Wiki-basiertes Assessmentsystem für die Erstellung von interaktiven Sprachübungen

Christoph Portsch

Wiki-based Assessment System for the Creation of Interactive Language Exercises

Master's Thesis

at

Graz University of Technology

submitted by

Christoph Portsch

Institute for Information Systems and Computer Media (IICM),
Graz University of Technology
A-8010 Graz, Austria

28th January 2010

Advisor: Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic

Wiki-basiertes Assessmentsystem für die Erstellung von interaktiven Sprachübungen

Masterarbeit

an der

Technischen Universität Graz

vorgelegt von

Christoph Portsch

Institut für Informationssysteme und Computer Medien (IICM),
Technische Universität Graz
A-8010 Graz

28. Januar 2010

Diese Arbeit ist in deutscher Sprache verfasst.

Begutachter: Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic

Abstract

Online assessment plays a vital role in today's learning processes. The different forms of assessments have led not only to an increase in learner's participation in the learning process, but also to their enhanced learning performance. These fundamental ideas form the basis of the international project "SprichWort", which this thesis reports upon.
The production of assessment data is a time-consuming venture. Thus, to keep the costs as low as possible, there are two objectives worth striving for. Firstly, to aim to keep the production of forms of assessment as simple as possible, and secondly, to reuse previously created data.
The de facto standard Question Test Interoperability (QTI) aims to fulfil the second of these objectives. The specification is an all-encompassing solution for the reuse and interoperability of assessment data. This thesis examines the QTI standard and the associated open source implementations. In this paper, some of the weaknesses will be identified and discussed, such as the separation of concerns, the high complexity of QTI and the current version policy. Indeed, the high complexity and the low number of authoring tools reveal the unsuitable nature of this data format for the "SprichWort" project.
Based on the knowledge gained, a new assessment system has been designed and implemented. The focus of this development is on the intuitive and easy creation of assessment data. Using Wiki frameworks and the associated syntax, it is possible for teachers without any special technical knowledge to build and manage assessment data. In addition, particular attention is paid to concealing the logic and visualization from the user.
The final section of the paper compares both data formats and thereby highlights some of the criticisms of the QTI specification.

Kurzfassung

Online Assessments spielen in der heutigen Zeit einen vitalen Part im Lernprozess. Durch die unterschiedlichen Assessmenttypen wird nicht nur die Beteiligung am Lernprozess erhöht, sondern auch die Lernperformanz gesteigert. Diese Grundideen bilden auch die Basis des internationalen Projektes „SprichWort“, mit dem sich diese Arbeit beschäftigt.
Die Erstellung solcher Assessmentdaten ist jedoch ein aufwendiges Unterfangen. Um den Generierungsaufwand so gering wie möglich zu halten, sind zwei Ziele erstrebenswert: Zum einen, die Erzeugung von Assessments so einfach wie möglich zu gestalten und zum anderen, bereits erstellte Daten wieder zu verwenden.
Der DeFacto Standard Question Test Interoperability (QTI) strebt die zweite Methode an. Die Spezifikation ist eine allumfassende Lösung für die Wiederverwendung und Interoperabilität von Assessmentdaten. Die hier vorgestellte Arbeit untersucht diesen Standard und die damit verbundenen frei verfügbaren Implementationen. Dabei werden einige Schwachstellen, wie die Trennung der Belange, die hohe Komplexität und die derzeitige Versionspolitik aufgezeigt und diskutiert. Auf Grund der hohen Komplexität und der geringen Anzahl an Autorenwerkzeugen zeigt sich schnell die Unanwendbarkeit dieses Datenformates für das Projekt „SprichWort“.
Basierend auf den gewonnenen Erkenntnissen wird ein neuer Weg eingeschlagen und somit ein neues Assessmentsystem konzipiert. Der Fokus dieser Entwicklung wird nun mehr auf die intuitive und einfache Erstellung von Assessmentdaten gelegt. Mit Hilfe eines Wikiframeworks und der damit verbunden Syntax wird es Pädagogen möglich, ohne besonderes technisches Vorwissen selbst Assessmentdaten zu erstellen und zu verwalten. Zudem wird vor allem darauf geachtet, dass die Logik und die Visualisierung vor dem Benutzer verborgen bleiben.
Den Abschluss dieser Arbeit bildet ein Vergleich beider Datenformate, der nochmals einige Kritikpunkte der QTI Spezifikation hervorhebt.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagung

„Leider lässt sich eine wahrhafte Dankbarkeit mit Worten nicht ausdrücken.“

[Johann Wolfgang von Goethe (1749-1832), dt. Dichter]

Die hier vorgestellte Masterarbeit ist am Institut für Informationssysteme und Computer Medien (IICM) der Technischen Universität Graz im Rahmen einer Mitarbeit am Projekt „SprichWort“ entstanden.

Ich möchte mich an dieser Stelle herzlich bei meinem Betreuer Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic bedanken, der mir nicht nur die Beteiligung an diesem Projekt ermöglichte, sondern mich auch mit zahlreichen Ratschlägen unterstützte. In diesem Zuge möchte ich auch Dipl.-Ing. Christoph Trattner danken, der mir als Projektkollege nicht nur technische sondern auch freundschaftliche Unterstützung gab. Vielen herzlichen Dank euch beiden!

Infolge des internationalen Kontextes dieses Projekts war auch eine intensive Teamarbeit vonnöten. Hier gilt mein Dank vor allem meinen lieben Kolleginnen und Kollegen: Dr. Brigita Kacjan, Dr. Darinka Chovaniaková, Dr. Věra Kozáková und Dr. Tamás Kispál. Unser angenehmes Arbeitsklima lag mir stets sehr am Herzen. Ebenso möchte ich mich auch beim gesamten „SprichWort“-Team, im Besonderen bei unserer Projektkoordinatorin Ao.Prof. Dr. Vida Jesenšek, für die gute Zusammenarbeit bedanken. Ich danke euch!

Ein Dankeschön auch an all meine Studienkollegen, die mich durch mein gesamtes Studium begleiteten. Vor allem Wolfgang Kandlbauer gilt hier mein Dank, der mir eine große moralische Stütze war. Vielen Dank!

Ich möchte mich auch sehr herzlich bei Mag. Dr.phil. Maria Löschnigg und Ao.Univ.-Prof. Mag. Dr.phil. Martin Löschnigg bedanken, die sehr viel Zeit für das Korrekturlesen meiner Masterarbeit aufwendeten. Vielen herzlichen Dank!

Mein besonderer Dank gilt meinen Eltern Ilse und Walter Portsch, die mich nicht nur in meiner ganzen Ausbildung unterstützten, sondern mir auch immer die Freiheit überließen, meine Entscheidungen selbst zu treffen. Auch meinen Geschwistern Lisa und Klaus möchte ich besonders für ihre mentale Unterstützung danken. Ich danke euch von Herzen!

Zu guter Letzt möchte ich einem ganz besonderen Menschen danken - meiner lieben Freundin Viktoria. Ohne ihre moralische Unterstützung, ihrer aufopfernde stilistische Hilfe und ihrer Liebe wäre diese Arbeit nicht so schnell entstanden. Ich danke dir von Herzen!

Kapitel 1

Einleitung

„Früh übt sich, was ein Meister werden will.“

[Friedrich von Schiller (1759 - 1805), Wilhelm Tell III]

1.1 Motivation

Die rapide Entwicklung der Internet-Technologie führte dazu, dass „E-Learning“ als Lehr- und Lernmethode den Lernprozess revolutionierte. Nach [Bodzin und Cates, 2003] und [Santally und Raverdy, 2006] zit. nach [Wang, 2008, S. 1248] bietet E-Learning mehr Ressourcen an und erhöht zugleich die Lerneffektivität. [Santally und Raverdy, 2006] zit. nach [Wang, 2008, S. 1248] zeigten weiters, dass die größte Stärke von „E-Learning“ die Auflösung der zeitlichen und räumlichen Restriktionen ist. Somit werden dem Lerner viel mehr Möglichkeiten geboten, um Wissen zu erwerben.
Diesen Aufschwung im World Wide Web erlebte auch die didaktische Methodik des Assessments mit und wurde dadurch zum „E-Assessment“. Diese neue Technologie führte zu einer Explosion der Aufgabenformate in solchen Assessments. Neben herkömmlichen Multiple Choice Fragen und Freitextaufgaben fanden auch multimediale Formate und interaktive Simulationen Anklang. Nach [Hambleton, 2004, S. 698] ist die größte Herausforderung in den nächsten 20 Jahren die Verwaltung dieser Prüfungen. Begrifflichkeiten wie Interoperabiltität, Austauschbarkeit und Wiederverwendbarkeit bilden die Schlagworte und Bedürfnisse in Bezug auf „E-Assessment“.
Das IMS Global Learning Consortium liefert mit dem De-Facto Standard „Question Test Interoperability“ (QTI), die Antwort auf diese Forderungen. Diese Spezifikation beschreibt ein Datenmodel zur Repräsentation von Fragen (*assessmentItem*), Tests (*assessment*) und den damit zusammenhängenden Ergebnisdarstellungen. QTI verspricht durch die Verwendung seines Standards viele neue Innovationen und Werkzeuge auf diesem Bereich. Weiters soll dieses Format nicht auf ein Gebiet beschränkt verwendet werden, sondern allen Fachrichtungen dienen.
Doch welcher Schritt kommt vor der Verwaltung? Die Erstellung darf nicht außer Acht gelassen werden. Schließlich müssen diese unzähligen Tests und Aufgaben auch produziert werden. Hierfür sind Fachgebietsexperten und Pädagogen zuständig, die meist wenig technisches Hintergrundwissen besitzen. Das trifft auch für das SprichWort[1] Projekt zu, eine Internet-Lernplattform für das Sprachenlernen, welche das Ziel verfolgt, Sprichwörter als wichtiges Kulturgut einer Sprachgemeinschaft in ihrem heutigen Gebrauch zu dokumentieren und zu vermitteln.
Hier stellt sich die Frage, ob sich QTI wirklich als ein praxistaugliches Testformat bewahrheiten kann und ob es auch durch Pädagogen ohne fundiertes Vorwissen verwendet werden kann. Die Komplexität dieser Spagatlösung zeigt jedoch schnell Grenzen in der Anwendbarkeit. Doch welcher Weg führt aus

[1] http://www.sprichwort-plattform.org

diesem Dilemma?
Einfachheit soll nun das neue Schlagwort für Assessments sein. Den Pädagogen, den Autoren unzähliger Aufgaben und Tests, soll eine intuitive Schnittstelle für die Erstellung dieser Daten geboten werden. Schließlich ist das ihre Aufgabe.

1.2 Struktur dieser Arbeit

Diese Arbeit ist in zwei Hauptteile und einem Appendix gegliedert. Der erste Teil (Kapitel 1 bis Kapitel 4) beschäftigt sich zum einen mit der theoretischen Auseinandersetzung der Assessment-Thematik und zum anderen mit dem De-Facto Standard QTI und dessen Anwendbarkeit auf das SprichWort Projekt. Der zweite Teil (Kapitel 5 bis Kapitel 9) beinhaltet die praktischen Überlegungen und beschreibt die alternative Implementation näher.

Die Masterarbeit startet mit dem Kapitel 1, welches einen Einblick in die Motivation und die Struktur dieser Arbeit bietet.

Kapitel 2 bildet die Grundlage für die weitere Verarbeitung der Thematik „Online Assessments“. Ebenso werden alle Begriffe definiert, die in weiterer Folge verwendet werden. Weiters gibt es einen Einblick in die Testgestaltung und gibt Anregungen, die in späteren Teilen der Arbeit aufgegriffen werden.

Kapitel 3 gibt einen Überblick über die QTI Spezifikation des IMS Global Learning Consortium. Dabei wird nicht nur der Aufbau dieses Standards näher untersucht, sondern auch dessen Schwächen. Zusätzlich werden auch freiverfügbare Implementationen unter die Lupe genommen und analysiert.

Kapitel 4 führt eine Anforderungsanalyse der Übungskomponente des internationalen SprichWort Projektes durch. Hierbei werden funktionale, wie auch nicht funktionale Anforderung in Hinsicht auf die Erzeugung, die Durchführung, die Bewertung und die Ergebnisbetrachtung getroffen. Im Anschluss darauf wird die Spezifikation IMS QTI als Grundlage für das Übungssystem diskutiert. In diesem Zuge wird auch die Einsetzbarkeit der QTI Werkzeuge betrachtet und evaluiert.

Kapitel 5 konzeptioniert ein Assessmentsystem, welches auf einem vereinfachten Datenformat basiert. Zudem werden auch die Technologien untersucht, die dieses Format unterstützen und die Grundlage für dieses System bilden. Dabei wird zum einen eine geeignete JavaScript Bibliothek selektiert, welche den verschiedenen Interaktionsanforderungen der Kandidaten entspricht und zum anderen ein passendes Wikiframework für die Erzeugung, Verwaltung und Durchführung von Assessments gewählt.

Kapitel 6 befasst sich mit der Implementation des Wikiframeworks und beschreibt das neue Datenformat näher. Dabei wird die Erstellung von Übungen, Tests, Aufgaben und die Feedback Komponente, die Selbstbewertungsbögen, vorgestellt. Zusätzlich werden auch die neuen Interaktionstypen der Aufgaben präsentiert.

Kapitel 7 stellt einen Vergleich zwischen dem Standard QTI und der proprietären Wiki Lösung auf. Hierzu werden beide Fokusse gegenübergestellt um eine Positionierung dieser Ansätze durchzuführen. Des Weiteren wird der XML Syntax von QTI und der Wikitext des Wikisystems untersucht. Zusätzlich werden auch die einzelnen Assessment Elemente wie Aufgaben, Übungen und Tests verglichen.

Kapitel 8 und Kapitel 9 fassen die Arbeit zusammen und geben zudem auch einen Ausblick über die zukünftige Entwicklung, der Spezifikation QTI, des Wikisystems und des SprichWort Projektes.

Kapitel 2

Assessment

„Prüfungen sind deshalb so unerträglich, weil der größte Dummkopf mehr fragen kann, als der gescheiteste Mensch zu beantworten vermag.“

[Charles Caleb Colton (1780 - 1832), englischer Aphoristiker u. Essayist]

Dieses Kapitel bildet die Grundlage für die weitere Verarbeitung der Thematik Online Assessments und definiert alle Begriffe, die in weiterer Folge verwendet werden. Weiters gewährt es einen Einblick in die Testgestaltung und verschafft Anregungen, die in späteren Teilen der Arbeit aufgegriffen und verarbeitet werden. Dieser Teil der Arbeit beschäftigt sich zunächst mit einer Einführung in die Didaktik, im speziellen in die Web-Didaktik. Hierzu wird die Medientheorie näher beleuchtet. Im Anschluss werden einige didaktische Methoden beschrieben, wobei das Thema Assessments in Form von Tests hervorgehoben wird. Dieser Exkurs in das Thema Wissensüberprüfung soll einen Überblick über die verschiedenen Formen und Arten der Assessments schaffen. Um noch tiefer in diese Materie zu gelangen, werden eine Vielzahl an verschiedenen Test Item Typen, d.h.: Testfrage Typen, diskutiert. In diesem Rahmen wird vor allem auf die Thematik der Multiple Choice Fragen eingegangen und deren Vor- und Nachteile erläutert. Ebenso gibt dieser Abschnitt einen Überblick über deren Anwendungsregeln und möglichen Hilfestellungen bei der Durchführung. Einen wichtigen Punkt behandelt der darauf folgende Abschnitt mit der Thematik Parametrisierung von Test Items. Die daraus gewonnen Informationen dienen zur Motivation der Testpersonen und erhöhen zugleich die Sicherheit von Prüfungen in Hinblick auf Betrug. Zum Abschluss wird das Thema Qualitätsmerkmale, wie zum Beispiel Testzuverlässigkeit und Testgültigkeit und die Problematik der korrekten Punktevergabe aufgezeigt.

2.1 Von der Didaktik zur Web-Didaktik

Die Didaktik wird definiert als die Theorie und Praxis des Lehrens und Lernens. Werner Jank und Hilbert Meyer formulieren die Aufgabe der Didaktik wie folgt: “Wer was von wem wann mit wem wo, wie, womit und wozu lernen soll?“ [Jank und Meyer, 1994, S. 16].
Hand in Hand mit dem World Wide Web entstand in den letzten Jahrzehnten ein neues Feld, welches als Web-Didaktik bzw. Online Didaktik beschrieben wird. Um diese Entwicklung näher zu betrachten wird zunächst die medientheoretische Grundlage behandelt. Laut [Swertz, 2005, S. 3] können nämlich mediale Veränderungen nur in Relation von Medien reflektiert werden, da das Nachdenken über Medien immer schon den Gebrauch von Medien voraussetzt. Welche Veränderung brachte also der Weg vom Buch zum Computer?
Das Buch besteht aus Farbe und Papier und besitzt eine feste Reihenfolge von einzelnen Blättern. Vordergründig wurden Buchstaben auf die Seiten gedruckt. [McLuhan, 1992] zit. nach [Swertz, 2005, S. 3]

ordnet dem Alphabet eine Linearität zu, diese Linearität geht einher mit der Reproduktion immer gleicher Exemplare von Büchern. Laut McLuhan zit. nach [Gehrke, 2009] führte dieser lineare Aufbau zu linearem Denken und darüber hinaus zu einer idealen Form des Unterrichts, dass alle das Gleiche lernen. Was ändert nun Computertechnologie an diesem Geltungsbestand? Das Medium besitzt die Eigenschaft, dass während der Darstellung von Information diese Darstellung, durch dahinter liegende Regeln, verändert werden kann. Dies führt zu einer Ablösung der immer gleichen Abfolge von Büchern und ermöglicht somit eine Individualisierung. Eine weitere Eigenschaft ist die Begrenzung der Visualisierung durch das Anzeige-Medium. Hier hat sich in der Theorie schon länger das Hypertextkonzept etabliert (Vgl. [Iske, 2001]). Wobei Hypertexte die zentrale Aufgabe der Navigation übernehmen und somit dem Lernenden die Möglichkeit bieten, zum nächsten Lernschritt zu springen [Swertz, 2005, S. 4]. Die Computertechnologie bringt somit zwei Veränderungen mit sich, zum einen die Individualisierung und zum anderen die Navigation.
Die Web-Didaktik baut auf beiden zuvor genannten Punkten auf. Der Ansatz der Individualisierung wird mit einer Vielfalt an Methoden (z. B. entdeckendes Lernen, *Blended Learning* oder aufgabenorientiertes Lernen) erfüllt und die Navigation in der Web-Didaktik wird systematisch und damit klar strukturiert gehalten. Die Aufbereitung des Lernmaterials erfolgt in zwei Schritten. Zuerst wird das Material von vorhandenen Quellen entnommen, d.h.: es wird dekontextualisiert und in geeignete granularisierte Elemente zerlegt. Danach werden diese Elemente wieder in die Lernzeit gebracht, d.h. sie werden rekontextualisiert. [Swertz, 2005, S. 6]

2.1.1 Dekontextualisierung

Der Aufbau einer didaktischen Wissensbasis wird Dekontextualisierung genannt. Hierbei wird, wie zuvor erwähnt, Wissen in verschiedenen Kontexten lokalisiert, entnommen und für den Lernprozess aufbereitet (Vgl. [Swertz, 2005, S. 6] und [Swertz, 2000]). Bei dieser Aufbereitung werden somit Lerneinheiten erzeugt. Diese Lerneinheiten setzen sich aus Wissenseinheiten zusammen und die wiederum bestehen aus Medieneinheiten. 2.1 erklärt diese Struktur näher:

- Lerneinheiten bestehen aus einer oder mehrerer Wissenseinheiten.
- Wissenseinheiten sind durch eine Wissensart bestimmt.
- Wissenseinheiten bestehen aus einer oder mehrerer Medieneinheiten. Medieneinheiten enthalten jeweils einen Medientyp (Abb. 12).

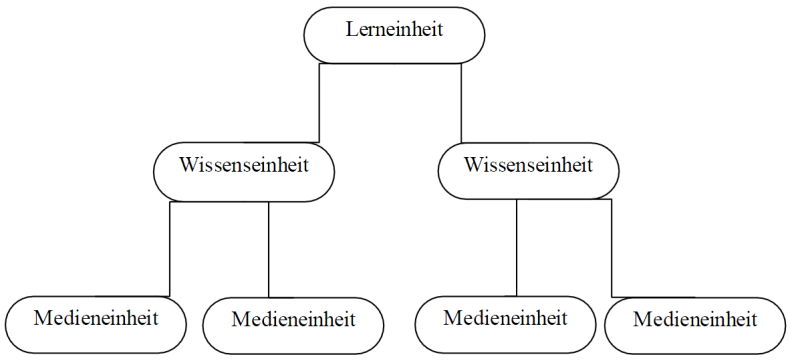


Abbildung 2.1: Aufbau von Lerneinheiten. [Swertz, 2005, S. 12]

Medieneinheiten setzen sich jeweils aus einem Medientyp zusammen. Aus didaktischer Sicht ist die mediale Variation entscheidend. Das heißt die eingesetzten Medien sollen sich unterscheiden, was durch die Multimedialität des Computers erleichtert wird. Medien setzen sich aus folgenden Gruppen zusammen [Swertz, 2005, S. 6]:

- Kommunikationsmedien (wie zum Beispiel: Chat, E-Mail)

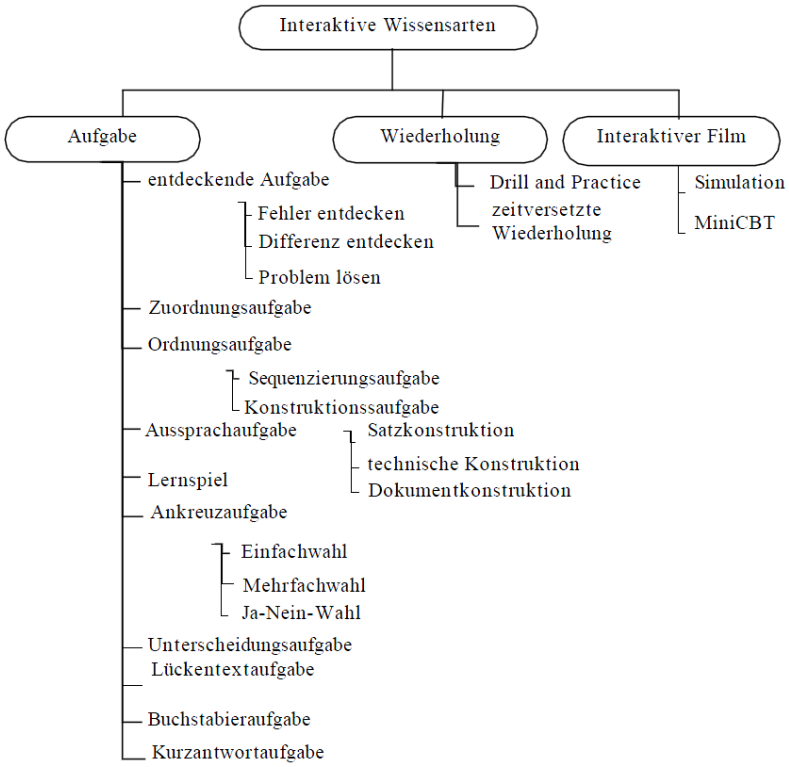
- Präsentationsmedien (wie zum Beispiel: Animation, Text)
- Interaktionsmedien (wie zum Beispiel: Formular)

Wissenseinheiten werden durch eine Wissensart bestimmt. In der Web-Didaktik existieren 3 Klassen von diesen Wissensarten [Swertz, 2005, S. 8]:

- rezeptives Wissen
- interaktives Wissen
- kooperatives Wissen

Rezeptives Wissen (Fakten, News) wird passiv wahrgenommen, interaktives Wissen wird durch die Interaktion mit der Maschine erlernt und kooperatives Wissen (Chat, Foren) besteht in der Kommunikation mit einer realen Person. Diese Arbeit umfasst vor allem das interaktive Wissen, welches in Abbildung 2.2 zu sehen ist [Swertz, 2005, S. 9]:

Abbildung 2.2: Zusammensetzung von interaktiven Wissen. [Swertz, 2005, S. 9]



Lerneinheiten sind Behälter für Wissenseinheiten und werden durch ein jeweiliges Thema festgelegt. Zusätzlich zu den zuvor genannten 3 Einheiten existieren noch Relationen. Diese Relationen ermöglichen die Lerneinheiten in Beziehung zu setzen und unterstützen somit eine Navigation durch die Themen. Diese Beziehungen bestehen zum Beispiel aus Assoziationsrelationen *(“ist neben“)* oder Hierarchieassoziationen *(“ist Teil von“)*. (Vgl. [Swertz, 2000] und [Swertz, 2005, S. 9])

2.1.2 Rekontextualisierung

Durch die Rekontextualisierung werden die erzeugten Lerneinheiten zu einem Ganzen zusammengeführt und bilden somit die Grundlage eines didaktischen Modells. So ergeben sich Lernpfade und Sequenzen, die entweder selbst gesteuert oder fremd gesteuert, durch eine Vorwärts-Rückwärts-Navigation, erlernt werden können. [Swertz, 2005, S. 14f]

2.1.3 Didaktische Methoden

Die Web-Didaktik beschränkt sich nicht auf eine didaktische Methode. Sie besitzt in Hinblick auf den Gedanken der Individualisierung eine Methodenvielfalt [Swertz, 2005, S. 5]. Die folgenden Methoden bilden laut [Tuparova und Tuparov, 2005, S. 3f] die Grundkomponenten der Didaktik:

- Vorlesung
- Diskussion
- Interaktive Simulation
- Projekt basierende Methode
- Übung
- Methode des Assessment
- Evaluation

Die Auswahl der richtigen Methodik hängt von der Form der Lehre, entweder *Distance Learning* oder *Blended Learning*, dem Typ des Lernens, selbstständig oder gemeinsam Lernen, und dem psychologischen Lernstil des Lernenden ab. [Tuparova und Tuparov, 2005, S. 3]

Die Arbeit konzentriert sich hierbei auf die Thematik der Wissensüberprüfung, der so genannten Assessments. Der nächste Abschnitt beschreibt dieses Gebiet näher und gibt einen Überblick der vielfältigen Definitionen und Arten von Assessment Typen und Prüfungselementen.

2.2 Assessments als didaktische Methode

Um den Begriff des Assessments in dieser Arbeit einzuschränken, wird er als Wissensüberprüfung oder Test übersetzt und mit folgender Definition behandelt: "*Ein Test ist ein standardisiertes Instrument zur Messung einer Merkmalsausprägung (Wissen, Intelligenz, Persönlichkeit, Lernerfolg etc.). Ein Test setzt sich aus Test-Elementen zusammen (Aufgaben, Bilder, Fragen), auf die ein Testteilnehmer reagiert bzw. antwortet. Diese Test-Elemente werden Test-Items oder einfach nur Items genannt. Items umfassen i.d.R. einen Stimulus (Frage, Abbildung, Aufgabe), eine Instruktion (Wählen Sie die richtige Antwort!) und die vorgegebenen Antwortmöglichkeiten.*" [Satow, 2006, S. 92]
Das Hauptziel von einem Assessment war ursprünglich eine faire Bewertung für die Leistung eines Lernenden. In den letzten Jahren änderte sich laut [Roberts, 2006, S. 3] dieser Fokus und ein weiteres Ziel, dass Assessments einen vitalen Part im Lernprozess spielen, setzte sich durch. Sogar die radikale Annahme, dass die primäre Rolle von Assessments das Lernen selbst sei und nicht mehr die Benotung, fand Anklang.
Assessments kann man in mehrere Arten unterteilen. Zum einen ist die Anzahl der beteiligten Personen mit deren jeweiliger Rolle entscheidend. Hier wird zwischen Self-, Peer- und Group Assessment unterschieden [Roberts, 2006, S. 3ff]. Zum anderen ist der Charakter des Assessments ausschlaggebend. Zeitpunktbezogen wird hier zwischen formativem und summativem Assessment getrennt. Weiters wird zwischen standardisiertem und unstandardisiertem Assessment separiert. Ebenso kann in Hinsicht auf die getesteten Merkmale zwischen Leistungs-, Diagnose- und Fortschrittstests unterschieden werden [Helen, 2009]. Die folgenden Abschnitte fassen die unterschiedlichen Arten zusammen.

2.2.1 Self-, Peer- und Group Assessments

Self-, Peer- und Group Assessments sind im E-Learningkontext sehr wichtig. Laut [McInnery und Roberts, 2004] und [Palloff und Pratt, 1999] ist ein Schwachpunkt von E-Learning, dass sich deren Nutzer isoliert fühlen und dadurch deren Motivation sinkt.
Genau hier wirken die oben genannten Assessments dagegen, da sie die Kommunikation zwischen den Lernenden fördern und sie gleichzeitig eine Gemeinschaft bilden (Vgl. [Roberts, 2006, S. 12] und [Garrison und Ehringhaus, 2009, S. 3]).
Weitere Vorteile sind nach [Noonan und Duncan, 2005, S. 3]:

- Erhöhung der Beteiligung der Lernenden im Lernprozess
- Erhöht soziale Interaktionen
- Ermöglicht individuelles Feedback
- Fokussiert Lernende auf den Lernprozess

2.2.1.1 Self Assessment

Das Self Assessment stellt einen Lern- und Bewertungsprozess dar. Der Lernende reflektiert sich dabei selbstkritisch, erfasst seinen Lernfortschritt und schlägt selbst seine Bewertung für seinen Lernerfolg vor. Die Reflexion ist eine essentielle Komponente von Self Assessments. Sie bietet dem Lernenden nicht nur die Möglichkeit, das Erlernte zu bedenken, sondern auch, wie der Lernprozess durchgeführt wurde. Aus dieser Reflexion resultiert ein wichtiges Feedback, um zukünftiges Lernen zu gestalten.
Wie bereits erwähnt, wird in einem Self Assessment der Lernfortschritt gemessen. Hierzu dient ein Lern-Portfolio, welches als Dokumentation des Erlernten und zur Darstellung des Wissenszuwachses dient.
Eine Problematik stellt jedoch die genannte Selbstbewertung dar. Laut [Falchikov und Boud, 1989] gibt es ohne Anleitung und Praxis zum Self Assessment nur eine kleine Korrelation zwischen den selbst vergebenen Noten und der Bewertung eines Lehrers. Aus diesem Grund ist die Ausbildung dieser Fähigkeiten zur Selbstbewertung vor einem solchen Prozess eine wichtige Komponente.
Laut [Buchanan, 2004] fördert diese Art von Assessment das Engagement von Lernenden, da sie aktiv am Lernprozess teilhaben und diesen auch selbst steuern können. Weiters ermöglicht ein solcher Prozess nach [Schunk, 2000], den Fokus auf Lerninhalte zu legen, die speziell vom Lernenden benötigt werden. [Cucchiarelli et al., 2000] behaupten, dass ein Self Assessment als Mix zwischen einem formativen und diagnostischen Assessment gesehen werden soll. Dieses ermöglicht dem Lernenden selbst, sein Wissen in Hinblick auf die Durchführung eines summativen Assessments zu kontrollieren [Roberts, 2006, S. 3].

2.2.1.2 Peer Assessment

Der Prozess des Peer Assessments beinhaltet eine gegenseitige Reflexion durch den jeweiligen Partner und gegebenenfalls auch einen Vorschlag zur Note des Partners [Roberts, 2006, S. 6ff]. Die Aufgabenstellung kann dabei vom Peer gestellt werden [Denny et al., 2008, S. 51] oder von einem Lehrer.
Auf Grund der aktiven Rolle beider Partner hat dies einen positiven Effekt auf den Lernprozess, so [McConnell, 2000]. Ebenso wird durch das aktive Entscheiden darüber wie, was und warum gelernt wird, das Engagement des Lernenden gesteigert. Auch die Bewertung, die Diagnostizierung von falschen Antworten und die Vermittlung von Feedback helfen dem Lernenden, sein eigenes Wissen zu vertiefen [Denny et al., 2008, S. 51].
Vorteile bringt auch die gleiche Perspektive beider Partner auf den Lernprozess, da dadurch Feedback besser aufgefasst wird als durch einen Lehrer.
Es gibt jedoch aus zwei Gründen Zurückhaltung bei Studenten. Zum einen fühlen sich manche der Situation nicht gewachsen, einen anderen zu bewerten. Und zum anderen schrecken manche davor zurück, dem Partner eine schlechte Note zu geben [Roberts, 2006, S. 8].

2.2.1.3 Group Assessment

Die Bedeutung dieses Begriffs zieht sich von einem Assessment der ganzen Gruppe, Assessment von Individuen innerhalb einer Gruppe bis hin zum Assessment einer Gruppe durch eine andere Gruppe.
Eine Schwierigkeit stellt hier aber die Bewertung dar, da nicht alle Mitglieder einer Gruppe pauschaliert beurteilt werden können. Aus diesem Grund wird in einer Gruppe selbst ein Peer Assessment eingeführt, indem sich die Gruppenmitglieder untereinander selbst beurteilen. Somit wird verhindert, dass so genannte „Free Rider“ ohne Aufwand ungerechtfertigt zu einer besseren Note kommen. [Roberts, 2006, S. 9ff]

2.2.2 Formatives und Summatives Assessment

Laut [Interactive Educational Systems Design, Inc., 2008, S. 5] verfolgen beide Assessments das gleiche Ziel, die Bewertung von Wissen. Sie unterscheiden sich aber im Zweck der Evaluierung.

2.2.2.1 Formatives Assessment

Dieses Assessment hat einen formenden Charakter. Das heißt, dass es Informationen und Richtungsweisungen für einen laufenden Lernprozess liefert [Interactive Educational Systems Design, Inc., 2008, S. 5]. Es kann dabei formend vor der Prüfung, durch das Lernen an sich, formend während der Prüfung, durch konstruktive Überlegungen einer Testperson oder formend nach einer Prüfung sein, durch das Feedback. [Interactive Educational Systems Design, Inc., 2008, S. 6f] weist folgende Charakteristika diesem Assessment zu:

- wird laufend und häufig durchgeführt
- wird benutzt um Informationen und Richtungsweisungen für den laufenden Lernprozess zu liefern
- liefert Feedback über die Leistung eines Lernenden
- beinhaltet formale und informale Methoden
- beinhaltet Self-, Peer-, Group Assessments
- macht Informationen für den Lehrer verfügbar

Bei der Frage, ob die erreichten Ergebnisse in die Note einfließen sollen, unterscheiden sich die Meinungen. Nach [Garrison und Ehringhaus, 2009, S. 2] dürfen die formenden Assessments nur als „Übungen“ gesehen werden und haben keine Auswirkung auf die Endnote. [UK Centre for Legal Education, 2008] widerspricht jedoch dieser Behauptung und sagt, dass es keinen Grund gäbe die Bewertung nicht in die Notengebung einfließen zu lassen.

2.2.2.2 Summatives Assessment

Das summative Assessment bestimmt zu einem gewissen Zeitpunkt das erlernte Wissen [Garrison und Ehringhaus, 2009, S. 1]. Es macht dabei sichtbar welche Lernziele vom Lernenden erreicht wurden und welche nicht [Helen, 2009]. Dieses Assessment reflektiert die Leistung des Lernenden und gibt die Grundlage für die Notengebung.

[Interactive Educational Systems Design, Inc., 2008, S. 8] weist folgende Charakteristika diesen Assessment zu:

- ermöglicht eine Leistungsüberwachung und macht den Lernfortschritt sichtbar

- stellt Informationen verschiedenen Stakeholdern zur Verfügung (Lehrer, Lernender, Universität)

Kritik an dieser Unterscheidung zwischen formativen und summativen Assessments liefert [UK Centre for Legal Education, 2008]. Denn alle Formen von Assessments haben ein formatives Element in sich. Alleine die Durchführung von summativen Prüfungen trainiert auf die Dauer die Fähigkeit eine Prüfung zu machen.

2.2.3 Standardisierte und unstandardisierte Assessments

Ein Assessment kann in Hinblick auf die beinhalteten Test-Elemente zwischen einem standardisierten, halb standardisierten und unstandardisierten Assessment unterschieden werden.

2.2.3.1 Standardisiertes Assessment

Bei einem standardisierten Test wird sowohl die Testsituation (Frage) als auch die Antwort auf die Frage vorgegeben. Somit kann dieses Assessment auch als objektives Assessment bezeichnet werden. Nach [Satow, 2006, S. 92] ist es zwingend, dass für jeden Teilnehmer der Test identisch ist. Dies ist nicht immer zutreffend, da durch Randomisierung der Testelemente zwar eine Standardisierung des Assessments möglich ist, aber jeder Teilnehmer unterschiedliche Tests erhält.
Diese Art von Assessment ist aufgrund der Objektivität voll automatisierbar und somit leicht auszuwerten.

2.2.3.2 Unstandardisiertes Assessment

Im Gegensatz zu den standardisierten Assessments sind hier weder die Testsituation, noch die Antworten einheitlich. Weiters gibt es eine Mischung aus beiden Formen, wenn die Antworten offen sind und nur die Testsituation festgelegt wird. In diesem Fall wird das Assessment als halb standardisiert bezeichnet. Beide Tests sind immer subjektiv zu beurteilen und können dadurch auch nur schwer automatisiert werden. [Satow, 2006, S. 92]

2.2.4 Leistungs-, Diagnose- und Fortschrittstests

Diese Tests werden in Bezug auf die getesteten Merkmale unterschieden. Leistungstests messen dabei das erlernte Wissen und werden meist am Ende des Kurses als summative Assessments durchgeführt.
Diagnosetests identifizieren die Stärken und Schwächen eines Lernenden. Sie stellen dem Lehrer grundlegende Informationen zur Verfügung und bieten ein Instrument zur Kurssteuerung.
Das Ziel eines Fortschrittstests ist den Leistungsfortschritt während eines Kurses zu messen. Die beiden letzten Tests haben einen formativen Charakter. [Helen, 2009]

2.2.5 Übungen und Tests

Um eine Klassifikation, basierend auf [Winkelmann, 2005, S. 6f], von beiden Arten vorzunehmen, muss zunächst eine Definition der Komponenten - Aufgabe und Übungsinteraktion - durchgeführt werden.

Aufgabe - diese Komponente bildet die kleinste Einheit in einer Übung oder einem Test. Aufgaben werden auch als Items verstanden, welche im Abschnitt 2.3.1.1 näher diskutiert werden. Die Aufgabe beinhaltet die Aufgabenstellung, die Antwortmöglichkeiten, die Benutzerinteraktion und die Auswertung (Feedback).

Übungsinteraktion - diese Interaktion tritt speziell bei Übungen auf und stellt die Interaktion zwischen Testperson und Übungsaufgabe dar. Der Umfang dieser Interaktion geht weit über Bedienungs- und Navigationsinteraktionen hinaus und kann als „echte Manipulation oder Veränderung“ verstanden werden. Diese Experimentiersituationen dienen dazu, neues Wissen zu erlernen und bereits Erlerntes zu überprüfen und zu festigen.

Test - diese Art von Assessment setzt sich aus vielen Aufgaben zusammen. Das Ziel eines Tests ist es den Lernerfolg zu überprüfen. Aus diesem Grund dürfen Tests nur auf die Materie fokussieren, welche auch zuvor im Unterricht oder bei Übungen vermittelt wurde. Typische Testfragen bestehen aus Wissens- oder Transferfragen. Der Wiederholungsgrad bei diesen Fragen ist meist null $(WG = 0)$, das heißt eine Frage darf nur einmal beantwortet werden.

Übung - die Übung beinhaltet ebenso eine gewisse Anzahl an Aufgaben. Diese Aufgaben werden aber im Gegensatz zu Tests mit Übungsinteraktionen und Kommunikationsmöglichkeiten erweitert. Der Fokus einer Übung ist die Erarbeitung des Lerninhaltes. Diese Erarbeitung kann auch mit Selbstüberprüfung verbunden sein. Die vorwiegenden Aufgabentypen sind Einschätzungs-, Vermutungs- und Bewertungsfragen. Die Fragen bei Übungen dürfen meist mehrmals wiederholt werden. Der Wiederholungsgrad ist somit n $(WG = n)$.

Die Abbildung 2.3 stellt einen exemplarischen Test und Übungsablauf dar. Zusätzlich wurde auch die Anwendung von Feedback illustriert. Bei Tests wird Feedback erst am Ende der Testdurchführung, in Form der Bewertung oder Punktevergabe, ausgeteilt. Hingegen wird bei Übungen direkt nach jeder Aufgabe ein Feedback durchgeführt. Dieses wird von der Kommunikationskomponente unterstützt.

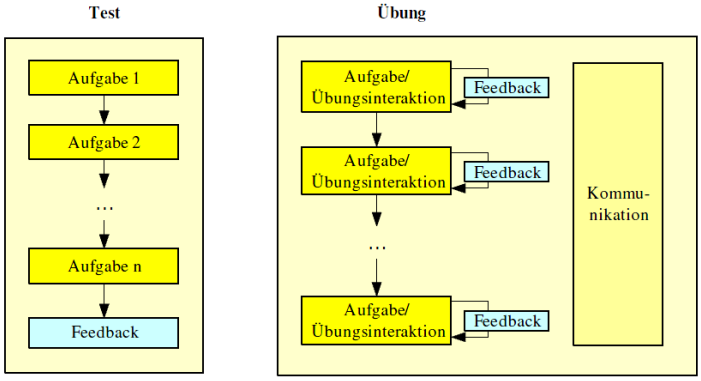


Abbildung 2.3: Ein möglicher Aufbau eines Tests bzw. einer Übung. [Winkelmann, 2005, S. 7]

2.3 Assessment Items

Im letzten Abschnitt wurden die verschiedenen Arten von Assessments betrachtet. Hier wird nun im Speziellen auf den Inhalt solcher Wissensüberprüfungen eingegangen. Der nachfolgende Abschnitt definiert als erstes Begriffe wie Assessment Item und Item Bank. Danach wird eine Reihe von Item Typen gelistet, wobei hier näher auf Multiple Choice Fragen eingegangen wird. Im Anschluss darauf wird das Thema der Parametrisierung von Testfragen untersucht.

2.3.1 Definitionen

2.3.1.1 Test Item

Ein Assessment setzt sich aus Testelementen zusammen. Diese Elemente werden als Test Items bezeichnet. Ein Test Item beinhaltet in der Regel einen Stimulus [Satow, 2006, S. 92]. Dieser kann sich entweder aus einer simplen Frage zusammensetzen oder auch ein Teil einer größeren Struktur sein. Eine solche Struktur besteht aus anderen Stimuli, wie zum Beispiel einer Passage aus einem Test oder anderen Referenzmaterialien [Downing und Haladyna, 2006, S. 264]. Ein Stimulus soll der Testperson eine Resonanz entlocken. Diese Resonanz kann gewertet werden und besitzt entweder einen binären Wert zwischen 1 und 0 oder kann sich in einer Werteskala befinden [Haladyna, 2004, S. 3]. Weiters enthält ein Test Item gewöhnlich eine Instruktion, die eine Anleitung zur Aufgabe darstellt, und die vorgegebenen Antwortmöglichkeiten. Test Items können optional oder auch als Voraussetzung gemeinsam auftreten. Ein Ausschluss eines Test Items durch ein anderes ist ebenso möglich. [Downing und Haladyna, 2006, S. 266] beschreibt hierzu die Gesellschaftsordnung unter Test Items.

2.3.1.2 Item Template

Ein Item Template repräsentiert ein Test Item. Der Unterschied liegt darin, dass es zwar eine Struktur und Template Variablen besitzt, jedoch noch keinen Inhalt. Ein Item Template wird nach [Haladyna, 2004, S. 150ff] auch als „Shell“ bezeichnet. Durch das Einfügen von Inhalt und Deklaration der Variablen werden diese zu Test Items.

2.3.1.3 Item Bank

Die Item Bank ist ein Repository für Test Items. Sie muss dabei zusammen mit einem Item Banker zwei Funktionen erfüllen: die Entwicklung und die Organisation von Items. Die Entwicklung setzt sich dabei aus der Erzeugung, laufenden Änderungen und Erhaltung zusammen. (Vgl. [Downing und Haladyna, 2006, S. 271] und [Foster, 2008])

2.3.2 Itemformate

Es gibt verschiedene Formate von Items. Dieser Abschnitt behandelt zunächst die Differenzierungen dieser Formate. Danach werden diese Formate benutzt um die Grundlage für die folgenden Typ Unterscheidungen der Test Items zu bilden. Zum Abschluss wird noch näher auf zwei Item Typen (MC Fragen und Zuordnung) eingegangen.
Die fundamentalste Unterscheidung zwischen Itemformaten ist die Differenzierung, ob abstraktes Wissen oder konkretes Wissen gemessen wird.
Bei einem Item, das abstraktes Wissen testet, steht nicht eindeutig fest, was genau gemessen wird. Hierzu werden Fachgebietsexperten benötigt, die die erzeugten abstrakten Konstrukte messen und beurteilen. Dadurch sind diese Formate sehr subjektiv und die Inferenz, aus diesem Wissen eine Folgerung zu ziehen, sehr hoch. Haladyna nennt solche Formate auch konstruktionszentriert.
Im Gegensatz dazu steht bei der Messung von konkretem Wissen genau fest, was gemessen wird. Eine Antwort kann somit richtig oder falsch sein. Fachgebietsexperten werden in diesem Fall nicht benötigt und die Inferenz ist sehr niedrig. Die nachfolgende Tabelle gibt einen beispielhaften Überblick, welche Aufgabenstellung die jeweiligen Formate beinhalten können. [Haladyna, 2004, S. 42ff]

Lernen mit hoher Inferenz	**Lernen mit niederer Inferenz**
Zusammenfassen von den wichtigsten Punkten	Anweisungen folgen
Komplexe Texte analysieren	Unterscheiden von Fakten und Meinungen
Einen Bericht mit eigenen Wörtern und eigener Meinung schreiben	26 Buchstaben aus dem Alphabet kopieren
Einen überzeugenden Aufsatz schreiben	Beobachtungen aufzeichnen
Formulierung von Vorhersagen basierend auf Daten	Konstruktion eines Venn Diagramms

Tabelle 2.1: Ergebnisse aus Lernen mit hoher und niederer Inferenz. [Haladyna, 2004, S. 43]

Hayadyna weist folgende Merkmale beiden Formaten in der Tabelle 2.2 zu. Diese Merkmale zielen auf die gemessene Konstruktion, die von der Aufgabe verlangt wird, die Item Erzeugung, die Bewertung und die Reliabilität ab.

Merkmale	hohe Inferenz	niedere Inferenz
Gemessenes Konstrukt	komplexe Fähigkeiten, Fähigkeiten von abstrakter Natur	Wissen, Kognition und Fähigkeiten von konkreter Natur
Leichtigkeit der Item Erzeugung	Items sind sehr komplex	Items nicht so komplex
Bewertung	kann sehr kostenintensiv sein, SMEs müssen die Arbeit bewerten	kann teilweise durch automatisiert oder durch reine Beobachtungspersonen bewertet werden
Bewertungstyp	subjektiv	objektiv
Bewertungseffekt	kann eine Gefahr für die Validität sein, da subjektiv bewertet wird	keine Gefahr für die Validität
Reliabilität	Reliablität ist ein Problem aufgrund von Bewertungsinkonsistenzen	Ergebnisse sind sehr zuverlässig

Tabelle 2.2: Merkmale von Itemformaten mit hoher und niederer Lerninferenz. [Haladyna, 2004, S. 45]

[Bennett, 1993, S. 47] zit. nach [Scalise und Gifford, 2006] benutzt eine andere Sicht auf Test Items. Er definiert sechs Grundformate, die nicht in Hinblick auf das getestete Wissen unterteilt werden, sondern durch die Art und Weise wie getestet wird.

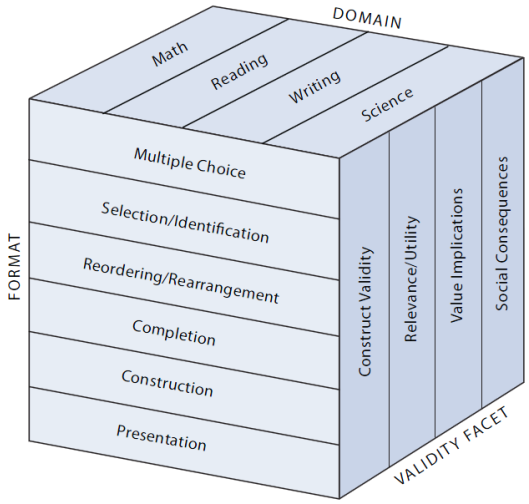


Abbildung 2.4: Vielseitiges Organisationsschema der sechs Aufgaben Grundformate. [Bennett, 1993, S. 47] zit. nach [Scalise und Gifford, 2006, S. 6]

2.3.3 Item Typen

Die Anzahl an Item Typen hat sich durch die Unterstützung von der IT in den letzten Jahren vervielfacht. Viele Item Arten wurden zwar nicht zur Gänze neu erfunden, doch durch die Informationstechnologie in dem einen oder anderen Punkt ergänzt. Durch Multimediale- und Interaktionsmöglichkeiten entstanden auch neue Formen von Items. [Hambleton und Pitoniak, 2002] zit. nach [Hambleton, 2004, 699] identifizierten mehr als 50 verschiedene Arten.

Die nachfolgende Matrix 2.5 baut auf diesen Grundformaten auf und bildet 28 ausgewählte Item Arten in einer Taxonomie, basierend auf dem Grad der Beschränkung des Antwortformates, ab. Die Antwort der Items in der linken Spalte ist voll selektierbar. In diesem Fall ist dieses Antwortformat am meisten beschränkt. Diese Beschränkung nimmt von links nach rechts ab. Zusätzlich wird in dieser Matrix noch die Komplexität der einzelnen Items abgebildet. Je weiter unten das Item platziert ist, desto höher ist dessen Komplexität.

Most Constrained → *Least* Constrained

Less Complex ↓	*Fully Selected*	*Intermediate Constraint Item Types*					*Fully Constructed*
	1. Multiple Choice	**2. Selection/ Identification**	**3. Reordering/ Rearrangement**	**4. Substitution/ Correction**	**5. Completion**	**6. Construction**	**7. Presentation/ Portfolio**
	1A. *True/False* (Haladyna, 1994c, p.54)	2A. *Multiple True/False* (Haladyna, 1994c, p.58)	3A. *Matching* (Osterlind, 1998, p.234; Haladyna, 1994c, p.50)	4A. *Interlinear* (Haladyna, 1994c, p.65)	5A. *Single Numerical Constructed* (Parshall et al, 2002, p. 87)	6A. *Open-Ended Multiple Choice* (Haladyna, 1994c, p.49)	7A. *Project* (Bennett, 1993, p.4)
	1B. *Alternate Choice* (Haladyna, 1994c, p.53)	2B. *Yes/No with Explanation* (McDonald, 2002, p.110)	3B. *Categorizing* (Bennett, 1993, p.44)	4B. *Sore-Finger* (Haladyna, 1994c, p.67)	5B. *Short-Answer & Sentence Completion* (Osterlind, 1998, p.237)	6B. *Figural Constructed Response* (Parshall et al, 2002, p.87)	7B. *Demonstration, Experiment, Performance* (Bennett, 1993, p.45)
	1C. *Conventional* or *Standard Multiple Choice* (Haladyna, 1994c, p.47)	2C. *Multiple Answer* (Parshall et al, 2002, p.2; Haladyna, 1994c, p.60)	3C. *Ranking & Sequencing* (Parshall et al, 2002, p.2)	4C. *Limited Figural Drawing* (Bennett, 1993, p.44)	5C. *Cloze-Procedure* (Osterlind, 1998, p.242)	6C. *Concept Map* (Shavelson, R. J., 2001; Chung & Baker, 1997)	7C. *Discussion, Interview* (Bennett, 1993, p.45)
↓ More Complex	1D. *Multiple Choice with New Media Distractors* (Parshall et al, 2002, p.87)	2D. *Complex Multiple Choice* (Haladyna, 1994c, p.57)	3D. *Assembling Proof* (Bennett, 1993, p.44)	4D. *Bug/Fault Correction* (Bennett, 1993, p.44)	5D. *Matrix Completion* (Embretson, S, 2002, p. 225)	6D. *Essay* (Page et al, 1995, 561-565) & *Automated Editing* (Breland et al, 2001, pp.1-64)	7D. *Diagnosis, Teaching* (Bennett, 1993, p.4)

Abbildung 2.5: Die Itemformat - Taxonomie nach [Scalise und Gifford, 2006, S. 9].

2.3.4 Ziele von Test Items

Alle Formate und deren einzelne Test Item Typen verfolgen die gleichen Ziele. Sie messen die geforderten Fähigkeiten oder Informationen und sind dabei in der Formulierung klar und präzise definiert. Test Items fokussieren auf den wichtigen, nicht trivialen Inhalt eines Sachgebietes. Sie beinhalten die passende Information zur Lösung der Aufgabe und werden in einer geeigneten Schwierigkeitsstufe verfasst. Ebenso müssen Test Items immer korrekt sein. Auf diese Validität wird im 2.7.3 noch näher eingegangen. [Kelly, 2008]

2.3.5 Multiple Choice Fragen - eine exemplarische Untersuchung eines Test Item Typs

Die Auswahl von Test Item Typen ist nicht einfach, da es viele Vor- und Nachteile der verschiedenen Arten gibt. Auch die existierenden Regeln sollten bei deren Verwendung beachtet werden, um effektive Fragestellungen zu erzeugen. Der folgende Abschnitt beschreibt exemplarisch Multiple Choice Fragen und setzt gewisse Anwendungsregeln hinsichtlich der Effektivität und Validität fest. Ebenso wird die Kritik von Multiple Choice Fragen diskutiert. Zusätzlich werden Hilfestellungen gegeben, die die Komplexität dieser Fragen bei formativen Assessments reduzieren können.

2.3.5.1 Definition

Multiple Choice Fragen können als Tripel $\langle \Theta, R, r_{ok} \rangle$ beschrieben werden. Bei diesem Tripel stellt Θ die Menge der Antworten auf die gestellte Frage dar. R bildet die Menge der möglichen Teilmengen die als Antwort gewählt werden können, so dass $R \subseteq 2^{\Theta}$. Die Teilmengen werden als r_i bezeichnet. Die korrekte Antwort wird als r_{ok} definiert und ist ebenso $r_{ok} \in R$ [Diaz et al., 2007, S. 2]. [Woodford

und Bancroft, 2005, S. 110] bezeichnen r_{ok} auch als „key“ und alle falschen Antwortmöglichkeiten als „distractor“. Nach dem Autor dieser Arbeit fehlt in dieser Definition das Q, welches die Frage an sich repräsentiert. Somit entsteht folgende Definition $\langle Q, \Theta, R, r_{ok} \rangle$.
Es werden zwei Typen von Multiple Choice Fragen unterschieden, zum einen MCQ-SR (Single Response) und zum anderen MCQ-MR (Multiple Response). MCQ-SR besitzen nur eine einelementige Menge von Antwortmöglichkeiten, d.h.: $R = \Theta$. Hierzu zählen Richtig/Falsch Fragen. Bei MCQ-MR hingegen setzt sich R aus verschiedenen Teilmengen aus Θ zusammen. Die richtige Antwort r_{ok} besteht dann auch aus einer Teilmenge von Θ. Die folgende Abbildung 2.6 veranschaulicht diese Definition. [Diaz et al., 2007, S. 2]

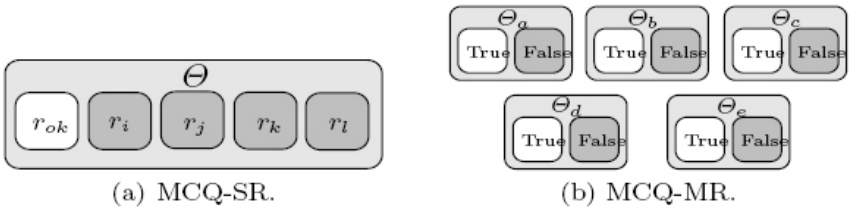


Abbildung 2.6: Antwortmöglichkeiten von Multiple Choice Fragen (MCQ-SR und MCQ-MR). [Diaz et al., 2007, S. 2]

2.3.5.2 Messung der Effektivität von Multiple Choice Fragen

Die Effektivität von Multiple Choice Fragen kann durch die Analyse der gegebenen Antworten von Studenten untersucht werden. Werden zu viele richtige Antworten gegeben, so kann darauf geschlossen werden, dass die „distracter“ zu schwach formuliert wurden. Treten hingegen zu viele falsche Antworten auf, so liegt es nahe, dass die Fragen nicht klar definiert wurden, oder die „distracter“ in die Irre leiten. Das Verhältnis zwischen richtigen und falschen Antworten, die durch Studenten abgegeben werden, nennt man auch „facility“ oder Leichtigkeit. In der Regel liegt bei Einstufungstests der Wert zwischen 0,4 und 0,6. Bei Prüfungen soll dieser Wert größer oder gleich 0,8 sein (Vgl. [Beasley et al., 1994] zit. nach [Woodford und Bancroft, 2005, S. 110]). Eine andere Möglichkeit die Effektivität zu messen besteht darin, festzustellen ob der gewünschte Grad der Kognition getestet wird. Kritiker behaupten, dass Multiple Choice Fragen nur Faktenwissen prüfen und somit nicht tiefere kognitive Grade testen. Der Abschnitt 2.3.5.3 widerspricht jedoch dieser Kritik [Woodford und Bancroft, 2005, S. 109]. Eine weitere Limitierung, der traditionelle Multiple Choice Fragen unterliegen, ist die Tatsache, dass nicht notwendigerweise das ganze Wissen über ein Subjekt getestet wird. Wenn eine richtige Antwort gegeben wird, kann zum Beispiel nicht darauf geschlossen werden, dass der Lernende alle anderen Antworten ausschließen kann. In Hinblick auf die Problematik wird von [Woodford und Bancroft, 2005, S. 110] eine hybride Version von Multiple Choice Frage vorgeschlagen, die in Abschnitt 2.3.5.3 behandelt wird.

2.3.5.3 Faktoren um die Validität zu beeinflussen

Die folgenden Faktoren beziehen sich entweder auf die Frage, den Stimulus selbst oder auf die Antwortmöglichkeiten, die Optionen. Teilweise sind diese Faktoren nicht eindeutig anwendbar, doch helfen die gegebenen Vor- und Nachteile die Verwendung dieser abzuwägen.

Richtige Grammatik und Formulierung
Durch eine falsch angewendete Grammatik oder Formulierung, kann die Testperson auf die richtige Antwort schließen. Um das zu verhindern, muss vor allem auf die gleichen Zeiten geachtet werden. Ebenso darf keine andere Formulierung für die richtige Antwort verwendet werden. Das heißt, es dürfen keine zusätzlichen qualifizierenden Wörter, höhere Anzahl an Wörtern oder auch Passagen aus einem Lehrbuch verwendet werden. (Vgl. [Learning Technologies at Virginia Tech., 2009] und [Woodford und Bancroft,

2005, S. 110])

Anzahl der vorgegebenen Optionen
In diesem Punkt gibt es eine intensive Diskussion zwischen 3, 4 und 5 Optionen. Die Vertreter der Anzahl von 3 Optionen argumentieren, dass es gleich effektiv zu einer höheren Anzahl sei. Ebenso wäre die Zeitersparnis bei der Erzeugung und Durchführung von Prüfungen enorm. Das signifikanteste Argument für eine höhere Anzahl von Antwortmöglichkeiten ist die damit verbundene fallende Wahrscheinlichkeit zum Erraten der richtigen Antwort (3 Optionen = 33%, 5 Optionen = 20%). (Vgl. [Learning Technologies at Virginia Tech., 2009] und [Woodford und Bancroft, 2005, S. 111])

Mehrfache richtige Antworten - die hybride Multiple Choice Frage
Wie bereits in Abschnitt 2.3.5.2 erwähnt, ist dieser Item Typ der Kritik ausgesetzt, dass eine Testperson die Antwort erraten kann. Um dem entgegenzuwirken, muss es mehrere richtige Antworten geben, um somit nicht nur die Fakten, sondern auch das Verständnis der Materie zu prüfen. Mehrere richtige Antworten erzeugen aber auch Verwirrung bei den Testpersonen und eine daraus resultierende Ineffektivität. Aus diesem Grund gibt es eine hybride Form, die zusätzlich eine Liste von möglichen Lösungskombinationen angibt. Ein Beispiel könnte wie folgt aussehen [Woodford und Bancroft, 2005, S. 111]:

Welche von den folgenden Statements beschreibt ein Java Interface?

1. *Es definiert „was“, nicht „wie“*
2. *Es muss den default constructor definieren*
3. *Es muss alle Methoden als abstract definieren*
4. *Es kann eine andere Interfaceklasse erweitern*

Antwortkombinationen:

1. *(a) 1, 4*
2. *(b) 1, 2*
3. *(c) 2, 4*

Plausible Distraktoren
Schwache Distraktoren verhelfen der Testperson dazu, schnell die richtige Antwort zu finden. Deshalb ist es wichtig, diese ähnlich der richtigen Antwort zu formulieren, um somit eine gewisse Komplexität in die Frage zu bringen. [Woodford und Bancroft, 2005, S. 111]

Mehr als nur der Abruf von Fakten
Ein weiterer Kritikpunkt besteht darin, dass Multiple Choice Fragen nur Faktenwissen prüfen. Dieser Fragetypus findet nur in formativen Assessments Platz, da in summativen Tests mehr als nur eine Auflistung von Fakten geprüft wird. Nach Blooms Grad der Kognition spiegelt dieses Faktenwissen die unterste Stufe der Pyramide wider, wie in Abbildung 2.7 zu sehen ist. Dem widersprechen jedoch [Woodford und Bancroft, 2005, S. 112] mit der Aussage, dass Multiple Choice Fragen auch Informationen aus tieferen kognitiven Ebenen hervorholen.

Die Prüfung von Verständnis wird zum Beispiel durch Fragen durchgeführt, die das Verstehen von Informationen erfordern. Hierbei muss Wissen in einen neuen Kontext übersetzt, Fakten müssen interpretiert und Konsequenzen vorhergesagt werden.

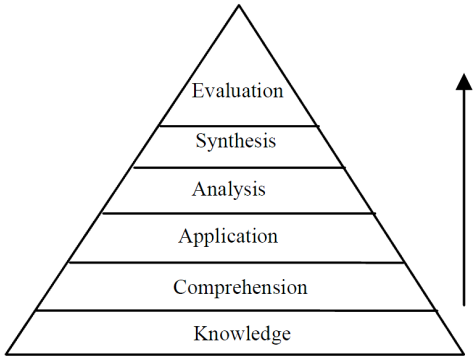


Abbildung 2.7: Bloom's kognitive Ziele. [Woodford und Bancroft, 2005, S. 109]

Der Grad der Anwendung erfordert die Fähigkeit, Probleme durch Anwendung von Wissen, Fakten und Techniken zu lösen. Um hier eine Multiple Choice Frage zu erstellen, kann zum Beispiel die Anwendung eines Algorithmus erfragt und die möglichen Resultate als Optionen gelistet werden.
Zur Prüfung des Analyselevels, wird die Fähigkeit, Informationen in Teile zu zerlegen, Muster darin zu erkennen und Beziehungen zu verstehen, getestet. [Woodford und Bancroft, 2005, S. 112] geben in ihrer Arbeit zu den oben genannten Ebenen Beispiele, die noch näher auf das Thema eingehen.

2.3.5.4 Hilfestellungen bei Multiple Choice Fragen (Ask-Hint Strategie)

Wenn die Komplexität von Multiple Choice Fragen zu hoch gewählt wurde und der Wert der „facility" nach unten sinkt, kann es passieren, dass viele Testpersonen demotiviert werden. Hierfür wurde speziell die Ask-Hint Strategie entwickelt. Sie erlaubt Testpersonen, die Schwierigkeiten bei der Beantwortung von Fragen haben, Hilfestellung in Anspruch zu nehmen. Ask-Hint setzt sich aus der Prune Strategy und der Call-In Strategy zusammen. [Wang, 2008, S. 1252]

Prune Strategie
Nach [Rodriguez, 2005] zit. nach [Wang, 2008, S. 1252] wird der Schwierigkeitsgrad von Multiple Choice Fragen durch die Anzahl der möglichen Optionen beeinflusst. Werden die Antwortmöglichkeiten reduziert, so wird auch die Komplexität zu einem gewissen Grad reduziert. Die Prune Strategie baut darauf auf, indem sie durch einen Befehl der Testperson die Auswahl der möglichen Antworten um eins reduziert.

Call-In Strategie
Aus soziologischer Sicht tendieren Personen, die keine klare Idee über die Lösung eines Problems haben, zur Imitation anderer. Diesen Umstand fördert die Call-In Strategie, wobei die Testperson die Antwortrate der vorliegenden Multiple Choice Frage anzeigen lassen kann. Das heißt, es wird dargestellt, wie oft eine Antwort bei dieser Frage von vorherigen Testpersonen ausgewählt wurde. [Wang, 2008, S. 1252]

Der Vorteil dieser Hilfestellungen liegt darin, dass bei formativen Assessments Lernende motiviert werden können. Zusätzlich können Informationen aus der Antwortrate einer Testfrage entnommen werden, die auf die Effektivität der Frage hinweisen. Ein sichtlicher Nachteil ist jedoch, dass es ausschließlich für formative und nicht für summative Assessments verwendet werden kann.

2.4 Parametrisierung von Test Items

Viele Prüfungssysteme unterstützen nur die Erstellung von statischen Testfragen. Sie bieten zwar die Möglichkeit die Test Fragen in Item Repositories zu speichern, jedoch ist die Generierung dieser Test Items sehr zeitaufwendig. Aus diesem Grund sind häufig nur sehr wenige Fragen in solchen Systemen vorhanden. Ein Problem, das dadurch entsteht, ist das ständige Durchlaufen der gleichen statischen Fragen. Diese Problematik führt zu vermehrtem Schwindeln in Prüfungssituationen und somit gleichzeitig zu einer schlechten Lernperformance bei den Studenten.
Abhilfe schaffen hier parametrisierte Testfragen. Diese Testfragen sind Test Item Templates mit speziell definierten Parametern. Zur Präsentationszeit werden diese Items mit einer zufälligen Auswahl, den zuvor definierten Parametern, instanziiert. Daraus resultierend kann durch eine kleine Anzahl an parametrisierten Fragen eine hohe Anzahl an verschiedenen eigentlichen Fragestellungen erzeugt werden. (Vgl. [Brusilovsky und Sosnovsky, 2005b, S. 1f] und [Brusilovsky und Sosnovsky, 2005a, S. 252])
Bei einer Multiple Choice Frage können zum Beispiel mehrere einzelne Fragen definiert werden, aus denen jedes Mal eine zufällig zur Präsentation ausgewählt wird. Zusätzlich werden auch die unten angeführten Optionen durchmischt.

```
1  [{SpMCQ
2  |
3  frage='Der ____ versetzt Berge.'
4  ok=1
5  options=('Gedanke';'Hinweis';'Glaube')
6  |
7  frage='Der ____ fällt nicht weit vom Stamm.'
8  ok=2
9  options=('Apfel';'Birne';'Melone')
10 |
11 }]
```

Listing 2.1: Template eines Multiple Choice Items

Vorteile, die sich aus dieser Technik ergeben, sind vor allem eine gesteigerte Motivation, die Verhinderung von Mogeleien und die Individualisierung von Prüfungen. Durch das Wiederholen gleicher Fragen entsteht oft Langeweile bei den Testpersonen, eine Abänderung dieses Zustandes führt somit direkt zu einer Erhöhung der Motivation. Ebenso wird durch eine kleine Anzahl an Testfragen das Auswendiglernen gefördert und die Schwindelgefahr erhöht. Auch hier wirken parametrisierte Fragen entgegen und fördern gleichzeitig die Lernperformance. Ein weiterer Vorteil ist, dass diese Art von Items wiederverwendbar ist. [Brusilovsky und Sosnovsky, 2005b, S. 1f]

2.5 Überlegungen zur Punktevergabe

Die Punkteauswertung verläuft nach einem einfachen Schema: Jedes Item besitzt eine Punkteanzahl, und die erreichten Punkte werden am Ende einer Prüfung zusammengezählt. Im Detail kann dieses Prinzip jedoch komplexer werden.

2.5.1 Anzahl der Punkte für eine Aufgabe

Die einfachste Lösung ist, jeweils einen Punkt pro Frage zu vergeben. Somit sind Prüfungen für Testpersonen viel transparenter. In häufigen Fällen ist dies aber nicht möglich, da nicht alle Fragen gleichwertig sind. Hierzu müssen mehrere Punkte nach gewissen Kriterien vergeben werden. Laut [Satow, 2006, S. 102f] sollte das aber nicht nach dem Kriterium der Komplexität einer Frage geschehen, sondern nach der Wichtigkeit im Lerngebiet. Schwere Aufgaben sollten als Zusatz-Aufgaben gehandhabt werden.

2.5.2 Punkte für teilweise richtige Aufgaben

Die Vergabe von Punkten bei nur teilweise richtig beantworteten Fragen sollte auf alle Fälle geschehen. Das ermöglicht eine genauere Unterscheidung der Leistungen von Testpersonen.

2.5.3 Minus-Punkte

Diese Art von Punkten hat den Vorteil, dass sie Rateversuche bei Multiple Choice Fragen verhindert. Ein gravierender Nachteil ist jedoch, dass richtige Fragen neutralisiert werden. Das heißt, wenn zum Beispiel eine Testperson viel weiß, aber auch vieles falsch beantwortet, bekommt sie wenig Punkte. Am Ende erhält somit eine andere Testperson, die nur wenig weiß, aber aus Vorsicht im Zweifel keine Versuche tätigt, mehr Punkte als die andere Testperson. Minuspunkte bestrafen somit Testpersonen, die weniger vorsichtig bei der Beantwortung von Fragen sind.

2.5.4 Beta-Aufgaben

Beta Aufgaben sollten nur zu einem sehr geringen Prozentsatz (1-2%) richtigen Tests hinzugefügt werden. Sie dürfen dabei nicht in das Endergebnis eines Tests mit einfließen, sondern müssen sich erst auf die Dauer im Einsatz bewähren, bis sich herausstellt, dass sie das gewünschte Lernziel überprüfen und deren Komplexität ausreichend ist.

2.6 Feedback

Das Feedback und die damit verbundene Bewertung bilden eine zentrale Komponente in einem Assessment. Durch das Feedback können dem Lernenden sofort oder am Ende einer Lernsequenz Informationen über dessen Lernerfolg zur Verfügung gestellt werden. Diese Information dient vorwiegend als Informations-, Verzweigungskomponente und ebenso als Motivation für die Testperson. Der Aufbau und die Organisation einer Feedbackstruktur sind somit elementar für eine Lernanwendung.
Bei elektronischen Tests kann Feedback in verschiedenen Medien übermittelt werden. Die einfachste Form bildet das textbasierte Feedback, zudem ist es aber auch möglich durch Grafik, Farben, Audio, Video oder verschiedene Animationen diese Informationen dem Lernenden zur Verfügung zu stellen. Der Kreativität sind hier wenig Grenzen gesetzt.
Feedback kann in verschiedene Typen eingeteilt werden. Diese Arten werden in den folgenden Abschnitten näher behandelt. Die hierfür verwendete Quelle ist [Winkelmann, 2005, S. 11ff].

2.6.1 Automatisiertes Feedback

Der Vorteil von automatisiertem Feedback ist die Eigenschaft, dass es schnell und direkt durch das System vorgenommen werden kann. Das System kann hierbei sofort auf die Interaktion zwischen Lernenden und Aufgabe reagieren. Dieser Vorgang kann dem Kandidaten unabhängig von seiner Lernsituation den derzeitigen Lernstand übermitteln. Somit kann sich der Lernende bei Übungen schnell auf die Situation einstellen, ob er einen Lerninhalt nochmals durcharbeiten muss oder eine neue Lektion beginnen darf. Auch bei Tests ist diese Form von Feedback einsetzbar. Der Vorteil liegt darin, dass ein Kandidat sofort über sein Ergebnis Bescheid weiß.
Ein Nachteil ist aber der Verlust der Individualisierung und der Subjektivität. Auch wenn eine hohe Komplexität durch verschiedene Lerneingaben möglich ist, so ist diese trotzdem endlich. Durch eine weitere Differenzierung ist zwar eine Erhöhung der Individualisierung möglich, doch steigen dadurch auch die Entwicklungskosten. Ein weiteres Problem ist, dass dieses Feedback nur bei objektiv beantwortbaren Fragestellungen verwendet werden kann.

Automatisiertes Feedback wird in zwei Arten unterteilt: einfach automatisiertes Feedback und differenziert automatisiertes Feedback.

2.6.1.1 Einfaches automatisiertes Feedback

Bei dieser Art von automatisiertem Feedback muss das System lediglich zwischen zwei möglichen Antwortalternativen unterscheiden. Die Antwort kann somit richtig oder falsch sein. Darauf folgend kann das System auf eines der zwei festdefinierten Rückmeldungen zurückgreifen. Das einfache Feedback ist unabhängig vom Wiederholungsgrad und der Lernsituation.
Ein Vorteil dieses Feedbacks ist seine Einfachheit, da es nur zwei Alternativen gibt. Somit kann es aber nur bei Tests eingesetzt werden die Wissen überprüfen und keinen formativen Charakter besitzen.
Bei diesem Feedback ist außerdem wichtig, dass bei einer inkorrekten Antwort nicht nur das Faktum „falsch“ angezeigt wird, sondern auch die Musterlösung präsentiert wird.

2.6.1.2 Differenziert automatisiertes Feedback

Dieses Feedback gibt als Rückmeldung den Grad der Übereinstimmung zwischen der Lernerlösung und der Musterlösung an. Es unterscheidet sich dabei vom einfachen Feedback indem es auf mehrere Antwortmöglichkeiten differenziert reagieren kann.
Das System muss dabei nicht auf jede einzelne Antwortmöglichkeit ein Feedback liefern, sondern kann zur Vereinfachungen auch Gruppierungen vornehmen. Die Tabelle 2.8 zeigt ein differenziertes Feedback auf eine Multiple Choice Frage mit drei Antwortmöglichkeiten und nimmt dabei eine Gruppierung vor:

Antwort-alternativen	A	B	C		D			E
A1 (richtig)		x	x		x	x		
A2 (richtig)		x		x		x	x	
A3 (falsch)					x	x	x	x
Feedback	Bitte treffen Sie eine Auswahl!	Sehr gut! Ihre Lösung ist richtig.	Da haben Sie etwas übersehen. Überprüfen Sie ihre Auswahl.		Sie haben richtige und falsche Antworten ausgewählt.			Ihre Lösung ist falsch.

Abbildung 2.8: Differenziertes Feedback bei einer MC-Aufgabe mit 3 Antwortalternativen. [Winkelmann, 2005, S. 13]

2.6.2 Nicht automatisiertes Feedback

Nicht jeder Inhalt oder jedes Lernziel kann durch objektive Fragestellungen abgedeckt werden. Auch kann nicht jeder Level der Kognition durch diesen Fragetypus geprüft werden. Aus diesem Grund müssen gegebenenfalls auch subjektive Lernfragen gestellt werden. Diese Lernfragen können aber nur durch subjektive Beurteilung bewertet werden. Hierbei gibt es zwei Möglichkeiten - die Selbstbewertung und die Bewertung durch einen Tutor.

2.6.2.1 Selbstbewertung

Bei einer Selbstbewertung muss dem Lernenden ermöglicht werden, selbst seinen Arbeits- und Lernprozess, sowie seine abgelieferte Qualität zu beurteilen. Im Abschnitt 2.2.1.1 wurde diese Thematik näher

beschrieben. Eine wichtige Rolle spielt dabei die Fähigkeit des Kandidaten, sich selbst zu prüfen. Um diesen Typ des Feedbacks zu vervollständigen, muss auch die Möglichkeit von Peer und Group Assessments erwähnt werden.

2.6.2.2 Feedback durch eine Lehrperson

Ein Lerner besitzt oft nicht die Fähigkeit Selbstbewertungen durchzuführen. Aus diesem Grund muss eine Lehrperson den Lernprozess begleiten. Diese Person übernimmt die Aufgabenbetreuung und die Bewertung von subjektiven Antworten.

2.7 Qualitätsmerkmale von Assessments

Qualität ist eine essentielle Komponente um effektive Assessments zu erzeugen. In diesem Abschnitt werden die wichtigsten Qualitätsanforderungen dargestellt.

2.7.1 Standardisierung und Objektivität

Vergleichende Aussagen (Person A ist besser als Person B) können nur getroffen werden, wenn unter gleichen Bedingungen (Testsituation, Fragen) getestet wird. Diese Objektivität ist jedoch nicht immer zu erreichen, da auch abstraktes Wissen getestet wird, welches eine Subjektivität in der Beurteilung erfordert. [Satow, 2006, S. 94]

2.7.2 Testzuverlässigkeit (Reliabilität)

Die Testzuverlässigkeit kann in mehreren Arten unterschieden werden. Die nachfolgende Auflistung 2.3 gibt einen Überblick dieser Typen.

Typ der Zuverlässigkeit	Definition
Zeitliche Zuverlässigkeit (Test- Retest)	Unter dieser Zuverlässigkeit versteht man, dass bei wiederholtem Testen einer Person, ohne das Messmerkmal zu verändern, immer wieder ein korrelierendes Ergebnis erzeugt wird [Pinellas School District]. Das Messen dieser Komponente stellt jedoch eine Problematik dar. Denn Testpersonen adaptieren das Test Format und tendieren so zu besseren Ergebnissen in späteren Prüfungen [Chong, 2008].
Formgleichheit	Zwei alternierende Versionen eines Tests, die zwar auf derselben Materie aufbauen, sich aber bei den Fragen ein wenig unterscheiden, müssen ein korrelierendes Ergebnis liefern. (Vgl. [Chong, 2008] und [Pinellas School District])
Innere Konsistenz	Die innere Konsistenz wird gemessen in dem eine Hälfte des Tests mit der anderen Hälfte verglichen wird. Methoden die hier verwendet werden, sind: Cronbach Alpha, KR20 oder Split-Half. (Vgl. [Chong, 2008] und [Pinellas School District])

Tabelle 2.3: Überblick der Testzuverlässigkeit.

2.7.3 Testgültigkeit (Validität)

Unter Testgültigkeit versteht man die Genauigkeit eines Tests. Hierbei wird gemessen, ob der Test das geforderte Wissen prüft oder nicht. Die Testzuverlässigkeit bildet hier eine Grundvoraussetzung für die Gültigkeit. In umgekehrter Richtung trifft das jedoch nicht zu. Wenn zum Beispiel ein Test zuverlässig ist (eine Waage misst jedes Mal bei einer Person 80 kg), heißt das nicht gleichzeitig, dass dieses Ergebnis stimmt (denn die Person wiegt nur 70 kg). Es werden verschiedene Arten von Testgültigkeit bewertet, eine Auflistung gibt die nachfolgende Tabelle 2.4 (Vgl. [Chong, 2008] und [Pinellas School District]):

Typ der Gültigkeit	Definition
Inhalts-Validität	Das abgefragte Wissen, welches durch die Testfragen geprüft wird, muss sich mit den Lernzielen decken. Es darf zum Beispiel ein Ganzjahrestest nicht nur die Materie der letzten sechs Wochen testen. Ein solcher Test ist invalide. [Pinellas School District]
Kriterien-Validität	Diese Validität wird durch Vergleich mit einem beobachtbaren Kriterium gemessen. Die Korrelation zwischen einem beobachteten Verhalten und einer Vorhersage wird Vorhersage-Validität genannt. Wird das beobachtbare Verhalten und die dazu in Korrelation gesetzte Messung gleichzeitig gemessen, so wird das als Übereinstimmungs-Validität bezeichnet [Wübbenhorst]. Ein Beispiel liefert hier die Matura eines bestimmten Gymnasiums, das eine hohe Korrelation mit den gesamten Matura der Bundeslandes haben muss (Übereinstimmungs-Validität).
Konstrukt Validität	Die Konstrukt Validität bildet das Ausmaß inwieweit sich die Prüfung mit verschiedenen Variablen, die durch eine Theorie definiert wurden, deckt. [Pinellas School District]

Tabelle 2.4: Überblick der Testgültigkeit.

2.7.4 Fairness

Testpersonen müssen immer fair behandelt werden. Keiner darf bevorzugt oder vernachlässigt werden. Ein Test darf ebenso nicht die physische oder psychische Gesundheit von Testpersonen verletzen. [Satow, 2006, S. 94]

2.7.5 Ökonomie

Bei der Gestaltung von Tests muss immer auf die Ökonomie geachtet werden. Das bedeutet, dass Prüfungen kostengünstig und schnell erstellt, angewendet und ausgewertet werden können. [Satow, 2006, S. 95]

2.8 Konklusion

Dieses Kapitel dient vorwiegend dazu, die Begrifflichkeiten rund um das Thema Assessments zu klären und gibt einen Einblick in den derzeitigen Stand der Forschung. Es bildet gleichzeitig die Grundlage für die folgenden Kapitel und soll als Einführung in die Thematik verstanden werden. Zudem werden Anregungen und Empfehlungen zur Testerzeugung, Testdurchführung und Testevaluierung getätigt. Zusätzlich wird der Aspekt der Parametrisierung von Testfragen diskutiert und generelle Qualitätsmerkmale wie Testzuverlässigkeit und Testgültigkeit aufgezeigt.

Kapitel 3

IMS Question and Test Interoperability (QTI)

„Wer etwas Großes will, der muss sich zu beschränken wissen, wer dagegen alles will, der will in der Tat nichts und bringt es zu nichts.“

[Georg Wilhelm Friedrich Hegel (1770 - 1831 v. Chr.), deutscher Philosoph]

3.1 Was ist QTI?

Question and Test Interoperability (QTI) ist eine Spezifikation von IMS Global Learning Consortium. Die Spezifikation beschreibt ein Datenmodell zur Repräsentation von Fragen (*assessmentItem*), Tests (*assessmentTest*) und den damit zusammenhängenden Ergebnisdarstellungen. QTI bildet einen offenen Standard, der Interoperabilität und Kommunikation zwischen verschiedenen Systemen, wie zum Beispiel: System zur Erstellung von Items, Item Datenbanken, Lernsysteme und Systeme die Assessments zur Verfügung stellen, fördert und standardisiert. Das Datenmodell von QTI ist abstrakt durch UML definiert, was eine weite Verwendung von Modellierungstools und Programmiersprachen ermöglicht. Das Austauschformat für Daten basiert auf den Industrie Standard eXtensible Markup Language (XML). Der Standard wurde so konzipiert, dass dieser nicht nur Interoperabilität fördert, sondern auch Innovation. Es ist somit möglich, dass an klar definierten Stellen Erweiterungen durchgeführt werden können. Erweiterungen können genutzt werden um proprietäre Daten zu verpacken. Durch diese Erweiterungen wird gewährleistet, dass auch stark spezialisierte Systeme den Standard unterstützen können. [IMS Global Learning Consortium, 2006c]

Die Hauptelemente des QTI Datenmodells bestehen aus:

- *AssessmentItem*: Das Assessment Item bildet das Kerndaten-Objekt in der QTI Spezifikation. Es beinhaltet die Fragen, die Antwortmöglichkeiten, Informationen zu der richtigen Antwort, Feedback, Hilfestellungen und Bewertungsszenarien. Es bildet die kleinste Einheit in der Spezifikation und kann keine Verschachtelungen, also keine weiteren Items, beinhalten. [Brinke et al., 2005, S. 190]
- *Section*: Die Section repräsentiert den Komposite Teil eines Assessment und kann weitere Sections und/oder Items enthalten. [Bouzo et al., 2007, S. 2]
- *AssessmentTest*: Dieses Element stellt einen einzelnen Test dar. Es beinhaltet eine Sammlung von Assessment Items, die eine Testperson zu einem speziellen Sachgebiet beantworten muss. Die Struktur eines Assessment Objektes setzt sich aus Sections und Subsections zusammen. Weiters

beinhaltet es sequentielle Informationen zur Errechnung des Endergebnisses, aus den Teilergebnissen der Antworten. [Bouzo et al., 2007, S. 2]

3.2 Von proprietären Entwicklungen zu einem offenen Standard

Mit dem Aufkommen der IT und des World Wide Web in den 1980er Jahren entstand auch die elektronische Form von Wissensprüfungen, so genannte E-Assessments. Diese computerbasierten Prüfungen brachten jedoch nicht nur eine Vielzahl an Vorteilen mit sich wie zum Beispiel Kostensenkung, verbesserte Testreliablität und Item Erweiterungen, sondern auch Nachteile, wie zum Beispiel spezielle proprietäre Entwicklungen am Markt. Durch diese eigenen Entwicklungen und Standards wurden Kunden an Produzenten solcher Applikationen stark gebunden und die Innovationen auf diesem Sektor wurden teilweise erstickt. Es fehlten verschiedene Anbieter von Diensten wie Testentwicklung, Zertifizierungsmanagement oder Item Banking. Die Migration von einem System auf ein anderes, wäre teilweise unmöglich oder nur unter hohen Kostenaufwand durchführbar gewesen.
Die Lösung für dieses Dilemma war ein offener Standard. Ein solcher Standard ist eine öffentliche Spezifikation um spezifische Ziele zu erreichen. Standards wie WWW, HTML oder XML gehören dazu. Aus der Makroebene gesehen, kann so eine Spezifikation, die Entwicklung der damit verbundenen Industrie um einiges vorantreiben. Das führt somit zu besseren Produkten, da die einhergehende Rivalität nur mit erhöhter Qualität oder niederen Preisen gewonnen werden kann. Der Hauptvorteil liegt jedoch darin, dass dieser offene Standard die Kommunikation und Interoperabilität zwischen verschiedenen Entwicklern, Systemen und Diensten fördert. [IMS Global Learning Consortium, 2006d, S. 1f]
Zusammengefasst können folgende Vorteile aus dieser Entwicklung gezogen werden:

- verbesserte Interoperabilität
- erhöht die Anzahl alternativer Lösungen
- schnellere Innovationen
- reduzierte Kosten

Die Entwicklung dieses Standards startete mit der Initial Spezifikation V0.5 im März 1999. Darauf folgend wurde die finale Version 1.0 im Mai 2000 verabschiedet. Die Spezifikation wurde zweimal erweitert und erneuert und somit im März 2001 mit der Version 1.1 und im Jänner 2002 (V1.2) herausgegeben.
Nachdem eine Reihe von Problemen durch Implementoren aufgezeigt wurden, überarbeitete das Projekt-Team den Standard und veröffentlichte die Erweiterung in Form eines Addendum als Version 1.2.1. Viele Probleme konnten mit dieser neuen Version aber nicht gelöst werden, da es signifikante Änderungen in der Spezifikation mit sich gezogen hätte. Dies hätte ebenso die Rückwärtskompatibilität verhindert.
IMS fertigte nicht nur QTI als Spezifikation an, sondern veröffentlichte auch Standards wie Content Packaging, Simple Sequencing und Learning Design. Um auch hier Interoperabilität zu gewährleisten, wurde im Jahr 2003 eine Überarbeitung quer über alle Spezifikationen durchgeführt. Durch diese Harmonisierung wurden einige Probleme mit QTI festgestellt. Im selben Jahr noch wurde ein Projekt aufgestellt, das beide Probleme, die Harmonisierung und die aufgezeigten Schwierigkeit der Implementoren aus V1.x, aufarbeiten sollte. Um das Vorhaben handhabbar zu machen, wurden einige Restriktionen festgelegt. Somit wurde beschlossen, dass sich das neue Release V2.0 ausschließlich mit individuellen Assessment Items auseinander setzt. Im Jänner 2005 wurde dies auch umgesetzt und V2.0 veröffentlicht. [IMS Global Learning Consortium, 2006c]
Nach [IMS Global Learning Consortium, 2009b] fokusiert QTI v2.0 auf das Item und verfolgt dabei drei Hauptziele:

1. Die Adressierung der Probleme aus v1.0: ein neues Informationsmodell mit einem neuen Interaktionsmodell und Unterstützung von XHTML wurde definiert. Die Platzierung und die Steuerung des

Verhaltens von Feedback werden somit erleichtert. Ebenso wird das Klonen durch Item Templates hinzugefügt.

2. Definition einer Methode, um QTI in Content Packages zu verpacken.

3. Entwicklung einer Methode, um QTI mit Learning Design, Simple Sequencing und CMI Datenmodelle zu verwenden.

Die Version 2.1 soll den Vorgänger komplettieren und die Teile der Spezifikation, die sich mit der Aggregation von Items in Sections und Tests beschäftigten, vervollständigen. Bis heute ist jedoch nur ein „public draft“ der Version 2.1 erhältlich. [IMS Global Learning Consortium, 2006c]

3.3 IMS Global Learning Consortium

Instructional Management Systems (IMS) Global Learning Consortium (GLC) ist eine globale non-profit Organisation, die Standards für die Entwicklung und Adaptierung von Lernsystemen erzeugt. Die Organisation ist dabei bemüht, weltweit das Technologie Wachstum im Erziehungs- und Lernsektor positiv zu beeinflussen.
IMS/GLC entstand im Jahr 1997 als Projekt innerhalb der US National Learning Infrastructure Initative von EDUCAUSE. Der damalige Fokus, der sich auf die Hochschulbildung legte, wurde erweitert und umfasst heute verschiedene Lernkontexte, wie Unternehmenstrainings, staatliche Trainings und K-12 Schulen.
IMS GLC wird von über 135 weiteren Organisationen unterstützt, die sich wie folgt zusammensetzen: 58% führende Unternehmen aus dem Lerntechnologiebereich, 24% führende Institutionen im Bereich Erziehung und 18% aus staatlichen Organisationen.
Das Betätigungsfeld von IMS setzt sich von on-/offline Lernen wie auch a-/synchrones Lernen zusammen und kann allgemein als „verteiltes Lernen“ beschrieben werden [IMS Global Learning Consortium, 2009a]. Im Laufe der Zeit veröffentlichte IMS über 20 Spezifikationen, darunter sind nach [Winkelmann, 2005, S. 18f] weit verbreitete Standards wie:

1. **IMS Meta-data** - Spezifikation zur Beschreibung von Lernmaterialien. Diese basieren auf dem LOM-Standard des IEEE LTSC.

2. **IMS Digital Repositories** - Spezifikation zur Erstellung von digitaler Bibliotheken.

3. **IMS Content Packaging** - Spezifikation zur Erstellung, Austausch und Wiederverwendung von Lernobjekten.

4. **IMS Simple Sequencing** - Spezifikation um Lernpfade durch verschiedene Lernaktivitäten zu beschreiben.

5. **IMS Learning Design** - Richtlinien zur Entwicklung von Lernszenarien.

Die IMS Spezifikationen können nach [Constandache et al., 2003, S. 2] in drei Hauptgruppen eingeteilt werden:

- Spezifikationen, die benutzt werden um Inhalte zu beschreiben, festzustellen und auszutauschen
- Spezifikationen für die Interaktion mit Inhalten und deren Verfolgung
- Spezifikationen für die Interoperabilität von Systemen

3.4 Überblick der Anwendungsfälle der Spezifikation

Die IMS Question and Test Interoperability Spezifikation bildet die Grundlage zum Austausch von Items, Assessments und deren Ergebnissen. So ermöglicht QTI die Interoperabilität einer Vielzahl von Systemen, die in der nachfolgenden Abbildung 3.1 zusammen mit ihren Akteuren dargestellt werden. (Vgl. [IMS Global Learning Consortium, 2006c])

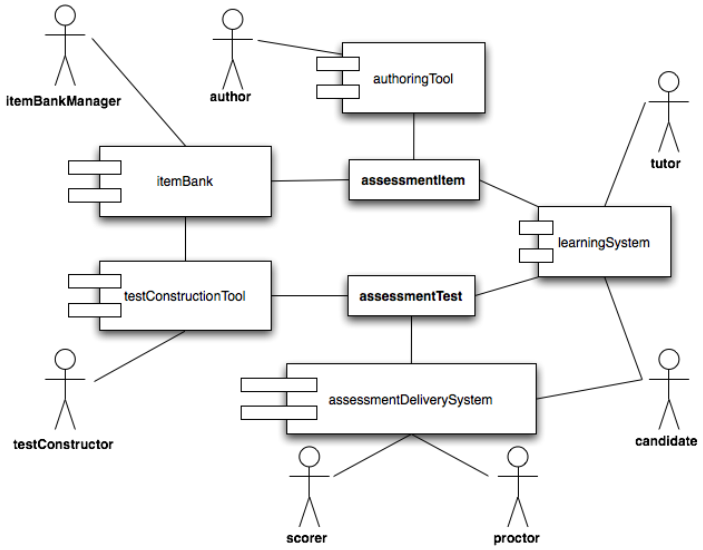


Abbildung 3.1: Überblick der Rollen und Systeme, die durch QTI berücksichtigt werden. [IMS Global Learning Consortium, 2006c]

Die Abbildung zeigt neben den Systemen und deren Akteuren auch indirekt die Ziele von QTI. Diese sind:

1. Ein klar definiertes Datenmodell für die Speicherung und dem Austausch von Items und Tests zur Verfügung zu stellen. Dieses Datenformat sollte unabhängig von Autorenwerkzeugen sein.
2. Die Unterstützung der Verteilung von Items aus einer Item Bank an Lernsysteme.
3. Den Austausch und die Speicherung von Testresultaten durch einen einheitlichen Standard zu gewährleisten.

3.4.1 Beteiligte Systeme

Die in Abbildung 3.1 aufgezeigten Systeme werden in der folgenden Tabelle 3.1 näher beschrieben.

3.4.2 Beteiligte Akteure

Die beteiligten Akteure wurden aus Komplexitätsgründen generalisiert und vereinfacht dargestellt. Die nachfolgende Tabelle 3.2 liefert eine kurze Beschreibung der Rollen.

3.5 QTIs Information Model

Dieser Abschnitt behandelt das von QTI spezifizierte Informationsmodell (Vgl. [IMS Global Learning Consortium, 2006a] und [JISC CETIS, 2008]). Es bildet den Kern des QTI Standards und dessen Datenformat wird durch den Industrie Standard eXtensible Markup Language (XML) beschrieben.

System	Beschreibung
authoringTool	In diesem System werden Items von Autoren erzeugt und modifiziert.
itemBank	Eine Item Bank (2.3.1.3) stellt ein Repository für die Verwaltung von Assessment Items dar.
testConstructionTool	Mit diesem System können Tests, durch die Zusammensetzung von Items, assembliert werden.
assessmentDeliverySystem	Die Aufgaben dieses Systems sind die Auslieferung von Tests an Testpersonen und die Erstellung der Test Bewertung (automatisch oder durch Bewerter). Ein solches System kann auch *Player* genannt werden.
learningSystem	In einem *learningSystem* können Lernende Lernaktivitäten durchführen, die teilweise von Tutoren unterstützt werden. Aktivitäten können hier zum Beispiel die Durchführung von Assessments sein.

Tabelle 3.1: Beschreibung der Systeme, die rund um das Thema „Assessment“ von QTI berücksichtigt werden.

Rolle	Beschreibung
author	Ein Autor erstellt mit Hilfe eines *authoringTool* die Assessment Items. Je nach Komplexität des Items und dessen Kontrollprozesses kann dieser Akteur auch stellvertretend für mehrere Autoren sein. Im Gegensatz zu einem Item Banker fokussiert dieser hauptsächlich auf den Inhalt eines Items.
itemBankManager	Ein *itemBankManager* ist für die Verwaltung von Items zuständig und arbeitet dabei mit einer itemBank.
testConstructor	Ein *testConstructor* fügt individuelle Items zu einem Test zusammen. Die Items werden dabei aus einer *itemBank* entnommen.
proctor	Der Aufgabenbereich eines Proktors ist die Prüfungsaufsicht. Die Rolle hat aber nichts mit der Bewertung von Prüfungen zu tun.
scorer	Die Bewertung einer Prüfung wird von dieser externen Person/-System vorgenommen. Viele Items können auch automatisch oder durch eine externe Instanz bewertet werden.
tutor	Ein Tutor unterstützt und dirigiert Lernende durch einen Lernprozess.
candidate	Eine Testperson wird als Kandidat bezeichnet.

Tabelle 3.2: Beschreibung der identifizierten Akteure durch die Spezifikation QTI.

Zunächst wird das Assessment Item, welches die kleinste Einheit in der Spezifikation bildet, und dessen Struktur und dessen Lebenszyklus behandelt. Darauf folgend werden verschiedene Item Variablen aufgezählt und deren Verwendung näher erläutert. Weiters wird das Content Model und die Verwendung von XHTML besprochen. Nachdem werden unterschiedliche Interaktionen, die Verarbeitung der Antworten und Feedback im Bereich Assessment ausgearbeitet. Um auf die Thematik der Assessment Parametrisierung näher einzugehen, wird der Punkt Item Template dargestellt. Den Abschluss bildet die Spezifikation von Assessment Test, wobei hier auch näher auf die Struktur und die Aggregation von Inhalten eingegangen wird.

3.5.1 Item Klasse

Das Item (*assessmentItem*) stellt das kleinste Objekt in einem Assessment dar. Es repräsentiert nicht nur die Testfrage, sondern beinhaltet auch die Verarbeitung der Antworten und das Feedback, welches sich aus Hilfestellungen und Lösungen zusammensetzt.
Die nachfolgende Auflistung zeigt die Elemente eines Items. Diese können je nach Restriktion einmal oder auch mehrfach in einem Assessment Item vorkommen. Die gezeigten Elemente werden in den nachfolgenden Abschnitten näher beschrieben:

- Attribute
- responseDeclaration [*]
- outcomeDeclaration [*]
- templateDeclaration [*]
- templateProcessing [0..1]
- stylesheet [0..*]
- itemBody [0..1]
- responseProcessing [0..1]
- modalFeedback [*]

3.5.1.1 Attribute eines Items

Ein Item besteht aus einer gewissen Anzahl von Attributen, die jedoch nichts mit der Frage oder den Antwortmöglichkeiten zu tun haben, sondern nur für die QTI Umgebung von Bedeutung sind. Zu diesen Attributen zählen unter anderem die Kennung, der Titel, die Definition der Adaptiviät und die Definition der Zeitabhängigkeit des Items. Ein mögliches Beispiel könnte wie folgt aussehen:

```
1 <assessmentItem xmlns="http://www.imsglobal.org/xsd/imsqti_v2p1"
2                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3                 xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1 imsqti_v2p1.xsd"
4 identifier="choice"
5 title="Unattended Luggage"
6 adaptive="false"
7 timeDependent="false"
8 >
```

Listing 3.1: Die Verwendung von Attributen in einer Multiple Choice Aufgabe. [JISC CETIS, 2008]

3.5.1.2 Item Session Lifecycle

Alle Versuche die eine Testperson an einer Item Instanz vornimmt um sie zu lösen werden in einer Item Session gesammelt. Es gibt zum Beispiel Test Typen wie *„Drill and Practice“*, bei denen eine Item Instanz mehrmals von einem Kandidaten gelöst werden muss. Jede Instanz eines Items besitzt seine eigene Session.
Das Ablaufdiagramm 3.2 stellt den Ablauf und die verschieden Stati der Item Session, in anderen Worten, den Lebenszyklus dar. Dieser Zyklus ist nicht bindend, sondern soll, je nach Lernkontext, angepasst werden.
Beispielsweise könnte das modale Feedback in einem speziellen Kontext nicht erwünscht, oder auch die Durchsicht der bereits absolvierten Fragen nicht gestattet sein.

Der Ablauf der Session beginnt, sobald das Item dem Kandidaten zugeordnet wird. Dieser Status wird als *initial* bezeichnet. Dieser Status endet, sobald das Item dem Kandidaten zur Verfügung gestellt wird. Ab diesem Zeitpunkt wird der Session Status solange verwaltet und aktualisiert, bis die Session wieder beendet wird. Zu jeder Zeit kann dabei die Session in ein *itemResult* Objekt übergeführt werden. Dieses beinhaltet unter anderem ein detailliertes Tracking aller Lösungsversuche.
Bei nicht adaptiven Tests wird bereits in der Anfangsphase festgelegt, welche Items dem Kandidaten zugeordnet werden. Die Interaktionen mit diesen Items werden zum Schluss einer Test Session gemeldet, egal ob alle Items versucht wurden. Aus diesem Grund werden die Item Sessions zur gleichen Zeit am Anfang erzeugt, in den *initial* Status versetzt und parallel verwaltet. Im Gegensatz dazu steht bei adaptiven Tests der Pfad durch die Items noch nicht fest, sondern wird je nach Antwort der Testperson gewählt. Hier wird jede Item Session erst erzeugt, wenn das Item durch ein *Delivery System* gewählt wurde. Die Interaktion zwischen einer Testperson und einem Item wird in Versuche unterteilt. Sobald ein Kandidat einen Versuch startet, wird eine *Candidate Session* angelegt. Verlässt aus irgendeinem Grund der Kandidat die Frage, ohne sie zu beantworten, und navigiert zu einer anderen Frage, wird die *Candidate Session* terminiert, aber der Versuch bleibt noch aufrecht. Der Versuch wird bei einem solchen Fragenwechsel lediglich auf den Status *suspended* gesetzt und kann wieder erweckt werden. Während einer *Candidate Session* bleibt die Item Session im Status *interacting*. Nachdem ein Versuch endet, wird die Verarbeitung der Antwort durchgeführt. Hierauf kann ein erneuter Versuch gestartet werden.
Wenn keine Versuche mehr durchgeführt werden dürfen, geht der Status in *closed* über. Nachdem alle Response Variablen gesetzt wurden, können, je nach *Delivery System*, die beantworteten Fragen noch mal durchgesehen werden. In diesem *review* Status kann ebenso summatives Feedback dargestellt werden.
Zum Schluss kann die Session, sofern das durch das *Delivery System* ermöglicht wird, in einen *solution* Status übergeführt werden. In diesem Zustand werden die Items mit ihren richtigen Lösungen, die in der *responseDeclaration* definiert wurden, dargestellt.

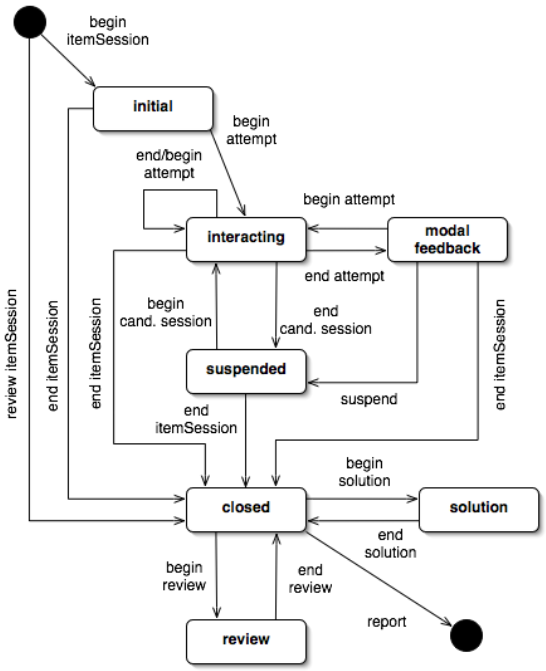


Abbildung 3.2: Darstellung des Lebenszyklusses einer Item Session in Form eines Ablaufdiagramms. [IMS Global Learning Consortium, 2006a]

3.5.2 Item Variablen

Die Item Variablen ähneln den lokalen Variablen in Programmiersprachen. Sie werden intern in einer Frage deklariert und dienen dabei zur Kommunikation. Item Variablen spielen eine wesentliche Rolle für verschiedene Funktionalitäten wie die Verarbeitung der Antwort oder das Item Klonen. Sie werden dazu genutzt um temporäre Informationen zur Laufzeit eines Items zu halten.

Es existieren drei verschiedene Typen von Item Variablen:

Response Variablen beinhalten die Antworten der Testperson. Sie werden durch die Klasse *responseDeclaration* angelegt und bekommen ihre Werte erst bei der Interaktion mit dem Item Body zugewiesen. Diese Variablen werden später bei der Verarbeitung der Antwort verwendet.

Outcome Variablen umfassen die erreichten Punktezahlen einer Testperson. Sie werden durch die Klasse *outcomeDeclaration* angelegt und bekommen während des Antwortverarbeitungsprozesses (siehe Abschnitt 3.5.5) einen Wert zugewiesen. Die Verwendung dieser Variablen findet im anschließenden Feedback statt.

Template Variablen werden verwendet, um Klone von Items zu erzeugen. Deklariert wird diese Variable durch die Klasse *templateDeclaration*. Die Initialisierung findet in der Verarbeitungsphase des Templates statt. Diese Variablen werden an zwei Stellen benötigt, zum einen innerhalb des Item Body und zum anderen während der Verarbeitung der Antwort.

Wie bereits oben angeführt werden diese Variablen mit Hilfe der Klasse *variableDeclaration* und deren Derivate angelegt. Das Klassendiagramm 3.3 gibt einen Überblick über die verschiedenen Deklarationsklassen und deren Attribute.

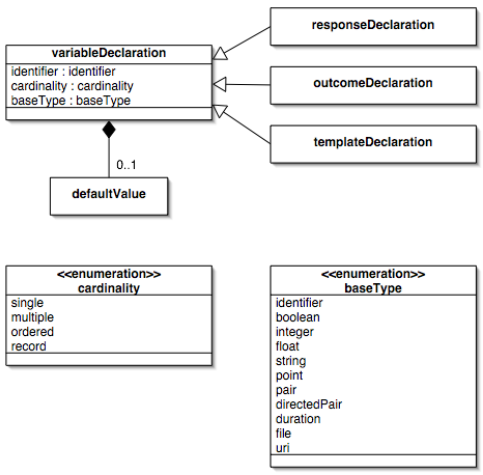


Abbildung 3.3: Überblick der verschiedenen Deklarationsklassen und deren Attribute in Form eines Klassendiagramms. [IMS Global Learning Consortium, 2006a]

Wichtige Attribute der Deklaration sind unter anderem der *identifier*, welcher eine genaue Bezeichnung der Variable festlegt. Innerhalb eines Items verwenden alle Variablen den gleichen Namensraum, das heißt jeder gewählte Variablen Name muss eindeutig sein. Weiters wird noch der Typ der Variable, durch *baseType*, definiert. Die verschiedenen Wertetypen können aus dem Klassendiagramm 3.3 entnommen werden.

3.5.3 Item Templates

Item Templates werden verwendet um eine große Anzahl ähnlicher Items zu erzeugen. Durch eine *Delivery Engine* wird ein so genannter Item Klon dynamisch erzeugt und dem Kandidaten zur Verfügung gestellt.
Die geklonten Items unterscheiden sich nur durch die variierenden Template Variablen, die im *Template Processing* gesetzt werden und die unterschiedlichen *Identifier*. Die Zuweisung der jeweiligen Werte für die Template Variablen erfolgt nach Regeln, die im *Template Processing* definiert wurden.
Eine Template Regel ist entweder eine *templateCondition*, die einem konditionalen Ausdruck ähnelt, oder einer einfachen Aktion. Diese Regeln bilden eine *light-weight* Programmiersprache, um solche Klone zu erzeugen, und stellen deshalb nur eine minimale Anzahl an Kontrollstrukturen zur Verfügung.
Die Template Variablen ermöglichen somit die Erstellung von parametrisierten Fragen, die sich je nach Variablenwerten unterscheiden. Zudem wird im *Template Processing* auch die richtige Antwort auf die Frage entsprechend verändert. Schließlich muss sich diese mit der neuerstellten parametrisierten Frage decken.
Nachdem den Variablen Werte zugewiesen wurden, können diese Werte im Item Body durch die Klasse *printedVariable* verwendet und dargestellt werden.

3.5.4 Content Model

Der Item Body bildet den Container für Texte, Grafiken, Multimedia Objekte, Interaktionen und die Informationen über die Strukturierung des Item Inhaltes. Der Inhalt unterliegt dabei den Formatierungen des Standards Cascading Style Sheets (CSS), die explizit oder implizit durch die Verwendung standardmäßiger Stile von *Delivery Engines* definiert werden.
Der Body enthält somit erstens alle Informationen, die für den Kandidaten sichtbar sind:

- Frage
- Antwortmöglichkeiten
- Feedback
- Hilfestellungen
- Lösungen

Zweitens enthält dieser die Interaktionsdefinitionen zwischen der Testperson und dem Item selbst.

3.5.4.1 Präsentation des Item

Der Inhalt eines Items wird sichtbar, sobald der Item Session Status auf *interacting* wechselt. Ab diesem Zeitpunkt kann die Testperson Interaktionen mit dem Inhalt durchführen, welche direkte Auswirkungen auf die Response Variablen haben. Im Session Status *closed* oder *review* wird der Body ebenso angezeigt, jedoch nur mit der abgegebenen Antwort, ohne, dass eine weitere Modifikation möglich ist. Im Gegensatz dazu wird im *solution* Status nicht nur die Interaktion deaktiviert, sondern auch die abgegebene Antwort durch die vordefinierte Lösung ausgetauscht.

3.5.4.2 Innere Struktur

Das Item Body Element besteht aus einem oder mehreren Blöcken. Die Abbildung 3.4 zeigt die innere Struktur eines Item Body Elementes mitsamt den vier verschiedenen Blocktypen.

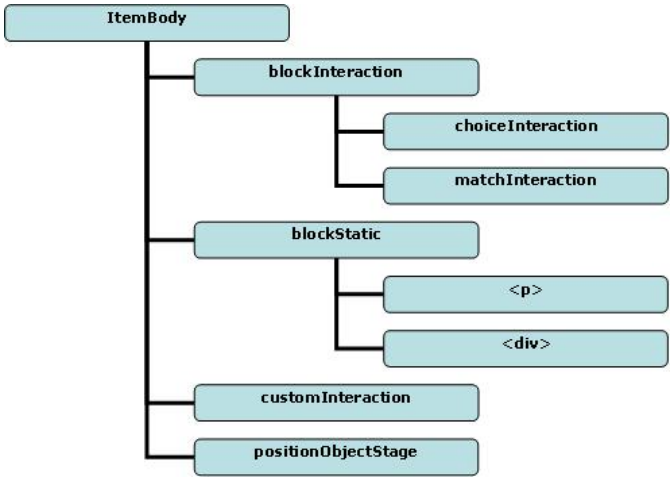


Abbildung 3.4: Die innere Struktur eines Item Body Elementes mitsamt den vier verschiedenen Blocktypen. [JISC CETIS, 2008]

blockInteraction ist der am häufigsten verwendete Blocktyp. Dieser Block implementiert ähnliche Interaktionstypen wie Multiple Choice und Zuordnungsinteraktionen.

blockStatic spezifiziert nur statische visuelle Komponenten wie Text oder Tabellen und beinhaltet keine Interaktionen.

customInteraction ermöglicht die Erweiterung der QTI Spezifikation mit eigenen Interaktionen.

positionObjectStage unterstützt Bilder auf denen der Kandidat weitere Objekte platzieren kann.

Ein kurzes Beispiel zeigt folgendes Code Snippet 3.2. In diesem Exempel wird der Kandidat angewiesen einen bestimmten Ort auf einer Karte zu finden. Hierzu werden vier Hotspot Möglichkeiten auf einem Bild definiert, welche jeweils von der Testperson ausgewählt werden können. Es werden dabei zwei Blocktypen verwendet: *blockStatic (p)* und *blockInteraction (hotspotInteraction)*

```
<itemBody>
<p>
The picture illustrates four of the most popular destinations for air travellers arriving
            in the United Kingdom: London, Manchester, Edinburgh and Glasgow.
</p>
<hotspotInteraction responseIdentifier="RESPONSE" maxChoices="1">
<prompt>Which one is Glasgow?</prompt>
<object type="image/png" width="206" height="280" data="images/ukair.png">UK Map</object>
<hotspotChoice shape="circle" coords="77,115,8" identifier="A"/>
<hotspotChoice shape="circle" coords="118,184,8" identifier="B"/>
<hotspotChoice shape="circle" coords="150,235,8" identifier="C"/>
<hotspotChoice shape="circle" coords="96,114,8" identifier="D"/>
</hotspotInteraction>
</itemBody>
```

Listing 3.2: Ein Hotspot Beispiel - Verwendung der verschiedenen Blocktypen.

3.5.4.3 Interaktionen

Die Interaktionen bilden einen Dialog zwischen Testperson und Item. Die Testperson kann durch eine Interaktion eine Antwort konstruieren und das Item gibt anschließend ein Feedback auf diese Response zurück.
Die abgegebenen Antworten werden in den bereits erwähnten Response Variablen gespeichert. Hierzu erzeugt eine Interaktion mindestens eine Response Variable.

In QTI existieren eine Vielzahl an Interaktionsmöglichkeiten, die in fünf Kategorien eingeteilt werden. Diese Kategorien setzen sich aus den simplen Interaktionen, textbasierten Interaktionen, grafischen Interaktionen, diversen Interaktionen und einer speziellen Interaktion zur Beendigung eines Versuches zusammen.

3.5.4.4 XHTML und Erweiterungen

Das Content Model von QTI baut sehr stark auf dem Konzept des Standards XHTML[1] auf. Jedoch ist nur eine Untermenge von XHTML in einem Item erlaubt. Zusätzlich werden auch eigene Elemente definiert, die es ermöglichen Interaktionen zu repräsentieren und integriertes Feedback wiederzugeben.

3.5.5 Response Processing - Verarbeitung der Antwort

Dieser Prozess besteht aus einer Sequenz von Verarbeitungsregeln, welche die Antwort der Testperson auswerten. Die *Delivery Engine* weist, je nach gesetzten Response Variablen aus dem Interaktionsprozess, das Ergebnis den Outcome Variablen zu. Diese werden später für den Bericht oder das Feedback verwendet. Ein solches Feedback kann entweder sofort nach einem Versuch eintreten oder auch zusammenfassend am Ende eines Tests.
Das folgende Ablaufdiagramm 3.5 zeigt die zeitliche Abfolge der verschiedenen Schritte eines Items:

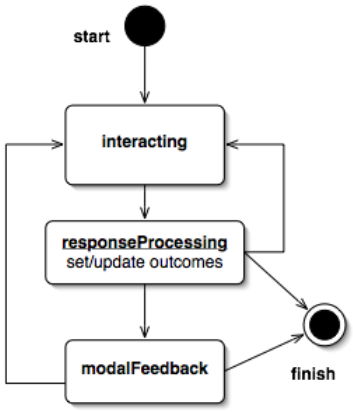


Abbildung 3.5: Zeitliche Abfolge der verschiedenen Schritte eines Items. [IMS Global Learning Consortium, 2006a]

Für diesen Verarbeitungsprozess stehen ebenso die gleichen Instruktionen und Kontrollstrukturen wie bei der Template Verarbeitung zur Verfügung. Diese Bewertungsinstruktionen bilden eine Hochsprache, die nicht an spezielle Score Engines gebunden ist.
Um das *Response Processing* besser darzustellen, wird hier ein kurzes Beispiel aus der Implementations Anleitung von QTI geben. In diesem Exempel sollen die Namen entsprechend derer Podiumsplätze in einem Grandprix gereiht werden.
Als erstes werden die Response und die Outcome Variable betrachtet. Die Response Variable beinhaltet bereits die korrekte Antwortreihung in Form einer Sequenz. Für das Ergebnis wird eine Variable mit dem Typ Integer verwendet:

```
<responseDeclaration identifier="RESPONSE" cardinality="ordered" baseType="identifier">
  <correctResponse>
    <value>DriverC</value>
    <value>DriverA</value>
    <value>DriverB</value>
  </correctResponse>
</responseDeclaration>
<outcomeDeclaration identifier="SCORE" cardinality="single" baseType="integer" />
```

[1] http://www.w3.org/TR/xhtml1/

Listing 3.3: Anwendung der Response Deklaration.

Der Item Body beinhaltet einen Block mit einer Reihungsinteraktion. In dieser Interaktion wird der bereits deklarierten Response Variable ein Wert zugewiesen.

```
<itemBody>
  <orderInteraction responseIdentifier="RESPONSE" shuffle="true">
    <prompt>The following F1 drivers finished on the podium in the first ever Grand Prix of Bahrain. Can you rearrange them
        into the correct finishing order?</prompt>
    <simpleChoice identifier="DriverA">Rubens Barrichello</simpleChoice>
    <simpleChoice identifier="DriverB">Jenson Button</simpleChoice>
    <simpleChoice identifier="DriverC" fixed="true">Michael Schumacher</simpleChoice>
  </orderInteraction>
</itemBody>
```

Listing 3.4: Erstellung des Item Bodys und der beinhalteten Reihungsinteraktion.

Nachdem die Interaktion durch den Kandidaten erfolgte, wird das Response Processing gestartet. Dieses Element beinhaltet eine Sequenz von konditionalen Ausdrücken. Wenn die Response Variable der korrekten Reihenfolge entspricht, wird die Outcome Variable auf 2 gesetzt. Trifft die Reihenfolge (C, B, A) zu, wird nur ein Punkt vergeben und falls keine Antwort richtig gegeben wurde, wird eine Bewertung von 0 Punkten vergeben.

```
<responseProcessing>
  <responseCondition>
    <responseIf>
      <match>
        <variable identifier="RESPONSE" />
    <correct identifier="RESPONSE" />
      </match>
    <setOutcomeValue identifier="SCORE">
      <baseValue baseType="integer">2</baseValue>
    </setOutcomeValue>
  </responseIf>
  <responseElseIf>
    <match>
      <variable identifier="RESPONSE" />
      <ordered>
        <baseValue baseType="identifier">DriverC</baseValue>
    <baseValue baseType="identifier">DriverB</baseValue>
    <baseValue baseType="identifier">DriverA</baseValue>
      </ordered>
    </match>
    <setOutcomeValue identifier="SCORE">
      <baseValue baseType="integer">1</baseValue>
    </setOutcomeValue>
  </responseElseIf>
  <responseElse>
    <setOutcomeValue identifier="SCORE">
      <baseValue baseType="integer">0</baseValue>
    </setOutcomeValue>
  </responseElse>
 </responseCondition>
</responseProcessing>
```

Listing 3.5: Verarbeitungsprozess der Antwort.

Die Regeln in einem Response Prozess können je nach Größe an Komplexität zunehmen. Itemautoren besitzen oft nicht das technische Verständnis, um diese Komplexität zu beherrschen. Aus diesem Grund kann dieser Umstand zu einer Barriere für QTI werden.
Um das Problem zu mindern, bietet QTI neben diesen integrierten Ausdrücken auch die Möglichkeit, Template Prozesse zu erstellen. In diesen Templates können Prozessregeln definiert werden, die von verschiedenen Items verwendet werden. Die Verarbeitungslogik wird somit vom Item selbst getrennt. Ein Itemautor muss deshalb nur deren Existenz kennen und nicht mehr die beinhaltete Logik.

3.5.6 Test Klasse

Die Test Klasse (*assessmentTest*) wurde mit der Spezifikation QTI 2.1 eingeführt. Diese Klasse stellt eine Gruppierung von Assessment Items dar. Ein Assessment beinhaltet Regeln, die darüber entscheiden, welche Reihenfolge der Items gewählt wird. Neben dieser Pfaddefinition durch ein Assessment wird auch der Zeitpunkt bestimmt, wann eine Antwort eingereicht oder ein Feedback angezeigt wird. Ein Test stellt somit eine logische Aggregation von einzelnen Items dar und beinhaltet zusätzliche Informationen über folgende Punkte:

- Definition der Gruppierung von Items
- Bestimmung der Reihenfolge von Items, durch eine bestimmte Sequenz oder durch Randomisierung
- Zeitpunkt, wann die Submission eines Items erfolgt
- Zeitpunkt, wann Feedback gegeben wird

Die nachfolgende Auflistung zeigt die Elemente eines Tests. Diese können je nach Restriktion einmal oder auch mehrfach in einem Assessment vorkommen:

- Attribute
- outcomeDeclaration [*]
- timeLimits [0..1]
- testPart [1..*]
- outcomeProcessing [0..1]
- testFeedback [*]

3.5.6.1 Navigation und Submission

Die Bestimmung der Navigation und der Submission sind Kernaufgaben eines Tests. Um diese Definition durchzuführen müssen beide Modi - *navigationMode* und *submissionMode* festgelegt werden.
QTI unterstützt für die Navigation zwei verschiedene Modi - linear und nicht-linear. Bei einer linearen Navigation wird ein Item nach dem anderen abgehandelt. Eine Umkehr oder eine nochmalige Bearbeitung ist dabei nicht möglich. Die nicht-lineare Navigation hingegen ermöglicht eine freie Pfadwahl und beinhaltet keine Restriktionen. Ein Item kann in diesem Modus zu jeder Zeit angesteuert werden.
Die Submission besitzt ebenso zwei Modi und definiert, wann die Antwort für die Response Verarbeitung freigegeben wird. Die beiden Modi setzen sich aus einer individuellen oder einer simultanen Submission zusammen. Bei dem individuellen Modus wird nach jeder Item-Bearbeitung eine Antwort abgegeben. In diesem Fall erfolgt das *Response Processing* während eines Testdurchlaufes nach jeder Item Submission. Dieser Vorgang ermöglicht, dass Feedback bereits während eines Tests entnommen werden kann. Im Gegensatz dazu definiert der simultane Modus eine gleichzeitige Abgabe aller Items zum Schluss eines Testteils. Kandidaten können daher erst am Ende dieses Testteils Feedback empfangen. Diese Form der Testdurchführung ähnelt einer Prüfung mit Papier und Bleistift.

3.5.6.2 Test Struktur

Die QTI Spezifikation unterstützt für Tests eine spezielle Strukturierung in einen oder mehreren Test- (*testPart*) und Sektionsteilen (*sectionPart*). Die Struktur des Sektionsteiles baut auf der Komposite Architektur auf. Der Sektionsteil kann somit weitere Sektionen und/oder Items enthalten.
Durch diese Unterteilung des Assessments kann ein Itemautor die lineare Abfolge von Items disaggregieren und Funktionalität in den Testablauf verpacken. Die folgende Abbildung 3.6 zeigt den generellen Aufbau einer Assessment Struktur in Form einer beispielhaften Konfiguration einer Prüfung:

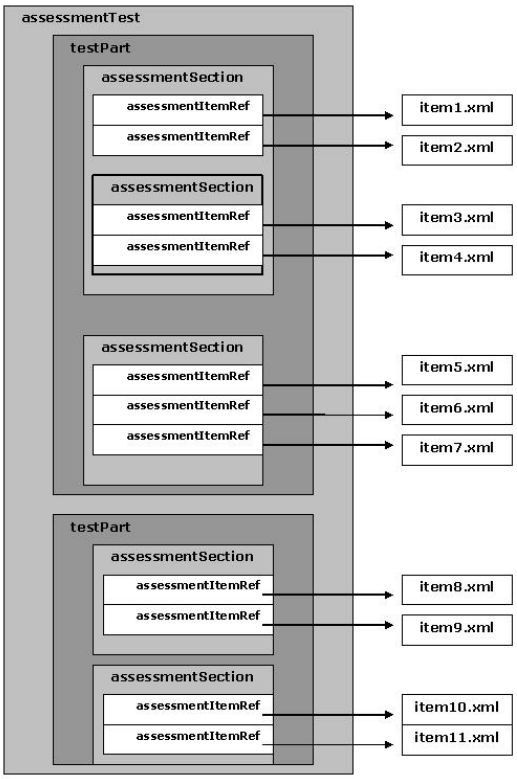


Abbildung 3.6: Der generelle Aufbau einer Assessment Struktur. [JISC CETIS, 2008]

Jede interne Gruppierung besitzt eine eigene Funktionalität. Diese Logik wird durch definierte Attribute im jeweiligen Teil gesteuert. Die Pfade, die eine Testperson durch eine Prüfung nimmt, werden somit durch diese Regeln gelenkt.
Der Testteil beinhaltet Attribute, die die Navigation und den Submission Modus beeinflussen. Beide wurden bereits in 3.5.6.1 diskutiert. Zusätzlich werden auch Vorbedingungen *(preconditions)* definiert, die je nach Konfiguration entscheiden, ob ein Testteil einem Kandidaten angezeigt wird. Nachdem ein Kandidat einen Testteil beendet, legt das optionale Attribut *branchRule* fest, welcher Testteil nachfolgend an die Reihe kommt. Mit dem Element *itemSessionControl* können verschiedene Funktionalitäten bestimmt werden. Eine Möglichkeit besteht darin, dass die maximalen Versuche der beinhalteten Items festgelegt werden oder definiert wird, ob diese Items teilweise bei der Testdurchführung übersprungen werden können. Zeitlimits der Testdurchführung können durch das Attribut *timeLimits* konfiguriert werden. Um Test Level Feedback zu erzeugen, existiert das Element *testFeedback*.
Der Sektionsteil beinhaltet die ähnlichen Attribute wie der Testteil mit Ausnahme des Elementes *fixed*. Durch dieses Element wird angegeben, ob die Reihenfolge der beinhalteten Items einer Sektion vor deren Anzeige gemischt wird. Die vom Sektionsteil abgeleiteten Klassen, Sektion und Item Referenz, erweitern die Elternklasse mit verschiedenen Attributen. Diese zusätzlichen Attribute und deren Klassen können aus der Abbildung 3.7 entnommen werden.

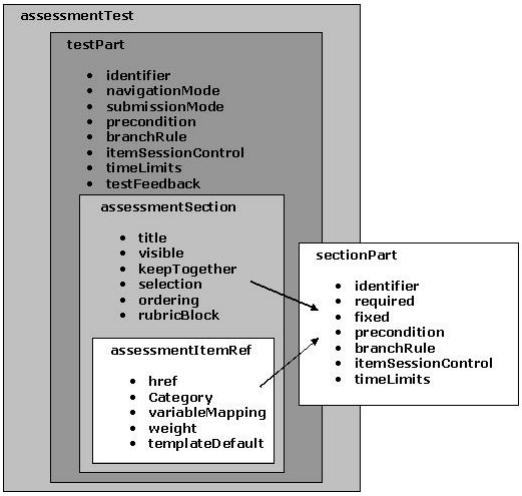


Abbildung 3.7: Übersicht der Testelemente, mitsamt deren Attribute. [JISC CETIS, 2008]

3.5.7 Time Limits

Um Zeitvorgaben für Prüfungen zu geben, definiert QTI Time Limits. Diese können je nach Gebrauch auf tiefere Ebenen herunter gebrochen werden. Somit gibt es nicht nur ein Zeitlimit für den gesamten Testteil, sondern auch für dessen Sektionen und Items.
Bei einer Zeitangabe kann auf zwei Typen zurückgegriffen werden, Minimal- und Maximalwert. Der Minimalwert beschreibt die kürzeste und der Maximalwert gibt die längste mögliche Zeit in Sekunden an.
Bei einer Zeitangabe im Testteil bezieht sich diese auf die gesamt Zeit, welche sich aus den einzelnen Item Sessions und der Navigationszeit zusammensetzt.
Die Aufgabe des Zeit Trackings und deren Auswertung obliegt der *Delivery Engine* und wird mit einer Response Zusatzvariablen im Testbereich mit dem Namen *duration* durchgeführt. Die Verarbeitung dieser Variablen wird im *Outcome Processing* durchgeführt. Dieser Verarbeitungsschritt bildet den nächsten Abschnitt der Arbeit.

3.5.8 Outcome Processing

Bei der Verarbeitung der Ergebnisse werden den zuvor deklarierten Outcome Variablen Werte zugewiesen. Diese Variablen werden als Attribute eines Assessment Tests angelegt und sind von den Outcome Variablen eines Items zu unterscheiden. Das Ziel dieser Variablen ist es, die Leistung eines Kandidaten in Werte zu fassen. Dabei stützen sich diese Variablen auf den Daten der Outcome Variablen aus den einzelnen darunter liegenden Items. So werden zum Beispiel bei einer Prüfung mit zehn Items alle zehn Ergebnis Variablen summiert und in eine Ergebnis Variable des Tests gespeichert. Dieser Vorgang nennt sich *Outcome Processing*.
Die Verarbeitung dieser Ergebnisse bildet einen Teilschritt in einer größeren Serie. Zunächst beginnt diese Abfolge mit der Submission einer Antwort durch eine Testperson. Darauf folgend wird das *Response Processing* des Items gestartet. In diesem Vorgang werden die lokalen Outcome Variablen des Items aktualisiert. Den nächsten Schritt bildet der hier diskutierte Prozess, welcher die globalen Ergebnis Variablen setzt.
Der Zeitpunkt der Antwort- und Ergebnisverarbeitung ist davon abhängig, welcher Submission Modus definiert wurde. Bei einem individuellen Modus werden diese beiden Vorgänge nach jedem Item Versuch durchgeführt. Das heißt, die Antwort und Ergebnis Variablen aktualisieren sich mehrmals bei einer Testdurchführung. Der simultane Modus hingegen ermöglicht nur eine Durchführung dieser Prozesse am Ende eines Testteils. Diese Form der Abarbeitung wird auch als *Batch-Job* bezeichnet, wobei wiederum

für jedes Item einzeln die Verarbeitungsschritte gestartet werden.
Für diesen Verarbeitungsprozess stehen ebenso die gleichen Instruktionen und Kontrollstrukturen wie bei der Template und Response Verarbeitung zur Verfügung.
Ziel des *Outcome Processing* und den damit verbundenen Ergebnis Variablen sind die Datenbereitstellung für das Test Level Feedback (siehe 3.5.9) und die Wahl des passenden Pfades durch die weiteren Sektionen und Items.

3.5.9 Feedback

QTI bezeichnet jede Information, die dem Kandidaten aufgrund von Outcome Variablen gezeigt wird, als Feedback. In der Spezifikation existieren drei verschiedene Arten von Feedback.

Modales Feedback wird dem Kandidaten einzeln angezeigt. Das heißt, es wird nicht im Item Body integriert. Die Anzeige erfolgt direkt nach der Response Verarbeitung. Der Kandidat muss hierfür seine Antwort einreichen und kann diese danach nicht mehr ändern. Zur Definition von diesen Feedback Informationen gibt es eine eigene Sektion im Assessment Item.

Integriertes Feedback hingegen wird direkt im Item Body mit angezeigt. Sobald der Item Body neu gerendert wird, wird auch das integrierte Feedback entsprechend dargestellt. In diesem Fall kann die Testperson eine Modifikation der Antwort vornehmen, da diese noch nicht eingereicht wurde.

Test Feedback wird durch die Outcome Variablen eines Tests bestimmt. QTI unterstützt zwei Orte in denen ein Test Feedback definiert werden kann - *assessmentTest* und *testPart*. Die Anzeige erfolgt somit entweder nach der Verarbeitung der Outcome Variablen, am Ende einer *testPart* Definition, oder am Ende einer *assessmentTest* Definition.

Die beiden folgenden Grafiken 3.8 und 3.9 bringen die Unterscheidung zwischen modalem und integriertem Feedback näher.

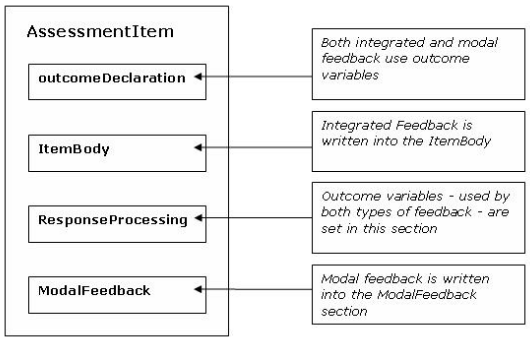


Abbildung 3.8: Feedback Sektionen in einem Assessment Item. [JISC CETIS, 2008]

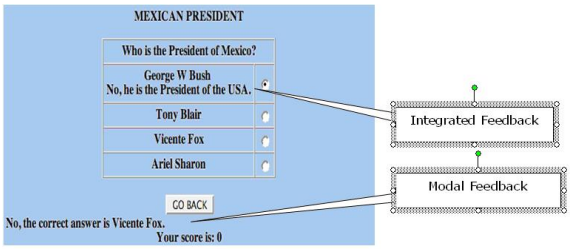


Abbildung 3.9: Darstellung von integriertem und modalem Feedback. [JISC CETIS, 2008]

3.6 Unterstützte Aufgabentypen

Die QTI Spezifikation unterstützt eine Vielzahl von Aufgabentypen und deren Interaktionen. Durch die Komposition dieser Typen kann die Artenvielfalt noch um einiges vergrößert werden. Im Vordergrund steht aber immer die Sicht des Testkandidaten. So sollten Item Interaktionen einfach gehalten werden und für die Testperson nachvollziehbar sein. Um festzustellen, ob mehrere Aufgaben oder eine Komposition erzeugt werden sollen, muss die Beziehung zwischen den Interaktionen untersucht werden. Funktionieren Interaktionen unabhängig von einer anderen, so müssen diese auch in einem separaten Item definiert werden. Besteht aber eine starke Verbindung unter den Interaktionen, so ist eine Komposition zu wählen [IMS Global Learning Consortium, 2006a]. In diesem Abschnitt werden mehrere Typen aufgelistet und klassifiziert. [Omar et al., 2005, S. 4f] beschreibt in der Tabelle 3.3 folgende Typen.

Aufgabentyp	Beschreibung
True/False	Dieser Fragetyp kann nur mit richtig oder falsch beantwortet werden.
Multiple Response	Eine Multiple Response Frage ist eine Mutiple Choice Frage mit mehreren Antwortmöglichkeiten. Um die Aufgabe korrekt zu erfüllen, muss die richtige Antwortkombination gewählt werden.
Fill-in-the-Blank	Diese Aufgabe stellt ein leeres Feld zur Verfügung, in das eine spezielle Information vom Kandidaten eingetragen werden muss. Diese Information kann entweder eine Zahl oder ein Text sein.
Slider	Bei einer Slider Aufgabe muss eine Testperson, entsprechend der Fragestellung, zwischen einem Zahlenbereich auswählen. Die Zahl kann über einen Schieberegler ausgewählt werden.
Drag Target	Zuvor definierte Objekte müssen in ein Zielobjekt gezogen werden.
Match Objects	Objekte aus einer Gruppe müssen Objekten einer Zielgruppe zugeordnet werden.
Multiple Choice	Bei dieser Frage kann eine Antwort aus verschiedenen Alternativen ausgewählt werden.
Image Hotspot	Der Image Hotspot ist eine definierte Koordinate im Bild, welche durch den Kandidaten gefunden werden muss.
Select Text	Die Testperson muss aus einem Paragraphen einen zuvor definierten Text bestimmen.
Drag Object	Bei dieser Aufgabe muss ein Kandidat verschiedene Objekte auf definierte Zielorte ziehen.
Order Objects	Um diese Frage zu lösen muss eine Testperson durchmischte Objekte, Texte oder Audios in eine spezielle Reihenfolge bringen.
Connect the Points	Der Benutzer muss verschiedene Punkte in einer definierten Reihenfolge in Verbindung bringen.
Short/Extended Text	Bei dieser Frage wird der Kandidat aufgefordert, einen Text zu schreiben.

Tabelle 3.3: Aufgabentypen, die von QTI unterstützt werden. [Omar et al., 2005, S. 4f]

Diese beschriebenen Aufgabentypen können nach [Winkelmann, 2005, S. 22] in Basistypen gruppiert werden. Die Basistypen bestehen aus *Logical Identifier*, *X-Y Coordinates*, *String*, *Numerical* und *Logical Group* [Risse, 2005].

Logical Identifier (LID) beinhalten alle Fragen, deren Antwortalternativen mittels ID oder ID Listen verwaltet werden. Ein Kandidat wählt eine oder mehrere Antworten aus und das System evaluiert deren Korrektheit und optional deren Reihenfolge.
X-Y Coordinates (X-Y) sind Fragetypen bei denen die Antwortmöglichkeiten über eine ID und X-Y Koordinaten verwaltet werden.
String (STR) oder auch Zeichenketten werden vom Lernenden eingegeben und durch das System bewertet.
Numerical (NUM) sind Fragen, die eine bestimmte Zahl vom Kandidaten verlangen. Der Zahlenwert wird vom System bewertet.
Logical Groups (GRP) definieren Positionen im Bildschirmbereich, welchen IDs zugewiesen werden. Diese Gruppen IDs werden ebenso den Antwortobjekten zugeordnet. Der Benutzer muss zur Lösung der Aufgabe die Antwort der entsprechenden Position zuweisen. Die Auswertung geschieht über deren Gruppen ID.

Zusätzlich wird hier noch zwischen den Wiedergabeformaten unterschieden. Ist nur eine Antwort möglich, so wird diese als *Single Response* bezeichnet. Bei mehrere möglichen Antworten wird das Wiedergabeformat als *Multiple Response* bezeichnet. Die *Ordered Response* erwartet zusätzlich noch eine spezielle Reihung der gegebenen Antworten. [Winkelmann, 2005, S. 23ff]

3.7 Integration in eine andere Spezifikationen

Um eine hohe Interoperabilität zu gewährleisten, kann QTI auch in andere IMS Standards integriert werden. Diese Standards zielen zum Teil auf den Datenaustausch zwischen Systemen, die Modellierung des Lernprozesses oder die Erstellung von Lernpfaden ab.

Das **IMS Content Packaging (CP)** stellt eine Spezifikation zur Verfügung, die es ermöglicht onlinebasierte Lernmaterialen auszutauschen. Der Standard definiert dabei ähnlich wie QTI ein Datenformat, welches auf XML aufbaut. Der Fokus liegt dabei auf der Interoperabilität zwischen Systemen die Importierung, Exportierung, Aggregation und Disaggreagtion von Lerninhalten benötigen. [IMS Global Learning Consortium, 2004]
QTI Formate können ebenso über Content Packages ausgetauscht werden. So werden einfach Assessment Objekte in ein Content Package Item Objekt verpackt. Diese Kollektion wird auch als *objectbank* bezeichnet. Durch verschiedene Metadaten, die aus den einzelnen Assessment Objekten gewonnen oder durch zusätzliche Definitionen erzeugt werden, wird die Wiederverwend- und Wiederfindbarkeit gefördert. Ebenso steigt die Austauschbarkeit durch die Verwendung eines weitverbreiteten Standards. [IMS Global Learning Consortium, 2006b]

Das **IMS Learning Design (LD)** und QTI sind direkte Partner im Lernprozess. Im Mittelpunkt dieser Spezifikation steht die Modellierung des Lernprozesses und seiner Lernaktivitäten und nicht mehr die Definition von Lerninhalten. Somit können umfassende Lernszenarios beschrieben werden. Lernaktivitäten können sich aus verschiedenen Aktivitäten, Assessments und Service oder Support durch Lehrpersonen zusammensetzen und werden mit verschiedenen Rollen verknüpft. [elive Learning Design, 2007]
Der primäre Fokus der Integration von QTI liegt auf der Durchführung von formativen Assessments. Denn bei dieser Art von Assessment muss ein Kandidat zuvor gewisse Lernaktivitäten durchführen, um spezielles Wissen zu erlernen. Nach Beendigung dieser Lernsequenz muss sich der Kandidat einer

Überprüfung unterziehen, welche auf QTI basiert. Durch die daraus resultierenden Ergebnisse, kann somit entschieden werden, ob der Lernende sein Wissen noch mehr vertiefen muss oder bereits auf die neue Lerneinheit wechseln kann. Die Kommunikation geht aber nicht nur in eine Richtung, sondern ist bidirektional. Das heißt, dass nicht nur Resultate aus QTI an LD übermittelt werden, sondern auch Lernpräferenzen eines Kandidaten von LD an QTI übergeben werden. [IMS Global Learning Consortium, 2006b]

Die **IMS Simple Sequencing (SS)** Spezifikation wird dafür genutzt, um Pfade durch eine Sammlung von Lernaktivitäten zu beschreiben. Es definiert dabei die relative Reihung, in der Lernaktivitäten einem Lernenden präsentiert werden. Diese Aktivitäten werden in einer Hierarchie verwaltet, einem so genannten *activity tree*. Jede einzelne Aktivität steht in Verbindung mit einem Zielobjekt. Diese Ziele zeichnen die Ergebnisse der Aktivitäten auf und dienen zur dynamischen Bestimmung des Lernpfades. [Bailey, 2005, S. 1]
Durch die Integration von QTI können auch Assessments oder sogar einzelne Assessment Items mit anderen Lernaktivitäten sequenziert werden. Das APIS Projekt[2] gibt hier näheren Aufschluss über diese Integration. [IMS Global Learning Consortium, 2006b]

3.8 Probleme von IMS QTI

Nach [Piotrowski und Fenske, 2007, S. 192] konnten einige Schwachstellen von QTI aufgedeckt werden. Dabei wurden die Probleme drei Bereichen zugeordnet: „Problematische Designentscheidungen“, „formale Schwächen“ und „Schwächen in der technischen Umsetzung von QTI in XML“.
Anzumerken ist, dass QTI nicht aus dem Praxis Umfeld gewachsen ist, sondern komplett neu geschrieben wurde. Ebenso existiert keine Referenzimplementierung von QTI 2.0+.

3.8.1 Problematische Designentscheidungen

3.8.1.1 Optionale Teile und eine liberale Spezifikation

Die Spezifikation QTI 2.0+ ist in vielen Teilen optional gestaltet. Um dennoch die Interoperabiltität zwischen Systemen zu gewähren, stellt IMS Profile zur Verfügung. Diese Profile teilen QTI in eine „Lite“ und eine „All“ Version. Laut [Piotrowski und Fenske, 2007, S. 192] sind diese Profile nur von geringem Nutzen. Das „Lite“ Profil beschreibt eine kleine Untermenge der Spezifikation und ist sogar für einfachste Tests nicht geeignet. So werden zum Beispiel keine Aufzählungszeichen oder Tabellen im Content Teil eines Items unterstützt. Bei den Bildformaten können nur JPEG und GIF benutzt werden, das PNG-Format wird von diesem Profil nicht unterstützt. Das „All“ Profil nimmt keine Einschränkungen vor und fordert somit eine Implementation der gesamten Spezifikation.
QTI bietet ein sehr liberales Konzept hinsichtlich der Inhalte von Items. So können zum Beispiel beliebige Interaktionen in einem Item kombiniert werden. Der Nachteil dieser Designentscheidung ist, dass bei Importen aus anderen unbekannten Systemen eine hohe Komplexität resultiert.

3.8.1.2 Trennung der Belange

Essentiell für eine gute Verwendbarkeit einer Datenstruktur ist die klare Trennung zwischen Präsentation und Inhalt. Im Zuge des Projektes EPHRAS[3] der Technischen Universität Graz wurden jedoch schwerwiegende Vermischungen dieser beiden Bereiche aufgedeckt. Zu dem Zeitpunkt der Projektdurchführung wurde die Spezifikation mit der Version 1.2 verwendet. Das folgende Code Listing 3.6 zeigt die Problematik bei einer Hotspot-Aufgabe:

[2] Assessment Provision through Interoperable Segments (http://www.jisc.ac.uk/whatwedo/projects/apis.aspx)
[3] EPHRAS - Ein mehrsprachiges phraseologisches Lernmaterial auf CD-ROM (http://www.ephras.org)

```
<presentation label="can_presentation">
  <flow>
    <material>
    </material>
    <response_lid ident="hotspot_a_2" rcardinality="Single">
        <material>
          <mattext texttype="text/html" x0="20" y0="50" width="600" height="60">
            <html><body><font face="Arial" size="4" color="#000033">Lesen Sie bitte den folgenden Text und versuchen Sie
                 das Phrasem zu entdecken! Markieren Sie den Satz oder Satzteil, in dem das Phrasem vorkommt!</font><br
                 /></body></html>
          </mattext>
        <render_choice>
          <material>
            <mattext texttype="text/html" x0="24" y0="135" width="310" height="20">
              <html><body><font face="Arial" size="4" color="#000033">Wer kennt sie nicht: diese fürchterlichen Tage,  </
                   font></body></html>
            </mattext>
          </material>
          <response_label ident="A">
            <material>
              <mattext texttype="text/html" x0="313" y0="125" width="280" height="20">
                <html><body><font face="Arial" size="4" color="#000033"> an denen einem wirklich gar nichts gelingt, </font
                     ></body></html>
              </mattext>
            </material>
          </response_label>
        </render_choice>
    </response_lid>
</flow>
</presentation>
```

Listing 3.6: Vermischung zwischen Präsentation und Inhalt von QTI 1.2.

Der Bereich eines Hotspots wird anhand dessen Koordinaten definiert (*Hotspot „A“:* `x0="313"-y0="125"width="280"height="20"`). Ändert in diesem Fall ein Itemautor die Schriftgröße oder erweitert den dargestellten Text, so müssen alle Hotspot Koordinaten geändert werden. Die Wartbarkeit leidet somit sehr unter diesem Umstand. Ein weiterer Kritikpunkt ist die daraus resultierende hohe Komplexität in der Verwendung der Spezifikation.
Das IMS Global Learning Consortium hat mit der Version 2.0 auf diese Problematik reagiert und führte einen neuen Interaktionstyp mit dem Namen Hottext ein. So können nun spezielle Hottext Bereiche ohne Koordinaten in einem Text definiert werden.
Weiterhin umständlich ist aber zum Beispiel die Erzeugung einer Aufgabe, indem ein vermischter Text entwirrt werden soll (Textsalat). Die Version 2.0 bietet hier zwei Interaktionsmöglichkeiten, die diese Aufgabe theoretisch umsetzen könnten. Die *gapMatchInteraction* ist aus Sicht der Datenrepräsentation, wie das Code Listing 3.7 zeigt, optimal. Dennoch scheitert die Verwendung dieser Interaktion an der vorgeschriebenen Darstellungsvariante von QTI (siehe Abbildung 3.10).

```
<itemBody>
    <gapMatchInteraction responseIdentifier="RESPONSE" shuffle="false">
        <prompt>Identify the missing words in this famous quote from Shakespeare's Richard III.</prompt>
        <gapText identifier="W" matchMax="1">winter</gapText>
        <gapText identifier="Sp" matchMax="1">spring</gapText>
        <gapText identifier="Su" matchMax="1">summer</gapText>
        <gapText identifier="A" matchMax="1">autumn</gapText>
        <blockquote>
            <p>Now is the <gap identifier="G1"/> of our discontent<br/> Made
                glorious <gap identifier="G2"/> by this sun of York;<br/> And all
                the clouds that lour'd upon our house<br/> In the deep bosom of the ocean buried.</p>
        </blockquote>
    </gapMatchInteraction>
</itemBody>
```

Listing 3.7: Eine Alternative in QTI 2.0 - *gapMatchInteraction*. [IMS Global Learning Consortium, 2005]

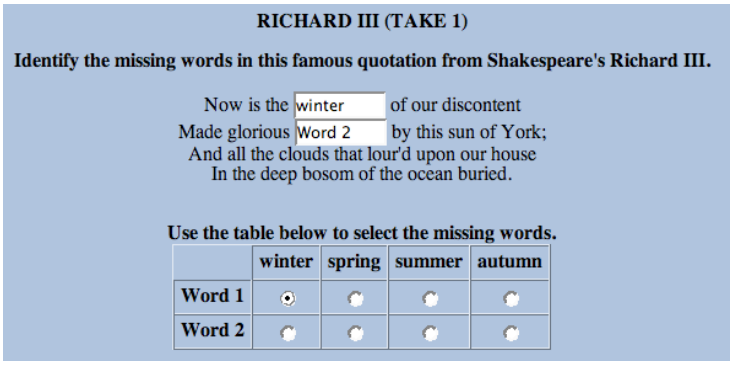


Abbildung 3.10: Empfohlene Darstellung von QTI 2.0. Aufgrund der Verwendung von Radio Boxen steigt die Komplexität des Lösungsvorganges mit der Lückenanzahl enorm. [IMS Global Learning Consortium, 2005]

Die zweite Alternative, die *graphicGapMatchInteraction*, entspricht der gewünschten Darstellungsform. Diese Interaktion ermöglicht zwar die Präsentation, doch zeigt sie die gleichen Schwächen, wie die alte Hotspot Interaktion aus der QTI Version 1.2. Auch hier müssen die Ziele aufwendig über Koordinaten definiert werden. Folgendes Code Listing 3.8 stellt diese Problematik dar.

```
<graphicGapMatchInteraction responseIdentifier="RESPONSE">
    <prompt>Some of the labels on the following diagram are missing: can you identify the
        correct three-letter codes for the unlabelled airports?</prompt>
    <object type="image/png" data="images/ukairtags.png" width="206" height="280"/>
    <gapImg identifier="CBG" matchMax="1">
        <object type="image/png" data="images/CBG.png" width="20" height="9"/>
    </gapImg>
    <gapImg identifier="EBG" matchMax="1">
        <object type="image/png" data="images/EBG.png" width="18" height="9"/>
    </gapImg>
    <associableHotspot identifier="A" matchMax="1" shape="rect" coords="12,108,39,121"/>
    <associableHotspot identifier="B" matchMax="1" shape="rect" coords="128,103,155,126"/>
</graphicGapMatchInteraction>
```

Listing 3.8: Eine weitere Alternative in QTI 2.0 - graphicGapMatchInteraction. [IMS Global Learning Consortium, 2005]

Im Gesamten betrachtet, bietet die Version 2.0+ eine klarere Trennung dieser beiden Belange. Dennoch sind auch weiterhin Schwächen vorhanden, die auch nicht durch die Verwendung von Cascading Style Sheets (siehe 3.5.4) gelöst werden können.
Auch im Bezug auf Logik und Inhalt weist der Standard eine Schwäche auf. Durch die Verwendung der eigenen Programmiersprache für die Auswertung von Items und die Template Verarbeitung, wird sehr viel Logik in das Datenformat verpackt. Da der Inhalt und die Verarbeitung von Items zwei unterschiedliche Aspekte sind, sollten diese nach [Piotrowski und Fenske, 2007, S. 192] in unterschiedliche Spezifikationen gegliedert werden. Somit würde auch die Komplexität von QTI gemindert werden.
Anzumerken ist weiters vor allem der Fakt, dass weiterhin die alte Version 1.2.1 als finale Version vom Konsortium empfohlen wird. Nähere Informationen dazu werden im Abschnitt 3.10 beschrieben.

3.8.2 Formale Schwäche

[Piotrowski und Fenske, 2007, S. 192ff] konnte einige formale Schwächen der Spezifikation identifizieren. QTI ist an vielen Stellen ungenau oder mehrdeutig beschrieben. Dadurch entstehen viele Graubereiche, die durch den Entwickler selbst erschlossen werden müssen. Nach den Autoren der zuvor genannten Arbeit ist es somit sehr wahrscheinlich, dass zwei Implementierungen eine unterschiedliche Interpretation von QTI besitzen. In Hinblick auf die Austauschbarkeit von Assessmentdaten würde dies schwerwiegende Folgen haben. Die Autoren zählen einige Beispiele dieser Unklarheiten auf. So wird zum Beispiel der *Identifier* in einer unklaren natürlichen Sprache beschrieben, anstatt eine kontextfreie

Gramatik zu verwenden. Weiters werden auch viele Attribute so wie deren Verwendung unverständlich spezifiziert.

3.8.3 Schwächen in der technischen Umsetzung

QTI umfasst neben dem Informationsmodell auch ein „XML Binding“ in dem durch ein XML Schema die Daten auf XML Elemente abgebildet werden. Laut [Piotrowski und Fenske, 2007, S. 192ff] werden die Möglichkeiten von XML und XML Schema in QTI nur ansatzweise verwendet, wodurch somit Restriktionen bei der Validation dieser QTI-Dokumente entstehen.
Ein Beispiel hierfür sind teilweise die Querverweise auf andere Elemente. So werden die richtigen Antwortmöglichkeiten bei Multiple Choice Frage im *correctResponse* Element mit einem oder mehreren *value* Elementen angegeben. Der Inhalt dieser *value* Elemente ist der Bezeichner einer korrekten *simpleChoice* Option.
Da eine bestimmte referenzierte *simpleChoice* Option auch fehlen könnte, ist dieser technische Ansatz nicht robust und somit fehleranfällig. Durch die Verwendung, der von XML zur Verfügung gestellten Attributtypen „ID“, „IDREF“ und „IDREFS“, könnte eine viel elegantere und robustere Lösung kreiert werden. In QTI werden diese Attributtypen jedoch an keiner Stelle eingesetzt.

3.9 Tools die QTI unterstützen

Die QTI Spezifikation kann, wie bereits unter 3.4 erwähnt wurde, von unterschiedlichen Werkzeugen unterstützt werden. Diese Tools haben verschiedene Aufgabenbereiche, wie die Erzeugung von Items, Verwaltung von Assessments/Items, Erzeugung von Assessments und Darstellung und Durchführung von Assessments. Die Werkzeuge werden unterteilt in:

- Authoring Tool
- Item Bank
- Test Construction Tool
- Assessment Delivery System
- Learning System

Im Zuge dieser Arbeit wurden verschiedene Implementationen, basierend auf einer Werkzeugmatrix (siehe Anhang A.1), untersucht. Auf Grund der Vermischungsproblematik von Präsentation und Inhalt aus Punkt 3.8.1.2, wurden die Tools auf die Spezifikation 2.0 und höher eingegrenzt. Eine weitere Einschränkung basiert auf einer vorweg gegriffenen Anforderung aus dem nächsten Kapitel. Die Werkzeuge sollen nicht kommerziell sein und für Erweiterungen offen stehen.
Diese Kriterien erfüllen nur die Applikationen Onyx[4], Elques[5], QTItools[6] und AQuRate[7]. Zudem wird der Fokus der Betrachtung auf die Erzeugung und Durchführung von Assessments und deren Items gelegt. In den folgenden Abschnitten wird diese Werkzeugauswahl, basierend auf den Quellen der Herstellerseiten, näher vorgestellt.

[4] http://onyx.bps-system.de
[5] http://elques.bps-system.de
[6] http://www.qtitools.org
[7] http://aqurate.kingston.ac.uk

3.9.1 Onyx

Onyx ist eine webbasierte Applikation zum Testen und Prüfen. Dieses Werkzeug ist als Open-Source verfügbar und unterstützt die QTI Spezifikation 2.1. Der Fokus der Applikation liegt auf der Testdurchführung. Somit fällt Onyx in die Kategorie einer *Assessment Delivery Engine*.
Onyx bietet eine Schnittstelle für die Einbindung in ein Lernmanagementsystem. So wird es bereits in OPAL[8], ein Lernmanagementsystem basierend auf LMS OLAT[9], eingesetzt. Dieses System bildet den Rahmen der Testdurchführung und übernimmt die administrativen Tätigkeiten.
Onyx unterstützt einige Interaktionen von QTI und bieten eine Medienunterstützung an, sowie diverse Auswertemöglichkeiten der Ergebnisse. Ebenso wird die Teststrukturierung in *testPart* und *sectionPart* ermöglicht. Durch die große Tragweite von QTI gibt es aber einige Restriktionen in der Konformität von Onyx. So werden zum Beispiel Interaktionen, wie *inlineChoiceInteraction*, *hottextInteraction* und *graphicGapMatchInteraction* oder auch Funktionalität in der Testdurchführung, wie der *submissionMode*, nicht implementiert.
Die Darstellung der Items baut sehr stark auf den Vorgaben von QTI auf. So wird zum Beispiel bei einer Zuordnungsaufgabe kein Drag and Drop unterstützt, was solche Aufgaben unübersichtlich gestaltet. Folgende Abbildung zeigt diese Problematik:

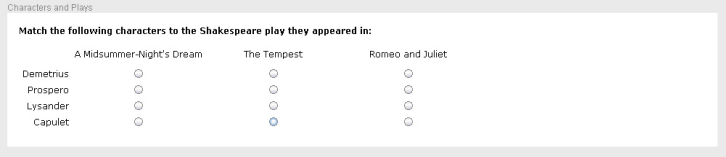


Abbildung 3.11: Darstellung einer Zuordnungsaufgabe von Onyx.

3.9.2 Elques

Elques ist eine Java-basierte Desktop Anwendung für die Erzeugung und Bearbeitung von Testdaten. Dieses Autorenwerkzeug ist in der Version 2.0 erhältlich und unterstützt ebenso die QTI Spezifikation 2.1 und ist als Basisversion freiverfügbar. Elques befindet sich noch im Entwicklungsstadium, da es bis jetzt nur die Grundfunktionalität zur Erstellung einfacher Testszenarien zur Verfügung stellt. Weiters unterstützt Elques zurzeit nur wenige Aufgabentypen wie: Auswahlaufgaben, Zuordnungsaufgaben, Audioaufgaben, Freitextaufgaben, Reihenfolgeaufgaben und Lückentexte. Die Abbildung 3.12 zeigt das Interface der Applikation und die verfügbaren Aufgabentypen:

3.9.3 QTItools

QTItools ist eine Sammlung von Werkzeugen, Bibliotheken und Web Services für die Verwendung von QTI 2.1. Alle Tools sind als Open-Source verfügbar. QTItools setzt sich aus den folgenden Projekten zusammen:

- **JQTI** - Eine Java-basierte Bibliothek für die Entwicklung von QTI Applikationen. Als zentrale Aufgabe übernimmt die Bibliothek die Funktion eines Interpreters für IMS QTI Inhalte. Ebenso stellt sie die Möglichkeit zur programmatischen Erzeugung von QTI Inhalten zur Verfügung, sowie deren Evaluierung.
- **R2Q2** - Ein Tool für das Rendering und die Durchführung von QTI Items (veraltet)
- **playr** - Ein Werkzeug für die Durchführung von Assessments (veraltet)

[8] http://www.olat.org
[9] https://bildungsportal.sachsen.de/opal

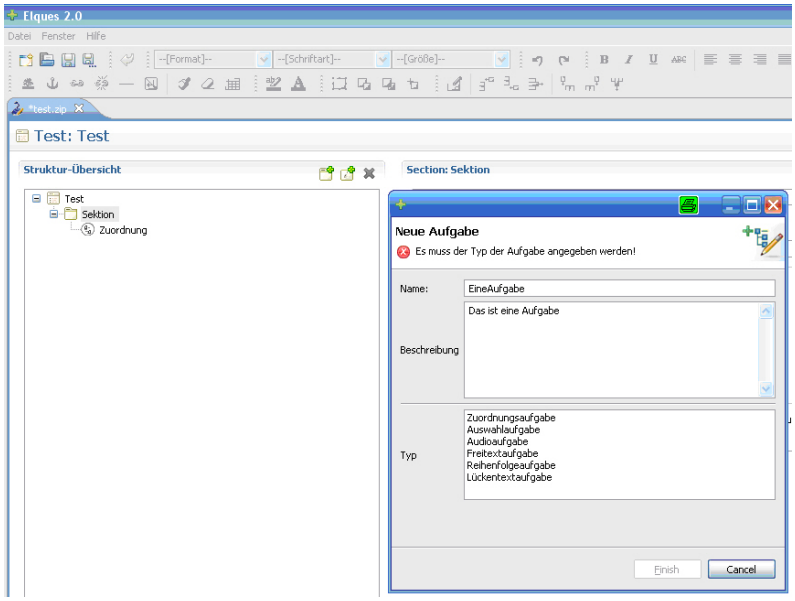


Abbildung 3.12: Interface von Elques und die verfügbaren Aufgabentypen.

- **assessr** - Eine webbasierte Applikation für die Ablaufsteuerung von Assessments (veraltet)
- **QTIEngine** - Eine Engine für die Durchführung von Assessments, basierend auf JQTI. Sie vereint die veralteten Projekte R2Q2, playr und assessr.
- **validatr** - Ein Werkzeug für die Validierung von Assessments
- **constructr** - eine webbasierte Applikation für die Erzeugung von Asessments mit Items aus einer Item Bank

Alle genannten Projekte befinden sich im Entwicklungsstadium und unterstützen nur Teile von QTI. Die Dokumentation und die Projekt Website[10] fallen sehr spärlich aus und bieten somit wenig Information.

3.9.4 AQuRate

AQuRate ist ein Open-Source Autorenwerkzeug zur Erzeugung von Assessment Items. Diese Desktopanwendung ist das Schwesterprojekt zu QTItools und baut ebenso auf der Version 2.1 von QTI auf.
Die Applikation unterstützt bis jetzt nur wenige Interaktionen von QTI. Wichtige Interaktionen wie Hottext oder Zuordnung von Elementen werden nicht angeboten. Auch die Kombination von verschiedenen Interaktionen ist nicht implementiert. Ebenso fokussiert die Applikation ausschließlich auf die Erzeugung von Items und bietet keine Assessment Unterstützung an. Die Abbildung 3.13 zeigt das Interface der Applikation und die verfügbaren Aufgabentypen:

[10] http://wiki.qtitools.org

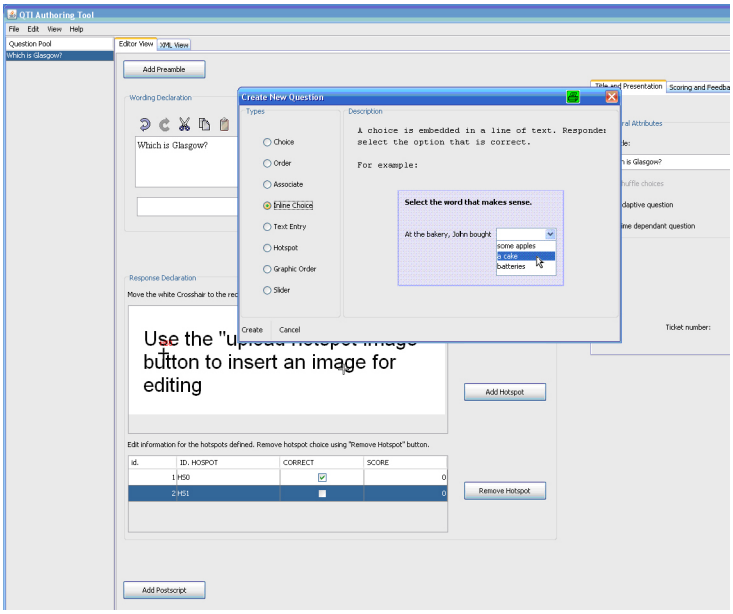


Abbildung 3.13: Interface von AQuRate und die verfügbaren Aufgabentypen.

3.9.5 Fazit

Alle bisher untersuchten Werkzeuge, die QTI 2.1 unterstützen und als Open-Source verfügbar sind, befinden sich noch im Entwicklungsstadium. Die Liste der unterstützten Aufgabentypen ist sehr beschränkt. Vor allem bei den Autorenwerkzeugen werden nur sehr wenige Itemtypen angeboten. Die QTIEngine von QTItools besitzt hier noch die größte Auswahl. Eine weitere Problematik ist die fehlende oder schlechte Dokumentation vieler Werkzeuge. Das trifft vorwiegend für die Sammlung der QTItools zu. Auch die geringe Anzahl an Implementationen, die in dieses Raster fallen, stellt ein Problem dar. Somit kann daraus geschlossen werden, dass IMS QTI nicht den erhofften Anklang findet. Ein weiterer Kritikpunkt ist die teilweise nicht komfortable Darstellung von Aufgaben. So könnte zum Beispiel häufiger die Drag und Drop Technik verwendet werden, ohne die Spezifikation von QTI zu verletzen. Alle Werkzeuge halten sich jedoch strikt an den Darstellungsvorgaben von QTI, obwohl das Rendering proprietär sein könnte.

3.10 Rückzug der Version 2.1 und seine zukünftige Entwicklung

Die aktuellste QTI Version 2.1 wurde als Public Draft im Frühjahr 2009 zurückgezogen. Der Grund für dieses Vorgehen war nach Blog Eintrag[11], dass in einer Zeitspanne von zwei Jahren nur wenige Implementationen erreicht wurden und die Rückmeldungen nicht ausreichend waren. IMS empfahl dabei die finale Version 2.0 nicht zu verwenden, sondern die alte Version 1.2.1 zu gebrauchen. Version 2.0 stellt schließlich nicht die ganze Spezifikation dar. Ebenso wurde angedeutet, dass eine neue Version 2.2 nicht unbedingt an der Entwicklung von 2.0 und 2.1 anknüpfen muss. Ein weiterer Punkt, warum diese Annullierung stattgefunden hat, war nach dem Kommentar[12] die Problematik, dass manche Firmen nicht die ganze Spezifikation implementieren wollten, sondern nur die Teile des Standards, die sich mit ihren Kundenwünschen decken. IMS plante hierzu ein Set von Profilen zu erstellen, die genau auf die unterschiedlichen Kundenwünsche passen.
Welche Konsequenzen hatte dieser Rückschritt? Firmen, die bereits die Spezifikation implementierten,

[11] http://blogs.cetis.ac.uk/rowin/2009/04/03/ims-withdraw-qti-v21-draft-specification/
[12] http://lists.ucles.org.uk/public/ims-qti/2009-March/001480.html

oder jene, die dies geplant hatten, wurden durch dieses Vorgehen sehr verunsichert. Jens Schlengel[13], CEO von BPS Bildungsportal Sachsen GmbH, bezeichnet den Rückzug sogar als großen Schaden für die zukünftige Akzeptanz der e-Learning Standardisierungsarbeit. Ebenso verteidigt er das geringe Feedback mit dem Argument, dass durch die hohe Qualität des Standards 2.1 nur wenige Probleme auftraten und somit auch wenig Feedback nötig war. Ein weiteres Problem ist der Rückschritt auf die Version 1.2.1. Mathematiker und Wissenschaftler konnten somit nicht mehr die Erweiterung von MathML nutzen. Weiters ist auch der Trend der Spezifikation, für jeden flexibel zu gestalten und propritäre Module zu erlauben, nach dem Kommentar[14] sehr gefährlich. Denn diese Entwicklung führt weg vom ursprünglichen Gedanken eines Standards.
Schlussendlich hat sich das IMS Konsortium nach Drängen vieler Implementoren und Mitglieder aber wieder dafür entschieden, IMS QTIv2.1 als Entwurf auf die Website zu stellen. Die finale Version von 2.1 ist jedoch noch ausständig.

3.11 Konklusion

Die QTI-Sepzifikation bietet ein sehr durchdachtes und umfassendes Informationsmodell zur Verwendung von Assessments und Assessment Items unter der Beachtung der Zeitkomponente. Das Ziel des Standards ist es, die Interoperabilität und die Kommunikation zwischen verschiedenen Lernsystemen zu fördern. Auch Innovation findet in diesem Standard Platz, indem proprietäre Module hinzugefügt werden können. Die Spezifikation bildet ein einheitliches System für eine große Anzahl an Testsituationen. Hierzu wird ermöglicht, dass QTI sich in viele andere etablierte IMS Standards integrieren lässt. Das Datenformat XML bietet QTI ebenso eine sehr gute Grundlage um diese Integrationen durchzuführen.
Nachteilig ist jedoch die Komplexität der Spezifikation. Zum einen für die Implementoren, da sie auf Grund des großen Umfanges vor eine große Herausforderung gestellt werden. Diese Entwickler müssen zudem auch entscheiden, in wie weit sie den Standard konform implementieren, da Kundenwünsche oft nur einen Teil dieser Spezifikation fordern. Zum anderen bereitet die Komplexität den Autoren Schwierigkeiten. Diese Sachgebietsexperten haben zwar ein tiefes Verständnis ihres Lehrinhaltes, jedoch fehlen ihnen meistens die Fähigkeiten um Aufgaben in diesem Datenformat zu erzeugen. Somit kann die Verwendung dieses Standards eher auf IT Didaktiker beschränkt werden. Die Arbeit mit dieser Spezifikation setzt somit sehr viel Fachwissen voraus. Einher mit der Formatkomplexität geht auch die Komplexität der verwendeten Ausdrücke und Kontrollstrukturen. Diese *light-weight* Programmiersprache kann schnell unübersichtlich und unverständlich werden. QTI bietet hier zwar die Möglichkeit, diese Schritte auszulagern und wieder zu verwenden, doch müssen auch diese Strukturen erst einmal erzeugt werden.
Bei einem weiteren Kritikpunkt kann die verwendete *light-weight* Programmiersprache auch von einem anderen Licht betrachtet werden. So können zum Beispiel nur ganz einfache Kontrollstrukturen bei der Verarbeitung der Antwort oder des Ergebnisses verwendet werden. Komplexe Aufgabenstellungen sind somit unmöglich zu generieren.
Schwierigkeiten bereitet auch die schlechte Dokumentation und Darstellung von Anwendungsszenarien. Die Spezifikation ist zwar eine gute Grundlage für Entwickler von Systemen, doch legt sie keinen Fokus auf deren Anwender. Es sind zwar viele Beispiele vorhanden, doch ist der Aufbau des Standards nur schwer aus den Dokumenten zu erfahren.
Nach [Winkelmann, 2005, S. 29] sind die Möglichkeiten zur Aufgabenerstellung begrenzt. Durch den Einsatz von XML ist es sehr schwer, Simulationen, virtuelle Experimente oder andere interaktiv multimedial gestaltete Übungsaufgaben umzusetzen. Zudem besteht für Winkelmann ein Problem darin, dass bei der Auswertung von Aufgaben die Antwort immer auf einen Endzustand reduziert wird. Somit ist die Überprüfung von konzeptionellem prozeduralen Wissen nur beschränkt möglich.
Ein besonderer Nachteil der QTI Spezifikation ist die unklare Trennung der verschiedenen Belange. Im besonderen betrifft dieser Umstand die zurzeit finale Version 1.2. Die neue Version 2.0+ reagiert auf diese

[13] http://lists.ucles.org.uk/public/ims-qti/2009-March/001465.html
[14] http://lists.ucles.org.uk/public/ims-qti/2009-March/001482.html

Problematik und ermöglicht zusätzlich die Verwendung von Cascading Style Sheets. Doch auch mit dieser neuen überarbeiteten Spezifikation entstehen solche Schwierigkeiten bei speziellen Aufgabentypen. Ein großer Schwachpunkt in der neuen Spezifikation ist vor allem die Vermischung von Verarbeitungslogik und Inhalt.
Die Anzahl der Werkzeuge, die auf QTI aufbauen, sind nur selten frei erhältlich. Die getesteten Tools sind noch dazu im Entwicklungsstadium und unterstützen meist nur wenige Aufgabentypen.
Ein großes Problem bildet auch der derzeitige Umstand der nicht vollendeten Version 2.1. Viele Firmen wurden dadurch verunsichert und die Entwicklung in diesem Segment verlangsamt. Weiters ist bis jetzt auch unklar, in welche Richtung die Spezifikation in Zukunft gehen wird.
Zu diesen bereits genannten Nachteilen kommt noch die Tatsache, dass die Wiederverwendung von Assessment Inhalten viel weniger nachgefragt wurde, als am Anfang angenommen wurde. [Gonzalez-Barbone und Llamas-Nistal, 2007]

Kapitel 4

Anforderungsanalyse der SprichWort - Übungen

„Wollen heißt: den Mut haben, sich einer Unannehmlichkeit auszusetzen; sich dieser auszusetzen heißt: den Zufall wagen, also spielen.“

[Stendhal (1783 - 1842), französischer Schriftsteller]

In diesem Kapitel wird für das Projekt „SprichWort“ eine Anforderungsanalyse des Übungssystems durchgeführt. Hierbei werden funktionale wie auch nicht funktionale Anforderung in Hinsicht auf die Erzeugung, die Durchführung, die Bewertung und die Ergebnisbetrachtung getroffen. Im Anschluss darauf wird IMS QTI als Grundlage für das Übungssystem diskutiert. In diesem Zuge wird auch die Einsetzbarkeit der QTI Werkzeuge betrachtet und evaluiert. Doch zunächst wird ein Einblick in das Projekt SprichWort gegeben.

4.1 Das Projekt SprichWort. Eine Internet-Lernplattform für das Sprachenlernen

SprichWort[1] ist ein internationales Projekt, in dem Gemeinsamkeiten und Unterschiede im heutigen Sprichwortgebrauch untersucht werden. Das Projekt wird vom Programm für Lebenslanges Lernen - (LLP) der Europäischen Kommission für zwei Jahre (2008-2010) finanziert. Das Projekt wird durch die Philosophische Fakultät der Universität Maribor (Slowenien) koordiniert. Ebenso beteiligt sind Sprachwissenschaftler der Universitäten Szeged (Ungarn), Trnava (Slowakei), Zlín (Tschechien), sowie des IDS Mannheim (Deutschland). Die technische Unterstützung erfolgt durch die Technische Universität Graz. Die in der Projektbeschreibung verwendete Quelle stammt aus [SprichWort-Team, 2009].

4.1.1 Die Ziele

Das Projekt verfolgt das Ziel, Sprichwörter als Kulturgut einer Sprachgemeinschaft im heutigen Gebrauch zu dokumentieren und zu vermitteln. Dabei werden Sprichwörter durch empirische Methoden, basierend auf einer großen elektronischen Textdatenbank, ermittelt und zugänglich gemacht. SprichWort baut auf drei Komponenten auf: SprichWort-Datenbank, SprichWort-Übungen, SprichWort-Community.

[1] http://www.sprichwort-plattform.org

4.1.2 Die Nutzer

Die Zielgruppe sind Fremdsprachenlerner und -lehrende, sowie Sprachwissenschaftler und Entwickler neuer Lernmaterialien. Ebenso bietet die Plattform für Interessierte Informationen zum heutigen Gebrauch verschiedener Sprichwörter.

4.1.3 Die Ergebnisse

Die SprichWort-Datenbank ermöglicht es, Sprichwort-Äquivalente in Deutsch, Slowenisch, Slowakisch, Tschechisch und Ungarisch vernetzt abzurufen. Durch die interaktiven SprichWort-Übungen wird der Lernprozess erleichtert und der Lernfortschritt kann dargestellt werden. Die SprichWort-Community wird durch ein integriertes Diskussionsforum gefördert. In diesem Forum wird der Austausch von Ideen und Konzepten rund um diese Thematik angeregt.
Die Projektgruppe erhofft sich neue Erkenntnisse in der linguistischen Forschung. Thematik ist vor allem die mehr oder weniger feste Verbindung von Wörtern. Sie bildet das zentrale Objekt der sprachwissenschaftlichen Diskussion. Durch die automatischen Recherchеverfahren können große elektronische Textdatenbanken systematisch analysiert werden. So eröffnen sich neue Sichtweisen auf Funktionsweisen der Sprache selbst. So hat sich herausgestellt, dass Menschen viel stärker mit sprachlichen Fragmenten kommunizieren, als bisher, ohne die elektronischen Verfahren, angenommen wurde. Diese Wortverbindungen haben sich als Muster in den Köpfen der Sprecher gefestigt und können variabel in der entsprechenden Kommunikationssituation abgerufen und modifiziert werden. Sprichwörter verkörpern diese Textbausteine in einer besonderen Weise, da anhand ihrer Verwendung viel Wissen über die Festigkeit und Variabilität sprachlicher Ausdrücke gewonnen werden kann.

4.2 Zielsetzung der SprichWort-Übungen

Ziel dieses Teilabschnittes des SprichWort Projekts ist es, autonomes Lernen in Bezug auf Sprichwörter zu fördern. Dies ist, wie es das Projekt vorsieht, jedoch nicht ohne sinnvolle Selbstkontrolle möglich. Ein Kandidat muss sich schließlich selbst über seinen Lernfortschritt überzeugen, da ihm keine reale Lehrkraft für Rückmeldungen zur Verfügung steht. Die Plattform sieht somit drei verschiedene Hilfsmittel vor:

- Übungen - besitzen formativen Charakter um Wissen zu festigen
- Tests - geben direkte Rückschlüsse über den eigenen Lernerfolg
- Selbstbewertungsbögen - ermöglichen eine Reflexion über das eigene Lernen

Die Plattform fordert, in anderen Worten, ein webbasiertes Übungs- und Testsystem mit formativem und summativem Charakter. Dieses muss sowohl standardisierte objektive als auch nicht standardisierte subjektive Fragen in Form von Textaufgaben verarbeiten und verwalten können.
Didaktiker oder andere Instruktoren können durch dieses System Übungs- oder Testsequenzen sowie die beinhalteten Fragen erstellen und verwalten. Im Vordergrund steht hierbei die einfache Verwendung des Systems, ohne allzu großes technisches Hintergrundwissen zu besitzen. Dabei wird gefordert, dass die Plattform eine einfache Syntax und einen Editor für diese Aufgaben zur Verfügung stellt.
Für die Prüfungsdurchführung werden die Aufgabenstellungen den Kandidaten (Schüler oder Studenten) zur Verfügung gestellt. Dazu müssen sich die Testpersonen auf der Plattform einloggen und im Anschluss die Selbstüberprüfungen über einen Webbrowser durchführen. Bei den Prüfungen wird zwischen Übungen und Tests unterschieden. Ein Test ermöglicht nur einen Lösungsversuch pro Frage und muss in einer bestimmten Zeitvorgabe gelöst werden. Bei den Übungen existiert diese Restriktion nicht. Die Evaluierung der Fragen kann entweder direkt nach jedem Lösungsversuch durch das System oder im

Anschluss einer Prüfung durch den Instruktor stattfinden. Der Prüfer kann die ausständigen Antworten, die sich aus Textaufgaben ergeben, über eine spezielle Schnittstelle bewerten.
Das System bietet ebenso verschiedene Ansichten auf die Prüfungsergebnisse. Zum einen können Prüfer die gesamten Assessments betrachten, zum anderen können Kandidaten ihre eigenen Prüfungen durchsehen. Zusätzlich können Testpersonen die subjektiven Antworten anderer Studenten betrachten und evaluieren.
Um eine Reflexion über das eigene Lernen zu gewährleisten, stellt das System verschiedene Selbstbewertungsbögen zur Verfügung. Diese können durch den Kandidaten ausgefüllt und gespeichert sowie nach Bedarf ausgedruckt werden.

4.3 Zielgruppen und Phasen

Die Zielgruppen des Systems bauen auf verschiedenen Schwierigkeitsstufen auf. Diese können in folgende Gruppen gegliedert werden:

- Stufen B1-B2
- Stufen C1-C2
- Germanistikstudenten

Für Germanistikstudenten sind nicht nur Fragen aus den Stufen B und C bestimmt, sondern auch linguistische Fragen, in denen die Klassifikation von Sprichwörtern und deren etymologischer Ursprung Thematik ist.
Die Aufgaben der Stufen B und C werden in folgende Phasen unterteilt:

1. Sprichwörter erkennen. Sprichwörter können erst in konkreten Texten ihre kommunikative Wirkung entfalten. Deshalb werden sie in authentischen und teilweise adaptierten Texten vorgestellt. Die Grundlage für die Texte bilden die Datenbank sowie auch verschiedene Textbelege.
2. Sprichwörter verstehen. Die Kandidaten werden hier zur Sinnsuche aktiviert. Sie sollten dabei einen selbstständigen Versuch der Entschlüsselung unternehmen und die Bedeutung mit Hilfe von kontextgebundenen Informationen suchen.
3. Sprichwörter festigen. Der Fokus liegt hier auf der Vielfältigkeit der Fragen. Die Übungsformen sollen somit verschiedene Lerntypen ansprechen. Zu den klassischen Aufgaben wie Multiple Choice oder Zuordnungsübungen sollen auch spielerische interaktive Aufgaben gestellt werden.
4. Sprichwörter verwenden. Diese Phase umfasst die metakommunikative Einbettung der Sprichwörter in Texten sowie die subjektive Beantwortung von Textaufgaben.

4.4 Rollen und Anwendungsfälle des Systems

Die Rollen des Systems bestehen aus einem Instruktor und einem Kandidaten.

Instruktor - Der Instruktor beinhaltet mehrere kleine Subrollen und stellt den lehrenden Part dar. Diese Subrollen setzen sich wiederum aus einem Itemautor, einem Item Verwalter, einem Testkonstrukteur, einem Proktor und einem Bewerter zusammen. Dadurch, dass eine Person all diese Aufgaben bewältigen muss, können diese auf die Rolle des Instruktors zusammengefasst werden.
Die Aufgaben eines Instruktors definieren sich wie folgt:

- Erstellen, Modifizieren und Löschen von Items
- Erstellen, Modifizieren und Löschen von Tests und Übungen
- Prüfungen den Kandidaten zur Verfügung stellen (damit verbunden eine Benutzer- und Gruppenverwaltung)
- Lernende durch einen Lernprozess dirigieren
- subjektive Antworten bewerten
- Kandidaten Kommentare zu anderen Ergebnissen kommentieren
- Testergebnisse evaluieren
- Erstellen, Modifizieren und Löschen von Selbstbewertungsbögen

Eine wichtige Anforderung dieser Rolle ist die einfache Verwaltung von Prüfungen und Prüfungsfragen.

Kandidat - Der Kandidat setzt sich aus den verschiedenen Zielgruppen des Systems zusammen und bildet den lernenden Part der beiden Rollen.
Folgende Aufgaben und Bedürfnisse bilden seinen Charakter:

- eigene Benutzerverwaltung (registrieren, einloggen, verwalten)
- Prüfungen durchführen
- Prüfungsfragen lösen
- die Ergebnisse der absolvierten Prüfungen betrachten
- Kommentieren der Textproduktionen anderer Kandidaten
- Selbstbewertungsbögen ausfüllen
- Selbstbewertungsbögen betrachten

Eine grundlegende Anforderung dieser Rolle ist eine einfache Navigation durch verschiedene Prüfungsaufgaben.
Die Diagramme 4.1 und 4.2 stellen die verschiedenen Anwendungsfälle des Systems dar:

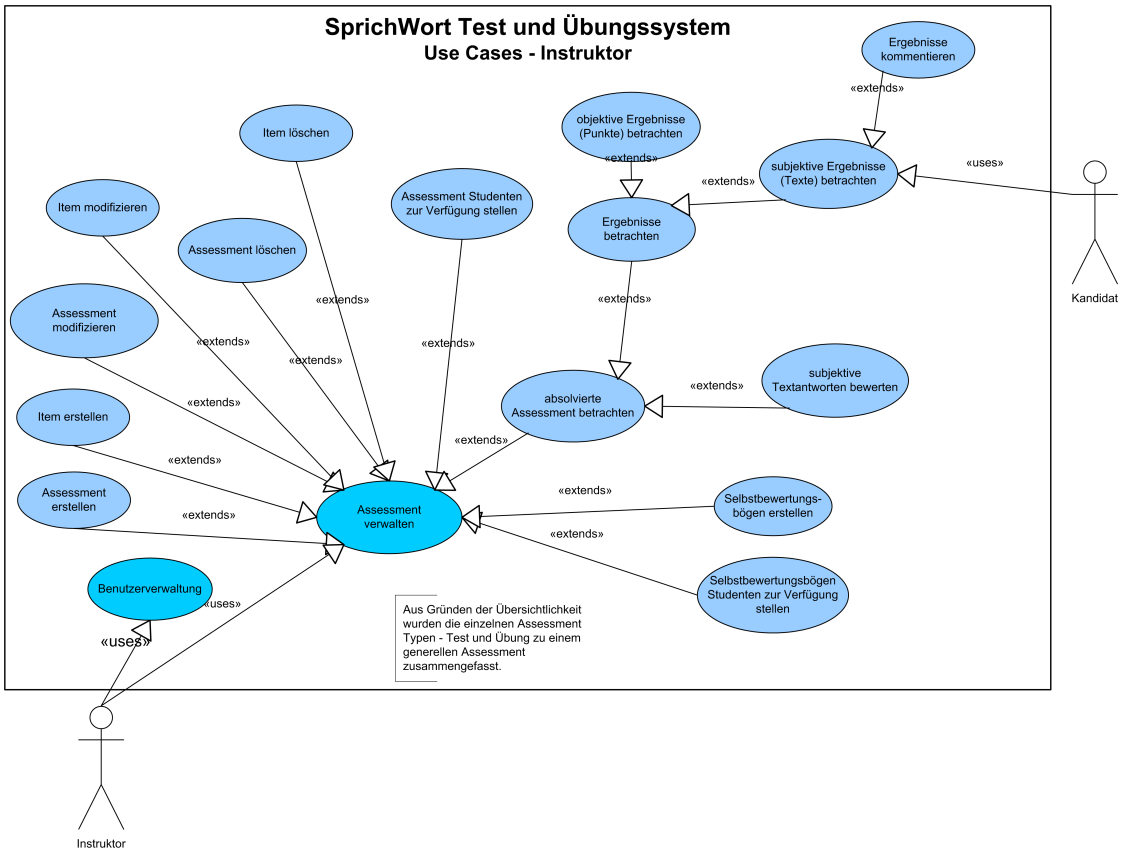


Abbildung 4.1: Anwendungsfalldiagramm für den Instruktor.

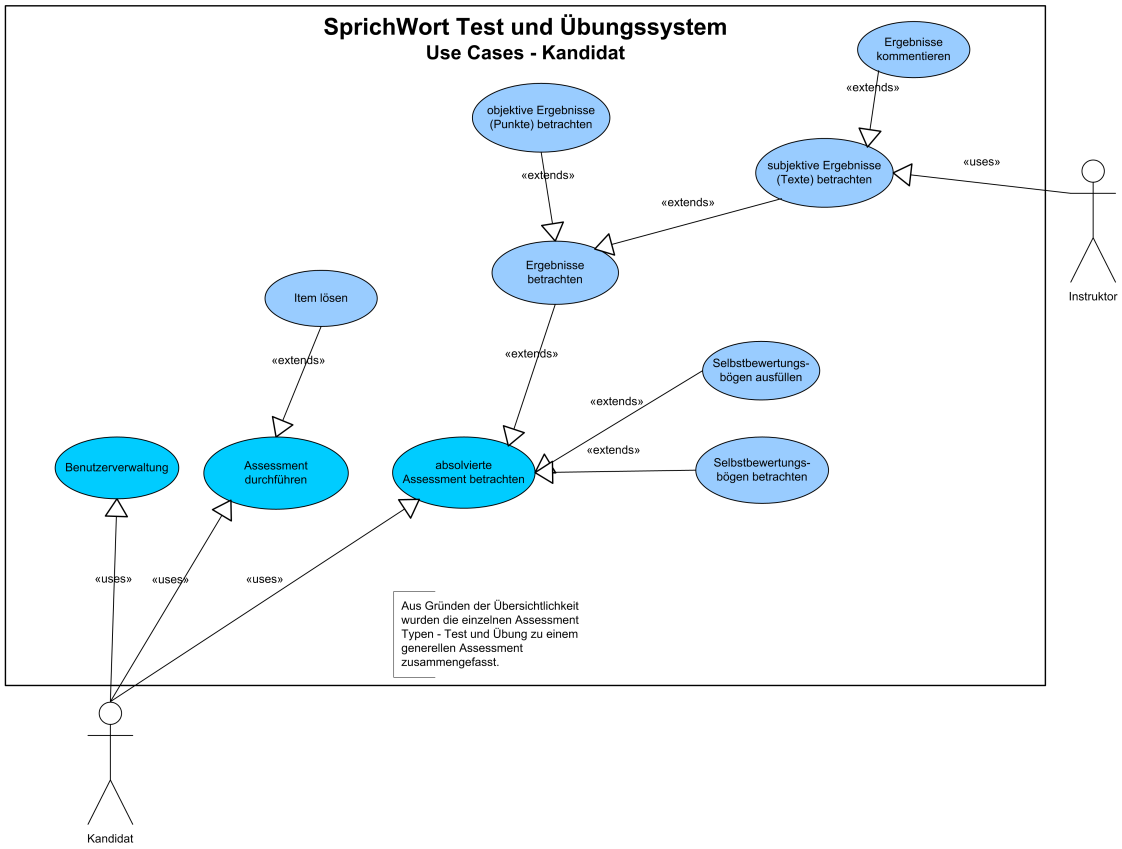


Abbildung 4.2: Anwendungsfalldiagramm für den Kandidaten.

4.5 Funktionale Anforderungen

4.5.1 Aufgabentypologie

Das Projekt fordert eine Fülle an verschiedenen Aufgabentypen. Diese Typenformate können in Hotspots, Multiple Choice Fragen, Lückentexte, Korrekturaufgaben, Zuordnungsaufgaben und Textproduktion unterteilt werden. Ebenso sind Aufgaben wie zum Beispiel ein Memory oder auch hybride Formen wie Hotspot Korrektur oder Hotspot Multiple Choice gefordert. Die nachfolgende Auflistung gibt einen Überblick über die detaillierten Anforderungen der Aufgabentypologie[2] aus dem Projekt.

4.5.1.1 Hotspot

Sprichwort kennzeichnen

Ablauf: Ein oder mehrere Sprichwörter müssen in einem Text erkannt werden. Dazu klickt der Kandidat auf die Textpassagen. Werden alle Sprichwörter erkannt, gibt es eine volle Punktzahl, ansonsten eine prozentuelle Aufteilung.

Zuordnung zum Aufgabenbedarf: [1.]

Fehler kennzeichnen

Ablauf: Ein oder mehrere Fehler werden in einem Text dargestellt. Der Kandidat muss diese Fehler markieren. Werden alle Fehler gefunden, erreicht der Kandidat die volle Punktzahl, ansonsten eine prozentuelle Aufteilung.

Zuordnung zum Aufgabenbedarf: [5.]

Fehler korrigieren

Ablauf: Ein oder mehrere Fehler werden in einem Text dargestellt. Der Kandidat muss diese Fehler markieren und korrigieren. Werden alle Fehler gefunden und ausgebessert, erreicht der Kandidat die volle Punktzahl, ansonsten eine prozentuelle Aufteilung.

Zuordnung zum Aufgabenbedarf: [6.]

Anmerkung: Diese Aufgabe ist eine hybride Form. Sie erfordert eine Hotspot- und eine Korrekturmöglichkeit.

Sprichwort ergänzen

Ablauf: Mehrere Fehler oder Lücken werden in einem Text dargestellt. Der Kandidat muss diese Fehler oder Lücken markieren und durch ein Multiple Choice Menü korrigieren oder ergänzen. Werden alle Fehler oder Lücken gefunden und ausgebessert, erreicht der Kandidat die volle Punktzahl, ansonsten eine prozentuelle Aufteilung.

[2] http://www.sprichwort-plattform.org/attach/Didaktik_Diskussion/Uebungs%20und%20aufgabentypen%20templatesbedarf.doc

Zuordnung zum Aufgabenbedarf: [8.]

Anmerkung: Diese Aufgabe ist eine hybride Form. Sie erfordert eine Hotspot- und eine Multiple Choice - Möglichkeit und wird nur verwendet, wenn mehrere Fehler oder Lücken in einem Text vorhanden sind.

4.5.1.2 Multiple Choice Fragen

Bedeutung von Sprichwort bestimmen

Ablauf: Ein Sprichwort wird angezeigt. Der Kandidat muss die Bedeutung aus einer Multiple Choice Liste auswählen. Wird die richtige Antwort gewählt, so gibt es die vollen Punkte, ansonsten werden keine Punkte vergeben.

Zuordnung zum Aufgabenbedarf: [2.]

Sprichwort durch Bedeutung bestimmen

Ablauf: Die Bedeutung eines Sprichwortes wird angezeigt. Der Kandidat muss das dazugehörige Sprichwort aus einer Multiple Choice Liste auswählen. Wird die richtige Antwort gewählt, so gibt es die vollen Punkte, ansonsten werden keine Punkte vergeben.

Zuordnung zum Aufgabenbedarf: [12.]

Sprichwort ergänzen

Ablauf: Ein Text mit einer Lücke wird angezeigt. Der Kandidat muss den passenden semantischen Teil aus einer Multiple Choice Liste auswählen. Wird die richtige Antwort gewählt, so gibt es die vollen Punkte, ansonsten werden keine Punkte vergeben.

Zuordnung zum Aufgabenbedarf: [8.]

Anderssprachige Entsprechung

Ablauf: Ein Sprichwort wird angezeigt. Der Kandidat muss die passende anderssprachige Entsprechung aus einer Multiple Choice Liste auswählen. Wird die richtige Antwort gewählt, so gibt es die vollen Punkte, ansonsten werden keine Punkte vergeben.

Zuordnung zum Aufgabenbedarf: [10.][9.]

4.5.1.3 Lückentext (Buchstabe, Wort, Satz)

Ablauf: Eine oder mehrere Lücken werden in einem Text dargestellt. Der Kandidat muss diese markieren und ausbessern. Werden alle Lücken ausgebessert, gibt es eine volle Punktzahl, ansonsten eine prozentuelle Aufteilung.

Zuordnung zum Aufgabenbedarf: [3.] [12.] [8.]

Anmerkung: Diese Aufgabe ist eine hybride Form. Sie erfordert eine Hotspot-, eine Lückentext- und eine Korrekturmöglichkeit.

4.5.1.4 Drag and Drop (Zuordnung)

Sprichwortsalat

Ablauf: Es wird ein Feld mit vermischten semantischen Teilen eines oder mehrerer Sprichwörter angezeigt. Ein Kandidat muss diese Anordnung entwirren und in dafür vorgesehene Felder ziehen, um einen korrekten Satz zu bilden. Die Punktevergabe erfolgt nach gelösten Sätzen.

Zuordnung zum Aufgabenbedarf: [4.]

Ein mögliches Beispiel kann der Abbildung 4.3 entnommen werden:

Abbildung 4.3: Ein Beispiel für eine Sprichwortsalat Aufgabenstellung.

Zuordnung - Objekt auf Ziel (Ziel frei platzierbar)

Ablauf: Es wird eine Liste mit Objekten (Texte, Bilder) angezeigt. Der Kandidat muss diese Elemente auf ein Zielobjekt ziehen. Werden alle Objekte richtig zugeordnet, gibt es eine volle Punktzahl, ansonsten eine prozentuelle Aufteilung.

Anmerkung: Das Zielobjekt ist bei dieser Aufgabe frei platzierbar.

Ein mögliches Beispiel kann der Abbildung 4.4 entnommen werden:

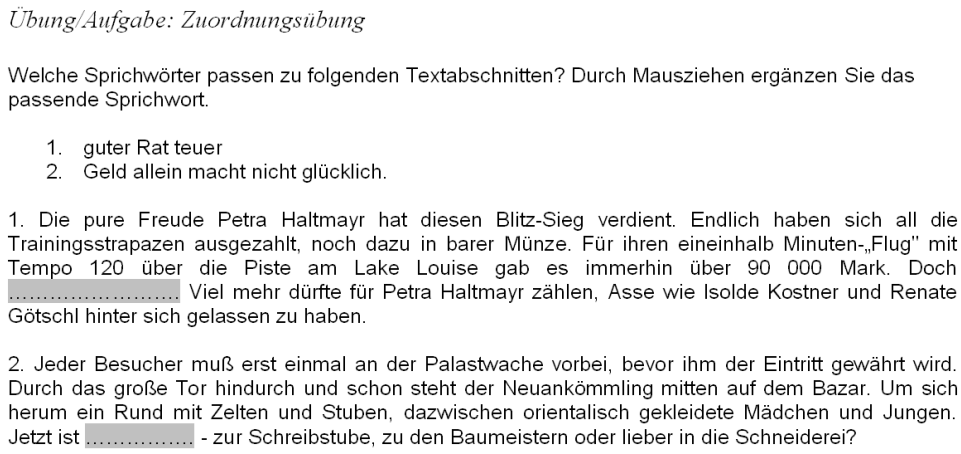


Abbildung 4.4: Ein Beispiel für die Zuordnung eines Objektes auf ein frei platzierbares Zielobjekt.

Zuordnung - Objekt auf Ziel (Ziel in Raster)

Ablauf: Es werden eine Liste mit Objekten (Texte, Bilder) und ein Raster mit Zielen angezeigt. Der Kandidat muss diese Elemente auf ein Zielobjekt ziehen. Werden alle Objekte richtig zugeordnet, gibt es eine volle Punktezahl, ansonsten eine prozentuelle Aufteilung.

Zuordnung zum Aufgabenbedarf: [10.] [11.] [12.]

Anmerkung: Zu diesem Typus zählen Aufgaben wie die Zuordnung von anderssprachigen Entsprechungen auf Sprichwörter, Bilder auf Sprichwörter, Sprichwörter auf Bilder, Sprichwörter auf Bedeutungen und Hörtexte auf Sprichwörter.

4.5.1.5 Datenbank-Aufgaben

Ablauf: Ein Kandidat sucht bestimmte Informationen aus der SprichWort Datenbank. Diese Informationen verwendet er, um die Aufgabe zu lösen.

Anmerkung: Diese Aufgabe kann auf allen Aufgabentypen basieren.

4.5.1.6 Textproduktion

Ablauf: Es wird eine offene Frage angezeigt. Der Kandidat muss dazu einen Text produzieren. Diese Frage kann nicht automatisch vom System korrigiert werden und muss somit an einen Instruktor weitergeleitet werden. Die Bewertung erfolgt im Anschluss nach Ermessen des Instruktors.

Zuordnung zum Aufgabenbedarf: [12.] [14.] [15.]

Anmerkung: Es werden bei der Durchführung Tipps und Hinweise gegeben.

4.5.1.7 Alphanumerische Zuordnung

Ablauf: Ein Kandidat muss einen semantischen Teil oder ein anderes Objekt einem weiteren Teil/Objekt zuordnen. Dies erfolgt durch die Eingabe verschiedener Indizes.

Zuordnung zum Aufgabenbedarf: [7.]

4.5.1.8 Interaktive Spiele - Memoryspiel

Ablauf: Es wird ein Memoryfeld angezeigt. Der Kandidat muss dieses lösen. Punkte werden bei dieser speziellen Übungsaufgabe keine verteilt.

Im Anhang B.1 wird eine Übersicht verschiedener interaktiver Sprachspiele nach [Kacjan, 2008, S. 69f] dargestellt. Diese Spiele könnten in einer Erweiterung eine Ideengrundlage bilden.

4.5.2 Unterscheidung zwischen Übungen und Tests

Das System unterstützt sowohl formative Prüfungen in Form von Übungen und summative Prüfungen in Form von Tests.

4.5.2.1 Übungen

Übungen können auch als formative Prüfungen bezeichnet werden. Sie dienen dazu, Informationen und Richtungsweisungen für den laufenden Lernprozess zu liefern. Ein Kandidat kann über ein Menü, je nach Phase und Zielgruppe, verschiedene Übungen auswählen.
Die Übungen bestehen aus den Instanzen der oben genannten Aufgabentypen. Die Aufgaben werden je nach Typus vom System oder von einem Instruktor bewertet. Somit sind diese Übungen nicht nur Selbstbewertungen, sondern besitzen auch einen Charakter der Fremdbewertung. Um einen weiteren Lerneffekt durch Kommunikation in die Übung zu bringen, ist es möglich, dass bereits produzierte Texte von Textaufgaben durch andere Kandidaten kommentiert werden können. Diese Kommentare erzeugen somit eine Gruppenbewertung.
Bei der Durchführung der Übungen ist es möglich, Fragen mehrmals zu beantworten. Die Ergebnisse dieser Übungen werden nicht gespeichert. Das gilt jedoch nicht für die Textaufgaben, da diese mit Kommentaren versehen werden. Übungen beinhalten neben Aufgaben, die Punkte erzeugen, auch Aufgaben, die rein zur Interaktion dienen. Solche Aufgaben sind zum Beispiel Memoryspiele.

4.5.2.2 Tests

Tests werden auch als summative Prüfungen bezeichnet. Sie dienen im Allgemeinen zur Untersuchung der Frage, ob und eventuell wie gut ein bestimmtes Lernziel erreicht wurde. (Vgl. [Lienert und Raatz, 1998] und [Doyé, 1988] beide zit. nach [Kacjan et al., 2009])
Der Kandidat wählt einen Test zu einem bestimmten Bedeutungsbereich wie z.B.: Handeln, Hoffnung oder Gemeinsamkeit aus und löst die vorgegebenen Aufgaben unter Einhaltung einer Prüfungszeit. Die Testaufgaben umfassen jeweils die drei angesprochenen Niveaus.
Die Tests bestehen aus den oben genannten Aufgabentypen mit der Ausnahme von Aufgaben, die keine Punkte liefern, und Textaufgaben. Diese Beschränkung entsteht durch den Umstand, dass keine reale Lehrkraft für einen Test zur Verfügung steht.
Die Aufgabenelemente eines Tests besitzen den gleichen Ablauf wie die einer Übung mit der Restriktion, dass die Aufgaben nur einmal gelöst werden dürfen.
Die Punkte werden im System gespeichert. Nach der erfolgten Bewertung können die Ergebnisse vom Kandidaten betrachtet werden. Bei Bedarf kann ein Test auch wiederholt werden.

Testkonstruktion und Aufgabenbeantwortung Für die Umsetzung von Tests wurden so genannte Testmodelle (TM) entwickelt. Sie bilden die Basis für die Sicherung und Erprobung von Sprichwortkenntnissen. Bis jetzt wurden drei TM aufbauend auf der Aufgabentypologie entwickelt. Die TM sind für die Kandidaten bestimmt, die feststellen wollen, auf welchem Niveau sie sich befinden und in wie weit sie die Sprichwortgruppen beherrschen. Tests dienen somit als Indikator zur Überprüfung des Lernfortschrittes.

Testmodell A - Einsprachige Variante In diesem Modell dient Deutsch als Ausgangs- und Zielsprache. Da Deutsch als Verbindungssprache zwischen den vier weiteren am Projekt beteiligten Sprachen gilt, ist das Modell in der deutschen Fassung dargestellt. Das Prinzip dieses Tests wird auch auf alle anderen beteiligten Sprachen übertragen, sofern es sich um rein einsprachige Tests handelt.

Testmodell B - Zweisprachige Variante Dieses Modell kombiniert die Sprache Deutsch mit einer der vier weiteren Sprachen (Slowenisch, Slowakisch, Tschechisch, Ungarisch). Eine dieser vier genannten Sprachen bildet die Muttersprache eines Kandidaten. Durch den eventuell möglichen sprachlichen Transfer kann das Verstehen und Beherrschen der zuvor erlernten Sprichwörter erleichtert werden. Zudem werden auch das Bewusstsein und die Beherrschung der interkulturellen und interlingualen Komponenten der Sprichwörter überprüft.

Testmodell C - Für Parömiologie-Interessierte und Germanistikstudenten In diesem Modell werden spezielle Testaufgaben zum parömiologischen Wissen vorgestellt. Die Testaufgaben sind in erster Linie für Germanistikstudenten und Parömiologie-Interessierte bestimmt, können aber natürlich auch von allen anderen genutzt werden, die ihr spezifisch parömiologisches Wissen einfach nur überprüfen möchten.

Aufgabenstrukturiertheit Gebundene Aufgabenbeantwortung (man muss zwischen vorgegebenen Antworten wählen):

1. Richtig/Falsch - es ist anzugeben, ob eine Aussage richtig oder falsch ist (True-False).
2. Mehrfachwahlaufgabe - eine aus mehreren Antworten ist auszuwählen (Multiple-Choice).
3. Zuordnungsaufgabe - zwei Reihen von Merkmalen sind einander zuzuordnen.

Freie Aufgabenbeantwortung (man muss selber die Antwort zu einer Aufgabenstellung finden - Lückentexte):

1. direkte Fragen
2. unvollständige Aussagen
3. Anweisungen zur Auswahl.

Alle Aufgaben haben nur eine richtige Lösung und sind den Niveaustufen B1-2 und C1-2 des gemeinsamen europäischen Referenzrahmens für Sprachen und dem Niveau parömiologischen Wissens zugeordnet. Die von den Benutzern erreichten Resultate werden in einer zentralen Datenbank auf der Internetplattform gespeichert und sind ausschließlich dem Benutzer und den didaktischen Begleitern (Entwickler der didaktischen Inhalte) zu beraterischen Zwecken zugänglich.

Verwendete Notenskala Die Testmodelle sind so konstruiert, dass sie den Grad der Fertigkeit bestimmen, den der Benützer durch das Lernen und Üben der Sprichwörter erworben hat. Die Ergebnisse ergeben sich aus der Zahl der richtigen Lösungen in der gegebenen Lösungszeit. Wenn alle Resultate korrekt sind, heißt das, dass der Test zu 100% richtig gelöst wurde.

Prozentuelle Korrektheit der Testaufgaben zu einer Sprichwort-Gruppe	Bewertung
90-100%	Super!
70-89%	Sehr gut!
50-69%	Gut!
49% und weniger	Schade. Du musst noch üben.

Tabelle 4.1: Bewertung von Tests.

4.5.3 Formale Gestaltung der Übungen und Tests

Bei den Übungen und Tests werden folgende Mittel eingesetzt:

- Farben - in Bezug auf die Phasen sind die Hintergrundfarben des Bildschirms unterschiedlich.

- Niveauangaben - die einzelnen Übungen sind den Niveaustufen des Referenzrahmens zugeordnet.
- Symbole - sie erleichtern das Lernen und die Orientierung.
- Lerntipps - sind didaktische Empfehlungen bzw. anklickbare Kommentare, wie effektiver gelernt werden kann.

4.5.4 Selbstbewertungsbögen

Um eine Reflexion über das eigene Lernen zu ermöglichen, stellt das System Selbstbewertungsbögen zur Verfügung. Diese Bögen streben folgende Ziele an:

- Förderung der Fähigkeit, sich selbst nach vorgegebenen Lernzielbeschreibungen zu bewerten
- Feststellung des Kompetenzniveaus anhand von Niveauzuordnungen
- Rückschlüsse ziehen auf Grundlage der erhaltenen Ergebnisse
- Motivation zum selbstbewertenden Lernen von Sprichwörtern

Ein Selbstbewertungsbogen wird nicht zu einzelnen Sprichwörtern angeboten, sondern umfasst jeweils eine Gruppe von Sprichwörtern zu einem bestimmten Bedeutungsbereich. Dabei beinhaltet ein solcher Bogen den bearbeiteten Bedeutungsbereich und umfasst die vier Phasen: Erkennen, Verstehen, Festigen und Anwenden. Die Phasen enthalten jeweils die unterschiedlichen Niveaus und die dazugehörigen Kompetenzen, wie auch die Lerntipps. Ein Kandidat kann zwischen drei verschiedenen Bewertungen wählen: „Das kann ich gut oder sehr gut“, „... einigermaßen“, „... noch nicht so richtig“. Je nach Selbstbeurteilung werden entsprechende Lerntipps für das weitere Vorgehen empfohlen.
Das Modell dieses Selbstbewertungsbogens wird anhand von folgendem Beispiel 4.5 veranschaulicht:

Bearbeiteter Bedeutungsbereich: z.B.: „Handeln“

Niveau	Kompetenzen (Kannbeschreibungen)	Das kann ich … gut oder sehr gut. ☺ ✓✓	einigermaßen. 😐 ✓	noch nicht so richtig. ☹ ✗	Lerntipp
SW erkennen					
B1-2	Ich kann die SW in (authentischen) Texten identifizieren.	x			Lies noch einmal die Belege zu den SW in der Datenbank (DB).
SW verstehen					
B1-2	Ich verstehe die Bedeutung der SW.		x		Schau die Bedeutung in der DB nach.
B1-2	Ich kenne die muttersprachliche Entsprechung der SW.	x			Diese Information findest du unter Äquivalente in anderen Sprachen in der DB.
SW festigen					
B1-2	Ich kann die Sprichwörter grammatisch korrekt bilden.		x		Überprüfe in der DB, wie die SW lauten bzw. lauten können.
B1-2	Ich kann SW korrekt ergänzen.			x	Hilf dir mit der DB.
Paröm.	Ich kann die SW parömiologisch angemessen analysieren.			x	Nimm die ganzen DB Seiten zur Hilfe.
SW anwenden					
B1-2	Ich kann die SW korrekt in kohärenten Texten verwenden.		x		Lies die Belege in der DB und die Beispiele anderer Lernenden.
Paröm.	Ich kann die SW bezüglich der parömiologischen Charakteristiken mit anderen vergleichen oder über sie schreiben.		x		Suche weitere Informationen auch außerhalb der Plattform.
Gesamtergebnis: Anzahl der ✓ und ✗		2x	4x	2x	**Lerntipp:** Bei mangelhaften Kompetenzen (✗), die entsprechenden Aufgaben und Übungen noch einmal machen.

Abbildung 4.5: Ein exemplarischer Selbstbewertungsbogen.

4.5.5 Parametrisierung und Randomisierung

Um Aufgaben interessanter zu gestalten, unterstützt das System Parametrisierung und Randomisierung der Aufgaben. So können Templates erstellt werden, die über spezielle Parameter konfigurierbar sind. Wird eine Instanz dieses Templates erstellt, wird eine Konfiguration zufällig gewählt und angezeigt.
Bei einer Hotspot-Aufgabe werden zum Beispiel drei zuvor konfigurierte Hotspots jeweils zufällig initialisiert. Die Abbildung 4.6 zeigt eine beispielhafte Darstellung.

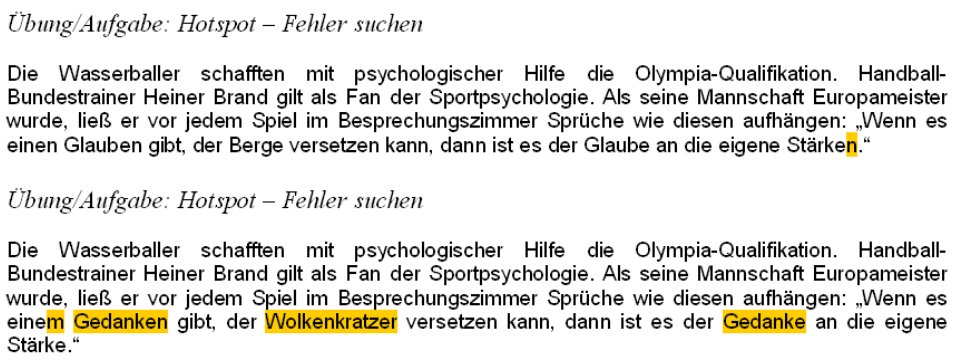


Abbildung 4.6: Ein Beispiel für eine Hotspot-Aufgabe, die drei zuvor konfigurierte Hotspots jeweils zufällig initialisiert.

4.5.6 Prozesse

Das Projekt beinhaltet vier Hauptprozesse. Diese Prozesse umfassen die Erzeugung, Durchführung und Evaluierung von Tests/Übungen und Aufgaben, sowie die Betrachtung derer Ergebnisse. Die nachfolgenden Abschnitte geben einen Überblick über den Ablauf und die Anforderungen der jeweiligen Prozesse. Auf das Usermanagement wird hier keine Rücksicht genommen, da es als generelle Anforderungen bereits erwähnt wurde.

4.5.6.1 Test-, Übung- und Aufgabenerzeugung

Bei der Erzeugung von Tests und Übungen kann aus den oben genannten Aufgabentypen gewählt werden. Für diese Erstellung ist ein Autorenwerkzeug vorgesehen. Dieses Werkzeug ermöglicht eine schnelle Erstellung über eine einfache Syntax oder ein UI.
Für eine Aufgabe müssen verschiedene Parameter definiert werden. Die wichtigsten Elemente hierfür sind der Aufgabentyp, die Frage oder der Stimulus, die Instruktion, die Antwortmöglichkeiten, wenn nötig die Distraktoren und die richtige Lösung. Zusätzlich können auch Hinweise definiert werden. Ebenso wird die Parametrisierung und Randomisierung berücksichtigt. Der Instruktor kann der erstellten Aufgabe auch eine spezielle Phase, sowie ein Niveau zuweisen. Diese dienen dem Kandidaten zur Orientierung.
Nachdem die Aufgabe erstellt wurde, kann diese einem Test oder einer Übung zugewiesen werden und dadurch Sequenzen erstellt werden. Bei Tests stellt das System die Möglichkeit einer Zeitkonfiguration zur Verfügung. Damit kann definiert werden, wie lange ein Kandidat eine Prüfung lösen darf. Im Anschluss werden die erstellten Übungen in ein dem Kandidaten zur Verfügung stehendes Menü eingetragen.
Nach der Erstellung dieser Daten werden diese vom System gespeichert. Der Instruktor kann auch nachträglich Änderungen vornehmen. Daraus entsteht das Bedürfnis einer Datenbank für Aufgaben, Tests und Übungen, welche zudem auch eine Revisionsverwaltung besitzt.

4.5.6.2 Test-, Übung- und Aufgabendurchführung

Die Durchführung von Übungen und Tests unterscheidet sich in einigen Punkten. Bei Übungen kann der Kandidat selbstständig aus einer Übungsliste auswählen. Der Kandidat kann während der Durchführung Hinweise und die richtige Lösung anzeigen. Wird die Funktion „Lösen“ gewählt, so werden alle richtig gelösten Teile in einer bestimmten Farbe angezeigt. Die falschen oder noch ausständigen Antworten, bei denen eine Lösung angezeigt werden müsste, werden mit einer anderen Farbe markiert. Ein Kandidat kann mehrmals eine Aufgabe lösen und selbst den Pfad durch die Übungssequenz wählen.
Bei Tests kann der Kandidat aus verschiedenen Bedeutungsbereichen bzw. Sprichwortgruppen wählen und die Durchführung beginnen. Bei dieser Prüfungsart werden keine Hilfestellungen angezeigt. Zusätzlich wird dem Kandidaten eine verbleibende Zeit angezeigt, in welcher er die Aufgaben lösen muss. Nach Ablauf dieser Zeit dürfen keine erneuten Lösungsversuche durchgeführt werden.
Beide Prüfungstypen beinhalten die gleiche Navigation. Ein Kandidat kann frei zwischen allen Aufgaben wählen, welche er lösen möchte. Bei Tests besteht die Restriktion, dass eine Aufgabe nur einmal gelöst werden darf.

4.5.6.3 Test-, Übung- und Aufgabenevaluierung

Die Evaluierung erfolgt entweder direkt nach der Beantwortung der Frage durch das System, oder indirekt durch einen Instruktor. Die Art der Bewertung hängt vom Aufgabentyp ab. Textproduktionen können nur subjektiv durch einen Instruktor bewertet werden. Das System unterstützt somit die Vergabe von Feedback in Form von Punkten oder Kommentaren.
Bei der subjektiven Evaluierung stellt das System dem Instruktor eine einfache Schnittstelle zur Bewertung zur Verfügung. Ebenso gibt es eine Schnittstelle für andere Kandidaten, um die Kommentierung von Texten ihrer Kollegen vorzunehmen.
Der Kandidat kann auch Selbstbewertungen mit Hilfe eines Selbstbewertungsbogens durchführen. Hierzu kann der Kandidat je nach Bedeutungsbereich einen Selbstbewertungsbogen öffnen und ausfüllen. Beurteilt ein Kandidat sich zu schlecht, so wird je nach betreffender Kompetenz ein Lerntipp angezeigt. Die Selbstbewertungsbögen werden in einer Datenbank gespeichert und können nach Bedarf ausgedruckt werden.

4.5.6.4 Ergebnisse betrachten

Das System bietet dem Kandidaten die Möglichkeit, seine Testergebnisse sowie auch die Selbstbewertungen anzusehen. Dazu wird eine einfache Schnittstelle zur Verfügung gestellt. Ebenso stellt das System eine Schnittstelle für die Betrachtung der Kommentare seiner Kollegen zu seinen Texten bereit.
Der Instruktor kann über das System die Ergebnisse aller durchgeführten Übungen und Tests betrachten.

4.5.7 Überblick der funktionalen Anforderungen

1) Generelle Anforderungen

   1.1) Unterstützung von Übungen und Tests: Das System soll Übungen wie auch Tests unterstützen.

   1.2) Hohe Aufgabenvielfalt: Das System soll eine große Anzahl an Aufgaben/Fragen zur Verfügung stellen.

   1.3) Parametrisierung/Randomisierung: Das System soll einen Mechanismus zur Parametrisierung und Randomisierung von Aufgaben besitzen.

   1.4) Authentifizierung/Userverwaltung: Das System soll einen Mechanismus für die Authentifizierung und Userverwaltung zur Verfügung stellen.

1.5) Zugriffskontrolle/Gruppenverwaltung: Das System soll den Zugriff von Unbefugten auf Tests oder Übungen verhindern und zusätzlich auch Schutz während der Erstellungsphase bieten.

1.6) Mediasupport: Das System soll multimediale Inhalte unterstützen.

1.7) Sprache: Die Menüpunkte sollen in unterschiedlichen Sprachen darstellbar sein.

2) Test-, Übung- und Aufgabenerzeugung

2.1) Einfache Erzeugung von Aufgaben: Das System soll eine einfache Syntax und einen Editor zur Erzeugung von Tests/Übungen und Aufgaben bieten.

2.2) Einfache Erzeugung von Test- und Übungssequenzen: Das System soll eine einfache Syntax und einen Editor zur Erzeugung von Test- oder Übungssequenzen bieten.

2.3) Zeiteinstellungen: Der Instruktor soll über das System einfache Zeitkonfigurationen für Tests vornehmen können.

2.4) Phasendefinition: Das System soll die Möglichkeit bieten, verschiedene Phasen zu definieren und diese in Tests/Übungen anzuzeigen.

2.5) Niveaudefinition: Das System soll die Möglichkeit bieten, verschiedene Niveaus zu definieren und diese in Tests/Übungen anzuzeigen.

2.6) Hinweise für Aufgabenerfüllung erzeugen: Das System soll die Möglichkeit bieten, Hinweise zu erzeugen.

2.7) Revisionshandling: Das System soll ein Revisionshandling von Aufgaben unterstützen.

3) Test-, Übungs- und Aufgabendurchführung

3.1) Auswahl von Tests/Übungen durch Kandidaten: Das System soll ein Menü zur Verfügung stellen, aus dem ein Kandidat die Übung oder den Test auswählen kann.

3.2) Navigation durch Test- und Übungssequenzen: Das System soll eine Navigation zwischen Aufgaben eines Tests oder einer Übung ermöglichen.

3.3) Zeitlimit: Der Kandidat soll durch das System mit einer zuvor definierten Zeitvorgabe bei der Testdurchführung beschränkt werden.

3.4) Hinweise zu Aufgaben: Das System soll bei Übungen definierte Hinweise anzeigen.

3.5) Lösungen zu Aufgaben: Bei Übungen soll das System die Option bieten, Lösungen anzuzeigen.

3.6) Punktevergabe: Das System soll eine Punktevergabe unterstützen.

4) Test-, Übungs- und Aufgabenevaluierung

4.1) Automatisierte objektive/nicht automatisierte subjektive Evaluierung von Tests/Übungen: - Aufgaben sollen entweder vom System oder von einem Instruktor bewertet werden können (abhängig von deren Typus).

4.2) Interface für Evaluierung durch Instruktor: Das System soll dem Instruktor ein einfaches Interface bieten, um eine Bewertung vorzunehmen.

4.3) Gruppenbewertung/Kommentarfunktion: Das System soll anderen Kandidaten ermöglichen, die Textproduktionen anderer Kollegen zu kommentieren.

4.4) Selbstbewertungsbögen ausfüllen: Das System soll die Möglichkeit bieten, Selbstbewertungsbögen auszufüllen und zu speichern.

5) Ergebnisse betrachten

5.1) Darstellung der Ergebnisse aus Sicht des Kandidaten: Das System soll ein benutzerfreundliches Interface bieten, um die eigenen Testergebnisse zu betrachten.

5.2) Darstellung der Ergebnisse aus Sicht des Instruktors: Das System soll ein benutzerfreundliches Interface bieten, um die Testergebnisse aller Kandidaten zu betrachten.

5.3) Selbstbewertungsbögen anzeigen: Das System soll die Möglichkeit bieten Selbstbewertungsbögen anzuzeigen.

4.6 Nicht funktionale Anforderungen

4.6.1 Zuverlässigkeit

Das System soll eine hohe Testzuverlässigkeit aufweisen. Eine inkonsistente Punktevergabe kann den Lernerfolg und die Akzeptanz durch die Kandidaten mindern. Ebenso muss das System an sich eine hohe Zuverlässigkeit bieten, da ein Absturz während einer Prüfung nicht von Vorteil ist.

4.6.2 Benutzerfreundlichkeit

Die einfache Benutzbarkeit hat die höchste Priorität unter den Anforderungen. Instruktoren müssen in der Lage sein, ohne großes technisches Hintergrundwissen Aufgaben, Übungen und Tests zu erstellen, durchzuführen und zu bewerten. Dabei steht vor allem die Erzeugung im Vordergrund, da eine große Menge an Aufgaben erzeugt werden soll.

4.6.3 Open-Source

Eine weitere wichtige Anforderung ist, dass das System auf Open-Source Software aufbaut. Somit soll gewährleistet werden, proprietäre Erweiterungen durchzuführen.

4.7 IMS QTI als Grundlage für das Übungssystem

In diesem Abschnitt wird die Einsetzbarkeit von IMS QTI für das zuvor beschriebene Übungssystem diskutiert. Auf Grund der unterschiedlichen Anforderungen muss bei dieser Betrachtung zwischen der Einsetzbarkeit der generellen Spezifikation und der möglichen Verwendung ihrer Implementationen unterschieden werden.
QTI beschreibt nur ein standardisiertes Datenformat für die Darstellung und Repräsentation von Assessmentdaten. Viele Aufgabentypen können durch dieses Datenformat repräsentiert werden.
Funktionale Anforderungen wie Benutzerverwaltung oder Gruppenbewertungen sind nicht die Aufgaben von QTI, sondern werden von verschiedenen Applikationen übernommen. Diese Implementationen, sofern sie vorhanden sind, sind jedoch in Bezug auf Benutzbarkeit oder Reife nicht brauchbar. Die Werkzeuge müssen je nach Aufgabengebiet unterschieden werden. So können diese, wie bereits im Abschnitt 3.4 gelistet wurde, in *Authoring Tools*, *Item Banks*, *Test Construction Tools*, *Assessment Delivery Systems* (auch *Player* genannt) und *Learning Systems* unterteilt werden. Im Zuge dieser Arbeit wurde der Fokus auf *Assessment Delivery Systems* und *Authoring Tools* gelegt. Auf Grund der Vermischungsproblematik von Präsentation und Inhalt aus Abschnitt 3.8.1.2 wurde die Begrenzung auf Tools gelegt, die auf der Spezifikation 2.1 basieren. Daraus folgend wird auch nur die QTI Spezifikation 2.1 den Anforderungen gegenüber gestellt.
Viele Anforderungen sind jedoch sehr proprietär, sodass einige Erweiterungen vorgenommen werden müssen. Deshalb ist es eine Grundvoraussetzung, dass dieses System auf Open-Source Basis zur Verfügung steht.
Die Tabelle 4.2 weist die Anforderungen des Übungs- und Testsystems der QTI Spezifikation und den entsprechenden Werkzeugen zu. Dadurch können weitere Untersuchungen über die Erfüllbarkeit in den nächsten Abschnitten vollzogen werden.

Funktionale Anforderung		Spezifikation / Werkzeug					
		QTI	Authoring / Test-construction Tool	Assessment Delivery System	Item Bank	Learning System	Proprietäre Anforderung
1.1	Unterstützung von Übungen und Tests	x	x	x			
1.2	Hohe Aufgabenvielfalt	x	x	x			
1.3	Parametrisierung / Randomisierung	x	x	x			
1.4	Authentifizierung / Userverwaltung					x	
1.5	Zugriffskontrolle / Gruppenverwaltung					x	
1.6	Mediasupport	x	x	x			
1.7	Sprache			x		x	
2.1	Einfache Erzeugung von Aufgaben	x	x				
2.2	Einfache Erzeugung von Test- und Übungssequenzen	x	x				
2.3	Zeiteinstellungen	x	x				
2.4	Phasendefinition						x
2.5	Niveaudefinition						x
2.6	Hinweise für Aufgabenerfüllung erzeugen	x	x				
2.7	Revisionshandling				x		
3.1	Auswahl von Tests / Übungen durch Kandidaten					x	
3.2	Navigation durch Test- oder Übungssequenzen	x		x			
3.3	Zeitlimit	x		x			
3.4	Hinweise zu Aufgaben	x		x			
3.5	Lösungen zu Aufgaben	x		x			
3.6	Punktevergabe	x		x			
4.1	Automatisierte objektive / nicht automatisierte subjektive Evaluierung von Tests / Übungen	x		x			
4.2	Evaluierung durch Instruktor			x			
4.3	Gruppenbewertung / Kommentarfunktion						x
4.4	Selbstbewertungsbögen						x
5.1	Darstellung Ergebnisse aus Sicht des Kandidaten				x		
5.2	Darstellung Ergebnisse aus Sicht des Instruktors				x		
5.3	Selbstbewertungsbögen anzeigen						x

Tabelle 4.2: Zuordnung der funktionalen Anforderungen auf die Spezifikation und die damit verbundenen Werkzeuge.

4.7.1 Erfüllbarkeit der Anforderung durch die QTI Spezifikation

4.7.1.1 (1.1) Unterstützung von Übungen und Tests

Die Spezifikation QTI bietet ein Datenformat für die Repräsentation von Tests und Übungen. Das Datenformat fasst beide Typen in einer Teststruktur zusammen. Zudem kann diese Struktur in weitere Sektionen sowie Aufgaben untergliedert werden.
Die Unterscheidung, ob Test oder Übung, kann durch die Definition des Autors bestimmt werden. So beinhalten Tests meist ein Zeitlimit (siehe Abschnitt 3.5.7) und eine Punktevergabe (siehe Abschnitt 3.5.5) und (siehe Abschnitt 3.5.8). Übungen hingegen erlauben eine mehrmalige Durchführung (siehe Abschnitt 3.5.6.1) einzelner Aufgaben. QTI unterstützt diese Anforderung durch verschiedene definierbare Attribute.

4.7.1.2 (1.2) Hohe Aufgabenvielfalt

QTI unterstützt eine sehr hohe Anzahl an Interaktionstypen. Die unterschiedlichen Typen können der Tabelle 3.3 entnommen werden. Zudem können Interaktionen auch mehrfach in einem Item in Form von Composite Items definiert werden.
Trotz dieser Kompositionsmöglichkeit konnte jedoch keine Lösung für eine Hotspot Fehlerkorrektur gefunden werden. Ebenso schwierig ist die Erzeugung einer Sprichwortsalat-Aufgabe. Machbar ist dieser Aufgabentyp nur über eine *graphicGapMatchInteraction*. Die Verwendung dieser Interaktion führt aber zu dem bereits diskutierten Vermischungsproblem (siehe Abschnitt 3.8.1.2) zwischen Präsentation und Inhalt.
Ein weiterer Kritikpunkt ist das von IMS QTI [IMS Global Learning Consortium, 2005] vorgeschlagene Design der Aufgaben. Dieser Aspekt kommt zwar bei der Betrachtung des Datenformats noch nicht zum Tragen, zum einen, weil der Fokus von QTI auf der inneren Repräsentation von Assessmentdaten liegt, zum anderen, weil es nur eine Empfehlung an die Implementationen ist. Dennoch bauen die meisten Darstellungen der Implementationen auf dieser Vorgabe auf, wie in Abschnitt 3.9 gezeigt wird.
Nach [Winkelmann, 2005, S. 29] unterliegt die Aufgabenvielfalt neben der zuvor erwähnten Beschränkung auch einer anderen Restriktion. So können zum Beispiel Simulationen, virtuelle Experimente oder andere interaktive multimedial gestaltete Übungsaufgaben nur schwer umgesetzt werden. Auch die Anforderung eines interaktiven Memoryspiels kann nicht erfüllt werden. Die QTI Spezifikation bietet ein sehr durchdachtes und umfassendes Angebot an Assessment Items. Dennoch können nicht alle Anforderung dieses Systems mit diesem Datenformat gedeckt werden.

4.7.1.3 (1.3) Parametrisierung/Randomisierung

Die Spezifikation sieht für die Parametrisierung so genannte Item Templates (siehe Abschnitt 3.5.3) vor. Eine *Delivery Engine* erzeugt zur Laufzeit dynamisch einen Itemklon und stellt diesen dem Kandidaten zur Verfügung. Die geklonten Items unterscheiden sich nur durch die variierenden Template Variablen, die im *Template Processing* gesetzt werden.
Die Randomisierung wird auf zwei Wegen erfüllt. Erstens wird diese Funktionalität für die Erzeugung von Itemklonen verwendet. Zweitens kann durch das Attribut *shuffle* eine Vermischung der Interaktionsinhalte veranlasst werden. So können zum Beispiel die zuvor definierten Auswahlmöglichkeiten einer *choiceInteraction* bei der Darstellung vertauscht werden.

4.7.1.4 (1.6) Mediasupport

QTI unterstützt neben Texten und Grafiken auch Medienobjekte. Diese Objekte können im Content Model, dem Item Body (siehe Abschnitt 3.5.4), definiert werden. Die Spezifikation entspricht somit dieser funktionalen Anforderung.

4.7.1.5 (2.1) & (2.2) Einfache Erzeugung von Aufgaben sowie von Test- und Übungssequenzen

Wie bereits erwähnt, stellt QTI ein umfassendes Informationsmodell in Form einer XML Struktur dar. Was auf den ersten Blick als großer Vorteil erscheint, kann nach längerer Betrachtung zu einer Hürde werden. Die daraus resultierende Komplexität der Spezifikation bereitet vor allem den Itemautoren Schwierigkeiten. Diese Sachgebietsexperten sind meist nur auf ihren Lehrinhalt spezialisiert. So besitzen sie oft nicht die Fähigkeit zur Verwendung dieses Datenformates. Die einzige Zielgruppe wären IT Didaktiker, die mit dieser Materie vertraut sind. Denn die Verwendung dieses Formates setzt viel Fachwissen voraus. Neben dieser Formatkomplexität bringt auch die Verwendung der Ausdrücke und Kontrollstrukturen von QTI Schwierigkeiten mit sich. Die dadurch entstehende *light-weight* Programmiersprache wird schnell unübersichtlich und schwer wartbar. Die Lesbarkeit der Daten kann zwar durch Auslagerung gewisser Verarbeitungsprozesse verbessert werden. Dennoch müssen auch diese Prozesse zumindest einmal erzeugt werden.
Ein weiterer Punkt im Zusammenhang der einfachen Erstellung von Aufgaben ist die Vermischung von Präsentation und Inhalt (siehe Abschnitt 3.8.1.2). Aus diesem Umstand ergeben sich für manche Aufgaben sehr komplexe Datenstrukturen, die natürlich auch sehr schwer wartbar sind.
Diese Anforderung wird von der Spezifikation nicht erfüllt und stellt ein großes Hindernis bei der Verwendung von QTI dar. Diese Problematik wird vor allem von der großen Aufgabenanzahl von 700 Aufgaben unterstrichen.

4.7.1.6 (2.3) & (3.3) Zeiteinstellungen

QTI ermöglicht die Definition von Time Limits (siehe Abschnitt 3.5.7). Diese Zeitvorgaben können, je nach Gebrauch, in verschiedenen Ebenen definiert werden. So kann zum Beispiel ein gesamtes Zeitlimit für den Test wie auch eine exakte zeitliche Begrenzung einzelner Aufgaben vorgenommen werden. Zudem können nicht nur maximale Beschränkungen angegeben werden, sondern auch eine Minimalzeit, in der die Aufgabe gelöst werden muss.
Die Anforderung des Übungssystems wird somit zur Gänze erfüllt, wobei auch nur ein Zeitlimit für einen gesamten Test gefordert wird.

4.7.1.7 (2.6) & (3.4) Hinweise zu Aufgaben

Hinweise werden bei QTI in Form von Feedback (siehe Abschnitt 3.5.9) dargestellt. QTI bezeichnet als Feedback jede Information, die dem Kandidaten aufgrund von Outcome Variablen zur Verfügung gestellt wird. Die Spezifikation unterscheidet zwischen modalem, integriertem und Test Feedback.
Für die Darstellung von Hinweisen in Aufgaben schlägt QTI die Form eines modalen Feedbacks vor [IMS Global Learning Consortium, 2005]. Die Definition eines modalen Feedbacks besagt jedoch, dass die Antwort für die Anzeige eines Hinweises eingereicht werden muss. Nach dieser Einreichung ist keine Modifikation der Antwort mehr möglich. Dieser Vorgang widerspricht aber der Idee eines Hinweises, der zuvor als Hilfestellung für die Auswahl der richtigen Antwort angezeigt werden soll.
Diese Anforderung sollte zwar von QTI erfüllt werden, dennoch konnte der zuvor genannte Widerspruch nicht aufgeklärt werden.

4.7.1.8 (3.2) Navigation durch Test- oder Übungssequenzen

Die Navigation (siehe Abschnitt 3.5.6.1) wird in Form von zwei Modi von QTI gewährleistet. Zum einen kann ein linearer Pfad durch eine Prüfungssequenz definiert werden. In diesem Pfad wird eine Aufgabe nach der anderen in einer definierten Reihenfolge angezeigt. Zum anderen kann eine nicht lineare Abfolge von Aufgaben gewählt werden, wobei der Kandidat selbst zwischen den verschiedenen Assessment

Items wählen kann. Die Definition des Navigationsmodus erfolgt im *testPart* eines Assessments.
Die Anforderung der Navigation wird zur Gänze und auf einem einfachen Weg von QTI erfüllt.

4.7.1.9 (3.5) Lösungen zu Aufgaben

Die Spezifikation ermöglicht durch den *Item Session* Lebenszyklus (siehe Abschnitt 3.5.1.2) und die damit verbundene Klasse *itemSessionControl*, die Repräsentation von verschiedenen Stati eines Assessment Items.
Für die Anzeige von Lösungen kann die *Item Session* in den *solution* Status übergeführt werden. In diesem Zustand werden alle richtigen Lösungen angezeigt, die in der *Response Declaration* definiert wurden.
Die Anzeige von Lösungen wird somit durch die Spezifikation zur Gänze unterstützt.

4.7.1.10 (3.6) Punktevergabe

Punkte werden bei QTI im *Response Processing* (siehe Abschnitt 3.5.5) gesetzt. Über Sequenzen von konditionalen Ausdrücken werden in diesem Prozess die Ergebnisse oder Punkte in Outcome Variablen gespeichert. Diese Regeln nehmen je nach Größe an Komplexität zu. Wie bereits erwähnt wurde, besitzen Itemautoren oft nicht das Fachwissen für die Erzeugung und Wartung dieser Regeln. So kann dieser Verarbeitungsprozess auch ausgelagert werden, um die komplexe Funktionalität vor dem Anwender zu verstecken. Diese Auslagerung verhindert aber eine spezifische Punktegewichtung gewisser Aufgaben, die vom gleichen Aufgabentypus sind.
Nachdem die Outcome Variablen definiert wurden, findet das *Outcome Processing* (siehe Abschnitt 3.5.8) des Tests statt. Dieser summiert in diesem Prozess die Outcome Variablen und liefert ein Gesamtergebnis.
QTI erfüllt diese Anforderung der Punktevergabe. Dennoch darf hier die einhergehende Komplexität nicht außer Acht gelassen werden.

4.7.1.11 (4.1) Objektive/Subjektive Evaluierung von Tests/Übungen

QTI bietet eine Evaluierungsmöglichkeit sowohl für automatisierbare als auch für nicht automatisierbare Aufgaben. Die automatisierbare Bewertung erfolgt im *Response Processing* (siehe Abschnitt 3.5.5), durch die Verwendung von verschiedenen Kontrollstrukturen. Durch dieses Regelwerk können somit Punkte oder andere Informationen als Ergebnisse verarbeitet werden.
Die nicht automatisierbare Bewertung unterstützt QTI durch das Datenformat an sich. Somit wird gewährleistet, dass Informationen zur Aufgabe selbst wie auch zur Benutzerinteraktion an einen Bewerter übermittelt werden. Dieser nimmt dann wiederum Modifikationen an den Daten in Form einer Bewertung vor.
Beide Anforderungen werden durch QTI Spezifikation gewährleistet, auch wenn die objektive Bewertung meist zu einem sehr komplexen Unterfangen wird.

4.7.2 Anforderungsdeckung durch freiverfügbare QTI Werkzeuge

Die Analyse der verschiedenen Werkzeuge, die auf QTI 2.1 basieren und als Open-Source verfügbar sind, wurde bereits im Abschnitt 3.9 durchgeführt. Da das Hauptaugenmerk dieser Arbeit auf der Betrachtung der QTI Spezifikation liegt, werden die daraus gewonnenen Analyseresultate nur noch zusammenfassend den Anforderungen gegenüber gestellt.
Die generelle Unterstützung von Übungen und Tests wird von allen *Delivery Systemen* unterstützt. Im Gegensatz dazu beschränkt sich der Funktionsumfang des Autorenwerkzeuges AQuRate nur auf die Erzeugung von Assessment Items.

Eine hohe Aufgabenvielfalt konnte nur bei der QTIEngine nachgewiesen werden. Alle anderen Werkzeuge ermöglichen nur die Verwendung von Basisaufgabentypen wie Multiple Choice oder Freitextaufgaben.
Die Parametrisierung und die Randomisierung werden von den verschiedenen *Player* Werkzeugen erfüllt. Die Autorenwerkzeuge bieten hingegen nur eine zufällige Anzeige von Antwortmöglichkeiten in Form einer *shuffle* Funktionalität.
Eine generelle Medienunterstützung bieten alle *Player*. Dennoch gibt es eine Restriktion auf Grund der nicht vorhandenen aber angeforderten Aufgabentypen, die somit auch keine Medien unterstützen. Das gleiche gilt für die Autorensysteme, wobei hier die Beschränkung durch eine noch geringere Anzahl an Typen höher ist.
Eine multilinguale Unterstützung konnte nur bei Onyx nachgewiesen werden.
Die einfache Erzeugung von Assessments und deren Items wird zum einen durch das geringe Aufgabentyprepertoire eingeschränkt. Somit muss eine Vielzahl der Items direkt in XML implementiert werden. Zum anderen ist die Funktionalität von QTI so komplex, dass trotz der unterstützenden Abstraktionsstufe von Autorenwerkzeugen ohne ein fachliches Grundwissen eine Erzeugung von Daten nur schwer möglich ist.
Zeiteinstellungen können nur bei Elques durchgeführt werden, da AQuRate keine Testerzeugung unterstützt.
Die Punktedefinition wird bei beiden Erzeugungstools ermöglicht. Auch hier gibt es aber eine Beschränkung, da keine differenzierte Punktevergabe möglich ist.
Die Erstellung von Hinweisen wird von beiden Autorenwerkzeugen nicht gewährleistet. Die Darstellung dieser Hinweise wird jedoch bei *Playern* in Form von Feedback ermöglicht.
Die Navigation durch Test- und Übungssequenzen wird von beiden *Delivery Systemen* umgesetzt.
Ebenso wird die Einhaltung von Zeitlimits von Tests von beiden *Playern* unterstützt.
Eine Darstellung von Lösungen wird von den *Delivery Systemen* nicht angeboten.
Die Evaluierung einer Aufgabe beschränkt sich bei beiden *Playern* auf die automatisierte objektive Bewertung. So bieten sie keine Schnittstelle für die Benotung durch einen Instruktor.

4.8 Konklusion

Zusammenfassend ist festzuhalten, dass die IMS Spezifikation QTI ein wirklich umfassendes Informationsmodell zur Darstellung von Assessments und Assessment Items liefert und somit auch dessen Austausch zwischen verschiedenen Systemen hervorragend ermöglicht. Die Spezifikation setzt dabei sehr auf das Zugpferd „Interoperabilität“. Ist jedoch die Interoperabilität so essentiell, dass wesentliche Anforderungen wie Einfachheit und Übersichtlichkeit beiseite geschoben werden können? Im Falle dieses Übungs- und Testsystems ist dieser Umstand nicht denkbar. So ist zwar die Vielfalt der Aufgabentypologie überzeugend, doch die Erstellung dieser Elemente für Instruktoren nicht tragbar. Diese Problematik steigert sich umso mehr, je weiter die angeforderten Aufgaben von den Basisaufgabentypen abweichen. Genau bei dieser Erstellung sollen Autorenwerkzeuge eingreifen und dabei eine Abstraktionsschicht zum darunter liegenden XML Format bilden. Diese Aufgabe wird aber nur sehr dürftig von den verschiedenen Implementationen wahrgenommen, da entweder der entsprechende Aufgabentyp nicht existiert oder die Konfiguration der Aufgabe trotz Eingabemaske sehr komplex ist.
Auch die Darstellungswerkzeuge sind nur begrenzt für dieses Übungs- und Testsystem einsetzbar, zumal entweder das Rendering zu sehr an die QTI Vorgaben angelehnt ist oder die gewünschte Aufgabentypologie nicht unterstützt wird.
Abschließend kann gesagt werden, dass alle bis jetzt untersuchten Werkzeuge, die QTI 2.1 unterstützen und als Open-Source verfügbar sind, sich noch im Entwicklungsstadium befinden und aus diesem und den oben genannten Gründen nicht einsetzbar sind.
Es wäre zwar wünschenswert, einen DeFacto Standard wie QTI für das angeforderte System zu verwenden, doch infolge der oben genannten Restriktionen ist diese Umsetzung impraktikabel.

Doch welche Möglichkeiten führen aus diesem Dilemma? Das nächste Kapitel befasst sich mit dieser Frage und findet eine Alternative zur Erfüllung der Anforderungen.

Kapitel 5

Konzeption eines Assessmentsystems

„Was man lernen muß, um es zu tun, das lernt man, indem man es tut.“

[Aristoteles (384 - 322), griechischer Philosoph, Begründer d. abendländ. Philosophie]

Ziel dieses Kapitels ist es, zum einen die Grundlage für ein eigenes Assessmentsystem, welches auf einem vereinfachten Datenformat basiert, zu schaffen und zum anderen die Technologien zu untersuchen, die dieses Format unterstützen und durch deren Basis das System implementiert wird.
Die Grundidee am Anfang dieser Arbeit war es, ein System basierend auf (X)HTML und JavaScript im Frontend-Bereich und einer Java-basierten Applikation im Backend-Bereich zu erzeugen. Die Instruktoren hätten dabei über normale Texteditoren die Aufgaben, Übungen und Prüfungen mit Hilfe von verschiedenen Assessment Widgets kreiert. Auf Grund der Komplexität und benötigten Fachkenntnisse regte sich aber schnell Widerstand aus den Reihen der Instruktoren gegenüber dieser Lösung. In Folge dessen wurde in die Konzeption des Systems ein Wikiframework integriert, um dadurch eine einfachere Erzeugung der testspezifischen Daten zu gewährleisten.
Die hierfür verwendeten Technologien und deren Rollen und Aufgaben für dieses System werden im nachfolgenden Teil näher beschrieben. Im Anschluss daran wird eine geeignete JavaScript Bibliothek selektiert, welche den verschiedenen Interaktionsanforderungen der Kandidaten entspricht. Zusätzlich wird auch ein passendes Wikiframework für die Erzeugung, Verwaltung und Durchführung von Assessments gewählt. Als Abschluss wird ein Überblick über die Architektur und die verschiedenen Abläufe und Funktionen im System gegeben.

5.1 Technologische Grundlagen

Um ein Grundverständnis für die verwendeten Technologien zu schaffen, werden diese in den folgenden Abschnitten näher untersucht und beschrieben.

5.1.1 (eXtensible) HyperText Markup Language

Die W3C Standards HyperText Markup Language (HTML) und eXtensible HyperText Markup Language (XHTML) sind Auszeichnungssprachen zur Strukturierung und semantischen Auszeichnung von Inhalten wie Texten, Bildern und anderen unterstützen Medien. Die Syntaxen bringen auch eine Interaktivität in Form von Hypertext Links in Dokumente.
Die Anwendungssprache HTML basiert auf der Standard Generalized Markup Language (SGML), die

auf Grund des enormen Umfanges nicht effektiv einsetzbar ist. XHTML hingegen baut auf einer Teilmenge von SGML auf. Diese Teilmenge mit dem Namen eXtensible Markup Language (XML) ist, ebenso wie SGML, eine formale Markup-Metasprache. In Hinblick auf die Vereinfachung unterstützt diese Sprachgrundlage aber nur eine beschränkte Auswahl an SGML Elementen. Somit besitzt XML eine viel einfachere aber auch striktere Syntax als SGML, was sich somit auch auf die darunter liegenden Anwendungssprachen auswirkt.
Der Fokus von HTML und XHTML liegt auf der Beschreibung von Inhalten, nicht auf deren Erscheinung. So geht es in beiden Sprachen um beschreibende und nicht um darstellungs- oder verfahrensorientierte Textauszeichnung. Zum Beispiel wird der Inhalt zwischen Tags nur als Überschrift gekennzeichnet, die genaue Darstellung, wie Schriftart oder Schriftgröße, wird nicht angegeben.
Die XHTML Version 1.0 bildet die Rekonstitution der aktuellen HTML Version 4.01. Die aktuelle XHTML Version 1.1 verabschiedet sich von den Layout-Tags, die die Präsentation von Dokumenten beeinflussen, und führt zusätzlich einen Mechanismus für die Erweiterung der Sprache in einer standardisierten Weise, in Form von XML Modulen, ein. (Vgl. [Holdener, 2008], [Musciano und Kennedy, 2006] und [Jendryschik, 2008, S. 36ff])

5.1.2 Cascading Style Sheets

Durch den Drang der Webautoren, mehr Einfluss auf die Präsentation von Dokumenten zu haben, entwickelte sich die Strukturs prache HTML in Richtung einer Präsentationssprache. Um diesen Abstraktionsrückschritt zu verhindern, wurde die Style Sheet Sprache Cascading Style Sheets (CSS) geschaffen.
CSS beschreibt Schriften, Farben und Layouts von Dokumenten und adressiert dabei die Funktionalität unterschiedlicher Darstellungen gleicher Inhalte. Somit können nicht nur die Bedürfnisse der Autoren, sondern auch die der Leser berücksichtigt werden. CSS sieht hierfür drei verschiedene Quellen für Style Sheets vor: Autoren, Leser und Browser. Die letztere Quelle, die Browser, beinhalten ein Standard Style Sheet für die Darstellung, wenn kein Style Sheet angegeben wurde. Auch die Kombination verschiedener Style Sheets wird bei CSS ermöglicht. Die daraus resultierenden möglichen Konflikte werden durch CSS verwaltet. Diese Prozedur nennt sich Kaskadierung und besitzt ein genau definiertes Regelwerk, welche Darstellung in einem Konfliktfall verwendet wird.
Neben der Trennung von Präsentation und Inhalt hat CSS auch den Vorteil, dass die Design Definition zentral gespeichert wird. Gewünschte Änderungen können somit an einer zentralen Stelle erfolgen. [Lie, 2005]
Die meisten heutigen Browser unterstützen die Spezifikation Cascading Style Sheets Level 2 (CSS2) Recommendation aus dem Jahr 1998. Diese Version basiert auf CSS1, die zur Gänze von allen Browsern unterstützt wird. Momentan wird an einer Revision von CSS2 gearbeitet, die Erfahrungen aus der alten Version berücksichtigt und Probleme korrigiert. Gleichzeitig wird auch an der Version CSS3 gearbeitet, die nicht nur neue Funktionalität in die Spezifikation mit einbringt, sondern auch eine Modularisierung für CSS vorsieht. [Holdener, 2008]

5.1.3 Document Object Model

Das Document Object Model bildet eine Programmierschnittstelle (API) für externe Programme wie Skriptsprachen. Es ermöglicht dabei dynamisch die Inhalte, Strukturen und Layouts von XML und HTML Dokumenten zu verändern.
Durch einen DOM Parser wird ein (X)HTML Dokument eingelesen und in ein Document Object verwandelt. Dieses Objekt repräsentiert das Dokument als Baumstruktur, in dem alle vorhandenen Elemente aus dem Dokument in Form von hierarchischen Knoten eingeordnet werden. Dazu zählt auch der Text, der von keinen Tags umgeben ist. Das $< html >$ Element bildet hierbei immer das Wurzelelement des Baumes. Ein DOM Baum unterstützt verschiedene Arten von Knoten wie Dokument-, Element-, Attribut- und Textknoten. Der daraus resultierende Nachteil ist, dass ein Dokument immer zur Gänze in

Form des Document Objects im Speicher liegt. (Vgl. [Schneider und Werner, 2001], [Maier, 2006, S. 5] und [Wenz, 2007])
Die durch das World Wide Web Consortium (W3C) standardisierte Schnittstelle basiert auf verschiedenen Modulen wie Core, View, Events, Style und Traversal und Range. Die DOM Level 1 Spezifikation wurde 1998 als erste offizielle W3C Recommendation verabschiedet. Zurzeit ist DOM Level 2 Spezifikation als Recommendation freigegeben. Die Level 3 Spezifikation, die zusätzlich Funktionalität für XML mit einbringt, ist noch nicht zur Gänze freigegeben. Einige Module sind hier noch als Working Draft gekennzeichnet. [Holdener, 2008]

5.1.4 JavaScript

JavaScript ist eine Skriptsprache und wurde Anfang der 90er Jahre von Brendan Eich bei Netscape entwickelt. Der Quellcode dieser Sprache wird nicht kompiliert sondern direkt durch einen Interpreter ausgeführt. Auf Grund der schwachen und dynamischen Typisierung wird die Sprache sehr oft als „einfache“ Skriptsprache aufgefasst. Dennoch ist sie eine umfassende, komplexe und vollwertige Programmiersprache.
Ursprünglich war der Name LiveScript vorgesehen, der sich in letzter Sekunde auf den Namen JavaScript änderte. Nach einer schnellen Entwicklung, wie es jede neue Technologie miterlebt, stabilisierte sich diese und wurde von der European Computer Manufacturers Association (ECMA) standardisiert. Der offizielle Name der standardisierten Version lautet ECMAScript und basiert auf der Spezifikation ECMA Standard 262. Der Name JavaScript bezieht sich eigentlich nur auf die technische Implementation dieses Standards von Mozilla und Netscape. Microsoft hingegen bezeichnet ihre ECMAScript Implementation als JScript. Die meisten Browser heutzutage unterstützen JavaScript 1.5, basierend auf ECMA Standard 262 Edition 3. Firefox und Google Chrome sind in Bezug auf JavaScript mächtiger als der Internet Explorer und unterstützen auch höhere Versionen dieser Sprache. Das Haupteinsatzgebiet von JavaScript ist auf der Seite des Clients. Wie bereits erwähnt, ist ein JavaScript Interpreter in einem Webbrowser integriert und kombiniert somit die Scripting-Fähigkeit mit dem vom Browser definierten Document Object Model (DOM). Durch diese Kombination wird einem statischen Inhalt ein ”Verhalten”verliehen. JavaScript bildet somit das Herz von Dynamic HTML (DHTML), oder auch DOM Scripting genannt, und AJAX. In weiterer Folge können durch diese neugewonnene Dynamik, Rich Client Applikationen erzeugt werden, die das Aussehen von einer Desktopanwendung auf eine Webanwendung übertragen. (Vgl. [Flanagan, 2006] und [Doernhoefer, 2006, S. 16f])
Das Sicherheitsmodell von JavaScript basiert auf dem Sandbox Prinzip und bietet einen recht guten Schutz auf Kosten der Funktionalität. So kann ein Script nicht auf die Daten eines Client-Rechners zugreifen oder diese verändern, mit der Ausnahme von Cookies. [Doernhoefer, 2006, S. 16f]
Die Eigenschaften von JavaScript werden im folgenden kurz zusammengefasst (Vgl. [Schyma, 2007, S. 14] zit. nach [Crane et al., 2006, S. 590f]):

- **Schwache Typisierung:** Bei der Deklaration von Variablen wird kein Typ angegeben. Erst während der Laufzeit wird dieser durch den beinhalteten Wert der Variable bestimmt. Eine Folge daraus ist, dass damit auch Fehler erst während der Laufzeit auftreten und somit unerwünschte Effekte hervorrufen.
- **Dynamische Interpretation des Quellcodes:** Der Quellcode wird als Plain-Text vom Server an den Client übertragen. Dieser wird somit erst zur Laufzeit des Browsers in Maschinencode übersetzt. Eine weitere Funktionalität von JavaScript ist die selbstständige Erzeugung von einem ausführbaren Code zum Beispiel in Form von JSON Objekten.
- **Funktionen sind Objekte:** Funktionen werden in JavaScript als Objekte gehandhabt und sind nicht, wie man es aus objektorientierten Programmiersprachen kennt, an andere Objekte gebunden. Diese Funktionen besitzen ebenso Methoden und Attribute und können zur Laufzeit anderen Objekten als Parameter übergeben werden.

- **Prototype:** Jedes in JavaScript erzeugte Objekt ist mit einem Prototype Objekt verknüpft und kann von diesem Eigenschaften erben. Wird eine Eigenschaft im Prototype Objekt geändert, so ändert sich auch die davon erzeugte Instanz.

5.1.5 eXtensible Markup Language

Die eXtensible Markup Language (XML) ist eine Auszeichnungssprache zur Beschreibung hierarchisch strukturierter Daten. Die Sprache ist eine Teilmenge von SGML und gibt im Gegensatz zu HTML kein Set von Elementen vor. Durch die Verwendung von einer Document Type Definition (DTD) können aber strukturelle und inhaltliche Beschränkungen vorgenommen werden. Somit können neue Sprachen durch die Metasprache XML erzeugt werden.
XML ist selbst dokumentierend, das heißt die Struktur an sich beschreibt bereits die beinhalteten Daten. Dadurch, dass XML im Plain Text Format vorliegt, gibt es keine Beschränkungen bei der Benutzung. Ein Vorteil von XML ist vor allem die Lesbarkeit für Mensch und Maschine, ohne die originale Struktur zu verändern.
Daten im XML Format unterliegen der Regel, dass alle Inhalte in einer hierarchischen Baumstruktur vorliegen. Es kann dabei aber nur einen Wurzelknoten geben. Jedem Start-Tag muss auch ein End-Tag folgen, mit der Ausnahme eines leeren Tags ($< tagName/ >$). Weiters muss jedes XML Dokument den definierten Regeln aus dem DTD Schema genügen und wohldefiniert sein.
XML findet vor allem als Austauschformat zwischen Server und Client Anklang. Konkurrenz erfährt XML in diesem Gebiet aber von JSON, dem eine kompaktere Kodierung der Daten nachgesagt wird. (Vgl. [Holdener, 2008] und [Maier, 2006, S. 7])
Die von W3C ins Leben gerufene XML Spezifikation 1.0 wurde 1998 erstmals als Recommendation verabschiedet und besitzt bis jetzt fünf Versionen. Zur gleichen Zeit der Veröffentlichung der 3. Version, im Jahr 2004, wurde auch die Spezifikation 1.1 publiziert. Diese Erneuerung ermöglicht in manchen Fällen eine einfachere Verwendung von XML. [Holdener, 2008]

5.1.6 eXtensible Stylesheet Language Transformation

Die eXtensible Stylesheet Language Transformation (XSLT) ist eine auf XML basierte Sprache zur Definition von Umwandlungsregeln für die Transformation eines XML Dokumentes. XSLT ist zusammen mit XPath und XSLF eine Untermenge von XSL. Ein solches Stylesheet enthält somit Informationen darüber, wie ein Knoten aus dem Quelldokument im Zieldokument auszusehen hat.
Die definierten Stylesheets werden von einem XSLT Prozessor eingelesen und darauf folgend die XML Dokumente entsprechend den wohldefinierten Regeln transformiert. Bei dieser Umwandlung bleibt das Originaldokument intakt und ein neues Dokument wird erstellt.
Im Jahr 1999 wurde XSLT erstmals vom W3C mit der Version 1.0 freigegeben. Seit 2007 hat die neue Version 2.0 die alte Spezifikation abgelöst. Dennoch unterstützen die meisten Browser bis jetzt nur den alten Standard 1.0. (Vgl. [Holdener, 2008] und [Maier, 2006, S. 8])

5.1.7 JavaScript Object Notation

Die JavaScript Object Notation (JSON) ist ein unabhängiges Format zum Austausch von Daten. Ähnlich wie XML ist auch dieses Format lesbar für Mensch und Maschine.
JSON ist auf zwei Strukturen aufgebaut:

- Eine Kollektion von Namen- und Wertepaaren
- Eine geordnete List von Werten

Die Aussage, dass JSON eine Leichtgewicht-Alternative zu XML sei und vor allem den Vorteil der Reduzierung des Overheads bietet, ist nicht ganz korrekt. Die folgenden Codesnippets zeigen den Unterschied in der Größe. Das XML Codesnippet 5.2 ist um 8 Byte kleiner als die JSON Variante im Code Listing 5.1.

```
{'details': {
    'id': 1,
    'type': 'book',
    'author': 'Anthony T. Holdener III',
    'title': 'Ajax: The Definitive Guide',
    'detail': {
        'pages': 960,
        'extra': 20,
        'isbn': 0596528388,
        'price': {
            'us': 49.99,
            'ca': 49.99
        }
    }
}}
```

Listing 5.1: Ein Beispiel im JSON Format.

```
<details id="1" type="book">
    <author>Anthony T. Holdener III</author>
    <title>Ajax: The Definitive Guide</title>
    <detail>
        <pages extra="20">960</pages>
        <isbn>0596528388</isbn>
        <price us="49.99" ca="49.99" />
    </detail>
</details>
```

Listing 5.2: Ein Beispiel im XML Format.

Tatsache ist aber, dass das JSON Format schneller verarbeitet werden kann und die Definition an sich ein valides JavaScript ist. Dadurch kann eine JSON Definition mit einer `eval()` Funktion direkt in JavaScript Objekt umgewandelt werden. XML hingegen ist ein weiter verbreiteter Standard. Die Wahl zwischen diesen Formaten ist nicht einfach zu beantworten und muss je nach Situation entschieden werden. [Holdener, 2008]

5.1.8 Asynchrones JavaScript und XML

Asynchrones JavaScript und XML (AJAX) ist eine Kombination von verschiedenen, bereits etablierten Technologien, um durch deren gemeinsame Anwendung mehr Dynamik in einer Webseite zu erzeugen. Der Begriff wurde von Jesse James Garret in seinem Aufsatz: „Ajax: A New Approach to Web Applications.“[1] erstmals geprägt und ist seitdem in aller Munde. Garret beschreibt AJAX als die Kombination folgender Technologien:

- (X)HTML und CSS für die standardisierte Präsentation
- DOM für die dynamische Änderung der Präsentation
- XML und XSLT für den Datenaustausch und die Transformation
- XMLHttpRequest für den asynchronen Datenaustausch
- JavaScript für die Logik

Nach [Holdener, 2008] sollte das X in AJAX nur mehr als kleines „x“ dargestellt werden. Der Grund dafür ist die Konkurrenz des zuvor genannten Datenformats JSON.

[1] http://adaptivepath.com/ideas/essays/archives/000385.php

Das klassische synchrone Webapplikationsmodell arbeitet nach dem Prinzip, dass ein Benutzer über ein Interface einen HTTP Request an den Server sendet, dieser die Daten verarbeitet und eine HTML Seite zurück an den Client schickt. Im AJAX-Konzept hingegen wird über JavaScript eine zusätzliche Kommunikation mit dem Server eingerichtet. Diese Kommunikation läuft über das XmlHttpRequest Objekt. Somit werden Daten asynchron im XML oder JSON Format an den Server gesendet, diese Daten werden auf dem Server verarbeitet und wiederum an den Client zurückgeleitet. Das JavaScript auf dem Client empfängt die Daten, parst und manipuliert diese und ändert darauf folgend über das DOM die Präsentation der Seite. Die Abbildung 5.1 zeigt die grafische Gegenüberstellung der beiden Modelle und deren Ablauf.
Der Vorteil, der sich aus diesem AJAX-Konzept ergibt, ist, dass nicht nach jeder Anfrage des Servers die gesamte Seite neu geladen werden muss, sondern nur Teilbereiche in einer Webseite erneuert werden. Datensätze werden sozusagen „on-demand“ geladen. Daraus entsteht der Eindruck einer kontinuierlichen Anwendung, wie man sie im Desktop Bereich kennt. Zudem wird auch die Datenlast um einiges verringert, da nur Inhalte und nicht jedes Mal das Design mit übertragen werden müssen. (Vgl. [Schyma, 2007, S. 16f] und [Doernhoefer, 2006, S. 22]) Der Schwerpunkt von AJAX liegt jedoch nicht auf der asynchronen Kommunikation, diese Technik bildet eher einen Teilaspekt. Der Hauptfokus liegt mehr auf der enormen Interaktivität. Mark Doernhoefer bezeichnet AJAX sogar als neues Gesicht von DHTML. Die daraus entstehenden Rich-Client-Applikationen sind gewöhnlichen Desktop-Applikationen sehr nahe. Die eigentliche Applikation läuft somit zum Großteil am Client ab.
Die Verwendung von AJAX zieht aber auch neue Herausforderungen mit sich. Probleme treten zum Beispiel bei der Browsernavigation auf. Wird eine Seite dynamisch über JavaScript manipuliert, so werden diese Informationen nicht im *history* Objekt des Browsers gespeichert. Als Resultat kann somit nicht die Funktionalität der „Vorwärts- Rückwärtsnavigation“ des Browsers verwendet werden. Zwar ist dieses Problem nach [Stenhouse, 2005] und [Neuberg, 2005] zit. nach [Trattner, 2009, S. 31]) seit geraumer Zeit gelöst, dennoch wird es nicht von allen Webbrowsern unterstützt. Zusätzlich ist auch die Barrierefreiheit von AJAX basierten Webapplikationen nicht gegeben, da nach dem Behindertengleichstellungsgesetzes Webseiten auch JavaScript funktionsfähig sein müssen. Dieser Umstand ist für AJAX natürlich unabdingbar. [Moussaoui und Zeppenfeld, 2008]

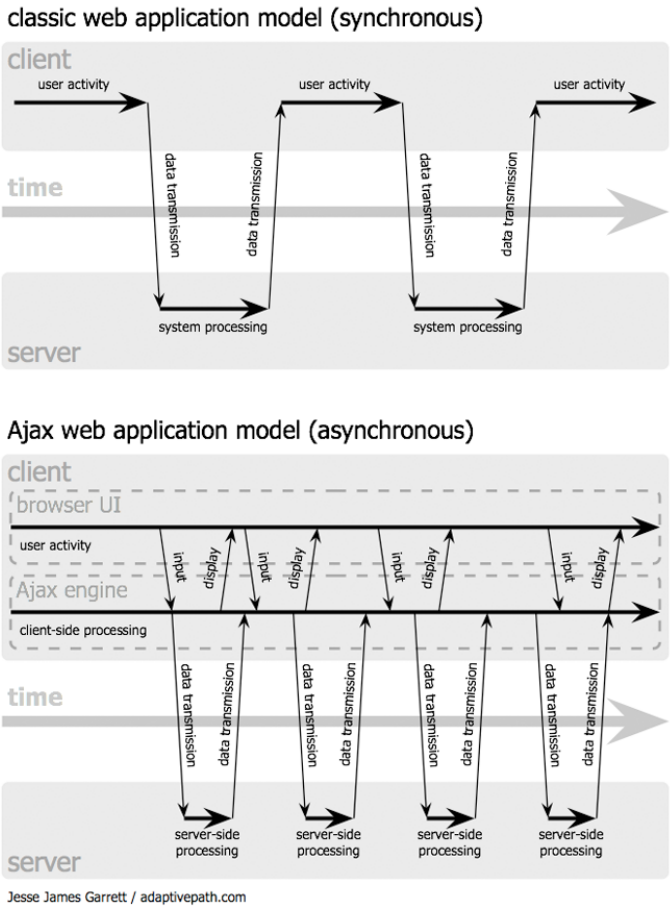


Abbildung 5.1: Das synchrone Interaktionsmuster traditioneller Webapplikationen verglichen zu der asynchronen Kommunikation einer AJAX Anwendung. [Garrett, 2005]

5.2 JavaScript Frameworks/Toolkits

Nach dem Sprichwort: „Das Rad muss nicht immer neu erfunden werden“ ist der Einsatz von JavaScript Frameworks/Toolkits für größere Applikationen unumgänglich. Ein weiterer Punkt ist die Browserkompatibilität, da JavaScript immer an die unterschiedlichen Browsergegebenheiten angepasst werden muss. Der Fokus eines Entwicklers sollte eher auf Business Logik liegen als auf den verschiedenen Browserimplementationen und der daraus resultierenden Komplexität. Ein Framework/Toolkit schafft hierbei Abhilfe, da es eine Abstraktionsstufe in die Entwicklung bringt und zusätzlich immer wiederkehrende Probleme aufgreift. Ein Nachteil darf hierbei aber nicht verschwiegen werden, denn je höher die Abstraktion, desto mehr Overhead wird erzeugt, was sich negativ auf die Performanz auswirkt. Trotzdem überwiegen die Entwicklungsunterstützungen und die dadurch gering gehaltenen Entwicklungskosten.
Die Unterscheidung zwischen einem Framework und einem Toolkit ist nach [Steyer, 2008] nicht standardisiert. Allgemein versteht man unter einem Framework ein Programmiergerüst, das einem Entwickler gewisse Funktionalität zur Verfügung stellt. Mit diesem Rahmen kann ein Entwickler seine eigene Applikation schreiben. Ein Framework beinhaltet ebenso eine Bibliothek mit verschiedenen Codestrukturen. Im Unterschied zu dieser Bibliothek legt ein Framework aber auch ein Verhalten fest, wie man diese Codestrukturen verwendet. Ein Toolkit fokussiert mehr auf die Sammlung von Tools. So stellt ein Web Toolkit zum Beispiel eine Sammlung von Widgets bzw. Komponenten zur Verfügung.
Zurzeit existieren nach der Quelle[2] über 100 verschiedene JavaScript Frameworks/Toolkits. Die nächsten

[2] http://ntt.cc/2008/02/13/the-most-complete-ajax-framework-and-javascript-libraries-list.html

Abschnitte untersuchen eine kleine Auswahl dieser JavaScript Frameworks/Toolkits. Die hierzu verwendeten Quellen sind, wenn nicht anders angegeben, die Hersteller-Websites. Aufbauend auf dieser Untersuchung wird im Anschluss die Wahl eines geeigneten JavaScript Frameworks/Toolkits vorgenommen.

5.2.1 Yahoo! UI Library

Die Yahoo! UI Library (YUI)[3] ist eine JavaScript und CSS Bibliothek von Yahoo und ist als Open-Source unter der BSD Lizenz verfügbar. Die im Jahr 2006 erstmalig veröffentlichte Bibliothek wird von einer nach und nach wachsenden Community getragen. Die Richtungsweisung der Entwicklung wird von Yahoo vorgenommen, was dieses Projekt von anderen Open-Source Projekten unterscheidet. Die derzeitige Version mit vollem Funktionsumfang ist YUI 2 (2.8.0r4), die im Jahr 2009 im September erschienene neue Version YUI 3 stellt bis jetzt nur eine Funktionalität ohne Widgets zur Verfügung.
YUI unterstützt eine Vielzahl an Komponenten (Widgets), auf die Rich Internet Applikationen (RIA) mit ereignisgesteuerten Interaktionen unter Verwendung von AJAX aufbauen. Nach [Steyer, 2008] wandelte sich die Bibliothek zu einem schwergewichtigen JavaScript und CSS Framework. So werden ebenso Rich-Text-Editoren, Image Loader oder Charts unterstützt. Diese Implementationen befinden sich aber teilweise noch im Beta-Stadium.
Neben diesen Komponenten bietet YUI auch eine objektorientierte Programmierung mit Klassen und Vererbung sowie Unterstützung von Animationen, XMLHttpRequest, Drag and Drop, Dom Scripting und Eventhandling.
YUI zeichnet sich auch durch eine sehr gute Dokumentation aus. Ein wesentliches Merkmal von YUI ist die Modularität. So werden beim Laden einer Seite nur diese Teile von YUI übertragen, die auch wirklich benötigt werden. YUI ist somit hoch performant, was sich auch in der geringen Core Größe von 19KB (gezipped) zeigt.
Ein Nachteil von YUI ist die Einschränkung auf A-Klasse-Browser. Laut einer offiziellen Stellungsnahme von Yahoo! gibt es keine leistungsfähige JavaScript Bibliothek, die alle Browser unterstützt. Das stimmt natürlich. Dennoch ist in der Auflistung von Yahoo nicht einmal ein auf Linux basierter Webbrowser sichtbar.

Die YUI Bibliothek teilt sich in verschiedene Bestandteile auf, die in Utilities und Komponenten und in CSS Bibliotheken zusammengefasst werden können [Steyer, 2008]:

5.2.1.1 Utilities und Komponenten

YUI Core: In dieser Gruppe ist der Kern der Bibliothek platziert. Hier ist das Yahoo Global Object zu finden, welches die Basisvoraussetzung für alle anderen YUI Komponenten liefert. Ebenso beinhaltet diese Gruppe die DOM Collection für zusätzliche Funktionalität mit dem DOM. Ebenso ist ein Event Utility für eine Event Normalisierung zu finden.

YUI Library Utilities: Diese Utilities umfassen verschiedene Funktionen für Animationen, Browser-History, Drag and Drop und JSON sowie einen Connection Manager, der bei AJAX Verwendung findet.

YUI Library Controls/Widgets: Wie der Name bereits verrät befinden sich darin die wesentlichen Kernkomponenten für die Erstellung von GUI Komponenten für RIA. So können damit zum Beispiel Buttons, Containers, Menüs und Kalender erzeugt werden.

[3] http://developer.yahoo.com/yui

YUI Developer Tools: Diese Sammlung beinhaltet Werkzeuge zum Analysieren und Testen von RIA. Darin enthalten sind zum Beispiel ein Logger Control, ein Profiler sowie das YUI Test Utility.

5.2.1.2 CSS Bibliotheken

YUI Library CSS Tools: Diese Werkzeuge haben ebenso eine wichtige Bedeutung in YUI. So kann zum Beispiel durch *reset.css* die browsereigenen Einstellungen, wie individuelle Paddings, komplett ausgeblendet werden. Das Werkzeug *font.css* hingegen setzt alle Schriften auf Standardwerte und die *base.css* nimmt gewisse Grundformatierungen vor.

5.2.2 Dojo Toolkit

Dojo[4] ist ein von Alex Russel ins Leben gerufenes JavaScript Toolkit, das von der Dojo Foundation und einer großen freien Community vorangetrieben wird. Das modular aufgebaute Open-Source Toolkit wurde im Jahr 2005 zum ersten Mal veröffentlicht und ist unter der Academic Free License 2.1 und der BSD Lizenz verfügbar. Die derzeitige Version 1.3.2 wurde im Juli released.
Dojo fokussiert sehr auf seine enorme Widget Bibliothek und beinhaltet eine große Anzahl an Funktionen für das DOM Scripting. Das Toolkit zeichnet sich dabei durch die Internationalisierung und Barrierefreiheit seiner Komponenten aus. Dazu bietet Dojo auch die Möglichkeit über verschiedene vor implementierte Themes das „Look and Feel“ der Seite zu beeinflussen.
Ebenso unterstützt Dojo eine sehr durchdachte Drag and Drop Architektur, wie auch ein Back Button Handling. Neben diesen dynamischen Aspekten einer Website, bietet Dojo auch eine Abstraktionsstufe für die Verwendung des XMLHttpRequest an und kapselt damit verschiedene Browser Eigenheiten. Das Event System, basierend auf dem Publisher-Subscriber Pattern, und die I/O APIs machen Dojo zu einer mächtigen Programmierumgebung.
Die Dojo Toolkit Architektur besteht, wie in Abbildung 5.2 zu sehen ist, aus mehreren Komponenten und kann durch eigene Widget Implementation erweitert werden:

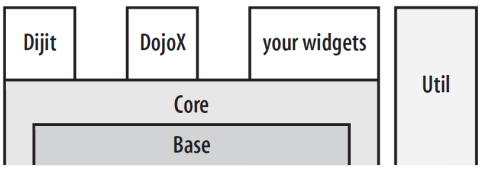


Abbildung 5.2: Die Komponentenarchitektur vom Dojo Toolkit. [Russell, 2008, S. 3]

5.2.2.1 Komponenten Base und Core

Die Komponente Base besteht lediglich aus der *dojo.js* Datei. Dieser Kernel beinhaltet in einer ultrakompakten Form, die wichtigsten Features von Dojo. Zu diesen Funktionalitäten gehören die CSS-basierten Queries, Event Handling, AJAX oder klassenbasierte Programmierunterstützung. Die Core Komponente baut auf dem Kernel auf und stellt Funktionalitäten wie erweiterte Animationen, I/O und Drag and Drop zur Verfügung.

5.2.2.2 Komponente Dijit

Dijit repräsentiert eine große UI Bibliothek. Die Widgets von Dijit ermöglichen die Verwendung von Themes und Accesibility Standards und sind einfach zu verwenden. Ebenso besteht die Möglichkeit, eigene Widgets zu erzeugen.

[4] http://www.dojotoolkit.org

5.2.2.3 Komponente DojoX

DojoX ist eine Sammlung von Subprojekten, die teilweise noch in einer experimentellen Phase sind. Darunter befinden sich Möglichkeiten für Grid-Widgets, bessere Animationen und Diagramme.

5.2.2.4 Komponente Util

Util beinhaltet Werkzeuge wie ein Unit-Test Framework oder verschiedene Build Tools.

Durch die Dojo Build-Tools wird die Möglichkeit geschaffen, den Code zu optimieren und Unit Tests durchzuführen. Das Toolkit unterstützt alle großen Browser, inklusive Linux Browser, und stellt eine gute API Dokumentation zur Verfügung.
Nachteilig ist aber, im Gegensatz zu YUI, der etwas größere Core von 27KB (gezipped).

5.2.3 Prototype + Scriptaculous

Das **Prototype**[5] JavaScript Framework wurde erstmals 2005 von Sam Stephenson ins Leben gerufen und bildet die Grundlage für Ruby on Rails. Der Name dieses Frameworks hängt mit der Prototyping Technologie von JavaScript zusammen. Das Framework, mit der derzeitigen Version 1.6.1, ist als Open-Source unter der MIT Lizenz verfügbar.
Der Fokus von Prototype liegt auf der effektiven Manipulation des DOM. Die Fähigkeit dieses Frameworks ist somit mit dem Core von Dojo vergleichbar. Prototype stellt für diese Aufgaben verschiedene Shortcut Funktionen zur Verfügung. So wird eine Standardfunktion wie `document.getElementById()` durch `$()` abgekürzt und mit zusätzlichen Features bestückt.
Das Framework kann in drei Sektionen unterteilt werden. Die erste Sektion ist die zuvor beschriebene DOM Sektion, die zweite ist die Sprach Sektion und die dritte bildet die AJAX Sektion.
In der Sprach Sektion sind viele Erweiterung zu finden, die die Sprache JavaScript erweitern. So werden zum Beispiel Ergänzungen für die eingebauten Typen, wie Number oder String, vorgenommen. Zudem ermöglicht Prototype, ähnlich den anderen Frameworks, auch eine objektorientierte Programmierung.
In der AJAX Sektion ist eine Wrapper API für XMLHttpRequest Objekt zu finden.
Die Abstraktionsstufe von Prototype ist im Gegensatz zu anderen Frameworks eher niedrig gehalten, deshalb ist ein fundiertes JavaScript und CSS nötig. Der vergleichsweise niedrige Abstraktionslevel zum großen Funktionsumfang ist aber auch der Grund, warum Prototype die Grundlage für verschiedene Frameworks bildet.
Die API Dokumentation ist vollständig und verfügt über eine Suchfunktion. [Steyer, 2008]

Scriptaculous[6] oder script.aculo.us ist ein Open-Source JavaScript Framework mit der MIT Lizenz und setzt auf den zuvor genannten Framework Prototype auf. Das Framework entwickelte Thomas Fuchs und wurde im Jahr 2005 erstmals öffentlich vorgestellt. Zurzeit ist die Version 1.8.3 die aktuellste.
Scriptaculous erweitert das darunter liegende Framework mit visuellen Effekten und interaktiven Funktionen. Es unterstützt eine Visual Effect Engine, eine Drag and Drop Bibliothek, verschiedene Controls, wie AJAX basierte Autovervollständigung oder Slider und Unit-Testing.
Nachteilig ist, dass Scriptaculous keine Widgets zur Verfügung stellt, sondern nur Helper Funktionen und visuelle Effekte. [Steyer, 2008]

[5] http://www.prototypejs.org
[6] http://script.aculo.us

5.2.4 MochiKit

MochiKit[7] ist eine *light-weight* JavaScript Bibliothek und unterliegt dem Motto: Javascript einfacher verwendbar zu machen. Die von Bob Ippolito entwickelte Bibliothek wurde im Jahr 2005 released und ist zurzeit in der Version 1.4.2 erhältlich. MochiKit ist unter der MIT Lizenz sowie der Academic Free License 2.1 verfügbar.
Die Bibliothek umfasst viele nützliche Hilfsfunktionen und erinnert dabei an die funktionale Sprache Lisp. MochiKit gliedert diese Funktionalitäten in verschiedene Pakete. So wird hier zum Beispiel zwischen dem DOM, Drag and Drop, Color, Iter, Logging, Base Paket und vielen mehr unterschieden. Ebenso stellt die Bibliothek eine Wrapper API für die Verwendung vom XMLHttpRequest Objekt zur Verfügung. Die Bibliothek importiert teilweise visuelle Effekte von Scriptaculous.
Trotz der Aufhängung von MochiKit auf der guten Dokumentation fällt diese aus Sicht des Autors eher mittelmäßig aus. Ein weiterer Nachteil ist, dass das letzte Release im November 2008 ausgegeben wurde. Daraus kann geschlossen werden, dass keine große Community dahinter steht. MochiKit stellt ebenso keine fertigen Widgets zur Verfügung. [Jäger, 2009, 318]

5.2.5 Anforderungsanalyse und Wahl eines geeigneten Frameworks/Toolkits

Das Aufgabengebiet von JavaScript ist im Übungs- und Testsystem die Gewährleistung von Interaktivität für den Benutzer. Für diese Aufgabe verwendet JavaScript die Funktion der dynamischen DOM Manipulation. Dabei beinhaltet es die clientseitige Logikschicht und bildet aus dem System eine Rich Internet Application. Für die Kommunikation zwischen dem Client und dem Server wird, soweit es möglich ist, in Hinblick auf einen kontinuierlichen Applikationsablaufes das AJAX Konzept verwendet.

Welche Anforderungen können nach dieser Positionierung von JavaScript identifiziert werden?

5.2.5.1 Hohe Anzahl an Widgets

Für das System ist eine Unterstützung von verschiedenen vorimplementierten Widgets essentiell. Dabei werden Komponenten wie Buttons, CheckBoxes, Tooltips, Tooltip Dialogs, Texteditoren, Inline Editboxes für die Darstellung von Aufgabenstellungen und Datagrids oder Charts für die Präsentation der Ergebnisse verwendet.

5.2.5.2 Eigene Widgets erzeugen

Für die Aufgabenstellung werden teilweise sehr spezielle Komponenten benötigt, die mehrmals in verschiedenen Übungen wieder verwendet werden. Aus diesem Grund wird eine einfache Widget Herstellung sowie eine simple Einbindung dieser Komponenten benötigt.

5.2.5.3 Drag and Drop

Viele Aufgabenstellungen basieren auf der Zuordnung von Objekten durch den Benutzer. Deshalb muss das JavaScript Framework/Toolkit Drag and Drop im Webbrowser unterstützen. Dabei ist eine durchdachte Architekur wünschenswert, um die Komplexität der verschiedenen Zuordnungsimplementationen gering zu halten.

[7] http://www.mochikit.com

5.2.5.4 Internationalisierung

Die Zielgruppe der Übungen und Tests besitzt unterschiedliche Sprachkenntnisse. Somit soll das JavaScript Framework/Toolkit die Möglichkeit bieten, eine Internationalisierung der Bedienungselemente vorzunehmen.

5.2.5.5 Hohe Abstraktionsstufe

Das Framework/Toolkit soll einen hohen Abstraktionsgrad einführen, um die verschiedenen Browserimplementationen zu verstecken und somit eine einfache Entwicklung zu gewährleisten.

5.2.5.6 Gute Dokumentation

Um die Entwicklungszeit gering zu halten, ist eine gute Dokumentation erforderlich. Ebenso wichtig ist die Verfügbarkeit verschiedener Bücher und Tutorials sowie die Unterstützung durch eine breite Community.

5.2.5.7 Open-Source

Aus Kosten und Erweiterungsgründen ist eine Anforderung an das Framework/Toolkit, dass es als Open-Source verfügbar ist.

5.2.5.8 Browserkompatibilität

Da die Browserverwendung der Zielgruppe nicht eingegrenzt wurde, soll das System durch einen Großteil der aktuellen Webbrowser unterstützt werden.

5.2.5.9 Framework-/Toolkit Wahl

Durch die Anforderung einer verfügbaren Widget Bibliothek scheiden die *light-weight* Bibliothek MochiKit sowie das Framework Scriptacoulus aus der Bewertung aus. Beide zielen nur auf die Gewährleistung von visuellen Effekten ab und stellen somit keine eigenen Widgets zur Verfügung. Auch das Prototype Framework bietet nur eine Erweiterung zu JavaScript, ähnlich der Dojo Core Funktionalität. Dojo und YUI liefern eine große Widget Bibliothek, worin alle benötigten Basiskomponenten enthalten sind. Das Dojo Toolkit stellt jedoch im Gegensatz zu YUI eine größere Auswahl zur Verfügung.
Das Erstellen von neuen Widgets wird vor allem im Dojo Toolkit erleichtert. Hierbei werden sowohl die Erstellung von komplett neuen Komponenten wie auch die Komposition bereits bestehender Elemente gefördert. Zudem ermöglicht Dojo auch die Einbindung dieser Widgets in den Markup Code einer (X)HTML Seite, was die Komplexität in der Verwendung sehr verringert. YUI stellt auch die Funktionalität der Erzeugung eigener Widgets bereit[8]. Im Gegensatz zu Dojo fokusiert YUI aber nicht so sehr auf diese Erzeugung. Weiters bietet YUI auch keine Markup Einbindung dieser Komponenten an.
Dojo wie auch YUI stellen eine sehr gute Drag and Drop Architektur zur Verfügung. Beide verwenden hierzu ein Set von erweiterbaren Klassen, die von einem Manager Objekt mediiert werden. Weiters bieten beide die Verwendung von Drag and Drop Events an.
Das Dojo Toolkit bietet als einziger ein Internationalisierungsmodul (I18N) an. Durch dieses Modul können sprachliche Anpassungen ohne Quellcodeänderung erfolgen.
Die YUI Library wie auch das Dojo Toolkit stellen dem Entwickler eine hohe Abstraktionsstufe zur Verfügung. Dadurch können browserabhängige Unterschiede abstrahiert und vernachlässigt werden. Zu dieser Abstraktion zählt im Framwork und Toolkit vor allem die Kapselung des XMLHttpRequest sowie

[8] http://yuiblog.com/blog/2008/06/24/buildingwidgets

des darunter liegenden Event Systems der verschiedenen Browser.
Laut [Schyma, 2007, S. 25] und [Crandall, 2006] besitzt das Dojo Toolkit eine schlechte Dokumentation. YUI hingegen liefert eine sehr gute Dokumentation. Aus Sicht des Autors trifft diese Aussage für Dojo nicht zu, da unter der Website[9] eine große Auswahl an Dokumentationen und Referenzen erhältlich ist. Zudem sind auch eine Vielzahl an Tutorials im Web verfügbar.
Dojo wie auch YUI sind als Open-Source unter der BSD Lizenz verfügbar (Dojo auch unter AFL). Anzumerken ist jedoch bei YUI, dass im Gegensatz zu Dojo nicht eine freie Community das Projekt trägt und erweitert, sondern das Unternehmen Yahoo!.
Die Browserkompatibilität zu den größten Browsern wird durch Dojo und YUI erreicht. Wie bereits angemerkt wurde, wird jedoch der Linux basierte Webbrowser Konqueror von YUI nicht unterstützt.
Der Geschwindigkeitsaspekt, wie schnell eine Seite geladen wird, wurde in den Anforderungen nicht erwähnt. Dennoch ist zu beachten, dass aufgrund der Mächtigkeit beider Bibliotheken auch die Performanz darunter leidet. YUI bietet durch seine neue Version im Gegensatz zu Dojo einen besonders performanten Core an, der diese Problematik erfolgreich aufgreift. Auch die ältere Version YUI 2 besitzt im Vergleich zu Dojo einen kleineren Kernel. Doch nicht nur die Kerngröße, sondern auch die Verarbeitungszeiten von verschiedenen Aufgaben sind entscheidend. So konnte Dojo bei unterschiedlichen Funktionen gegenüber YUI punkten (siehe Anhang C.1).

Abbildung 5.3: Eine Trendanalyse der Begriffe „dojo javascript“ und „yui javascript“ von Google Trends

Das Dojo Toolkit und die YUI Library weisen eine ähnliche Mächtigkeit auf. YUI überzeugt vor allem durch seine bestechende Dokumentation und Performanz. Dojo hingegen überwiegt bei der Erzeugung eigener Komponenten sowie durch seine große Widget Bibliothek. Auch die sprachliche Anpassung seiner Komponenten zeichnet das Dojo Toolkit aus. Die Trendanalyse 5.3 von Google Trends[10] zeigt eine nahezu identische Beliebtheit. Aus den oben genannten Stärken beider Bibliotheken überwiegen für diese Aufgabenstellung jedoch die des Dojo Toolkits.

5.2.6 Tieferer Einblick in das Dojo Toolkit

Nachdem eine geeignete JavaScript Bibliothek gefunden wurde, wird noch näher auf die Besonderheiten der einzelnen Funktionalitäten und Komponenten eingegangen. Zunächst wird hierzu die AJAX Kommunikation näher untersucht und die Wrapper API dargestellt. Danach wird die Drag and Drop Architektur und Funktionaltität besprochen. Zum Abschluss werden die verschiedenen Widgets der Dijit Bibliothek aufgezeigt und deren Lifecycle und Anatomie dargestellt.

[9] http://www.dojotoolkit.org/docs
[10] http://www.google.at/trends

5.2.6.1 AJAX Kommunikation

Webseiten holen sich heutzutage die Inhalte über asynchrone Anfragen vom Server. Die retour übermittelten Daten werden über sogenannte Callback Funktionen weiterverarbeitet.
Um diese Kommunikation zu unterstützen, bietet Dojo im Base Modul eine Sammlung von Funktionen an, die in einer REST Architektur[11] eingesetzt werden können. Die REST Funktionen (GET, POST, PUT, DELETE) werden hier aufgelistet:

`dojo.xhrGet(/*Object*/ args)`
Mit dieser Funktion fordert der Client über das XMLHttpRequest Objekt Daten vom Server an.

`dojo.xhrPost(/*Object*/ args)`
Hiermit werden die Daten oder Ressourcen vom Client auf dem Server abgelegt.

`dojo.xhrPut(/*Object*/ args)`
Diese Funktion aktualisert bereits vorhanden Daten.

`dojo.xhrDelete(/*Object*/ args)`
Mit dieser Funktion löscht der Client Daten am Server.

`dojo.xhr(/*String*/ method, /*Object*/ args, /*Boolean?*/ hasBody)`
Diese Funktion ist eine allgemeine XHR Funktion um eine beliebige HTTP Methode asynchron auszuführen. Alle zuvor beschriebenen Funktionen sind Wrapper dieser Funktion.

Das `args` Objekt beinhaltet die Konfiguration für die Anfrage. Die wichtigsten Elemente sind nachfolgend beschrieben:

- `url` - die URL um die Anfrage zu lenken.
- `content` - der Inhalt in Form von Key/Value Paaren.
- `form` - der DOM Knoten, der den Inhalt zur Verfügung stellt.
- `timeout` - die Wartezeit für eine Antwort. Nach Ablauf dieser Zeit wird die Fehlerroutine aufgerufen.
- `handleAs` - das Format, in dem die Antwort retouniert wird (JSON, XML).
- `load` - die Funktion, die nach erfolgreicher Antwort aufgerufen wird.
- `error` - die Funktion für den Aufruf nach einem Fehler.

Ein möglicher XHR GET Aufruf könnte nach [Russell, 2008, S. 86] wie im Code Listing 5.3 aussehen.

```
1  //...snip...
2  dojo.addOnLoad(function( ) {
3    dojo.xhrGet({
4      url : "someText.html", //the relative URL
5      // Run this function if the request is successful
6      load : function(response, ioArgs) {
7        console.log("successful xhrGet", response, ioArgs);
8        //Set some element's content...
9        dojo.byId("foo").innerHTML= response;
10       return response; //always return the response back
11     },
12     // Run this function if the request is not successful
13     error : function(response, ioArgs) {
14       console.log("failed xhrGet", response, ioArgs);
```

[11] Ressource zentrierte Architektur

```
15        /* handle the error... */
16        return response; //always return the response back
17      }
18    });
19  });
20  //...snip...
```

Listing 5.3: Ein möglicher XHR GET Aufruf im Dojo Toolkit.

Wird für das `load` oder `error` Argument eine externe Funktion verwendet, so tritt hier ein Fehler auf. Der Grund dafür ist, dass bei einem asynchronen Callback der Scope wechselt und somit die externe Funktion nicht mehr erkannt wird. Doch auch hier bietet Dojo eine Lösung an. Die Funktion `dojo.hitch` garantiert, dass eine Funktion in einem entsprechenden Kontext ausgeführt wird. Die Verwendung dieser Funktion wird im Code Listing 5.4 nach [Dojo Toolkit Community] näher veranschaulicht.

```
1  var myObj = {
2      foo: "bar"
3  };
4  var func = dojo.hitch(myObj, function() {
5      console.log(this.foo);
6  });
7  func();
```

Listing 5.4: Verwendung der Funktion `dojo.hitch` Funktion.

Bei der Verwendung von asynchronen Callbacks entsteht bereits nach einer geringen Anzahl dieser Funktionen, eine hohe Komplexität. Dojo schafft durch sogenannte `dojo.deferred` Objekte Abhilfe (Vgl. [Dojo Foundation] und [Russell, 2008, S. 89ff]). Diese Objekte bieten einen Weg zur Abstrahierung von nicht-blockierenden Ereignissen. Hierfür stellen *deferred* Objekte eine Antwort zu einem zukünftigen Zeitpunkt in Aussicht und bieten dabei eine einfache Möglichkeit, sich zu registrieren um diese Antwort zu erhalten. Dafür werden folgende Methoden durch die Klasse *Deferred* zur Verfügung gestellt:

- `addCallback(handler)` - fügt einen Callback Handler hinzu.
- `addErrback(handler)` - fügt einen Errorback Handler hinzu.
- `callback(result)` - wird aufgerufen, sobald ein Callback stattfindet.
- `errback(result)` - wird aufgerufen, wenn ein Fehler in einem Callback auftritt.

Deferreds unterstützen auch eine Verkettung verschiedener Callbacks, wie auch Errorback Handler. So könnte ein möglicher Ablauf der verschiedenen Callbacks wie in Abbildung 5.4 aussehen:

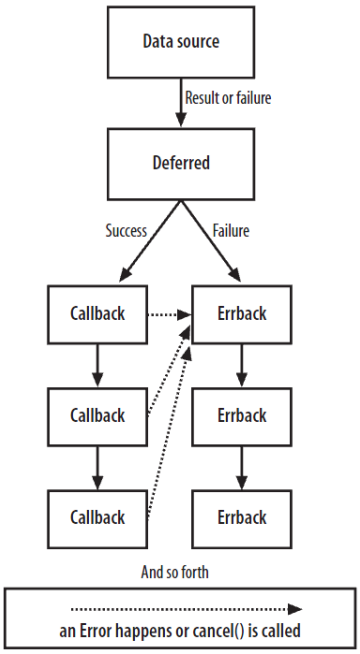


Abbildung 5.4: Ein beispielhafter Eventablauf unter der Verwendung von *Deferred*. [Russell, 2008, S. 95]

Dojo verwendet diese *Deferreds* auch für die XHR Funktionen. Wird eine solche Funktion erzeugt, so wird automatisch ein *Deferred* Objekt zurückgeliefert. Diesem Objekt können im Anschluss verschiedene Callback Handler angefügt werden. Sobald die Daten asynchron durch den Server zurückgesendet werden, wird die Callback Funktion des `load` Argumentes aufgerufen. Im Anschluss dieses Aufrufes werden alle Callback Handler der Reihe nach informiert. Dieser beschriebene Ablauf trifft speziell für eine fehlerfreie Abhandlung zu.

5.2.6.2 Drag and Drop

Durch Drag and Drop erhält eine webbasierte Applikation das Feeling einer Desktop Applikation. Dojo bietet für diese Anforderung ein sehr durchdachtes Modul. In diesem Modul enthalten sind unter anderem die sechs wichtigsten Klassen: *Container*, *Selector*, *Source*, *Manager*, *Avatar*, *Moveable*.
Die Klassen *Container*, *Selector* und *Source* sind für die Quelle und das Zielobjekt zuständig. *Manager* und *Avatar* sind Singleton Klassen und orchestrieren das Drag and Drop auf der Seite.
Die Klasse *Moveable* verleiht einem DOM Knoten die Eigenschaft bewegbar zu sein.

Container - bildet eine einheitliche lineare Sammlung von Items. Ein Item ist ein abstraktes Datenobjekt und kann alles Mögliche beinhalten. Wird ein *Container* angelegt, so wird auch eine *Creator* Funktion mit übergeben. Diese Funktion wird aufgerufen, sobald ein Item visualisiert wird. Zudem gibt es auch die Möglichkeit, zwischen der Präsentation in einem *Container* und der innerhalb eines *Avatars* zu differieren. Neben den unterschiedlichen Erzeugungsfunktionen wird auch der Typ des beinhalteten Items angegeben. Ein *Container* kann somit nur jene Items halten, die seinem Typ entsprechen. Zusätzlich besitzt ein *Container* auch das Wissen, wann die Maus des Benutzers über ihn selbst oder ein beinhaltetes Item wandert.

Selector - baut auf dem *Container* auf und fügt die Fähigkeit hinzu, beinhaltete Items zu selektie-

ren. Dabei können nicht nur einzelne Elemente ausgewählt werden, sondern auch eine Liste von Items. Zusätzlich erweitert er den *Container* mit einer Anker-Funktion, die es ermöglicht, Items an einem speziellen Platz einzufügen.

Source - wird von der Klasse *Container* abgeleitet und erweitert die beiden darunter liegenden Klassen durch die Fähigkeit, Drag and Drop Operationen zu starten und an der Orchestrierung teilzunehmen. Ein Benutzer kann durch ein instanziiertes Objekt dieser Klasse eine Drag Operation starten und beinhaltete Elemente von einer Quelle zu einem Ziel ziehen, oder auch nur die Reihenfolge der Präsentation dieser Elemente in einer Quelle ändern.

Manager - implementiert die Business Logik und übernimmt die Orchestrierung der Visualisierung von Drag and Drop. Dabei akzeptiert der *Manager* die Events eines *Source* Objektes, erzeugt den *Avatar* und validiert das Droppen eines Objektes.

Avatar - ist eine Singleton Klasse, die das Element während einer Drag Operation repräsentiert.

Moveable - verleiht einen DOM Knoten die Fähigkeit, bewegbar zu sein. Hierbei können verschiedene Einstellungen konfiguriert werden. Zum Beispiel ist es möglich, eine Verzögerung zu definieren, die eine Bewegung erst nach einer bestimmten Distanz des Mauszeigers startet.

Zu diesen Klassen werden auch eine Fülle an Events angeboten, die entweder über `dojo.subscribe` oder `dojo.connect` abonniert werden können. Für diesen Abschnitt wurde die Quelle [Dojo Toolkit Community] verwendet.

5.2.6.3 Dijit

Die Widget Bibliothek zeigt einen solchen Erfolg, dass von den meisten Benutzern Dojo als Synonym für diese Sammlung von Komponenten gesehen wird. Dennoch ist Dijit ein Subprojekt des Toolkits und wird vom Kern getrennt verwaltet. Der Name Dijit stammt von „Dojo Widget“ und bezeichnet zum einen das Projekt selbst und zum anderen die darin enthaltenen Widgets.
Eine Besonderheit, die Dojo ermöglicht, ist die Einbindung von Widgets in einem HTML Dokument über Markup Code. Ein Entwickler platziert hierzu die Komponente über einen `dojoType` Tag in den entsprechenden HTML Knoten. Sobald die Seite geladen wird, ersetzt der Parser von Dojo diesen Tag und transplantiert ein dynamisches DHTML Widget in die Seite.
Dojo stellt für Widgets die Verwendung von Themes zur Verfügung. Diese Themes sind eine konsistente Sammlung von CSS Regeln, die auf die darunter liegenden Komponenten angewendet werden können. Das Toolkit bietet hierzu drei verschiedene Themes: Tundra, Soria und Nihilio, die auch selbst erweitert werden können.
Wie bereits erwähnt, ist die Komponenten-Bibliothek sehr umfangreich. Um einen Überblick zu schaffen, kategorisiert Dijit die Widgets in drei Hauptgruppen: Formular Widgets, Layout Widgets und Applikations-Widgets. Die Formular Widgets wie *Button*, *ComboBox*, *Slider* oder *TextBox*, sind vor allem für die Benutzung in Formularen geschaffen. Die Layout Widgets wie *ContentPane*, *Tree*, *TabContainer* oder *AccordionContainer* ermöglichen komplexe Layouts und kapseln dabei aufwendige CSS Konfigurationen. Die Applikations-Widgets wie *Menu*, *Tooltip* oder *TooltipDialog* sind häufig verwendete Komponenten in RIAs. [Russell, 2008]

Die Anatomie eines Dijit

Ähnlich wie „Standalone“-Klassen sind Widgets eigenständig und werden innerhalb eines Ordners, der

den Namensraum bestimmt, gespeichert. In diesem Ordner befinden sich auch die restlichen Ressourcen, wie das Template und das Theme. Die Abbildung 5.5 nach [Russell, 2008, S. 274] zeigt einen Überblick dieser Anatomie:

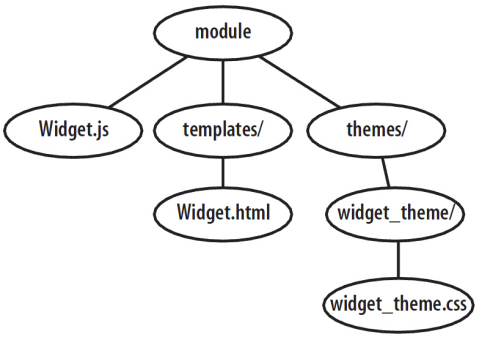


Abbildung 5.5: Anatomie eines Dijits auf der Festplatte. [Russell, 2008, S. 274]

Das JavaScript Widget.js beinhaltet die Logik der Komponenten. Im HTML Dokument *Widget.html* wird hingegen die Struktur der Komponente definiert, was eine gute Trennung zwischen Strukturierung und Logik bringt. Das *widget_theme.css* übernimmt die Style Definitionen betreffend der Farben und Layouts. [Russell, 2008]

Der Lifecycle eines Dijit

Dijit lehnt seine Lebenszyklus Events gewissen Mustern aus der objektorientierten Welt an. Jedes Dijit wird hierfür von der Klasse `dijit._Widget` abgeleitet. Diese Superklasse packt verschiedene Lebenszyklus-Methoden in ein Dijit, die je nach Zeitpunkt der Erstellung oder Zerstörung aufgerufen werden. In der folgenden Auflistung nach [Russell, 2008] wird ein Überblick über die wichtigsten Methoden gegeben:

- `preamble` - Die Preamble Methode ermöglicht eine Manipulation von Konstruktor Argumenten bevor sie diesen erreichen.
- `constructor` - Diese Methode wird vor dem „Mixin“ der Parameter aufgerufen und kann für Array Initialisierungen verwendet werden.
- `postMixInProperties` - Nachdem Dojo die Ableitungshierachie durchlaufen hat und alle entsprechenden Eigenschaften in die Klasse „gemixt“ wurden, wird diese Methode aufgerufen. Dies geschieht aber noch vor dem Rendering des Widgets.
- `buildRendering` - Diese Methode wird durch die Klasse `dijit._Templated` implementiert und ist für das Rendering der Komponente zuständig.
- `postCreate` - Diese Methode wird aufgerufen, nachdem das Dijit erzeugt und sichtbar in der Seite platziert wurde. Anzumerken ist hierbei, dass enthaltene Kind Widgets noch nicht verfügbar sind.
- `startup` - Die Startup Methode wird aufgerufen, nachdem alle Child Widgets erzeugt wurden.
- `destroy` - Sobald eine Komponente zerstört wird, wird diese Methode aufgerufen. Hierin können spezielle Aufräumarbeiten vorgenommen werden.

5.3 Wiki

Der ursprüngliche Gedanke des Übungs- und Testsystems war, über einen Texteditor vorgefertigte Aufgaben, Übungs- und Testkomponenten in (X)HTML Seiten einzufügen. Doch wie bereits erwähnt, setzte diese Arbeitsweise eine gewisse Technologieerfahrung und -vertrautheit voraus. Durch die fehlenden Fachkenntnisse wurde somit nach einer neuen Lösung gesucht. Abhilfe für dieses Problem wurde im Wiki-Konzept gefunden, das es ermöglicht, eine eigene einfache Syntax zu verwenden und Inhalte zu publizieren. Doch was ist ein Wiki? Die nachfolgenden Abschnitte beschreiben diese Technologie näher und gehen dabei auf die verschiedenen Funktionen eines Wikis ein.

5.3.1 Was ist ein Wiki?

„*The simplest online database that could possibly work. [...] Wiki is a piece of server software that allows users to freely create and edit Web page content using any Web browser. Wiki supports hyperlinks and has a simple text syntax for creating new pages and crosslinks between internal pages on the fly.*“ [Wiki.org, 2002]
Das Konzept, auf dem ein Wiki basiert, wurde ursprünglich von Wissenschaftlern der Carnegie-Mellon Universität in Form des ZOG Datenbanksystems im Jahr 1972 entwickelt. Diese Multiuser Datenbank bestand aus Text Frames (WikiPages), die einen Titel, eine Beschreibung und Verlinkungen zu anderen Frames beinhalteten. Diese Frames konnten über verschiedene ZOG Kommandos erstellt werden. [C2.com]
Das erste Wikisystem wurde vom Wissenschaftler Ward Cunningham Anfang 1995 entwickelt. Dieser benötigte eine Plattform für den Wissensaustausch im Bereich des Softwaredesigns. Der Name Wiki ist die Kurzform für das Wiki Wiki Web. „Wiki Wiki“ stammt aus dem Hawaiianischen und bedeutet so viel wie „schnell“. Ein Wiki soll somit eine schnelle Publikation von Inhalten im Web ermöglichen. (Vgl. [Ebersbach et al., 2007] zit. nach [Trattner, 2009, S. 41])

Die Merkmale eines Wikis fasst Cunnigham nach [Szugat et al., 2006, 47f] wie folgt zusammen:

- **Online Bearbeitung:** Ein Benutzer kann ohne zusätzliche Software direkt im Browser eine Seite verändern oder erstellen.
- **Intuitiver Zugang:** Wikis unterstützen die Verwendung einer eigenen Wiki Syntax, die sich an der natürlichen Sprache orientiert.
- **Einfache Vernetzung:** Durch eine intuitve Art der Verlinkung fördern Wikis eine aussagekräftige thematische Zuordnung verschiedener Seiten. Diese Links werden nicht über URL Definition vorgenommen, sondern der Seitenname dient als Verknüpfungsanker.

Eine zusätzliche aber ähnliche Zusammenfassung liefert (Vgl. [Ebersbach et al., 2007, S. 18] zit. nach [Trattner, 2009, S. 45]):

- **Nichtlineare Hypertextstruktur:** Jede Wikiseite beinhaltet verschiedene Querverweise zu anderen Seiten. Der Benutzer kann entscheiden, welche Seite er als nächstes betrachtet. Wikis ermöglichen somit die Entstehung von assoziativen Hypertexten mit einer nichtlinearen Navigationsstruktur.
- **Einfacher und weitgehender Zugriff:** Durch ein Wikisystem werden die Fachkenntnisse auf ein Minimum reduziert, da jederzeit „on-the-fly“ Inhalte mit einfachen Regeln geändert werden können.
- **Keine Client-Software:** Um Inhalte zu lesen oder zu verändern, wird nur ein Browser vorausgesetzt, der ohne jegliche zusätzliche Plugins oder Applets auskommt.

- **Soziale Prozesse im Vordergrund:** Bei Wikis steht nicht die Technologie im Mittelpunkt, sondern die mit den Beiträgen einhergehenden Diskussionen und philosophischen Debatten.

Die Idee von Cunningham fand bei der Community einen regen Anklang, und so wurden verschiedene Wiki Klone in allen möglichen serverseitigen Programmiersprachen entwickelt. Heutzutage existieren laut (Vgl. [CosmoCode] zit. nach [Trattner, 2009, S. 44]) über 300 verschiedene Wiki Implementationen.

5.3.2 Wiki Technik

Obwohl sich diese Klone teilweise beträchtlich in der Funktionalität unterscheiden, unterliegen sie trotzdem alle dem gleichen technischen Prinzip. Wikis sind Client-Server Anwendungen, die im World Wide Web agieren.
Die Kommunikation zwischen einem Client und Server basiert somit auf dem HTTP Protokoll und wird durch die Request-Methoden GET und POST ausgeführt. Diese beiden Methoden wurden bereits im Abschnitt 5.2.6.1 beschrieben.
Möchte nun ein Benutzer einen Eintrag des Wikisystems im Browser anzeigen, wird folgender Prozess durchlaufen [Trattner, 2009, S. 46]:

1) Der Benutzer sendet über den Browser einen GET Request an den Server. Der Server verarbeitet diesen Request und erkennt über die URL, welche Wikiseite angefragt wurde.

2) Das Wikisystem erzeugt im so genannten Verarbeitungsschritt aus der Wikisyntax der angefragten Seite einen HTML Quellcode.

3) Dieser Quellcode wird durch das Wikisystem in eine Vorlage eingebettet, um somit die fertige HTML Seite zu erzeugen.

4) Das HTML Dokument wird vom Server zurück an den Browser gesendet, der im Anschluss dieses dann visualisiert.

5.3.3 Wahl eines geeigneten Wikis

Die Wahl des geeigneten Basis-Wiki baut auf der Arbeit von [Trattner, 2009] auf, der im Zuge dessen eine Evaluation von den fünf wichtigsten Wiki Implementationen (MediaWiki[12], FlexWiki[13], JSPWiki[14], TWiki[15] und DokuWiki[16]) vorgenommen hat.
Das Resultat dieser Bewertung kann der Tabelle 5.6 entnommen werden. Die Wahl fiel dabei auf das JSPWiki, das durch verschiedene Vorteile punkten konnte. So zeichnet sich das System vor allem durch sein leicht erweiterbares Datastoreage-Modul wie auch durch die gute Userverwaltung aus. Auch andere Erweiterungen können durch die hervorragende Codebase, die verfügbaren Plugins und Filter und den guten Support einfach durchgeführt werden. Zudem überzeugt auch die Ausgabe in XHTML 1.0 Strict. Auch die Programmiersprache spielt eine große Rolle, da durch Java eine klare Trennung in der Architektur möglich ist und auch andere Punkte wie Typsicherheit oder Exception Handling unterstützt werden.
Die Evaluation wurde ursprünglich für das Austria Lexikon[17] ausgelegt, doch das JSPWiki stellte auch für das Schwester Projekt SprichWort-Datenbank eine gute Basis dar. Aus diesem Grund liegt es nahe, auch das Übungs- und Testsystem auf der gleichen Grundlage aufzubauen. Somit sind die Mechanismen zur Erzeugung und Verwaltung von Inhalten bereits bekannt.

[12] http://www.mediawiki.org
[13] http://www.flexwiki.com
[14] http://www.jspwiki.org
[15] http://twiki.org
[16] http://www.dokuwiki.org
[17] http://www.austria-lexikon.at

Kriterien	*Media Wiki*	*Flex Wiki*	*JSP Wiki*	*TWiki*	*Doku Wiki*
Security Features	o	++	+	++	+
Userverwaltung	+	++	++	+	++
Strukturierung	++	o	+	++	++
Suche	++	+	++	--	+
Kommentare, Forumen, Blogs	o	--	+	+	o
XHTML,CSS, RSS, PDF	+	-	++	+	+
Media (Video, Audio, Flash)	++	--	++	++	o
Conflict Management	++	o	o	++	o
Hersteller Support	o	--	++	o	++
Datastorage	o	+	++	o	o
Erweiterbarkeit	o	++	++	+	o
Code&Programmiersprache	+	+	++	o	o
Printer Friendly	++	-	++	+	++
Syntax	+	-	+	+	+
Usability	++	--	+	-	+

++ Sehr Gut, + Gut, o Befriedigend, - Genügend, -- Nicht Genügend

Abbildung 5.6: Vergleich der fünf Wikis - MediaWiki, FlexWiki, JSPWiki, TWiki und DokuWiki nach [Trattner, 2009, S. 79]

5.3.4 JSPWiki

Das JSPWiki baut, wie der Name schon sagt, auf der Verwendung von Java Server Pages auf und ist unter der Apache License 2.0 erhältlich. Das JSPWiki, zurzeit in der Version 2.8.2 erhältlich, zeichnet sich durch einen qualitativ hochwertigen Code aus, sodass es von der Apache Software Foundation aufgenommen wurde.
Das Wikisystem unterstützt nicht nur die übliche Wiki Funktionalität, wie Beiträge kollaborativ zu erstellen, sondern bietet auch zusätzliche Elemente, Kommentarfunktionen, Foren, Blogs oder RSS-Feeds an.
Neben dieser Fülle an Funktionen legen die Entwickler des JSPWikis auch einen Fokus auf die Erweiterbarkeit des Systems. So wird zum Beispiel ein File-Provider-Modul zur Verfügung gestellt, auf dem sämtliche Speicherkonzepte abgebildet werden können. Bereits in der Standardversion sind ein CVS-, ein File-System und ein Buffer-File-System Provider vorhanden. Aber auch verschiedene Datenbanken werden unterstützt.
Hinsichtlich der Visualisierung des Systems stellt das JSPWiki ein Template-Konzept zur Verfügung, das entsprechend angepasst werden kann. So können auch einige Vorlagen von der Herstellerwebsite integriert werden.
Für das Usermanagement wird das Java Authentication and Authorization Service (JAAS) verwendet, das einem Benutzer, ähnlich einer HttpSession, eine WikiSession zuordnet.
Betreffend den Mediensupport können neben Bildern auch Videos oder Flashanimationen in eine Seite integriert werden. Diese Dateien werden als Attachment einer Wikiseite gehandelt und können somit auch referenziert werden.
Die Dokumentation fällt sehr gut aus, und die Herstellerwebsite stellt unzählige Informationen über das System zur Verfügung. Auch die regen Community-Tätigkeiten sind von Vorteil. Die in diesem Abschnitt verwendeten Quellen sind, wenn nicht anders angegeben, [JSPWiki Community] und [Trattner, 2009].

5.3.4.1 Architektur

Das JSPWiki folgt dem Model-View-Controller (MVC) Architektur Muster und kann in folgende Teile zerlegt werden. Die Architektur Abbildung 5.7 veranschaulicht die Verwendung des Design Patterns näher. Zusätzlich werden in Abbildung 5.8 die wichtigsten Kompontenen des JSPWiki dargestellt.

Das Model wird durch die *WikiEngine* und zusätzliche Hilfsobjekte verwaltet. Durch verschiedene Manager Objekte wird zum Beispiel der Page Speicher verwaltet. Jeglicher Zugriff auf Wiki Services erfolgt über die *WikiEngine*.

Die View wird durch mehrere JSP Seiten umgesetzt. Diese befinden sich im */template* Ordner und können je nach Bedarf angepasst werden.

Der Controller basiert ebenso auf JSP Seiten. Diese befinden sich aber im Top Level Directory. Der Controller beinhaltet die Logik des Frameworks, wie zum Beispiel das Editieren oder Löschen von Beiträgen. Der Vorteil durch die Verwendung von JSP Seiten, im Gegensatz zu Servlets, ist die große Flexibilität. So können diese Seiten modifiziert werden, ohne den Code neu zu kompilieren. Noch dazu sind JSP Seiten eigentlich Servlets.

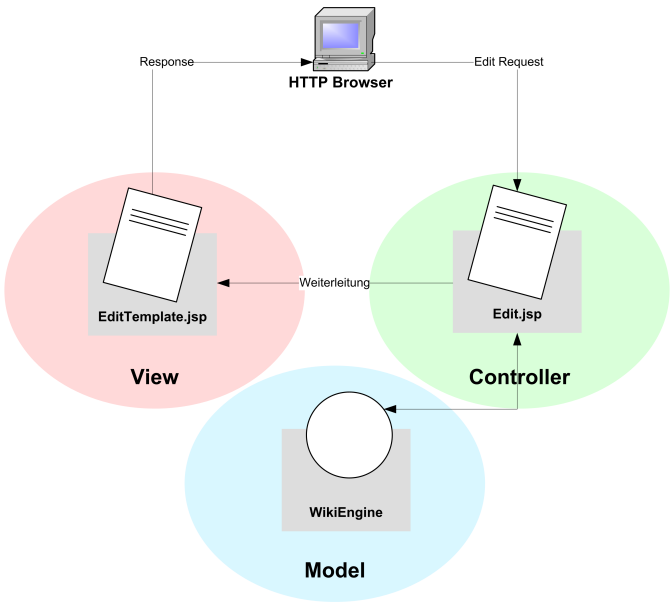


Abbildung 5.7: Model-View-Client Architektur des JSPWiki.

5.3.4.2 Plugins

Die wichtigste Erweiterungsmöglichkeit in funktionaler Hinsicht ist das Konzept der Plugins. Das JSPWiki unterstützt die Einbindung von Plugins in eine Wikiseite. Diese Plugins automatisieren Aktionen auf Wikiseiten (wie zum Beispiel das Search Plugin). Das JSPWiki unterstützt 23 Core Plugins und über 100 Contributed Plugins, die durch die Community entwickelt wurden.
Die formale Definition eines Plugins, zum Einfügen auf einer Seite, lautet wie folgt:

```
[INSERT <plugin class> WHERE <param1=value1>, <param2=value2>, ...]
```

Diese Definition kann aber auch verkürzt werden auf:

```
[<plugin class> <param1=value1>]
```

Ein Plugin setzt sich aus drei Elementen zusammen:

- die Plugin Klasse

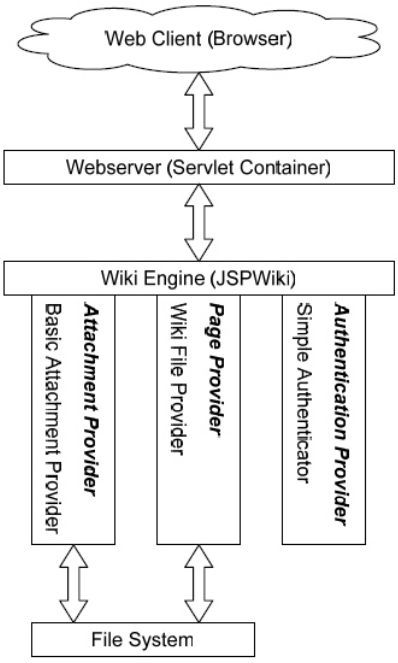


Abbildung 5.8: Die wichtigsten Komponenten des JSPWiki im Überblick. [Hartmetz, 2005]

- die Plugin Parameter
- der Plugin Body

Durch die Implementation des Interfaces `com.ecyrd.jspwiki.plugin.WikiPlugin` können eigene Plugins erzeugt werden. Wird ein Plugin in einer Seite eingebunden, so wird dessen *execute*-Methode aufgerufen und die Darstellung und Funktionalität der Wikiseite entsprechend manipuliert. Als Resultat wird im Anschluss die Wikiseite in Form von HTML durch *execute* retourniert.

5.3.4.3 Forms

Die Erstellung von einfachen HTML Formularen wird durch das JSPWiki leicht gemacht. So wird hierzu eine Sammlung von verschiedenen Wiki Forms Plugins bereitgestellt, die diese Funktionalität abstrahieren. Die Resultate des Formulars werden an einen benutzerdefinierten Handler weitergeleitet, der wiederum auf einem Plugin basiert.

Das JSPWiki unterstützt hierzu folgende Plugins:

FormOpen - ist zuständig für die Eröffnung eines Formulars. Dieses Formular ist dem HTML Schema angelehnt und benötigt somit verschiedene Parameter wie form, submit, method oder hide.

```
[FormOpen form='meinFormName' submit='/servlets/halloWelt'
method='post' hide='onsuccess']
```

CloseForm - übernimmt den Abschluss eines Formulars indem der Tag $< /form >$ angehängt wird.

```
[FormClose]
```

FormSet - ist zuständig für die verborgene Übermittlung von Parametern und Wertzuweisungen, sowie für die Konfiguration von Formaten.

```
[FormSet form='testForm' format='EEE, d MMM yyyy mm:ss:HH Z']
```

FormOutput - spezifiziert den Handler für das Ergebnis des Formulars.

```
[FormOutput form='testForm' handler='CurrentTimePlugin']
```

FormInput, FormSelect, FormTextarea - bilden die Steuerelemente eines Formulars. Durch Parametrisierung werden diese Elemente, ähnlich wie in HTML, vorkonfiguriert.

```
[FormInput type='text' name='format']
[FormSelect name='breakfast' value='egg;bacon;spam;']
[FormTextarea name='whyIlikeMontyPython' rows=5 cols=40]
```

5.3.4.4 Dynamic styles

Dynamische Styles ermöglichen ein erweitertes Design für Wikiseiten. Diese Styles können in einer Wikiseite ähnlich wie Plugins definiert werden und sind eine Kombination aus JavaScript und CSS. Ein möglicher Style ist zum Beispiel der *slimbox* Style. Durch ihn wird es ermöglicht, ein Bild oder ein IFrame in Form des bekannten Lightbox[18] Stiles anzuzeigen.

Die Anwendung eines solchen Styles wird mit folgender Syntax durchgeführt:

```
%%<style name> ... %%
```

Anzumerken ist, dass Styles jedoch nur die Präsentation der Seite beeinflussen sollten.

5.3.4.5 Filter

Ein neues elementares Element von JSPWiki wurde mit der Version 2.2 eingeführt - Filter. Diese dienen dazu, den Ein- und Ausgabeprozess zu manipulieren. So können zum Beispiel Spamfilter eingesetzt werden, um unerwünschte Inhalte von Seiten zu löschen oder diese komplett zu sperren.
Filter können zu vier verschiedenen Zeitpunkten aufgerufen werden:

- vor der Konvertierung der Wikisyntax auf HTML Code (`preTranslate`)
- nach der Konvertierung der Wikisyntax auf HTML Code (HTTP Request - SEND wurde noch nicht abgesetzt) (`postTranslate`)
- vor der Speicherung der Wikiseite im Datastorage-Modul (`preSave`)
- nach der Speicherung der Wikiseite im Datastorage-Modul (`postSave`)

Wird eine Methode des Filters zu einem der vier Zeitpunkte aufgerufen, so werden der `WikiContext` und der `Content` übergeben. Der Filter kann dadurch Informationen über den Kontext beziehen oder diesen befüllen. Der Content stellt den Inhalt der Wikiseite zum entsprechenden Prozessschritt dar.
Durch die Implementation des Interfaces `com.ecyrd.jspwiki.filters.PageFilter` können auch hier wieder eigene Filter erzeugt werden.

[18] http://en.wikipedia.org/wiki/Lightbox_(JavaScript)

5.3.4.6 Variablen

Das JSPWiki stellt ein Variablen-Modul zur Verfügung, durch das verschiedene Variablentypen unterstützt werden.

- **Konstante Variablen** - oder auch System Variabeln genannt, sind durch das System vordefinierte Variablen und können global von jeder Seite aus referenziert werden.
- **Kontext Variablen** - werden in verschiedenen Kontexten, wie zum Beispiel Plugins, verwendet. Diese Variablen werden verwendet, um Werte auszutauschen oder eine Anwendung zu konfigurieren.
- **Property Variablen** - sind zum Auslesen von Property Information aus der *jspwiki.properties* Datei gedacht.
- **Page Variablen** - dienen dazu, selbst Variablen auf einer Wikiseite zu definieren.

Die Definition von JSPWiki Variablen erfolgt durch folgende Syntax:

```
[SET <name>='<value>']
```

Um den Wert dieser Variable auf einer WikiPage auszulesen, wird folgende Syntax verwendet:

```
[<name>]
```

5.4 Übungs- und Testsystem

Die Konzeption des Übungs- und Testsystems stützt sich auf die zuvor zusammengefassten Technologien. Die Hauptbausteine hierfür sind das Wikiframework auf der Serverseite und die freie und modulare JavaScript-Bibliothek Dojo auf der Clientseite. Für das Wikiframework wurde das Java basierte JSPWiki, für den Einsatz von JavaScript wurde das Dojo Toolkit gewählt.
Das JSPWiki stellt einfache Mechanismen zur Erzeugung und Verwaltung von Inhalten zur Verfügung. So können Inhalte am Server durch eine einfache Wikisyntax erstellt oder modifiziert werden. Zudem können Benutzer nicht nur über einen Editor Inhalte ändern, sondern auch Plugins oder Styles platzieren. Auf der Clientseite bietet das Dojo Toolkit nicht nur eine effektive Methode zu dynamischen Änderungen von Website Inhalten, sondern bietet auch die Möglichkeit zu AJAX Kommunikation. Dojo abstrahiert verschiedenste Browser Implementationen und erleichtert die JavaScript Entwicklung, um die Cross-Browser Fähigkeit zu bewahren.
Das folgende Architektur Konzept 5.9 zeigt einen Überblick über die Komponenten des Systems. Dabei wurde darauf geachtet, dass auf beiden Seiten, Server wie auch Client, eine Art Model-View-Controller Pattern verwendet wurde. Das JSPWiki baut generell seine Architektur auf diesem Design Pattern auf. Im Clientbereich musste aber ein Weg gefunden werden, um die DOM Manipulationen von der AJAX Kommunikation zu trennen. Die Grundlage hierzu lieferte [Chisholm, 2008] und [Rueth].

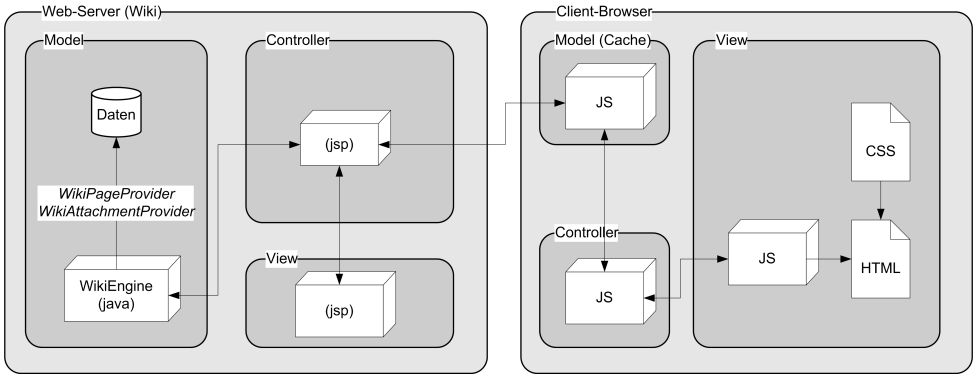


Abbildung 5.9: Architektur des Übungs- und Testsystems mit dem Fokus auf einer AJAX Kommunikation. Die synchrone Kommunikation wird bei dieser Darstellung nicht berücksichtigt.

5.4.1 Abstraktionsschicht (1) - Dojo Widgets deklartiv in Markup einbinden

Als die Implementation begonnen wurde, war die Aufgabe ein Template System für Aufgaben in Form von Dojo Widgets zu erstellen. Instruktoren hätten über diese Abstraktionsschicht HTML Seiten erzeugt und diese Aufgaben Widgets deklarativ über Markups eingebunden. Nach kurzer Zeit zeigte sich aber die Komplexität dieser Erzeugung. Instruktoren hätten dafür HTML und Javascript Hintergrundwissen gebraucht, da viele Einzelkonfigurationen vorgenommen werden mussten. Aus diesem Grund wurde eine weitere Abstraktionsschicht eingeführt. Diese Schicht beruht auf der Verwendung der zuvor beschriebenen Mechanismen des Wikiframeworks.

```
1  <div dojoType="dispue.AssessmentController" id="c1"></div>
2
3  <p><b>Aufgabe: </b>Markieren sie die Fehler in den Sprichw&ouml;rtern. <br>
4  <hr>
5  <div class="content">
6    Die Wasserballer schafften mit psychologischer Hilfe die Olympia-Qualifikation.
7    Handball-Bundestrainer Heiner Brand gilt als Fan der Sportpsychologie.
8    Als seine Mannschaft Europameister wurde, ließ er vor jedem Spiel im
9    Besprechungszimmer Sprüche wie diesen aufhängen: " Wenn es
10   eine<div dojoType="dispue.HotspotMarking" controller="c1"
11     params="{ 'params': [{
12           'content':'n',
13           'isCorrect':true,
14           'hint':'Hmm. Das passt doch oder?'},
15         {   'content':'m',
16           'isCorrect':false,
17           'hint':'Komisch oder?'}
18         ]}"
19     isRandomized=true></div>&nbspGlauben gibt, der
20     <div dojoType="dispue.HotspotMarking" controller="c1"
21     params="{ 'params': [{
22           'content':'Berge',
23           'isCorrect':true,
24           'hint':'Das passt hier her!'},
25         { 'content':'Wolkenkratzer',
26           'isCorrect':false,
27           'hint':'Das stimmt nicht ganz so!'}
28         ]}"
29     isRandomized=true>
30     </div>&nbspversetzen kann, dann ist es der Glaube an die eigene Stärke.
31 </div>
```

Listing 5.5: Abstraktionsstufe 1 - Verwendung von Dijits durch deklarative Einbindung in Markup Code.

5.4.2 Abstraktionsschicht (2) - Wikiseite als Container für Tests/Übungen und Aufgaben

In dieser Abstraktionsschicht kann der Instruktor auf einer Wikiseite ein Aufgaben Plugin definieren. Durch diese Abstraktion wird die Verwendung von JavaScript, HTML und CSS zur Gänze gekapselt. Das Konzept ist nicht nur für Aufgaben, sondern auch für Tests und Übungen gedacht. Eine Wikiseite bildet hier, je nach Konfiguration, eine Aufgabe, einen Test oder eine Übung. Die Elemente dieser verschiedenen Typen werden über Plugins hinzugefügt.

```
1  Ein alt bekanntes Sprichwort ist:
2  [{SpHotspotKennzeichnung
3  |
4  inhalt='Eigener Herd ist goldes wert.'
5  korrekt=true
6  hinweis='Hmm. Das passt doch oder?'
7  |
8  inhalt='Eigener Ofen ist silber wert.'
9  korrekt=false
10 hinweis='Das stimmt nicht ganz so!'
11 |
12 }]
```

Listing 5.6: Abstraktionsstufe 2 - Verwendung einer eigenen Wikisyntax um Dijits zu abstrahieren.

5.5 Konklusion

Dieses Kapitel sollte als Grundidee und Konzeption des angeforderten Systems verstanden werden. Dabei wurden alle grundlegenden Technologien diskutiert und verschiedene Entscheidungen getroffen, welche Frameworks/Toolkits verwendet werden. Weitere Details, die die exakte Umsetzung betreffen, werden im folgenden Kapitel „Implementation eines Assessmentsystems“ erläutert.

Kapitel 6

Implementation eines Assessmentsystems

„Wenn du gerne lernst, wirst du auch viel lernen. Was du gelernt hast, erhalte durch Übung.“

[Isokrates (436 - 338 v. Chr.), griechischer Redner]

6.1 Grundkonzept - Wikiseite als Container für Tests/Übungen/-Aufgaben und Selbstbewertungen

Die Hauptanforderung an das System ist eine einfache Erzeugung von übungs- und testbezogenen Inhalten. Um dieser Anforderung gerecht zu werden, baut das System stark auf den Wiki üblichen Funktionalitäten und Eigenschaften auf. Diese Funktionen und Eigenschaften setzen sich aus folgenden Punkten zusammen:

- die Online Bearbeitung von Inhalten, wobei keine weitere Client Software benötigt wird.
- die Wikisyntax, die auf Grund ihrer Nähe zur natürlichen Sprache einen intuitiven Zugang bietet.
- die Unterstützung von nicht linearen Hypertextstrukturen, die sich vor allem für die Verknüpfung von Übungs- und Testinhalten eignen und somit eine Navigation ermöglichen.

Zusätzlich stützt sich das System auch auf spezielle JSPWiki Konzepte. Darunter zu finden sind Werkzeuge wie Wiki Plugins, Filters und Styles. Ebenso werden auch bereits profilierte Plugins verwendet, die zum Beispiel eine unterstützende Tätigkeit im Erstellungsprozess durchführen.
Die dadurch entstehenden Wikiseiten bilden somit einen Container für test- und übungsspezifischen Inhalt. Je nach Konfiguration kann eine Wikiseite eine Aufgabe, einen Test, eine Übung oder eine Selbstbewertung darstellen. Die Konfiguration dieser verschiedenen Typen wird über Plugins und die herkömmliche Wikisyntax vorgenommen. Da Wiki Plugins aber nur einfache Datentypen und keine Datenstrukturen wie Arrays unterstützen, wird auch auf die Hilfe von Filtern zurückgegriffen. Diese Filter übernehmen die Aufgabe der Datenaufbereitung und stellen den Plugins die geeigneten Daten zur Verfügung.
Die Abbildungen 6.1 und 6.2 zeigen eine beispielhafte Wikisyntax und die daraus resultierende Darstellung.

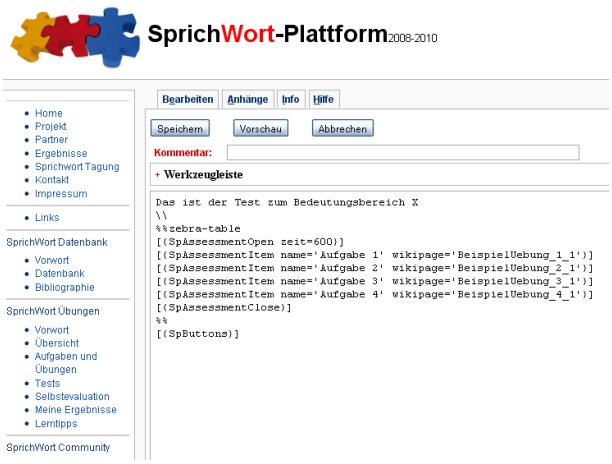


Abbildung 6.1: Eine beispielhafte Wikisyntax für die Erstellung eines Tests.

Abbildung 6.2: Die Darstellung einer exemplarischen Test Syntax in Form einer Wikiseite.

6.1.1 Warum Filters?

Die Plugin Erweiterungen verwenden eine spezielle Form der Syntax. Dieser Umstand beruht auf der Verwendung von mehreren Konfigurationen für ein Aufgaben-Template. Mit der herkömmlichen Syntax für Plugins können nur jeweils unterschiedliche Parameter definiert werden. Durch die Erweiterung in diesem System können aber auch Gruppen von Parametern mit übergeben werden. Zudem werden auch Arrays unterstützt, die zum Beispiel bei den Antworten für die Multiple Choice Fragen verwendet werden. In den folgenden Code Snippets 6.1 und 6.2 zeigen sich deutlich die Unterschiede zwischen einem herkömmlichen Plugin und der Erweiterung dieses Systems.

Herkömmliche Verwendung von Plugins:

```
1 [{SpAssessmentItem name='Aufgabe 1 (Hotspot finden)' wikipage='Aufgabe1'}]
```

Listing 6.1: Ein herkömmliches Plugin Beispiel.

Parameter Erweiterung für verschiedene Parametergruppen und Arrays:

```
Der [{SpHotspotMCQ
|
inhalt='____'
korrekt=false
korrekterInhalt='Glaube'
mcqs=('Glaube';'Hinweis';'Riese')
|
inhalt='Riese'
korrekt=false
korrekterInhalt='Glaube'
mcqs=('Glaube';'Gedanke';'Traum';'Schlaf')
|
}] versetzt Berge
```

Listing 6.2: Ein Plugin Beispiel mit Parametergruppen.

Um nicht in den Parsing Prozess für Plugins des Wikiframeworks einzugreifen, wird ein Filter angewendet, der diese Aufgabe übernimmt. Ein Filter zerlegt vor der Erstellung des HTML Codes mit der Methode *preTranslate* die Wikisyntax und verarbeitet die übergebenen Parameter. Diese Parameter werden als Variablen in den Wiki Kontext geschrieben. Die Wikisyntax wird so verändert, dass lediglich eine speziell generierte ID dem Plugin Prozess übergeben wird. Diese ID verhilft später dem Plugin dazu, die eigenen Parameter aus dem Kontext zu finden und die Parameterliste zu laden.
Ein weiterer Aufgabenbereich des Filters ist es, Header Information für die Integration des Dojo Frameworks in die HTML Seite einzubinden. Hierzu werden in der *postTranslate* Methode, nach der Konvertierung von Wikisyntax zu HTML Code, alle benötigten Informationen eingebunden. Das folgende Code Snippet 6.3 zeigt einen kleinen Auszug aus dem zentralen Filter.

```
public class AssessmentFilter extends BasicPageFilter
{
    private static final String PLUGIN_INSERT_PATTERN = "\\{?(INSERT)?\\s*([\\w\\._]+)[ \\t]*(WHERE)?[ \\t]*";
    private Pattern m_pluginPattern;
    protected WikiEngine m_engine;
    private static Logger log = Logger.getLogger(AssessmentFilter.class);

    public void initialize( WikiEngine engine, Properties properties )
        throws FilterException
    {
        // [..] Do initializing work
    }

    // modify the Wikisyntax before the wiki to html translation
    public String preTranslate( WikiContext context, String content )
    {
        AssessmentParser parser = new AssessmentParser();
        ArrayList<String> plugins = parser.parseLinks(content);
        Iterator it = links.iterator();
        // [..]
        while(it.hasNext()) {
            String line = ((String)it.next());
            // [..]
            PatternMatcher  matcher  = new Perl5Matcher();
            if( matcher.contains( line, m_pluginPattern ) )
            {
                MatchResult res = matcher.getMatch();
                String plugin   = res.group(2);

                // check if plugin is an item plugin
                if (plugin.equalsIgnoreCase("SpHotspotKorrektur") ||
                    // [..]
                    plugin.equalsIgnoreCase("SpDnd")) {
                    // [..] find parameter location
                    List paramslist = parser.parse( line.substring( beginOfParams, endOfParams + 1 ) );
                    context.setVariable( "paramslist" + id, paramslist );
                    // [..] set parameter list id to the plugin line and modify the content
                }
            }
        }
        return content;
    }

    // modify the html after the wiki to html translation
    public String postTranslate( WikiContext wikiContext, String htmlContent )
    {
      // check if generated html content contains any dojo elements of the dispue package
        if (htmlContent.indexOf( DISPUE_CHARACTERISTIC )>=0) {
            StringBuilder sb = new StringBuilder();
            // append the dojo for the html site
            sb.append("<script type='text/javascript' src='/js/dojo/dojo/dojo.js'" +
                      "  djConfig='parseOnLoad: true, isDebug: true, usePlainJson: true, modulePaths:{dispue:\"../../../
                          dispue\"}'></script>" +
```

```
                    "<script type='text/javascript'>                        " +
                    "    function init() { dojo.query('body').addClass('nihilo'); }" +
                    "    dojo.addOnLoad(init);" +
                    "</script>                                              " +
                    "<style type='text/css'>                                " +
                    "    @import '/js/dojo/dijit/themes/nihilo/nihilo.css';  " +
                    "    @import '/dispue/css/player.css';                   " +
                    "</style>" +

                    "<div id='assessment'>");

            sb.append( htmlContent );
            sb.append( "</div>");
            return sb.toString();
        }
        // [..] do additional html modification (IE specialities, Controller attachments)

        return htmlContent;
    }
}
```

Listing 6.3: Auszug aus dem zentralen Filter der Applikation.

6.1.2 Warum Plugins?

Das System verwendet zur Konfiguration neben der herkömmlichen Wikisyntax den Mechanismus der Plugins. Diese Plugins sind Platzhalter und können unterschiedlichen HTML und JavaScript Code in die Seite einfügen. Je nach Plugintyp und Parameter wird meist ein Dojo Widget in die Webseite eingebunden.
Aufgrund der Parametrisierungs- und Randomisierungsanforderung verwenden die Aufgaben Plugins eine spezielle Form der Parameterübergabe. Durch Parametergruppen können unterschiedliche Darstellungen der Aufgabe iniitiert werden. Ein mögliches Beispiel könnte wie im Code Snippet 6.4 aussehen:

```
[{SpHotspotKennzeichnung
|
inhalt='Eigener Herd ist goldes wert.'
hotspot=false
|
inhalt='Eigener Ofen ist silber wert.'
hotspot=true
|
}]
```

Listing 6.4: Wikisyntax für die Verwendung von Parametergruppen.

In diesem Plugin werden zwei Parametergruppen definiert, die nach dem Zufallsprinzip dem Kandidaten angezeigt werden. Diese Gruppen werden mit dem Zeichen „|“ eingegrenzt.

```
Parametergruppe 1 = (inhalt='Eigener Herd ist goldes wert.'
hotspot=false)
Parametergruppe 2 = (inhalt='Eigener Ofen ist silber wert.'
hotspot=true)
```

Wird die Seite durch den Kandidaten geöffnet, wird bei diesem Hotspot immer nur eine Parametergruppe angezeigt. Bei dieser Konfiguration können beliebig viele Gruppen hinzugefügt werden, um somit die Aufgaben interessanter und abwechslungsreicher zu gestalten. Die Verarbeitung dieser Parameter wird im Abschnitt 5.3.4.2 näher beschrieben und wird hier nicht näher ausgeführt.
Eine weitere Anforderung ist, dass mehrere Aufgaben des gleichen Typus in der gleichen Seite verwendet werden. Um eine mehrfache Initialisierung des Dojo Headers eines bestimmten Widgets zu vermeiden wird eine „INIT“ Varibale verwendet. Diese Variable wird nach dem ersten Aufruf eines Widget Typus in den Wiki Context gesetzt und nimmt somit die Initalisierungsarbeit vor. Das Code Snippet 6.5 der *execute* Methode eines Plugins zeigt die Verwendung dieser Variable und das laden der Parameterliste.

```
1  public String execute( WikiContext context, Map parameterMap ) throws PluginException
2  {
3      /** look for any supported parameter * */
4      Integer id = null;
5      List paramslist = null;
6
7      try
8      {
9          id = Integer.parseInt( getParameter( ID, parameterMap ) );
10     }
11     catch( NumberFormatException e )
12     {
13         throw new PluginException( ERROR_ID_NO_NUMBER );
14     }
15     /** get the parameterlist, which includes the different parametermaps, from the context **/
16     paramslist = (List) context.getVariable( PARAMSLIST + id );
17     if( paramslist == null )
18         throw new PluginException( ERROR_NO_PARAMSLIST );
19
20     StringBuilder sb = new StringBuilder();
21
22     /**
23      * set init Header of Widget (will be used by all widgets of the same
24      * type)
25      *
26      */
27     if( context.getVariable( INIT_PARAM + this.getClass().toString() ) == null )
28     {
29         context.setVariable( INIT_PARAM + this.getClass().toString(), true );
30         sb.append( "<script type='text/javascript'>"
31                    + "dojo.require('dispue.HotspotCorrection');"
32                    + "</script>"
33                    + "<style type='text/css'>"
34                    + "@import '/dispue/css/hotspot.css';"
35                    + "</style>" );
36     }
37     // [..] do the main plugin work and integrate the widget into the html code
38 }
```

Listing 6.5: Die execute Methode eines Plugins im Wikiframework.

Das System unterstützt folgende Plugins, die in den nachfolgenden Abschnitten näher beschrieben werden:

Allgemeine Plugins

- SpAnleitung
- SpButtons
- SpNiveau
- SpPhase

Aufgaben Plugins

- SpDnd
- SpDndListe
- SpDndQuelle
- SpDndSalat
- SpDndZiel
- SpHotspotKennzeichnung
- SpHotspotKorrektur
- SpHotspotLuecke

- SpHotspotMCQ
- SpMCQ
- SpMemory
- SpTextfeld

Test und Übungs Plugins

- SpAssessmentOpen
- SpAssessmentItem
- SpAssessmentClose
- SpAssessmentErgebnisse
- SpTexte

Selbstbewertungsbogen Plugins

- SpSbb
- SpSbbErgebnisse

6.1.3 Warum Styles?

Dynamische Styles ermöglichen dem Benutzer die Einbindung und Konfiguration von erweiterten Designs für Wiki Elemente. Diese Styles können in einer Wikiseite ähnlich wie Plugins definiert werden und sind eine Kombination aus JavaScript und CSS. Das System verwendet speziell im Übungsbereich eine Erweiterung des „Lightbox“ Styles. Dadurch wird einem Link die Eigenschaft hinzugefügt, dass nach der Aktivierung ein schwarzer Container als IFrame mit der Aufgabenseite angezeigt wird. Das folgende Code Snippet 6.6 zeigt eine beispielhafte Anwendung für eine Übungsaufgabe.

```
1 %%slimbox-question
2 [Aufgabe Memory|Memory]
3 %%
```

Listing 6.6: Die Anwendung des Lightbox (Slimbox) Styles auf einen Aufgaben Link.

6.1.4 Verwaltung von Attachments

Die Aufgaben des Projektes verwenden häufig verschiedene Attachments wie zum Beispiel Bild-, Audio- oder Videodateien. Häufig teilen sich mehrere Aufgaben die gleichen Ressourcen. Aus diesem Grund wurde ein zentrales Lager für diese Ressource, getrennt nach Typ (Bild, Audio, Video), konzipiert.
Konkret heißt das, dass jeweils eine Wikiseite erstellt wird, in der alle benötigten Dateien hinzugefügt werden. Eine Aufgabe, Übungs- oder Testseite verweist im Anschluss auf diese zentrale Ressource um somit zusätzlichen Verwaltungsaufwand zu beschränken und Konformität zu wahren. Die Abbildung 6.3 zeigt die Ressourcensammlung für Bilder. Im Code Snippet 6.7 wird für eine Aufgabe eine Verlinkung auf das Bild „geld.jpg“ vorgenommen.

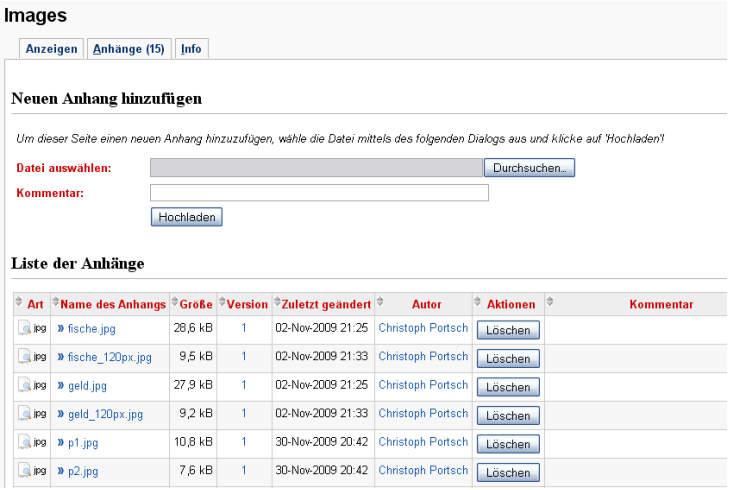


Abbildung 6.3: Das zentrale Ressourcenlager in Form einer Wikiseite.

```
[{SpPhase phase=4}]

[{SpAnleitung aufgabe='16.1 Schreiben Sie in ein paar Sätzen, welches Sprichwort mit dem Bild gemeint ist und warum sich
     das Bild zur Illustration des Sprichwortes eignet.'}]\\

[Images/geld.jpg]

[{SpTextfeld}]

[{SpButtons}]

[Zu den geschriebenen Texten|TexteBeispielUebung_16_1]
```

Listing 6.7: Die Verwendung von multimedialen Ressourcen in einer Wikiseite.

6.1.5 Erstellungsprozess der Aufgaben (für Übungen und Tests)

Aufgrund der enormen Aufgabenzahl ist es wichtig, im Erstellungsprozess den Status einer Aufgabe zu verfolgen. Im Fall des Sprichwort Projektes durchläuft eine Aufgabe drei verschiedene Phasen, die sich nach dem Bearbeitungsstand richten: „Aufgabe (Übung/Test) in Arbeit“, „Aufgabe (Übung/Test) fertig für interne Zwecke“, „Aufgabe (Übung/Test) fertig“. Um dieser Anforderung gerecht zu werden, wurde das Prinzip des *ReferringPagesPlugins* verwendet. Hierbei werden auf einer Übersichtsseite, welche dieses Plugin integriert, alle eingehenden Links angezeigt. Das folgende Code Snippet zeigt die Erstellung solcher Listen.

```
[ReferringPagesPlugin before='*' after='\n'
exclude='Intern,Didaktik_Beispiele']
```

Eine Aufgabe muss somit einen Link auf die entsprechende Übersichtsseite einbinden. Dieser Link wird manuel nach dem Bearbeitungszustand geändert. Das folgende Code Snippet 6.8 zeigt die Einbindung dieses Verweises.

```
%%(display:none)
[{ALLOW view Editors}] //Kommentar: die Anzeige und Editierung dieser Seite ist nur Editoren vorbehalten
[{ALLOW edit Editors}]
[Aufgabe_fertig] //Kommentar: dieser Link muss je nach Bearbeitungsstatus geändert werden
%%
```

Listing 6.8: Die Verwendung einer Erstellungsphase.

In diesem Fall wird die Aufgabe in der Seite: Liste aller Aufgaben (Übung/Test) in Arbeit angezeigt. Insgesamt werden folgende Listen bzw. Übersichtsseiten unterstützt.

- Liste aller Aufgaben (Übung/Test) in Arbeit
  (benötigter Link: `[Aufgabe_in_Arbeit]`)
- Liste aller Aufgaben (Übung/Test) fertig für interne Zwecke
  (benötigter Link: `[Aufgabe_intern_fertig]`)
- Liste aller Aufgaben (Übung/Test) fertig
  (benötigter Link: `[Aufgabe_fertig]`)

6.2 Architektur der Rich Internet Application

6.2.1 Generelle Architektur

Das *Model-View-Controller* Muster bildet nicht nur die Grundlage für die Architektur des JSPWiki auf der Server Seite, sondern wurde auch als Richtline für die Clientseite der Applikation genommen. Hierbei wurde darauf geachtet, dass die Daten, die Logik und die Darstellung von einander getrennt werden. Die Abbildung 6.4 zeigt das Klassendiagramm der clientseitigen Architektur:

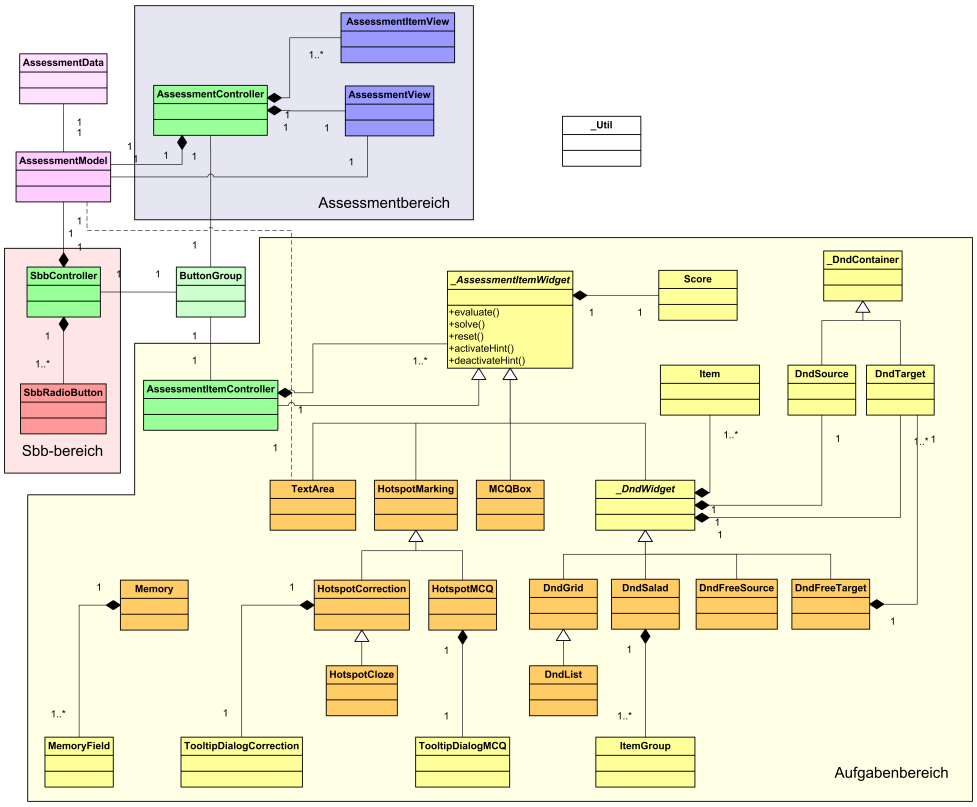


Abbildung 6.4: Das Klassendiagramm der clientseitigen Architektur.

Die verschiedenen *Controller* (`AssessmentController`, `SbbController` und `Assessment-ItemController`) übernehmen im System die Prozesssteuerung. Das *Model*, oder auch in diesem Fall als Cache bezeichenbar, ist zuständig für die Datenbereitstellung. Hierbei übernimmt das *Model* die Kommunikation zwischen Client und Server, wie auch die Speicherung und das Laden von Cookie Informationen. Die verschiedenen *Views* hingegen sind verantwortlich für die Darstellungen und die Interaktionslogik, wie zum Beispiel bei Drag and Drop.

Die Architektur kann in drei Bereiche unterteilt werden: der Assessment-, der Selbstbewertungsbogen- und der Aufgabenbereich. Der Assessmentbereich beinhaltet die Funktionalität für die administrative Darstellung und Durchführung von Tests. Der Selbstbewertungsbogenbereich implementiert die Anforderung der Selbstbewertungen. Im Aufgabenbereich sind alle Aufgaben Widgets und deren Interaktionen zu finden, die durch die Aufgabentypologie gefordert werden. Dieser Bereich wird von Tests wie auch von Übungen verwendet. Das *Model* und die `ButtonGroup` werden keinem konkreten Bereich zugeordnet, da sie eine zentrale Aufgabe übernehmen und von jedem Bereich kontaktiert werden. Die Notwendigkeit der Ausgliederung des Bedienungselements (`ButtonGroup`) von den verschiedenen *Views* wird im folgenden Abschnitt 6.2.2 näher besprochen.

6.2.2 Trennung der Aufgaben View

Die Anforderung des Systems sieht vor, dass die Aufgaben der Typologie sowohl für Tests als auch für Übungen verwendet werden sollen. Dabei entsteht die Problematik, dass eine Aufgabe in jedem Kontext (Test oder Übung) andere Bedienungselemente voraussetzt. Das heißt im Kontext eines Tests kann eine Aufgabe keine „Lösungs“ Funktionalität anbieten.
Aus diesem Grund wurde eine Trennung der *View* „Aufgabe“ durchgeführt. So wird nun zwischen dem wesentlichen Assessment Item und den Bedienungselementen unterschieden.
Ein weiterer Aspekt für diese Teilung ist die Verwendung von Sub Items. Diese Sub Items kommen vor allem bei Hotspot-Aufgaben oder bei zusammengesetzten Aufgaben vor. Dabei wird zum Beispiel jedes Hotspot Element als Sub Item bezeichnet. Durch diesen Umstand kann keine vordefinierte *View* verwendet werden, da erst während der Laufzeit feststeht, welche Elemente die gesamte Aufgabe verwendet und wo die Bedienungselemente platziert werden können.

6.3 Umsetzung der Aufgabentypologie

Nach der Anforderungsanalyse der Aufgabentypologie konnten verschiedene Aufgabentypen und hybride Formen identifiziert werden. Um den Anforderungen gerecht zu werden, wurden folgende Grundtypen definiert: Hotspot, Multiple Choice, Drag and Drop, Textaufgabe und Memory. Um Datenbankaufgaben zu erzeugen, kann ebenso aus den gelisteten Grundtypen gewählt werden. Zusätzlich wird aber ein Link zur SprichWort Datenbank zur Verfügung gestellt, um weitere Informationen zu recherchieren.
Die im Aufgabenbereich unter der Abbildung 6.4 dargestellten JavaScript Widgets ermöglichen die Interaktion zwischen dem Kandidaten und der Aufgabe. Die dunkel markierten Widget Klassen werden jeweils von einem JSPWiki Plugin gekapselt. Durch diese Kapselung wird dem Benutzer ermöglicht, diese Interaktionselemente in die Aufgabenseite zu integrieren. Das *Model* und der dazugehörige *Controller* werden automatisch vom Plugin hinzugefügt und dem neuen Widget übergeben.

6.3.1 Grundfunktionalitäten der JavaScript Widgets

Jede Interaktionsklasse (außer Memory) wird von `_AssessmentItemWidget` abgeleitet. Die Basis Klasse definiert fünf Grundfunktionen, die von den Ableitungen implementiert werden. Diese öffentlichen Funktionen setzen sich aus `evaluate`, `solve`, `reset`, `activateHint` und `deactivateHint` zusammen. Wird zum Beispiel die Bedienungsfunktion „Auswerten“ über einen Button aufgerufen, so wird die Funktion `evaluate` der Interaktionsklasse durchgeführt.
Es besteht aber die Möglichkeit, dass eine Aufgabe aus verschiedenen Interaktionen besteht, wie die folgende Abbildung 6.5 zeigt:

Ein Button darf hierbei nicht nur die Bewertungsfunktion eines einzelnen Hotspots aufrufen, sondern muss jeden Hotspot evaluieren. Abhilfe schafft hier der `AssessmentItemController`. Sobald eine

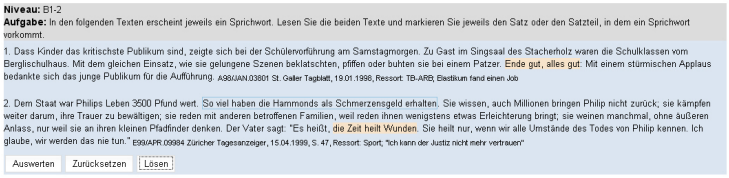


Abbildung 6.5: Die Verwendung von mehreren Interaktionen in einer Aufgabe.

Instanz der Interaktionsklassen erzeugt wird, wird dem *Controller* eine Referenz übergeben. Der *Controller* verwaltet diese Referenzen und kann dadurch gesammelte Evaluierungen erzeugen. Ein Button kommuniziert somit nur mit einem *Controller*.

6.3.2 Hotspot

Diese Klassen ermöglichen die Interaktion zwischen einem Kandidaten und einer Hotspot-Aufgabe. Die Basis bildet hier die Klasse `HotspotMarking` und implementiert damit die rudimentären Mouse Events. Durch dieses Widget können Aufgaben wie „Sprichwort kennzeichnen“ durchgeführt werden. `HotspotCorrection`, `HotspotCloze` und `HotspotMCQ` bilden eine Erweiterung der Basis, indem sie bei der Aktivierung eines Hotspots einen Dialog, mit einer erneuten Interaktionsmöglichkeit erscheinen lassen. `HotspotCorrection` deckt die Aufgaben wie „Sprichwörter korrigieren“ ab. `HotspotCloze` baut auf dieser Grundfunktionalität der Korrektur auf und ermöglicht dazu eine Darstellung in Form einer Lücke. `HotspotMCQ` gibt nach der Aktivierung eines Hotspots mehrere Antwortmöglichkeiten vor. Die folgenden Abbildungen zeigen die jeweiligen Darstellungen der Aufgaben und die verwendeten Wikisyntaxen, um diese zu erzeugen:

Markierung von Hotspots:
JS Widget: `HotspotMarking`

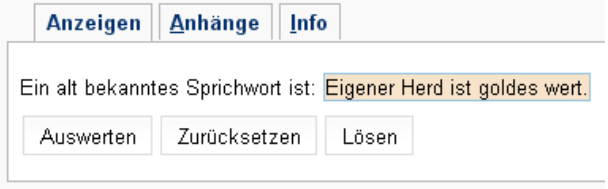


Abbildung 6.6: Die Darstellung eines Markierungs Hotspots.

```
1 Ein alt bekanntes Sprichwort ist:
2 [{SpHotspotKennzeichnung
3 |
4 inhalt='Eigener Herd ist goldes wert.'
5 hotspot=true
6 |
7 }]
8 [{SpButtons}]
```

Listing 6.9: Syntax für die Erstellung eines Markierungs Hotspots.

Korrektur von Hotspotelementen (in Form von Text):
JS Widget: `HotspotCorrection`

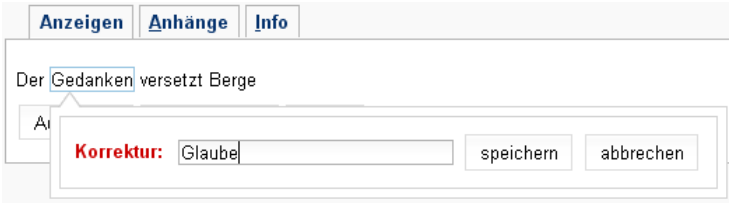


Abbildung 6.7: Die Darstellung eines Korrektur Hotspots.

```
Der [{SpHotspotKorrektur
|
inhalt='Gedanken'
korrekt=false
korrekterInhalt='Glaube'
|
}] versetzt Berge
[{SpButtons}]
```

Listing 6.10: Syntax für die Erstellung eines Korrektur Hotspots.

Korrektur von Lückentext (in Form von Text):
JS Widget: `HotspotCloze`

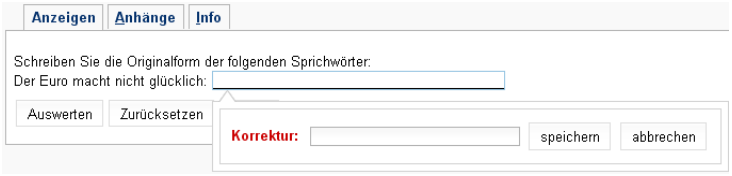


Abbildung 6.8: Die Darstellung eines Lückentext Hotspots.

```
Schreiben Sie die Originalform der folgenden Sprichwörter:\\
Der Euro macht nicht glücklich: [{SpHotspotLuecke
|
korrekterInhalt='Geld alleine macht auch nicht glücklich'
|
}]
[{SpButtons}]
```

Listing 6.11: Syntax für die Erstellung eines Lückentext Hotspots.

Korrektur von Hotspotelementen (in Form von Multiple Choice):
JS Widget: `HotspotMCQ`

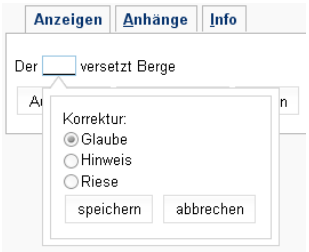


Abbildung 6.9: Die Darstellung eines Multiple Choice Hotspots.

```
Der [{SpHotspotMCQ
|
inhalt='____'
```

```
4| korrekt=false
5| korrekterInhalt='Glaube'
6| mcqs=('Glaube';'Hinweis';'Riese')
7| |
8| }] versetzt Berge
9| [{SpButtons}]
```

Listing 6.12: Syntax für die Erstellung eines Multiple Choice Hotspots.

6.3.3 Multiple Choice Fragen

Diese Klasse `MCQBox` repräsentiert die Interaktion zwischen einem Kandidaten und einer Multiple Choice Aufgabe. Die Anforderung des Systems ist, nur Single Responses zu unterstützen. Das heißt, dass ein Kandidat nur jeweils eine Antwort wählen kann. Der Interaktion kann aber mühelos ein Multi Response Verhalten hinzugefügt werden. Somit kann ein Kandidat mehrere Antworten auswählen. Folgende Abbildung zeigt die Darstellung einer Multiple Choice-Aufgabe und die verwendete Wikisyntax, um diese zu erzeugen:

JS Widget: `MCQBox`

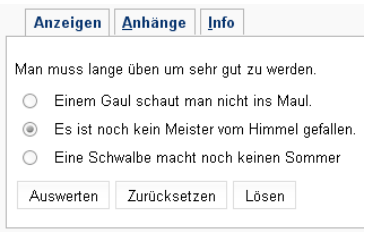


Abbildung 6.10: Die Darstellung eines Multiple Choice Frage.

```
1| [{SpMCQ
2| |
3| inhalt='Es ist noch kein Meister vom Himmel gefallen.'
4| idKorrekterInhalt=0
5| mcqs=('Man muss lange üben, um sehr gut zu werden.';'Ich kann schon alles.';'Unfälle passieren eben mal.')
6| |
7| }]
8| [{SpButtons}]
```

Listing 6.13: Syntax für die Erstellung eines Multiple Choice Frage.

6.3.4 Drag and Drop

Drag und Drop-Aufgaben stellen einen großen Anteil der Aufgabentypologie dar. Um die Grundfunktionalität dieser Interaktionen zu repräsentieren, wurde die Basisklasse `_DndWidget` eingeführt. Diese Klasse beinhaltet die zwei Hauptattribute einer Drag and Drop Interaktion. Zum einen den Source Container und zum anderen die/den Target Container, der die Elemente aus der Source aufnimmt. Beide Container werden zwar von der Dojo Bibliothek zur Verfügung gestellt, dennoch wurden sie mit verschiedenen Methoden erweitert. Unter anderem besitzen sie spezielle `Creator` Objekte für die Erstellung von Elementen und Methoden zur Feststellung der beinhalteten Elemente und deren Reihenfolge.
Die Klassen `DndGrid`, `DndList`, `DndFreeSource` und `DndSalad` leiten von dieser Basisklasse ab. Im Fall der nicht strukturgebundenen Drag and Drop Zuordnung wurde eine Trennung zwischen Source (`DndFreeSource`) und den verschiedenen Targets (`DndFreeTarget`) vorgenommen. So können Target Container in Tabellen oder auch in Texte integriert werden. Es wird hier lediglich einem Target der übergeordnete Source Container übergeben, der als Drag and Drop *Controller* fungiert und über die Targets wacht.

Die folgenden Abbildungen zeigen die jeweiligen Darstellungen der Zuordnungsaufgaben und die verwendeten Wikisyntaxen, um diese zu erzeugen:

Zuordnung von verschiedenen Medien (in Tabellenform):
JS Widget: `DndGrid`

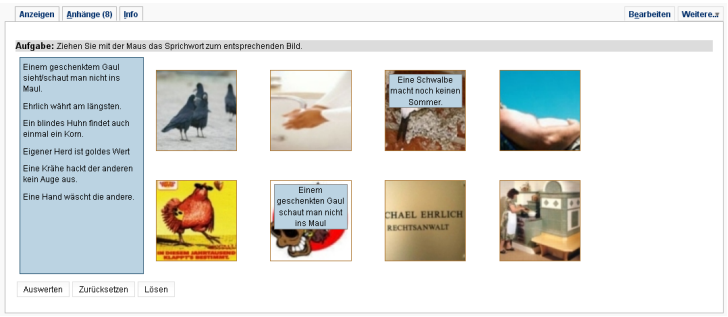


Abbildung 6.11: Die Darstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Texten.

```
[{SpDnd format=text
|
quelle='Eigener Herd ist goldes Wert.'
zielBild='Images/p1.jpg'
|
quelle='Einem geschenkten Gaul schaut man nicht ins Maul.'
zielBild='Images/p2.jpg'
|
[...]
}]
[{SpButtons}]
```

Listing 6.14: Syntax für die Erstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Texten.

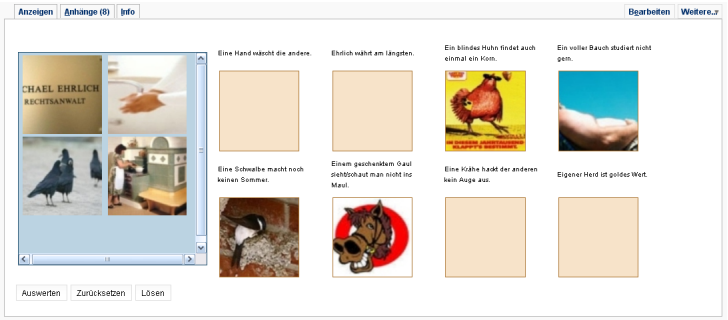


Abbildung 6.12: Die Darstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Bildern.

```
[{SpDnd format=image
|
quelle='Images/p1.jpg'
zielText='Eigener Herd ist goldes Wert.'
|
quelle='Images/p2.jpg'
zielText='Ein voller Bauch studiert nicht gern.'
|
[...]
}]
[{SpButtons}]
```

Listing 6.15: Syntax für die Erstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Bildern.

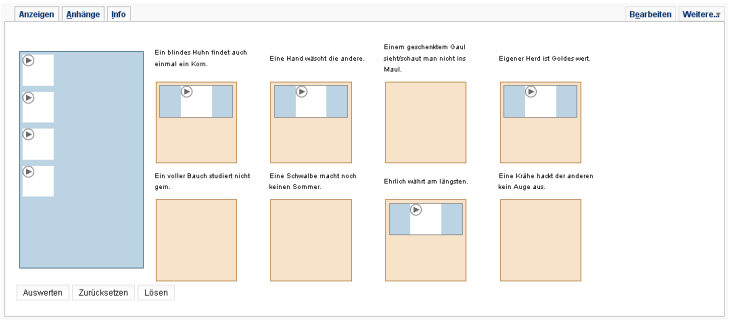


Abbildung 6.13: Die Darstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Audioelementen.

```
[{SpDnd format=audio
|
quelle='Audios/audio1.mp3'
zielText='Eigener Herd ist Goldes wert.'
|
quelle='Audios/audio1.mp3'
zielText='Ein voller Bauch studiert nicht gern.'
|
[...]
}]
[{SpButtons}]
```

Listing 6.16: Syntax für die Erstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Audioelementen.

Zuordnung von Texten (in Listenform): JS Widget: `DndList`

Abbildung 6.14: Die Darstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Texten in Listenform.

```
[{SpDndListe
|
quelle='Es ist wichtig einen Herd zu haben :).'
zielText='Eigener Herd ist Goldes wert.'
|
quelle='Ehrlichkeit ist auf Dauer am besten.'
zielText='Ehrlich währt am längsten.'
|
}]
[{SpButtons}]
```

Listing 6.17: Syntax für die Erstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Texten in Listenform.

Zuordnung von Texten (ungebundene Form): JS Widget: `DndFreeSource` und `DndFreeTarget`

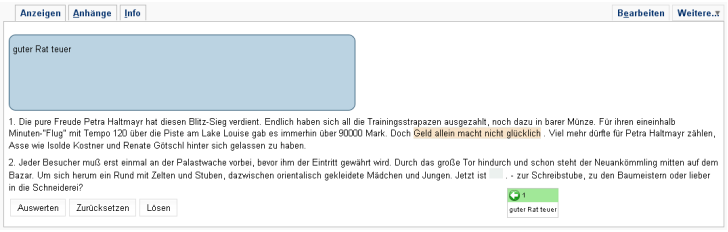


Abbildung 6.15: Die Darstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Texten in ungebundener Form.

```
[{SpDndQuelle idQuelle='quelle1'
|
inhalte=('Geld allein macht nicht glücklich';'guter Rat teuer')
|
}]

1. Die pure Freude Petra [...]. Doch [{SpDndZiel idQuelle='quelle1' inhalt='Geld allein macht nicht glücklich'}]. Viel
     [...] hinter sich gelassen zu haben.

2. Jeder Besucher muß erst einmal an der Palastwache [...] Jetzt ist [{SpDndZiel idQuelle='quelle1' inhalt='guter Rat teuer
    '}]. - zur Schreibstube, zu den Baumeistern oder lieber in die Schneiderei?

[{SpButtons}]
```

Listing 6.18: Syntax für die Erstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Texten in ungebundener Form.

Zuordnung von Textteilen (Salatform): JS Widget: `DndSalad`

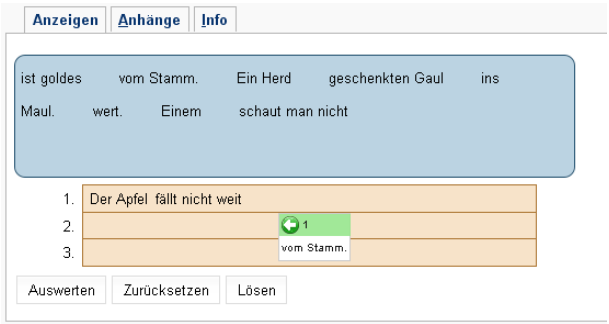


Abbildung 6.16: Die Darstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Textteilen in Salatform.

```
[{SpDndSalat
|
inhalt=('Einem';'geschenkten Gaul';'schaut man nicht';'ins Maul.')
|
inhalt=('Der Apfel';'fällt nicht weit';'vom Stamm.')
|
inhalt=('Ein Herd';'ist goldes';'wert.')
|
}]
[{SpButtons}]
```

Listing 6.19: Syntax für die Erstellung einer Drag and Drop-Aufgabe mit der Zuordnung von Textteilen in Salatform.

6.3.5 Textfeld

Textaufgaben beinhalten in ihrer Interaktion eine Kommunikation zwischen dem Server und dem Client. Dieser Unterschied zu den zuvor genannten Interaktionen macht sie zur Sonderheit.

Der Kandidat produziert einen Text, der nach Abschluss über eine AJAX Kommunikation an den Server gesendet wird. Hierzu steht eine Java Server Page SaveItemResponse.jsp zur Verfügung, die weiters die JSON und Kontext Parameter ausliest und entsprechend dem JSPWiki *Model* übergibt. Das *Model* speichert die Informationen über Data Access Objects (DAO) in der Datenbank. Die Abbildung 6.17 zeigt einen Teilausschnitt aus der Datenbankstruktur der Applikation.

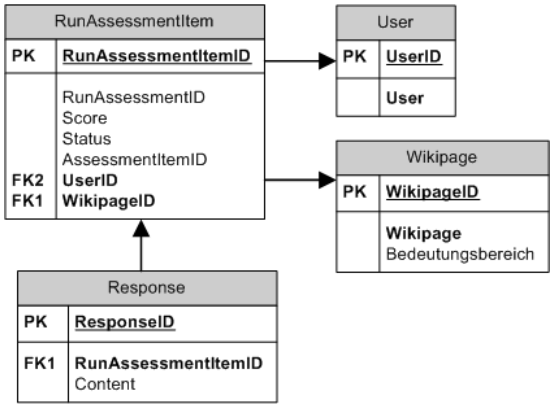


Abbildung 6.17: Verantwortliche Datenbankstruktur für die Verwendung von Textfeldern.

Die Durchführung einer Aufgabe wird in der Tabelle `RunAssessmentItem` abgebildet. Dabei werden die Wikiseite, die die Aufgabe beinhaltet, und der derzeitige Benutzer gespeichert. Die Wikiseite übernimmt zugleich die Identifikationsfunktion der Aufgabe. Die Textantwort wird in der Tabelle `Response` persistiert, dabei bleibt die Möglichkeit offen, mehrere Antworten zu einer Aufgabe abzugeben. Zur Zeit wird aber nur eine 1:1 Beziehung verwendet. Die zentralen Tabellen `User` und `Wikipage` werden auch für weitere Aspekte verwendet, die später noch näher beschrieben werden.
Die Anzeige dieser produzierten Texte wird unter dem Punkt 6.4 erläutert.
Die folgende Abbildung zeigt die Darstellung des Textfeldes in einer Textaufgabe und die verwendete Wikisyntax, um diese zu erzeugen:

JS Widget: `TextArea`

Abbildung 6.18: Die Darstellung einer Textfeld Aufgabe.

```
1 [{SpAnleitung aufgabe='Was brauchen Sie, um glücklich zu werden? Schreiben Sie ein paar Sätze dazu.'}]\\
2 [{SpTextfeld}]
3 [{SpButtons}]
```

Listing 6.20: Syntax für die Erstellung einer Textfeld Aufgabe.

6.3.6 Memory

Die Memory Klasse ist unabhängig von den anderen Aufgaben, da sie ausschließlich eine Interaktion darstellt, ohne Punkte zu vergeben. Ein Memory Objekt kann mit einer verschiedenen Anzahl von Feldern initialisiert werden. Somit kann der Schwierigkeitsgrad durch den Instruktor variiert werden. Eine weitere Eigenschaft von Memory ist die Verwendung von unterschiedlichen Zuordnungsmedien. Das heißt, dass sowohl zwei Textelemente als auch ein Text- und ein Bildelement als Memory Paar gültig sind. Die unterschiedlichen Medien und deren Inhalt werden durch den Instruktor bestimmt.
Die folgende Abbildung zeigt die Darstellung eines Memoryspiels.

JS Widget: `Memory`

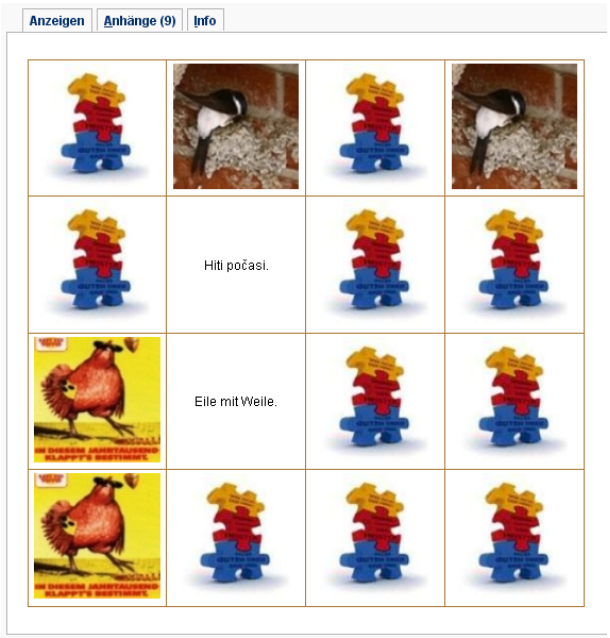


Abbildung 6.19: Die Darstellung einer Memory Aufgabe.

```
1
2  [{SpMemory deckblatt='Images/puzzle.jpg'
3  |
4  inhaltA='Images/p1.jpg'
5  formatA='image'
6  inhaltB='Images/p2.jpg'
7  formatB='image'
8  |
9  inhaltA='Eile mit Weile.'
10 formatA='text'
11 inhaltB='Hiti pocasi.'
12 formatB='text'
13 |
14 inhaltA='Images/p3.jpg'
15 formatA='image'
16 |
17 inhaltA='Images/p4.jpg'
18 formatA='image'
19 inhaltB='Eigener Herd ist goldes wert.'
20 formatB='text'
21 |
22 [...]
23 }]
```

Listing 6.21: Syntax für die Erstellung einer Memory Aufgabe.

6.3.7 Allgemeine Aufgaben Plugins

Neben den bereits erwähnten Interaktionselementen einer Aufgabe werden auch zusätzliche Elemente für statische Informationen in einer Aufgabenseite angefordert. Durch diese Elemente kann entweder die Phase, in der die Aufgabe einzuordnen ist, oder das Niveau, welches einen Indikator für die Komplexität der Aufgabe bildet, dargestellt werden. Ebenso kann eine Anleitung dem Kandidaten zur Verfügung gestellt werden, die zum einen die Vorgehensweise, zum anderen die Aufgabenstellung der Aufgabe beinhaltet.
Die Anforderung dieser Elemente könnte auch mit einer einfachen Wikisyntax umgesetzt werden. Auf Grund der hohen Aufgabenzahl kann somit aber keine Konformität in der Darstellung sichergestellt werden. Durch die Verwendung dieser Plugins wird aber der Visualisierungsprozess zentralisiert. Änderungen können somit an einer einzigen Stelle durchgeführt werden.
Die Syntax der Plugins wird im folgenden beispielhaften Code Listing 6.22 veranschaulicht. Die Plugins werden meist als Kopf einer Aufgabenseite definiert:

```
1 [{SpPhase phase=3}]
2 [{SpNiveau niveau=1}]
3 [{SpAnleitung aufgabe='Ergänzen Sie die fehlenden Buchstaben. Klicken Sie die Lücke an und schreiben Sie das Fehlende in
      das Schreibfeld.'}]
4 [...]}]
```

Listing 6.22: Syntax für die Erstellung allgemeiner Aufgaben Plugins.

6.4 Umsetzung der Übungen

Der Begriff Übung hat im SprichWort Projekt zwei Bedeutungen. Zum einen bezeichnet der Begriff einen Container, der verschiedene Aufgaben (Items) beinhaltet und zum anderen wird auch die Aufgabe selbst als Übung deklariert. Im Fall der Implementation wird die erste Begriffsdefinition verwendet.
Eine Übung stellt somit eine Wikiseite dar, in die verschiedene Aufgaben eingebunden werden können. Zusätzlich kann eine Übung auch Erklärungen, Kommentare oder andere Informationen enthalten.
Um eine Aufgabe einzubinden, gibt es zwei Möglichkeiten. Die erste Option ist das Einfügen der Aufgabe über einen normalen Wiki Link (`[die Aufgabe 1|Aufgabe1Link]`). Der Nachteil hierbei ist, dass beim Öffnen der Aufgabe die Übungsseite verloren geht. Es kann natürlich auch das Zielfenster des Verweises festgelegt werden, um somit dieses Problem zu lösen. Dennoch geht der Eindruck der Kontinuität der RIA verloren. Aus diesem Grund wurde der bekannte Lightbox [1] Style verwendet, um ein modales Fenster auf der Übungsseite zu öffnen. Hierzu wurde auf die Lightbox Implementation des JSPWiki zurückgegriffen, die neben der ursprünglichen Funktionalität, Bilder zu öffnen, auch das Öffnen von Webseiten zur Verfügung stellt. Der Code 6.6 zeigt die Integration einer Aufgabe in eine Übungsseite.
Ein weiterer großer Vorteil dieser Vorgehensweise ist, dass der Kandidat immer den Überblick über die verfügbaren Aufgaben behalten kann. Ebenso kann der Kandidat selbst über den Übungspfad entscheiden und durch die verschiedenen Aufgaben frei navigieren. Die Resultate der Übungsaufgaben werden im Gegensatz zu den Testaufgaben nicht gespeichert. Deshalb ist auch keine zusätzliche Logik für Übungen nötig.
Eine Ausnahme bilden jedoch die Textaufgaben, die ausschließlich zum Kontext einer Übung gehören. Die Vorgangsweise dieser Textaufgaben wurde unter dem Punkt 6.3.5 beschrieben. In diesem Abschnitt wird weiters noch die Funktionalität zur Anzeige von produzierten Texten und deren Kommentierung veranschaulicht. Über das folgende Plugin können in einer Wikiseite die erzeugten Texte der Kandidaten angezeigt werden.

[1] http://en.wikipedia.org/wiki/Lightbox_(JavaScript)

```
[SpTexte wikipage='TextfeldUebung1']
```

Dem Plugin `SpTexte` wird der Name der Aufgabeseite übergeben. Somit können zur Laufzeit alle Textantworten dieser Aufgabe aus der Datenbank gelesen und in die Seite integriert werden.
Um die Kommentierung einer Textaufgabe vorzunehmen, wird die Kommentierungsfunktion des JSPWiki verwendet. Durch diese werden am Ende einer Seite die Bemerkungen der anderen Benutzer eingefügt. Die Abbildung 6.20 zeigt eine mögliche Darstellung.

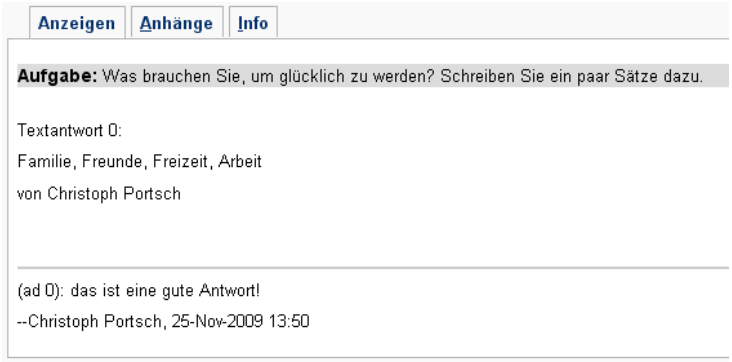


Abbildung 6.20: Die Darstellung von Textantworten zu einer Textfeld Aufgabe.

6.5 Umsetzung der Tests

Ein Test ist im Wesen ähnlich einer Übung. Das heißt, ein Test beinhaltet ebenso verschiedene Aufgaben und fungiert als Container. Der Unterschied liegt dabei aber auf den verwendeten Aufgabentypen. Ein Test kann auf Grund der automatisierten Bewertung nur objektive und standardisierte Aufgabentypen beinhalten. Die Verwendung von Textaufgaben ist hier nicht möglich. Weiters wird ein Test durch ein Zeitlimit beschränkt, das, sobald es überschritten wird, keine weitere Eingabe ermöglicht.
Durch das Zeitlimit und die Punktevergabe muss ein Test eine Logik beinhalten, die diese Funktionalitäten unterstützt. Diese Logik wird durch den `AssessmentController` aus dem Klassendiagramm 6.4 im Assessmentbereich übernommen.
Die Aufgaben in diesem Kontext werden ebenso wie bei Übungen mit dem Prinzip der Lightbox durchgeführt. Um die erreichten Punkte einer Aufgabe zu verarbeiten, wird hier jedoch eine Kommunikation zwischen dem `AssessmentController` und dem `AssessmentItemController`, aus dem erzeugten IFrame der Lightbox, vorausgesetzt. Diese Informationsübermittlung findet über das DOM mit Frame Objekten statt. Auch die Zeitlogik nimmt diesen Weg der Interaktion, um dadurch eine Aufgabe zu sperren und zu schließen.
Für die Erstellung eines Tests sind unterschiedliche Plugins vorhanden. Diese Plugins setzen sich aus `SpAssessmentOpen`, `SpAssessmentItem` und `SpAssessmentClose` zusammen. Das Plugin zum Öffnen eines Tests ist zuständig für die Initialisierung des *Controllers*, des *Models* und der generellen *View*. Weiters wird Markup Code für eine Tabellenstruktur, in der die verschiedenen Aufgaben gelagert werden, eingebunden. Die Aufgaben werden über das Plugin `SpAssessmentItem` eingefügt, die jeweils eine Reihe in der Tabelle bilden. Da ein Aufgabenelement keine Informationen über die Gesamtanzahl der beinhalteten Aufgaben besitzt, wird das Plugin `SpAssessmentClose` verwendet. Dieses Plugin schließt die erzeugte Tabelle und beendet somit den Testbereich. Die Wikisyntax 6.1 zeigt einen exemplarischen Test und dessen Visualisierung unter der Abbildung 6.2.
Die Daten einer Testdurchführung werden persistiert, um später dem Kandidaten Rückschlüsse über den eigenen Lernerfolg zu geben. Die dazu verwendeten Tabellen werden in der Abbildung 6.21 dargestellt.
In der Tabelle `RunAssessment` werden alle durchgeführten Tests abgebildet. Dazu wird die Wikiseite gespeichert, um eine exakte Zuordnung der Ergebnisse auf die Testseite zu erzeugen. Zusätzlich

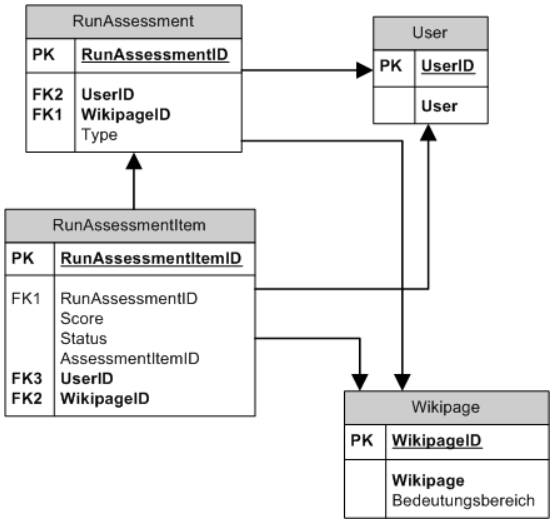


Abbildung 6.21: Ausschnitt aus der Datenbank, der für die Testdurchführung verantwortlich ist.

wird auch die User ID abgelegt, um den Test einem Kandidaten zuzuordnen. Die Tabelle `RunAssessmentItem` stellt alle gelösten Aufgaben mit ihren Ergebnissen dar. Auch hier wird wieder eine exakte Zuordnung auf die Aufgabenseite durchgeführt. Zusätzlich werden auch die erreichten Punkte und die ID der Aufgabe, die unterstützend für den Lightbox Prozess ist, gespeichert.

Die Initalisierung eines Tests nach Aufruf der Testseite kann auf unterschiedliche Arten erfolgen:

1) Der Kandidat führt den Test zum ersten Mal aus.

2) Der Kandidat hat mitten in der Testdurchführung das Fenster geschlossen und möchte fortsetzen.

3) Der Kandidat hat den Test bereits durchgeführt.

Um diese Initialisierungsarten zu unterstützen, werden die Daten je nach Status durch das *Model* aus dem Cookie oder der Datenbank geladen. Folgende Code Snippets 6.23 und 6.24 zeigen die `loadData` Methode aus dem `AssessmentController` und die `loadAssessmentData` Methode aus dem `AssessmentModel`. Hier zeigen sich die Prioritäten der Datenquellen. Sind Daten im Cookie zu finden, so wurde der Test noch nicht abgeschlossen. Sind keine Daten im Cookie vorhanden, sondern nur Daten aus der Datenbank, so wurde der Test bereits abgeschlossen. Gibt es keine verfügbaren Daten, so wird ein neuer Test angelegt.

```
1  loadData: function() {
2      // try to get Data from Model (Cookie or Database)
3      this.assessmentData = this.model.loadAssessmentData(this.getWikipage());
4      if (this.assessmentData.status == this.Status.FINISHED) {
5        this.handleFinishedAssessment();
6      } else if (this.assessmentData.status == this.Status.RUNNING) {
7        this.resumeAssessment();
8      } else if (this.assessmentData.status == this.Status.INIT) {
9        this.startAssessment();
10     }
11 }
```

Listing 6.23: Methode zum Laden der Daten durch den AssessmentController.

```
1  loadAssessmentData: function(){
2      // Status of Assessment is RUNNING
3      var assessmentData = this.loadAssessmentDataFromCookie(this.getWikipage());
4      // Status of Assessment is FINISHED
5      if (assessmentData == null) {
6        assessmentData = this.loadAssessmentDataFromServer(this.getWikipage());
7      }
```

```
8       // Status of Assessment is INIT
9       if (assessmentData == null) {
10        assessmentData = new AssessmentData();
11      }
12      return assessmentData;
13    }
```

Listing 6.24: Methode zum Laden der Daten durch das AssessmentModel.

Die Ergebnisse der absolvierten Tests können entweder einzeln über die entsprechende Testseite oder zusammengefasst in einer Ergebnisseite angezeigt werden. Die Ergebnisseite verwendet hierzu das Plugin `[SpAssessments anzeige='EIGENE']`, das sowohl die eigenen Ergebnisse als auch die Resultate aller anzeigen kann. Zu jedem Testresultat wird zusätzlich ein Link auf den Test angezeigt. Über diesen Verweis gelangt ein Kandidat zum Test, um ihn zu wiederholen.

6.6 Umsetzung der Selbstbewertungsbögen

Die Selbstbewertungsbögen ermöglichen eine Reflexion über das eigene Lernen. Diese Formulare können entweder vor oder nach der Durchführung eines Tests ausgefüllt werden und dienen neben der Motivation zum selbstbewertenden Lernen auch zur Feststellung des Kompetenzniveaus.
Diese Formulare werden in der Datenbank persistiert. Die dazu notwendige Datenstruktur wird im Datenbankdiagramm 6.22 veranschaulicht:

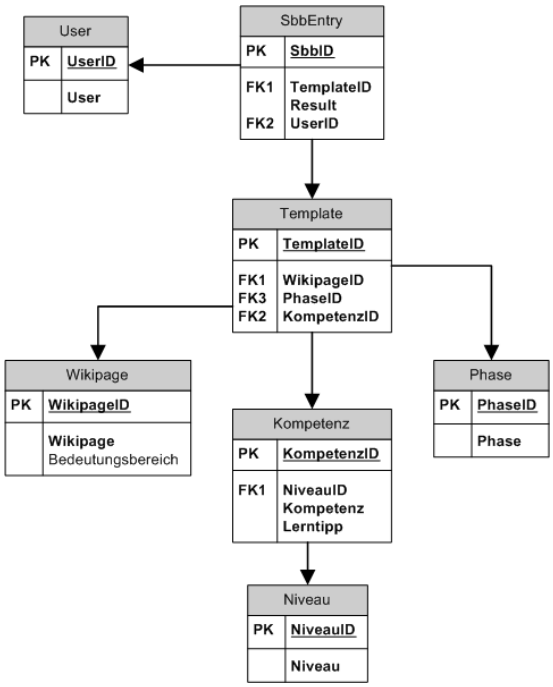


Abbildung 6.22: Ausschnitt aus der Datenbank, der für die Selbstbewertungsbögen verantwortlich ist.

Die Tabelle `Template` ist die zentrale Tabelle für die Verwaltung der unterschiedlichen Selbstbewertungsbögen-Vorlagen. Mit Hilfe der Wikiseite kann der entsprechende Selbstbewertungsbogen identifiziert werden. Dieser Bogen enthält unterschiedliche Phasen (Erkennen, Verstehen, Festigen und Anwenden), die in der Tabelle `Phase` gehalten werden. Jeder Phase können unterschiedliche Kompetenzen aus der Tabelle `Kompetenz` zugeordnet werden, die einem bestimmten Niveau entsprechen. Zusätzlich beinhalten Kompetenzen auch einen Lerntipp, der als Hilfe für den Kandidaten dient, falls ein Kompetenzniveau nicht ausreichend vorhanden ist. Diese genannten Tabellen bilden die statische Datenstruktur und werden einmalig konfiguriert. Die dynamischen Daten werden in der Tabelle `SbbEntry` persistiert.

Hierin werden alle Bewertungsergebnisse eines Kandidaten gespeichert. Dazu wird das Resultat, der dazugehörige Kandidat und der entsprechende Datensatz der Vorlage gespeichert.
Wie bereits erwähnt, werden die statischen Daten der Vorlagen nur einmal erzeugt. Aus diesem Grund ist für diese Aufgabe kein Editor vorgesehen.
Die Erstellung dieses Formulars läuft über ein eigenes Plugin ab. Ähnlich wie bei Aufgaben, Übungen und Tests wird auch bei Selbstbewertungen eine Wikiseite wieder als Container herangezogen. Um so ein Formular zu erstellen, ist lediglich das Plugin `[SpSbb]` in die Seite einzufügen. Der Seitenname muss natürlich zuvor in der `Template` Tabelle konfiguriert werden.
Während der Durchführung eines solchen Formulars kann der Kandidat zwischen unterschiedlichen Selbstbewertungen wählen. Je nach Kompetenzniveau wird dabei ein Lerntipp angezeigt. Folgende Darstellung 6.23 zeigt einen beispielhaften Bewertungsbogen.

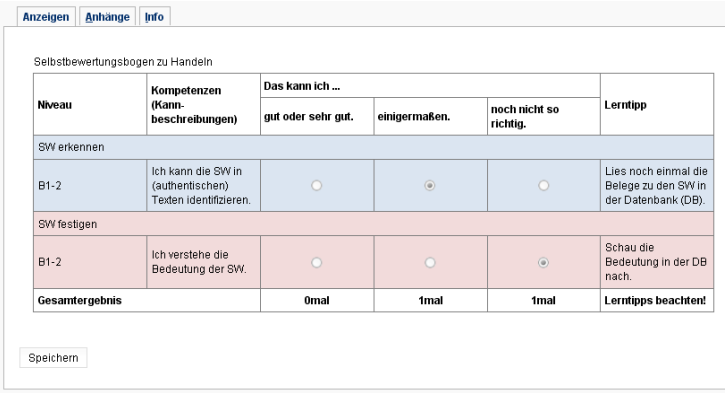


Abbildung 6.23: Darstellung eines exemplarischen Selbstbewertungsbogen zu einem bestimmten Bedeutungsbereich.

Die Ergebnisse können entweder einzeln in der entsprechenden Seite des Selbstbewertungsbogens angezeigt werden oder in einer Ergebnisseite ähnlich der Testergebnisse. Hierzu wird in der Ergebnisseite das Plugin `[SpSbbAnzeige anzeige='ALLE']` eingebunden. Bei diesem Plugin kann wieder zwischen den Resultaten aller und der Anzeige der eigenen Resultate unterschieden werden.

Kapitel 7

Vergleich zwischen der QTI Spezifikation und dem Wikisystem

„Das Vergleichen ist das Ende des Glücks und der Anfang der Unzufriedenheit."

[Søren Kierkegaard (1813 - 1855), dänischer Essayist, Theologe u. Philosoph]

Dieses Kapitel stellt einen Vergleich zwischen dem DeFacto Standard QTI und der proprietären Wiki Lösung auf. Zunächst werden beide Fokusse gegenübergestellt, um eine Positionierung durchzuführen. Des Weiteren wird die XML Syntax von QTI und der Wikitext des Wikisystems untersucht. Im Anschluss werden die Assessment-Elemente wie Aufgaben, Übungen und Tests verglichen. Im Fall der Aufgaben werden zusätzlich die Anforderungen des SprichWort Projektes und deren Erfüllung bearbeitet. Hierunter fallen Thematiken wie Parametrisierung, Punktevergabe und Lösungshilfen. Darauf folgend wird die Erweiterbarkeit beider Formate unterstucht. Zum Schluss werden beide Datenformate hinsichtlich des Produktionsprozesses näher analysiert.

7.1 Fokus

Die Spezifikation QTI bietet ein klar definiertes, umfassendes Informationsmodell zur Erstellung, Verwendung und zum Austausch von assessmentbezogenen Daten. Hierbei wird der Fokus vor allem auf die Interoperabilität und dem Austausch dieser Daten gelegt. Durch den enormen Umfang der Spezifikation wird eine Vielzahl von Aufgabentypen unterstützt. Ebenso werden unterschiedliche Testdurchläufe ermöglicht, die durch verschiedene Konfigurationen angepasst werden können. Bei dieser Spagatlösung, die alle Fachbereiche mit einer einheitlichen Lösung abdecken möchte, wird somit aber auch viel Overhead an Daten und Informationen verwendet. Oft sind jedoch die Anforderungen an die Funktionalität nicht annähernd so komplex und umfassend, wie es QTI zur Verfügung stellt.
Das Wikisystem des SprichWort Projektes zielt vor allem auf die einfache und intuitive Verwendung dieser Technik. So ist es möglich, dass Sachgebietsexperten mit wenig technischem Hintergrundwissen Aufgaben, Übungen und Tests selbst erstellen und verwalten können. Ein weiteres Ziel sind die speziellen Aufgabenformen, ausgerichtet auf den parömiologischen Kontext. Diese Aufgabenformate sind aber zum Großteil auch auf andere Fachgebiete anwendbar. Auch die Interoperabilität wurde nicht außer Acht gelassen, was im nächsten Abschnitt näher erläutert wird.

7.2 Syntax

QTI verwendet für die Datenrepräsentation das XML Format. Die Spezifikation 2.1 setzt sich aus zwei Hauptmodellen zusammen. Zum einen aus dem Testmodell und zum anderen aus dem Itemmodell.

Die kleinste Einheit, das *assessmentItem*, bildet das Kernobjekt und beinhaltet die Frage, die Antwortmöglichkeiten, die richtige Antwort, das Feedback, die Hilfestellungen und die Bewertungsszenarien. Der Testbereich wird durch das *assessmentTest* Objekt abgedeckt, welches über verschiedene Sektionen Aufgaben einbindet und zusätzlich eine konfigurierbare Ablaufsteuerung beinhaltet.
Das Wikisystem unterteilt sich in zwei verschiedene Abstraktionsschichten. Die erste Ebene baut vor allem auf der standardisierten Auszeichnungssprache HTML auf. In der zweiten Abstraktionsschicht wird die Auszeichnungssprache Wikitext, im speziellen die Version des JSPWiki, verwendet. Auch diese Sprache lehnt sich an einer standardisierten Sprache an - Creole[1]. Die Creole Syntax versucht eine einheitliche Wiki Sprache zu definieren, um Interoperabilität zwischen Wikisystemen zu gewährleisten.
Die folgenden Code Snippets 7.1 und 7.2 zeigen den Programmieraufwand für eine Lückentextaufgabe in beiden Sprachen.

```
<?xml version="1.0" encoding="UTF-8"?>
<assessmentItem xmlns="http://www.imsglobal.org/xsd/imsqti_v2p1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1 http://www.imsglobal.org/xsd/imsqti_v2p1.xsd"
    identifier="textEntry" title="Lückenaufgabe" adaptive="false" timeDependent="false">
    <responseDeclaration identifier="RESPONSE_1" cardinality="single" baseType="string">
        <correctResponse>
            <value>ache</value>
        </correctResponse>
        <mapping defaultValue="0.0">
            <mapEntry mapKey="ache" mappedValue="1" />
        </mapping>
    </responseDeclaration>
    <responseDeclaration identifier="RESPONSE_2" cardinality="single" baseType="string">
        <correctResponse>
            <value>eiligt</value>
        </correctResponse>
        <mapping defaultValue="0.0">
            <mapEntry mapKey="eiligt" mappedValue="1" />
        </mapping>
    </responseDeclaration>
    <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float">
        <defaultValue>
            <value>0</value>
        </defaultValue>
    </outcomeDeclaration>
    <itemBody>
        <p>Phase: Festigen</p>
        <p>Niveau: B1-2</p>
        <p>Ergänzen Sie die fehlenden Buchstaben. Klicken Sie die Lücke an und schreiben Sie das Fehlende in das
             Schreibfeld.</p>
        <blockquote>
            <p>R<textEntryInteraction responseIdentifier="RESPONSE_1" expectedLength="4"/> ist süß.</p>
            <p>Der Zweck h<textEntryInteraction responseIdentifier="RESPONSE_2" expectedLength="6"/> die Mittel.</p>
        </blockquote>
    </itemBody>
    <responseProcessing>
        <responseCondition>
            <responseIf>
                <not>
                    <isNull>
                        <variable identifier="RESPONSE_1" />
                    </isNull>
                </not>
                <setOutcomeValue identifier="SCORE">
                    <sum>
                        <variable identifier="SCORE" />
                        <mapResponse identifier="RESPONSE_1" />
                    </sum>
                </setOutcomeValue>
            </responseIf>
        </responseCondition>
        <responseCondition>
            <responseIf>
                <not>
                    <isNull>
                        <variable identifier="RESPONSE_2" />
                    </isNull>
                </not>
                <setOutcomeValue identifier="SCORE">
                    <sum>
                        <variable identifier="SCORE" />
                        <mapResponse identifier="RESPONSE_2" />
                    </sum>
                </setOutcomeValue>
            </responseIf>
        </responseCondition>
```

[1] http://www.wikicreole.org

```
67      </responseProcessing>
68  </assessmentItem>
```

Listing 7.1: Eine Lückentextaufgabe im QTI Format.

```
1  [{SpPhase phase=3}]
2  [{SpNiveau niveau=1}]
3
4  [{SpAnleitung aufgabe='Ergänzen Sie die fehlenden Buchstaben. Klicken Sie die Lücke an und schreiben Sie das Fehlende in
       das Schreibfeld.'}]\\
5
6  R[{SpHotspotLuecke | korrekterInhalt='ache' | }] ist süß.\\
7  Der Zweck h[{SpHotspotLuecke|korrekterInhalt='eiligt'|}] die Mittel.\\
8
9  [{SpButtons}]
```

Listing 7.2: Eine Lückentextaufgabe im Wiki Text Format.

Im QTI Beispiel werden zunächst zwei Response Variablen deklariert. Jede dieser Response Variablen wird einer Lücke in der Aufgabe zugeordnet. Durch ein spezielles Mapping in der Response Deklaration können unterschiedliche Punkte bei leicht variierenden Textantworten gegeben werden. In der Outcome Deklaration wird die Ergebnisvariable der Aufgabe deklariert, die am Schluss in der Verarbeitung der Antwort gesetzt wird. Im Item Body werden die Interaktionen mit dem Inhalt der Aufgabe verwoben. Im Schlussteil der Aufgabe wird das *Response Processing* durchgeführt. Hierin wird die Ergebnisvariable, je nach den Werten der Response Variablen, berechnet. Bei diesem Prozess ist schnell zu erkennen, dass die Codelänge enorm mit der Anzahl der Lücken Positionen im Text anwächst. Folglich ist auch die Auslagerung dieses Prozesses nur mit einer einheitlichen Anzahl von Interaktionen gegeben, da der Prozess in seinem Wesen variiert.
Das Wikitext Beispiel baut auf der Plugin Technik des JSPWiki auf. Zunächst werden Orientierungshilfen für den Kandidaten gesetzt, die im Gegensatz zu QTI Konformität unter den verschiedenen Aufgaben versichern. Weiters wird der Stimulus definiert, der auch hier wieder einer einheitlichen Verwendung unterliegt. Die Interaktionen werden, ähnlich wie bei QTI, in den Textbereich integriert. Hierbei wird nicht zwischen Head- und Bodybereich unterschieden, da im Prinzip das Wikisystem diese Zuordnung der Head- und Bodyinformationen übernimmt. Die Aufgabe stellt, im Unterschied zu QTI, keine Variablen Deklarationen zur Verfügung. Die Variablen für den Auswertungsprozess werden intern durch das System erstellt bzw. verwaltet und bleiben somit dem Benutzer verborgen. Der Instruktor kann lediglich Eingriff auf die Parameter der Interaktion nehmen. Das Gleiche wie für die Variablen gilt auch für den Verarbeitungsprozess. Auch dieser bleibt im Verborgenen und muss nicht jedes Mal durch den Instruktor konfiguriert werden. Im Anschluss zur Aufgabenstellung werden die Bedienungselemente hinzugefügt, die je nach Kontext unterschiedliche Funktionalitäten zur Verfügung stellen.
Zusammenfassend ist zu bemerken, dass zwar QTI mit seiner Funktionalität das Wikisystem übertrifft, jedoch der Kontext nicht vergessen werden darf, in dem diese Applikation verwendet wird. Auch der Kostenfaktor, mit dem eigentlich QTI beworben wird, ist durch die geringe Komplexität auf Seiten des Wikisystems.

7.3 Übungen und Tests

In Hinblick auf die Anforderungen des SprichWort Projektes bietet QTI die Möglichkeit zur Differenzierung zwischen Übungen und Tests. Das Testmodell liefert hierbei genügend Funktionalität, um beide Typen umzusetzen. Hervorragend ist vor allem die Unterteilung in Testsektionen, die Gruppierungen von Aufgaben erlaubt. Weiters sind auch die zeitlichen Konfigurationsmöglichkeiten zu bemerken, die ebenso auf die einzelnen Sektionen angewendet werden können. QTI unterstützt unterschiedliche Navigationsformen, die entweder einen linearen oder nicht-linearen Durchlauf der Aufgaben bieten. Zusätzlich gibt es auch die Option von adaptiven Tests, die über Vorbedingungen und vorherige Testergebnisse eine individuell angepasste Testsequenz zur Laufzeit bereitsstellen. Auch der Abgabemodus ist in QTI

frei konfigurierbar. Hier kann zwischen einer simultanen und einer individuellen Abagbe der Aufgaben gewählt werden.
Im Vergleich dazu fällt die Funktionalität des Wikisystems etwas mager aus. Die Applikation wurde jedoch speziell auf die Anforderungen des Projektes abgestimmt. So wird zum Beispiel nicht zwischen verschiedenen Sektionen unterschieden, da dies zusätzliche Komplexität in die Anwendung bringt. Die zeitlichen Einstellungen gelten somit für den gesamten Test. Der Abgabemodus wurde einheitlich gestaltet und ähnelt dem individuellen Modus von QTI. Der Unterschied besteht jedoch darin, dass die Ergebnisdaten erst am Ende eines Testdurchlaufes übertragen werden. Die Versuche, im Fall von QTI auch *attempts* genannt, sind beschränkt auf die Anzahl eins. Dieser Umstand rührt daher, dass bereits während der Abgabe Fehlerkennzeichnungen erfolgen und der Kandidat somit sofort Feedback erhält. Anzumerken ist, dass die Tests des Projektes ein Teil der SprichWort-Übungen sind und somit auch mehr den Charakter einer Übung besitzen. Das heißt zum Beispiel, dass Tests wiederholt werden können. Dennoch wird zwischen Übungen und Tests getrennt. Tests ermöglichen dem Benutzer, die Ergebnisse zu speichern. Übungen erfüllen diese Funktionalität nicht, mit Ausnahme der Textaufgaben. Beide Typen beinhalten jedoch eine bestimmte Aufgabenkollektion zu einem speziellen Sachgebiet.

Die folgenden Code Snippets 7.3 und 7.4 zeigen die Datenrepräsentation eines Tests in QTI und im Wikisystem.

```
<?xml version="1.0" encoding="UTF-8" ?>
<assessmentTest
  xmlns="http://www.imsglobal.org/xsd/imsqti_v2p1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1 http://www.imsglobal.org/xsd/imsqti_v2p1.xsd"
  identifier="Test01"
  title="NI - timeLimits - test">
  <timeLimits maxTime="10.0"/>
  <testPart identifier="P01" navigationMode="nonlinear" submissionMode="individual">
    <assessmentSection identifier="S01" title="Section 1" visible="true">
      <assessmentItemRef identifier="item01" href="item01.xml"/>
      <assessmentItemRef identifier="item02" href="item02.xml"/>
      <assessmentItemRef identifier="item03" href="item03.xml"/>
    </assessmentSection>
  </testPart>
</assessmentTest>
```

Listing 7.3: Datenrepräsenation eines Tests im QTI Format.

```
[{SpAssessmentOpen zeit=600}]
[{SpAssessmentItem name='Aufgabe 1' wikipage='Aufgabe1'}]
[{SpAssessmentItem name='Aufgabe 2' wikipage='Aufgabe1'}]
[{SpAssessmentItem name='Aufgabe 3' wikipage='Aufgabe1'}]
[{SpAssessmentClose}]

[{SpButtons}]
```

Listing 7.4: Datenrepräsenation eines Tests im Wiki Text Format.

Beide Codebeispiele zeigen die Datenrepräsenation eines Tests. QTI bindet als Aufgabe bereits vorhandene Item Daten ein. Im Wikisystem hingegen wird auf Wikipages zurückgegriffen, die die Container für Aufgaben bilden. Beide Varianten sind einfach zu erstellen. Zu erwähnen ist jedoch, dass die QTI Spezifikation 2.1, die zum ersten Mal die Verwendung von Tests einführt, nur als „Draft“ erhältlich ist und noch nicht freigegeben wurde.

7.4 Aufgaben

Anhand der Codebeispiele 7.1 und 7.2 wurde bereits eine Gegenüberstellung beider Verwendungen vorgenommen. Herausstechend ist hier vor allem die Kapselung der Interaktions- und Bewertungslogik durch das Wikisystem. Bei der Erstellung einer Aufgabe sind somit nur die wesentlichen Aufgabeninhalte durch den Instruktor anzugeben.
Ein weiterer wichtiger Unterschied zwischen beiden Formaten ist die Verlagerung der Renderinglogik

in Richtung der *Player* Implementation im Fall des Wikisystems. Das heißt, dass der Instruktor sich nicht mehr um Positionierungen von Elementen kümmern muss, denn ein besonderer Nachteil der QTI Spezifikation ist die Vermischung von Inhalt und Darstellung. Im Besonderen betrifft dieser Umstand die zurzeit finale Version 1.2, wie unter Abschnitt 3.10 näher erläutert wurde. Die neue Version 2.0+ reagiert auf diese Problematik und ermöglicht zusätzlich die Verwendung von Cascading Style Sheets. Trotz dieser Überarbeitung bestehen weiterhin Schwierigkeiten. Dies gilt vor allem für den Aufgabentyp des Textsalates. Bei dieser Zuordnungsaufgabe müssen weithin die genauen Koordinaten der Textelemente angegeben werden. Dieser Umstand ist nicht nur bei der Erstellung ärgerlich, sondern erzeugt auch einen hohen Änderungsaufwand. Sobald der Inhalt einer Aufgabe verändert wird, muss sich auch die Präsentation ändern. (Vgl. [Helic, 2006, S. 56])
Hinsichtlich der Aufgabenvielfalt ist zu erwähnen, dass QTI eine Vielzahl an Interaktionstypen unterstützt. Dennoch werden für den Kontext der Linguistik keine Sprachspiele wie Memory zur Verfügung gestellt. Diese Gegebenheit ist natürlich verständlich, da sie eine spezielle Anforderung aus diesem Fachbereich ist. Trotzdem zielt QTI darauf ab, interdisziplinär eingesetzt zu werden.

7.5 Parametrisierung und Randomisierung

Der Begriff Parametrisierung wird in diesem Fall für die Fähigkeit verwendet, unterschiedliche Parameter für eine Eigenschaft einer Interaktion einzustellen. Diese Parameter werden im Anschluss durch das Zufallsprinzip während der Initalisierungsphase der Aufgabe ausgewählt und verwendet. Dieser Prozess könnte auch als Templating bezeichnet werden.
Die Spezifikation QTI bietet für diese Fähigkeit einen Template Prozess an, der bei der Initalisierung Template Variablen verarbeitet und somit eine abwechslungsreiche Aufgabe erzeugt. In diesem Prozess können auch Abhängigkeiten unter den vorhandenen Template Variablen verarbeitet werden.
Das Wikisystem bietet eine ähnliche Funktionalität mit Hilfe von Parametergruppen an. Die Parameter einer Gruppe bilden die vollständige Information zur Darstellung einer Interaktionsvariante. Dabei verzichtet das System auf Abhängigkeiten zwischen Parametern anderer Interaktionen. Vorteile verschafft diese Fähigkeit vor allem im Bereich von Hotspots, da in einer Darstellungsvariante zum Beispiel ein Hotspot gezeigt wird und in einer anderen Darstellung dieser ausbleibt.
Die Randomisierung wird nicht nur für das Templating verwendet, sondern bildet auch das Schlagwort für die eigentliche Darstellung einer Interaktion. Mit anderen Worten heißt das, dass Inhalte in einer zufälligen Reihung visualisiert werden.
QTI verwendet hierzu den Tag *shuffle*, der je nach Interaktion platziert werden kann. Das Wikisystem hingegen verzichtet auf diese Konfiguration und versetzt automatisch alle Interaktionsinhalte in diese Präsentationsform.

7.6 Punkteverarbeitung

QTI ermöglicht durch sein spezielles Verarbeitungsverfahren der Antworten die Verwendung von unterschiedlichen Punkten. Das bedeutet, dass sowohl die Punkte einer Interaktion je nach Antwort variieren können als auch die Punkte verschiedener Interaktionen unterschiedlich sein können.
Das Wikisystem hingegen arbeitet mit Prozenten, somit sind in jeder Aufgabe 100% erreichbar. Die Interaktionselemente in einer Aufgabe teilen sich diesen Prozentsatz. Somit wird in einer Aufgabe mit fünf Interaktionselementen, von denen vier richtig gelöst werden, ein Prozentsatz von 80% erreicht. Auf die Gesamtprozentzahl eines Tests wird verzichtet, da durch die Verwendung von Prozenten ein ungleichmäßiges Verhältnis von gleichen Interaktionen aus unterschiedlichen Aufgaben entsteht.

7.7 Hinweise zu Aufgaben

QTI präsentiert Hinweise in Form von modalem Feedback. Die Spezifkation bezeichnet als Feedback jede Information, die dem Kandidaten aufgrund von Outcome Variablen zur Verfügung gestellt wird. Wie bereits unter Abschnitt 4.7.1.7 untersucht worden ist, widerspricht sich die Spezifikation in Bezug auf Hinweise und die dafür vorgesehene Technik des modalen Feedbacks.
Das Wikisystem stellt für die meisten Interaktionen einen Hinweis-Parameter zur Verfügung. Dieser Parameter kann je nach Anforderung gefüllt werden und bezieht sich jeweils auf eine Parametergruppe. Das bedeutet, dass bei Zuordnungsübungen, in denen mehrere Parametergruppen gleichzeitig visualisiert werden, die Hinweise separat für jede Gruppe sichtbar sind.

7.8 Lösungen zu Aufgaben

In der Spezifikation QTI unterliegt jedes Item einem Lebenszyklus (siehe Abschnitt 3.5.1.2), der durch eine Item Session repräsentiert wird. In dieser Session werden die unterschiedlichen Stati einer Aufgabe festgehalten. Für die Anzeige von Lösungen kann die Item Session in den *solution* Status übergeführt werden. In diesem Zustand werden alle richtigen Lösungen angezeigt, die in der *Response Declaration* definiert wurden.
Das Wikisystem verwendet keine Item Session, da dies auf Grund der Anforderung nicht nötig ist. Trotzdem können alle Interaktionselemente in einer Aufgabe gelöst werden. Diese Funktionalität wird jedoch nur im Kontext der Übungen unterstützt, da diese einen formativen Charakter besitzen.

7.9 Erweiterung

Der Erweiterungsaspekt wird hinsichtlich der Anforderungen eher kleingeschrieben.
Die Spezifikation stellt im Bereich der Interaktionen eine *customInteraction* zur Verfügung, die für Erweiterungen verwendet werden kann. In der Arbeit von [Pfähler und Holzer, 2009] wird die Umsetzung eines neuen Aufgabentyps erforscht. Zu bemerken ist hier, dass solche individuellen Lösungen nur in einer proprietären *Player* Implementation dargestellt werden und die Interoperabilität gefährden.
Die Logik der Aufgaben Interaktionen befindet sich in der Wiki Applikation auf der Seite des Clients und wurde mit DOM Scripting umgesetzt. Durch die Verwendung einer Schnittstelle können neue Interaktionen schnell integriert werden. Um die neue Interaktion dem Benutzer zur Verfügung zu stellen, muss lediglich ein neues Plugin integriert werden.

7.10 Erstellungsprozess

Der geforderte Erstellungsprozess unterteilt den Lebenszyklus einer Aufgabe, einer Übung oder eines Tests in verschiedene Phasen.
QTI ab der Version 2.0 verwendet hierzu die IMS Learning Resource Meta-Data Spezifikation, die auf dem IEEE 1484.12.1-2002 Standard for Learning Object Meta-data (LOM)[2] aufbaut und diese um assessmentspezifische Daten erweitert. Die Meta-Data Spezifikation erlaubt die Beschreibung von QTI Datensätzen in verschiedenen Kategorien. Diese Kategorien setzen sich aus „General“, „Lifecycle“, „Meta-Metadata“, „Technical“, „Educational“, „Rights“, „Relation“, „Annotation“ und „Classification“ zusammen. Die relevante Kategorie für die Unterstützung des Produktionsprozess ist der „Lifecycle“, wobei der Parameter „version“ die exakte Anlaufstelle bildet.
Im Wikisystem wird diese Aufgabe durch die Verwendung von Links gelöst. Diese unsichtbaren Links

[2] http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

können im Bearbeitungsmodus durch den Betrachter auch als Metadaten interpretiert werden. Ein solcher Link verweist auf eine zentrale Sammelseite. Für die Phasen im Erstellungsprozess existieren unterschiedliche Sammelseiten, die jeweils alle Aufgaben oder auch Tests dieser Phase visualisieren.

7.11 Konklusion

Es wird deutlich, wie umfangreich und ausgeklügelt die QTI Spezifikation ist. Vor allem dem Ziel, ein Assessment Datenformat für die meisten Fachbereiche zu liefern, wird die Spezifikation gerecht. Dennoch hat auch das seine Kehrseite, denn durch das Bestreben für alle Eventualitäten und Anforderungen gewappnet zu sein, wird die Spezifikation globig und komplex. Dieses Faktum wird noch dazu von den Implementoren unterstrichen, die von der neuen Version 2.1 spezielle Profile verlangen. Diese Profile sollen nur Teile der Spezifikation umfassen, die sich mit den entsprechenden Bedürfnissen der einzelnen Implementoren decken. Doch wo führt diese Forderung hin? Wieder zu proprietären Lösungen.
Das Wikisystem verfolgt einen anderen Weg und legt den Fokus auf die Bedürfnisse einer speziellen Anwenderschicht. Einfachheit und Verwendbarkeit sind die Schlagworte dieser Lösung. Zusätzlich wird durch die Verwendung von weitverbreiteten Technologiestandards und Datenformaten aber auch Interoperabilität gewährleistet. Erweiterungen können durch die Verwendung von klar definierten Schnittstellen mit wenig Aufwand umgesetzt werden.
QTI wurde in diesem Kapitel nur hinsichtlich seiner Spezifikation untersucht. Zu erwähnen sind aber auch die Implementationen selbst. Dabei ist zu bemerken, dass es nur wenige freiverfügbare Werkzeuge gibt. Alle getesteten Tools befanden sich noch im Entwicklungsstadium und unterstützen nur wenige Teile der Spezifikation. Somit ist vor allem die Spezifikation Version 2.0 noch nicht praxistauglich.

Kapitel 8

Ausblick

> *„It's very easy to predict the future. People do it all the time. What you can't do, is get it right."*
>
> [Don Norman, The Front Desk, BBC Video, 1995.]

In dieser Arbeit muss in dreierlei Hinsicht ein Ausblick gegeben werden.
Zunächst möchte der Autor, die etwas trübere Perspektive der Spezifikation QTI diskutieren. Laut [Piotrowski und Fenske, 2007, S. 195f] sind die Schwächen von QTI letztlich in der Designphilosophie begründet. So ist ein komplett neuer Ansatz dieser Spezifikation nötig. Viele Probleme rühren auch daher, dass dieser Standard nicht aus dem praktischen Einsatz entwickelt wurde, sondern durch ein Komitee „am grünen Tisch". Der Autor schlägt somit vor, dass zukünftig viel mehr praktische Erfahrungen in den Standard einfließen müssten. [Piotrowski und Fenske, 2007, S. 195f] gehen sogar soweit vorzuschlagen, dass die Anwendergemeinschaft auf Basis ihrer Praxiserfahrungen selbst ein Datenformat für Tests entwickeln sollte.
Das Wikisystem zeigt, wie einfach Testentwicklung sein kann, was auch durch die Anwender dieser Applikation bestätigt wurde. Wenn es in Zukunft zu einem neuen und „anwendbaren" Standard im Bereich Assessments kommt, ist es sicher erstrebenswert, auf diesem aufzubauen. So könnte zum Beispiel die Wikisyntax eine höhere Abstraktionsstufe des darunterliegenden Testformates bilden. Die Lösung dieser Arbeit fokusiert sehr auf Self-, Peer- und Group Assessments und stellt herkömmliche Tests und ihre geforderten Sicherheitsmaßnahmen hinten an. Deshalb sollte das System in diese Richtung noch erweitert werden, um sein Einsatzgebiet zu vergrößern.
Die Etablierung des SprichWort Projektes und die Akzeptanz der Lern Community in Hinsicht auf die Übungen wird sich erst in nächster Zeit zeigen. Auch hier möchte der Autor eine Anregung geben. Das SprichWort Team sollte den Weg „student-driven" in Richtung „assessment-driven" weiterschreiten. Diese Idee stammt von [Brusilovsky und Sosnovsky, 2005a, S. 251] und schlägt die Verwendung der online verfügbaren Übungen in üblichen Papier- und Bleistifttests vor. Dadurch könnte die Motivation vielfach gesteigert werden, da die Plattform Vorteile für Studenten bildet.

Kapitel 9

Konklusion

„Sobald man in einer Sache Meister geworden ist, soll man in einer neuen Schüler werden.“

[Gerhart Hauptmann, (1862 - 1946), deutscher Schriftsteller]

Die didaktische Methodik der Online Assessment erfreut sich immer größerer Beliebtheit unter den Pädagogen. Diese Assessments dienen heutzutage nicht mehr ausschließlich der Notengebung, sondern spielen einen vitalen Part im Lernprozess.
Durch unterschiedliche Assessment Arten wie Self-, Peer- und Group Assessments wird die Beteiligung am Lernprozess erhöht. Zudem werden auch die Komponente der sozialen Interaktion gestärkt und individuelle Feedbackmöglichkeiten erreicht.
Auch Mechanismen wie Parametrisierung spielen eine wichtige Rolle in diesem Bereich. So wirkt eine solche Funktionalität nicht nur dem hohen Generierungsaufwand von Aufgaben entgegen, sondern bringt auch Abwechslung, Motivation und Sicherheit in die Durchführung von Assessments und steigert somit die Lernperformanz.
Schlagworte in Bezug auf Assessments sind weiters auch Interoperabilität sowie Austauschbarkeit und Wiederverwendbarkeit. Die assessmentbezogenen Daten sollten unabhängig vom Prüfungssystem sein. Die IMS Spezifikation QTI ist die Antwort auf diese Forderung und bietet hierfür ein sehr durchdachtes und umfassendes Informationsmodell an. QTI verspricht eine verbesserte Interoperabilität, eine Anzahlssteigerung der am Markt befindlichen Lösungen, schnellere Innovationen und reduzierte Kosten.
Dieser DeFacto Standard weist jedoch einige Schwächen auf. Zu diesen Schwächen gehört vor allem die Versionspolitik von IMS. So wurde im Frühjahr 2009 die Version 2.1 als Public Draft zurückgezogen. Dieser Rückzug wurde mittlerweile zwar wieder aufgehoben, dennoch empfiehlt IMS noch immer die Verwendung der 2003 releasten Version 1.2.1. Dieser Umstand verunsicherte vor allem die Implementoren, denen auch eine geplante, komplett neue Version 2.2 mitgeteilt wurde.
Eine weitere Schwierigkeit bildet die Vermischungsproblematik von Inhalt, Form und Logik, wobei die Version 2.0+ auf diesen Aspekt reagierte und das Format neu überarbeitet wurde. Dennoch beinhaltet auch das neue Format weiterhin eine unklare Trennung zwischen Inhalt und Logik.
Die Komplexität, der Umfang und die Versionspolitik der gesamten Spezifikation verursachen weiters auch einen Mangel an Implementationen. Werkzeuge, die bereits vorhanden sind, befinden sich meist in einem Entwicklungsstadium und sind somit für die Praxis untauglich.
Es bleibt somit nichts anderes übrig, als mit einem herkömmlichen Editor diese assessmentbezogenen Daten zu erstellen. Doch diese Vorgehensweise ist auf Grund der hohen Komplexität keinem Pädagogen zumutbar. Zu diesen bereits genannten Nachteilen kommt noch die Tatsache, dass die Wiederverwendung von Assessmentinhalten einen geringeren Zuspruch fand, als am Anfang angenommen wurde.
Das System des internationalen SprichWort Projektes geht einen anderen Weg und setzt auf Einfachheit und Intuitivität. Die Bedürfnisse des Anwenders stehen im Mittelpunkt.

Hauptbausteine hierfür sind das Wikiframework auf der Serverseite und die freie und modulare JavaScript-Bibliothek auf der Clientseite.
Das Wikiframework liefert durch seine Eigenschaften, die Online Bearbeitung von Inhalten, die Verwendung einer Wikisyntax und die Unterstützung nicht-linearer Hypertextstrukturen, dem Anwender die entsprechende Funktionaltität und Einfachheit im Erstellungsprozess.
Die Applikation verbirgt im Gegensatz zu QTI die Form und Logik des Assessments. Die Anwender müssen sich nicht mehr um Punktevergabe oder Visualisierung ihrer Prüfungen kümmern. Somit kann sich ein Pädagoge nur mehr auf den Inhalt seiner Fragen und sein Sachgebiet konzentrieren.

Diese Arbeit ist entstanden im Rahmen des Forschungsprojekts SprichWort (143376-2008-LLP-SIKA2-KA2MP). Das Projekt wird mit Unterstützung der Europäischen Kommission finanziert (Programm für Lebenslanges Lernen / Multilaterale Projekte / Schwerpunktaktivität 2: Sprachen). Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Autor. Die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

Anhang A

QTI-basierte Applikationen

Name	QTI version	Type of tool	Comment
ANGEL Learning Management Suite	2.1	LMS	also supports IMS Common Cartridge
APIS QTIv2 Assessment Engine	2.0 draft	Java library & demo application.	Incomplete. Author recommends using QTITools instead.
AQuRate	2.1	authoring tool	see QTITools
ASDEL	2.1	assessment delivery system	see QTITools
ATutor	1.2, 2.1	LCMS	
Canvas Learning	1.2.1	Authoring tools and SCORM compatible item renderer available as middle-ware solutions.	Creators - Can Studios contributed to the development of the QTI specification. A number of LMS systems used the Canvas Learning Player to achieve compatibility with the Becta learning platform conformance regime. The system is currently being distributed to schools in the UK as a result of this integration work.
CCReader	1.2.1 CC Profile	Common Cartridge Viewer	
Cognero	not stated	Assessment authoring and delivery system	
Content-e	1.2 & 2.0	Professional authoring tool Content-e.	Imports QTI 1.2 and 2.0.
DB Primary	2.0	LMS	
Diploma	1.2, 2.1		export QTI 1.2 & 2.1
Dokeos	1.2 and 2.0	LMS/LCMS	export QTI 1.2 & 2.0 (1.2 disabled by default but available) (supports SCORM 1.2)
Elques	2.1	authoring tool	exports QTI 2.1 and QTI 1.2 (for LMS OLAT only); imports QTI 2.1, Tests from Blackboard and OLAT (kind of QTI 1.2 too)
it's learning	2.1	VLE	import and export questions in QTI 2.1 format
ILIAS	not stated	LMS	supports SCORM 1.2 and SCORM 2004
Lectora	not stated	authoring tool	supports SCORM 1.2 and SCORM 2004
Mathqurate	2.1	authoring tool	see QTITools. Embedded Gecko engine and support for multiple interactions
Moodle	not stated	LCMS	supports adaptive questions
Online Learning And Training	not stated	LCMS	
ONYX	2.1	modular assessment delivery system	open-source, QTI 2.1 import and export, Report Viewer for QTI-Result-Files
OWL Testing Software	not stated	test management system	can import IMS QTI
QTITools	2.1	collection of tools and libraries	
QuestionMark Perception	not stated	authoring tool and delivery system	can export IMS QTI, an online tool provides QTI 1.2 import
Question Writer 2.0 Publisher Edition	1.2	authoring tool	Exports as QTI 1.2 and SCORM 1.2
Question Writer 3.5 Professional	1.2	authoring tool	Exports as QTI 1.2 and SCORM 1.2. Also specific QTI Export for Pearson VUE
Respondus	1.2	authoring tool	QTI export
RM Test Authoring System	2.1	authoring tool	
Sakai	1.2	LMS	
SToMP (Software Teaching of Modular Physics)	2.1	assessment system	mostly unavailable as of July 2008
Studywiz	1.2	Virtual Learning Environment Module	An optional module for creating and assigning QTI v1.2 questions to students. Available as of June 2008
Wimba Create	QTI Lite	authoring tool	only export

Tabelle A.1: Liste der Applikationen, die die Spezifikation IMS QTI unterstützen. [Wikipedia]

Anhang B

Spieletypologie

Spielkategorie	*Spielarten*	*Spielunterarten*	Einsatzalter (Jahre)				
			-6 Vorschulstufe	7-10 Primarstufe	11-14 Sek.-stufe I	14-20 Sek.-stufe II	20-45 Tertiärstufe
Sprachelementspiele	Kartenspiele	• Paare, Terzette, Quartette sammeln	+	+	+	+	+
		• Frage-/Antwortkarten		+	+	+	
		• Aufgabenkarten		+	+	+	+
		• ursprüngliches Kartenspiel					
	Würfelspiele	• Brettspiele mit Würfel	+	+	+	+	+
		• Figuren würfeln, Rechnen	+	+	+		
		• Buchstaben-, Wörter-, … würfeln			+	+	+
	Anlegespiele (Bild-Bild, Bild-Wort, Wort-Wort)	• zweiseitiges Domino	+	+	+	+	+
		• dreiseitiges Domino		+	+	+	+
		• vierseitiges Domino			+	+	+
	Merkspiele	• Memory	+	+	+	+	+
		• Kim-Spiele		+	+	+	(+)
		• Veränderungen finden			+	+	+
		• Kofferpacken			+	+	+
	Lotteriespiele	• Bingo, Lotto		+	+	+	+
		• andere "Glücksspiele"			+	+	+
	Rätsel	• Bilderrätsel		+	+	+	+
		• Rätsel			+	+	+
	Kreuzworträtsel	• Bilderkreuzworträtsel		+	+	+	+
		• K. mit/ohne Erklärungen		+	+	+	+
		• übliche Kreuzworträtsel			+	+	+
	Puzzles	• Text-, Satz-, Informationspuzzle		+	+	+	+
	Buchstaben-, Lautspiele	• Laute identifizieren, variieren	+	+	+	+	
		• Buchstabensalat		+	+	+	+
		• Geheimschriften			+	+	
	Wortspiele	• Buchstabenquadrat		+	+	+	+
		• Wörter sammeln			+	+	
		• Wörterschlangen		+	+	+	+
		• Wortende ist Wortanfang			+	+	+
		• Silbenrätsel			+	+	+
		• Versteckte, falsche Wörter			+	+	+
	Satzspiele	• Kettenspiel	+	+	+	+	+
		• Sätze bilden, formulieren		+	+	+	+
	Textspiele	• Textteile ordnen			+	+	+
		• Texte abwandeln			+	+	+
		• T. weiter schreiben / -spielen			+	+	+
		• Texte aus Stichwörtern bilden			+	+	+
	Nachahmungsspiele	• Pantomime	+	+	+	+	(+)
		• Laut-, Sprechimitation		(+)	(+)	(+)	

Abbildung B.1: Spieletypologie (Ausschnitt aus dem Gesamtrepertoire). [Kacjan, 2008, S. 69f]

Anhang C

JavaScript Frameworks im Vergleich

Abbildung C.1: JavaScript Framework Task Matrix. [Higgins]

Anhang D

CD-ROM

Literaturverzeichnis

[Bailey 2005] BAILEY, Warwick: *What Is... IMS Simple Sequencing?* Institute for Educational Cybernetics (IEC), University of Bolton. September 2005. – URL http://www.cetis.ac.uk/lib/media/ss2brief.pdf. – Zugriffsdatum: 08.09.2009 (Zitiert auf Seite 41.)

[Beasley et al. 1994] BEASLEY ; VIC ; ISAACS ; GEOFF: *Multiple Choice Testing : HERDSA Green Guide No 16*. Higher Education Research & Development Society of Australasia Incorporated, Jänner 1994. – 54 S (Zitiert auf Seite 15.)

[Bennett 1993] BENNETT, R. E.: *On the Meaning of Constructed Response. In R. E. Bennett, Ward, W. C. (Ed.), Construction versus Choice in Cognitive Measurement: Issues in Constructed Response, Performance Testing, and Portfolio Assessment*. Hillsdale, NJ : Lawrence Erlbaum Associates, 1993 (Zitiert auf den Seiten vii und 13.)

[Bodzin und Cates 2003] BODZIN, Alec M. ; CATES, Ward M.: Enhancing Preservice Teachers' Understanding of Web-based Scientific Inquiry. In: *Journal of Science Teacher Education* 14 (2003), November, Nr. 4, S. 237–257. – URL http://www.springerlink.com/content/j11v2564022152t2/. – Zugriffsdatum: 14.12.2009 (Zitiert auf Seite 1.)

[Bouzo et al. 2007] BOUZO, José ; BATLLE, Helena ; NAVARRETE, Toni ; BLAT, Josep: Enhancing IMS QTI assessment with web maps. In: *Ten Competence Open Workshop* Departament of Information and Communication Technologies, Universitat Pompeu Fabra (Veranst.), Juni 2007 (Zitiert auf den Seiten 23 und 24.)

[Brinke et al. 2005] BRINKE, Desiree J. ; GORISSEN, Pierre ; LATOUR, Ignace: *Learning Design - A Handbook on Modelling and Delivering Networked Education and Training*. Kap. Integrating Assessment into E-learning Courses, S. 185–202, Springer Berlin Heidelberg, 2005 (Zitiert auf Seite 23.)

[Brusilovsky und Sosnovsky 2005a] BRUSILOVSKY, Peter ; SOSNOVSKY, Sergey: Engaging students to work with self-assessment questions: a study of two approaches. In: *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*. New York, NY, USA : ACM, 2005, S. 251–255. – ISBN 1-59593-024-8 (Zitiert auf den Seiten 18 und 131.)

[Brusilovsky und Sosnovsky 2005b] BRUSILOVSKY, Peter ; SOSNOVSKY, Sergey: Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK. In: *J. Educ. Resour. Comput.* 5 (2005), Nr. 3, S. 6. – ISSN 1531-4278 (Zitiert auf Seite 18.)

[Buchanan 2004] BUCHANAN, E. A.: *Online assessment in higher education: Strategies to systematically evaluate student learning*. Kap. Distance learning and university effectiveness: Changing educational paradigms for online learning. Hershey PA. : Information Science Publishing, 2004 (Zitiert auf Seite 7.)

[C2.com] C2.COM: *Wiki Wiki Origin*. – URL http://c2.com/cgi/wiki?WikiWikiOrigin. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 91.)

[Chisholm 2008] CHISHOLM, Matt: *Building a Proper MVC Pattern for the web*. August 2008. – URL http://welcome.totheinter.net/2008/08/05/mvc-pattern-for-the-web/. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 97.)

[Chong 2008] CHONG, Ho Y.: *Reliability and Validity*. 2008. – URL http://www.creative-wisdom.com/teaching/assessment/reliability.html. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten 21 und 22.)

[Constandache et al. 2003] CONSTANDACHE, Ionut ; ZAMFIR, Cristian ; UDREA, Octavian: *Implementing the IMS QTI specification: an architectural overview*. University Politehnica of Bucharest. 2003. – URL http://jupiter.ksi.edu/~changsk/chronobot/ref-scorm.pdf. – Zugriffsdatum: 08.09.2009 (Zitiert auf Seite 25.)

[CosmoCode] COSMOCODE: *WikiMatrix*. – URL http://www.wikimatrix.org. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 92.)

[Crandall 2006] CRANDALL, Chuck: *Dojo JS Toolkit*. September 2006. – URL http://www.ja-sig.org/wiki/display/UP3/Dojo+JS+Toolkit. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 85.)

[Crane et al. 2006] CRANE, Dave ; PASCARELLO, Eric ; JAMES, Darren: *Ajax in Action*. Manning Publications, 2006 (Zitiert auf Seite 75.)

[Cucchiarelli et al. 2000] CUCCHIARELLI, A. ; PANTI, M. ; VALENTI, S.: *Web-based learning and teaching technologies: Opportunities and challenges*. Kap. Web-based Assessment in Student Learning, Idea Group Publishing, 2000 (Zitiert auf Seite 7.)

[Denny et al. 2008] DENNY, Paul ; HAMER, John ; LUXTON-REILLY, Andrew ; PURCHASE, Helen: PeerWise: students sharing their multiple choice questions. (2008), S. 51–58. ISBN 978-1-60558-216-0 (Zitiert auf Seite 7.)

[Diaz et al. 2007] DIAZ, Javier ; RIFQI, Maria ; BOUCHON-MEUNIER, Bernadette: Evidential Multiple Choice Questions. In: P.BRUSILOVSKY, K. P. (Hrsg.): *Proceedings of Workshop on Personalisation in E-Learning Environments at Individual and Group Level* 11th International Conference on User Modeling (Veranst.), 2007, S. 61–64 (Zitiert auf den Seiten vii, 14 und 15.)

[Doernhoefer 2006] DOERNHOEFER, Mark: Surfing the net for software engineering notes. In: *SIGSOFT Softw. Eng. Notes* 31 (2006), Nr. 1, S. 5–13. – ISSN 0163-5948 (Zitiert auf den Seiten 75 und 78.)

[Dojo Foundation] DOJO FOUNDATION: *Dojo API*. – URL http://api.dojotoolkit.org/. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 87.)

[Dojo Toolkit Community] DOJO TOOLKIT COMMUNITY: *The Official Dojo Documentation*. http://docs.dojocampus.org. – URL http://docs.dojocampus.org. – Zugriffsdatum: 14.10.2009 (Zitiert auf den Seiten 87 und 89.)

[Downing und Haladyna 2006] DOWNING, Steven M. (Hrsg.) ; HALADYNA, Thomas M. (Hrsg.): *Handbook of Test Development*. Lawrence Erlbaum Associates, Jänner 2006. – 792 S (Zitiert auf Seite 11.)

[Doyé 1988] DOYÉ, Peter: *Typologie der Testaufgaben für den Unterricht Deutsch als Fremdsprache.* Berlin u. München: Langenscheidt KG., 1988 (Zitiert auf Seite 60.)

[Ebersbach et al. 2007] EBERSBACH, Anja ; GLASER, Markus ; HEIGL, Richard: *WikiTools : Kooperation im Web*. 2. Berlin : Springer, 2007 (Zitiert auf Seite 91.)

[elive Learning Design 2007] ELIVE LEARNING DESIGN: *IMS Learning Design*. Februar 2007. – URL http://www.elive-ld.com/content/e65/e114/index_ger.html. – Zugriffsdatum: 08.09.2009 (Zitiert auf Seite 40.)

[Falchikov und Boud 1989] FALCHIKOV, Nancy ; BOUD, David: Student Self-Assessment in Higher Education: A Meta-Analysis. In: *Review of Educational Research* 59 (1989), S. 395–430 (Zitiert auf Seite 7.)

[Flanagan 2006] FLANAGAN, David: *JavaScript: The Definitive Guide*. 5. O'Reilly Media, August 2006. – 1032 S (Zitiert auf Seite 75.)

[Foster 2008] FOSTER, D.: *Online Readings in Testing and Assessment*. Kap. Item development: Where do good questions come from?, International Test Commission, 2008. – URL http://www.psyweb.nl/research/orta/ZKhfgREqq/pdf/4a.pdf. – Zugriffsdatum: 25.08.2009 (Zitiert auf Seite 11.)

[Garrett 2005] GARRETT, Jesse J.: *Ajax: A New Approach to Web Applications*. Februar 2005. – URL http://www.adaptivepath.com/ideas/essays/archives/000385.php. – Zugriffsdatum: 14.10.2009 (Zitiert auf den Seiten viii und 79.)

[Garrison und Ehringhaus 2009] GARRISON, Catherine ; EHRINGHAUS, Michael: *Formative and Summative Assessments in the Classroom*. 2009. – URL http://www.nmsa.org/portals/0/pdf/publications/Web_Exclusive/Formative_Summative_Assessment.pdf. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten 7 und 8.)

[Gehrke 2009] GEHRKE, Jens: *Medien/McLuhans Medienbegriff*. MedienKulturWiki. August 2009. – URL http://www.uni-lueneburg.de/medienkulturwiki/medienkulturwiki2/index.php/Marshall_McLuhan. – Zugriffsdatum: 25.08.2009 (Zitiert auf Seite 4.)

[Gonzalez-Barbone und Llamas-Nistal 2007] GONZALEZ-BARBONE, V. ; LLAMAS-NISTAL, M.: eAssessment: Trends in content reuse and standardization. In: *Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE '07. 37th Annual*, Oct. 2007, S. T1G–11–T1G–16. – ISSN 0190-5848 (Zitiert auf Seite 49.)

[Haladyna 2004] HALADYNA, Thomas M.: *Developing and Validating Multiple-Choice Test Items*. 3. Lawrence Erlbaum, April 2004. – 320 S (Zitiert auf den Seiten ix, 11 und 12.)

[Hambleton und Pitoniak 2002] HAMBLETON, R. K. ; PITONIAK, M. J.: *Stevens' Handbook of Experimental Psychology*. Bd. 4. Kap. Testing and Measurement: Advances in Item Response Theory and Selected Testing Practices, S. 517–561. NJ : John Wiley and Sons, 2002 (Zitiert auf Seite 13.)

[Hambleton 2004] HAMBLETON, Ronald K.: Theory, Methods, and Practices in Testing for the 21st Century. In: *Psicothema* 16 (2004), Nr. 4, S. pp. 696–701 (Zitiert auf den Seiten 1 und 13.)

[Hartmetz 2005] HARTMETZ, Sven O.: *Unterstützung unstrukturierter Geschäftsprozess in einer wissensbasierten Arbeitsumgebung*, Universität Zürich, Diplomarbeit, 2005. – URL http://www.ifi.uzh.ch/ifiadmin/staff/rofrei/DA/DA_Arbeiten_2005/Hartmetz_Sven.pdf. – Zugriffsdatum: 14.10.2009 (Zitiert auf den Seiten viii und 95.)

[Helen 2009] HELEN: *Assessment types and activities*. Mai 2009. – URL http://www.britishcouncil.org/russia-projects-education-elt-seminars-assessment_types_and_activities.ppt. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten 6, 8 und 9.)

[Helic 2006] HELIC, Denis: *A Didactics-Aware Approach to Management of Learning Scenarios in E-Learning Systems*. Institut für Informationssysteme und Computer Medien IICM, Technische

Universität Graz, A-8010 Graz, Österreich, Technischen Universität Graz, Dissertation, November 2006 (Zitiert auf Seite 127.)

[Higgins] HIGGINS, Peter: *Taskspeed. Basic task test for frameworks.*. – URL http://dante.dojotoolkit.org/taskspeed/. – Zugriffsdatum: 16.01.2010 (Zitiert auf den Seiten viii und 140.)

[Holdener 2008] HOLDENER, Anthony T.: *Ajax: The Definitive Guide*. Jänner 2008. – 992 S (Zitiert auf den Seiten 74, 75, 76 und 77.)

[IMS Global Learning Consortium 2004] IMS GLOBAL LEARNING CONSORTIUM: IMS Content Packaging Summary of Changes - Version 1.1.4 Final Specification / IMS Global Learning Consortium. URL http://www.imsglobal.org/content/packaging/cpv1p1p4/imscp_sumcv1p1p4.html. – Zugriffsdatum: 08.09.2009, Oktober 2004. – Spezifikation (Zitiert auf Seite 40.)

[IMS Global Learning Consortium 2005] IMS GLOBAL LEARNING CONSORTIUM: IMS Question and Test Interoperability Implementation Guide - Version 2.0 Final Specification / IMS Global Learning Consortium. URL http://www.imsglobal.org/question/qti_v2p0/imsqti_implv2p0.html. – Zugriffsdatum: 08.09.2009, Jänner 2005. – Spezifikation (Zitiert auf den Seiten vii, 42, 43, 68 und 69.)

[IMS Global Learning Consortium 2006a] IMS GLOBAL LEARNING CONSORTIUM: IMS Question and Test Interoperability Assessment Test, Section, and Item Information Model - Version 2.1 Public Draft revision 2 Specification / IMS Global Learning Consortium. URL http://www.imsproject.org/question/qtiv2p1pd2/imsqti_infov2p1pd2.html. – Zugriffsdatum: 08.09.2009, Juni 2006. – Spezifikation (Zitiert auf den Seiten vii, 26, 29, 30, 33 und 39.)

[IMS Global Learning Consortium 2006b] IMS GLOBAL LEARNING CONSORTIUM: IMS Question and Test Interoperability Integration Guide - Version 2.1 Public Draft (revision 2) Specification / IMS Global Learning Consortium. URL http://www.imsproject.org/question/qtiv2p1pd2/imsqti_intgv2p1pd2.html. – Zugriffsdatum: 08.09.2009, Juni 2006. – Forschungsbericht (Zitiert auf den Seiten 40 und 41.)

[IMS Global Learning Consortium 2006c] IMS GLOBAL LEARNING CONSORTIUM: IMS Question and Test Interoperability Overview - Version 2.1 Public Draft (revision 2) Specification / IMS Global Learning Consortium. URL http://www.imsglobal.org/question/qtiv2p1pd2/imsqti_oviewv2p1pd2.html. – Zugriffsdatum: 08.09.2009, Juni 2006. – Spezifikation (Zitiert auf den Seiten vii, 23, 24, 25 und 26.)

[IMS Global Learning Consortium 2006d] IMS GLOBAL LEARNING CONSORTIUM: *IMS/GLC Announces: An Open Standard for Question and Test Interoperability (QTI)*. 2006. – URL http://www.imsproject.org/question/QTI2p1brochure.pdf. – Zugriffsdatum: 08.09.2009 (Zitiert auf Seite 24.)

[IMS Global Learning Consortium 2009a] IMS GLOBAL LEARNING CONSORTIUM: *Background*. 2009. – URL http://www.imsglobal.org/background.html. – Zugriffsdatum: 08.09.2009 (Zitiert auf Seite 25.)

[IMS Global Learning Consortium 2009b] IMS GLOBAL LEARNING CONSORTIUM: *IMS Global Learning Consortium - Question and Test Interoperability Project Group*. 2009. – URL http://www.imsglobal.org/QTI.html. – Zugriffsdatum: 08.09.2009 (Zitiert auf Seite 24.)

[Interactive Educational Systems Design, Inc. 2008] INTERACTIVE EDUCATIONAL SYSTEMS DESIGN, INC.: *How Jamestown Reading Navigator Supports Research-Based Instruction for Struggling Adolescent Readers. Formative and Summative Assessment*. McGraw-Hill Glencoe. Jänner

2008. – URL http://www.nmsa.org/portals/0/pdf/publications/Web_Exclusive/Formative_Summative_Assessment.pdf. – Zugriffsdatum: 25.08.2009 (Zitiert auf Seite 8.)

[Iske 2001] ISKE, Stefan: *Hypertext als Technologie des Umgang mit Informationen.* Bertelsmann: Bielefeld, 2001 (Zitiert auf Seite 4.)

[Jank und Meyer 1994] JANK, Werner ; MEYER, Hilbert: *Didaktische Modelle*. Bd. 3. Aufl. Cornelsen Berlin, 1994 (Zitiert auf Seite 3.)

[Jendryschik 2008] JENDRYSCHIK, Michael: *Einführung in XHTML, CSS und Webdesign: Standardkonforme, moderne und barrierefreie Websites erstellen*. München : Addison-Wesley, Dezember 2008. – 564 S (Zitiert auf Seite 74.)

[Jäger 2009] JÄGER, Kai: *Ajax in der Praxis: Grundlagen, Konzepte, Lösungen*. Springer-Verlag Berlin Heidelberg, 2009 (Zitiert auf Seite 83.)

[JISC CETIS 2008] JISC CETIS: *QTI Training Guide*. Institute for Educational Cybernetics (IEC), University of Bolton. Jänner 2008. – URL http://wiki.cetis.ac.uk/QTI_Training_Guide. – Zugriffsdatum: 08.09.2009 (Zitiert auf den Seiten vii, 26, 28, 32, 36, 37 und 38.)

[JSPWiki Community] JSPWIKI COMMUNITY: *JSPWiki Documentation*. – URL http://www.jspwiki.org/wiki/JSPWikiDocumentation. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 93.)

[Kacjan 2008] KACJAN, Brigita: *Sprachelementspiele und Wortschatzerwerb im fremdsprachlichen Deutschunterricht mit Jugendlichen und jungen Erwachsenen*. Maribor : Institut für slawische Sprachen und Literatur, Philosophische Fakultät, 2008. – 265 S (Zitiert auf den Seiten viii, 59 und 137.)

[Kacjan et al. 2009] KACJAN, Dr. B. ; CHOVANIAKOVÁ, Darina ; KOZÁKOVÁ, Mgr. V. ; KISPÁL, Dr. T.: *AP 3-2 Entwicklung von Mustertests*. November 2009. – URL http://www.sprichwort-plattform.org/attach/Ergebnisse/Musertests_AP.pdf. – Zugriffsdatum: 15.12.2009 (Zitiert auf Seite 60.)

[Kelly 2008] KELLY, Melissa: *Assessment Item Creation and Review. Creating Effective Assessments*. About.com. 2008. – URL http://712educators.about.com/cs/assessment/a/assessments.htm. – Zugriffsdatum: 25.08.2009 (Zitiert auf Seite 14.)

[Learning Technologies at Virginia Tech. 2009] LEARNING TECHNOLOGIES AT VIRGINIA TECH.: *Selecting Test Items*. 2009. – URL http://www.edtech.vt.edu/edtech/id/assess/items.html. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten 15 und 16.)

[Lie 2005] LIE, Håkon W.: *Cascading Style Sheets*. Norway, University of Oslo, Dissertation, 2005. – URL http://people.opera.com/howcome/2006/phd. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 74.)

[Lienert und Raatz 1998] LIENERT, Gustav A. ; RAATZ, Ulrich: *Testaufbau und Testanalyse*. Bd. 6. Aufl. BeltzPVU, Juli 1998. – 432 S (Zitiert auf Seite 60.)

[Maier 2006] MAIER, Gerhild: *AJAX von A bis X*. März 2006. – URL http://krottmaier.cgv.tugraz.at/docs/seminar/sem2005_ajax.pdf. – Zugriffsdatum: 14.10.2009 (Zitiert auf den Seiten 75 und 76.)

[McConnell 2000] MCCONNELL, D.: *Implementing computer supported cooperative learning.* London : Kogan Page, 2000 (Zitiert auf Seite 7.)

[McInnery und Roberts 2004] MCINNERY, J. ; ROBERTS, T.S.: Online learning: Social Interaction and the Creation of a Sense of Community. In: *Educational Technology & Society* 7(3) (2004), S. 73–81 (Zitiert auf Seite 7.)

[McLuhan 1992] MCLUHAN, Marshall: *Die magischen Kanäle. Understanding Media.* Düsseldorf: Econ, 1992 (Zitiert auf Seite 3.)

[Moussaoui und Zeppenfeld 2008] MOUSSAOUI, H. E. ; ZEPPENFELD, K.: *AJAX - Geschichte, Technologie, Zukunft*. Kap. Herausforderungen, S. 111–120, Springer Berlin Heidelberg, 2008 (Zitiert auf Seite 78.)

[Musciano und Kennedy 2006] MUSCIANO, Chuck ; KENNEDY, Bill: *HTML & XHTML: The Definitive Guide - Creating Effective Web Pages*. 6. O'Reilly Media, Oktober 2006. – 688 S (Zitiert auf Seite 74.)

[Neuberg 2005] NEUBERG, Brad: *AJAX: How to Handle Bookmarks and Back Buttons*. Oktober 2005. – URL http://onjava.com/pub/a/onjava/2005/10/26/ajax-handling-bookmarks-and-back-button.html. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 78.)

[Noonan und Duncan 2005] NOONAN, Brian ; DUNCAN, C. R.: Peer and Self-Assessment in High Schools. In: *Practical Assessment Research & Evaluation. A peer-reviewed electronic journal.* 10 (2005), November, Nr. 17 (Zitiert auf Seite 7.)

[Omar et al. 2005] OMAR, M. ; DALWAI, A. ; SULEMAN, H.: *An Evaluation of the QTI E-Learning Standard in Resource-Limited Environments*. Department of Computer Science, University of Cape Town. 2005. – URL http://pubs.cs.uct.ac.za/archive/00000275/01/www_2005_qti_final_edited_hs.pdf. – Zugriffsdatum: 08.09.2009 (Zitiert auf den Seiten ix und 39.)

[Palloff und Pratt 1999] PALLOFF, R. M. ; PRATT, K.: *Building learning communities in cyberspace*. San Francisco : Joessey-Bass, 1999 (Zitiert auf Seite 7.)

[Pfähler und Holzer 2009] PFÄHLER, Michael ; HOLZER, Matthias: Erweiterung des QTI-Standards zur Unterstützung von LongMenu-Fragen sowie Abbildung spezifischer Metadaten in QTI 2.1. In: *GMS Med Inform Biom Epidemiol.* 5(1) Doc06 (2009) (Zitiert auf Seite 128.)

[Pinellas School District] PINELLAS SCHOOL DISTRICT: *C. Reliability and Validity*. – URL http://fcit.usf.edu/assessment/basic/basicc.html. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten 21 und 22.)

[Piotrowski und Fenske 2007] PIOTROWSKI, Michael ; FENSKE, Wolfram: Interoperabilität von elektronischen Tests. In: *DeLFI 2007: 5. e-Learning Fachtagung Informatik*, GI-Verlag, September 2007, S. 185–196 (Zitiert auf den Seiten 41, 43, 44 und 131.)

[Risse 2005] RISSE, Thomas: *Erweiterung von eTests um graphische Eingaben*. Institut für Informatik & Automation, IIA, Hochschule Bremen, HSB. Oktober 2005. – URL http://www.weblearn.hs-bremen.de/risse/papers/MathEng4/graphische_eTests_s.pdf. – Zugriffsdatum: 08.09.2009 (Zitiert auf Seite 40.)

[Roberts 2006] ROBERTS, T.S.: *Self, peer, and group assessment in e-Learning*. Hershey, PA : Information Science Publishing, 2006. – 333 S (Zitiert auf den Seiten 6, 7 und 8.)

[Rodriguez 2005] RODRIGUEZ, Michael C.: Three options are optimal for multiple-choice items: A meta-analysis of 80 years of research. In: *Educational Measurement: Issues and Practice* 24 (2005), Nr. 2, S. pp. 3–13 (Zitiert auf Seite 17.)

[Rueth] RUETH, Marion: *event.based.network*. – URL http://www.marion-rueth.de/event.based.network/05_sw-design.php?link=Event.based.network&name=05_sw-design.php. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 97.)

[Russell 2008] RUSSELL, Matthew: *Dojo: The Definitive Guide*. 1. O'Reilly Media, Juli 2008. – 486 S (Zitiert auf den Seiten viii, 81, 86, 87, 88, 89 und 90.)

[Santally und Raverdy 2006] SANTALLY, Mohammad I. ; RAVERDY, Jerome: The Masters Program in Computer-Mediated Computer Communications: A Comparative Study of Two Cohorts of Students. In: *Educational Technology Research and Development* 54 (2006), June, Nr. 3, S. 312–326. – URL http://www.springerlink.com/content/t8j1xn22563pn150/. – Zugriffsdatum: 14.12.2009 (Zitiert auf Seite 1.)

[Satow 2006] SATOW, Dr. L.: *e-Learning: Eine Einführung für Autoren, Tutoren und Instructional Designer*. URL http://userpage.fu-berlin.de/~satow/elearning.pdf. – Zugriffsdatum: 25.08.2009, 2006 (Zitiert auf den Seiten 6, 9, 11, 18, 21 und 22.)

[Scalise und Gifford 2006] SCALISE, Kathleen ; GIFFORD, Bernard: Computer-Based Assessment in E-Learning: A Framework for Constructing Intermediate Constraint Questions and Tasks for Technology Platforms. In: *The Journal of Technology, Learning, and Assessment* 4 (2006), Juni, Nr. 6. – URL http://www.jtla.org. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten vii, 13 und 14.)

[Schneider und Werner 2001] SCHNEIDER, Uwe ; WERNER, Dieter: *Taschenbuch der Informatik*. 4. Fachbuchverlag Leipzig, 2001 (Zitiert auf Seite 75.)

[Schunk 2000] SCHUNK, D.: *Learning theories: An educational perspective.* Upper Saddle River, NJ : Prentice Hall, 2000 (Zitiert auf Seite 7.)

[Schyma 2007] SCHYMA, Christian J.: *Untersuchung der Verwendbarkeit und Überführbarkeit von Ajax-Ansätzen in das wingS Webframework für komfortablere und effizientere User-Interface-Komponenten*, Hochschule Albstadt-Sigmaringen, Diplomarbeit, 2007 (Zitiert auf den Seiten 75, 78 und 85.)

[SprichWort-Team 2009] SPRICHWORT-TEAM: *SprichWort. Eine Internet-Lernplattform für das Sprachenlernen*. Juli 2009. – URL http://www.sprichwort-plattform.org/sp/Projekt. – Zugriffsdatum: 16.01.2010 (Zitiert auf Seite 51.)

[Stenhouse 2005] STENHOUSE, Mike: *Fixing the Back Button and Enabling Bookmarking for AJAX Apps*. Juni 2005. – URL http://www.contentwithstyle.co.uk/content/fixing-the-back-button-and-enabling-bookmarking-for-ajax-apps. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 78.)

[Steyer 2008] STEYER, Ralph: *Ajax Frameworks*. 1. München : Addison-Wesley, September 2008. – 368 S (Zitiert auf den Seiten 79, 80 und 82.)

[Swertz 2000] SWERTZ, Christian: *Ausbildung zum Gebrauch didaktischer Ontologien*. 2000. – URL http://homepage.univie.ac.at/christian.swertz/texte/ausbildung/ausbildung_ontologien.html. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten 4 und 5.)

[Swertz 2005] SWERTZ, Christian: *Web-Didaktik. Eine didaktische Ontologie in der Praxis*. September 2005. – URL www.medienpaed.com/04-2/swertz04-2.pdf. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten vii, 3, 4, 5 und 6.)

[Szugat et al. 2006] SZUGAT, Martin ; GEWEHR, Jan E. ; LOCHMANN, Cordula: *Social Software*. Entwickler.Press, 2006. – URL http://www.worldcat.org/isbn/3939084093. – Zugriffsdatum: 14.10.2009. – ISBN 3939084093 (Zitiert auf Seite 91.)

[Trattner 2009] TRATTNER, Christoph: *Vom Austria-Forum zum Wiki-Konzept*. Institut für Informationssysteme und Computer Medien IICM, Technische Universität Graz, A-8010 Graz, Österreich, Technischen Universität Graz, Diplomarbeit, Jänner 2009 (Zitiert auf den Seiten viii, 78, 91, 92 und 93.)

[Tuparova und Tuparov 2005] TUPAROVA, Daniela ; TUPAROV, Georgi: Didactical Issues of E-learning- Problems and Future Trends. In: *International Conference on Computer Systems and Technologies - CompSysTech'2005* (2005), S. IV.12 1–5 (Zitiert auf Seite 6.)

[UK Centre for Legal Education 2008] UK CENTRE FOR LEGAL EDUCATION: *Formative vs summative assessment*. Dezember 2008. – URL http://www.ukcle.ac.uk/resources/assessment/formative.html. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten 8 und 9.)

[Wang 2008] WANG, Tzu-Hua: Web-based quiz-game-like formative assessment: Development and evaluation. In: *Comput. Educ.* 51 (2008), Nr. 3, S. 1247–1263. – ISSN 0360-1315 (Zitiert auf den Seiten 1 und 17.)

[Wübbenhorst] WÜBBENHORST, Klaus: *Stichwort: Validität*. Gabler Wirtschaftslexikon, Gabler Verlag. – URL http://wirtschaftslexikon.gabler.de/Archiv/978/validitaet-v3.html. – Zugriffsdatum: 25.08.2009 (Zitiert auf Seite 22.)

[Wenz 2007] WENZ, Christian: *JavaScript und AJAX - Das umfassende Handbuch*. 7. Galileo Computing, 2007. – 841 S. – URL http://openbook.galileocomputing.de/javascript_ajax/. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 75.)

[Wiki.org 2002] WIKI.ORG: *What Is Wiki*. Juni 2002. – URL http://wiki.org/wiki.cgi?WhatIsWiki. – Zugriffsdatum: 14.10.2009 (Zitiert auf Seite 91.)

[Wikipedia] WIKIPEDIA: *QTI*. – URL http://en.wikipedia.org/wiki/QTI. – Zugriffsdatum: 16.01.2010 (Zitiert auf den Seiten ix und 135.)

[Winkelmann 2005] WINKELMANN, Yvonne: *Aufgabentypen und Aufgabengenerierung*. Technischen Universität Dresden. März 2005. – URL http://www.yviwin.de/project/documents/Beleg.pdf. – Zugriffsdatum: 25.08.2009 (Zitiert auf den Seiten vii, 9, 10, 19, 20, 25, 40, 48 und 68.)

[Woodford und Bancroft 2005] WOODFORD, Karyn ; BANCROFT, Peter: Multiple choice questions not considered harmful. In: *ACE '05: Proceedings of the 7th Australasian conference on Computing education*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2005, S. 109–116. – ISBN 1-920682-24-4 (Zitiert auf den Seiten vii, 14, 15, 16 und 17.)