Fachhochschule Oldenburg Ostfriesland Wilhelmshaven
University of Applied Sciences

Bachelorarbeit

Integration von PDF-Formularen als Benutzerschnittstelle von computergenerierten Aufgaben in LON-CAPA

Vorgelegt von:
Thomas Onken

Emden, Juli 2008

Betreut von :
Prof. Dr. Gerd von Cölln
Prof. Dr. Peter Riegler

Bachelorarbeit an der
Fachhochschule Oldenburg/Ostfriesland/Wilhemshaven
Fachbereich Technik
Studiengang Medientechnik

Thema : Integration von PDF-Formularen als Benutzerschnittstelle von computergenerierten Aufgaben in LON-CAPA
Autor : Thomas Onken

Thomas Onken
Königstr. 17
26789 Leer
Email: tonken@et-inf.fho-emden.de

ERKLÄRUNG

Soweit meine Rechte berührt sind, erkläre ich mich einverstanden, dass die Bachelor-Arbeit Angehörigen der Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven für Studium/ Lehre/Forschung uneingeschränkt zugänglich gemacht werden kann.

EIDESSTATTLICHE ERKLÄRUNG

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Bachelor-Arbeit bis auf die offizielle Betreuung selbst und ohne fremde Hilfe angefertigt habe und die benutzten Quellen und Hilfsmittel vollständig angegeben sind.

Emden, den 11.07.2008

.....

Ort, Datum (Thomas Onken)

Danksagung

Als ich meine erste Vorlesung in der Fachhochschule in Emden hörte, dachte ich mir im Stillen: Hier wirst du also mindestens die nächsten drei Jahre verbringen - wenn denn alles klappt. Mir schien, als läge eine kleine Ewigkeit vor mir, doch die Zeit verging wie im Flug.

Viele Menschen kreuzten meinen Weg und nun ist es an der Zeit einigen davon zu danken. Eine schwierige Sache, Dank auszusprechen, für ein Ziel, das man sich selber gesteckt hat und erreichen wollte. Trotzdem möchte ich es an dieser Stelle versuchen.

Ich bedanke mich bei:

Meinen Eltern **Annegret** und **Focko**, die mich mein ganzes Leben lang unterstützt haben.

Björn Lünemann und **Claus Lachmann**, ohne deren Bekanntschaft ich nie oder erst später auf die Idee gekommen wäre ein Studium zu beginnen.

Florian Ast, der nach einiger Zeit ein enger und treuer Wegbegleiter wurde.

Dem **OMI-Team** für die fortwährende Unterstützung und das mir entgegengebrachte Vertrauen.

Prof. Dr. Gerd von Cölln, **Prof. Dr. Peter Riegler** und **Dipl.-Inf. Stefan Bisitz** für die Unterstützung während dieser Abschlussarbeit.

Am meisten jedoch möchte ich mich bei meiner Lebensgefährtin **Birgit Meyer** bedanken, die immer da war und hoffentlich auch sein wird.

Alles Wissen ist vergeblich ohne die Arbeit.
Und alle Arbeit ist sinnlos ohne die Liebe.
– Khalil Gibran, Sämtliche Werke –

Kurzzusammenfassung

Die vorliegende Bachelorarbeit befasst sich mit der Integration von PDF-Formularen als Benutzerschnittstelle zur Bearbeitung von computergenerierten Aufgaben ins Online-Lernraumsystem LON-CAPA (The Learning Online Network with Computer-Assisted Personalized Approach).

Es wird untersucht, wie Aufgaben bisher im Web-Browser und über die Druckfunktion in LON-CAPA dargestellt und bearbeitet werden können. Die daraus gewonnenen Erkenntnisse werden zur Erweiterung der Druckfunktion angewandt, so dass optional zu druckbaren PDF-Dokumenten PDF-Formulare erstellt werden können. In diese können die Ergebnisse der Aufgaben eingetragen und zur Bewertung eingereicht werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

The Learning Online Network with Computer-Assisted Personalized Approach (LON-CAPA) ist ein Online-Lernraumsystem mit dem Schwerpunkt zur automatisierten Erstellung und Bewertung von rechnergestützten Aufgaben. LON-CAPA wird unter anderem an der Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven und der Fachhochschule Braunschweig/Wolfenbüttel eingesetzt. Die von LON-CAPA bereitgestellten Aufgaben können von Studierenden über einen Web-Browser genutzt werden. Die Lösungen der Aufgaben werden in ein HTML-Formular oder mit Hilfe von Java-Applets eingetragen und an LON-CAPA gesendet. Die Auswertung der Aufgaben wird direkt erzeugt und an die Studenten zurückgesendet. Bei diesem Verfahren wird während der Aufgabenbearbeitung eine Verbindung mit dem LON-CAPA-System benötigt. Eine sinnvolle Erweiterung wäre daher das Generieren von PDF-Dokumenten, die sowohl die Aufgabenstellungen als auch eine Eingabemöglichkeit zur Aufnahme und Speicherung der Antworten bereitstellen. Nach der Bearbeitung wird das Dokument an LON-CAPA gesendet. Darauf werden die Antworten aus dem Dokument ausgelesen und in das System eingepflegt, so dass die Studenten die Auswertung wie gewohnt im Web-Browser einsehen können.

Um einen Einstieg in die Thematik zu geben, werden in Kapitel 2 und 3 die Grundlagen von LON-CAPA, eine Einführung in den PDF-Standard sowie die Erzeugung von PDF-Formularen mit LaTeX beschrieben. In Kapitel 4 werden die Anforderungen an die Bearbeitung durch PDF-Formulare beschrieben. In Kapitel 5 wird die Aufgabenbearbeitung per Web-Browser analysiert. Darauf folgt in Kapitel 6 das Konzept des Bearbeitungsablaufs mit PDF-Formularen und die Planung der neuen Funktion in LON-CAPA. Das Kapitel 7 beschreibt die Umsetzung der geplanten Erweiterung. Zuletzt folgt noch eine Zusammenfassung der gewonnenen Erkenntnisse und weiterführende Ideen und Möglichkeiten, was zukünftig mit PDF-Dokumenten umgesetzt werden kann.

In der vorliegenden Bachelorarbeit wird an mehreren Stellen gezielt auf Originalquellen verwiesen, um den Leser nicht durch die zusätzliche Fülle an Informationen vom Hauptthema wegzuführen.

Für diese Abschlussarbeit werden Grundkenntnisse der Programmiersprachen Perl, Java und JavaScript, sowie Kenntnisse in HTML und LaTeX vorausgesetzt.

Als Anhang liegt eine DVD-ROM bei, auf der sich die eingesetzten freien Software-Produkte, die erstellten Software-Module (inkl. Sourcecodes) und eine Kopie der in LON-CAPA erweiterten Perl-Module befinden.

2 LON-CAPA

In diesem Kapitel soll eine Übersicht über die Einsatzgebiete und die Geschichte von LON-CAPA gegeben werden. Anschließend folgt eine kurze Beschreibung der für die Arbeit relevanten technischen Details.

2.1 Was ist LON-CAPA

The Learning Online Network with Computer-Assisted Personalized Approach (LON-CAPA) ist eine E-Learning-Plattform, vergleichbar mit einem Content- oder Learning-Management-System. Jedoch unterscheidet sie sich von anderen Systemen insofern, dass LON-CAPA die Möglichkeit bietet, dem „LearningOnline Network“, einem weltweiten Netzwerk, beizutreten. Dieses Netzwerk erlaubt es den LON-CAPA-Servern, auch als „LON-CAPA domains“ bezeichnet, Kurs-Materialien und Aufgaben untereinander auszutauschen. Die Autoren[1] und Kurs-Koordinatoren[2] erhalten Zugriff auf alle veröffentlichten Materialen und Aufgaben des Netzwerkes. LON-CAPA bietet für Aufgaben eine auf Perl und XML basierte Programmierschnittstelle, mit deren Hilfe beispielsweise auf ein Computer-Algebra-System zurückgegriffen werden kann. Die Aufgaben können für jeden Kursteilnehmer personifiziert werden. Das bedeutet, dass die einzelnen Aufgabenstellungen unterschiedliche Werte bei Mathematikaufgaben oder unterschiedliche Antworten bei Multiple-Choice-Aufgaben haben.

[1] Autor ist die Bezeichnung für eine Benutzerrolle in LON-CAPA, die die Berechtigung zur Erstellung und Änderung von Aufgaben hat.

[2] Kurs-Koordinator ist die Bezeichnung für eine Benutzerrolle in LON-CAPA, die die Berechtigung zur Verwaltung und Organisation eines Kurses hat.

Weiter können Aufgaben zu Übungsblättern oder Testaten zusammengefügt werden. Darüber hinaus kann den Kursteilnehmern eine begrenzte Zeit mit einer limitierten Anzahl von Beantwortungsversuchen zur Bearbeitung freigeschaltet werden. Bei der Bearbeitung, insbesondere bei Fehlern, erhalten die Kursteilnehmer eine sofortige Rückmeldung von System. Dabei ist es auch möglich, nach einer falschen Antwort zusätzliche Tipps zur Aufgabenstellung hinzuzufügen.

LON-CAPA ist ein Open-Source-Projekt und wird stetig weiterentwickelt. Es wird unter GNU General Public License (GPL) kostenlos zur Verfügung gestellt. Das Kern-Entwickler-Team befindet sich in der Michigan State University in den Vereinigten Staaten von Amerika. Im Jahre 1999 wurde LON-CAPA als Zusammenschluss aus zwei parallel entwickelten Systemen gegründet, zum einen „Computer-Assisted Personalized Approach(CAPA)“ und zum anderen „LectureOnline“. [lon]

Die Entwicklung von CAPA begann bereits 1992 und bot Studierenden der Physik personifizierte Übungsaufgaben und Testate. Bei der Bearbeitung wurden sofortige Rückmeldungen gegeben und Hinweise bei falschen Antworten waren möglich. Die Aktivität und die Leistung der Teilnehmer wurde im System gespeichert und konnte von Studierenden und von Lehrenden eingesehen werden. [lon]

Das LectureOnline-Projekt startete 1997 mit dem Ziel, ein komplett per Web-Interface steuerbares System zur Verteilung von Physik-Lehrmaterialien zu erstellen. Zusätzlich sollten individuelle Online-Hausaufgaben erzeugt werden. Benotungen, Kommunikation, Gruppenarbeit, und Benutzerverwaltung wurden ebenfalls von LectureOnline unterstützt. [lon]

2.2 Relevante technische Details

In diesem Abschnitt werden die für die Bachelorarbeit relevanten technischen Details von LON-CAPA beschrieben. Dazu gehören Betrieb und Aufbau des Servers, die verschiedenen Benutzerrollen, Aufgabentypen und bereits vorhandene Ausgabemöglichkeiten für Aufgaben.

2.2.1 Betrieb und Aufbau des Web-Servers

Für den Betrieb von LON-CAPA wird ein eigenständiger Linux-Server mit Fedora, SuSE oder Redhat Enterprise Distribution empfohlen. Bei anderen Linux-Distributionen kann es zu Problemen mit Versionskonflikten oder fehlenden Komponenten kommen. [lon] Um eine gute Performance bei der Verwaltung einer großen Anzahl von Benutzern zu erhalten, besteht die Möglichkeit, eine „LON-CAPA domain“ auf mehrere Server zu verteilen. Diese Verteilung bietet unter anderem den Vorteil, dass bei einem Ausfall eines einzelnen Zugangsservers die Interaktionen eines Benutzers auf einem der funktionsfähigen Server zwischengespeichert werden kann. Steht der ausgefallene Server wieder zur Verfügung, werden die Daten mit dem Zwischenspeicher abgeglichen. [AKH02]

LON-CAPA läuft auf einem Apache-Web-Server über die Apache-Perl-API[3]. Das hat zur Folge, dass die verschiedenen Bereiche und Funktionen von LON-CAPA in Perl-Module aufgeteilt werden. Eine ausführliche Liste der Zuständigkeiten der Module kann in der LON-CAPA Entwicklerdokumentation [AKH02] nachgelesen werden. Dieses Dokument kann auf der LON-CAPA Internetseite [lon] heruntergeladen werden. In dieser Dokumentation wird an einigen Stellen auf ein „Eagle book“ verwiesen, wobei es sich um das Buch „Writing Apache Modules with C and Perl“ von Lincoln Stein und Doug MacEachern [SM99] handelt. Dieses Buch gibt eine gute Einführung in die Architektur der Apache Perl- und C-API. An späterer Stelle wird in der Analyse auf diese Thematik detaillierter eingegangen.

2.2.2 XML als Grundstruktur

Die Darstellung einer Web-Seite und die Verarbeitung der Benutzer-Interaktion werden in LON-CAPA als XML-Struktur abgebildet. Dabei stehen die einzelnen öffnenden und schließenden XML-Elemente für verschiedene Methoden in den Perl-Modulen. Ein öffnendes XML-Element entspricht dabei einer Methode *start_ Name des Elements()* und ein schließendes Element einer Methode *end_ Name des Elements()*. Alle Methoden werden in der Reihenfolge der XML-Strukur ausgeführt. Um festzulegen welches Modul für welches XML-Element zuständig ist, werden diese im Modul *lonxml.pm* registriert. Dieses geschiet, so bald die einzelnen Module geladen werden in einem *BEGIN{}* Block. Anschließend kann in den start-Methoden ein bereits registriertes XML-Element überlagert werden, so dass die Start- und End-Methoden eines anderen Moduls benutzt werden. In den End-Methoden

[3] Eine genauere Beschreibung der Apache-Perl-API folgt in Kapitel 5

können diese Überlagerungen wieder zurück genommen werden, so dass die ursprüngliche Zuordnung wieder gültig wird. Ein Beispiel dafür wird in der Analyse der Aufgabendarstellung in Kapitel 5 genauer betrachtet.

2.2.3 Benutzerrollen

In LON-CAPA existieren zur Organisation und Aufteilung der Berechtigung im wesentlichen drei Benutzerrollen: Student, Kurs-Koordinator und Autor. Dabei können einem Benutzer mehrere Rollen zugewiesen werden. Mit der Rolle Student kann der Benutzer in einen Kurs aufgenommen werden und hat Zugriff auf die im Kurs verfügbaren Inhalte. Der Kurs-Koordinator legt Kurse an, in die er andere Benutzer mit Studentenrolle als Kursteilnehmer aufnehmen kann. Um einen Kurs aufzubauen, können vom Kurs-Koordinator Lernmaterialen in Form von Web-Seiten und einfachen Aufgabentypen erstellt werden. Er besitzt jedoch auch die Berechtigung, öffentliche komplexere Inhalte und Aufgaben in den Kurs zu importieren. Diese Inhalte werden mit der Rolle Autor erstellt und bereitgestellt. Änderungen an öffentlichen Inhalten werden allen Kurs-Koordinatoren, die diesen Inhalt importiert haben, durch ein Verwaltungskonzept angezeigt. Sie können dann entscheiden, ob die alte gegen die neue Version ausgetauscht werden soll.

2.2.4 Aufgabentypen

LON-CAPA bietet die Möglichkeit computergestützte Aufgaben in einen Kurs zu integrieren. Dafür stehen mehrere Aufgabentypen zur Verfügung. Diese sollen hier kurz beschrieben werden.

Radio Response

Bei einer Radio Response Aufgabe werden zu einer Frage mehrere Antworten in Multiple-Choice Form angeboten von denen nur eine richtig ist.

Match-, Option- und Rank Response

Match- und Option Response sind Aufgaben, bei denen der Kursteilnehmer die richtige Zuordnung aus einer Drop-Down-Box auswählen muss. Bei Rank Response müssen über

dasselbe Verfahren logische Sortierungen vorgenommen werden.

String Response

String Response gibt dem Kursteilnehmer die Möglichkeit seine Antwort als Zeichenkette anzugeben. Ein Beispiel dafür sind Vokabeltests, eindeutige Antworten auf Fragen und chemische Formeln, für die ein spezieller chemischer Formeleditor benutzt werden kann.

Numerical Response

Numerical Response ermöglicht die Eingabe von Zahlenwerten, optional können auch Einheiten angegeben werden. Zusätzlich kann vom Aufgabenautor die Anzahl der anzugebenden Nachkommastellen definiert werden.

Formula Response

Bei Formula Response werden mathematische Formeln als Antwort entgegengenommen. Die Antworten werden algebraisch oder numerisch geprüft.

Math Response

Mit Math Response werden die Antworten mit Hilfe des freien Computer-Algebra-Systems „Maxima“ ausgewertet. Dadurch wird es möglich freie Aufgaben an die Kursteilnehmer zu stellen, die mehr als eine richtige Antwort erlauben. Ein Beispiel dafür ist: „Nennen Sie eine ungerade Funktion!“.

Klick-ins-Bild

Klick-ins-Bild ermöglicht Aufgaben zu erstellen, die eine Grafik anzeigen in der der Kursteilnehmer per Mouse-Klick einen bestimmten Bereich auswählen muss.

Weitere Aufgabentypen

LON-CAPA bietet noch weitere Aufgabentypen, die zum Teil spezielle Formen der oben aufgeführten Typen darstellen. Eine Übersicht der vordefinierten Aufgabentypen im Autor-Bereich von LON-CAPA sollen in Abbildung 2.1 aufgezeigt werden.

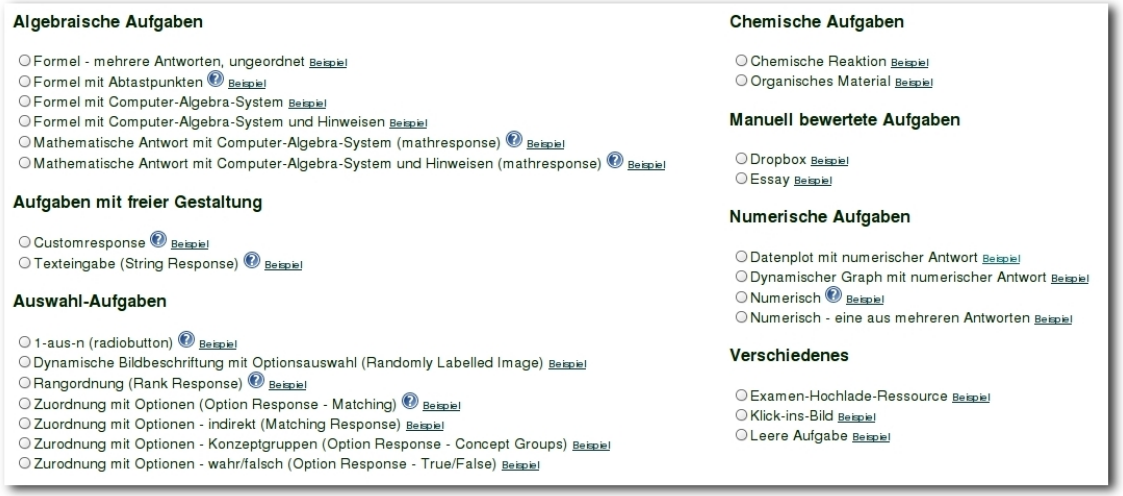

Abbildung 2.1: Übersicht der vordefinierten Aufgabentypen (Screenshot aus LON-CAPA)

2.2.5 Ausgabemöglichkeiten für Aufgaben

In LON-CAPA wird neben der Bearbeitung per Web-Browser die Möglichkeit geboten, einzelne und gesammelte Aufgaben als PDF-Dokument herunterzuladen. Es besteht auch die Möglichkeit Aufgabensätze als Multiple-Choice für einen ganzen Kurs zu erstellen, um sie beispielsweise in einer Klausur einzusetzen. Die Kursteilnehmer können ihre Antworten auf einem Bubble-Sheet[4] eintragen, das später mit einem Scanner zur automatischen Auswertung eingelesen wird.

[4] Bubble-Sheets werden hauptsächlich an amerikanischen Schulen eingesetzt. Es handelt sich dabei um einen genormten Vordruck, auf dem die Antworten zu den einzelnen Aufgaben angekreuzt werden.

3 Grundlagen zum Portable Document Format (PDF)

In den letzten Jahren hat sich das PDF-Format immer mehr als Standard für elektronische Dokumente durchgesetzt. Vor allem im Internet findet es großen Zuspruch. Im Laufe der Zeit wurde der für PDF zu Grunde liegende Standard in mehreren Schritten erweitert, so dass heute verschiedene Versionen vorliegen. Im Folgenden soll ein kurzer Einstieg in die derzeitigen Eigenschaften des PDF gegeben werden. Um nur dessen Grundidee zu vermitteln, wird im Folgenden nicht zu stark auf Details eingegangen.

3.1 Informationen zu PDF

Das Portable Document Format (PDF) ist das grundlegende Datei-Format für die Adobe Acrobat Produktfamilie. Ziel dieses Formats ist es, den Benutzern einen einfachen Austausch von Dokumenten zwischen verschiedenen Plattformen und Geräten zu bieten. Das Format wurde von Adobe offen gelegt und in mehreren Versionen nach ISO spezifiziert. PDF basiert auf dem Model von PostScript[1], jedoch ist die Datenstruktur im PDF dahin erweitert worden, dass eine schnelle Betrachtung ermöglicht wird. Zudem bietet das PDF-Format zusätzliche Funktionen, die dem Benutzer Interaktionsmöglichkeiten im Dokument geben. Dazu gehören in der aktuellen Version beispielsweise 3D-Objekte und per JavaScript programmierbare Formularfelder. [Ado06]

Zur Darstellung und Erzeugung von PDF-Dokumenten gibt es neben dem kostenpflichtigen Adobe Acrobat und dem im Funktionsumfang eingeschränkten AcrobatReader eine Vielzahl von freien Softwareprodukten, bei denen in der Regel nicht alle im Standard beschriebenen Funktionen und Eigenschaften unterstützt werden. Besonders das Speichern

[1] Das PostScript-Format stammt ebenfalls von Adobe Systems. Es wird auch heute noch im Printbereich zum geräteunabhängingen Datenaustausch eingesetzt.

von Formulardaten wird in freier Software sehr selten unterstützt. Eine Liste mit Informationen zu kommerzieller und freier PDF-Software findet sich auf verschieden Seiten im Internet[2].

3.1.1 Aufbau von PDF-Dokumenten

PDF-Dateien können in binärer Form mit 8-Bit oder in lesbarer Form mit 7-Bit im ASCII-Code (American Standard Code for Information Interchange) codiert werden. Zum Austausch zwischen verschiedenen Plattformen wird die binäre Form vorgeschlagen, wodurch auf eine Umrechnung der plattformabhängigen Zeichensätze verzichtet werden kann. Der grundlegende Aufbau einer PDF-Datei ist in Abbildung 3.1 dargestellt. [Ado06]

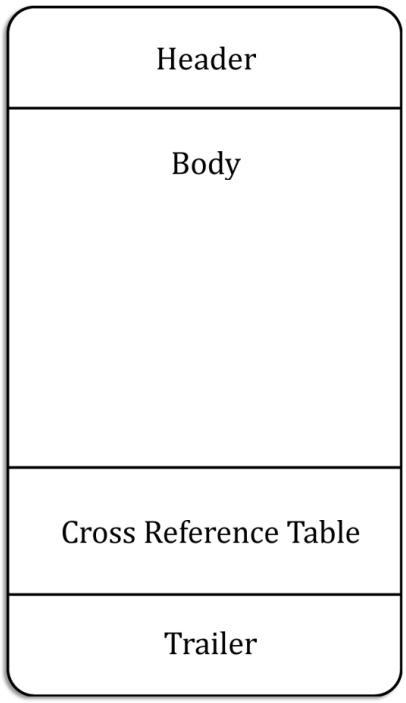

Abbildung 3.1: Grundlegender Aufbau einer PDF-Datei

[2] *http://en.wikipedia.org/wiki/List_of_PDF_software*
http://www.dmoz.org/Computers/Data_Formats/Document/Publishing/PDF/Software/

Eine PDF-Datei beginnt mit einem einzeiligen *Header*, der Aufschluss über die verwendete PDF-Version gibt, zum Beispiel „%PDF-1.7“. Danach folgt der *Body*, in dem einzelne Inhalte des PDF-Dokuments in sogenannten *Objects* oder *Object-Streams* (Sammlung von *Objects*) abgelegt werden. Diese können, um die Dateigröße zu reduzieren, je nach Typ per CCITT(Group 3 oder Group 4), JPEG, JPEG2000 und LZW (Lempel-Ziv-Welch) komprimiert sein. Damit später wieder auf die Inhalte zugegriffen werden kann, wird jedem *Object-Stream* eine Beschreibung (*Dictionary*) voran gestellt, die Informationen zum Inhalt, zum Komprimierungsverfahren und zur Länge der enthaltenen Daten gibt. Der *Cross Reference Table* stellt eine Art Inhaltsverzeichnis mit Positionen der einzelenen *Objects* innerhalb der Datei dar. Am Ende der Datei folgt ein *Trailer*, der die Position des *Cross Reference Table* angibt. Diese Art der Zuordnung ermöglicht es auf einzelne *Objects* zuzugreifen ohne den Inhalt des gesamten Dokuments auszuwerten. Auch die Bearbeitung eines PDF-Dokuments profitiert von dieser Struktur. So werden bei Änderung oder Erweiterung eines PDF-Dokuments neue *Objects* gefolgt von einem aktualisierten *Cross Reference Table* und *Trailer* am Ende der Datei angehängt.

Der PDF-Standard bietet noch eine ganze Reihe an weiteren Spezifikationen, die zusammen mit der oben beschriebenen Datenstruktur das Lesen von PDF-Dateien mit einem einfachen Text-Editor nur bedingt zulassen. Daher wird ein PDF-Dokument in der Regel als hierachisches Objekt-Modell betrachtet. Dabei steht an oberster Stelle ein Objekt mit dem Namen *Document catalog*, welches auf Objekte verweist, die das Dokument beschreiben. Ein solches Objekt-Modell vereinfacht besonders bei der Programmierung die Navigation innerhalb von PDF-Dokumenten.

3.1.2 Formulare innerhalb von PDF

PDF-Dokumente bieten zur Erstellung von Formularen verschiedene Komponenten. Dazu gehören Textfelder, Auswahllisten, Comboboxen, Radiobuttons, Checkboxen und Schaltflächen. Alle Komponenten können auf Benutzereingaben reagieren, indem der jeweiligen Komponente ein JavaScript hinzugefügt wird. Das ermöglicht zum Beispiel eine Validierung von Daten während der Eingabe. In Abbildung 3.2 werden exemplarisch Formular-Komponenten aufgezeigt.

Zur Speicherung oder Auswertung der Formulardaten bietet der PDF-Standard unterschiedliche Verfahren. Sie können direkt im Dokument gespeichert oder in eine separate FDF-Datei extrahiert werden. „FDF (Form Data Format) ist ein spezielles Textdateiformat

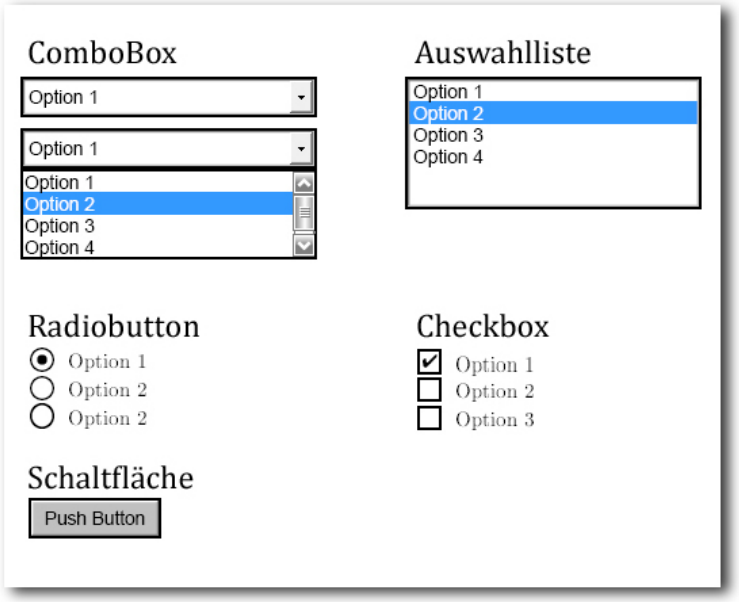

Abbildung 3.2: Übersicht der PDF-Formular-Komponenten

für Daten, die aus PDF-Formularfeldern exportiert wurden. FDF-Dateien sind wesentlich kleiner als PDF-Dateien, da sie nicht das gesamte Formular, sondern ausschließlich Felddaten enthalten“ [ado]. Eine dritte Methode zum Austausch der Formulardaten ist das direkte Senden an einen Web-Server. Dazu kann eine Schaltfläche zum Absenden in das Dokument integriert werden. In den Web-Server muss dafür eine Software zur Verarbeitung der FDF-Datei integriert sein.

3.2 LaTeX und PDF

LaTeX ist eine textbasierte Sprache zur Erstellung von Dokumenten mit *komplexem* Layout. Es wurde 1980 von Leslie Lamport entwickelt. Die Grundlage von LaTeX ist TeX, ein Textsystem, das 1977 von Donald Knuth an der Stanford Universität entwickelt wurde. TeX wurde durch LaTeX um funktionelle Hilfsprogramme zur Erstellung von Indizes, Literaturverzeichnissen, Querverweisen und verschiedenem mehr erweitert, so dass dem Anwender eine vereinfachte Benutzung zur Textgestaltung geboten wird. [Jür00] Ermöglicht wird dies durch das Einbinden von Makros und bereitgestellten Markropaketen, in denen komplexe Funktionen und Formatierungen zusammengefasst und per Aufruf eines einfachen Befehls

ausgeführt werden können. Aufgrund der programmierähnlichen Syntax konnte sich LaTeX nicht in allen Computerbenutzergruppen etablieren. Jedoch findet es starken Zuspruch im wissenschaftlichen und akademischen Bereich, da LaTeX eine sehr gute Darstellung von mathematischen Formeln ermöglicht.

Um ein Dokument mit Layout zu erzeugen, wird nach dem Kompilieren der TeX-Datei(en) mit LaTeX[3] eine dvi-Datei(DeVice Independend) erzeugt. Diese ist zunächst plattformunabhängig und kann später mit einem so genannten *Treiber* in eine andere Datenform umgewandelt werden. Um aus einer dvi-Datei ein PDF-Dokument zu erzeugen, kann unter Linux der Befehl „dvipdf“ in der Konsole aufgerufen werden. Dabei können Eigenschaften, die von PDF aber nicht von dvi unterstützt werden, verloren gehen. Um dieses Problem zu umgehen, gibt es einen speziellen Befehl „pdflatex“, der LaTeX-Dateien direkt ins PDF-Format kompiliert, wobei man auch vom PDF-Modus spricht. Auf diese Weise können Hyperlinks, Formularfelder und neuerdings auch multimediale Inhalte wie Audio, Video und 3D-Animationen in das PDF-Dokument geschrieben werden. Dafür werden spezielle Makropakete benötigt. Bei CTAN (the Comprehensive TeX Archive Network „http://tug.ctan.org“) wird eine Vielzahl von Makropaketen und Informationen kostenlos bereit gestellt. Über die integrierte Suchfunktion lassen sich bei CTAN sehr schnell Dokumentationen und Beispiele zu bestimmten Themen finden. Wird zum Beispiel nach „PDF Form“ gesucht, findet man das „AcroTeX education bundle“, auf das noch später in der Planung der PDF-Formularfunktion eingegangen wird.

[3] Das Kompilieren unter Linux erfolgt beispielsweise durch den Aufruf „latex Dateiname“

4 Anforderungen

Im Folgenden sollen die Anforderungen an die Abschlussarbeit bezüglich der Erweiterung von PDF-Formularen in LON-CAPA beschrieben werden. Dazu werden zunächst die bereits vorhandenen Benutzerschnittstellen genannt. Darauf folgt eine Beschreibung von Einsatzszenarien, die durch den Einsatz von PDF-Formularen ermöglicht werden könnten. Weiter wird noch der grundsätzliche Prozessablauf einer Aufgabenbearbeitung per PDF-Formular und die Anforderungen an das PDF-Formular selbst beschrieben.

4.1 Benutzerschnittstellen

Ein Kursteilnehmer hat in LON-CAPA die Möglichkeit Aufgaben über drei Verfahren zu beantworten. Diese sollen im folgenden noch einmal kurz genannt werden.

Web-Browser

Der Web-Browser stellt die wichtigste und umfangsreichste Benutzerschnittstelle für Aufgaben in LON-CAPA dar. Über Sie kann ein Kursteilnehmer über alle Funktionen der Aufgabenbearbeitung von LON-CAPA verfügen. Dazu gehören zum Beispiel spezielle Editoren für chemische Formeln oder interakive Funktionsplotter, aus denen eine bestimmte Funktion ausgewählt werden muss. Sendet ein Kursteilnehmer eine Antwort über den Web-Browser, bekommt er unmittelbar eine Rückmeldung über die Bewertung seiner Lösung.

Bubble-Sheet

Die Benutzerschnittstelle wird LON-CAPA intern als Target 'scantron' bezeichnet. Sie kann vom Kurs-Koordinator genutzt werden, um Aufgabensammlungen in einer Form zu erstellen, die es ermöglicht, Antworten auf einem Bubble-Sheet anzukreuzen. Das Bubble-Sheet wird nach der Bearbeitung eingelesen, so dass die angekreuzten Antworten der Kursteilnehmer in den LON-CAPA Kurs eingelesen werden können.

Clicker

Clicker ermöglicht es in einer Vorlesung Multiple-Choice Aufgaben an die Kursteilnehmer zu stellen, die direkt mittels kleinen Radiotransmittern beantwortet werden können.

4.2 Mögliche Einsatzszenarien

Durch das Erzeugen von Aufgaben oder Aufgabensammlungen in PDF-Formularen kann eine Reihe von neuen Bearbeitungsmöglichkeiten realisiert werden. Ein Szenario ist die offline-Bearbeitung auf Seiten der Kursteilnehmer. Die Druckfunktion bietet bereits die Möglickeit Aufgaben im PDF-Format auf den lokalen Rechner herunterzuladen. Die Lösungen zu den Aufgaben müssen allerdings nach der Bearbeitung in die Web-Schnittstelle eingetragen und einzeln gesendet werden. Ein PDF-Formular ermöglicht es dem Kursteilnehmer, die Lösung direkt nach der Bearbeitung in die integrierten Formularfelder einzutragen. Diese könnten nach dem Hochladen auf den LON-CAPA-Server automatisch zur Bewertung eingereicht werden, so dass der Kursteilnehmer die Bewertungen nach dem Hochladen per Web-Brower einsehen kann.

Optional könnte dieses Vorgehen auch mit einer E-Mail-Funktion ergänzt werden, so dass der Kurs-Koordinator Aufgabensammlungen als Hausarbeit an die Kursteilnehmer per E-Mail verschickt. Dabei hat der Kurs-Koordinator die Möglichkeit, Termine und Umfang der Aufgabensammlung im Kurs zu hinterlegen, so dass die Hausarbeiten automatisch generiert und gesendet werden. Die Kursteilnehmer könnten ihre beantworteten PDF-Formulare an eine bestimmte E-Mail-Adresse zurücksenden. LON-CAPA liest die empfangenen PDF-Formulare aus, bewertet die Antworten und schickt im Anschluss eine bewertete Version des PDF-Formulars zurück. Je nach Anzahl von erlaubten Versuchen kann der Kursteilnehmer diesen Vorgang wiederholen. Bei dieser Funktion könnte LON-CAPA komplett in den Hintergrund rücken, so dass sich die Kursteilnehmer nicht in die Funktionen des Kurssystems in LON-CAPA einarbeiten müssten.

Vorstellbar wäre auch, dass PDF-Formulare in Lehrbüchern als Übungsblätter auf einer CD-ROM beiliegen. Diese würden wiederum an eine E-Mail-Adresse gesandt und korrigiert zurück gesendet.

4.3 Prozessablauf

Als nächstes soll der Prozessablauf bei der Bearbeitung durch PDF-Formulare dargestellt werden. In den Schritten 1-6 wird der Prozessablauf der typischen Kursteilnehmer-Interaktion per PDF-Formular beschrieben.

1. Das PDF-Formular wird serverseitig erstellt. Denkbar wäre eine zusätzliche Formularfunktion zur bereits vorhandenen Druckfunktion.

2. Das PDF-Formular wird auf den Rechner des Kursteilnehmers heruntergeladen (ggf. per E-Mail verschickt).

3. Der Kursteilnehmer bearbeitet die Aufgaben offline und trägt die Lösungen in das PDF-Formular ein.

4. Anschließend lädt der Kursteilnehmer das PDF-Formular auf den LON-CAPA Server (ggf. per E-Mail verschickt).

5. Die Antworten werden von LON-CAPA aus dem PDF-Formular gelesen und bewertet. Wie die Lösungen LON-CAPA intern bewertet werden, kann eventuell im Target 'scantron' oder 'clicker' eingesehen werden.

6. Eventuell wird eine bewertete Version des PDF-Formulars an den Kursteilnehmer zurückgesendet. Das wäre im Falle des Szenarios mit der in Lehrbüchern beiliegenden CD-ROM besonders sinnvoll.

4.4 Anforderungen an das PDF-Formular

Die zur Erstellung der PDF-Formulare verwendete Software und die verwendeten Software-Module müssen wie alle Software-Komponenten in LON-CAPA frei verwendet werden dürfen. Die PDF-Formulare dürfen keine Informationen über die korrekte Lösung einer Aufgabe enthalten. Dieses ist zum einen aus Sicherheitsgründen notwendig und zum anderem wäre es dem Kursteilnehmer möglich mit Hilfe geneigneter Software diese Informationen aus dem PDF-Formular auszulesen. Die PDF-Formulare müssen Daten enthalten, die eine eindeutige Zuordnung zum Kursteilnehmer gewährleisten.

4.5 Abgrenzung der Umsetzung in der Bachelorarbeit

Im Rahmen der vorliegenden Bachelor of Engineering Arbeit sollen mindestens Lösungen für die im Prozessablauf genannten Punkte 1, 2 und 5 gefunden werden. Die für die anderen Schritte erforderlichen Erweiterungen würden den Umfang dieser Bachelorarbeit überziehen.

5 Analyse der Aufgabendarstellung in LON-CAPA

In diesem Kapitel wird beschrieben, wie von LON-CAPA Aufgaben für verschiedene Ausgabemöglichkeiten (Targets) generiert werden. Zuvor soll jedoch erst einmal die Funktion des Apache-Web-Servers mit der Perl-API genauerwird eine Übersicht der betrachtet werden, da ohne ein Verständnis von der serverseitigen Verarbeitung nur schwer nachvollzogen werden kann, wie die verschiedenen Perl-Module zusammen arbeiten.

5.1 Apache-Perl-API

Um die Perl-API im Apache-Web-Server zu nutzen, muss ein Apache-Modul mit dem Namen „mod_perl“ in den Apache-Server integriert werden. „mod_perl“ ermöglicht es, Perl-Skripte und -Module direkt im Server auszuführen, so dass kein zusätzlicher Interpreter für das Ausführen von CGI-Skripten aufgerufen werden muss. Dadurch spart man viel Rechenzeit, so dass der Web-Server wesentlich effizienter läuft.

„mod_perl is more than CGI scripting on steroids. It is a whole new way to create dynamic content by utilizing the full power of the Apache web server to create stateful sessions, customized user authentication systems, smart proxies and much more. Yet, magically, your old CGI scripts will continue to work and work very fast indeed. With mod_perl you give up nothing and gain so much!“–Lincoln Stein, http://perl.apache.org

„mod_perl“ bietet die Möglichkeit, in die interne Verarbeitungsstruktur des Apache-Web-Servers einzugreifen. Dieses wird in [SM99] als *Request-Loop* bezeichnet. Eine schematische Darstellung zeigt die Abbildung 5.1. Um die Aufgabenerzeugung in LON-CAPA nachzuvollziehen, wird ein Verständnis der *URI-Translation* benötigt. Die anderen Phasen des *Request-Loops* werden daher an dieser Stelle nicht betrachtet.

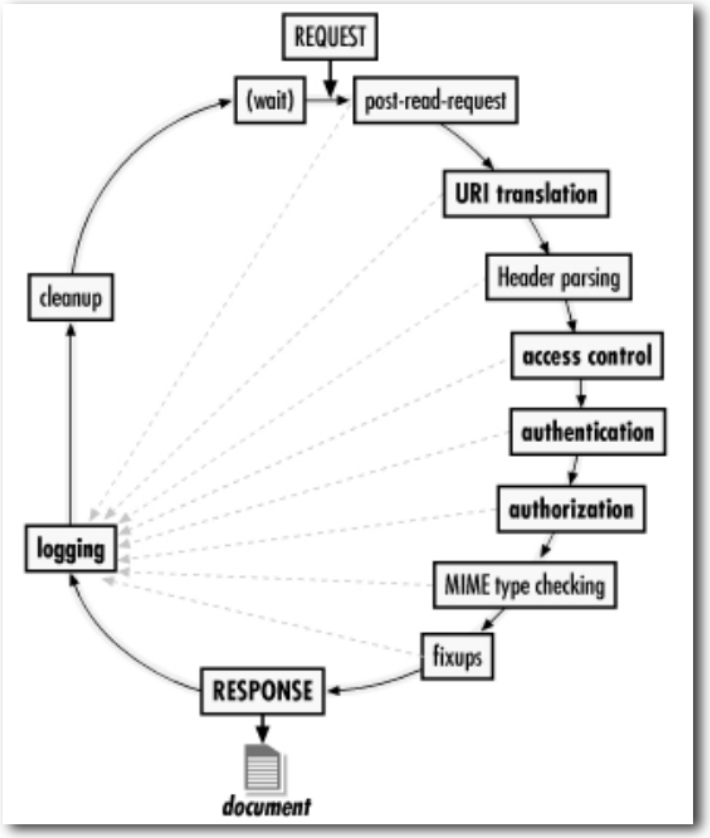

Abbildung 5.1: Schematische Darstellung des *Request-Loops* [SM99]

5.1.1 URI-Translation

Die *URI-Translation* ermöglicht es, eine vom Browser angeforderte Internetadresse URL (Uniform Resource Locator) auszuwerten. Ohne diese Auswertung wird der Apache-Server aufgefordert, vom Dateisystem eine Datei, die durch die URL beschrieben wird, zu verarbeiten und an den Browser zurückzusenden. Dabei kann es sich um eine statische HTML-Seite oder ein serverseitiges Skript in Perl, PHP, Ruby, etc. handeln. Um Einfluss auf die *URI-Translation* und andere Phasen des *Request-Loops* zu nehmen, werden in der Apache-Konfigurationsdatei Direktiven eingetragen, die mittels regulärer Ausdrücke auf bestimmte URLs geprüft werden. Beim LON-CAPA-Server sind diese Direktiven in */etc/httpd/conf/loncapa_ apache.conf* abgelegt. Listing 5.1 zeigt einen Auszug aus dieser Datei.

Listing 5.1: Auszug aus der Apache-Konfiguration

```
1  [...]
2  <LocationMatch "^/+(uploaded|res|\~).*\.(xml|html|htm|xhtml|xhtm|sty)$">
3  SetHandler perl-script
4  PerlHandler Apache::londatecheck
5  PerlHandler Apache::lonxml
6  </LocationMatch>
7
8  <LocationMatch "^/+(res|\~).*\.(task|problem|exam|quiz|assess|survey|form|
       library)$">
9  SetHandler perl-script
10 PerlHandler Apache::lonhomework
11 </LocationMatch>
12
13 <LocationMatch "^/adm/wrapper/">
14 AuthType LONCAPA
15 Require valid-user
16 PerlAuthzHandler       Apache::lonacc
17 PerlHandler Apache::londatecheck
18 SetHandler perl-script
19 PerlHandler Apache::lonwrapper
20 ErrorDocument      403 /adm/login
21 ErrorDocument      500 /adm/errorhandler
22 </LocationMatch>
23 [...]
```

In den Zeilen 8 bis 11 steht die Direktive, die für die Verarbeitung der Aufgaben zuständig ist. <LocationMatch „Regulärer Ausdruck“> beschreibt eine *URI-Translation*. Die übermittelte URL wird dazu mit dem angegebenen Regulären Ausdruck verglichen. Wird eine Übereinstimmung festgestellt, wird die Anfrage des Web-Browsers an die in der Konfiguration angegebenen „PerlHandler“ weitergereicht. Unter einem „PerlHandler“ versteht man im Zusammenhang mit mod_perl ein Perl-Modul, das eine Methode *sub handler()* implementiert hat. An diese Methode wird später die Anfrage des Browsers in Form eines Request-Objekts übergeben. Zuvor werden jedoch die anderen Phasen des *Request-Loops* durchlaufen. Für diese Phasen können, wie beispielsweise in Zeile 16, spezielle Perl-Module als Handler angegeben werden. Für eine genauere Beschreibung der verschiedenen Phasen soll auf [SM99] verwiesen werden.

5.2 Umwandeln der Aufgaben

Um den weiteren Verlauf der Aufgabendarstellung zu beschreiben, wird an dieser Stelle davon ausgegangen, dass sich ein Benutzer mit der Rolle Student in einen LON-CAPA Kurs eingeloggt hat. So können im Inhaltsverzeichnis des Kurses die Aufgaben über Hyperlinks aufgerufen werden. Die Adresse dieser Hyperlinks entspricht folgendem Muster: */res/domainname/benutzername/(Name der Aufgabe).problem&(Parameter)*. Die Parameter beschreiben die personifizierte Version einer Aufgabe und werden an späterer Stelle noch genauer betrachtet. Durch die in der loncapa_apache.conf angegebene Direktive (siehe Listing 5.1 Zeile 8 bis 11) wird die Anfrage des Browsers im Perl-Modul *lonhomework.pm* bearbeitet. Dort werden die Aufgaben gemäß ihres Typs für die Ausgabe vorbereitet und an den Browser zurückgesendet. Abbildung 5.2 aus der LON-CAPA-Entwickler-Dokumentation beschreibt die Vorgehensweise des Handlers von *lonhomework.pm*.

Zu Beginn erfolgt eine Anfrage vom Web-Browser, der eine Aufgabe aufruft. Daraufhin wird das Target, in das die Aufgabe dargestellt werden soll, bestimmt. Anschließend werden die zur Aufgabe gehörigen Daten eingelesen und global zur Verfügung gestellt. Wird dabei festgestellt, dass eine Antwort auf eine Aufgabe im Browser gesendet wurde, wird eine Bewertung durchgeführt und gespeichert. Dann wird die Ausgabe gemäß des zuvor bestimmten Targets generiert und an den Browser gesendet. Zuletzt wird geprüft, ob in einem weiteren Target dargestellt werden soll. Das ist zum Beispiel der Fall, wenn eine Aufgabe per Web-Browser beantwortet wurde und anschließend die bewertete Rückmeldung gesendet wird.

Da XML als Beschreibungssprache für die Aufgaben genutzt wird, müssen die Aufgaben für die verschiedenen Targets angepasst werden. Für die Ausgabe im Web-Browser wird dazu in HTML und für die Druckausgabe über LaTeX in PDF gewandelt. Dazu werden zunächst die aufgabenspezifischen Daten gelesen und global verfügbar gemacht. Der Code zu den verschiedenen Targets wird für jeden Aufgabentyp in einem zugehörigen Perl-Modul erzeugt. Die Zuständigkeiten der Module sind in der LON-CAPA-Entwicklerdokumentation beschrieben. Sie sind gemäß ihres zugehörigen Aufgabentyps benannt, so dass sie auch ohne einen Blick in die Dokumentation gefunden werden können. Zum Beispiel werden die Bestandteile einer Radiobutton-Aufgabe in *radiobuttonresponse.pm* und die von Sortierungsaufgaben in *rankresponse.pm* erzeugt. Die von LON-CAPA verwendeten Module liegen auf dem Server unter */home/httpd/lib/perl/Apache/*.

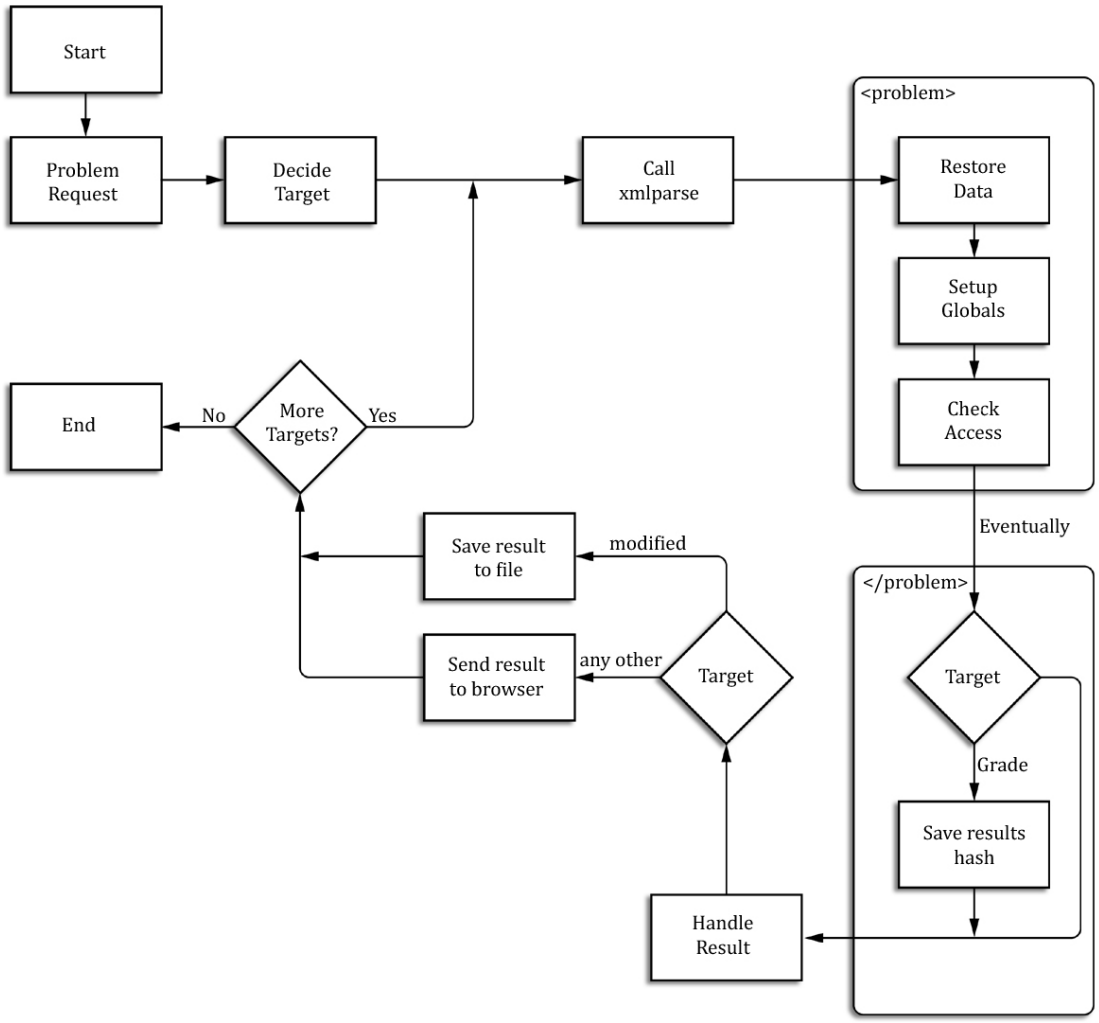

Abbildung 5.2: Grundsätzlicher Ablauf beim Aufruf des *lonhomework*-Handlers [AKH02]

Wie in den relevanten technischen Details in Kapitel 2.2 beschrieben, werden die zu den intern verwendeten XML-Elementen zugeordneten Module in *lonxml.pm* registriert. Das folgende Listing 5.2 soll dieses noch einmal an Hand einer vereinfachten XML-Beschreibung einer Radiobutton-Aufgabe verdeutlichen. Dabei wird aus Platzgründen auf die Attribute der einzelnen XML-Elemente sowie auf den Aufgabentext verzichtet.

Listing 5.2: Beispiel zu Start- und End-Methoden

```
<problem>

  <radiobuttonresponse> -> start_radiobuttonresponse();

    <foilgroup> -> start_foilgroup();

      <foil>  -> start_foil();
      </foil>  -> end_foil();

      <foil>  -> start_foil();
      </foil>  -> end_foil();

      ...weitere foils

    </foilgroup> -> end_foilgroup();

  </radiobuttonresponse> -> end_radiobuttonresponse();

</problem>
```

Das erste XML-Element <problem> beschreibt, dass eine Aufgabe definiert wird. Die zugehörige Start-Methode *start_problem()* ist in *structuretags.pm* definiert und erstellt beim Aufruf ein öffnendes HTML-Form-Element und versteckte HTML-Formularfelder, in die die Daten zur Zuordnung der Aufgabe geschrieben werden. Danach folgt die Spezifikation des Aufgabentyps mit <radiobuttonresponse>. Diese führt dazu, dass beim Generieren der Ausgabe die Methode *start_radiobuttonresponse* ausgeführt wird. In dieser Methode werden die im Anschluss folgenden Elemente <foilgroup> und <foil> in *lonxml.pm* registriert, so dass die Methoden für deren Ausführung aus *radiobuttonresponse.pm* verwendet werden. Als nächstes folgt ein schließendes XML-Element </foil>, durch das die *end_foil()*-Methode aufgerufen wird. Mit den übrigen schließenden XML-Elementen verhält es sich ebenso. Jede dieser Methoden bekommt Parameter übergeben, worin unter anderem der Name des Targets enthalten ist, in welchem die Aufgabe dargestellt wer-

den soll. Dazu werden größtenteils stark verschachtelte if-Anweisungen verwendet, die den übergebenen Targetnamen mit 'tex', 'web', 'analyse', etc. vergleichen. Dadurch wird der zum Target passende Code in eine Variable *$result* geschrieben. Diese wird am Ende der Methode zurückgegeben. Die Rückgabewerte aller Methoden, die durch die XML-Elemente aufgerufen werden, ergeben den Code für die Darstellung der Aufgabe. Im Falle des Aufrufs einer Aufgabe mit HTML-Schaltflächen im Web-Browser werden die meisten Code-Teile in der Methode *displayfoils* oder *displayallfoils* generiert. Diese werden in der Methode *end_foilgroup()* aufgrufen. Wie der Code für die Ausgabe in HTML und PDF aussieht wird in den nächsten zwei Abschnitten beschrieben.

5.3 Target - web

Die Aufgaben im Target 'web' werden in HTML dargestellt. Der genaue Aufbau der HTML-Struktur kann sehr gut mit einem Web-Browser mit Web-Entwicklungs-Plug-In analysiert werden. Als besondere Empfehlung soll hier kurz der Mozilla Firefox 3.0 mit „firebug"-Plug-In genannt werden, mit dessen Hilfe sehr einfach durch den HTML-Code einer Web-Seite navigiert werden kann. Im Listing 5.3 wird zunächst der HTML-Code für eine Radiobutton-Aufgabe aufgezeigt. Dabei wird aus Platzgründen der umgebene HTML-Code nicht aufgeführt und die Leerzeichen und Zeilenumbrüche angepasst. In Abbildung 5.3 wird ein Screenshot des HTML-Formulars zusammen mit der Aufgabenstellung im Web-Browser dargestellt.

Listing 5.3: Radiobutton-Aufgabe im Target 'web'

```
1  <form name="lonhomework"
2        enctype="multipart/form-data" method="post"
3        action="/res/lib/templates/simpleproblem.problem">
4    <input type="hidden" name="symb"
5           value="uploaded/fhwfdev/6o32055631bf647a7fhwfldev1/
                default_1212826295.sequence___4___lib/templates/simpleproblem.
                problem" />
6    <input type="hidden" name="submitted" value="yes" />
7    <a name="radio" />
8    Wie lautet das Ergebnis folgender Aufgabe im binären Zahlensystem?
9    1 + 1 = ?<br /><br />
10   <label>
11     <input type="radio"
12            onchange="javascript:setSubmittedPart('radio');"
```

```
13              name="HWVAL_radiosegment"
14              value="0" checked="on" />
15      0
16    </label><br />
17    <label>
18      <input type="radio"
19              onchange="javascript:setSubmittedPart('radio');"
20              name="HWVAL_radiosegment"
21              value="1"  />
22      10
23    </label><br />
24    <label>
25      <input type="radio"
26              onchange="javascript:setSubmittedPart('radio');"
27              name="HWVAL_radiosegment"
28              value="2"  />
29      110
30    </label><br />
31    <label>
32      <input type="radio"
33              onchange="javascript:setSubmittedPart('radio');"
34              name="HWVAL_radiosegment"
35              value="3"  />
36      2
37    </label><br />
38    <table><tr><td>
39          <input onmouseup="javascript:setSubmittedPart('radio')"
40                  onsubmit="javascript:setSubmittedPart('radio')"
41                  type="submit" name="submit_radio"
42                  value="Sumbit Answer" />
43        </td>
44        <td/>
45      </tr>
46    </table>
47  </form>
```

Anhand eines Vergleichs des HTML-Codes mit dem Code in den einzelnen Start- und End-Methoden in *radiobuttonresponse.pm*, kann die Generierung nun relativ einfach nachvollzogen werden. Dieser Vorgang lässt sich sehr gut auf die Darstellung von anderen Aufgabentypen übertragen.

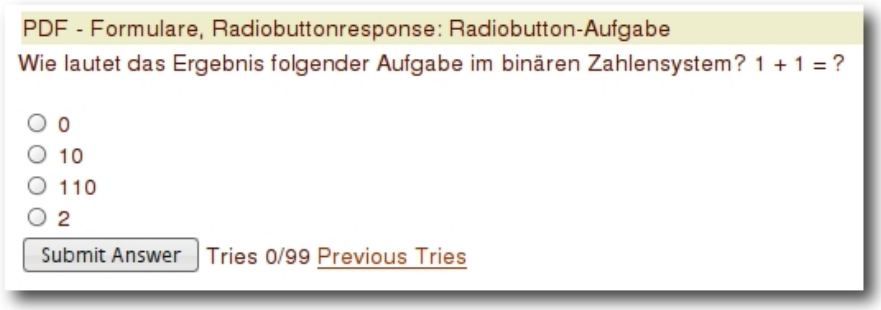

Abbildung 5.3: Radiobutton-Aufgabe im Web-Browser

5.4 Target - tex

Um eine Darstellung im Target 'tex' zu erzeugen, muss eine Aufgabe über die Druckfunktion erstellt werden. Dieses Vorgehen soll ebenfalls an Hand eines Beispiels analysiert werden. Dazu soll die bereits zuvor betrachtete Radiobutton-Aufgabe über das Druckmenü gedruckt werden. Dabei wird die Aufgabe über das Kurs-Inhaltsverzeichnis ausgewählt und das Druckmenü über den Link „Drucken“ aufgerufen. Der Link gibt dabei keine URL an, sondern führt eine JavaScript-Funktion *javascript:gopost('/adm/printout',currentURL);* aus (Listing 5.4).

Listing 5.4: Die JavaScript-Funktion *gopost()*

```
function gopost(url,postdata) {
   if (url!='') {
      this.document.server.action=url;
      this.document.server.postdata.value=postdata;
      this.document.server.command.value='';
      this.document.server.url.value='';
      this.document.server.symb.value='';
      this.document.server.submit();
   }
}
```

Diese Funktion bewirkt, dass die Werte der nicht-sichtbaren Felder und die Zieladresse eines HTML-Formulars namens „server“ gesetzt werden. Am Ende der Funktion wird das Formular abgeschickt. Das Formularfeld besitzt nach den Änderungen durch das JavaScript die im Listing 5.5 aufgeführten Werte.

Listing 5.5: Geändertes HTML-Formular

```
1 <form name="server" action="/adm/printout" method="post" target="_top">
2   <input type="hidden" name="postdata" value="/res/lib/templates/
      simpleproblem.problem" />
3   <input type="hidden" name="command" value="" />
4   <input type="hidden" name="url" value="" />
5   <input type="hidden" name="symb" value="" />
6 </form>
```

Aus den Attributen des HTML-Form-Elements geht hervor, dass es durch den *submit()*-Aufruf am Ende des JavaScripts per *POST-Request* an die URL */adm/printout* gesendet wird. Durch die zugehörige Direktive in der LON-CAPA-Konfigurationsdatei *loncapa_apache.conf* geht hervor, dass die Handler-Methode im Modul *lonprintout.pm* für die Weiterverarbeitung zuständig ist. Der Ablauf dieses Handlers ist durch viele methodeninterne Aufrufe um einiges komplexer als das Darstellen einer Aufgabe im Web-Browser, da zum einen für jede Rolle ein unterschiedliches Druckmenü erstellt wird und zum anderem Informationen zu den verfügbaren Aufgaben gesammelt werden. Um nicht in den Feinheiten der LON-CAPA-internen Benutzer- und Inhaltsverwaltung zu versinken, soll im Folgenden nur beschrieben werden, wie die Aufgaben im Target 'tex' gewandelt werden. Im folgenden Szenario wird davon ausgegangen, dass im Druckmenü nur der zuletzt betrachtete Inhalt gedruckt wird. Das soll im Weiteren die bereits oben beschriebene Radiobutton-Aufgabe sein.

Beim erstem Betreten des Druckmenüs wird ein HTML-Formular mit verschiedenen Druckoptionen angezeigt. Der vereinfachte HTML-Code (Listing 5.6) des Formulars zeigt an, das kein *action*-Attribut im <form>-Element vorhanden ist. Das bedeutet, dass das Formular beim Absenden an die aktuelle Adresse im Web-Browser geschickt wird. Daher ist das Modul *lonprintout.pm* weiterhin für die Verarbeitung zuständig.

Listing 5.6: Vereinfachter HTML-Code des Druckmenüs in Rolle Student

```
<form method="post" name="helpform">
  <input type="hidden" value="START" name="CURRENT_STATE"/>
  <input type="hidden" value="fb00447350623bd5e1cd006ff24ced44" name="TOKEN"/>
  <input type="hidden" value="" name="RETURN_PAGE"/>

  <input type="radio" id="id0" checked="checked"
         value="current_document" name="PRINT_TYPE_forminput"/>
  <input type="radio" id="id1" value="map_problems"
         name="PRINT_TYPE_forminput"/>
  <input type="radio" id="id2" value="map_problems_pages"
         name="PRINT_TYPE_forminput"/>
  <input type="radio" id="id3" value="select_sequences"
         name="PRINT_TYPE_forminput"/>

  <input type="radio" value="L" name="FORMAT.layout"/>
  <input type="radio" checked="1" value="P" name="FORMAT.layout"/>
  <select name="FORMAT.cols">
    <option value="1">1</option>
    <option selected="" value="2">2</option>
  </select>
  <select name="FORMAT.paper">
    <option value="letter" selected="">letter [8 1/2x11 in]</option>
    <option value="legal">legal [8 1/2x14 in]</option>
    <option value="a4">a4 [210x297 mm]</option>
  </select>
  <input type="button" onclick="history.go(-1)"
         value="<- Zurück" name="back"/>
  <input type="submit" value="Weiter ->" name="SUBMIT"/>
</form>
```

An Hand der übermittelten Formulardaten kann LON-CAPA dann entscheiden, welche Aufgaben und Inhalte gedruckt werden sollen. Bei der Wahl von mehreren Aufgaben und Inhalten wird ein zweites Menü mit einer Übersicht der verfügbaren Aufgaben und Inhalte der Ordner im Kurs ausgegeben. Die TEX-Version der Aufgaben wird anschließend mit einem serverinteren Request, ähnlich der Darstellung im Web-Browser, mit Hilfe von *lonxml.pm* gewandelt. In die für das Target verantwortliche Variable *$target* wird dabei ein String 'tex' gesetzt, so dass die Rückgaben in den Start- und End-Methoden den Code für eine LATEX-Datei zurückgeben. Im Listing 5.7 wird ein Teil des zuständigen Codes aus *radiobuttonresponse.pm* dargestellt.

Listing 5.7: Auszug aus *radiobuttonresponse.pm*

```
[...]
  if ($target ne 'tex') {
    # Ausgabe, wenn $target ungleich dem String 'tex' ist
    $result.= '<label>';
    $result.= "<input type=\"radio\"
                onchange=\"javascript:setSubmittedPart('$part');\"
                name=\"HWVAL_$Apache::inputtags::response['-1']\"
                value=\"$temp\" ";
    if (defined($lastresponse{$name})) {
      $result .= 'checked="on"';
    }
    $result .= ' />'.$Apache::response::foilgroup{$name.'.text'};
    $result .= "</label>";

  } else {
    # Ausgabe, wenn $target gleich dem String 'tex' ist
    if ($Apache::lonhomework::type eq 'exam') {
      # Hier wird eine besondere Ausgabe für
      # BobbleSheet Aufgaben erzeugt
      [...]
        } else {
      # Hier die normale Druckausgabe
      $result .= '\vspace*{-2 mm}\item ';
      $reslut .= $Apache::response::foilgroup{$name.'.text'};
    }
  }
[...]
```

Ein weiterer Unterschied zwischen Target 'web' und 'tex' ist, dass die Darstellung im Web-Browser in ein HTML-Gerüst eingebettet wird. Daher kann der generierte HTML-Code für das Target 'web' ohne Probleme hinter den Code der LON-CAPA-Navigationsleiste angefügt werden. Bei LaTeX müssen, bevor die Aufgaben eingefügt werden, in einer Präambel das Dokumentformat und die verwendendeten Makropakete angegeben werden. Die Präambel wird im Modul *londefdef.pm* in der Methode *start_ html()* (Listing 5.8) definiert.

Listing 5.8: Auszug aus *londefdef.pm*

```
sub start_html {
    my ($target,$token) = @_;
    my $currentstring = '';

    if ($target eq 'web' || $target eq 'edit' || $target eq 'webgrade' ) {
        # start_body() takes care of emitting the <html>
    } elsif ($target eq 'tex') {
        # Ab hier wird die Präambel definiert
        $currentstring .=
            '\documentclass[letterpaper,twoside]{article}\raggedbottom';
        if (($env{'form.latex_type'}=~'batchmode') ||
            (!$env{'request.role.adv'})) {$currentstring .='\batchmode';}
        $currentstring .= '\newcommand{\keephidden}[1]{}'.
                  '\renewcommand{\deg}{$^{\circ}$}'.
                  '\usepackage{multirow}'.
                  '\usepackage{longtable}'.
                  '\usepackage{textcomp}'.
                  '\usepackage{makeidx}'.
                  '\usepackage[dvips]{graphicx}'.
                  '\usepackage{wrapfig}'.
                  '\usepackage{picins}'.
                  '\usepackage[T1]{fontenc}'."\n".
                  '\usepackage[postscript]{ucs}'."\n".
                  '\usepackage[utf8x]{inputenc}'."\n".
                  '\usepackage{pifont}' ."\n".
                  '\usepackage{latexsym}'."\n".
                  '\usepackage{epsfig}'.
                  '\usepackage{calc}'.
                  '\usepackage{amsmath}'.
                  '\usepackage{amssymb}'.
                  '\usepackage{amsfonts}'.
                  '\usepackage{amsthm}'.
                  '\usepackage{amscd}'.
                  [...] Hier werden noch Markos überschrieben
                        und neue Umgebungen definiert
    }
    return $currentstring;
}
```

Nachdem die Präambel erzeugt ist, werden die ausgewählten Aufgaben jeweils mit einem serverinternen Request generiert und angehängt. Für den Request wird die Methode *ssi_with_retries()* verwendet, die eine Anfrage vom Web-Browser simuliert. Die Anwort vom Server wird zurückgegeben und kann in einer Variablen gespeichert werden. Wurden alle Aufgaben erzeugt, wird ein Copyright-Vermerk hinzugefügt und mit *end{document}* das Ende des TEX-Codes angegeben. Anschließend wird eine TEX-Datei mit dem erzeugten TEX-Code im Druckordner(*/home/httpd/prtspool/*) gespeichert. Der Dateiname setzt sich dabei wie folgt zusammen:

*Benutzername*_*Domainname*_printout_*Zeitstempel*_*PID*_(*eindeutige Nummer*).tex

Konkretes Beispiel auf dem Testserver:
/home/httpd/prtspool/tonken_fhwfdev_printout_1214392758_29212_1.tex

Nachdem die Datei gespeichert wurde, wird über das Senden eines <meta>-HTML-Elements an den Web-Browser bewirkt, dass das CGI-Skript */cgi-bin/printout.pl?(Parameter)* aufgerufen wird. Im folgenden Listing 5.9 ist die Ausgabe des <meta>-HTML-Elements dargestellt. Die Variable *$identifier* enthält dabei die Parameter des oben genannten Aufrufs, an Hand dessen *printout.pl* erkennen kann welche TEX-Datei zuvor gespeichert wurde. Im Falle eines Fehlers bei der Verarbeitung des TEX-Codes wird ein Fehler-Protokoll im Web-Browser angezeigt und eine Nachricht an den Kurs-Koordinator gesendet.

Listing 5.9: Weiterleitung an */cgi-bin/printout.pl*

```
$r->print(<<FINALEND);
   <br />
   <meta http-equiv="Refresh"
         content="0; url=/cgi-bin/printout.pl?$identifier" />
   <a href="/cgi-bin/printout.pl?$identifier">$continue_text</a>
   $end_page #
FINALEND
```

In */cgi-bin/printout.pl* werden zunächst die Berechtigungen des Benutzers geprüft. Wird der Benutzer zugelassen, wird an Hand des Parameters in der URL der Pfad zur zuvor erstellten TEX-Datei wiederhergestellt. Das weitere Vorgehen beim Erzeugen der PDF soll an dieser Stelle etwas vereinfacht beschrieben werden, da je nach Rolle des Benutzers eine Vielzahl von verschiedenen Ausgaben an den Web-Browser gesendet werden. Zudem wird noch geprüft, ob Grafiken in die PDF-Datei eingebunden werden sollen. Diese müssen vor der Verarbeitung der TEX-Datei in EPS(Encapsulated PostScript)-Grafiken umgewandelt

werden. Davon ausgehend, dass alle Vorgänge ohne Fehler durchgeführt werden, erzeugt *printout.pl* per **latex** eine DVI-Datei. Diese wird per **dvips** in ein PostScript gewandelt. Im Anschluss werden an dem PostSkript einige hier nicht näher beschriebene Anpassungen vorgenommen. Im letzten Schritt wird das angepasste PostScript mit **ps2pdf** in eine PDF-Datei gewandelt. Wenn all diese Umwandlungen berechnet und gespeichert wurden, wird von *printout.pl* ein Link zur PDF-Datei an den Web-Browser gesendet. Im Druckordner */home/httpd/prtspool* befinden sich neben der PDF-Datei auch die Dateien der Zwischenschritte. Listing 5.10 zeigt die Liste der erzeugten Dateien. Diese werden im einzelnen in der Tabelle 5.1 kurz beschrieben werden.

Listing 5.10: Erstellte Dateien im Druckordner

```
username_domain_printout_1214398775_2246_1.aux
username_domain_printout_1214398775_2246_1.dvi
username_domain_printout_1214398775_2246_1.html
username_domain_printout_1214398775_2246_1.log
username_domain_printout_1214398775_2246_1.pdf
username_domain_printout_1214398775_2246_1.ps
username_domain_printout_1214398775_2246_1temporar.ps
username_domain_printout_1214398775_2246_1.tex
```

Tabelle 5.1: Übersicht der beim Druckvorgang erzeugten Dateien

.aux	Die aux-Datei entsteht in der ersten Verarbeitungsphase von TEX. In ihr werden die Verweise des gesamten Dokuments gespeichert, damit TEX in der nächsten Phase auf Stellen verlinken kann, die erst später im Dokument erzeugt werden.
.dvi	Die dvi-Datei ist die Darstellung des Dokuments in einem plattformunabhängigen Format.
.html	Die html-Datei enthält die .log-Datei, die zur Darstellung im Web-Browser von einem HTML-Gerüst umgeben ist.
.log	Die log-Datei enthält das Protokoll, das während der Verarbeitung der tex-Datei erstellt wird. Warnung, Fehler und andere hilfreiche Informationen können in ihr eingesehen werden. Besonders bei Problemen kann die log-Datei wichtige Hinweise geben.
.pdf	Die pdf-Datei ist das endgültige Resultat des Druckvorgangs. Diese Datei wird nach der erfolgreichen Verarbeitung im Web-Browser als direkter Download verlinkt.
.ps	Das PostScript, das direkt aus der dvi-Datei erzeugt wurde.
temporar.ps	Das angepasste PostScript aus dem die pdf-Datei erzeugt wird.
.tex	Die tex-Datei enthält den von LON-CAPA generierten TEX-Code.

5.5 Rücksendung von Aufgaben

Nachdem in den vorigen Abschnitten die Generierung von Aufgabenausgaben im Target „web“ und „tex“ beschrieben wurde, soll hier die Rücksendung der Antworten zu den Aufgaben per Web-Browser analysiert werden. Dabei soll erneut an der zuvor beschriebenen Radiobutton-Aufgabe festgehalten werden. Im Anschluss soll die Übertragbarkeit auf andere Aufgabetypen beschrieben werden.

5.5.1 Absenden einer Antwort

Im Listing 5.3 auf Seite 25 wurde das HTML-Formular für die Radiobutton-Aufgabe bereits abgebildet. Wie dort zu erkennen ist, wird das HTML-Formular an die Adresse */res/-lib/templates/simpleproblem.problem* gesendet. Dazu sollte noch erwähnt werden, dass es

sich bei dieser Radiobutton-Aufgabe um eine vom Kurs-Koordinator erstellte Aufgabe handelt. In der Beschreibung zu den verschiedenen Rollen wurde erwähnt, das der Kurs-Koordinator einfache Aufgaben erstellen kann. Die Vorlagen dafür liegen auf jeder Domain unter textit/home/httpd/res/lib/templates/. Bei Aufgaben, die vom LON-CAPA-Network in den Kurs importiert wurden, steht in dem HTML-Formular eine Adresse in der Form */res/Domainname/Autorname/Aufgabenname.problem*. Der hier betrachtete Vorgang ist aber in beiden Fällen derselbe.

Zunächst soll das Versenden des HTML-Formulars betrachtet werden. Anschließend wird beschrieben, welche Funktion die einzelen Formularfelder darstellen. Da das Formular per *POST-Request* versendet wird, kann zum Feststellen der zu übertragenden Daten der HTML-Code angeschaut werden oder die Kommunikation während des Sendens der Antwort an den Server mit einem Netzwerkanalyse-Werzeug aufgezeichnet werden. Der Vorteil bei der Aufzeichnung ist, dass alle Daten die vom Web-Browser gesendet werden inklusive HTTP-Header aufgezeichnet werden. Zwei konstenlose Software-Lösungen für diesen Vogang sind Wireshark von Gerald Combs und das bereits genannte Firebug-Plug-In für den Firefox 3.0. Listing 5.11 zeigt den mit Wireshark aufgezeichneten HTTP-Stream.

Listing 5.11: Der HTTP-Stream beim Einsenden einer Antwort

```
1  POST /res/fhwfdev/tonken/Radiobutton.problem HTTP/1.1
2  Host: polya.informatik.fh-wolfenbuettel.de
3  User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9) Gecko/2008061015
       Firefox/3.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
6  Accept-Encoding: gzip,deflate
7  Accept-Charset: UTF-8,*
8  Keep-Alive: 300
9  Connection: keep-alive
10 Referer: http://polya.informatik.fh-wolfenbuettel.de/res/fhwfdev/tonken/
       Radiobutton.problem?symb=uploaded%2ffhwfdev%2f6o32055631bf647a7fhwfldev1
       %2fdefault_1212826295%2esequence___5___fhwfdev%2ftonken%2fRadiobutton%2
       eproblem
11 Cookie: lonID=tonken_1214492867_fhwfdev_fhwfldev1
12 Content-Type: multipart/form-data; boundary
       =---------------------------59864454817761740331085019777
13 Content-Length: 648
14
15 -----------------------------59864454817761740331085019777
16 Content-Disposition: form-data; name="symb"
17 uploaded/fhwfdev/6o32055631bf647a7fhwfldev1/default_1212826295.
       sequence___5___fhwfdev/tonken/Radiobutton.problem
18 -----------------------------59864454817761740331085019777
19 Content-Disposition: form-data; name="submitted"
20
21 part_0
22 -----------------------------59864454817761740331085019777
23 Content-Disposition: form-data; name="HWVAL_12"
24
25 3
26 -----------------------------59864454817761740331085019777
27 Content-Disposition: form-data; name="submit_0"
28
29 Antwort einreichen
30 -----------------------------59864454817761740331085019777--
```

Der im Listing 5.11 dargestellte Netzwerkmitschnitt zeigt die komplette Anforderung einer Web-Seite per HTTP1.1-Protokoll. Informationen zum HTTP1.1-Protokoll können unter *http://tools.ietf.org/html/rfc2616* nachgelesen werden. Im Zusammenhang mit der Einsendung von Antworten sind vor allem die Zeilen 1, 11 und die Zeilen 12 bis 30 von Bedeutung. Diese werden kurz in der Tabelle 5.2 beschrieben.

Tabelle 5.2: Erläuterung zu wichtigen HTTP-Variablen

Zeile 1	POST gibt an, dass die zur Anfrage an den Server gehörigen Daten im Content-Bereich nach dem HTTP-Header beschrieben werden. Hinter POST folgt die Adresse, an die die Anfrage gesendet wird. Diese wird für die URI-Translation ausgewertet. Danach folgt noch die Version des HTTP-Protokolls.
Zeile 11	Das Cookie wird von LON-CAPA gesetzt, nachdem sich ein Nutzer eingeloggt hat. An Hand der Cookies werden in LON-CAPA die eingehenden Anfragen den angemeldeten Benutzern zugeordnet.
Zeile 12ff	Content-Type beschreibt in welcher Form die zur Anfrage gehörenden Daten beschreiben werden. Content-Length gibt die Länge der Daten an. Durch die leere Zeile 14 wird der HTTP-Header vom Content-Bereich getrennt, so dass in den folgenden Zeilen 15 bis 30 die angehängten Formulardaten stehen.

Für die Bearbeitung von Adressen mit der Endung '.problem' ist das Modul *lonhomework.pm* zuständig. Dessen Handler-Methode zeigt, dass zuerst die Benutzerberechtigung geprüft und etwas später die Methode *renderpage* aufgerufen wird. Dort werden gleich zu Beginn über den Aufruf von *&get_targets()* die Targets für den Request festgestellt. Dazu werden verschiedene Servervariablen ausgewertet. Im Falle der zuvor gesendeten Antwort sind *$env{'request.state'} == 'uploaded'* und *$env{'from.submitted'}* gesetzt. Daraus ergibt sich in der Prüfung, dass das Target 'grade' und das Target 'web' zurückgegeben werden. Danach werden beide Targets nach ihrer Reihenfolge, d.h erst 'grade' und danach 'web', per *lonxml.pm* verarbeitet. Zur Bewertung (Target 'grade') werden auch die Start- und End-Methoden zu den einzelnen XML-Elementen ausgeführt. Dabei werden nur die Codezeilen ausgeführt, die benötigt werden, um die personifizierte Version der Aufga-

be einzulesen. Anschließend wird je nach Aufgabentyp eine Methode aufgerufen, um die eingesendete Antwort zu bewerten. Wie das im Einzelnen geschieht, soll nicht weiter betrachtet werden. Erst nachdem die Bewertung durchgeführt wurde, wird die Aufgabe im Target 'web' an den Browser zurückgesendet, so dass in der Ansicht der Aufgabe die zuvor vorgenommene Bewertung berücksichtigt wird.

5.5.2 Eindeutige Zuordnung einer Antwort zur Aufgabe

Im Listing 5.11 wurde die Übertragung einer Antwort aufgezeigt. Die Daten im HTTP-Content-Bereich werden von LON-CAPA benötigt, um eine personifizierte Version einer Aufgabe wiederherzustellen und diese mit der eingegebenen Antwort zu bewerten. Es wurde bereits beschrieben, dass die Aufgabenstellung oder die Anordnung der Antworten in Aufgaben bei unterschiedlichen Kursteilnehmern variieren. Dazu wird jede Aufgabenvariation abhängig von einem sogenannten *seed* erstellt. Der *seed* kann als ein Zufallswert angesehen werden, der bei der Zuordnung von Aufgaben zu einem Kursteilnehmer generiert wird. Er wird aus der Adresse im Kurs (*Map*), der Nummer im Kurs und der Adresse der Aufgabe berechnet. Diese drei Informationen stehen im *symb*. Sie sind durch '___' der Reihe nach getrennt. Für den im Listing 5.11 aufgeführten *symb* würde das folgendes bedeuten [AKH02]:

Adresse im Kurs
fhwfdev/6o32055631bf647a7fhwfldev1/default_1212826295.sequence

Nummer im Kurs
5

Adresse der Aufgabe
fhwfdev/tonken/Radiobutton.problem

Aufgrund des Zusammenhangs von *symb* und *seed* kann die personifizierte Version einer Aufgabe allein aus dem *symb* beschrieben werden.

Die Antwort zu der oben aufgeführten Aufgabe ergibt sich aus den Daten der Formularfelder 'submitted', 'HWVAL_12' und 'submit_0'. Der Wert von 'submitted' und die Zahl hinter 'submit_' geben dabei Auskunft über den Teil (*part*) einer Aufgabe. Das ist nötig, da eine Aufgabe aus mehreren Teilaufgaben bestehen kann. Die einzelnen Teile werden innerhalb der gesamten Aufgabe durchnummeriert. 'HWVAL_12' enthält den Antwort-

wert. Im Falle einer Radiobutton-Aufgabe entspricht er einer Nummer, wobei bei anderen Aufgabentypen auch Zeichenketten vorkommen können. Auch der Zahlenwert 12 im Namen der 'HWVAL_12'-Bezeichnung kann je nach Aufgabentyp variieren. Anstatt einer einfachen Zahl können hier ebenfalls Zeichenketten an den Namen 'HWVAL_' angehängt werden. Da Aufgaben wie beispielsweise *Rank*- und *Matchresponse* mehr als einen Wert zur Beantwortung benötigen, müssen diese mehreren Bezeichnungen zugeordnet werden. Diese Bezeichnungen unterscheiden sich durch verschiedene Zeichenketten, die hinter 'HWVAL_' angehangen werden.

5.6 Zusammenfassung der Aufgabenanalyse

Das Vorgehen von LON-CAPA folgt beim Ausgeben und Bewerten von Aufgaben in vereinfachter Betrachtung immer dem gleichen Verfahren. Zuerst werden die Benutzerrechte geprüft, das/die Target(s) bestimmt und durch die in *lonxml.pm* registrierten Start- und End-Methoden für die einzelnen Targets verarbeitet. Im Target 'tex' müssen zusätzlich zu diesem Vorgehen das Druckmenü ausgegeben und der generierte TEX-Code in PDF gewandelt werden. Um PDF-Dateien mit Formularen zu ermöglichen, ist die Planung und die Implementierung eines neuen Targets sehr aufwändig, da in den verschiedenen Methoden, die über *lonxml.pm* aufgerufen werden, stark verschachtelte if-Anweisungen stehen. Diese unterscheiden aufgrund des angegebenen Targets, welcher Code generiert werden soll. Dabei werden oft Bereiche zusammengefasst, die für mehrere Targets gültig sind. Die Erstellung eines neuen Targets würde daher bedeuten, dass in allen betroffenen Methoden der Source-Code zuerst analysiert und dann geändert werden muss. Dieser Aufwand wäre enorm und könnte ohne massive Unterstützung des Kernentwickler-Teams nicht umgesetzt werden. Eine Alternative stellt die Erweiterung der Druckfunktion aufgrund der Ähnlichkeit des zu generierenden Codes dar. Durch das Einfügen eines neuen Auswahl-Feldes im Druckmenü, das bestimmt, ob eine Druck- oder Formularversion des PDFs generiert werden soll, kann an den zuständigen Source-Code-Stellen im Target 'tex' unterschieden werden, welcher TEX-Code generiert werden soll.

6 Planung der PDF-Formularfunktion

In diesem Kapitel werden die einzelnen Schritte zur Realisierung der PDF-Formularfunktion in LON-CAPA dargestellt werden. Dazu soll zunächst der grundsätzliche Ablauf einer Aufgabenbearbeitung per PDF-Formular beschrieben werden. Danach folgt die Planung der Änderungen und Erweiterungen, die in LON-CAPA eingebracht werden müssen, um die im Ablauf beschriebene Funktionalität zu ermöglichen. Dabei soll in der Planung noch nicht auf die in Kapitel 7 beschriebene Implementierung eingegangen werden.

6.1 Ablauf einer PDF-Formularbearbeitung

Aus der Analyse geht hervor, dass ein eigenes Target zum Generieren und Einsenden von Aufgaben einen zu großen Aufwand bedeuten würde. Daher soll die Generierung von PDF-Forumlaren als eine Erweiterung zur Druckfunktion (Target 'tex') realisiert werden, so dass allen Benutzerrollen die Möglichkeit geboten wird, wahlweise eine PDF-Druckversion oder ein PDF-Formular zu erstellen. Für die Bearbeitung der PDF-Formulare muss auf Seiten der Kursteilnehmer eine Software installiert sein, die es erlaubt, Formulardaten innerhalb von PDF zu speichern. Mögliche Softwarelösungen dazu werden im Abschnitt „Bearbeiten von PDF-Formularen“ genannt. Das Einsenden der bearbeiteten Formulare soll über eine Upload-Funktion per Web-Browser erfolgen. Für das anschließende Auslesen der Antworten wird ein lizenzfreies Software-Modul verwendet. Die ausgelesenen Daten aus dem PDF-Formular werden danach für die Bewertung vorbereitet und nacheinander durch serverinternen Request eingereicht. Die Rückmeldung zu den einzelnen Aufgaben kann der Kursteilnehmer nach dem Upload-Vorgang per Web-Browser in der Kursumgebung einsehen.

6.2 Erzeugung von PDF-Formularen

Wie im Abschnitt „PDF und LaTeX“ bereits beschrieben, werden zusätzliche Funktionen für LaTeX in Form von Makropaketen angeboten. Eine Recherche nach Makropaketen, die das Erstellen von PDF-Formularen ermöglichen, führte zu einer Auswahl von drei als besonders geeignet erscheinenden Paketen. Die Tabelle 6.2 soll diese kurz vorstellen.

Tabelle 6.1: Makropakete zur Erstellung von PDF-Formularen

hyperref	Das hyperref-Paket basiert auf HyperTeX[1] und ermöglicht die Integration von Links innerhalb von DVI und PDF-Dokumenten. Zudem bietet *hyperref* das Integrieren von PDF-Formularfeldern. [Sto07b]
acrotex	Das AcroTeX education bundle ist eine Sammlung von mehreren Makropaketen. Der Schwerpunkt von AcroTeX liegt in der Erstellung von Aufgabenblättern im Bildungsbereich, die durch PDF-internes JavaScript eine interaktive Benutzerschnittstelle zur Beantwortung und Bewertung von Aufgaben ermöglicht. Für die Erstellung der PDF-Formularfelder wird das Paket *eforms* verwendet. Es benutzt dazu Anpassungen von *hyperref*, so dass zur Verwendung von *eforms* auch das hyperref-Paket installiert sein muss. Zur Integration von JavaScript wird das zum AcroTeX gehörende Paket *insdljs* verwendet. [Sto07b]
aeb_pro	Das AcroTeX education bundle Professional bietet spezielle PDF-Erweiterungen zum AcroTeX education bundle (acrotex). Diese Erweiterungen setzten aber den Einsatz von Acrobat Professional ab Version 7.0 oder höher, sowie den zugehörigen Adobe Distiller voraus. [Sto07c]

Aufgrund der Anforderung, dass die verwendeten Software-Module linzenzfrei verfügbar sein müssen, kann das aeb_pro-Paket wegen der Abhängigkeit von Acrobat Professional nicht eingesetzt werden. Die beiden anderen Pakete *hyperref* und *acrotex* hingegen dürfen unter der LaTeX Project Public License (LPPL) frei verwendet werden. Ein weiterer Vorteil für den Einsatz dieser Pakete ist, dass sie in den meisten LaTeX-Distributionen bereits integriert sind.

Im Vergleich bieten *hyperref* und *eforms* den gleichen Umfang an PDF-Formularfeldern, jedoch haben sie Unterschiede in der Syntax und in der Art, wie die einzelnen Formu-

[1] Informationen zu HyperTeXkönnen unter *http://arxiv.org/hypertex/* eingesehen werden.

larfelder in ein PDF-Dokument eingebunden werden. So muss beispielsweise in *hyperref* eine zusätzliche Formularumgebung für die Formularfelder angegeben werden. Diese Umgebung darf nur einmal im gesamten Dokument vorkommen. [RO06] Zudem werden auch für gruppierte Schaltflächen, zum Beispiel eine „1-aus-n"-Auswahl per Radiobuttons, spezielle Umgebungen verwendet. Diese müssen zusätzlich durch eine Größenangabe in Höhe und Breite beschrieben werden, so dass die Umgebungen für unterschiedliche Beschriftungen individuell angepasst werden müssen. Das eforms-Paket vereinfacht die Verwendung, so dass Formularfelder beliebig in den normalen Textfluss integriert werden können. Gruppierte Radiobuttons werden bei *eforms* durch einen gleichen Formularfeldnamen erkannt. Im Allgemeinen ist die Syntax von *eforms* wesentlich vielseitiger einzusetzen, so dass besonders mit Blick in die Zukunft, der Einsatz von *eforms* dem Einsatz von *hyperref* vorzuziehen ist. Deshalb wird im Weiteren die Realisierung der PDF-Formulare mit *eforms* aus dem AcroTEX education bundle vorgenommen. Damit die PDF-Formulare überhaupt erzeugt werden können, sollte geprüft werden, ob das hyperref- und das acrotex-Paket auf dem Server installiert sind, damit gegebenfalls die fehlenden Pakete nachinstalliert werden. Das kann je nach TEX-Distribution auf verschiedene Weise erfolgen, so dass an dieser Stelle auf die jeweilige Dokumentation der installierten Distribution verwiesen werden soll.

In Listing 6.1 ist als Beispiel eine TEX-Datei mit Radiobuttons, Textfeldern, einer Combobox und einer Schaltfläche zum Löschen der Formularwerte aufgeführt. Die erzeugten PDF-Formularfelder sind in Abbildung 6.1 als Screenshot abgebildet.

Listing 6.1: eforms Minimalbeispiel (ohne Formatierung)

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[dvips]{eforms}
\usepackage{hyperref}

\begin{document}
Herr: \radioButton{gruppe}{10bp}{10bp}{Mann}
Frau: \radioButton{gruppe}{10bp}{10bp}{Frau}\\
Name: \textField {name}{1.5in}{12bp}\\
Vorname: \textField {firstname}{1.5in}{12bp}\\
Familienstand:
\comboBox[]{ComboBox1}{1in}{14bp}{(ledig)(verheiratet)(geschieden)}\\
\pushButton[\CA{Formular löschen}\A{/S/ResetForm}]{myButton}{}{12bp}\\
\end{document}
```

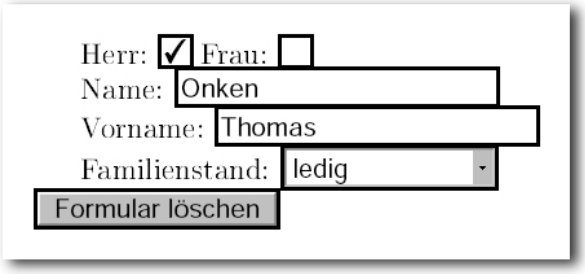

Abbildung 6.1: Radiobutton-Aufgabe mit Formularfeldern

Die Befehle zur Erzeugung der Formularfelder haben sehr umfangreiche Parameter und variieren je nach Typ. Sie sind in der eforms-Dokumentation [Sto07a] detailliert an Hand von Beispielen beschrieben, weshalb auch im Weiteren auf eine Beschreibung der Parameter verzichtet wird. Als zweites Beispiel zum Umgang mit *eforms* soll die bereits bekannte Radiobutton-Aufgabe von Hand um eine PDF-Formularfunktion erweitert werden. Die von der Druckfunktion erzeugte TEX-Datei liegt im Druckordner unter */home/httpd/prtspool*. Im Listing 6.2 wird ein Auszug mit den für die Erweiterung relevanten Stellen dargestellt.

Listing 6.2: Auszug einer per Druckfunktion erstellten TEX-Datei

```
[...]
\usepackage{fancyhdr}
\addtolength{\headheight}{\baselineskip}
\pagestyle{fancy}
\fancyhead{}\
fancyhead[LO]{
\textbf{Thomas Onken} - PDF - Formulare \hfill \thepage \\
\textit{Radiobuttonresponse}}
\begin{document}
\voffset=-0.8 cm\setcounter{page}{1}
\typeout{STAMPOFPASSEDRESOURCESTART Resource [...]}
\noindent \end{minipage}\vskip 0 mm \noindent
\begin{minipage}{\textwidth}\noindent
\noindent Wie lautet das Ergebnis folgender Aufgabe im binären Zahlensystem?
1 + 1 = ?\strut \\ \strut
\renewcommand{\labelenumi}{\Alph{enumi}.}
\begin{enumerate}
  \vspace*{-2 mm}\item
  \noindent 0
  \vspace*{-2 mm}\item
```

```
  \noindent 10
  \vspace*{-2 mm}\item
  \noindent 110
  \vspace*{-2 mm}\item
  \noindent 2
  \vskip 0 mm
  \keephidden{\noindent Denken Sie digital!}
\end{enumerate}
[...]
```

Um die Formularfunktion von LaTeX nutzen zu können, muss das Paket *eforms* in der TeX-Datei mit dem Befehl „\usepackage[dvips]{eforms“ geladen werden. Die zusätzliche Treiberangabe [dvips] bezweckt, dass *eforms* für die PDF-Generierung über eine PostScript-Datei angepasst wird. [Sto07a] Als nächstes sollen die einzelnen \items in der *enumerate*-Umgebung durch eine Radiobutton-Gruppe ersetzt werden, so dass sie ähnlich der Darstellung im Web-Browser angezeigt werden. Um dieses zu erreichen, wird jeder Radiobutton mit dem zugehörigen Antworttext in der Tabelle formatiert. Um sicherzustellen, dass auch längere Texte passend umgebrochen werden, wird dazu das Makro \tabularx verwendet. Dieses muss wie das eforms-Paket vor Beginn des Dokumenten-Bereichs mit \usepackage{tabularx} geladen werden. Listing 6.3 zeigt die Änderungen in der TeX-Datei. Das Ergebnis nach dem Generieren der PDF-Datei zeigt die Abbildung 6.2.

Listing 6.3: Änderung der TeX-Datei

```
[...]
\usepackage[dvips]{eforms}
\usepackage{tabularx}
\usepackage{fancyhdr}
\addtolength{\headheight}{\baselineskip}
\pagestyle{fancy}
\fancyhead{}\
fancyhead[LO]{
\textbf{Thomas Onken} - PDF - Formulare \hfill \thepage \\
\textit{Radiobuttonresponse}}
\begin{document}
\voffset=-0.8 cm\setcounter{page}{1}
\typeout{STAMPOFPASSEDRESOURCESTART Resource [...]}
\noindent \end{minipage}\vskip 0 mm \noindent
\begin{minipage}{\textwidth}\noindent
\noindent Wie lautet das Ergebnis folgender Aufgabe im binären Zahlensystem?
1 + 1 = ?\strut \\ \strut
```

```
\begin{tabularx}{\textwidth}{p{2mm}X}
 \radioButton[\symbolchoice{circle}]{gruppenname}{10bp}{10bp}{R1} & 0
\end{tabularx}
\hspace{3mm}
\begin{tabularx}{\textwidth}{p{2mm}X}
 \radioButton[\symbolchoice{circle}]{gruppenname}{10bp}{10bp}{R2} & 10
\end{tabularx}
\hspace{3mm}
\begin{tabularx}{\textwidth}{p{2mm}X}
 \radioButton[\symbolchoice{circle}]{gruppenname}{10bp}{10bp}{R3} & 110
\end{tabularx}
\hspace{3mm}
\begin{tabularx}{\textwidth}{p{2mm}X}
 \radioButton[\symbolchoice{circle}]{gruppenname}{10bp}{10bp}{R4} & 2
\end{tabularx}
\hspace{3mm}
  \keephidden{\noindent Denken Sie digital!}
[...]
```

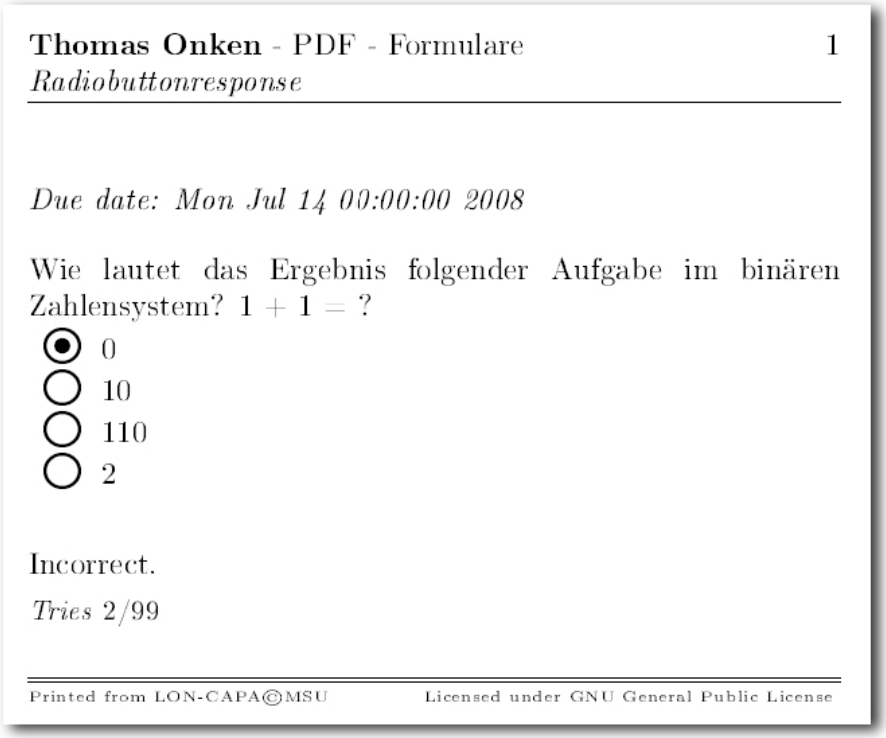

Abbildung 6.2: Radiobutton-Aufgabe mit Formularfeldern

In der erweiterten TEX-Datei der Radiobutton-Aufgabe haben die Namen der Felder die Bezeichnung „gruppenname“ und die zugehörigen Werte die Werte R1-R4. In der späteren Implementierung muss die Verbindung von Namen und Wert des Formularfeldes eine eindeutige Zuordnung zu einer Aufgabe in LON-CAPA inklusive Antwort haben. Dazu mehr im Abschnitt „Bewertung der Formulardaten“.

6.3 Bearbeitung von PDF-Formularen

Im Kapitel 2 wurde bereits erwähnt, dass eine ganze Reihe von PDF-Tools als kommerzielle und freie Software angeboten werden. Die sicher bekanntesten dieser Art sind der kostenpflichtige Adobe Acrobat und der freie Adobe Reader. Zwei vielversprechende freie PDF-Tools sind CABAReT Stage von CABAReT-Solutions[2] und PDF-XChange Viewer von Tracker Software Products[3]. Jedoch ermöglichen beide PDF-Tools die Bearbeitung und Speicherung der mit *eforms* erstellten PDF-Formlare nur eingeschränkt. CABAReT Stage hat Probleme mit der Bedienung von Radiobuttons. Diese können wie Checkboxen ein- und ausgeschaltet werden, obwohl eine „1-aus-n“-Auswahl mit *eforms* erzeugt wurde. Das hat zur Folge, dass die Radiobuttons nach dem Deaktivieren den Wert „Off“ annehmen. Die Funktion der anderen Formularfeldtypen ist wie gewohnt. PDF-XChange Viewer zeigt keine Radiobuttons und Checkboxen an und bricht häufig bei der Bearbeitung von Formularfeldern mit einer Windows-Fehlermeldung ab. Als einzige getestete Software funktioniert der kostenpflichtige Adobe Acrobat ohne erkennbare Probleme. Ein besonderer Effekt tritt auf, wenn die per *eforms* erzeugten PDF-Formulare mit dem Adobe Acrobat nach der Bearbeitung gespeichert werden. Danach stimmen in beiden freien Programmen die Funktionen und die Darstellung der Formularfelder. Wird jedoch das von *eforms* erzeugte Dokument zuerst mit CABAReT Stage bearbeitet, stimmen die Darstellungen im Adobe Acrobat nicht mehr. Es lässt sich daher vermuten, dass die von *eforms* erstellten Formulare nicht genau dem PDF-Standard entsprechen, aber der Adobe Acrobat in der Lage ist, diese Dokumente richtig zu interpretieren. Beim Speichern werden von Acrobat geänderte Formulardaten als neue *Objects* (siehe Kapitel 3.1) an eine PDF-Datei angehangen. Diese werden gemäß der PDF-Spezifikation erstellt und überlagern die per *eforms* erstellten, so dass beim erneuten Laden der PDF-Datei nur noch die von Acrobat erstellten Formular-*Objects* interpretiert werden. Eine optimale Lösung zu diesem Problem konnte im Rahmen der vorliegenden Ba-

[2] http://www.cabaret-solutions.com
[3] http://www.pdfxviewer.com

chelorarbeit nicht gefunden werden. Aufgrund der ständigen Weiter- und Neuentwicklug im Bereich der PDF-Software sind die Chancen nicht schlecht, dass in zukünftigen freien Produkten die Unterstützung von per *eforms* erstellten PDF-Formularen ermöglicht wird. In der späteren Implementierung soll die Bearbeitung mit CABARet Stage und, aufgrund der weiten Verbreitung, die Verwendung des Adobe Acrobat unterstützt werden.

6.4 Auslesen der Formulardaten

Die Datenstruktur von PDF-Dokumenten lässt es nicht zu, Formulardaten mit einem einfachen Suchverfahren, wie beispielsweise regulären Ausdrücken, aufzufinden. Daher muss auf eine freie Software oder ein freies Software-Modul zurückgegriffen werden. LON-CAPA ist in Perl programmiert, daher liegt es nahe, eine Lösung in dieser Programmierspache zu verwenden. Bei CPAN(Comprehensive Perl Archive Network)[4] gibt es drei umfangreiche Perl-Module, die sich mit der Verarbeitung von PDF-Dokumenten beschäftigen.

- CAM::PDF von Chris Dolan
- PDF von Antonio Rosella
- Text::PDF von Martin Hosken

Als einziges Modul bietet CAM::PDF die Möglichkeit eine Liste aller Formularfeldnamen auszugeben und mit Hilfe einer Methode die Werte eines bekannten Formularfeldes zu setzten, jedoch fehlt auch hier eine Möglichkeit, die bereits vorhandenen Werte eines Formularfeldes auszulesen. Das Modul um diese Funktionalität zu erweitern erscheint möglich, würde jedoch den Zeitraum der vorliegenden Abschlussarbeit überziehen. Daher soll als eine Alternative ein Java-Paket mit dem Namen *jPod* eingesetzt werden, bis ein PerlModul für diese Funktion bereitsteht. Es wurde ursprünglich von Intarsys[5] entwickelt und wird als Open-Source-Projekt frei zur Verfügung gestellt. Zudem basiert CABAReT Stage auf diesem Paket, so dass die Kompatibilität zwischen von CABAReT Stage bearbeiteten Dokumenten und *jPod* am Größten erscheint. *jPod* kann bei SOURCEFORGE.NET unter *http://sourceforge.net/projects/jpodlib/* heruntergeladen werden. Im Download-Archiv ist neben Sourcecode und Java-Klassen eine Vielzahl an Minimalbeispielen zur Verwendung von *jPod* gegeben. Darunter findet sich auch ein Beispiel, das alle Formularfelder zusammen mit den zugehörigen Werten aus einem PDF-Dokument ausliest. Dabei entsprechen

[4] http:/www.cpan.org
[5] http://www.intarsys.de/

Feldbezeichner und Wert einer Zeichenkette. Dieses Beispiel wird später als Grundlage einer Java-Anwendung verwendet, die die eingetragenen Antworten aus der hochgeladenen PDF-Datei ausliest und in eine Datei schreibt. Der Aufruf dieser Anwendung erfolgt aus einem Perl-Modul, das auch für das Hochladen der PDF-Datei zuständig ist. Dabei soll der Aufruf in einer separaten Methode stehen, damit er, sobald eine Perl-Lösung verfügbar ist, ohne großen Aufwand geändert werden kann.

6.5 Bewertung der Formulardaten

Zur Bewertung der Aufgaben soll die Web-Schnittstelle genutzt werden. Daher müssen die ausgelesenen Antworten aus dem PDF-Formular mit einem serverinternen Request, wie bei der Beantwortung per Web-Browser eingereicht werden. In der Analyse wurde festgestellt, dass zur eindeutigen Zuordnung der *symb*, der *part*, die HTML-Formularfeld-Bezeichnung (*HWVAL_*) und der im HTML-Feld stehende Wert benötigt werden. Anhand dieser Daten kann kein Rückschluss auf den Aufgabentyp geschlossen werden, daher wird der Name des Aufgabentyps zusätzlich angegeben. Dadurch ist es möglich den Auslesevorgang für einzelne Aufgabetypen anzupassen. Zur Unterbringung der Zuordnungsdaten stehen nach der Auslesung per *jPod* der Formularfeldname und der Wert als Zeichenkette zur Verfügung. Die benötigten Zuordnungsdaten müssen daher mit Hilfe eines Trennzeichen voneinander geteilt in diesen Zeichenketten untergebracht werden. Um in Aufgaben mit freien Antworten wie beispielsweise Textresponse zu vermeiden, dass ein in der Antwort verwendetes Trennzeichen das spätere Aufteilen der zusammengesetzten Zeichenkette stört, werden alle Zuordnungsdaten im Formularfeldnamen zusammengefügt. Das bietet zusätzlich den Vorteil, dass jeder Formularfeldname nur einmal im PDF-Dokument vorkommen kann. Als Trennzeichen müssen Zeichen gewählt werden, die nicht in den Zuordnugsdaten vorkommen. Da die Zuordnungsdaten für die Verwendung im Web-Browser von LON-CAPA angepasst sind, kann eines der nicht erlaubten Zeichen für HTML-Formularnamen benutzt werden. Zu diesen Zeichen gehört unter anderem das Fragezeichen(?) und das Kaufmännische-Und(&). Auch im Namen der Aufgaben, die von Autoren erstellt werden, können diese beiden Zeichen nicht vorkommen, da LON-CAPA sie automatisch entfernt.

Zusätzlich sollen Daten in das PDF-Dokument eingebracht werden, die es ermöglichen zu erkennen, welcher Benutzer das Dokument erzeugt hat. Damit wird erreicht, dass ein Benutzer nur seine eigenen Aufgaben zur automatischen Bewertung einreichen kann. Um diese zusätzliche Sicherheitsmaßnahme zu erreichen, wird bei jeder Aufgabe ein nicht-sichtbares Formularfeld mit Informationen zum Kursteilnehmer eingefügt.

Die Web-Schnittstelle erwartet als Antwort auf eine Aufgabe eine Anfrage mit POST-Daten, die vom Browser an die personifizierte Version der Aufgabe geschickt wird. Die POST-Daten müssen dabei dieselben sein, wie bei der Beantwortungen aus der Kursumgebung per Web-Browser. Nachdem alle Antworten aus dem PDF-Formular zur Bewertung gesendet wurden, soll eine Übersicht über die erfolgreich eingesendeten Antworten und ein Link zum Kurs-Inhaltsverzeichnis im Web-Browser angezeigt werden.

7 Implementierung der Formularfunktion

In diesem Kapitel soll beschrieben werden, wie die zuvor geplanten Änderungen und Erweiterungen in LON-CAPA implementiert werden. Dazu werden alle Implementierungen beschrieben, die zur Umsetzung der PDF-Formularfunktion für Radiobutton-Aufgaben gemacht werden müssen. Die Übertragbarkeit auf andere Aufgabentypen wird im letzten Abschnitt dieses Kapitels dargestellt. Alle geänderten Zeilen werden im Sourcecode auf der DVD-ROM mit einem Kommentar „#TO“ markiert, so dass die geänderten Stellen schnell per Suchfunktion gefunden werden können.

7.1 Erweiterung des Druckmenüs

Das Druckmenü hat verschiedene Ansichten für jede Benutzerrolle. Die unteren Auswahlfelder zum Festlegen des Seitenlayouts sind jedoch bei allen Rollen gleich, so dass sich ein Einbringen des Auswahlfeldes, um zwischen Druck- und Formularversion zu unterscheiden, hier anbietet. Auch von der Thematik her passt die Auswahl an diese Stelle. Der untere Teil des Druckmenüs wird als HTML-Code im Perl-Modul *lonprintout.pm* in der Methode *render()* erzeugt. Als Auswahlfeld wird eine ComboBox mit dem Namen „FORMAT.pdfFormFields“[1] verwendet. Sie hat den Wert *yes* wenn *with PDF-Formfields* und *no* wenn *without PDF-Formfields* in ihr ausgewählt wird. Um die ComboBox zu integrieren, wird zu Beginn der *render()*-Methode die Überschrift und am Ende die ComboBox hinzugefügt. Damit die angezeigten Texte im Druckmenü für verschiedene Sprachen angepasst werden können, sollten sie über die Methode *mt()* definiert werden. Listing 7.1 zeigt die geänderten Code-Zeilen. Das erweiterte Druckmenü wird in der Abbildung 7.1 dargestellt.

[1] „FORMAT“ wird im Sourcecode durch die Variable $var dargestellt

Listing 7.1: Änderung in *render()*

```
[...]
my $landscape=&mt('Landscape');
my $portrait=&mt('Portrait');
my $pdfForm=&mt('PDF-Formfields');#TO

$result .= <<STATEHTML;
<hr width="33%" />
<table cellpadding="3">
  <tr>
    <td align="center"><b>$PageLayout</b></td>
    <td align="center"><b>$NumberOfColumns</b></td>
    <td align="center"><b>$PaperType</b></td>
    <td align="center"><b>$pdfForm</b></td><!--#TO -->
  </tr>
  <tr>
[...]
$result .= "</select></td>";#TO
$result .= "<td align='center'><select name='${var}.pdfFormfields'>";
$result .= "<option selected value='no' />".&mt('without Formfields')."</option>"; #TO
$result .= "<option value='yes'>".&mt('with Formfields')."</option></select></td>";#TO
$result .= "</tr></table>";#TO
return $result;
}
```

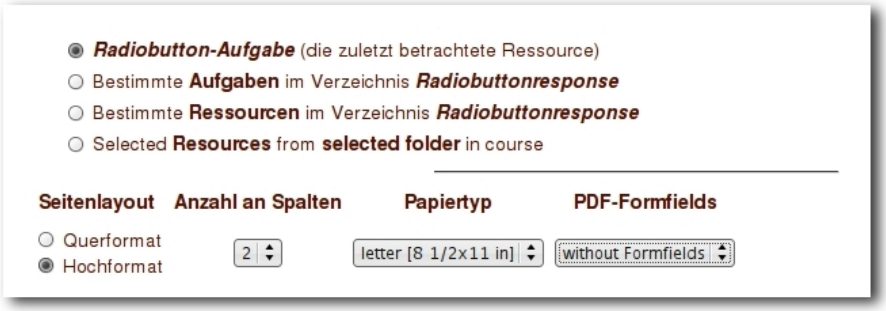

Abbildung 7.1: Erweitertes Druckmenü

Damit später geprüft werden kann, ob eine Druck- oder Formularversion ausgegeben werden soll, muss der Wert von „FORMAT.pdfFormfields“ so wie die Werte der anderen Auswahlfelder zwischengespeichert werden. Dazu werden die Werte der Auswahlfelder nach dem Absenden der Druckoption in der Methode *postprocess* mit „|“ als Trennzeichen in eine Zeichenkette zusammengefasst und im *Helper*[2] gespeichert.

Listing 7.2: *postprocess()*

```
sub postprocess {
    my $self = shift;

    my $var = $self->{'variable'};
    my $helper = Apache::lonhelper->getHelper();
    $helper->{VARS}->{$var} =
        $env{"form.$var.layout"}.
        '|'.$env{"form.$var.cols"}.
        '|'.$env{"form.$var.paper"}.
        '|'.$env{"form.$var.pdfFormFields"};
    return 1;
}
```

Die zusammengesetzte Zeichenkette wird in *outputdata()* mit einem *split*-Befehl auseinander geschnitten und in einzelne Variablen gespeichert. Anhand der Werte aus dem Druckmenü wird entschieden, wie und was gedruckt werden soll. Die tex-Versionen der im Druckmenü ausgewählten Aufgaben werden etwas später im Code mit dem Aufruf von *ssi_with_retries($currentURL, $ssi_retry_count, %form))* generiert. Dabei wird ein serverinterner Request an die einzelnen Aufgaben geschickt, dessen Antwort die tex-Version der angeforderten Aufgabe enhält. Die Daten im übergebenen Hash *%form* stehen in den generierenden Modulen im globalen Hash *%env* und können mit *$env'form.(Schlüssel aus %form)* abgefragt werden. Das bedeutet, dass die Variable, die zur Unterscheidung zwischen Druck- und Formularversion herangezogen wird, in *%form* geschrieben werden muss, bevor die tex-Version der Aufgabe generiert wird. Listing 7.3 zeigt Auschnitte der geänderten Code-Zeilen in *output_data()*.

[2] Helper ist ein Modul von LON-CAPA, das es erlaubt, Variablen global in einem Hash-ähnlichen Objekt zu speichern.

Listing 7.3: Änderungen in *output_data()*

```
[...]
my $number_of_columns = 1; #used only for pages to determine width of the cell
my @temporary_array=split /\|/,$format_from_helper;
my ($laystyle,$numberofcolumns,$papersize,$pdfFormfields)=@temporary_array;
if ($laystyle eq 'L') {
[...]
my %form;
$form{'grade_target'} = 'tex';
$form{'textwidth'}    = &get_textwidth($helper, $LaTeXwidth);
$form{'pdfFormfields'} = $pdfFormfields;#TO
[...]
```

7.2 Erzeugen der PDF-Formularfelder

Nach den Änderungen in *lonprintout.pm* kann bei der Generierung der TEX-Version einer Aufgabe anhand der Variablen $env{'form.pdfFormfields} geprüft werden, ob PDF-Formularfelder im späteren PDF-Dokument benutzt werden sollen. Um später Formularfelder benutzen zu können, müssen im nächsten Schritt die Makropakete *eforms* und *tabularx* in der Präambel der TEX-Datei eingefügt werden. Damit die Formularfunktion keinen Einfluss auf die bisherige Druckfunktion nimmt, sollen die zusätzlichen Makropakete nur geladen werden, wenn eine Formularversion ausgegeben werden soll. Dazu wird eine if-Anweisung in *start_html()* von *londefdef.pm* eingefügt, die diese Aufgabe übernimmt. Listing 7.4 zeigt die eingefügten Änderungen.

Listing 7.4: Einfügen der zusätzlichen Makropakete

```
[...]
                    '\usepackage{amsfonts}'.
                    '\usepackage{amsthm}'.
                    '\usepackage{amscd}';
  if($env{'form.pdfFormfields'} eq 'yes') {
    $currentstring .= "\n".
                      '\usepackage{eforms}' . "\n".
                      '\usepackage{tabularx}' . "\n";
  }
  $currentstring .= '\newenvironment{choicelist}{\begin{
[...]
```

Als nächstes soll die Formularfunktion für verschiedene Aufgabentypen implementiert werden. Die Generierung einzelner PDF-Formularfelder soll in eigenen Methoden, denen die Daten für Feldbezeichnung und Feldwert als Parameter übergeben werden, untergebracht werden. Das hat den Vorteil, dass Änderungen an den Feldern an zentraler Stelle vogenommen werden können. Auf Rat von Gerd Kortemeyer vom LON-CAPA Kernentwicklerteam werden diese Methoden in *lonxml.pm* untergebracht. In der Analyse wurde bereits detailliert der Generierungsablauf einer Radiobutton-Aufgabe beschrieben. Daher soll anhand dieses Aufgabentyps auch das weitere Vorgehen beschrieben werden. Im Modul *radiobuttonresponse.pm* werden in der Methode *displayfoils()* oder *displayallfoils()* die HTML- und TEX-Zeilen für die einzelnen Antwortmöglichkeiten ausgegeben. In der Druckversion werden die einzelnen Antworten in einer *enumerate*-Umgebung als *items* dargestellt. Die Umgebung wird in *start_ radiobuttonresponse()* mit *\begin{enumerate}* eingeleitet und in *end_ radiobuttonresponse()* mit *\end{enumerate}* beendet. Um die Radiobuttons sauber in das PDF zu integrieren, darf die *enumerate*-Umgebung nur erzeugt werden, wenn eine Druckversion erzeugt wird. Das kann ähnlich wie in *londefdef.pm* mit Hilfe von if-Anweisungen realisiert werden. Anstatt des Einfügens von *\begin{enumerate}* bietet es sich an, die Informationen zum Ersteller einzubringen. Dazu wird ein sehr kleines, nichtsichtbares Formulartextfeld mit Daten zu Domain und Benutzernamen des Erstellers eingefügt. Erzeugt wird dieses Feld mit dem Aufruf von *Apache::lonxml::print_ pdf_ hiddenfield()*. Als Parameter werden die Bezeichnung des Feldes, Benutzername und Benutzerdomain übergeben. Als Bezeichnung soll bei allen Benutzerinformationen die Zeichenkette 'meta' angegeben werden, damit sie beim späteren Auslesen eindeutig ausgewertet werden kann. Benutzername und Benutzerdomain werden mit einem Kaufmännischen-Und (&) getrennt, wodurch sie nach dem Auslesen einfach getrennt werden können. Hinter das Textfeld muss noch ein Zeilenumbruch eingefügt werden. Dieser wird beim Darstellen einer *enumerate*-Umgebung automatisch erzeugt. Listing 7.5 zeigt die Änderungen an der *enumerate*-Umgebung und das Listing 7.6 zeigt die Methode zur Erzeugung des versteckten Textfeldes. *radiobuttonresponse.pm*

Listing 7.5: Anpassung der *enumerate*-Umgebung

```
sub start_radiobuttonresponse {
[...]
    if($env{'form.pdfFormFields'} eq 'yes') {
        $result .= &Apache::lonxml::print_pdf_hiddenfield('meta',
                                                          $env{'user.name'},
                                                          $env{'user.domain'});
        $result .= "\n\\\\\n\\\\\n";
    } else {
        $result .= '\begin{enumerate}';
    }
[...]
sub end_radiobuttonresponse {
    my ($target,$token,$tagstack,$parstack,$parser,$safeeval,$style)=@_;
    my $result;
    if ($target eq 'edit') { $result=&Apache::edit::end_table(); }
    if ($target eq 'tex') {
        if($env{'form.pdfFormFields'} eq 'yes') {
            #do nothing
        } else {
            $result .= '\end{enumerate}';
        }
    }
[...]
```

Listing 7.6: Implementierung von *print_pdf_hiddenfield()*

```
sub print_pdf_hiddenfield {
    my $result;
    my ($fieldname, $user, $domain) = @_;
    $result .= '\textField [\F{\FHidden}\F{-\FPrint}'.
               '\V{'.$domain.'&'.$user.'}]{'.$fieldname.'}'.
               '{0in}{0in}'."\n";
    return $result;
}
```

Als nächstes werden die *items* beim Erzeugen einer Formularversion in *displayfoils()* und *displayallfoils()* durch den Aufruf einer neuen Methode *print_pdf_radiobutton()* in *lonxml.pm* ersetzt. Diese Methode bekommt die Bezeichnung und den Wert des Radiobuttons und den Antworttext als Parameter übergeben. Aus der Bezeichnung des Radiobuttons müssen alle Informationen zur Zuordnung der Aufgabe untergebracht werden. Diese Daten können aus den in Tabelle 7.1 aufgezeigten Variablen gelesen werden.

Tabelle 7.1: Variablen mit Zuordnungsdaten

symb	$env{'request.symb'}
part	$Apache::inputtags::part
*HWVAL_*	$Apache::inputtags::response['-1']

Die Zuordnungsdaten und der Aufgabentyp werden mit einem Kaufmännischen-Und (&) getrennt in eine Zeichenkette geschrieben, so dass sich die Reihenfolge *symb*, *part*, *Aufgabentyp* und *HWVAL_* ergibt. Diese wird als Bezeichnung des Radiobuttons angegeben. Der Wert des Radiobuttons kann aus einer Variablen *$temp* gelesen werden. Der Aufgabentext steht in der Variable *$Apache::response::foilgroup{$name.'.text'}*. Listing 7.7 zeigt die Implementierung in *displayfoils()*. In *displayallfoils()* sind die benötigten Änderungen sehr ähnlich, so dass sie hier nicht näher betrachtet werden. In Listing 7.8 wird die Methode *print_pdf_radiobutton()* zur Generierung des TEX-Codes für einen Radiobutton mit Antworttext abgebildet.

Listing 7.7: Auszug aus *displayfoils()*

```
[...]
    $bubble_number++;
    } else {
        if($env{'form.pdfFormFields'} eq 'yes') {
            my $fieldname = $env{'request.symb'}.
                                '&part_'. $Apache::inputtags::part.
                                '&radiobuttonresponse'.
                                '&HWVAL_' . $Apache::inputtags::response['-1'];
            my $value = $temp;
            my $text = $Apache::response::foilgroup{$name.'.text'};
            $result .= &Apache::lonxml::print_pdf_radiobutton($fieldname,
                                                              $value,
                                                              $text);
            $result .= '\newline'."\n";
        } else {
            $result .= '\vspace*{-2 mm}\item '.
            $Apache::response::foilgroup{$name.'.text'};
        }
    }
}
[...]
```

Listing 7.8: *print_pdf_radiobutton()*

```
sub print_pdf_radiobutton {
    my $result = '';
    my ($fieldName, $value, $text) = @_;
    $result .= '\begin{tabularx}{\textwidth}{p{0cm}X}'."\n";
    $result .= '\radioButton[\symbolchoice{circle}]{'.
               $fieldName.'}{10bp}{10bp}{'.$value.'}&'.$text."\n";
    $result .= '\end{tabularx}' . "\n";
    $result .= '\hspace{2mm}' . "\n";
    return $result;
}
```

Nachdem alle in diesem Abschnitt aufgezeigten Änderungen in LON-CAPA durchgeführt wurden, können Radiobutton-Aufgaben im Target 'tex' wahlweise als PDF-Formularversion gedruckt werden. Die Radiobuttons beinhalten dabei alle Informationen die benötigt werden, um sie später bei der Bewertung zu ihren personifizierten Aufgaben zuzuordnen. Die bisher nicht geänderten Aufgabentypen werden wie zuvor als normale Druckversion ausgegeben.

7.3 Hochladen der PDF-Formulare

Für die Rücksendung der bearbeiteten PDF-Formulare wird ein neues Perl-Modul mit dem Namen *lonpdfupload.pm* in LON-CAPA eingefügt. Das Modul soll eine Funktion zum Hochladen der bearbeiteten PDF-Formulare bieten, aus denen anschließend die Antworten ausgelesen und bewertet werden. Die nötigen Schritte werden in den folgenden Unterkapiteln beschrieben.

7.3.1 Ein Perl-Modul in LON-CAPA hinzufügen

LON-CAPA läuft auf einem Apache Web-Server mit mod_perl, weshalb neue Module den Spezifikationen der mod_perl Schnittstelle entsprechen müssen. Diese gibt vor, dass das Modul mindestens eine Methode mit dem Namen *handler()* besitzen muss. Die Methode *handler()* wird aufgerufen, wenn das Modul über eine Direktive in der Apache-Konfigurationsdatei[3] mit einer URL verbunden wird. Das neue Modul soll unter der URL

[3]Bei LON-CAPA besteht die Konfigurationsdatei aus mehreren Dateien. Die Direktiven zur Zuordnung der URLs werden in der Datei /etc/httpd/conf/loncapa_apache.conf definiert [AKH02]

(Adresse des Servers)/pdfupload/ zu erreichen sein. Daher muss die im Listing 7.9 aufgeführte Direktive in die Konfigurationsdatei hinzugefügt werden.

Listing 7.9: Direktive für *lonpdfupload.pm*

```
<LocationMatch "^/pdfupload/">
SetHandler perl-script
PerlHandler Apache::lonpdfupload
</LocationMatch>
```

Zusätzlich muss das Modul beim Starten des Web-Servers geladen werden. Dazu wird das neue Modul in die Liste der bereits integrierten Module im Perl-Skript *startup.pl*[4] eingetragen. Listing 7.10 zeigt einen Auszug aus dieser Datei. Danach kann das Modul *lonpdfupload.pm* unter oben angegebener URL erreicht werden.

Listing 7.10: Auszug aus *startup.pl*

```
use Apache::longroup();
use Apache::groupboards();
use Apache::lonclonecourse();
use Apache::lonuserutils();
use Apache::lonpdfupload();
1;
__END__
```

Beim Ausführen der *handler()*-Methode wird vom Server ein Request-Objekt übergeben. Darin können alle Informationen von der Anfrage des Web-Browsers eingesehen werden. Dieses Objekt ist zusätzlich für die Antwort des Servers verantwortlich. Die Rückmeldungen, die an den Browser in einem normalen CGI-Skript per *print* zurückgesendet werden, müssen im mod_perl-Betrieb in das Request-Objekt geschrieben werden. Zuvor muss jedoch der Typ der Rückmeldung angegeben werden. Im Falle einer Web-Seite wird über die Methoden *content_type(„text/html“)* und *send_http_header* im Request-Objekt bewirkt, dass der Web-Browser die Rückantwort als Web-Seite interpretiert. Danach kann mit der Methode print() des Request-Objekts der HTML-Code der Seite gesendet werden. Besonders hilfreich sind Methoden in anderen Modulen von LON-CAPA, die automatisch HTML-Code-Teile der LON-CAPA Web-Seite generieren. Beispielsweise erzeugt der Aufruf von *&Apache::loncommon::start_page('(Titel der Seite)')* eine an die Benutzerrolle angepasste Navigationsleiste, wie sie in anderen Bereichen von LON-CAPA dargestellt wird. Wie dieses im einzelnen eingesetzt wird, kann im Listing 7.11 eingesehen werden.

[4] /etc/httpd/conf/startup.pl

Listing 7.11: Handler von *lonpdfupload.pm*

```
sub handler() {
  my $r = shift;

  #Testen ob der Benutzer ein gueltiges Cookie besitzt
  if(!&checkpermission($r)) {
    return OK;
  }

  $Apache::lonxml::request=$r;
  $Apache::lonxml::debug=$env{'user.debug'};
  $env{'request.uri'}=$r->uri;

  $r->content_type('text/html');
  $r->send_http_header();
  $r->print(&Apache::loncommon::start_page('Upload-PDF-Form'));

  #lade die per POST gesendenten daten in %env
  &Apache::lonacc::get_posted_cgi($r);

  if($env{'form.Uploaded'} && $env{'form.file'}) {
    #Upload-Formular wurde gesendet
    &processPDF($r);

  } else {
    #erster Aufruf Upload-Formular wird ausgeben
    $r->print(&get_javascripts);
    $r->print(&get_uploadform);

  }

  return OK;
}
```

7.3.2 Hochladen der PDF-Formulare

Dateien können vom Web-Browser leicht über ein HTML-Formular zu einen Server gesendet werden. Dazu wird in *lonpdfupload.pm* ein HTML-Formular benutzt, das Formulardaten, wie in LON-CAPA üblich, per POST-Methode mit „multipart/formdata“-Kodierung sendet. Dieses Formular wird wieder an *lonpdfupload.pm* gesendet. Damit unterschieden

werden kann, ob eine Datei hochgeladen werden soll oder eine Datei gesendet wurde, müssen zunächst mögliche POST-Daten in der Serverumgebung bekannt gemacht werden. Dafür kann die Methode *&Apache::lonacc::get_posted_cgi($r)*[5] benutzt werden. Sie schreibt die POST-Daten in der Form *$env'form.Formularfeldname'* in die Serverumgebung. In der *handler()*-Methode wird überprüft, ob die Daten gemäß des Formulars zum Hochladen der PDF-Datei vorhanden sind. Bei erfolgreicher Prüfung werden die PDF-Formularfelder aus der hochgeladenen PDF-Datei ausgelesen.

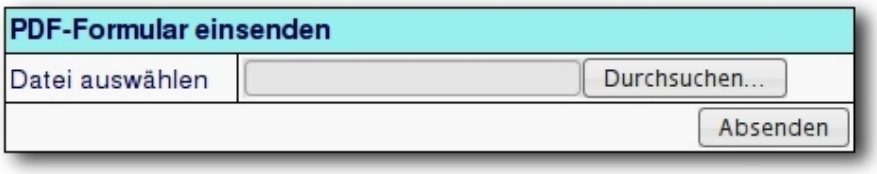

Abbildung 7.2: Formular zum Hochladen der PDF-Formulare

7.3.3 Auslesen der Antworten aus dem PDF-Formular

Das Auslesen der PDF-Formulardaten soll, bis eine geeignete Möglichkeit in Perl verfügbar ist, über eine Java-Anwendung *dumbPDF*, basierend auf dem jPod-Paket von Intarsys[6], realisiert werden. Um nicht zu weit in die Programmierung dieser Anwendung einsteigen zu müssen, soll hier nur die Funktionsweise beschrieben werden. Der dokumentierte Source-Code befindet sich auf der DVD-ROM zur vorliegenden Bachelorarbeit. Die Anwendung besteht aus einem Jar-Archiv, das über den Konsolenbefehl „java -jar dumpPDF.jar *Infile*[7] *Outfile*[8]“ ausgeführt wird. Von *dumpPDF* werden alle PDF-Felder, die einen Wert besitzen, in die für *Outfile* angegebene Datei geschrieben. Die Formularfelder werden in dieser Datei zeilenweise in der Form „Formularfeldname?Wert“ gespeichert. Um diese Java-Anwendung per Perl nutzen zu können, wird sie mit dem *system()*-Befehl aufgerufen. Dafür wurde ein neues Verzeichnis */home/httpd/pdfspool/* auf dem Server erstellt, in das das Jar-Archiv zum Ausführen von dumpPDF abgelegt wurde. In der Methode *get_pdf_data()* wird in *lonpdfupload.pm* das Auslesen der PDF-Formularfelder durchgeführt, was in drei Schritten geschieht. Zuerst wird die hochgeladene PDF-Datei aus der Serverumgebung als Datei ins

[5] $r steht für das Request-Objekt
[6] http://www.intarsys.de/
[7] Infile - steht für das PDF-Dokument mit Formulardaten
[8] Outfile - steht für die Datei mit den ausgelesenen PDF-Formulardaten

pdfspool-Verzeichnis kopiert. Anschließend wird per *system()* die dumpPDF.jar aufgerufen, die die Formulardaten aus der hochladenen PDF-Datei in eine Datei schreibt. Diese wird daraufhin in ein Array eingelesen, so dass in jeder Zelle des Arrays die Daten von einem PDF-Formularfeld stehen. Zuletzt werden die angelegten Dateien wieder gelöscht und das Array mit den PDF-Formulardaten zurückgegeben. Listing 7.12 zeigt den Sourcecode von *get_pdf_data()*.

Listing 7.12: *get_pdf_data()* aus *lonpdfupload.pm*

```
sub get_pdf_data() {
  my @data = ();
  my $file_path = "/home/httpd/pdfspool/".time."_".
                  int(rand(100000)).".pdf";
  my $file_data = $file_path;
     $file_data =~ s/\.pdf$/\.data/;

  # zwischenspeichern der hochgeladenen PDF
  my $temp_file = Apache::File->new('>'.$file_path);
  binmode($temp_file);
  print $temp_file $env{'form.file'};
  $temp_file->close;

  #Java PDF-Auslese-Programm starten
  my @command = ("java", "-jar",
                 "/home/httpd/pdfspool/dumpPDF.jar",
                 $file_path, $file_data);
  system(@command);


  #Einlesen der extrahierten Daten
  $temp_file = new IO::File->new('<'.$file_data);
  while (defined (my $line = $temp_file->getline())) {
    push(@data, $line);
  }
  $temp_file->close;
  undef($temp_file);

  #zwischengespeicherte Dateien loeschen
  if( -e $file_path) {
    unlink($file_path);
  }
  if( -e $file_data) {
```

```
    unlink($file_data);
  }
  return @data;
}
```

7.3.4 Bewerten der Aufgaben

Zur Bewertung müssen die ausgelesenen Antworten wieder in ihre einzelnen Bestandteile zerlegt werden. Dazu wird zuerst der Wert vom Formularfeldnamen getrennt. Anschließend wird der Formularfeldname in die Zuordungsdaten zur Aufgabe zerteilt. Die Adresse, an die die Antwort zur Bewertung gesendet werden muss, kann über die Methode *decode_symb()* aus dem Modul *lonnet.pm* bestimmt werden. Die Bezeichnung der HTML-Schaltfläche zum Senden einer Aufgabe per Web-Browser entspricht der Zeichenkette 'submit_', der die Bezeichnung des Aufgabenteils(*part*) angehangen wird. Diese Daten reichen bei Aufgaben mit einem Wert aus, um einen serverinternen Request zur Bewertung einer Aufgabe abzuschicken. Bei Aufgabentypen, die mehrere Antwortwerte zur Beantwortung benötigen, müssen zuerst alle 'HWVAL_'-Bezeichner inklusive ihres Werts gesammelt werden. Um diesen Unterschied nicht für jeden Aufgabentyp separat behandeln zu müssen, werden alle im PDF-Dokument enthaltenen Formularfelder nach *symb* und *part* in einen Hash sortiert. Dieser Hash enthält danach für jeden Aufgabenteil einen Eintrag auf einem untergeordneten Hash, der alle Informationen und alle Antwortwerte zur Beantwortung enthält. Beim Durchlaufen der PDF-Formularfelder werden auch alle eingefügten Benutzerdaten, die jeder Aufgabe in einem nicht sichtbaren Formularfeld vorangestellt werden können, überprüft. Stimmen die Benutzerdaten von PDF und Benutzerrolle nicht überein, wird der Bewertungsvorgang abgebrochen, ohne dass eine Aufgabe bewertet wurde. Das Listing 7.13 zeigt den Source-Code zur Überprüfung der Benutzerdaten und das Füllen des Hash *%problems* mit allen einzusendenden Aufgaben. Die ausgelesenen Daten des PDF-Formulars stehen dabei zeilenweise aufgeteilt im Array *@pdfdata*.

Listing 7.13: Sourcecode zur Sortierung der PDF-Formulardaten

```
foreach my $entry (sort(@pdfdata)) {
  if ($entry =~ /^meta.*/) {
      #Benutzerdaten prufen
      $debug .= 'found: metadata -> '.$entry;
      my ($label, $value) = split('\?', $entry);
      my ($domain, $user) = split('&', $value);
      $user =~ s/(.*)\n/$1/;
      if($user ne $env{'user.name'}
         or  $domain ne $env{'user.domain'}) {
          #Abbruch bei falschen Benutzerdaten
          return "<pre>".&mt('Wrong username in PDF-File').
                 ": $user $domain -> $env{'user.domain'}".
                 " $env{'user.name'} </pre>";
      }
  } elsif($entry =~ /^upload.*/)  {
      # PDF-Formularfeld einsortieren
      $debug .= 'found: a problem -> '.$entry;
      my ($label, $value) = split('\?', $entry);
      my ($symb, $part, $type, $HWVAL) = split('&', $label);
      my ($map,$id,$resource)=&Apache::lonnet::decode_symb($symb);
      $value =~ s/(.*)\n/$1/; #Zeilenumbruch entfernen
      if($type eq 'radiobuttonresponse' && $value eq 'Off' ) {
          #fehlerhafte Radiobuttons filtern (CABAReT Stage)
          next;
      }
      my $submit = $part;
      $submit =~ s/part_(.*)/submit_$1/;
      $problems{$symb.$part} =  { 'resource' => $resource,
                                  'symb' => $symb,
                                  'submitted' => $part,
                                  $submit => 'Answer',
                                  $HWVAL => $value};
  } else {
      # unbekannte Felder ignorieren
      next;
  }
}
```

Nachdem die Daten zu allen PDF-Formulardaten einsortiert sind, enthält jeder Eintrag von *%problems* einen Hash, der alle Informationen zur Bewertung einer Aufgabe beziehungsweise eines Aufgabenteils enthält. Die Informationen haben aufgrund der Sortierung das passende Format, um per serverinternen Request über die Web-Schnittstelle zur Bewertung eingereicht zu werden. Das geschieht über den bereits erwähnten Aufruf von *ssi_with_retries()*. Die zurückgegebene Anwort des Servers wird in der Variablen *$content* gespeichert. Mittels regulärem Ausdruck wird aus ihr der Bereich mit dem Aufgabentitel herausgefiltert. Um Aufschluss über die Bewertung zu erhalten wird nach dem Einreichen mit der Methode *restore()* aus *lonnet.pm* ein Hash (*%problemhash*)mit dem akutellen Status der Aufgabe abgerufen. Im Eintrag *$hashresource.(part).award* steht die Bewertung der Aufgabe. Die Titel und Bewertungen aller eingereichten Aufgaben werden in einer Tabelle formatiert und an den Web-Browser zurückgesendet. Listing 7.14 zeigt das Einreichen der Aufgaben und wie die Informationen zur Bewertung abgerufen werden. In der Abbildung 7.3 wird die Übersicht nach erfolgreichem Hochladen einer Aufgabe dargestellt.

Listing 7.14: Sourcecode der Rücksendung per serverseitigen Request

```
sub grade_problem {
  my %problem = @_;
  my ($content) =  &Apache::loncommon::ssi_with_retries('/res/'.
          $problem{'resource'}, 5, %problem);

  $content =~ s/.*class="LC_current_location".*>(.*)<\/td>.*/$1/g;
  $content = $1;

  my $part = $problem{submitted};
  $part =~ s/part_(.*)/$1/;
  $content .= " - Part $part";

  my %problemhash = &Apache::lonnet::restore($problem{'symb'});
  my $grade = $problemhash{"resource.$part.award"};
  return ($content, $grade);
}
```

Abbildung 7.3: Übersicht nach dem Hochladen eines PDF-Formulars

7.4 Übertragbarkeit auf andere Aufgabentypen

Die in diesem Kapitel beschriebene Implementierung zeigt auf, wie eine Bearbeitungsmöglichkeit von Radiobutton-Aufgaben durch PDF-Formulare realisiert werden kann. Dieses Vorgehen kann aufgrund der gleichen Struktur in der Verarbeitung beim Darstellen der Aufgaben sehr gut auf andere Aufgabentypen übertragen werden. Dazu müssen die Code-Zeilen, die für die Generierung des TEX-Codes zuständig sind, in den zugehörigen Response-Modulen analysiert und erweitert werden. Um festzustellen, welche Informationen zur erfolgreichen Rücksendung benötigt werden, kann die Bearbeitung über den Web-Browser, wie in Kapitel 5 aufgezeigt, analysiert werden. Grundsätzlich könnten alle einfachen Aufgaben, die mit HTML-Formularfeldern beantwortet werden, mit PDF-Formularen umgesetzt werden. Dazu könnten bei speziellen Aufgabentypen weitere Informationen in den PDF-Formularfeldern untergebracht werden, die bei der späteren Auslesung und Rücksendung separat behandelt werden.

8 Erreichte Ziele

Im Rahmen der Bachelorarbeit wurde aufgezeigt, dass eine Schnittstelle in LON-CAPA zur Bearbeitung von Aufgaben mittels PDF-Formularen realisiert werden kann. Dazu wurde das Druckmenü um einen Schalter zur Erstellung von PDF-Formularen erweitert. Dadurch wird es möglich, bei der Generierung von TEX-Code im Target 'tex' über die globale Variable *$env{'pdfFormFields'}* zu unterscheiden, ob eine Druck- oder eine Formularversion erstellt werden soll. Dabei wurde eine Möglichkeit gezeigt, wie die Daten zur späteren Zuordnung der bearbeiteten Aufgaben in den PDF-Formularfeldern untergebracht werden. Das beschriebene Vorgehen am Beispiel der Radiobutton-Implementierung kann dabei auf andere Aufgabentypen, die im Target 'web' über HTML-Formularfelder realisiert sind, übertragen werden. Zum Hochladen der bearbeiteten PDF-Formulare wurde ein neues Perl-Modul *lonpdfupload.pm* in LON-CAPA eingefügt. Dieses kann zunächst nur über die direkte Eingabe der über die Server-Konfiguration zugeordneten URL *Hostname/pdfuploa-d/*in der Adresszeile des Web-Browsers erreicht werden. Dieses Modul übernimmt dabei alle Funktionen, um die bearbeiteten Aufgaben aus einem PDF-Formular auszulesen und im System über die Web-Schnittstelle einzureichen. Aufgrund der komplexen Thematik konnte in der begrenzten Bearbeitungszeit dieser Abschlussarbeit eine Realisierung nur am Beispiel von Radiobutton-Aufgaben aufgezeigt werden. Dabei wurde besonderen Wert darauf gelegt, die einzelnen Schritte zur Umsetzung detailliert darzustellen, so dass für die Umsetzung der anderen Aufgabentypen die grundlegende Vorgehensweise nachempfunden werden kann. Die bisher umgesetzten Funktionen können nur einen Eindruck davon geben, wie die Aufgabenbearbeitung über PDF-Formulare zukünftig in LON-CAPA benutzt werden kann. Für den produktiven Einsatz sind jedoch noch einige weitere Anpassungen und Planungen nötig. Hier kann beispielsweise die Implementierung der weiteren Aufgabentypen, sowie eine genaue Abgrenzung, welche Funktionen durch PDF-Formulare im produktiven Einsatz ermöglicht werden sollen, gesehen werden.

9 Fazit

Im Hinblick auf die anfangs gesetzte Zielsetzung, LON-CAPA um eine Funktion zur Bearbeitung von Aufgaben per PDF-Formulare zu erweitern, hat sich ergeben, dass der Aufwand für eine produktiv einsetzbare Lösung in der Zeitspanne dieser Bachelorarbeit nicht umgesetzt werden kann. Aufgrund des komplexen Umfangs von LON-CAPA konnten nur die Bereiche analysiert werden, die unmittelbar mit der Generierung und der Einsendung von Aufgaben in Verbindung stehen. Dazu musste zunächst ein Grundverständnis der internen Arbeitsweise von LON-CAPA erlangt werden. Die in der Entwicklerdokumentation [AKH02] aufgeführten Informationen spiegelten oft nur die Grundidee der internen Verarbeitung wider, so dass zusätzlich mehrere tausend Zeilen Source-Code interpretiert werden mussten. Um diesen Aufwand für zukünftige Entwickler, die sich mit der Generierung und Bewertung von Aufgaben in LON-CAPA beschäftigen wollen, zu vereinfachen, habe ich versucht, auch meine Erkenntnisse zur internen Struktur und Verarbeitung an den passenden Stellen in dieser Abschlussarbeit einzubringen.

Die Erweiterung der Benutzerschnittstelle durch PDF-Formulare ist meiner Meinung nach eine sinnvolle Funktionalität, da unabhängig der im Rahmen der Arbeit erreichten Ziele, der Grundstein für viele neue Funktionalitäten gelegt wurde. Es wäre sicher in naher Zukunft möglich, die in den Anforderungen beschriebenen Szenarien durch die zusätzliche Erweiterung einer Email-Funktion umzusetzten. Man könnte die Benutzerdaten verschlüsselt in den PDF-Dokumenten integrieren, so dass die Emails ohne eine aktive Sitzung in LON-CAPA eingereicht werden können. Würde dann von LON-CAPA auch noch eine korrigierte Version des PDF-Dokuments per Email zurück an den Studierenden gesendet, wäre es sogar denkbar, dass Lehrende personifizierte Aufgabensammlungen für einen Kurs in LON-CAPA erstellen und die PDF-Dateien über andere Lernraumsysteme verteilen. Optimal wäre eine Schnittstelle zu anderen Lernraumsystemen, die die Aufgabensammlung direkt in den Kurs dieser Systeme einpflegt. Bevor aber die Umsetzung der oben genannten Funktionalität begonnen wird, sollten zunächst die anderen in der Arbeit nicht

umgesetzten Aufgabentypen über das Target 'tex' hinaus als PDF-Formularversion erstellt werden können.

Im Rahmen dieser Bachelorarbeit habe ich viel Fachwissen dazu gewonnen, welches mir mit großer Sicherheit im weiteren Verlauf meiner beruflichen Zukunft von Nutzen sein wird.

10 Anhang

Als Anhang liegt der vorliegenden Bachelorarbeit eine DVD-ROM bei auf der sich eine die geänderten Versionen der Perl-Module von LON-CAPA, die Java-Anwendung zum Auslesen der Daten aus PDF-Formularen, ein Teil der für die Bachelorarbeit herangezogenen Dokumente sowie eine elektronische Version dieser Abschlussarbeit als PDF.

Literaturverzeichnis

[ado] *Adobe Acrobat support.* http://www.adobe.com/de/support/toptech/acrobat/325874/, Abruf: 29.06.2008

[Ado06] *PDF Reference.* sixth edition: Adobe Portable Document Format version 1.7. Adobe Systems Incorporated, 2006

[AKH02] Albertelli, Guy ; Kortemeyer, Gerd ; Harrison, Scott: *The guts of LON-CAPA.* 2002 http://www.lon-capa.org

[Jür00] Jürgens, Manuela: *LaTeX- eine Einführung und ein bisschen mehr... /026/0003.* FernUniversität, 2000

[lon] *The LearningOnline Network with CAPA.* http://www.lon-capa.org, Abruf: 29.06.2008

[RO06] Rahtz, Sebastian ; Oberdiek, Heiko: *Hypertext marks in LaTeX: a manual for hyperref.* 2006

[SM99] Stein, Lincoln ; MacEachern, Doug: *Writing Apache Modules with Perl and C.* O'Reilly, 1999

[Sto07a] Story, D. P.: *AcroTeX Bundle eForm Support.* 2007 http://www.acroTeX.net

[Sto07b] Story, D. P.: *The AcroTeX eDucation Bundle.* 2007 http://www.acroTeX.net

[Sto07c] Story, D. P.: *AcroTeX eDucation Bundle Professional Enhanced AeB Features using Acrobat Pro.* 2007 http://www.acroTeX.net