verfahren *Umwandlung über Potenztabellen*, *Divisions-Rest-Verfahren* oder dem *Horner Schema* von einem Zahlensystem in ein anderes umwandeln können. Abbildung 7.16 zeigt die Umwandlung der Dezimalzahl 123, in das Dualsystem. Es können beliebige Einträge aus der Potenztabelle ausgewählt werden, die dann von der Zahl im Quellsystem subtrahiert werden.

Eine weitere Interaktionsform für eine Explorationsumgebung ist der Baustein *Quizzes für Komplexe Zahlen*. Studierende können die zuvor erlernten Rechenregeln für Komplexe Zahlen anhand einer Quiz-Umgebung prüfen und erkennen, ob sie die Regeln richtig anwenden können (vgl. Abbildung 7.17).

Zudem existiert der, an der Universität Paderborn entwickelte, Baustein *Überlagerung zweier harmonischer Schwingungen* mit dem Studierende das Überlagerungsverhalten zweier harmonischer Schwingungen mit unterschiedlicher Amplitude experimentell erlernen. Abbildung 7.18 zeigt das, auf Geonext [Geo05] basierende Applet, mit dem Studierende interaktiv zwei rotierende Zeiger bewegen können. Die resultierende Schwingung wird dann in das Applet gezeichnet, wodurch der Zusammenhang beider Schwingungen verdeutlicht wird.

Abschließend wird ein weiterer Baustein der Explorationsumgebung *Komplexe Zahlen* vorgestellt, der ebenfalls an der Universität Paderborn entstanden ist. Mit dem Baustein *Rechnen mit komplexen Zahlen* können zwei komplexe Zahlen definiert werden, die in

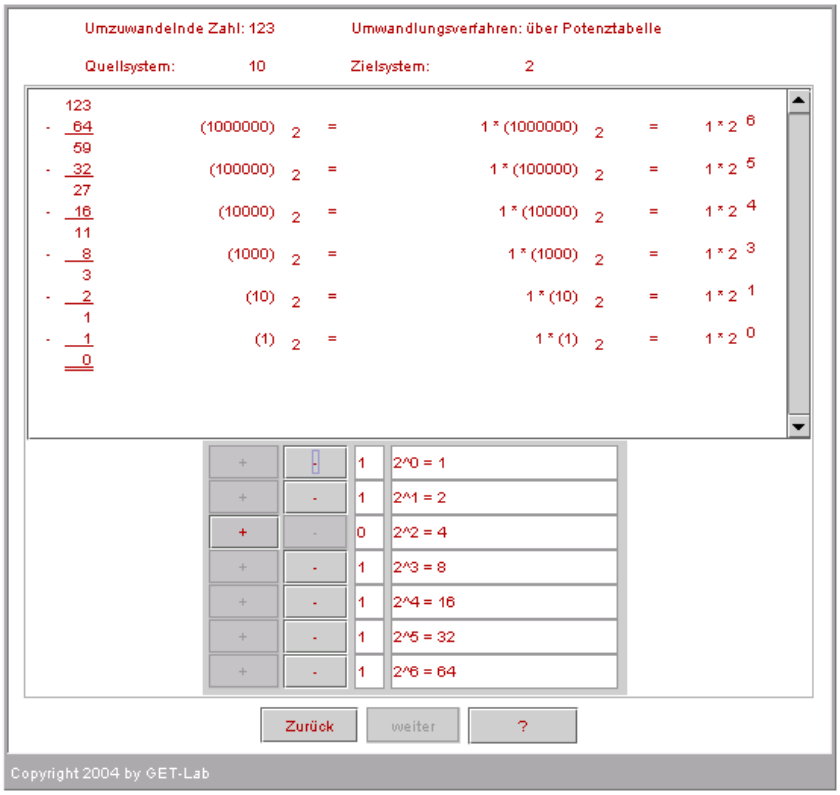


Abbildung 7.16: Baustein zur Zahlenumwandlung

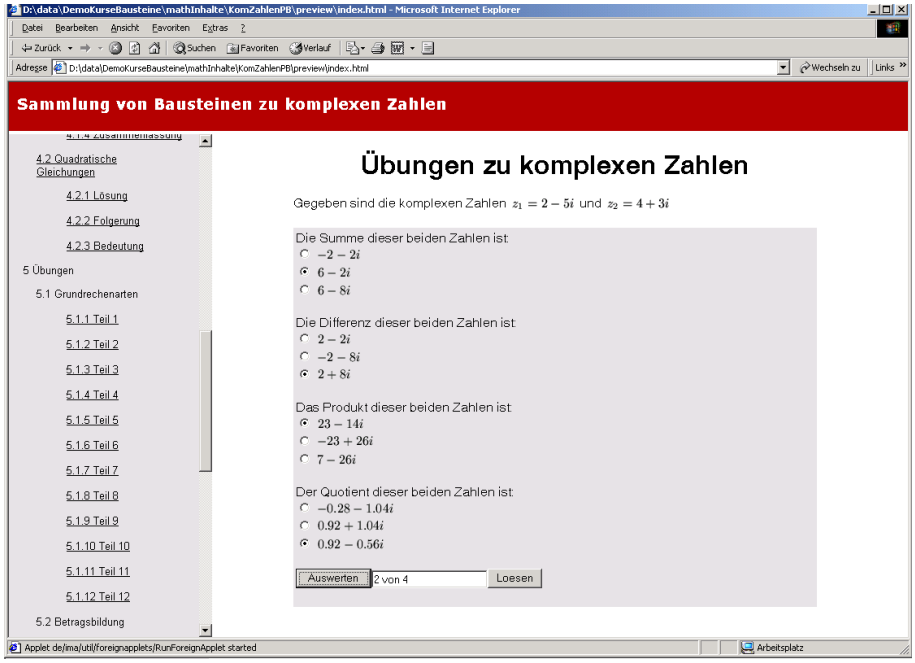


Abbildung 7.17: Baustein Quiz-Umgebung Komplexe Zahlen

$$x(t) := x_1(t) + x_2(t) = (A_1 + A_2)\, e^{i(\omega t + \varphi)}$$

mit der gleichen Phase aber der Amplitude $A_1 + A_2$. Daraus ist sofort ersichtlich, dass die Überlagerung zweier harmonischer Schwingungen wieder eine solche ist.

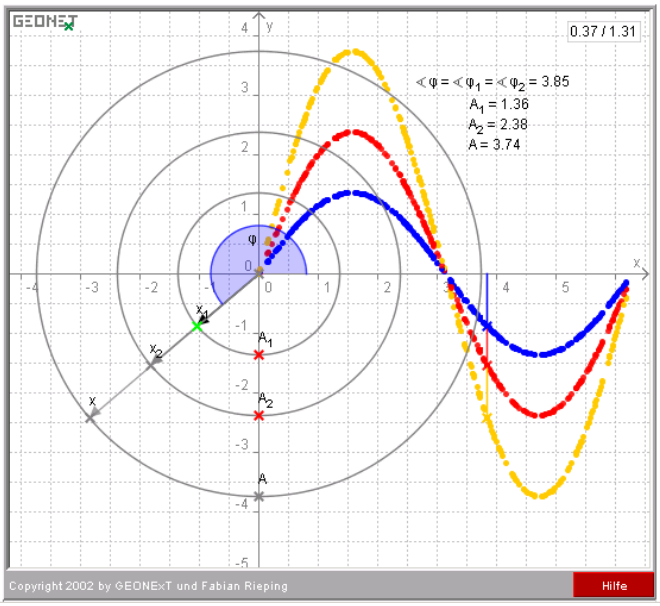


Abbildung 7.18: Überlagerung zweier harmonischer Schwingungen

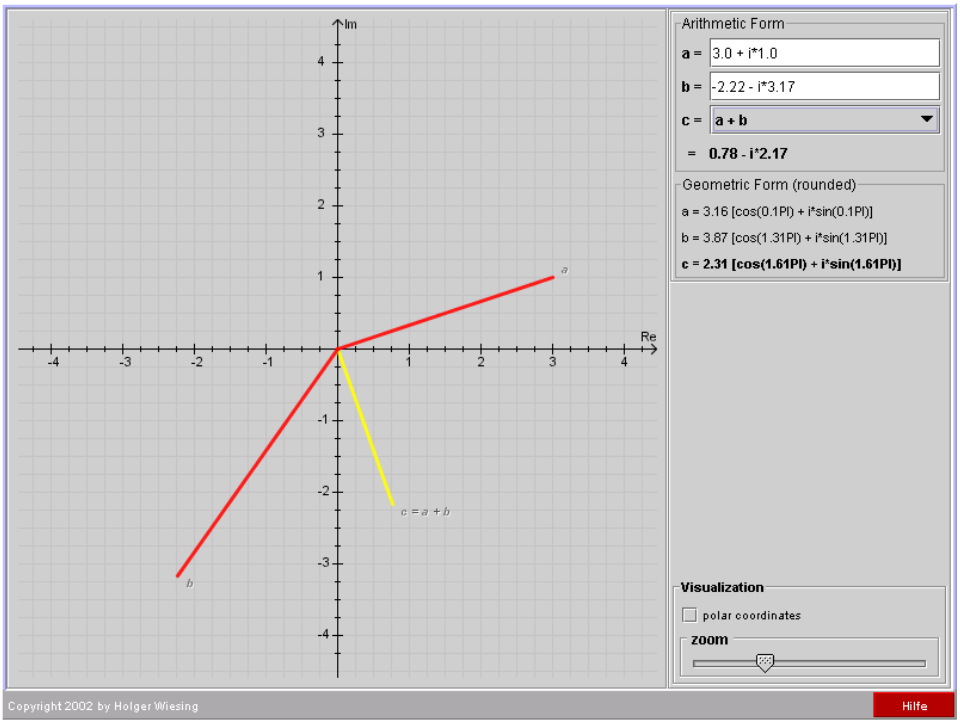


Abbildung 7.19: Rechnen mit komplexen Zahlen (Kartesische Koordinaten)

einem kartesischen Koordinatensystem dargestellt werden (vgl. Abbildung 7.19). Studierende können verschiedene Rechenoperationen auswählen, wodurch die resultierende komplexe Zahl in das Applet eingezeichnet wird. Durch interaktives Einwirken können Studierende die eingezeichneten Werte manipulieren, so dass die anderen sich, entsprechend der ausgewählten Operation, neu positionieren. Mit diesem Baustein erlernen Studierende konstruktivistisch die Grundlagen für das Rechnen mit komplexen Zahlen.

Teil III

Analyse

Kapitel 8

Zusammenfassung

Ziel dieser Arbeit war die Entwicklung eines technischen Modells zur Konstruktion modularer und wiederverwendbarer Lerneinheiten, die speziell für den interdisziplinären Einsatz nutzbar sein sollten. Hierfür war die Entwicklung eines abstrakten Modells notwendig, das neben der strukturellen Kopplung von Lerneinheiten auch deren inhaltliche Kompatibilität und Integrität unterstützt. Um dieses Ziel zu erreichen, wurden in Kapitel 2 zunächst die Begriffe *Multimediales Lernen* und *E-Learning* definiert, denn diese dienten als Grundlage für die nachfolgenden Abhandlungen. Ebenso war die Darlegung lerntheoretischer Grundlagen für die Modellierung von Kursmaterialien äußerst wichtig. Hierzu gehört der Lernprozess des Menschen, der chronologisch vom Anfänger mit wenig Erfahrung Schritt für Schritt zum Experten avanciert. Das Fortschreiten zu der nächst höheren Fertigkeitsstufe wird durch unterschiedliche Methoden, den Lernparadigmen, erreicht. In dieser Arbeit wurden die Paradigmen *Behaviourismus*, *Kognitivismus* und *Konstruktivismus* eingeführt. Des weiteren wurden wichtige Standardisierungen aus dem E-Learning-Bereich vorgestellt, deren Prinzipien bei der Entwicklung des Modell berücksichtigt wurden. Hierzu gehören vor allem die folgenden Standards: *ADL-SCORM*, *IMS Content Packaging*, *AICC* und *LOM*. Sie lieferte wichtige Erkenntnisse für die Aggregation von Kursen und deren Kompatibilität. Neben diesen technischen Modellen existieren auch theoretische Überlegungen, die versuchen, Lernmaterialien als abstrakte Objekte zu beschreiben und deren Zusammenspiel zu analysieren. Es wurde vor allem der metaphorische Vergleich zu anderen Modellen, wie z. B. dem Atom-Modell, hergestellt, um die aus anderen Bereichen gewonnenen Erfahrungen auf das E-Learning zu übertragen.

In Kaptiel 3 fand die Analyse bereits bestehender Systeme und Verfahren statt, an denen zurzeit geforscht und entwickelt wird. Dieses sind im industriellen Zweig Learning Management Systeme (LMS) und Autorenwerkzeuge sowie formale Beschreibungssprachen und Verfahren zur Wiederverwendung von Materialien im Universitären Umfeld. Die Analyse ergab jedoch, dass kein System und kein Verfahren die in dieser Arbeit geforderten Ziele zufriedenstellend umgesetzt hat.

Als Ergebnis stellt Kapitel 5 das Modell zur Entwicklung konsistenter Lernmaterialien vor. Bei diesem völlig neuem Ansatz wird erstmals die Lücke zwischen den bestehenden technischen Modellen, die im Wesentlichen die strukturierte Datenhaltung spezifizieren,

und den theoretischen Modellen für Lernobjekte geschlossen. Lernobjekte werden nicht nur in ihrer Struktur, sondern vielmehr durch ihren Inhalt spezifiziert und ermöglichen somit die *echte* Aggregation, wie sie durch die Metaphern gefordert wird. Das Ergebnis waren formatierbare Lernbausteine und Kurse sowie das abstrakte Kursmodell, die eine höhere Abstraktionsebene als bisherige Verfahren aufweisen. Diese Abstraktionsebene setzt jedoch die Separation von Formatierungsregeln voraus. Durch Transformationspakete wurden die für eine spezialisierte Repräsentation benötigten Regeln genau festgelegt, mit dem Ziel, ein hohes Maß an Anpassbarkeit und Wiederverwendbarkeit zu erhalten. Diese Kriterien sinddie grundlegende Voraussetzung für den kooperativen Entwicklungsprozess, bei dem keine spezialisierten Inhalte bearbeitet werden, sondern Kursfragmente auf rein abstrakter Ebene entwickelt, aggregiert und wiederverwendet werden.

Aufbauend auf den in Kapitel 6 eingeführten theoretischen Grundlagen, hat das Kapitel Umsetzung die technische Realisierung vorgestellt. Zunächst wurde das Autorensystem Lyssa präsentiert, welches als Referenzimplementierung aus dem Projekt mαth-kit hervorgegangen ist. In Relation zu dem Kursmodell wurden die jeweiligen Funktionalitäten des Systems nacheinander beschrieben und deren Umsetzung diskutiert. Das Augenmerk hat sich vor allem auf die Kernfunktionalität, auf die sich das Kursmodell bezieht, gerichtet. Aufgrund der verschiedenen Forschungsaspekte und der Funktionsvielfalt des Systems, wurden lediglich die für diese Arbeit relevanten Aspekte beleuchtet. Dieses sind vor allem die formatierbaren Bausteine und Kurse, die Abbildung bestehender Formate sowie die Kurstransformation in Fremdformate. Dieses Kapitel hat insbesondere die technische Realisierbarkeit des theoretischen Modells verdeutlicht und hat zudem einen möglichen Realisierungsansatz vorgestellt. Zu dem Autorenwerkzeug wurde ein weiteres Werkzeug konzipiert, mit dem sich Transformationspakete entwickeln und modifizieren lassen. Experten und Expertinnen wird hierdurch sowohl die Entwicklung neuer Präsentationsformen als auch die einfache Modifikation vorab definierter Layoutmerkmale ermöglicht. Der letzte Abschnitt hat die technische Umsetzung des CKS gezeigt, der die Verwaltung von Lernbausteinen in einem kooperativen Arbeitsumfeld unterstützt. Dieser Server wurde als Web-Applikation realisiert und kann wahlweise über einen Web-Zugang oder durch die direkte Einbindung in das Autorensystem eingesetzt werden.

Kapitel 7 hat exemplarisch die Entwicklung eines Kurses zum Thema *Technische Informatik* demonstriert. Ziel war es, die Leitidee des Modells zu vermitteln und dessen praktische Realisierung zu verdeutlichen. Entwickler und Entwicklerinnen von Onlinekursen sind nun in der Lage, Lernbausteine zu entwickeln und mit anderen Bausteinen wiederholt zu aggregieren und disaggregieren. Zur Veranschaulichung dieser neuartigen Entwicklungsphilosophie wurde das Thema *Komplexe Zahlen* modular aufbereitet und abstrakt formuliert.

8.1 Evaluation

Ein Teil der wissenschaftlichen Arbeit ist die Evaluation der erzielten Ergebnisse. Gewiss lassen sich bei der Definition einer neuen Formel oder eines physikalischen Experiments

die Ergebnisse beweisen bzw. auf ihre Richtigkeit hin untersuchen. In dieser Arbeit soll auf eine solche Untersuchung nicht verzichtet werden, allerdings sind die Ansatzpunkte nicht so klar ersichtlich, wie zuvor beschrieben. In dieser Arbeit wurde für die Lösung eines Problems ein technisches Modell entwickelt, mit dem es möglich ist, konsistente Lernmaterialien zu produzieren. Um zu beweisen, dass das Modell praxistauglich ist, wurde daher ein Prototyp entwickelt, der das Modell als Referenzimplementierung umsetzt. Die reine Umsetzung reicht jedoch als Beweis zur Lösung des Problems nicht aus. Vielmehr müssen Untersuchungen durchgeführt werden, wie sich das Modell bei dem Zusammenspiel mehrerer Autoren und Autorinnen verhält, und ob das zugrunde liegende Konzept überhaupt akzeptiert wird, denn es wurde bereits mehrmals darauf hingewiesen, dass modulare Konstruktionskonzepte einen höheren Einarbeitungsaufwand verursachen. Aus diesem Grund soll nun das Testumfeld erläutert werden.

Der Prototyp Lyssa wurde im Projektverlauf an den vier Standorten Paderborn, Hamburg, Hagen und Bayreuth ausgiebig getestet. Bei dem Test handelte es sich um einen Funktionstest, der zum Auffinden von Fehlern diente und zudem die Anwendbarkeit durch verschiedene Benutzer und Benutzerinnen untersuchen sollte. Durch ständige Autor-Kritiker-Zyklen [Züllighoven98] wurden immer wieder von Seiten der Anwender und Anwenderinnen Verbesserungsvorschläge eingebracht. Ein Beispiel hierfür ist die Zusammenlegung des Übersetzers und des Baukastens in einen Prozessraum, wodurch das Übersetzungsrahmenwerk entstanden ist. Des weiteren wurden zahlreiche Anregungen für die Erweiterung des Sprachumfangs von BrickML gegeben, die daraufhin umgesetzt und aufs Neue getestet wurden. Insgesamt wurde der Prototyp von 19 Personen ausgiebig getestet. Davon waren 5 Personen in Paderborn, 4 in Hagen, 8 in Hamburg und zwei an der Hochschule in Bayreuth beteiligt. Des weiteren wurde während des Förderzeitraums eine Testversion des Prototypen auf der Projekt-Homepage[1] bereitgestellt, die sich Benutzer und Benutzerinnen bei näherem Interesse herunterladen konnten. Über einen Registrierungsdialog wurden einhundert Personen registriert, die ebenso Anregungen zur aktuellen Version zuschickten. Insgesamt wurde das neue Konzept gut angenommen und Lerninhalte fachübergreifend entwickelt. Die Resultate sind ebenfalls auf der Projekt-Homepage zu finden und zeigen Online-Präsentationen übersetzter formatierbarer Kurse.

[1] www.math-kit.de/download (29.10.2005)

Kapitel 9

Bewertung

Die vorliegende Dissertation schlägt ein technisches Modell vor, anhand dessen sich Autorensysteme entwickeln lassen, die modulare und wiederverwendbare Kurse erzeugen. Der Stand der Forschung und Technik, der in Kapitel 2 und 3 dargelegt wurde, zeigt, dass es sowohl theoretische als auch praktische Lösungen gibt, die allerdings die eingehende Fragestellung aus Kapitel 1 nicht umfassend beantworten. Diese Arbeit schlägt erstmals eine Brücke zwischen der Theorie von Lernobjekten und deren technische Umsetzung und stellt Lösungen für die auftretenden Probleme bezüglich der Granularität, Modularität und Kompatibilität vor.

Ein Ziel dieser Arbeit war die Sensibilisierung für die Probleme, die bei der technischen Aggregation von Lernobjekten auftreten. Sie wird dadurch erreicht, dass Entwickler und Entwicklerinnen sich von der Vorstellung einer konkreten technischen Umsetzung distanzieren und technische Lernobjekte vielmehr als ein abstraktes Objekt ansehen, das auf eben diesem Niveau Operationen und Schnittstellen anbietet. Durch diese Art der abstrakten Modellierung und die zugehörige technische Umsetzung werden, die am Anfang der Arbeit aufgeworfenen Probleme vollständig gelöst. Die Entwicklung des Prototypen und der praktische Einsatz haben gezeigt, dass sich das Konzept in der Praxis bewährt, weil Anwender und Anwenderinnen den Mehrwert erkennen und ihn bei der Modellierung aufgreifen (vgl. Abschnitt 8.1).

Abschließend findet eine verfeinerte Bewertung der Ergebnisse in Bezug auf die in Kapitel 1 definierte Problembeschreibung statt. Hierzu werden die in Tabelle 2.1 formulierten Fragestellungen separat aufgegriffen und beantwortet.

Formatierbarkeit: Das Problem der Formatierbarket wurde gelöst, indem sowohl die Struktur von Kursen als auch deren Inhalt vom Präsentationsformat getrennt wurden. Durch Einsatz von Transformatoren (vgl. Abschnitt 5.5) können abstrakt formulierte Inhalte individuell adaptiert werden und Präsentationen einfach angepasst bzw. neu entwickelt werden.

Modularität: Kompatibilität zwischen Lernbausteinen wird erreicht, indem der Aufbau der Lerneinheit, also die Struktur, von dem eigentlichen Inhalt getrennt wird. Der IMS Content Package Standard (vgl. Abschnitt 2.3.2) bietet hierfür bereits elegante

Lösungen an, betrachtet jedoch nicht den konkreten Inhalt. Das Modell schlägt als Lösungsansatz Bausteine vor, die formatierungsfrei definiert werden (vgl. Abschnitt 5.2), und darüber hinaus keine relevanten Informationen für die Kursstrukturierung enthalten. Die Aggregationsfähigkeit ergibt sich durch die Einhaltung von Regeln, wie z. B. die Abgeschlossenheit (vgl. Abschnitt 7.1.2).

Granularität: Das Modell definiert grundsätzlich keine Restriktionen bezüglich der Granularität, sondern lässt eine freie Skalierbarkeit von Lernbausteinen zu. Sie ist notwendig, da sich der Gesamtumfang einer Lerneinheit zum Zeitpunkt der Entwicklung nicht abschätzen lässt. Aufgrund der rekursiven Definition von Lernbausteinen können sie schlichtweg zu jedem Zeitpunkt aggregiert bzw. dissaggregiert werden.

Interdisziplinarität: Das Ziel, Lernmodule interdisziplinär einsetzen zu können, d. h. über Fachgrenzen hinaus, ist vor allem durch die thematische Nähe der naturwissenschaftlichen Fächer innerhalb des mαth-kit Projekts bedingt. Die Fragen nach Modularität und Formatierbarkeit sind wichtige Voraussetzungen zur Lösung dieser Fragestellung und tragen einen erheblichen Anteil dazu bei, denn das technische Modell bietet erst hiermit die Rahmenbedingungen, um Lerneinheiten aus anderen Fachdomänen auseinander zu nehmen, relevante Teile aufzugreifen und diese wieder zu verarbeiten. Im Projekt mαth-kit sind Lernbausteine entwickelt worden, die in den Disziplinen Mathematik, Informatik, und Maschinenbau wiederverwendet wurden (vgl. Abschnitt 7.1.3).

Nachhaltigkeit: Die mit dem Modell erzeugten Lerninhalte lassen sich zu jedem Zeitpunkt in ein neues Format überführen. Auf diese Weise kann der langfristige Erhalt gesichert werden. Ferner lassen sich transformierte Inhalte in ein standardisiertes Format generieren, so dass die Nachhaltigkeit ebenso gesichert ist (vgl. Abschnitt 2.3).

Integrierbarkeit: Das Problem, digitale Ressourcen für die Entwicklung neuer Kurse wiederzuverwenden, wurde durch das Konzept des abstrakten Kursmodells gelöst. Dieses entkoppelt funktional die modulare Erzeugung von der Verarbeitung des Quellformats und ermöglicht daher die einfache Erweiterung durch neue Interpreter [1].

Kooperation: Die Evaluation (vgl. Abschnitt 8.1) hat gezeigt, dass kooperatives Arbeiten durch das Modell bzw. durch die Implementierung des CKS umfassend unterstützt wird. Verteilte Teams sind nunmehr in der Lage, gemeinsam mit dem Baukasten und den enthaltenen Bausteinen zu arbeiten.

Mathematik: Mathematische Inhalte zeichnen sich vor allem durch den hohen Anteil an Formeln und Gleichungen aus. Das Kapitel 2 hat jedoch gezeigt, dass es zur Darstel-

[1] Eine solche Erweiterung wurde z. B. in der Diplomarbeit *Entwicklung einer Software-Komponente zur Integration bestehender Lehr- und Lernmaterialien und deren Metadaten in das mαth-kit System* untersucht [Sünneli04].

lung ganz unterschiedliche Verfahren gibt. Hierbei hilft das Konzept der Prozessoren, die beliebige Formatkonvertierungen durchführen können.

Didaktik: Das in dieser Arbeit entwickelte Modell sieht die technische Entwicklung von Lerneinheiten vor. Hierbei sollen jedoch keine didaktischen Aspekte modelliert werden. Fragen der Didaktik beschäftigen sich mit der Gestaltung, Lehrtechnik und Vermittlung von Lerneinheiten, die bereits in der Planungsphase beantwortet werden müssen (vgl. [Pawlowski02, Pawlowski01]). In dieser Phase muss entschieden werden, welchen Lernparadigmen gefolgt wird, und welche Präsentation bzw. Gestaltung durch eine Transformation erzeugt wird. Das Modell muss jedoch in der Lage sein, alle Varianten zu bedienen, was durch die Gestaltungsfreiheit bei der Entwicklung von Transformationspaketen und der Verankerung verschiedener Techniken in die Lerneinheiten zweifellos möglich ist.

Kapitel 10

Ausblick

Das grundlegende Ziel dieser Arbeit war die Entwicklung eines Modells zur orthogonalen Definition universell einsetzbarer Lernmodule. Da es sich hierbei um einen neuen Ansatz handelt, der zahlreiche interessante Fragestellungen aufwirft, konnten in der vorliegenden Dissertation nicht alle Aspekte behandelt und untersucht werden. Dieses Kapitel zeigt Ansatzpunkte für weiterführende Forschungsfragen, die das Modell verbessern und erweitern sollen. Des weiteren werden Vorschläge für Weiterentwicklungen des Prototypen Lyssa formuliert, die insbesondere die Anwendbarkeit des Systems verbessern sollen.

Die Anforderungen an die Auszeichnungssprache BrickML sind aus dem Bedarf des mαth-kit Projekts entstanden. Die beteiligten Personen haben Vorschläge und Wünsche geäußert, die mit in die Kernentwicklung eingeflossen sind. Dieser Fundus ist keineswegs vollständig und kann in Zukunft immer weiter verfeinert werden, um eine möglichst große Akzeptanz zu schaffen. Ferner können weitere domänenspezifische Sprachen abgeleitet werden, die die Erzeugung spezialisierter Inhalte ermöglicht, ohne den Basissatz der Grundsprache zu erweitern.

Des weiteren gibt es Ansätze, Lehrinhalte an den Wissensstand von Lernenden anzupassen. Das kann dadurch erreicht werden, indem Inhalte mit Metadaten angereichert werden, die die Qualifikation bzw. das Grundwissen von Lernenden definieren. Eine solche Vorbedingung ermöglicht die selektive Zuordnung von Lernmodulen durch eine Lernumgebung. Diese Methode wird im Allgemeinen als *Adaptives Lernen* [Brusilovsky98, Brusilovsky01] bezeichnet und wurde bisher nicht in das Modell integriert.

Bei der Abbildung bestehender digitaler Ressourcen auf Kurse kristallisieren sich weitere interessante Fragestellungen heraus. Ein Gedankenanstoß, der in dieser Arbeit begründet ist, wendet Methoden des Refactorings an, die von dem Paradigma der objektorientierten Programmierung abgeleitet (vgl. Abschnitt 5.4.3) sind. Die hier vorgestellten Lösungen sind jedoch recht allgemein gefasst und lösen die auftretenden Probleme nur im Ansatz. Ferner kann über Prozesse zur Nachbearbeitung nachgedacht werden, die Hinweise auf inkonsistente Lernbausteine geben und deren Überarbeitung anregen.

Ein ebenso interessanter Forschungsansatz bildet die Integration zeitvarianter Inhalte in das Modell. Hiermit ist insbesondere die zeitliche Veränderung grafischer Objekte gemeint, die z. B. durch den SMIL-Standard [Consortium05] spezifiziert wird. In Abschnitt

3.4 wurde bereits darauf hingewiesen, dass Autorenwerkzeuge in dokumentbasierte und programmbasierte Systeme aufgeteilt werden. Bisher waren lediglich Komponenten innerhalb der Bausteine interaktiv steuerbar. Wie sähe jedoch ein Modell aus, bei dem der Baustein interaktiv verändert werden kann und dessen Zustand sich zeitlich ändert? Dieses Modell könnte die bestehende Lücke zwischen programmbestimmten und dokumentbestimmten Werkzeugen schließen und die Kompatibilität von Inhalten durch entsprechende Schnittstellen erhöhen.

Als letztes wird das Autorensystem Lyssa betrachtet. Der Prototyp Lyssa verfügt zurzeit über eine Bausteinansicht, in der Inhalte direkt in XML gesetzt werden. Unterstützt wird dieser Prozess durch einen XML-Editor, der je nach Vorlieben des Anwenders bzw. der Anwenderin, ausgewählt werden kann. Eine sinnvolle Erweiterung des Prototypen würde darin bestehen, von der technischen Ansicht zu abstrahieren und einen Editor zu entwickeln, der kein technisches Grundwissen voraussetzt. Diese Maßnahme würde die Benutzbarkeit und damit den Zugang zu dem Prototypen vereinfachen.

Teil IV

Anlagen

Anhang A

Abkürzungsverzeichnis

ADL	–	Advanced Distributed Learning Initiative
AICC	–	Aviation Industry Computer Based Training Committee
ANSI	–	American National Standards Institute
ARIADNE	–	Aliance of Remote Instructional Authoring and Distibution Networks for Europe
CAM	–	Content Aggregation Model
CBT	–	Computer Based Training
CP	–	Content Package
CKS	–	Construction Kit Server
CMI	–	Computer Manage Instruction Systems
DTD	–	Data Type Definition
GUI	–	Graphical User Interface
EML	–	Educational Markup Language
FB	–	Formatierbarer Baustein
FK	–	Formatierbarer Kurs
FO	–	Formating Objects
FOP	–	Formating Objects Processor
FTP	–	File Transfer Protocol
HTML	–	Hypertext Markup Language
HTTP	–	Hypertext Transfer Protocol
ISO	–	International Organization for Standardization
IEEE	–	Institute of Electrical and Electronics Engineers
IMS	–	Instructional Management System
JDOM	–	Java Document Object Model

nächste Seite

KPS	–	Knowlage Pool System
LCMS	–	Learning Content Management System
LMS	–	Learning Management System
LO	–	Learning Object
LOM	–	Learning Object Metadata
LMML	–	Learning Material Markup Language
LTSC	–	Learning Technology Standard Committee
MDI	–	Multiple Device Interface
MSC	–	Mupad Computing Server
OOP	–	Object Oriented Programming
PIF	–	Package Interchange File
SCO	–	Sharable Content Object
SCORM	–	Sharable Content Object Reference Model
SGML	–	Standard Generalized Markup Language
TP	–	Transformationspaket
UML	–	Unified Modelling Language
W3C	–	World Wide Web Consortium
WBT	–	Web Based Training
WWW	–	World Wide Web
XML	–	Extensabile Markup Language
XSD	–	extended Stylesheet
XSL	–	extendable Stylesheet Language

Anhang B

Glossar

ADL: ADL ist eine Initiative des amerikanischen Verteidigungsministerium zur Standardisierung von Lerneinheiten.

AICC: Das AICC ist ein internationaler Zusammenschluss von Herstellern computerbasierter Lerneinheiten. Ziel ist die Verabschiedung von Standardisierungen im Bereich Lernobjekte.

ARIADNE: Die *Aliance of Remote Instructional Authoring and Distibution Networks for Europe* (ARIADNE) ist ein EU-Projekt mir der Zielsetzung, Werkzeuge und Methoden zur Produktion, Verwaltung und Wiederverwendung von rechnergestützten und pädagogischen Lerneinheiten sowie für die Erstellung von Curricula zu entwickeln. Hierbei wurde maßgeblich an der Spezifikation des LOM Standards mitgewirkt und kulturelle Aspekte eingebracht.

Asset: Asstes, wie z.B. Bilder, Videos oder Applets, sind physikalische Dateien, aus denen eine Lerneinheit zusammengesetzt ist.

CP: Das Content Package wurde ursprünglich vom IMS Consortium verabschiedet und ist in den SCORM Standard aufgenommen worden. Es enthält das Manifest mit den Metadaten sowie die zugehörigen Assets.

DTD: Die Documenttyp-Definition (DTD) dient zur Definition eines Dokumentenmodells. Mit ihr lassen sich semantische Abhängigkeiten in XML Dokumenten formulieren.

FB: Ein formatierbarer Baustein (FB) besteht aus einem generellen, darstellungsfreien Dokument beliebiger Granularität zur strukturierten Speicherung des Inhalts und wird erst in Verbindung mit einem Regelwerk für ein beliebiges Präsentationsformat zu einem Lernobjekt. Formatierbare Bausteine besitzen einen definierten Rahmen, der die Aggregation mit anderen formatierbaren Bausteinen ermöglicht.

FK: Ein formatierbarer Kurs (FK) besteht aus mindestens einem formatierbaren Baustein und einer beliebigen Anzahl Assets und definiert deren Organisation. Erst in

Verbindung mit einem Regelwerk für ein beliebiges Präsentationsformat entsteht eine einsetzbare Kurseinheit. Formatierbare Kurse besitzen einen definierten Rahmen, der die Aggregation mit anderen formatierbaren Kursen ermöglicht.

IEEE-LTSC: Das LTSC (Learning Technology Standard Committee) entwickelt Richtlinien zur Standardisierung im Bereich webbasierter Lern- und Lehrsysteme.

LT: Anhand des Layout-Typs wird entschieden, wie ein Wert zur Layout-Manipulation vom Lyssa-Designer angezeigt wird. Die Einstellung eines Farbwerts ist z. B. an einen Auswahldialog für Farben gekoppelt und Zahlenwerte an eine Textbox.

Learning Management System: Unter einer webbasierten Lernplattform ist eine serverseitig installierte Software zu verstehen, die beliebige Lehrinhalte über das Internet vermittelt und die Organisation der dabei notwendigen Lernprozesse unterstützt ([Baumgartner02], Seite 14).

Lernobjekt: Learning Objects bestehen in der Regel aus einer Menge an Informationseinheiten oder Datenobjekten, wie z.B Text, Videos oder Programmen. Aus ihnen können durch Aggregation oder Komposition verschiedene Kurse zusammengestellt werden, die wiederum selbst zu kompletten Lehrgängen kombiniert werden können (Baumgartner [Baumgartner02], Seite 33).

Manifest: Das Manifest ist Bestandteil des Content Package und bündelt strukturelle Kursinformationen, wie z.B. Metadaten oder den hierachischen Aufbau der Kapitel.

Metadaten: Metadaten sind Daten über andere Daten. Im Fall des SCORM-Standards und der IMS-Content Package Spezifikation werden sie, nach dem LOM-Standard, in XML kodiert. Sie helfen beim Auffinden von Kursinhalten in Repositories und beim Einstellen einer Lerneinheit.

SCO: Ein SCO *(Sharable Content Object)* ist eine austauschbare Lehreinheit, die in verschiedenen Kontexten integriert werden kann. Damit diese Vorderung eingehalten wird, dürfen keine Referenzen auf andere Lerneinheiten vorhanden sein.

SGML: SGML ist ein allgemeiner Standard zur Definition von Auszeichnungssprachen (Markup Sprachen) und ist der Vorläufer von XML. Mit SGML können Dokumente unabhängig von ihrer Darstellung strukturell gespeichert werden.

TP: Ein Transformationspaket kapselt die für die Übersetzung notwendigen Dateien und stellt somit einen flexiblen und modularen Mechanismus zur Erzeugung unterschiedlicher Ausgabeformate dar. Transformationspakete werden im Zusammenhang mit formatierbaren Kursen eingesetzt um Lernobjekte zu erzeugen.

Workbench: Der Zugriff auf Dateiressourcen erfolgt bei dem Autorensystem Lyssa über die Workbench. Von dort aus können sie, per Drag'n'Drop, mit Bausteinen oder Kursfenstern, ausgetauscht werden.

XML: die extensabile Markup Language ist der Nachfolger von SGML und wird zur darstellungsunabhängigen Beschreibung von Dokumenten verwendet. Im Gegensatz zu SGML wird ein reduzierter Sprachumfang eingesetzt, der die Implementierung von Parsern vereinfacht.

XML-Schema: Hierbei handelt es sich um ein auf XML basierendes Dokumentenmodell, welches es ähnlich wie die DTD ermöglicht, die Grammatik der Sprache zu definieren.

Anhang C

Attribute

Nummer	Name	Erklärung	Benötigt	Typ
1	Manifest	Kontainer für Transformatoren und Anwendungen	B	Container
1.1	desc	Kurszbeschreibung des Transformationspakets	B	String
1.2	identifierRef	Verweist auf die Icon-Datei	B	String
1.3	archive	Gibt an, ob das Ergebnis in einem komprimierten Archiv vorliegt	O	Boolean
1.4	ext	Gibt das Archivsuffix an	O	String
1.5	starthref	Verweist auf die Start-Datei des übersetzten Kurses	B	String
1.6	transformers	Kontainer für Transformatoren	B	Container
1.6.2	transformer	Kontainer für Transformatoren und Anwendungen	B	Container
1.6.2.1	name	Name des Transformators	B	String
1.6.2.2	type	Typ des Transformators	B	String
1.6.2.3	ext	Gibt das Suffix für die Zieldateiena an	B	String
1.6.2.4	id	Id des Transformators	B	String
1.6.2.5	identifierRef	Verweist auf die Quellen für den Tranformator	B	String
1.6.2.5	srcRef	Verweist auf zu transformierende Resourcen	O	String
1.6.2.6	prozessor	Definiert einen Prozessor für einen Transformator	O	Container
1.6.2.6.1	identifierRef	Verweist auf die Resourcen des Prozessor	B	String

nächste Seite

Nummer	Name	Erklärung	Benötigt	Typ
1.6.2.6.2	active	Aktiviert den Prozessor	B	Boolean
1.6.2.6.3	order	Bestimmt die Reihenfolge der Prozessoren	O	Integer
1.7	tasks	Kontainer für Anwendungen	B	Container
1.7.1	task	Liste aller Anwendung	O	Container
1.7.1.1	type	Definiert den Zeitpunkt, an dem die Anwendung ausgeführt wird	B	String
1.7.1.2	lang	Programmiersprache	B	String
1.7.1.3	class	Definiert die zu startende Klasse	B	String
1.7.1.4	identifierRef	Referenziert die Resource für die ausführbare Datei	B	String
1.8	sceme	Kontainer fü Schema-Dateien	O	Container
1.8.1	identifierRef	Referenziert die Resource der Schema-Datei	O	String
1.9	layout	Beschreibt das Layout des Transformationspakets	B	String
1.9.1	identifierRef	Referenziert die Resource des Layouts	O	String
1.10	resources	Gibt die Liste der Resourcen an	B	Container
1.10.1	resource	Definiert eine Resourcen an	B	Container
1.10.1.1	file	Definiert eine Datei innerhalb einer Resource	B	String
1.10.1.2	href	Verweis auf die Datei	B	String
1.10.2	identifier	Setzt die Id mit der die Resource angesprochen wird	B	String

Tabelle C.1: Attribute und Elemente des Transformationspakets

Nummer	Name	Erklärung	Benötigt	Typ
1	Course	Kontainer für Lektionen	B	Container
1.1	Metadaten	LOM Metadaten für den Kurs	O	Container
1.2	Lesson	LOM Metadaten für den Kurs	O	Container
1.2.1	courseObject	Gibt an, ob es sich um einen Subkurs handelt	B	Booolean
1.2.2	titel	Enthält den Titel einer Lektion	B	String
1.2.3	Metadaten	LOM Metadaten für die Lektion	O	Container
1.2.4	Lesson	Unterlektion mit den gleichen Eigenschaften aus 1.2	O	Container

nächste Seite

Nummer	Name	Erklärung	Benötigt	Typ
1.2.5	Paragraph	Definiert einen Absatz	O	Container
1.2.6	Table	Definiert einen Tabelle	O	Container
1.2.6.1	border	Legt fest, ob eine Tabelle einen Rahmen hat	O	Integer
1.2.6.2	width	Legt fest, ob eine Tabelle einen Rahmen hat	O	Decimal
1.2.6.2	Column	Konfigurerit die Spalten	O	Container
...				
1.2.8	MMO	Gibt ein Mediaobjekt an	O	Container
1.2.8.1	href	Verweist auf die zugehörige Date	O	String
...				

Tabelle C.2: Attribute und Elemente des Kursmodells

Nummer	Name	Erklärung	Benötigt	Typ
1	Layout	Container für Master, Properties und Map-Elemente	B	Container
2	Master	Container für Dokumentattribute	B	Container
2.1	Page	Container für Seitenattribute	B	Container
2.1.1	width	Definiert die Seitenbreite	B	Dezimal
2.1.2	height	Definiert die Seitenhöhe	B	Dezimal
2.2	Body	Container für Seitenausrichtung	B	Container
2.2.1	marginleft	Linker Rand	B	Dezimal
2.2.2	marginright	Rechter Rand	B	Dezimal
2.2.3	margintop	Oberer Rand	B	Dezimal
2.2.4	marginbottum	Unterer Rand	B	Dezimal
2.2.5	columns	Anzahl der Spalten	B	Integer
2.3	Header	Container für die Kopfzeile	O	Container
2.3.1	width	Breite	O	Integer
2.3.2	height	Höhe	O	Integer
2.3.3	margintop	Abstand zum oberen Rand	O	Integer
2.4	Footer	Container für die Fußzeile	O	Container
2.4.1	width	Breite	O	Integer
2.4.2	height	Höhe	O	Integer
2.4.3	marginbottum	Abstand zum unteren Rand	O	Integer

nächste Seite

Nummer	Name	Erklärung	Benötigt	Typ
3	Styles	Container für Styles	B	Container
3.1	Style	Definiert einen Style	O	Container
3.2	identifier	Identifizierung	O	String
3.3	Property	Eigenschaften	O	Container
3.3.1	type	Layouttyp	O	String
3.3.2	identifier	Identifizierung	O	String
3.3.3	desc	Beschreibung	O	String
4	Map	Container	O	Container
4.1	Reference	Definiert das Mapping	O	Container
4.1.1	indentifierRef	Referenzierte ID	O	String
4.1.2	resourceRef	Referenzierte Ressource	O	String

Tabelle C.3: Attribute und Elemente des Layouts

Index

ADL, 19, 20, 53
AICC, 19, 28
Andendung, 101
ANSI, 20
ARIADNE, 19, 30, 189
Asset, 21, 24
Atom, 35
Authorware, 60
Autorenwerkzeuge, 58

Baustein, 72
Behaviorismus, 12, 16
BrickML, 83

CAM, 21
CBT, 11, 28
CMI, 28
Content Aggregation Model, 21
Content Packaging, 21
CP, 25

DazzlerMax, 61
DocBook, 48, 132
DTD, 43, 45, 46, 52

E-Learning, 11
EML, 53

Formatentwickler/-in, 106
Formatierbaren Bausteine, 124
Formatierbarer Baustein, 77, 84, 86–90, 92, 95, 97, 189
Formatierbarer Kurs, 87, 90, 91, 93, 95, 100, 189
Framework, 138
FTP, 27

Grafische Minimierung, 153
Granularität, 37

Heuristische Lernmodell, 17
HTML, 27, 55, 58
Huffman-Kodierung, 166

IEEE, 31
IEEE LTSC, 19
IMS, 19, 25, 53
IMS Learning Design Information Model, 55
IMS Question and Test, 25
ISO, 31
ITO, 40

Java, 47
JDOM, 47

Kognitivismus, 12, 17
Komplexe Zahlen, 149
Konstruktivismus, 12, 17
KPS, 28
Kurs, 73
Kurs-Framework, 140

Latex, 134
Layout, 105, 107
Layoutentwickler/-in, 106
Layouttyp, 141
Layouttypen, 108
LCMS, 27
Learning Object, 31
LEGO, 35
Lernobjekte, 19
Lernplattformen, 56
LMML, 51, 63
LMS, 19, 21, 56, 58
Loader, 95

LOM, 25, 27
Lyssa, 134
Lyssa-Designer, 140

Mathkit, 3, 5
MathML, 49, 135
Metadaten, 19, 21, 27, 34
Metapher, 34
Multimedia, 10

OMDoc, 49
OpenMath, 49
OpenOffice, 40, 131

PaKMaS, 63
PIF, 22
Prozessor, 101

Rollen, 106

schema, 101
SCO, 24
SCORM, 20
SGML, 43
Simplified DocBook, 133
Slicing-Book-Technologie, 39
staroffice, 131

Task, 101
Technische Informatik, 149, 150
Telemedien, 11
tex4ht, 135
toolbook, 60
Transformationspaket, 101
Transformator, 46, 101–103, 105

W3C, 43
WBT, 11
Wechselstromschaltungen, 155
WWW, 11

Xalan, 136
XML, 42, 43
XML-Schemata, 46
XSD, 46, 52
XSL, 44, 47, 164

Literaturverzeichnis

[ADL04] Sharable Content Object Reference Model (SCORM). http://www.adlnet.org/ (viewed 27.11.2004), 2004.

[AIC03] Aviation industry computer-based training committee (aicc). http://www.aicc.org/ (viewed 02.03.2003), 2003.

[Ari05] Alliance of Remote Instructional Authoring and Distribution Networks for Europe. http://www.ariadne-eu.org/ (viewed 29.10.2005), 2005.

[Balzert00] Helmut Balzert: *Lehrbuch der Software-Technik - 2. Auflage*. Spektrum Akademischer Verlag, Heidelberg, Berlin, ISBN: 3-8274-0480-0, 2000.

[Bauch03] M. Bauch and L. Unger: Interactive mathematics with math-kit - distance learning versus face-to-face learning. In: *21st ICDE World Conference on Open Learning & Distance Education*, Hong Kong, 2003.

[Bauch04a] Manfred J. Bauch, S. Hartlieb and Luise Unger: I - You - We: Teaching Linear Algebra the constructivist way. In: *Proceedings of ED-MEDIA 2004–World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Lugano, Switzerland, January 2004.

[Bauch04b] Manfred J. Bauch, Bianca Thiere and Luise Unger: Visualizing mathematics on the web - constructivist approaches in open and distance education. In: *Proceedings of the 21st ICDE World Conference on Open Learning and Distance Education*, Hong Kong, January 2004.

[Baudry02a] Andreas Baudry, Michael Bungenstock and Bärbel Mertsching: Architecture of an E-Learning System with Embedded Authoring Support. In: Margaret Driscoli and Thomas C. Reeves (eds.): *E-LEARN 2002 - World Conference on Educational in Corporate, World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education (E-LEARN 2002)*, Montreal, Canada, January 2002. pp. 110 – 116.

[Baudry02b] Andreas Baudry, Michael Bungenstock and Bärbel Mertsching: Ein multimediales Rahmenwerk für die Mathematiklehre nach der Baukastenmetapher. In: Klaus P. Jantke, Wolfgang S. Wittig and Jörg Herrmann (eds.): *Von e-Learning bis e-Payment. Das Internet als sicherer Marktplatz (LIT '02)*. Infix, 2002, pp. 300–307.

[Baudry03] Andreas Baudry, Michael Bungenstock and Bärbel Mertsching: Nyx - A Tool for Generating Standard Compatible E-Learning Courses with Consistent and Adaptable Presentation. In: *The IASTED International Conference on Computers and Advanced Technology in Education (CATE 2003), ISBN: 0-88986-361-X*, Rhodes, Greece, June 2003. IASTED, pp. 265 – 269.

[Baudry04a] Andreas Baudry, Michael Bungenstock and Bärbel Mertsching: Administration and Development of Modular Learning Units with the Construction Kit Server. In: *ED-MEDIA 2004*, P.O. Box 3728, Norfolk, VA 23514-3728, June 2004. Ass. for the Advancement of Computing in Education, pp. 783 – 790.

[Baudry04b] Andreas Baudry, Michael Bungenstock and Bärbel Mertsching: Reusing Document Formats for Modular Course Development. In: *IASTED International Conference on WEB-BASED EDUCATION (WBE 2004), ISBN: 0-88986-377-6*, Innsbruck, February 2004. ACTA Press, pp. 535 – 537.

[Baudry05] A. Baudry, M. Bungenstock and B. Mertsching: An e-learning system for standard compatible and uniform course development. In: *International Journal on E-Learning (IJEL) Corporate, Government, Healthcare, & Higher Education.*

[Baumgartner95] P. Baumgartner: *Information und Lernen mit Multimedia*, chapter Didaktische Anforderungen an (multimediale) Lernsoftware. BELTZ Psychologie Verlags Union, 1995.

[Baumgartner97] P. Baumgartner and S. Payr. Erfinden lernen. Konstruktivismus und Kognitionswissenschaft. Kulturelle Wurzeln und Ergebnisse. Zu Ehren Heinz von Foersters, 1997. Wien-New York: Springer.

[Baumgartner99] P. Baumgartner and S. Payr: *Lernen mit Software.* Studienverlag GmbH, ISBN: 3706514443, 1999.

[Baumgartner02] Peter Baumgartner, Hartmut Häfele and Kornelia Maier-Häfele: E-Learning: Fachbegriffe, didaktische und technische Grundlagen. Technical report, bm:bwk, 2002. Sonderheft e-Learning.

[Belqamsi02] Youssef Belqamsi: Using XML in Elearning Courses. In: Margaret Driscoli and Thomas C. Reeves (eds.): *E-LEARN 2002–World*

Conference on Educational in Corporate, World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education, 2002, pp. 1183 – 1186.

[Bergstrom01] Bergstrom: CMI Guidelines for Interoperability AICC. Technical Report Revision 3.5 release 2, Alliance of Remote Instructional Authoring and Distribution Networks for Europe, AICC CMI Subcommittee, 2001.

[Bla03] The Blackboard e-Education. http://www.blackboard.com/ (viewed 23.01.2003), 2003.

[Bohl02] Oliver Bohl, Jörg Schellhhase and Udo Winand: A Critical Discussion of Standards for Web-based Learning. In: Margaret Driscoli and Thomas C. Reeves (eds.): *E-LEARN 2002–World Conference on Educational in Corporate, World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education*, 2002, pp. 850–855.

[Brusilovsky98] P. Brusilovsky, J. Eklund and Schwarz: Web-based education for all: A tool for developing adaptive courseware. In: *Computer Networks and ISDN Systems*, pp. 291–300.

[Brusilovsky01] P. Brusilovsky: adaptive hypermedia. In: *User Modeling and User Adapted Interaction, The Journal of Personalization Research.*

[Bungenstock02] Michael Bungenstock, Andreas Baudry and Bärbel Mertsching: The Construction Kit Metaphor for a Software Engineering Design of an E-Learning System. In: Philip Barker and Samuel Rebelsky (eds.): *Proc. ED-MEDIA 2002 - World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Denver, Colorado, USA, January 2002. pp. 216–217.

[Bungenstock03a] Michael Bungenstock, Andreas Baudry and Bärbel Mertsching: Datenaustausch zwischen Lyssa und Learning Management Systemen. In: *Von e-Learning bis e-Payment. Das Internet als sicherer Marktplatz (LIT '03)*, Leipzig, September 2003.

[Bungenstock03b] Michael Bungenstock, Andreas Baudry and Bärbel Mertsching: Lyssa - An Authoring System Complying with E-Learning Standards. In: *ED-MEDIA 2003 - World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Honolulu, Hawaii, USA, June 2003. pp. 502 – 508.

[Bungenstock04] Michael Bungenstock, Andreas Baudry and Bärbel Mertsching: Design of a Common API for Learning Objects. In: *7th IASTED International Conference on Computers and Advanced Technology in*

Education (CATE 2004), August 16-18, 2004, Kauai, Hawaii, USA, ISBN: 0-88986-430-6. IASTED, August 2004, pp. 345 – 350.

[Caprotti02] O. Caprotti, D. P. Carlisle and A. M. Cohen. The OpenMath Standard, the openmath esprit consortium. http://www.openmath.org/ (viewed 29.10.2005), October 2002. 1.1b.

[Chappell03] David A. Chappell and Tyler Jewell: *Java Web Services*. O'Reilly, 2003. ISBN: 3897212846.

[Consortium05] World Wide Web Consortium. Synchronized Multimedia. http://www.w3.org/AudioVideo/ (viewed 29.10.2005), 2005.

[Cuthbert02] Alex Cuthbert and Frances Himes: *Creating Learning Objects with Macromedia Dreamweaver MX*. Macromedia white paper, 2002.

[Dahn01] I. Dahn: Slicing book technology - providing online support for textbooks. In: *ICDE*, 2001.

[Dahn02] Ingo Dahn and Gerhard Schwabe: Personalizing textbooks with slicing technologies - concept, tools, architecture, collaboration use. In: *35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, 2002.

[Dahn05] Ingo Dahn, Michael Armbruster, Ulrich Furbach and Gerhard Schwabe. Slicing Books - The Authors' Perspective, 2005. `http://citeseer.ist.psu.edu/487947.html(viewed29.10.2005)`.

[Daniel98] John Daniel and Russell Edgerton: *Mega-Universities and Knowledge Media: Technology Strategies for Higher Education*. Kogan Page Ltd, 1998.

[Downes00] S. Downes. Learning Objects. Leaders in Learning 2000, May 5 2000. University of Alberta.

[Downes04] Stephen Downes. The Learning Marketplace Meaning, Metadata and Content Syndication in the Learning Object Economy. http://www.downes.ca/, Januar 2004. Moncton, New Brunswick.

[Dreyfus86] H. L. Dreyfus, T. Anthanasiou and S. E. Dreyfus: *Mind Over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*. The Free Press, ISBN: 0743205510, 1986.

[Duval02] E. Duval: Learning Technology Standardization: Too Many? Too Few? In: *Workshop: Standardisierung im eLearning*, Universität Frankfurt/Main, April 2002.

[Edelmann96] W. Edelmann: *Lernpsychologie 5., vollst. überarbeitete Auflage.* Beltz Psychologie-Verlags-Union, Weinheim, Basel, 1996.

[Electrical02] Institute of Electrical and Electronics Engineers Inc. Draft standard for learning object metadata, July 2002.

[Engesser93] Hermann Engesser (ed.): *Duden Informatik.* Dudenverlag, 1993.

[Farinetti00] L. Farinetti, F. Bota and A. Rarau. An authoring tool for building flexible on-line courses using XML, 2000. XML Europe 2000 Conference, Paris.

[Fielding99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. http://www.w3.org/Protocols/rfc2616/rfc2616.html (viewed 12.12.2002), Juni 1999.

[Finsterle02] Lutz Finsterle and Martin Rotard: Mit konventionellen Autorensystemen zum E-Learning Portal. In: Klaus P. Jantke, Wolfgang S. Wittig and Jörg Herrmann (eds.): *Von e-Learning bis e-Payment. Das Internet als sicherer Marktplatz (LIT '02).* Infix, 2002, pp. 111–121.

[Fowler99] Martin Fowler: *Refactoring.* Addison-Wesley Professional, 1999.

[Freitag02a] Burkhard Freitag, Christian Süß and Claus Dziarstek: Adaptation und Wiederverwendung von XML-basiertem eLearning-Content. In: Sigrid E. Schubert, Bernd Reusch and Norbert Jesse (eds.): *Informatik bewegt: Informatik 2002 - 32. Jahrestagung der GI*, 2002. LNI 19 GI.

[Freitag02b] Burkhard Freitag, Christian Süß and Claus Dziarstek: LMML-Eine XML-Sprachfamilie für eLearning Content. In: Sigrid E. Schubert, Bernd Reusch and Norbert Jesse (eds.): *Informatik bewegt: Informatik 2002 - 32. Jahrestagung der GI*, 2002. LNI 19 GI.

[Frerich00] Alwin Frerich and Günter Gerharz: *Lernen mit Neuen Medien 2000 Software-Ratgeber für die Sekundarstufe I/II.* DruckVerlag Kettler GmbH, Landesinstitut für Schule und Weiterbildung, ISBN: 3-8165-1792-7, 2000.

[Fünfstück00] F. Fünfstück, R. Liskowsky and K. Meißner: Softwarewerkzeuge zur Entwicklung multimedialer Anwendungen. Eine Übersicht. In: *Informatik Spektrum*, 23(1), pp. 11 – 25.

[Gamma95] Erich Gamma et al.: *Design Patterns - Elements of Reusable Object-Oriented Software.* Addison Wesley, Reading, Massachusetts, 1995.

[Geo05] Geonext - dynamische mathematiksoftware. http://geonext.uni-bayreuth.de/ (viewed 29.10.2005), 2005.

[Gersdorf02] Ruben Gersdorf, Berit Jungmann and Eric Schoop: Chancen und Herausfordeungen bei der Abbildung didaktischer Anforderungen mit XML. In: Klaus P. Jantke, Wolfgang S. Wittig and Jörg Herrmann (eds.): *Von e-Learning bis e-Payment. Das Internet als sicherer Marktplatz (LIT '02)*. Infix, 2002, pp. 339–346.

[Goland99] Y. Goland, E. Whitehead, A. Faizi and D. Jensen: *HTTP Extensions for Distributed Authoring – WEBDAV, RFC 2518*. Network Working Group, http://www.webdav.org/ (viewed 29.10.2005), 1999.

[Goldfarb90] Charles F. Goldfarb: *The SGML Handbook*. Oxford University Press, 1990.

[Goosen99] Goosen, Rahtz, Gurari, Moore and Sutor: *The LaTEX Web Companion*. Addison-Wesley, 1999.

[Griffel98] Frank Griffel: *Componentware: Konzepte und Techniken eines Softwareparadigmas*. dpunkt-Verlag, Heidelberg, 1998.

[Gurari00] E. Gurari and R. Rahtz: From LaTEX to MathML and Back with TEX4ht and PassiveTEX. In: *The first MathML International Conference*, Urbana-Champaign, Illinois 2000, pp. 76–82.

[Hansch02] Matthias Hansch, Stefan Kuhlins and Martin Schader: XML-Schema. In: *Informatik Spektrum*, 25(5), 2002, pp. 363–366.

[Harold02a] Elliotte Rusty Harold: *Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and TrAX*. Addison-Wesley Pub Co, ISBN: 0201771861, 2002.

[Harold02b] Elliotte Rusty Harold: *The XML Bible, 2nd Edition*. John Wiley & Sons; 2nd Book and CD-ROM edition, ISBN: 0764547607, 2002.

[Heafele05] Hartmut Heafele. Autorenwerkzeuge für Learning Content. http://virtual-learning.qualifizierung.com/ (viewed 29.10.2005), 2005.

[Heng03] Simon Heng. Learning objects - where next? 3rd Annual CM316 Conference on Multimedia Systems, 2003.

[Hodgins02] Wayne Hodgins: The future of learning objects. In: *Proceedings of the 2002 eTEE Conference*, August 2002, pp. 76–82.

[Ignatiadis03] Michael Ignatiadis. Learning Objects - does size matter? 3rd Annual CM316 Conference on Multimedia Systems, 2003.

[ILI04] Integriertes Lern-, Informations- und Arbeitskooperationssystem. http://www.ilias.uni-koeln.de/ios/index.html (viewed 25.11.2004), 2004.

[IMS03a] IMS Content Packaging Information Model, Version 1.1.3 Final Specification. http://www.imsglobal.org/ (viewed 29.10.2005), Juni 2003.

[IMS03b] IMS Digital Repositories Interoperability - Core Functions Information Model, Version 1.0 Final Specification, 2003.

[IMS03c] IMS Learning Design Information Model Version 1.0 Final Specification. http://www.imsglobal.org/learningdesign/ (viewed 29.10.2005), 2003.

[IMS03d] IMS Learning Resource Meta-Data Information Model, Version 1.2.1 Final Specification. http://www.imsglobal.org/ (viewed 02.03.2003), 2003.

[IMS05a] IMS Question and Test Interoperability Addendum, Version 1.2.1 Final Specification. http://www.imsglobal.org/ (viewed 29.10.2005), 2005.

[IMS05b] Instructinal Management Systems Project (IMS) Homepage. http://www.imsglobal.org/ (viewed 29.10.2005), 2005.

[JDO05] JDOM-Project. http://www.jdom.org (viewed 29.10.2005), 2005.

[JSP05] Java server pages, the apache software foundation. http://jakarta.apache.org/ (viewed 29.10.2005), 2005.

[Kerres98] M. Kerres: *Multimediale und telemediale Lernumgebungen - Konzeption und Entwicklung.* Oldenbourg Verlag, ISBN: 3-486-24539-2, 1998.

[Klamma03] R. Klamma, M. Spaniol and M. Jarke: Digital media knowledge management with MPEG-7. In: *The Twelfth International World Wide Web Conference (WWW 2003)*, Budapest, Hungary, May 2003.

[Kohlhase00] Michael Kohlhase: OMDoc: An Infrastructure for OpenMath Content Dictionary Information. In: *Bulletin of the ACM Special Interest Group for Algorithmic Mathematics SIGSAM*, 2000.

[Kohlhase02] Michael Kohlhase: *OMDoc: An open Markup Format for Mathematical Documents (Version 1.1).* Carnegie Mellon University, January 2002.

[Koper01] Rob Koper. Modeling Units of Study from a pedagogical perspective: the pedagogical meta-model behind EML, June 2001. Open University of the Netherlands.

[Koper04] Rob Koper and Jocelyn Manderveld: Educational Modelling Language: modelling reusable, interoperable, rich and personalised units of learning. In: *British Journal of Educational Technology.*

[LON05] The learning online network with capa. http://www.lon-capa.org/ (viewed 29.10.2005), 2005.

[Lytras] Miltiadis D. Lytras and Athanasia Pouloudi. E-Learning: Just a waste of time. http://citeseer.ist.psu.edu/lytras01elearning.html (viewed 29.10.2005).

[Mac05] Macromedia Authorware 6. http://www.macromedia.com (viewed 29.10.2005), 2005.

[Mat05a] Ein multimedialer Baukasten für die Mathematik-Ausbildung im Grundstudium. http://www.math-kit.de (viewed 29.10.2005), 2005.

[Mat05b] W3C's Math Home Page, World Wide Web Consortium. http://www.w3.org/Math/ (viewed 29.10.2005), 2005.

[Melis03] Erica Melis, Georgi Goguadze, Paul Libbrecht and Carsten Ullrich: Wissensmodellierung und -nutzung in ActiveMath. In: *KI - Kuenstliche Intelligenz*, 1/03, pp. 12–17. http://www.kuenstliche-intelligenz.de/Artikel/WissensmodellierungundnutzunginActiveMa.htm (viewed 12.05.2003).

[Mertsching01] Bärbel Mertsching. Material zur Vorlesung Systemtheorie. Universität Hamburg, Fachbereich Informatik, Arbeitsgruppe IMA, unveröffentlichtes Skript, 2001.

[Mertsching02] Bärbel Mertsching. Material zur Vorlesung Technische Informatik T1/T2. Universität Hamburg, Fachbereich Informatik, Arbeitsgruppe IMA, unveröffentlichtes Skript, 2002.

[Meyer97] Bertrand Meyer: *Object-Oriented Software Construction - SECOND EDITION.* Prentice Hall PTR, Upper Saddle River, ISBN: 0-13-629155-4, 1997.

[National Defence02] Canadian Department of National Defence. SCORM Dynamic Appearance Model, White Paper 1.0, 2002.

[Negroponte96] Nicholas Negroponte: *Being Digital.* Vintage Books USA, 1996.

[Net05] TML Language Specification. http://www.ilrt.bris.ac.uk/netquest/about/lang/ (viewed 29.10.2005), 2005.

[Niegemann03] Helmut M. Niegemann, Silvia Hessel, Dirk Hochscheid-Mauel Kristina Aslanski, Markus Deimann and Gunther Kreuzberger: *Kompendium E-Learning*. Springer Verlag, 2003.

[OOS02] OpenOffice.org XML File Format, Technical Reference Manual, Sun Microsystems. http://xml.openoffice.org/xml_specification.pdf, December 2002.

[Pawlowski01] Jan Martin Pawlowski: *Das Essener-Lern-Modell (ELM): Ein Vorgehensmodell zur Entwicklung computerunterstützter Lernumgebungen*. PhD thesis, Essen, Universität Essen, Fachbereich Wirtschaftswissenschaften (BWL, VWL, Rechtswissenschaft und Wirtschaftsinformatik), 2001. http://deposit.ddb.de/cgi-bin/dokserv?idn=963514113.

[Pawlowski02] Jan M. Pawlowski: Reusable Models of Pedagogical Concepts - a Framework for Pedagogical and Content Design. In: *Proc. of ED-MEDIA 2002, World Conference on Educational Multimedia, Hypermedia & Telecommunications*. Philip Barker AND Samuel Rebelsky, 2002.

[Pomberger96] Gustav Pomberger and Günther Blaschek: *Software Engineering*, vol. 2. Hanser, 1996.

[Ray01] Erik T. Ray: *Einführung in XML*, vol. 1. Auflage. OReilly, 2001.

[Reglin03] Thomas Reglin and Eckart Severing: *eLearning für die betriebliche Praxis*. No. 30 in ISBN 3-7639-3112-0. W. Bertelsmann Verlag, 2003.

[Rieser00] Nicole Rieser (ed.): *Der Brockhaus von A-Z*, vol. 2. Weltbild Verlag GmbH, ISBN: 3-7653-3642-4, 2000.

[Roisin98] C. Roisin: Authoring Structured Multimedia documents. In: *25th Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'98*. Springer, 1998, pp. 222 – 239.

[Rumbaugh91] James Rumbaugh: *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[Saddik01a] Abdulmotaleb El Saddik: *Interactive Multimedia Learning. Shared Reusable Visualization-based Modules*. Springer-Verlag Berlin Heidelberg, ISBN: 3540419306, 2001.

[Saddik01b] Abdulmotaleb El Saddik, Stephan Fischer and Ralf Steinmetz: Reusable Multimedia Content in Web-based learning Systems. In: *IEEE Multimedia*, July 2001, pp. 30–38.

[Saddik02] Abdulmotaleb El Saddik: E-Learning Standards: The Dawn has broken. In: Philip Barker and Samuel Rebelsky (eds.): *Proc. ED-MEDIA 2002–World Conference on Educational Multimedia, Hypermedia & Telecommunications*, 2002.

[Schulmeister81] R. Schulmeister: Lerntheorien - Lernprozesse, r8366. Technical report, Interdiszipliäres Zentrum für Hochschuldidaktik der Universität Hamburg, Juni 1981.

[Schulmeister96] R. Schulmeister: *Grundlagen hypermedialer Lernsysteme: Theorie - Didaktik - Design*, vol. 1. Addison-Wesley, ISBN: 3-89319-923-3, 1996.

[Schulmeister03] R. Schulmeister: *Lernplattformen für das virtuelle Lernen.* Oldenbourg Wissenschaftsverlag, 2003.

[Seufert02] Sabine Seufert and Peter Mayr: *Fachlexikon e-lerning. Wegweiser durch das e-Vokabular.* managerSeminare, ISBN: 3931488640, http://www.managerseminare.de/msemi/1412896/frontend/ elexikon_liste.html?kat=2000 (viewed 29.10.2005), 2002.

[Shen02] Zhongnan Shen, Yuanchun Shi and Guangyou Xu: A Learning Resource Metadata Management System Based on LOM Specification. In: *The 7th International Working Group on Computer Supported Cooperative Work in Design (CSCWD2002)*, Rio de Janeiro, 2002.

[Sim05] The Simplified DocBook Document Type, Committee Specification V1.1. http://www.docbook.org/specs/cs-docbook-simple-1.1.html, (viewed 29.10.2005), 2005.

[Simon02] Bernd Simon: *E-Learning an Hochschulen - Gestaltungsräume und Erfolgsfaktoren von Wissensmedien.* Josef Eul Verlag, 2002.

[Skinner78] B. Skinner: *Was ist Behaviorismus?* Rohwolt, Reinbek bei Hamburg, 1978.

[Snell02] James Snell, Doug Tidwell and Pavel Kulchenko: *Webservice-Programmierung mit SOAP.* O'Reilly, ISBN: 3897211599, 2002.

[Str05] Struts, The Apache Software Foundation. http://struts.apache.org/ (viewed 29.10.2005), 2005.

[Süß99] Christian Süß, Burkhard Freitag and Peter Brössler: Metamodeling for Web-Based Teachware Managment. In: P.P. Chen, D.W. Embley, J. Kouloumdijan, S.W. Liddle and J.F. Roddick (eds.): *Advances in Conceptual Modeling. ER'99 Workshop on the World-Wide Web and Conceptual Modeling, Paris, France, Nov. 1999*, vol. 1727 of *LNCS*, Berlin, 1999. Springer-Verlag, pp. 360–373.

[Süß00] Christian Süß and Burkhard Freitag: Entwicklung und Nutzung von Teachware: Das Passauer Knowledge Management System (PaK-MaS). In: *Computerunterstütztes Lernen*, 2000. Oldenbourg-Verlag, München.

[Szillat95] Horst Szillat: *SGML Eine Proktische Einführung*, vol. 1. International Thomson Publishing GmbH, ISBN: 3-929821-75-3, 1995.

[Szyperski97] Clemens Szyperski: *Component Software - Beyond Object-Oriented Programming.* Addison Wesley, Reading, Massachusetts, 0-201-17888-5, 1997.

[Sünneli04] Yildiz Sünneli and Nurdan Turan: Entwicklung einer Software-Komponente zur Integration bestehender Lehr- und Lernmaterialien und deren Metadaten in das math-kit-System. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, May 2004.

[Taylor96] David A. Taylor: *Objektorientierte Technologien, Ein Leitfaden für Manager.* Addison-Wesley, 1996.

[Thissen97] F. Thissen: Das Lernen neu erfinden, Grundlagen einer konstruktivistischen Multimedia-Didaktik. In: U. Beck and W. Sommer (eds.): *LearnTec 97, Europäischer Kongress für Bildungstechnologie und betriebliche Bildung*, Karlsruhe, 1997. pp. S. 69–79.

[Thissen99a] F. Thissen: Im Informationsreichtum verhungern? In: U. Beck and W. Sommer (eds.): *LearnTec 99*, Karlsruhe, 1999. pp. S. 725–730.

[Thissen99b] Frank Thissen: Lerntheorien und ihre Umsetzung in multimedialen Lernprogrammen - Analyse und Bewertung. In: *BIBB Multimedia Guide Berufsbildung.*

[Unger04] L. Unger, M. J. Bauch, A. Baudry, M. Bungenstock, B. Mertsching, G. Oevel, K. Padberg and B. Thiere: Ein multimedialer Bauksaten für die Mathematikausbildung im Grundstudium. In: *Softwaretechnik-Trends, Gesellschaft für Informatik e.V.*, Band 24(Heft 1), 2004, pp. 62–71.

[Valerius01] Marianne Valerius and Anna Simon: Slicing Book Technology eine neue Technik für eine neue Lehre? Technical report, Fachberichte Informatik 8/2001, Universitaet Koblenz-Landau, 2001. http://www.uni-koblenz.de/fb4/publikationen/gelbereihe/RR-8-2001/ (viewed 29.10.2005).

[Valk80] Rüdiger Valk: *Rechensysteme*. Springer, 1980.

[Verbert04] Katrien Verbert and Erik Duval: Towards a Global Component Architecture for Learning Objects: A Comparative Analysis of Learning Object Content Models. In: *ED-MEDIA 2004–World Conference on Educational Multimedia, Hypermedia & Telecommunications*, P.O. Box 3728, Norfolk, VA 23514-3728, June 2004. Ass. for the Advancement of Computing in Education, pp. 202 – 208.

[W3C99] W3C. XML Path Language (XPath). http://www.w3.org/TR/xpath (viewed 29.10.2005), 1999.

[W3C05a] XML Schema. http://www.w3.org/XML/Schema (viewed 29.10.2005), 2005.

[W3C05b] XML Syntax for XQUERY 1.0. http://www.w3.org/TR/xqueryx (viewed 29.10.2005), 2005.

[W3C05c] XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery/ (viewed 29.10.2005), 2005. W3C Working Draft 12 November 2003.

[Walsh02] Norman Walsh and Leonard Muellner: *DocBook: The Definitive Guide*. O'Reilly & Associates, Inc, ISBN: 1-56592-580-7, 2002.

[WEB05] WebCT Campus Edition - Course Management System, WebCT Inc. http://www.webct.com/ (viewed 29.10.2005), 2005.

[Weitl04] Franz Weitl, Rudolf Kammerl and Monica Göstl: Context Aware Reuse of Learning Resources. In: *ED-MEDIA 2004–World Conference on Educational Multimedia, Hypermedia & Telecommunications*, P.O. Box 3728, Norfolk, VA 23514-3728, June 2004. Ass. for the Advancement of Computing in Education, pp. 202 – 208.

[Wiley99] David Wiley. The Post-LEGO Learning Object. http://wiley.ed.usu.edu/docs/post-lego.pdf (viewed 29.10.2005), November 5 1999.

[Wiley00a] D. A. Wiley. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. D. Wiley, The Instructional Use of Learning Objects, 2000. Bloomington: Association for Educational Communications and Technology.

[Wiley00b] D. A. Wiley: *Learning object design and sequencing theory.* PhD thesis, Brigham Young University, 2000.

[Wiley04] Wiley, Gibbons and Recker. A reformulation of the issue of learning object granularity and its implications for the design of learning objects. http://www.reusability.org/granularity.pdf (viewed 29.10.2005), 2004.

[Wilson99] M. Wilson: Standards and Reference Architectures: Their Purpose and Establishment. In: *In Proceedings of the AISB'99 Workshop on Reference Architectures and Data Standards for NLP*, Edinburgh, 1999.

[Wollowski02] Michael Wollowski: XML Based Course Websites. In: Margaret Driscoli and Thomas C. Reeves (eds.): *E-LEARN 2002–World Conference on Educational in Corporate, World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education*, 2002, pp. 1043–1048.

[Xer05] Xerces, XML - parsers in Java und C++, Apache XML Project. http://xml.apache.org/xerces2-j/index.html (viewed 29.10.2005), 2005.

[XSL99] XSL Transformations (XSLT) Version 1.0. http://www.w3.org/TR/xslt (viewed 16.04.2002), 1999.

[Züllighoven98] Heinz Züllighoven: *Das objektorientierte Konstruktionshandbuch.* dpunkt-Verlag, Heinz Zuellighoven.-Heidelberg : dpunkt-Verl., ISBN: 3-932588-05-3, 1998.