Standards im e-learning Umfeld: SCORM

Ausarbeitung zum Seminarvortrag

Steffen Glückselig

Inhaltsverzeichnis

1. Was ist SCORM und was sind dessen Ziele?

Die Abkürzung **SCORM** steht für *Sharable Content Object Reference Model*. Es handelt sich bei SCORM um eine Sammlung von Spezifikationen und Standards, die aus mehreren Quellen (siehe Abschnitt 3, „Geschichte von SCORM“) zusammengetragen wurden, um eine verständlichen Satz von **web**-basierten *e-learning*-Lösungen zu definieren (siehe [1] und [2], Seite 1-21).

SCORM beschreibt ein 'Content Aggregation Model' und ein 'Run-Time Envrionment' für Lernobjekte, um adaptives Instruieren basierend auf den Zielen des Lerners, seinen Vorlieben, seiner Leistung und anderen Faktoren zu unterstützen (siehe [2], Seite 1-21), wie es im CBI[i] und ITS[ii]-Umfeld verlangt wird.

SCORM wurde für das US-Militär durch die ADL[iii]-Initiative entwickelt. Im Jahr 2000 wurde der Standard veröffentlicht (siehe [6] und Abschnitt 3, „Geschichte von SCORM“).

SCORM besteht aus vier Dokumenten, die zusammengehörende und aneinander angepasste Sammlungen von Spezifikationen und Standards darstellen und in der SCORM-Terminologie *Bücher* genannt werden. Jedes Buch wird seit *SCORM 2004* **eigenständig versioniert**.

[i] *Computer-Basiertes Instruieren*
[ii] *Intelligente Tutor-Systeme*
[iii] **Advanced Distributed Learning**

Die vier SCORM-Bücher

1. **Overview**
2. **Content Aggregation Model (CAM)**
3. **Run-Time Environment (RTE)**
4. **Sequencing and Navigation (SN)**

Das erste Buch - **Overview** - bietet einen *Überblick* über alle weiteren Bücher. Es beschreibt die Herkunft und Ziele des Standards. Auf technische Details des Standards wird in den nachfolgenden Büchern eingegangen.

Im **CAM**-Buch werden Schlüsselkonzepte (wie z.B. die SCO[iv]s) definiert und genauer beschrieben. Desweiteren wird das Zusammenstellen, das Auszeichnen[v] für eine automatisierte Suche innerhalb der Lerninhalte, das Verpacken[vi] für den Austausch zwischen Systemen und die Regeln zur Ablaufsteuerung zwischen Komponenten beschrieben. (cf. [2], Seite 1-29).

Das **RTE**-Buch betrachtet hauptsächlich das Starten von aktivem Inhalt (SCOs) durch das LMS[vii], die Kommunikation und den Datentransfer zwischen Inhalt und LMS, das Verfolgen[viii] und Fehlerbehandlung (cf. [2], Seite 1-27 f.).

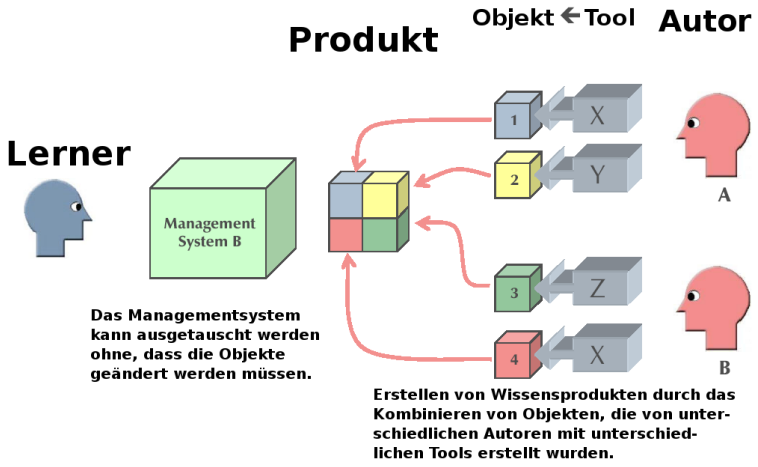
Das **SN**-Buch behandelt die Details der *Ablaufsteuerung und der Navigation* in SCORM. Das heißt es beschreibt, wie SCORM-konformer Inhalt traversiert werden kann und zwar anhand von *lerner*- und/ oder *system*-initiierten Navigationsereignissen[ix] .

2. Motivation von SCORM

[iv] *Sharable Content Object*
[v] Beispielsweise sind *Metadaten* eine Art der Auszeichnung.
[vi] Das Verstauen der physikalischen Dateien in einem ZIP-Archiv.
[vii] *Learning Management System*
[viii] **tracking**: das Speichern von Informationen über die Interaktionen des Lerners mit den Inhaltsobjekten und Aktivitäten (siehe weiter unten).
[ix] **Navigationsereignisse** sind beispielsweise Vorwärts, Rückwärts und die Navigation durch *Wahl* in einem Menü.

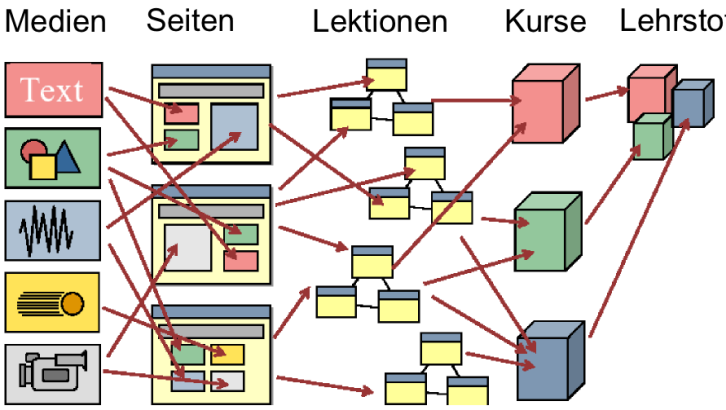
Neben der Zusammenführung und Vereinheitlichung der Arbeiten verschiedener Organisationen im Bereich der e-learning-Technologien, waren folgende Ansprüche weitere Motive für den Standard (siehe [2], Seite 1-22):

- **Zugänglichkeit**: die Möglichkeit instruktionale Komponenten von einem entfernten Ort zu lokalisieren, auf sie zuzugreifen und sie an viele andere Orte zu liefern. Zugänglichkeit ermöglicht so die *Verteiltheit* von Lernangeboten.
- **Anpassungsfähigkeit**: die Möglichkeit Instruktionen an individuelle oder organisationelle Gegebenheiten anzupassen. D.h. die *Adaptivität* der Lernerfahrung.
- **Erschwinglichkeit**: die Möglichkeit Effizienz und Produktivität zu erhöhen während die benötigte Zeit und somit die Kosten für das Liefern der Instruktionen reduziert werden.
- **Beständigkeit**: die Möglichkeit technologischer Entwicklung und Änderungen ohne kostenintensives Redesign, Rekonfiguration oder Neuprogrammierung standzuhalten.
- **Interoperabilität**: die Möglichkeit an einem Ort mit einer Menge an Werkzeugen entwickelte instruktionelle Komponenten zu nehmen und sie an einem anderen Ort mit anderen Werkzeugen zu verwenden.



Konzept der Interoperabilität (cf. [21])

- **Wiederverwendbarkeit**: die Flexibilität instruktionelle Komponenten in mehreren *Anwendungen* und *Zusammenhängen* einzubinden.



Konzept der Wiederverwendbarkeit (cf. [21])

Neben der Wiederverwendbarkeit der einzelnen Inhalte ist die Wiederverwendbarkeit für verschiedene *Ausgabemedien* ebenfalls wichtig. SCORM ist zwar hauptsächlich ein Standard für web-basierte Lerninhalte, diese Lerninhalte sollen (und sind) jedoch auch einfach auf andere Medien, wie z.B. CD-ROM, übertragbar.

3. Geschichte von SCORM

Die unüberschaubare Menge von Standards im Bereich des e-learnings und der Metadaten führte zum Wunsch einer Vereinheitlichung. Dieses Streben nach einer Vereinheitlichung führte im Januar 1999 zur Unterzeichnung einer Durchführungsverordnung, die das *Verteidigungsiminsterium* der Vereinigen Staaten von Amerika an die Spitze verschiedener Bundesbehörden und privater Organisationen stellte, die allgemeingültige Spezifikationen und Standards für technologiebasiertes Lernen erarbeiten sollten. Anfang 1999 war ein erster Entwurf des *Sharable Content Object Reference Model* entiwckelt. Dieser Standard sollte die Arbeit dieser Organisationen integrieren und die **Advanced Distributed Learning**-Initiative des Verteidigungsministeriums unterstützen. ADL entwickelte SCORM um diese Fülle an sich abzeichnenden Standards zusammenzubringen und in ein gemeinsames *Referenz*modell zu integrieren. (cf. [7])

Die ADL-Initiative selbst wurde im November 1997 durch das amerikanische Verteidigungsministerium und dem *Weißen Haus Büro für Wissenschafts- und Technologiepolitik* [x] gegründet. Die Aufgabe der Initiative besteht darin, Zugang zu qualitativ hochwertiger Bildung und Training, die auf die Bedürfnisse des Einzelnen zugeschnitten sind und kosteneffizient jederzeit und überall angeboten werden können zu ermöglichen. (siehe [2], Abschnitt 1.1)

Die unter SCORM zusammengefassten Spezifikationen basieren auf Arbeiten von *Aviation Industry CBT*[xi] *Commitee* **(AICC)**, des *IMS*[xii] *Global Learning Consortium* **(IMS global)**, des *Institute of*

[x] White House Office of Science and Technology Policy (OSTP)

Electrical and Electronics Engineers **(IEEE)** / *Learning Technology Standards Commitee* **(LTSC)**, des *Alliance for Remote Instructional Authoring and Distribution Networks for Europe* **(ARIADNE)** und anderen (cf. [15]).

Im Januar 2000 veröffentlichte ADL Version 1.0 des SCORM-Standards. Darauf folgten erste Beispielimplementationen und Software für das Testen der Standardkonformität von Lerninhalten.

Im folgenden Jahr wurde SCORM 1.1 verabschiedet. In der Version 1.1 wurden Verbesserungsvorschläge und Korrekturen - ausgehend von Version 1.0 - einbezogen. Die Testphase von SCORM wurde beendet. Ab dieser Version steht SCORM nicht mehr für *Sharable* ***Courseware*** *Object Reference Model* sondern für *Sharable* ***Content*** *Object Reference Model*. Mit dieser Namensänderung soll verdeutlicht werden, dass der Standard für mehrere Ebenen von Inhalt einsetzbar ist und nicht bloß auf die Kursebene beschränkt ist. D.h. mit SCORM lassen sich nicht bloß Inhalte auf Kursebene beschreiben, sonder auch kleinere Einheiten (z.B. *Lektionen*) und größere Einheiten (z.B. *Lehrpläne*).

In der Version 1.2 wurden die *content packaging application profiles* eingeführt. Diese Version des SCORM erweiterte auch die Metadaten, die für die Beschreibung des Lerninhalts Verwendung finden.

SCORM 2004 führte ein viertes Buch ein, *Sequencing and Navigation*. Diese neue Version verbesserte auch die anderen drei Bücher um die Evolution der Standards wiederzuspiegeln und besser mit dem neuen vierten Buch im Einklang zu stehen. Seit dieser Version ist SCORM als stabil anzusehen. Zukünftige Versionen der vier Bücher werden Lektionen berücksichtigen, die bei der Implementierung dieser Version von SCORM gelernt wurden. (cf. [7])

SCORM dient als Grundlage für das Erstellen von *wiederverwendbaren* Lerninhalten innerhalb eines gemeinsamen Rahmens für computer- und webbasiertes Lernen (siehe [2], Abschnitt 1.1). Die CBI-Gemeinde erkannte recht bald den Vorteil von web-basierten Inhalten, nämlich die Verwendung einheitlicher Kommunikationsstrukturen, gemeinsamer Standards und den stetig wachsenden Anteil von web-Anwendungen und portierte ihre proprietären Lösungen in web-basierte Anwendungen. So ist das Hauptmedium von SCORM das Web, jedoch mit dem Hintergedanken, dass jede im Web zugängliche Information auch relative leicht auf andere Medien übertragen werden kann. So ist beispielsweise eine Auslieferung auf CD-ROM leicht zu bewerkstelligen. (cf. [16], Seite 1-13). Dies kommt dem Gedanken des *write once, deploy anyhwere* entgegen und ermöglicht es dem Entwickler sich auf Inhalte, nicht auf Technologien zu deren Auslieferung zu konzentrieren.

Spezifikationen und Standards, die in SCORM vereinigt wurden (cf. [2], Seite 1-5) sind:

- **IEEE Data Model** für die Kommunikation zwischen Inhaltsobjekt und LMS

[xi] Computer-Based Training
[xii] Instructional Management System

- **IEEE ECMAScript Application Programming Interface** für die Kommunikation zwischen Inhalt und Laufzeitdienst[xiii] for Content to Runtime Services Communication
- **IEEE Learning Object Metadata** (LOM) zum Auszeichnen der Daten mit Metainformation.
- **IMS Content Packaging** - siehe Abschnitt 4.1.3, „Content Packaging "
- **IMS Simple Sequencing** - siehe Abschnitt 4.3, „Ablaufsteuerung und Navigation"

4. Der Standard im Detail

4.1. Das *Content Aggregation Model* [xiv]

Es besteht aus drei Teilen, dem **Inhaltsmodell**, den **Meta-Daten** und dem **Verpacken von Inhalt** [xv] und wird im CAM-Buch spezifiziert.

4.1.1. Inhaltsmodell

Das Inhaltsmodell beschreibt den Inhalt, dessen Struktur und Beziehungen[xvi] . Falls der Inhalt aus mehr als einem *Modul* [xvii] besteht, beschreibt das Inhaltsmodell alle Beziehungen zwischen den verschiedenen Modulen (die *Aggregationen* genannt werden). Das Inhaltsmodell beschreibt außerdem die physische Struktur des Inhalts (benötigte Dateien, verlangte Datei*namen* und Verzeichnisstrukturen etc.). Das Inhaltsmodell besteht aus *Assets*, *SCOs* und der *Inhaltsorganisation* [xviii] (cf. [3], Seite 2-1 bis 2-7).

Sowohl Assets, als auch SCOs werden im XML der Inhaltsorganisation (in der Datei `imsmanifest.xml`, siehe Abschnitt 4.1.1.3, „Organisation") als `resource`-Elemente wiedergegeben. Dabei wird das `scormType`-Attribute entweder auf den Wert `asset` oder `sco` gesetzt.

Aktivitäten[xix] werden im XML der Inhaltsorganisation mit `item`-Elementen oder `organization`-Elementen repräsentiert. `item`-Elemente, die Blätter im Beum des Inhaltsmodells sind, verweisen per

[xiii] Entspricht der *Laufzeitumgebung* bzw. dem LMS.
[xiv] cf. [20]
[xv] content packaging
[xvi] Solche **Beziehungen** sind z.B. *Ist-Teil-Von* oder *Ist-Ressource-Für*.
[xvii] Als **Modul** wird eine Sammlung von Inhalten bezeichnet. Dies entspricht beispielsweise einem *Kurs*.
[xviii] content organization
[xix] Siehe Abschnitt 4.3.1, „Aktivitäten"

`id`-Attribut auf `resource`n. Mehr dazu findet sich unter Abschnitt 4.1.1.3, „Organisation“.

4.1.1.1. Asset

Ein Asset ist die grundlegenste From einer Lernressource. Assets sind *elektronische Repräsentationen von Medien* wie Text, Bilder, Klang, Bewertungsobjekte[xx] oder jedes andere Datum, das von einem Web-Client dargestellt und einem Lerner präsentiert werden kann. Man kann mehrere Assets zu einem neuen Asset bündeln.

Assets können mit *Metadaten* ausgezeichnet werden. Dies ermöglicht die Suche und das Finden innerhalb von Repositories und somit die Wiederverwendung. Die Metadaten werden durch das *Content Package* (siehe Abschnitt 4.1.3, „Content Packaging “) mit einem Asset assoziiert.

4.1.1.2. SCO

Ein SCO ist eine Sammlung von einem oder mehreren Assets, die eine einzige im obigen Sinn aktive Lernressource repräsentieren. Ein SCO stellt die kleinste Einheit einer Lernressource dar, die von einem LMS mittels der RTE verfolgt werden kann.

Während der Entwicklung des Inhalts sollte über eine sinnvolle Größe der einzelnen SCOs nachgedacht werden. Dabei sollte die kleinste logische Einheit, die vom LMS vernünftig verwendet werden kann, berücksichtigt werden. Generell sollte jedes *Lernziel* auf ein SCO abgebildet weden.

Der einzige Unterschied zwischen einem SCO und einem Asset besteht darin, dass ein SCO mit dem LMS über die *IEEE ECMAScript API* kommuniziert und ein Asset nicht.

Um *Wiederverwendbarkeit* zu erreichen, sollte ein SCO unabhängig vom Lernzusammenhang sein. So kann ein SCO in verschiedenen Kursen Verwendung finden. Außerdem können verschiedene SCOs zu neuen Kursen zusammengefasst werden.

SCOs können einzelne Webseiten umfassen, aber auch Module mit mehreren hundert Seiten und anderen Assets darstellen.

Auch SCOs können - wie Assets - mit Metadaten beschrieben werden. Auch hier erfolgt die Assoziation von Metadaten und SCO durch das *Content Package*.

Anmerkung: Trennung von Metadaten und Daten

Es ist auffällig, dass im Standard die Metadaten von den Daten (den Assets und SCOs) getrennt ist. Das bedeutet, dass eine Suche nach Inhalt durch die Beschreibungen des Inhalts anhand der

[xx] assessment objects

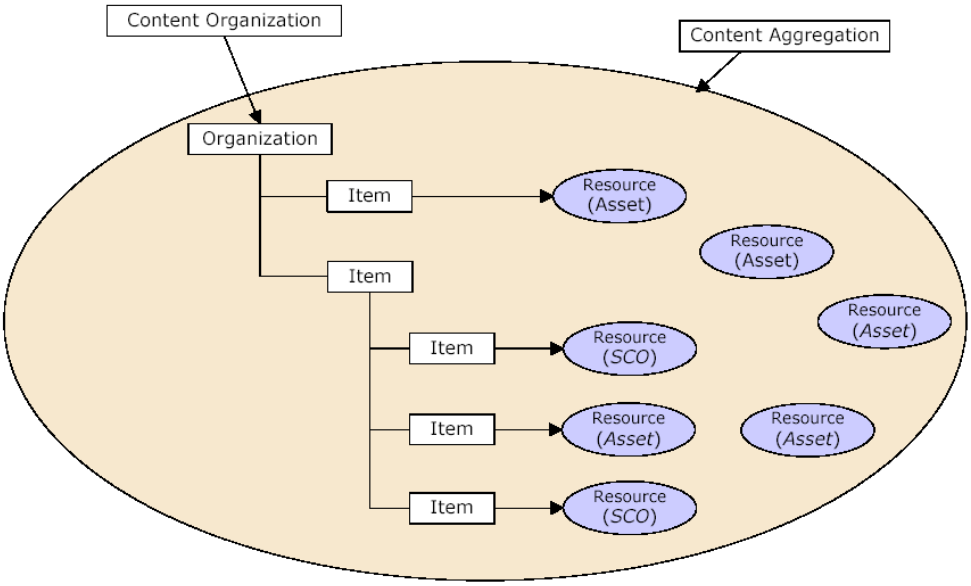
Metadaten in einer separaten Datei geschehen muss.

Das ist im Grunde so, also würde man die *Dublin Core*-Metadaten[xxi] einer Internetseite getrennt von der Seite Abspeichern.

Außerdem muss ein SCO die Anforderungen des SCORM RTE erfüllen. D.h. es muss imstande sein, die durch das LMS zur Verfügung gestellte API zu finden und muss mindestens die Funktionen zur Initialisierung und zur Terminierung aufrufen. Andere Methoden der API müssen nicht aufgerufen werden. Der Aufruf dieser zusätzlichen Methoden hängt vom Inhalt des SCO ab. Siehe hierzu Abschnitt 4.2.2, „Application Programming Interface“.

4.1.1.3. Organisation

Eine Inhaltsorganisation entspricht einer *Karte*, die den beabsichtigten Gebrauch des Inhalts darstellt. Dies wird durch das Strukturieren von Einheiten - Aktivitäten - erreicht. Die Karte zeigt ebenfalls, wie Aktivitäten zueinander in Beziehung stehen und spezifiziert, wie Aktivitäten traversiert werden können. Siehe hierzu Abschnitt 4.3.2, „Aktivitätsbäume“.



Schema einer *Content Organization*

Die in einer Inhaltsorganisation repräsentierten Aktivitäten können selbst wieder aus Aktiviäten bestehen und werden dann *Cluster* genannt (siehe Abschnitt 4.3.2.2, „Cluster“). Es ist keine Grenze für die Tiefe dieser Rekursion im Standard festgelegt. Aktivitäten, die nicht wiederum aus Aktivitäten bestehen sind mit einer Lernressource assoziiert (entweder einem SCO oder einem Asset), die dafür verwendet wird, die Aktivität auszuführen. D.h. Aktivitäten entsprechen verschiedenen Ebenen in der Lernmittelhierarchie.

[xxi] Mehr zu diesem Standard findet sich unter http://dublincore.org/.

Auch das Inhaltsmodell kann mit Metadaten assoziiert werden und ebenso Aktivitäten.

Ablaufsteuerung bezieht sich auf Aktivitäten. Der beabsichtigte Ablauf von Aktivitäten ist als Teil der Inhaltsorganisation definiert. Hier werden Aktivitäten in Beziehung zueinander gesetzt, indem Ablaufinformationen mit jeder Aktivität assoziiert wird. Ablaufinformationen und -regeln werden durch das `sequencing`-Element im Inhaltsmodell gekapselt.

Das LMS ist verantwortlich für die Interpretation der Ablaufinformationen in der Inhaltsorganisation und für die Anwendung des Ablaufverhaltens, um die tatsächliche Reihenfolge der Lernressourcen zur Laufzeit festzustellen.

SCORM-konforme Lerninhalte trennen die Ablaufsteuerung vom eigentlichen Inhalt. Dies erleichtert die Wiederverwendbarkeit und den Austauch zwischen verschiedenen Autorensystemen. SCORM-konforme SCOs sollten nicht kontextsensitiv sein, sie sollten nicht andere SCOs referenzieren und verlinken.

4.1.2. Metadaten

Zum Beschreiben des Inhalts existiert ein vordefinierter Wortschatz[xxii], der in neun Kategorien unterteilt ist:

1. **General** dient zum beschreiben der allgemeinen Informationen über die Ressource.
2. **Lifecycle** gruppiert die Informationen zu Geschichte und dem gegenwärtigen Stand der Ressource und gibt an, wer die Ressource verändert hat.
3. **Meta-Metadaten** speichert Informationen zu den Metadaten selbst. Beispielsweise kann festgehalten werden, *wann* die Metadaten von *wem* eingegeben wurden und in *welcher Sprache* die Metadaten angegeben sind.
4. **Technical** beschreibt die technischen Anforderungen und die Eigenschaften der Ressource. Zum Beispiel kann hier eine *URL* angegeben werden, unter der die neueste Version des Inhalts heruntergeladen werden kann.
5. **Educational** gibt die erzieherischen und pädagogischen Eigenschaften der Ressource wieder.
6. **Rights** speichert die Rechte für geistiges Eigentum und Bedingungen für die Verwendung der Ressource.
7. **Relation** beschreibt die Beziehungen dieser Ressource zu anderen Ressourcen. Hier sind Angaben

[xxii] Der Wortschatz basiert auf dem *LOM Information Model*. LOM steht für *Learning Object Metadata*.

wie *isbasedon* [xxiii]

8. **Annotation** beinhaltet Kommentare zum erzieherischen Nutzen der Ressource und Informationen wann und von wem die Kommentare angelegt wurden.

9. **Classification** reiht die Ressource in ein gegebenes Klassifikaitonssystem ein.

Im Manifest des Content Package werden diese neun Kategorien in einem `lom`-Element zusammengefasst. Dabei muss nicht jede Kategorie vorhanden sein.

Beispiel 1. Metadaten

```
<lom xmlns="http://ltsc.ieee.org/xsd/LOMv1p0">
  <general>
    <title>
      <string xml:lang="de">Seminarvortrag zu e-learning Standards</string>
    </title>
    <description>
      <string>
        Die Seite bietet die Ausarbeitung zum Seminarvortrag über e-learning-Standards
         als PDF und docbook/XML sowie interessante und hilfreiche Links zum Thema.
      </string>
    </description>
    <keyword>
      <string>e-learning</string>
    </keyword>
    <keyword>
      <string>xml Anwendung</string>
    </keyword>
    <keyword>
      <string>Standards</string>
    </keyword>
    <keyword>
      <string>SCORM</string>
    </keyword>
    <keyword>
      <string>Metadaten</string>
    </keyword>
    <language>de</language>
  </general>
  <technical>
    <location type="URI">
      http://www.gungfu.de/studium/e-learning/index.html
    </location>
    <format>application/xml</format>
  </technical>
</lom>
```

[xxiii] *'Basiert auf'*, d.h. das aktuelle Dokument ist eine Weiterentwicklung des angegebenen Dokuments.

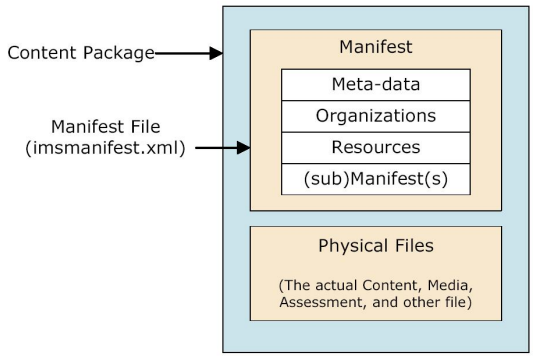
4.1.3. Content Packaging[xxiv]

Dieser Teil des Standards beschreibt, wie die Inhalte auf einem Speichermedium bzw. in einem Archiv abgelegt werden sollten, damit das gemeinsame Verwenden der Lernressource unterstützt wird.

Das Inhaltverpackungs-Modell gibt prinzipiell an, dass die Daten des Inhaltsmodells[xxv] in einer XML-Datei mit dem Namen `imsmanifest.xml` gespeichert sein müssen, die sich im Wurzelverzeichnis des Pakets befindet.

Diese Datei - ähnlich einem Packzettel - beschreibt den Inhalt des Pakets anhand der Metadaten und kann optional eine Beschreibung der Inhaltsstruktur beinhalten.

Ressourcen werden durch entsprechende Einträge in der `imsmanifest.xml`-Datei referenziert. Das Content Packaging spezifiziert auch, wie die physischen Dateien auf dem Speichermedium verpackt werden müssen und wie die Daten in eine PIF[xxvi]-Datei komprimiert werden.



Schema eines *Content Package*

Ein Inhaltspaket fasst Inhaltsobjekte[xxvii] in einer Inhaltsorganisation zusammen, die in einem Manifest beschrieben wird. Ein solches Paket kann einen Kurs, eine Lektion, ein Modul oder einfach eine Sammlung von zusammenhängenden Inhaltsobjekten repräsentieren. Es können auch *alternative* Organisationen definiert werden.

Beispiel 2. Alternative Organisationen

Die Definition von mehreren Organisationen in einem Manifest macht beispielsweise dann Sinn, wenn auf unterschiedliche Zielgruppen eingegangen werden soll, im Grunde jedoch derselbe Lernstoff vermittelt werden soll. Solche Alternativen können sich z.B. auf *Anfänger*, *Fortgeschrittene* und

[xxiv] cf. [2], Seite 1-29
[xxv] Also Metadaten, Organisation mit evtl. Alternativen und Sequencing-Informationen.
[xxvi] *Package Interchange File* - entspricht im Wesentlichen ZIP
[xxvii] Ein Inhaltsobjekt ist entweder ein SCO oder ein Asset (cf. [4])

Experten beziehen.

4.2. Die Run-Time Environment

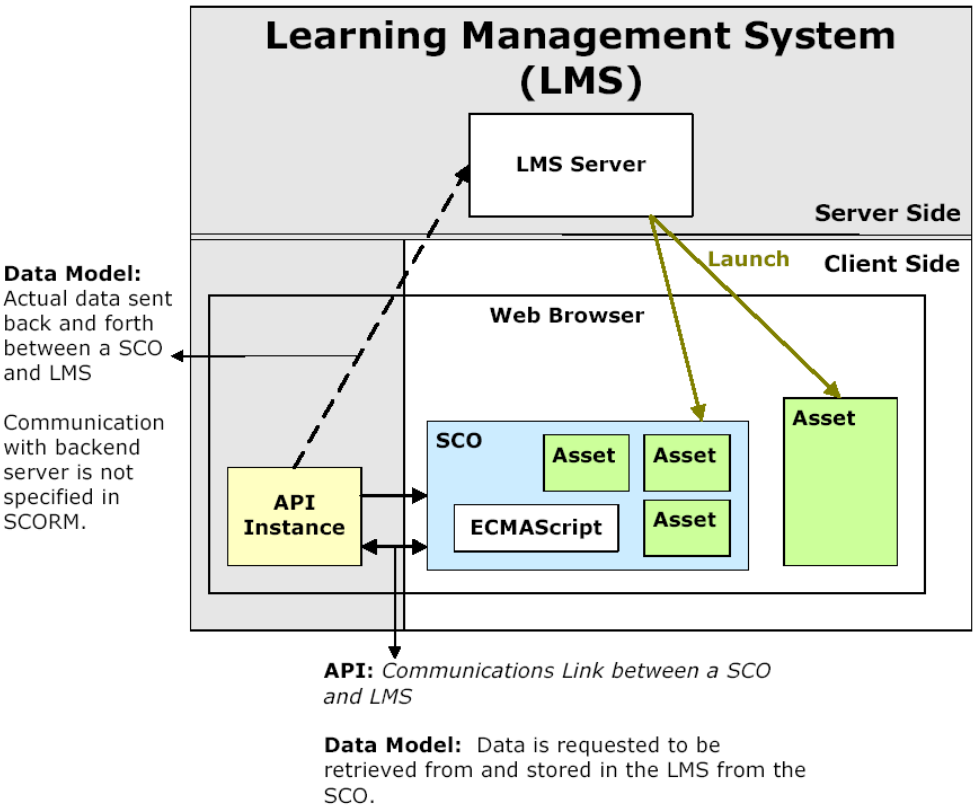
Der Hauptzweck des RTE besteht im Zurverfügungstellen von Mitteln für das Zusammenarbeiten von SCO und LMS. Da SCORM die Möglichkeit des Zusammenarbeitens von Lerninhalten in verschiedenen LMS, ohne dass die Werkzeuge zum Erstellen der Inhalte berücksichtigt werden müssen, ermöglichen möchte, ist es notwendig, dass Inhalt in einer vereinheitlichten Weise gestartet werden kann (cf. [2], Seite 1-30). Außerdem muss festgelegt werden, wie sich der Inhalt verhält, wenn er einmal vom LMS gestartet wurde (cf. [20]). D.h. der Inhalt ist *aktiv*. Sein Verhalten wird durch JavaScript[xxviii] -Funktionen definiert.

Die drei Bestandteile des SCORM RTE (cf. [2], Seite 1-30 f.) werden als

- **Launch**
- **Application Programming Interface (API)** und
- **Data Model**

bezeichnet.

[xxviii] Eigentlich **ECMA**Script - European Computer Manufacturers Association.



Funktionsweise der Run-Time Environment

4.2.1. Launch

Launch definiert den Startvorgang eines SCORM-Inhalts durch das LMS. So daß jeder SCORM-konforme Inhalt mit SCORM-konformen LMS ausgeliefert und dem Lerner angezeigt werden kann. Außerdem wird hier bestimmt, welcher Inhalt als nächstes angezeigt werden soll. Weiterführendes findet sich im *sequencing and navigation*-Buch (siehe hierzu Abschnitt 4.3, „Ablaufsteuerung und Navigation“.

Zum Starten eines Lerninhalts ist es notwendig, dass das LMS in der Lage ist, die nächste zu startende Lernaktivität festzustellen. Ausgelöst durch ein *Navigationsereignis* findet das LMS den Inhalt, der mit dieser neuen Aktivität verknüpft ist, anhand der Informationen im Content Package. Dort sind die Lerninhalte mit Ressourcen verknüpft, die ihrerseits anhand von URLs[xxix] auf physikalische Ressourcen (Dateien) verweisen.

Nach dieser Ressourcenauflösung wird der gegenwärtige Inhalt mit dem neuen Inhalt ersetzt, indem der alte Inhalt beendet und der neue gestartet wird.

Beispiel 3. Beispiel für den Verweis von Lernaktivität auf Ressource

[xxix] Universal Resource Locator

```
<manifest>
  <organizations>
    <organization>
      <item identifier="INTRO" identifierref="RESOURCE_INTRO">
        <title>Einleitung</title>
      </item>
      <item identifier="MODULE1">
        <title>Module 1 -- Basics</title>
        <item identifier="LESSON1" identifierref="RESOURCE_LESSON1">
          <title>Lesson 1 -- Interface</title>
        </item>
       ...
      </item>
    </organization>
  </organizations>
  <resources>
    <resource identifier="RESOURCE_INTRO" scormType="asset" type="webcontent" href="intro.htm">
      <file href="intro.htm"/>
      <file href="images/headerside.gif"/>
      <file href="images/headertop.gif"/>
      <file href="images/pssidebar.gif"/>
    </resource>
    <resource identifier="RESOURCE_LESSON1" scormType="asset" type="webcontent" href="Lesson1.htm">
      <file href="Lesson1.htm"/>
      <file href="images/EndOfLesson.gif"/>
      <file href="images/LessonTitle1.gif"/>
      <file href="images/headerside.gif"/>
    </resource>
    <resource identifier="RESOURCE_LESSON2" scormType="asset" type="webcontent" href="Lesson2.htm">
      <file href="Lesson2.htm"/>
      <file href="images/EndOfLesson.gif"/>
      <file href="images/HandTool.gif"/>
      <file href="images/LessonTitle2.gif"/>
      <file href="images/ZoomToolIcon.gif"/>
    </resource>
    ...
   </resources>
</manifest>
```

In diesem Beispiel verweist die Lernaktivität *INTRO* auf die Ressource *RESOURCE_INTRO*, die ein Bündel an Ressourcen kapselt.

Das Beispiel zeigt außerdem, dass eine Lernaktivität durchaus aus mehreren weitern Lernaktivitäten bestehen kann.

Für die verschiedenen Inhaltsobjekte[xxx] wird beim **Laden** unterschiedlich vorgegangen:

4.2.1.1. Laden von Assets

Da Assets *passiven Inhalt* (z.B. Bilder) darstellen, wird vom Standard nur verlangt, dass das LMS das

[xxx] Im Standard existieren *Asset* und *SCO*.

Asset über das HTTP-Protokoll startet. Assets kommunizieren nicht über die API und dem Datenmodell mit dem LMS.

4.2.1.2. Laden von SCOs

Für SCOs, dem *aktiven Inhalt* im SCORM-Standard, wird verlangt, dass das LMS ihn startet, Daten vom SCO speichert und auf Anfrage zum SCO sendet. Dabei ist jeweils nur *ein* SCO *pro Lerner* aktiv. Das LMS startet nicht mehrer SCOs für einen Lerner gleichzeitig.

Es ist jedoch erlaubt, dass ein SCO selbst die API implementiert - quasi als *Proxy* agiert - und es so untergeordneten, vom SCO geladenen SCOs ermöglicht, auf die API zuzugreifen. Dieser Zugriff ist jedoch *indirekt* über das aufrufende SCO und niemals direkt auf die API des LMS. Andernfalls könnten leicht Inkonsistenzen im Datenbestand des LMS entstehen.

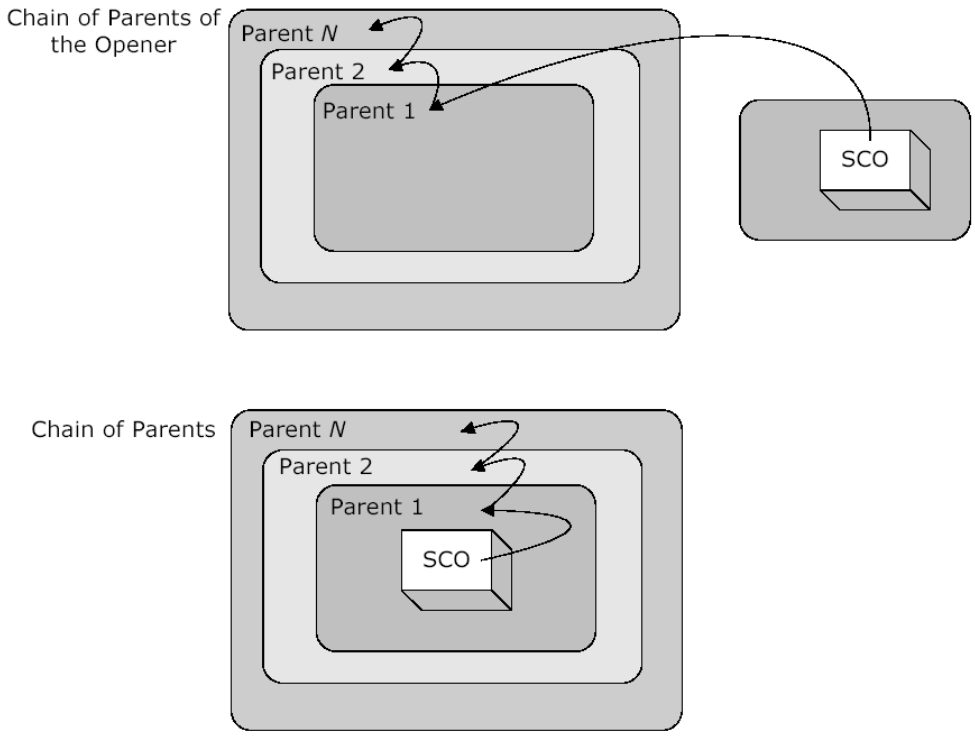
4.2.1.2.1. Suchen der API

Nachdem ein SCO durch das LMS gestartet wurde, ist das SCO dafür verantwortlich die API des LMS zu finden. Hierfür wird der Lokalisation der API auf der LMS-Seite folgende Restriktionen auferlegt:

- Das LMS muss das SCO in einem abhängigen Browserfenster[xxxi] starten. Oder
- Das SCO wird in einem *Kind*frame des LMS-Fensters gestartet, in dem die API als DOM[xxxii]-Objekt entweder direkt oder durch das Durchsuchen der *Eltern*frames zugänglich ist.

Diese Einschränkungen ermöglichen es einem SCO durch rekursive Suche in der Eltern- und/oder der Fensteröffnerhierarchie die API des LMS zu finden. Im DOM hat die API den Namen `API_1484_11` und kann so *eindeutig* identifiziert werden.

[xxxi] einem *Popup*
[xxxii] *Document Object Model* - ein Standard des W3C-Consortiums



Die zwei Möglichkeiten die API zu finden.

4.2.2. Application Programming Interface

Die *API* bietet eine Menge von vordefinierten Funktionen, auf die sich LMS-Anbieter und Autorensystem-Hersteller geeinigt haben, um die Kommunikation zwischen einem LMS und den SCOs, die das LMS startet, zu ermöglichen.

Ein SCORM-konformes LMS muss die nachfolgend aufgeführten acht API-Funktionen implementieren und den Großteil des SCORM Datenmodells unterstützen. Der schwierige Teil besteht in der Handhabung der Browser-zu-Server Kommunikation. Meistens wird diese Funktionalität in Form eines Java-**Applets** in einem versteckten Frame im Browserfenster realisiert. Es existieren jedoch auch Implementierungen mit Flash, ActiveX-Kontrollen und reinem JavaScript (cf. [20]).

SCORM-konformer Inhalt greift über die oben bereits angesprochene API auf das LMS zu. Hierfür muss das LMS folgende 8 API-Funktionen implementieren (cf. [20]:

1. `return_value = Initialize(parameter)` dient zur Initialisierung der Kommunikation zwischen SCO und LMS. Das LMS führt LMS-spezifische Aufgaben durch. Der Parameter ist der leere String "".

2. `return_value = Terminate(parameter)` beendet die Kommunikation zwischen aktivem SCO und LMS. Das SCO ruft diese Methode auf, wenn es zu dem Schluß kommt, dass

keine weitere Kommunikation mit dem LMS nötig ist[xxxiii] . Daraufhin speichert das LMS die vom SCO empfangenen Daten[xxxiv] nach einem impliziten Aufruf der `Commit`-Methode (siehe dort).

Nach dem Aufruf der `Terminate`-Methode darf das SCO nur noch *Support Methoden* aufrufen (siehe weiter unten). Der Parameter ist der leere String "".

3. `return_value = GetValue(parameter)` fordert vom LMS einen Wert an. Der Wert wird durch das Datenmodell (bzw. durch das als Parameter übergebenes Datenmodell*element*) spezifiziert.

4. `return_value = SetValue(parameter1, parameter2)` bittet um den Transfer von parameter2 als Wert für das als parameter1 angegebenen Datenmodellelements an das LMS.

5. `return_value = Commit(parameter)` verlangt das Weiterreichen der evtl. in der API-Instanz zwischengespeicherten Daten an das LMS. Einer der Hauptgründe für die Existenz dieser Methode ist, dass duch das Zwischenspeichern zeitraubende direkte und häufigere Kommunikation zwischen Client (Browser) und LMS verhindert werden kann und die tatsächliche Übertragung wahrscheinlich erst angeregt werden muss. Nur nach Aufruf dieser Methode werden die in der API gesammelten Daten über das Netzwerk an das LMS geschickt. Der Parameter ist der leere String "".

6. `return_value = GetLastError()` verlangt den Fehlercode des gegenwärtigen Fehlerzustands der API-Instanz.

7. `return_value = GetErrorString(parameter)` liefert eine textuelle Repräsentation des übergebenen Fehlercodes.

8. `return_value = GetDiagnostic(parameter)` existiert für LMS-spezifische Anwendungen. Es bietet dem LMS die Möglichkeit, zusätzliche diagnostische Informationen durch die API-Instanz zu definieren.

Die Methoden 1 und 2 sind sogenannte **Session Methoden**. Mit ihnen wird der Anfang und das Ende der Kommunikation zwischen SCO und LMS markiert.

Die Methoden 2, 3 und 4 sind die sogenannten **Datentransfer Methoden** und dienen zum Datenaustausch mittels des Datenmodells zwischen SCO und LMS.

Die letzten drei Methoden, die sogenannten **Support Methoden**, dienen zur unterstützenden Kommunikation (z.B. Fehlerbehandlung) zwischen SCO und LMS.

Alle Methoden außer den Support-Methoden setzen einen **Fehlercode** bei ihrem Aufruf. Dieser Fehlercode kann und sollte vom SCO durch die `LMSGetLastError`-Methode abgefragt werden.

[xxxiii] Beispielsweise, weil der Lerner den Submit-Button des SCOs geklickt hat.

[xxxiv] Meist in einer Datenbank.

Error Code Category	Error Code Range
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

Die im Standard definierten Fehlercodes.

Beispiel 4. Fehlercodes

- **103 - Already Initialized** gibt an, dass das SCO die Kommunikation initialisieren wollte, nachdem diese bereits initialisiert war.
- **123 - Retrieve Data After Termination** zeigt an, dass das SCO versucht hat, Daten vom LMS zu beziehen, nachdem die Kommunikation bereits erfolgreich beendet wurde. D.h. das SCO hat die `GetValue(parameter)`-Methode des API aufgerufen, nachdem zuvor `Terminate("")` erfolgreich war.

Für die SCORM-Konformität eines Inhalts ist es ausreichend, dass dieser nach dem Finden der API die Initialisierungs-Funktion zu Beginn und die Terminierungs-Funktion am Ende der SCO-Sitzung aufruft.

Jede Kommunikation zwischen dem LMS und einem SCO wird durch das SCO angeregt. Es gibt gegenwärtig keine standardisierte Möglichkeit mit der ein LMS auf ein SCO zugreifen kann (cf. [4], Seite 3-4).

Durch die API wird die konkrete Implementierung der Kommunikation zwischen LMS und Browser vor dem SCO versteckt. So muss sich der SCO-Entwickler muss sich nicht um diese Art der Server-Kommunikation kümmern, sondern kann sich auf die Entwicklung des Inhalts konzentrieren.

4.2.3. Data Model

Das **Data Model** stellt den Wortschatz zur Verfügung, der für den Informationsaustausch zwischen aktivem Inhalt und LMS benutzt werden kann. Über das Datenmodell werden Daten vom LMS bezogen bzw. an es gesendet, indem die im Standard definierten `set`- und `get`-Methoden API-Funktionen aufgerufen werden. D.h. das Datenmodell gibt an, welche Daten das LMS für ein Inhaltsobjekt speichert und was das LMS mit den Daten macht (z.B. Reihenfolgesteuerung).

Beispiel 5. Datenmodell

Wenn das Testergebnis eines Lerners weitergegeben werden soll, wird ein SCO das SCORM Datenmodell-Element `cmi.score.scaled` verwenden um das LMS darüber zu informieren, wie der Lerner abgeschnitten hat. Hierfür ruft das SCO die API-Funktion `SetValue` mit dem Parameter *`cmi.score.scaled`* und dem Score auf:

`SetValue`(*`cmi.score.scaled`*, *`5`*)

Im Datenmodell können verschiedenste Informationen gespeichert werden:

- Informationen über den Lerner
- Interaktionen, die der Lerner mit dem SCO hatte
- Informationen zu Lernzielen[xxxv]
- Erfolgsstatus
- Komplettheitsstatus

Diese Daten haben auch Einfluß auf die Ablaufsteuerung indem die Ergebnisse des Lerners an das LMS gemeldet werden und darauf, wie im Manifest spezifiziert, reagiert wird.

4.3. Ablaufsteuerung und Navigation

Seit *SCORM 2004* gibt es als viertes Buch das **Sequencing and Navigation**-Buch. Es beschreibt grundlegende Verantwortungen für ein LMS und ist somit hauptsächlich für Hersteller von LM-Systemen und Autorenwerkzeugen interessant.

Themen des SN-Buchs

- **Konzepte und Terminologie** - es werden Begriffe wie *Lernaktivität*, *Aktivitätsbaum* und *Cluster* eingeführt und definiert.
- **Seqeuncing Definition Model** beschreibt die Sequenzsteuerungsregeln, die auf Lernaktivitäten angewandt werden können.

[xxxv] *Objectives*

- **Sequencing Behavior Model** beschreibt das Verhalten des LMS bei Abarbeitung von Sequenzregeln
- **Navigation Data Model**

Generell beschreibt das SN-Buch ein *Laufzeitdatenmodell*, das LMSs nutzen können, um dem Lerner die durch Navigationsanfragen angeforderten Inhalte anzuzeigen, oder das Anzeigen abzulehnen.

4.3.1. Aktivitäten

Schon mehrmals erwähnt, soll hier eine kurze Definition von *Lernaktivitäten* gegeben werden.

Umgangssprachlich können Lernaktivitäten als "bedeutungstragende *Einheiten* von Lerninformation" bezeichnet werden. Ein Lernaktivität ist etwas, das der Lerner 'tut', während er durch das Lernangebot fortschreitet.

Eine Aktivität kann aus weiteren Aktivitäten bestehen und wird dann 'Cluster' genannt (siehe dort). Eine Aktivität kann jedoch auch ein *Blatt* im Aktivitätsbaum sein und dann mit einer Ressource (oder einem weiteren Manifest, siehe Abschnitt 4.3.2.1, „Erstellen eines Aktivitätsbaums und dessen Struktur“) assoziiert sein.

Welche Aktivität dem Lerner vom LMS als nächste zu bearbeitende Aktivität angezeigt wird, wird zur Laufzeit des Systems bestimmt und hängt vom Fortschritt des Lerners in vorhergehenden Aktivitäten, der Lernabsicht und den durch den Autor des Lerninhalts vorgegebenen Ablaufsteuerungsinformationen (siehe Abschnitt 4.3.3.3, „Sequencing Rule“) ab.

Es sind immer die Blätter im Aktivitätsbaum, die dem Lerner prästentiert werden. Die Folge von den mit den Blattaktivitäten assoziierten Lerninhalten bezeichnet man als **Lernerfahrung**.

4.3.2. Aktivitätsbäume

Aktivitätsbäume werden aus der *Inhaltsstruktur* (siehe oben) abgeleitet. Sie beschreiben die Struktur der Lernaktivitäten. Mit ihr können Ablaufsteuerungs- und Navigationsverhalten[xxxvi] in einer implementierungsunabhängigen Weise beschrieben werden.

Es wird erwartet, aber es ist nicht vorgeschrieben, dass Systeme, die Ablaufsteuerung unterstützen, auch intern eine Repräsentation als Aktivitätsbaum besitzen.

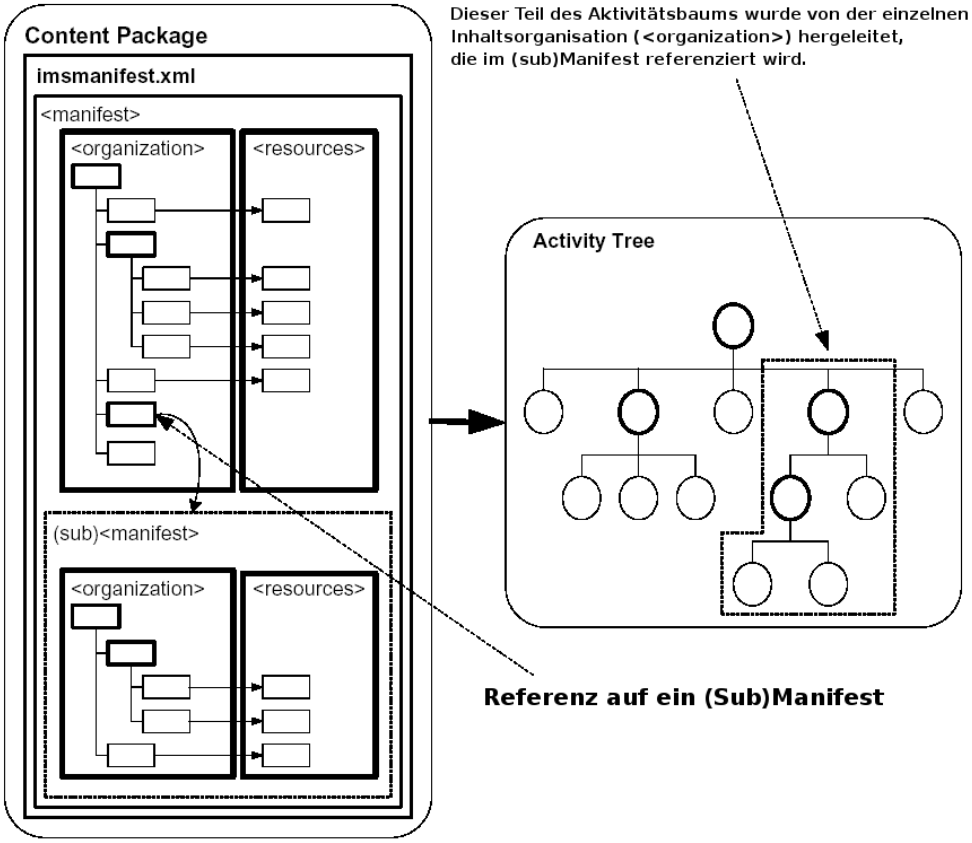
Aktivitätsbäume repräsentieren also eine *Instanz* [xxxvii] von hierarchischen Lernaktivitäten und die

[xxxvi] Z.B. Algorithmen zur Ablaufsteuerung

dazugehörenden Ablaufsteuerungsinformationen.

4.3.2.1. Erstellen eines Aktivitätsbaums und dessen Struktur

Ein Aktivitätsbaum kann direkt aus der *Inhaltsorganisation* eines *Content Package* hergeleitet werden. Hierfür werden die in einem `organization`-Element des `manifests` gespeicherten Einträge ausgewertet. Diese Einträge enthalten auch die Ablaufsteuerungsregeln, die entweder in `sequencing`-Elementen oder `sequencingCollection`-Elementen gekapselt sind. Mehr dazu findet sich in Abschnitt 4.4, „Mögliche Angaben zum Sequencing im Content Package“.



Herleiten eines *Aktivitätsbaums* aus einer Inhaltsstruktur

Das Bild stellt schematisch die Herleitung eines Aktivitätsbaums aus der Inhaltsstruktur, wie sie im Manifest definiert wird, dar. Zusätzlich wird die Möglichkeit des *Einbeziehens* von bereits vorhandenen Manifesten in ein neues Manifest angedeutet. Hierfür wird ein Blatt-`item`-Element mit einem `manifest`-Element anstelle eines `resource`-Elements assoziiert. Die von diesem `manifest` referenzierte Default-`organization` wird dann anstelle des Blatt-`item`-Elements in den Aktivitätsbaum inkludiert. Dies ermöglicht eine *Modularität* von Lerninhalten und den Rückgriff auf bereits bestehende Lerninhalte.

Das Bild soll auch deutlich machen, dass die einzelnen `item`-Elemente des Manifests eins-zu-eins als

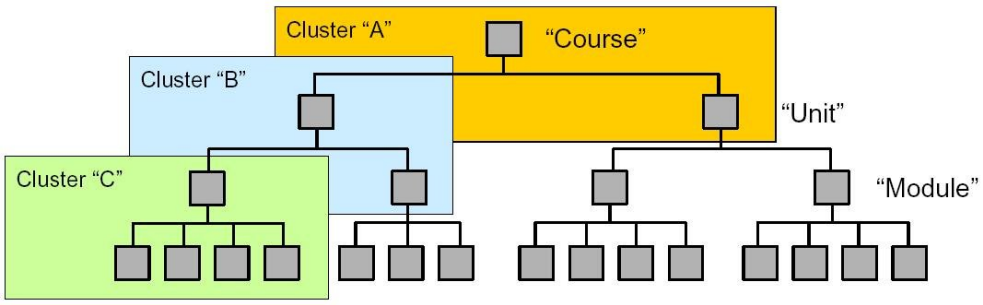
[xxxvii] Für jeden Kurs eines Lerners wird ein selbständiger *Aktivitätsbaum* assoziiert.

Aktivitäten übernommen werden können, d.h. Aktivitäten entsprechen `items` und wie diese können `items`-Elemente*geschachtelt* sein. Diese Abbildungsmöglichkeit erlaubt den Austausch von Ablaufsteuerungsinformationen zwischen verschiedenen (SCORM-konformen) LMSs.

Wie oben bereits erwähnt existiert für jeden Lerner und für jeden Kurs ein Aktivitätsbaum. In diesem werden auch *tracking*-Informationen[xxxviii] gespeichert.

4.3.2.2. Cluster

Die Schachtelung von Aktivitäten, also die Möglichkeit, dass eine Aktivität aus weiteren *Sub-Aktivitäten* besteht, nennt man *Clustering*. Eine solche abstraktere Aktivität wird **Cluster** genannt. Ein Cluster enthält jedoch nicht alle Sub-Aktivitäten beliebieger Tiefe, sondern besteht nur aus einer einzelnen *Eltern*aktivität und deren *Kinder*aktivitäten. Cluster sind wesentiche Bausteine eines Aktivitätenbaums und viele Elemente des *Sequencing Definition Models* beziehen sich auf die Einheit des Clusters. Dabei enthält die Elternaktivität eines Clusters die Ablaufsteuerungsinformationen für die Kinder. Für die Beziehungen zwischen Clustern und dem Sequencing Definition Model siehe beispielsweise Abschnitt 4.3.3.1, „Sequencing Control Modes“.



Schema eines *Aktivitätsbaums* mit Clustern

4.3.2.3. Versuch

Sobald mit dem Bearbeiten einer Aktivität begonnen wird, wird ein *Versuch* zum Lösen dieser Aktivität gestartet. Handelt es sich dabei um eine Blattaktivität, so werden für alle Aktivitäten auf dem Weg zur Wurzelaktivität des Aktivitätsbaums Versuche gestartet. Für jeden Aktivitätsbaum kann pro Lerner nur eine Blattaktivität versucht werden[xxxix]. Durch einen Versuch können ein oder mehrere **Lernziele** erreicht werden.

Es darf zwar nur jeweils eine Blattaktivität versucht werden. Es besteht jedoch die Möglichkeit des *Unterbrechens* [xl] einer Aktivität. So kann es vorkommen, dass mehrere Aktivitäten unvollständig sind.

xxxviii Z.B. Informationen darüber, wie viel einer Lernaktivität der Lerner bereits abgearbeitet hat, welche die aktuelle Aktivität ist, etc.
xxxix Dies hängt sicherlich damit zusammen, dass der Standard das gleichzeitige Starten mehrerer SCOs untersagt.
xl suspend

4.3.3. Sequencing Definition Model

Die Möglichkeiten zur Beeinflussung des Verhaltens bei system- und/oder lernerinitiierten Navigationsereignissen in SCORM sind mannigfaltig. Im folgenden werde ich auf die wichtigsten *Möglichkeiten zur Definition der Ablaufreihenfolge* eingehen.

4.3.3.1. Sequencing Control Modes

Um zu spezifizieren, welche Navigationsnachfragen auf ein Cluster angewendet werden können, gibt es die *Sequencing Control Modes*. Sie haben einen binären Wertebereich, können also entweder `wahr` oder `falsch` sein.

Die verschiedenen Modes

- **Sequencing Control Choice** - erlaubt die *Auswahl* einer Kindaktivität[xli] . *Default:* `wahr`
- **Sequencing Control Choice Exit** - erlaubt die *Terminierung* dieser Aktivität, wenn eine andere Aktivität[xlii] ausgewählt wird. *Default:* `wahr`
- **Sequencing Control Flow** - erlaubt die Navigation durch die Kinderaktivitäten anhand von *Vorwärts*- und *Rückwärts*-Navigationsereignissen. *Default: falsch*
- **Sequencing Control Forward Only** - bedeutet, dass der Lerner nur 'nach vorne' durch das Lernangebot navigieren kann. Es ist ihm nicht gestattet zu vorhergehenden Lernaktivitäten zurück zu wechseln. *Default:* `falsch`
- **Use Current Attempt Objective Information** - *Default:* `wahr`
- **Use Current Attempt Progress Information** - *Default:* `wahr`

Die beiden letzten Optionen geben an, ob die entsprechende Information vom gegenwärtigen Versuch, oder vom letzten Versuch in einem der Kindaktivitäten des Clusters verwendet werden soll.

Vom LMS wird verlangt, aber nicht erzwungen, dass es dem Lerner Navigationsmöglichkeiten entsprechend der für die aktuellen Aktivität spezifizierten Modes bietet. D.h. gute LMS bieten dem Lerner gar nicht erst die Möglichkeit falsche Navigationsereignisse auszulösen.

4.3.3.2. Constrain Choice Controls

[xli] Als *Kindaktivitäten* werden immer direkte Nachfolger verstanden.

[xlii] Also weder ein anderes Kind dieser Aktivität noch eine Aktivität auf dem Weg von dieser Aktivität zur Wurzelaktivität.

Für die Wahl der zu bearbeitenden Aktivität gibt es zwei weitere Einschränkungsmöglichkeiten.

- **Constrain Choice** - erlaubt eine Wahl nur dann, wenn die gewählte Aktivität direkt vor oder nach der aktuellen Aktivität liegt. *Default:* `falsch`
- **Prevent Activation** - verhindert den Start eines *Versuchs* dieser Aktivität durch die direkt Auswahl des Lerners. *Default:* `falsch`

Durch beide Regeln soll verhindert werden, dass der Lerner zu tief in ein Lerngebiet *springt*.

Beispiel 6. Prevent Activation

Durch Verwendung dieser Einschränkung kann der Lerner daran gehindert werden, den Abschlußtest einer Lerneinheit direkt anzuwählen.

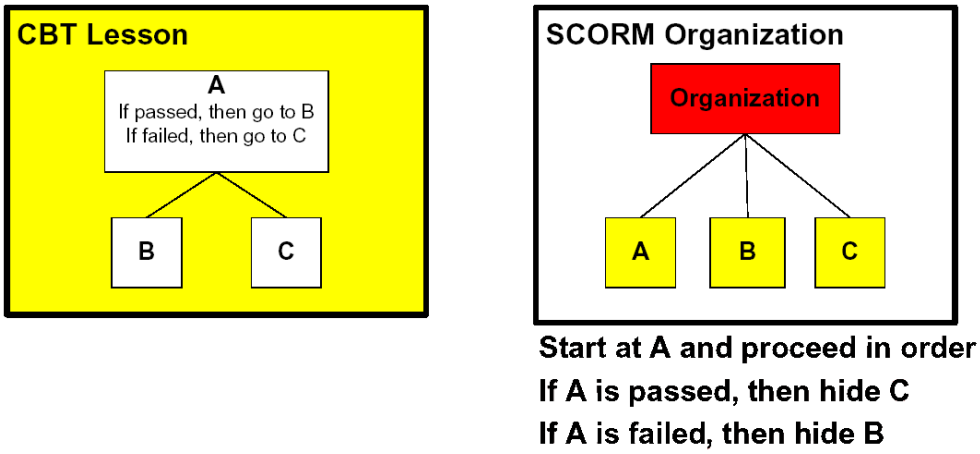
4.3.3.3. Sequencing Rule

Ablaufsteuerungsregeln verändern die standardmäßig lineare Reihenfolge des Lerninhalts bedingt durch Fortschritt und Beherrschen eines Gebiets durch den Lerner. Beispielsweise wird eine Aggregation (bzw. ein Modul) *verlassen*, wenn der Lerner das Lernziel erreicht hat.

Außerdem verändern Ablaufsteuerungsregeln den Ablaufzustand eines SCOs oder einer Aggregation. Zum Beispiel wird ein SCO deaktiviert, wenn es erfolgreich bearbeitet wurde (d.h. es ist nich mehr möglich dieses SCO auszuwählen) oder ein SCO wird übersprungen, wenn in einem Test ausreichendes Wissen demonstriert wurde.

Im Gegensatz zum Verzweigen zwischen Lerninhalten in CBT[xliii]-Systemen, wo die verschiedenen Möglichkeiten direkt in den einzelnen Lerninhalten angegeben werden, bietet SCORM eine neue Möglichkeit der Ablaufsteuerung. Hier werden die Interaktionens- und Navigationsmöglichkeiten zwischen den einzelnen Lerninhalten - SCOs - und dem Lerner vom LMS gesteuert und durch die Inhaltsorganisation definiert. Es besteht *keine direkte, explizite Verknüpfung* zwischen einzelnen SCOs. Im nachfolgenden Bild (cf. [19]) soll dieser Sachverhalt verdeutlicht werden.

[xliii] Computer-Based Training



Gegenüberstellung vom Verzweigen in CBTs und der Ablaufsteuerung in SCORM

Für die Planung von Ablaufsteuerungen werden **Inhaltsstrukturdiagramme** erstellt. Die rechte Seite des obigen Bilds ist ein einfaches Beispiel für ein solches Diagramm. Das Bild zeigt ebenfalls einfache *Ablaufsteuerungsregeln* in Umgangssprache.

4.3.3.4. Rollup rule

Als *Roll Up* wird das Weiterreichen von Lernergebnissen in unteren Teilen des Aktivitätsbaums an weiter oben liegende Cluster-Aktivitäten bezeichnet. Durch *rollup-Regeln* wird beschrieben, wie sich der Lernfortschritt in Cluster-Aktivitäten[xliv] aus den Lernergebnissen der Kinderaktivitäten herleitet.

Einfügen des Bilds aus der Sequencing-Präsentation!

4.3.4. Sequencing Behaviors

Der vorhergehende Abschnitt 4.3.3, „Sequencing Definition Model“ beschrieb die im Standard gegebenen Möglichkeiten zur Definition der Ablaufsteuerung. In diesem Abschnitt wird erläutert werden, wie ein LMS auf die verschiedenen Ereignisse im System reagiert, d.h. es wird das geforderte Verhalten des LMS beschrieben.

Das System leitet sein Verhalten aus dem Zustand von drei Modellen her:

- Das **Tracking Model** speichert für jede Aktivität deren Zustand. Dieser wird durch die Interaktionen des Lerners verändert. Dieses Modell wird zur Laufzeit *dynamisch* verändert.

  Während einer Lernerfahrung wird das Trackingmodell ständig aktuallisiert und spiegelt so den Fortschritt des Lerners wieder.

[xliv] Also Aktivitäten, die aus weiteren Aktivitäten bestehen.

- Das **Activity State Model** verwaltet den Ablaufzustand jeder Aktivität des Aktivitätsbaums[xlv] und den globalen Zustand des Baums. Dies ist ebenfalls ein *dynamisches* Modell.
- Das **Sequencing Definition Model** [xlvi] beschreibt, wie die verschiedenen Ablaufsteuerungsprozesse[xlvii] das Trackingmodell verwenden, um die Ablaufsteuerung, wie sie in der Inhaltsorganisation definiert ist, umzusetzen.

In vorhergehenden Versionen von SCORM[xlviii] war das Datenmodell der Laufzeitumgebung (siehe Abschnitt 4.2.3, „Data Model“) das einzig existente Modell. D.h. vor *SCORM 2004* wurde der Fortschritt des Lerners verfolgt und daraus Abläufe hergeleitet.

Wie bei den Definitionsmöglichkeiten für die Ablaufsteuerung gibt es auch bei den Verhaltensweisen zahlreiche Vorschriften seitens des Standards. Im folgenden gehe ich auf die wichtigsten kurz ein. Alle drei aufgeführten Verhaltensweisen leiten ihr Verhalten aus dem *Trackingmodell* ab.

4.3.4.1. Navigationsverhalten

Legt fest, wie Navigationsanfragen validiert und in Terminierungs- und Ablaufanfragen übersetzt werden.

Beispiel 7. Übersetzen einer Start-Anfrage in eine Ablaufanfrage

Nach dem Start eines Kurses durch den Lerner initiiert das LMS eine *Start*-Navigationsanfrage. Das Navigationsverhalten übersetzt diese Anfrage in die entsprechende Ablaufanfrage und verarbeitet[xlix] diese.

4.3.4.2. Rollupverhalten

Beschreibt, wie die Trackinginformation für Cluster-Aktivitäten aus den Trackinginformationen der Kinderaktivitäten des Cluster hergeleitet werden.

4.3.4.3. Ablaufsteuerungsverhalten

Beschreibt, wie eine Ablaufanfrage anhand eines Aktivitätsbaums verarbeitet wird, um die 'nächste' abzuliefernde Aktivität zu bestimmen.

xlv Dies kann z.B. das *Verstecken* einer Aktivität - wie in der Studie zur Verzweigung in CBTs und SCORM - sein.
xlvi siehe Abschnitt 4.3.3, „Sequencing Definition Model“
xlvii sequencing processes
xlviii Vor *SCORM 2004* existierte das SN-Buch noch nicht.
xlix D.h. im Aktivitätsbaum wird eine passende, der angestoßenen Ablaufsteuerung entsprechende Blatt-Aktivität gesucht.

4.3.4.4. Ablaufschleife

Der Ablauf einer *Sequencing Session* [l] mit den verschiedenen Verhalten sieht grob wie folgt aus:

1. Der Lerner startet das LMS und wählt einen Kurs o.ä.
2. Das LMS initiiert einen Ablaufprozess durch eine *Start*-, *Resume All*- oder *Choice*-Navigationsanfrage.
3. Das Navigationsverhalten übersetzt diese Anfrage in die entsprechende Ablaufanfrage und verarbeitet diese.
4. Anhand der Ablaufanfrage, dem Trackingmodell und dem Sequencing Definition Model, das in durch die Inhaltsorganisation definiert ist, wird eine Aktivität bestimmt, die dem Lerner präsentiert werden soll. Falls keine entsprechende Aktivität gefunden wird, springe zu Schritt 9.
5. Das Auslieferungsverhalten[li] bestimmt, ob die ermittelte Aktivität ausgeliefert werden kann und bereitet den Start des assoziierten Inhaltsobjekts vor. Falls die Aktivität nicht ausgeliefert werden kann, springe zu Schritt 9.
6. Der Lerner interagiert mit dem Inhalt. Die Ablaufprozesse ruhen.
7. Das Inhaltsobjekt kann Werte rückmelden, die verschiedene Trackingmodell-Elemente verändern.
8. Der Lerner, das Inhaltsobjekt oder das System erzeugt eine Navigationsereignis, z.B. *Continue*, *Previous*, *Wähle Aktivität X*, *Beenden*.
9. Das LMS informiert die Ablaufsteuerungs*implementierung* über das Navigationsereignis durch eine Navigationsanfrage.
10. Das Navigationsverhalten übersetzt die Anfrage in eine Terminierungsanfrage (für die verlassene Aktivität) und in eine Ablaufanfrage. Falls diese Anfrage bedeutet, dass der Lerner den Versuch auf das Wurzelelement des Aktivitätsbaums beenden möchte, so endet die Sequencing Session.
11. Falls das Inhaltsobjekt die Navigationsanfrage durch seine Terminierung ausgelöst hat, hat es jetzt noch die Möglichkeit Werte zu liefern, die das Trackingmodell aktuallisieren. Danach endet der Versuch an dieser Aktivität. Jetzt wird das *Rollupverhalten* aufgerufen, um die Auswirkungen der Zustandsveränderungen durch die Interaktionen des Lerners mit dem Inhaltsobjekt zu bestimmen. Das Rollupverhalten aktuallisiert das Trackingmodell der vorherigen Aktivität und für alle *Vorgänger*-Aktivitäten im Aktivitätsbaum.

[l] Als *Sequencing Session* bezeichnet man den Zeitraum zwischen Beginn eines Versuchs auf die Wurzelaktivität eines Baums (also immer, sobald *irgendeine* Aktivität des Baums versucht wird) bis zum Ende dieses Versuchs.

[li] *delivery behavior* - wurde im Text nicht erläutert.

12. Die Ablaufschleife wiederholt sich beginnend bei Schritt 4 bis die Seqeuncing Session beendet ist.

Dei eigentliche Schleife befindet sich zwischen den Schritten 4 und 12.

4.4. Mögliche Angaben zum Sequencing im Content Package

Für die Definition der Ablaufsteuerungsregeln gibt es das `sequencing`- und das `sequencingCollection`-Element, das in der Datei `imsmanifest.xml` innerhalb des `organization`-Elements verwendet werden kann. Hier möchte ich kurz auf mögliche Angaben eingehen.

Mit dem `sequencingCollection`-Element können mehrere `sequencing`-Element zusammengefasst werden. Das `sequencingCollection`-Element dient also als *Container* für einzelne Ablaufsteuerungsregeln.[lii]

Das `sequencing`-Element kann als direktes Kind von `item`-Elementen (die keine Blätter sind), oder als Kind eines `organization`-Elements im Manifest vorkommen.

Alle oben angesprochenen[liii] Ablaufsteuerungsmöglichkeiten können durch ein `sequencing`-Element kodiert werden.

Beispiel 8. Eine sequencing-Regel und ihre Kodierung als `sequencing`-Element

```
<sequencingCollection>
  <sequencing ID = "pretest">
    <controlMode choice = "false" choiceExit = "false" flow = "true" forwardOnly = "true"/>
    <sequencingRules>
      <preConditionRule>
        <ruleConditions>
          <ruleCondition condition = "completed"/>
        </ruleConditions>
        <ruleAction action = "disabled"/>
      </preConditionRule>
      <preConditionRule>
        <ruleConditions>
          <ruleCondition condition = "satisfied"/>
        </ruleConditions>
        <ruleAction action = "skip"/>
      </preConditionRule>
    </sequencingRules>
```

[lii] Im Folgenden meine ich mit `sequencing`-Element meist auch das `sequencingCollection`-Element.
[liii] generell: alle im Standard möglichen

```
  </sequencing>
</sequencingCollection>
```

Obiges `sequencingCollection`-Element enthält ein `sequencing`-Element. Dieses kodiert das Verhalten, dass keine Kindaktivität des Clusters ausgewählt werden darf (`choice="false"`), dass die Aktivität nicht durch die Auswahl einer anderen Aktivität beendet werden kann (`choiceExit="false"`) und dass die Navigation innerhalb der Kindaktivitäten durch Vorwärts- und Rückwärts-Buttons durch das LMS ermöglicht werden soll (`flow="true"`), jedoch mit der Einschränkung auf Vorwärts (`forwardOnly="true"`).

Der Inhalt des `sequencingRules`-Elements legt das Ablaufsteuerungs*verhalten* fest. Die Regeln werden entsprechend der Reihe, in der sie im Element vorkommen, abgearbeitet. Es gibt drei Regelklassen:

- **preConditionRule** - bestimmt Ablaufentscheidungen und das Ausliefern einer Aktivität, wenn ein Versuch *startet*.
- **exitConditionRule** - wird bearbeitet, wenn ein Versuch auf einer Nachfolgeraktivität *beendet* wird.
- **postConditionRule** - wird berücksichtigt, wenn ein Versuch auf der Aktivität beendet wird.

Jede Regel besteht aus:

- **ruleCondition** - Bedingungen, die zutreffen müssen.
- **ruleAction** - Aktionen, die ausgeführt werden.

Die im Beispiel zu sehenden Regeln bedeuten also:

1. Wenn die Aktivität vollständig bearbeitet (= *completed*) wurde, wird sie deaktiviert (= *disabled*) (d.h. beispielsweise, dass sie nicht mehr ausgewählt werden kann)
2. Wenn die Aktivität befriedigend bearbeitet wurde (= *satisfied*) oder ausreichendes Vorwissen vorhanden ist, dann wird sie übersprungen (= *skip*).

Es besteht z.B. auch die Möglichkeit, bei einer Aktivität, die *versucht* wird, das Vorwärtsgehen zu unterbinden. Das ist beispielsweise bei *Vortests* sinnvoll.

Wenn keine Sequencing-Regeln im Content Package, d.h. dem Manifest, angegeben wurden, so wird

ein *linearer Ablauf* - entsprechend der Reihenfolge der `item`-Elemente im Manifest - standardmäßig angenommen.

Die Optionen für die einzelnen Attributwerte sind vielfältig. So gibt es für das `condition`-Attribut der Regeln mindestens 12 Werte, die die Prämisse der Regel definieren. Diese Tatsache verdeutlicht die hohe Anpassungsfähigkeit des Lerninhalts und der Ablaufsteuerung im SCORM-Standard.

5. Wie funktioniert SCORM?

In diesem Abschnitt werde ich einen relativ abstrakten Überblick über die Funktionsweise von SCORM und das Zusammenspiel der einzelnen Komponenten geben.

5.1. Aus Sicht des Anwenders

1. Das LMS lädt einen aktiven Inhalt (SCO) oder ein Asset. Im Falle eines Assets werden die Schritte bis Nummer 4 und danach bis Nummer 7 übersprungen.

   Dies geschieht entweder durch lerner- oder system-initiierte Ereignisse wie z.B. dem Starten eines neuen Kurses oder durch das Befolgen einer Reihenfolgevorgabe (Sequencing) im Inhaltsmodell durch das LMS.

2. Das SCO sucht die SCORM-API. Im RTE-Buch wird festgelegt, dass sich die API immer in einem Elternframe des Frames, in dem der Inhalt geladen wird, befinden muss.

3. Nachdem die API gefunden wurde, wird die Funktion `Initialize` aufgerufen. Das SCO meldet sich so beim LMS an.

4. Der Lerner bearbeitet den Inhalt.

5. Das SCO schreibt und liest Daten in/vom LMS über die set- und get-Methoden der API und dem Datenmodell.

   Die API kommuniziert meist über ein Applet in einem versteckten Frame mit dem LMS-Server. Das Datenmodell wird meist durch eine Datenbank auf dem Server realisiert.

6. Wenn der Inhalt bearbeitet wurde, terminiert das SCO und ruft die API-Funktion `Terminate` auf, um sich beim LMS abzumelden.

7. Entsprechend der mit dem gerade bearbeiteten SCO assoziierten Ablaufssteuerungsregeln und dem Datenmodell wird der nächste zu ladende Inhalt bestimmt (dann: zurück zu Punkt 1).

5.2. Aus Sicht des Lerninhalt-Entwicklers

Beim Entwurf von Lerninhalten sollte der Entwickler auf eine sinnvolle Größe der einzelnen Lernobjekte achten, um eine Wiederverwendung zu ermögliche. Je größer und umfassender ein Objekt wird, desto spezifischer und weniger geeignet zur Wiederverwendung wird es.

Für die Entwicklung von SCORM-konformen Inhalten stehen eine Reihe von Werkzeugen zur Verfügung. So besteht mit dem RELOAD-Editor (downloadbar unter [23]) die Möglichkeit komfortable Content Packages zusammenzustellen und Metadaten einzufügen. Für das Erstellen von Lernobjekten an sich, genügt schon ein normaler HTML-Editor. Kommerzielle Produkte wie *Macromedia Authorware* bieten eine spezifische Unterstützung der Objektentwicklung.

Die Liste der SCORM-konformen LearningManagmentSystem ist ebenfalls umfassend. Eine ansprechende Präsentation des Lerninhalts ist somit gesichert.

6. Demonstration zur Funktionsweise anhand der Referenzimplementierung

Auf der ADL-Seite zu SCORM (unter [17] besteht die Möglichkeit des Downloads einer *Referenzimplementierung* eines SCORM-konformen Run-Time Environments. Die Implementierung ist kein vollwertiges LMS, ist jedoch in der Lage die Funktionsweise von SCORM zu verdeutlichen.

Die Implementierung benötigt das **Sun Java 2 SDK Standard Edition 1.4.2_02** um lauffähig zu sein.

Nach der Installation des Downloads kann ein mitinstallierter Apache-Server gestartet werden, der die Funktionalität des LMS bereitstellt.

Im Browser muss **JavaScript** aktiviert sein.

Das Installationsscript richtet im Programme-Ordner des *Windows*-Startmenüs einen Ordner `ADL` ein, unter dem ein Ordner `Sample RTE 1.3` eingebunden wird. Hier kann man den Server starten und stoppen. Der Server ist unter der Adresse *http://localhost:8080/adl/runtime/LMSMain.htm* erreichbar.

Nach dem Start des 'LMS' gliedert sich das Browserfenster in drei *Frames*:

- **Der oberste Frame** [liv] beinhaltet den *API-Adapter* als `Applet` mit der `id` 'APIAdapter'. Das Applet besitzt keine visuellen Komponenten und ist daher unsichtbar für den Benutzer.

[liv] In der Datei `LMSFrame.jsp`.

Außerdem zeigt dieser Frame die *Next*-, *Previous*-, *Suspend*- und *Quit*-Buttons für die Navigation im Inhalt. Die Anzeige dieser GUI-Elemente ist dynamisch und reagiert auf die in Abschnitt 4.3.3, „Sequencing Definition Model“ beschriebenen Modelle.

- **Der linke Frame** bietet später ein Menü für das gegenwärtig aktive SCO, wenn dies durch die Ablaufsteuerung erlaubt ist.
- **Der rechte Frame** zeigt den eigentlichen Inhalt. Beim Start des LMS wird hier eine kurze Einführung gezeigt.
- Im **Frameset** wird eine JavaScript-Variable mit dem Namen `API_1484_11` angelegt. Also ein Objekt im DOM, wie das vom Standard verlangt wird. Diese Variable dient als Relais für den eigentlichen API-Adapter, der nicht in einem Elterframe des Inhaltsframe platziert wurde (bzw. werden konnte). Über diese Variable ist dann der Zugriff des SCO auf die API wie im Standard verlangt möglich.

Durch Klick auf den Button *Log In* im oberen Frame gelangt man zum Login. Defaultmäßig existiert ein Benutzer **`admin`** mit dem Passwort **`admin`**.

Im darauffolgenden Fenster bestehen verschiedene Möglichkeiten für das weitere Vorgehen. Die zum Zwecke der Demonstration interessanten Optionen sind:

1. **Import a Course** zum Einfügen eines Kurses. Hier bietet sich einer der Kurse zu *Photoshop* der Referenzimplementierung an (downloadbar unter [17]).
2. **Register For a Course** ermöglicht das Sich Anmelden für einen Kurs. Das Anmelden für mehrere Kurse ist möglich.
3. **View Registered Courses** startet einen der Kurse, für die der Lerner sich angemeldet hat. Hierfür werden die entsprechenden Kurse in einer Liste als Links angezeigt.

6.1. Beispielkurse der Referenzimplementierung

Die Referenzimplementierung bietet eine Sammlung von Kursen zum Thema *Photoshop* an. Diese Kurse stellen denselben Inhalt dar. Jedoch ist das *Sequencing* und die *Lehrstrategien* verschieden[lv].

Ich stelle die verschiedenen Organisationen hier kurz dar, da sie einen praxisnahen Einblick in die

[lv] Die nachfolgenden Beschreibungen sind eine sinngemäße Übersetzung der PowerpointPräsentation, die den Beispielkursen (siehe [17] beiliegt.

Möglichkeiten der Ablaufsteuerung bieten.

- **Keine Sequenzregeln** - Im Manifest wurde keine Reihenfolge festgelegt. Der Standard erlaubt dann die freie Wahl der Reihenfolge durch den Lerner.
- **Linear** - Der Lerner muss den Inhalt in einer vorgegebenen Reihenfolge abarbeiten. Er kann zwar zurück-, nicht jedoch vorgehen.
- **Linear mit Kontrolle** fasst die Lektionen eines Themengebiets in Modulen zusammen. Nicht das LMS, sondern das SCO des Moduls bietet Navigationsmöglichkeiten. Navigation zwischen Lektionen eines Moduls erledigt das zuständige SCO. Navigation zwischen Modulen erledigt das LMS.
- **Linear mit Wahl** ermöglicht dem Lerner die Wahl eines Moduls nach Beenden der Einführung. Sobald ein Modul gewählt wurde, muss dessen Inhalt in der vorgegebenen Reihenfolge abgearbeitet werden. Nach dem Durchgehen eines Moduls muss er Fragen beantworten.
- **Beschränkte Wahl** - nach der Einleitung kann der Lerner ein Modul oder eine Lektion wählen. Nach dieser Wahl kann er nur noch ein Modul direkt vor oder direkt nach dem gewählten Modul auswählen. Nach der Wahl eines Moduls werden dessen Lektionen in einer definierten Reihenfolge abgearbeitet, der Lerner kann jedoch in beliebiger Reihenfolge in den verschiedenen Lektionen 'springen'.
- **Angepasst an das Wissen des Lerners** - nach der Einleitung kann der Lerner mit dem *Vortest* von Modul 1 weiterfahren, oder den Vortest eines anderen Moduls auswählen. Der Lerner kann zwischen Vortests 'springen' und Lektionen in beliebiger Reihenfolge auswählen. *Abschließende Tests* für Module können vom Lerner nicht ausgewählt werden.

  Wenn der Lerner einen Test nicht besteht, wird er zum Lerninhalt des Moduls geleitet. Nach dessen Durcharbeiten muss er wieder den Test machen.

  Nachdem alle Module abgearbeitet wurden, wird eine Zusammenfassung zur Leistung des Lerners gegeben.
- **Angepasst an das Wissen des Lerners mit Wiederverwendung** - die Informationen zur Ablaufsteuerung werden wiederverwendet.
- **Verbesserung** - nach der Einleitung wird der Lerner *linear* durch den Kurs geführt (siehe Linear). Wenn der Lerner den Verständnistest besteht, ist der Kurs beendet.

  Für jeden Teil des Tests, den den Lerner nicht bestanden hat, wird er zum entsprechenden Modul weitergeleitet, damit er den Lerninhalt wiederholen kann. Danach muss er wieder den Test machen.

- **Beurteilung des Verständnisses** ist eine Variante der vorhergehenden Strategie. Nach der Einleitung wird dem Lerner ein Beurteilungs-SCO präsentiert, das intern das Verständnis der einzelnen Ziele der Module auswertet.

  Dem Lerner wird dann Material zu den nicht bestandenen Modulen gezeigt.

  Nachdem der Lerner dieses Material durchgearbeitet hat, muss er wiederum einen Test machen, der diese Ziele prüft.

Beim Arbeiten mit den verschiedenen Kursen fällt auf, dass zusätzlich zum 'Submitten' einer Antwort im Frame des SCOs auch immer auf den 'Continue'-Button des LMS geklickt werden muss, um zur nächsten Seite zu gelangen. Dies macht offensichtlich, dass das Wissen für die Bewertung der Eingaben im SCO-Frame liegt. Dieser wertet die Eingaben des Benutzer aus. Eine automatische Weiterleitung zur nächsten Seite ist - auch wegen der Trennung der Ablaufsteuerung von den SCOs - nicht möglich.

7. Schlußbemerkung

SCORM ist ein sehr umfassender Standard, in dem der Inhalt von der Navigation getrennt wird. Somit werden die einzelnen Lernobjekte, Assets und SCOs als Bausteine des Standards, umfassend austauschbar, erweiterbar und weiderverwendbar. Module fassen mehrere Lernobjekte zusammen. Zur Navigation im Inhalt besteht die Möglichkeit der Angabe von Ablaufsteuerungsregeln. Auch hier sind die Möglichkeiten umfassend.

Eine erstaunlich große Anzahl von LearningManagmentSystemen, die sich derzeit auf dem Markt befinden, unterstützen SCORM (cf. [26]), oder geben dies zumindest an.

Es gibt eine Fülle an Lernobjekten, die, das sie SCORM-konform sind, wiederverwendet werden können. Diese Objekte sind meist jedoch nicht öffentlich zugänglich.

Das Erstellen von SCORM-konformen Inhalt wird durch die Menge an vorhandenen Autorensystemen unterstützt. Beispielsweise gibt es einen *SCORM-Wrapper* für das in Webentwickler-Kreisen weit verbreitete *Flash*-Autorenwerkzeug von Macromedia.

Somit kann die *Qualität* von Lerninhalten durch das Verwenden des SCORM-Standards erheblich verbessert werden.

Literaturverzeichnis

[1] *Introduction to the SCORM for Instructional Designers*. Betsy Spigarelli. http://www.adlnet.org/index.cfm?fuseaction=developer&pageview=viewarticle&ID=4&pcatid=15. Addresse gültig am: Juni 2004. 2004.

[2] *Sharable Content Object Reference Model (SCORM) 2004 - book 1 - Overview*. ADL Technical Team. http://www.adlnet.org/index.cfm?fuseaction=rcdetails&libid=648. Addresse gültig am: Juni 2004. 2004.

[3] *Sharable Content Object Reference Model (SCORM) 2004 - book 2 - Content Aggregation Model*. ADL Technical Team. http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=994. Addresse gültig am: Juni 2004. 2004.

[4] *Sharable Content Object Reference Model (SCORM) 2004 - book 3 - Run-Time Environment*. ADL Technical Team. http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=996. Addresse gültig am: Juni 2004. 2004.

[5] *Sharable Content Object Reference Model (SCORM) 2004 - book 4 - Sequencing and Navigation*. ADL Technical Team. http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=998. Addresse gültig am: Juni 2004. 2004.

[6] *SCORM Overview*. http://www.adlnet.org/index.cfm?fuseaction=scormabt. Addresse gültig am: Juni 2004.

[7] *SCORM History*. http://www.adlnet.org/index.cfm?fuseaction=SCORMHistory. Addresse gültig am: Juni 2004.

[8] *SCORM in a teacup?*. Dr. Tabetha Newman. http://www.trainingfoundation.com/articles/default.asp?PageID=945. Addresse gültig am: Juni 2004. 2002.

[9] *Sharable Content Object Reference Model (SCORM) Version 1.2*. http://www.adlnet.org/index.cfm?fuseaction=rcdetails&libid=40&bc=false. Addresse gültig am: Juni 2004. 2001.

[10] *Homepage der Advanced Distributed Learning-Initiative*. http://www.adlnet.org/. Addresse gültig am: Juni 2004.

[11] *Homepage von ARIADNE*. http://www.ariadne-eu.org/. Addresse gültig am: Juni 2004.

[12] *Homepage des AICC*. http://www.aicc.org/. Addresse gültig am: Juni 2004.

[13] *Homepage des IEEE/LTSC*. http://ltsc.ieee.org/. Addresse gültig am: Juni 2004.

[14] *Homepage des IMS*. http://www.imsglobal.org/. Addresse gültig am: Juni 2004.

[15] *Developer Tutorial*. Introduction to the SCORM for Instructional Designers. http://www.adlnet.org/index.cfm?fuseaction=print&prntTPK=4&prntTypeID=10. Addresse gültig am: Juni 2004.

[16] *Sharable Content Object Reference Model (SCORM) 2004*. ADL Technical Team. http://www.adlnet.org/index.cfm?fuseaction=rcdetails&libid=648. Addresse gültig am: Juni 2004. 2004.

[17] *Referenzimplementation zu SCORM*. ADL Technical Team. http://www.adlnet.org/index.cfm?fuseaction=SCORDown&listing=Examples. Addresse gültig am: Juni 2004. 2004.

[18] *Course Development*. Jeff Krinock. http://www.isn.ethz.ch/adl-wg/documents/Mr.%20Krinnock's%20presentation%20about%20sequencing%20in%20SCORM.ppt. Addresse gültig am: Juni 2004. 2002.

[19] *Putting ADL and SCORM into Practice*. http://www.tpic.org/Conference03/TPIC-SCORM-2.pdf. Addresse gültig am: Juni 2004. 2002.

[20] *SCORM Overview*. Rustici Software LLC. http://www.rusticisoftware.com/SCORMOverview.pdf. Addresse gültig am: Juni 2004.

[21] *Don't Bother Me With Objects! I've Got a Course to Teach*. A non-objective view. William Horton. http://www.designingwbt.com/content/madison/objectkeynote.pdf. Addresse gültig am: Juni 2004. 2002.

[22] *API based data exchange*. http://www.globalteach.com/GtWeb/Content/2/10_Product/30_Standards/aicc/V3/ApiBasedExchange.htm. Addresse gültig am: Juni 2004. 2002.

[23] *RELOAD Project*. Reusable eLearning Object Authoring & Delivery. http://www.reload.ac.uk/. Addresse gültig am: Juni 2004.

[24] *SCORM im Projekt SYMPOL*. Rüdiger Henrici. http://www.jkaref.de/Information/89.html. Addresse gültig am: Juli 2004. 2002.

[25] *Mr. Gysel's introductory presentation*. Gysel. http://www.isn.ethz.ch/adl-wg/documents/Mr.%20Gysel's%20introductory%20presentation.ppt. Addresse gültig am: Juli 2004. 2002.

[26] *Mr. Krinnock's presentation giving an introduction to SCORM*. Krinnock. http://www.isn.ethz.ch/adl-wg/documents/Mr.%20Krinnock's%20presentation%20giving%20an%20introduction%20to%20SCORM.ppt. Addresse gültig am: Juli 2004. 2002.