Interoperabilität von elektronischen Tests

Michael Piotrowski, Wolfram Fenske

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

Abstract: Die Erstellung qualitativ hochwertiger Tests ist aufwändig. Daher ist es wünschenswert, einmal erstellte Tests wieder- und weiterverwenden zu können. Um eine Abhängigkeit von einer einzelnen Testplattform zu vermeiden, werden standardisierte Austauschformate benötigt. In diesem Beitrag formulieren wir Desiderata für derartige Formate und untersuchen den derzeitigen De-Facto-Standard, die IMS Question & Test Interoperability Specification (QTI), auf seine Eignung. Das erklärte Ziel von QTI ist es, den Austausch von Tests zwischen verschiedenen Systemen zu ermöglichen. Nach der Analyse der Spezifikation und aufgrund unserer Erfahrungen bei der Implementierung von QTI im System „ECQuiz“ kommen wir zu dem Schluss, dass QTI jedoch als Austauschformat ungeeignet ist.

1 Einleitung

Formative Tests tragen entscheidend zur kontinuierlichen Verfolgung des Lernprozesses bei. Webbasierte objektive[1] Tests (wie Multiple-Choice-Tests) sind in diesem Zusammenhang besonders nützlich, da sie schnell und häufig durchgeführt und zeitlich und räumlich flexibel eingesetzt werden können.

Die Erstellung qualitativ hochwertiger Tests ist jedoch aufwändig. Schon alleine aufgrund des damit verbundenen zeitlichen und personellen Aufwands ist es wünschenswert, dass einmal erstellte Tests wieder- und weiterverwendbar bleiben. Dies gilt sowohl für den Fall, dass sich die verwendete Testplattform ändert als auch für den Austausch mit anderen Lehrenden.

Um dies zu ermöglichen, wird ein Austauschformat benötigt, also eine Repräsentation von Tests, die von verschiedenen Systemen geschrieben und gelesen werden kann. Im Kontext von Tests handelt es sich bei den Systemen üblicherweise um Lernplattformen mit integrierten Testmöglichkeiten (z. B. Blackboard, ILIAS, Moodle, OLAT oder WebCT) oder um eigenständige Testsysteme (z. B. Hot Potatoes, Questionmark Perception oder Test Pilot).

[1] Im Sinne der Testtheorie, d. h., Antwortalternativen sind eindeutig richtig oder falsch (vgl. [LR98]) und somit automatisch überprüfbar.

In diesem Beitrag gehen wir zunächst auf einige Aspekte ein, die unseres Erachtens für den Austausch von Tests relevant sind und definieren Anforderungen an Austauschformate für Tests. Wir berichten über unsere Erfahrungen mit der IMS Question & Test Interoperability Specification (QTI) in dem von uns entwickelten System „ECQuiz“. Es folgt eine kritische Betrachtung von QTI.

2 Desiderata für Austauschformate

Ob ein (direkter) Datenaustausch zwischen Systemen möglich ist, hängt zunächst davon ab, ob es ein gemeinsames Datenformat gibt. Das alleine ist aber in der Praxis leider noch keine Garantie für einen reibungslosen Datenaustausch. Vielmehr hängt die Zuverlässigkeit des Austauschs fast unmittelbar von der Spezifikation des Austauschformats ab. Ist das Austauschformat unzureichend spezifiziert, ist es möglich – und sogar wahrscheinlich –, dass es von verschiedenen Entwicklern auf verschiedene Weise interpretiert und implementiert wird; auch wenn alle Varianten dabei prinzipiell der Spezifikation entsprechen, kann ein Datenaustausch dennoch unmöglich sein.

Weiterhin kann man davon ausgehen, dass mit der Komplexität der Spezifikation auch die Zahl der Fehler in den Implementierungen wächst. Schreibt oder liest ein System das Austauschformat nicht korrekt, ist es wiederum wahrscheinlich, dass der Datenaustausch gar nicht oder nur fehlerhaft funktioniert. Dabei ist die Situation, dass ein Datenaustausch zwar stattfindet, die Daten aber z. B. vom Zielsystem nicht richtig interpretiert werden, potentiell gefährlicher, als wenn der Datenaustausch von vornherein nicht möglich ist: Die dabei entstehenden Fehler können lange unentdeckt bleiben und im Falle von Tests etwa zu einer falschen Bewertung der Kandidaten führen.

Formale Definition

Aus diesen Überlegungen ergeben sich eine Reihe von Anforderungen an Austauschformate für Tests. Zunächst sollten sich Standards – oder als solche intendierte Spezifikationen – soweit wie möglich auf bereits vorhandene Standards stützen. Im Bereich von Datenformaten bedeutet dies heutzutage die Verwendung von XML [Wo06]. Auf diese Weise kann die Spezifikation des eigentlichen Austauschformats kompakt gehalten werden und es kann bei der Implementierung auf bereits vorhandene und insbesondere bereits getestete Werkzeuge zurückgegriffen werden.

Die Verwendung von XML in einer Spezfikation ist jedoch nur dann sinnvoll, wenn auch eine formale Definition in einer Schemasprache (Relax NG [ISO03] oder W3C XML Schema [Wo04]) erfolgt. Die Möglichkeiten der Schemasprachen sollten dabei voll ausgeschöpft werden, um die Konformität möglichst weitgehend bereits durch einen XML-Parser sicherstellen zu können; natürlichsprachliche Einschränkungen und Anforderungen, sind dagegen weit schwerer automatisch zu überprüfen, somit fehlerträchtig und sollten daher vermieden werden. Durch ein Schema wird die Syntax beschrieben. Für eine Implementierung, die das Format schreiben oder lesen soll, muss darüberhinaus auch die

Semantik, also die Bedeutung der einzelnen Elemente, genau spezifiziert sein, so dass Tests auch tatsächlich in der vom Ersteller intendierten Form übertragen werden.

Trennung von Inhalt und Form

Beim Austausch und bei der Wiederverwendung von Tests gibt es verschiedene Szenarien. In manchen Fällen sollen Tests vollständig übernommen werden, während in anderen Fällen z. B. nur einzelne Fragen aus einem Test oder einer Aufgabensammlung („item bank“) in einen anderen Test integriert werden sollen. Daher ist es wichtig, die verschiedenen Aspekte von Tests – insbesondere Inhalt, Erscheinungsbild und Verhalten – klar voneinander zu trennen, so dass etwa der Austausch von Inhalten nicht dadurch behindert wird, dass Inhalt und Erscheinungsbild miteinander vermischt sind.

Umfang

Um vollständige Tests mit allen ihren Eigenschaften austauschen zu können, erscheint es zunächst wünschenswert, dass ein Austauschformat die gesamte Funktionalität aller Systeme abbilden kann. Bei näherer Betrachtung wird jedoch klar, dass diese Anforderung illusorisch ist: Zu vielfältig und zu verschieden sind die Möglichkeiten von Testsystemen, wobei kein System alle Testtypen und Auswertungsfunktionen unterstützt.

Daher sollte sich ein Austauschformat zunächst auf einen relativ kleinen Kern von Test- und Fragetypen beschränken; weitere Typen können in späteren Versionen standardisiert werden, wenn klar ist, welche Beschreibungsmöglichkeiten in der Praxis tatsächlich benötigt werden. Die Aufnahme von optional zu implementierenden Teilen oder mehrerer Alternativen für eine Funktion ist dagegen zu vermeiden, da sich gezeigt hat, dass dies die Entwicklung interoperabler Implementierungen stark behindert. [2]

Langlebigkeit

Schließlich sollte ein Austauschformat langlebig sein, d. h., im Austauschformat beschriebene Tests sollten über einen möglichst langen Zeitraum verarbeitbar bleiben. Zur Korrektur von Fehlern und zur Erweiterung der Beschreibungsmöglichkeiten werden immer wieder neue Revisionen des Formats notwendig sein. Hierbei sollte jedoch soweit wie möglich vermieden werden, dass neue Revisionen zu Problemen beim Datenaustausch führen.

---

[2] Ein gutes Beispiel dafür sind SGML und XML. SGML enthält eine Vielzahl optionaler Teile: Trotz des nunmehr 20jährigen Bestehens von SGML gibt es bisher keinen Parser, der den Standard vollständig implementiert. XML ist eine Untermenge von SGML, bei der auf alle optionalen Teile verzichtet wurde: In kürzester Zeit waren eine große Zahl von konformen und interoperablen Implementierungen verfügbar und XML fand eine praktisch universelle Verbreitung.

3 IMS QTI: Überblick

Für die Beschreibung von Multiple-Choice-Tests und verwandten Testtypen ist die IMS Question & Test Interoperability Specification (QTI) [IMS05] zur Zeit die einzige öffentliche, von einer Implementierung unabhängige Spezifikation. Darüberhinaus kann das IMS-Konsortium im E-Learning-Bereich als De-Facto-Standardisierungsinstanz betrachtet werden.

QTI beschreibt ein Datenmodell und eine XML-Repräsentation für die Kodierung von Testfragen (sog. „assessment items“) bzw. Tests. Das erklärte Ziel der Spezifikation ist es, den Austausch dieser Daten zwischen Autorenwerkzeugen, Aufgabensammlungen, Lernplattformen und Testsystemen zu ermöglichen; QTI ist also als Austauschformat gedacht.

QTI Version 1.0 wurde im Jahr 2000 veröffentlicht und mehrfach überarbeitet. Wenn bei Systemen „Unterstützung für QTI“ angegeben ist (beispielsweise bei Respondus, WebCT oder OLAT), ist in den meisten Fällen damit eine Version von QTI 1.x gemeint. Eine kurze Beschreibung von QTI 1.0 durch Mitglieder der QTI-Arbeitsgruppe bietet [SR00].

Im Einsatz zeigten sich jedoch grundsätzliche Mängel in QTI 1.x, so dass die QTI-Arbeitsgruppe einen kompletten Neuentwurf für nötig erachtete. Dieser Neuentwurf ist die Version 2.0, die 2005 veröffentlicht wurde. Entgegen den Zusagen weitgehender Kompatibilität durch IMS – „software that is compliant with the V1.0 DTD will be able to import V2.0 Items providing it ignores the optional tags“ [SR00] – verwendet QTI 2.0 ein grundsätzlich anderes Modell und eine vollkommen andere XML-Struktur und ist mit QTI 1.x nicht kompatibel. Darüberhinaus deckt QTI 2.0 nicht alle Bereiche ab, die in QTI 1.x verfügbar waren; so können etwa mit QTI 2.0 nur einzelne Items, aber keine kompletten Tests beschrieben werden. Im Folgenden gehen wir auf QTI 2.0 ein; dies ist z. Z. die neueste offizielle Version der Spezifikation.

Die QTI-Spezifikation besteht aus mehreren Teilen. Im Teil „Information Model“ wird zunächst ein abstraktes Datenmodell beschrieben. Hier wird beispielsweise behandelt, was eine Frage ist und über welche Attribute sie verfügt. Das „XML Binding“ definiert dann eine Abbildung dieses Modells in eine konkrete XML-Repräsentation. Für die XML-Repräsentation werden ein W3C XML Schema und eine DTD definiert. Die weiteren Teile der Spezifikation beschäftigen sich mit verschiedenen Teilaspekten und geben Hinweise zur Implementierung und Nutzung von QTI.

Das grundlegende Element von QTI 2.0 ist das Item, also eine Frage mit den dazugehörigen Antwortmöglichkeiten. Für die Auszeichnung des Iteminhalts wird dabei eine Untermenge von XHTML verwendet, die um testspezifische Elemente ergänzt wird.

Abbildung 1 zeigt ein einfaches Beispiel für ein in QTI 2.0 kodiertes Item, in dem eine Multiple-Choice-Frage mit Mehrfachwahl definiert wird. Das Element <itemBody> enthält dabei die eigentliche Frage (Element <prompt>) und die Antwortmöglichkeiten. Da der Kandidat mit den Antworten „interagieren“ kann, wird dieser Teil in QTI als „Interaction“ bezeichnet. Im Beispiel soll eine Auswahl getroffen werden, daher wird das Element <choiceInteraction> verwendet, das in den <simpleChoice>-Elementen die Antwortmöglichkeiten enthält. Welche der Antwortmöglichkeiten korrekt sind, wird im Element <correctResponse> am Anfang der Datei festgelegt.

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

<assessmentItem identifier="EX1" title="Formel-1"
                adaptive="false" timeDependent="false">
 <responseDeclaration identifier="R-EX2" cardinality="multiple">
  <correctResponse>
   <value>choice1</value>
   <value>choice2</value>
  </correctResponse>
 </responseDeclaration>

 <itemBody>
  <choiceInteraction responseIdentifier="R-EX2"
                     shuffle="true" maxChoices="0">
   <prompt>In welchen Jahren wurde Michael Schumacher
    Formel-1-Weltmeister?</prompt>
   <simpleChoice identifier="choice1">1994</simpleChoice>
   <simpleChoice identifier="choice2">2000</simpleChoice>
   <simpleChoice identifier="choice3">2006</simpleChoice>
  </choiceInteraction>
 </itemBody>
</assessmentItem>
```

Abbildung 1: Einfaches Beispiel für eine QTI-2.0-Datei

4 Beispiel: QTI 2.0 in „ECQuiz“

„ECQuiz“[3] [PR05] ist ein Modul der von uns entwickelten eduComponents [RPA07], das die Integration von Multiple-Choice-Tests in das freie Content-Management-System Plone[4] ermöglicht. Für den Import und Export von Tests haben wir in „ECQuiz“ eine Untermenge von QTI 2.0 implementiert und dabei Erfahrungen mit diesem Standard gesammelt, die die Grundlage für diesen Artikel bilden. Wir beschreiben im Folgenden den konkreten Einsatz von QTI in „ECQuiz“ und unsere Erfahrungen bei der Implementierung.

Da es zu Beginn der Entwicklung bereits konkrete Anforderungen gab, welche Funktionen „ECQuiz“ bereitstellen sollte, war unser Ansatz, zuerst die benötigte Funktionalität mittels eines geeigneten Modells zu implementieren und diese dann für den Import und Export auf QTI 2.0 abzubilden.

Das kleinste Element im Modell von „ECQuiz“ ist eine Frage. „ECQuiz“ bietet z. Z. zwei grundsätzliche Fragetypen: Multiple-Choice-Fragen, bei denen die Kandidaten aus mehreren Antwortmöglichkeiten auswählen müssen, und Textfragen, bei denen eine freie Antwort formuliert werden muss. Mehrere Fragen, die sich z. B. auf dieselbe Textpassage oder dasselbe Bild beziehen oder anderweitig inhaltlich verwandt sind, können in einer Fragegruppe zusammengefasst werden. Der gemeinsame Inhalt wird in den sogenannten „Bearbeitungshinweisen“ der Fragegruppe abgelegt, vgl. Abb. 2. Mehrere Fragen und Fra-

[3] Frühere Versionen hießen „LlsMultipleChoice“.
[4] `http://www.plone.org/`

Abbildung 2: Beispiel eines Tests (hier in der Ergebnisansicht) in „ECQuiz“ mit Fragegruppen (❶ und ❷) und den dazugehörigen Bearbeitungshinweisen (❸ und ❹).

gegruppen bilden schließlich einen Test. Analog zu den Fragegruppen können auch für den Test als ganzes Bearbeitungshinweise gegeben werden.

Für „ECQuiz“ haben wir QTI zunächst soweit implementiert, dass ein „round trip“ möglich ist, d. h., dass von „ECQuiz“ exportierte Dateien ohne Informationsverlust wieder importiert werden können. Bei der Abbildung des Modells von „ECQuiz“ auf QTI traten mehrfach Probleme auf, die es erforderlich machten, „ECQuiz“-spezifische Erweiterungen vorzunehmen.

Wie bereits oben erwähnt, deckt QTI 2.0 im Gegensatz zu QTI 1.x 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. Da „ECQuiz“ jedoch sowohl komplette Tests als auch Gruppen von zusammengehörigen Fragen unterstützt, musste ein Weg gefunden werden, diese Strukturen dennoch in einer möglichst portablen Weise zu beschreiben. Die Teile „Integration Guide“ und „Migration Guide“ der QTI-Spezifikation schneiden einige dieser Aspekte an, es bleiben jedoch viele Fragen bezüglich der konkreten Umsetzung offen:

> 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 an associated learning design or set of sequencing rules. [IMS05, Integration Guide, S. 4]

Sie machen auch deutlich, dass die Integration der verschiedenen IMS-Spezifikationen noch nicht 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, S. 7 bzw. S. 9]

Wir haben den im Folgenden beschriebenen Ansatz gewählt. Eine Frage mit ihren zugehörigen Antworten wird gemäß QTI 2.0 in ein „assessmentItem“ abgebildet. Die Zusammenstellung der Fragen zu einem Test erfolgt gemäß der IMS Content Packaging Specification (CP) [IMS04]. „Packaging“ bedeutet hier, dass alle Items zusammen mit einem sog. „Manifest“ in ein ZIP-Archiv gepackt werden. Das Manifest ist eine XML-Datei mit dem Namen „imsmanifest.xml“ im Wurzelverzeichnis des Archivs und beschreibt die im Archiv enthaltenen Ressourcen.

Die von „ECQuiz“ vorgesehenen Bearbeitungshinweise für Test und Fragegruppen werden weder von QTI noch von CP explizit unterstützt. Wir behandeln diese Hinweise als „assessmentItem“ ohne „Interaction“. Auf diese Weise lässt sich die Erweiterung syntaktisch konform modellieren, es ist allerdings nicht sicher, ob andere Systeme diese Verwendung des <assessmentItem>-Elements korrekt interpretieren können.

Die Randomisierung der Antworten innerhalb einer Frage wird von QTI abgedeckt, „ECQuiz“ unterstützt jedoch auch die Randomisierung von Fragen, einschließlich der zufälligen Auswahl einer Untermenge der vorhandenen Fragen. Das Verhalten kann für jede Fragegruppe 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.

Der Implementierungsaufwand für die QTI-Unterstützung in „ECQuiz“ war sehr hoch, denn um einen Test ohne Informationsverlust exportieren zu können, mussten neben QTI auch IMS CP sowie Teile von IMS Simple Sequencing implementiert werden, die nicht immer perfekt aufeinander abgestimmt sind. Außerdem galt es, Lösungen für Eigenheiten und Einschränkungen von QTI zu finden. Das QTI-Modul macht in „ECQuiz“ fast 50 % des gesamten Codes aus. Um den Aufwand in Grenzen zu halten, ist der QTI-Import primär darauf ausgerichtet, von „ECQuiz“ selbst exportierte Tests zu importieren. Experimente mit QTI-Dateien bzw. Content-Packages aus verschiedenen Quellen waren, außer bei sehr einfachen Items, unbefriedigend. Beispielsweise sind die von Moodle exportierten QTI-2.0-Items und Content-Packages nicht mit den Spezifikationen konform, so dass der Import in „ECQuiz“ fehlschlägt.

5 IMS QTI: Probleme

Beim Entwurf von QTI 2.0 sind Rückmeldungen von der Anwendergemeinde zu QTI 1.x eingeflossen. Trotzdem sind uns während der Implementierung in „ECQuiz“ (vgl. Abschnitt 4) eine Vielzahl von Schwachstellen aufgefallen, von denen wir einige im Folgenden beschreiben. Die Probleme lassen sich dabei drei Bereichen zuordnen: „Problematische Designentscheidungen“, „formale Schwächen“ und „Schwächen in der technischen Umsetzung in XML“.

Vorab ist anzumerken, dass QTI 2.0 keine existierende Praxis kodifiziert, sondern komplett neu geschrieben wurde. Im Gegensatz zum Standardisierungsverfahren für RFCs der IETF [Br96] verlangt IMS nicht mindestens zwei voneinander unabhängig entwickelte, interoperable Implementierungen; es gibt auch keine Referenzimplementierung von QTI 2.0.

5.1 Problematische Designentscheidungen

QTI 2.0 ist eine überaus umfangreiche Spezifikation mit vielen optionalen Teilen. Um die Interoperabilität zwischen Systemen zu sichern, die nicht den gesamten Standard umsetzen, sieht die QTI-Spezifikation die Definition von Profilen vor. Sie erlauben es, die von einem System implementierte Untermenge von QTI zu beschreiben. Zwei Profile, „QTI-Lite“ und „QTI-All“, sind vordefiniert. Beide sind praktisch leider von geringem Nutzen. QTI-Lite beschreibt eine minimale Untermenge, die selbst für einfachste Tests zu beschränkt sein dürfte: Beispielsweise dürfen in QTI-Lite-konformen Items keine Aufzählungen oder Tabellen und lediglich die Bildformate JPEG und GIF, nicht aber das vom W3C standardisierte PNG-Format benutzt werden. Diese Einschränkungen sind aus technischer Sicht nicht nachvollziehbar. QTI-All hingegen fordert die Implementierung der gesamten Spezifikation. Zur Zeit ist uns keine vollständige Implementierung von QTI 2.0 bekannt; angesichts des Umfangs erscheint es uns fraglich, ob es eine solche jemals geben wird.

Die QTI-Spezifikation ist in vielen Punkten recht liberal; innerhalb eines Items sind praktisch beliebige Strukturen zugelassen. So ist etwa ein komplett leeres Item vollkommen standardkonform, ebenso ein Item ohne <itemBody> (und damit ohne Fragetext) sowie ein Item mit mehreren „Interactions“, d. h. ein Item, das z. B. gleichzeitig Multiple-Choice-Frage als auch Lückentext ist. Ein Vorteil dieser Philosophie ist, dass prinzipiell viele Fragetypen auf QTI abgebildet werden können. Der Nachteil ist, dass der Import von QTI-Items aus unbekannten Quellen sehr komplex wird. Da der Standard zudem nicht klärt, welche Bedeutung etwa ein leeres Item oder eines mit mehreren „Interactions“ hat, kann kaum sichergestellt werden, dass ein Item genau so importiert wird, wie vom Autor ursprünglich vorgesehen. Somit wird der Hauptzweck eines Austauschformats nicht erreicht.

Zusätzlich zur Beschreibung von Fragen und Tests spezifiziert QTI mit dem sogenannten „Response Processing“ eine Programmiersprache zur Auswertung von Tests. Da Beschreibung und Auswertung von Tests aber zwei vollkommen unterschiedliche Aspekte sind,

hätte das „Response Processing“ unserer Meinung nach in eine separate Spezifikation ausgegliedert werden sollen. Dies hätte zur Vereinfachung des ohnehin sehr umfangreichen QTI-Standards beigetragen.

Die oben genannte Trennung von Inhalt, Erscheinungsbild und Verhalten findet sich im Design von QTI praktisch nicht wieder. In den Definitionen vieler Elemente, die den Inhalt des Items beschreiben, z. B. <feedbackBlock>, <feedbackInline> oder <responseDeclaration>, finden sich Abhängigkeiten zum per „Response Processing“ beschriebenen Verhalten des Items. Selbst wenn die Auswertung eines Tests nicht mit „Response Processing“ erfolgt, müssen bestimmte Elemente und Attribute vorhanden sein, damit ein Item dem QTI-Standard entspricht.

5.2 Formale Schwächen

Da sie an vielen Stellen ungenau oder mehrdeutig ist, genügt die QTI-Spezifikation nicht unserer Forderung nach exakter Formulierung. Zahlreiche Fragen bleiben ungeklärt oder müssen vom Leser selbst erschlossen werden. Wir halten es daher für unwahrscheinlich, dass zwei Implementierungen von QTI in ihrer Interpretation des Standards genug übereinstimmen, dass ein reibungsoser Austausch zwischen ihnen möglich ist.

Beispielsweise wird der Datentyp „language“ im [IMS05, XML Binding, S. 52] mit dem knappen Satz „A trivial restriction of xsd:string.“ definiert. Ob Sprachbezeichnungen z. B. nach RFC 3066 anzugeben sind, wird nicht näher festgelegt.

Die Definition des Formats des Typs „identifier“ ist hingegen übermäßig lang ausgefallen. Üblicherweise werden für derartige Definitionen reguläre Ausdrücke oder kontextfreie Grammatiken in Backus-Naur-Form (BNF) verwendet. In der QTI-Spezifikation wird jedoch eine umständliche natürlichsprache Definition gegeben:

> An identifier is a string of characters that must start with a Letter or an underscore (’_’) and contain only Letters, underscores, hyphens (’-’), period (’.’, a.k.a. full-stop), Digits, CombiningChars and Extenders. Identifiers containing the period character are reserved for future use. The character classes Letter, Digit, CombiningChar and Extender are defined in the Extensible Markup Language (XML) 1.0 (Second Edition) [XML]. Note particularly that identifiers may not contain the colon (’:’) character. Identifiers should have no more than 32 characters. for compatibility with version 1 They are always compared case-sensitively.

Die Interpunktionsfehler sind im Original enthalten; dadurch ist unklar, worauf sich die Kompatibilität mit Version 1 bezieht. Abweichend von der obigen Spezifikation ist im Schema der Typ „identifier“ als „NMTOKEN“ deklariert, das Schema erlaubt somit u. a. Punkt und Doppelpunkt, so dass die Anwendung die weiteren Restriktionen implementieren muss, obwohl dies bereits im Schema möglich gewesen wäre. Ebenso wird nicht verbindlich geregelt, ob Bezeichner mehr als 32 Zeichen lang sein dürfen und innerhalb welches Bereiches sie eindeutig sein müssen – tatsächlich macht die Spezifikation überhaupt keine Aussage zur Eindeutigkeit von Bezeichnern.

Ein anderes Beispiel findet sich in der Definition des Elements <extendedTextInteraction> für Freitextaufgaben. Dieses Element hat unter anderem die Attribute „expectedLines“ und „expectedLength“. Beide sind dafür vorgesehen, dem Kandidaten einen Anhaltspunkt zu geben, wie umfangreich seine Antwort ausfallen sollte (vgl. [IMS05, Information Model, S. 29]):

> Attribute: expectedLines [0..1]: integer
>
> The expectedLines attribute provides a hint to the candidate as to the expected number of lines of input required. A Delivery Engine should use the value of this attribute to set the size of the response box, where applicable.
>
> Attribute: expectedLength [0..1]: integer
>
> The expectedLength attribute provides a hint to the candidate as to the expected overall length of the desired response. A Delivery Engine should use the value of this attribute to set the size of the response box, where applicable.

Aus den nahezu identischen Beschreibungen dieser beiden Attribute ist nicht ersichtlich, worin sie sich in ihrer Funktion unterscheiden. Ebensowenig wird erläutert, welcher Wert Vorrang hat, falls beide Attribute angegeben wurden. Zudem ist bei beiden unklar, worauf sich der anzugebende Wert bezieht. Die Vorgabe einer bestimmten Anzahl von Zeilen mittels „expectedLines“ ergibt nur dann einen Sinn, wenn auch festgelegt ist, wie lang die Zeilen sind. Der Wert von „expectedLength“ könnte z. B. die Anzahl an Wörtern meinen oder die Anzahl der Sätze oder die Breite in Zentimetern eines Eingabefeldes auf einer Webseite. Bei Items aus unbekannten Quellen ist es unmöglich festzustellen, was beabsichtigt war.

5.3 Schwächen in der technischen Umsetzung in XML

Die QTI-Spezifikation umfasst auch das „XML Binding“, in dem ein XML-Schema definiert wird, das vorgibt, wie die im „Information Model“ beschriebenen Items auf XML-Elemente abgebildet werden. Leider werden die Möglichkeiten von XML und XML-Schema nur ansatzweise ausgenutzt, so dass QTI-Dokumente mit Standard-XML-Werkzeugen nur zum Teil validiert werden können. Damit wird eine weitere unserer Anforderungen an Austauschformate nicht erfüllt. Einige Beispiele sollen dies illustrieren.

An vielen Stellen in QTI wird für Querverweise auf andere Elemente ein Attribut verwendet, das den Bezeichner des Zielelements enthält. Teilweise werden Querverweise in QTI-Dokumenten aber auch anders realisiert. Wie in Abbildung 1 zu sehen ist, werden z. B. die richtigen Antwortmöglichkeiten auf eine Multiple-Choice-Frage mit Hilfe des <correctResponse>-Elements angegeben, das wiederum ein oder mehrere Elemente vom Typ <value> enthält. Der Inhalt jedes <value>-Elements ist der Bezeichner einer korrekten <simpleChoice>.

Problematisch ist an diesem Ansatz, dass bei konsequenter Nutzung der XML eine deutlich robustere und elegantere Lösung möglich gewesen wäre. Zur Realisierung von Querverweisen stellt XML eigens die Attributtypen „ID“, „IDREF“ und „IDREFS“ zur Verfügung.

Werden Attribute dieses Typs verwendet, ist garantiert, dass alle Bezeichner vom Typ „ID“ eindeutig sind und dass alle per „IDREF“ oder „IDREFS“ referenzierten Element auch tatsächlich existieren. Vermutlich ist diese Semantik auch in QTI intendiert; im QTI-Schema werden jedoch an keiner Stelle die Attributtypen „ID“, „IDREF“ oder „IDREFS“ verwendet.

Stellvertretend für zahlreiche weitere Schwächen in der XML-Umsetzung sei noch das Element <rubricBlock> genannt. Im [IMS05, Information Model, S. 23] findet sich in der Definition dieses Elements der Hinweis „Although rubric blocks are defined as simpleBlocks they must not contain interactions.“ Im XML-Schema wird diese Einschränkung aber nicht umgesetzt, obwohl W3C XML Schema durchaus die dazu notwendigen Mittel bietet.

6 Zusammenfassung und Schlussfolgerung

Aufgrund unserer Erfahrungen bei der Implementierung einer Untermenge von QTI 2.0 in „ECQuiz“ und der Analyse der QTI-Spezifikation kommen wir zu dem Schluss, dass QTI 2.0 als Standard für den Austausch von Tests nicht geeignet ist: Es erlaubt zwar prinzipiell die Beschreibung einer großen Zahl von Testtypen und Auswertungsverfahren, eine vollständige Implementierung ist aber nur mit extrem hohem Aufwand möglich – unvollständige Implementierungen erreichen jedoch nicht die angestrebte Interoperabilität.

Zur Zeit (Juni 2007) ist QTI 2.1 in Vorbereitung; diese Version soll u. a. die bislang fehlende Möglichkeit zur Beschreibung vollständiger Tests nachliefern. In den aktuellen Entwürfen sind darüberhinaus keine grundsätzlichen Änderungen gegenüber Version 2.0 vorgenommen worden, so dass die o. g. Kritikpunkte weiterhin zutreffen. Es ist jedoch bereits absehbar, dass QTI 2.1 nicht vollständig abwärtskompatibel zu QTI 2.0 sein wird: Nach QTI 2.0 kodierte Tests werden also ohne Änderungen keine gültigen QTI-2.1-Tests sein.

Gorissen hat 2003 und 2006 [Go03, Go06] bei einer Auswahl von Systemen[5] die Unterstützung für QTI evaluiert. Seine Untersuchungen zeigen, dass alle Systeme nur sehr kleine Untermengen von QTI unterstützen; in den meisten Fällen gehen beim Import Informationen verloren. Er bemerkt, dass sich die Situation seit 2003 – trotz des offensichtlichen Bedarfs für Austauschmöglichkeiten von Tests – praktisch nicht verbessert hat.

Angesichts dieser Situation und der Komplexität von QTI schlägt Gorissen die Entwicklung einer freien Referenzimplementierung durch die „educational community“ vor. Da die Probleme mit QTI letztendlich in der Designphilosophie begründet sind, lassen sie sich nicht dadurch beheben, dass lediglich die o. g. Schwächen ausgebessert werden. Vielmehr ist aus unserer Sicht ein grundsätzlich anderer Ansatz notwendig, um die in Abschnitt 2 definierten Kriterien erfüllen zu können. Daher halten wir auch eine Beteiligung am QTI-Spezifikationsprozess nicht für Erfolg versprechend.

[5] 2006: Respondus, QuestionMark Perception, N@tschool!, Blackboard, Learn eXact (QTI 1.2) und TOIA (QTI 2.1)

Viele der Probleme im Ansatz von QTI rühren auch daher, dass QTI nicht aus dem praktischen Einsatz heraus entwickelt wurde, sondern von einem Kommittee „am grünen Tisch“ entworfen wurde (vgl. Abschnitt 5). Wir schlagen deshalb vor, dass die Anwendergemeinde (also die „educational community“) stattdessen auf der Basis ihrer praktischen Erfahrungen selbst ein Austauschformat für Tests entwickelt, das die o. g. Desiderata umsetzt und so die Implementierung tatsächlich interoperabler Testsysteme ermöglicht.

Literaturverzeichnis

[Br96] Scott Bradner. The Internet Standards Process – Revision 3. RFC 2026 (BCP 9), Internet Engineering Task Force, 1996.

[Go03] Pierre Gorissen. Quickscan QTI. Usability study of QTI for De Digitale Universiteit, 2003.

[Go06] Pierre Gorissen. Quickscan QTI – 2006. Usability study of QTI for De Digitale Universiteit, 2006.

[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.

[ISO03] ISO (International Organization for Standardization). ISO/IEC 19757-2:2003. Information technology – Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG, 2003.

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

[PR05] Michael Piotrowski und Dietmar Rösner. Integration von E-Assessment und Content-Management. In Jörg M. Haake, Ulrike Lucke und Djamshid Tavangarian, Hrsg., DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V., number P-66 in Lecture Notes in Informatics, Seiten 129–140, Bonn, 2005. GI-Verlag.

[RPA07] Dietmar Rösner, Michael Piotrowski und Mario Amelung. A Sustainable Learning Environment based on an Open Source Content Management System. In Wilhelm Bühler, Hrsg., Proceedings of the German e-Science Conference (GES 2007). Max-Planck-Gesellschaft, 2007.

[SR00] Colin Smythe und P. Roberts. An Overview of the IMS Question & Test Interoperability Specification. In Proceedings of the 4th CAA Conference, Loughborough, England, 2000. Loughborough University.

[Wo04] World Wide Web Consortium. XML Schema, 2004.

[Wo06] World Wide Web Consortium. Extensible Markup Language (XML) 1.0, 2006.