Integration von E-Assessment und Content-Management

Michael Piotrowski, Dietmar Rösner

Otto-von-Guericke-Universität Magdeburg
Institut für Wissens- und Sprachverarbeitung
Postfach 4120
39016 Magdeburg
{mxp,roesner}@iws.cs.uni-magdeburg.de

Abstract: Formative Tests können für Lehrende und Lernende gleichermaßen nützlich sein. Webbasierte Multiple-Choice-Tests können helfen, den Aufwand für formative Tests zu senken und somit einen breiteren und häufigeren Einsatz zu ermöglichen. Wir stellen ein Modul für das Content-Management-System Plone vor, das es erlaubt, MC-Tests genau wie andere Ressourcen einzusetzen und zu verwalten. Auf diese Weise können vor allem in Präsenzveranstaltungen, für die üblicherweise keine Lernplattform verwendet wird, Tests eng mit den anderen online verfügbaren Lehr- und Lernmaterialien (z. B. Vorlesungsskripten oder Aufgabenblättern) verknüpft werden. Das Modul erlaubt auch den Import und Export von Aufgaben gemäß IMS QTI; in diesem Zusammenhang diskutieren wir auch unsere Erfahrungen mit dieser Spezifikation.

1 Einleitung

Tests und Prüfungen spielen in der Lehre eine wichtige Rolle. Dabei geht es heute nicht mehr nur um die Benotung am Ende einer Lehrveranstaltung (*summative Tests*), sondern Tests dienen auch der kontinuierlichen Verfolgung des Lernprozesses, ohne notwendigerweise in die Gesamtbewertung einzugehen (*formative Tests*). Formative Tests können Lernende motivieren, indem ihr Lernfortschritt sichtbar wird und sie Bereiche identifizieren können, in denen sie noch lernen müssen, um sich weiter zu verbessern.

Auch für Lehrende sind formative Tests von großem Nutzen, da sie ihnen zeigen, ob die Lernziele tatsächlich erreicht werden und wie erfolgreich sie die Lerninhalte präsentieren und wo evtl. noch eine intensivere Beschäftigung mit dem Material nötig ist. Formative Tests stellen somit ein Instrument zur Qualitätssicherung der Lehre dar.

Es ist deshalb wünschenswert, die Anwendung formativer Tests auszubauen. Elektronische Tests – auch als *e-assessment* bezeichnet – sind in diesem Zusammenhang besonders nützlich, da sie die häufige Durchführung von Tests erleichtern: Sie können die Kosten der Durchführung reduzieren, da kein Papier nötig ist und die Handhabung und Auswertung der abgegebenen Tests vereinfacht wird – bis hin zur automatischen Auswertung – und sie ermöglichen größere zeitliche und räumliche Flexibilität. Bei automatischer Testauswertung ermöglichen sie darüber hinaus die sofortige Rückmeldung an die Kandidaten.

Nicht alle Testtypen bieten sich gleichermaßen für die automatische Bewertung an: Am einfachsten ist eine automatische Auswertung bei Tests mit *gebundener Aufgabenbeantwortung*[1] möglich. Zu den Aufgabentypen mit gebundener Aufgabenbeantwortung gehören Richtig-Falsch-Aufgaben, Mehrfach-Wahl-Aufgaben, Zuordnungsaufgaben und Umordnungsaufgaben. Der wichtigste und am weitesten verbreitetste dieser Aufgabentypen ist die Mehrfach-Wahl- oder Multiple-Choice-Aufgabe.

Die automatische Bewertung von Aufsätzen ist dagegen vergleichsweise aufwendig (vgl. z. B. [BCL03, Mil03, SB03]). Im Bereich der Informatikausbildung ist noch die automatische Bewertung von Programmen von Interesse (vgl. z. B. [SMK01, HST02, RA05]).

Auf Bloom [BEFH76] geht eine Taxonomie von sog. *kognitiven Lernzielen* zurück, mit denen sich Lernaufgaben klassifizieren lassen. Die Bloomsche Taxonomie definiert die Kategorien *Wissen*, *Verstehen*, *Anwendung*, *Analyse*, *Synthese* und *Bewertung*. Nicht alle Lernziele lassen sich mit Hilfe von Multiple-Choice-Tests überprüfen, die Möglichkeiten von Multiple-Choice-Tests beschränken sich jedoch keineswegs nur auf die Überprüfung von Wissen; [CC98, Cla04] zeigen speziell für die Informatik weitergehende Möglichkeiten auf.

2 Webbasierte Multiple-Choice-Tests

Es gibt eine große Zahl von Systemen, die webbasierte Multiple-Choice-Tests (MCTs) realisieren. Die meisten dieser Systeme gehören zu einer der beiden folgenden Kategorien:

1. Lernplattformen (oft auch bezeichnet als *learning management system* oder *virtual learning environment*) mit integrierten Testmöglichkeiten. Beispiele für derartige Systeme sind WebCT[2], Blackboard[3], moodle[4] oder ILIAS[5].

2. Eigenständige Testsysteme, d. h., Systeme, die sich allein auf die Bereitstellung und Auswertung von Tests konzentrieren und ohne eine Lernplattform verwendet werden können. Beispiele sind Questionmark Perception[6], Hot Potatoes[7], Test Pilot[8] oder TOIA[9].

Der Hauptvorteil von Testmöglichkeiten, die in Lernplattformen integriert sind, ist sicherlich die Verknüpfung mit Kursen, die in demselben System verwaltet werden.

[1] Das Gegenstück ist die *freie Aufgabenbeantwortung*. Im Englischen wird von *selected-response tests* oder *fixed-responsed tests* (Gegensatz: *constructed response*) oder *objective tests* (Gegensatz: *essays*) gesprochen.

[2] http://webct.com

[3] http://blackboard.com

[4] http://moodle.org

[5] http://ilias.de

[6] http://questionmark.com/

[7] http://web.uvic.ca/hrd/halfbaked/

[8] http://clearlearning.com/

[9] http://www.toia.ac.uk/

Da die meisten Lernplattformen jedoch primär für Online-Kurse konzipiert sind, sind sie nicht unbedingt für Präsenzlehrveranstaltungen geeignet; der Aufwand für die Verwendung einer Lernplattform nur für ihre Testmöglichkeiten ist in den meisten Fällen nicht zu rechtfertigen.

In dieser Situation sind eigenständige Testsysteme interessant: Viele dieser Systeme bieten Möglichkeiten zur Anbindung an Lernplattformen, sie können aber auch ohne sie verwendet werden, wenn dafür kein Bedarf besteht.

Während jedoch Präsenzveranstaltungen nicht unbedingt eine Lernplattform verwenden, werden viele von ihnen durch elektronische Materialien ergänzt, wie z. B. Vorlesungsskripte oder Übungsaufgaben. Die meisten dieser Materialien sind statisch, denkbar sind jedoch auch etwa veranstaltungsbezogene Diskussionsforen. Sinnvollerweise sollten diese Materialien mittels eines *Content-Management-Systems* (CMS) verwaltet werden, um eine einheitliche, vom Inhalt getrennte, Präsentation und Zugriffssteuerung zu erhalten.

Wird aber ein CMS zur Verwaltung der statischen Lehr- und Lernmaterialien genutzt, wäre es ungünstig, noch ein Testsystem mit eigenem Webserver, eigener Datenbank, eigener Benutzerverwaltung usw. einrichten und verwalten zu müssen. Vielmehr wäre es wünschenswert, wenn Tests genau wie andere Inhaltstypen (Texte, Bilder usw.) im CMS verwaltet werden könnten. Dies hätte einige Vorteile

- für die Studierenden: Sie finden die Tests an derselben Stelle wie die anderen Veranstaltungsmaterialien, mit gleichem Erscheinungsbild und gleicher Bedienung.
- für die Testersteller: Sie müssen nicht die Benutzung eines weiteren Systems erlernen, wenn sie mit der Bedienung des CMS bereits vertraut sind.
- für die Systemadministration: Nur ein System muss verwaltet werden, insbesondere müssen auch die Benutzer in nur einem System verwaltet werden.

3 LlsMultipleChoice

Wir benutzen das Content-Management-System Plone für unsere Webseiten, die auch Lehr- und Lernmaterialien für die von unserer Arbeitsgruppe angebotenen Lehrveranstaltungen umfassen.

Plone[10] ist ein Open-Source-CMS, das auf dem Web-Application-Framework Zope[11] basiert, das ebenfalls quelloffen ist.

Zope und Plone sind in Python[12] geschrieben, einer weitverbreiteten und sehr portablen Open-Source-Programmiersprache. Dadurch können Zope und Plone auf einer sehr großen Zahl von Plattformen eingesetzt werden, darunter praktisch alle UNIX- und UNIX-ähnlichen Systeme (z. B. Linux, BSD-Varianten, Mac OS X) sowie Microsoft Windows.

[10] http://plone.org/
[11] http://zope.org/
[12] http://python.org/

Zope und Plone können durch *Produkte* genannte Module erweitert werden, die z. B. Inhaltsobjekte, Anbindungen an RDBMS und andere externe Datenquellen oder andere zusätzliche Funktionalität bereitstellen können.

Wir haben ein Plone-Produkt namens LlsMultipleChoice entwickelt und in Lehrveranstaltungen getestet, das die Integration von Multiple-Choice-Tests in das CMS ermöglicht.

LlsMultipleChoice ermöglicht zur Zeit die Erstellung, Präsentation und Auswertung von Multiple-Choice-Tests, die sowohl Fragen mit genau einer möglichen Antwort als auch Fragen mit mehreren auswählbaren Antworten (Mehrfach-Antwort-Aufgaben, *multiple-answer questions*) beinhalten können.

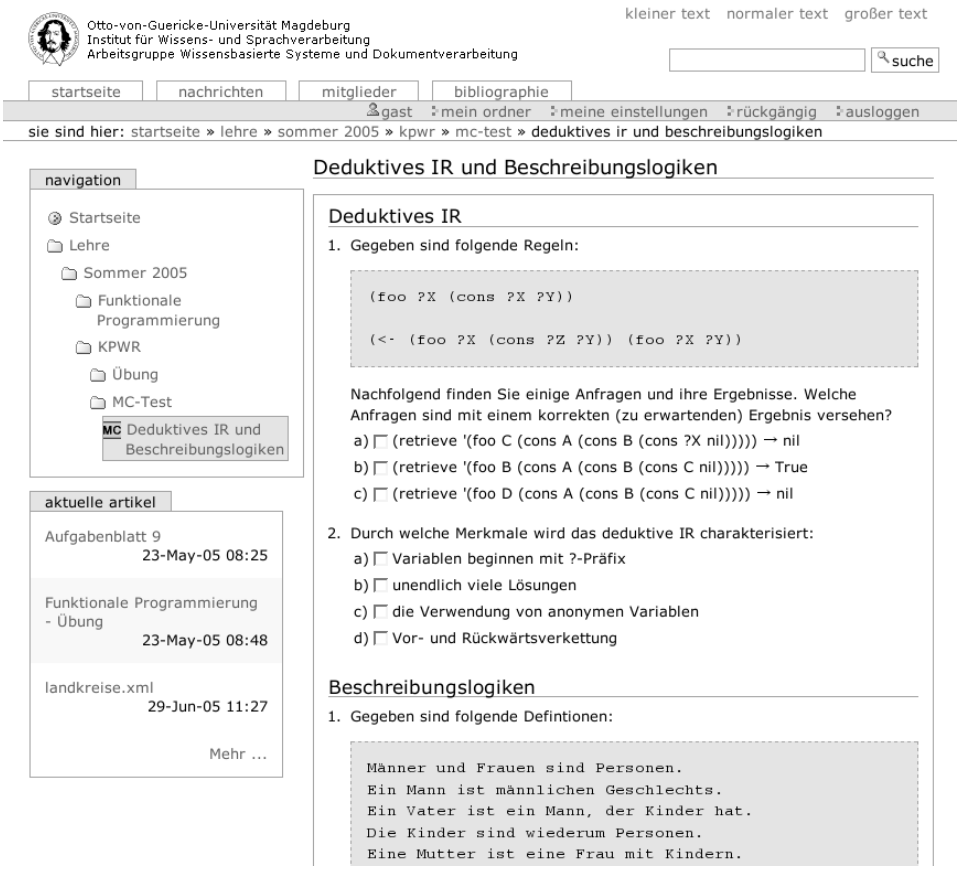

Abbildung 1: Kandidatensicht auf einen Test in LlsMultipleChoice

LlsMultipleChoice ist vollständig in Plone integriert: Benutzer, die mit Plone vertraut sind, finden sich schnell in LlsMultipleChoice zurecht, da sich Tests und ihre Bestandteile wie die anderen Inhaltsobjekte (Texte, Bilder, Termine, Ordner usw.) verhalten. Ein Test ist eine Art Ordner, der Frageobjekte enthält, die wiederum die dazugehörigen Antwortobjek-

te enthalten. Die testspezifischen Objekte sind von den allgemeinen Plone-Objektklassen abgeleitet. Auf diese Weise erben sie die grundlegende CMS-Funktionalität, wie etwa Zugriffssteuerung, zeitgesteuerte Veröffentlichung, Metadaten, volltext- und metadatengestützte Indizierung und Suche. Ebenso können z. B. auch Diskussionen über Testobjekte ermöglicht werden, was für die kollaborative Erstellung von Tests hilfreich sein kann. Zur Verwaltung der Testobjekte stehen alle Möglichkeiten des CMS zur Verfügung, so dass das CMS auch als Testsammlung (sog. *item bank*) dient.

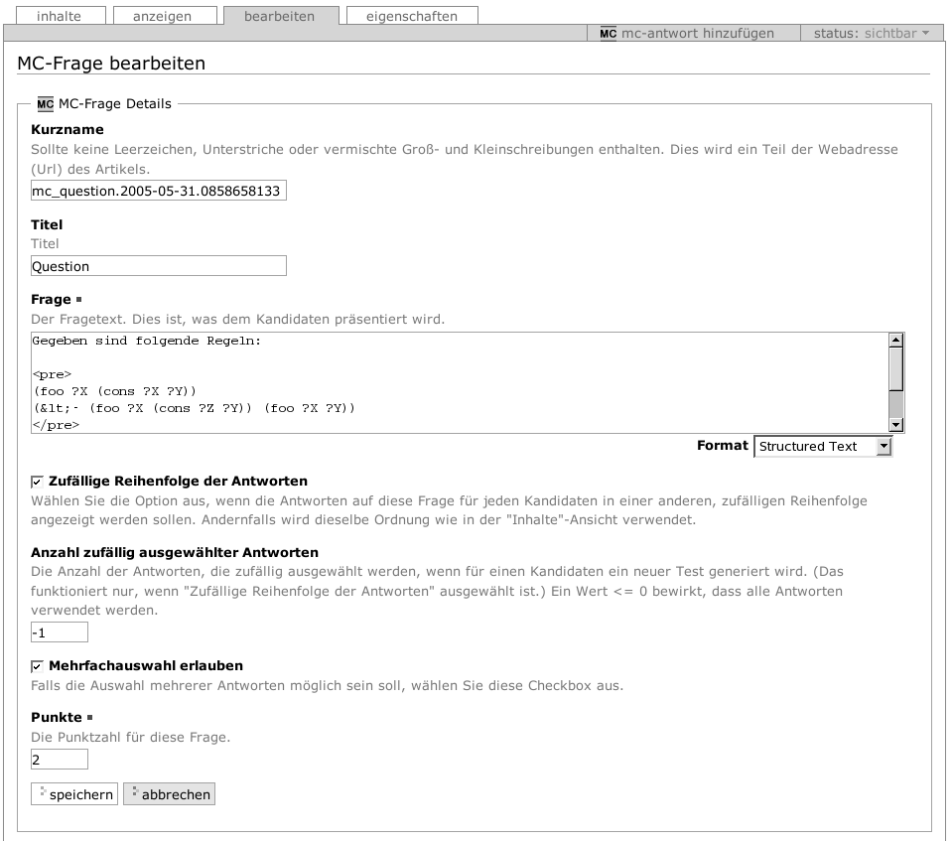

Abbildung 2: Bearbeiten einer Antwort in der Web-Oberfläche

Wichtige Eigenschaften von Testobjekten sind: Für Tests kann festgelegt werden, ob der Kandidat sofort eine Rückmeldung über seine Testergebnisse erhält (*Instant Feedback*) und ob ein Test mehrfach absolviert werden darf (z. B. für Selbsttests).

Fragen und Antwortmöglichkeiten können in fester oder zufälliger Reihenfolge angezeigt werden. Es ist auch möglich, sowohl nur eine zufällige Auswahl aller im Test vorhandenen Fragen zu präsentieren, als auch bei einzelnen Fragen nur eine Untermenge aller definierten Antwortmöglichkeiten anzubieten. Selbst wenn man Studierenden Multiple-Choice-

Tests nur zur Selbstkontrolle zur Verfügung stellt, ist es sinnvoll, diese zu randomisieren. Wenn auf Basis von Tests eine Vergabe von Leistungspunkten erfolgt, ist die Randomisierung nahezu zwingend. Bei einem nicht randomisierten Test wären sonst sehr schnell Aufstellungen mit den korrekten Antwortmöglichkeiten über die üblichen Austauschkanäle kommuniziert und würden den Test dann entwerten. Einen randomisierten Test korrekt zu beantworten, erfordert zumindest, genau die jeweiligen Antwortmöglichkeiten zu lesen.

Fragen und Antworten können beliebiges XHTML enthalten, einschließlich Bildern oder anderen Medienobjekten. Zusammengehörige Fragen können zu Fragegruppen zusammengefasst werden, die dann als Einheit behandelt werden können. So können z. B. innerhalb einer Fragegruppe andere Randomisierungseinstellungen als für den enthaltenden Test gelten.

Antworten können mit Kommentaren versehen werden – z. B., warum eine Antwort falsch oder richtig ist –, die den Kandidaten angezeigt werden, wenn die sofortige Rückmeldung von Testergebnissen aktiviert ist.

Neben der Auswahl vordefinierter Bewertungsmethoden ist es möglich, sogenannte *Bewertungsskripte* hochzuladen, um die Bewertung von Tests, Fragegruppen und Fragen an spezielle Bedürfnisse anzupassen. Ein Bewertungsskript ist ein Python-Programm, das eine Methode `getCandidatePointsCustom` definiert, die die ID des Kandidaten und eine Liste der gegebenen Antworten als Parameter nimmt.

```
def getCandidatePointsCustom(self, candidateId, givenAnswerIds):
  try:
    if(givenAnswerIds is None):
      return 0
    else:
      if(type(givenAnswerIds) != list):
        givenAnswerIds = [givenAnswerIds]

      shownAnswerIds = getParent(self).getAnswers(candidateId, self.UID())
      correctAnswerIds = self.getCorrectAnswerIds(candidateId)

      numShownAnswers = len(shownAnswerIds)
      numCorrectAnswers = len(correctAnswerIds)
      numWrongAnswers = numShownAnswers - numCorrectAnswers

      score = R = F = 0.0
      for givenAnswerId in givenAnswerIds:
        maxPoints = float(self.getPoints())
        if givenAnswerId in correctAnswerIds:
          R = R + maxPoints / numCorrectAnswers
        else:
          F = F + maxPoints / numWrongAnswers

      score = R - F
      return max(score, 0)
  except:
    return None
```

Listing 1: Beispiel eines Bewertungsskripts

Listing 1 zeigt ein Beispiel für ein Bewertungsskript, das eine einfache Bewertungsstrategie mit negativen Punkten für falsche Antworten implementiert: Wenn m die mögliche Punktzahl für eine Frage ist und R und F die Anzahl richtiger bzw. falscher vorgegebener Antworten sind, dann erhält der Kandidat für jede richtig gewählte Antwort m/R, für jede falsch gewählte Antwort wird dagegen m/F von der Punktzahl abgezogen.

Eine solche Bewertungsstrategie kann zur Zufallskorrektur dienen, d. h., um zu vermeiden, dass Kandidaten allein durch Raten eine hohe Punktzahl erreichen können – hierfür gibt es verschiedene Möglichkeiten. Die Vergabe negativer Punkte für falsche Antworten soll hier nur als Beispiel dienen, nicht als Empfehlung; [LR98, Bus99, SHMJB02, Joh03] (um nur einige Arbeiten zu nennen) diskutieren verschiedene Ansätze. Durch die Möglichkeit, prinzipiell beliebige Bewertungsstrategien zu implementieren, ist man bei LIsMultipleChoice nicht auf eine – möglicherweise im konkreten Fall ungeeignete – Strategie festgelegt.

Die Testergebnisse aller Kandidaten können in einer Übersicht angezeigt werden, von der aus man detailliertere Darstellungen der Einzelergebnisse aufrufen kann. Die Testergebnisse können außerdem für die weitere Auswertung in Tabellenkalkulations- oder Statistikprogrammen exportiert werden.

Die Oberfläche von LIsMultipleChoice ist vollständig internationalisiert, d. h., dass sie leicht für verschiedene Sprachumgebungen angepasst werden kann, und zwar erfolgt die Lokalisierung genau wie für Plone selbst. Wir haben zur Zeit deutsche und englische Texte.

Einzelne Aufgaben können im QTI-Format [IMS05] importiert und exportiert werden. Tests, d. h. Zusammenstellungen von Aufgaben, können als IMS Content Packages [IMS04] importiert und exportiert werden. Im folgenden Abschnitt wird darauf noch näher eingegangen. Aufgaben und Tests können somit entweder über die Web-Oberfläche (siehe Abbildung 2) in Plone oder offline in einem XML-Editor oder einem speziellen Editor für MC-Tests bzw. einem Package-Editor (wie z. B. dem RELOAD Editor[13]) erstellt werden.

4 Einige Anmerkungen zu QTI

Die IMS Question & Test Interoperability Specification (QTI) [IMS05] beschreibt ein Datenmodell und eine XML-Repräsentation für die Kodierung von Testfragen (sog. *assessment items*), siehe Listing 2. Das Ziel der Spezifikation ist es, den Austausch dieser Daten zwischen Autorenwerkzeugen, Testsammlungen (*item banks*), Lernplattformen und Testsystemen zu ermöglichen.

Da sich QTI (IMS Question & Test Interoperability Specification) Version 1.x inzwischen als weithin unterstützter Standard für den Austausch von Testfragen etabliert hat, stand bei der Entwicklung von LIsMultipleChoice von Anfang an fest, dass dieser Standard unterstützt werden soll. Da QTI Version 2.0 bereits im Entwurfsstadium war, haben wir uns die Unterstützung dieser Version für den Import und Export von Aufgaben zum Ziel gesetzt.

[13] http://www.reload.ac.uk/editor.html

QTI diente jedoch nicht als Entwurfsspezifikation, da zum einen die Version 2.0 zum Beginn der Entwicklung noch nicht fertiggestellt war und zum anderen QTI auf Grund der großen Zahl optionaler Elemente auch nicht als Entwurfsspezifikation geeignet ist.

Unser Ansatz war daher, zuerst die benötigte Funktionalität mittels eines geeigneten Modells zu implementieren und dann eine Abbildung in QTI vorzunehmen.

Das Ziel von QTI ist zwar, den Datenaustausch zwischen verschiedenen Systemen zu erleichtern, allerdings wird durch QTI alleine die Interoperabilität verschiedener Implementierungen noch nicht sichergestellt: Zum einen sind praktisch alle Elemente optional, zum anderen ist die Semantik mancher Elemente (oder ihrer Kombination) nicht vollständig definiert. Da die Spezifikation sehr umfassend ist, wird kaum ein System alle Teile vollständig implementieren; somit entsteht die Problematik, dass jede Implementierung ihre eigene Untermenge von QTI erzeugt und versteht, und dass auch die Interpretation eines in zwei Implementierungen benutzten Elements sich unterscheiden kann.

QTI definiert deshalb sogenannte *conformance profiles* [IMS05, Conformance Guide], d. h., Beschreibungen, welche Teile der Spezifikation von einer Implementierung unterstützt werden. Diese Profile können in XML-Form kodiert werden, so dass es zumindest prinzipiell möglich ist, automatisch festzustellen, ob eine bestimmte Frage von einem bestimmten System verarbeitet werden kann. Es gibt zwei vordefinierte Profile: *QTI-All* umfasst alle Möglichkeiten von QTI, während *QTI-Lite* durch Beschränkung auf die wichtigsten und am häufigsten verwendeten Möglichkeiten ein Minimalprofil darstellt.

Für LIsMultipleChoice haben wir QTI zunächst soweit implementiert, um einen *round trip* zu ermöglichen, d. h., dass von LIsMultipleChoice exportierte Dateien ohne Verlust von Informationen wieder importiert werden können.

Da LIsMultipleChoice alle in QTI-Lite definierten Möglichkeiten bietet, ist als nächster Schritt ein Feinabgleich mit QTI-Lite vorgesehen, um – im Hinblick auf die bestmögliche Interoperabilität – möglichst genau diesem Profil zu entsprechen.

Im Gegensatz zu QTI 1.x deckt jedoch QTI 2.0 bislang nur einzelne Fragen ab und lässt die Teile von QTI 1.x aus, die sich mit der Aggregation von Fragen in Abschnitte und Tests beschäftigten. [14]

Da LIsMultipleChoice jedoch sowohl komplette Tests als auch Gruppen von zusammengehörigen Fragen unterstützt, musste ein Weg gefunden werden, diese Strukturen in einer standardkonformen und portablen Weise zu beschreiben.

QTI Integration Guide und QTI Migration Guide schneiden einige dieser Aspekte an, es bleiben jedoch viele Fragen bezüglich der konkreten Umsetzung offen, wie das folgende Zitat zeigt:

> As this version of the QTI specification does not define either an information model or a binding for section, assessment and objectbank objects no recommendations on how to interpret collections of packaged version 2 items are made. However, packaged items may be referred to individually in

[14] Die Beschreibung von Tests ist für QTI Version 2.1 vorgesehen [Lay05].

```
<?xml version="1.0"?>
<!DOCTYPE assessmentItem SYSTEM "imsqti_v2p0.dtd">

<assessmentItem identifier="EX1" title="Brötchen"
                adaptive="false" timeDependent="false">
 <responseDeclaration identifier="R-EX1" cardinality="single">
  <correctResponse><value>choice2</value></correctResponse>
 </responseDeclaration>

 <itemBody>
  <choiceInteraction responseIdentifier="R-EX1"
                     shuffle="true" maxChoices="1">
   <prompt>7 Brötchen kosten 3,15 DM. Was kosten 11 Brötchen?</prompt>
   <simpleChoice identifier="choice1">5,05 DM</simpleChoice>
   <simpleChoice identifier="choice2">4,95 DM</simpleChoice>
   <simpleChoice identifier="choice3">4,85 DM</simpleChoice>
   <simpleChoice identifier="choice4">4,75 DM</simpleChoice>
   <simpleChoice identifier="choice5">4,65 DM</simpleChoice>
  </choiceInteraction>
 </itemBody>
</assessmentItem>
```

Listing 2: Beispiel eines nach QTI V. 2.0 ausgezeichneten *Assessment Item*

> an associated learning design or set of sequencing rules. [IMS05, Integration Guide]

Sie machen auch deutlich, dass die Integration der verschiedenen IMS-Spezifikationen noch nicht wirklich optimal ist:

> IMS Learning Design and IMS QTI are natural partners in the learning process. [...] However, the type systems used in IMS LD and IMS QTI differ: [...] A final complicating factor is the presence of multi-valued variables in QTI which have no equivalent in IMS LD. [IMS05, Integration Guide]

Wir haben den im Folgenden beschriebenen Ansatz gewählt.

Eine Frage mit ihren zugehörigen Antworten wird gemäß QTI 2.0 in ein `assessmentItem` und damit in eine eigene Datei abgebildet.

Die Zusammenstellung der Fragen zu einem Test erfolgt nach der IMS Content Packaging Specification (CP) [IMS04]. *Packaging* bedeutet hier, dass alle Dateien zusammen mit einem *Manifest* in ein ZIP-Archiv gepackt werden. Das Manifest, das die im Archiv enthaltenen Ressourcen beschreibt, ist eine XML-Datei mit dem Namen `imsmanifest.xml` im Wurzelverzeichnis des Archivs.

Unser Modell sieht vor, dass sowohl Tests als auch Fragegruppen vorangestellte Bearbeitungshinweise enthalten können, was weder in QTI noch in CP explizit vorgesehen ist. Wir behandeln diese Hinweise als `assessmentItem` ohne Interaktion. Auf diese Weise lässt sich die Erweiterung konform modellieren.

Außerdem kann im Element `organization` des Manifests die Struktur des enthaltenen Materials beschrieben werden. Auf diese Weise können Fragegruppen repräsentiert werden und die Bearbeitungshinweise den entsprechenden Einheiten zugeordnet werden.

Die Randomisierung der Antworten innerhalb einer Frage wird von QTI abgedeckt, LlsMultipleChoice unterstützt jedoch auch die Randomisierung von Fragen, einschließlich der zufälligen Auswahl einer Untermenge der vorhandenen Fragen. Das Verhalten kann in Fragegruppen separat eingestellt werden. Um diese Eigenschaften zu beschreiben, greifen wir auf die IMS Simple Sequencing Specification [IMS03] zurück. Diese Spezifikation definiert Elemente, mit denen die Abfolge von Lernobjekten beschrieben werden kann. Diese Elemente können im `organization`-Element des Manifests verwendet werden. Wir benutzen sie, um die Randomisierung von Elementen, die Anzahl erlaubter Versuche und die zeitliche Freigabe von Tests zu beschreiben.

5 Verwandte Arbeiten

Im Wintersemester 2003/2004 haben wir ein selbstentwickeltes, eigenständiges System für webbasierte MC-Tests in der Veranstaltung »Programmierkonzepte und Modellierung« eingesetzt. Die Erfahrungen damit waren grundsätzlich positiv, jedoch stellte sich die fehlende Integration mit den anderen Ressourcen (z. B. Vorlesungsskripten oder Aufgabenblättern) als ein deutliches Hindernis dar.

Bevor wir mit der Entwicklung von LlsMultipleChoice begonnen haben, haben wir die Zope-Produkte Exam[15], XQuizz[16] und Survey[17] evaluiert. Diese Produkte sind jedoch zum einen nicht in Plone integriert, zum anderen fehlen ihnen wichtige Funktionen.

Kürzlich sind zwei Plone-Produkte veröffentlicht worden, die einen Ansatz ähnlich dem unseren verfolgen: eXam[18] und LTOnlineTest [19]. Beide Produkte scheinen sich noch in einem recht frühen Entwicklungsstadium zu befinden. Eduplone eXam bietet zur Zeit mehr Fragetypen als LlsMultipleChoice, ansonsten fehlen beiden Produkten jedoch einige der oben beschriebenen Eigenschaften von LlsMultipleChoice (darunter Internationalisierung, QTI-Import und -Export, benutzerdefinierbare Bewertungsskripte und die Randomisierung von Fragen und Antworten).

6 Zusammenfassung und Ausblick

LlsMultipleChoice ist noch sehr neu, so dass wir Erfahrungen bislang nur in geringem Umfang sammeln konnten; wir haben es jedoch im Wintersemester 2004/2005 bereits erfolgreich wieder in »Programmierkonzepte und Modellierung« eingesetzt und verwen-

[15] `http://www.zope.org/Members/J.A.R.Williams/exam`
[16] `http://zope.org/Members/gillou/XQuizz`
[17] `http://zope.org/Members/jwashin/Survey/`
[18] `http://www.janus-projekte.de/exam/`
[19] `http://lawtec.net/projects/ltonlinetest/`

den es in den Lehrveranstaltungen des laufenden Semesters (Sommer 2005). Der Ansatz, Testmöglichkeiten als zusätzlichen Objekttyp in ein allgemeines (d. h., nicht e-learning-spezifisches) CMS zu integrieren, fügt sich sehr gut in unsere Lehrumgebung ein und stellt eine stabile Infrastruktur für die Erstellung und Durchführung von Tests dar. Bei der Entwicklung hat sich Plone als sehr leistungsfähige Entwicklungsumgebung gezeigt, auch wenn die Entwicklerdokumentation teilweise noch Lücken aufweist.

Zur Zeit arbeiten wir an der Verallgemeinerung des QTI-Imports und -Exports und weiteren Testtypen, einschließlich Short-Answer-Aufgaben und manuell korrigierter Fragen für Aufsatzaufgaben. Außerdem werden wir jetzt selbstverständlich LlsMultipleChoice verstärkt in den praktischen Lehrbetrieb integrieren.

Es ist wichtig sich daran zu erinnern, dass die größte Herausforderung nicht die Durchführung von Tests ist, sondern die Erstellung qualitativ hochwertiger Tests. Wir sehen die Hauptaufgabe elektronischer Tests nicht im Ersatz traditioneller Methoden, sondern darin, Lehrenden mit einfach zu nutzenden Werkzeugen den breiteren und häufigeren Einsatz von vor allem formativen Tests zu ermöglichen.

Wir stellen LlsMultipleChoice als quelloffene Software kostenfrei unter `http://wwwai.cs.uni-magdeburg.de/sw/llsmc/` zur Verfügung.

Angaben zur Förderung

Die Arbeiten an LlsMultipleChoice sind Teil des vom Land Sachsen-Anhalt geförderten Projekts *XML-Technologie zur Unterstützung der Entwicklung und Wiederverwendung von Lehr- und Lernmaterialien* (Förderkennzeichen: 0047M1/0002A) .

Danksagungen

Wir danken Herrn Wolfram Fenske für die Implementierung von LlsMultipleChoice und für seine Beiträge zum konzeptuellen Entwurf.

Literatur

[BCL03] Jill Burstein, Martin Chodorow und Claudia Leacock. Criterion: Online essay evaluation: An application for automated evaluation of student essays. In *Proceedings of the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence*, Acapulco, Mexico, Aug 2003.

[BEFH76] Benjamin S. Bloom, Max D. Engelhart, Edward J. Furst und Walker H. Hill. *Taxonomie von Lernzielen im kognitiven Bereich*. Beltz, Weinheim, 5. Auflage, 1976.

[Bus99] Martin Bush. Alternative marking schemes for online multiple-choice tests. In *Proceedings of the 7th Annual Conference on the Teaching of Computing*, Belfast, 1999.

[CC98] Kevin Cox und David Clark. The Use of Formative Quizzes for Deep Learning. *Computers & Education*, 30(3/4):157–167, 1998.

[Cla04] David Clark. Testing Programming Skills with Multiple Choice Questions. *Informatics in Education*, 3(2):161–178, 2004.

[HST02] Colin Higgins, Pavlos Symeonidis und Athanasios Tsintsifas. The marking system for CourseMaster. In *ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education*, Seiten 46–50. ACM Press, 2002.

[IMS03] IMS Global Learning Consortium. *IMS Simple Sequencing Specification Version 1.0*. 2003.

[IMS04] IMS Global Learning Consortium. *IMS Content Packaging Specification Version 1.1.4*. 2004.

[IMS05] IMS Global Learning Consortium. *IMS Question and Test Interoperability Version 2.0 Final Specification*. 2005.

[Joh03] Alex Johnstone. *Effective Practice in Objective Assessment*. LTSN Physical Sciences Centre, 2003.

[Lay05] Steve Lay. What's new in IMS QTI v2.0? Presentation at the 15th CETIS Assessment SIG meeting, University of York, 26 Jan 2005.

[LR98] Gustav A. Lienert und Ulrich Raatz. *Testaufbau und Testanalyse*. Psychologie Verlags Union, Weinheim, 6. Auflage, 1998.

[Mil03] Tristan Miller. Essay Assessment with Latent Semantic Analysis. *Journal of Educational Computing Research*, 28(3), 2003.

[RA05] Dietmar Rösner und Mario Amelung. A Web-Based Environment to Support Teaching of Programming Paradigms. In *Proceedings of the 4th IASTED International Conference on Web-based Education (WBE 2005), February 21–23, 2005, Grindelwald, Switzerland*, Seiten 655–660. IASTED, ACTA Press, 2005.

[SB03] Mark D. Shermis und Jill Burstein, Hrsg. *Automated Essay Scoring: A Cross Disciplinary Perspective*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2003.

[SHMJB02] Gary A. Schaeffer, Dianne Henderson-Montero, Marc Julian und Nancy H. Bené. A Comparison of Three Scoring Methods for Tests With Selected-Response and Constructed-Response Items. *Educational Assessment*, 8(4):317–340, Dec 2002.

[SMK01] Riku Saikkonen, Lauri Malmi und Ari Korhonen. Fully automatic assessment of programming exercises. In *ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education*, Seiten 133–136. ACM Press, 2001.