

Lernmodule mit quelltextbasierter Interaktivität

Dissertation

der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Johannes Görke

aus Karlsruhe

Tübingen
2007

Tag der mündlichen Qualifikation:

11.07.2007

Dekan:

Prof. Dr. Michael Diehl

1. Berichterstatter:

Prof. Dr. Wolfgang Straßer

2. Berichterstatter:

Prof. Dr. Andreas Schilling

Danksagung

Diese Arbeit entstand am Lehrstuhl für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts der Universität Tübingen. Besonders herzlicher Dank gilt meinem Doktorvater Prof. Wolfgang Straßer. Außerdem möchte ich Prof. Andreas Schilling für die Unterstützung danken.

Dr. Frank Hanisch möchte ich sehr herzlich für die enge Zusammenarbeit und die befruchtenden Diskussionen danken, die Quelle von Inspiration waren.

Weiterhin möchte ich all meinen Projektpartnern danken. Hierbei sind zu nennen Prof. Dr. Michael H. W. Hoffmann, Ravinder Beemagani und Dr. Frank Bögelsack von der Universität Ulm. Zu danken ist ebenfalls Prof. Dr. Ertl, Dr. Martin Rotard und Christiane Taras von der Universität Stuttgart.

Außerdem möchte ich mich bei einigen Studenten bedanken. Dies sind Marco Kugel für seine HiWi-Tätigkeit, Christoph de Mattia, Rihab Salih und Stephan Griehl für ihre Diplomarbeiten, Ralf Olleck und Carsten Soemer für ihre Studienarbeiten sowie Tina Egerter für ihre Diplomarbeit.

Gesonderter Dank gilt Prof. Straßer, Prof. Schilling und Prof. Fischer wegen der Ermöglichung und Durchführung einer Evaluierung des vorgestellten Systems.

Weiterhin möchte ich allen Kollegen am WSI/GRIS danken. Besonderer Dank gilt dabei Dr. Peter Biber, Ralf Sonntag, Sven Fleck, Timo Schairer, Robert Kuchar, Johannes Mezger, Dr. Zein Salah, Dr. Ingmar Peter, Dr. Johannes Hirche, Dr. Dirk Staneker und Jörg Edelmann.

Ganz besonderer Dank gilt meiner Frau Christine und meiner Familie.

Inhaltsverzeichnis

Zusammenfassung	vii
Abstract	ix
Abbildungsverzeichnis	ix
Tabellenverzeichnis	xii
1 Einleitung	1
1.1 Techniken	2
1.2 Inhalte und Didaktik	3
1.3 Beiträge dieser Arbeit	4
1.4 Aufbau der Arbeit	5
2 Auszeichnungssprachen im Internet	7
2.1 Die Geschichte von XML und den Vorläufern	7
2.2 Die Funktionsweise von XML	8
2.3 Programmierschnittstellen und Transformation	12
2.4 Wichtige XML-Anwendungen	15
2.4.1 Scalable Vector Graphics	16
2.4.2 eXtensible 3D	18
2.5 XML im Web	19
2.6 Dynamische Inhalte im Web	21
2.7 Speicherung von XML	23

2.8	Trennung von Inhalt und Layout	24
2.9	Neue Entwicklungen im Web	24
2.10	Zusammenfassung und Ausblick	26
3	Lernen	27
3.1	Einleitung	27
3.2	Lerntheorien	27
3.3	Geschichte neuer Techniken zum Lernen	29
3.4	Lehr- und Lernsysteme	31
3.5	Begriffe und Abkürzungen	32
3.6	Kategorisierung von Lernmanagementsystemen	33
3.7	Standards für Lernobjekte	34
3.7.1	Learning Object Metadata (LOM)	35
3.7.2	Sharable Content Object Reference Model (SCORM)	36
3.8	Weitere Standards	38
3.9	Lernobjektsammlungen	39
3.10	Zusammenfassung	40
4	Moderne Courseware	41
4.1	Moderne Courseware	41
4.2	Cocoon	41
4.3	Einbindung mathematischer Formeln	44
4.4	Integration hochgradig interaktiver Inhalte	46
4.5	Transformation	49
4.6	Zusammenfassung und Ausblick	53
5	Ein interaktives Online-Lehrbuch	55
5.1	Einführung	55
5.2	Einordnung	57
5.3	Quelltextbasierte Interaktivität	60
5.4	Motivation für quelltextbasierte Interaktivität	62
5.5	Die Technik der Lernmaterialsammlung	64
5.5.1	Eine Auszeichnungssprache für Inhalte	64
5.5.2	Verwaltung von Inhalten	66
5.5.3	Serverseitiges Übersetzen	68
5.5.4	Versionierung der Quelltexte	69
5.5.5	Ein webbasierter Quelltexteditor	70

5.6	Einsatzszenarien	71
5.7	Evaluierung	76
5.8	Zusammenfassung und Ausblick	79
6	Workflows und ihr Management	81
6.1	Workflowmanagement	81
6.2	Geschichte der Workflowsysteme	82
6.3	Technische Aspekte	84
6.3.1	Content-Managementsystem	85
6.3.2	Ein Workflow-Editor	86
6.3.3	Transformation	87
6.3.4	Aktionen	88
6.3.5	Usecases	88
6.3.6	Integration	89
6.4	Eine Bibliothek für Workflowmanagement	89
6.4.1	Gebräuchliche Elemente einer Bibliothek	90
6.5	Usecase: Peer Reviewing	91
6.6	Zusammenfassung und Ausblick	92
7	Zusammenfassung und Ausblick	95
	Literaturverzeichnis	99
	Index	110

Zusammenfassung

Die Verfügbarkeit von Computern in Haushalten sowie schnelle Internetzugänge zu bezahlbaren Preisen für jedermann treiben die Entwicklung webbasierter Plattformen an. Lehr- und Lerntechnologien, die mittels Kollaborationstechniken schon seit längerem die Zusammenarbeit unter Lernenden fördern, passen sich diesem Trend vermehrt an. Die Weiterentwicklung von Auszeichnungssprachen eröffnet dabei eine Vielzahl an neuen Möglichkeiten, wie Inhalte aufbereitet und dem Anwender präsentiert werden können. Auch die Entstehung vermehrt interaktiver Plattformen im Web eröffnet neue Lernszenarien für Lernende. Die Computergrafik nimmt hier eine Vorreiterstellung ein, da sie maßgeblich an der Entwicklung neuer Dialekte für grafische Auszeichnungssprachen beteiligt ist und sich schon länger mit der Präsentation multimedialer Inhalte im Internet beschäftigt.

Diese Arbeit stellt die Neuerungen im Bereich der Auszeichnungssprachen vor, wobei sie die Auswirkungen auf die Erstellung von Lehr-/Lernplattformen aufzeigt. Hierbei wird dargelegt, dass die zu Grunde liegende Technik nicht nur die Datenhaltung beeinflusst sondern auch die Flexibilität bei der Darstellung der Inhalte in verschiedenen Formaten. Auch die Integration schon bestehender Inhalte ist bei geschickter Umsetzung ohne Schwierigkeiten zu bewältigen.

Unter Beachtung dieser Grundsätze zur Erstellung von Lernplattformen wird ein neuartiges System vorgeschlagen, welches analog zu einem Textbuch Lernmaterialien im Web anbietet, angereichert um interaktive Illustrationen mit eingebetteten Quelltexten. Die Quelltexte dieses Online-Lehrbuchs kann der Anwender individuell modifizieren und anpassen. Diese *quelltextbasierte Interaktivität* wird mittels der Dienste *serverseitiges Übersetzen* und *serverseitige Versionierung* verwirklicht. Dies ist der wesentliche, innovative

Beitrag dieser Arbeit. Ebenfalls werden die didaktischen Szenarien einer solchen Plattform für Unterricht und Selbststudium dargelegt. Die dazu benötigten Dienste zum Übersetzen und Versionieren der Quelltexte werden erläutert. Weitere Komponenten dieses Systems sind ein webbasierter Editor, der die Syntax der Quelltexte farblich markiert, und die Auszeichnungssprache für Autoren, die sie in ihren Editor zur erleichterten Erstellung solcher Lehreinheiten einbinden. Abschließend wird ein grafisches Werkzeug zur halbautomatischen Anpassung von Arbeitsabläufen dieser Plattform vorgestellt.

Abstract

The availability of computers in households and the availability of low-cost fast internet access for everybody pushes the development of web-based platforms. Teaching and learning technologies, which already encourage learners to work together by means of collaboration techniques, follow this trend. The ongoing development of markup languages opens up a whole set of new possibilities of how to prepare and how to present content to users. The raise of interactive web platforms on the web permits new learning scenarios for inquiring learners. Computer graphics has an outstanding position in this process as it is the main source for new dialects of markup languages for graphical content, and as it is dealing for long with the presentation of multimedia content on the internet.

This work presents innovation in the domain of markup languages and their impact on the development of educational systems. We show that the technical basis not only affects data management but also the flexibility of presenting content in different formats. Existing content can be integrated easily by the use of handy mechanisms.

Based on these design principles for educational platforms, we propose a new system that offers web-based textbooks enriched with interactive illustrations and embedded source code. User may individually modify and adapt source code. We detail the involved didactics of this platform for tutorials and self studies. We explain the main services for compiling and versioning source code. We present other components of this newly proposed system like a web-based source code editor with syntax highlighting and a specific markup language for authors that use it with their editor in order to facilitate their work of integrating interactive illustrations. We close with a graphical tool for adapting workflows of the platform semi-automatically.

Abbildungsverzeichnis

2.1	Das XML-Universum nach Michel	9
2.2	Das XHTML-Resultat im Browser	15
2.3	SVG-Primitive, Bezierkurve und Text	18
3.1	Das SCORM Content Aggregation Modell nach SCORM CAM	38
4.1	Verarbeitung eines HTTP-Requests in Cocoon	42
4.2	Ablauf der Cocoon-Pipeline	43
4.3	Die Courseware des LIVE-Projekts	46
4.4	Ablauf in einem Drag-And-Drop-fähigen Forum	48
4.5	Transformation nach HTML oder PDF per XSLT	50
4.6	Klickstatistiken mit SVG	52
4.7	Der Nutzer-Fortschritt als SVG-Chart	52
4.8	Der Fortschritt aller Nutzer als Chart	53
5.1	Beispiel einer interaktiven Illustration	57
5.2	Aufrufablauf bei interaktiven Illustrationen	63
5.3	Inhaltserfassung mit Schema-Unterstützung in JEdit	66
5.4	Datenfluss zwischen Nutzer und Repository	70
5.5	Materialien im interaktiven Online-Lehrbuch	73
5.6	Das Lehrmodul Histogrammeinebnung	74
5.7	Das Lehrmodul Houghtransformation	76
5.8	Interaktive Illustration mit Flash statt Java	77

6.1	Der <i>Hypecycle</i> von Workflows im Vergleich mit dem relationaler Datenbanken	83
6.2	Der Standard-Lebenszyklus eines Dokuments in Apache Lenya.	85
6.3	Der Peer-Review-Workflow, wie er in YAWL modelliert wird.	92

Tabellenverzeichnis

2.1	Neue W3C-Standards in Webbrowsern (Stand 2006)	21
3.1	Vergleich von Lerntheorien von Blumstengel	28
4.1	Vergleich von Cocoon mit JSP/ASP und PHP anhand ausgewählter Kriterien	44
6.1	Testergebnisse der grafischen Workflow-Editoren	87

KAPITEL 1

Einleitung

Bildung zählt heute zum wichtigsten Kapital einer modernen Gesellschaft, welche sich im Weltmarkt nur durch ständige Innovation behaupten kann. Lernen und Fortbildung werden deswegen mehr und mehr zu einer lebenslangen Aufgabe, welche nicht nur in der Verantwortung des Einzelnen liegt, sondern inzwischen auch von Firmen als wichtige Maßnahme zur Erhaltung der Wettbewerbsfähigkeit erkannt wurde. Deswegen wird der Markt für Lernsoftware weiter stark wachsen. Das Bundesministerium für Bildung und Forschung fördert entsprechend die Entwicklung neuer E-Learning-Technologien.

Universitäten, welche seit Humboldts Bildungsreform Forschung und Lehre als von der Gesellschaft übertragenen Auftrag haben, wissen um deren Bedeutung und auch um die Schwierigkeit, umfangreiches Wissen an nachwachsende Generationen weiter zu geben. Da sich der Wissenszuwachs der Menschheit mit der weiteren Spezialisierung der verschiedenen Fachbereiche zunehmend zu beschleunigen scheint, gibt es schon lange Bestrebungen, den Wissenstransfer der dabei gewonnen Erkenntnisse zu verbessern, indem technische Hilfsmittel entwickelt werden, um die Inhalte eingängiger darzulegen, die Verfügbarkeit der Lernmaterialien zu erhöhen und das Wissen besser abrufbar zu gestalten, so dass das Lernen erleichtert wird.

Neben der klassischen Lernsoftware, welche als Programm lokal installiert wird, gewinnt das Internet als Wissensquelle zunehmende Bedeutung. Ein wichtiger Vorteil des Internet liegt in der Aktualität, da die Datenbasis ständig gepflegt werden kann und dem Anwender dann sofort zur Verfügung steht – im Gegensatz zu Updates, welche bei lokal installierter Software erst eingepflegt werden müssen. Im Internet gibt es dabei mehrere Arten, wie Wissen vermittelt wird. Einerseits sind Online-Lexika wie Wikipedia.org zu nennen, welche einen direkten Anlaufpunkt mit eigener Suchfunktion darstellen. Weiterhin gibt es unstrukturierte Seiten, welche man mittels einer Suchmaschine wie Google

sich selbst erschließen und einordnen muss. Und drittens gibt es Repositorien wie Merlot [140] oder die OpenCourseWare des Massachusetts Institute of Technology (MIT) [95], die Kurse, Lektionen oder Lerneinheiten zu vielen verschiedenen Fachgebieten anbieten. Wegen der zunehmenden Verbreitung breitbandiger Internetzugänge im privaten Umfeld verschiebt sich der Entwicklungsfokus auf webbasierte Lernumgebungen.

Die Computergrafik erlebt eine besonders eindrucksvolle Fortentwicklung, welche sich in grafisch sehr realistischen Computerspielen, bei Anwendungen wie Google Earth, das internetbasiert dem Anwender die ganze Erde in ungeahnter Detailschärfe präsentiert, oder bei komplexen und großen Volumendatensätzen, wie sie beispielsweise in der Medizin oder in der Klimaforschung anfallen, manifestiert. Sie ist exemplarisch für ein Forschungsgebiet, bei dem in schnellem Tempo neues Wissen weltweit erforscht wird und anfällt, welches nicht nur im universitären sondern auch im industriellen Umfeld von Bedeutung ist. Der schnelle Transfer neuer Inhalte in die Curricula und Wissensbasen ist also von großer Bedeutung. Der Lehrstuhl für Graphisch-Interaktive Systeme am Wilhelm-Schickard-Institut der Universität Tübingen (WSI/GRIS) bietet hier also ein Umfeld, in dem sowohl Forschung betrieben wird als auch die Lehre mittels neuartiger Techniken verbessert werden kann.

Die Forschung im Bereich der Verbesserung der Lehre und des Lernens ist ein klassisches Querschnittsthema, welches viele universitäre Bereiche berührt. Während natürlich Pädagogen und Psychologen sich schon lange mit dem Phänomen "Lernen" beschäftigen, werden von vielen naturwissenschaftlichen Fachrichtungen, insbesondere von der Informatik, aus zweierlei Gründen ebenfalls in dieser Richtung Anstrengungen unternommen: Erstens besteht durch die schnelle Wissenserneuerung ein immenses Interesse an der schnellen Verfügbarmachung dieser Erkenntnisse, und zweitens verfügt bei den Naturwissenschaften gerade die Informatik mit der rasanten Entwicklung der Software und der Hardware originär über die neuen Techniken, welche neue Arten des Lernens ermöglichen. Die Computergrafik ist unter den Fächern der Informatik besonders ausgezeichnet, was ihre Fähigkeit und ihr Wissen bezüglich multimedialer Inhalte, interaktiver Software mit Mensch-Maschine-Schnittstellen oder grafischer Darstellung am Computer angeht.

1.1 Techniken

Während das Internet eine Fülle neuer Möglichkeiten bietet, schränkt es Nutzer immer auch ein. Zur Darstellung mathematischer Formeln in Webbrowsern war in den Anfangszeiten des Internet nur der Umweg über eine Generierung von Bildern mit den Formeln als Inhalt möglich. Erst mit der Standardisierung von der Auszeichnungssprache für mathematische Formeln (MathML) und einer neuen Generation von Browsern ist es also möglich, naturwissenschaftliche Inhalte so in das *World Wide Web* (WWW) zu stellen, dass die Formeln auch beim Benutzer als Text und nicht als Bild ankommen. Dies hat natürlich verschiedene Auswirkungen auf die Datenhaltung der zu Grunde liegenden Webseite, die Erstellung neuer Inhalte, die Pflege und Wartung der bestehenden

Inhalte und somit die Aktualität der Webseite. Durch den großen Erfolg der *eXtended Markup Language* (XML) als Format für teils unstrukturierten Text müssen also bisherige Techniken überdacht und möglicherweise zugunsten neuer Techniken aufgegeben werden, da zum Beispiel ein Zwischenschritt zur Generierung von Bildern für mathematische Formeln nicht mehr nötig ist.

XML beeinflusst auch andere Bereiche der Publikation von Inhalten im Netz. So ermöglicht die Transformationssprache *eXtended Stylesheet Language* (XSL), selbst eine in XML geschriebene Sprache, die weitgehend problemlose Übersetzung von XML-Inhalten in andere XML-Dialekte wie zum Beispiel *Scalable Vector Graphics* (SVG), mittels derer sich Grafiken einfach erstellen lassen. Auch hierfür können somit Prozesse zur Inhaltserstellung vereinfacht werden. Desgleichen existieren Bibliotheken, welche die automatische Generierung von Dokumenten im Format *Portable Document Format* (PDF) ermöglichen. Aus einer Datenbasis können nach Erstellung verschiedener XSL-Transformationen zur Übersetzung in andere XML-Formate somit eine ganze Schar von Dokumenten vollautomatisch erzeugt werden.

Ein weiterer wichtiger Punkt bei der Vermittlung von Wissen ist Interaktivität. Hochgradig interaktive Lernobjekte, wie sie Hanisch in [68] vorschlägt, können in einer XML-basierten Lernplattform gewinnbringend eingesetzt werden. Dies wird anhand der Beispielanwendung "Forum" erläutert, welches hochgradig interaktive Lernobjekte nach dem Drag-And-Drop-Paradigma [62] [57] integriert.

Bei der Erstellung einer neuartigen Lernplattform mit didaktischen Konzepten zu interaktiven Illustrationen wird heutzutage üblicherweise ein Framework als technische Basis verwendet, da damit eine Reihe benötigter Dienste wie beispielsweise Versionierung, eine Benutzerverwaltung mit einem Rollenkonzept oder Suchfunktionalität zur Verfügung stehen. Dies bedingt jedoch die Anpassung des zu Grunde liegenden Frameworks, sei es ein Dokumentenmanagementsystem oder ein Content-Managementsystem, die mit heutigen Systemen nicht unproblematisch ist. Deswegen werden in dieser Arbeit ebenfalls die Forschungen im Gebiet der Workflow-Managementsysteme nachvollzogen. Gerade bei der neueren Forschung gewonnene Erkenntnisse können verwendet werden, um die Einrichtung und Konfiguration sowie die Wartung einer webbasierten Lernplattform zu vereinfachen, indem grafische Werkzeuge bei der Verwaltung der Workflows in dem System eingesetzt werden.

1.2 Inhalte und Didaktik

Da am Institut für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts der Universität Tübingen schon seit dem Jahr 1995 im Bereich der multimedialen Inhalte und ihrer Präsentation über das Internet geforscht wird, ist inzwischen ein großer Satz an interaktiven Inhalten entstanden. Diese wurden teils erweitert, teils angepasst, damit sie auch in dem neu vorgeschlagenen System mit erweiterten didaktischen Szenarien funktionieren. Es sei hier ausdrücklich erwähnt, dass die Erstellung von Inhalten, seien sie multimedialer Art, reine Schriftstücke, Präsentationen oder Bilder, die mit Abstand teuer-

ste Arbeit ist, da hierfür viel Zeit selbst für eingearbeitete und mit dem Gebiet vertraute Autoren vonnöten ist [124]. Diese Kosten erzwingen, dass Inhalte möglichst wiederverwendet oder zumindest in angepasster Form übernommen werden, was bei entsprechender Qualität der Inhalte ein erster Schritt zur Umsetzung im neuen Medium ist.

Bei der Publikation von Inhalten im Internet soll ein möglichst großer Lernfortschritt erzielt werden. Deswegen werden in dem in dieser Arbeit vorgeschlagenen Online-Lehrbuch (siehe Kapitel 5) verschiedene didaktische Szenarien, wie sie bei [67] und [109] zu finden sind, verwirklicht. Hierbei werden folgende Szenarien umgesetzt: Illustration von Konzepten durch eine Beispielimplementierung, Exploration solcher Illustrationen, das allseits bekannte Mehrfachauswahlszenario (*Multiple Choice*), des weiteren die Programmieraufgabe, die Fehlersuche, der Vergleich von Quelltexten nach verschiedenen Aspekten wie Komplexität, Geschwindigkeit oder Genauigkeit, die Bestimmung der Funktionalität bestimmter Quelltextpassagen, und schließlich die Quelltextbesprechung. Die Umsetzung dieser didaktischen Ansätze findet sich in der vorgeschlagenen Auszeichnungssprache wieder. Ein Großteil der interaktiven Szenarien, welche im Umgang mit interaktiven Illustrationen möglich sind, sind abgedeckt, da die umfangreiche IMS-QT-Spezifikation [11] als Basis für die Szenarien diene.

1.3 Beiträge dieser Arbeit

Diese Arbeit stellt folgende wesentliche Neuerungen vor. Im Rahmen der Arbeit wurde basierend auf dem XML/XSL-Framework Cocoon eine vorlesungsbegeleitende Courseware entwickelt, welche als eine der ersten im Internet die Verwendung von mathematischen Formeln per MathML im Textformat und nicht als generiertes Bild gleichermaßen für die Browserformate Netscape und Internet Explorer ermöglichte [54]. Die Auswirkungen von XML/XSL-basierten Frameworks auf die Publikation von Lernmaterialien im Internet mit den verschiedenen verfügbaren Formaten, in welche mittels XSL gerendert werden kann, wird in Kapitel 4 beschrieben.

Mittels der dabei erarbeiteten Techniken wird das Konzept des Online-Lehrbuchs mit interaktiven Illustrationen als *quelltextbasierte Interaktivität* vorgeschlagen [55] [56]. Dabei wird erstmalig das Konzept solcher Illustrationen erläutert. Damit diese Interaktionen im Rahmen eines Repositoriums von vielen Benutzern in den oben genannten, verschiedenen didaktischen Szenarien genutzt werden können, werden die Dienste *serverseitiges Übersetzen* und *serverseitige Versionierung* eingeführt. Um als Autor neue Inhalte erstellen zu können, wird eine XML-basierte Auszeichnungssprache eingeführt, welche für die verschiedenen didaktischen Szenarien entsprechende Auszeichnungselemente bereit hält. Somit kann ein beliebiger Texteditor wie beispielsweise JEdit, welcher XML-Schemata versteht, verwendet werden, um bequem und übersichtlich Inhalte aufzubereiten. Desgleichen wird ein webbasierter Editor angepasst, so dass Anwender zur Bearbeitung der Illustrationen mit quelltextbasierter Interaktivität ein Werkzeug zur Verfügung haben, welches die Quelltexte mittels farblicher Markierung leichter lesbar darstellt.

Die Mächtigkeit des vorgestellten Ansatzes wird dadurch deutlich, dass keine Einschränkung auf eine bestimmte Programmiersprache stattfindet, sondern prinzipiell beliebige Sprachen verwendet werden können. Die Arbeit konzentriert sich auf Java, aber es wird auch anhand eines Beispiels gezeigt, dass dieser Ansatz ebenso mit Macromedias Flash funktioniert.

Da sich bei der Implementation des System zeigte, dass sich die Anpassung der Abläufe der zu Grunde liegenden Plattform sehr mühsam gestaltete, wird außerdem ein neues halbautomatisches Verfahren vorgeschlagen, wie sich die Workflows grafisch gestalten und dann in das System einpflegen lassen [56] [59].

1.4 Aufbau der Arbeit

Diese Arbeit legt die verwendeten Grundlagen im zweiten und dritten Kapitel dar. Im zweiten Kapitel wird XML mit verschiedenen Dialekten vorgestellt sowie die Transformation von XML mittels XSL. Die dynamische Erzeugung von auf XML basierenden Webseiten wird erläutert und auf die persistente Speicherung der Inhalte eingegangen. Ein weiterer Schwerpunkt liegt auf der Trennung von Inhalt und Layout, ein Paradigma, welches von XML stark gefördert wird. Im dritten Kapitel werden didaktische Grundlagen des Lernens im Internet erläutert und didaktische Szenarien für ein interaktives Online-Lehrbuch vorgeschlagen. Dann werden verschiedene ausgewählte Lehr- und Lernsysteme vorgestellt, sowie eine Kategorisierung dieser System versucht. Die heutzutage verwendeten grundsätzlichen Standards für Lerninhalte werden dargelegt. Anhand einer neu entwickelten Lernsoftware werden im vierten Kapitel die Neuerungen, welche sich mit der Einführung von XML zur Speicherung der Inhalte ergeben, aufgezeigt. Besondere Betonung liegt dabei auf der korrekten Integration von mathematischen Formeln, auf der Integration hochgradig interaktiver Inhalte in ein Diskussionsforum und auf der Transformation der bereitgestellten Inhalte in verschiedene Ausgabeformate. Im fünften Kapitel wird das Konzept des interaktiven Online-Lehrbuchs vorgestellt, welches unter Einsatz der bisher gewonnenen Erkenntnisse für interaktive Illustrationen mit quelltextbasierter Interaktivität verwirklicht wird. Dazu wird eine Auszeichnungssprache definiert, die dafür erforderlichen Dienste *serverseitiges Übersetzen* und *serverseitige Versionierung* werden erläutert und ein webbasierter Quelltexteditor wird eingeführt. Anschließend werden Einsatzszenarien sowie Ergebnisse der bisher vorgenommenen Evaluierungen des Systems dargelegt. Im sechsten Kapitel wird schließlich auf die Bedeutung von Arbeitsabläufen und deren Management bei der Erstellung einer solchen Lehr-/Lernplattform eingegangen. Es wird anhand der verwendeten technischen Basis eine Lösung aufgezeigt, um schneller von einer grafischen Erstellung eines neuen Workflows mittels eines halbautomatischen Ablaufs zu einer um diesen Ablauf erweiterten Lernplattform zu kommen. Dafür verwendete Grundlagen und Techniken sowie eine Bibliothek oft verwendeter Werkzeuge einer Lernplattform werden bereit gestellt. Das siebte Kapitel gibt schließlich eine Zusammenfassung und einen Ausblick auf zukünftige Arbeiten.

Auszeichnungssprachen im Internet

Dieses Kapitel beschreibt die Entstehung der Auszeichnungssprache *eXtended Markup Language* (XML) sowie ihre Bedeutung als Datenaustauschformat. Ein besonderes Augenmerk wird auf die historische Entwicklung von XML mit dem Vorläuferformat SGML gelegt sowie auf die diversen so genannten Anwendungen wie XSL, Docbook, MathML und weitere. Schließlich wird auf Adressierungsarten und Namensräume eingegangen sowie eine Einordnung von XML in die Welt des Web und des Datenaustauschs zwischen Computern vorgenommen.

2.1 Die Geschichte von XML und den Vorläufern

XML geht auf die *Standard Generalized Markup Language* (SGML) zurück und bezeichnet eine allgemeine Metasprache. SGML seinerseits ist ein Nachfolger der *Generalized Markup Language* (GML), die von Goldfarb, Mosher und Lorie Ende der sechziger Jahre bei IBM entworfen wurde. Hauptidee von GML und den Nachfolgern ist die Umsetzung des Konzeptes der generischen Programmierung im Zusammenspiel mit einer formalisierten Dokumentenstruktur mit festgelegten Elementauszeichnungen. IBM selbst verwendete GML erfolgreich für Dokumentationen, die zeitweise zu neunzig Prozent in diesem Format erfolgten [51]. Die Entwicklung von SGML begann im Jahre 1978 auch unter Beteiligung von Goldfarb, und bereits im Jahre 1980 wurde ein erster Arbeitsentwurf (*working draft*) von SGML verabschiedet. 1983 konnte der sechste Entwurf dann als Industriestandard empfohlen werden, und bereits 1986 wurde SGML als ISO-Standard 1986:8879 [77] verabschiedet. Obwohl SGML mit den Anwendungen "elektronisches Manuskript" zum erleichterten Austausch von Manuskripten zwischen Autoren und Verlagen und der rechnergestützten Beschaffungs- und Logistikunterstützung [130] kleinere

Erfolge in den achtziger Jahren und mit der *Hypertext Markup Language* (HTML) einen sehr großen und bis jetzt anhaltenden Erfolg in den neunziger Jahren feiern konnte, wurde das Bedürfnis nach einer verbesserten Version von SGML Anfang der neunziger Jahre immer größer. Hauptproblem von SGML war die Komplexität des Standards - es handelte sich um über 150 Seiten technische Dokumentation. Daraus resultierte eine nur mangelhafte Umsetzung des Standards in Software - es existiert kaum eine vollständige Umsetzung des Standards. 1996 begann eine Arbeitsgruppe des World Wide Web Consortium (W3C) mit der Arbeit an einer vereinfachten SGML-Version. Die Ergebnisse dieser Arbeit wurden 1998 als XML 1.0 verabschiedet.

Nach 1998 wurden vor allem verschiedene Anwendungen von XML entwickelt. Hierbei sind die Namensräume zu nennen, die für eine Unterscheidung von Elementen gleichen Namens sorgen, dann die Schnittstellen Document Object Model (DOM) und Simple API for XML (SAX), um XML-Dokumente zu parsen und zu verarbeiten, und die eXtensible Stylesheet Language (XSL), welche eine regelbasierte Transformation von XML-Dokumenten ermöglicht. Mittels Dokumenttypdefinitionen (DTD) und später XML-Schema können Anwender ihren XML-Dialekt festlegen. Zur genauen Adressierung von Elementen in Dokumenten wurde XPath geschaffen, und zur Verbindung von XML-Dokumenten beziehungsweise Teilen von ihnen wurden XLink [37], XPointer [60] und XInclude [93] entwickelt. Zusätzlich wurde eine große Menge an XML-Dialekten oder XML-Anwendungen entworfen. Für Texte im Allgemeinen existiert Docbook seit 1991 [149]. Für alle naturwissenschaftlichen Fächer gibt es spezielle XML-Schemata wie beispielsweise MathML (*Mathematical Markup Language*) für mathematische Formeln, CML (*Chemical Markup Language*) für die Chemie, GML (*Geography Markup Language*) oder SBML (*Systems Biology Markup Language* [75]) für Biologie. Für die Informatik existieren sehr viele Dialekte, da einerseits viele Programme eigene Dialekte vorschlagen, um ihre Daten zu speichern und andererseits in der Informatik das Potenzial von XML bereits früh erkannt wurde. Hier sind mit Scalable Vector Graphics (SVG), einer Beschreibungssprache für zweidimensionale Vektorgrafiken, und eXtending 3D (X3D) für dreidimensionale grafische Szenen zwei XML-Dialekte für die Computergrafik zu nennen, sowie SMIL (*Synchronized Multimedia Integration Language*), eine Auszeichnungssprache für multimediale Inhalte.

Generell lässt sich feststellen, dass jegliche Standardsoftware wie Textverarbeitungssoftware (beispielsweise Microsoft Office oder OpenOffice), Serversoftware wie Applikationsserver oder Datenbanken (beispielsweise JBoss, Apache Tomcat, Oracle oder DB2) XML heutzutage entweder nativ oder über Im- und Exportfunktionen unterstützen.

2.2 Die Funktionsweise von XML

Ein XML-Dokument ist ein in Unicode verfasstes Textdokument. Damit ist es sprachenspezifisch und plattformunabhängig. Es besteht aus Elementen ("tags"), welche in spitze Klammern gesetzte Wörter sind (<para>). Auf ein öffnendes Element folgt immer ein korrespondierendes schließendes Element, welches durch ein führendes umgekehrten Schrägstrich

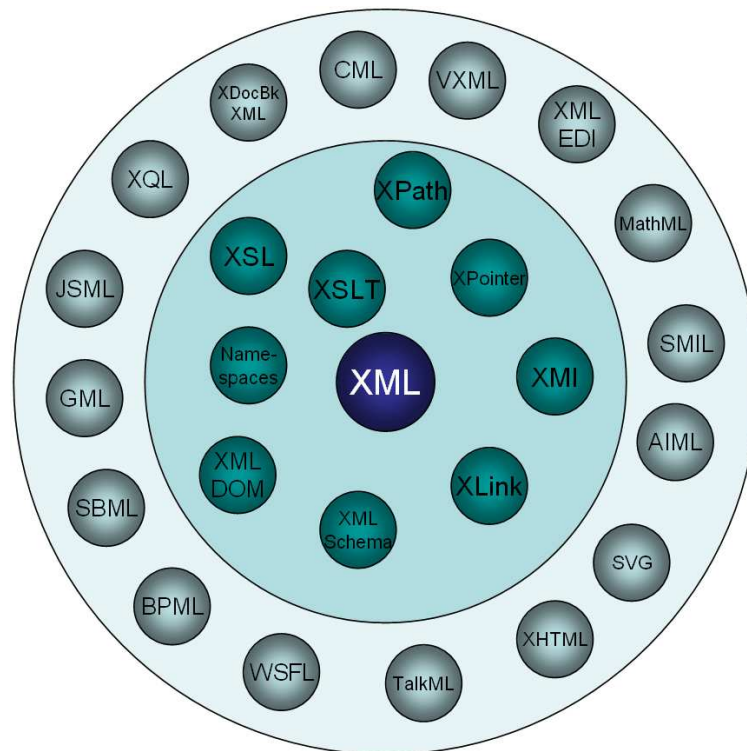


Abbildung 2.1 Das XML-Universum nach Michel

markiert wird (`</para>`). Elemente können ein oder mehrere Attribute unterschiedlichen Namens enthalten, deren Werte in einfache oder doppelte Hochkommata gesetzt werden müssen (`<para id="1" style="justify">`). Elemente können leer sein oder Text oder weitere Elemente umschließen. Ein leeres Element kann entweder als `<para></para>` oder verkürzt `<para/>` notiert werden. Ein XML-Dokument darf nur ein so genanntes Wurzelement enthalten. Sind diese und einige weitere weniger wichtige Regeln erfüllt, so ist das XML-Dokument *wohlgeformt* (*well-formed*). Ein nicht wohlgeformtes XML-Dokument ist genau genommen kein XML-Dokument. Will man ein XML-Dokument speichern, so legt man es als Textdatei ab, fügt jedoch noch eine so genannte XML-Deklaration in einer Kopfzeile hinzu, in welcher die XML-Version und die Codierung (z.B. UTF-8, UTF-16, ISO-8859-1) als Hilfe für Parserprogramme festgelegt werden. Eine solche Kopfzeile wird mit der Auszeichnung `<? ... ?>` umgeben. Ein Beispiel für ein wohlgeformtes XML-Dokument sieht dann so aus:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <book>
3   <titel >XML ganz kurz gefasst</ titel >
4   <autor>
5     <vorname>Johannes</vorname>
6     <nachname>Görke</nachname>
7   </autor>

```

```
8 <jahr>2006</jahr>
9 <text>Hier steht der Text des Buches</text>
10 </book>
```

Listing 2.1 Ein einfaches wohlgeformtes XML-Dokument

Bisher wurde noch keine Aussage darüber getroffen, wie die Elemente und Attribute heißen, welche Elemente innerhalb welcher anderer Elemente auftauchen dürfen oder wie oft sie dort auftauchen dürfen. Will man diese Informationen spezifizieren, so benötigt man eine Dokumenttypdefinition (*document type definition - DTD*) oder ein XML-Schema. DTDs wie XML-Schema können in einer weiteren Prologzeile im Kopf eines XML-Dokuments referenziert werden. DTDs können auch im XML-Dokument direkt enthalten sein. DTDs wurden bereits zur Definition von SGML-Dokumenten verwendet und benutzen eine erweiterte Backus-Naur-Form [157], welche natürlich kein XML darstellt. In einer DTD werden Elemente mit ihren Namen aufgezählt sowie die Abhängigkeiten untereinander. Als Datentypen gibt es nur Buchstabenmengen (*parsed character data - PCDATA*) oder Aufzählungen. Zahlen oder gar Zahlenbereiche kommen nicht vor, man müsste sie sich mühsam selbst definieren. Eine DTD für Beispiel 2.1 könnte folgendes Dokument sein:

```
1 <!ELEMENT book (titel, autor+, jahr , text )>
2 <!ELEMENT titel (#PCDATA)>
3 <!ELEMENT autor (vorname, nachname)>
4 <!ELEMENT jahr (#PCDATA)>
5 <!ELEMENT vorname (#PCDATA)>
6 <!ELEMENT nachname (#PCDATA)>
7 <!ELEMENT text (#PCDATA)>
```

Listing 2.2 Eine Beispiel-DTD für ein einfaches XML-Dokument

Mittels dieser DTD kann ein Parser nun überprüfen, ob ein XML-Dokument den Anforderungen dieser DTD entspricht. Ist dies der Fall, so wird das XML-Dokument *gültig* (*valid*) genannt. Die Prüfung auf Gültigkeit ist nicht zwingend, beim Parsen kann trotz referenzierter DTD diese unterbleiben. Einige Nachteile von DTDs sind offensichtlich: Es wird eine andere Notation als für das XML-Dokument verwendet, was für Entwickler eine Umstellung erfordert und für Parser bedeutet, dass sie mehr als nur XML parsen können müssen. Vor allem sind DTDs bezüglich ihrer Aussagekraft über Inhalte von Elementen sehr eingeschränkt. Als Folge hat das W3C XML-Schema entworfen, was sich als in XML notierte DTD mit erweiterten Definitionsmöglichkeiten beschreiben lässt. Spezifizieren lassen sich jetzt auch Datentypen wie beispielsweise ganze Zahlen, boolesche Werte, Datumsangaben, aber auch base64-kodierte Binärdaten. Mittels einfacher und

komplexer Datentypen können über reguläre Ausdrücke Zahlen- und Buchstabenkombinationen wie Sozialversicherungsnummern selbst definiert werden und diese dann über Ableitung und Vererbung und Namensräume weiterverwendet werden. Die Häufigkeit der Vorkommnisse von Elementen kann ebenso in beliebigen Maßen festgelegt werden. Ein der DTD 2.2 entsprechendes Schema hätte folgendes Aussehen:

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3   targetNamespace="http://www.gris.uni-tuebingen.de/bookSchema"
4   xmlns="http://www.gris.uni-tuebingen.de/bookSchema"
5   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6   xsi:schemaLocation="http://www.w3.org/2001/XMLSchema http://www.w3.org/2001/
7     XMLSchema.xsd"
8   elementFormDefault="qualified">
9   <xsd:element name="book" type="bookType"/>
10  <xsd:complexType name="bookType" mixed="true">
11    <xsd:sequence>
12      <xsd:element name="titel" type="xsd:string"/>
13      <xsd:element name="autor" type="AutorType" minOccurs="1" maxOccurs="5"/>
14      <xsd:element name="jahr" type="xsd:integer"/>
15      <xsd:element name="text" type="xsd:string"/>
16    </xsd:sequence>
17  </xsd:complexType>
18  <xsd:complexType name="AutorType" mixed="true">
19    <xsd:sequence>
20      <xsd:element name="vorname" type="xsd:string"/>
21      <xsd:element name="nachname" type="xsd:string"/>
22    </xsd:sequence>
23  </xsd:complexType>
24 </xsd:schema>
```

Listing 2.3 Ein Schema für das einfache XML-Dokument aus Listing 2.1

XML-Schema ist ein sehr mächtiges Werkzeug zum Testen der Gültigkeit von Dokumenten, jedoch werden die Definitionen relativ schnell sehr umfangreich. Selbst dieses kurze Beispiel ist schon umfangreich, wobei aus Gründen der Übersicht die Jahreszahlen oder die maximale Länge der Namen von Autoren nicht genauer spezifiziert wurden. Da die Länge von XML-Schema-Dokumenten als Nachteil gegenüber DTDs empfunden wurde, wurden auch weitere Schemasprachen wie Relax NG [111] und Schematron [120] entwickelt. Diese weichen wiederum von einer durchgängigen XML-Syntax ab, ohne jedoch auf die Mächtigkeit der Definitions- und Spezifikationsmöglichkeiten zu verzichten.

Wie im obigen Schemabeispiel 2.3 zu sehen, werden die Elementnamen mit einem vorangehenden "xsd:"-Präfix bezeichnet. Hiermit kann man in XML-Dokumenten die *Namensräume* von XML-Elementen definieren. Wenn wie beispielsweise in einem XHTML-Dokument Elemente verschiedener Schemata wie MathML, SVG und XHTML gemischt werden sollen, so schreibt man diese Elementen mit den Präfixen ihrer Schemata. Damit vermeidet man Verwechslungen und erlaubt Parsern die Validierung gegen mehrere verschiedene Schemata gleichzeitig. Die Präfixe müssen im Dokumentenkopf einmal mittels einer eindeutigen Ressourcen-Bezeichnung (*Uniform Resource Locator* - URI) definiert werden, dann können sie im Dokument verwendet werden. Durch Definition eines Default-Namensraums werden Elemente ohne Präfix diesem Namensraum zugeordnet, so dass man sich Schreibarbeit sparen kann. Namensräume wurden erst nach Vorstellung des Standards XML 1.0 eingeführt, sind jedoch abwärtskompatibel, so dass Parser, welche Namensräume nicht kennen, diese Elemente einfach als Elemente mit enthaltenen Doppelpunkten erkennen.

Eine kurze Übersicht zu XML und den grundlegenden Techniken findet sich in [102]. Sehr ausführlich haben sich Harold und Means mit XML beschäftigt [69]. Alle Standards rund um XML findet man auf den Webseiten des W3C. Sehr viele praktische Hinweise mit Programmertipps finden sich im WWW, eine sehr gute ausführliche Übersicht gibt hier Stefan Münz mit Selfhtml [103].

2.3 Programmierschnittstellen und Transformation

Die Verarbeitung von XML-Dokumenten aus Programmen heraus ist prinzipiell durch zwei Ansätze möglich. Ein einfacher Ansatz besteht darin, das Dokument sequenziell zu verarbeiten und bei jedem Auftreten von XML-Elementen diese per Ereignis an das Programm zurück zu melden. Dieser Ansatz wird *Simple API for XML* (SAX [27], [99], bei letzterem findet man Bibliotheken beispielsweise für Java) genannt und liegt inzwischen in der Spezifikation SAX2 vor. Durch die Verarbeitung des XML-Dokuments als Eingabestrom von Events ist die Verarbeitung beliebig großer Dokumente in hoher Geschwindigkeit möglich, da immer nur die aktuell gelesenen Elemente im Speicher zu halten sind. Allerdings muss das aufrufende Programm dafür sorgen, dass es auf die richtigen Bereiche des Dokuments zugreift. Ebenso kann die Verarbeitung von Informationen aus dem Dokument sofort durchgeführt werden, da nicht auf das vollständige Parsen des Dokuments gewartet werden muss. Deswegen können mit SAX sehr gut Filter aufgebaut werden.

Der zweite wichtige Ansatz ist der Aufbau einer Baumrepräsentation des XML-Dokuments im Speicher. Da XML-Dokumente ein einziges Wurzelement besitzen und Elemente korrekt geschachtelt sein müssen, liegt die Darstellung als Baummodell auf der Hand. Methoden erlauben den Zugang zu allen Knoten, man kann die Knoten manipulieren und umsortieren. Seit DOM Level 3 ist der Zugriff auch über XPath [162] möglich. Da das ganze Dokument als Baum im Speicher vorgehalten wird und dabei auch Attribute und ihre Werte sowie die Textinhalte von Elementknoten als eigene Blät-

ter gespeichert werden, ist der Speicherverbrauch nicht zu vernachlässigen. Wenn es sich nicht um Datenbankabfragen beschränkter Größe handelt, können XML-Dokumente leicht mehrere hundert Megabyte groß werden. Dafür ist die DOM-Repräsentation sicher nicht angemessen und es empfiehlt sich der Einsatz von SAX. Es gibt für DOM ebenfalls frei verfügbare Bibliotheken, eine häufig eingesetzte Variante für Java ist unter [38] zu finden.

Um ein XML-Dokument vom Ursprungsformat beispielsweise nach XHTML [105] zu übersetzen, kann man also ein Programm schreiben, welches SAX oder DOM benutzt, um alle Elemente des Ursprungdokuments in entsprechende Elemente des Zielformats zu transformieren. Dabei würde beispielsweise das Element "<title>" des Ursprungdokument in das Element "<h1>" (dieses Element zeichnet Überschriften in XHTML aus) übersetzt. Man bräuchte vor allem einen Regelkatalog, welche Elemente in welche abgebildet werden und was mit dem enthaltenen Text- und Attributwerten passieren soll. Da diese Anwendung sehr häufig gebraucht wird, wurde ein entsprechender Standard für XML, die *eXtended Stylesheet Language - XSL* und *XSL Transformations* [32] entwickelt. Mittels eines so genannten Stylesheets können Regeln definiert werden, nach denen ein gegebenes XML-Dokument transformiert werden soll. Das bekannteste Programm für derartige Transformationen ist Xalan [4], ursprünglich eine IBM-Entwicklung, welche später an die Apache Software Foundation übergeben wurde und unter deren Regie weiterentwickelt wurde. Frei verfügbare Bibliotheken sind plattformübergreifend in C++ und Java verfügbar. XSL ist Turing-vollständig, es ermöglicht die Definition von Variablen und Parametern, ermöglicht die Definition eigener Funktionen, welche rekursiv aufgerufen werden können und stellt bedingte Verarbeitung sowie Iterationen zur Verfügung. Aus dem Ursprungdokument ausgelesene Werte können als mathematische Ausdrücke ausgewertet werden, für Zeichenketten liegt ebenfalls eine Fülle von vordefinierten Funktionen vor. Zusätzlich können auch Erweiterungen in Java oder anderen Programmiersprachen eingebunden werden. Der Zugriff auf bestimmte Elemente des Originaldokuments erfolgt unter Verwendung von XPath [162]. XPath ermöglicht eine direkte Adressierung von Elementen, Attributen und Werten. Dabei wird eine UNIX-ähnliche Syntax verwendet: Das Wurzelement ist durch "/" bezeichnet, der Knoten "vorname" aus dem obigen Beispieldokument 2.1 wird mit "/book/autor/vorname" angesprochen. Es gibt Wildcards und verkürzte Notationen, Vergleiche von Zeichenketten mit Elementinhalten zur Selektion von eingeschränkten Knotenmengen und vieles mehr. Eine Beispieltransformation für Dokument 2.1 könnte folgendes Stylesheet sein:

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2
3 <xsl:stylesheet version="1.0"
4 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
5
6 <xsl:template match="/">
7 <html>
```

```

8   <xsl:apply-templates select="buch"/>
9   </html>
10  </xsl:template >
11
12  <xsl:template match="buch">
13    <head>
14      <title ><xsl:value-of select="titel" /> written by
15        <xsl:value-of select="autor/nachname"/>
16    </title >
17  </head>
18  <body>
19    <h1><xsl:value-of select="titel" /></h1>
20    <h2><xsl:value-of select="autor/vorname"/>&#160;
21      <xsl:value-of select="autor/nachname"/></h2>
22    <p><xsl:value-of select="text" /></p>
23  </body>
24 </xsl:template >
25 </xsl:stylesheet >

```

Listing 2.4 Ein Beispiel-XSL-Dokument zur Transformation des Dokuments in Listing 2.1

```

1  <html>
2  <head>
3    <META http-equiv="Content-Type" content="text/html;
4      charset=UTF-8">
5    <title >XML ganz kurz gefasst written by G&ouml;rke</title>
6  </head>
7  <body>
8    <h1>XML ganz kurz gefasst</h1>
9    <h2>Johannes&nbsp;G&ouml;rke</h2>
10   <p>Hier steht der Text des Buches</p>
11 </body>
12 </html>

```

Listing 2.5 XHTML-Resultat

Im Browser wird das Ergebnis der beiden vorangegangenen Listings wie in Abbildung 2.2 gerendert.

Neben dieser Transformation, welche aus einem XML-Dokument ein neues XML-Dokument generiert, gibt es auch XSL-FO (*XSL Formatting Objects*), eine XML-Anwen-



Abbildung 2.2 Das XHTML-Resultat im Browser

ung, welche sich vollständig auf das Seitenlayout konzentriert. Zu Beginn eines FO-Dokumentes kann eine Seitenvorlage mit Angaben zu Seitenrändern, Seitengröße usw. erstellt werden. Der Text wird dann blockweise aufgeteilt, hier können wiederum Angaben zu Größen, Rändern oder Schrifttypen gemacht werden. Für ein derartig formatiertes Dokument existieren schließlich automatische Transformierer wie FOP [7], welcher aus einer XML-FO-Vorlage unter anderem Dateien im PDF-Format (Adobes Portable Document Format), im PCL-Format (Hewlett-Packards Printer Command Language), PS-Format (PostScript, von John Warnock und Chuck Geschke bei Adobe entwickelt) oder in SVG (siehe 2.4.1) erzeugt.

Für eine vertiefte Übersicht zu XSL sei auf das Buch von Harold [69] und die Erläuterung von Münz zur Verwendung von XSL [103] verwiesen.

2.4 Wichtige XML-Anwendungen

In diesem Abschnitt sollen wichtige Anwendungen von XML vorgestellt werden. Hierunter fallen vor allem Docbook und Dublin Core, MathML, SVG und X3D auch im Hinblick auf die in dieser Arbeit verwendeten Techniken.

Docbook [149] wurde 1991 unter Führung des O'Reilly-Verlags als SGML-Anwendung entworfen. Für technische Dokumentationen sollte Docbook die plattformübergreifenden Fähigkeiten des Unix-Kommandos `troff` zur Aufbereitung von Text erhalten. Der Standard ist öffentlich verfügbar und wird seit Mitte 1998 unter Führung der OASIS-Gruppe auch als XML-Schema weiterentwickelt. Docbook selbst ist als Notationsschema außer als Text am Bildschirm nicht darstellbar. Aufgrund der Offenheit des Standards gibt es aber eine Reihe von Konvertern, welche Docbook-formatierte Texte beispielsweise nach HTML, ins *Rich Text Format* (RTF) für Microsoft Office Word oder nach LaTeX [88] übersetzen. Die Konvertierung nach LaTeX ermöglicht eine weitere Transformation nach PostScript oder Adobes *Portable Document Format* (PDF). Im Jahr 2006 liegt Docbook in Version 5 vor, das entsprechende XML-Schema ist 750 Kilobyte groß.

Verschiedene Initiativen haben sich mit der Beschreibung von Quellen verschiedener

Herkunft bemüht. Mit dem *Resource Description Framework* (RDF) [160] von 1999 schuf das W3C ein Rahmenwerk als Basis für das Semantische Web [20]. Bereits seit 1995 bemüht sich dagegen die *Dublin Core Metadata Initiative* (DCMI) um die Beschreibung von Objekten im Internet. Hierfür können im HTML-Seitenkopf über Meta-Elemente Zusatzinformationen zum Autor, zum Thema, zu Referenzen oder auch zu Veröffentlichungsrechten und Modifikationen definiert werden. Je nach verwendetem System ist es auch möglich, diese Daten in einer auf RDF basierenden XML-Sprache abzulegen. Dies wird insbesondere von Content-Managementsystemen übernommen, die bei Bedarf die vorhandenen Daten gesondert aufbereitet ausliefern. Verwendung findet Dublin Core heute im OpenDocument-Format [79] und in verschiedenen Content-Managementsystemen wie Apache Lenya [8].

Die *Mathematical Markup Language* (MathML) ist eine XML-Anwendung des W3C und wurde im Sommer 1998 verabschiedet [159]. Bereits seit Februar 2001 liegt eine Definition von MathML 2.0 vor. Mittels MathML können mathematische Formeln in HTML-Seiten dargestellt werden, ohne den üblichen Umweg über eingebettete Bilder zu gehen. Eine Spezialität von MathML ist die Unterteilung des Standards in zwei Teile, einerseits gibt es die so genannte Präsentationsschicht (*Presentation Markup*), in der das Aussehen der Formel im Layoutformat ähnlich wie bei LaTeX definiert wird, und andererseits gibt es die so genannte Inhaltsschicht (*Content Markup*), bei der die Semantik der Formel beschrieben wird. Die semantische Schicht ist immer dann wichtig, wenn Algebrasysteme oder ähnliche Programme mathematische Formeln importieren sollen. Zur Darstellung im Web genügt im Allgemeinen die Präsentationsschicht. MathML-Ausdrücke fallen sehr groß aus, weswegen die Bearbeitung als Textdatei nicht von Vorteil ist. Inzwischen gibt es mehrere Softwarepakete (OpenOffice, MathType), welche die Eingabe über einen Formeleditor erlauben und MathML generieren. Dazu existiert eine große Anzahl an Transformatoren teils auf XSL-Basis, welche auch LaTeX als Ein- oder Ausgabeformat für Formeln erlauben.

2.4.1 Scalable Vector Graphics

Für die Darstellung von Vektorgrafiken im Netz wurde seit 1998 der Standard *Scalable Vector Graphics* (SVG) [166] unter Führung des W3C entwickelt. Die Version 1.0 wurde im September 2001 als Empfehlung des W3C veröffentlicht. Im Januar 2003 folgte dann Version 1.1 und aktuell wird an Version 1.2 gearbeitet. SVG geht auf die zwei Standards *Vector Markup Language* (VML) und der *Precision Graphics Markup Language* (PGML) zurück, deren Ergebnisse in SVG einfließen. Vektorgrafiken beschreiben die enthaltenen Objekte (oder Grafikprimitive) mathematisch genau mittels Vektoren. Daher sind derartige Grafiken nicht auf eine Auflösung beschränkt, sondern können skaliert werden. Dadurch unterscheiden sie sich von Bildern, so genannten Rastergrafiken, die beispielsweise im Format BMP (Bitmap von Microsoft, unkomprimiert), GIF (*Graphics Interchange Format* von CompuServe, verlustfrei komprimiert), PNG (*Portable Network Graphics*, verlustfrei komprimiert, im Gegensatz zu GIF nicht belastet durch Patentstreitigkeiten [121]), JPEG (*Joint Photographic Experts Group*, verlustbehaftet komprimiert)

oder in vielen weiteren vorliegen können. Die Speicherung von Grafik als Vektoren ermöglicht eine hohe Kompression, eine auflösungsunabhängige Speicherung, die synthetische Generation von Grafiken als Kurvenverläufe oder Charts und die Einbettung von textueller Information.

SVG-Dokumente bestehen aus einem Zeichenblatt mit kartesischem Koordinatensystem. Der Ursprung befindet sich in der linken oberen Ecke, die x-Achse verläuft von links nach rechts, die y-Achse von oben nach unten. Der sichtbare Bereich des Zeichenblattes wird mittels der Attribut-Angaben "width" und "height" im Dokument festgelegt. Es gibt drei grundlegende Grafikobjekte in SVG:

- Grafikprimitive
- Rastergrafiken
- Zeichen und Zeichenketten

Die Grafikprimitive sind einfachste geometrische Objekte wie Rechteck, Kreis, Ellipse, Linie, Polylinie, Polygonzüge (also geschlossene Polylinien) und Pfade, mittels welcher auch Linien oder quadratische und kubische Bezierkurven definiert werden und an denen andere Objekte wie Text ausgerichtet werden können. Transformationen innerhalb von SVG beinhalten Translation, Rotation, Skalierung und Scherung. Hinzu kommen Effekte wie lineare und radiale Verläufe sowie sehr viele Filter, beispielsweise diffuse Beleuchtung, Weichzeichnen, Blenden, Zusammenfügen von Objekten, Morphologische Operationen, spekulare Beleuchtung usw. Schließlich existiert in SVG ein Animationsmodell, welches über SMIL oder mittels ECMAScript [78] gesteuert wird. Interaktivität wird ebenfalls mittels ECMAScript erreicht, ein Eventhandler kann für Ereignisse wie *onload* (die Seite wurde fertig geladen), *onmouseover* (der Mauszeiger befindet sich über dem entsprechenden Objekt), *onclick* (das Objekt wurde angeklickt) definierte Funktionen aufrufen, welche dann auf das Ereignis reagieren. Der Standard ist mittlerweile so umfangreich, dass selbst Anwendungen wie Kartenbetrachter [92] oder Spiele wie Tetris [47] möglich sind. Wie anhand der genannten Beispiele deutlich wird, ist SVG eine relativ umfangreiche Sprache, die insbesondere bei Verwendung von Animationen einige Rechenkraft erfordert. Deswegen wurden zur Verwendung in Mobiltelefonen und *Personal Digital Assistants* (PDAs) eingeschränkte Profile in SVG 1.1 entworfen: SVG Basic und SVG Tiny. Während SVG Basic vor allem auf das Skripting von SVG verzichtet, ist SVG Tiny auch im Umfang der zur Verfügung stehenden Elemente reduziert. Filter und Verläufe sind hier nicht vorhanden.

```
1 <?xml version="1.0" standalone="no"?>
2 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
3   "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd" >
4 <svg width="12cm" height="6cm" viewBox="0 0 1200 600" version="1.1"
5   xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
6 <defs>
```

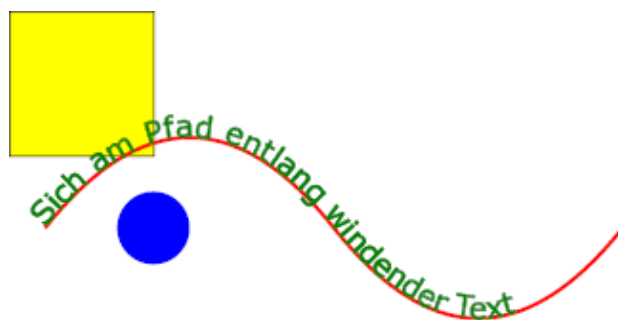


Abbildung 2.3 SVG-Primitive, Bezierkurve und Text

```

7 <path id="Kurve" d="M50,300 q200,-250 400,0 t400,0" />
8 </defs>
9 <title>SVG-Beispiel</title>
10 <desc>Dieses Bild zeigt verschiedene grafische Primitive, eine quadratische Bezierkurve und
    Text</desc>
11 <rect x="1" y="1" width="200" height="200"
12       fill="yellow" stroke="black" stroke-width="2" />
13 <circle cx="200" cy="300" r="50" fill="blue" />
14
15 <use xlink:href="#Kurve" fill="none" stroke="red" stroke-width="5" />
16 <text font-family="Verdana" font-size="42" stroke="green" fill="green" >
17   <textPath xlink:href="#Kurve" spacing="auto">Sich am Pfad entlang windender Text</
    textPath >
18 </text >
19 </svg>

```

Listing 2.6 Der zu Abbildung 2.3 gehörende Quellcode

2.4.2 eXtensible 3D

Im Gegensatz zu SVG, welches für zweidimensionale Daten gedacht ist, ist X3D ein Standard für dreidimensionale Szenen. Er wurde vom Web3D Consortium entwickelt, welches X3D als Nachfolge-Format der Virtual Reality Modeling Language (VRML97 [29]) etablieren will. Seit 2004 ist X3D ein ISO-Standard (ISO/IEC 19776 [151]). X3D besteht aus einem Kern (*core*), Profilen (*profiles*) und Komponenten (*components*). Profile können Gruppen von Komponenten sein. Beide werden separat installiert, ein X3D-Betrachter zeigt bei nicht installierten Erweiterungen entsprechende Funktionalitäten nicht an. Der reine Kern bietet im Verhältnis zu VRML97 eine reduzierte Funktionalität, da VRML97 als zu komplex und zu wenig modular empfunden wurde. Erst mit dem *immer-*

sive profile erreicht man Kompatibilität zu VRML97. Komponenten können Objekte wie Interpolation, Texturierung, 2D-Geometrie, 3D-Geometrie, NURBS, Beleuchtung usw. sein. Um die Migration von VRML97 zu erleichtern, wird auf den Webseiten des Web3D-Consortium ein XSL-Schema zur Übersetzung nach X3D angeboten.

Ein einfaches X3D-Dokument enthält einen Szenengraphen. Er besteht aus geometrischen Primitiven wie Kugel, Zylinder, Kegel und Box sowie auch aus zweidimensionalen Objekten. Das Aussehen wird durch Materialeigenschaften und Belechtungsknoten beschrieben. Transformationen ordnen die Objekte im Raum an, sie können dazu gruppiert werden. Sensoren regeln Kollisionen, wie auch Interaktionen über Mausclicks und Zeitintervalle. Interpolatoren sind für Animationen zuständig. Programmcode kann über Skriptknoten eingebunden werden.

Das Definitionspaket von X3D ist umfangreich. Es können nicht nur animierte interaktive Darstellungen von komplexen 3D-Szenen unter Einsatz von Hardware-Beschleunigung wie Pixel- und Vertex-Shadern mit Kollisionsdetektion eingebunden werden sondern auch vom Nutzer definierte Objekte. Dies geht bis hin zur Komposition von Szenen aus unterschiedlichen, über das Netz verteilten Einheiten. Die Einbindung von audiovisuellen Quellen ist möglich wie physische Simulation und Animation von Humanoiden. Mittels des *interchange profile* können Rohdaten von verschiedenen CAD-Programmen eingelesen werden.

2.5 XML im Web

Erfinder des *World Wide Web* (WWW), welches kurz auch Web genannt wird, ist Tim Berners-Lee, der es ab 1989 am CERN in Genf zur einfacheren Publikation wissenschaftlicher Arbeiten und Forschungsergebnisse entwickelte. Dafür entwickelte er nicht nur einen Webserver sondern auch einen Browser, also ein Programm, welches HTML darstellt, und das Protokoll HTTP (*HyperText Transport Protocol*), welches die Kommunikation zwischen Webserver und Browser regelt. Wichtiges Element war die unidirektionale Verlinkung per URL (*Uniform Resource Locator*): Beliebige Worte können als Link markiert auf andere Dokumente verweisen. Diese Dokumente können sich auch auf anderen Webservern befinden. Die erste Browserversion konnte nur Text darstellen, Bilder kamen erst 1992 mit Viola [48] hinzu. Als erster grafischer Browser wurde Mosaic vom National Center for Supercomputer Applications (NCSA) unter Führung von Marc Andreessen entwickelt und 1993 vorgestellt. Mosaic war sehr schnell der am weitesten verbreitete Browser. Andreessen gründete 1994 Netscape Communication Corporation, welche den Netscape Navigator veröffentlichte, der bald Mosaic als beliebtesten Browser verdrängte. Jede neue Version der Browser brachte Erweiterungen zum bis dahin standardisierten HTML, welche dann bald in den Standard aufgenommen wurden. So wurde die erste Version im Juni 1993 als Arbeitsentwurf der *Internet Engineering Task Force* (IETF) verabschiedet. Tim Berners-Lee gründete 1994 das *World Wide Web Consortium* (W3C), dessen Leitung er übernahm und das die HTML-Standards im folgenden verabschiedete. 1995 folgte dabei HTML 2.0 mit der Neuerung "Formulare". HTML 3.2 erschien An-

fang 1997, Tabellen werden Teil des Standards. Ende 1997 erschien HTML 4.0, *Cascading Stylesheets*, JavaScript (standardisiert unter dem Namen ECMAScript) und Rahmen (*Frames*) waren wesentliche Neuerungen. Im Dezember 1999 wurde HTML 4.0 zum Standard gemacht, einen Monat später erschien HTML 4.01 und die "xml-ifizierter" Version von HTML 4.01 namens XHTML 1.0 [161].

Zusätzlich zu den in die Standards aufgenommenen Erweiterungen setzten sich drei grundsätzliche Prinzipien durch: Über eine Plugin-Schnittstelle können alle Browser externe Programme starten, um andere Inhalte darzustellen. Die wichtigsten Plugins sind Java (Sun Microsystems), Flash (Adobe), Windows Media Format (Microsoft), Quicktime (Apple) und RealMedia (Real Networks). Des Weiteren unterstützen alle Browser die Interpretation von JavaScript, einer Skriptsprache, die im Browser ausgeführt wird und unter anderem die Manipulation des aktuell geladenen HTML-Dokuments erlaubt. Und drittens kann man beliebige Dateitypen und Formate über das WWW verbreiten, da Browser bei unbekanntem Dateitypen, die sie über den MIME-Type erkennen, dem Benutzer über einen Dialog das Abspeichern der Datei anbieten, oder ein Programm starten, welches diese Datei verarbeiten kann.

Die rasche Entwicklung des HTML-Standards war maßgeblich vom so genannten Browserkrieg getrieben, einer Auseinandersetzung zwischen Microsoft und Netscape um die Vorherrschaft bei den Webbrowsern. Diese begann 1995, als Microsoft die Bedeutung des WWW erkannte, und nun mit aller Macht versuchte, einen eigenen Browser zu etablieren und damit auch die zukünftigen HTML-Standards zu setzen. Dieser Krieg wurde 1998 gewonnen, Netscape wurde von AOL übernommen und der Netscape Navigator beziehungsweise dessen Nachfolgeversion Communicator wurde nicht mehr weiterentwickelt. Danach nahm auch die Geschwindigkeit der Veröffentlichung neuer Versionen des Internet Explorer, des Browsers von Microsoft, ab. Die aktuelle Version 6.0 erschien schon 2001. Erst seit 2004 gibt es wieder Bewegung im Browsermarkt mit dem Erscheinen von Mozillas Firefox, einem Nachfolger des Netscape-Browsers, Apples Safari, dem Konqueror (der Browser des KDE-Projekts) unter Linux und Opera.

Der relative Stillstand ab 2001 hatte Vorteile für den Anwender, der nun nicht mehr ständig seine Webseiten erneuern musste, um neue Möglichkeiten der Standards bei sich einzubauen. Aber es gab auch viele Nachteile: Die Browser sind nach wie vor nicht vollständig kompatibel zum Standard, bieten einerseits mehr Funktionalität als der Standard an und decken ihn gleichzeitig in anderen Bereichen nicht vollständig ab [103]. Da HTML 4.01 im Wesentlichen alle Bedürfnisse der Anwender befriedigt, um Seiten im WWW darzustellen, war für die Anwender kein sonderlich großer Druck da, zu XHTML zu migrieren, zumal zunächst nur Arbeit investiert wird und kein Nutzen aus einer derartigen Migration gezogen werden kann.

Aufgrund der immer noch mangelhaften Unterstützung der W3C-Standards muss also nach wie vor für jede Anwendung im Web genau überlegt werden, auf welche Technik man setzt und wieviele Anwender man durch möglicherweise nicht vorhandene Plugin-Installationen verliert, wenn sie nicht über die nötigen Rechte zur Installation solcher Software verfügen.

Browser	SVG	MathML	XSL
Netscape/Mozilla	mit Einschränkungen	Ja	Nein
Internet Explorer	Mit Einschränkungen/Plugin	Plugin	Ja
Opera	Ja	Nein	Nein

Tabelle 2.1 Neue W3C-Standards in Webbrowsern (Stand 2006)

2.6 Dynamische Inhalte im Web

Die wichtigsten Erfolgsfaktoren für das WWW sind die Einfachheit von HTML und die kostenlose Verfügbarkeit von Browsern. Die Verlinkung von Webseiten kann im allgemeinen vorgenommen werden, ohne dass man das Einverständnis desjenigen einholen muss, dessen Dokumente man verlinkt, da diese ja nicht lokal kopiert werden, sondern vom Nutzer immer auf der Originalwebseite abgerufen werden. Mit Tabellen, Formularen und Cascading Stylesheets sind schön gestaltete Webseiten möglich, die Funktionalität ist ebenfalls beachtlich. Jedoch fehlte die Möglichkeit, beispielsweise personalisierte Daten anzuzeigen oder einen Account in einem Online-Shop zu verwalten. Dafür wurde das *Common Gateway Interface* (CGI) [30] erfunden: eine standardisierte Schnittstelle auf der Serverseite. Normalerweise wird, wenn man im Browser ein Webdokument ansteuert, der entsprechende aufgerufene Webserver in seinem Verzeichnisbaum nach der Datei diesen Namens suchen und sie an den Browser zurückliefern. Im Falle von CGI wird nun kein Dokument abgerufen, sondern der Webserver interpretiert das aufgerufene Dokument als ausführbares Programm, welches er startet und dessen Ausgabe er an den Aufrufer zurückliefert. Dabei werden eventuelle Formulardaten aus dem Aufruf an das Programm als Eingabewerte weitergeleitet. Somit kann ein Programm die aktuelle Uhrzeit oder auch nutzerbezogene Daten zurückgeben.

Da das Starten eines externen Programms relativ viel Zeit beansprucht, insbesondere, wenn ein Interpreter wie Perl [148] geladen werden muss, suchte man bald eine Möglichkeit zur Beschleunigung der externen Programmaufrufe, um die Serverlast zu reduzieren. Hierzu wurde FastCGI [128] entworfen: Der Interpreter wird im Unterschied zu CGI einmal geladen und bleibt dann im Speicher, der Webserver leitet die Aufrufe über Sockets an diesen Interpreter weiter. Trotz der guten Performanz setzte sich FastCGI nie wirklich durch, begründet auch durch den Erfolg der so genannten aktiven Serverseiten (*active server pages*).

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <%@page language="java"%>
3 <html>
4   <head>
5     <title>WSI/GRIS – Datum</title>
6   </head>
7   <body>
8     <h2>Datum</h2>

```

```
9
10 <% java.text.SimpleDateFormat sdf = new java.text.SimpleDateFormat();
11     String myDate = sdf.format(java.util.Calendar.getInstance().getTime());%>
12
13     Heute ist der <%= myDate %>
14
15 </body>
16 </html>
```

Listing 2.7 JSP-Beispielseite

Diese aktiven Serverseiten gibt es für mehrere Sprachen: Java (*Java Server Pages - JSP*), VBScript von Microsoft (*Active Server Pages - ASP*), CFML (*Coldfusion Markup Language* von der Allaire Corporation), Python (Zope) und PHP (*Personal Home Page Tools*, später *PHP: Hypertext Preprocessor*, von Perl abgeleitet, inzwischen eigenständiger Dialekt). Ihr Hauptmerkmal ist, dass man im Unterschied zu CGI oder FastCGI wieder HTML-Seiten schreibt, in welche der Programmcode umgeben von speziellen Elementen eingebettet ist. Der Applikationsserver übernimmt die Kompilierung und das dynamische Ausführen der Anwendung.

Alle derartigen serverseitigen Sprachen unterstützen insbesondere die Anbindung von Datenbanken, um auch mit großen Mengen an Daten umgehen zu können. Insgesamt gesehen versuchen all diese Ansätze, alle Intelligenz und alle Rechenlast auf den Server zu übertragen und dem Nutzer mit seinem Webbrowser nur einfache Textseiten zu präsentieren, da beim Nutzer nur von einem einfachen Browser ohne jegliche installierte Plugins ausgegangen werden kann. Zur Verteilung der großen Last bieten diese Softwaresysteme oft auch Möglichkeiten der Lastverteilung auf mehrere Instanzen auf mehreren Rechnern an.

Eine relativ neue Entwicklung im Bereich der Programmierung dynamischer Webseiten ist die Idee der Continuation [153] [35] [6]. Eine Continuation stellt den Ausführungszustand eines Programmes dar. Besonders nützlich kann dieses Programmierschema im Bereich von dynamischen Webseiten eingesetzt werden. Stellt man sich das Szenario eines Einkaufs vor, muss meist zunächst ein Produkt gewählt werden, dann wird eine Adresse mit Namen eingegeben, zuletzt werden Informationen zur Zahlungsweise benötigt. Zerlegt man die Anfrage in einzelne Webseiten, müssen diese bei fehlerhaften Eingaben es ermöglichen, dass man vor- und zurücknavigiert, wobei Formularfelder natürlich gleich mit bereits vorgenommenen Eingaben befüllt werden sollten, damit der Anwender diese nicht zweimal eingeben muss. Dies geschieht entweder per Hand, indem man jede Seite mit den Variablen befüllt oder indem man auf dem Server so genannte Continuations einsetzt. Hierbei wird ein Programm durchlaufen (zum Beispiel in JavaScript [78]), welches an den Stellen, an denen Nutzereingaben erforderlich sind, mittels eines speziellen Befehls (*sendPageAndWait*) eine Webseite an den Nutzer sendet, während das Programm an dieser Stelle wartet. Mittels einer Schablone werden Werte in der Webseite

dargestellt. Sobald der Anwender seine Eingaben gemacht hat und das Ergebnis zurücksendet, wird dann auf dem Server an dieselbe Stelle im Programm zurückgesprungen, wobei die Nutzereingaben zur Verfügung stehen. Damit kann ein Entwickler die Nutzereingaben serverseitig wie Eingaben in einem Desktopprogramm behandeln und leichter iterativ vorgehen. Die Verwaltungsarbeit mit Rücksprungadressen wird dabei vom System dem Entwickler zur Verfügung gestellt. Kritiker sehen allerdings Ähnlichkeiten mit dem "GOTO"-Befehl, welcher in der Informatik heutzutage eher abgelehnt wird, da er unübersichtliche Programme ermöglicht beziehungsweise die Ausführungsreihenfolge von Programmen beeinträchtigt. Im Webbereich stellt dieser Ansatz allerdings einen nicht zu unterschätzenden Fortschritt dar, insbesondere wenn man, wie in [36] geschildert, die Schablonen automatisiert generiert. Die Entwicklung von Webanwendungen kann je nach Szenario auf diese Art deutlich beschleunigt werden.

2.7 Speicherung von XML

Bei der Verarbeitung von Inhalten geht es bei jedweder Anwendung immer auch um die persistente Speicherung der Informationen. Für XML eröffnen sich dem Nutzer verschiedene Ansätze: Die direkte Speicherung in Dateien im Dateisystem, die Speicherung mittels eines Versionierungssystems wie *Concurrent Versions System (CVS)* [147] oder Subversion [33], oder die Speicherung der Daten in einer Datenbank. Während erstere Möglichkeit sich durch sehr hohe Geschwindigkeit auszeichnet, bietet die zweite Möglichkeit eine gewisse Sicherheit gegen Datenverlust, ist durch die zusätzliche Verwaltungsarbeit jedoch nicht so schnell. Am langsamsten sind Datenbanken, welche jedoch in Bezug auf Transaktionen, konkurrierenden Zugriff und Dauerhaftigkeit bei weitem mehr Funktionalität aufweisen (die klassischen ACID-Eigenschaften Atomizität, Konsistenz, Isolation und Dauerhaftigkeit [89]).

Für die Speicherung von XML in Datenbanken gibt es verschiedene Ansätze: Zunächst muss zwischen so genanntem narrativem (oder auch dokumentenzentriertem) XML und datenzentriertem XML unterschieden werden. Narratives XML wäre ein in XML geschriebenes Buch oder eine Anleitung, datenzentriertes XML könnte eine Liste von Adressen mit Straße, Hausnummer, Postleitzahl und so weiter sein. Datenzentriertes XML lässt sich sehr leicht in eine klassische relationale Datenbank abbilden, indem man eine entsprechende Tabelle anlegt. Ein Mapping in objektorientierte Datenbanken ist ebenfalls möglich, indem man ähnlich wie bei DOM eine Klassenhierarchie entwirft, welche dann mittels des XML-Dokuments gefüllt wird. Dieses objektrelationale Mapping wird auch *XML data binding* genannt [98]. Für narrative XML-Dokumente existiert eine derartige Möglichkeit nicht, da die Elemente unterschiedlich oft vorkommen, keine durchgehende Strukturierung existiert und die Dokumente sehr irregulär aufgebaut sind. Würden die Dokumente in Tabellen abgebildet werden, gäbe es sehr viele Leereinträge oder unterschiedlich gefüllte Einträge. Daher ist hier der Einsatz nativer XML-Datenbanken wie Xindice [5] oder Tamino der Software AG zu empfehlen. Diese ermöglichen Abfragen über XPath und XQuery [165], können im Gegenzug als Ergebnis nur XML liefern. Will

man nur Teile des Ergebnisses als Eingabe in ein weiterverarbeitendes Programm, ist man gezwungen, dieses XML wieder zu parsen, um sich dann das entsprechende Element heraus zu greifen. Oft wird eine native XML-Datenbank auch in einem System verwendet, welches die Ergebnisse mittels XSL weiterverarbeiten kann. Möglicherweise bietet sich auch ein gemischter Ansatz an: Narrative Dokumente werden in einer anderen Datenbank gespeichert als die stark strukturierten datenzentrierten Informationen. Eine umfangreiche Diskussion der verschiedenen Möglichkeiten finden sich bei [23].

2.8 Trennung von Inhalt und Layout

Als einer der Leitgedanken beim Entwurf von HTML, der auch bei XML immer wieder betont wird, stand die strikte Trennung von Layout und Inhalt. Da es sich um reine Textdokumente handelt, steht der Inhalt im Vordergrund, eine Formatierung findet zunächst nicht statt, nur die Auszeichnung verschiedener Bereiche durch Tags. Mittels dieser Tags wird die umschlossene Information inhaltlich klassifiziert.

Zur Formatierung von XML steht XSL (siehe 2.4) zur Verfügung, für HTML wurden 1994 die *Cascading Stylesheets* vorgeschlagen. Diese Textdokumente mit eigener Syntax werden von HTML-Dokumenten referenziert und enthalten für HTML-Elemente (wie "h1", "h2", "a", "li" und so weiter) Formatierungsanweisungen wie Schriftgröße, Schriftart oder Schriftfarbe. Damit wurden die althergebrachten Formatierungsattribute, welche direkt in den HTML-Dateien die Elemente weiter spezifizierten, abgelöst. Dies sollte mit XHTML verstärkt durchgesetzt werden, weil man feststellte, dass HTML-Elemente zur Layout-Gestaltung zweckentfremdet wurden. Insbesondere kann man dies mit dem Element "Tabelle" beobachten, welches kaum für Tabellendarstellungen von Zahlen- oder Datenreihen sondern fast ausschließlich zur Formatierung und Gestaltung (z.B der Farbe) verwendet wurde. Dies widerspricht der Denkweise von XML und kann zu Schwierigkeiten bei der dynamischen Generation von Webseiten führen.

Jedoch sollte man sich bewusst werden, dass eine inhaltliche Auszeichnung von Text immer auch eine gestalterische Auszeichnung zur Folge hat: Eine Hauptüberschrift ("h1") wird immer mit einer größeren Schrift dargestellt werden als eine Unterüberschrift ("h2"). Ein Paragraf wird immer durch einen Zeilenumbruch von anderen Paragrafen abgesetzt. Deswegen kann die Trennung der Auszeichnung von Inhalt und Layout nie absolut sein, da die inhaltliche Auszeichnung eine gestalterische Auszeichnung impliziert.

2.9 Neue Entwicklungen im Web

XML ist im Internet nicht nur wegen XHTML stark verbreitet, sondern wird auch zum direkten Datenaustausch, zum Beispiel zwischen Firmen, verwendet. Hierfür hat Microsoft 1999 *Simple Object Access Protocol* (SOAP) vorgeschlagen. Beruhend auf einer Übertragung per HTTP sollen Funktionsaufrufe in XML verpackt werden; die Antwort erfolgt ebenfalls in XML. Eine Beispielanwendung hierfür könnte ein Mehrwertsteuer-

rechner sein, welcher für einen gegebenen Preis die nötige Mehrwertsteuer berechnet. Einen solchen Service nennt man Webservice. Zum Auffinden des Webservices ist die Unterstützung eines Verzeichnisdienstes vonnöten wie *Universal Description, Discovery and Integration* (UDDI) und zur Spezifikation der vorgehaltenen Methoden einschließlich Daten und Datentypen wird die *Web Services Description Language* (WSDL) [164] benutzt. Insgesamt ist der Einsatz von Webservices also relativ komplex trotz inzwischen vermehrter Unterstützung durch Entwicklungsumgebungen.

Ein einfacherer Ansatz ist der weitere Vorschlag von Microsoft, einen so genannten XMLHttpRequest einzuführen. Diese als ActiveX-Plugin für Internet Explorer 5.0 eingeführte Software vermag über JavaScript gesteuert asynchron XML-Anfragen über HTTP an einen Webserver abzusetzen und die Antwort über Skripting in das angezeigte XHTML-Dokument einzufügen. Diese Technik wird heute auch Ajax (*asynchronous JavaScript and XML*) genannt. Seit diese Technik auch von anderen Browsern als dem Internet Explorer unterstützt wird und Google mit seinem kostenlosen, webbasierten Emaildienst GMail und auch Google Maps davon Gebrauch macht, hat sie zunehmende Verbreitung gefunden, da Webseiten mit mehr Interaktivität ermöglicht werden. Ein Beispiel hierfür ist die direkte Anzeige von gefundenen Begriffen in einem Formular, ohne dass der Anwender die Anfrage hätte senden müssen. XMLHttpRequest liegt inzwischen dem W3C zur Standardisierung vor [167].

Ein weiterer starker Wandel im Web betrifft die Entstehung von so genannten Communities, welche meist an Plattformen gebunden sind. Als wichtigster Vertreter ist hier Wikipedia, gegründet von Larry Sanger und Jimmy Wales (www.wikipedia.org), zu nennen, ein durch Nutzer erstelltes Online-Lexikon zur freien Benutzung, welches qualitativ mit bekannten Enzyklopädien wie Brockhaus und Britannica zumindest mithalten kann [86]. Wikipedia beruht auf der Idee des Wikis, einer Webseite, die jedem Besucher das Lesen aber auch die Änderung und Erweiterung der Inhalte erlaubt [87]. Ein wesentlicher Bestandteil dieser Idee ist auch, dass HTML noch zu kompliziert ist und weiter vereinfacht werden muss. Für Wikis existiert deshalb eine weiter vereinfachte Syntax oder ein webbasierter WYSIWIG-("What You See Is What You Get")-Editor, welcher die Eingaben erleichtert. Gegen Vandalismus und Fehlinformationen gibt es keinen besonderen Schutz außer der Aufmerksamkeit anderer Besucher, welche Fehler korrigieren können und sollen, und der bisherigen Erfahrung nach auch tun.

Diese und weitere Entwicklungen, wie die zunehmende Verfügbarkeit von Breitbandzugängen oft zu Festpreisen und damit vereinfachte Distribution auch großer Mengen von Daten oder das immer mehr um sich greifende "Bloggen" (Führen eines Webtagebuchs - *web log*), Plattformen zum Tausch von Videos wie bei Youtube oder bei Google ([170] [52]) veranlassten Tim O'Reilly zurückgehend auf einen Vorschlag von Dale Dougherty, Vizepräsident des O'Reilly-Verlags [110], den Begriff des Web 2.0 zu schaffen.

2.10 Zusammenfassung und Ausblick

Wenn auch heutzutage das W3C Probleme hat, sich mit seinen HTML-Standards durchzusetzen [24], so kann man trotzdem sagen, dass XML inzwischen ein etabliertes Format zur Verwaltung und zum Austausch von Daten ist. Hierbei kann man sogar oft beobachten, dass Webseiten, welche XML-basierte Werkzeuge zur Aufbereitung ihrer Inhalte benutzen, zwar XML, jedoch kein korrektes XHTML einsetzen. Diese Entwicklung zeigt den großen Erfolg von XML. Mit den neuen Entwicklungen um das Web 2.0, welche im Wesentlichen durch XML als Datenaustauschformat im Zusammenspiel mit dem browserbasierten JavaScript eine neue Interaktivität und Reaktivität bis hin zu browserbasierten Anwendungen wie Textverarbeitungsprogramme oder Spiele ermöglicht.

Kollaborative Aspekte, wie sie erfolgreich bei der Internet-Enzyklopädie Wikipedia oder bei anderen Mitmach-Webseiten wie Youtube gezeigt werden, stehen dabei auch unter dem Schlagwort Web 2.0 für eine neue Generation von Internetanwendungen, welche mehr und mehr standardmäßig von den Anwendern genutzt werden und langsam beginnen, sich einen festen Platz im Leben der Anwender zu erobern.

XML wird weiterhin bestimmend sein bei der Vorhaltung und Aufbereitung von Informationen, die dabei erreichte Trennung von Inhalt und Layout hilft in vielen Anwendungsbereichen. Die Flexibilität beim Einsatz von XML-Parsern und das leichte Übersetzen von XML in verschiedene XML-Dialekte mittels XSLT führt auch dazu, dass XML-Formate vermehrt Komma-separierte Textdateien zum Datenaustausch ablösen. XML ist somit für viele aktuelle und neue Webseiten das Datenformat der Wahl.

repetitio est mater studiorum

3.1 Einleitung

Bei der Gestaltung eines Lehr- und Lernsystems im Internet ist es sinnvoll, vorher die theoretischen Grundlagen des Lernens darzulegen, die verschiedenen Lehr- und Lernsysteme kennen zu lernen und einzuordnen sowie vergleichbare Ansätze zu erörtern. Des Weiteren wird in diesem Kapitel ein Überblick über verschiedene Ansätze für Lehr- und Lernsysteme gegeben.

3.2 Lerntheorien

Es werden in der Literatur vor allem drei Lerntheorien unterschieden: der Behaviorismus, der Kognitivismus und der Konstruktivismus. Sie unterscheiden sich grundsätzlich in der Art und Weise, wie das Wissen erarbeitet wird, bezüglich der Lernziele, sowie in der Art und Weise, wie der Lehrer auftritt und wie dem Lernenden Rückmeldungen gegeben werden. Einen sehr guten Überblick dazu bietet Blumstengel [22] in ihrer Dissertation (siehe Tabelle 3.2). Sie hat dazu die von Baumgartner und Payr entwickelte Aufstellung [18] weiterentwickelt.

Behaviourismus wurde von Watson im Jahre 1919 vorgeschlagen, der in seiner Schrift "*Psychology as the Behaviorist views it*" ähnlich zu den Pawlow-Versuchen den Menschen als Reiz-Reaktions-Maschine interpretierte. Der Mensch wird in dieser Theorie als Black-Box gesehen, man untersucht alle von außen (oder innen) zugefügten Reize und ihre Reaktionen darauf. Der Verzicht auf eine - für unmöglich gehaltene - Betrachtung

Kategorie	Behaviourismus	Kognitivismus	Konstruktivismus
Hirn ist ein	Passiver Behälter	Informations- verarbeitendes "Gerät"	Informationell ge- schlossenes System
Wissen wird	Abgelagert	Verarbeitet	Konstruiert
Wissen ist	Eine korrekte Input- Output-Relation	Ein adäquater interner Verarbeitungsprozess	Mit einer Situation operieren zu können
Lernziele	Richtige Antworten	Richtige Methoden zur Antwortfindung	Komplexe Situa- tionen bewältigen
Paradigma	Reiz-Reaktion	Problemlösung	Konstruktion
Strategie	Lehren	Beobachten und helfen	Kooperieren
Lehrer ist	Autorität	Tutor	Coach, Trainer
Feedback	Extern vorgegeben	Extern modelliert	Intern modelliert
Interaktion	Starr vorgegeben	Dynamisch in Ab- hängigkeit des ex- ternen Lernmodells	Selbstreferenziell, zir- kulär, strukturde- terminiert (autonom)
Programm- merkmale	Starrer Ablauf, quantitative Zeit- und Ant- wortstatistik	Dynamisch gesteuerter Ablauf, vorgegebene Problemstellung, Antwortanalyse	Dynamisch, komplex vernetzte Systeme, keine vorgegebene Problemstellung
Software- Paradigma	Lernmaschine	Künstliche Intelligenz	Sozio-technische Umgebungen
"idealer" Soft- waretypus	Tutorielle Systeme, Drill & Practice	Adaptive Systeme, ITS	Simulationen, Mikro- welten, Hypermedia

Tabelle 3.1 Vergleich von Lerntheorien von Blumstengel

der inneren Denk- und Fühlvorgänge sollte eine objektivere Sicht auf den Menschen ermöglichen. Skinner erweiterte diese Ansichten 1954 um spontanes Verhalten, welches auch auftreten könne, und welches er durch Verstärker konditionieren wollte. Bekannt wurde vor allem seine Skinnerbox, in welcher er mittels Belohnung kleinster Lernerfolge Ratten und Tauben verschiedene Verhaltensweisen beibrachte wie etwa Klavier oder Tischtennis spielen.

Der Behaviorismus wurde ab dem zweiten Weltkrieg und zunehmend in den sechziger und siebziger Jahren vom Kognitivismus abgelöst. Dieser bezieht in die Betrachtung des Lernens vermehrt die inneren Vorgänge im Menschen ein. Sie werden analog zu den aufkommenden Computern als Aufnahme von Eingabereizen, ihrer Verarbeitung und anschließender Reaktion betrachtet. Jean Piaget definierte die zwei Verhaltensweisen *Akkommodation* und *Assimilation*. Der Mensch handelt nach Schemata, welche durch Akkommodation an die Umwelt angepasst werden, durch Assimilation jedoch die Umwelt verändern (also die Umwelt an das Schema anpassen).

Die dritte Lerntheorie ist aus dem Konstruktivismus abgeleitet. Der Konstruktivismus geht davon aus, dass ein Lerner sich seine eigene Darstellung der Welt schafft. Hierfür rekonstruiert, konstruiert und dekonstruiert er die Welt [119]. Da die Konstruktionen mit

dem Handeln des Lernenden verknüpft sind, ist es am besten, wenn man dem Lernenden größtmögliche Freiheit beim Lernen lässt, da jeder selbst am besten weiß, wie er lernt. Geschichtlich gesehen ist der Konstruktivismus schon sehr alt, der Begriff lässt sich auf den italienischen Philosophen Giambattista Vico zurückführen. Auch die Methode *Lernen durch Lehren* kann als dem Konstruktivismus zugehörig bewertet werden.

Heutzutage wird Lernen auch über die Lernsituation definiert. Hier steht die klassische Konditionierung entsprechend dem Behaviourismus neben dem Lernen am Modell (kognitives Lernen), bei welchem man in einer Situation bei einem Vorbild abschaut. Dieses Vorbild kann die Person sein (Identifikationslernen) oder der Vorgang (Imitationslernen). Hinzu kommt die Erkenntnis, dass Lernen und Anwenden nicht unbedingt in Einklang steht. Mögen in gewissen Situationen wie beispielsweise im Unterricht Handlungen oder Wissen korrekt verstanden und wiedergegeben werden, so mag es in einer anderen Situation doch dazu kommen, dass das Gelernte überhaupt nicht zur Anwendung kommt, weil die Situation nicht mehr übereinstimmt.

Das Lernen und insbesondere das optimierte Lernen ist immer wieder Ziel von Pädagogen und Psychologen, es werden auch heute noch ständig neue Methoden zum besseren oder schnelleren Lernen vorgeschlagen.

3.3 Geschichte neuer Techniken zum Lernen

Schon immer wurden neue Techniken und insbesondere auch neue Medien umgehend auf ihre Verwendung zum Lernen untersucht und meist Anwendungsszenarien dafür vorgeschlagen. So hat Friedrich Schiller in seiner Rede *Die Schaubühne als eine moralische Anstalt betrachtet* vorgeschlagen [123], Theater unter anderem als ein Instrument der Aufklärung zu sehen und damit als Bildungseinrichtung zu nutzen. Das Medium Film dient ebenfalls als Bildungsmedium. Dies schlägt sich in Einrichtungen wie den Landes- und Kreisbildstellen nieder, welche Lehrfilme (zumeist im Schulunterricht für die Fächer Biologie und Erdkunde verwandt) verleihen. Für das Radio als Lehreinrichtung warb Bertolt Brecht in seiner Schrift *Der Rundfunk als Kommunikationsapparat* (1922/1923) [25], in der er das Radio allerdings zu einem Kommunikationsgerät ausbauen wollte. Jeder sollte Radiosendungen verbreiten können. Das Fernsehen wurde neben der Unterhaltung auch für Unterrichtssendungen wie das Telekolleg (Start: 1968) genutzt, das heute noch unter dem Namen Telekolleg multimedial fortgeführt wird. In allen Schulen und Bildungseinrichtungen finden sich Fernseher und Videorekorder, um Lehrfilme abspielen zu können. Der Computer als technische Neuerung ging und geht dabei den gleichen Weg.

Es gibt viele Utopien und Visionen zu technischen Entwicklungen, eine davon ist die Utopie der universellen Bibliothek, in welcher das Wissen der Menschheit komplett gespeichert ist. Sie fand auch Mitte des zwanzigsten Jahrhunderts ihren Fortgang beispielsweise in Memex, dem *Memory Extender* von Vannevar Bush [28], welcher als im Schreibtisch integrierte Maschine über Mikrofilme alles Wissen vorzuhalten und durch Verlinkung (*associations*) schneller abrufbar machen sollte. Diese Idee spiegelte sich im Docuverse von Ted Nelson wider, das er im Rahmen seines Projektes Xanadu schaffen

wollte [106]. Er schlug wesentliche Elemente des späteren Web vor: Dokumente mit eindeutiger Adresse, gespeichert in einem globalen Speicher. Verweise zwischen den Dokumenten wurden bidirektional abgespeichert, was Konsistenz garantieren sollte, zugleich aber auch verhindern sollte, Dokumente unnötigerweise zu kopieren. So genannte Transklusionen ermöglichen das Zitieren anderer Werke, wobei der Zugriff darauf mittels Zahlen kleinster Beiträge an die Autoren vergütet werden sollte. Xanadu scheiterte auch bei einem zweiten Anlauf in den achtziger Jahren, wahrscheinlich vor allem an seiner Komplexität. Das WWW kann als die Implementation einer Teilmenge des Xanadu-Projektes gesehen werden.

Seymour Papert entwickelte die funktionale Programmiersprache Logo 1967, um die Wirksamkeit seiner Theorie vom Konstruktivismus, eine Art erweiterter Konstruktivismus, zu demonstrieren [114]. Menschen beziehungsweise Kinder sollen etwas lernen, indem sie Sachen zusammen bauen - eben konstruieren. Logo ist Lisp sehr ähnlich und auch Turing-vollständig [70] [71] [72]. Bekannt wurde es vor allem durch seine Schildkrötengrafiken (*turtle graphics*), mit denen man durch wiederholtes Aufrufen von einfachen Grundfunktionen wie Box oder Vielecke als Basis komplexere Zeichnungen erstellen beziehungsweise konstruieren konnte [90]. Papert bereitete damit vielen anderen Ideen den Weg. Hier ist Alan Key mit seinem Dynabook zu nennen, welches er Ende der sechziger Jahre bei Xerox Parc vorschlug. Seine Idee, die man sich als modernes Notebook vorstellen mag, umfasste jedoch nicht nur den Computer als tragbares Gerät sondern auch die Software, die es dem Benutzer auf besonders intuitive Art und Weise ermöglichen sollte, das Gerät zu bedienen und damit letztendlich zu programmieren. Kay sagt dazu in seiner kurzen Geschichte über Smalltalk: "*Doing with Images makes Symbols*", was er in Anlehnung an Piaget und Bruner verstanden wissen will [81]. Hiermit meint er, dass der Fortschritt beim Lernen durch das Bewegen und Bearbeiten von dynamischen Bildern zu einer symbolischen Repräsentation von Ideen führt.

Einen wesentlichen Beitrag zur Entwicklung von Vorläufern des Web hat Douglas Engelbart, der vor allem als Erfinder der Computermaus bekannt wurde, mit seinem so genannten Online-System (NLS) geleistet, welches er Ende 1968 der Öffentlichkeit präsentierte [42]. Inspiriert von den Memex-Ideen von Bush schuf er ein ganzes System aus Soft- und Hardware. Die Daten wurden auf einem zentralen Server abgelegt, bearbeitet wurden sie über mehrere Terminals, welche über Rastermonitore, eine Drei-Tasten-Maus und eine spezielle, ebenfalls von Engelbart entworfene Tastatur mit fünf Tasten verfügten. Dieses System bot auch kollaborative Ansätze in Form eines von mehreren Autoren gemeinsam genutzten Journals. Allerdings war die Bedienung nicht intuitiv, was der Akzeptanz des Systems abträglich war.

Mit HyperCard schuf Bill Atkinson 1987 eine bekannte Anwendung für den Macintosh-Computer von Apple. Sie wurde von vielen Leute als eine der wichtigsten Vorläuferanwendungen des Web angesehen [124] [107], und erfreute sich wegen ihrer Einfachheit großer Beliebtheit. HyperCard kann als Karteikartenanwendung gesehen werden, auf denen GUI-Elemente wie Texte, Köpfe oder Bilder gespeichert werden können. Die Karten sind als Stapel (*stacks*) gespeichert. Sie können über Links, die allerdings nicht textgebunden sind, verknüpft werden. Neben der Funktionalität Datenbank, die Anwendungen

wie Adressbücher oder Serienbriefe möglich machte, gibt es zusätzlich eine einfache Programmiersprache namens Hypertext, mit welcher die Karteikarten programmiert werden. Programme können als Text in einer Art Basic erstellt werden. Externe Programme können über eine Plugin-Schnittstelle eingebunden werden.

Es gibt viele weitere Anwendungen im Bereich Hypertext und Hypermedia, die mit dem Aufkommen des erfolgreichen Web meist verschwanden oder darin aufgingen. Zu nennen sind HyperTIES von Shneiderman [127], entwickelt im Jahr 1983, NoteCards von Xerox Parc (1985) [61], ein Vorläufer von HyperCard, und Intermedia von van Dam aus dem Jahr 1985 [169]. Für viele weitere Entwicklungen sei auf Schulmeister [124] verwiesen.

3.4 Lehr- und Lernsysteme

Neben den in Abschnitt 3.3 vorgestellten Systemen zur Wissensaufbereitung und Wissensverwaltung gibt es schon länger Versuche, reine Lernsoftware zu erstellen. Ein unter anderem von NLS inspiriertes System ist PLATO (*Programmed Logic for Automated Teaching Operations*) von Donald Bitzer, der für die Erfindung von Plasma-Bildschirmen bekannt ist [158]. 1960 wurde PLATO I an der Universität von Illinois vorgestellt, es lief auf einem Großrechner. Bereits 1961 wurde mit PLATO II eine Weiterentwicklung eingeführt, welche zwei Benutzern gleichzeitig die Arbeit beziehungsweise das Lernen am Rechner ermöglichte. Bis 1966 wurde PLATO überarbeitet und erlaubte in der dritten Version Benutzern die Erstellung eigener Lektionen unter Verwendung der eigens entwickelten Programmiersprache TUTOR [126]. Durch die Verwendung von Terminals konnte sich die Nutzergemeinde mit neu geschaffenen Programmen wie *Personal Notes* Emails schicken, miteinander online kommunizieren (*Talkomatic*) sowie Sofortnachrichten austauschen (*instant messaging*) oder sich gegenseitig Einsicht in die Bildschirm Inhalte gewähren (*remote screen sharing*). Plato war damit einer der Vorläufer von Lotus Notes, die bekannteste und zeitweise am weitesten verbreitete Groupware (Software für Gruppenarbeit). Die ersten Versionen von PLATO waren vor allem erfolgreich, weil Bitzer rein behaviouristische Lernmethoden umsetzte (*drill & practice*) und auf moderne kognitive Ansätze verzichtete, was die technische Umsetzbarkeit erleichterte.

TICCIT (*Time-shared, Interactive, Computer-Controlled Information Television*) wurde von der Mitre Corporation entwickelt und basierte auf Merrills Component-Display-Theorie [100]. Sie stellte Lerneinheiten auf die drei Arten Regel, Beispiel und Übung dar. Der Lernende konnte seinen Schwierigkeitsgrad einstellen sowie Hilfe anfordern. PLATO wie TICCIT waren sehr teure Einrichtungen, alleine die zum Betrieb nötigen Großrechner kosteten mehrere Millionen US-Dollar. Die Erstellung der Lektionen war ebenfalls sehr teuer, die eigentlich erwartete Wiederverwendbarkeit nicht im gewünschten Maße vorhanden [31].

Ab Ende der siebziger Jahre gab es an vielen Universitäten Initiativen zur Erstellung von E-Learning-Systemen. Wichtige hierbei sind Cyclops an der Open University in Großbritannien (1976); das Projekt Athena des Massachusetts Institute of Technology

(MIT) (1983); Toolbook der Firma Asymetrix (1984), welche später in Click2Learn umbenannt wurde (heute Sumtotal); das Projekt NEOS (MIT, 1986, *Networked Educational Online System*); und Hyper-G, entwickelt 1989 an der Universität Graz. Ab den neunziger Jahren zeigte sich eine zunehmende Kommerzialisierung des Marktes, teilweise wurden viele Systeme nicht mehr frei verteilt, da Spinoffs, welche als Ausgründungen von Universitätsprojekten entstanden, die Software kommerziell vertrieben und weiterentwickelten. Beispiele hierfür sind WebCT, welches 1995 von Murray Goldberg an der Universität von British Columbia entwickelt und 1997 als Firma ausgegründet wurde. Ein weiteres Beispiel ist Hyper-G, das von 1996 bis zum Konkurs im Februar 2006 von der Firma HyperWave weiterentwickelt wurde. Blackboard, eigentlich eine Beratungsfirma, begann nach der Fusion mit CourseInfo 1998 mit dem Verkauf von E-Learning-Software. Nach mehreren anderen Zukäufen im Markt für Lernsoftware hat Blackboard im Frühjahr 2006 auch WebCT übernommen. Um die teils horrenden Lizenzkosten zu vermeiden, die beim Einsatz von kommerzieller Lernsoftware oft allein dadurch entstehen, dass pro Nutzer (also an Universitäten pro Student) eine jährliche Lizenz-Gebühr fällig wird, haben sich verschiedene Open-Source-Initiativen entwickelt. Moodle (*Modular Object-Oriented Dynamic Learning Environment*), 1999 von Martin Dougiamas gestartet, ist kostenlos erhältlich und erfreut sich einer zunehmend großen Nutzerbasis. Dougiamas hat bei seinem System als ehemaliger Administrator von WebCT an der australischen Curtin University einige pädagogische Prinzipien verwirklicht, die es so in einem Lernmanagementsystem noch nicht gab. Studenten können in Moodle ihre Rollen tauschen, also beispielsweise von der Rolle Schüler in die Rolle Lehrer wechseln. Ilias, ein Lernmanagementsystem der Universität Köln, sowie Metacoön der Bauhaus-Universität Weimar sind weitere bekannte Open-Source-Projekte neueren Ursprungs. Eine umfangreiche Übersicht über Lernsysteme findet man auf Wikipedia unter [155].

3.5 Begriffe und Abkürzungen

Bei der Betrachtung einer Anwendung aus dem Bereich des E-Learning ist es nützlich, sich zunächst über die Bedeutung der Worte klar zu werden. Unter *E-Learning* versteht man das Lernen unter Einsatz des Computers, sei es zur Verteilung von digitalen Medien, also der Lernmaterialien, sei es für die Präsentation der Medien oder sei es bei der Kommunikation während des Lernvorgangs.

E-Learning-Systeme oder Lernmanagementsysteme (LMS) präsentieren also digitale Inhalte. Geschieht das nur lokal, so spricht man von *Computer based training* (CBT) oder *Computer aided training* (CAT), werden die Inhalte über das Web angeboten, so nennt man es *Web based training* (WBT). Weitere Abkürzungen mögen das Wort Training durch *teaching*, *tutoring*, *instruction*, *education* oder *learning* ersetzen. Genauso wird *based* gerne durch *support* oder *managed* ersetzt. Im Grunde sind hier neue Abkürzungen eher als Aufmerksamkeitserreger denn als inhaltliche oder technische Neuerung zu sehen. Weiterhin werden gerne die Adjektive kooperativ und kollaborativ (*cooperative* und *collaborative*) verwendet. Diese Unterscheidung ist eine eher spitzfindige, zumal die

deutsche Übersetzung beide Male "gemeinschaftlich", "zusammen" oder "in Gruppen" lautet. Manchmal wird dabei auf den Unterschied des lokalen Zusammenarbeitens oder des Zusammenarbeitens über das Netz angespielt, dies wird jedoch nicht durchgängig verwendet. Eine äußerst differenzierte, jedoch nicht klärende Auflistung ist unter [152] zu finden.

Eine neuere Wortschöpfung im Bereich des E-Learning ist das *blended learning* (integriertes Lernen). Sie drückt einerseits aus, dass man durch Einsatz von E-Learning trotzdem auf Präsenzveranstaltungen nicht verzichten kann, und wird verdeutlicht, dass E-Learning eher eine ergänzende denn eine ausschließliche Komponente im Lernprozess ist.

3.6 Kategorisierung von Lernmanagementsystemen

Bei der Fülle an vergangenen und aktuellen Projekten ist es sinnvoll, die Kerneigenschaften eines Lernmanagementsystems darzustellen, sowie die verschiedenen Aufgabengebiete, welche ein Lernmanagementsystem abdeckt, zu erläutern.

Ein Lernmanagementsystem besteht aus verschiedenen Bereichen, welche durchaus auch als Sammlung von Diensten (*services*) gesehen werden können. Diese werden in der Kombination als Lernmanagementsystem gesehen. Folgende wichtige Elemente sind oft Teil eines Lernmanagementsystem (zusammengetragen und aufbereitet nach [65] [124] [154]):

- Autorensystem: interne oder externe Werkzeuge und Programme, welche für die Erstellung der Lerninhalte nötig sind
- Verwaltung: einerseits Nutzerverwaltung, andererseits Kurs- und Inthalteverwaltung, dazu gehört eine Rechteverwaltung und die Abwicklung der Kurse (wie Kurs, Übung, Test) und Auswertung
- Kommunikation: ein Lernmanagementsystem bietet oft verschiedene Elemente zur synchronen und asynchronen Kommunikation wie News, Forum, Chat, Instant Messaging, Whiteboard oder auch Videokonferenz
- Datenvorhaltung: das Lernmanagementsystem verwaltet die gespeicherten Inhalte dauerhaft, meist bietet es hierzu ergänzend Versionierung oder ähnliches an
- Sonstige Werkzeuge: zum Beispiel eine integrierte Suche, persönliche Daten wie Notizen, Export der Inhalte in Formate wie SCORM (siehe Abschnitt 3.7, Internationalisierung, Tracking von Nutzeraktivitäten und so weiter)

Diese Übersicht erhebt keinen Anspruch auf Vollständigkeit, tatsächlich implementieren Lernmanagementsysteme nicht unbedingt alle Bereiche. Dies gilt insbesondere für die Autorenwerkzeuge, welche oftmals nicht integriert sind, sondern als eigenständige Software verteilt werden. Eine noch losere Kopplung besteht, wenn das Lernmanagementsystem keine Unterstützung zum Erstellen von Dokumenten anbietet, sondern nur

den Import von Dateien bestimmter Formate, wie PDF-Dateien oder Dateien, welche mit Microsoft Office oder OpenOffice erstellt werden. Dies kann durchaus im Sinne der Autoren sein, ist die Einarbeitung der Autoren doch ein wesentlicher Kostenfaktor für Autorenssysteme [124].

Ein weiterer Aspekt in diesem Zusammenhang ist, dass je nach verwendeter Basistechnologie häufig bereits vorhandene Teilkomponenten modulartig hinzugefügt werden können. Ein Beispiel hierfür ist die Funktionalität "Forum". Es gibt diverse Implementierungen von Foren auf Basis von PHP und der Datenbank MySQL, so dass z.B. bei Moodle, einem Lernmanagementsystem auf Basis von PHP, mit relativ wenig Aufwand ein Forum hinzugefügt werden kann. Mit wenigen Skripten ist eine gemeinsame Nutzerverwaltung erreicht, die automatisierte Erstellung von Diskussionsbereichen für neu angelegte Kurse erreicht man ebenfalls mit Skripten.

Die Suche kann ebenfalls über Suchmaschinen wie Google, Yahoo oder MSN Search abgewickelt werden, dafür wird der normalen Suchanfrage eine Einschränkung auf die Seite, unter welcher das Lernmanagementsystem verfügbar ist, hinzugefügt. Allerdings erweist sich das Ranking von mitgelieferten internen Suchmaschinen oft als besser, da diese besseren Zugriff auf die Metadaten haben.

Die Ausgestaltung der Lerninhalte variiert ebenfalls erheblich. Während einfache Systeme sich auf die Vorhaltung von statischen Inhalten wie Texten, Bildern oder Filmen beschränken, unterstützen neuere Systeme zumeist mindestens eingeschränkt die Metadatenstandards LOM und SCORM (siehe Abschnitt 3.7), womit die Lernobjekte besser wieder verwendbar werden. Betrachtet man Lernobjekte von höherer Interaktivität wie das Exploratories-Projekt der Brown University [87], oder das ILO-Projekt am GRIS der Universität Tübingen mit hoch interaktiven Applets [66], so stellt man zwar eine bessere Interaktivität für den Lerner fest, jedoch gleichzeitig eine weniger enge Integration der Lernobjekte in das darunter liegende Lernmanagementsystem.

Umfassende Evaluationen bestehender Lernsysteme haben beispielsweise Schulmeister [125] oder Baumgartner [15] [16] [17] gemacht. Die dabei entworfenen Kriterienkataloge sind äußerst umfangreich. Sie erstrecken sich analog zu den oben genannten Bereichen eines Lernmanagementsystems von Bewertungen der Kommunikationsmöglichkeiten über die Ausgestaltung der Administration, hin zu den didaktischen Möglichkeiten, die Lerneinheiten zu gestalten, und weiter zu den technischen Grundlagen des Systems, der Dokumentation, den Preisen und so weiter. Eine relativ umfangreiche Vergleichsseite im Internet, welche allerdings nur ausgewählte Lernmanagementsysteme vergleicht - dafür aber auch verschiedene Versionen der Lernmanagementsysteme - ist unter [39] zu finden. Diese Kriterien könnten analog zur Bewertung von Content-Managementssystemen wie bei [96] noch sehr viel detaillierter ausfallen.

3.7 Standards für Lernobjekte

Als *Lernobjekt* bezeichnet man kleinste Teile eines Kurses, welcher einen bestimmten Inhalt vermittelt. Dies mag eine Seite mit Text sein, ein Film, eine Grafik oder ähn-

liches. Mehrere solcher Lernobjekte kann man zu Lerneinheiten und schließlich Kursen gruppieren. Kann man Lernobjekte in ein Lernmanagementsystem exportieren und importieren, steigert man die Wiederverwendbarkeit der geschaffenen Inhalte und spart somit Geld, da die Inhalte in der Regel sehr teuer sind [124].

3.7.1 Learning Object Metadata (LOM)

Der Standard *Learning Object Metadata* (LOM) ist 2002 von der IEEE-Organisation unter der Beteiligung des *IMS Global Learning Consortium* entstanden. Das *IMS Global Learning Consortium* hat LOM seinerseits in seiner Spezifikation der *Learning Resource Meta-data* (IMS LRM) verwendet. LOM entspricht also einerseits dem IEEE-Standard 1484.12.1 und gleichzeitig der IMS-Spezifikation 1.3. LOM wurde entworfen, um Lernobjekte auszeichnen zu können. Lernobjekte werden dabei als "jegliches digitale oder nicht digitale Gebilde, welches für das Lernen, die Lehre oder zur Schulung verwendet werden kann" [14] definiert.

LOM kategorisiert Lernobjekte in neun Klassen [76]: *General Category* (allgemeine Informationen zum Lernobjekt), *Lifecycle Category* (Geschichte und Zustand des Lernobjekts, Abhängigkeiten von anderen Lernobjekten), *Meta-Metadata Category* (Metadaten zu den vorhandenen Metadaten), *Technical Category* (technische Einzelheiten), *Educational Category* (Merkmale zur pädagogischen Beschreibung), *Rights Category* (Copyright-Informationen und Nutzungsbedingungen), *Relation Category* (Relation zu anderen Lernobjekten), *Annotation Category* (Anmerkungen zum Nutzen des Lernobjekts, Kommentare können eingeordnet werden), *Classification Category* (Klassifikationsmöglichkeiten). Die aufgeführten Kategorien sind hierarchisch weiter unterteilt, wiederholt auftretende Knoten sind dabei kontextabhängig zu interpretieren.

```
1 <lom>
2   <general>
3     < identifier >
4       <catalog>URI</catalog>
5       <entry>http://www.gris.uni-tuebingen.de/content/CO_01</entry>
6     </ identifier >
7     < title >
8       <string language="de">Titel des Lernobjekts</ string >
9     </ title >
10    <language>de</language>
11    <description >
12      <string language="de">Textuelle Beschreibung</ string >
13    </ description >
14    <keyword>
15      <string language="de">Lernobjekt</ string >
16    </keyword>
17    <coverage>
```

```
18 <string language="de">2006, Deutschland</ string >
19 </coverage>
20 < structure >
21 < structure >LOMv1.0</structure>
22 <value>atomic</value>
23 </ structure >
24 <aggregationLevel>
25 <source>LOMv1.0</source>
26 <value>2</value>
27 </ aggregationLevel >
28 </general>
29 </lom>
```

Listing 3.1 Ein Beispiel einer LOM-Beschreibung

Die Kategorisierung ermöglicht somit effizientes Suchen und die Einordnung des Nutzens der Lernobjekte für die Lernziele. Die IEEE-Arbeitsgruppe für LOM arbeitet für die Umsetzung des Standards auch mit der Dublin Core - Initiative zusammen (siehe Abschnitt 2.4).

3.7.2 Sharable Content Object Reference Model (SCORM)

Das *Sharable Content Object Reference Model* SCORM ist ein Standard, welcher von der *Advanced Distributed Learning* - Initiative (ADL), eine vom US-Verteidigungsministerium 1997 gegründete Einrichtung, getragen wird. Dabei sind wichtige weitere Organisationen an der Erstellung beteiligt. Wichtigste Partnerorganisationen hierbei sind die folgenden Einrichtungen:

- Alliance of Remote Instructional Authoring & Distribution Networks for Europe (ARIADNE)
- Aviation Industry CBT Committee (AICC)
- Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Committee (LTSC)
- Instructional Management System (IMS) Global Learning Consortium, Inc.

Aktuell wird an SCORM 2004 gearbeitet, inzwischen ist der Standard bei der dritten Edition angelangt. SCORM 2004 setzt sich aus drei Büchern zusammen: (*Content Aggregation Model* (CAM), *Run-Time Environment* (RTE) und *Sequencing and Navigation* (SN)). SCORM basiert auf verschiedenen Technologien der beteiligten Mitglieder. Die wichtigsten verwendeten Technologien sind dabei:

- IEEE Data Model For Content Object Communication
- IEEE ECMAScript Application Programming Interface for Content to Runtime Services Communication
- IEEE Learning Object Metadata (LOM)
- IEEE Extensible Markup Language (XML) Schema Binding for Learning Object Metadata Data Model
- IMS Content Packaging
- IMS Simple Sequencing

SCORM definiert im CAM als kleinste Einheit von Lernobjekten so genannte *assets*, welche verschiedene Ausprägungen haben können: eine Webseite, eine Javascript-Funktion, ein HTML-Fragment, eine Bild, ein Flash-Objekt, eine Musikdatei usw. Mehrere dieser Assets können zu einem *Sharable Content Object* (SCO) gruppiert werden [1]. Eine Organisationsstruktur legt die Hierarchie so genannter Einheiten (*items*) fest. Die Einheiten können wiederum auf Assets oder die SCOs verweisen (siehe Bild 3.1). Zusammen angeordnet und gruppiert ergeben sie ein Paket (*package*). Ein Paket hat eine Dateistruktur und enthält so genannte Manifest-Dateien, welche eine Art Inventarliste des Pakets darstellt. Die Manifest-Dateien sind XML-Dokumente. Das Paket selbst wird als Zip-Datei komprimiert.

Im SCORM RTE werden die drei Teilbereiche *Launch*, *API* und *Data Model* definiert [2]. Während der Launch-Prozess beschreibt, wie SCOs gestartet werden und die Kommunikation zwischen dem Lernmanagementsystem und dem SCO abläuft, geschieht dies unter Verwendung der API. Diese kann Zustände des SCO (initialisiert, beendet, Fehler) und Daten zwischen SCO und Lernmanagementsystem austauschen und speichern. Das dazu verwendete Datenmodell enthält alle nötigen Felder, um Ergebnisse oder Lernschritte der absolvierten Lerneinheiten abzuspeichern. Da die Idee des SCORM die Wiederverwendung der Lernobjekte ist, müssen alle Daten, welche gespeichert werden sollen, in der SCO-Beschreibung vorhanden sein.

Das SCORM SN ist der neueste Teil des SCORM-Standards und basiert auf dem IMS Simple Sequencing. SCORM SN erzeugt eine Sequenzialisierung der Lerninhalte durch Einordnung der SCOs in einen so genannten Aktivitätsbaum (*activity tree*). Dieser entspricht der hierarchischen Ordnung des CAM. Durch Cluster können zwei Ebenen des Aktivitätsbaums (immer ein Elternknoten mit seinen Kindknoten) gruppiert werden. Über Vorbedingungen können dann Aktionen mit und ohne Nachbedingungen ausgeführt werden. Weiterhin werden die Aktivitäten im Lernmanagementsystem nachverfolgt. Dies entspricht der Erstellung und Verfolgung von Lernpfaden der klassischen Didaktik.

Der SCORM-Standard ist mit LOM zwar der wichtigste Standard im Bereich des E-Learning und wird von den maßgeblichen Institutionen mitgetragen, spiegelt jedoch die allgemeine Schwierigkeiten im Bereich des E-Learning wider. Der Standard wird

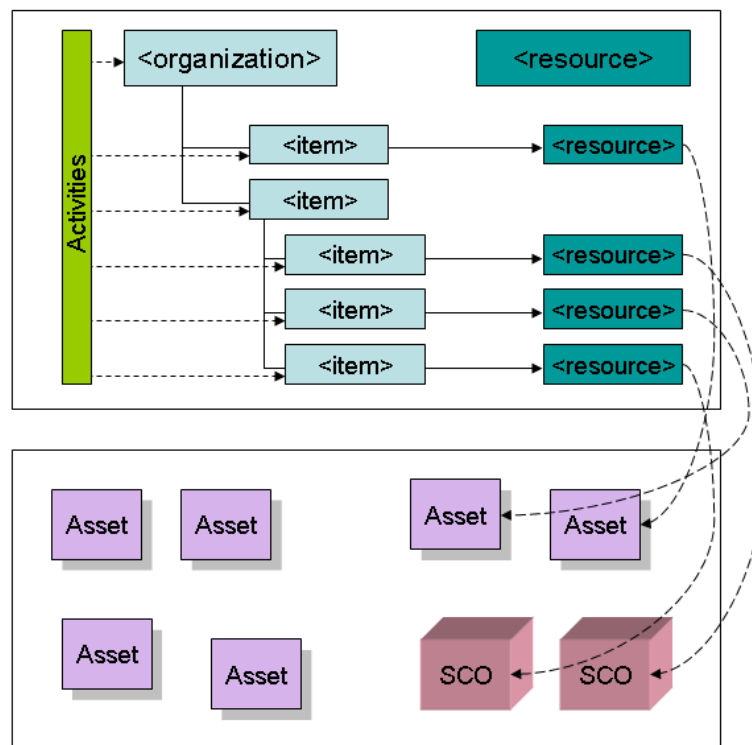


Abbildung 3.1 Das SCORM Content Aggregation Modell nach SCORM CAM

ständig weiterentwickelt und erweitert, auf der anderen Seite gibt es kaum ein Lernmanagementsystem, welches überhaupt die bisher definierten Bereiche vollständig unterstützt. Meist unterstützen Lernmanagementsysteme nur ausgewählte Bereiche wie SCORM CAM oder aber verschiedene ältere Versionen der SCORM-Teilbereiche [39]. Als Beispiel für nötige und sinnvolle Ergänzungen sei die mangelnde Unterstützung hochgradiger Interaktivität, wie sie Frank Hanisch vorschlägt [68], genannt. Für weitere Beispiele sei auf die neu entwickelten Ideen aus Kapitel 5 verwiesen.

3.8 Weitere Standards

Eine Reihe weiterer XML-basierter Standards im Bereich des E-Learnings zur Auszeichnung von Lerninhalten sind:

- EML: Die Educational Modelling Language der Open University der Niederlande ist Basis der IMS Learning Design Spezifikation [85].
- LMML: Die Learning Material Markup Language [46] wurde an der Universität Passau entwickelt. Sie wird noch weiterentwickelt, und Kurse basierend auf LMML werden an verschiedenen Universitäten eingesetzt.
- TeachML: Dieser XML-Dialekt wurde im Rahmen des Targeteam Projekts (TAR-

geted Reuse and GEneration of TEAching Materials) an der Technischen Universität München und der Universität der Bundeswehr München entwickelt [135].

Eine guter Vergleich dieser und weiterer Modellierungs-Sprachen wie Palo [91] ist bei [118] nachzulesen.

3.9 Lernobjektsammlungen

Anders als Lernmanagementsysteme legen Lernobjektsammlungen (*Repositories*) weniger Wert auf eine durchstrukturierte Kursarchitektur mit Lerngruppen und Kommunikationsdiensten, sondern mehr auf abgeschlossene gut aufbereitete Lernobjekte, die als Bibliothek vorgehalten werden. Hauptaugenmerk liegt damit auf der durchgängigen Indizierung der vorhandenen Materialien und ihrer Verfügbarkeit über einen langen Zeitraum hinweg. Eine schöne Zusammenfassung bisheriger Projekte gibt Hanisch [65]. Exemplarisch sollen hier nur die zwei Projekte Exploratives der Brown University und Merlot der Californian State University genannt werden.

Das Exploratives-Projekt unter Andries van Dam sammelt seit 1995 interaktive Illustrationen in Form von Java-basierten Applets im Bereich der Computergrafik und Bildverarbeitung. Um die Wiederverwendbarkeit der Applets zu erhöhen, wird versucht, Softwaretechniken wie Java Beans einzusetzen, sowie die Applets nach Kategorien (*Core Technologies, Application Technologies, Support Technologies* [87]) einzuteilen. Verschiedene Lernszenarien wie Labor, Visualisierung, Simulation, Rollenspiel usw. können mit den Applets abgedeckt werden [131]. Ziel sind rekombinierbare, hochgradig interaktive, selbst erforschbare Mikrowelten [87]. Schwierigkeiten entstehen vor allem im Umgang mit Java, sei es die langwierige und teure Entwicklungszeit oder die Probleme im Ausführen der Applets auf Nutzerseite.

Das Merlot-Projekt [140] [129] ist nicht auf Java als Softwarebasis von Lernobjekten eingeschränkt. Materialien können zum Beispiel auch als Flash, Shockwave, ausführbares Programm, CD-Rom, Film etc. angelegt werden. Merlot klassifiziert die Materialien in elf verschiedene Kategorien: Simulation, Animation, Tutorial, Drill and Practice, Quiz/Test, Vorlesung, Case Study, Sammlung und Referenzmaterialien [139]. Ein Peer-Reviewing-Prozess sichert die Qualität der eingestellten Lernmaterialien, ehe sie dem Publikum verfügbar gemacht werden [138]. Die Lernmaterialien sind auf keine bestimmte Fachrichtung festgelegt, derzeit sind neben vielen naturwissenschaftlichen Fächern wie Informatik, Mathematik, Biologie, Physik und Chemie auch geisteswissenschaftliche Fächer vorhanden. Die Sammlung umfasst inzwischen knapp 15000 Materialien bei über 36000 eingetragenen Mitgliedern. Die Materialien bei Merlot werden nicht im System abgelegt sondern als externe Ressourcen verlinkt.

3.10 Zusammenfassung

Es existieren viele verschiedene Lernplattformen, welche zum großen Teil einen weitgehend gleichen Funktionsumfang ausweisen. Mächtigere Plattformen werden nur kommerziell angeboten, hierbei sind die Lizenzkosten allerdings oft exorbitant hoch, da hier gerne pro Nutzer abgerechnet wird. Trotz dieser Tatsache und trotz guter Prognosen bezüglich der weiteren Entwicklung des Marktes scheint die nachhaltige Entwicklung auch großer bekannter Plattformen nur unzureichend gesichert, wenn man sich das Beispiel von WebCT und Hyper-G (siehe Abschnitt 3.4) vor Augen führt. Zunehmende Konkurrenz aus dem Open-source-Lager bezeugt, dass auch in Zukunft der Markt heiß umkämpft sein wird.

Die Plattformen werden, um der Komplexität und der Vielfalt der Anwendungsgebiete Herr zu werden, modular ausgestaltet. Treibende Kraft hinsichtlich der Entwicklung der Funktionalität sind universitäre Einrichtungen, welche viele Lernplattformen in Form von Spinoffs oder als Ergebnis von Projekten hervorgebracht haben.

Unterscheidungen und Klassifikationen der Lernplattformen sind nicht trivial, eine wahre Explosion an Namensgebungen für Plattformen im Bereich des webbasierten Lernens erschwert die Einordnung, zumal die Unterscheidungen teilweise willkürlich und aus Moden begründet erscheinen.

4.1 Moderne Courseware

Zur Implementierung moderner Courseware ist die Nutzung aktueller Technologien für eine effiziente Umsetzung neuer Ideen wichtig. In diesem Kapitel wird zunächst das zu Grunde liegende Framework Cocoon (Abschnitt 4.2) beschrieben, mittels dessen Hilfe eine Courseware neu implementiert wurde. Diese erlaubt eine einfache Handhabung von MathML im Web (Abschnitt 4.3), erleichtert eine Umsetzung des von Frank Hanisch vorgeschlagenen Drag-And-Drop-Paradigmas für ein Benutzer-Forum (Abschnitt 4.4) und ermöglicht die Darstellung der Inhalte in verschiedenen Formaten beziehungsweise unterstützt die Generierung von Grafiken (Abschnitt 4.5).

4.2 Cocoon

Cocoon ist ein so genanntes Web-Development-Framework. Im Jahr 1998 wurde das Projekt von Stefano Mazzocchi und Jon Stevens gestartet [97]. Ab 1999 wurde es als Unterprojekt von der Apache Software Foundation übernommen. Zunächst ging es darum, eine Sammlung von Skripten zur Generierung von Dokumentation von Open-Source-Projekten zu automatisieren. Mazzocchi programmierte dazu ein Servlet, welches die Skriptaktionen überflüssig machte, indem es serverseitig XSL-Stylesheets auf XML-Dokumente anwendete und das Ergebnis unmittelbar an einen Browser zur Darstellung sandte. Da die serverseitige Verarbeitung von XML auf großes Interesse der Community stieß, wurde das Projekt eigenständig und erlangte dann 2001 mit Version 2 eine Reife, die seiner Verbreitung als Software zur Publikation von Dokumenten auf Basis von XML und XSL im Web förderlich war. Cocoon bietet dem Anwender folgende Eigen-

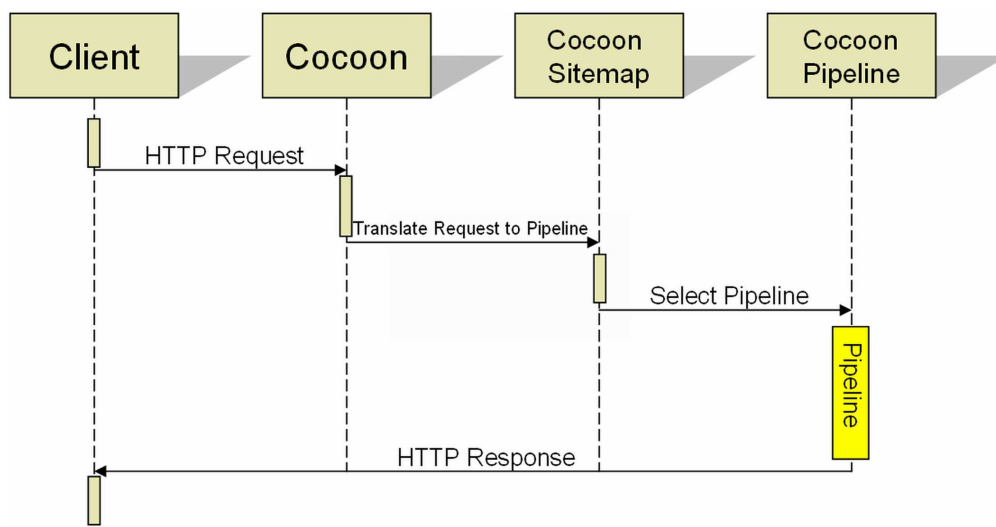


Abbildung 4.1 Verarbeitung eines HTTP-Requests in Cocoon

schaften: durch SAX-basierte Verarbeitung von XML-Dokumenten eine gute Performanz und Skalierbarkeit, das Prinzip der Trennung von Logik, Inhalt und Design (*separation of concerns: content, logic and style*), eine zentralisierte Konfiguration mit eingebauten Cache-Mechanismen, und damit insgesamt eine solide Basis, um mittels moderner Technik webbasierte Courseware zu entwickeln und zu implementieren.

Im Vergleich zu herkömmlichen Techniken wie Servlets oder ASP, JSP und PHP ist Cocoon strikt XML-orientiert. Alle Dokumente, Konfigurationen und Steuerungselemente sind in XML geschrieben. Die grundsätzlichen Dienste, die Cocoon anbietet, sind (siehe [3] und [171]): Zuordnen der Anfrage zur richtigen Pipeline (*Dispatching*), Generierung der XML-Dokumente aus Dateien oder Datenbankabfragen (*Generation*), Transformierung der Dokumente (*Transformation*), Zusammensetzen verschiedener Dokumente zu größeren Dokumenten (*Aggregation*) und das Rendern der XML-Dokumente mittels Serialisierern (*Serialization*). Die Dienste sind in einer Java-Bibliothek implementiert, in XML-Dokumenten wie der Sitemap werden sie über spezielle Tags angesprochen. Mit einem Web-Development-Framework wie Cocoon findet ein Paradigmenwechsel hinsichtlich der Vorhaltung und Aufbereitung der Daten statt. Während in einer Webumgebung wie ASP, JSP oder PHP statischer Inhalt mit aktiven Elementen angereichert wird, wird bei Cocoon das Dokument selbst belassen und nur durch Transformationen aktive oder sich ändernde Inhalte beigefügt. Das impliziert das Konzept der so genannten Pipeline, welche die Verarbeitung von Dokumenten steuert. Dabei wird ein HTML-Aufruf einer Seite wie in Abbildung 4.1 durch die Sitemap einer gewissen Pipeline zugeordnet, wobei die aufrufende URL (*Uniform Resource Locator*) ausgewertet wird. Dann wird die Pipeline, die ebenfalls in der Sitemap definiert ist, abgearbeitet (siehe Abbildung 4.2). Ein Generator liest beispielsweise ein XML-Dokument aus einer Datei oder aus einer Datenbankabfrage ein und gibt es als SAX-Ereignisse in die Pipeline. Im nächsten Schritt wird dieses Dokument in Form von SAX-Ereignissen durch einen Transformer verändert, zum Beispiel könnten alle Links um Attribute erweitert werden. Ein

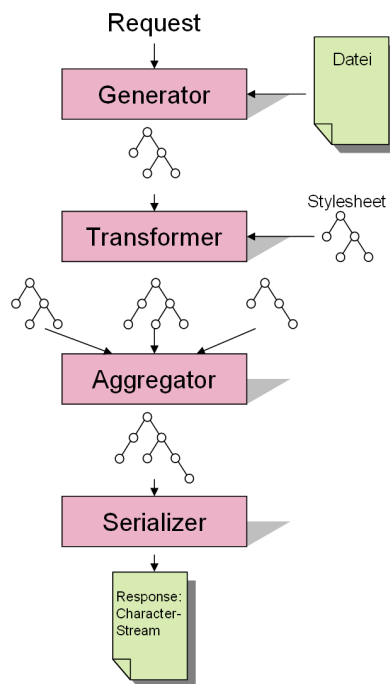


Abbildung 4.2 Ablauf der Cocoon-Pipeline

optionaler Aggregator kombiniert das transformierte Dokument mit weiteren Elementen wie zum Beispiel mit einem Titlelement oder einem Fußzeilenelement. Der Serialisierer setzt die Ereignisse wieder in einen seriellen Byte- oder Character-Strom um und erzeugt so ein PDF- oder HTML-Dokument.

Stellt man die verschiedenen Systeme in einer groben Übersicht zusammen, so gelangt zur Vergleichstabelle 4.2. Die Kriterien sollen noch einmal kurz erläutert werden:

- **Modernität:** Cocoon bietet aufgrund der verwendeten Techniken XML und XSL sowie Java als Programmiersprache und der Dokumentenzentriertheit den neuesten Ansatz und ist somit am modernsten. PHP fällt vor allem wegen seiner lange ungenügenden Umsetzung des modernen Konzepts der Objektorientiertheit, den mangelhaften und unübersichtlichen Bibliotheken mit wenig Dokumentation sowie einer unübersichtlichen Spracharchitektur zurück.
- Am schlechtesten schneidet PHP ab, weil es mit seiner Architektur viel Angriffsfläche für Buffer-Overflow-Fehler und bei Datenbankabfragen zum Beispiel SQL-Injektionen erlaubt. JSP bietet hier deutlich bessere Mechanismen, einzig SQL-Injektionen sind auch hier gefährlich. Cocoon schneidet hier wiederum am besten ab, dass es gerade für Datenbankabfragen zusätzliche Mechanismen der Abstraktion anbietet, welche SQL-Injektionen verhindern (z.B. der *Authentication Manager*, welcher über eine in XML definierte Zwischenschicht auf eine Datenbankta-belle zugreift).
- Bezüglich der Performanz schlägt PHP die beiden anderen Ansätze. Der PHP-

Interpreter ist sehr schnell, insgesamt werden wenig Ressourcen verbraucht. JSP und Cocoon werden erst ähnlich schnell, wenn man die eingebauten Caching-Algorithmen einsetzt.

- Hingegen ist die Skalierbarkeit von JSP sowie Cocoon ungeschlagen, da man diese Frameworks auf Tomcat laufen lassen kann, ein Applikationsserver, welcher auf verschiedenen Rechnern verteilt laufen und Anfragen dynamisch zwischen diesen Rechnern ausbalancieren kann.
- Datenbankverbindungen sind heutzutage mit jedem Framework gut einzurichten. PHP hat mit den neuesten Versionen endlich auch eine Abstraktionsschicht (*Database Connectivity* - DBC), die wie bei Cocoon und JSP (*Java Database Connectivity* - JDBC [133]) die Anbindung verschiedener Datenbanken ohne Änderung der Aufrufe in den Anwendungen ermöglicht. PHP wird deswegen schlechter gewichtet, weil diese Erweiterung so spät umgesetzt wurde.
- XML-Unterstützung ist in allen Systemen vorhanden. Nur in Cocoon ist sie jedoch ein derart zentrales Element und derartig durchgängig umgesetzt. In JSP wie PHP muss man für die Verarbeitung von XML-Dokumenten Bibliotheken einbinden und diese dann aufrufen. Die Unterstützung ist also nur über Erweiterungen zu erreichen.

	Cocoon	JSP/ASP	PHP
Modernität	++	+	0
Sicherheit	++	+	-
Performanz	+	+	++
Skalierbarkeit	++	++	+
Datenbanken	++	++	+
XML-Unterstützung	++	0	0

Tabelle 4.1 Vergleich von Cocoon mit JSP/ASP und PHP anhand ausgewählter Kriterien

4.3 Einbindung mathematischer Formeln

Seit 1998 liegt eine offizielle Spezifikation von MathML des W3C vor (siehe Abschnitt 2.4 und [159]). Trotz dieses recht hohen Alters gab es und gibt es immer noch nicht sehr viele Seiten, welche MathML einbinden. Dies hat vor allem drei Gründe: Inkompatibilität mit Browsern (siehe Tabelle 2.5), erhöhter Aufwand bei der Erstellung der Formeln beziehungsweise bei der Einbettung in Webseiten wegen der verschiedenen Browser und die mangelnde Notwendigkeit, von der hergebrachten Herangehensweise, also dem Einbetten der Formeln als Grafiken, abzuweichen.

Am Institut für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts der Universität Tübingen wurden schon seit 1998 erste Erfahrungen mit webbasierten Grafikkursen und den dazugehörigen Lernumgebungen gesammelt [63] [82] [83] [83]. Gestartet als Online-Kurs für die Computergrafik ("Computer-Graphik spielend lernen"), wurde das System sukzessive ausgebaut. Eine schichtenbasierte Datenbank erlaubt die Abspeicherung der Inhalte in verschiedenen Sprachen sowie in verschiedenen Ebenen der Expertise (Laie, Amateur, Experte) [65]. Links und Verweise werden konsistent gehalten, in dem sie in der Datenbank vorgehalten werden und in einer Generierungsphase der Webseiten erst erstellt werden. Die Architektur funktioniert Template-basiert und Metastrukturen wie Kurse können mittels eines Autorenwerkzeugs offline erstellt, geändert oder erweitert werden. Online-Assistenten nach dem Model-View-Control-Paradigma erlauben für Inhalte-Erstellung auch webbasierten Zugang und webbasierte Änderungsmöglichkeiten. Alle Modifikationen werden jedoch aus Performanzgründen nicht dynamisch im Web vorgehalten, sondern in einer Generierungsphase erstellt. Dieser halbautomatische Ansatz konnte erst nach dem Aufkommen neuerer Techniken für Webframeworks abgelöst werden (siehe Abschnitt 2.6).

Im Rahmen des vom Land Baden-Württemberg finanzierten Projekts LIVE ("Lernen und Lehren durch Interaktive Visualisierung am Beispiel eines Kurses der Elektrotechnik") wurden diese Grundlagen fortentwickelt. Neben dem Hauptziel einer vollständig dynamischen vorlesungsbegleitenden Courseware mit Benutzerverwaltung, Lernobjekten, Kursstruktur und Lernpfad wurden hier Inhalte einer Vorlesung über Signalausbreitung mittels MathML browserübergreifend im Web präsentiert [54]. Weitere Elemente der Courseware sind kollaborative Elemente wie ein integriertes Forum und Multiple-Choice-Selbsttests zur Selbstbefragung.

Das Autorensystem für die Kursinhalte einschließlich der Strukturierung der Abfolge erfolgt mittels Online-Eingabefeldern komplett webbasiert. Inhalte werden einschließlich ihrer Strukturinformationen in einer Datenbank abgelegt. Die Navigation der Courseware wird ebenfalls in einer Datenbanktabelle abgelegt und ist somit ebenfalls vollständig dynamisch. Eine Webseite der Courseware wird mittels der Cocoon-Aggregation aus Kopfzeile, Fußzeile, Navigationsbereich und Inhaltsbereich zusammengesetzt. Während die Kopf- und Fußzeile aus statischen Dokumenten im Rahmen der Pipeline zusammengesetzt werden, werden die Navigation und der Inhalt dynamisch aus der Datenbank generiert. Um eine korrekte Darstellung von MathML im Internet Explorer und in Netscape/Firefox zu gewährleisten, müssen den jeweiligen Browsern angepasste Stylesheet-Transformationen durchgeführt werden (siehe Bild 4.3), im Internet Explorer muss zusätzlich ein Plugin zum Rendern der Formeln installiert sein. Trotz der komplett webbasierten Eingabemöglichkeiten mit Korrekturmöglichkeiten ist festzustellen, dass MathML, wie in Abschnitt 2.4 beschrieben, nicht oder nur in geringem Umfang mittels der webbasierten Eingabemöglichkeiten erstellt werden kann, da der zu schreibende Code sehr schnell umfangreich wird und ohne Unterstützung durch einen Formeleditor wie MathType oder OpenOffice nicht zu handhaben ist.

Die im Projekt erreichten Ergebnisse decken die möglichen erreichbaren Browser im Internet weitgehend ab, da selbst im Jahr 2006 keine weiteren MathML-kompatiblen

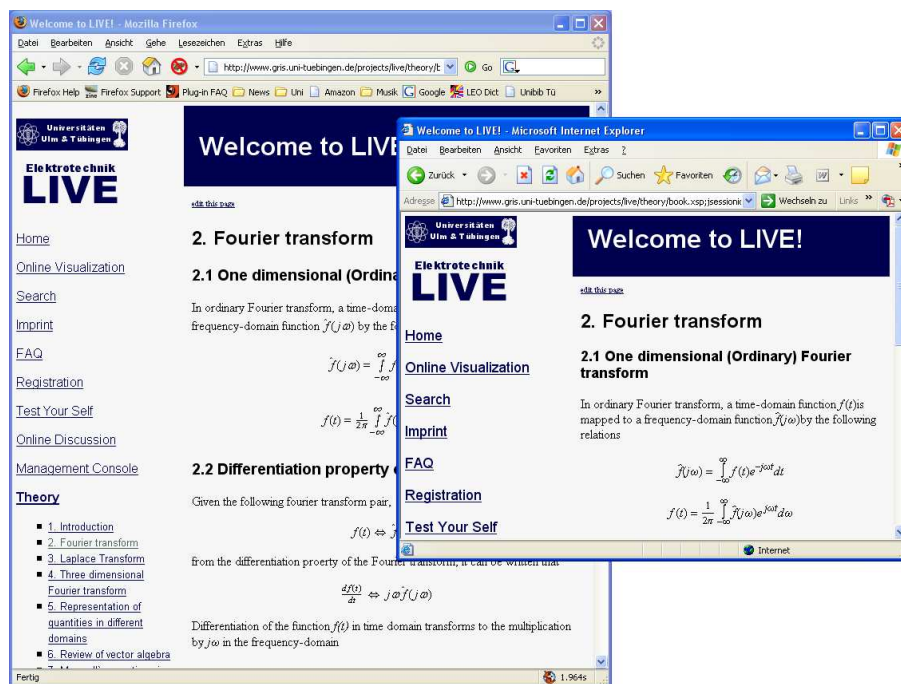


Abbildung 4.3 Die LIVE-Courseware im Web, zu sehen ist MathML gleichermaßen in den Browsern Firefox und Internet Explorer

Browser im Markt hinzugekommen sind [168]. Mit der LIVE-Courseware können unter Windows Firefox, Mozilla ab Version 0.9.4, Netscape ab Version 6.1, der Internet Explorer ab Version 5.5 mit MathPlayer-Plugin und Amaya benutzt werden, um die Inhalte korrekt dargestellt abzurufen. Trotz der relativ langen Zeitspanne von vier Jahren, die zwischen der Veröffentlichung des MathML-Standards durch das W3C und der Erstellung der Courseware des LIVE-Projekts lag, war diese Courseware eine der ersten, die durchgängig MathML verwendete. Die vorlesungsbegleitenden Seiten des MIT zum Beispiel boten die Unterstützung von MathML erst 2003 an (siehe [94] [80]).

4.4 Integration hochgradig interaktiver Inhalte

Das Drag-And-Drop-Paradigma nach Frank Hanisch [62] ist auf einer modernen technischen Plattform wie Cocoon ebenfalls mit relativ wenig Aufwand zu verwirklichen und zu erweitern. Hanisch schlägt in seiner Arbeit vor, Lernobjekte nach seinem Drag-And-Drop-Paradigma zu modifizieren, um hochgradig interaktive Lernobjekte zu erhalten. Kriterien für eine korrekte Umsetzung dieses Paradigmas spezifiziert Hanisch folgendermaßen:

1. Eine dem Lernen förderliche Darstellung des Wissensgebiets
2. Anschauliche Bedienung
3. Direkte Manipulation der Ansicht des Lernobjekts

4. Direkte Manipulation aller wesentlichen Parameter des Lernobjekts
5. Direkte Manipulation der Funktionalität des Lernobjekts
6. Angemessene Rückmeldungen sowie Hilfestellung

Dieses Paradigma erweitert die bis dahin angestrebte Skripting-Fähigkeiten von Lernobjekten [64] um anwendungsübergreifende Drag-And-Drop-Fähigkeit. Für eine moderne Lernsoftware wurde diese Technik eingesetzt, um ein um Drag-And-Drop-Fähigkeiten erweitertes Forum einzurichten.

Während Skripting ohne Drag-And-Drop die Steuerung eines Lernobjekts über definierte Zugriffe von außen ermöglicht, um beispielsweise bei einem in eine HTML-Seite eingebetteten Lernobjekt bestimmte Parameter durch einen Klick auf einen Link im Text einzustellen, wird beim Drag-And-Drop-Skripting der Zustand eines Lernobjekts in einem Script abgespeichert, sobald der Nutzer eine Drag-And-Drop-Aktion ausführt. Dieser Zustand in einem gleichzeitig erstellten Foto des Lernobjekts abgespeichert. Damit kann der Nutzer verschiedene Zustände eines Lernobjekts einfach und komfortabel abspeichern und als Bild auf seinem Rechner vorhalten. Um den abgespeicherten Zustand seines Lernobjekts wieder zu restaurieren, genügt es, das Bild wieder auf das Lernobjekt zu ziehen.

Die Speicherung erfolgt direkt im Bild oder in zugeordneten Bildeigenschaften. Lernobjekte müssen natürlich dem Drag-And-Drop-Schema angepasst werden und Funktionen zum Setzen von Parametern oder Aktionen nach außen anbieten. Für die Verwendung in einer Online-Lernumgebung sollte neben der Präsentation der Lernobjekte jedoch auch eine komfortable Nutzer-zu-Nutzer-Interaktion unterstützt werden (siehe [57]). Als Grundidee dient hierfür, dass derartige Zustandsabzüge von Lernobjekten ideal für die gemeinsame Besprechung von Fragen und Problemen in Lerngruppen, aber auch bei Fragen an die Tutoren einsetzbar sind. Dies geschieht am einfachsten im Rahmen eines Forums, welches neben der Eingabe von Texten auch das Einfügen von solchen Snapshots ermöglicht.

Hierfür ist eine Erweiterung der Courseware erforderlich, da Foren in aller Regel das gleichzeitige Hochladen eines Fotos beim Schreiben eines Beitrages nur in Form eines Dateialogs ermöglichen, welcher das Drag-And-Drop-Paradigma durchbricht. Der Nutzer ist gezwungen, die Konfiguration seines Lernobjekts erst im Dateisystem seines Rechners abzuspeichern, mittels des Dateialogs an die richtige Stelle zu navigieren und die korrekte Datei auszuwählen. Wird das Drag-And-Drop-Paradigma in einem Forum korrekt umgesetzt, so gestaltet sich der Ablauf in einem Forum folgendermaßen (siehe auch Abbildung 4.4):

1. Nutzer A erforscht ein Lernobjekt, dabei stößt er auf eine Frage, zu der er Hilfe von außen benötigt.
2. Nutzer A meldet sich beim zugehörigen Forum an und schreibt eine kurze Nachricht, wobei er ein Zustandsfoto seines Lernobjekts per Drag-And-Drop beifügt.



Abbildung 4.4 Ablauf in einem Drag-And-Drop-fähigen Forum

3. Er schickt die Nachricht ab, sie wird somit auf dem Server gespeichert.
4. Andere Nutzer lesen die Nachricht.
5. Dabei können sie ihrerseits per Drag-And-Drop ihr Lernobjekt in den der Ursprungs-
nachricht entsprechenden Zustand (Parameter, Zeitablauf etc.) versetzen.

Eine Antwort von Nutzer B kann ebenfalls eine Konfiguration des Lernobjekts enthalten. Das Erstellen und Abschicken der Nachricht erfolgt wiederum wie Schritt 1-3. Nutzer A kann dann wie in Schritt 5 per Drag-And-Drop sein Lernobjekt mit der korrekten Konfiguration anpassen.

Die Neuerstellung von Nachrichten im Forum ist durch die Möglichkeit, per Drag-And-Drop Bilder hinzuzufügen, angereichert. Diese Dateien werden beim Abschicken der Nachricht gleichzeitig mit der Nachricht selbst verschickt. Genauso sind alle Seiten, welche Nachrichten darstellen, um die Möglichkeit der Darstellung von Bildern erweitert. Um beim Verfassen der Nachrichten im Browser Bilder ablegen zu können, ist die Bereitstellung zum Beispiel eines Java-Applets nötig, welches einen Rahmen zum Ablegen darstellt und die Kommunikation zum Verschicken der Nachricht übernimmt.

4.5 Transformation

Die in 2.6 und 2.8 dargestellten Vorteile von XML gegenüber herkömmlichen Techniken der Datenaufbereitung und Datenspeicherung sollen in diesem Abschnitt anhand zweier praktischer Umsetzungen belegt werden. Als Beispiel wird hierzu die Aufbereitung von Inhalten für verschiedenartige Ausgabegeräte sowie die dynamische Auswertung von statistischen Daten behandelt.

Ein bekanntes Problem bei der Unterhaltung von Auftritten im Web ist die Umsetzung der Inhalte in verschiedene Formate, um dem Besucher eine größtmögliche Flexibilität im Umgang zu ermöglichen. Als wichtigste Formate werden heutzutage HTML zur Anzeige im Browser und PDF zum Abspeichern und Ausdrucken angesehen. Deswegen konzentriert sich der hier vorgestellte Ansatz auf diese beiden Formate, wobei auch weitere Formate wie WML (Wireless Markup Language, ein stark eingeschränktes XHTML), Bildformate wie SVG, PNG oder JPG oder auch das Druckformat Postscript auf analoge Art und Weise erzeugt werden können.

Prinzipiell gibt es zwei Möglichkeiten, die Bereitstellung der Formate zu ermöglichen. Entweder man überträgt diese Aufgabe den Autoren, welche dann beim Einstellen neuer Inhalte auch gleichzeitig dafür Sorge zu tragen haben, dass die ebenfalls erforderlichen Formate hinzugefügt werden. Oder man erweitert das System derart, dass es diese Aufgabe mit übernimmt. Der erste Ansatz ist natürlich sehr fehlerträchtig, da vom System nicht überprüft werden kann, ob ein neu eingestelltes HTML-Dokument der dazugehörigen Version eines PDF-Dokument entspricht. Weiterhin muss dann jeder Autor die Software zur Erzeugung der verschiedenen Formate bei sich installiert haben. Soll ein neues zusätzliches Format unterstützt werden, müsste der gesamte Datenbestand durchforstet und entsprechende Dateien hinzugefügt werden. Deswegen ist offensichtlich der zweite Ansatz der elegantere, zumal XML angetreten ist, genau diese Problematik zu lösen. Die Inhalte müssen von Autoren also ohne Layout nur in einem definierten XML-Dialekt erstellt und ins System übertragen werden, damit dann das System die Transformation in die gewünschten Ausgabeformate übernehmen kann. Je nach Implementierung kann diese Transformation dynamisch auf Anfrage erfolgen oder in einem Durchlauf einmalig erzeugt werden. Das Optimum wird mit der dynamischen Variante erreicht, die die Aktualität der Daten sichert, wenn durch Cachingmechanismen unnötige Mehrfachgenerierung verhindert wird.

Am Lehrstuhl des WSI/GRIS wurden für verschiedene Vorlesungen (Bildkommunikation, Bildverarbeitung I und Visualisierung) die Übungsblätter mit dieser Technik erstellt und verbreitet. Die Eingabe erfolgt in einem HTML-ähnlichen Dialekt, die Darstellung wird, wie in Abbildung 4.5 zu sehen, einerseits in HTML für die Browser-Ansicht oder als PDF zum Ausdrucken ohne störende Navigationsleisten oder Überschriften angeboten.

Da die Darstellung in XHTML durch Cocoon schon gegeben ist, ist die Lernumgebung nur um die Transformation nach PDF zu erweitern. Das zu Grunde liegende Framework bringt eine dafür erforderliche Bibliothek (XML Formatting Objects [7]) mit sich, allein ein XSL-Stylesheet mit Anweisungen, wie Strukturelemente in PDF umgesetzt werden sollen, muss bereitgestellt werden. Hierin wird nach einer Definition des Seitenlay-

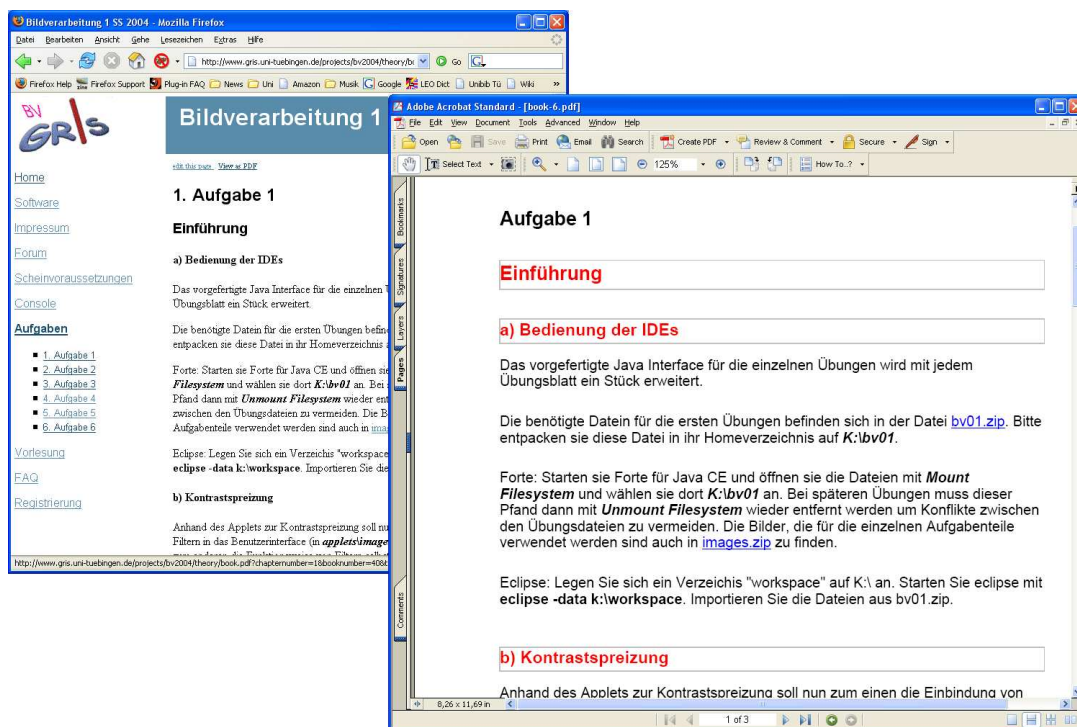


Abbildung 4.5 Die erste Aufgabe der Vorlesung Bildverarbeitung I: links nach HTML transformiert im Browser und rechts als dynamisch generiertes PDF-Dokument ohne Navigation und Titel- oder Fußzeile

outs (Größe der Seite, Definition der Seitenränder) für jedes Strukturelement aufgeführt, wie es im PDF-Dokument dargestellt werden soll. So können Zwischenüberschriften verschiedene Schriftarten und -farben zugewiesen werden, Paragraphen umrandet und Verweise hervorgehoben werden. Somit wird beim Klick auf den PDF-Verweis der Inhalt statt in XHTML in PDF gerendert.

Gleiches gilt für die Erstellung von RSS-Feeds, welche im Umfeld von Newstickern dazu dienen, dem Anwender, welcher solche Nachrichtenkanäle abonniert, automatisch die neuesten Informationen zu kommen zu lassen. Hierfür wurde vom W3C ein Standard geschaffen, welcher einen sehr reduzierten XML-Dialekt zur Beschreibung von Informationseinheiten festlegt (siehe [163]). Dieses Schema ermöglicht die Zusammenfassung von Webseiten, wobei man sich auf reine Textinformationen beschränken kann und beispielsweise bei Neuigkeiten nur einen Kanal, eine Überschrift und einen Link auf die vollständige Neuigkeit angeben kann. Auch dieses Format lässt sich aus den gegebenen Inhalten in einem Repository durch einfaches Anlegen einer entsprechenden XSL-Transformation erzeugen und funktioniert damit auf die gleiche Weise wie die Generation der PDF-Dokumente mittels FOP.

Die dynamische Erzeugung von Grafiken zur Nutzung einer Courseware ist eine weitere Anwendung, die von der von XML propagierten Trennung von Inhalt und Layout profitiert. Einerseits bietet sich dabei der Einsatz von XSL zur passenden Transformation von Daten in das Vektorformat SVG, welche dann mittels vorhandener Bibliotheken

in PNG oder JPG umgewandelt werden, andererseits unterstützt das zu Grunde liegende Framework Cocoon die Erfassung von Nutzungsstatistiken.

Die Generierung von NutzungsstatistikenindexNutzungstatistik im Web wurde durch die Nachfrage nach Informationen zum Verkehr im Internet angetrieben [108]. Größere Einrichtungen wie Universitäten wollten aktualisierte Informationen über die Auslastung ihrer Netzanbindung intern und nach außen auswerten und Nutzern zur Verfügung stellen. Hierfür wurden anfangs eigene externe Programme über CGI (siehe Abschnitt 2.6) täglich oder wöchentlich gestartet, welche aus den gesammelten Daten Grafiken generierten. Aktuellere Grafiken bekommt man, wenn die Bilder durch Aufruf aktiver Serverseiten erstellt werden. PHP [116] bot als eine der ersten Sprachen diese Möglichkeit. Durch Kapselung von frei verwendbaren Bibliotheken des GIMP-Projekts [137] sowie der freien Bibliotheken libpng und libjpeg in PHP-Funktionen können auf aktiven Webseiten aktuelle Daten in Grafiken übersetzt werden. Mit SVG liegt eine deutlich modernere und einfachere Art vor, um vorhandene Daten in Grafiken umzusetzen.

Um derartige Statistiken zu generieren, bleibt jedoch das Problem der Datenerhebung. In einer Courseware will man üblicherweise für jede besuchte Seite den Nutzer und die Verweildauer messen (siehe [58]). Daraus kann man Häufungen erkennen und seine Webseiten neu sortieren oder anders gruppieren. Informationen über betrachtete Seiten liefert der weit verbreitete Apache Webserver in einer Log-Datei. Hier steht in einer Textzeile unter anderem die IP-Adresse des abfragenden Rechners, unter welcher Benutzerkennung ein Nutzer eventuell eingeloggt ist, ein Datumsstempel, der HTTP-Befehl mit Antwortcode, die referierende Seite sowie der benutzte Browser.

```
1 134.2.176.67 -- [30/Oct/2006:15:53:34 +0100] "GET /images/bg.gif HTTP/1.1"  
2 304 -- "http://www.gris.uni-tuebingen.de/frame.html"  
3 "Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.8.0.7) Gecko/20060909 Firefox/1.5.0.7"
```

Listing 4.1 Eine Beispiel-Logzeile des Apache Webservers

Um diese Informationen aufbereiten zu können, müsste eine Anwendung, die hieraus Statistiken generieren will, diese Datei erst parsen. Eine bessere Möglichkeit wäre, diese Informationen gleich in einer Datenbank abzulegen. Hierfür wäre unter PHP eine relativ aufwändige Modifikation nötig. Gibt es keine generelle Kopfdatei, welche von allen Seiten ausgeführt wird, müsste man jede Seite mit einem Datenbankaufruf versehen, welcher einen Klick in der Datenbank speichert. Unter Cocoon kann man jedoch das Konzept der Sitemap sehr effizient nutzen, in dem man für alle Aufrufe notiert, dass eine so genannte *Action* durchgeführt wird, welche anhand einer kleinen Definitionsdatei genau diese Informationen in einer Datenbanktabelle speichert. Eine Modifikation der Seiten oder einer Kopfseite ist somit nicht erforderlich, der Code ist sehr kompakt einzubinden.

Damit hat man alle nötigen Informationen in einer Datenbank und kann daraus die Grafiken generieren. Verschiedene Übersichten über die verschiedenen Teilbereiche einer

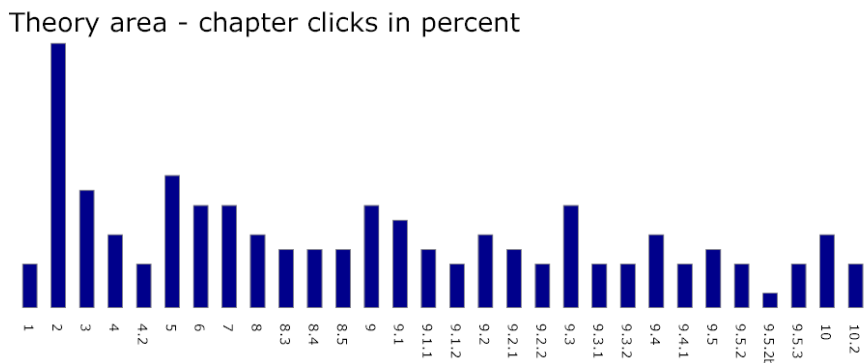


Abbildung 4.6 Eine Statistik der Klicks auf den Inhaltsseiten der Courseware in Prozent

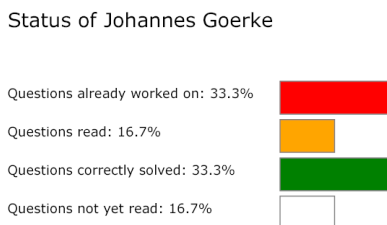


Abbildung 4.7 Diese Grafik gibt einem angemeldeten Benutzer einen Überblick über sein Fortkommen im Multiple-Choice-Selbsttest

Courseware mögen von Interesse sein: allgemein sind die Zugriffszahlen/Seite interessant. Werden Multiple-Choice-Selbsttests angeboten, so will jeder Nutzer individuell eine Statistik seiner Antwortqualität, der Administrator jedoch ein allgemeines Bild aller abgegebenen Antworten. Ähnlich kann bei Diskussionsforen von Interesse sein, welche Foren am meisten frequentiert werden.

Das erste dynamisch erzeugte SVG-Bild 4.6 zeigt einen Überblick über die Nutzungsfrequenz im Bereich der Theorieseiten der Courseware in Relation zueinander. Die Startseite ist natürlicherweise die meistbesuchte Seite. Alle Unterkapitel sind, da sie auf einer eigenen HTML-Seite angezeigt werden, als eigene Seite ausgewertet worden, eine weitere Möglichkeit wäre, alle Seiten eines Kapitels zusammenzufassen.

Das Bild 4.7 zeigt beispielhaft die Statistik des Multiple-Choice-Selbsttests für den angemeldeten Benutzer. Er kann anhand der Daten seinen Fortschritt ersehen, sowie das Verhältnis von richtig und falsch beantworteten Fragen einschätzen. Eine simple Erweiterung dieser Statistik auf alle Benutzer (Figur 4.8) zeigt ein großes Übergewicht an ungelesenen Fragen, was sich darauf zurückführen lässt, dass neu angemeldete Benutzer sofort in diese Statistik einfließen, auch wenn sie den Selbsttestbereich der Courseware noch gar nicht besucht haben.

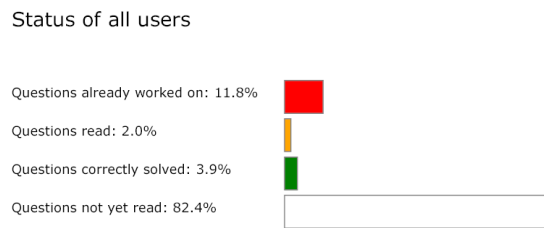


Abbildung 4.8 Analog zu Grafik 4.7 ein Überblick über die Erfolgsquote aller im System angemeldeten Benutzer beim Multiple-Choice-Selbsttest - viele neu angemeldete Benutzer lesen sich erst ein, ehe sie den fakultativen Selbsttest bearbeiten, entsprechend hat der Großteil der Nutzer den Selbsttest noch nicht durchgeführt.

4.6 Zusammenfassung und Ausblick

In diesem Kapitel konnte gezeigt werden, dass zur Erstellung moderner Courseware die Verwendung einer modernen technischen Basis hilft und damit die Entwicklung von zusätzlichen Modulen und Diensten unterstützt und beschleunigt. Die Mächtigkeit der zu Grunde liegenden Plattform konnte in verschiedener Hinsicht gezeigt werden. Einerseits können mit einer korrekten Verwendung des XML-Standards MathML mathematische Formeln in aktuellen Browsern, wie sie zum großen Teil verwendet werden, erstmals direkt im Web angezeigt werden, ohne den bekannten Umweg über die Generierung von Bildern, welche die Formeln beinhalten, mit all den bekannten Nachteilen mangelhafter Weiternutzung sowie der komplizierter Aktualisierung beziehungsweise Korrektur, gehen zu müssen.

Anhand der Beispiele Erzeugung von PDF-Versionen und Erzeugung eines RSS-News-Feed wurde die Mächtigkeit der Trennung von Inhalt und Layout im verwendeten Framework sowie die einfache Erweiterbarkeit auf neue Formate durch die Benutzung verschiedener Transformationsvorlagen innerhalb des Framework gezeigt.

Weitere Aspekte der durch XML verfügbaren Flexibilität ist die dynamische Generierung von Anwendercharts, welche verschiedene Aspekte wie Häufigkeit der Klicks auf den verschiedenen Seiten der Courseware oder Fortschritt beim Lernerfolg grafisch anzeigen. Dabei ist keine Verwendung externer Software notwendig, sondern die Grafiken können unter Verwendung des SVG-Standards direkt aus dem System heraus generiert werden.

Auch die Integration von hochgradig interaktiven Lernobjekten in die neue Courseware ist keine Hürde. Neue Interaktionsmöglichkeiten zwischen Studenten und Tutoren werden über die neue Möglichkeit des Drag-And-Drop bei Diskussionsbeiträgen im Forum eröffnet. Den Studenten wird es erleichtert, mit ihren Tutoren beispielsweise ihre Parametrisierungen von Applets bei Problemen zu diskutieren.

Die hier gemachten Erfahrungen zeigen, dass sich die Verwendung einer geeigneten technischen Basis empfiehlt und sich hiermit ein deutlicher Vorteil gegenüber anderer Courseware, welche noch mit veralteten Softwareframeworks arbeitet, ergibt.

Ein interaktives Online-Lehrbuch

5.1 Einführung

Die mit Lehre befassten Teile der internationalen Computergrafik-Gemeinschaft haben die Materialsammlung "Computer Graphics Educational Material Source (CGEMS)" ins Leben gerufen, ein weltweit hoch angesehenes Repositorium zum Zwecke, Lehrinhalte eines Computergrafikcurriculums gemeinsam zu benutzen. Diese Sammlung wurde im Jahr 2004 einem größeren Publikum vorgestellt [43] und es wird seitdem dazu aufgerufen, dort Beiträge einzustellen, um eine aktive Nutzergemeinschaft aufzubauen und um die Inhalte auch regelmäßig zu nutzen. Um für diese Materialsammlung Autoren zu gewinnen werden hier einige neue Technologien vorgestellt: quelltextbasierte Interaktivität und serverseitiges Kompilieren, welches in einem eigenen Repositorium auch umgesetzt wurde.

Die Hauptschwierigkeit aller Materialsammlungen, seien es Bilderarchive, Videosammlungen oder Lehrmaterialien, ist es, eine kritische Masse an Beiträgen und Beitragenden zu gewinnen, damit ein Mehrwert für alle Beteiligten entsteht. Im Augenblick ist Anerkennung unter Kollegen die einzige Belohnung für CGEMS-Autoren, welche die schwierige und mühsame Aufgabe auf sich nehmen, Lerninhalte aufzuarbeiten und zu modularisieren [132], um sie in das Repositorium einstellen zu können. Es wäre schöner, wenn Autoren für ihre Beiträge zusätzlich zur Reputation noch durch anderen Mehrwert belohnt würden, als Beispiel wäre hier vereinfachtes und beschleunigtes Erstellen von Inhalten oder Vereinfachungen beim Konvertieren ihrer Arbeiten in verschiedene Formate zu nennen. Der derzeitige Stand des CGEMS-Projekts wird in Abschnitt 5.2 erläutert, dabei wird gleichzeitig auch ein Blick auf andere Materialsammlungen und deren Konzepte geworfen. Vor allem wird dargelegt, wie eine einfache XML-basierte Auszeichnungssprache und dazugehörige Transformationen eine einfache, übersichtliche Basis

zum Erstellen, Verweisen und Veröffentlichen von Lernmaterialien mit mathematischem und grafisch-interaktivem Inhalt sein können. Als Ergebnis dieser Technik unterstützt ein solches Repositorium Autoren mit Schablonen und Templates, welche die Entwicklung und die Produktion von Kursen, Modulen und kleinsten Lerneinheiten beschleunigen und vereinfachen.

Viele wertvolle Materialien existieren bereits und könnten sofort verwendet werden. In diesem Fall werden Autoren ein zusätzlich zu erstellendes Markup nur akzeptieren, wenn das Repositorium eine beträchtliche didaktische Verbesserung bietet. Ein möglicher Aufhänger dafür ist die Tatsache, dass in der Computergrafik und in verwandten Bereichen Programmierbeispiele das Verständnis der Theorie sehr effektiv vertiefen und erweitern, diese Programmierung jedoch oftmals getrennt von der Theorie durchgeführt wird. Deswegen wird hiermit ein neues didaktisches Element eingeführt, die *quelltextbasierte Interaktivität*, welche Lerninhalte mit Lernbeispielen zum Anfassen bereichert. Dieser Ansatz verwendet und kombiniert das Konzept des Textbuches mit dem des technischen Journals, dazu Quelltext-Implementation und Diskussionsforen, wie es zum Beispiel Heckbert in seinem Vorwort zu *Graphic Gems IV* [73] fordert. Damit wird zum ersten Mal Quelltext direkt in Lernmaterialien integriert und dynamisch in die korrespondierenden visuellen Elemente (wie Applets) oder in interaktive Objekte geladen. Die Didaktik quelltextbasierter Interaktivität wird in Abschnitt 5.3 motiviert.

Technisch gesehen bietet das Repositorium neue zusätzliche Dienste an, welche als *"server side compiling"* (serverseitiges Kompilieren) und dazu gehöriges *"server side versioning"* (serverseitiges Versionieren) bezeichnet werden. Die dafür nötigen Erfordernisse für die Realisierung eines Repositoriums mit einem solchen Dienst werden in Abschnitt 5.5 diskutiert, gleichzeitig wird eine Beispielimplementierung vorgestellt. In diesem Zusammenhang wird die Bezeichnung "Quelltext" nicht nur für Software eingesetzt, sondern erweitert und zum Beispiel auch für Pseudocode-Quelltexte, XML-basierte Grafiken oder Flash-Objekte verwendet. Dieser Ansatz erlaubt weiterhin neue Lizenzmodelle wie eine "Teil-OpenSource"-Lizenz, welche von Autoren verwendet werden kann, welche ansonsten auf die Veröffentlichung ihrer Arbeiten im Repositorium verzichten würden, da sie aus Gründen des Urheberrechts oder des Datenschutz möglicherweise dazu gezwungen wären. Unser Ansatz zeigt nur den Quelltext, der für das gewünschte Lernziel vonnöten ist und blendet alle anderen Quelltexte und Bibliotheken aus. Das fördert Datenschutz und Geheimhaltung bei größtmöglicher Granularität, indem bis auf einen einzigen Algorithmus kein weiterer Quellcode offen gelegt wird.

Diese Ideen werden in Abschnitt 5.6 anhand von Beispielen illustriert. Ein Prototyp ist unter der Adresse www.gris.uni-tuebingen.de/gris/ilo online erreichbar, ein Gast-Konto ist eingerichtet, mittels dessen quelltextbasierte Interaktivität eingebettet in Theorieseiten im Verbund mit serverseitiger Kompilierung ausprobiert werden können.

Proposition: The harmonic set

$$u_n(t) = \cos(nt), n=0,1,2,\dots$$

$$v_n(t) = \sin(nt), n=0,1,2,\dots$$

is orthogonal.

Example: Multiply two arbitrary harmonics. Resulting areas sum to zero.

Proof: Consider the scalar product of

- two cosine waves:

$$(u_m, u_n) = \int_{-\pi}^{\pi} \cos(mt)\cos(nt)dt = \begin{cases} 0 & \text{if } m \neq n \\ \pi & \text{if } m = n = 1, 2, \dots \\ 2\pi & \text{if } m = n = 0 \end{cases}$$

- two sine waves:

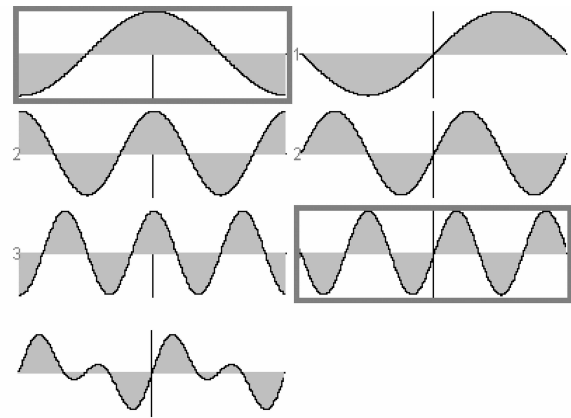


Figure (Harmonic orthogonal set)

Interaction: multiply (click), degree (scroll)

Abbildung 5.1 Der visuelle Beweis für die Orthogonalität harmonischer Sinuskurven. Anwender können durch Skripting spezielle Schwingungen aus den Theorieseiten heraus miteinander multiplizieren. Die Theorie einschließlich mathematischer Formeln wird in XML/MathML formuliert.

5.2 Einordnung

Das CGEMS-Projekt [10] wurde aufgrund der fruchtbaren Diskussionen vor, während und nach dem *Eurographics/SIGGRAPH Workshop On Computer Graphics Education* in Bristol (CGE'02) und in Hangzhou (CGE'04) ins Leben gerufen. Ausgangspunkt waren Ideen während des *Visualization Education Workshop* [34] in Coimbra (GVE'99). Im Augenblick unterstützt das Repositorium die Online-Verwaltung des Begutachtungs- und des Veröffentlichungsworkflows [43]. Während das einem Journal ähnliche Veröffentlichungsmodell darauf angelegt ist, der Qualität der Inhalte und der Anerkennung in der Community zum Durchbruch zu verhelfen, soll in der geplanten Kollaborationsfunktionalität Nutzerrückmeldungen und Versionierung unterstützt werden.

Der Veröffentlichungs- und Begutachtungsprozess von Lehrmaterialien unterscheidet sich von dem eines Forschungsartikels. Dies ist bekannt und wird durch verschiedene Begutachtungen und andere Prozesse nach der Veröffentlichung abgedeckt [84]. Kurz zusammengefasst erfordert die Verwendung von Lernmaterialien in Tutorien oder anderen Umgebungen nach der Veröffentlichung, dass man sich auf keine endgültige Version festlegen darf, sondern die Gemeinschaft dabei unterstützen muss, die Inhalte zu pflegen und weiter zu entwickeln. Lehrende ändern für gewöhnlich das Design der Materialien, natürlich auch die Sprache und Terminologie, so dass diese ihren spezifischen Bedürfnissen entspricht, dabei werden Teile der Inhalte modifiziert, z.B. Folien, Beispiele oder Übungsaufgaben ersetzt oder angepasst. Wichtiger noch ist im Zusammenhang mit der vorgesehen Verwendung durch Lehrende die Möglichkeit, technische Updates durchführen zu können, da die Software-Umgebungen einem ständigen Erneuerungsprozess unterworfen sind. Deswegen wurde auf der CGE'04 der gegenseitige Begutachtungsprozess bei verschiedenen Versionen der Inhalte diskutiert.

Die Verwendung eines kohärenten XML-Schemas zur Auszeichnung der Inhalte wird

beispielhaft im Connexions-Projekt (<http://cnx.rice.edu>, [13]) gezeigt, es handelt sich hierbei um ein Repositorium mit Kollaborationsmodul, welches 1999 von der Rice University gestartet wurde. In diesem Projekt sind einige Standardkonzepte verwirklicht, welche zum Erfolg des Repositoriums beitrugen, inzwischen ist die Materialsammlung auf circa 2200 frei erhältliche Lehrmodule aus verschiedenen Wissensgebieten angewachsen. Die dabei definierte Auszeichnungssprache *Connexions markup language* (CNML) definiert die gemeinsame Basis zur Erstellung der Module, der Autor ist somit nicht mehr frei in der Wahl der Auszeichnungssprache für seine Inhalte. Zur Zeit gibt es nach wie vor keinen allgemein gültigen Standard zur Auszeichnung von Lehrinhalten. Betrachtet man die rund 400 Docbook-Elemente [149], führt dieser Umstand in aller Regel dazu, dass viele Autoren ihre eigene Auszeichnungssprache wählen oder gar selbst definieren, um Inhalte zu organisieren (Zusammenfassung, Abschnitt, Satz, Beweis, Beispiel, Gleichung, Programm usw.), Wissen auszuzeichnen (Konzept, Fähigkeit usw.), und um die Didaktik zu bezeichnen (Übung, Hausaufgabe, Test usw.). Ohne eine allgemeingültige Auszeichnungssprache (dies kann auch eine ganze Gruppe von Auszeichnungssprachen mit dazugehörigen Transformationen sein) kann eine Lernmaterialsammlung Materialien nicht automatisiert verarbeiten. Verschiedene Module können nicht oder nur unter großen Anstrengungen verknüpft oder aneinander gereiht werden. Das Connexions-Projekt dagegen kann Materialien auf verschiedene Art präsentieren, indem einfach die dazu benötigten Transformationen (XSLT) und Stylesheets angewendet werden. Dabei sollte beachtet werden, dass Kompatibilität zu Docbook erwägt werden sollte, da mit der Kompatibilität zu Docbook automatisch eine große Menge an Transformationen nach HTML, PDF, Folien und RTF (Word) verfügbar wären. Connexions bietet Autoren neben einem schnellen Entwicklungsprozess für Inhalte und einer Vielfalt von Veröffentlichungsformaten auch zusätzliche Dienste wie einen Kursgenerator, Anmerkungen und Druckformatierungen an. Hinzu kommt noch ein Plugin für Webbrowser, welches als Ersatz für die Navigation mit gewöhnlichen Hyperlinks nutzerdefinierte Abfolgen und Navigation anbietet, damit es Nutzern möglich wird, Lernpfade durch die Materialien, also die Kurse und Curricula, zu definieren und abzuarbeiten.

Eine weitere Idee, um Autoren dazu zu gewinnen, Inhalte beizusteuern leuchtet ein, wenn man die speziellen Charakteristiken von Lernmaterialien im Gebiet der Computergrafik bedenkt. Im Augenblick sind die besten Auszeichnungsmöglichkeiten für das korrekte Rendern von mathematischen Formeln (MathML), für Grafiken (SVG und X3D) und für parameterbasierte Interaktionen, welche Berechnungen auslösen (MeML [150]), gegeben. Direkte Manipulation grafischer Inhalte, sei es die Ansicht oder das Modell, werden hiermit nicht erfasst. Stattdessen werden proprietäre Sprachen in XML-Dokumente eingebettet, um Texte und Formeln mit visuellen und interaktiven Elementen zu verbinden. Skripting wird meist dafür benutzt, um spezielle Parameterkonfigurationen zu illustrieren, dabei wird nur äußerst selten mehr als oberflächlich an den Lehrmaterialien manipuliert oder gar Teile dieser Materialien in andere übertragen. Hanisch schlägt in [62] vor, wie materialübergreifende Aktionen durch visuelles Skripting in einem Repositorium bewerkstelligt werden könnten. Weitergehende Modifikationen werden in den den Materialien zu Grunde liegenden Quelltexten durchgeführt, zum Beispiel in Java- oder C++-

Quelltexten und den dazugehörigen Makefiles, oder bei Übersetzern, dies ist aber auch der Fall bei Textdateien, Grafiken und Folien (wie den Ausgangsformaten LaTeX, Word, MathPage, Photoshop, Powerpoint und Flash FLA im Gegensatz zu Veröffentlichungsformaten wie PDF, DHTML und Flash SWF). Folgerichtig halten Lehrmaterialsammlungen wie CGEMS oder Merlot [129] Autoren dazu an, die Quellen für die Inhalte beizufügen - als Option, da manche Autoren dies nicht erfüllen mögen oder dürfen, wenn sie beispielsweise die Quelltexte nicht weitergeben dürfen. Damit allein ist es jedoch nicht getan, da die Konfrontation des Lernenden mit Quelltexten und einem Makefile zusätzliche technische Hürden aufbaut und es oftmals erforderlich macht, dass sich der Lernende zusätzliche Softwarepakete installieren muss. Der hier vorgetragene Ansatz des serverseitigen Übersetzens wird hingegen direkt wie andere Dienste wie Versionierung, Anmerkungen und Bewertungen [112] in das Repositorium integriert und kann auf verschiedene Art, entweder als CGI, als Servlet oder als Web Service implementiert werden.

Dieser Vorschlag verwandelt die Quellen von Lernmaterialien selbst in didaktische Elemente. Im Falle von Software resultiert das in quelltextbasierter Interaktivität, welche direkt in Theorieseiten und Diskussionsforen integriert werden kann. Dadurch wird Glassners Idee der Graphic Gems (<http://www.graphicsgems.org>, [49]) erweitert, die Glassner 1990 begründete und später im *ACM Journal of graphics tools* weiterführte. Während jedes Graphic Gem ein eigenes Paket mit Quellen, Makefiles und Installationshilfen darstellt, werden jetzt Lehrmaterialsammlungen mit Quelltextsammlungen zusammengeführt, was das Einbetten von veränderbarem Quelltext in Module ermöglicht, wobei das Kompilieren serverseitig erfolgt und nur Standardfunktionalität von Browsern erfordert.

Die Beispiele wurden von Browns Exploratories-Projekt [87] und Tübingens Applet-basiertem Computergrafikkurs [83] inspiriert. Es wurden interaktive Materialien entwickelt, welche wie ein klassisches Textbuch aussehen, aber gleichzeitig die Möglichkeiten interaktiver Benutzung maximiert. Abbildungen sind interaktive Illustrationen [19], d.h. Softwareelemente ohne Knöpfe, Eingabefelder oder ähnliche Vorrichtungen. Parameter können direkt manipuliert werden, weitergehende Funktionalität wird aus den Theorieseiten heraus ausgelöst oder programmiert. Die Didaktik unterstützt dabei visuelle Kognition und ermöglicht visuelle Beweise [26].

Betrachtet man allein die Kompilierung von Quelltexten im Web, so wurde dies in verschiedenen Projekten bereits erreicht. Beispiele sind ELP (Environment for Learning to Program) [142], ein System, welches in Informatik-Anfängervorlesungen zur Lehre von Programmierung verwendet wird. Dieses unterstützt die wichtigsten imperativen Programmiersprachen wie Java, C++, C und Pascal. Dabei basiert die Eingabe auf einem per Java-Webstart gestarteten Java-Programm, d.h. es ist nicht Browser-basiert. Eine Demoversion ist unter [117] verfügbar.

Im WebToTeach-Projekt [9] wird ebenfalls auf dem Server kompiliert. Der Studierende erhält eine kleine Aufgabe, wobei der Tutor die Ergebnisse auf dem Server anhand von Testfällen überprüft. Meist werden nur kurze Statements vom Studenten verlangt. Das Projekt ging im kommerziellen CodeLab auf, welches die Sprachen Java, C, C++, Fortran, Ada und Pascal unterstützt, und wozu unter [143] eine Demonstrationsseite zu finden ist.

Für C++ existiert ein webbasiertes Projekt von Hitz und Kögeler [74], welche es 1997 vorstellten. Die Eingabe von C++-Lückentexten erfolgt webbasiert, der Anwender bekommt Rückmeldungen vom System über Erfolg oder Misserfolg des Übersetzens. Dabei sollen gezielt verschiedene Programmierkonstrukte eingesetzt werden.

CodeSaw [136] ist hingegen ein kommerzielles Produkt, welches zusammen mit dem Verlag Addison-Wesley Lernmaterialien zum Erlernen von Programmiersprachen anbietet. Die Programmierbeispiele werden als Zusatzmaterialien zu Büchern angeboten. Unterstützt werden die Sprachen C, C++, Java, Perl, Python, Ruby, XML, HTML, PHP, SQL, PL/SQL und JavaScript. Die Auswahl wird ständig erweitert.

Für neue Interpretersprachen gibt es ebenfalls mehrere Projekte, welche diese über ein Webschnittstelle online kompilieren. Als Beispiel sei hier die Webseite <http://tryruby.hobix.com> genannt, welche mittels der Technik Ajax einen interaktiven Interpreter für Ruby [141] stellt. Bei diesem Beispiel wird jedoch immer das ganze Programm auf dem Server ausgeführt und nur das Ergebnis dem Benutzer in den Browser zurückgeschickt.

Ein weitere Anwendung, wie man Browser zur Programmierung von Shader-Programmen, kleinen in einer C-ähnlichen Syntax geschriebenen Programmen, welche auf modernen Grafikkarten direkt ausgeführt werden, verwendet, wurde von Scott Owen [113] vorgeschlagen. Er benutzt das Bitmanagement Contact Plugin [21], um die Programmierergebnisse in einem Browser darzustellen. Da die Shader-Programme nur interpretiert werden, kann man in einer komplett Browser-basierten Softwareumgebung arbeiten.

5.3 Quelltextbasierte Interaktivität

Quelltextbasierte Interaktivität vereinigt drei verschiedene Lehrkonzepte, und zwar (1) visuelle oder interaktive Illustration von (2) Theorie unter Einbeziehung von (3) echten Quelltexten (im Gegensatz zum oftmals verwendeten Pseudocode). Dadurch wird der allseits bekannte Ansatz des "Textbuchs mit Programmquelltexten" in dynamische Webseiten verwandelt, welche Quelltexte interaktiv editierbar präsentieren und auf Nutzereingaben mit dem serverseitigen Kompilieren der visuellen Interaktivität reagieren. Im Folgenden werden die didaktischen Ansätze der quelltextbasierten Interaktivität beschrieben; das technische Framework wird dann in Abschnitt 5.5 vorgestellt.

Die ursprüngliche Idee des Textbuchs mit Programmtexten ist, die umfassende und gute Erläuterung eines Algorithmus mit einer Implementierungsvorlage zu bereichern [73]. Dies ist vor allem auch in mit der Computergrafik verwandten Bereichen wichtig, da hier viele Lernende Schwierigkeiten zu bewältigen haben, wenn sie den Pseudo-Quellcode tatsächlich als Programm umsetzen und ausführen wollen. Hinzu kommt, dass Forschungsartikel sich auf zu Grunde liegende theoretische Aspekte konzentrieren und das als "problemlos" angesehene Programmieren nicht weiter erwähnen, was den Leser mit praktischen Details alleine lässt. Bücher und Online-Journale bieten deswegen zusätzliche Webseiten mit Diskussionsforen und Anwenderanmerkungen an, weiterhin existieren häufig inoffizielle Diskussionsforen, auf denen Quelltexte und Probleme diskutiert werden. Jedoch wird auf diese Foren nicht verwiesen, schon gar nicht aus den Theorieseiten

heraus.

Zunächst bereichert das Einbetten von quelltextbasierter Interaktivität Theorieseiten um interaktive Elemente. Lernende können die tatsächliche Implementation eines Algorithmus direkt ausprobieren und können wesentliche Parameter manipulieren. Anspruchsvolle interaktive Illustrationen wie die Microworlds [114] erlauben zusätzlich die Anpassung der zu Grunde liegenden Modelle, d.h. der internen Teile und Strukturen. Die Verbindung solcher Interaktivitäten mit Theorie kann sehr gut mittels Skripting erreicht werden. Im Beispielbild 5.1 wird gezeigt, wie ein mathematischer Satz mittels einer Illustration visuell bewiesen werden kann. Dabei wurde die traditionelle Art und Weise des "Satz-Beweis-Beispiel" durchbrochen, indem das Beispiel durch eine interaktive Illustration ersetzt wurde. Weiterhin wurde eine passende XML-basierte Auszeichnungssprache, welche mathematische Formeln (MathML) beinhaltet, erstellt. Deswegen können Nutzer den Satz und wichtige weitere Teile der Gleichungen auswählen, um sie um die Interaktivität neu anzuordnen und so das beschriebene Konzept visuell zu erleben. Lehrende können ein solches Modul benutzen, um zuerst die visuelle Kognition des Lernobjekts zu erreichen, und den korrekten Beweis dann erst nach zu liefern.

Ein weiterer Aspekt ist, dass nun Materialien echte funktionierende Quelltexte beinhalten. Inhalte können wie zuvor präsentiert werden, als Textbuch mit Quelltexten, jedoch können Nutzer diese präsentierten Quelltexte manipulieren und verändern, womit sie gleichzeitig die Interaktivität verändern. Deswegen müssen Autoren den Quelltext auszeichnen, was der Auszeichnung der anderen Inhalte entspricht. Im Regelfall werden nur Teile der Quelltexte in die Lehrmaterialien eingefügt, vor allem die nützlichen Teile, welche zum besseren Verständnis beitragen oder aber aus technischen Gründen notwendig sind. Weiterer Quelltext kann dann auf Nachfrage angezeigt werden. Im Abschnitt mit Beispielen (Abschnitt 5.6) sind als so genannte weitere Quelltexte die "siehe auch"-Verweise aufgeführt. Autoren können also Quelltexte offen legen oder den Zugang beschränken im Falle, dass die Quellen ohne Wert für den Lernfortschritt sind oder gar zur Verwirrung beitragen. Lernende können hingegen mit dem Programmieren ausgehend von funktionierenden Quelltexten beginnen, ohne ihren Lernprozess unterbrechen zu müssen.

Dieser Ansatz zeigt damit didaktische Elemente auf, welche früher nur im Rahmen von Laborarbeiten oder Gruppenarbeiten bekannt waren. Autoren können ihre Materialien für verschiedene Lernszenarien aufbereiten (nach [67] [109]).

- Illustration (*Illustration*): Ein Konzept wird anhand einer Beispielimplementierung demonstriert
- Exploration (*Exploration*): Eine erweiterte Illustration mit mehr zugänglichem Quellcode, welcher über das reine Konzept hinausgeht
- Mehrfachauswahl (*Multiple Choice*): Der Lernende kann zwischen Implementierungsalternativen wählen
- Programmierung (*Code Development*): Das Konzept ist vom Lernenden zu implementieren

- Fehlersuche im Quelltext (*Code Debugging*): Typische Fehlerquellen werden präsentiert und sind vom Lernenden zu korrigieren
- Quelltextvergleich (*Code Matching*): Verschiedene Implementierungen werden vom Lernenden nach verschiedenen Aspekten wie Geschwindigkeit, Genauigkeit oder Ähnlichem klassifiziert
- Auswahl im Quelltext (*Within-Code Selection*): Der Lernende muss innerhalb des Quelltextes bestimmte Details identifizieren
- Quelltextbesprechung (*Software-Walkthrough & Discussion*): Hier können Lernende gegenseitig ihre Ergebnisse bewerten und diskutieren

Bei den hier vorgestellten Lernszenarien ist zu beachten, dass durch das serverseitige Kompilieren alle Eingaben der Lernenden verfolgt und somit analysiert werden können. Der im Repository enthaltene Versionierungsdienst speichert jegliche Modifikation der Quelltexte (siehe Abbildung 5.2. Dieser Versionierungsdienst kann auf effektive Art für kollaborative Arbeit verwendet werden: Lernende können ihr Werk mit anderen teilen oder es den Tutoren zeigen. Ein Tutor könnte so beispielsweise eine Lerngruppe beauftragen, ein bestimmte Version der Quelltexte auszuchecken, sie zu überarbeiten und ihren Lösungsvorschlag wieder einzuchecken. Die Mitglieder dieser Lerngruppe können gemeinsam eine Lösung entwerfen, diskutieren und weiterentwickeln, da alle Versionen weiterhin serverseitig abgespeichert werden. Neben der Versionierung muss ein Repository eine Nutzerverwaltung einschließlich einer Rechte- und Rollenverwaltung anbieten. Die gesammelten Materialien des Repositoriums werden also im Laufe der Zeit immer weiterentwickelt.

Derartige visuell-interaktive Illustrationen können auch auf rein optischen Techniken beruhen. Wie oben beschrieben, können Anwender mit den Illustrationen die Quelltexte bearbeiten, ihr Verständnis verbessern und die Illustration weiterentwickeln, ohne dass es sich zwangsläufig um Software handeln muss. Alternative Ansätze könnten das Zielformat serverseitig generieren und anschließend in ein passendes Ausgabeformat transformieren, um dabei das Lernmodul zu erneuern. Als Beispiel für ein solche optische Illustration können SVG-basierte Grafiken genannt werden. Ein Anwender könnte bei einer SVG-Grafik die betreffenden XML-Quellen modifizieren, sie wiederum serverseitig abspeichern und dann neu im Browser mit entsprechendem SVG-Plugin darstellen. Alternativ könnte der Server die SVG-Grafik mittels Stylesheets in ein Bild verwandeln und so die Lernmaterialien als Bild anstelle von SVG ausliefern. Dieser letzte Ansatz bietet deutliche Vorteile im Falle, dass Quelltexte zu schützen sind, beispielsweise in Anwendungsszenarien Übung oder Klausur.

5.4 Motivation für quelltextbasierte Interaktivität

Die *Cognitive Load Theorie* von Sweller [134] beschreibt, wie unter der Annahme, dass die geistige Aufnahmekapazität eines Menschen insgesamt beschränkt ist, verschie-

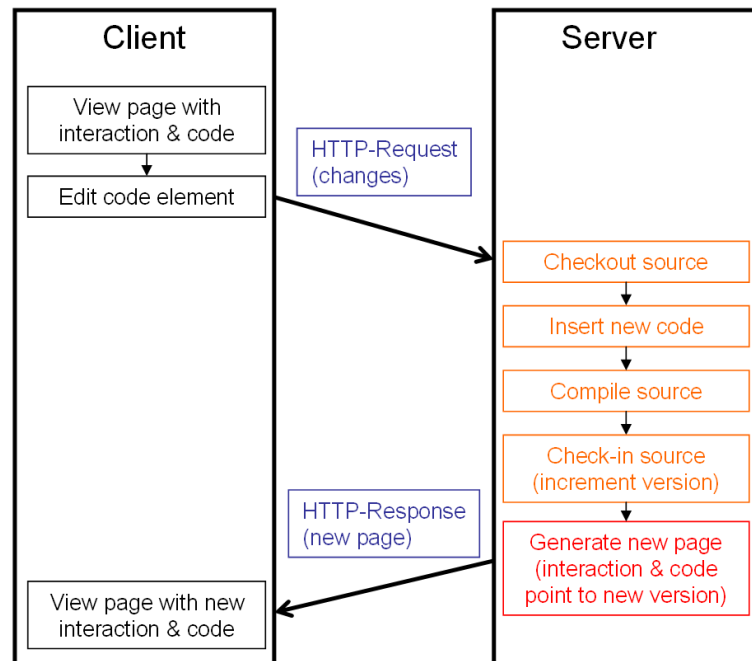


Abbildung 5.2 Illustrative interaktive Applets ("live graphics gems") werden mit echtem Quelltext gesteuert. Das Repositorium bietet Dienste an, welche das Kompilieren und die Versionierung von Anwenderänderungen übernehmen, dies erlaubt Anwendern, eigene Erweiterungen und Veränderungen gemeinsam zu diskutieren und diese weiterentwickelten Materialien per Web zu verteilen.

dene Aufgaben beim Lernen diese Kapazität auslasten. Da nicht alle Aufgaben dem Lernziel dienen, gilt es, die Verteilung zwischen den Aufgaben zu optimieren. Sweller unterscheidet dabei zwischen folgenden drei Lasten (siehe [134], auch kurz dargestellt bei [36]):

- *Intrinsic Cognitive Load*: Hierbei handelt es sich um die Last des Verständnisses eines Lernziels. Sie lässt sich nur verringern, indem dem Lernenden mehr Vorwissen mitgegeben wird.
- *Extraneous Cognitive Load*: Das sind diejenigen kognitiven Lasten, welche beim Lernen anfallen, die jedoch eher begleitender Natur sind. Beispiele können Schwierigkeiten beim Einarbeiten oder Einrichten einer passenden Umgebung zum Erlernen eines Lernziels sein, konkret Schwierigkeiten bei der Installation und Adaption einer passenden Softwareumgebung.
- *Germane Cognitive Load*: Diese Last entspricht der Anstrengung, die beim Lernprozess aufgewendet werden muss, und wird oft auch als Motivation oder Interesse beschrieben. Diese Last kann durch geeignete Aufbereitung der Aufgabenstellung gesteigert werden.

Die Vereinfachung für den Anwender, die darin besteht, dass er das System ohne weitere Installationshürden nutzen kann, die Einbettung in entsprechenden Theorieseiten mit Erläuterungen, Verweisen und Illustrationen und die Leichtigkeit, mit der sich ein Anwender an ein Themengebiet annähern kann, setzt darauf, die *Extraneous Cognitive Load* zu reduzieren zugunsten einer höheren *Germane Cognitive Load* und damit zugunsten eines besseren Lernerfolgs.

5.5 Die Technik der Lernmaterialsammlung

Prinzipiell gesehen sind Repositorien Sammlungen von gemeinsam verwendeten oder gemeinsam zu verwendenden Daten, welche entweder in einer Datenbank oder im Dateisystem abgespeichert werden (siehe auch Abschnitt 3.9). Lernmaterialsammlungen sammeln Materialien zum Gebrauch in Klassenräumen, für Selbststudien und um die Anwendungsgemeinschaft mit Diensten zur kollaborativen Arbeit, zum gemeinsamen Verwenden und zum Weiterentwickeln der Materialien zu versorgen. Allgemeine Dienste beinhalten die Veröffentlichung von Daten mit eindeutiger Referenz (*Uniform Resource Names - URN*), zur Qualitätsbemessung durch Begutachtung und Anwenderbewertungen, Lizenzsysteme, Nutzerverwaltung einschließlich Rechte- und Rollenverwaltung sowie Stöber- und Suchfunktionalitäten [43] [112]. Im Folgenden wird das technische Framework des hierzu entwickelten prototypischen Repositorien beschrieben. Dieses Framework basiert auf XML und bietet für die Java-basierten Illustrationen Dienste zum Auszeichnen und Verwalten der Inhalte, zur Verwaltung und Versionierung der Quelltexte und einen web-basierten Quelltext-Editor an. Obwohl für die hier vorgestellten Applets der Hauptfokus auf Java-Quelltexten liegt, ist das Framework nicht eingeschränkt und kann genauso Quelltexte in anderen Sprachen und Formaten verwalten. Anwender und Lernende können Quelltexte direkt im Browser modifizieren und sie auf dem Server übersetzen lassen, ohne dass zusätzliche Hürden wie Installation von Software oder weitere unabdingliche Arbeitsschritte den Lernprozess behindern.

5.5.1 Eine Auszeichnungssprache für Inhalte

Die Hauptintention bei der Definition einer Auszeichnungssprache war es, Autoren bei der Strukturierung und Kategorisierung der Lehrinhalte behilflich zu sein. Dabei soll das Repository in der Lage sein, die Materialien feingranular zu organisieren und sie in verschiedenen Präsentationsformaten zu veröffentlichen. Für die oben skizzierten Anforderungen lag es nahe, ein XML-Schema zu verwenden, welches einfach in das Docbook-Format [34] transformiert werden kann, was es erlaubt, die vielfältigen, schon vorhandenen Docbook-Transformationen zu Webseiten, nach PDF, in Folien oder in RTF (Word) zu verwenden. Bisher sind die wichtigen Transformationen nach HTML, PDF und SVG bereitgestellt; andere Formate können abgeleitet werden, indem man vorhandene Stylesheets anpasst.

In unserem XML-Schema werden Standardelemente des Lehrbetriebs wie Definitionen, Regeln, Notizen, Probleme, Beispiele, Übungen, Gleichungen, Abbildungen oder auch einfache Paragraphen definiert. HTML- und MathML-Tags können ebenfalls verwendet werden, ihre Schemata bzw. DTDs sind über Namensräume eingebunden. Listen, Tabellen und Textformatierungen folgen in dem Schema, viele dieser Elemente können rekursiv benutzt werden. Die Haupterweiterung ist jedoch die Definition von Interaktionen, Quelltexten und Codeelementen. Interaktionen können wie Abbildungen verwendet werden, wirken also wie Illustrationen, beinhalten jedoch Java-Applets oder Flash-Objekte. Quelltexte und Codeelemente stellen die Implementation oder Teile einer Implementation eines interaktiven Elements dar und werden über ihre Signatur definiert. Während ein Quelltext ein passives Element ist, welches nicht modifiziert werden kann, werden Codeelemente in einem Quelltexteditor dargestellt (siehe unten). Die Auszeichnungssprache bietet desgleichen die Möglichkeit, die in der Aufzählung der Lernszenarien in Abschnitt 5.3 genannten verschiedenen didaktischen Elemente für interaktive Illustrationen zu definieren.

Abbildung 5.3 zeigt, wie das XML-Schema dafür benutzt wird, Lernmaterialien mit interaktiven Illustrationen zu erstellen. Autoren können zur Erstellung solcher Inhalte beliebige XML-Editoren verwenden, je nach Editor werden sie dann mittels Syntax-Highlighting, automatischer Vervollständigung der Auszeichnungselemente und Überprüfung der Elemente unterstützt. Editoren wie der frei erhältliche jEdit (www.jedit.org) bieten außerdem Mittel an, um Stylesheet-Transformationen innerhalb des Editors durchzuführen und sich das Ergebnis der Transformationen anzeigen zu lassen. Dies erlaubt die schnelle Entwicklung (*rapid prototyping*) von Lerninhalten, ohne sie überhaupt ins Repository laden zu müssen. Im gezeigten Beispiel bezeichnet das *Interaction*-Element die Resource mit einem eindeutigen Identifizierer ("dftInter"). Der eingeschlossene *object*-Marker wird als Container verwendet und kann entweder ein Java-Applet oder Flash-Objekte beinhalten. In diesem Beispiel wird ein Java-Applet mit Höhe und Breite zusammen mit der Hauptklasse (*main class*), welche zu starten ist, und zusammen mit einer *codebase*-Auszeichnung, die benötigte Java-Bibliotheken definiert, spezifiziert. Der "src"-Tag deklariert, welche die Zielklasse des serverseitigen Kompilierungsdienstes ist, in aller Regel ist dies der gleiche Wert wie die referenzierte interaktive Illustration. Gleich danach wird der Quelltext in ein Editor-Fenster mit 15 Zeilen Größe eingefügt. Das Code-Element spezifiziert das Applet, zu dem es gehört, dieses wird neu geladen, wenn ein Anwender Quelltexte modifiziert. Der "signature"-Tag beinhaltet einen vollständig qualifizierten Pfad, um das gewählte Quelltextstück in der Quelltextsammlung finden zu können. Da in diesem Beispiel Java-Code verwendet wird, wird die Signatur analog zu Java-Bibliotheken und -Klassennamen ("*package name*" und "*class name*") durch Punkte separiert, gefolgt von Methodennamen und den Parametern; diese müssen auch angegeben werden, um verschiedene Methoden unterschiedlicher Signatur aber gleichen Namens voneinander unterscheiden zu können. Soll der Lernende gleich die ganze Klasse bearbeiten, wird jeglicher Methodennamen ausgelassen.

Die kleinste im Quelltext adressierbare Einheit, welche mit einer solchen Signatur spezifiziert werden kann, ist eine Java-Methode, dies ist konsistent mit den Konventio-

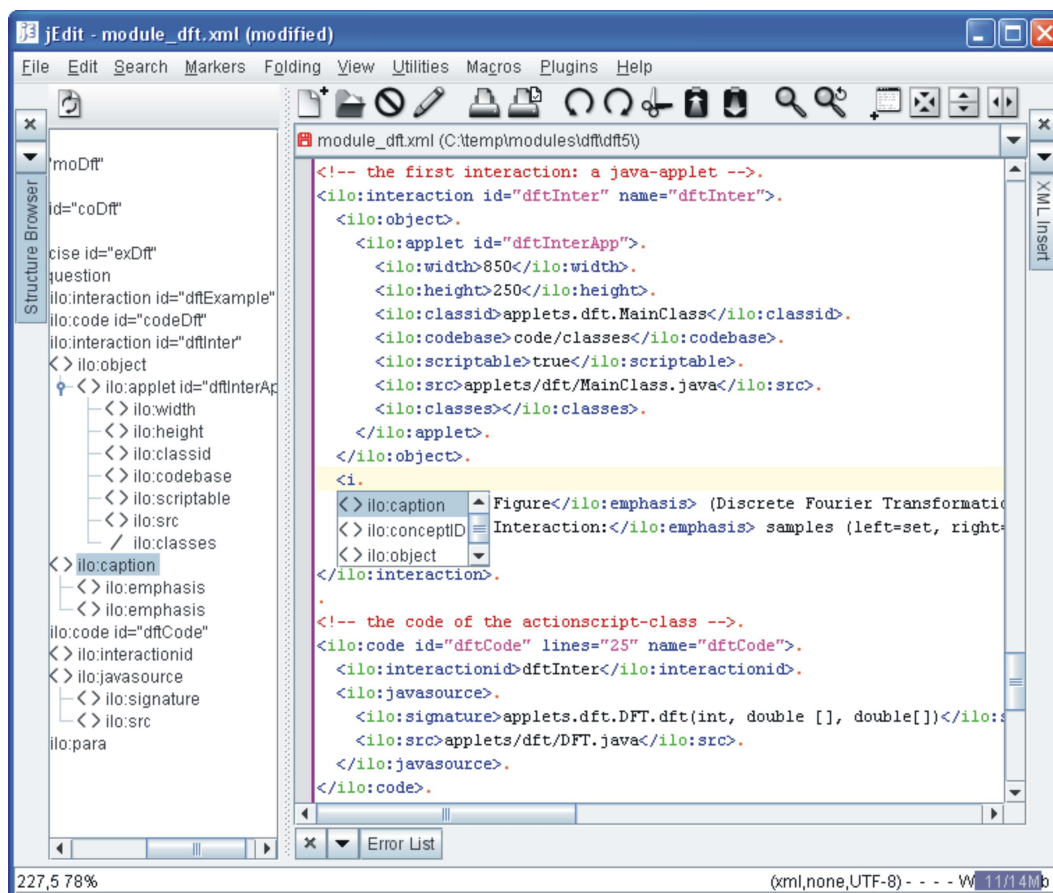


Abbildung 5.3 Eine XML-Auszeichnungssprache für Lehrmaterialien mit interaktiven Illustrationen wurde definiert. Man sieht, wie Lehrmaterialien mit einer Interaktion und dem dazugehörigen Quelltext spezifiziert werden. Autoren werden dabei durch Syntax-Highlighting, Marker-Vervollständigung und Auszeichnungsvalidierung unterstützt. Das Quelltextelement referenziert die Methode, die dem Anwender dargestellt werden soll, mittels ihrer Java-Signatur; der Methodenrumpf wird dann in einem webbasierten Quelltexteditor dargestellt.

nen des Java-Dokumentationssystems JavaDoc. Daher sollten Autoren ihre Materialien so aufbereiten, dass Quelltexte ohne didaktischen Nutzen auf andere Methoden verteilt werden, damit der Lernende in einer ihm präsentierten Methode nur wichtige Inhalte zu sehen bekommt. Um weitere Informationen zu geben, können Java-Quelltexte, welche in Programm-Listings verwendet werden, durch zusätzliche JavaDoc-"siehe auch"-Kommentare ("see also") markiert werden, um auf weitere interessante Methoden zu verweisen.

5.5.2 Verwaltung von Inhalten

Auf der Seite des Servers wird ein Content-Managementsystem benötigt, welches Werkzeuge für Kollaboration und den Veröffentlichungsprozess bietet. Als Grundlage für das Repository wurde Apache Lenya (<http://lenya.apache.org>, [8], siehe Abschnitt

2.4) ausgewählt, ein Cocoon-basiertes Content-Managementsystem, welches seinerseits auf XML/XSL und Java basiert. Damit steht ein funktionierendes Framework mit verschiedenen Basisdiensten zur Verfügung, bei der weiteren Entwicklung konnte das Hauptaugenmerk also auf der Integration der interaktiven Illustrationen liegen. Lenya stellt eine Nutzerverwaltung mit Authentifizierung und Autorisierung zur Verfügung, zur Bearbeitung von Inhalten gibt es bereits verschiedene integrierte webbasierte Texteditoren. Außerdem unterstützt Lenya die Versionierung von Inhalten, Mehrsprachigkeit, Kategorisierung, Zeitplanung und zeitgesteuerte Abläufe, Schablonen für das Layout und Suchfunktionalität. Die von Cocoon bekannte Pipeline (siehe Abschnitt 4.2) steuert basierend auf Dokumenten die Erzeugung von Webseiten durch Aggregation von verschiedenen XML-Quellen, auf welche XSL-Schablonen angewendet werden. Die Resultate werden dann als XHTML an den Browser gesendet. In Lenya wird dadurch wiederum der Inhalt vom Aussehen getrennt und eine einheitliche Darstellung ("look & feel") geschaffen. Dabei wird auf Seite des Servers jegliche Ablauflogik in der Pipeline gekapselt und somit getrennt verwaltet, andere Teile des Ablaufs werden in Java- oder JavaScript-Bibliotheken vorgehalten.

Lenya erleichtert die Erstellung von Inhalten, indem das Einstellen der Inhalte und der Metadaten Schritt für Schritt vom Autor erfragt werden. Die Verwaltung der Benutzer im System ist unkompliziert: Administratoren definieren Rechte und Rollen, um verschiedene Zugangsrechte zu den Materialien zu gewähren. Deswegen können Lehrende und Lernende die Inhalte nur in veröffentlichten Formaten anschauen, während Autoren und andere berechtigte Benutzer auch Zugang zu den Quellen der Materialien haben.

Nach dem Anpassen des Aussehens des Repositoriums durch einfache Modifikationen in einigen Stylesheets, wurde die Workflowengine erweitert, um ein einfacheres Hochladen und ein einfacheres Arbeiten mit interaktiven Illustrationen zu ermöglichen. Lenya-Workflows werden durch einen endlichen Zustandsautomaten abgebildet. Entweder ein Benutzer des Repositoriums oder die Zeitsteuerung von Lenya starten ein Workflow-Ereignis durch Aktivierung einer Hyperlinks oder Abschicken eines Webformulars. Dieses Ereignis ändert den Zustand des Workflows und sorgt für das Auslösen aller Aktionen, welche für die betreffende Transition spezifiziert sind. Für jegliche Transition können Bedingungen zum Beispiel bezüglich der Rollen und Rechte gestellt werden. Der Lenya-Workflow-Zustandsautomat überprüft diese Bedingungen und löst die Aktionen automatisch aus. Die Workflow-Zustände werden dabei anmeldungsübergreifend abgespeichert, d.h. selbst wenn Anwender sich ausloggen, wird ein angestoßener Workflow weiter ausgeführt. Um neue Dienste zu definieren, werden neue Workflow-Aktionen als Java-Objekte hinzugefügt. Als Beispiel dient der Datei-Upload; weiterhin wird das serverseitige Übersetzen und die Versionierung der Quelltexte im nächsten Abschnitt erläutert.

Workflow-Aktionen betreffen immer Dokumente eines bestimmten Typs, deswegen wurde der Dokumenttyp "Modul" neu definiert, welcher ein zip-komprimiertes Paket aus XML-Daten, Quelltexten und weiteren Dateien wie Bildern und Bibliotheken (meist Java-Archive - JAR) darstellt. Alle zum Kompilieren notwendigen Quelltexte und Materialien müssen enthalten sein. Immer wenn ein Autor ein neues Modul ins System hochlädt, entpackt der Repositoriumsworkflow das komprimierte Paket und erstellt eine ini-

tiale Version aller Dateien. Jedes Modul erhält eine eindeutige Identifikation (URN), um die Gültigkeit von Verweisen auf Materialien im Repositorium in Textbüchern oder wissenschaftlichen Arbeiten über Jahre zu garantieren. Dabei werden nicht Verweise der referenzierten Materialien gespeichert sondern die Inhalte selbst, wie es sich angesichts der sich schnell ändernden Web-Inhalte empfiehlt. Autoren ordnen ihre hochgeladenen Materialien ein, diese Informationen nutzt das Repositorium zur Indexierung und erweiterten Suche.

5.5.3 Serverseitiges Übersetzen

Das dynamische Übersetzen der Quelltexte geschieht vollständig auf der Seite des Servers. Jedesmal, wenn ein Benutzer mit interaktiven Illustrationen mit Quelltextelementen interagiert, wird serverseitig ein Ereignis, welches die Übersetzung anstößt, ausgelöst. Die dazu gehörige Workflow-Aktion bestimmt den Ablauf. Erst werden die Quelltexte gemäß den Benutzeränderungen angepasst, dann wird das Interaktions-Ziel übersetzt, etwaige Übersetzungsfehler werden verwaltet und ausgegeben oder es wird im Falle von fehlerfreier Übersetzung eine neue Seite mit der neuen Version der Materialien zusammengestellt und dem Benutzer angezeigt (siehe Abbildung 5.2).

Die Workflow-Aktion erstellt dabei zunächst eine neue Version der Quelltextdateien in Abhängigkeit der Identität des jeweiligen Benutzers. Dabei ersetzt die Aktion Code-Teile der Originalversion durch die vom Benutzer geänderten Texte und ruft im Anschluss den Java-Compiler auf. Verschiedene Versionen der Ausführungsumgebung auf Seiten der Anwender können durch entsprechende Flags bei der Übersetzung unterstützt werden, was dazu führt, dass die erzeugten Klassendateien kompatibel zur Ausführungsumgebung des Nutzers sind. Die erzeugten Binärdateien werden ebenfalls auf dem Server gespeichert. Dem Anwender wird eine neue Seite mit angepassten Inhalten und angepassten interaktiven Illustrationen präsentiert. Die auf der Seite enthaltene Illustration verweist auf die neu übersetzte Version, ebenfalls werden in einem Quelltexteditor die modifizierten Texte angezeigt. Der Anwender sieht das Ergebnis seiner Modifikationen. Wenn das Übersetzen fehlschlagen sollte, da Programmierfehler eingebaut wurden, werden die Fehlermeldungen des Übersetzers auf der gleichen Seite dem Benutzer zur Korrektur angezeigt.

Im Regelfall wird es nur einem authentifizierten Anwender gestattet, das serverseitige Kompilieren einschließlich des Workflow anzustoßen. Dies ist nicht nur dem Rechtemanagement geschuldet, sondern wird vor allem zur Identifikation benötigt. Jedem Anwender soll seine zuletzt erstellte Version gezeigt werden. Diese spezielle Version muss erst geladen werden, um dem Benutzer eine Seite mit den Lehrinhalten auszuliefern. Anonyme Anwender bekommen dagegen eine Seite mit den initialen Inhalten und müssen sich, so sie Änderungen durchführen wollen, erst am System anmelden.

Wie oben schon gesagt wurde, ermöglicht dieses hier vorgestellte Modell des Übersetzens von Quelltexten auf Seite des Servers ein neues Lizenzmodell der "teilweise offenen gelegten Quellen", um dadurch Geheimhaltung zu erzielen beziehungsweise Urheberrechte zu befolgen. Das Repositorium präsentiert dem Anwender nur die markierten

Quellen eines Lernobjekts, alle anderen Quellen, welche der Autor wegen des Lernerfolgs oder aus anderen Gründen nicht offen legen will, werden ausgeblendet und dem Nutzer nur in Form von binären Inhalten in die Hand gegeben. Dieses Modell funktioniert auf der Ebene von Paketen, Klassen und auch Methoden, also hinreichend feinkörnig. Im Fall von Methoden werden die Quelltexte, welche die Methode beinhalten, zum Übersetzen dem System zur Verfügung stehen; sie werden jedoch nicht dem Anwender gezeigt.

5.5.4 Versionierung der Quelltexte

Um Quelltextversionen verschiedener Anwender effizient zu verwalten, empfiehlt es sich, ein Versionskontrollsystem zu verwenden. Dabei soll gewährleistet werden, dass jeder Nutzer eine eigene private Version seines webbasierten Arbeitsbereiches erhält. Zu jeder Zeit sollten Anwender eine vorherige Version oder auch die initiale Version auswählen können, um mit ihren Modifikationen neu starten zu können. Desgleichen soll es Benutzern ermöglicht werden, ihre Arbeit und somit ihre Quelltextversionen mit anderen Anwendern zu teilen, d.h. andere ihre Arbeit bewerten zu lassen oder eine Entwurfsversion ihrem Tutor zukommen zu lassen. Außerdem erlaubt ein Versionskontrollsystem, einfache Multiple-Choice-Tests und ähnliche didaktische Elemente zu verwenden, welche darauf beruhen, alternative Quelltexte zu präsentieren.

Für die am WSI/GRIS entwickelte Software wurde Subversion [33] als Versionskontrollsystem gewählt. Subversion ist ein Nachfolger des *Concurrent Versions Systems* (CVS), eines der meist verbreiteten Systeme zur Versionskontrolle. Subversion ist für viele Betriebssysteme frei erhältlich. Im Gegensatz zu CVS bietet Subversion einige Erweiterungen wie Versionierung ganzer Verzeichnisse, atomare Commits, versionierte Metadaten, effiziente Versionierung von Binärdaten, verschiedene Netzwerkprotokolle zur Anbindung von Klienten und eine effiziente Verwaltung von Kopien ganzer versionierter Verzeichnisse (*branching*). Gerade letzteres ist für das implementierte Repository von großer Bedeutung: Branching, welches für die Verwaltung von Entwicklungszweigen gedacht ist, kann für verschiedene Nutzer erstellte Kopien von Verzeichnissen effizient verwalten. Die Kommunikation des Repositoriums und den Workflow-Aktionen mit dem Subversion-Server erfolgt über eine frei erhältliche Java-Schnittstelle, welche die gleiche Mächtigkeit wie der Kommandozeilen-Client von Subversion besitzt.

Die hier vorgestellte Lösung speichert jede Modifikation der Quelltexte in Subversion ab. Wenn ein Anwender zum ersten Mal Quelltexte bearbeiten will, erstellt der Server in einem neu erstellten Verzeichnis Kopien der ursprünglichen Quelltextdateien. Dieses Verzeichnis wird nach der eindeutigen Benutzer-Identität benannt. Modifizierte Texte werden dabei gleich eingefügt bzw. ersetzt. Da bei Java der Paketname die Verzeichnisstruktur reflektiert, muss bei allen Quelltexten der Paketname entsprechend angepasst werden. Nachdem die Quelltexte übersetzt sind, wird diese neue Version in Subversion eingchecked. Da jeder Benutzer sein eigenes Verzeichnis hat, in dem initial eine Kopie der Originaldaten liegt, werden diese Daten als Branch in Subversion gespeichert. Deswegen werden die Dateien von Subversion stark komprimiert. Um die Arbeit in Kleingruppen zu ermöglichen, was dadurch geschieht, dass Anwender einer Gruppe sich von verschiede-

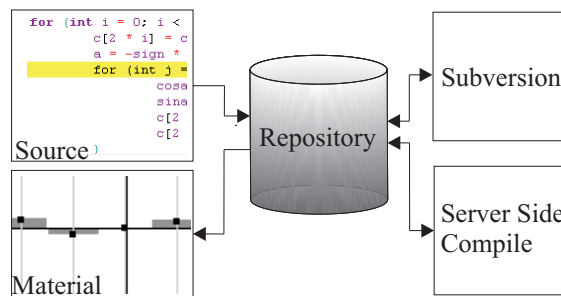


Abbildung 5.4 Quelltextbasierte Interaktivität benutzt richtigen Programmcode. Veränderungen von Nutzern werden auf dem Server übersetzt und versioniert, die neue Version wird dann wieder zum Benutzer gesendet.

nen Rechnern einloggen, müssen sie, wenn sie die Quellen nicht nur studieren, sondern auch modifizieren wollen, sich um eine Sperre (*lock*) bewerben, welche eindeutig verteilt wird - nur der Anwender mit Sperre hat das Recht, auch wirklich Änderungen durchzuführen.

Dieser Ansatz erfordert eine durchdachte Nomenklatur für Nutzer und Versionen, um die Quelltexte zu benennen. Abgesehen von der dauerhaften Speicherung in Abhängigkeit vom Anwender, muss die Nomenklatur auch dafür sorgen, dass jegliche Caching-Mechanismen verhindert werden. Caching gibt es bei Webbrowsern, bei Proxies und im Java-Plugin des Browsers. Dieser Effekt könnte dazu führen, dass eine neue Version einer Klassendatei nicht neu geladen würde, wenn sie unter dem alten Namen weitergeführt würde. Deswegen wurde die Namensgebung nach dem Muster "Benutzer-Identität": "Versionsnummer":"originaler Java-Klassenname" entworfen. Dies garantiert, dass für jede Version neue Klassendateien mit neuen, dem Browser unbekannt Namen, welche der Browser in der gewünschten Version herunterlädt, erstellt werden. Klassendateien können dadurch in der Versionskontrolle leicht erkannt und anderen Klassen korrekt zugeordnet werden.

Die erzeugten Binärdateien werden nicht im Versionskontrollsystem abgelegt, da sie einerseits dort nicht gebraucht werden und andererseits nicht so effizient wie die Textdateien gespeichert werden können. Gleichfalls werden Software-Bibliotheken und ähnliche weitere Ressourcen direkt im Dateisystem gespeichert, wo sie dem Link-Prozess und dem Browser auf Seite des Klienten zur Verfügung stehen.

5.5.5 Ein webbasierter Quelltexteditor

Die Programmierung im Internet basierend auf dem Zugang per Webbrowser ist vor allem im Falle, dass eine integrierte Entwicklungsumgebung (*Integrated Development Environment* - IDE) nicht vorhanden ist oder deren Einrichtung den Lernprozess unterbrechen würde, vorzuziehen. IDEs sind relativ umfangreiche Softwarepakete, welche der Anwender herunterladen, installieren und einrichten müsste - Aufgaben, welche oft Hürden für die Nutzer sind. Gelegentliche Anwender werden diesen zusätzlichen Ar-

beitsaufwand nicht akzeptieren und haben möglicherweise auch gar nicht die technischen Voraussetzungen dazu, weil ihnen beispielsweise die nötigen Rechte zur Installation oder benötigte Bibliotheken fehlen. Selbst die Studenten am Lehrstuhl, welche durch ihr Studium Hintergrundwissen zur Programmierung erworben haben, schaffen es nicht immer auf Anhieb oder brauchen gar Hilfe, wenn sie zur Lösung von Übungsaufgaben oder im Rahmen der gegenseitigen Begutachtung die richtigen Arbeitspakete auswählen und einrichten sollen. Des Weiteren sind viele IDEs wenig oder gar nicht darauf eingerichtet, innerhalb eines webbasierten Arbeitsflusses verwendet zu werden oder gar Ereignisse auf Webservern auszulösen, wie es im hier vorgestellten Ansatz der interaktiven Illustrationen benötigt wird.

Wenige Anwender würden gerne Quelltexte in Form von einfachem Text bearbeiten. Deswegen wird für das Repositorium eine Lösung gebraucht, welche Anwendern die Bearbeitung erleichtert, selbst wenn nur Webbrowser und JavaScript zur Verfügung stehen. Das Minimum an Unterstützung für Nutzer des Repositoriums stellt ein Editor für Quelltexte mit Syntax-Highlighting und automatischem Einrücken dar. Ein solcher Editor ist Helene (<http://www.muze.nl>). Helene ist frei verfügbar und komplett in JavaScript geschrieben, unterstützt allerdings in der derzeitigen Version nur Internet-Explorer und Mozilla-Browser (also Browser mit der Gecko-Engine). Kleine JavaScript-Klassen ermöglichen Syntax-Highlighting, automatisches Einrücken und einen Zeilenzähler. Eigentlich für die Skriptsprache PHP sowie JavaScript entworfen, war die Erweiterung dieses Editors auf Java als Programmiersprache relativ leicht zu implementieren.

Code-Ergänzung (code completion), eine integrierte Dokumentation zu den verwendeten Java-Klassen und jegliche Fehlerprüfung (debugging) sind im webbasierten Editor nicht verfügbar. Deswegen eignet sich die rein webbasierte Variante vor allem für kurze Quelltextauszüge von 10-20 Zeilen Länge, d.h. Algorithmen sind in diesem Texteditor gut zu bearbeiten. In den unter Studenten durchgeführten Befragungen ergaben sich keine größeren Schwierigkeiten mit solch relativ kurzen Code-Fragmenten. Für längere Textfragmente wird empfohlen, die Arbeit in einem lokalen Editor oder einer lokalen Entwicklungsumgebung vorzubereiten, um sie dann in den webbasierten Editor zu kopieren und hochzuladen. Die Möglichkeit, mittels selbst entwickelten Eclipse-Plugins aus einer Entwicklungsumgebung heraus die Quelltextentwicklung vorzunehmen, ist in Planung. Dieses Plugin würde gegenüber dem Repositorium wie ein Webbrowser auftreten und böte dem Programmierer den üblichen Komfort einer Entwicklungsumgebung. Da Versionen weiterhin im Repositorium gespeichert sind, könnte der Anwender die Ergebnisse wiederum in einem Browser betrachten.

5.6 Einsatzszenarien

Für Einsatzszenarien von interaktiven Illustrationen gibt es mehrere Beispiele. Ein erstes wurde 2005 vorgestellt. Mittels interaktiver Illustrationen werden Lehrmaterialien zur Abtasttheorie aufbereitet. Diese Theorie ist eine wichtige Grundlage bei vielen Themen der Computergrafik und auch anverwandter Fachgebiete. Sie wird beispielsweise bei

Glassner [50] ausführlich vorgestellt. Im klassischen Computergrafik-Curriculum [45] dient diese Lehreinheit zur Behandlung des Phänomens der Aliasing-Effekte und als Basis von Bildverarbeitung, hier insbesondere von Filterung. Zu diesen Fachgebieten gibt es sehr gute Lehrmaterialien im Web, zum Beispiel MathWorld (www.mathworld.com) mit verständlichen Zeichnungen, Bourkes Webseiten (astronomy.swin.edu.au/~pbourke) mit Quelltextauszügen, oder die entsprechende Lerneinheit des Connexion-Kurses. Zusammen mit den Lernobjekten aus Tübingen [83] und dem Exploratories-Projekt [87] werden sie als Lerninstrumente in Tutorien und als Selbstlernhilfen für Studenten verwendet.

In den genannten Materialien wird die Theorie von den interaktiven Experimenten getrennt dargestellt, die Quelltexte sind meist nur in Anhängen zu finden. Untereinander sind die Materialien nicht austauschbar, da sie verschiedene Namensgebungen, verschiedene Lösungsansätze und verschiedene Ansichten über Problematik bieten. Im Fall der Abtasttheorie wird die Lehre und das Lernen erschwert, weil die Fouriertransformation verschiedene Normalisierungen an verschiedenen Stellen benutzt, weil die Fourierkoeffizienten oftmals verschieden angeordnet werden und weil es verschiedene Visualisierungen des Fourier-Spektrums gibt.

Die hier vorgestellten überarbeiteten Materialien integrieren die drei Lehrkonzepte Theorie, interaktive Illustration und verwendeten Quelltext dagegen auf einer einzigen Webseite. Das hier vorgestellte Beispiel (Abbildung 5.5) zeigt ein Lernmodul, welches die diskrete Fouriertransformation (DFT) auf einer Webseite erläutert. Beim ersten Hinsehen sieht die vorgestellte Seite wie ein Textbuch mit Programmtexten und Grafiken aus, als einzige Ausnahme erkennt man einen "Execute"-Knopf. Tatsächlich sind alle Bilder und die Quelltexte dynamisch. Anwender können die Illustrationen bearbeiten, zum Beispiel indem sie die Abtastrate und die Abtastwerte entweder im Orts- oder im Frequenzraum verändern. Während der Interaktionen werden die dargestellten Signale in beiden Domänen ständig den neuen Werten angepasst. Der Programmquelltext, hier auf den Kernalgorithmus einer DFT reduziert, kann ebenfalls verändert werden. Klickt ein Anwender auf den "Execute"-Knopf, werden die Eingaben des Benutzers an das Repository gesandt, welches diese Eingaben in die restlichen Quelltexte an der richtigen Stelle einfügt, diese als neue Version abspeichert, sie kompiliert, und das neu erzeugte Kompilat an den Anwender als neue interaktive Illustration ausliefert, indem es die Webseite mit den neuen Daten aktualisiert. Die Rückmeldungen des Übersetzers werden dann ebenfalls auf der Webseite ausgegeben.

Lehrende können derartige Lehrmodule als Basis ihrer Übungsaufgaben verwenden. Als Beispiel könnte ein Lehrender eine nicht funktionierende Version der DFT auf der Webseite initial anbieten, um den Lernenden dann aufzufordern, eine korrigierte Version zu entwickeln. Oder Lernende sollen eine DFT durch eine schnelle Fouriertransformation (FFT) ersetzen. Da jede Version auf dem Server abgespeichert wird, können Lehrende nachvollziehen, welche Verständnisschwierigkeiten häufig auftreten und entsprechend ihren Unterricht anpassen.

Die Software ist in Java geschrieben, normalerweise sollte der Autor seine Algorithmen mit JavaDoc-Kommentaren versehen haben. Ähnliche Kommentarauszeichnungen werden verwendet, um die Parameter der fraglichen Methode zu erläutern oder Verweise

GRIS/ILO Interactive Learning Objects - Microsoft Internet Explorer

File Edit View Favorites Tools Help Address D:\LocalProject\edu\modules\zip\dft\index.html

Discrete Fourier transformation

Sampling a continuous signal f creates discrete samples $f(m) = f(x_0 + \Delta x)$, $m = 0, \dots, N-1$ and simplifies the Fourier integral to

$$F(k) = \frac{1}{N} \sum_{m=0}^{N-1} f(m)(\cos(2k\pi) - i\sin(2k\pi)), \quad k = 0, \dots, N-1$$

$$f(m) = \sum_{k=0}^{N-1} F(k)(\cos(2k\pi) + i\sin(2k\pi)), \quad m = 0, \dots, N-1$$

Figure (Discrete Fourier transformation)
A sampled periodic signal in spatial domain (left) is the sum of harmonics (cosine=center, sine=right).
Interaction: samples (left=set, right=clear), sampling rate (drag N), reset (esc)

Algorithm: (Discrete Fourier transformation)

Parameters

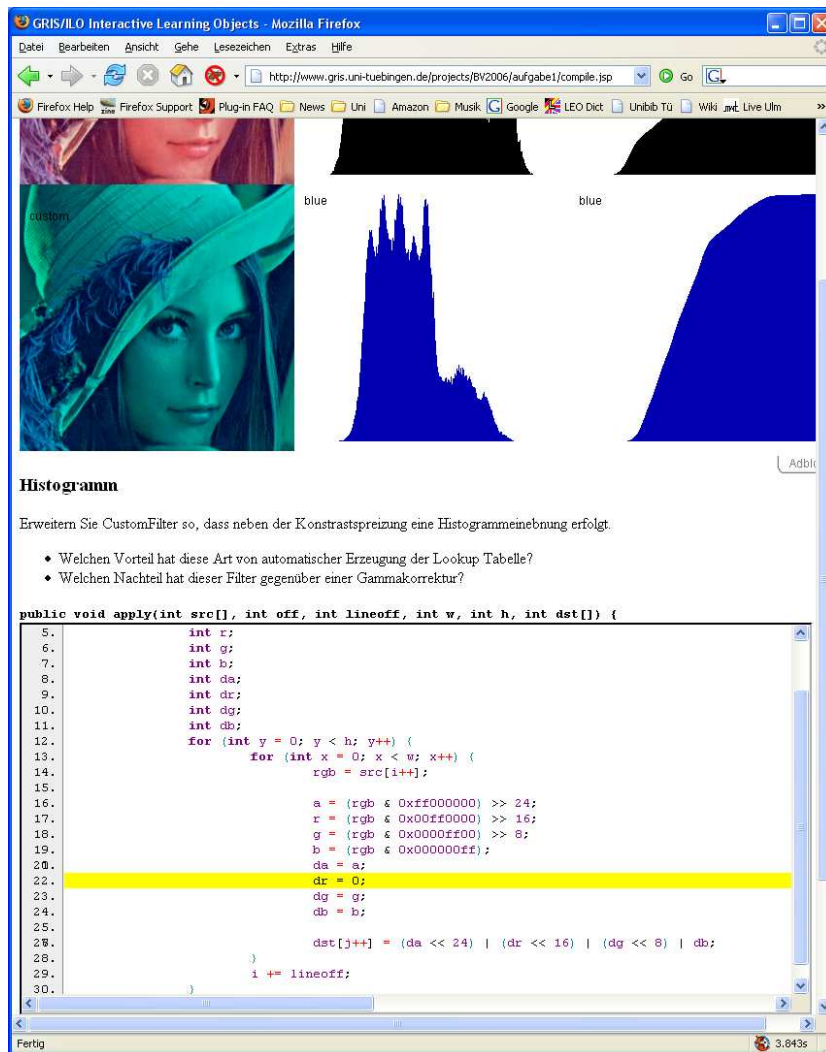
```
void dft(int sign, double[] re, double[] im) {
1. int n = re.length;
2. double a, cosa, sina;
3.
4. for (int i = 0; i < n; i++) {
5.     c[2 * i] = c[2 * i + 1] = 0;
6.     a = -sign * 2 * Math.PI * i / (double) n;
7.     for (int j = 0; j < n; j++) {
8.         cosa = Math.cos(j * a);
9.         sina = Math.sin(j * a);
10.        c[2 * i] += (re[j] * cosa - im[j] * sina);
11.        c[2 * i + 1] += (re[j] * sina + im[j] * cosa);
12.    }
13. }
}
```

Execute

See also:

- real () Returns the transformed samples' real-valued part.
- imaginary () Returns the transformed samples' imaginary-valued part.
- magnitude () Calculates the result's amplitude.
- phase () Calculates the result's phase.
- setScale () Normalizes with 1, 1/N, or 1/√N.
- setOrdering () Sorts Fourier coefficients symmetric about the Nyquist rate, DC, or removes negative ones.

Abbildung 5.5 Das serverseitige Kompilieren verwandelt ein normales Textbuch mit Programmtexten in dynamische Lehrmaterialien. Abtastwerte und die Abtastrate können direkt im Orts- und im Frequenzraum verändert werden. Die Implementation der Fouriertransformation kann desgleichen direkt bearbeitet werden. Da die Quellen vom Repository verwaltet, übersetzt und versioniert werden, können die Materialien dynamisch verändert werden.



Histogramm

Erweitern Sie CustomFilter so, dass neben der Kontraststreckung eine Histogrammebnung erfolgt.

- Welchen Vorteil hat diese Art von automatischer Erzeugung der Lookup Tabelle?
- Welchen Nachteil hat dieser Filter gegenüber einer Gammakorrektur?

```
public void apply(int src[], int off, int lineoff, int w, int h, int dst[]) {
5.     int r;
6.     int g;
7.     int b;
8.     int da;
9.     int dr;
10.    int dg;
11.    int db;
12.    for (int y = 0; y < h; y++) {
13.        for (int x = 0; x < w; x++) {
14.            rgb = src[i++];
15.
16.            a = (rgb & 0xff000000) >> 24;
17.            r = (rgb & 0x00ff0000) >> 16;
18.            g = (rgb & 0x0000ff00) >> 8;
19.            b = (rgb & 0x000000ff);
20.            da = a;
21.            dr = 0;
22.            dg = g;
23.            db = b;
24.
25.            dst[j++] = (da << 24) | (dr << 16) | (dg << 8) | db;
26.
27.
28.
29.            i += lineoff;
30.        }
    }
}
```

Abbildung 5.6 Lehrmodul Histogrammebnung, aufbereitet als Aufgabe, wie sie in der Vorlesung Bildkommunikation I im Sommersemester 2006 am WSI/GRIS gestellt wurde

auf weitere, ähnliche Methoden zu geben. Diese werden dann direkt unter dem Quelltext auf der Webseite eingefügt. Anwender können diese Methoden auf- und zuklappen, um sich zusätzliche Dokumentationen oder weitere Quelltexte anzuschauen.

Die unter Abbildung 5.5 gezeigten Materialien können unter <http://www.gris.uni-tuebingen.de/gris/ilo> eingesehen werden. Das Anmelden am System erfolgt über ein bereit gestelltes Gast-Konto. Als Browser sind entweder Internet Explorer oder Mozilla-basierte Browser erforderlich.

In weiteren Beispielen wurde das System im Rahmen der Vorlesung Bildverarbeitung I im Sommersemester 2006 am WSI/GRIS verwendet. Hierbei wurde es für Übungsaufgaben eingesetzt, bei denen Studenten in Gruppen ihre Programmieraufgaben online erarbeiten und abgeben konnten. Hier wird die Aufgabe Histogrammebnung vorgestellt.

Bei der Aufgabe der Histogrammebnung geht es darum, das Histogramm eines

Bildes durch eine Spreizungsoperation so anzupassen, dass eine möglichst gleichmäßige Verteilung der Intensitätsstufen (oder Farben) entsteht. Betrachtet man das kumulierte Histogramm, wird dieses durch eine Einebnung möglichst gut an die erste Winkelhalbierende angepasst, Unregelmäßigkeiten, d.h. Abweichungen von dieser entstehen dabei nur durch die Diskretisierung. In Abbildung 5.6 ist das Applet mit der Aufgabenstellung zu sehen, unter der Aufgabenstellung folgt direkt der Quelltexteditor, welcher die Filtermethode darstellt. Das Applet zeigt links oben (ein wenig abgeschnitten) verschiedene Originalbilder, rechts davon das Histogramm für Grauwert, den Rot-, Grün- oder Blaukanal und ganz rechts das kumulierte Histogramm ebenfalls für Grauwerte, oder den Rot-, den Grün- oder den Blaukanal. In der zweiten Zeile sieht man das Bild, nun aber nach einer Filteroperation, welche dem im Quelltexteditor aufgeführten Programmtext entspricht. Dann kommt das Histogramm des Blaukanals sowie das kumulierte Histogramm des Blaukanals. Zum besseren Verständnis kopiert der Filter im dargestellten Beispiel alle Farbwerte des Eingangsbildes in das resultierende Ausgangsbild bis auf den roten Farbkanal, welcher auf Null gesetzt wird (zu sehen in der gelb hinterlegten Zeile). Mangels roten Farbkanals erscheint das gefilterte Bild bläustichig.

Eine weitere Aufgabe, diesmal zum Thema Houghtransformationen, ebenfalls im Rahmen der Vorlesung Bildverarbeitung I im Sommersemester 2006 am WSI/GRIS gestellt, ist in Abbildung 5.7 zu sehen. Man sieht wiederum im oberen Bereich die Aufgabenstellung, darauf schließt sich das Applet an, welches bearbeitet werden soll, dann werden verschiedene Quelltexteditorfenster präsentiert, von welchen man nur den ersten sieht. So können Studenten mehrere Methoden aus verschiedenen Klassen gleichzeitig auf einer Seite editieren. Um die Erarbeitung der Lösung zu erleichtern, empfiehlt es sich, die Aufgabe so aufzubereiten, dass pro Aufgabenteil nur eine Methode komplettiert werden muss. Dies kann bedeuten, dass man die Lösung ausfaktoriert, um jeweils für Teilaufgaben den Rest der nötigen Schritte eines Algorithmus dem Studenten an die Hand geben zu können.

Die Houghtransformation ist ein Verfahren, um Gerade oder Kreise oder andere parametrisierbare Gebilde in einem Bild zu erkennen. Sie wurde 1962 von Paul V. C. Hough entwickelt. Das Verfahren läuft in drei Schritten ab. Zuerst werden aus dem Originalbild mit einem geeigneten Operator (zum Beispiel mit dem Sobel-Operator) alle Kanten extrahiert. Im daraus entstandenen Grauwertbild werden durch Punkte, welche auf einer Kante liegen, alle möglichen Geraden (oder Kreise bzw. andere Gebilde) gelegt und im Houghraum (oder Dualraum) aufaddiert. Für die tatsächlichen Geraden gibt es im Houghraum somit Häufungspunkte, welche man per Schwellwerterkennung auswertet und somit die genauen Parameter der gesuchten geometrischen Gebilde im Ausgangsbild gefunden hat.

Es gibt für Studenten also drei Schritte zu programmieren: Filterung des Originalbildes mit dem Sobel-Operator, Akkumulieren von Houghtransformierten im Dualraum und Erkennen und Einzeichnen der gefundenen Formen im Ausgangsbild. Für manchen Schritt mag ein vorgegebener Akkumulator nützlich sein, welcher aber in einem späteren Schritt von Studenten noch zu implementieren ist. Dafür kann man eine solche Methode als Kompilat vorgeben, und die Aufgabe dann im Rahmen einer anderen Klasse in einem

Abbildung 5.7 Lehrmodul Houghtransformation, ebenfalls als Aufgabe in der Vorlesung Bildkommunikation I im Sommersemester 2006 am WSI/GRIS gestellt

späteren Schritt stellen.

Ein letztes Beispiel zeigt, dass dieses System sich prinzipiell für alle Programmiersprachen eignet, für die ein Plugin für Browser existiert. Hierbei wird zu Demonstrationzwecken ein einfaches Flash-Programm ebenfalls als Modul auf dem Server gespeichert (siehe Abbildung 5.8). Da es inzwischen frei verfügbare Compiler für Flash gibt, konnte mittels des gezeigten Moduls die Sprachunabhängigkeit demonstriert werden.

5.7 Evaluierung

Die hier vorgestellte Plattform wurde in mehreren Feldversuchen eingesetzt und evaluiert. Der größte Feldversuch geschah im Rahmen der Vorlesungen Grafische Datenverarbeitung II und Bildverarbeitung I, beide gehalten im Sommersemester 2005, unter Beteiligung von insgesamt circa sechzig Studenten. Durchgeführt wurde eine Feldstudie

The screenshot shows a Mozilla Firefox browser window displaying the GRIS ILO Repository for Learning-Modules. The page content is as follows:

Home
News
Login
Search
Courses
Modules

Two examples

Example $f(m) = f(x_0 + \Delta x)$, $m = 0, \dots, N-1$
and simplifies the Fourier integral to

$$F(k) = \frac{1}{N} \sum_{m=0}^{N-1} f(m)(\cos(2k\pi) - i \sin(2k\pi)), \quad k = 0, \dots, N-1$$

$$f(m) = \sum_{k=0}^{N-1} F(k)(\cos(2k\pi) + i \sin(2k\pi)), \quad m = 0, \dots, N-1$$

flash

Hallo!!

Hello World Applet

flashCode

```
import test.TestClip;
class test.Test extends MovieClip
{
  5.     private var scopeRef:MovieClip;
  6.
  7.     function Test(scope:MovieClip) {
  8.
  9.         scopeRef = scope;
 10.
 11.         // --- Creates a 'tf' TextField size 100x600 at pos 100, 100
 12.         scopeRef.createTextField("tf", 0, 100, 100, 800, 600);
 13.         // --- Write some text into it
 14.         scopeRef.tf.text = "Hallo!!"; // --- Dynamic MovieClip linkage
 15.         var linkedClip:MovieClip = scopeRef.attachMovie(TestClip.SymbolName,
 16.         linkedClip._x = 200;
 17.         linkedClip._y = 200;
 18.         linkedClip._rotation = 30;
 19.         linkedClip._alpha = 70;
 20.
 21.         linkedClip.onPress = function() {
 22.         this.startDrag();
 23.         };
 24.
 25.         linkedClip.onRelease = function() {
 26.         this.stopDrag();
 27.         };
 28.
 29.
 30.
}
```

Compile-Results

Abbildung 5.8 Demonstration eines Modules in Flash: Oben Formeln in MathML, darunter das Flash-Objekt mit dem dazugehörigen Quelltext im Editor unten.

zur Fragestellung, inwiefern Kooperationskripte in Verbindung mit Peer-Reviews den individuellen wie auch den Gruppenlernerfolg verbessern [40]. Während der objektive Lernerfolg der Skriptgruppe im Vergleich zur Kontrollgruppe nicht signifikant besser war, wurden die analytischen Fähigkeiten der Skriptgruppe signifikant in objektiver wie subjektiver Hinsicht gesteigert. Weiterhin verwendeten die Studenten der Skriptgruppe signifikant mehr Zeit für die Bearbeitung und Lösung der gestellten Aufgaben - ein gewünschter Effekt. Untersucht wurde in dieser Arbeit auch, ob die stärkere Strukturierung des Lernens durch die Kooperationskripte einen motivationsmindernden Einfluss auf die Studenten hat, was im Rahmen der Studie als nicht zutreffend bewertet werden konnte. In subjektiver Hinsicht bewerteten die Studenten ihren Lernerfolg sogar signifikant höher als die Kontrollgruppe. Die Ergebnisse sind ausführlich nachzulesen bei [40].

Das System wurde auch im Rahmen der Vorlesung Bildverarbeitung I im Sommersemester 2006 am WSI/GRIS eingesetzt, abschließend wurden die teilnehmenden Studenten zu ihrer Meinung bezüglich des Online-Systems befragt. Teilgenommen haben zehn Studenten in fünf Zweier-Lerngruppen. Prinzipiell wurde die Idee der Online-Übungsabgabe von allen Studenten begrüßt. Kritik wurde an der Gesamt-Performanz des Systems, an der Kompatibilität unter verschiedenen Browsern und am Editor geübt. Im Einsatz mit mehreren Studenten, welche gleichzeitig am System über Browser arbeiteten, kam es tatsächlich zu Problemen, die erst im Laufe des Übungsbetriebs gelöst werden konnten. Zusätzlich lief das System noch im Probebetrieb und somit auf einem nicht mehr aktuellen Rechner (512 MB Speicher, Intel Pentium 4 mit 2 Gigahertz CPU-Takt). Insbesondere der begrenzte Speicher im Zusammenspiel mit nicht ausreichend groß konfigurierter Auslagerungspartition (Swap) führte anfangs aufgrund des Speicherverbrauchs zu Problemen, die mit der Vergrößerung der Auslagerungspartition aber weitgehend verschwanden. Insgesamt ist natürlich klar, dass bei starker Benutzung das System auf mehrere Rechner verteilt laufen muss. Die Grundlagen hierfür sind gegeben, da einerseits die Dienste wie die Versionierung auf externe Rechner ausgelagert werden können, andererseits Apache Lenya und Tomcat dafür vorgesehen sind, verteilt konfiguriert zu laufen. Außerdem bieten gerade Apache Lenya und Tomcat vielerlei Möglichkeiten zur Optimierung des Betriebs, die allerdings im beschriebenen Probebetrieb nicht ausgeschöpft wurden.

Bei verschiedenen Demonstrationen wurde das System von potenziellen Autoren wie Anwendern positiv aufgenommen. Die positive Einschätzung kam gerade auch aus anderen naturwissenschaftlichen Fachgebieten wie Biologie und Physik, welche im Studium oftmals auch Programmierkenntnisse erfordern. Zusätzlich zu den bereits beschriebenen Erkenntnissen kann man noch folgende Thesen zusammenfassen:

- Dem unbedarften Anwender erleichtert der Ansatz die Beschäftigung mit den Quelltexten von Algorithmen.
- Die fehlenden Debug-Möglichkeiten werden von unerfahrenen Anwendern eher als Nachteil empfunden als von versierten Programmierern.
- Ab einer gewissen Komplexität oder Länge des Quelltextes lohnt sich die Verwendung einer integrierten Entwicklungsumgebung.

Das System hat sich im Einsatz als stabil und verwendbar erwiesen. Die Erstellung von Lehrmaterialien ist deutlich erleichtert.

5.8 Zusammenfassung und Ausblick

Die grundlegende Idee ist, die Computergrafik-Gemeinschaft dazu zu bringen, Lehrmaterialien zu sammeln und zu teilen. Es hat sich in der Vergangenheit oft genug gezeigt, dass es für Autoren nicht reicht, Anerkennung nur in der Community zu bekommen.

Die vorgestellte Auszeichnungssprache reduziert die Aufgabe der Inhalteerstellung auf den Kerninhalt, die Publikation in verschiedenen Formaten wird vom System übernommen. Mittels der neu entworfenen Auszeichnungssprache werden interaktive Illustrationen ermöglicht, dies stellt einen Mehrwert da, der ohne ein zu Grunde liegendes mächtiges Repositorium nicht möglich wäre. Die interaktiven Illustrationen ersetzen die bis dahin üblichen Pseudocode-Quelltexte durch echte Quelltexte, welche mit den dazugehörigen interaktiven Elementen verknüpft werden. Theorie wird somit durch Implementierungen erläutert und der Anwender kann diese Implementierungen online ändern. Unter der Voraussetzung, dass die interaktiven Illustrationen mit den Quelltexten angemessen entworfen sind, werden somit Implementations-zentrierte Elemente eingeführt, welche die alte Zweiteilung zwischen Laborarbeit und Aufgaben aufhebt. Weitere bisher noch nicht verwendete didaktische Szenarien (siehe Abschnitt 5.1) können mit diesem Werkzeug verwirklicht werden.

Die Materialien werden im Repositorium verwaltet, kompiliert und versioniert, dies senkt die technischen Hürden für Lehrende wie Lerner und ermöglicht tatsächlich den Austausch der Materialien unter Gleichen. Der Quelltextzugang kann feingranular gesteuert werden, damit wird ein neues Lizenzierungsmodell ermöglicht, welches nur Teile der Quellen offen legt. Die Materialien im Repositorium unterliegen einer ständigen Weiterentwicklung, die wachsende Anzahl an alternativen Implementierungen können analysiert oder in anderen didaktischen Szenarien weiterverwendet werden, zum Beispiel in Multiple-Choice-Tests oder für die Quelltext-Fehlersuche. Im Falle einer Klassifizierung der Quelltextversionen, können auch adaptive Eigenschaften verwirklicht werden. Je nachdem, welcher Anwender gerade angemeldet ist, könnten diejenigen Quelltexte gewählt werden, welche am besten zu seinen Einstellungen passen. Dies bedeutet, dass Forschung über adaptive Hypermedien-Technologien die hier vorgestellten Quelltextimplementierungen ebenfalls abdecken könnte. Evaluationen haben gezeigt, dass die hier vorgestellten Werkzeuge von Studierenden als förderlich und sinnvoll bewertet wurden.

Workflows und ihr Management

6.1 Workflowmanagement

Wie in den vorangegangenen Kapiteln beschrieben, können mit den bisher vorgestellten Techniken virtuelle Lerngemeinschaften mit einem großen Mehrwert geschaffen werden. Neben den üblichen Diensten eines Repositoriums wie Benutzermanagement, Internationalisierung, Versionierung, Kategorisierung, einheitliches Aussehen, Suchmöglichkeiten und weiteren sowie den neuen im letzten Kapitel vorgestellten Dienste *code based interactivity* und *server side compiling* [55] müssen in einem flexiblen System, welches zum Arbeiten in Gruppen genutzt wird, verschiedene Abläufe ermöglicht werden. Ein bekanntes Beispiel hierfür ist das gegenseitige Begutachten von Entwürfen und Lösungen, ehe diese Arbeiten veröffentlicht werden. Die Integration eines solchen Metaablaufs, welcher mehrere Dokumente betrifft, welche in mehreren Zwischenschritten von verschiedenen Personen oder Personengruppen bearbeitet werden, ist in einem klassischen Content-Managementsystem komplex, da es die Bearbeitung sehr vieler unterschiedlicher Komponenten des Content-Managementsystem zur Abbildung dieses Ablaufs erfordert, oder gar unmöglich. Erst in neuerer Zeit gibt es Ansätze, Content-Managementsysteme mit einem Rahmenwerk zur Workflow-Interpretation auszustatten (z.B. wurde in dem Opensource-Content-Managementsystem Typo3 erst in der im Frühjahr 2006 erschienenen Version 4.0 ein Workflow-Mechanismus eingebaut), moderne Content-Managementsystem wie Apache Lenya basieren im Kern auf einer Workflow-Engine. Die Arbeiten an den neuen Diensten *code based interactivity* und *server side compiling* profitierten bereits von einer vorhandenen Workflow-Engine, was auch die Wahl des genutzten Content-Managementsystem mit begründete. Tatsächlich sind Content-Managementsysteme unterschiedlich gut auf solche Erweiterungen vorbereitet, ermöglichen dann aber Implementierungen auf

unterschiedliche Art und Weise [53] [59]. Im folgenden Kapitel wird erläutert, wie eine vorhandene funktionierende Workflow-Engine die Erweiterung eines Content-Managementsystems um neuartige wie im vorigen Kapitel vorgestellte Dienste erleichtern und beschleunigen kann und wie in Zukunft die Workflow-zentrierte Sicht auf Applikationen zunimmt, was sich bei Webservices schon jetzt abzeichnet.

6.2 Geschichte der Workflowsysteme

Generell ist zu sagen, dass aktuelle Dokumentenmanagementsysteme - ob webbasiert oder desktopbasiert, ob kommerziell oder frei erhältlich - keine oder nur extrem eingeschränkte Möglichkeiten zur Veränderung oder Erweiterung der bestehenden Abläufe bieten. Bessere Dokumentenmanagementsysteme bieten Bibliotheken mit Programmierschnittstellen an, allerdings meist auf einem sehr niedrigen Niveau. Workflows sind im akademischen Umfeld als Verallgemeinerungen von Petrinetzen bekannt [115]. Diese wurden 1962 von Carl Adam Petri an der Universität von Bonn entwickelt. Er schuf eine grafische Beschreibung von gleichzeitig ablaufenden Programmfäden in verteilten Systemen. Obwohl diese Arbeiten vor allem für die Automatentheorie entworfen wurden, können sie gut für Workflows verwendet werden. Mitte der siebziger Jahre zeigte Officetalk [41] und SCOOP von Wharton (nachzulesen bei [101]), beide bei Xerox Park entwickelt, wie sich die Forschung in Richtung automatisierte Geschäftsabläufe (*business processing*) verlagerte. Ab 1985 wurden dann kommerzielle Workflow-Systeme entwickelt und verkauft. Einen guten Überblick dazu liefert zur Mühlen in [104]. Akademische Forschung konzentrierte sich erst wieder in den neunziger Jahren auf Workflow-Prozesse. Erwähnenswert sind hier die Arbeiten von van der Aalst [146], der nachwies, wie nah Petrinetze und Workflows sind, indem er systematisch existierende Workflow-Sprachen analysierte und sie auf die entsprechenden Konstrukte in Petrinetzen abbildete [145].

Die Workflow Management Coalition (WfMC), welche im August 1993 gegründet wurde, ist eine Organisation, welche versucht, allgemein gültige Standards im Bereich des Workflow-Management durchzusetzen. Ihre Vorschläge mündeten in einer Workflow-Schnittstelle (*Workflow API - WAPI*) und einer Definitionssprache für Workflows (*Workflow Process Definition Language - WPD*L). Diese wurden beide 2002 weiterentwickelt und im gleichen Zug kompatibel zu XML umgestaltet, der hierbei entwickelte Standard nennt sich XPDL. Seither wurden von verschiedenen Organisationen und Firmen weitere, von einander unabhängige Sprachstandards geschaffen. Darunter finden sich die OMG Workflow Facility, eine Erweiterung des UML-Standards, die BPML (*Business Process Modelling Language*), eine Fassung der Business Process Management Initiative (BPMI), WS-CDL vom W3C, WSFL (*Web Service Flow Language*) von IBM, XLANG von Microsoft und BPEL4WS (*Business Process Execution Language For Workflow Services*) von der Oasis-Gruppe, welche den IBM- und den Microsoft-Ansatz unter Beteiligung anderer Firmen wie SAP zusammenführt. Diese neu entstandene Diversität resultiert daher, dass die verschiedenen Firmen ihre Workflow-Entwicklungen unabhängig voneinander oft auch als Konkurrenzprodukte starteten und die verschiedenen Ansätze alle verschiedene

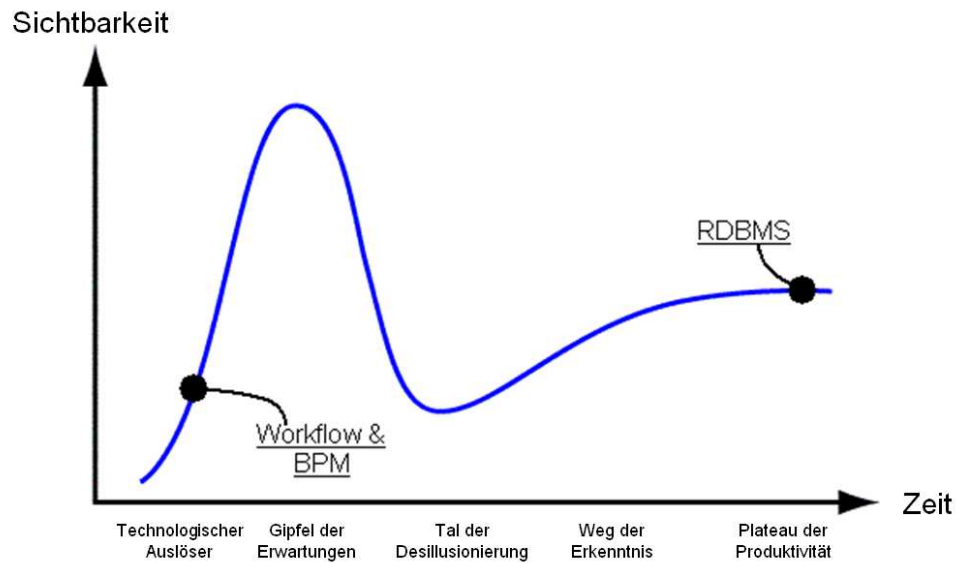


Abbildung 6.1 Der Hypecycle von Workflow-Management im Vergleich mit relationalen Datenbanken nach Baeyens [12] und Gartner Group.

Fähigkeiten besitzen. Workflowmanagementsysteme können mindestens eine der standardisierten Sprachen interpretieren und Anwendungen danach steuern. Diese Systeme können als Applikationsserver mit zusätzlichen Workflow-Diensten verwirklicht sein wie z.B. Beas WebLogic Server oder SAPs Netweaver, oder als Sammlung verschiedener Werkzeuge und Dienste, welche ein eher lose gekoppeltes Rahmenwerk darstellen, wofür anschließend noch einiges an Integrationsarbeit geleistet werden muss (wie bei jBPM von JBoss oder OpenEJBXML als OpenSource-Produkt). Weitere Systeme, die sich zwischen diesen zwei Extrema ansiedeln, sind Enhydra Shark (www.enhydra.org), eine Workflow-Umgebung, welche auf der XPDL-Sprache der WfMC beruht, oder auch Con:cern (concern.org), einem Rahmenwerk, welches auf Einzelfallunterscheidungen beruht. Im Allgemeinen bieten Workflow-Applikationsserver nur Programmierschnittstellen (APIs wie Java Management Interfaces, Enterprise Java Beans oder Webservices mit ihrer API) um externe Anwendungen zu steuern. Diese eher lose Kopplung führt dazu, dass während der Umsetzung von Workflows noch viel Implementierungsarbeiten und Anpassungen durchzuführen sind. Selbst Workflow-Rahmenwerke mit grafischen Benutzerschnittstellen für die Modellierung von Workflows wie EnhydraShark, das eben genannte Con:cern, Twister (www.smartcomps.org/twister/) oder auch Yawl von van der Aalst [144] ermöglichen mit dieser Schnittstelle nur die Steuerung des Frameworks selbst, erlauben jedoch nicht die einfache Anpassung und Integration externer Anwendungen.

Laut Baeyens ist die Entwicklung der Workflowmanagementsysteme erst am Beginn des Verwendungszyklus [12] (siehe Abbildung 6.1). Dieser so genannte *Hypecycle* lässt sich laut Gartner Group auf alle Technologien anwenden und beschreibt die Produktivität, mit welcher eine Technologie eingesetzt wird. Workflowmanagementsysteme stehen im Gegensatz zu den allseits bekannten und benutzten relationalen Datenbanken

erst ganz am Anfang ihrer Nutzung.

6.3 Technische Aspekte

Workflowmanagementsysteme sind Software, welche formale Beschreibungen eines Workflow interpretieren, die Zustände der Aktivitäten abspeichern und Aufgaben entweder den Anwendungen oder Anwendern zuzuweisen. Diese Beschreibungen sind oft in einem XML-Dialekt geschrieben und können grafisch erstellt und bearbeitet werden. Die grafische Bearbeitung eines Workflowmanagementsystems hat im Verhältnis zum textorientierten Ansatz Vorteile in Bezug auf die Entwicklungsrisiken, ermöglicht die zentrale Verwaltung von angepassten Workflows, verkürzt die Entwicklungszeit von Workflows und ermöglicht eine größere Flexibilität im Umgang mit ihnen [104]. Technisch gesehen, kann man Workflows in vier unterschiedliche technische Domänen aufteilen: in Zustände, in den Kontext, in Programmlogik und in Anwenderschnittstellen [12].

- Die Domäne der Zustände: Mittels der Zustände werden die Ablauffolgen eines Workflows und somit seine Ablauflogik bestimmt. Denkt man in Kategorien der Petrinetze, bestimmt ein Aktivitätstoken, welcher Zustand aktiv ist. Über Transitionen kann der Aktivitätstoken weiter wandern.
- Die Domäne des Kontexts: Über Variablen im Kontext interagiert ein Workflow mit externen Anwendungen und kann Daten abspeichern und einlesen.
- Die Domäne der Programmlogik: Nach außen werden verschiedene Aktionen angestoßen, welche je nach Anforderung sehr komplex werden können. Als Beispiel sei das Versenden einer Email genannt, welches natürlich über Programmlogik abgewickelt wird.
- Die Domäne der Benutzerschnittstelle: Im Laufe eines Workflows sind oft Benutzereingaben erforderlich, oder der Workflow wird gar von einem Benutzer angestoßen. Als Beispiel sei das Ausfüllen und Absenden eines Begutachtungsformulars genannt, womit ein Dokument akzeptiert oder verworfen wird.

Normalerweise werden Workflow-Beschreibungen als Zustandsautomat angesehen, welcher aus einer Menge von Zuständen, einer Menge von Transitionen, einer Menge von Ereignissen, einer Menge von Bedingungen, einer Menge von Aktionen, einer Menge von Variablen und einem initialen Zustand besteht. Eine Instanz eines Workflows wird aus diesem Zustandsautomat, einem laufenden Zustand und Instanzen der Variablen gebildet. Nachfolger werden bestimmt, indem der laufende Zustand verändert wird. Dazu werden die Transitionen untersucht, um unter Beachtung der gegebenen Bedingungen und Ereignisse einen Nachfolgezustand zu ermitteln. Während einer Transition können auch Variablen neu oder anders gesetzt werden. Um ein Content-Managementsystem grafisch anpassen zu können, werden eine Reihe von Werkzeugen benötigt, welche in den folgenden Abschnitten beschrieben werden.

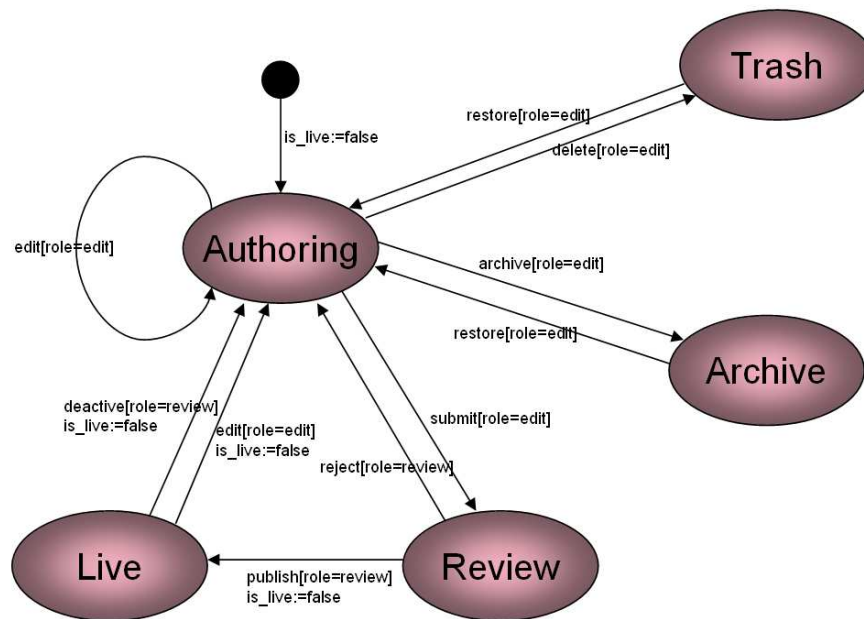


Abbildung 6.2 Der Standard-Lebenszyklus eines Dokuments in Apache Lenya.

6.3.1 Content-Managementsystem

Für das zu Grunde liegende Content-Managementsystem wurde Apache Lenya [8] gewählt, welches von vornherein wegen seiner eingebauten Workflow-Fähigkeiten überzeugt. Im System gibt es bereits einen Basis-Workflow, welcher den Lebenszyklus eines Dokuments abbildet (siehe Abbildung 6.2).

Ein Dokument wird neu angelegt und befindet sich damit im Zustand *Authoring*. Wenn der Benutzer das Dokument veröffentlichen will, legt er es einem Reviewer vor, das Dokument wandert also in den Zustand *Review*. Ein Benutzer mit der Rolle *reviewer* muss das Dokument akzeptieren, damit es der Allgemeinheit zugänglich wird, was sich durch eine Transition in den Zustand *Live* ausdrückt. Damit ein Dokument archiviert werden kann, muss es durch eine Deaktivierung oder ein erneutes Editieren wieder in den Zustand *Authoring* überführt werden. Von dort aus gelangt es mittels einer Archivieren-Aktion in den Zustand *Archive* oder mittels einer Löschen-Aktion in den Zustand *Trash*, von denen aus ein Dokument jeweils mit der Aktion Restaurieren (*restore*) wieder hergestellt werden kann.

Die Konfiguration eines solchen Workflows ist in Lenya in einer in einem XML-Dialekt verfassten Workflow-Konfigurationsdatei abgespeichert. Es ist mittels der Erstellung weiterer Konfigurationsdateien leicht möglich, andere Workflows zu entwerfen oder auch den bestehenden Workflow zu erweitern. In der Konfigurationsdatei werden, wie im folgenden Auszug zu sehen ist, zuerst die Zustände eingeführt. Dann werden die Transition mit Ausgangs- und Zielzustand sowie die Bedingungen und verwendeten Usecases definiert, letztere legen hier den Layer der Benutzerschnittstelle fest:

```
1 <state id="authoring" initial="true"/>
2 <state id="review"/>
3 <state id="live"/>
4 <state id="trash"/>
5 <state id="archive"/>
6 <variable name="is_live" value="false"/>
7 <transition source="authoring" destination="review">
8   <event id="submit"/>
9   <condition class="org.apache.lenya.cms.workflow.RoleCondition">edit</condition>
10   <usecase>
11     <name>submit</name>
12     <step>showscreen</step>
13   </usecase>
14 </transition >
```

Listing 6.1 Auszug aus der Konfigurationsdatei des Standardworkflows von Apache Lenya

Die Usecases als Benutzerschnittstellen müssen natürlich noch programmiert werden, da hier sämtliche Informationen über zu setzende Variablen fehlen.

6.3.2 Ein Workflow-Editor

Um derartige Konfigurationsdateien nicht mühsam per Hand erstellen und verwalten zu müssen, ist die Wahl eines geeigneten Editors zur Modellierung von Workflows notwendig. Hierfür wurden hauptsächlich frei erhältliche Editoren getestet, welche sich durch bereits weit verbreiteten Einsatz auszeichnen. In die Testauswahl kamen die Editoren der Projekte Twister, Con:cern, Enhydra JaWE, YAWL, OSWorkflow und OpenOffice Draw. Ziel ist, aus einem grafisch modellierten Workflow per XML-Export eine Datei zu gewinnen, welche per XSL-Transformation als eine korrekte Apache Lenya Konfiguration umgesetzt werden kann und welche als Grundlage genommen werden kann, um die zusätzlich benötigten Skripte und Code-Segmente zumindest als Skelett-Dateien daraus erzeugen zu können.

In der Tabelle 6.3.2 werden die Editoren nach Namen aufgeführt, die drei Spalten U (*Usability*), S (*Stability*), und C (*generated Code*) stehen für die Benutzbarkeit und den Komfort, die Ausgereiftheit des Programms sowie schließlich für den generierten XML-Code. Die Bewertungen erstrecken sich von Minus für schlecht, über Null wie neutral bis hin zu Plus für gute Bewertung.

Insgesamt ergaben sich große Unterschiede zwischen den Editoren. Die Bedienbarkeit in Hinblick auf die Aufgabe der Flussgraph-Modellierung war vor allem bei den spezialisierten Anwendungen, also Twister, Con:cern, Enhydra JaWE, YAWL und OSWorkflow, gegeben. Allerdings waren bei diesen oft die Menüstrukturen unübersichtlicher und im Gegensatz zu OpenOffice Draw neu zu erlernen. Die Stabilität war dann auch nur bei

Editor	U	S	C	Bemerkungen
Twister	0	0	-	Erfordert zusätzliche Werkzeuge
Con:cern	+	0	-	Generiert Java-Code
Enhydra JaWE	0/+	0	+	Wesentliche Eigenschaften fehlen (keine Labels möglich)
YAWL	0/+	+	+	Bester Kompromiss
OSWorkflow	-/+	-	+	Vergisst das modellierte Layout nach dem Abspeichern
OpenOffice Draw	+	+	-	Generiert binäres XML mit darin enthaltenen Layout-Informationen

Tabelle 6.1 Testergebnisse der grafischen Workflow-Editoren

OpenOffice hinreichend gut, sowie mit leichten Abstrichen bei YAWL, da dieser im Betrieb manchmal nicht mehr auf Benutzereingaben reagierte. Die anderen Editoren sind deutlich weniger stabil, manche beenden sich im laufenden Betrieb, ohne dass der Benutzer die Chance gehabt hätte, seine laufende Arbeit abzuspeichern. Twister generiert keine XML-Dateien ohne weitere Werkzeuge, welche extra zu installieren sind. OpenOffice Draw vermischt den generierten XML-Code mit binären Daten, welche das Layout beschreiben. Con:cern schließlich generiert Java-Code statt XML. Als bester Kompromiss der hier aufgeführten Editoren hat sich der YAWL-Editor erwiesen. Er generiert ein recht übersichtliches XML, welches ohne große weitere Hürden transformiert werden kann. Außerdem läuft er welcher recht stabil und bietet Möglichkeiten zur Angabe aller benötigten Bemerkungen und Annotationen. Hierbei sei insbesondere auf die Bedingungen verwiesen, welche im Workflow-Flussgraph modelliert werden sollen und bestimmen, ob Transitionen überhaupt ausgeführt werden können. Gerade für Annotationen zu Transitionen bietet beispielsweise der Enhydra-Editor keinerlei Möglichkeit, weswegen im generierten XML-Code die Zuordnung dieser Informationen zu den Transitionen nicht mehr gemacht werden kann, was diesen Editor von der Benutzung für die gewünschten Zwecke ausschließt.

6.3.3 Transformation

Der nächste Schritt in der Abfolge von einem grafischen Modell zu einem vollständig ins System integrierten Workflow ist die korrekte Übersetzung des mittels des grafischen Workflow-Editors generierten Codes in eine für Apache Lenya verständliche Konfigurationsdatei. Der Workflow-Editor YAWL generiert eine XML-Beschreibung des Modells, welche leider nicht mit der Workflow-Konfiguration in Lenya identisch ist. Da aber alle benötigten Informationen wie Namen der Zustände, Namen der Transitionen, Bedingungen und so weiter in der exportierten Datei vorhanden sind, kann mittels einer XSL-Transformation (siehe Abschnitt 2.3 und [32]) eine Lenya-kompatible Version der Konfiguration erzeugt werden. Der Hauptunterschied zwischen den beiden XML-Dialekten des YAWL-Editor und der Lenya-Konfigurationsdatei liegt darin, wie Zustände

und Transitionen beschrieben werden. Bei YAWL wird jeder Zustand als XML-Knoten beschrieben. Kindknoten dieses Zustands verweisen darauf, welche anderen Zustände unter welchen Bedingungen von diesem Zustand aus erreicht werden können. In der Lenya-Konfiguration werden Zustände ebenfalls als Knoten dargestellt. Die Transitionen werden jedoch unabhängig davon in eigenen Knoten beschrieben, welche durch Namen sowie Start- und Zielzustand definiert werden. Eine XSL-Transformation, welche eine YAWL-Konfiguration in einen Lenya-Workflow überführt, besteht aus zwei Hauptschritten. Zuerst müssen alle Zustände aus der YAWL-Konfiguration extrahiert werden, im zweiten Schritt werden in einer Schleife die Transitionen unter Berücksichtigung ihrer Namen umgesetzt. Damit dies korrekt durchgeführt werden kann, sind im YAWL-Editor die Labels und Bedingungen analog zu den Editieranweisungen, wie sie in [122] erläutert werden, durchzuführen.

6.3.4 Aktionen

Mit so genannten Aktionen von Apache Lenya ist die Domäne der Programmlogik gemeint (siehe Abschnitt 6.3). Aktionen werden ohne Nutzerinteraktion im Rahmen eines Workflow-Ablaufs gestartet und steuern externe Programme. Beispiele hierfür mögen das Kopieren von Dateien sein, das Versenden von Emails, das Einplanen von Terminen für weitere Aktionen oder auch Backup-Aufgaben und ähnliches. Der Workflow in Lenya löst Aktionen aus, indem spezielle Java-Klassen, welche eine vorher festgelegte Schnittstelle befolgen, aufgerufen werden. Diese Klassen erweitern eine abstrakte Elternklasse und müssen eine so genannte *execute*-Methode, in welcher Entwickler ihren speziellen gewünschten Code einbetten können, implementieren. Dabei wird der aktuelle Anwender beziehungsweise der Anwender, welcher zuletzt in den Workflow eingegriffen hat, sowie das betroffene Dokument als Parameter übergeben und stehen somit zur Verfügung. Die Parameter, welche im grafischen Workflow-Modell definiert wurden, stehen ebenfalls zur Verfügung. Ist die Programmierung einer neuen Java-Aktions-Klasse erforderlich, wird automatisch ein Rahmenskelett per Skript automatisch generiert, der Programmierer muss nur noch seine spezielle *execute*-Methode mit Inhalt ausfüllen. Bereits vorhandene Klassen können natürlich neu parametrisiert weiter verwendet werden. Hiermit steht dem Anwender neben der Fülle der Java-Umgebung auch die Möglichkeit zur Verfügung, Shell-Skripte und Ant-Skripte (Ant ist ein XML-basierter Nachfolger des bekannten Make-Programms, welches das Übersetzen von Programmpaketen automatisiert) anzusteuern und ausführen zu lassen.

6.3.5 Usecases

Useases dagegen entsprechen der Domäne der Benutzerschnittstelle (siehe Abschnitt 6.3). Usecases sind ein Lenya-Konstrukt, welches hiermit die Integration der Cocoon-Flow-Technik in Lenya beschreibt. Die Cocoon-Flow-Technik erlaubt es, Abfolgen von Formularseiten, welche Benutzereingaben ermöglichen, serverseitig zu steuern und somit

imperativ zu programmieren (siehe Abschnitt 2.6). Als Programmiersprache kann beispielsweise JavaScript [44] eingesetzt werden. Dabei muss im Rahmen von Workflows natürlich nicht nur das serverseitige JavaScript erstellt, sondern auch eine einfache Möglichkeit konzipiert werden, wie die entsprechenden Benutzerschnittstellen, d.h. Webseiten mit verschiedenen Formularfeldern, weitgehend automatisiert erstellt werden können. Hierfür schlägt de Mattia in seiner Arbeit [36] die Verwendung von generalisierten Schablonen vor, welche über Standardfelder wie Überschrift, Text, Knopf, Texteingabefeld, Checkbox, Link, Sendeknopf, Auswahl und Tabelle mittels einer Java-Bibliothek direkt verwendet werden können. Dies führt dazu, dass der Entwickler sich weitgehend auf die serverseitige Programmierung konzentrieren kann, da alle benötigten Layoutelemente schon zur Verfügung stehen. Trotz aller Vereinfachung durch die generellen Schablonen und die relativ leicht zu erlernende Sprache Javascript ist der hier durchzuführende Entwicklungsaufwand leider weitgehend per Hand zu erbringen und kann nicht aus dem grafischen Workflow-Modell generiert werden.

6.3.6 Integration

Die bis jetzt erzeugten Dateien werden in die Ablaufumgebung von Lenya übertragen, indem Skripte ausgeführt werden, welche die Dateien in die richtigen Verzeichnisse kopieren und dabei die Dateien auch so erweitern, dass sie den Anforderungen Lenyas entsprechen. Lenya lädt Workflow-Konfigurationen dynamisch neu nach. Werden nur kleine Änderungen vorgenommen, wie die Umbenennung von Zuständen oder das Hinzufügen einer neuen Transition, wozu keine Programmierarbeit notwendig ist, muss das System nicht neu gestartet werden. Neue Dokumentschablonen können ebenfalls hinzugefügt werden. Im Falle, dass ein neuer Workflow angelegt werden soll, muss dies sogar erfolgen, da ein Workflow eng an eine bestimmte, ihm zugewiesene Dokumentschablone gekoppelt ist. Dokumentschablonen steuern weiterhin das interne Pipeline-Verhalten von Lenya, welches die Webseiten zusammenstellt und ausliefert. Für neue Workflows muss man somit eine neue Dokumentschablone erstellen, an welcher der Workflow hängt. Dadurch wird andererseits eine neue Kategorie geschaffen, welche die Suchfunktionalität erweitert und verbessert.

6.4 Eine Bibliothek für Workflowmanagement

Damit die Arbeit eines Entwicklers erleichtert wird, ist es von Vorteil, eine Bibliothek zu entwickeln, die wiederverwendet werden kann, weil sie nötige und oft verwendete Elemente vorhält. Das Ziel ist dabei, dass der Entwickler keinen neuen Programmcode schreiben muss, sondern sich vollständig auf vorgefertigte Bibliothekenfunktionalität verlassen kann. Damit wird außerdem erreicht, dass Entwicklern neben dem vollen Potenzial der Bibliotheken auch eine Möglichkeit an die Hand gegeben wird, über Schnittstellen Programme in einer Programmiersprache ihrer Wahl zu verfassen. Dies ist natürlich von

großem Vorteil im Verhältnis zu einer Einschränkung der Entwickler auf eine bestimmte Programmiersprache.

6.4.1 Gebräuchliche Elemente einer Bibliothek

Für die Erstellung der oben angesprochenen Bibliothek ist es nützlich, die gebräuchlichsten Aufgaben, welche im Rahmen eines Online-Repositoriums und der Workflow-Modellierung anfallen, zu identifizieren:

- Benutzerbenachrichtigung
- Kopieren, Verschieben und Löschen von Dateien
- Verschieben von Dateien zwischen den verschiedenen Bereichen der Materialsammlung wie zum Beispiel aus dem Bereich *authoring* in den Bereich *published*
- Aufgaben planen
- externe Programme starten
- Skripte starten
- andere entfernte Dienste starten wie Webservices

Diese hier vorgestellten Elemente sind Aufgaben, welche recht unterschiedlich anspruchsvolle Fähigkeiten beinhalten. Die Möglichkeit, externe Skripte anzustoßen umfasst im Endeffekt alle anderen Fähigkeiten, da mittels eines Skriptes Emails versendet werden können. Ein Skript kann Dateien kopieren, verschieben und löschen. Mittels eines Skripts können externe Programme gestartet werden, es können weitere Skripte aufgerufen werden und mittels kleiner Helferprogramme könnte man auch entfernte Dienste starten, wenn diese nicht schon direkt skriptfähig (*scriptable*) sind. Allein die Planung von Aufgaben ist für Skripte nur machbar, wenn eine geeignete Schnittstelle dafür vorgesehen ist, was im Falle von Lenya jedoch gegeben ist. Nichtsdestotrotz ist es von Vorteil, wenn man diese verschiedenen Aufgaben auch ohne Skripte mittels vorgefertigter Programme oder Programmeinheiten direkt in Lenya durchführen kann, da dies einerseits die Aufgabenstellung erleichtert, die Ergebnisse zur weiteren Verarbeitung unmittelbar vorliegen und eine weitere Fehlerquelle, welche durch die Externalität gegeben ist, minimiert. Zudem ist man, wenn man mittels Lenya-eigener Werkzeuge oder einer dafür entworfenen Bibliothek zum Beispiel Emails versendet davor gefeit, Seiteneffekte wie die Synchronisierung und die Nachverfolgung solcher Ausführungselemente nicht genügend beachtet zu haben.

6.5 Usecase: Peer Reviewing

Als ein Beispiel für eine Workflowmodellierung soll das Szenario des Peer-Reviewing-Prozess dienen. Dieser Prozess wurde schon im Kapitel 5 erwähnt und ist einer der bekanntesten Abläufe zur Evaluierung und Qualitätssicherung im wissenschaftlichen Bereich. Seit 1950 ist er etabliert und wird allgemein eingesetzt, insbesondere im Bereich der Auswahl von Artikeln für Fachkonferenzen. Gegen diese Art von Evaluierungsprozess richtet sich auch Kritik wie unter [156] nachzulesen.

Für den hier aufgezeigten Fall soll das Peer-Review-System als Prozess für Studenten in Übungsgruppen, welche sich gegenseitig bewerten, umgesetzt werden. Studenten besuchen vorlesungsbegleitende Tutorien im Rahmen des normalen Lehrbetriebs an einer Universität. Diese Tutorien sind oft in Lerngruppen organisiert, was bedeutet, dass sich zwei oder drei Studenten zusammenfinden, um die oftmals umfangreichen Übungsaufgaben zu bearbeiten. Während im herkömmlichen Betrieb nach dem Bearbeiten einer Aufgabe die Abgabe beim Tutor erfolgt, welcher die Ergebnisse begutachtet, soll in einem Peer-Reviewing-Prozess jede Gruppe ihre Lösungsvorschläge anonymisiert in das Repository einstellen. Dann begutachten andere Gruppen diese Vorschläge und bewerten diese über Formulare, welche sie ausfüllen. Diese Bewertungen werden wiederum dem Tutor zur Verfügung gestellt, welcher dann abschließend darüber entscheidet, ob die Lösung so veröffentlicht wird oder ob sie noch einmal überarbeitet werden muss. Dieser Peer-Review-Workflow ist hier als beispielhaft zu sehen, er kann später noch erweitert und vertieft werden. Um diesen im Rahmen des gegebenen Systems zu implementieren, sind einige Schwierigkeiten, welche auftreten können, zu beachten:

- Workflowinstanzen sind wie oben beschrieben an Dokumente gekoppelt. Dies bedeutet, dass auch auszufüllende Formulare als Dokumente behandelt werden müssen. Bei der Erstellung müssen sie also erstellt werden und Anwendern zugeordnet werden.
- Die Synchronisierung des Ablaufs muss zugesichert werden.
- Für die Anonymisierung der Formulardaten müssen besondere Mechanismen eingeführt werden, damit aus der Zuordnung von Dokumenten zu Anwendern keine Rückschlüsse gezogen werden können.

Der Workflow wird in Abbildung 6.3 gezeigt. Viele wichtige Angaben, welche für die korrekte Generierung der Konfigurationsdateien und Skript-Dateien in Lenya notwendig sind, wie die Rollenbedingungen, die Variablenzuweisungen usw. sind im Modell nicht zu sehen, da sie in Kontextmenüs der Transitionen modelliert werden.

Der Prozess wird damit gestartet, dass ein Dokument im System angelegt wird, indem es initial eingereicht wird. Der Anwender legt es damit dem Tutor (in Abbildung 6.3 *Editor in Chief - EiC* genannt) vor, welcher neu erzeugte Auswertungsbögen drei unabhängigen Reviewergruppen zuordnet und somit die Aktivität an diese Gruppen weiterreicht. Das System erleichtert dem Tutor diese Arbeit, indem es nach dem Rollenkriterium

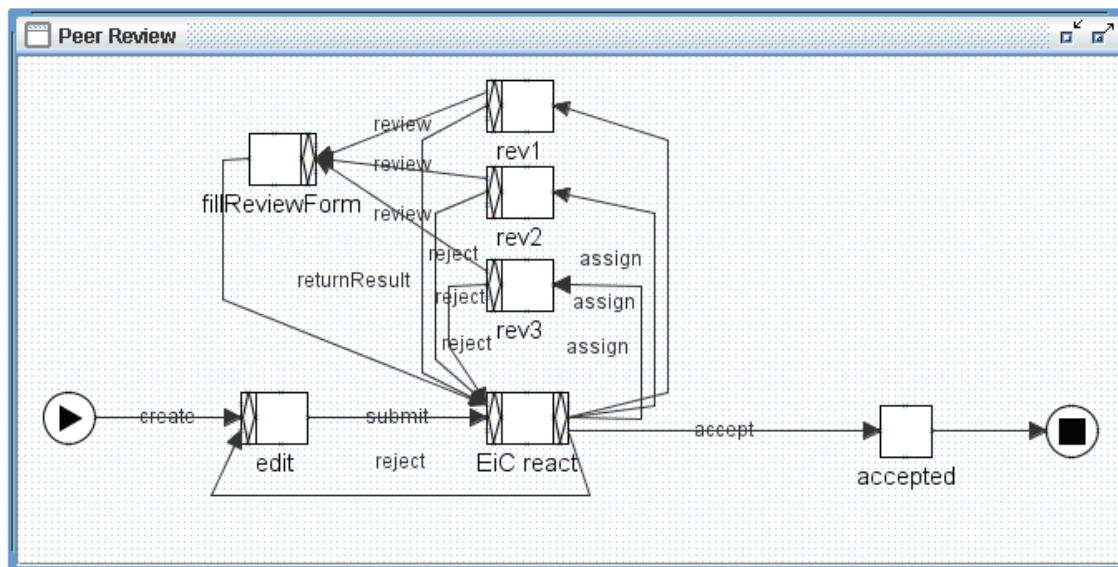


Abbildung 6.3 Der Peer-Review-Workflow, wie er in YAWL modelliert wird.

mögliche Gruppen vorselektiert. Akzeptiert eine Evaluierungsgruppe die Aufgabe, füllt sie den Bewertungsbogen aus. Will oder kann eine Gruppe die Evaluierungsarbeit nicht leisten, kann sie diese Aufgabe ausschlagen, das System schlägt dem Tutor dann eine neue zufällig ausgewählte Gruppe vor, welche die Aufgabe übernehmen soll. In einer zukünftigen Version könnten hier Verbesserungen greifen, wie beispielsweise die Auswertung des Verhältnisses durchgeführter Evaluierungen zu ausgeschlagenen Evaluierungen, was dann gewisse Werte nicht unterschreiten dürfte. Dadurch würden im Schnitt alle Gruppen gleichviel Bewertungsarbeit zu leisten haben. Wenn von einer Gruppe das Bewertungsformular ausgefüllt ist, wird diese dem Tutor über die *Submit*-Funktionalität vorgelegt. Bei seinem nächsten Login präsentiert das System die Ergebnisse dem Tutor, daraus kann er dann sein Fazit bezüglich Veröffentlichung oder Verwerfen des Lösungsvorschlags ziehen.

6.6 Zusammenfassung und Ausblick

Der hier aufgezeigte Ansatz zeigt, wie die Anpassung oder Neumodellierung von Prozessen in virtuellen Gemeinschaften, welche auf einem Content-Managementsystem wie Apache Lenya beruhen, erleichtert und beschleunigt werden. Dabei wird in diesem Ansatz die Anpassungsfähigkeit des Systems, welche als zu komplex und unzureichend empfunden wurde, als die in Kapitel 5 vorgestellten Dienste implementiert wurden, stark verbessert. Es wurde erfolgreich die Möglichkeit geschaffen, neue Workflows im System zu erstellen, welches anhand des Beispiels eines Peer-Reviews gezeigt wurde. Wie auch zu sehen ist, ist der Prozess der Definition, Modellierung und Implementation im System immer noch nicht voll automatisiert, da die Skripte zur Übertragung der Konfigu-

rationsdateien noch per Hand angestoßen werden müssen. Dies kann in zukünftigen Versionen noch weiter verbessert werden. Der hier vorgestellte Ansatz ist so allgemein, dass auch andere virtuelle webbasierte Gemeinschaften, welche nicht der Lehre dienen und nur mangelnde grafische Unterstützung bei der Erstellung neuer Arbeitsabläufe haben, davon profitieren können. Der Ansatz basiert auf so genannten Usecases und Aktionen, welche zusätzliche Programmierung nur erfordern, wenn die Bibliothek mit allgemeinen Elementen, welche im Rahmen dieses Projekts identifiziert wurden, nicht ausreicht. Selbst in diesem Fall wird ein Entwickler mit Programmschablonen, welche generiert werden, unterstützt. Damit wird die Bedienbarkeit, die möglichen Einsatzszenarien und die Wartbarkeit eines webbasierten Lehr-/Lernsystems stark verbessert.

Zusammenfassung und Ausblick

Diese Dissertation stellt Grundlagen in technischer wie auch didaktischer Hinsicht zur Entwicklung und Weiterentwicklung von webbasierter Lernsoftware vor. Angesichts des weiterhin rasanten technischen Fortschritts von Software und Softwaretechniken im Bereich von Internetanwendungen wurde verdeutlicht, wie wichtig es ist, für Software und insbesondere für Lernplattformen eine geeignete technische Basis zu benutzen, um die heute benötigte Flexibilität im Umgang mit den teuren Inhalten zu erreichen. Auch neue Entwicklungen des Internet, welche unter dem Schlagwort Web 2.0 vermehrt kollaborative Aspekte und größere Interaktivität bei Webanwendungen betonen, erzwingen geradezu, dass sich Lernplattformen diesen Entwicklungen, welche sie teilweise schon vorweggenommen haben, anschließen.

Bei der Transformation der Lernmaterialien spielt die Verwendung von XML heutzutage eine große Rolle. Die durch XML erreichte größere Flexibilität wird in dieser Dissertation anhand der Verwendung eines XML/XSL-basierten Frameworks demonstriert. Inhalte können sehr viel einfacher als früher dem Anwender in verschiedenen Formaten präsentiert werden, die Verwendung von mathematischen Formeln ist in Form von Schrift anstelle von Bildern möglich und grafische Auswertungen können leichter und dynamisch ohne Verwendung von externen Programmen generiert werden. Auch die Weiterverwendung von Lernobjekten in solch einer neuen Umgebung stellt kein Problem dar. Die Courseware ermöglicht die schnelle Entwicklung eines Forums, welches nach dem Drag-And-Drop-Paradigma [62] den Austausch zwischen Tutoren und Studenten weiter erleichtert.

Nachdem die Bedeutung einer modernen technischen Grundlage für Lernplattformen erläutert wurde, werden zwei neuartige Dienste für Lernplattformen eingeführt: quelltextbasierte Interaktivität zusammen mit serverseitiger Kompilierung und Versionierung. Diese beiden Techniken ermöglichen einen Lernfortschritt, da nun neben der Quelltextpro-

grammierung auch andere wesentliche Aufgaben wie Illustration, Exploration, Mehrfachauswahl, Quelltextvergleich, Auswahl im Quelltext und Quelltextbesprechung ermöglicht werden. Ein weiteres Ziel bei der Entwicklung dieser Dienste ist, die Computergrafik-Gemeinschaft dazu zu bringen, Lehrmaterialien selbstständig zu sammeln und zu teilen. Es hat sich erwiesen, dass es für Autoren nicht reicht, Anerkennung nur in der Community zu bekommen. Mittels der hier vorgestellten Dienste ergibt sich ein deutlicher Mehrwert auch für Autoren, zumal sie Teile ihrer Materialien nicht öffentlich einstellen können. Hiervon wird eine größere Attraktivität eines solchen Repositoriums erwartet.

Zur Implementierung der Dienste im Rahmen einer Lernplattform wurde eine XML-basierte Auszeichnungssprache definiert. Diese reduziert die Aufgabe der Inhalteerstellung auf den Kerninhalt, die Publikation in verschiedenen Formaten wird vom System übernommen. Die automatische Eingliederung neuer Inhalte wäre ohne ein zu Grunde liegendes mächtiges Repository nicht möglich. Die interaktiven Illustrationen ersetzen die bis dahin üblichen Pseudocode-Quelltexte durch echte Quelltexte verknüpft mit dem dazugehörigen interaktiven Elementen. Theorie wird somit durch Implementierungen erläutert und der Anwender kann diese Implementierungen online ändern. Dem Anwender bietet sich eine neuartige Sicht der verfügbaren Experimente und Interaktionen, Theorie wird anschaulicher als zuvor dargestellt.

Durch das eingängige XML-Schema, welches die Auszeichnungssprache definiert, wird Autoren die Erstellung der Inhalte erleichtert, wenn sie das Schema in ihren Editor einbinden. XML-fähige Editoren bieten nach dem Einbinden zulässige Auszeichnungselemente über komfortable Kontextmenüs an, so dass sich der Autor diese Auszeichnungselemente nur ungefähr aneignen muss beziehungsweise diese Elemente durch interaktives Blättern in den Menüs leicht erlernen kann. Für den Anwender ergibt sich die bequeme Situation, dass er seine Version immer bei sich hat, da die Versionierung auf dem Server vorgenommen wird und nach dem Einloggen dem jeweiligen Nutzer zur Verfügung steht. Dies erleichtert auch die Abgabe von Übungen, wie in Evaluationen gezeigt werden konnte. Erweiterungen dieses Systems lassen sich im Rahmen der Adaptierbarkeit vorstellen, welche, da der Benutzer sich in aller Regel am System anmeldet, ebenfalls anwenderspezifisch implementieren lassen. Der bisherige Erfolg des Systems, der sich in Evaluationen gezeigt hat, verdeutlicht, wie wichtig und zukunftsweisend diese neuartigen Dienste *serverseitige Versionierung*, *serverseitige Kompilierung* und *quelltextbasierte Interaktivität* sind.

Schließlich geht das letzte Kapitel auf die Problematik der Workflows in Lernplattformen ein. Insbesondere die Benutzung eines Content-Managementsystem zum Aufbau einer Lernplattform führt zur Problematik, wie die erweiterten und sich ändernden Arbeitsabläufe in Lernplattformen verwaltet und entwickelt werden können. Gibt es in einem klassischen Content-Managementsystem oft nur das Dokument als kleinste Einheit, welches entweder editiert, veröffentlicht oder gelöscht wird, so entstehen mit den Lernmaterialien sofort verschiedene Arbeitsabläufe bezüglich der Begutachtung und der Durchführung von Tutorien, welche oft umständlich modelliert und implementiert werden müssen. Dazu wurde ein semi-automatischer Entwicklungsprozess entworfen, welcher es ermöglicht, neue Workflows grafisch zu entwickeln und dann auf der Lernplattform zu

benutzen. Hierfür wurden auch Elemente einer Bibliothek von Hilfsmitteln und Diensten, welche die Umsetzung neuer Workflows erleichtern, definiert. Damit werden die Bedienbarkeit, die möglichen Einsatzszenarien und die Wartbarkeit eines webbasierten Lehr-/Lernsystems stark verbessert.

Zukünftige Entwicklungen bringen zunächst die Integration der neuen Dienste in weitere Repositorien. Wichtig werden Verbesserungen in der webbasierten Schnittstelle sein, hier könnten die in Entwicklungsumgebungen gebräuchlichen Features "Code-Ergänzung" sowie eine bessere Fehleranzeige im Online-Modus genannt werden. Ein weiterer Punkt sind die Adaptierbarkeit der Lernumgebung, welche ebenfalls Erleichterungen im Umgang mit den interaktiven Illustrationen verspricht. Die Verwendbarkeit auch mit anderen Programmiersprachen wurde anhand von Macromedias Flash verifiziert, jedoch werden die bekannten und verbreiteten Sprachen C und C++ mangels eines geeigneten Browser-plugins derzeit nicht unterstützt.

Lebenslanges Lernen wird in Wissensgesellschaften, wie sie westliche Industrienationen darstellen, zunehmend wichtiger. Jegliche Verbesserung in diesem Bereich, zumal wenn sie über das Internet abrufbar persistent überall verfügbar ist, wird hierzu beitragen. Lernmaterialrepositorien werden zunehmend wichtiger im Sinne, dass hier spezielle Themen interaktiv illustriert aufbereitet für jedermann begreifbar vorgehalten werden. Damit stellen sie eine mögliche technische Grundlage, um das immer schneller zunehmende Wissen aus der weltweiten Forschung breiten Teilen der Bevölkerung unkompliziert, einfach und didaktisch sinnvoll aufbereitet zugänglich zu machen. Ein großes Problem ist dabei die sehr teure Inhalteerstellung. Hierfür kollaborative Ansätze nach allgemein akzeptierten didaktischen Regeln zu erschaffen, bleibt eine große Herausforderung für Wissenschaft und Forschung.

Literaturverzeichnis

- [1] Advanced Distributed Learning. *Scorm 2004: 3rd Edition Content Aggregation Model*. <http://www.adlnet.gov/scorm/20043ED/Documentation.cfm> 2004. 3.7.2
- [2] Advanced Distributed Learning. *Scorm 2004: 3rd Edition Run-time Environment*. <http://www.adlnet.gov/scorm/20043ED/Documentation.cfm> 2004. 3.7.2
- [3] Apache Cocoon Project. *Understanding Apache Cocoon*. <http://cocoon.apache.org/2.1/userdocs/concepts/index.html> 2006. 4.2
- [4] Apache Software Foundation. *Xalan*. <http://xml.apache.org/xalan-j/index.html>. 2.3
- [5] Apache Software Foundation. *Xindice*. <http://xml.apache.org/xindice/>. 2.7
- [6] Apache Software Foundation. *Cocoon Continuation*. <http://cocoon.apache.org/2.1/userdocs/flow/index.html> 2006. 2.6
- [7] Apache Software Foundation. *XML Formatting Objects Project*. <http://xmlgraphics.apache.org/fop/> 2006. 2.3, 4.5
- [8] Apache Software Group. *Apache Lenya*. <http://lenya.apache.org/>. 2.4, 5.5.2, 6.3.1
- [9] D. Arnow und O. Barshay. *WebToTeach: An Interactive Focused Programming Exercise System*. In *Proceedings of the twenty-ninth ASEE/IEEE Frontiers in Education* Seite 12a9/39–12a9/44. IEEE Press 1999. 5.2
- [10] S. A. Assunção, F. C. Figueiredo und J. A. Jorge. *Proposal for a CG Education Content Online Submission and Reviewing System*. In *Eurographics/SIGGRAPH Workshop on Computer Graphics Education (CGE02)* 2002. 5.2
- [11] D. Bacon, N. Barr, L. Blackstone, J. Ferrante, P. Gorissen, R. Hoag, G. Krishan, S. Lay und R. Young. *IMS Question & Test Interoperability Specification*. <http://imsproject.org/question/> 2006. 1.2

- [12] T. Baeyens. *The State of Workflow*. TheServerSide.com. Mai 2004. 6.1, 6.2, 6.3
- [13] R. G. Baraniuk, C. S. Burrus, D. H. Johnson und D. L. Jones. *Connexions – Sharing Knowledge and Building Communities in Signal Processing*. IEEE Signal Processing Magazine. Mai 2004. 5.2
- [14] P. Barker. *What is IEEE Learning Object Metadata / IMS Learning Resource Metadata?* <http://metadata.cetis.ac.uk/guides/WhatIsLOM.pdf> 2005. 3.7.1
- [15] P. Baumgartner, H. Häfele und K. Maier-Häfele. *Evaluierung von Lernmanagement-Systemen: Theorie - Durchführung - Ergebnisse*. In A. Hohenstein und K. Wilbers, Editoren, *Handbuch E-Learning*. Fachverlag Deutscher Wirtschaftsdienst. Köln. 2002. 3.6
- [16] P. Baumgartner, H. Häfele und K. Maier-Häfele. *Evaluationsverfahren für den Vergleich virtueller Lernplattformen*. In H. Apel und S. Kraft, Editoren, *Online lehren. Planung und Gestaltung netzbasierter Weiterbildung* Seite 219–236. Bertelsmann. Bielefeld. 2003. 3.6
- [17] P. Baumgartner, H. Häfele und K. Maier-Häfele. *Lernplattformen im Feldtest*. In D. M. Meister, S.-O. Tergan und P. Zentel, Editoren, *Evaluation von E-Learning* Band 25 Seite 108–122. Waxmann. 2004. 3.6
- [18] P. Baumgartner und S. Payr. *Lernen mit Software*. Österreichischer StudienVerlag. 1994. 3.2
- [19] J. E. Beall, A. M. Doppelt und J. F. Hughes. *Developing an Interactive Illustration: Using Java and the Web to Make It Worthwhile*. Proceedings of 3D and Multimedia on the Internet, WWW and Networks. 1996. 5.2
- [20] T. Berners-Lee, J. Hendler und O. Lassila. *The semantic web*. Scientific American 284 (5), Seite 34–43. 2001. 2.4
- [21] Bitmanagement Software GmbH. *BS Contact VRML / X3D*. http://www.bitmanagement.com/products/bs_contact_vrml.de.html 2006. 5.2
- [22] A. Blumstengel. *Entwicklung hypermedialer Lernsysteme*. Dissertation Universität Paderborn 1998. 3.2
- [23] R. Bourret. *XML and Databases*. <http://www.rpbouret.com/xml/XMLAndDatabases.htm> 2005. 2.7
- [24] H. Braun. *Webstandards im Wandel*. c't - Computer und Technik 1, Seite 162–169. Januar 2007. 2.10
- [25] B. Brecht. *Der Rundfunk als Kommunikationsapparat*. In *Bertolt Brecht: Gesammelte Werke in 20 Bänden* Band 18 Seite 127–134. Frankfurt a. M. 3.3

- [26] J. Brown, Editor. *Visual Learning for Science and Engineering. Report of Eurographics/SIGGRAPH Visual Learning Campfire*. 2002. 5.2
- [27] D. Brownell. *SAX2*. O'Reilly & Associates, Inc. Sebastopol, California. 2002. 2.3
- [28] V. Bush. *As We May Think*. The Atlantic Monthly Seite 101 ff. Juli 1945. 3.3
- [29] R. Carey, G. Bell und C. Marrin. *ISO/IEC 14772-1:1997 Virtual Reality Modeling Language (VRML97)*. <http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/> 1997. 2.4.2
- [30] *Common Gateway Interface (CGI)*. <http://hoohoo.ncsa.uiuc.edu/cgi/overview.html>. 2.6
- [31] J. A. Chambers und J. W. Sprecher. *Computer assisted instruction: current trends and critical issues*. Communications of the ACM 23 (6), Seite 332–342. 1980. 3.4
- [32] J. Clark (Eds). *XML Transformations (XSLT) Version 1.0*. W3C Recommendation November 1999. <http://www.w3.org/TR/xslt/>. 2.3, 6.3.3
- [33] B. Collins-Sussman, B. W. Fitzpatrick und C. M. Pilato. *Version Control with Subversion for Subversion 1.1*. O'Reilly & Associates, Inc. Sebastopol, California. 2004. 2.7, 5.5.4
- [34] S. Cunningham. *GVE '99: Report of the 1999 eurographics/SIGGRAPH workshop on graphics and visualization education*. ACM SIGGRAPH Computer Graphics 33 (4), Seite 96–102. 2000. 5.2, 5.5.1
- [35] O. Danvy. *On Evaluation Contexts, Continuations, and the Rest of Computation* Januar 2004. 2.6
- [36] C. de Mattia. *Konzeption und Integration eines webbasierten Repository für interaktive Lernobjekte in ein Content-Management-System*. Diplomarbeit WSI/GRIS, Universität Tübingen 2006. 2.6, 5.4, 6.3.5
- [37] S. J. DeRose, E. Maler und D. Orchard, Editoren. *XML Linking Language (XLink) Version 1.0*. W3C Proposed Recommendation Dezember 2000. <http://www.w3.org/TR/xlink/>. 2.1
- [38] *DOM for Java*. <http://www.dom4j.org> 2006. 2.3
- [39] EduTools.info. *Comparison of selected LMS*. http://www.edutools.info/item_list.jsp?pj=8 2006. 3.6, 3.7.2
- [40] T. Egerter. *Kooperatives Lernen mit Hilfe von Kooperationskripten*. Diplomarbeit Universität Tübingen 2005. 5.7

- [41] C. A. Ellis und M. Bernal. *OfficeTalk-D: An experimental office information system*. In *Proceedings of the SIGOA conference on Office information systems* Seite 131–140 New York, USA. ACM Press. 1982. 6.2
- [42] D. C. Engelbart. *The Augmented Knowledge Workshop*. In A. Goldberg, Editor, *History of Personal Workstations* Seite 187–236. ACM Press. New York. August 1988. 3.3
- [43] F. C. Figueiredo, D. E. Eber und J. A. Jorge. *Computer graphics educational materials source policies and status report*. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Educators program* Seite 11 New York, USA. ACM Press. 2004. 5.1, 5.2, 5.5
- [44] D. Flanagan. *JavaScript: The Definitive Guide*. O'Reilly Media Sebastopol, California. 4. Edition 2001. 6.3.5
- [45] J. D. Foley, A. van Dam, F. Feiner und J. F. Hughes. *Computer Graphics, Principles and Practice, Second Edition*. Addison-Wesley Reading, Massachusetts. 1996. 5.6
- [46] B. Freitag. *LMML - Eine XML - Sprachfamilie für eLearning Content*. In S. E. Schubert, B. Reusch und N. Jesse, Editoren, *GI Jahrestagung* Band 19 von *LNI* Seite 349–353. GI 2002. 3.8
- [47] A. Fritze. *SVG Tetris for SVG-enabled Mozilla*. <http://www.croczilla.com/svg/samples/svgtetris/svgtetris.svg> 2004. 2.4.1
- [48] J. Gillies. *How the Web Was Born*. Oxford Paperbacks. 2000. 2.5
- [49] A. S. Glassner, Editor. *Graphics Gems I*. Academic Press. 1990. 5.2
- [50] A. S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann San Francisco, CA. 1995. 5.6
- [51] C. Goldfarb. *A Brief History of the Development of SGML*. SGML Users' Group 11. Juni 1990. 2.1
- [52] Google.com. *Google Video*. <http://video.google.com> 2006. 2.9
- [53] J. Görke. *Graphical Interactive Modeling Of Workflows In Web Based Repositories*. IADIS Februar 2006. 6.1
- [54] J. Görke, R. Beemagani, M. Hoffmann und W. Straßer. *Interactive Courseware for Teaching Field Theory*. International Conference on Engineering Education. 2003. 1.3, 4.3
- [55] J. Görke, F. Hanisch und W. Straßer. *Live graphics gems as a way to raise repositories for computer graphics education*. SIGGRAPH 2005 Education Paper. 2005. 1.3, 6.1

- [56] J. Görke, F. Hanisch und W. Strasser. *Introducing live graphics gems to educational material*. Computers & Graphics 30 (6). 2006. 1.3
- [57] J. Görke und M. H. W. Hoffmann. *Drag And Drop Supporting Interactive Courseware*. In *Proceedings of the 15th Annual Conference on Innovation in Education for Educational and Information Engineering* Mai 2004. 1.1, 4.4
- [58] J. Görke, M. H. W. Hoffmann und W. Straßer. *Dynamically Generated User Charts In Interactive Courseware*. In *Proceedings of the 16th Annual Conference on Innovation in Education for Educational and Information Engineering*. EAEEIE Juni 2005. 4.5
- [59] J. Görke und W. Straßer. *Setting up online repositories - a graphical workflow centric approach*. In *Proceedings of 17th EAEEIE annual conference*. EAEEIE Mai 2006. 1.3, 6.1
- [60] P. Grosso, R. D. Jr., E. Maler, S. DeRose, N. Walsh und J. Marsh. *XML Pointer Language (XPointer)*. W3C working draft W3C August 2002. <http://www.w3.org/TR/2002/WD-xptr-20020816/>. 2.1
- [61] F. G. Halasz. *NoteCards: An Experimental Environment for Authoring and Idea Processing*. In *BTW* Seite 56–67 1987. 3.3
- [62] F. Hanisch und W. Straßer. *Drag & Drop Scripting: How To Do Hypermedia Right*. Eurographics 2003 Education Presentations (EG'03). September 2003. 1.1, 4.4, 5.2, 7
- [63] F. Hanisch. *Hypermediales Lernen und Lehren von Computergraphik*. Dissertation Universität Tübingen 1998. 4.3
- [64] F. Hanisch. *Using Scripting to Increase the Impact of Highly Interactive Learning Objects*. In *ED-MEDIA 2002: the 14th World Conference On Educational Multimedia, Hypermedia & Telecommunications*. Association for the Advancement of Computing in Education (AACE), USA 2002. 4.4
- [65] F. Hanisch. *Highly Interactive Web-Based Courseware*. Dissertation Wilhelm-Schickard-Institut, Universität Tübingen Februar 2004. 3.6, 3.9, 4.3
- [66] F. Hanisch, J. Görke und W. Straßer. *A Visual Scripting Technique for the GRIS/ILO Interactive Learning Objects*. Conference on Educational Multimedia, Hypermedia and Telecommunications 2004 (1), Seite 1347–1350. 2004. 3.6
- [67] F. Hanisch, J. Görke und W. Straßer. *The educational use of live graphics gems: a walkthrough for aliasing*. WSI Report (WSI-2005-9). 2005. 1.2, 5.3
- [68] F. Hanisch und W. Straßer. *How to include visuals and interactivities in an educational computer graphics repository*. Computers & Graphics 29 (2), Seite 237–243. 2005. 1.1, 3.7.2

- [69] E. R. Harold und W. S. Means. *XML in a nutshell*. O'Reilly & Associates, Inc. Sebastopol, California. Second Edition 2002. 2.2, 2.3
- [70] B. Harvey. *Computer Science Logo Style - Vol. 1: Intermediate Programming*. MIT Press Cambridge, Mass. Second Edition 1985. 3.3
- [71] B. Harvey. *Computer Science Logo Style - Vol. 2: Projects, and Techniques*. MIT Press USA. First Edition 1985. 3.3
- [72] B. Harvey. *Computer Science Logo Style - Vol. 3: Beyond Programming*. MIT Press USA. 1985. 3.3
- [73] P. S. Heckbert, Editor. *Graphics gems IV* Band 4 von *Graphics Gems*. AP Professional Boston, MA, USA. 1994. 5.1, 5.3
- [74] M. Hitz und S. Kogeler. *Teaching C++ on the WWW*. In *Proceedings of the 2nd conference on Integrating Technology into Computer Science Education* Seite 11–13 Uppsala, Sweden. 1997. 5.2
- [75] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano und I. Goryainin. *The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models*. Technischer Report EDIINFRR0565 The University of Edinburgh März 2003. 2.1
- [76] IEEE Learning Technology Standards Committee. *Draft Standard for Learning Object Metadata*. http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf 2002. 3.7.1
- [77] International Organization for Standardization. *Standard Generalized Markup Language (SGML)*. ISO 8879:1986 Edition 1986. 2.1
- [78] International Organization for Standardization. *Information Technology — ECMA-Script Language Specification*. ISO/IEC 16262 Juni 2002. 2.4.1, 2.6
- [79] International Organization for Standardization. *Open Document Format for Office Applications (OpenDocument) v1.0*. ISO/IEC 26300 2006. 2.4
- [80] D. Jamous. *MathML for Mathematical Equations on the Web*. The Insider Fall/Winter Issue. 2003. 4.3
- [81] A. C. Kay. *The Early History of Smalltalk*. In *History of Programming Languages* Seite 511–579. ACM Press/Addison-Wesley 1996. 3.3
- [82] R. Klein und F. Hanisch. *Using a modular construction kit for the realization of an interactive Computer Graphics course*. In T. Ottmann und I. Tomek, Editoren, *ED-MEDIA & ED-TELECOM; world conference on educational multimedia and hypermedia & world conference on educational telecommunications*. Association for the Advancement of Computing in Education (AACE), USA 1998. 4.3

- [83] R. Klein, F. Hanisch und W. Straßer. *Web based Teaching of Computer Graphics: Concepts and Realization of an Interactive Online Course*. In M. Cohen, Editor, *SIGGRAPH 98 Conference Proceedings Annual Conference Series*. ACM SIGGRAPH Addison Wesley 1998. 4.3, 5.2, 5.6
- [84] D. Knox, D. Goelman, S. Fincher, J. Hightower, N. Dale, K. Loose, E. Adams und F. Springsteel. *The Peer Review Process of Teaching Materials: Report of the ITiCSE'99 Working Group on Validation of the quality of teaching materials*. In *ITiCSE-WGR '99: Working group reports from ITiCSE on Innovation and technology in computer science education* Seite 87–100. ACM Press 1999. 5.2
- [85] R. Koper, H. Spoelstra und D. Burgos. *Learning Networks using Learning Design. A first collection of papers*. OpenUniversiteitNederland. November 2004. 3.8
- [86] M. Kurzdin. *Wissenswettbewerb - Die kostenlose Wikipedia tritt gegen die Marktführer Encarta und Brockhaus an*. c't - Computer und Technik 21, Seite 132–139. Oktober 2004. 2.9
- [87] J. R. Laleuf und A. M. Spalter. *A component repository for learning objects: a progress report*. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries* Seite 33–40. ACM Press 2001. 2.9, 3.6, 3.9, 5.2, 5.6
- [88] L. Lamport. *Latex User's Guide and Reference Manual*. Addison Wesley. 2nd Edition 1994. 2.4
- [89] S. M. Lang und P. C. Lockemann. *Datenbankeinsatz*. Springer. 1995. 2.7
- [90] H. Lieberman. *The TV Turtle, a Logo Graphics System for Raster Displays*. Computer Graphics 10 (1), Seite 66–72. Spring 1976. 3.3
- [91] V. M. F. R.-A. M. *High Level Design of Web-Based Environments for Distance Education*. Kluwer Academic Publishers. 2000. 3.8
- [92] T. Mailänder. *Visualisierung von ALK-Daten mittels SVG*. Diplomarbeit Fachhochschule Stuttgart - Hochschule für Technik <http://www.bgrundviewer.de/> 2003. 2.4.1
- [93] J. Marsh, D. Orchard und D. Veillard. *XML Inclusions (XInclude) Version 1.0, Second Edition*. World Wide Web Consortium, Recommendation REC-xinclude-20061115 November 2006. 2.1
- [94] Massachusetts Institute of Technology. *MathML at MIT*. <http://web.mit.edu/ist/topics/webpublishing/mathml/index.html> 2006. 4.3
- [95] Massachusetts Institute Of Technology. *MITOpenCourseWare*. <http://ocw.mit.edu/index.html> 2006. 1
- [96] C. Matrix. *CMS Comparison*. <http://cmsmatrix.org/> 2006. 3.6

- [97] S. Mazzocchi. *Short History of Cocoon*. <http://cocoon.apache.org/history.html> 2003-2005. 4.2
- [98] B. McLaughlin. *Java and XML data binding*. Nutshell handbook. O'Reilly & Associates, Inc. Sebastopol, California. 2002. 2.7
- [99] D. Megginson und D. Brownell. *Saxproject*. <http://www.saxproject.org> 2006. 2.3
- [100] M. D. Merrill. *Learner control in computer based learning*. Computers & Education 4 (2), Seite 77–95. 1980. 3.4
- [101] B. Meyer. *Object-Oriented Software Construction*. Prentice Hall Professional Technical Reference Santa Barbara, California. Second Edition 1997. 6.2
- [102] T. Michel. *XML kompakt - Eine praktische Einführung*. Hanser. 1999. 2.2
- [103] S. Münz. *Selfhtml*. <http://www.selfhtml.org> 2006. 2.2, 2.3, 2.5
- [104] M. Muehlen. *Workflow-based Process Controlling*. Logos Verlag Berlin. 2004. 6.2, 6.3
- [105] C. Musciano und B. Kennedy. *HTML & XHTML: The Definitive Guide*. O'Reilly & Associates, Inc. Sebastopol, California. Fifth Edition 2002. 2.3
- [106] T. H. Nelson. *The Xanadu Ideal*. <http://www.xanadu.com.au/general/ideal.html> 8. Oktober 1993. 3.3
- [107] J. Nielsen. *Multimedia and Hypertext. The Internet and Beyond*. Boston: Academic Press. 1995. 3.3
- [108] T. Oetiker. *MRTG: The Multi Router Traffic Grapher*. In *Proceedings of the 12th Conference on Systems Administration (LISA-98)* Seite 141–148 Berkeley, CA. USENIX Association. 6.–11. Dezember 1998. 4.5
- [109] R. Olleck. *Lernszenarien für interaktive Online-Lehrbücher mit Quelltext*. Studienarbeit, Universität Tübingen, WSI/GRIS 2006. 1.2, 5.3
- [110] T. O'Reilly. *What Is Web 2.0*. <http://www.oreillyn.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html> 2005. 2.9
- [111] Organization for the Advancement of Structured Information Standards (OASIS). *RELAX NG Compact Syntax*. <http://www.oasis-open.org/committees/relax-ng/compact-20021121.html> November 2002. 2.2
- [112] G. S. Owen, R. Sunderraman und Y. Zhang. *The development of a digital library to support the teaching of computer graphics and visualization*. Computers and Graphics 24 (4), Seite 623–627. August 2000. 5.2, 5.5

- [113] G. S. Owen, Y. Zhu, J. Chastine und B. R. Payne. *Teaching Programmable Shaders: Lightweight versus Heavyweight Approach*. SIGGRAPH 2005 Education Paper. 2005. 5.2
- [114] S. Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books New York. 1980. 3.3, 5.3
- [115] C. A. Petri. *Kommunikation mit Automaten*. Institut für Instrumentelle Mathematik Bonn. 1962. 6.2
- [116] PHP. *PHP - Pretty HomePage*. <http://www.php.net> 2006. 4.5
- [117] Queensland University of Technology, Brisbane Australia. *ELP: Environment for Learning to Program*. <http://elp.fit.qut.edu.au/> 2006. 5.2
- [118] A. Rawlings, P. van Rosmalen, R. Koper, M. Rodríguez-Artacho und P. Lefre-re. *Survey of Educational Modelling Languages (EMLs)*. Technischer report CEN/ISSS WS/LT Learning Technologies Workshop 2002. 3.8
- [119] K. Reich. *Konstruktivistische Didaktik - ein Lehr- und Studienbuch inklusive Methodenpool auf CD*. Beltz-Verlag Weinheim u.a. 2006. 3.2
- [120] E. Robertsson. *An Introduction to Schematron*. xml.com. November 2003. 2.2
- [121] G. Roelofs. *The Story of PNG*. 1999. 2.4.1
- [122] R. Salih. *Interaktive grafische Modellierung von Workflows in einem Content Management System*. Diplomarbeit Universität Tübingen 2006. 6.3.3
- [123] F. Schiller. *Die Schaubühne als eine moralische Anstalt betrachtet*. <http://gutenberg.spiegel.de/schiller/anstalt/anstalt.htm> 1784. 3.3
- [124] R. Schulmeister. *Grundlagen hypermedialer Lernsysteme*. Oldenbourg Verlag München. 1997. 1.2, 3.3, 3.6, 3.7
- [125] R. Schulmeister. *Lernplattformen für das virtuelle Lernen. Evaluation und Didaktik*. R. Oldenbourg Verlag. 2003. 3.6
- [126] B. A. Sherwood. *The TUTOR Language*. 1977. 3.4
- [127] B. Shneiderman, C. Plaisant, R. A. Botafogo, D. Hopkins und W. J. Weiland. *Designing to Facilitate Browsing: A Look Back at the Hyperties Workstation Browser*. Hypermedia 3 (2), Seite 101–117. 1991. 3.3
- [128] P. Simons. *FastCGI | The Forgotten Treasure* Juni 19 2001. 2.6
- [129] K. Smith-Gratto, D. Wicks und C. Berger. *MERLOT: Reaping the On-line Vineyard*. In *Proceedings of ED-MEDIA* 2002. 3.9, 5.2

- [130] J. M. Smith. *CALS in Europe: an integrated information base*. Logistics Information Management 4 (1), Seite 26–31. 1991. 2.1
- [131] A. M. Spalter, R. M. Simpson, M. Legr und S. Taichi. *Considering a Full Range of Teaching Techniques for Use in Interactive Educational Software: A Practical Guide and Brainstorming Session* 5. Juni 2000. 3.9
- [132] A. M. Spalter und A. van Dam. *Problems with using components in educational software*. Computers & Graphics 27 Seite 329–337. 2003. 5.1
- [133] Sun Developer Network. *Java SE - Java Database Connectivity (JDBC)*. <http://java.sun.com/javase/technologies/database.jsp> 2006. 4.2
- [134] J. Sweller, J. Merrienboer und F. Paas. *Cognitive architecture and instructional design*. Educational Psychology Review (10), Seite 251–296. 1998. 5.4
- [135] Targeteam. *TeachML 1.2*. <http://www11.in.tum.de/forschung/projekte/targeteam/extension/TeachML.dtd> 2006. 3.8
- [136] The CodeSaw Project. *CodeSaw*. <http://www.codesaw.com> 2006. 5.2
- [137] The GIMP Team. *GIMP - The GNU Image Manipulation Program*. <http://www.gimp.org> 2006. 4.5
- [138] The Merlot Project. *About the Peer Review Process*. <http://taste.merlot.org/peer-reviewprocess.html> 2006. 3.9
- [139] The Merlot Project. *MERLOT Learning Materials*. <http://taste.merlot.org/learning-materials.html> 2006. 3.9
- [140] The Merlot Project. *The Merlot Project*. <http://www.merlot.org/> 2006. 1, 3.9
- [141] The Ruby Language Project. *The Ruby Language*. <http://www.ruby-lang.org/> 2006. 5.2
- [142] N. Truong, P. Bancroft und P. Roe. *Learning to program through the web*. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education* Seite 9–13 New York, USA. ACM Press. 2005. 5.2
- [143] Turing's Craft. *CodeLab: A Powerful Tool for Programming Instruction*. <http://www.turingscraft.com/> 2006. 5.2
- [144] W. M. P. van der Aalst, L. Aldred, M. Dumas und A. H. M. ter Hofstede. *Design and Implementation of the YAWL System*. Januar 2004. 6.2

- [145] W. M. P. van der Aalst und A. H. M. ter Hofstede. *Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages*. In K. Jensen, Editor, *Proceedings of the Fourth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2002)* Band 560 von *DAIMI* Seite 1–20 Aarhus, Denmark. University of Aarhus. August 2002. 6.2
- [146] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski und A. P. Barros. *Workflow Patterns*. *Distributed and Parallel Databases* 14 (1), Seite 5–51. 2003. 6.2
- [147] J. Vesperman. *Essential CVS*. O'Reilly & Associates Sebastopol, California. Juni 2003. 2.7
- [148] L. Wall, T. Christiansen und J. Orwant. *Programming Perl*. O'Reilly Sebastopol, California. 3 Edition 2000. 2.6
- [149] N. Walsh und L. Muellner. *DocBook: The Definitive Guide*. O'Reilly & Associates Sebastopol, California. Juli 1999. 2.1, 2.4, 5.2
- [150] P. S. Wang, N. Kajler, Y. Zhou und X. Zou. *WME: towards a web for mathematics education*. In *ISSAC '03: Proceedings of the 2003 international symposium on Symbolic and algebraic computation* Seite 258–265. ACM Press 2003. 5.2
- [151] Web3D Consortium. *ISO/IEC 19775:2004, Extensible 3D (X3D)*. <http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/> 2004. 2.4.2
- [152] Wikipedia. *Collaboration*. <http://en.wikipedia.org/wiki/Collaboration> 2006. 3.5
- [153] Wikipedia. *Continuation*. <http://en.wikipedia.org/wiki/Continuation> 2006. 2.6
- [154] Wikipedia. *E-Learning*. <http://de.wikipedia.org/wiki/E-learning> 2006. 3.6
- [155] Wikipedia. *History of virtual learning environments*. http://en.wikipedia.org/wiki/History_of_virtual_learning_environments 2006. 3.4
- [156] Wikipedia. *Peer-Review*. <http://en.wikipedia.org/wiki/Peer-review> 2006. 6.5
- [157] N. Wirth. *What can be do about the unnecessary diversity of notation for syntactic definitions?* *Communications of the ACM* 20 (11), Seite 882. November 1977. 2.2
- [158] D. R. Woolley. *PLATO: The Emergence of Online Community*. <http://thinkofit.com/plato/dwplato.htm> 1994. 3.4
- [159] World Wide Web Consortium. *Mathematical Markup Language - MathML*. <http://www.w3.org/TR/MathML2/>. 2.4, 4.3
- [160] World Wide Web Consortium. *Resource Description Framework - RDF*. <http://www.w3.org/RDF>. 2.4

- [161] World Wide Web Consortium. *XHTML 1.0 The Extensible HyperText Markup Language*. <http://www.w3.org/TR/xhtml1/>. 2.5
- [162] World Wide Web Consortium. *XML Path Language (XPath) Version 1.0, W3C Recommendation*. <http://www.w3c.org/TR/xpath> November 1999. 2.3
- [163] World Wide Web Consortium. *RSS 1.0 Specification - RDF Site Summary (RSS) 1.0*. <http://web.resource.org/rss/1.0/spec> 2001. 4.5
- [164] World Wide Web Consortium. *Web Services Description Language (WSDL) 1.1*. <http://www.w3.org/TR/wsdl.html> 2001. 2.9
- [165] World Wide Web Consortium. *XQuery: The W3C query language for XML – W3C working draft*. <http://www.w3.org/TR/xquery/> 2001. 2.7
- [166] World Wide Web Consortium. *Scalable Vector Graphics - SVG, W3C Recommendation*. <http://www.w3.org/TR/SVG11/> Januar 2003. 2.4.1
- [167] World Wide Web Consortium. *The XMLHttpRequest Object*. <http://www.w3.org/TR/XMLHttpRequest/> 2006. 2.9
- [168] World Wide Web Consortium. *XSLT stylesheets for MathML*. <http://www.w3.org/Math/XSL/Overview-tech.html> 2006. 4.3
- [169] N. Yankelovich, N. Meyrowitz und N. V. Dam. *Reading and writing the electronic book*. IEEE Computer 18 (10), Seite 15–30. Oktober 1985. 3.3
- [170] Youtube.com. *Youtube*. <http://www.youtube.com> 2006. 2.9
- [171] C. Ziegeler und M. Langham. *Cocoon: Building XML Applications*. New Riders. 2002. 4.2

- ACID, 23
- Aggregation, 45, 67
- Ajax, 25, 60
- Akkomodation, 28
- Amaya, 46
- Andreessen, Marc, 19
- Apache Lenya, 66, 81, 85
- Apache Tomcat, 8
- ASP, 42
- Assimilation, 28
- Atkinson, Bill, 30
- Attribut, 9
- Aufnahmekapazität
 - geistige, 62
- Auswahl im Quelltext, 62
- Auszeichnungssprache, 7, 64
- Authentifizierung, 67
- Autorisierung, 67

- Backus-Naur-Form, 10
- Baeyens, Tom, 83
- Baumgartner, Peter, 27, 34
- Behaviourismus, 27
- Berners-Lee, Tim, 19
- Bibliothek
 - libjpeg, 51
 - libpng, 51
 - universelle, 29
- Bildungsmedium, 29
- Bitzer, Donald, 31
- Blackboard, 32

- Blended Learning, 33
- Blog, 25
- Blumstengel, Astrid, 27
- branching, 69
- Brecht, Bertolt, 29
- Browserkrieg, 20
- Bruner, Jerome S., 30
- Bush, Vannevar, 29, 30

- Caching, 70
- CAT, 32
- CBT, 32
- CGEMS, 55
- CGI, 21
- CMS, 66, 81, 85
- Cocoon, 41, 49, 51, 67
- CodeLab, 59
- CodeSaw, 60
- Cognitive Load Theorie, 62
- Computergrafik, 55, 58
- Computergrafikkurs, 59
- Connexions-Projekt, 58, 72
- Continuation, 22
- Curriculum, 55
- CVS, 69

- DB2, 8
- de Mattia, Christoph, 89
- DFT, 72
- Diskretisierung, 75
- Document Object Model (DOM), 8, 12

- Docuverse, 29
Dougherty, Dale, 25
Dougiamas, Martin, 32
Drag-And-Drop-Paradigma, 46, 47
Dynabook, 30
- E-Learning, 32
Element, 8
ELP, 59
Engelbart, Douglas, 30
Erneuerungsprozess, 57
Exploration, 61
Exploratories, 39, 59, 72
- Farbkanal, 75
Filtermethode, 75
Filteroperation, 75
Firefox, 20, 46
Flash-Objekt, 65
Flussgraph-Modellierung, 86
Forschungsartikel, 57
Forum, 47
- gültig, 10
Geschke, Chuck, 15
GIMP, 51
Glassner, Andrew S., 59, 72
GML, 7
Goldberg, Murray, 32
Goldfarb, Charles, 7
Graphic Gems, 59
- Hanisch, Frank, 39, 41, 58
Harold, Elliotte Rusty, 15
Heckbert, Paul S., 56
Helene, 71
Histogramm
 kumuliertes, 75
Histogrammeinebnung, 74
Hitz, Martin, 60
Hough, Paul V. C., 75
Houghraum, 75
Houghtransformation, 75
HTML, 8, 19
- HTTP, 19, 24
Hyper-G, 32
HyperCard, 30
- IDE, 70
Identifikationslernen, 29
Illustration, 61
ILO, 34, 72, 74
Imitationslernen, 29
Interaktivität
 quelltextbasierte, 56, 60, 96
Internet Explorer, 46
- Java-Applet, 65
JavaDoc, 66
JBoss, 8
jEdit, 65
JSP, 42
- Kögeler, Stefan, 60
Kay, Alan, 30
Kognitivismus, 28
kollaborativ, 32
Kompilieren
 serverseitiges, 56, 60, 68, 96
Konqueror, 20
Konstruktionismus, 30
Konstruktivismus, 28
kooperativ, 32
Kopfzeile, 9
- Lehrmaterialien, 57
Lehrmaterialsammlung, 64
Lernmanagementsystem, 33
Lernobjekt, 34
Lernobjektsammlung, 39
Lernszenarien, 61
Lerntheorie, 27
LIVE, 45
Lizenzmodell, 68
LMS, 32
Load
 Extraneous Cognitive, 63
 Germane Cognitive, 63

- Intrinsic Cognitive, 63
- Logo, 30
- LOM, 35
- Lorie, Raymond, 7
- Münz, Stefan, 12, 15
- Masse
 - kritische, 55
- MathPlayer-Plugin, 46
- MathType, 45
- Mazzocchi, Stefano, 41
- Memex, 29
- Merlot, 39, 59
- Merrill, M. David, 31
- Metacoon, 32
- Microsoft Office, 8
- Microworlds, 61
- Model-View-Control-Paradigma, 45
- Moodle, 32
- Mosaic, 19
- Mosher, Edward, 7
- Mozilla, 46
- Multiple-Choice, 45, 52, 61
- Nelson, Ted, 29
- NEOS, 32
- Netscape, 19, 46
- NLS, 30
- Nomenklatur, 70
- Nutzerverwaltung, 67
- O'Reilly, Tim, 25
- Officetalk, 82
- OpenOffice, 8, 45
 - Draw, 86
- Oracle, 8
- Owen, Scott, 60
- Papert, Seymour, 30
- Paradigmenwechsel, 42
- Payr, Sabine, 27
- PDF, 15, 49
- Peer-Reviewing, 91
- Petri, Carl Adam, 82
- Petrinetz, 82
- PHP, 42, 51
- Piaget, Jean, 30
- Pipeline, 42, 67
- PLATO, 31
- Plugin, 20
- Programmierung, 61
- Quelltext, 61
- Quelltext-Editor
 - webbasierter, 64
- Quelltextbesprechung, 62
- Quelltextvergleich, 62
- rapid prototyping, 65
- Reiz-Reaktions-Maschine, 27
- Relax NG, 11
- Repositorium, 39, 55
- Ruby, 60
- Safari, 20
- Sanger, Larry, 25
- Schematron, 11
- Schiller, Friedrich, 29
- Schulmeister, Rolf, 31, 34
- SCORM, 36
- Selbstbefragung, 45
- SGML, 7
- Shader-Programm, 60
- Shneiderman, Ben, 31
- Simple API for XML (SAX), 8, 12
- Skinner, Burrhus Frederic, 28
- Skinnerbox, 28
- Skripting, 47, 61
- SOAP, 24
- Sobel-Operator, 75
- Spreizungsoperation, 75
- Stevens, Jon, 41
- Subversion, 69
- Sweller, John, 62
- Syntax-Highlighting, 71
- tag, 8
- Textbuch, 61

- TICCIT, 31
- Toolbook, 32
- Transklusion, 30
- Trennung von Inhalt und Layout, 24
- Trennung von Logik, Inhalt und Design, 42
- TUTOR, 31
- Typo3, 81

- UDDI, 25
- Urheberrecht, 56
- URI, 12
- URL, 19

- valid, 10
- van Dam, Andries, 31, 39
- van der Aalst, Wil M. P., 82
- Versionieren
 - serverseitiges, 56, 96
- Vico, Giambattista, 29

- Wales, Jimmy, 25
- WAPI, 82
- Warnock, John, 15
- Watson, John Broadus, 27
- WBT, 32
- Web 2.0, 25
- WebCT, 32
- Webseite
 - dynamische, 21
- WebToTeach, 59
- well-formed, 9
- WfMC, 82
- Wiki, 25
- Wikipedia, 25
- wohlgeformt, 9
- Workflow, 82
 - Konfigurationsdatei, 85
- Workflowengine, 67
- WSDL, 25
- WSI/GRIS, 45, 69, 78
- Wurzelement, 9
- WWW, 19

- Xalan, 13

- Xanadu, 29
- XML
 - Anwendungen, 8
 - BPEL4WS, 82
 - CML, 8
 - CNML, 58
 - datenzentriertes, 23
 - Docbook, 8, 15
 - DTD, 10
 - Dublincore, 16
 - GML, 8
 - MathML, 8, 16, 44, 58
 - MeML, 58
 - Namensraum, 12
 - narratives, 23
 - orientiert, 42
 - PCDATA, 10
 - RDF, 16
 - RSS-Feed, 50
 - SBML, 8
 - SMIL, 8
 - Speicherung, 23
 - SVG, 8, 16, 50, 58, 62
 - WS-CDL, 82
 - WSFL, 82
 - X3D, 8, 18, 58
 - XHTML, 20, 49
 - XInclude, 8
 - XLANG, 82
 - XLink, 8
 - XML-Schema, 10
 - XPath, 8, 13
 - XPDL, 82
 - XPointer, 8
 - XSL, 8, 13, 24
 - FO, 14, 49
 - Transformation, 87
 - XML-Editor, 65
 - XMLHttpRequest, 25

- YAWL, 87

- zur Mühlen, Michael, 82
- Zustandsautomat, 67, 84