Hochschule
Ravensburg-Weingarten
Technik | Wirtschaft | Sozialwesen

Studiengang Angewandte Informatik Bachelor
Fakultät Elektrotechnik und Informatik

Bachelor-Thesis

SCORM Export von Lernmodulen aus Moodle

zur Erlangung des akademischen Grades

Bachelor of Science der Informatik

vorgelegt von:

Bastian Rosenfelder

10. Juli 2013

1. Gutachter: Prof. Dr Martin Hulin
2. Gutachter: Michel Tokic

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Die Arbeit wurde bisher, in gleicher oder ähnlicher Form, keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Unterschrift

Ort, Datum

Vorwort

Abstract

Das Ziel dieser Bachelorarbeit ist es, einen neuen Moodle[1]Block zu entwerfen, über den erstellte Lernmodule, in das SCORM[2] Format exportiert werden können. SCORM wurde ursprünglich von ADLNET entwickelt, einer Initiative, die vom US-Verteidigungsministerium gegründet wurde. Die Idee war, einen einheitlichen Mechanismus zu schaffen, um Lerninhalte zwischen verschiedenen Systemen auszutauschen.

Zu Beginn meiner Arbeit werde ich die Möglichkeiten von SCORM detailliert beschreiben. Danach gehe ich auf die verwendeten API's[3] ein. Diese Vorkenntnisse sind von essentieller Bedeutung, um die Funktionsweise und Abläufe des geschriebenen Programms zu verstehen. Zuletzt werde ich die Erfahrungen, die während der Arbeit gewonnen wurden, beschreiben und ein Fazit ziehen

Danksagung

An diesem Punkt möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelor Arbeit unterstützt und motiviert haben.

Danken möchte ich in erster Linie meinem Betreuer Prof. Dr. Hulin, für seine Unterstützung. Während der Entwicklung gab er mir immer wieder wertvolle Hinweise, die maßgeblich zum erfolgreichen Abschluss beigetragen haben.

Daneben gilt mein Dank Herrn. Michel Tokic, der mir vor allem in der Implementierungsphase mit seinem Fachwissen zur Seite stand.

Auch möchte ich mich bei all denjenigen bedanken, die einige Stunden mit dem Korrekturlesen verbracht haben. Zahlreiche Rechtschreibfehler wurden so entdeckt.

Nicht zuletzt spreche ich meinen Eltern Dank aus, da sie mich während des gesamten Studiums sowohl finanziell als auch emotional unterstützt haben.

[1]Moodle(Modular Object Oriented Dynamic Learning Environment) ist eine weit verbreitet E-Learning Plattform auf Basis von PHP

[2]Sharable Content Object Reference Model

[3]API(Application Programming Interface) bezeichnet eine Schnittstelle, die Programmen zur Kommunikation mit einem System dient

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

E-Learning Systeme werden heutzutage an fast allen Universitäten und vielen Schulen, unterstützend zum Unterricht, eingesetzt. Bei Moodle handelt es sich um ein kursbasiertes LMS[4], d.h., alle Lernmodule (z.B. Lektionen, Tests), werden dem Teilnehmer in einer Kursstruktur zur Verfügung gestellt.

Um die Portabilität dieser Inhalte zwischen den verschiedenen Systemen zu gewährleisten, muss neben einer Import auch eine Exportfunktion existieren. So können bei einem Wechsel des LMS bestimmte Teile des Kurses übernommen werden. Daraus ergeben sich vielfältige Vorteile. Zum einen wird dem Kurserstel-ler Arbeit abgenommen. Aber auch der Austausch von Lernmaterialien zwischen unterschiedlichen Hochschulen wird vereinfacht. Ein didaktisch besonders wertvoller Kurs könnte beispielsweise bundesweit angeboten werden.

1.2 Problemstellung

Das SCORM beschreibt eine Sammlung von Standards und Referenzen, um Lerninhalte in einem definierten Format zu speichern. Dadurch ist es möglich, mithilfe von spezieller Autorensoftware Lerneinheiten zu erstellen, die unabhängig vom verwendeten LMS eingesetzt werden können.

Durch die rasante technologische Entwicklung und vor allem durch das Web 2.0, haben sich E-Learning Systeme an Schulen und Universitäten mittlerweile etabliert. Moodle ist mit mehr als 80 000 Registrierungen weltweit, allein über 3000 davon in Deutschland, eines der meist genutzten LMS[5]. Die Lerninhalte werden nun nicht mehr mit Autorensoftware, sondern innerhalb des LMS erstellt.

Aktuell besitzt Moodle lediglich eine Importfunktion für SCORM Pakete. Somit müssen bei einem Umzug auf ein neues LMS alle Lernobjekte neu angelegt werden. Vor allem für größere Kurse stellt dies einen hohen Zeitaufwand dar. Aus diesem Grund habe ich mich der Aufgabe angenommen, eine Exportfunktion zu schreiben.

[4] LMS(Learning Management System) ist ein anderer Begriff für ein E-Learning System
[5] https://moodle.org/stats

1.3 Zielsetzung

Ziel der Bachelor Arbeit ist es, ein neues Plugin für Moodle 2.4 zu entwickeln. Dadurch soll es möglich sein, bestimmte Kursmodule als SCORM Objekt zu exportieren und diese in anderen Systemen wiederzuverwenden.

Vorbild für den SCORM Export stellt die Moodle-eigene Exportfunktion dar. Der Benutzer befindet sich in einem Kurs, in dem er mindestens Trainer Rechte besitzt. Er bekommt die Möglichkeit, den SCORM Export zu starten. Nach dem Start erfolgt eine Auswahlliste aller Lernobjekte des Kurses. Der Nutzer wählt Objekte für den Export aus. Auftretende Fehler oder Warnungen werden in einem Logfile gespeichert.

Erste Priorität haben Lektionen, zweite Priorität Tests. Diese Lernobjekte sollten so vollständig wie möglich und natürlich auch fehlerfrei exportiert werden. Bei jeder Einschränkung gegenüber der Moodle eigenen Exportfunktion wird begründet, warum diese notwendig ist.

Der Nutzer soll zwischen SCORM 1.2 und SCORM 2004 wählen können. Falls dies nicht realisierbar sein sollte, ist das zu begründen. Das Plugin sollte dann so programmiert werden, dass eine Erweiterung auf SCORM 2004 möglichst einfach umgesetzt werden kann.

2 SCORM Grundlagen

> *Das SCORM (Sharable Content Object Reference Model) ist ein Referenzmodell für austauschbare elektronische Lerninhalte der Advanced Distributed Learning Initiative. SCORM umfasst eine (Variablen-) Sammlung von Standards und Spezifikationen aus verschiedenen Quellen, um einfache Austauschbarkeit, einen allgemeinen Zugriff und Wiederverwendbarkeit in verschiedenen Umgebungen von web-basierenden Lerninhalten (E-Learning) zu ermöglichen. SCORM besteht seit der Version SCORM 2004 aus vier wesentlichen Dokumenten, die in englischer Sprache verfügbar sind: Overview, Content Aggregation Model, Run-Time Environment und Sequencing and Navigation. Autorenwerkzeuge und Lernplattformen unterstützen derzeit allerdings meist nur die vorherige Version SCORM 1.2, in der insbesondere der Aspekt des Sequencing and Navigation noch nicht berücksichtigt ist.*[6]

Für die vorliegende Arbeit sind vor allem das Content Aggregation Model und die Run Time Environment von Bedeutung, die ich nachfolgend im Detail beschreiben werde. Das Overview Book befasst sich mit der Entwicklung und technischen Spezifikationen von SCORM. Da diese Daten für mein Programm weniger relevant sind, werde ich darauf nicht eingehen.

[6] Auszug aus der deutschsprachigen Wikipedia[16]

2.1 Content Aggregation Model

2.1.1 Komponenten

Ein SCORM Paket besteht aus mehreren Komponenten:

- Assets
- Sharable Content Objects (SCO's)
- Content Aggregation

Mithilfe von Metadaten ist es möglich, die Komponenten genauer zu beschreiben. In Kapitel 2.1.2 wird darauf genauer eingegangen.

Assets
Bei Assets handelt es sich um die elektronische Form von Bildern, Text und anderen Daten, die ein Web-Browser interpretieren kann.

Sharable Content Objects
Ein Sharable Content Object, im weiteren Verlauf SCO genannt, stellt eine Sammlung von ein oder mehreren Assets dar. Abhängig von der Rolle, in der man sich befindet, sieht man ein SCO entweder als HTML Seite (Programmierer) oder als Lerneinheit (LMS Nutzer). Je nach Lernziel ist es möglich, SCOs zu gruppieren (Aggregation). Dadurch lassen sich Aktivitäten, wie z.B. Tests oder Lektionen, realisieren.
Für das Erstellen eines SCO gelten einige Grundsätze:

- **Wiederverwendbarkeit** Es sollte sichergestellt werden, dass keine Abhängigkeiten zum restlichen Lerninhalt bestehen. Somit kann ein SCO in mehreren Kursen verwendet werden.
- **kleine Lerneinheiten** Das SCORM macht zwar keine Angaben über die maximale Größe eines SCOs, meistens sind kleine SCOs jedoch sinnvoller. Je größer das SCO, desto schwieriger wird es, das Prinzip der Wiederverwendbarkeit zu erfüllen.

Bei einem SCO handelt es sich um die einzigste Komponente, die Befehle an das LMS absetzen und empfangen kann. Das geschieht über die SCORM API. In Kapitel 2.2 wird diese detailliert beschrieben.

Content Aggregation Content Aggregation beschreibt einen Mechanismus, um eine Struktur auf Lernressourcen anzuwenden. Das kann z.B. ein Kurs oder eine Lektion sein. Auch der inhaltliche Aufbau und der Ablauf, in dem der Lernende die Inhalte sieht, wird darüber definiert.

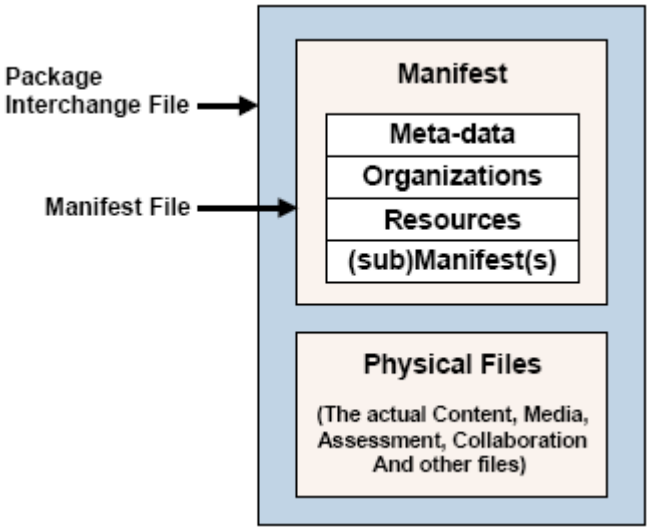

Abbildung 1: Aufbau imsmanifest.xml

2.1.2 Content Packaging

Die Aufgabe des Content Packaging ist es, eine standardisierte Methode bereitzustellen, um Lernobjekte zwischen verschiedenen Systemen auszutauschen. Auch die Reihenfolge, in der die Lerninhalte präsentiert werden, kann darüber definiert werden. Bei einem SCORM Content Package, im weiteren Verlauf PIF[7] genannt, handelt es sich um ein ZIP Archiv[8]. Es enthält alle benötigten Dateien, sowie eine XML Datei (Manifest), bestehend aus:

- Metadaten
- einem optionalen Bereich namens *organization*.
- Pfade zu den benötigten Assets und SCO's innerhalb des Pakets

Die Pfade innerhalb des PIF sind nicht standardisiert und können frei gewählt werden. Es ist jedoch ratsam, Ordner für die verschiedenen Komponenten anzulegen. Vor allem bei größeren Kursen könnten ansonsten Namenskonflikte zwischen den enthaltenen Komponenten auftreten. Die einzigste Ausnahme stellt die Manifest Datei dar.Sie muss im Top Level Directory des Archivs liegen und den Namen *imsmanifest.xml* tragen. Um die Funktionsweise eines PIF zu verstehen, ist vor allem der Aufbau der Manifest Datei wichtig (Abb. 1).

[7]PIF(Package Interchange File) ist die offizielle Bezeichnung für das SCORM Content Package
[8]SCORM 1.2 unterstützt desweiteren jar,cab und tar. Ab Version 1.3 ist nur noch zip erlaubt

Ich werde nachfolgend den internen Aufbau der Manifest Datei näher beschreiben. Dazu werden zunächst einige Begriffe erklärt. Die untenstehenden Begriffe stellen bei weitem nicht alle verfügbaren xml Attribute dar, das würde an dieser Stelle zu weit reichen. Meiner Ansicht nach sind es jedoch diejenigen, die für das Verständis der Funktionsweise grundlegend sind. Listing 1 zeigt ein exemplarisches Beispiel einer Manifest Datei.

Listing 1: imsmanifest.xml

```
2  <manifest>
3   <organizations>
4    <organization>
5      <title>Titel des Lernobjekts</title>

7      <item identifier = "block_1">
8       <title>Titel eines Blocks</title>
9       <item identifier = "page_1" identifierref = "sco_1">
10        <title>Titel des ersten SCO</title>
11      </item>
12     </item>

14     <metadata>
15             <schema>ADL SCORM</schema>
16             <schemaversion>1.2</schemaversion>
17             <adlcp:location>Course1.xml</adlcp:location>
18         </metadata>
19   </organization>
20  </organizations>
21  <resources>
22    <resource identifier = "sco_1" type = "webcontent"
23    adlcp:scormtype = "sco" href = "sco/sco_01.html">
24     <file href = "sco/sco_01.html" />
25     <dependency identifierref = "asset_1" />
26    </resource>

28    <resource identifier="asset_1" adlcp:scormtype="asset" type="webcontent">
29          <file href="assets/SCOFunctions.js"/>
30          <file href="assets/APIWrapper.js"/>
31       </resource>
32  </resources>
33 </manifest>
```

Manifest Jede Manifest Datei beginnt mit *<manifest>* und endet mit *</manifest>*. Innerhalb dieses Bereichs wird die Struktur und der Ablauf des Inhalts definiert. Darüberhinaus werden die benötigten Dateien referenziert.

Metadaten Metadaten beschreiben Dateien. Sie können auf alle Komponenten angewendet werden. Die Definition erfolgt entweder in einer externen XML[9] Datei, oder in der Manifest Datei selbst. Über das Attribut *adlcp:location* wird die Datei referenziert. SCORM Metadaten sind ein sehr umfangreiches und komplexes Thema. In Kapitel 2.1.3 werde ich darauf genauer eingehen.

Organization Der *Organization* Teil innerhalb des Manifests ist optional. Die Struktur der Lerneinheit wird dadurch bestimmt.

Item Das Item Element wird in dem Bereich *organization* verwendet. Es stellt entweder einen Block oder ein SCO dar. Bei einem Block handelt es sich um eine Sammlung von SCO's. Ein Item kann also mehrere Items enthalten. Dadurch wird der Lerninhalt strukturiert (z.B. in Kapitel).

Ressourcen In diesem Bereich wird auf externe Ressourcen verwiesen.

Dependency Dependencies werden vor allem für SCO Ressourcen genutzt. Ein SCO kann beispielsweise von einem oder mehreren Assets abhängig sein (z.B., wenn es ein Bild oder ein Video enthält). In diesem Fall wird zu der Ressource das Dependency Element für das jeweilige Asset hinzugefügt.

[9] XML(Extensible Markup Language) wird verwendet um Daten hierarchisch zu strukturieren

2.1.3 Metadaten

Metadaten werden genutzt um SCORM Komponenten näher zu beschreiben. In Repositories kann dadurch z.B. nach bestimmten Kapiteln gesucht werden. In meinem Programm werden Metadaten ausschliesslich zur Kennzeichnung der SCORM Version verwendet. Ich werde deshalb dieses Kapitel auf die wesentlichsten Aspekte beschränken.
Das Meta Data Information Model unterscheidet neun Kategorien von Metadaten.

- **General** enthält allgemeine Informationen über die Lernressource
- **Lifecycle** Status der Lerneinheit,Erstelldatum werden hier beschrieben
- **Meta-metadata** Daten über Metadaten
- **Technical** hier werden technische Spezifikationen und Anforderungen beschrieben.
- **Educational** enthält Einträge über den didatischen Aufbau der Lernressource
- **Rights** Infos über das Copyright
- **Relation** Einträge zu bestehenden Beziehungen zu anderen Lernobjekten
- **Annotation** Kommentare
- **Classification** gibt an, in welche Kategorie die Lernressource eingeordnet werden kann.

Metadaten werden, ebenfalls wie die Manifest Datei, in XML definiert.[10](Vgl.[11]).

[10] EnthältBeispielefürdenEinsatzvonMetadatenhttp://www.adlnet.gov/resources/scorm-1-2-content-packages?type=software_downloads
[11] Spezifikation des Content Aggregation Model[1]

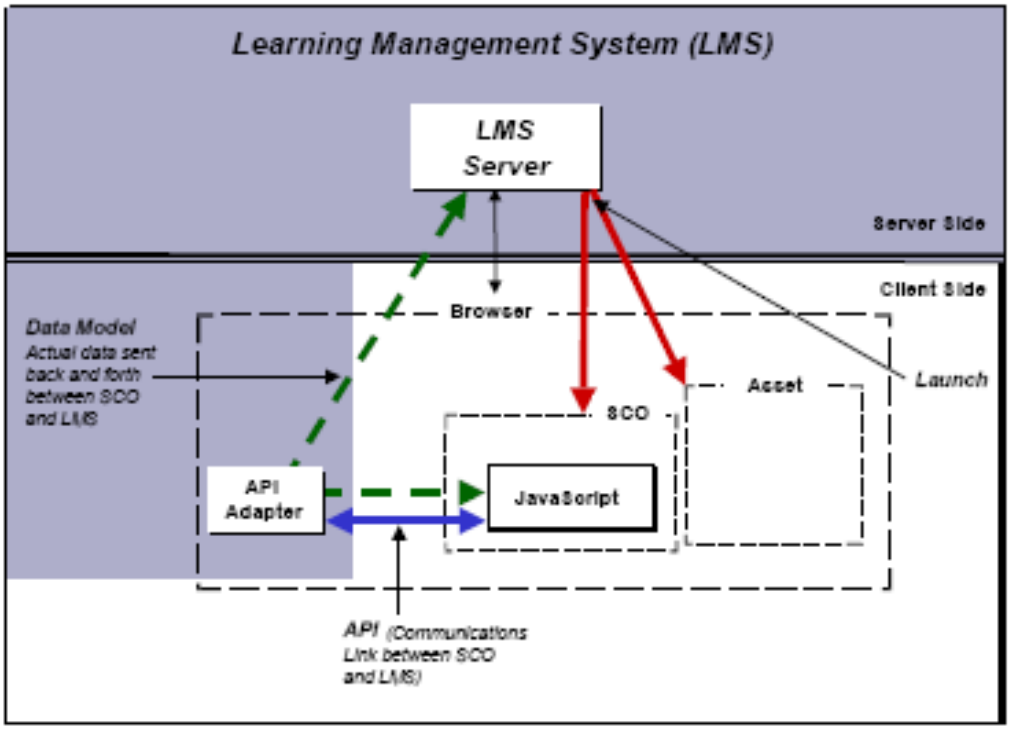

Abbildung 2: Funktionsweise der Run Time Environment

2.2 Run Time Environment

Ein Hauptziel des SCORM ist die Wiederverwendbarkeit von Lerninhalten. Um das zu erreichen, ist ein einheitliches Interface[12] zu dem verwendeten LMS notwendig (Abb.2). Die Run Time Environment setzt sich aus 3 Komponenten zusammen:

- **Launch** definiert einen einheitlichen Mechanismus, um Lerneinheiten zu starten.
- **API** dient zur Kommunikation des SCO mit dem LMS. Der Informationsweg ist hierbei bidirektional.
- **Data Model** spezifiziert, welche Art von Informationen zwischen dem SCO und dem LMS ausgetauscht werden

[12]Schnittstelle, durch die mit einem System kommuniziert werden kann

2.2.1 Launch

Die folgenden zwei SCORM Komponenten sind ausführbar: Assets und SCO's. Sobald ein Event eintritt, das kann beispielsweise der Aufruf einer Lerneinheit sein, wird das LMS getriggert[13]. Welches SCO dann ausgeführt wird, ist abhängig von der Struktur des Lernobjekts bzw. der Wahl des Nutzers. Auch die verwendete SCORM Version spielt eine Rolle. Mittels Sequencing & Navigation ist es möglich, die Reihenfolge festzulegen, in der dem Lernenden die Inhalte präsentiert werden. Bei einem Test wäre es denkbar den Nutzer erst dann den Zugriff auf die nächste Frage zu erlauben, wenn er die vorherige Frage korrekt beantwortet hat.
Handelt es sich bei einer Lerneinheit um ein Asset, geschieht die Ausführung über das HTTP[14] Protokoll. Das reicht in diesem Fall aus, da ausschliesslich SCO's über die SCORM API mit dem LMS kommunizieren können.
Für SCO's gelten andere Regeln. Das SCORM schreibt folgendes vor:

- es dürfen nie mehrere SCO's gleichzeitig aktiv sein.
- SCO's werden ausschließlich vom LMS gestartet. Die Ausführung eines SCO durch ein anderes SCO ist verboten.

Das SCORM API ist in Javascript, einer clientseitigen Skriptsprache implementiert. Dadurch ist es nicht möglich, auf direktem Weg Befehle mit dem LMS auszutauschen. In Abb. 2 wird deutlich, dass die Kommunikation zwischen SCO und LMS über den API Adapter 2.2.2 abgewickelt wird. Um ein SCO zu starten, muss eine Instanz dieses Adapters existieren, ansonsten erscheint eine Fehlermeldung und die Ausführung wird abgebrochen.

[13] Ein Trigger (engl. Abzug) bezeichnet in der Informatik das Auslösen einer Funktion, durch ein vorangegangenes Event

[14] HTTP(Hypertext Transfer Protocol) wird zur Übertragung von Daten verwendet. Meistens zwischen Browser und Webserver

2.2.2 API

Das SCORM API stellt einen einheitlichen Weg für die Kommunikation zwischen SCO's und E-Learning Plattformen dar. Es handelt sich dabei um eine Sammlung von JavaScript Funktionen, die der API Adapter interpretieren kann. Details über die Implementierung des Adapters spielen hierbei für den Programmierer keine Rolle. Wichtig ist lediglich, dass die Nutzung der Schnittstelle gleich bleibt. Nur so kann die Plattformunabhängigkeit der PIFs garantiert werden.
Die Kommunikation wird immer vom SCO initiiert. Sobald eine Instanz der API gefunden wurde, kann kommuniziert werden. Die API Funktionen lassen sich in 3 Gruppen einteilen:

- **Execution State** Funktionen zum Initialisieren (*LMSInitialize()*) und Beenden(*LMSFinish()*) des SCO
- **State Management** wird zum Error Handling benötigt, z.B. *LMSGetLastError()*
- **Data Transfer** Das SCO kann mithilfe dieser Befehle, Daten mit dem LMS austauschen. Beinhaltet *LMSGetValue(param)*, *LMSSetValue(datamodel param,value)* und *LMSCommit()*. Die Parameter für *datamodel* und *param* sind durch das SCORM standardisiert. Welche Werte erlaubt sind, ist unter http://www.adlnet.gov einsehbar.

API Adapter

Der API Adapter wird zur Kommunikation des SCO mit dem LMS benötigt. Abhängig vom verwendeten LMS kann dieser in verschiedenen Programmiersprachen implementiert sein. Auf das Interface, das er zur Verfügung stellt, darf das keinen Einfluss haben. Ansonsten wäre das Prinzip der Wiederverwendbarkeit verletzt.

API Wrapper

Der SCORM API Wrapper erweitert die API um bestimmte Funktionalitäten, wie z.B. Error Handling. Dem Kursersteller wird dadurch viel Schreibarbeit abgenommen [15].

[15] Die offiziellen SCORM API Wrapper: http://www.adlnet.gov/resources/official-adl-scorm-api-wrappers?type=software_downloads

2.2.3 Data Model

Das SCORM Data Model dient zum Austausch von Daten über bestimmte SCO's zwischen Systemen. Möchte man Informationen an das LMS übermitteln, z.B. die erreichte Punktzahl in einem Test, so muss dafür ein einheitlicher Weg zur Verfügung stehen, den alle Plattformen unterstützen. Um das zu erreichen, wurden einige Regeln definiert:

- Die Namen der Elemente sind case-sensitive[16].
- SCO's können nur auf ihre eigenen Daten zugreifen.
- alle Arrays sind null basiert.

Während der Entwicklung des SCORM wurden mehrere Datenmodelle entwickelt. Ich verwende in meinem Programm das AICC[17] CMI Data Model, da es sich mittlerweile durchgesetzt hat. An dem Präfix ist der Typ des Datenmodells zu erkennen. In diesem Fall wäre das *cmi*.
Die Verwendung der Elemente des Datenmodells ist optional. Ausschließlich die Funktionen *LMSInitialize()* und *LMSFinish()* müssen eingebunden werden, um ein SCO als solches zu kennzeichnen.
Listing 2 zeigt ein praktisches Einsatzbeispiel des API. Eine Übersicht aller möglichen Elemente ist im Anhang zu finden. (Vgl.[18])

```
2  function checkTrueFalseTest()
3  {
4   //defines the type of interaction (e.g. choice,true-false,fill-in etc)
5   doLMSSetValue("cmi.interactions.0.type","true-false");

7   /*
8    *variable definitions and program logic
9    */

11   if(totalScore == parseFloat(maxScore[0].value))
12   {
13   //in this case result can be correct or wrong.
14   doLMSSetValue("cmi.interactions.0.result","correct");

16   //sends a message to the LMS with the reached score
17   doLMSSetValue("cmi.core.score.raw",totalScore.toString());

19   //calls LMSFinish and sets the session time
20     return unloadPage("completed");
21   }
```

[16] case sensitive bedeutet, dass zwischen Gross und Kleinschreibung unterschieden wird.
[17] AICC = Aviation Industry CBT Committee
[18] Spezifikation der Runtime Environment (Version 1.2)[2]

```
22    if(totalScore < parseFloat(maxScore[0].value))
23    {
24    doLMSSetValue("cmi.interactions.0.result","wrong");
25    doLMSSetValue("cmi.core.score.raw",totalScore.toString());
26      return unloadPage("incomplete");
27    }

29  }
```

Listing 2: praktisches Beispiel für die Nutzung der SCORM API

2.3 Unterschiede zwischen SCORM 1.2 und 2004

Für SCORM 1.2 existieren lediglich die Bücher: Overview, Content Aggregation Model und Run Time Environment. Die Version 2004 umfasst ein weiteres Buch: Sequencing and Navigation. Dadurch kann der Lernprozess präziser gesteuert werden. Die Möglichkeit, den Ablauf zu kontrollieren, bestand bereits in SCORM 1.2, allerdings nur in sehr eingeschränktem Umfang.
In den folgenden Abschnitten werde ich die Änderungen zwischen den Versionen beschreiben. Der Schwerpunkt wird dabei auf Sequencing and Navigation liegen.

2.3.1 Änderungen in der Run Time Environment (RTE)

In dem RTE Buch hat sich vor allem die Namensgebung der API Methoden geändert (Tabelle 2.3.1). Darüberhinaus bietet SCORM 2004 mehr Errorcodes an. Dadurch wird das Error Handling vereinfacht, da genauere Fehlermeldungen angezeigt werden. Eine ausführliche Liste mit den Errorcodes findet sich im Anhang (Tabelle 7.2). Auch das Datenmodell wurde erweitert, um den Sequencing and Navigation Mechanismus zu ermöglichen. (Vgl.[19])

1.2	**2004**
LMSInitialize()	*Initialize()*
LMSFinish()	*Terminate()*
LMSGetValue(param)	*GetValue(param)*
LMSSetValue(dataModel element,value)	*SetValue(dataModel element, value)*
LMSCommit()	*Commit()*
LMSGetLastError()	*GetLastError()*
LMSGetErrorString(param)	*GetErrorString(param)*
LMSGetDiagnostic(param)	*GetDiagnostic(param)*

Tabelle 1: Unterschiede in der SCORM API

[19]Spezifikation der Run-Time Environment unter SCORM 2004[12]

2.3.2 Änderungen im Content Aggregation Model

An folgenden Komponenten des CAM Buches wurden Änderungen vorgenommen:

- Content Packaging
- Content Model
- Metadaten
- Dateiformat des PIF

Durch die Einführung von Sequencing and Navigation mussten einige Änderungen am Content Packaging vorgenommen werden (Tabelle 2.3.2). Die Tabelle zeigt nicht, welche Elemente in Version 2004 dazukamen, sondern lediglich die Änderungen bereits bestehender Elemente.

SCORM 1.2	**SCORM 2004**
<adlcp:prerequisites>	entfällt. wird durch Sequencing and Navigation Mechanismen überflüssig
<adlcp:maxtimeallowed>	entfällt
<adlcp:timelimitaction>	wurde umbenannt in *<adlcp:timeLimitAction>*
<adlcp:datafromlms>	wurde umbenannt in *<adlcp:dataFromLms>*
<adlcp:masteryscore>	wurde umbenannt in *<adlcp:masteryScore>*

Tabelle 2: Unterschiede im Content Packaging

Am Content Model hat sich nur die Namensgebung geändert. Die Content Aggregation Komponente aus Version 1.2 wurde in Organization umbenannt.
Auch die Namen der Metadaten wurden geändert. Da ich diese in meiner Arbeit nicht verwende, verzichte ich an dieser Stelle darauf, die Unterschiede zu erklären. Während in SCORM 1.2 noch mehrere Dateieendungen für das PIF erlaubt waren (jar, cab, tar, zip), wird in Version 2004 nur noch zip akzeptiert (Vgl.[20]).

2.3.3 Unterschiede im Sequencing and Navigation Buch

In SCORM 1.2 ist Sequencing nur über die Verwendung des *<adlcp:prerequisites>* Element möglich. In der Manifest Datei wird es einem *<item>* als Attribut hinzugefügt. Dadurch kann erreicht werden, dass der Lernende die Lektionen in einer

[20]Spezifikation des Content Aggregation Model unter SCORM 2004[11]

vorgegebenen Reihenfolge besucht (z.B. Vorbedingung für Lektion 1 ist Lektion 0). Die Navigation wird in dem Bereich <*organization*> definiert.

In Version 2004 stehen dem Programmierer mehr Möglichkeiten zur Verfügung, um den Lernfluss zu steuern. Über folgende Mechanismen kann der Sequencing Mechanismus realisiert werden:

- Control Modes
- Rollups
- Global Objectives

Control Modes
Während in SCORM 1.2 der Teilnehmer noch frei zwischen den SCO's auswählen konnte, kann der Kursersteller nun bestimmte Lernpfade definieren. Das gibt ihm z.B. die Möglichkeit, bestimmte didaktische Konzepte zu realisieren.
Es gibt drei Control Modes, die entweder true oder false als Wert besitzen dürfen (Tab.2.3.3). Jeder Kurs besitzt Control Modes, auch wenn der Programmierer sie weglässt (keine Einträge in der Manifest Datei). In diesem Fall werden die Default Werte angegeben. Control Modes können in der Manifest Datei an Komponenten

Control Mode	**Default Wert**	**Beschreibung**
Choice	true	erlaubt dem Teilnehmer, frei zwischen den Lerninhalten zu navigieren
Flow	false	zwingt den Teilnehmer dazu, die Lerninhalte in einer vorgegebenen Reihenfolge zu bearbeiten.
Choice Exit	true	gibt an, ob der Teilnehmer ein SCO ausserhalb des Blocks, anwählen darf.

Tabelle 3: Die unterschiedlichen Control Modes und ihre Bedeutung

vom Typ Content Aggregation gebunden werden. Die definierten Regeln betreffen alle Kindelemente, das können entweder SCO's und/oder Content Aggregations sein. Folgender Manifest Eintrag erlaubt dem Teilnehmer die Bearbeitung der SCO's in beliebiger Reihenfolge:

<imsss:controlMode flow=false choice = true />

Soll die Reihenfolge der SCO's vorgegeben sein, muss die Zeile so aussehen:

<imsss:controlMode flow=true choice = false />

Rollups
Rollups werden eingesetzt, um den Status von Kindelementen, dem Elternelement mitzuteilen. So können beispielsweise folgende Szenarien umgesetzt werden:

- die Content Aggregation ist bestanden, wenn alle Kindelemente den Status passed besitzen.
- die Content Aggregation ist nicht bestanden, wenn mindestens ein Kindelement den Status failed besitzt.
- die Content Aggregation ist bestanden, wenn mindestens 75 % der erreichbaren Punktzahl erzielt wurde.

Die Regeln werden in der Manifest Datei definiert und können, ebenfalls wie Control Modes, nur an Komponenten vom Typ Content Aggregation gebunden werden. Für den ersten Fall müsste der Code Ausschnitt aus Listing 3 in die Manifest Datei eingetragen werden.

```
1 <imsss:rollupRule childActivitySet = all>
2  <imsss:rollupConditions conditionCombination=any>
3   <imsss:rollupCondition condition = satisfied />
4  </imsss:rollupConditions>
5  <imsss:rollupAction action = satisfied/>
6 </imsss:rollupRule>
```

Listing 3: Exemplarische Implementierung von Rollup Regeln

Manchmal ist es erwünscht, dass die Punktzahl eines SCO nicht in die Gesamtpunktzahl einfließt. Über das *imsss:rollupRules* Element kann dieses Verhalten erreicht werden. Es sind drei Werte möglich:

- **rollupObjectiveSatisfied** true, falls das Element in die Berechnung des satisfied status des Parent miteinbezogen werden soll.
- **rollupProgressCompletion** true, falls das Element ind ie Berechnung des completion status des Parent miteinbezogen werden soll.
- **objectiveMeasureWeight** wird zur Berechnung der Durchschnittspunktzahl angegeben

Global Objectives Bei Objectives handelt es sich um Variablen, die Informationen über ein SCO aufnehmen können. Jedes SCO kann solche Objectives setzen und auslesen. Dazu gehören:

- success_status
- completion_status

- score(scaled,raw,min,max)
- progress_measure

Dadurch kann z.B, die Punktzahl eines SCO mit einem anderen SCO geteilt werden (Vgl.[21]).

[21]Sequencing and Navigation Buch für SCORM 2004[13]

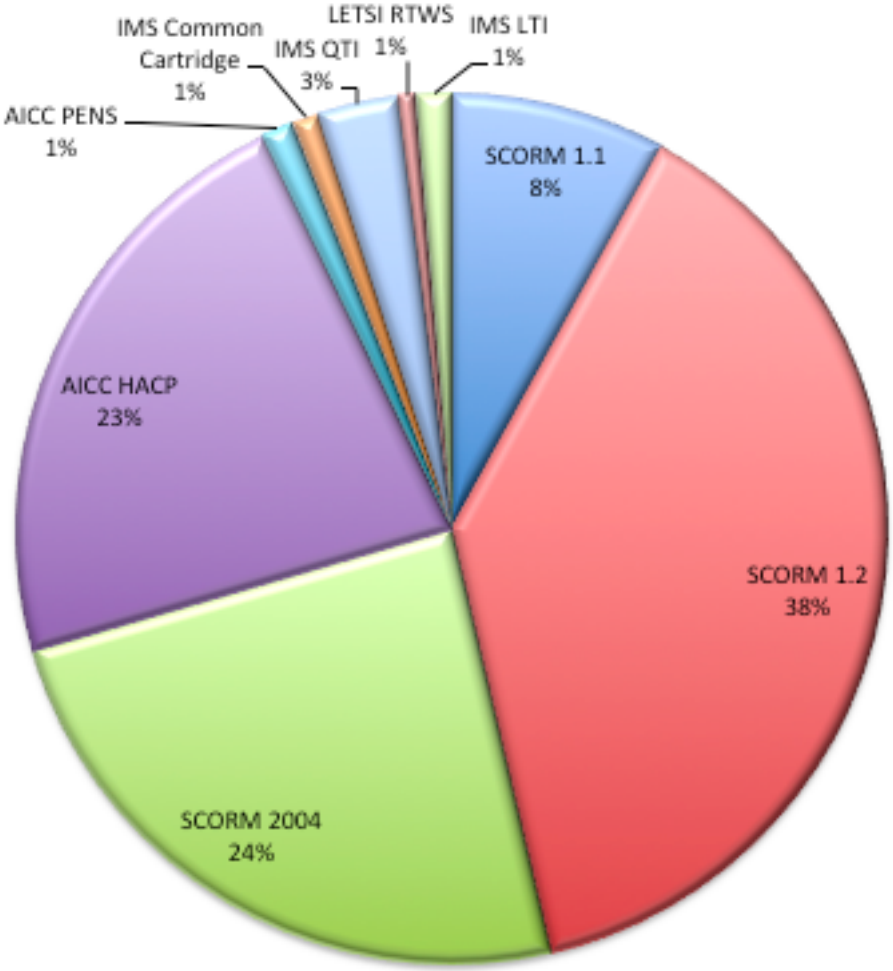

Abbildung 3: prozentuale Verteilung der gängigsten E-Learning Standards

2.4 SCORM Unterstützung

Derzeit existieren eine Vielzahl an E-Learning Systemen. Tab.2.4 soll einen Überblick über die unterstützten SCORM Versionen, der bekanntesten, geben. Zu erwähnen ist, dass es sich bei SCORM konformanten Produkten nicht zwingend um ein LMS handeln muss. [22][23]
Abb.3 zeigt die prozentuale Verteilung, bezogen auf E-Learning Standards.

[22]Liste aller SCORM zertifizierten Produkte: http://www.adlnet.gov/wp-content/uploads/2012/04/SCORMCertifiedProductsLocked.xlsx
[23]Liste aller Unternehmen, die SCORM verwenden: http://www.adlnet.gov/wp-content/uploads/2012/01/SCORMAdoptersLocked.xlsx

3 Moodle Grundlagen

Moodle's Grundstruktur ist in Kursbereichen und Kurse organisiert. Kurse sind Seiten oder Bereiche innerhalb von Moodle, in denen Lehrende Arbeitsmaterialien und Aktivitäten für die Kursteilnehmer/innen zur Verfügung stellen können. Kurse können verschieden gestaltet sein und vom Design her unterschiedlich aussehen, in der Regel besteht eine Kursseite jedoch aus einzelnen Kursabschnitten, in denen die Arbeitsmaterialien und Aktivitäten bereitgestellt werden, sowie aus Blöcken an den Seiten (links und/oder rechts) mit zusätzlichen Informationen[25]. Es gibt eine Reihe von Standardblöcken, die bei jeder Installation vorhanden sind, z.B. der Kalender oder die Liste aller Teilnehmer.
Bei Moodle handelt es sich um Open Source Software, d.h., jeder hat die Möglichkeit, an der Entwicklung teilzuhaben. Für mein Projekt habe ich einen Block programmiert, der zur Laufzeit der Benutzeroberfläche hinzugefügt und wieder entfernt werden kann.
Nachfolgend beschreibe ich die Vorgehensweise bei der Entwicklung eines Blocks.

[25] Aufbau einer Moodle Site[4]

3.1 Verzeichnisstruktur von Moodle

3.1.1 Das admin Verzeichnis

Beinhaltet alle Funktionen für den Website Administration Block. Die cron.php ist für die Entwicklung eines Blockes eine der wichtigsten Dateien in diesem Verzeichnis Viele Module müssen in zyklischen Abständen Funktionen ausführen. Das kann bspw. das Senden einer Benachrichtigung an die Teilnehmer oder ein Backup des Kurses sein. Die cron.php nutzt dafür den cronjob service von UNIX. Der Ablauf des Skriptes sieht folgendermaßen aus:

1. durchsucht die mdl_modules[26] Tabelle in der Datenbank nach Modulen, für die cron ausgeführt werden muss.

2. in dem jeweiligen Modul-Verzeichnis wird die Funktion module-name_cron in der lib.php ausgeführt

3. durchsucht die mdl_block Tabelle in der Datenbank nach Blöcken, für die cron ausgeführt werden muss.

4. für jeden Block wird die zugehörige cron Methode ausgeführt

3.1.2 Das blocks Verzeichnis

Blöcke werden links und rechts von der Kursseite angezeigt. Beispiele für bereits vorinstallierte Blöcke sind: Kalender, Teilnehmer, Blogs. Welche Blöcke angezeigt werden, liegt in der Hand des Kurs-Administrators. Es können jederzeit neue Blöcke hinzugefügt bzw. bestehende gelöscht werden. Abb.4 veranschaulicht den Aufbau eines Kurses.

3.1.3 Das lang Verzeichnis

Enthält lokalisierte Dateien, um die Benutzeroberfläche von Moodle in einer gewünschten Sprache anzuzeigen. Das lang Verzeichnis ist an dieser Stelle relevant, da es auch innerhalb eines Plugins vorhanden sein kann. Es enthält eine Sammlung von Strings (z.B. zur Beschriftung eines Buttons oder Textfeldes). So kann ein Block in mehreren Sprachen installiert werden.
Damit Moodle ein Language file als solches identifizieren kann, muss die Namensgebung standardisiert sein. Die Unterordner von lang müssen das offizielle Länderkürzel tragen (en = englisch, de = deutsch)[27] . Die Datei mit den Language Strings trägt den Namen block_blockname (für meine Arbeit wäre der blockname scormexport).

[26] mdl_ ist das Standard Präfix für Datenbank Tabellen unter Moodle.
[27] Unterstützte Sprachen: http://download.moodle.org/langpack/2.0/

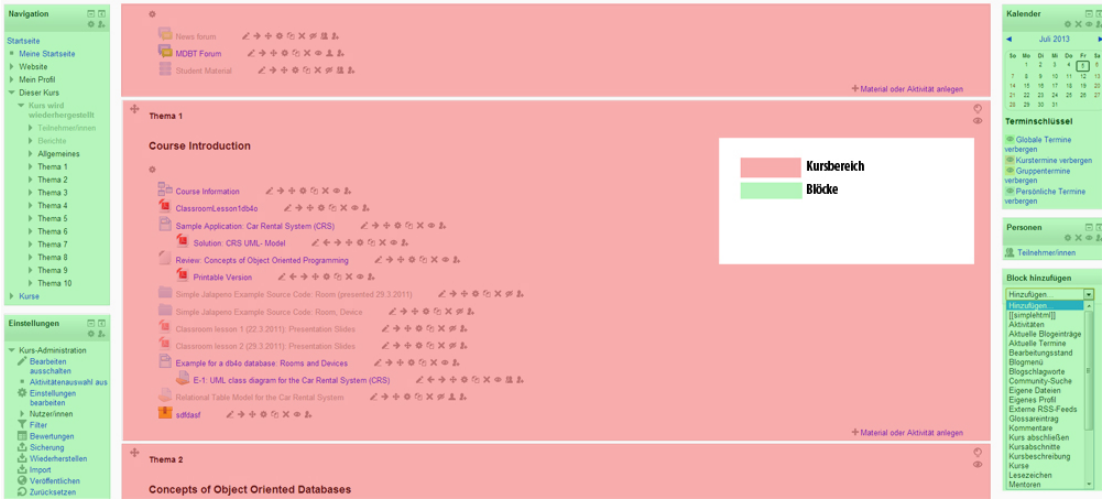

Abbildung 4: Aufbau eines Moodle Kurses

3.1.4 Das mod Verzeichnis

Enthält Moodle Module (z.B. Quiz oder Lesson). Es beinhaltet eine Vielzahl von Funktionen, die ich in meiner Arbeit verwende. In Kapitel 4 werde ich darauf genauer eingehen.

3.2 Block Programmierung unter Moodle

In diesem Kapitel werde ich aufzählen, was bei der Entwicklung unter Moodle zu beachten ist. Zuerst gehe ich auf reservierte Dateinamen ein. Solche Dateien werden von Moodle erkannt und interpretiert. Ich werde nachfolgend aufzählen, welche standardisierten Dateien in meinem Programm verwendet wurden und welchen Zweck sie erfüllen.

- **view.php**
  Enthält die gesamte Programmlogik. Dazu gehört:
  - Moodle API Funktionen sowie selbst definierte Funktionen
  - Zugangskontrolle. Welche Rolle wird für die Nutzung dieses Blocks benötigt (Trainer, Teilnehmer etc.)?
  - Verwaltung von globalen Variablen (PAGE,COURSE,DB).
  - Erzeugung der Benutzeroberfläche

- **version.php**
  Gibt an, welche Moodle Version benötigt wird. Darüberhinaus wird auch die Versionsnummer des Blocks angegeben. Wenn sich diese ändert, weiß Moodle dass Änderungen am Block vorgenommen wurden. Die Datenbank wird daraufhin aktualisiert.

- **settings.php**
  Unter Moodle können Konfigurationsseiten für Blöcke angelegt werden. In dieser Datei werden die Elemente definiert, die vom Nutzer einstellbar sein sollen. In meinem Programm ist das derzeit nur der Speicherort für die Logfiles. Damit die Einstellugen in Moodle angezeigt werden, muss in block_scormexport.php die Funktion *has_config()* überschrieben werden.

- **block_scormexport.php**
  Sobald ein Block hinzugefügt wird, erzeugt Moodle eine Instanz von block_scormexport und ruft *init()* auf. Diese Klasse definiert den Inhalt und die Einstellungen des Blocks. Anstatt „scormexport“ wird der Name des jeweiligen Blockes eingetragen.

- **lang/de/block_scormexport.php** Wird zur Definition von lokalisierten Strings benötigt. Anstatt „de“ und „scormexport“, wird das jeweilige Länderkürzel und der Blockname angezeigt. Es ist auch möglich, den Block mehrsprachig zu programmieren. In diesem Fall müsste neben „de“ einfach ein weiteres Verzeichnis erstellt werden (Vgl.[28] und [29]).

[28] Blocks/Appendix A[3]
[29] Blocks[5]

LMS	SCORM Unterstützung	URL
ILIAS	SCORM 1.2 und SCORM 2004 (3rd Edition) wird unterstützt	http://www.ilias.de
Moodle	Version 1.2 wird ab Moodle 2.1 vollständig unterstützt. Es wurden zwar Teile der SCORM 2004 API implementiert, allerdings nicht Sequencing and Navigation. Falls ein voll funktionsfähiger SCORM 2004 Player in Moodle benötigt wird, ist dies in Kombination mit der SCORM Cloud möglich[24]. Auf dieses Thema werde ich später noch eingehen.	http://docs.moodle.org/23/en/SCORM_FAQ
Chamilo	SCORM 1.2 Import wird unterstützt	http://classic.chamilo.googlecode.com/hg/documentation/readme.html
imc Content Studio	SCORM 1.2 und SCORM 2004 Import wird unterstützt	http://www.im-c.de/produkte/imc/software-solutions/content-studio/highlights/
Metacoon	SCORM 1.2 Import wird unterstützt	http://www.metacoon.net/
Claroline	SCORM 1.2 Import wird unterstützt	http://doc.claroline.net/de/index.php/Wie_erstelle_ich_einen_SCORM_Inhalt%3F_(Englisch)
OLAT	SCORM 1.2 Import wird unterstützt	http://www.olat.org/images/olat/downloads/manuals/help_de.pdf"
Ganesha	SCORM 1.2 und 2004 wird unterstützt.	http://www.ganesha-lms.de
DotNetSCORM	Interessantes Projekt, das sich derzeit noch in der Entwicklung befindet. SCORM 1.2 und 2004 soll unterstützt werden.	http://dotnetscorm.codeplex.com/

4 Programmierung

4.1 Verwendete Werkzeuge

Für die Reallisierung des Projekts wurden diverse Tools benötigt. Tab.4.1 gibt eine Übersicht über die verwendeten Programme.

Bereich	**Programm**
Programmierung	Microsoft Visual Studio 2012 Premium[30]+ php Tools for Visual Studio[31]
Versionsverwaltung	Subversion (AnkhSVN Plugin)[32]
Repository	Google Code
Dokumentation	TeXnicCenter + pdflatex[33]
Testumgebung	Moodle, SCORM Cloud[34]
Diagramme	ArgoUML[35]

Tabelle 5: Verwendete Werkzeuge

4.2 Programmablauf

Um den SCORM Export zu starten, muss man sich in einem Kurs befinden. Falls der Block noch nicht erzeugt wurde, kann man das über

Blocks hinzufügen->SCORM Export

tun. Nun müsste der Block vorhanden sein (Abb.5). Klickt man auf Export, erscheint das Export Menü (Abb.6) Dort werden alle Lernmodule die exportiert werden können, aufgelistet. Das sind derzeit Lektionen, Tests und Textpages. Der Benutzer kann nun auswählen, welche Objekte er exportieren möchte. Zu erwähnen ist, dass immer nur ein PIF erzeugt wird, unabhängig davon, wie viele Elemente angewählt werden. Soll die gleiche Kursstruktur erhalten bleiben, muss jedes Element einzeln exportiert werden. Über den Export Button wird der Vorgang gestartet. Wird auf Abbrechen gedrückt, wird wieder der Kurs angezeigt.

[35]http://www.microsoft.com/visualstudio/deu/products/visual-studio-premium-2012
[35]http://visualstudiogallery.msdn.microsoft.com/6eb51f05-ef01-4513-ac83-4c5f50c95fb5
[35]http://ankhsvn.open.collab.net/
[35]http://www.texniccenter.org/resources/downloads
[35]http://scorm.com/scorm-solved/scorm-cloud-features/?gclid=CIrT3ZT4obgCFdDJtAodQV0ABg
[35]http://argouml.tigris.org/

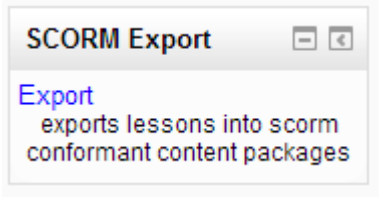

Abbildung 5: Screenshot: Export Block

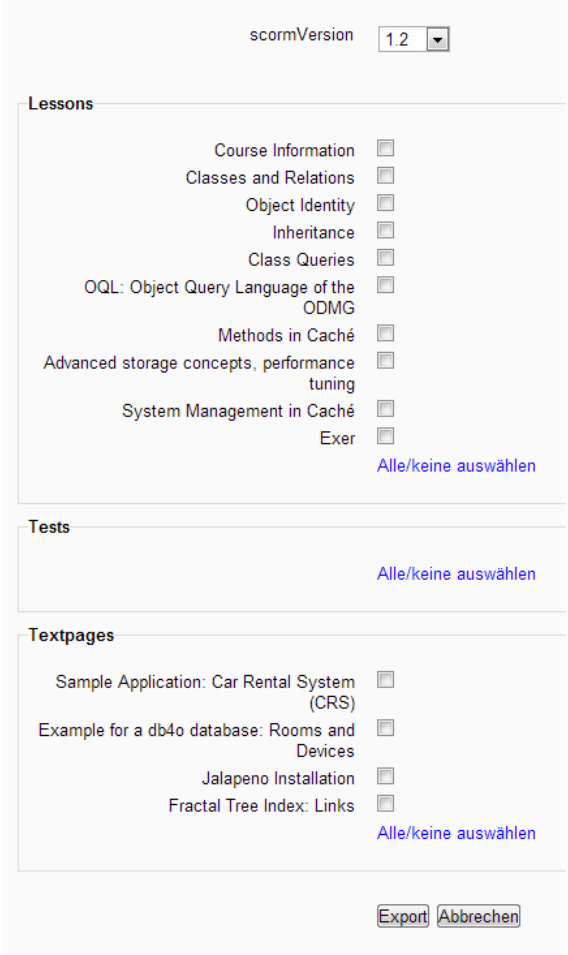

Abbildung 6: Screenshot: SCORM Export Menü

4.3 Block

4.3.1 Klassen

Die folgenden Klassen wurden für die Programmierung des Blocks benötigt. Abb. 7 zeigt das zugehörige Klassendiagramm.

- **block_scormexport**
  erbt von der Klasse block_list. Hier werden die Eigenschaften des Blockes definiert. Folgende Methoden wurden überschrieben:
  - *init()*
    erste Methode, die nach der Erzeugung eines Blocks aufgerufen wird.
  - *get_content()*
    definiert den Inhalt des Blocks. Dazu gehört der Header, der Inhalt an sich und der Footer
  - *applicable_formats()*
    legt fest, in welchen Moodle Bereichen der Block sichtbar ist. Da ich Lernmodule aus Kursen exportiere, habe ich mich dazu entschieden, den Block nur in Kursen anzuzeigen. Das Kursformat (social oder weeks) spielt dabei keine Rolle.
  - *has_config()*
    gibt an, ob der Block eine Konfigurationsseite besitzt.
- **block_list**
  erbt von block_base. Gedacht für Blöcke, die Elemente listenartig anordnen (z.B. Navigation oder Einstellungen).
- **block_base**
  Diese Klasse muss von jedem Block eingebunden werden. Sie enthält allgemeine Funktionen, die für jeden Blocktyp gelten

4.4 Benutzeroberfläche

4.4.1 Klassen

Folgende Klassen wurden für die Programmierung der Oberfläche benötigt. Abb.8 zeigt die Zusammenhänge.

- **scormexport_form**
  erbt von der Klasse moodleform. Die Checkboxen für Lektionen, Tests und Textpages werden hier gezeichnet. Buttons zum Absenden der Benutzereingaben werden hinzugefügt.

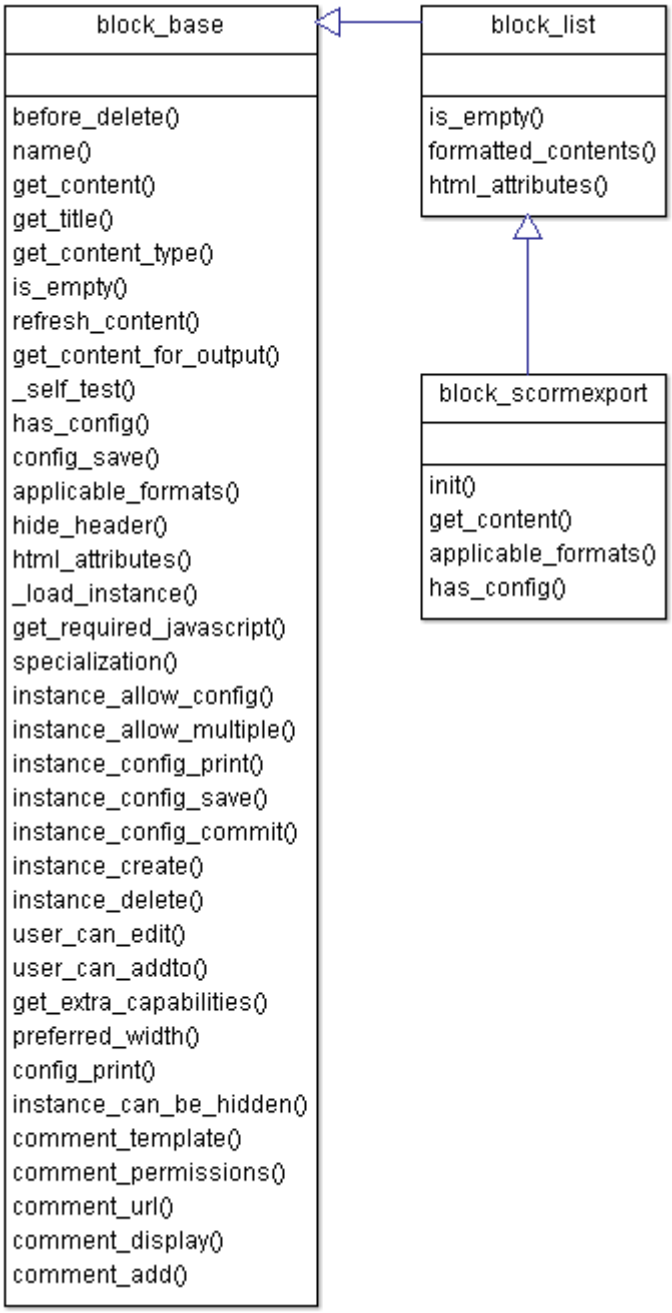

Abbildung 7: Klassendiagramm für den Block

  - *definition()* Die Funktion *definition()* muss implementiert werden, da diese in *moodleform* als abstrakt[36] deklariert ist. In dieser Funktion wird die Oberfläche gezeichnet. Über den Konstruktor der Formularklasse besteht die Möglichkeit, Werte zu übergeben. Das ist vor allem dann sinnvoll, wenn die Oberfläche abhängig vom Kursinhalt oder anderen Parametern ist. Über das Attribut *_customdata* kann auf diese Werte zugegriffen werden. In meinem Programm ist *_customdata* ein zweidimensionales Array, das alle Lektionen, Tests und Textpages enthält.

- **moodleform** Abstrakte Klasse[37], die Funktionen zum Hinzufügen von Formular Elementen enthält[38]. Um eine Oberfläche zu zeichnen, muss diese Klasse geerbt werden.

4.5 Logik

4.5.1 Klassen

- **scormData**
  Abstrakte Klasse, die Funktionen definiert, welche für jedes SCORM Lernmodul implementiert werden müssen. Dazu gehören u.a. die abstrakten Funktionen: *setTitle(title), setName(name), setId(id)*. Die Funktionen wurden als protected deklariert, da sie in den Unterklassen sichtbar sein müssen. Desweiteren enthält scormData die Funktion *getHeaderInformation(param)*, die den Quellcode der benötigten Javascript Dateien zurückgibt. Dazu gehören:
  - SCORM API Wrapper
  - Funktionen zur Auswertung von Tests und Lektionen
  - Funktionen zur Steuerung des SCO

  Der Parameter *param* gibt die verwendete SCORM Version an. Das ist notwendig, da die Wrapper Funktionen für SCORM 1.2 und 2004 unterschiedlich sind.

- **scormLesson**
  Erbt von scormData. Jede zu exportierende Lektion stellt eine Instanz vom Typ scormLesson dar.

[36] Abstrakte Methoden werden in der Oberklasse definiert und müssen von Unterklassen implementiert werden. Die Oberklasse gibt lediglich die Siganatur der Methode vor.
[37] Eine Klasse von der keine Instanz erzeugt werden kann.
[38] In der Moodle Form API http://docs.moodle.org/dev/Form_API wird im Detail auf die Möglichkeiten von moodleform eingegangen

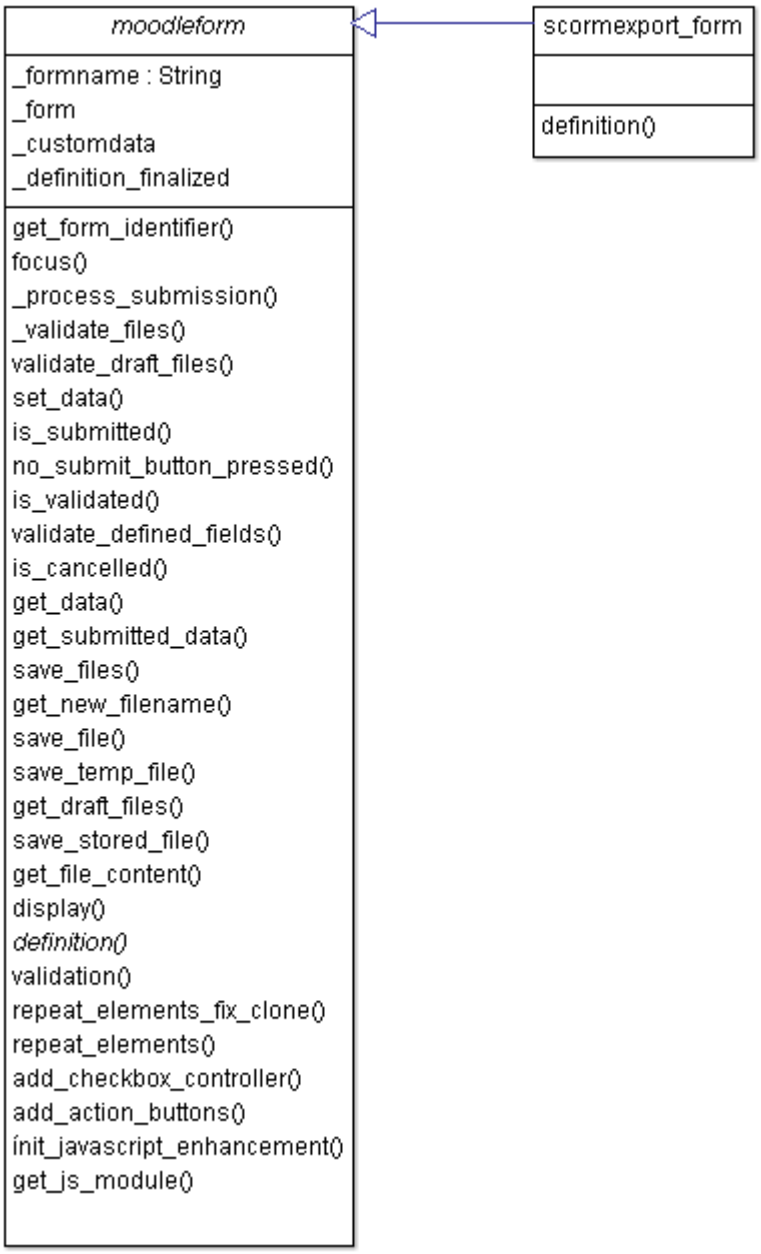

Abbildung 8: UML Klassendiagramm der Benutzeroberfläche

- **scormTest**
  Erb von scormData. Jeder zu exportierende Test stellt eine Instanz vom Typ scormTest dar.

- **htmlWriter**
  Es handelt sich hierbei um eine Hilfsklasse, die eingesetzt wird, um die SCO's zu erstellen. Enthält z.B. Methoden wie *openTag()* oder *addInputTag()*. Über *getHtmlString()* wird der erzeugte HTML Code zurückgegeben. Durch die Verwendung der htmlWriter Klasse wird der Quellcode angenehmer zu lesen, da keine Programmiersprachen „vermischt" werden. Außerdem wird die Wartbarkeit des Programms einfacher. Das ist ein wesentlicher Aspekt, da diese Arbeit nach Vollendigung der Moodle Community zur Verfügung gestellt wird.

- **apiWrapper12** Enthält Wrapper Funktionen des SCORM API[39] (Version 1.2), z.B. *doLMSInitialize()*. Das Attribut *content* enthält die Funktionen als String. Über die Methode *getWrapperFileAsString()* kann auf diese zugegriffen werden.

- **quizFuncs** Diese Klasse beinhaltet alle Javascript Funktionen, zur Auswertung von Tests (innerhalb von Lektionen). Nach der Instantiierung, kann über die Methode *getQuizFunctionsAsString()* auf die Funktionen zugegriffen werden. Die Datei mit den quizFuncs ist im Ordner assets, des PIF zu finden.

- **scoFuncs** Beinhaltet Javascript Funktionen, die für die Steuerung der SCO's benötigt werden. Wird ein SCO aufgerufen, wird *loadPage()* aufgerufen. Daraufhin wird ein Timer gestartet. Das ist z.B. dann notwendig, wenn das Lernmodul zeitlich begrenzt ist. Wird das SCO verlassen, wird der Timer gestoppt und die *session_time* dem API Adapter mitgeteilt. Über die Funktion *unloadPage(param)* wird das SCO wieder „entladen"'. Der Parameter *param* gibt an, mit welchem Status das SCO verlassen wurde. Mögliche Werte sind z.B. completed oder incomplete. Die Datei scoFuncs beinhaltet noch mehr Funktionen. Das oben beschriebene Szenario sollte nur zum Verständis dienen.

- **testFuncs** Diese Klasse ist ähnlich aufgebaut wie quizFuncs, mit dem Unterschied, dass hier Tests ausgewertet werden (bezogen auf das gleichnamige Lernmodul).

- **pageFuncs** Enthält Funktionen zum Export von Lektionsseiten. Grundsätzlich wird zwischen Frage und Inhaltsseiten unterschieden. Der Ablauf innerhalb der Methoden ist, unabhängig vom Seitentyp, immer ähnlich.

[39] Die Wrapper Funktionen wurden übernommen von: http://www.adlnet.gov/resources/official-adl-scorm-api-wrappers?type=software_downloads

1. Inhalt des SCO erzeugen. Die erstellte HTML Datei wird dem PIF hinzugefügt.
2. Zugehörige Einträge in die Manifest Datei schreiben
3. Assets, die von dem SCO benötigt werden (z.B. Bilder,Videos etc.) , dem PIF hinzufügen

- **typeFuncs** Hier werden die Methoden zum Export von Tests definiert. Ablauf:
  1. Die Methode *addHeader()* wird aufgerufen. Diese fügt den SCO's den *<head>* Bereich hinzu. Da der Header für jedes SCO gleich ist (Metadaten, einbinden von externen JS Dateien), habe ich mich dazu entscheiden, diesen in eine eigene Methode auszulagern. Das erspart dem Programmierer Schreibarbeit.
  2. Die SCO und Assets werden erzeugt und dem PIF hinzugefügt. Der Ablauf gestaltet sich ähnlich wie bei der pageFuncs Klasse.
  3. Für das Hinzufügen von Assets und Manifest Einträgen existiert hier eine eigene Methode.
- **errorLog** Klasse, die zum Error Handling benötigt wird. Enthäld die Methode *writeSyslog()*, die zum Schreiben von Logfiles verwendet wird.
- **manifestWriter** Hilfsklasse, die zum Schreiben von Manifest Einträgen benötigt wird. Bisher werden nur SCORM 1.2 Elemente unterstützt. Falls zukünftig auch ein SCORM 2004 Export möglich sein soll, kann diese Klasse geerbt und erweitert werden. Beinhaltet z.B. Methoden wie *addSchema()* oder *setMaxtime()*.
  Während der gesamten Laufzeit wird nur eine manifestWriter Instanz zu Beginn des Programms erzeugt. Nach Vollendigung aller Exportfunktionen wird über *getXmlAsString()* der XML String zurückgegeben. Daraufhin wird die imsmanifest.xml erzeugt und dem PIF hinzugefügt.
- **ZipStream**[40] Ermöglicht das Generieren eines ZIP Archivs „on the fly“. Durch die Verwendung dieser Klasse, wird die Datei nicht mehr auf dem Server abgespeichert, sondern direkt als Stream per HTTP an den Client gesendet. Das ist mit der ZipArchive Klasse von PHP nicht möglich[41]. Durch den Aufruf von *close()* wird die Datei und alle enthaltenen Komponenten, auf dem Server abgelegt.

[40] Basiert auf: http://www.phpclasses.org/package/2322-PHP-Create-ZIP-file-archives-and-serve-for-download.html
[41] http://www.php.net/manual/de/function.ziparchive-close.php

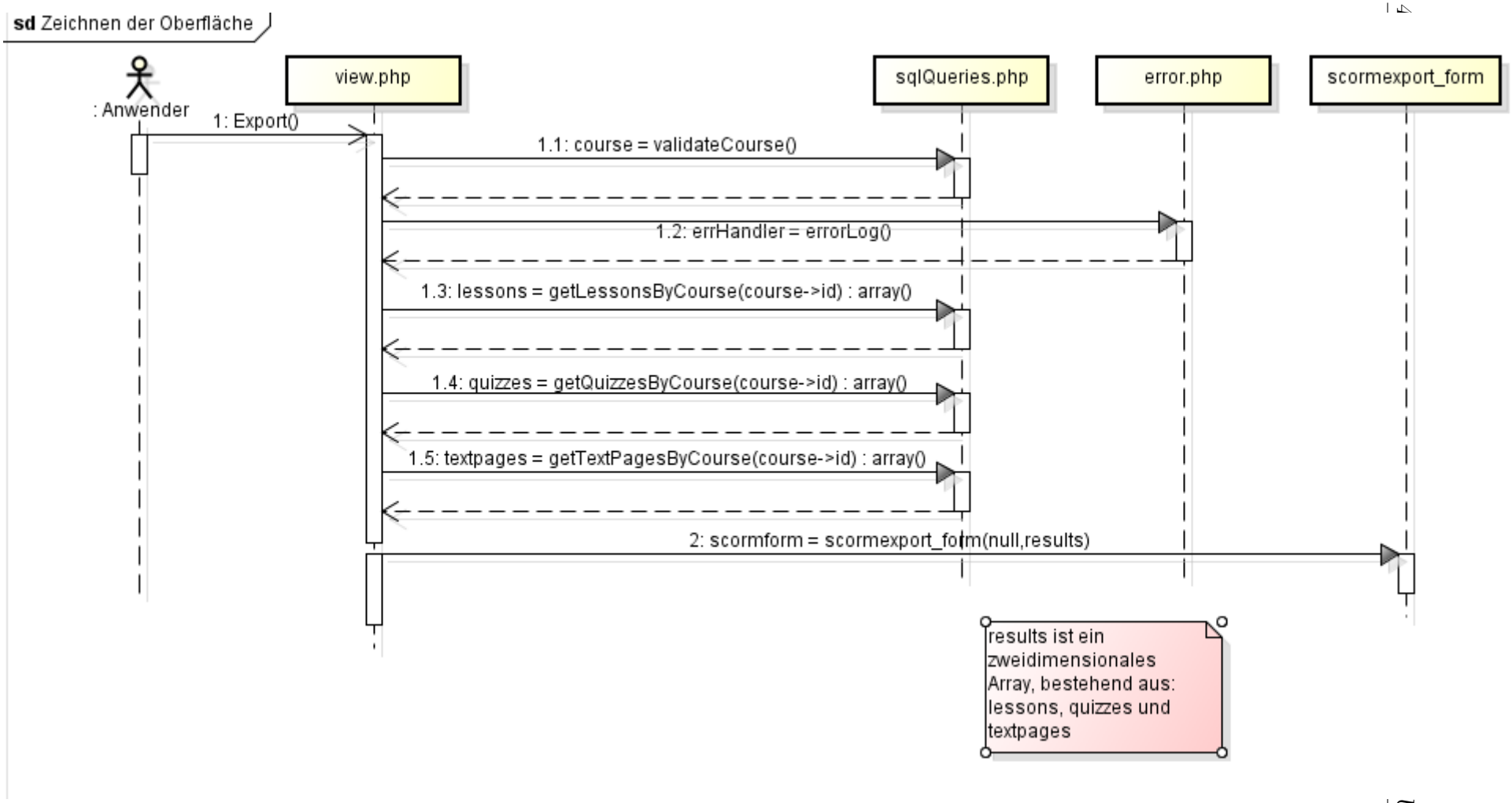

Abbildung 9: Sequenzdiagramm für das Zeichnen der Oberflaeche

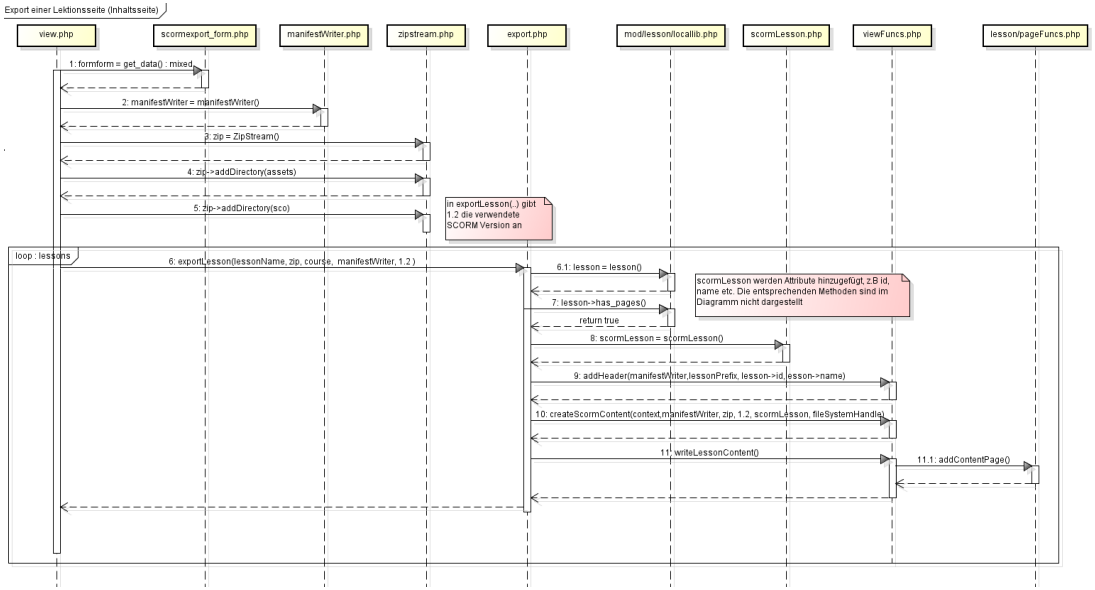

Abbildung 10: Sequenzdiagramm für den Export einer Lektionsseite

Das Sequenzdiagramm in Abb. 10 schliesst sich vom Programmablauf direkt an 9 an.

5 Tests

Um das entwickelte Programm zu testen wurden die generierten Content Packages auf verschiedenen Plattformen getestet.

5.1 SCORM Cloud

Bei der SCORM Cloud[42] handelt es sich um ein Repository für Lernpakete. Dadurch ist es möglich:

- SCORM 1.2 und 2004 Pakete zu importieren und abzuspielen
- Die Pakete in der Cloud mit anderen Personen zu teilen
- SCORM 2004 Pakete als SCORM 1.2 zu exportieren.

Durch die Verwendung bieten sich einige Vorteile. Zum einen können im SCORM Player von Moodle nun auch SCORM 2004 Inhalte angezeigt werden. Desweiteren kann von überall auf die erstellten Kurse zugegriffen werden.
Die SCORM Cloud bietet zahlreiche Möglichkeiten zur Konfiguration. Sie kann bspw. um Apps erweitert werden. Derzeit existieren Plugins für Moodle[43] und Ilias[44], die es ermöglichen, SCORM Pakete direkt aus dem LMS in die Cloud zu übertragen
Befindet sich ein Lernpaket in der Cloud, ändert sich das Look and Feel des Kurses(Abb.11).

[42] https://cloud.scorm.com
[43] SCORM Cloud Plugin für Moodle: https://github.com/RusticiSoftware/SCORMCloud_MoodleModule/downloads
[44] SCORM Cloud Plugin für ILIAS: https://github.com/RusticiSoftware/SCORMCloud_IliasPlugin

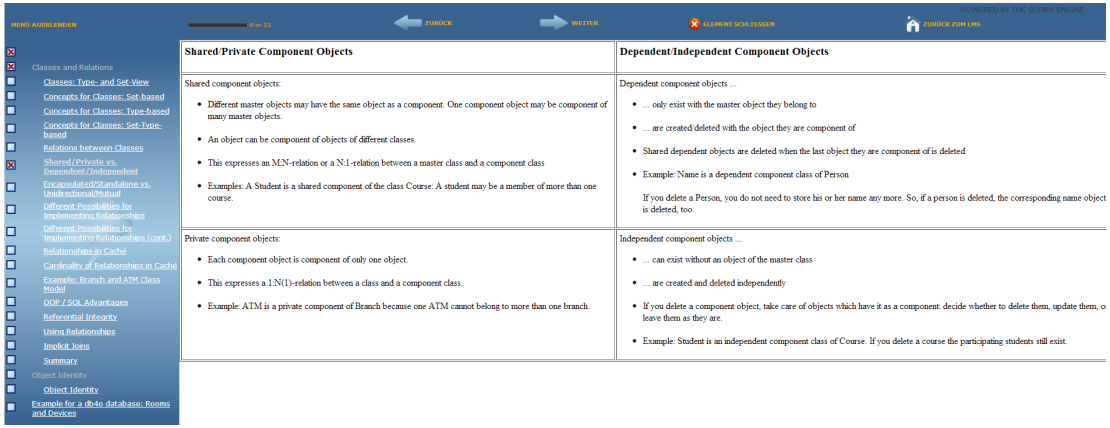

Abbildung 11: Screenhot: Exportierter Moodle Kurs in der SCORM Cloud

5.2 Moodle

Auch Moodle bietet sich als vollwertige Testplattform an. SCORM PIFs können dort als Lernpakete inportiert und abgespielt werden. Damit Fehlermeldungen und Warnungen angezeigt werden, sollte der Tester in den Debugging Modus wechseln. Das ist möglich unter:

Website Administration->Entwicklung->Debugging

Dort wird die Option Debug-Meldungen auf „ALLE“ gesetzt. Unter *Entwicklung* ist es weiterhin möglich, PHP Unit Tests durchzuführen. Aufgrund der zeitlichen Beschränkung, war es mir nicht mehr möglich Testfälle zu definieren. Dieser Punkt wird deshalb auf die nächste Phase verschoben.

6 Zusammenfassung und Ausblick

Das Ziel dieser Bachelorarbeit war es einen Moodle Block zu programmieren, über den der Export von Lernmodulen in das SCORM Format möglich sein soll. Darüberhinaus soll der Leser einen Einblick in die Moodle Programmierung und die Möglichkeiten von SCORM erhalten.

Der Stand des Projekts zum Ende dieser Arbeit, entspricht den definierten Anforderungen zu Beginn. Mindestens Lektionen und Tests sollten exportiert werden. Die Möglichkeit Lernmodule als SCORM 2004 Paket zu speichern, war anfangs eingeplant wurde aber nicht implementiert, da Moodle diesen Standard noch nicht vollständig unterstützt. Es wurde allerdings darauf geachtet, dass ein Upgrade möglichst einfach zu programmieren ist.

Aufgrund von Schwierigkeiten in der Entwicklungsphase, konnten nicht alle Ziele erreicht werden. Es ist zum jetzigen Zeitpunkt nicht möglich „berechnete Fragen“ von Tests zu exportieren, da notwendige Einträge in der Datenbank fehlen. Auch der Import in ILIAS funktioniert noch nicht. Allerdings können dort auch die „offiziellen“ Testkurse von ADL nicht importiert werden, was wiederum auf serverspezifische Probleme hindeutet

Der Quelltext des Programms wurde nach Fertigstellung der Moodle Community zur Verfügung gestellt. Somit ist sichergestellt, dass das Projekt weiterhin gepflegt wird.

7 Anhang

7.1 Data Model Elements

Element	Eigenschaften
cmi.core._children	liefert alle unterstüzten Elemente als komma separierte Liste
	LMSGetValue()
	CMIString255
cmi.core.student_id	gibt die student id des angemeldeten Nutzers zurück
	LMSGetValue()
	CMIIdentifier
cmi.core.student_name	liefert den Namen des angemeldeten Nutzers
	LMSGetValue()
	CMIString255
cmi.core.lesson_location	setzt bzw. liefert den Teil des SCO, den der Nutzer zuletzt bearbeitet hat. Der Nutzer kann dadurch die aktuelle Session unterbrechen und zu einem späteren Zeitpunkt, am gleichen Punkt des SCO weiterarbeiten.
	LMSGetValue(), *LMSSetValue()*
	CMIString255
cmi.core.credit	liefert credit oder no-credit zurück. Je nachdem ob das SCO, dass der Nutzer bearbeitet benotet wird oder nicht
	LMSGetValue()
	CMIVocabulary
cmi.core.lesson_status	erlaubt das Setzen bzw. Abfragen eines Status. Erlaubte Werte sind: passed, completed, failed, incomplete, browsed und not attempted
	LMSGetValue(), *LMSSetValue*
	CMIVocabulary
cmi.core.entry	betritt der Lernende ein SCO zum ersten Mal, wird dem Entry Flag der Wert ab-initio zugewiesen. Wird die Session unterbrochen und zu einem anderen Zeitpunkt fortgesetzt, trägt das Flag den Wert Resume. Die Werte können vom Kursersteller lediglich abgefragt werden. Das Setzen übernimmt das LMS.
	LMSGetValue()

	CMIVocabulary
cmi.core.score._children	liefert alle Kindelemente von score zurück, die vom LMS unterstützt werden.
	LMSGetValue()
	CMIString255
cmi.core.score.raw	gibt an, inwieweit sich der Student im Vergleich zum vorherigen Versuch verbessert/-verschlechtert hat.
	LMSGetValue(), *LMSSetValue()*
	CMIDecimal or CMIBlank
cmi.core.score.max	die Maximalpunktzahl, die der Studierende erreichen kann.
	LMSGetValue, *LMSSetValue*
	CMIDecimal or CMIBlank
cmi.core.score.min	die Minimalpunkzahl, die der Studierende erreichen muss
	LMSGetValue(), *LMSSetValue()*
	CMIDecimal or CMIBlank
cmi.core.total_time	Gesamtzeit, die für die Lektion/Test nötig war (Summe aller SCO Session Times)
	LMSGetValue()
	CMITimespan
cmi.core.lesson_mode	gibt den Modus an, in dem das SCO ausgeführt wird (browse, normal, review).
	LMSGetValue
	CMIVocabulary
cmi.core.exit	gibt den Ursache an, warum das SCO beendet wurde
	LMSSetValue
	CMIVocabulary
cmi.core.session_time	Zeit, die der Nutzer für die Bearbeitung eines SCO benötigt hat
	LMSSetValue()
	CMITimespan

cmi.comments	erlaubt das Absetzen/Abfragen von Kommentaren an das LMS
	LMSGetValue(), *LMSSetValue*
	CMIString4096
cmi.comments_from_lms	erlaubt das Abfragen von Kommentaren über ein bestimmtes SCO.
	LMSGetValue()
	CMIString4096
cmi.objectives._children	listet alle Kindelemente von objectives auf, die vom LMS unterstützt werden.
	LMSGetValue()
	CMIString255
cmi.objectives._count	gibt die Anzahl der Kindelemente von objectives zurück.
	LMSGetValue()
	CMIInteger
cmi.objectives.n.id	identifiziert ein Objective
	LMSGetValue()
	CMIIdentifier
cmi.objectives.n.score._children	listet alle Kindelemente von objectives.n.score auf, die vom LMS unterstützt werden
	LMSGetValue()
	CMIString255
cmi.objectives.n.score.raw	macht die Fortschritte des Lernenden in Bezug auf vorherigen Versuch deutlich
	LMSGetValue(), *LMSSetValue()*
	CMIDecimal or CMIBlank
cmi.objectives.n.score.max	maximal zu erreichende Punktzahl für ein Objective
	LMSGetValue(), *LMSSetValue()*
	CMIDecimal or CMIBlank
cmi.objectives.n.score.min	minimale Punktzahl, die der Student erreichen muss

	LMSGetValue(), *LMSSetValue*
	CMIDecimal or CMIBlank
cmi.objectives.n.status	Informationen über den Status eines Objective. Mögliche Werte sind: passed,completed, failed, incomplete, browsed und not attempted
	LMSGetValue(), *LMSSetValue()*
	CMIVocabulary
cmi.student_data._children	listet alle Kindelemente von student_data auf, die vom LMS unterstützt werden
	LMSGetValue()
	CMIString255
cmi.student_data.mastery_score	Punktzahl, die mindestens benötigt wird, um die Lektion zu bestehen
	LMSGetValue()
	CMIDecimal
cmi.student_data.max_time_allowed	gibt das Zeitlimit für eine Lektion/Test zurück. Falls ein Limit gesetzt ist, wird dieses in der Manifest Datei definiert.
	LMSGetValue()
	CMITimespan
cmi.student_data.time_limit_action	Aktion, die ausgeführt werden soll falls die Zeitgrenze überschritten wurde. In der Manifest Datei wird der Wert dieses Elements definiert.
	LMSGetValue()
	CMIVocabulary
cmi.student_preference._children	gibt alle Kindelemente von cmi.student_preference zurück, die das LMS unterstützt
	LMSGetValue()
	CMIString255
cmi.student_preference.audio	Über dieses Element können Audio Einstellungen gesetzt werden Mögliche Optionen sind: lautlos=-1, lautstärke(1-100) oder keine Änderungen vornehmen (0)

	LMSGetValue(), *LMSSetValue*
	CMIInteger
cmi.student_preference.language	für mehrsprachige SCO's geeignet. Mögliche Parameter wären z.B. French oder English.
	LMSGetValue(), *LMSSetValue()*
	CMIString255
cmi.student_preference.speed	die Geschwindigkeit in der Content übermittelt wird, kann dadurch geändert werden.
	LMSGetValue(), *LMSSetValue()*
	CMIInteger
cmi.student_preference.text	vor allem für audiobasierte Lektionen ist dieses Element wichtig. Parameter: -1 = nur audio, 0 = keine Änderung, 1 = Text wird zusätzlich zur Audiospur angezeigt.
	LMSGetValue(), *LMSSetValue()*
	CMIInteger
cmi.interactions._children	listet alle Kindelemente von interactions auf, die vom LMS unterstützt werden.
	LMSGetValue()
	CMIString255
cmi.interactions._count	liefert die Anzahl der Interaktionen, die in Gebrauch sind.
	LMSGetValue()
	CMIString255
cmi.interactions.n.id	erlaubt das Zuweisen einer Id für eine Interaktion. Die Id muss eindeutig sein.
	LMSSetValue()
	CMIIdentifier
cmi.interactions.n.objectives._count	liefert die Anzahl der gespeicherten Objectives, für eine Interaktion
	LMSGetValue()
	CMIInteger
cmi.interactions.n.objectives.n.id	erlaubt das Setzen einer Id für ein Objective
	LMSSetValue()
	CMIIdentifier

cmi.interactions.n.time	gibt an, wann die Interaktion beendet wurde
	LMSSetValue()
	CMITime
cmi.interactions.n.type	gibt an um welche Art von Interaktion es sich handelt. Erlaubte Werte sind: true-false, choice, fill-in ,matching, performance, sequencing, likert, numeric
	LMSSetValue()
	CMIVocabulary
cmi.interactions.n.correct_responses._count	liefert die Anzahl an korrekten Antworten, für eine Interaktion
	LMSGetValue()
	CMIInteger
cmi.interactions.n.correct_responses.n.pattern	definiert die Antwortmöglichkeiten für eine Interaktion.
	LMSSetValue()
	CMIFeedback
cmi.interactions.n.weighting	gibt die Gewichtung der Interaktion an
	LMSSetValue()
	CMIDecimal
cmi.interactions.n.student_response	übermittelt die gewählte Option an das LMS
	LMSSetValue()
	CMIFeedback
cmi.interactions.n.result	setzt das Ergebnis einer Interaktion. Mögliche Werte: correct, wrong, unanticipated, neutral, x.x
	LMSSetValue()
	CMIVocabulary
cmi.interactions.n.latency	die Zeit von dem Laden der Interaktion bis zur Beantwortung
	LMSSetValue()
	CMITimespan

7.2 Errorcodes

Kategorie	**Bereich**
No Error	0
General Errors	100 - 199
Syntax Errors	200 -299
RTS Errors	300 - 399
Data Model Errors	400 - 499
Implementation-defined Errors	1000 - 65535

Tabelle 6: Zahlenbereiche der unterschiedlichen Fehlertypen (SCORM 2004)

SCORM 1.2	**SCORM 2004**
0 - No Error	0 - No Error
101 - General Exception	101 - General Exception
	102 - General Initialization Failure
	103 - Already Initialized
	104 - Content Instance Termninated
	111 - General Termination Failure
	112 - Termination Before Initialization
	113 - Termination After Termination
	122 - Retrieve Data Before Initialization
	123 - Retrieve Data After Termination
	132 - Store Data Before Initialization
	133 - Store Data After Termination
	142 - Commit Before Initialization
	143 - Commit After Termination
201 - Invalid Argument Error	201 - General Argument Error
202 - Element cannot have children	
203 - Element not an array. Cannot have count	
301 - Not Initialized	301 - General Get Failure
	351 - General Set Failure
	391 - General Commit Failure
401 - Not implemented error	401 - Undefined Data Model Element
402 - Invalid set value, element is a keyword	402 - Unimplemented Data Model Element
403 - Element is read only	403 - Data Model Element Value Not Initialized
404 - Element is write only	404 - Data Model Element Is Read Only
405 - Incorrect Data Type	405 - Data Model Element is Write Only
	406 - Data Model Element Type Mismatch
	407 - Data Model Element Value Out Of Range
	408 - Data Model Dependency Not Established

Tabelle 7: Vergleich der Errorcodes zwischen SCORM 1.2 und 2004

Literatur

[1] ADL: *The SCORM Content Aggregation Model*. PDF Dokument, 2001. Online verfügbar unter: http://www.adlnet.gov/resources/SCORM-1-2-Specification?type=technical_documentation.

[2] ADL: *The SCORM Run-Time Environment*. PDF-Dokuments, 2001. Online verfügbar unter: http://www.adlnet.gov/resources/SCORM-1-2-Specification?type=technical_documentation.

[3] Docs, Moodle: *Blocks/Appendix A*. Website, 2011. Online verfügbar unter: http://docs.moodle.org/dev/Blocks/Appendix_A#after_install.28.29.

[4] Docs, Moodle: *Aufbau einer Moodle-Site*. Website, 2013. Online verfügbar unter: http://docs.moodle.org/25/de/Aufbau_einer_Moodle-Site.

[5] Docs, Moodle: *Blocks*. Website, 2013. Online verfügbar unter: http://docs.moodle.org/dev/Blocks.

[6] Docs, Moodle: *Data Manipulation API*. Website, 2013. Online verfügbar unter: http://docs.moodle.org/dev/Data_manipulation_API.

[7] Docs, Moodle: *File API*. Website, 2013. Online verfügbar unter: http://docs.moodle.org/dev/File_API#Read_file.

[8] Docs, Moodle: *SCORM FAQ*. Website, 2013. Online verfügbar unter: http://docs.moodle.org/23/en/SCORM_FAQ.

[9] Docs, Moodle: *Setting up Eclipse*. Website, 2013. Online verfügbar unter: http://docs.moodle.org/dev/Setting_up_Eclipse.

[10] Fremaux, Valery: *Moodle Technical Documentation*. Website, 2013. Online verfügbar unter: http://phpdocs.moodle.org/HEAD/index.html.

[11] Jesukiewicz, Paul: *Content Aggregation Model [CAM]*. PDF-Dokument, 2009. Online verfügbar unter: http://www.adlnet.gov/resources/SCORM-2004-4th-Edition-Specification?type=technical_documentation.

[12] Jesukiewicz, Paul: *Run-Time Environment [RTE]*. PDF-Dokument, 2009. Online verfügbar unter http://www.adlnet.gov/resources/SCORM-2004-4th-Edition-Specification?type=technical_documentation.

[13] Jesukiewicz, Paul: *Sequencing and Navigation [SN]*. PDF-Dokument, 2009. Online verfügbar unter: http://www.adlnet.gov/resources/SCORM-2004-4th-Edition-Specification?type=technical_documentation.

[14] Moodle: *Moodle Developer Courses*. Website, 2013. Online verfügbar unter: http://dev.moodle.org/login/index.php (Anmeldung erforderlich).

[15] PHPXref: *PHP Cross-Reference for Moodle*. Website, 2013. Online verfügbar unter: http://xref.schoolsict.net/moodle/2.3/nav.html?index.html.

[16] Wikipedia: *SCORM Artikel in der deutschsprachigen Wikipedia*. Website, 2013. Online verfügbar unter: http://de.wikipedia.org/wiki/Scorm.