Hochschule für Technik und Wirtschaft Dresden (FH)
Fachbereich Informatik/Mathematik

Diplomarbeit

im Studiengang Medieninformatik

Thema:	**Untersuchung und Bewertung von SCORM 2004 sowie prototypische Realisierung von Online-Kursmaterial nach dieser Spezifikation**
eingereicht von:	Anne-Kathrin Gericke
eingereicht am:	23.12.2004
Betreuer:	Prof. Dr. Teresa Merino, Hochschule für Technik und Wirtschaft Dresden Dipl.-Wirtsch.-Inf. Robert Hossenfelder, Media Design Center, TU Dresden

Inhaltsverzeichnis

Einleitung

Es ist des Lernens kein Ende.
(Robert Schumann, 1810 – 1856, dt. Komponist)

Diese Aussage aus dem 19. Jahrhundert hat bis heute ihre Gültigkeit nicht verloren. Im Gegenteil, ständige Weiterbildung erlangt in der so genannten Informations- oder Wissensgesellschaft einen immer höheren Stellenwert. Zum einen für Privatpersonen, die vor allem im Beruf wettbewerbsfähig bleiben wollen und zum anderen für Unternehmen, die qualifizierte Mitarbeiter beschäftigen wollen. Dabei soll natürlich der Zeit- und Kostenaufwand möglichst gering gehalten werden. Dies kann durch den Einsatz von E-Learning erreicht werden. Es ist so für größere Unternehmen beispielsweise möglich, Mitarbeiter aus verschiedenen Unternehmensstandorten für den Einsatz neuer Technologien oder Produktionsabläufe zu schulen, ohne dass diese alle zu einem Schulungsort reisen müssen. Privatpersonen profitieren z. B. davon, dass keine festen Schulungstermine eingehalten werden müssen und sich die Weiterbildung flexibler in den Tagesablauf integrieren lässt.
Die Nutzung solcher E-Learning-Bildungsangebote setzt das Vorhandensein von entsprechenden Lernmanagementsystemen und Lerninhalten voraus. Diese werden von zahlreichen verschiedenen Herstellern produziert. In diesem Zusammenhang ist es wünschenswert, dass Produkte unterschiedlicher Hersteller miteinander kompatibel sind, so dass einmal erstellte Lerninhalte wiederverwendbar und auf verschiedenen Systemen nutzbar sind. Solche Produkte sollten also standardkonform erzeugt werden. Ein umfassender akkreditierter Standard existiert auf diesem Gebiet nicht, jedoch gibt es zahlreiche Standardisierungsprojekte, die sich mit dieser Thematik beschäftigen. Eine Spezifikation, welche Arbeitsergebnisse mehrerer Standardisierungsprojekte in sich vereint, ist das Sharable Content Object Reference Model (SCORM) der Advanced Distributed Learning Initiative (ADL).

Die Untersuchung der Anfang des Jahres veröffentlichten Version SCORM 2004 ist Gegenstand dieser Arbeit. Es werden die Konzepte und Komponenten des Referenzmodells sowie die Veränderungen hinsichtlich der Vorgängerversion SCORM 1.2 beschrieben. Der Schwerpunkt liegt dabei auf dem SCORM Sequencing, welches ein neuer Bestandteil des

Referenzmodells ist. Dieser Teil der Spezifikation definiert verschiedene Modelle und Prozesse zur Ablaufsteuerung. Ziel der Arbeit ist es, mit Beispielen und tabellarischen Übersichten dem Leser die Funktionsweise des Sequencing anschaulich und systematisch zu vermitteln. Im praktischen Teil der Arbeit sollen außerdem beispielhaft konkrete Anwendungsmöglichkeiten für das Sequencing gezeigt werden.

Im ersten Kapitel werden wichtige Begriffe geklärt, die in dieser Arbeit verwendet werden. Außerdem gibt das Kapitel einen Überblick über aktuelle Standardisierungsprojekte auf dem Gebiet des E-Learning und zeigt ihre Einwirkung auf die SCORM-Spezifikation.
Das Content Aggregation Model und das Run-Time Environment werden im zweiten Kapitel beschrieben. Dabei werden die Änderungen zur Vorgängerversion SCORM 1.2 aufgezeigt. Im Mittelpunkt des dritten Kapitels steht das Sequencing. Einleitend werden wichtige Konzepte vorgestellt, im Anschluss wird das Tracking Model beschrieben. Das Sequencing Definition Model bildet den Hauptteil des dritten Kapitels, dort wird ausführlich dargestellt, wie Sequencinginformationen in ein Content Package integriert werden. Anschließend werden kurz die vom Lernmanagementsystem durchzuführenden Sequencingprozesse vorgestellt. Das Kapitel endet mit der Beschreibung des Navigation Models. Den Abschluss der Arbeit bildet die prototypische Umsetzung von drei konkreten Anwendungsbeispielen. Es wird gezeigt, wie verschiedene Sequencingstrategien mit den Elementen des Sequencing Definition Models umgesetzt werden können.

1 E-Learning und Standards

Einführend werden in diesem Kapitel wichtige Begriffe dieser Arbeit geklärt. In den weiteren Abschnitten wird die Bedeutung von Standards für das E-Learning aufgezeigt und es werden die wichtigsten Standardisierungsprojekte vorgestellt und ihr Einfluss auf die SCORM-Spezifikation verdeutlicht.

1.1 Begriffsklärungen

Bevor näher auf die Bedeutung von Standards im E-Learning und auf konkrete Standardisierungsbestrebungen eingegangen wird, sollen zunächst verschiedene Begriffe erläutert werden, die in diesem Zusammenhang und in anderen Teilen dieser Arbeit verwendet werden.

1.1.1 E-Learning

Für E-Learning existieren sowohl unterschiedliche Schreibweisen (auch e-Learning, e-learning, eLearning) als auch unterschiedliche Definitionen. Geht man davon aus, dass das "E" die gleiche Bedeutung hat wie bei E-Mail oder E-Commerce, ist nicht nur allgemein die elektronische Bereitstellung von Lerninhalten gemeint, sondern explizit die Bereitstellung unter Nutzung von Webtechnologien [Neubauer, 2002]. E-Learning ist also nicht nur "Lernen am Computer", wozu z. B. auch das Lesen von Texten am Bildschirm zählen würde, sondern setzt unter anderem die Nutzung von Multimedia-Technologien und elektronischen Daten- und Kommunikationsnetzen voraus und beinhaltet auch den Kontakt und Austausch mit einem Tutor und anderen Lernenden in einem virtuellen Lernverbund [Bauer, 2001].

1.1.2 Standard und Spezifikation

Der Begriff Standard wird im Allgemeinen als Synonym für eine Norm verwendet. "Eine Norm ist eine allseits rechtlich anerkannte und durch ein Normungsverfahren beschlossene, allgemeingültige sowie veröffentlichte Regel zur Lösung eines Sachverhaltes." [Wikipedia] Eine Spezifikation (auch Empfehlung oder Richtlinie) beschreibt ebenfalls Einzelheiten zur Lösung eines Sachverhaltes, nur dass sie nicht durch eine anerkannte Normungseinrichtung zertifiziert ist. Spezifikationen bilden jedoch meist die Grundlage für die Normungs- bzw. Standardisierungsarbeit. Bei der Entwicklung eines Standards werden in der Regel vier Phasen durchlaufen [Masie, 2003]:

1. die Suche nach Problemlösungen
2. Dokumentierung der Lösung in einer Spezifikation
3. Test der Spezifikation in der Praxis (und Verbesserung)
4. Bearbeitung und Zertifizierung durch ein Standardisierungsgremium

Neben den akkreditierten Standards, die im englischen Sprachgebrauch auch als de-jure-Standard (lat. de jure "vom Recht her") bezeichnet werden, gibt es noch so genannte de-facto-Standards (lat. de facto "von der Tatsache aus"). Das sind Spezifikationen, Empfehlungen oder Richtlinien, die zwar nicht durch ein offizielles Normungsverfahren anerkannt sind, die sich aber durch die Praxis vieler Anwender und verschiedener Hersteller als technisch nützlich und richtig erwiesen haben. De-facto-Standards werden auch als Industriestandards bezeichnet [Wikipedia]. In dieser Arbeit wird der Begriff Standard im Sinne von Norm verwendet.

1.1.3 Lernmanagementsystem

Ein Lernmanagementsystem (LMS) ist ein serverbasiertes System, das über eine entsprechende Oberfläche (die Lernumgebung, vgl. 1.1.4) bestimmte Funktionalitäten, wie den Aufruf und die Administration von Lernern, Lerninhalten, Übungsaufgaben, Kommunikationstools usw. von einer zentralen Stelle aus ermöglicht. Das LMS verwaltet Lerninhalte und stellt sie den Lernenden zur Verfügung. Dabei wird der individuelle Lernprozess (z. B. Auswahl und Aufruf von Kursen, Bearbeitungszeiten, Testergebnissen) vom System verfolgt und protokolliert [Baumgartner, 2002].
Auch ADL beschreibt ein LMS mit ähnlichen Funktionalitäten: es unterstützt die Bereitstellung, Verfolgung, Aufzeichnung und Verwaltung von Lerninhalten, Lernfortschritten und Interaktionen des Lernenden [SCORM, 2004a].

1.1.4 Lernumgebung

Die Lernumgebung ist die Gesamtsituation, in die ein Lernprozess eingebettet ist. Im Zusammenhang mit E-Learning ist damit in der Regel die medial gestaltete Lernumgebung gemeint [Baumgartner, 2002]. In dieser Arbeit wird der Begriff im Sinne von der Benutzeroberfläche, auf der sich der Lernende bewegt, verwendet.

1.2 Bedeutung und Zielsetzung von Standards im E-Learning

In vielen Bereichen unseres Lebens sichern Standards und Normen die Qualität, Kompatibilität und Marktfähigkeit von Produkten und Technologien. Die Verbreitung und die

unkomplizierte Nutzung von Mobilfunk und Internet wären ohne das Vorhandensein von standardisierten Technologien wie GSM, TCP/IP und HTTP kaum möglich. Auch andere alltägliche Dinge wie Elektrizität oder das Eisenbahnwesen könnten ohne Normung von Steckerverbindungen oder Spurweite nicht existieren. Standardisierungen führen also zu einer Vereinheitlichung von Entwicklungs- und Anwendungsprozessen und gewährleisten die Kompatibilität von Produkten unterschiedlicher Hersteller. Außerdem erreichen Entwickler mit der Verwendung von Standards eine höhere Investitionssicherheit und bleiben wettbewerbsfähig [Gries, 2003].
Auch auf dem Gebiet des E-Learning zeigt sich die Notwendigkeit von Standards. Dabei lassen sich aus verschiedenen Perspektiven Ziele formulieren, die durch die Verwendung von einheitlichen Richtlinien erreicht werden können [Collier, 2002]. Aus der Sicht von Bildungsanbietern und Unternehmen können z. B. folgende Dinge durch Standards gesichert werden: plattform- und systemunabhängige Kombinationsmöglichkeit von Produkten unterschiedlicher Hersteller oder die Möglichkeit der Suche nach Lernangeboten in Kursdatenbanken über bestimmte Auswahlkriterien. Eine breite Auswahl an Lernangeboten, die Anpassung der Lernangebote an individuelle Bedürfnisse und ebenfalls die Suche nach Lernangeboten in Kursdatenbanken sind aus Sicht des Lernenden unter anderem erstrebenswerte Ziele einer Standardisierung. Für Entwickler von E-Learning-Komponenten (Lerninhalte, Autorenwerkzeuge, LMS) bedeutet die Verwendung von Standards, dass die verschiedenen Komponenten kombinierbar und wiederverwendbar sind, dass die Entwicklung von proprietären Schnittstellen für viele verschiedene Produkte entfällt und dass dadurch die Entwicklungskosten reduziert werden.

1.3 Wichtige Standardisierungsprojekte

Schon seit mehreren Jahren gibt es Standardisierungsbestrebungen zu unterschiedlichen Bereichen des E-Learning. Themenbereiche wie Metadaten, Content Packaging, Content Sequencing u.a. werden dabei von verschiedenen Organisationen, Initiativen und Projekten bearbeitet. Die Arbeit vier dieser Organisationen fließt in die SCORM Spezifikation ein. Diese Organisationen sowie die ADL Initiative werden nachfolgend vorgestellt.

1.3.1 AICC

Das Aviation Industry CBT (Computer Based Training) Committee ist eine internationale Vereinigung von Spezialisten aus dem Bereich des technologiebasierten Lernens. Seit der Gründung 1988 arbeitet das AICC an Richtlinien für die Entwicklung, Bereitstellung und Auswertung von CBT und verwandten Lerntechnologien zum Einsatz in der

Luftfahrtindustrie. Diese Richtlinien sind aber allgemein gehalten und werden daher auch außerhalb der Luftfahrtindustrie eingesetzt.
Unter der Bezeichnung AICC Guidelines and Recommendations (AGR) wurden bisher zehn technische Richtlinien veröffentlicht. AGR-001 ist eine Auflistung aller AICC-Publikationen und wird nicht als Empfehlung behandelt. Die AGR-002 bis AGR-010 geben jeweils Empfehlungen für einen speziellen Bereich und sind meist kurz gehalten. Für tiefergehende technische Details wird auf Whitepaper oder Technical Reports verwiesen.
AGR-006 Computer Managed Instruction (CMI) enthält Empfehlungen, welche die Interoperabilität von CMI-Systemen unterstützen. Mit der AGR-010 Web-based Computer Managed Instruction wurde diese Empfehlung auf web-basierte CMI ausgeweitet. Beide AGR verweisen auf ein weiteres Dokument, CMI-001 CMI Guidelines for Interoperability. Es enthält detaillierte Richtlinien für die Kommunikation zwischen CMI-System und Lerninhalt, für den Austausch von Kursmaterial zwischen verschiedenen CMI-Systemen und für die Speicherung von Nutzerdaten. [AICC]
Der 2003 vom IEEE LTSC verabschiedete Standard for Learning Technology - ECMAScript Application Programming Interface for Content to Runtime Services Communication (P1484.11.2-2003) basiert auf CMI-001 und wird in SCORM als Basis des Run-Time Environment Version 1.3 verwendet. Frühere Versionen von SCORM, einschließlich Version 1.2 basierten direkt auf CMI-001.

1.3.2 ARIADNE Foundation

Die Alliance of Remote Instructional Authoring and Distribution Networks for Europe Foundation wurde gegründet, um die Arbeit der Projekte ARIADNE und ARIADNE II umzusetzen und weiterzuentwickeln. Diese Projekte beschäftigten sich von 1996 bis 2000 mit der Entwicklung von Werkzeugen und Methoden zur Produktion, Verwaltung und Wiederverwendung von computerbasierten Lernelementen und wurden von der EU und der Schweizer Regierung finanziert.
Ein wichtiger Beitrag zu den bestehenden Standardisierungsbestrebungen im E-Learning ist die ARIADNE Educational Metadata Specification, welche zusammen mit der IMS Learning Resource Meta-data Specification die Grundlage für die Entwicklung des Standard for Learning Object Metadata durch das IEEE/LTSC bildete. [ARIADNE]

1.3.3 IEEE LTSC

Das Learning Technology Standards Committee (LTSC) ist eine Einrichtung des Institute of Electrical and Electronic Engineers (IEEE) und entwickelt akkreditierte Standards,

Praxisempfehlungen und Richtlinien für Lerntechnologien. Die Standards werden von mehreren Arbeitsgruppen entwickelt, die jeweils unterschiedliche Themengebiete bearbeiten [IEEE].
IMS und ARIADNE reichten 1998 eine gemeinsame Spezifikation beim LTSC ein, auf deren Basis die Working Group 12 ihre Arbeit an einem Entwurf für Learning Object Metadata (LOM) begann. Dieser Entwurf wurde am 12. Juni 2002 mit dem IEEE Standard for Learning Object Metadata (P1484.12.1-2002) ein anerkannter Standard und legt die Syntax und Semantik der Metadaten fest. Diese Metadaten beschreiben Lernobjekte und sollen die Suche, Auswertung und Benutzung der Lernobjekte ermöglichen [IMS, 2004]. Der LOM-Standard wird in SCORM 2004 als Grundlage für das Content Aggregation Model genutzt.
Weitere Standards, die in SCORM Verwendung finden sind der IEEE Standard for Learning Technology - ECMAScript Application Programming Interface for Content to Runtime Services Communication (P1484.11.2 –2003) und der Draft Standard for Learning Technology - Data Model for Content Object Communication (P1484.11.1).

1.3.4 IMS Global Learning Consortium

1997 startete die National Learning Infrastructure Initiative (NLII) das IMS Project. Die NLII ist ein spezielles Forschungsprogramm innerhalb von EDUCAUSE, einer US-amerikanischen Vereinigung zur Förderung des Einsatzes von Informationstechnologien in der Bildung. Mitglieder von EDUCAUSE sind US-Bildungseinrichtungen sowie Unternehmen und Organisationen aus dem Gebiet der Lerntechnologie. Auf Grund des weltweiten Interesses an der Arbeit des IMS Projects fand 1999 eine Umwandlung in das IMS Global Learning Consortium statt.
IMS entwickelt offene technische Spezifikationen für Lerntechnologien und unterstützt deren Einsatz. Diese Spezifikationen beziehen sich sowohl auf online- als auch auf offline-Lernanwendungen und zielen darauf ab, die Interoperabilität von Lerntechnologien zu sichern. Insgesamt gibt es Spezifikationen zu 13 verschiedenen Aspekten, wobei jede Spezifikation drei Dokumente umfasst: die genaue Spezifikation, die XML-Binding Specification sowie einen Best Practise and Implementation Guide.[IMS]
In SCORM werden berücksichtigt:

IMS Learning Resource Meta-data Specification

Diese Spezifikation enthält Richtlinien zur Beschreibung von Lerninhalten mit Metadaten und ermöglicht damit das Suchen und Auffinden und die Katalogisierung von Lerninhalten und somit deren Wiederverwendbarkeit. Die IMS Learning Resource Meta-data Specification

bildete die Grundlage für die Erarbeitung des IEEE Standard for Learning Object Metadata (vgl. Kap. 1.3.3).

IMS Content Packaging

Das IMS Content Packaging wurde entwickelt, um die Struktur und den Austausch von Lerninhalten zu standardisieren. Es beschreibt Datenstrukturen, deren Benutzung die Interoperabilität zwischen Lerninhalten, Tools und LMS sichert. IMS Content Packaging ist Bestandteil des SCORM Content Aggregation Models.

IMS Simple Sequencing Specification

Die IMS Simple Sequencing Specification wurde im März 2003 veröffentlicht. Sie stellt Methoden bereit, die gewährleisten, dass der vom Autor vorgesehene Ablauf eines Kurses in jedem LMS einheitlich interpretiert und dargestellt wird. Durch definierte Regeln wird entsprechend der Interaktion des Lerners mit dem Lerninhalt der Ablauf der Lerninhalte gesteuert.
SCORM verwendet ab der Version 1.3 die IMS Simple Sequencing Specification als Basis für Sequencing and Navigation.

Laut IMS sind mehrere IMS-Spezifikationen weltweit als de-facto-Standard anerkannt, einzigster akkreditierter Standard ist der IEEE 1484.12.1 - 2002 Standard for Learning Object Metadata.

1.3.5 ADL

Die Advanced Distributed Learning Initiative wurde 1997 vom US-amerikanischen Verteidigungsministerium und dem White House Office of Science and Technology Policy (OSTP) ins Leben gerufen. In kollaborativer Zusammenarbeit von Regierung, Industrie und Universitäten verfolgt ADL die Vision, individuell anpassbare und qualitativ hochwertige Bildungsmaterialien zeit- und ortsunabhängig einer Vielzahl von Benutzern zugänglich zu machen. Um diese Vision zu verwirklichen, hat die ADL-Initiative Anforderungen formuliert, die von Werkzeugen, Lerninhalten und Lernmanagementsystemen erreicht werden sollten [SCORM, 2004a]:

- Accessibility: Zugänglichkeit und Nutzbarkeit von unterschiedlichen Orten aus
- Adaptability: Anpassbarkeit an individuelle Bedürfnisse
- Affordability: Erschwinglichkeit durch Erhöhung der Lerneffizienz und Produktivität und durch Kostensenkung in der Erstellung

- Durability: Beständigkeit gegenüber Weiterentwicklungen von Betriebssystemen und Software
- Interoperability: gemeinsame Nutzung von Produkten unterschiedlicher Hersteller, unabhängig von Eigenschaften des Betriebssystems oder der Software
- Reusability: Wiederverwendbarkeit in verschiedenen Anwendungen und Kontexten

Diese Anforderungen sind gleichzeitig Ausgangspunkt für die Arbeit an SCORM, dem Sharable Content Object Reference Model. Dieses Referenzmodell definiert die technischen Grundlagen für web-basierte Lernmanagementsysteme und integriert dabei bestehende Spezifikationen, Standards und Richtlinien anderer Standardisierungsgremien in einem gemeinsamen Dokument [SCORM, 2004a]. Im Januar 2000 wurde die erste Version veröffentlicht. Die aktuelle Version SCORM 2004 wird im zweiten und dritten Kapitel ausführlich beschrieben.

Die folgende Abbildung soll abschließend die Zusammenhänge zwischen den verschiedenen Standardisierungsprojekten darstellen:

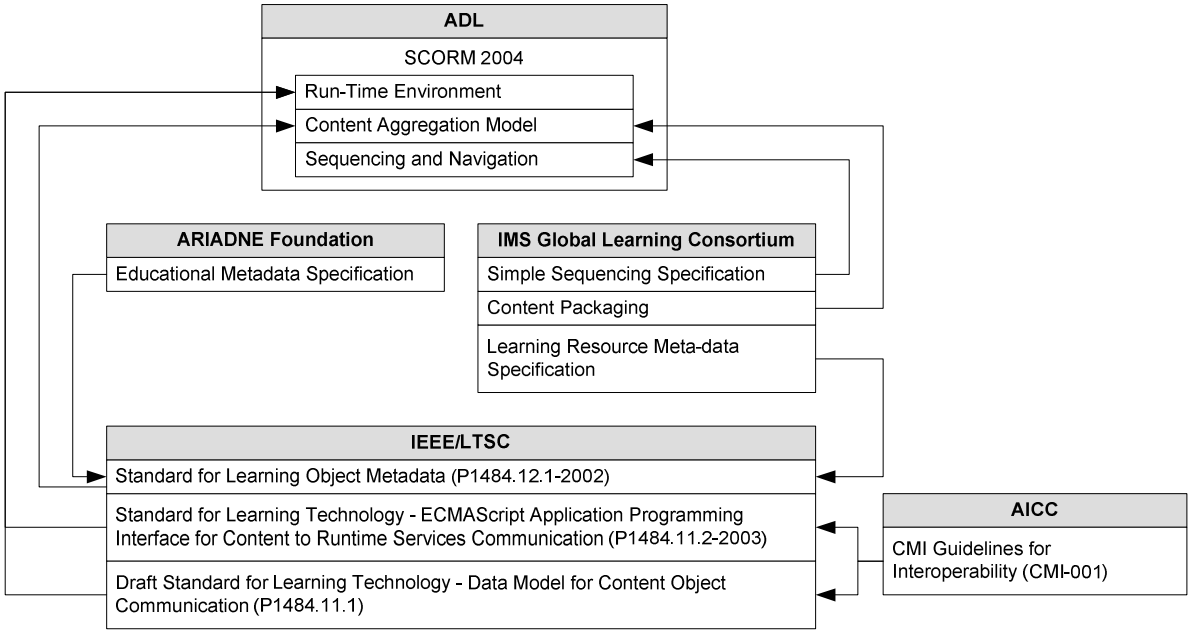


Abbildung 1: Zusammenhang zwischen den verschiedenen Standardisierungsprojekten

2 SCORM

In diesem Kapitel wird die aktuelle Version des Sharable Content Object Reference Models detailliert vorgestellt. Es wird auf die Geschichte und den Aufbau von SCORM eingegangen und die einzelnen SCORM-Bücher werden beschrieben. Dabei werden jeweils auch die Veränderungen zur Vorgängerversion hervorgehoben.
Da das SCORM Sequencing and Navigation Book ein zentraler Bestandteil dieser Arbeit und außerdem sehr umfangreich ist, wird es im Anschluss an dieses in einem eigenen Kapitel behandelt.

2.1 Überblick

Die im vorangegangenen Kapitel vorgestellten Organisationen, Initiativen und Projekte beschäftigen sich mit verschiedenen Aspekten auf dem Gebiet der Lerntechnologie und erarbeiten dabei eine Vielzahl von Spezifikationen und Richtlinien. Entwickler stehen daher oft vor der Entscheidung, welche Spezifikationen sie anwenden und implementieren sollen. Die ADL Initiative stellt eine Verbindung zwischen den verschiedenen Arbeiten her und vereint die einzelnen Spezifikationen in einem gemeinsamen Referenzmodell, dem Sharable Content Object Reference Model (SCORM). Dieses Referenzmodell liefert konkrete implementierbare Richtlinien, deren Umsetzung die Interoperabilität, Zugänglichkeit und Wiederverwendbarkeit von web-basierten Lerninhalten ermöglicht.
Jede in SCORM verwendete Spezifikation kann als einzelnes Buch innerhalb einer Bibliothek betrachtet werden. Diese Bücher sind zu drei Hauptthemen zusammengefasst, die jeweils auch als Buch bezeichnet werden. Insgesamt besteht SCORM 2004 aus vier einzelnen Büchern, siehe Abbildung 2:

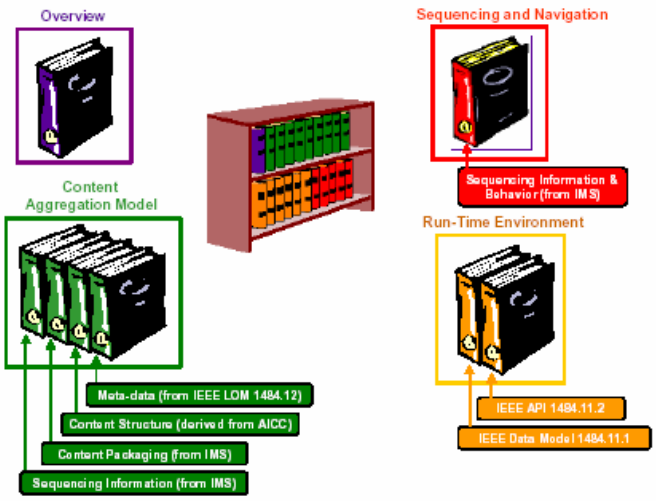


Abbildung 2: Die verschiedenen SCORM-Bücher [SCORM, 2004a]

Das SCORM 2004 Overview Book [SCORM, 2004a] dokumentiert die Geschichte und die Zielsetzung von ADL und SCORM sowie die Beziehungen der einzelnen SCORM-Bücher untereinander. Im Content Aggregation Model Book [SCORM, 2004b] werden Methoden für die Zusammenstellung und die Beschreibung von Lerninhalten beschrieben. Mechanismen zum Starten von Lerninhalten und für die Kommunikation und den Datenaustausch zwischen Lerninhalt und Lernmanagementsystem sind im Run-Time Environment Book [SCORM, 2004c] enthalten. Das Sequencing and Navigation Book [SCORM, 2004d] beschreibt die Ablaufsteuerung der Lerninhalte und die Navigation.
Die erste Ausgabe von SCORM (Version 1.0) wurde im Januar 2000 veröffentlicht und hieß damals noch Sharable Courseware Object Reference Model. Ein Jahr später wurde mit der Version 1.1 der Name in Sharable Content Object Reference Model umgewandelt, um zu verdeutlichen, dass SCORM nicht nur auf einen ganzen Kurs sondern auch auf verschiedenen Aggregationsstufen des Lerninhalts anwendbar ist. Im Oktober 2001 veröffentlichte ADL die Version 1.2. Dort wurde zum ersten Mal die oben beschriebene Einteilung in Bücher vorgenommen, damals waren es noch drei. Das vierte Buch, Sequencing and Navigation, ist erst seit der aktuellen Version vom Januar 2004 Bestandteil von SCORM. Mit dieser Veröffentlichung wurde auch das Versions-Prinzip verändert. Jedes Buch wird zukünftig unabhängig von den anderen fortgeführt, beginnend mit der Version 1.3. Damit sollen die Änderungen und Korrekturen übersichtlicher werden. Mittlerweile (Juli 2004) ist eine zweite Edition zu SCORM 2004 erschienen, die einzelnen Bücher tragen jetzt die Versionsnummer 1.3.1 und wurden überarbeitet. Die Version 1.3.1 bildet die Grundlage für diese Arbeit.
Mit jeder neuen Version wird SCORM an technische Entwicklungen und die Veränderungen an den zugrunde liegenden Spezifikationen angepasst.

2.2 Content Aggregation Model

Das Content Aggregation Model ist ein Modell für die Zusammenstellung von Lerninhalten zu komplexen Lernangeboten und beschreibt die Komponenten, aus denen sich ein Lernangebot zusammensetzt. Es wird definiert, wie diese Komponenten strukturiert und zusammengefasst (gepackt) werden müssen, um sie zwischen verschiedenen Systemen auszutauschen und wie sie mittels Metadaten beschrieben werden können, um die Suche und Identifizierung zu ermöglichen. Außerdem wird erläutert, wie den Komponenten Elemente für die Ablaufsteuerung zugeordnet werden können. Diese Inhalte sind in vier Teile gegliedert:

- Content Model
- Content Packaging
- Metadaten
- Sequencing und Navigation

2.2.1 Content Model

Lernangebote können unterschiedlich strukturiert und aufgebaut sein. Das Content Model beschreibt die verschiedenen Komponenten und erläutert, wie einfache Lernressourcen zu komplexen Lerneinheiten zusammengestellt werden können. Folgende Komponenten werden unterschieden: Asset, Sharable Content Object (SCO) und Content Organization.

Asset

Ein Asset ist die kleinste Form einer Lernressource und kann jegliche Form von elektronischen Daten repräsentieren, die ein Webbrowser darstellen oder verarbeiten kann, z. B. Texte, Bilder, Musik oder Javascript-Funktionen. Mehrere Assets können zu einem weiteren Asset zusammengesetzt werden.

Sharable Content Object (SCO)

Ein Sharable Content Object ist eine Zusammenstellung aus einem oder mehreren Assets oder auch aus Dateien, die nicht extra als Asset definiert sind. Der Unterschied zu einem Asset besteht darin, dass ein SCO über die Laufzeitumgebung mit dem LMS kommuniziert. Ein SCO muss in der Lage sein, die vom LMS bereitgestellte API-Instanz zu lokalisieren und muss mindestens die API-Methoden zum Initialisieren und Beenden der Kommunikation aufrufen. Der Aufruf weiterer API-Methoden (z. B. zum Abfragen oder Senden von Daten) ist optional und hängt vom Inhalt des SCOs ab (vgl. auch Kap. 2.3.2).
Um die Wiederverwendbarkeit der Lernressourcen zu gewährleisten, sollten SCOs unabhängig sein vom Gesamtkontext, in den sie eingebettet sind, sie sollten eine in sich abgeschlossene, sinnvolle Lerneinheit darstellen. SCORM enthält keine Beschränkung für die Größe eines SCOs, es sollte jedoch darauf geachtet werden, dass innerhalb eines SCOs ein bestimmtes Lernergebnis erreicht werden kann.

Content Organization

Eine Content Organization bildet die Struktur der Inhalte eines Lernangebots ab. Die Wurzel der baumförmigen Struktur bildet das `<organization>`-Element, darunter folgen verschachtelte `<item>`-Elemente. Das `<organization>`-Element und jedes `<item>`-Element

entspricht einer Activity (vgl. Kap. 3.1.1). Diese Struktur wird in der Manifest-Datei festgelegt, vgl. Kap. 2.2.2 .
An die einzelnen Activities können Sequencinginformationen gebunden werden. Das Sequencing wird nur auf die Activities angewendet, dadurch sind die Lernressourcen unabhängig von den Sequencinginformationen und können auch in verschiedenen Kontexten verwendet werden. Mehr dazu in Kapitel 3.

Im Vergleich zu SCORM 1.2 wurde das Content Model etwas verändert. Anstelle des Begriffs Content Aggregation wird nun Content Organization verwendet und Content Aggregation steht jetzt für die Zusammenstellung der Inhalte, also im Prinzip für das ganze Content Package. Abbildung 3 enthält alle Komponenten des Content Aggregation Models:

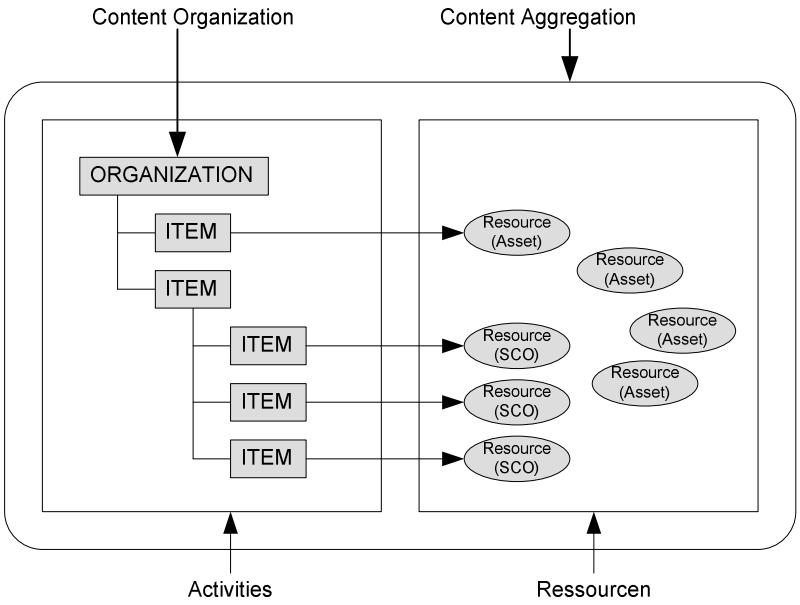


Abbildung 3: Komponenten des Content Aggregation Models, nach [SCORM, 2004b]

2.2.2 Content Packaging

Das SCORM Content Packaging liefert ein einheitliches Format, um Lerninhalte zwischen Lernmanagementsystemen, Werkzeugen und Kursdatenbanken auszutauschen und somit deren Wiederverwendbarkeit und Interoperabilität zu gewährleisten. SCORM Content Packaging ist ein Anwendungsprofil der IMS Content Packaging Specification, hält sich streng an diese Spezifikation und beinhaltet darüber hinaus zusätzliche Anforderungen und Implementationsrichtlinien.
Ein Content Package kann einen Teil eines Kurses, einen ganzen Kurs oder eine Sammlung von Kursen umfassen. Das Content Package muss eigenständig sein und muss alle benötigten Informationen für die Nutzung der enthaltenen Lernressourcen bereitstellen. SCORM

empfiehlt, Content Packages in Form einer komprimierten Archiv-Datei zu erstellen. Diese Datei wird als Package Interchange File (PIF) bezeichnet und sollte im .zip-Format erstellt werden.

Die zwei Komponenten eines Content Packages sind die Manifestdatei und die physischen Dateien, siehe Abbildung 4.

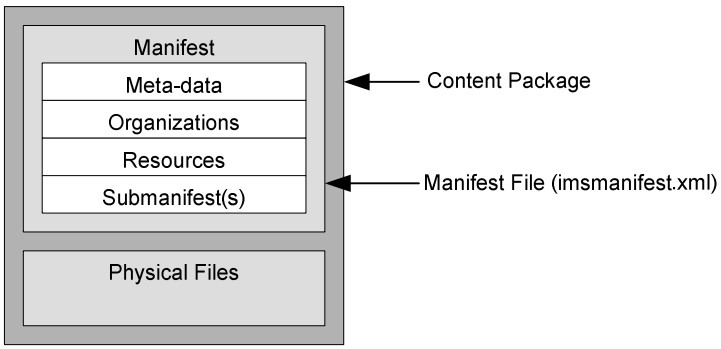


Abbildung 4: Content Package, nach [SCORM, 2004b]

Die Manifestdatei ist ein XML-Dokument, welches die Struktur des Inhalts sowie die Ressourcen eines Content Packages beschreibt. Der Gültigkeitsbereich der Manifestdatei ist entsprechend des Umfangs eines Content Packages elastisch, sie kann

- einen Teil eines Kurses, der unabhängig vom Kontext des ganzen Kurses ist
- einen kompletten Kurs
- eine Sammlung von Kursen oder
- eine Sammlung von Inhalten ohne Kursstruktur (Resource Package)

beschreiben.

Ein Package enthält genau ein Top-Level-Manifest, welches für das ganze Package steht. Dieses Top-Level-Manifest kann ein oder mehrere Sub-Manifeste enthalten. Außerdem ist festgelegt, dass die Manifestdatei `imsmanifest.xml` heißen und sich im Wurzelverzeichnis des Content Packages befinden muss. Codebeispiel 1 zeigt ein Beispiel für eine Manifestdatei:

```
<?xml version="1.0"?>
<manifest identifier="SAMPLE1" version="1.3"
               xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
               xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
               imscp_v1p1.xsd
               http://www.adlnet.org/xsd/adlcp_v1p3 adlcp_v1p3.xsd">
<!-------------------------------------------------------------------------------------------->
<metadata> <!-- Content Aggregation Metadata -->
  <schema>ADL SCORM</schema>
  <schemaversion>CAM 1.3</schemaversion>
  <adlcp:location>packageMetadata.xml</adlcp:location>
</metadata>
<!-------------------------------------------------------------------------------------------->
<organizations default="TOC1">
  <organization identifier="TOC1">
    <metadata> <!-- Content Organization Metadata -->
      <adlcp:location>orgTOC1Metadata.xml</adlcp:location>
    </metadata>
    <title> Introduction to SCORM for LMS Vendors </title>
```

```
    <item identifier="ITEM01">
      <title>SCORM Run-Time Environment Requirements</title>
      <item identifier="ITEM01-01" identifierref="RES01-01">
        <title>Launching</title>
      </item>
      <item identifier="ITEM01-02" identifierref="RES01-02">
        <title>API</title>
      </item>
      <metadata> <!-- Activity Metadata -->
        <adlcp:location>act01Metadata.xml</adlcp:location>
      </metadata>
    </item>
    <item identifier="ITEM02" identifierref="RES02">
      <title>LMS Conformance Requirements</title>
      <metadata> <!-- Activity Metadata -->
        <adlcp:location>act02Metadata.xml</adlcp:location>
      </metadata>
      <adlnav:presentation>
        <adlnav:navigationInterface>
          <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
        </adlnav:navigationInterface>
      </adlnav:presentation>
      <imsss:sequencing>
        <imsss:rollupRules rollupObjectiveSatisfied="false"/>
      </imsss:sequencing>
    </item>
  </organization>
</organizations>
<!-------------------------------------------------------------------------------------------->
<resources>
  <resource identifier="RES01-01" type="webcontent" adlcp:scormType="sco" href="sco1.html">
    <metadata> <!-- SCO Metadata -->
      <adlcp:location>SCOMetadata.xml</adlcp:location>
    </metadata>
    <file href="sco1.html"/>
    <file href="SCOFunctions.js"/>
  </resource>
  <resource identifier="RES01-02" type="webcontent" adlcp:scormType="sco" href="sco2.html">
    <file href="sco2.html"/>
    <file href="grafik1.jpg"/>
  </resource>
  <resource identifier="RES02" type="webcontent" adlcp:scormType="sco" href="sco3.html">
    <file href="sco3.html"/>
    <file href="SCOFunctions.js">
      <metadata> <!-- Asset Metadata -->
        <adlcp:location>asset01Metadata.xml</adlcp:location>
      </metadata>
    </file>
  </resource>
</resources>
<!-------------------------------------------------------------------------------------------->
<imsss:sequencingCollection>
  <!-- Sequencing data -->
</imsss:sequencingCollection>
</manifest>
```

Codebeispiel 1: imsmanifest.xml

Wie man sieht, ist die Datei in vier Abschnitte gegliedert. Im Bereich `<metadata>` können Metadaten für das gesamte Package (Content Aggregation Meta-data) eingefügt werden. Obwohl das Element `<metadata>` sowie dessen Unterelemente `<schema>` und `<schemaversion>` im CAM 1.3.1 obligatorisch sind, sind die Metadaten an sich weiterhin optional. Sie sollten aber für eine bessere Identifikation und Wiederverwendbarkeit der Lerninhalte nicht weggelassen werden. Die Metadaten können entweder direkt in der Manifestdatei als Inline-Metadaten hingefügt werden oder es wird mit dem `<location>`-Element eine externe XML-Datei referenziert, in der die entsprechenden Metadaten enthalten sind.

Im Abschnitt <organizations> wird die Struktur des Packages festgelegt. Wie schon unter 2.2.1 beschrieben, besteht eine solche Struktur aus dem Wurzelelement <organization> und darunter folgen unterschiedlich tief verschachtelte <item>-Elemente. Ist ein <item>-Element nicht weiter verschachtelt, verweist es auf eine Lernressource (SCO oder Asset) oder auf ein Sub-Manifest. In der Version 1.2 konnten auch Parent-<item>-Elemente auf ein <resource>-Element verweisen, dies ist jetzt nicht mehr erlaubt.
Das Element <organization> muss mindestens einmal, kann aber auch mehrmals enthalten sein. Mit verschiedenen Organizations kann der Autor unterschiedliche Abläufe der Lerninhalte vorschreiben, z. B. für Anfänger und Fortgeschrittene. Eine Organization muss dabei als "default" gekennzeichnet sein, falls das LMS mehrere nicht unterstützt.
In diesem Bereich können auch Vorschriften für die Ablaufsteuerung und die Anzeige von Navigationselementen eingefügt werden. Diese Vorschriften werden an die Activities gebunden, also an die <item>- und <organization>-Elemente. Dafür wurden die Elemente <imsss:sequencing> und <adlnav:presentation> eingeführt. In SCORM 1.2 war es lediglich möglich, so genannte "Prerequisites" an die <item>-Elemente zu binden. Prerequisites sind Vorbedingungen (z. B. der erfolgreiche Abschluss einer vorangegangenen Lektion), die erfüllt sein müssen, bevor die Lernressource vom System bereitgestellt wird. Das entsprechende Element <adlcp:prerequisites> wurde aus dem XML-Schema entfernt, ebenso <adlcp:maxtimeallowed> und <adlcp:masteryscore>, an deren Stelle treten die Sequencingangaben. Mit der Einführung des Simple Sequencing sind die Möglichkeiten zur Ablaufsteuerung umfangreicher geworden, im Kapitel 3 werden die Methoden sowie das XML-Binding in der Manifestdatei genauer erläutert.
Das <organization>-Element liefert dem LMS die Informationen für die Bereitstellung der Lerninhalte. Das LMS generiert aus dem Aufbau der Organization einen Pfad, auf dem sich der Lernende durch das Lernangebot bewegt. Je nachdem, ob Sequencinginformationen enthalten sind, wird der Pfad während der Abarbeitung des Lernangebots durch das LMS angepasst.
Sowohl dem <organization>-Element als auch den <item>-Elementen können optional Metadaten zugeordnet werden.
Der Bereich <resources> enthält eine Auflistung der Ressourcen, auf welche innerhalb der Content Organization verwiesen wurde. Innerhalb des <resource>-Elements wird auf die physische Datei verwiesen und es können auch hier Metadaten hinzugefügt werden.

Durch die Einführung des Simple Sequencing ist der Bereich `<imsss:sequencingCollection>` neu hinzugekommen. Dieser stellt einen Container für mehrfach verwendbare Sequencinginformationen dar. Mehr dazu im Kapitel 3.3.

2.2.3 Metadaten

Alle Komponenten des Content Model können mittels Metadaten beschrieben werden, so dass sie katalogisiert, gesucht und identifiziert werden können. Grundlage für die im Content Aggregation Model verwendeten Metadaten bildet der IEEE Standard for Learning Object Metadata. SCORM orientiert sich exakt an diesem Standard und beinhaltet außerdem zusätzliche Anforderungen. Alle im IEEE-Standard definierten Metadaten-Elemente sind optional, SCORM definiert aber bestimmte Elemente als obligatorisch, um die Identifizierung und somit die Wiederverwendbarkeit der Komponenten zu gewährleisten.
Die Metadaten-Elemente sind in neun Kategorien eingeteilt und jede Kategorie enthält zahlreiche Unterelemente. Tabelle 1 gibt einen Überblick über die Kategorien:

Kategorie	**Beschreibung**
`<general>`	Allgemeine Informationen, welche die Ressource als Ganzes beschreiben (z. B. Titel, Stichworte, Beschreibung)
`<lifeCycle>`	Geschichte und aktueller Stand der Ressource, sowie an der Entwicklung beteiligte Personen
`<metaMetadata>`	Informationen über den Metadatensatz selbst
`<technical>`	Beschreibung der technischen Anforderungen und Charakteristika der Ressource
`<educational>`	Beschreibung der pädagogischen Charakteristika der Ressource
`<rights>`	Informationen über Nutzungsrechte, Kosten, Copyright
`<relation>`	Definition von Beziehungen der Ressource zu anderen Ressourcen
`<annotation>`	Kommentare zum pädagogischen Nutzen der Lernressource
`<classification>`	Einordnung der Ressource in ein Klassifizierungsschema

Tabelle 1: Metadatenkategorien [Kammer, 2003]

Im Vergleich zum CAM 1.2 gibt es einige Veränderungen bei den Metadaten. Eine Änderung betrifft die Schreibweise der Elemente: Elementnamen aus mehreren Wörtern werden künftig in der so genannten Kamelschreibweise (camel case)[1] notiert (z. B. `<lifeCycle>` statt `<lifecycle>`). Andere Änderungen beziehen sich auf die Struktur des Metadatenschemas: Das Element `<identifier>` (Unterelement in verschiedenen Kategorien) soll die beschriebene Komponente eindeutig identifizieren. Es galt in CAM 1.2 als reserviert und sollte nicht verwendet werden, da noch keine Methode für die Erstellung einer einheitlichen ID festgelegt wurde. In CAM 1.3.1 wird es an Stelle des Elements `<catalogentry>`

[1] Bei Bezeichnungen aus mehreren Wörtern wird jedes Wort mit einem Großbuchstaben begonnen (außer das erste), das "Auf und Ab" der Buchstaben erinnert dabei an Kamelhöcker.

verwendet und enthält dessen Unterelemente <catalog> und <entry>. Tabelle 2 verdeutlicht diese Veränderung:

CAM 1.2	CAM 1.3.1
<general> <identifier> <!-- Reserved --> </identifier> <catalogentry> <catalog>ISBN</catalog> <entry>123-456</entry> </catalogentry> </general>	<general> <identifier> <catalog>ISBN</catalog> <entry>123-456</entry> </identifier> </general>

Tabelle 2: Änderungen des Metadatenschemas – Element <identifier>

Außerdem wurde dem Element <requirement>, welches die notwendigen technischen Voraussetzungen für das Benutzen des Lernangebots beschreibt, das Unterelement <orComposite> hinzugefügt. Damit können jetzt alternative Voraussetzungen beschrieben werden (z. B. verschiedene Browser).

CAM 1.2	CAM 1.3.1
<technical> <requirement> <type>...</type> <name>...</name> </requirement> </technical>	<technical> <requirement> <orComposite> <type>...</type> <name>...</name> </orComposite> </requirement> </technical>

Tabelle 3: Änderungen des Metadatenschemas – Element <requirement>

Eine weitere Änderung: das Unterelement <person> der Kategorie <annotation> wurde in <entity> umbenannt.

CAM 1.2	CAM 1.3.1
<annotation> <person>...</person> <date>...</date> <description>...</desription> </annotation >	<annotation> <entity>...</entity> <date>...</date> <description>...</desription> </annotation >

Tabelle 4: Änderungen des Metadatenschemas – Element <entity>

Im Metadaten-Anwendungsprofil ist festgelegt, welche Metadaten-Elemente optional und welche obligatorisch sind. Im Content Aggregation Model gibt es Metadaten-Anwendungsprofile für:

- Content-Aggregation-Metadaten
- Content-Organization-Metadaten
- Activity-Metadaten
- SCO-Metadaten und
- Asset-Metadaten,

wobei die Anwendungsprofile für die Content Organizations, Activities und SCOs identisch sind. Im CAM 1.2 gab es nur drei verschiedene Anwendungsprofile. Tabelle 5 zeigt die Unterschiede zum aktuellen Content Aggregation Model:

Position im Manifest innerhalb	CAM 1.2 Anwendungsprofil	CAM 1.3.1 Anwendungsprofil
`<manifest>`	Package-level-Metadaten (kein Anwendungsprofil, alle Elemente sind optional)	Content Aggregation Metadaten
`<organization>`	Content Aggregation Metadaten	Content Organization Metadaten
`<item>`	Content Aggregation Metadaten	Activity Metadaten
`<resource>` (wenn `adlcp:scormType="sco"`)	SCO Metadaten	SCO Metadaten
`<resource>` (wenn `adlcp:scormType="asset"`), `<file>`	Asset Metadaten	Asset Metadaten

Tabelle 5: Metadaten-Anwendungsprofile in CAM 1.2 und CAM 1.3.1, nach [SCORM 2004e]

Die Anwendungsprofile wurden entsprechend der Komponenten des CAM präziser benannt, um die Beschreibung und Strukturierung der Manifestdatei für den Content-Entwickler zu verbessern. Content-Organization-Metadaten (Metadaten innerhalb des `<organization>`-Elements in der Manifestdatei) und Activity-Metadaten (Metadaten innerhalb eines `<item>`-Elements in der Manifestdatei, siehe auch Codebeispiel 1) gab es im Prinzip auch im CAM 1.2, sie wurden nur nicht so bezeichnet, sondern als Content-Aggregation-Metadaten behandelt.

Grundsätzlich sind alle Metadaten optional, da die jeweiligen `<metadata>`-Elemente in der Manifestdatei optional sind, werden aber Metadaten eingefügt, müssen die Anforderungen der Anwendungsprofile eingehalten werden. Bei den Content-Aggregation-Metadaten sind bis auf zwei obligatorische Elemente (`<metaMetadata>` und `<metadataSchema>`) wie im IEEE LOM Standard alle Angaben optional, bei den Metadaten der restlichen Komponenten wurden einige Elemente als obligatorisch definiert. Die Tabelle im Anhang A enthält alle Metadaten-Anwendungsprofile.

2.3 Run-Time Environment

Um zu gewährleisten, dass einmal erstellte Lerninhalte ohne Modifizierung in verschiedenen LMS ausgeführt werden können, stellt SCORM ein Run-Time-Environment bereit. Dazu werden im SCORM RTE Book einheitliche Mechanismen für das Starten von Lernressourcen, für die Kommunikation zwischen Lernressourcen und LMS sowie ein Datenmodell für die Erfassung und Speicherung von verschiedenen Informationen beschrieben. Diese Inhalte sind in drei Teile gegliedert:

- Launch

- API
- Datenmodell

2.3.1 Launch

SCORM definiert im RTE Book das Verhalten von Lernmanagementsystemen beim Starten (Launch) von Lernressourcen. Die Implementierung im LMS wird dabei jedoch nicht näher spezifiziert.
SCOs und Assets sind die Komponenten des Content Models, die von einem LMS gestartet werden können. Sie werden auch als Content Objects bezeichnet. Je nach Typ des Content Objects werden unterschiedliche Anforderungen an das LMS gestellt. Für Assets gibt es lediglich die Vorgabe, dass sie über das HTTP-Protokoll gestartet werden. Wird ein SCO gestartet, muss dies in einem Child-Fenster bzw. einem Child-Frame des Fensters geschehen, welches die API-Instanz (vgl. Kap. 2.3.2) enthält, um zu gewährleisten, dass das SCO eine Kommunikationsverbindung zum LMS aufbauen kann. Außerdem ist festgelegt, dass ein LMS nie mehr als ein SCO gleichzeitig starten darf.
Die Identifizierung des Content Objects, welches als nächstes gestartet werden soll, findet innerhalb des LMS statt, z. B. über die in der Content Organization definierte Struktur oder über Sequencing-Mechanismen. Anhand der Launch-Location, die innerhalb des Content Package für die Ressource definiert wurde (Attribut `href` im `<resource>`-Element), navigiert das LMS zur gewünschten Ressource und stellt sie dem Lernenden zur Verfügung.
Dem Launch-Prozess liegt ein Zeitmodell zugrunde. Dieses definiert verschiedene Begriffe, welche für die Verwaltung des Run-Time Environment eines SCOs von Bedeutung sind.

Learner Attempt

Ein Learner Attempt bezeichnet den Versuch eines Lernenden, eine Activity zu bearbeiten. Dieser Versuch kann sich über eine oder mehrere Learner Sessions erstrecken und kann zwischen Learner Sessions unterbrochen werden. Wurde eine Activity zur Bereitstellung identifiziert, beginnt der Learner Attempt. Er wird beendet, wenn eine Learner Session in einem normalen Status beendet wird, d.h. wenn die Bearbeitung des SCOs abgeschlossen und nicht nur unterbrochen ist.

Learner Session

Eine Learner Session ist eine zusammenhängende Zeitspanne, in der der Lernende auf ein Content Object zugreift, also vom Starten bis zum Entfernen des Content Objects. Endet eine Learner Session, gilt das SCO entweder als abgeschlossen und somit ist auch der Learner

Attempt beendet oder es gilt als unterbrochen und der Learner Attempt wird mit der nächsten Learner Session fortgesetzt.

Communication Session

Die Zeitspanne, in der eine aktive Verbindung zwischen SCO und API besteht, wird als Communication Session bezeichnet. Sie beginnt, sobald das SCO mit der Methode `Initialize()` die Kommunikation aufbaut und endet mit dem Aufruf der Methode `Terminate()`, siehe auch Kap. 2.3.2.

Login Session

Die Login Session umfasst den Zeitraum, in dem der Lernende beim LMS angemeldet (eingeloggt) ist.

Abbildung 5 stellt die Zusammenhänge zwischen den Begriffen dar:

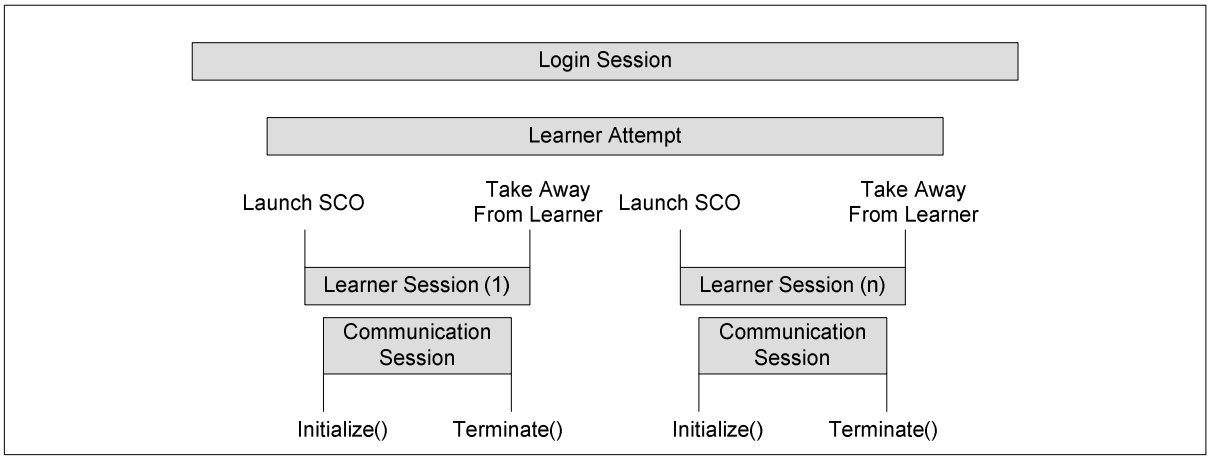


Abbildung 5: Temporal Model, nach [SCORM, 2004c]

2.3.2 API

Das Application Programming Interface (API) ermöglicht einen standardisierten Datentransfer zwischen den Lerninhalten und dem Lernmanagementsystem, indem es verschiedene Funktionen als Schnittstelle bereitstellt. Von den bereits erläuterten Komponenten des SCORM Content Models sind nur SCOs in der Lage, mit dem LMS zu kommunizieren. Dies geschieht über eine API-Instanz, die vom LMS zur Verfügung gestellt werden muss. Vorteilhaft ist dabei, dass der Entwickler der Lerninhalte sich nicht mit Einzelheiten der Kommunikation beschäftigen muss, da er lediglich die standardisierten Schnittstellen nutzt. Für die API-Implementation innerhalb des LMS gibt es bezüglich der Programmiersprache keine Festlegungen, der Zugriff durch ein SCO auf die API-Instanz

muss über ECMAScript (JavaScript) erfolgen. Zugrunde liegender Standard ist der IEEE P1484.11.2 - Standard for Learning Technology - ECMAScript Application Programming Interface for Content to Runtime Services Communication.

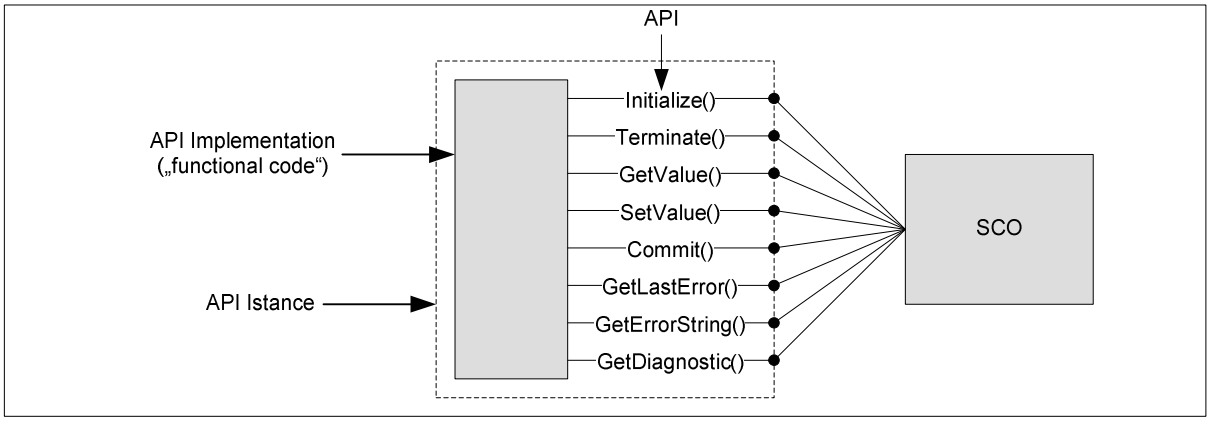


Abbildung 6: API, nach [SCORM, 2004c]

Nach dem Launch eines SCOs muss dieses die API-Instanz lokalisieren. Damit dies mit einem einheitlichen Mechanismus realisiert werden kann, muss die API-Instanz als Objekt mit dem Namen "`API_1484_11`" (in SCORM 1.2 "`API`") über das Document Object Model (DOM) ansprechbar sein. Außerdem muss das SCO entweder in einem Child-Browserfenster oder einem Child-Frame des LMS-Browserfensters, welches die API-Instanz enthält, gestartet werden. Codebeispiel 2 enthält Funktionen für das Auffinden der API-Instanz:

```
var nFindAPITries = 0;
var API = null;
var maxTries = 500;
var APIVersion = "";

function ScanForAPI(win)
{
  while ((win.API_1484_11 == null) && (win.parent != null) && (win.parent != win))
  {
    nFindAPITries++;
    if (nFindAPITries > maxTries)
    {
      alert("Error in finding API instance -- too deeply nested.");
      return null;
    }
    win = win.parent;
  }
  return win.API_1484_11;
}
function GetAPI()
{
  if ((win.parent != null) && (win.parent != win))
  {
    API = ScanForAPI(win.parent);
  }
  if ((API == null) && (win.opener != null))
  {
    API = ScanForAPI(win.opener);
  }
  if (API != null)
  {
    APIVersion = API.version;
  }
}
```

Codebeispiel 2: Auffinden der API-Instanz [SCORM, 2004c]

Die insgesamt acht bereitgestellten API-Funktionen sind in drei Kategorien unterteilt. Im Vergleich zu SCORM 1.2 wurden die Funktionsnamen verändert. Tabelle 6 enthält einen Überblick zu den Namensänderungen sowie eine kurze Funktionsbeschreibung:

RTE 1.2	**RTE 1.3.1**	**Beschreibung**
Session-Methoden		
LMSInitialize("")	Initialize("")	Start der Kommunikations-Session
LMSFinish("")	Terminate("")	Beenden der Kommunikations-Session
Datentransfer-Methoden		
LMSGetValue(data model element)	GetValue(data model element)	Abfrage von Informationen vom LMS
LMSSetValue(data model element, value)	SetValue(data model element, value)	Transfer von Daten zum LMS
LMSCommit("")	Commit("")	Übertragung zwischengespeicherter Daten zum LMS
Support-Methoden		
LMSGetLastError()	GetLastError()	Abfrage des aktuellen Fehlerzustands (Fehlernummer)
LMSGetErrorString(error code)	GetErrorString(error code)	Abfrage einer wörtlichen Beschreibung des Fehlers
LMSGetDiagnostic(value)	GetDiagnostic(value)	Abfrage von zusätzlichen Informationen zu dem Fehler

Tabelle 6: API-Methoden

Für ein SCO ist es lediglich erforderlich, dass es die Methoden `Initialize()` und `Terminate()` aufruft, die Anwendung der übrigen Methoden ist optional. Mit den Methoden `GetValue()` und `SetValue()` können die Werte für die Elemente des Datenmodells (vgl. 2.3.3) abgefragt bzw. gesendet werden. Dabei wird als Parameter der Name des Elements übergeben, bei `SetValue()` außerdem der Wert für das Element. Alle Parameter müssen als Characterstring angegeben werden. Dies ist in einigen Fällen umständlich, da z. B. ein berechneter Punktwert, der für *cmi.score.scaled* gesetzt werden soll, vor der Übertragung noch in einen String umgewandelt werden muss.
Es ist nicht zwingend vorgeschrieben, dass die Daten durch die API-Instanz sofort zum Server übertragen werden müssen, sie können auch im lokalen Cache des Clients gespeichert werden. Ist letzteres der Fall werden die zwischengespeicherten Daten erst mit dem Aufruf der Methode `Commit()` dauerhaft auf der Serverseite gespeichert. Dies soll Verzögerungen verhindern, die entstehen, wenn jeder Wert einzeln und sofort übertragen wird.
Das Abfragen und Senden von Daten ist nur dann möglich, wenn vorher die Verbindung zur API-Instanz hergestellt wurde. Dazu kann bereits beim Laden der HTML-Seite über das Ereignis "onLoad" im `<body>`-Tag die Funktion `Initialize()` aufgerufen werden. Ebenso kann über das Ereignis "onUnload" beim Verlassen der Seite die Funktion `Terminate()` aufgerufen werden. Codebeispiel 3 enthält ein HTML-Dokument mit dem entsprechenden (vereinfachten) JavaScript-Code:

```
<html>
  <head>
    <script language="JavaScript">
      function doInitialize()
      {
        var api = findAPI();
        api.Initialize("");
      }

      function doTerminate()
      {
        var api = findAPI();
        api.Terminate("");
      }

      function findAPI()
      {
        ...
        return APIHandle;
      }
    </script>
  </head>
  <body onload=" doInitialize()" onunload="doTerminate()">
    Inhalt
  </body>
</html>
```

Codebeispiel 3: Initialisierung und Beendigung der Kommunikation zwischen LMS und SCO

Bei jedem Aufruf einer Funktion, bis auf die Support-Methoden, setzt die API-Instanz den aktuellen Errorcode neu. Mit der Funktion `GetLastError()` sollte innerhalb des SCOs der Errorcode überprüft werden und gegebenenfalls eine Fehlermeldung angezeigt werden. Der Rückgabewert der Funktion `GetErrorString()` enthält eine textuelle Beschreibung zum übergebenen Fehlercode und kann somit für die Erstellung der Fehlermeldung genutzt werden. Für noch genauere Fehlermeldungen kann die Funktion `GetDiagnostic()` verwendet werden. Diese Methode ist LMS-spezifisch und erlaubt die Definition von zusätzlichen Fehlerbeschreibungen durch den Hersteller des LMS.
Neu im RTE 1.3.1 ist, dass der Inhalt des Rückgabewerts der Methode `GetErrorString()` nicht mehr festgelegt ist, sondern innerhalb des LMS frei wählbar ist. Außerdem wurden die Errorcodes geändert und erweitert, siehe Tabelle 7.

RTE 1.2 Error Code	**RTE 1.3.1 Error Code**
0 – No Error	0 – No Error
101 – General Exception	101 – General Exception
	102 – General Initialization Failure
	103 – Already Initialized
	104 – Content Instance Terminated
	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

RTE 1.2 Error Code	RTE 1.3.1 Error Code
202 – Element cannot have children 203 – Element not an array. Cannot have count	301 – General Get Failure
	351 – General Set Failure
	391 – General Commit Failure
401 – Not implemented error	401 – Undefined Data Model Element
401 – Not implemented error	402 – Unimplemented Data Model Element
301 – Not initialized	403 – Data Model Element Value Not Initialized
403 – Element is read only	404 – Data Model Element Is Read Only
404 – Element is write only 402 – Invalid set value, element is a keyword	405 – Data Model Element Is Write Only
405 – Incorrect Data Type	406 – Data Model Element Type Mismatch
	407 – Data Model Element Value Out Of Range

Tabelle 7: Error Codes in RTE 1.2 und RTE 1.3.1, nach [SCORM, 2004e]

2.3.3 Datenmodell

Das RTE-Datenmodell stellt einen festgelegten Satz von Elementen zur Verfügung, mit denen zur Laufzeit über die API-Instanz Informationen zwischen SCO und LMS ausgetauscht werden können. Codebeispiel 4 zeigt einen Auszug aus einem JavaScript, welches anhand eines berechneten Punktwertes den Erfolgsstatus für das SCO ermittelt und den Punktwert und den Erfolgsstatus zum LMS überträgt:

```
var score = 0;
var minRequiredScore = 2;
var anzahlAufgaben = 3;

function calcScore()
{
  //Berechnung des Punktestandes
  ...

  if ( score >= minRequiredScore )
  {
    setObjToPassed();
  }
  else
  {
    setObjToFailed();
  }
}

function setObjToPassed()
{
  doSetValue("cmi.success_status","passed");
  var tempScoreScaled = (score / anzahlAufgaben);
  tempScoreScaled = tempScoreScaled.toString();
  doSetValue("cmi.score.scaled",tempScoreScaled);
}

function setObjToFailed()
{
  doSetValue("cmi.success_status","failed");
  var tempScoreScaled = (score / anzahlAufgaben);
  tempScoreScaled = tempScoreScaled.toString();
  doSetValue("cmi.score.scaled",tempScoreScaled);
}

function doSetValue(name, value)
{
  var api = getAPIHandle();
  api.SetValue(name, value);
}
```

Codebeispiel 4: Nutzung des RTE-Datenmodells

Durch die Benutzung eines einheitlichen Datenmodells ist sichergestellt, dass die Informationen in verschiedenen LMS übereinstimmend verarbeitet werden können. Basierte das Datenmodell in der Version 1.2 noch direkt auf den AICC CMI001 Guidelines for Interoperability, bildet jetzt der IEEE 1484.11.1 – Draft Standard for Data Model for Content Object Communication die Grundlage.
Im RTE-Datenmodell können verschiedene Informationen gespeichert und abgerufen werden: u.a. Daten über den Lernenden, über seine Interaktionen mit dem Lerninhalt, über den Lernerfolg oder den Abschluss von SCOs. Diese Daten sind auch Grundlage für LMS-interne Sequencing-Entscheidungen, da die Werte einiger Elemente in die Ablaufsteuerung einbezogen werden.
Alle Elemente des RTE-Datenmodells müssen von einem SCORM-konformen LMS unterstützt werden, in der Version 1.2 waren nur bestimmte Elemente obligatorisch. Für SCOs sind weiterhin alle Elemente optional. Zur Kennzeichnung des zugrunde liegenden Datenmodells beginnen alle Elemente mit "cmi.", danach folgt der Elementname entsprechend des hierarchischen Aufbaus des Modells. Jedem Datenelement ist ein Datentyp zugeordnet. Im RTE 1.2 wurden vom AICC definierte Datentypen verwendet, diese wurden jetzt in ISO-Datentypen umgewandelt. In der Tabelle im Anhang B sind alle Datenelemente mit den Änderungen zum aktuellen Datenmodell aufgelistet. Die in SCORM 2004 neu hinzugekommenen Elemente werden nun genauer erläutert.

Neue Datenelemente im RTE 1.3.1- Datenmodell

cmi.success_status

Dieses Datenelement ersetzt zusammen mit *cmi.completion_status* das Element *cmi.core.lesson_status* und zeigt an, ob der Lernende das SCO erfolgreich abgeschlossen hat. Wie der erfolgreiche Abschluss definiert ist, ist in SCORM nicht festgelegt und muss vom Entwickler des SCOs z. B. anhand von erreichten Punkten in einem Test oder anhand der Anzahl von erfolgreich absolvierten Übungen festgelegt werden. Das Element hat den Datentyp *state* und kann folgende Werte annehmen:

- passed – der Lernende hat das SCO erfolgreich absolviert
- failed – der Lernende hat das SCO nicht erfolgreich absolviert
- unknown – es ist keine Angabe zum Status des SCOs möglich (default)

Die Ermittlung des Erfolgsstatus geschieht innerhalb des SCOs, es ist jedoch auch möglich, dass *cmi.success_status* vom LMS gesetzt werden muss: der Wert von *cmi.success_status* wird von zwei anderen Elementen beeinflusst, *cmi.scaled_passing_score* und

cmi.score.scaled. Ist *cmi.scaled_passing_score* definiert und überträgt das SCO den Wert für *cmi.score.scaled*, muss das LMS den Wert für *cmi.success_status* entsprechend setzen.
Der Wert von *cmi.success_status* beeinflusst die Tracking-Informationen der mit dem SCO verbundenen Activity (*Objective Satisfied Status* des Primary Objectives der Activity, vgl. Kap. 3.2.1) und damit das Sequencing.

cmi.completion_status

Dieses Datenelement gibt an, ob der Lernende das SCO vollständig abgearbeitet hat, ungeachtet einer erreichten Punktzahl oder ähnlichem. Die Definition der vollständigen Abarbeitung liegt beim Entwickler des SCOs und ist in SCORM nicht festgelegt. Der Status könnte z. B. anhand der Anzahl von besuchten Seiten (bei mehrseitigen SCOs) oder des Lesens eines bestimmten Dokuments festgelegt werden. Das Element hat den Datentyp *state* und kann folgende Werte annehmen:

- completed – der Lernende hat das SCO ausreichend abgearbeitet
- incomplete – das SCO ist noch nicht ausreichend abgearbeitet
- not_attempted – der Lernende hat das SCO nur unwesentlich genutzt (z. B. gestartet, aber nur das Inhaltsverzeichnis betrachtet)
- unknown – es ist keine Angabe zum Status des SCOs möglich (default)

Der Wert "browsed", der für *cmi.core.lesson_status* noch gültig war, wurde entfernt.
Die Ermittlung des Abarbeitungstatus ist Sache des SCOs, es ist jedoch auch möglich, dass *cmi.completion_status* vom LMS gesetzt werden muss: *cmi.completion_status* wird von zwei anderen Elementen beeinflusst, *cmi.completion_threshold* und *cmi.progress_measure.* Ist *cmi.completion_threshold* definiert (über das Element `<adlcp:completionThreshold>` in der Manifestdatei) und *cmi.progress_measure* wird vom SCO gesetzt, muss das LMS den Wert für *cmi.completion_status* entsprechend setzen. Der Wert von *cmi.completion_status* beeinflusst die Tracking-Informationen der mit dem SCO verbundenen Activity (*Attempt Completion Status* des Primary Objectives der Activity, vgl. Kap. 3.2.3) und damit das Sequencing.

cmi.score.scaled

Mit diesem Element wird die Leistung des Lernenden für das gesamte SCO als Zahlenwert wiedergegeben, skaliert im Bereich von -1 bis 1. Das SCO ist verantwortlich für die Berechnung des Wertes. Wird dieser Wert gegenüber *cmi.scaled_passing_score* ausgewertet, kann sich der Wert von *cmi.success_status* ändern. Außerdem beeinflusst der Wert des Elements *cmi.score.scaled* den *Objective Normalized Measure* des Primary Objective der mit

dem SCO verbundenen Activity. Übermittelt das SCO einen Wert für *cmi.score.scaled* an das LMS, erhält *Objective Normalized Measure* den Wert von *cmi.score.scaled*, vgl. Kap. 3.2.1.

cmi.objectives.n.score.scaled

Mit diesem Element wird die Leistung des Lernenden für ein lokales Objective einer Activity (vgl. Kap. 3.1.3) in Form eines Punktwertes wiedergegeben, skaliert im Bereich von -1 bis 1. Festgelegt wird der Wert innerhalb des SCOs. Übermittelt das SCO einen Wert für *cmi.objectives.n.score.scaled*, wird *Objective Normalized Measure* (vgl. Kap. 3.2.1) für das entsprechende Objective dem Wert von *cmi.objectives.n.score.scaled* gleichgesetzt.

cmi.objectives.n.success_status

Dieses Datenelement ersetzt zusammen mit *cmi.objectives.n.completion_status* das Element *cmi.objectives.n.status* und gibt an, ob der Lernende das Objective erfolgreich absolviert hat. Wie eine erfolgreiche Absolvierung definiert ist, ist in SCORM nicht festgelegt und muss vom Entwickler des SCOs z. B. anhand von erreichten Punkten in einem Test oder anhand der Anzahl von absolvierten Übungen, die zu dem Objective gehören, festgelegt werden.
Das Element hat den Datentyp *state* und kann folgende Werte annehmen:

- passed – der Lernende hat das Objective erfolgreich absolviert
- failed – der Lernende hat das Objective nicht erfolgreich absolviert
- unknown – es ist keine Angabe zum Status des Objective möglich (default)

Die Ermittlung des Erfolgsstatus ist Sache des SCOs, wird *cmi.objectives.n.success_status* nicht vom SCO gesetzt, sollte das LMS "unknown" als Wert annehmen. Der Wert von *cmi.objectives.n.success_status* beeinflusst die mit dem SCO verbundene Activity (*Objective Satisfied Status* des entsprechenden lokalen Objectives, vgl. Kap. 3.2.1) und damit das Sequencing.

cmi.objectives.n.completion_status

Dieses Datenelement gibt an, ob der Lernende das Objective vollständig absolviert hat. Die Definition der vollständigen Absolvierung liegt beim Entwickler des SCOs (z. B. anhand der Anzahl beantworteter Testfragen) und ist in SCORM nicht festgelegt. Das Element hat den Datentyp *state* und kann folgende Werte annehmen:

- completed – der Lernende hat das SCO ausreichend bearbeitet, um das Objective als abgeschlossen zu betrachten
- incomplete – das SCO ist noch nicht ausreichend bearbeitet, um das Objective als abgeschlossen zu betrachten

- not_attempted – der Lernende hat zwar das SCO bearbeitet, aber das Objective selber wurde nicht bearbeitet
- unknown – es ist keine Angabe zum Status des Objectives möglich (default)

Der Wert "browsed", der für *cmi.objectives.n.status* noch gültig war, wurde entfernt.
Die Ermittlung des Abarbeitungsstatus ist Sache des SCOs, wird *cmi.objectives.n.completion_status* nicht vom SCO gesetzt, sollte das LMS "unknown" als Wert annehmen.

cmi.objectives.n.description

In diesem Element kann eine kurze Beschreibung für das Objective gespeichert werden. Es ist vom Datentyp localized_string_type und der Wert wird vom SCO initialisiert.

cmi.interactions.n.description

In diesem Element kann eine kurze Beschreibung für die Interaktion gespeichert werden. Es ist vom Datentyp localized_string_type und der Wert wird vom SCO initialisiert.

cmi.completion_threshold

Dieses Element kann genutzt werden, um den Completion Status für ein SCO zu ermitteln, es enthält den Schwellenwert für den Completion Status. In der Manifest-Datei kann für ein `<item>`-Element, welches auf eine SCO-Ressource verweist, mittels `<adlcp:completionThreshold>` dieser Schwellenwert angegeben werden. Das LMS sollte das Element mit dem dort angegebenen Wert initialisieren. Der Datentyp des Elements ist real(10,7).

cmi.progress_measure

Das Maß für den Fortschritt des Lernenden bei der Abarbeitung des SCOs wird mit diesem Datenelement festgelegt. Der Wert wird innerhalb des SCOs ermittelt, z. B. basierend auf der Absolvierung einer bestimmten Anzahl von Objectives oder anhand einer bestimmten Anzahl von Seiten, die dem Lernenden präsentiert wurden. Der Datentyp des Elements ist real(10,7). Ist *cmi.completion_threshold* definiert und das SCO übermittelt einen Wert für *cmi.progress_measure*, muss das LMS den Wert für *cmi.completion_status* entsprechend setzen.

Die Umstrukturierung im RTE-Datenmodell bringt deutliche Verbesserungen für das Verständnis des Content-Entwicklers bei der Arbeit mit diesem Modell. Es erfolgte eine

schlüssigere Beschreibung der Elemente (z. B. Umbenennung von *cmi.comments* in *cmi.comments_from_learner*) und eine sinnvolle Aufteilung von Einzelelementen aus SCORM 1.2 auf mehrere Elemente bei SCORM 1.3.1 zur besseren Identifizierung von kombinierten Zuständen des SCOs (z. B. Trennung von *cmi.core.lesson_status* in *cmi.success_status* und *cmi.completion_status*). Es entsteht allerdings dadurch die Problematik der Inkompatibilität zwischen SCORM 1.2 und 1.3.1, durch die Umbenennungen der Elemente ist die Abwärtskompatibilität nicht mehr gegeben.
Eine weitere Verbesserung ist die Möglichkeit zur Beschreibung von Objectives und Interactions (*cmi.objectives.n.description* und *cmi.interactions.n.description*). Diese Beschreibungen haben in SCORM 1.2 völlig gefehlt, sind aber nützlich für Auswertungsanzeigen im LMS nach Verlassen des Kurses.
Neben den Umbenennungen bei den API-Methoden wurden die Fehlermeldungsfunktionen gegenüber der Version 1.2 verbessert und lassen aussagekräftigere Fehlermeldungen durch das LMS zu. Dies ist für Content-Entwickler wichtig, da durch eine genauere Beschreibung Fehler schneller zu finden sind.

3 Sequencing und Navigation

Dieses Kapitel beschäftigt sich mit dem SCORM Sequencing and Navigation Book. Es werden die zugrunde liegenden Konzepte und Modelle erläutert und es wird detailliert beschrieben, wie der Content-Entwickler die Sequencinginformationen im Content Package integrieren kann. Es folgen die Darstellung der verschiedenen Sequencingprozesse sowie ein Vergleich zum Sequencing aus SCORM 1.2. Den Abschluss des Kapitels bildet die Beschreibung des Navigation Models.

3.1 Überblick

Ein Ziel von SCORM ist es, Lernangebote so zu gestalten, dass die Inhalte wiederverwendbar sind. Bei herstellerspezifischen Anwendungen sind die Struktur und auch die Ablaufsteuerung für die Lerninhalte meist untrennbar mit dem Inhalt verbunden. Mit der Einführung des Sequencing wurde eine Möglichkeit geschaffen, die Ablaufsteuerung vom Inhalt zu trennen, so dass einmal erstellte Lernmaterialien wiederverwendbar sind und die Ablaufsteuerung trotzdem nach individuellen Bedürfnissen gestaltet werden kann.
Das SCORM Sequencing basiert auf der IMS Simple Sequencing Spezifikation [IMS, 2003], welche definiert, wie das gewünschte Ablaufverhalten innerhalb eines Lernangebots festgelegt wird, so dass es in jedem SCORM-konformen LMS gleichermaßen umgesetzt wird.
Das SCORM Sequencing and Navigation Book definiert die Anwendung der IMS-Spezifikation und SCORM-spezifische Erweiterungen. Es wird beschrieben

- wie die Interaktionen des Lernenden erfasst und verarbeitet werden
- welche Sequencinginformationen angegeben werden können, um die gewünschte Sequencingstrategie umzusetzen
- welche Prozesse und Verhalten SCORM-konforme LMS umsetzen müssen, um das Sequencing zu steuern
- wie die Navigation und die Benutzeroberfläche vom Content-Entwickler beeinflusst werden können

Dabei bezieht sich SCORM Sequencing lediglich auf die Ablaufsteuerung zwischen den einzelnen Activities (vgl. Kap. 3.1.1), die Navigation innerhalb des Content Objects einer Activity (z. B. mehrseitige SCOs) wird vom LMS nicht gesteuert und somit auch nicht erfasst (getrackt).
Wichtige Konzepte des Sequencing sind die Activity, der Activity Tree und die Objectives. Sie werden im Folgenden erläutert.

3.1.1 Activity

Das Sequencing wird auf Activities angewendet. Anhand der vom Entwickler der Lerninhalte definierten und mit der Activity verbundenen Sequencinginformationen und der Interaktionen des Lernenden ermittelt das LMS zur Laufzeit die sequenzielle Abfolge der einzelnen Activities.

Eine Activity ist eine Lernhandlung, etwas was der Lernende während der Abarbeitung eines Lernangebots "tut". Eine Activity ist entweder eine Blatt-Activity und referenziert eine Lernressource oder sie besteht aus weiteren Sub-Activities, welche wiederum aus weiteren Activities aufgebaut sein können. Für die Verschachtelungstiefe gibt es keine Begrenzung. In Abbildung 7 ist eine Activity mit Sub-Activities dargestellt. "Take Lesson" ist eine Activity und ist aus den Sub-Activities "Take a Pre-Test", "Experience Content" und "Take a Final-Test" aufgebaut. Die Sub-Activities sind Blatt-Activities und verweisen auf die entsprechende Lernressource.

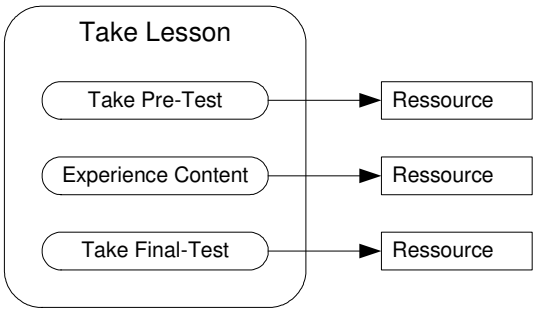


Abbildung 7: Activity mit Sub-Activities, nach [SCORM, 2004d]

Solche verschachtelten Activities werden Cluster genannt. Ein Cluster umfasst die Eltern-Activity und ihre direkten Kind-Activities, deren Kinder wiederum zählen nicht dazu, Kinder eines Clusters sind also entweder Blatt-Activities und verweisen auf eine Lernressource oder sind selbst wieder Cluster, siehe Abbildung 8.

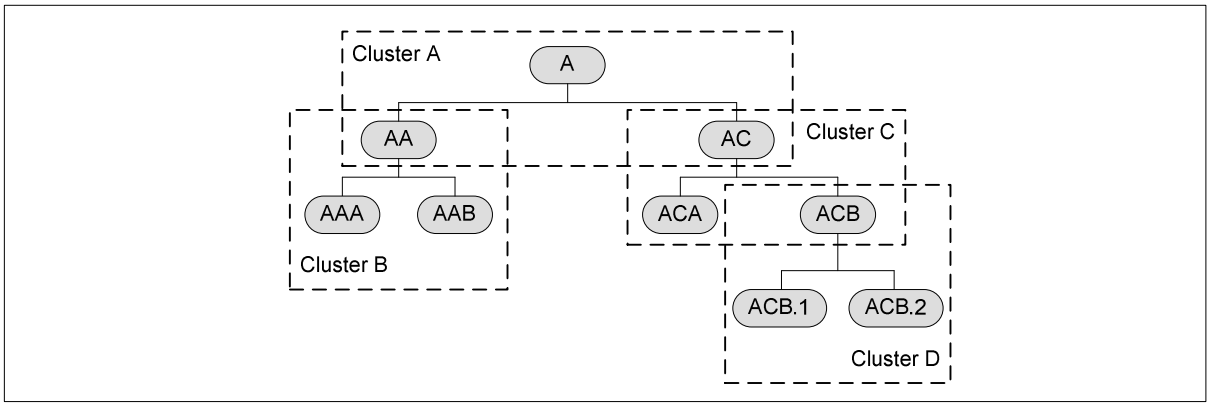


Abbildung 8: Cluster-Activities, nach [SCORM, 2004d]

Activities stehen in direktem Zusammenhang mit der Content Organization und ihre Struktur wird mit einem Activity Tree beschrieben.

3.1.2 Activity Tree

Ein Activity Tree wird aus der Content Organization abgeleitet, wobei die Organization (`<organization>`-Element) die Wurzel-Activity des Baumes bildet und jedes `<item>`-Element einer Activity entspricht.

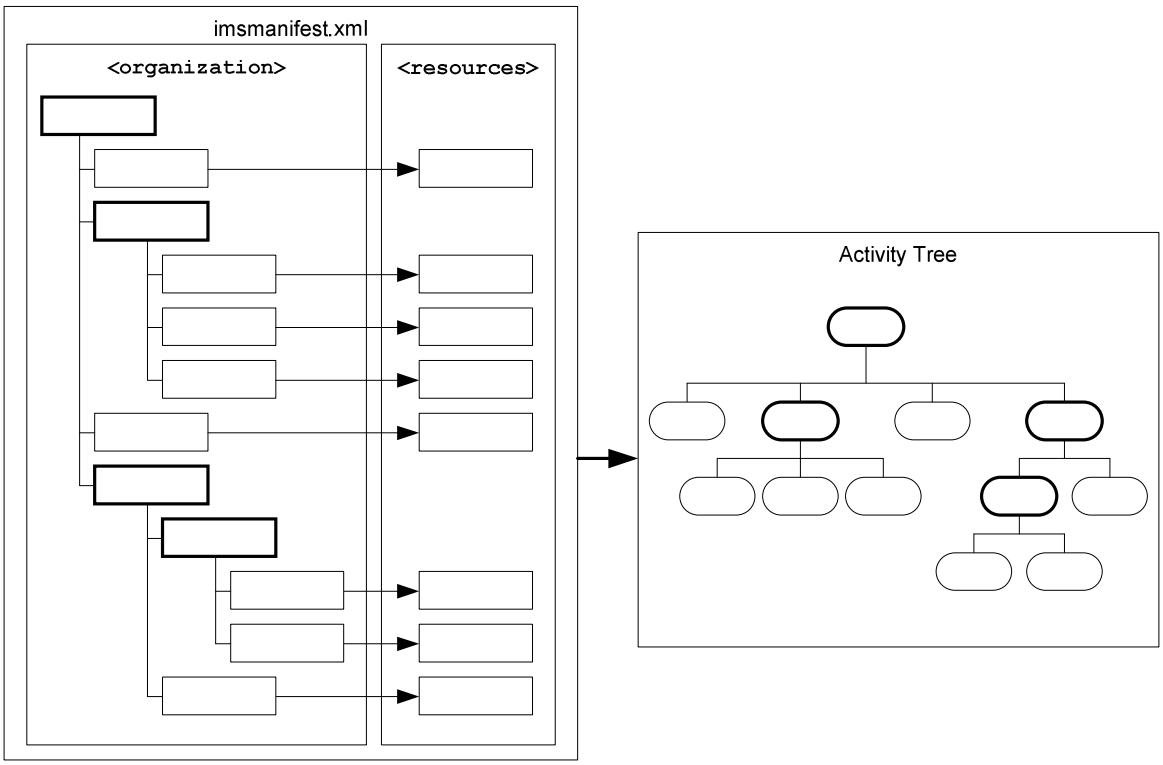


Abbildung 9: Ableitung des Activity Tree aus der Content Organization, nach [SCORM, 2004d]

Die Sequencing-Implementation des LMS arbeitet mit dem Activity Tree. Dies ermöglicht, dass die Sequencinginformationen unabhängig von den Lerninhalten sind.
Aus der Struktur des Activity Trees kann das LMS u.a. auch ein Navigationsmenü für die Lernumgebung generieren.

3.1.3 Objectives

Activities können mit einem oder mehreren Objectives verbunden werden. Objectives sind keine Lernziele im didaktischen Sinne sondern eher Variablen, mit denen Statuswerte (bestanden/nicht bestanden und ein Ergebniswert) gespeichert und von verschiedenen Activities gemeinsam genutzt werden können. Man kann mit Objectives also den Lernerfolg des Lernenden festhalten. SCORM macht aber keine Angaben zur Bedeutung eines

Objectives, es kann z. B. einer Frage in einem Abschlusstest, einer Übungsaufgabe oder einem ganzen Test mit vielen Fragen entsprechen.
Es gibt lokale und globale Objectives. Lokale Objectives können nur von der Activity genutzt werden, an die sie gebunden sind. Globale Objectives sind allen Activities zugänglich und nicht explizit an eine Activity gebunden. Ihre Werte können von den lokalen Objectives gelesen werden und lokale Objectives können die Werte der globalen überschreiben.
Abbildung 10 zeigt ein Beispiel für die Verwendung von Objectives, "objective1" ist ein globales Objective und wird von zwei Activities genutzt, alle anderen Objectives sind lokal.

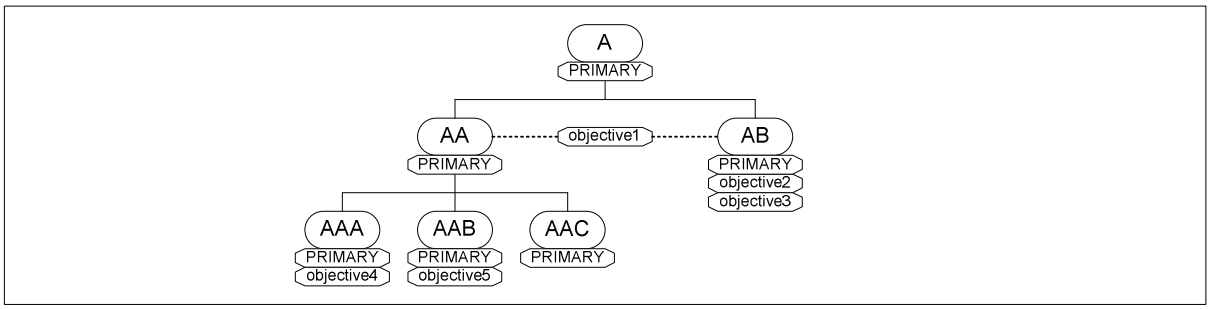


Abbildung 10: Objectives

Wie aus Abbildung 10 ersichtlich, ist jeder Activity automatisch ein Primary Objective zugeordnet. Dort wird der Status der Activity gespeichert, also ein Ergebniswert für die gesamte Activity und ob die Activity erfolgreich bearbeitet wurde oder nicht. Dieses Primary Objective wird für das Rollup (vgl. Kap. 3.3.5) genutzt.
Die Definition von Objectives wird unter Kapitel 3.3.7 näher beschrieben.
Die Objectives finden sich im Datenmodell der Laufzeitumgebung und im Tracking Model wieder und sind über eine ID in beiden Modellen identifizierbar. So können Daten, die von einem SCO zum LMS übertragen werden (z. B. *cmi.success_status*), im Tracking Model genutzt werden. Mehr dazu im Abschnitt 3.2.

Weitere wichtige Konzepte des SCORM Sequencing sind das Sequencing Definition Model (statisches Datenmodell zur Definition der beabsichtigten Sequencingstrategie innerhalb des Content Packages), das Tracking Model (dynamisches Datenmodell, welches verschiedene Informationen über die Activities zur Laufzeit registriert) und der Overall Sequencing Process, welcher die verschiedenen Sequencingverhalten und -prozesse beinhaltet. Diese Konzepte werden in den folgenden Abschnitten beschrieben.

3.2 Tracking Model

In älteren SCORM-Versionen gab es nur das Datenmodell der Laufzeitumgebung, welches die Interaktionen des Lernenden mit einem SCO registriert hat. Durch die Einführung des Sequencings müssen nun auch die Interaktionen des Lernenden in Bezug auf die Activities registriert werden, um sie in den verschiedenen Sequencingprozessen verarbeiten zu können. Für jede Activity werden vom LMS so genannte Tracking-Informationen gespeichert, z. B. ob und wie oft auf eine Activity zugegriffen wurde, welche Activity gerade aktiv ist oder wie erfolgreich der Lernende beim Bearbeiten einer Activity ist usw. Die Gesamtheit der Elemente, welche die Tracking-Informationen enthalten, bilden das Tracking Model.
Für jedes Element innerhalb des Tracking Models sind default-Werte definiert, die dann zur Laufzeit durch die Interaktionen des Lernenden verändert werden. Das Tracking Model ist also eine dynamische Sammlung von Status-Informationen.
Einige Elemente des RTE-Datenmodells werden direkt auf Elemente im Tracking Model abgebildet. So entspricht zum Beispiel das Element *cmi.completion_status* des RTE-Datenmodells dem Element *Attempt Completion Status* im Tracking Model. Tracking-Informationen von Activities, die auf SCOs verweisen, basieren also zum Teil auf Daten, die direkt vom SCO zum LMS übertragen wurden. Abbildung 11 zeigt den allgemeinen Zusammenhang zwischen den beiden Modellen, die genauen Zusammenhänge werden unter Kapitel 3.2.1 und 3.2.3 beschrieben.

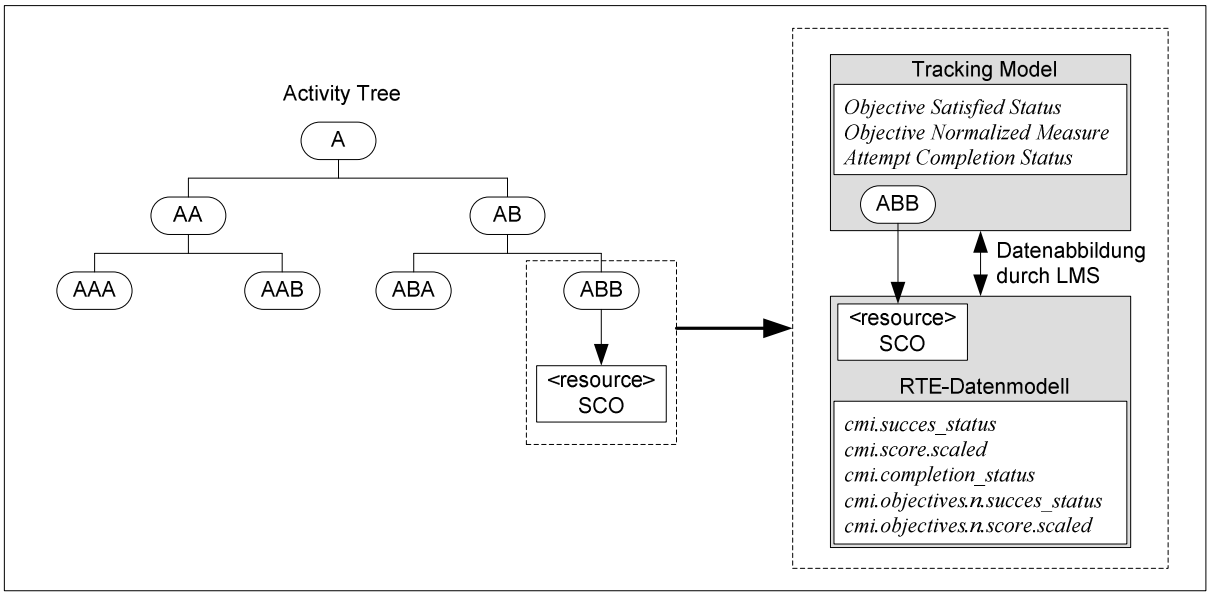


Abbildung 11: Zusammenhang RTE-Datenmodell und Tracking Model, nach [SCORM, 2004d]

Welche Tracking-Informationen erfasst werden, wird in den folgenden Abschnitten beschrieben.

3.2.1 Objective Progress Information

Die Objective Progress Information beschreibt den Erfolg des Lernenden bezüglich eines Objectives und wird für jedes Objective einer Activity registriert. Durch das Primary Objective gibt es also mindestens einen Satz dieser Informationen pro Activity. Die Objective Progress Information besteht aus vier Elementen, siehe Tabelle 8:

Element	Beschreibung
Objective Progress Status	Gibt an, ob das Objective einen gültigen *Objective Satisfied Status* hat ("true") oder nicht ("false", default).
Objective Satisfied Status	Gibt an, ob das Objective erfolgreich bearbeitet wurde ("true") oder nicht ("false", default).
Objective Measure Status	Gibt an, ob *Objective Normalized Measure* für das Objective gültig ist ("true") oder nicht ("false", default).
Objective Normalized Measure	Enthält den Ergebniswert für das Objective (default = 0).

Tabelle 8: Objective Progress Information

Die Elemente können zu Paaren zusammengefasst werden, das eine Element enthält den eigentlichen Wert, das andere Element definiert, ob der Wert gültig ist. Dieses Vorgehen ist nötig für die Umsetzung der verschiedenen Sequencingprozesse, zum leichteren Verständnis wird aber nur ein Element genutzt, um das Paar zu beschreiben (z. B. bei der Auswertung der Bedingungen von Sequencing Rules, vgl. Kap. 3.3.3). Ist vom *Objective Satisfied Status* die Rede, wird folgendes Vokabular mit entsprechender Bedeutung benutzt:

Status	Bedeutung
satisfied	*Objective Progress Status* "true" *Objective Satisfied Status* "true"
not satisfied	*Objective Progress Status* "true" *Objective Satisfied Status* "false"
unknown	*Objective Progress Status* "false"

Tabelle 9: Objective Satisfied Status

Für *Objective Normalized Measure* gilt:

Status	Bedeutung
Zahlenwert	*Objective Measure Status* "true"
unknown	*Objective Measure Status* "false"

Tabelle 10: Objective Normalized Measure

Aus dem RTE-Datenmodell können einige Elemente auf die Objective Progress Information abgebildet werden. Diese Datenabbildung findet innerhalb des LMS statt, wie das realisiert wird, ist in SCORM nicht festgelegt.
Wird innerhalb eines SCOs der Wert für das Datenelement zum LMS übertragen, muss das entsprechende Element des Tracking Models aktualisiert werden. In Tabelle 11 sind die betroffenen Elemente aufgeführt:

RTE-Datenmodell	Objective Progress Information
cmi.success_status passed failed unknown	*Objective Satisfied Status*, für das Primary Objective satisfied not satisfied unknown
cmi.score.scaled	*Objective Normalized Measure*, für das Primary Objective
cmi.objectives.n.success_status passed failed unknown	*Objective Satisfied Status* , für das lokale Objective mit der entsprechenden ID (*cmi.objectives.n.id*) satisfied not satisfied unknown
cmi.objectives.n.score.scaled	*Objective Normalized Measure*, für das lokale Objective mit der entsprechenden ID (*cmi.objectives.n.id*)

Tabelle 11: Datenabbildung zwischen RTE-Datenmodell und Objective Progress Information

Der folgende Auszug aus einer Manifest-Datei soll den Zusammenhang zwischen den beiden Datenmodellen noch einmal verdeutlichen:

```
<item identifier="AUFGABEN_01"identifierref="RES_AUFGABEN_01">
  <title>Aufgaben Lektion 1</title>
  <imsss:sequencing>
    <imsss:objectives>
      <imsss:primaryObjective objectiveID="PRIMARY">
        <imsss:mapInfo targetObjectiveID="obj_lektion01"
           readSatisfiedStatus="false" readNormalizedMeasure="false"
           writeSatisfiedStatus="true" writeNormalizedMeasure="true" />
      </imsss:primaryObjective>
    </imsss:objectives>
  </imsss:sequencing>
</item>

<!-- other manifest data -->

<resource identifier="RES_AUFGABEN_01" href="Lektion01_Aufgabe.htm" adlcp:scormType="sco"
   type="webcontent">
  <file href="Lektion01_Aufgabe.htm" />
  <file href="style.css" />
</resource>
```

Codebeispiel 5: Zusammenhang zwischen RTE-Datenmodell und Objective Progress Information

Angenommen das Codebeispiel 4 von Seite 28 ist in der Datei `Lektion01_Aufgabe.htm` enthalten: mit der Übertragung von *cmi.success_status* und *cmi.score.scaled* werden die Tracking-Informationen für das Primary Objective der Activity `AUFGABEN_01` entsprechend Tabelle 11 aktualisiert. Zusätzlich wird in diesem Beispiel die Objective Progress Informationen noch auf das global Objective `obj_lektion01` übertragen.
Sind für eine Activity lokale Objectives definiert und referenziert diese Activity ein SCO, werden beim Start des SCOs für jedes lokale Objective die Elemente der Kategorie *cmi.objectives* des RTE-Datenmodells initialisiert. Anhand eines konkreten Beispiels wird dieser Zusammenhang in Kapitel 4.3 genauer beschrieben.

3.2.2 Activity Progress Information

Die Activity Progress Information beschreibt den Fortschritt des Lernenden bezüglich der ganzen Activity. Diesen Informationssatz gibt es genau einmal pro Activity und er umfasst

alle Zugriffe auf die Activity. Folgende vier Elemente definieren die Activity Progress Information:

Element	Beschreibung
Activity Progress Status	Gibt an, ob die Werte der restlichen drei Elemente für die Activity von Bedeutung sind ("true") oder nicht ("false", default). Dieses Element wird auf "true" gesetzt, sobald das erste Mal auf die Activity zugegriffen wird.
Activity Absolute Duration	Gibt die summierte Dauer aller Zugriffe auf die Activity an (default=0). Dazu gehört auch die Zeit, in der auf Kind-Activities dieser Activity zugegriffen wird. Allerdings ist in SCORM nicht festgelegt, wie diese Dauer ermittelt wird und es ist für ein SCORM-konformes LMS nicht erforderlich, dieses Element zu unterstützen.
Activity Experienced Duration	Gibt die summierte Dauer aller Zugriffe auf die Activity an (default=0). Es wird nur die Zeit berechnet, in der sich der Lernende direkt mit dieser Activity beschäftigt, dieser Wert wird also nur für Blatt-Activities ermittelt. Auch dieses Element muss von SCORM-konformen LMS nicht unterstützt werden.
Activity Attempt Count	Enthält die Anzahl der Zugriffe auf die Activity (default=0) und wird bei jedem Zugriff hochgezählt. Der aktuelle Zugriff ist inklusive, der Wert 0 bedeutet also, dass auf die Activity noch nicht zugegriffen wurde.

Tabelle 12: Activity Progress Information

3.2.3 Attempt Progress Information

Die Attempt Progress Information beschreibt den Fortschritt des Lernenden bezüglich des aktuellen Zugriffs. Für jeden Zugriff auf eine Activity wird ein Satz der Attempt Progress Information angelegt. Folgende Elemente sind enthalten:

Element	Beschreibung
Attempt Progress Status	Gibt an, ob die Werte der restlichen vier Elemente von Bedeutung sind ("true") oder nicht ("false", default). Sobald während des Zugriffs eine Tracking-Information registriert wird (z. B. wenn *Activity Attempt Count* hochgezählt wird), wird dieser Wert auf "true" gesetzt.
Attempt Completion Status	Gibt an, ob der Zugriff komplett ist ("true") oder nicht ("false", default), also ob die Activity vollständig bearbeitet wurde oder nicht (siehe unten, Zusammenhang mit RTE-Datenmodell).
Attempt Completion Amount	Gibt ein Maß für den Abschluss des Zugriffs im Wertebereich von 0 bis 1 an. 1 bedeutet, dass der Zugriff komplett ist. SCORM macht keine Angaben zur Ermittlung dieses Wertes und das Element muss nicht unterstütz werden.
Attempt Absolute Duration	Gibt die Dauer des Zugriffs auf die Activity an (default=0). Dazu gehört auch die Zeit, in der auf Kind-Activities dieser Activity zugegriffen wird. Allerdings ist in SCORM nicht festgelegt, wie diese Dauer ermittelt wird und es ist für ein SCORM-konformes LMS nicht erforderlich, dieses Element zu unterstützen.
Attempt Experienced Duration	Gibt die Dauer des Zugriffs auf die Activity an (default=0). Es wird nur die Zeit berechnet, in der sich der Lernende direkt mit dieser Activity beschäftigt, dieser Wert wird also nur für Blatt-Activities ermittelt. Auch dieses Element muss von SCORM-konformen LMS nicht unterstützt werden.

Tabelle 13: Attempt Progress Information

Auch hier können *Attempt Progress Status* und *Attempt Completion Status* als Paar betrachtet werden. *Attempt Progress Status* gibt an, ob der Status gültig ist und *Attempt Completion Status* enthält den eigentlichen Status. Ist vom *Attempt Completion Status* die Rede, wird folgendes Vokabular mit entsprechender Bedeutung genutzt:

Status	Bedeutung
completed	*Attempt Progress Status* "true" *Attempt Completion Status* "true"
incomplete	*Attempt Progress Status* "true" *Attempt Completion Status* "false"
unknown	*Attempt Progress Status* "false"

Tabelle 14: Attempt Completion Status

Für den *Attempt Completion Status* gibt es ein entsprechendes Element im RTE-Datenmodell: *cmi.completion_status*. Wird der Wert dieses Elements vom SCO gesetzt, muss der Wert von *Attempt Completion Status* entsprechend Tabelle 15 angepasst werden:

RTE-Datenmodell	Attempt Progress Information
cmi.completion_status completed incomplete unknown	*Attempt Completion Status* completed incomplete unknown

Tabelle 15: Zusammenhang zwischen RTE-Datenmodell und Attempt Progress Information

Wird vom mit der Activity verbundenen Content Object kein Wert für den Completion-Status gesetzt (was z. B. bei Assets auch nicht möglich ist), wird vom LMS angenommen, dass der *Attempt Completion Status* "completed" ist, wenn der Zugriff beendet wird.

3.2.4 Activity State Information

Mit der Activity State Information werden für jede Activity noch zusätzliche Status-Informationen registriert, die während des Overall Sequencing Process (vgl. Kap. 3.4) benötigt werden. Die drei Elemente sind in Tabelle 16 aufgeführt:

Element	Beschreibung
Activity is Active	Gibt an, ob gerade auf eine Activity zugegriffen wird ("true") oder nicht ("false", default). Mit dem Beginn des Zugriffs wird der Wert "true" gesetzt, mit dem Ende des Zugriffs auf "false".
Activity is Suspended	Gibt an, dass eine Activity gerade unterbrochen ist ("true").
Available Children	Enthält eine geordnete Liste aller Kind-Activities, auf die zugegriffen werden kann. Dieses Element wird nur für Cluster-Activities registriert. Durch Selection Controls und Randomization Controls kann diese Liste beeinflusst werden, vgl. 3.3.8.

Tabelle 16: Activity State Information

Durch das Element *Activity is Active* erhält der Activity Tree einen Active-Pfad. Dieser Pfad beginnt an der Wurzel des Activity Trees und endet an der *Current Activity* (vgl. Kap. 3.2.5). Alle Activities entlang des Pfades haben den Wert "true" für *Activity is Active*.

Ebenso verhält es sich mit dem Element *Activity is Suspended*: wird die Bearbeitung einer Blatt-Activity unterbrochen, gelten auch alle Vorfahren bis zur Wurzel als unterbrochen.

3.2.5 Global State Information

Mit der Global State Information werden Status-Informationen bezüglich des Activity Trees registriert, welche ebenfalls während des *Overall Sequencing Process* benötigt werden. Dies geschieht mit folgenden Elementen:

Element	Beschreibung
Current Activity	Dieses Element kennzeichnet die Activity, die dem Lernenden gerade präsentiert wird.
Suspended Activity	Enthält die Activity, von der aus die Unterbrechung ausgelöst wurde.

Tabelle 17: Global State Information

3.3 Sequencing Definition Model

Das Sequencing Definition Model besteht aus einer Reihe von Elementen, mit denen der Content-Entwickler seine beabsichtigte Sequencingstrategie umsetzen kann. Der Content-Entwickler kann verschiedene Dinge festlegen, z. B. ob Activities innerhalb eines Menüs angezeigt werden, unter welchen Bedingungen bestimmte Activities übersprungen werden, wie oft auf eine Activity maximal zugegriffen werden darf usw. Er definiert also mit Hilfe dieser Elemente, wann dem Lernenden welche Inhalte präsentiert werden und liefert dem LMS die Informationen, die es benötigt, um den Ablauf, die Auswahl und Bereitstellung der Activities so zu steuern, wie es vom Content-Entwickler gewünscht ist.
Im SCORM CAM wird beschrieben, wie mittels dieser Elemente die Sequencingstrategie innerhalb der Manifest-Datei codiert wird. Jedem `<item>`-Element und dem `<organization>`-Element können diese Sequencinginformationen zugeordnet werden, da diese Elemente jeweils einer Activity entsprechen. Dies geschieht durch das Element `<sequencing>`, siehe Codebeispiel 6:

```
<organizations default="ORG_01">
  <organization identifier="ORG_01">
    <title>Sicherung der Ablaufintegritaet</title>
    <item identifier="EINLEITUNG" identifierref="RES_EINLEITUNG">
      <title>Einleitung</title>
      <imsss:sequencing>
        <imsss:rollupRules rollupObjectiveSatisfied="false" />
      </imsss:sequencing>
    </item>
    <imsss:sequencing>
      <imsss:controlMode flow="true" />
    </imsss:sequencing>
  </organization>
</organizations>
```

Codebeispiel 6: Einfügen von Sequencinginformationen in der Manifest-Datei

Jedes Element hat einen default-Wert, welcher vom LMS verwendet wird, wenn kein anderer Wert explizit gesetzt wird. Weiterhin enthält jedes Element den Namensraum-Präfix `imsss`, um auf die Herkunft der Elemente aus der IMS-Spezifikation zu verweisen[2].
Nachfolgend werden die verschiedenen Steuerungsmöglichkeiten erläutert und die Verwendung der entsprechenden Elemente in der Manifest-Datei dargestellt.

3.3.1 Sequencing Control Modes

Mit den Sequencing Control Modes wird in erster Linie der Inhalt des Navigationsmenüs beeinflusst. Der Content-Entwickler legt fest, auf welche Weise der Weg durch den Activity Tree zurückgelegt werden kann und welche Activities direkt im Menü ausgewählt werden können. Zum Beispiel kann er bestimmen, dass die einzelnen Activities nur durch Blättern über "Vor"- und "Zurück"-Buttons ausgewählt werden können oder dass nur die Activities auf der Ebene "Kapitel" direkt über ein Menü auswählbar sind, die darunterliegenden "Lektionen" jedoch nicht.
Das entsprechende Element in der Manifest-Datei ist <`controlMode`>, siehe Codebeispiel 7. In diesem Beispiel wird festgelegt, dass die einzelnen Lektionen nicht direkt angewählt werden können, sondern dass das Kapitel nur durch vor- und zurückblättern durchlaufen werden kann.

```
<item identifier="KAPITEL_01">
  <title>Kapitel 1</title>
  <item identifier="LEKTION_01" identifierref="RES_LEKTION_01">
    <title>Lektion 1</title>
  </item>
  <item identifier="LEKTION_02" identifierref="RES_LEKTION_02">
    <title>Lektion 2</title>
  </item>
  <imsss:sequencing>
    <imsss:controlMode choice="false" flow="true" />
  </imsss:sequencing>
</item>
```

Codebeispiel 7: <controlMode>

Das Element <`controlMode`> enthält keine Unterelemente, nur optionale Attribute, entsprechend der verschiedenen Modi, mit einem binären Wertebereich (true/false), siehe Tabelle 18:

[2] Elemente, die von ADL zusätzlich zur IMS-Spezifikation eingeführt wurden, enthalten den Namensraum-Präfix `adlseq`.

Ebene	Element
1	<controlMode> **Attribute:** • choice • choiceExit • flow • forwardOnly • useCurrentAttemptObjectiveInformation • useCurrentAttemptProgressInformation

Tabelle 18: Übersicht <controlMode>

choice

Wenn dieses Attribut den Wert "true" besitzt (default), kann der Lernende jede Activity innerhalb des Clusters in beliebiger Reihenfolge und ohne Beschränkungen auswählen, sofern dies nicht andere Steuerungsmodi oder Sequencing Rules (vgl. Kap. 3.3.3) verhindern. Das LMS muss dem Lernenden die Möglichkeit geben, die Activities auszuwählen, z. B. über ein Menü. Abbildung 12 macht deutlich, wie dieser Control Mode den Inhalt des Menüs beeinflusst:

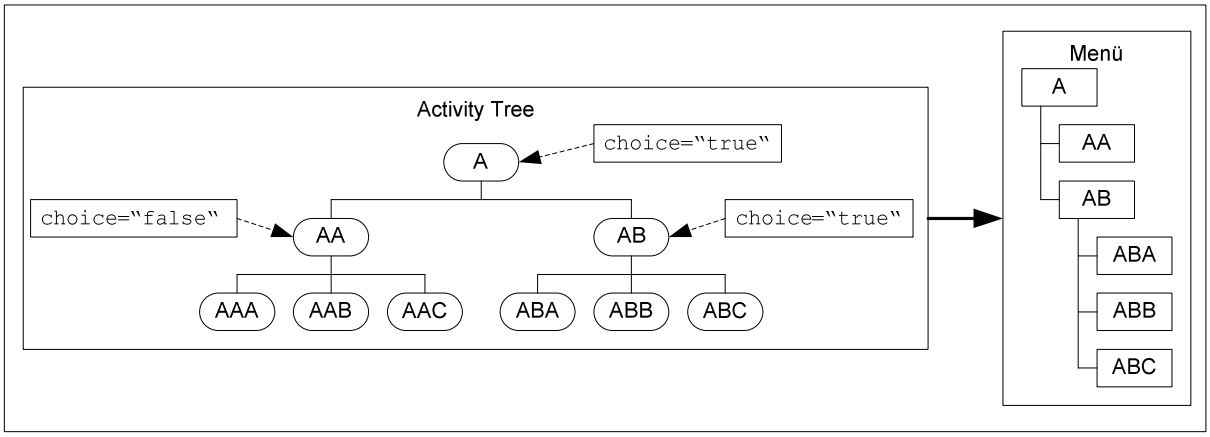


Abbildung 12: Einfluss des Control Modes "choice" auf das angezeigte Menü

Die Activities innerhalb des Clusters AA sind für die direkte Auswahl nicht freigegeben und erscheinen somit nicht im Menü. Die Activity AA sollte allerdings zusätzlich den Control Mode `flow="true"` erhalten, damit AAA, AAB und AAC über "Vor"- bzw. "Zurück"-Buttons ausgewählt werden können.

choiceExit

Dieses Attribut gibt an, ob von der betroffenen Activity aus andere Activities per *Choice* Navigation Request ausgewählt werden dürfen und dadurch die betroffene Activity beendet werden darf. Hätte z. B. die Activity AA in Abbildung 12 zusätzlich noch das Attribut `choiceExit="false"`, würde bei der Bereitstellung dieser Activity kein Menü mehr

dargestellt werden (bzw. wären die Activities im Menü nicht mehr auswählbar; wie dies vom LMS umgesetzt wird, ist in SCORM nicht festgelegt), da es nicht erlaubt ist, die aktuelle Activity durch Auswählen anderer Activities zu beenden. Es ist dann lediglich möglich, durch vor- oder zurückblättern über Buttons des LMS die Activity zu verlassen. Dieses Verfahren ist z. B. sinnvoll, wenn dem Lernenden die Auswahl verschiedener Kapitel (hier AA und AB) ermöglicht werden soll, ein einmal begonnenes Kapitel aber komplett bearbeitet werden muss, bevor das nächste ausgewählt werden kann.
Der default-Wert des Attributs ist "true".

flow

Das Attribut `flow` gibt an, ob eine vom LMS gelenkte Steuerung durch die Activities eines Clusters unterstützt wird, das heißt, ob das Blättern mit "Vor"- und "Zurück"-Buttons möglich ist. Hat das Attribut den Wert "true" (default-Wert ist "false"), muss das LMS dem Lernenden Mittel zur Verfügung stellen (i.d.R. Buttons), mit denen er vorangegangene und nachfolgende Activities auswählen kann.

forwardOnly

Hat das Attribut forwardOnly den Wert "true" (default-Wert ist "false"), können die Activities des Clusters nur vorwärtsgerichtet durchlaufen werden. Ein *Previous* Navigation Request ist nicht erlaubt und auch kein *Choice* Navigation Request, der innerhalb des Clusters zu einer rückwärts gelegenen Activity führen würde.

useCurrentAttemptObjectiveInformation

`useCurrentAttemptObjectiveInformation` gibt an, ob während der verschiedenen Sequencingpozesse die Objective Progress Information (vgl. Kap. 3.2) des aktuellen Zugriffs ("true", default) oder des vorangegangenen Zugriffs ("false") auf die Activity verwendet wird.

useCurrentAttemptProgressInformation

`useCurrentAttemptProgressInformation` gibt an, ob während der verschiedenen Sequencingprozesse die Attempt Progress Information (vgl. Kap. 3.2) des aktuellen Zugriffs ("true", default) oder des vorangegangenen Zugriffs ("false") auf die Activity verwendet wird.

Die einzelnen Modi schließen einander nicht aus, sie können miteinander kombiniert werden. Sie können für jede Activity innerhalb des Aktivitätsbaumes angewendet werden, allerdings

sind die Attribute bis auf choiceExit ohne Bedeutung, wenn sie an Blatt-Activities angewendet werden.
Es wird vorausgesetzt, dass das LMS verbotene Navigation Requests verhindert, also dass es keine Activities zur Auswahl anbietet, die eine Verletzung des Steuerungsmodus hervorrufen. Wird kein Modus explizit gesetzt, werden die default-Werte verwendet, das heißt, alle Activities sind vom Lernenden z. B. über ein Menü beliebig auswählbar.

3.3.2 Constrain Choice Controls

Jede Activity, deren Eltern-Activity das Attribut `choice="true"` besitzt, ist ein gültiges Ziel für einen *Choice* Navigation Request. Um diese Flexibilität einzuschränken, gibt es zusätzlich zum IMS Simple Sequencing das Element `<constrainedChoiceConsiderations>`. Im Codebeispiel 8 stehen die Lektionen von Kapitel 2 erst zur Auswahl, wenn die Activity `KAPITEL02` aktiv ist, wenn sie also z. B. über Vorwärtsblättern erreicht wurde. Es soll verhindert werden, dass die Lektionen von Kapitel 2 ausgewählt werden, bevor Kapitel 1 bearbeitet wurde.

```
<item identifier="KAPITEL_02">
  <title>Kapitel 2</title>
  <item identifier="LEKTION_01" identifierref="RES_LEKTION_01">
    <title>Lektion 1</title>
  </item>
  <item identifier="LEKTION_02" identifierref="RES_LEKTION_02">
    <title>Lektion 2</title>
  </item>
  <imsss:sequencing>
    <adlseq:constrainedChoiceConsiderations preventActivation="true" />
  </imsss:sequencing>
</item>
```

Codebeispiel 8: <constrainedChoice>

Das Element `<constrainedChoiceConsiderations>` enthält keine Unterelemente, nur optionale Attribute, mit einem binären Wertebereich (true/false), siehe Tabelle 19:

Ebene	Element
1	`<adlseq:constrainedChoiceConsiderations>` **Attribute:** • `constrainedChoice` • `preventActivation`

Tabelle 19: Übersicht <constrainedChoice>

constrainedChoice

Dieses Attribut beschränkt die auszuwählenden Activities auf die, die relativ zur betroffenen Activity im Activity Tree davor oder dahinter liegen. Das soll verhindern, dass der Lernende zu weit im Inhalt "springt" und zu viele Activities auslässt. Abbildung 13 zeigt ein Anwendungsbeispiel:

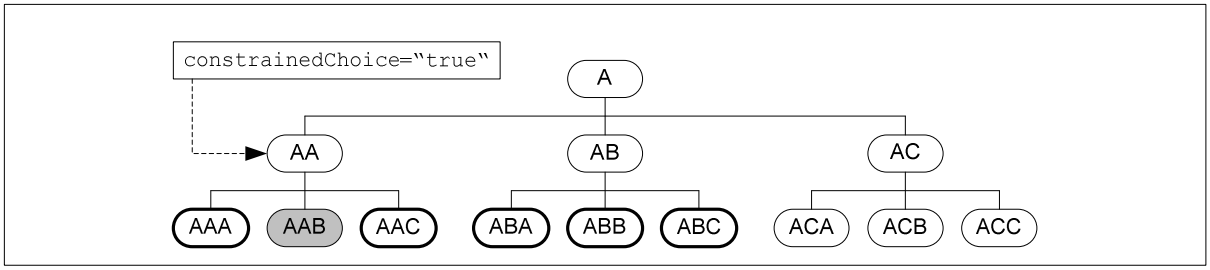


Abbildung 13: Anwendungsbeispiel constrainedChoice

Bearbeitet der Lernende gerade Activity AAB sind nur die dick umrandeten Activities auswählbar, die übrigen sind zu weit entfernt und von der Auswahl ausgeschlossen. Das LMS sollte von der Auswahl ausgeschlossene Activities nicht im Menü anzeigen.
Der default-Wert des Attributs ist "false".

preventActivation

Wenn das Attribut preventActivation den Wert "true" besitzt, wird erreicht, dass sich der Lernende im betroffenen Cluster "frei bewegen" kann (choice="true"), die Activities des Clusters stehen aber erst zur Auswahl, wenn die Cluster-Activity aktiv ist, das heißt, sie wurde z. B. mittels eines *Flow* Navigation Requests erreicht. Abbildung 14 zeigt ein Anwendungsbeispiel:

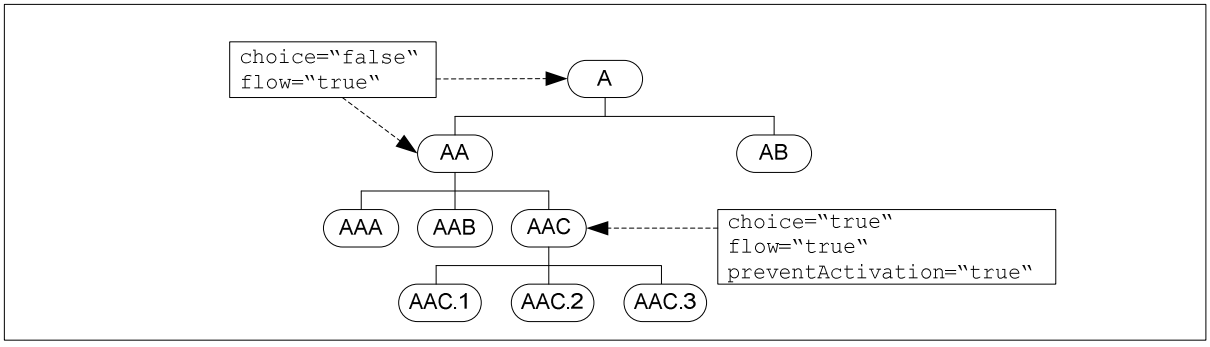


Abbildung 14: Anwendungsbeispiel preventActivation

Erst wenn die Activity AAC über Vorwärtsblättern erreicht wurde, erscheinen die Activities AAC.1 bis AAC.3 zur Auswahl im Menü.
Der default-Wert des Attributs ist "false".

3.3.3 Sequencing Rules

Für eine Activity können eine oder mehrere Sequencing Rules definiert werden, welche zu verschiedenen Zeitpunkten zur Laufzeit angewendet werden. Mit diesen Regeln kann festgelegt werden, wann Activities übersprungen werden, ob und wann sie im Menü angezeigt werden, wann sie verlassen werden uvm. Das entsprechende Element in der Manifest-Datei

ist <sequencingRules>, siehe Codebeispiel 9. In diesem Beispiel wird beim Ermitteln der nächsten bereitzustellenden Activity geprüft, ob die Anforderungen für diese Activity schon erfüllt sind und sie gegebenenfalls übersprungen werden kann.

```
<item identifier="WIEDERHOLUNG_01" identifierref="RES_LEKTION_01">
  <title>Lektion 1</title>
  <imsss:sequencing>
    <imsss:sequencingRules>
      <imsss:preConditionRule>
        <imsss:ruleConditions>
          <imsss:ruleCondition condition="satisfied" />
        </imsss:ruleConditions>
        <imsss:ruleAction action="skip" />
      </imsss:preConditionRule>
    </imsss:sequencingRules>
  </imsss:sequencing>
</item>
```

Codebeispiel 9: <sequencingRules>

Jede Sequencing Rule besteht aus einer oder mehreren Bedingungen und einer bestimmten Aktion. Die Bedingungen werden anhand der Tracking-Informationen der Activity (vgl. Kap. 3.2) geprüft und der gesamte Bedingungsteil der Regel (welcher aus mehreren einzelnen Bedingungen bestehen kann) enthält entweder den Wert "true" oder "false". Die angegebene Aktion wird ausgeführt, wenn der Bedingungsteil erfüllt ("true") ist. Eine Sequencing Rule kann also folgendermaßen zusammengefasst werden:

```
if [condition_set] then [action]
```

Entsprechend dem Zeitpunkt ihrer Prüfung gibt es drei Typen für Sequencing Rules:

- Pre-Condition-Rule: wird ausgewertet, wenn die nächste bereitzustellende Activity ermittelt wird (Element <preConditionRule>)
- Post-Condition-Rule: wird ausgewertet, wenn der Zugriff auf die Activity beendet wird (Element <postConditionRule>)
- Exit-Condition-Rule: wird ausgewertet, wenn der Zugriff auf eine untergeordnete Activity beendet wird (Element <exitConditonRule>)

Die folgende Tabelle gibt einen Überblick über die Unterelemente von <sequencingRules> mit den dazugehörigen Attributen:

Ebene	Element
1	<sequencingRules>
1.1	<preConditionRules>
1.1.1	<ruleConditions> **Attribute:** • conditionCombination o all o any
1.1.1.1	<ruleCondition> **Attribute:** • referencedObjective • measureThreshold • operator o noOp o not • condition o satisfied o objectiveStatusKnown o objectiveMeasureKnown o objectiveMeasureGreaterThan o objectiveMeasureLessThan o completed o activityProgressKnown o attempted o always o attemptLimitExceeded
1.1.2	<ruleAction> **Attribute:** • action o skip o disabled o hiddenFromChoice o stopForwardTraversal
1.2	<postConditionRules>
1.2.1	<ruleConditions> siehe 1.1.1 und Unterpunkte
1.2.2	<ruleAction> **Attribute:** • action o exitParent o exitAll o retry o retryAll o continue o previous
1.3	<exitConditionRules>
1.3.1	<ruleConditions> siehe 1.1.1 und Unterpunkte
1.3.2	<ruleAction> **Attribute:** • action o exit

Tabelle 20: Übersicht <sequencingRules>

Bedingungen

Das Element <ruleConditions> ist der Container für die Bedingungen einer Sequencing Rule, das Unterelement <ruleCondition> enthält jeweils eine einzelne Bedingung. Wie die

Bedingungen miteinander kombiniert werden, legt das Attribut `conditionCombination` im Element `<ruleConditions>` fest. In Tabelle 21 sind die möglichen Werte aufgeführt:

Wert	Beschreibung
`all`	default-Wert; der Bedingungsteil wird nur "true", wenn jede einzelne Bedingung "true" ist
`any`	der Bedingungsteil wird "true", wenn mindestens eine Bedingung "true" ist

Tabelle 21: conditionCombination im Element <ruleConditions>

Die Bedingung selbst wird mit dem Attribut `condition` im Element `<ruleCondition>` festgelegt, Tabelle 22 zeigt die verschiedenen möglichen Werte des Attributs:

Wert	Beschreibung
`satisfied`	Die Bedingung wird "true" gesetzt, wenn der *Objective Satisfied Status* des referenzierten Objectives "satisfied" ist (vgl. Kap. 3.2.1). Dies ist der Fall, wenn *cmi.objectives.n.success_status* des referenzierten Objectives bzw. *cmi.success_status* (Primary Objective) den Wert "passed" enthält.
`objectiveStatusKnown`	Die Bedingung wird "true" gesetzt, wenn der *Objective Satisfied Status* des referenzierten Objectives entweder "satisfied" oder "not satisfied" ist (vgl. Kap. 3.2.1). Dies ist der Fall, wenn *cmi.objectives.n.success_status* des referenzierten Objectives bzw. *cmi.success_status* (Primary Objective) entweder den Wert "passed" oder "failed" enthält.
`objectiveMeasureKnown`	Die Bedingung wird "true" gesetzt, wenn der *Objective Normalized Measure* des referenzierten Objectives einen Wert enthält (vgl. Kap. 3.2.1). Dies ist der Fall, wenn *cmi.objectives.n.score.scaled* bzw. *cmi.score.scaled* (Primary Objective) vom SCO gesetzt wurde oder der Wert über den Overall Rollup Process (vgl. Kap. 3.4) ermittelt wurde.
`objectiveMeasureGreaterThan`	Die Bedingung wird "true" gesetzt, wenn der *Objective Normalized Measure* (vgl. Kap. 3.2.1) des referenzierten Objectives größer ist als der Wert des Attributs `measureThreshold`. *Objective Normalized Measure* entspricht dem Wert von *cmi.objectives.n.score.scaled* bzw. *cmi.score.scaled* (Primary Objective).
`objectiveMeasureLessThan`	Die Bedingung wird "true" gesetzt, wenn *Objective Normalized Measure* kleiner ist als der Wert des Attributs `measureThreshold.`
`completed`	Die Bedingung wird "true" gesetzt, wenn der *Attempt Completion Status* der Activity "completed" ist (vgl. Kap. 3.2.3). Dies ist der Fall, wenn *cmi.completion_status* den Wert "completed" enthält oder der aktuelle Zugriff auf die Activity beendet wird.
`activityProgressKnown`	Die Bedingung wird "true" gesetzt, wenn der *Activity Progress Status* und der *Attempt Progress Status* der Activity "true" ist (vgl. Kap. 3.2.3). Dies ist der Fall, sobald beim aktuellen Zugriff auf die Activity irgendeine Tracking-Information registriert wird.
`attempted`	Die Bedingung ist "true", wenn der *Activity Attempt Count* größer 0 ist (vgl. Kap. 3.2.3). Dies ist der Fall, sobald das erste Mal auf die Activity zugegriffen wird (d.h. sobald sie das erste Mal vom LMS bereitgestellt wird).
`always`	Die Bedingung wird immer auf "true" gesetzt.
`attemptLimitExceeded`	Die Bedingung ist "true", sobald der *Activity Attempt Count* größer oder gleich dem Wert von `attemptLimit` (vgl. Kap. 3.3.4) ist.

Tabelle 22: Bedingungen für Sequencing Rules

Außerdem gibt es laut IMS Simple Sequencing Spezifikation noch die Bedingungen

timeLimitExceeded und outsideAvailableTimeRange. Sequencing Rules mit diesen beiden Bedingungen müssen von SCORM-konformen LMS nicht beachtet werden, da das SCORM Sequencing generell die Auswertung von zeitbasierten Tracking-Informationen nicht unterstützt.

Eine Bedingung kann zusätzlich mit folgenden optionalen Attributen versehen werden:

Attribut	Beschreibung
referencedObjective	Das Attribut wird nur bei bestimmten Bedingungen (welche sich auf ein Objective beziehen, die ersten fünf in Tabelle 22) angewendet. Es gibt an, welches Objective der Activity für die Prüfung der Bedingung genutzt wird. Als Wert wird die ID eines lokalen Objectives der Activity angegeben. Wird nichts angegeben, wird das Primary Objective genutzt.
measureThreshold	Das Attribut wird nur bei den Bedingungen objectiveMeasureGreaterThan und objectiveMeasureLessThan verwendet. Es beinhaltet einen Schwellenwert (Wertebereich -1 bis 1, default 0) für den Vergleich mit einem Punktwert des Objectives.
operator	Logischer Operator, welcher auf die Bedingung angewendet wird. Werte: • noOp – default, die Bedingung bleibt unverändert • not – die Bedingung wird negiert

Tabelle 23: Attribute für das Element <condition>

Aktionen

Das Element <ruleAction> enthält die Aktion, welche ausgeführt werden soll, wenn der Bedingungsteil "true" ist. Die Aktion wird mit dem Attribut action festgelegt und je nach Typ der Bedingung sind verschiedene Aktionen möglich:

Aktion	Beschreibung
Pre-Condition-Rule	
skip	Die Activity ist kein gültiges Ziel für einen *Previous* bzw. *Continue* Sequencing Request (vgl. Kap. 3.4), sie wird beim Ermitteln der nächsten bereitzustellenden Activity übersprungen.
disabled	Die Activity ist kein gültiges Ziel mehr für einen Sequencing Request.
hiddenFromChoice	Die Activity ist kein gültiges Ziel für einen *Choice* Navigation Request (vgl. Kap. 3.4). Durch diese Aktion wird erreicht, dass die Activity im Menü des LMS nicht angezeigt wird.
stopForwardTraversal	Nachfolgende Activities (entsprechend der Reihenfolge im Activity Tree) sind keine gültigen Ziele für einen Choice Navigation Request, für einen Continue Navigation Request sind sie aber weiterhin gültige Ziele.
Post-Condition-Rule	
exitParent	Das LMS beendet die übergeordnete Activity und wartet auf einen Navigation Request.
exitAll	Das LMS beendet alle Activities und wartet auf einen Navigation Request.
retry	Das LMS wiederholt die Activity (stellt sie noch einmal bereit).
retryAll	Das LMS beendet alle Activities und startet noch einmal mit der Wurzel-Activity.
continue	Das LMS stellt die nachfolgende Activity im Activity Tree bereit.
previous	Das LMS stellt die vorhergehende Activity im Activity Tree bereit.
Exit-Condition-Rule	
exit	Die Activity wird beendet.

Tabelle 24: Aktionen für Sequencing Rules

3.3.4 Limit Conditions

Der Content-Entwickler kann Bedingungen definieren, unter welchen eine Activity nicht bereitgestellt werden darf. Es gibt in der IMS-Spezifikation zeitbasierte Bedingungen und eine Beschränkung der Anzahl der Zugriffe auf eine Activity. Zeitbasierte Bedingungen werden von SCORM jedoch nicht unterstützt und sind daher auch nicht in der SCORM-Spezifikation aufgeführt.
Das Element für das Einfügen solcher Bedingungen ist `<limitConditions>`. Im Codebeispiel 10 ist die Anzahl der Zugriffe auf die Activity auf zwei beschränkt. Versucht der Lernende, ein drittes Mal auf die Activity zuzugreifen, wird sie nicht angezeigt bzw. wird sie schon vorher gar nicht mehr zur Auswahl bereitgestellt.

```
<item identifier="EINLEITUNG" identifierref="RES_EINLEITUNG">
  <title>Einleitung</title>
  <imsss:sequencing>
    <imsss:limitConditions attemptLimit="2" />
  </imsss:sequencing>
</item>
```

Codebeispiel 10: <limitConditions>

Das Element `<limitConditions>` enthält keine weiteren Unterelemente, Tabelle 25 zeigt eine Übersicht der Attribute:

Ebene	Element
1	`<limitConditions>` **Attribute:** • `attemptLimit` • `attemptAbsoluteDurationLimit`

Tabelle 25: Übersicht <limitConditions>

attemptLimit

Wird dieses Attribut benutzt, enthält es einen Zahlenwert, der angibt, wie oft diese Activity bereitgestellt werden darf. Wird solch eine Beschränkung nicht definiert, ist eine unbegrenzte Anzahl von Zugriffen möglich, sofern dies nicht z. B. durch Sequencing Rules (vgl. Kap. 3.3.3) verhindert wird.

attemptAbsoluteDurationLimit

Dieses Attribut bezeichnet die einzigste zeitbasierte Bedingung, die von SCORM-konformen LMS unterstützt werden muss. Dies ermöglicht, dass das Datenmodell-Element *cmi.max_time_allowed* mit dem Wert des Attributes initialisiert werden kann. Allerdings muss das LMS diesen Wert nicht weiter auswerten oder anwenden.
Als Wert des Attributes wird die Anzahl der Sekunden angegeben.

3.3.5 Rollup Rules

Cluster-Activities sind nicht mit einem Content Object verbunden, deshalb kann für solche Activities die Objective Progress Information und Attempt Progress Information (vgl. Kap. 3.2) nicht direkt gesetzt werden. Mit Rollup Rules kann abhängig von den Tracking-Informationen der Activities innerhalb des Clusters auch für die Cluster-Activity die entsprechende Tracking-Information ermittelt werden, sie wird sozusagen von den Kind-Activities zur Cluster-Activity "hochgerollt".

Jeder Cluster-Activity können eine oder mehrere Rollup Rules zugeordnet werden, welche während des *Overall Rollup Process* (vgl. Kap. 3.4) ausgewertet werden. Der *Overall Rollup Process* wird ausgelöst, wenn eine Activity beendet wird oder auch wenn sich der Status eines globalen Objectives ändert.

Innerhalb des Elements `<rollupRules>` werden die Rollup Rules für eine Activity definiert. Im Codebeispiel 11 wird durch eine Rollup Rule festgelegt, dass die Activity `TEST` als erfolgreich bearbeitet gilt, wenn jede ihrer Kind-Activities erfolgreich bearbeitet wurde.

```
<item identifier="TEST">
  <title>Test</title>
  <item identifier="AUFGABEN_01" identifierref="RES_AUFGABEN_01">
    <title>Aufgaben Lektion 1</title>
  </item>
  <item identifier="AUFGABEN_02" identifierref="RES_AUFGABEN_02">
    <title>Aufgaben Lektion 2</title>
  </item>
  <item identifier="AUFGABEN_03" identifierref="RES_AUFGABEN_03">
    <title>Aufgaben Lektion 3</title>
  </item>
  <imsss:sequencing>
    <imsss:rollupRules>
      <imsss:rollupRule childActivitySet="all">
        <imsss:rollupConditions>
          <imsss:rollupCondition condition="satisfied" />
        </imsss:rollupConditions>
        <imsss:rollupAction action="satisfied" />
      </imsss:rollupRule>
    </imsss:rollupRules>
  </imsss:sequencing>
</item>
```

Codebeispiel 11: <rollupRules>

Jede Rollup Rule besteht aus den zu berücksichtigenden Kind-Activities, einer oder mehreren Bedingungen und einer Aktion. Die Bedingungen werden anhand der Tracking-Informationen der einzubeziehenden Kind-Activities[3] geprüft, und der gesamte Bedingungsteil der Rollup Rule (welcher aus mehreren einzelnen Bedingungen bestehen kann) enthält entweder den Wert "true" oder "false". Die angegebene Aktion wird auf die Cluster-Activity angewendet,

[3] Grundsätzlich wird jede Kind-Activity in die Auswertung einbezogen. Durch verschiedene Elemente kann die Activity jedoch ausgeschlossen werden: z.B. `<deliveryControls tracked="false">`, `<rollupRules rollupObjectiveSatisfied="false" />`, `<rollupRules objectiveMeasureWeight="0.0" />` oder `<rollupRules rollupProgressCompletion="false" />`.

die die Rollup Rule enthält und wird nur ausgeführt, wenn der gesamte Bedingungsteil "true" ist. Eine Rollup Rule kann wie folgt zusammengefasst werden:

```
if [condition_set] TRUE for [child activity set] then [action]
```

Die folgende Tabelle gibt einen Überblick über die Unterelemente von `<rollupRules>` mit den dazugehörigen Attributen:

Ebene	Element
1	`<rollupRules>` **Attribute:** • `rollupObjectiveSatisfied` • `rollupProgressCompletion` • `objectiveMeasureWeight`
1.1	`<rollupRule>` **Attribute:** • `childActivitySet` o `all` o `any` o `none` o `atLeastCount` o `atLeastPercent` • `minimumCount` • `minimumPercent`
1.1.1	`<rollupConditions>` **Attribute:** • `conditionCombination` o `all` o `any`
1.1.1.1	`<rollupCondition>` **Attribute:** • `operator` o `noOp` o `not` • `condition` o `satisfied` o `objectiveStatusKnown` o `objectiveMeasureKnown` o `completed` o `activityProgressKnown` o `attempted` o `attemptLimitExceeded`
1.1.2	`<rollupAction>` **Attribute:** • `action` o `satisfied` o `notSatisfied` o `completed` o `incomplete`

Tabelle 26: Übersicht <rollupRules>

Das Element `<rollupRules>` schließt alle Rollup Rules für die Activity ein. Es enthält drei Attribute, die Rollup Controls. Mit diesen Rollup Controls kann der Content-Entwickler

festlegen, ob und wie (mit welcher Wichtung) eine Activity für das Rollup ihrer Cluster-Activity einbezogen wird. In Tabelle 27 werden diese Attribute näher erläutert:

Attribut	Beschreibung
`rollupObjectiveSatisfied`	Gibt an, ob der Status der Activity für die Auswertung der Rollup Rules (mit `action="satisfied"` bzw. `"notSatisfied"`) der Eltern-Activity einbezogen wird ("true", default) oder nicht ("false").
`rollupProgressCompletion`	Gibt an, ob der Status der Activity für die Auswertung der Rollup Rules (mit `action="completed"` bzw. `"incomplete"`) der Eltern-Activity einbezogen wird ("true", default) oder nicht ("false").
`objectiveMeasureWeight`	Der Ergebniswert für die Eltern-Activity ergibt sich aus dem Durchschnitt der Ergebniswerte der Kind-Activities. Mit diesem Attribut (Wertebereich 0 bis 1) kann eine Wichtung der einzelnen Ergebniswerte vorgenommen werden.

Tabelle 27: Rollup Controls

In Abbildung 15 ist der Einfluss der Rollup Controls beispielhaft dargestellt: ob Activity A (z. B. ein Kapitel) erfolgreich bearbeitet wurde, hängt davon ab, wie erfolgreich die Activities AA bis AD bearbeitet werden. Angenommen, AA, AB und AC sind die Lektionen des Kapitels und AD enthält einen Abschlusstest. Für AA, AB und AC kann nur ermittelt werden, ob sie vollständig bearbeitet wurden, aber nicht, ob sie erfolgreich bearbeitet wurden, da der Lernerfolg nur in Activity AD anhand des Tests überprüft wird. Also wird der Erfolgsstatus für Activity A nur anhand des Erfolgsstatus von Activity AD ermittelt, die anderen Activities werden nicht mit einbezogen (`rollupObjectiveSatisfied="false"`).

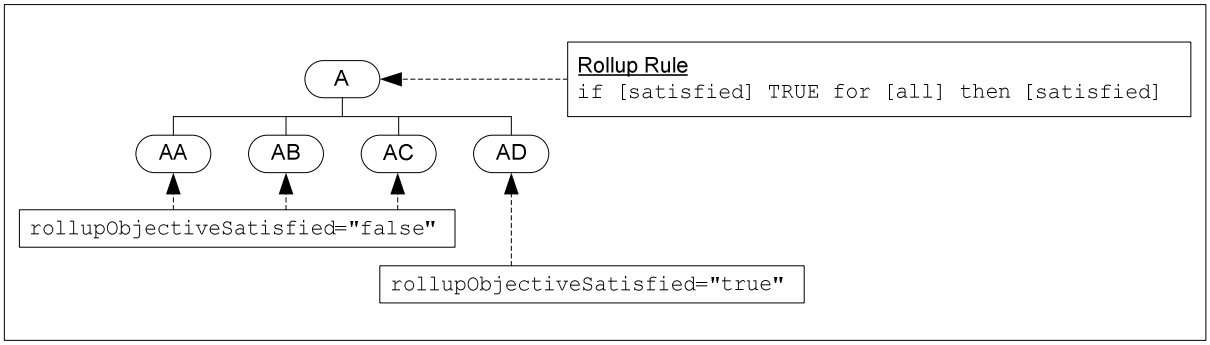


Abbildung 15: Anwendungsbeispiel Rollup Controls

Das Element `<rollupRule>` beinhaltet eine einzelne Rollup Rule und gibt durch das Attribut `childActivitySet` an, wie die in die Auswertung einbezogenen Kind-Activities zu berücksichtigen sind, ob z.B alle die Bedingung erfüllen müssen oder ob der Bedingungsteil bereits "true" wird, wenn eine Kind-Activity die Bedingung erfüllt. Die folgende Tabelle zeigt die möglichen Werte für `childActivitySet` und deren Bedeutung:

Wert	Beschreibung
all	default-Wert; wenn für alle einbezogenen Kind-Activities der Bedingungsteil "true" wird, wird die angegebene Aktion ausgeführt.
any	Wenn für mindestens eine der einbezogenen Kind-Activities der Bedingungsteil "true" wird, wird die angegebene Aktion ausgeführt.
none	Wenn für keine der einbezogenen Kind-Activities der Bedingungsteil "true" wird, wird die angegebene Aktion ausgeführt.
atLeastCount	Wenn für eine Mindestanzahl der einbezogenen Kind-Activities der Bedingungsteil "true" wird, wird die angegebene Aktion ausgeführt. Die Mindestanzahl wird mit dem Attribut minimumCount angegeben.
atLeastPercent	Wenn für eine Mindest-Prozentzahl der einbezogenen Kind-Activities der Bedingungsteil "true" wird, wird die angegebene Aktion ausgeführt. Die Mindest-Prozentzahl wird mit dem Attribut minimumPercent angegeben.

Tabelle 28: childActivitySet

Codebeispiel 12 macht die Verwendung von atLeastCount deutlich: wenn mindestens zwei der drei Aufgaben-Activities die Bedingung "satisfied" erfüllen, gilt auch die Activity TEST als erfolgreich bearbeitet.

```
<item identifier="TEST">
  <title>Test</title>
  <item identifier="AUFGABEN_01" identifierref="RES_AUFGABEN_01">
    <title>Aufgaben Lektion 1</title>
  </item>
  <item identifier="AUFGABEN_02" identifierref="RES_AUFGABEN_02">
    <title>Aufgaben Lektion 2</title>
  </item>
  <item identifier="AUFGABEN_03" identifierref="RES_AUFGABEN_03">
    <title>Aufgaben Lektion 3</title>
  </item>
  <imsss:sequencing>
    <imsss:rollupRules>
      <imsss:rollupRule childActivitySet="atLeastCount" minimumCount="2">
        <imsss:rollupConditions>
          <imsss:rollupCondition condition="satisfied" />
        </imsss:rollupConditions>
        <imsss:rollupAction action="satisfied" />
      </imsss:rollupRule>
    </imsss:rollupRules>
  </imsss:sequencing>
</item>
```

Codebeispiel 12: Anwendungsbeispiel für atLeastCount

Bedingungen

Das Element <rollupConditions> ist der Container für die Bedingungen einer Rollup Rule, das Unterelement <rollupCondition> enthält jeweils eine einzelne Bedingung. Wie die Bedingungen miteinander kombiniert werden, legt das Attribut conditionCombination im Element <rollupConditions> fest. In Tabelle 29 sind die möglichen Werte aufgeführt:

Wert	Beschreibung
all	der Bedingungsteil wird nur "true", wenn jede einzelne Bedingung "true" ist
any	default-Wert; der Bedingungsteil wird "true", wenn mindestens eine Bedingung "true" ist

Tabelle 29: conditionCombination im Element <rollupConditions>

Die Bedingung selbst wird mit dem Attribut condition im Element <rollupCondition> festgelegt. Die möglichen Bedingungen sind bereits in Tabelle 26 aufgeführt und sollen hier

nicht näher erläutert werden, da sie den gleichnamigen Bedingungen der Sequencing Rules entsprechen (siehe Tabelle 22). Allerdings beziehen sich die Bedingungen, die anhand von Objectives überprüft werden (`satisfied`, `objectiveStatusKnown` und `objectiveMeasureKnown`) immer auf das Primary Objective, es kann kein anderes Objective referenziert werden wie bei den Sequencing Rules.
Jede Bedingung kann mit einem Operator versehen werden. Das entsprechende Attribut `operator` wurde bereits in Tabelle 23 beschrieben.

Aktionen

Das Element `<rollupAction>` enthält die Aktion, welche ausgeführt werden soll, wenn der Bedingungsteil "true" ist. Die Aktion wird mit dem Attribut `action` festgelegt. Folgende Aktionen sind möglich:

Wert	Beschreibung
`satisfied`	Der *Objective Satisfied Status* des Primary Objective der Activity wird "satisfied".
`notSatisfied`	Der *Objective Satisfied Status* des Primary Objective der Activity wird "not satisfied".
`completed`	Der *Attempt Completion Status* der Activity wird "completed".
`incomplete`	Der *Attempt Completion Status* der Activity wird "incomplete".

Tabelle 30: Aktionen für Rollup Rules

Sind keine Rollup Rules angegeben, wird innerhalb der Subprozesse des *Overall Rollup Process* ein Rollup nach default-Regeln durchgeführt.

3.3.6 Rollup Consideration Controls

Grundsätzlich sind alle Kind-Activities in das Rollup der Cluster-Activity einbezogen, es sei denn

- die Activity wird nicht registriert (`tracked="false"`, vgl. Kap. 3.3.9)
- oder sie ist durch Rollup Controls vom Rollup ausgeschlossen, vgl. Kap. 3.3.5

Wenn eine Kind-Activity in das Rollup einbezogen ist, die auszuwertende Tracking-Information (*Attempt Completion Status* und *Objective Satisfied Status*) hat aber den Status "unknown", dann kann auch meist nur der Status "unknown" zur Cluster-Activity übertragen werden. Dies ist im Beispiel in Abbildung 16 der Fall: wenn z. B. Activity AAB übersprungen wird, wird kein Bearbeitungsstatus für diese Activity registriert, der Status ist unbekannt und kann somit auch nicht für die Cluster-Activity definiert werden. Die Cluster-Activity ist aber im Prinzip vollständig bearbeitet, wenn AAA und AAC bearbeitet wurden.

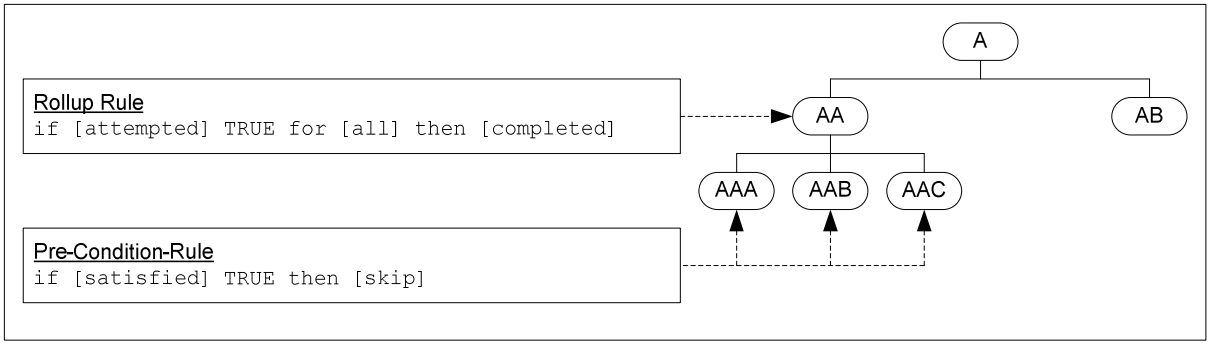


Abbildung 16: Notwendigkeit von Rollup Consideration Controls

Übersprungene und ausgeblendete oder auch gerade unterbrochene Activities sollten also in manchen Fällen für die Ermittlung des Bearbeitungs- und Erfolgsstatus nicht mit einbezogen werden. Um dies explizit festzulegen, hat ADL die Rollup Consideration Controls eingeführt, welche anhand des Elements `<rollupConsiderations>` der jeweiligen Activity zugeordnet werden können. Im Codebeispiel 13 soll die Activity `WIEDERHOLUNG_01` übersprungen werden, falls die Voraussetzungen dafür erfüllt sind. Der Bearbeitungsstatus dieser Activity soll nur dann in das Rollup mit einbezogen werden, wenn sie nicht übersprungen wurde.

```
<item identifier="WIEDERHOLUNG_01" identifierref="RES_LEKTION_01">
  <title>Wiederholung Lektion 1</title>
  <imsss:sequencing>
    <imsss:sequencingRules>
      <imsss:preConditionRule>
        <imsss:ruleConditions>
          <imsss:ruleCondition condition="satisfied" />
        </imsss:ruleConditions>
        <imsss:ruleAction action="skip" />
      </imsss:preConditionRule>
    </imsss:sequencingRules>
    <adlseq:rollupConsiderations requiredForCompleted="ifNotSkipped" />
  </imsss:sequencing>
</item>
```

Codebeispiel 13: <rollupConsiderations>

Das Element `<rollupConsiderations>` kann mit verschiedenen Attributen versehen werden, deren Wert die Bedingung enthält, wann die Tracking-Information in das Rollup einbezogen wird. Tabelle 31 gibt einen Überblick über das Element und seine möglichen Attribute:

Ebene	Element

1	`<rollupConsiderations>`
	Attribute:
	• `requiredForSatisfied`
	• `requiredForNotSatisfied`
	• `requiredForCompleted`
	• `requiredForIncomplete`
	  o `always`
	  o `ifNotSuspended`
	  o `ifNotSkipped`
	  o `ifAttempted` (diese Werte sind für jedes der vier Attribute möglich)
	• `measureSatisfactionIfActive`

Tabelle 31: Übersicht <rollupConsiderations>

In der folgenden tabellarischen Übersicht werden die Attribute näher erläutert:

Attribut	Beschreibung
`requiredForSatisfied`	Die Activity wird zur Festlegung des *Objective Satisfied Status* "satisfied" für die Cluster-Activity nur einbezogen, wenn die angegebene Bedingung erfüllt ist.
`requiredForNotSatisfied`	Die Activity wird zur Festlegung des *Objective Satisfied Status* "not satisfied" für die Cluster-Activity nur einbezogen, wenn die angegebene Bedingung erfüllt ist.
`requiredForCompleted`	Die Activity wird zur Festlegung des *Attempt Completion Status* "completed" nur einbezogen, wenn die angegebene Bedingung erfüllt ist.
`requiredForIncomplete`	Die Activity wird zur Festlegung des *Attempt Completion Status* "incomplete" nur einbezogen, wenn die angegebene Bedingung erfüllt ist.
`measureSatisfactionIfActive`	Wird der *Objective Satisfied Status* der Activity über einen Ergebniswert ermittelt (Vergleich von *Objective Normalized Measure* mit *Objective Minimum Normalized Measure*), kann mit diesem Attribut festgelegt werden, ob der Vergleich stattfindet, wenn sie noch aktiv ist ("true", default) oder erst, wenn der Zugriff auf die Activity beendet wurde ("false").

Tabelle 32: Attribute des Elements <rollupConsiderations>

Folgende Bedingungen können angegeben werden:

Wert	Beschreibung
`always`	default-Wert; der entsprechende Status der Kind-Activity wird immer in die Auswertung einbezogen.
`ifNotSuspended`	Der entsprechende Status der Kind-Activity wird nur dann in die Auswertung einbezogen, wenn die Activity zum Zeitpunkt der Auswertung nicht unterbrochen ist.
`ifNotSkipped`	Der entsprechende Status der Kind-Activity wird nur dann in die Auswertung einbezogen, wenn die Activity nicht übersprungen wurde.
`ifAttempted`	Der entsprechende Status der Kind-Activity wird nur dann in die Auswertung einbezogen, wenn auf die Activity zum Zeitpunkt der Auswertung bereits zugegriffen wurde.

Tabelle 33: Bedingungen für Roullup Consideration Controls

Die Rollup Consideration Controls haben keinen Einfluss auf Activities, welche nicht in das Rollup einbezogen sind.

3.3.7 Objectives

Wie schon im Überblick zum Sequencing beschrieben, kann mit Hilfe von Objectives der Lernerfolg des Lernenden festgehalten werden. Dafür müssen die Objectives den Activities zugeordnet werden. Dies geschieht mit dem Element `<objectives>`. Im Codebeispiel 14 enthält die Activity `AUFGABEN_01` nur das Primary Objective und keine weiteren lokalen Objectives. Das Primary Objective nutzt das globale Objective `obj_lektion01` und bildet die eigenen Statuswerte auf dieses ab. Später wird von der Activity `WIEDERHOLUNG_01` das globale Objective ausgelesen, um zu entscheiden, ob die mit der Activity verbundene Lernressource dem Lernenden präsentiert wird oder nicht.

```
<item identifier="AUFGABEN_01"identifierref="RES_AUFGABEN_01">
  <title>Aufgaben Lektion 1</title>
  <imsss:sequencing>
    <imsss:objectives>
      <imsss:primaryObjective objectiveID="PRIMARY">
        <imsss:mapInfo targetObjectiveID="obj_lektion01"
           readSatisfiedStatus="false" readNormalizedMeasure="false"
           writeSatisfiedStatus="true" writeNormalizedMeasure="true" />
      </imsss:primaryObjective>
    </imsss:objectives>
  </imsss:sequencing>
</item>
.
.
.
<item identifier="WIEDERHOLUNG_01"identifierref="RES_LEKTION_01">
  <title>Wiederholung Lektion 1</title>
  <imsss:sequencing>
    <imsss:objectives>
      <imsss:primaryObjective objectiveID="PRIMARY">
        <imsss:mapInfo targetObjectiveID="obj_lektion01"
           readSatisfiedStatus="true" readNormalizedMeasure="true"
           writeSatisfiedStatus="false" writeNormalizedMeasure="false" />
      </imsss:primaryObjective>
    </imsss:objectives>
    <imsss:sequencingRules>
      <imsss:preConditionRule>
        <imsss:ruleConditions>
          <imsss:ruleCondition condition="satisfied" />
        </imsss:ruleConditions>
        <imsss:ruleAction action="skip" />
      </imsss:preConditionRule>
    </imsss:sequencingRules>
  </imsss:sequencing>
</item>
```

Codebeispiel 14: <objectives>

In Tabelle 34 sind zur Übersicht alle Unterelemente und Attribute innerhalb des Elementes `<objectives>` aufgeführt:

Ebene	Element
1	`<objectives>`
1.1	`<primaryObjective>` **Attribute:** • `satisfiedByMeasure` • `objectiveID`
1.1.1	`<minNormalizedMeasure>`
1.1.2	`<mapInfo>` **Attribute:** • `targetObjectiveID` • `readSatisfiedStatus` • `readNormalizedMeasure` • `writeSatisfiedStatus` • `writeNormalizedMeasure`
1.2	`<objective>` **Attribute:** • `satisfiedByMeasure` • `objectiveID`
1.2.1	`<minNormalizedMeasure>`
1.2.2	`<mapInfo>` siehe 1.1.2

Tabelle 34: Übersicht <objectives>

Jeder Activity ist automatisch ein Primary Objective zugeordnet und es können beliebig viele lokale Objectives festgelegt werden. Das Primary Objective wird mit dem Element

<primaryObjective> beschrieben, die lokalen Objectives mit dem Element <objective>. Sobald für eine Activity Objectives definiert werden, ist das Element <primaryObjective> obligatorisch, es kann allerdings leer bleiben (<primaryObjective />), solange es kein Element <mapInfo> enthält. Beide Elemente enthalten zwei Attribute, siehe Tabelle 35.

Attribut	Beschreibung
objectiveID	Dieses Attribut enthält eine ID für das Objective, welche innerhalb der Activity eindeutig sein muss. Diese ID wird genutzt, um *cmi.objectives.n.id* zu initialisieren.
satisfiedByMeasure	Dieses Attribut gibt an, ob für die Ermittlung des *Objective Satisfied Status* der Inhalt des Elements <minNormalizedMeasure> mit *Objective Normalized Measure* verglichen wird ("true") oder ob der *Objective Satisfied Status* anders ermittelt wird ("false", default).

Tabelle 35: Attribute für <primaryObjective> und <objective>

Codebeispiel 15 zeigt die Anwendung des Attributes satisfiedByMeasure. Der *Objective Satisfied Status* des Primary Objective der Activity TEST wird über den Vergleich des *Objective Normalized Measure* mit einem angegebenen Schwellenwert ermittelt (satisfiedByMeasure="true"). Der Schwellenwert für den Vergleich wird mit dem Element <minNormalizedMeasure> in der Manifest-Datei angegeben, in einem Wertebereich von –1 bis 1. Der Ergebniswert des Primary Objective der Activity TEST ist der Durchschnitt der *Objective Normalized Measure* der Aufgaben-Activities.

```
<item identifier="TEST "identifierref="RES_TEST">
  <title>Abschlußtest</title>
  <item identifier="AUFGABEN_01" identifierref="RES_AUFGABEN_01">
    <title>Aufgaben Lektion 1</title>
  </item>
  <item identifier="AUFGABEN_02" identifierref="RES_AUFGABEN_02">
    <title>Aufgaben Lektion 2</title>
  </item>
  <item identifier="AUFGABEN_03" identifierref="RES_AUFGABEN_03">
    <title>Aufgaben Lektion 3</title>
  </item>
  <imsss:sequencing>
    <imsss:objectives>
      <imsss:primaryObjective objectiveID="PRIMARY" satisfiedByMeasure="true">
        <imsss:minNormalizedMeasure>0.6</minNormalizedMeasure>
      </imsss:primaryObjective>
    </imsss:objectives>
  </imsss:sequencing>
</item>
```

Codebeispiel 15: Anwendungsbeispiel für satisfiedByMeasure

Lokale Objectives innerhalb einer Activity können ihre Werte auf globale Objectives abbilden und umgekehrt. Das Element `<mapInfo>` legt fest, wie die Werte abgebildet werden können. Dies geschieht über die Attribute, welche in Tabelle 36 erläutert werden:

Attribut	Beschreibung
`targetObjectiveID`	Enthält die ID des globalen Objectives.
`readSatisfiedStatus`	Gibt an, ob der *Objective Satisfied Status* des globalen Objectives auf das lokale übertragen wird ("true", default) oder nicht ("false").
`readNormalizedMeasure`	Gibt an, ob der *Objective Normalized Measure* des globalen Objectives auf das lokale übertragen wird ("true", default) oder nicht ("false").
`writeSatisfiedStatus`	Gibt an, ob der *Objective Satisfied Status* des lokalen Objectives auf das globale übertragen wird ("true") oder nicht ("false", default).
`writeNormalizedMeasure`	Gibt an, ob der *Objective Normalized Measure* des lokalen Objectives auf das globale übertragen wird ("true") oder nicht ("false", default).

Tabelle 36: Attribute für <mapInfo>

3.3.8 Selection Controls und Randomization Controls

Mit diesen Sequencing-Eementen kann der Content-Entwickler festlegen, ob und wie die Liste der *Available Children* (vgl. Kap. 3.2.4) zur Laufzeit noch verändert werden kann. Die Anzahl der auswählbaren Activities kann eingeschränkt werden und die Reihenfolge kann verändert werden. Diese Sequencing-Informationen werden im Element `<randomizationControls>` definiert. Im Codebeispiel 16 werden bei jedem neuen Zugriff auf die Activity `KAPITEL_01` zwei der drei Child-Activities ausgewählt, die im Menü angezeigt werden und die Reihenfolge wird per Zufall bestimmt.

```
<item identifier="KAPITEL_01">
  <title>Kapitel 1</title>
  <item identifier="LEKTION_01" identifierref="RES_LEKTION_01">
    <title>Lektion 1</title>
  </item>
  <item identifier="LEKTION_02" identifierref="RES_LEKTION_02">
    <title>Lektion 2</title>
  </item>
  <item identifier="LEKTION_03" identifierref="RES_LEKTION_03">
    <title>Lektion 3</title>
  </item>
  <imsss:sequencing>
    <imsss:controlMode choice="true" choiceExit="true" flow="true" />
    <imsss:randomizationControls selectCount="2" selectionTiming="onEachNewAttempt"
          randomizationTiming="onEachNewAttempt" reorderChildren="true"/>
  </imsss:sequencing>
</item>
```

Codebeispiel 16: <randomizationControls>

Das Element `<randomizationControls>` enthält vier Attribute und keine weiteren Unterelemente, siehe Tabelle 37:

Ebene	Element
1	<randomizationControls> **Attribute:** • selectionCount • reorderChildren • selectionTiming o never o once o onEachNewAttempt • randomizationTiming o never o once o onEachNewAttempt

Tabelle 37: Übersicht <randomizationControls>

In der folgenden Tabelle werden die Attribute mit den möglichen Werten und ihre Verwendung erläutert:

Attribut	Beschreibung
selectionCount	Enthält einen Zahlenwert, welcher festlegt, wie viele der Child-Activities in *Available Children* aufgenommen wird. Die Activities werden zufällig ausgewählt, die relative Reihenfolge der Activities bleibt dabei gleich. Ist der Wert gleich oder größer der Anzahl der tatsächlichen Child-Activities, werden alle ausgewählt.
reorderChildren	Legt fest ob die Reihenfolge der Child-Activities entsprechend randomizationTiming neu ermittelt wird ("true") oder nicht ("false", default):
selectionTiming	Diese Attribut legt fest, wann selectionCount angewendet wird: • never – default, selectionCount wird nie angewendet, alle Child-Activities sind in Available Children enthalten • once – selectionCount wird vor dem ersten Zugriff auf die Activity angewendet • onEachNewAttempt – selectionCount wird vor jedem Zugriff auf die Activity angewendet
randomizationTiming	Diese Attribut legt fest, wann die Child-Activities neu geordnet werden: • never – default, die Child-Activities bleiben immer in der festgelegten Reihenfolge • once – die Reihenfolge der Child-Activities wird vor dem ersten Zugriff auf die Activity neu ermittelt • onEachNewAttempt – die Reihenfolge der Child-Activities wird vor jedem Zugriff auf die Activity neu ermittelt

Tabelle 38: Attribute des Elements <randomizationControls>

Mit den Selection Controls soll dem Content-Entwickler die Möglichkeit gegeben werden, dem Lernenden nur eine Teilmenge der Child-Activities eines Clusters zu präsentieren. Dies ist z. B. dann sinnvoll, wenn aus einer Aufgabensammlung für einen Abschlusstest zufällig eine bestimmte Anzahl ausgewählt werden soll, so dass jeder Lernende unterschiedliche Aufgaben bearbeitet.

3.3.9 Delivery Controls

Mit Delivery Controls kann die Erfassung der Tracking-Informationen im Tracking Model beeinflusst werden. Der Content Entwickler kann festlegen, ob der Zugriff auf und die Interaktion mit einer Activity vom LMS registriert wird und er kann explizit festlegen, ob der *Attempt Completion Status* und der *Objective Satisfied Status* nur über die Werte der entsprechenden RTE-Datenelemente ermittelt werden darf, dass das LMS also nicht "selbständig" den Status auf "completed" bzw. "satisfied" setzen darf[4].
Delivery Controls werden mit dem Element `<deliveryControls>` definiert. In Codebeispiel 17 wird festgelegt, dass für die Activities, die die Lektionen enthalten, keine Tracking-Informationen erfasst werden.

```
<item identifier="KAPITEL_01">
  <title>Kapitel 1</title>
  <item identifier="LEKTION_01" identifierref="RES_LEKTION_01">
    <title>Lektion 1</title>
    <imsss:sequencing>
      <imsss:deliveryControls tracked = "false"/>
    </imsss:sequencing>
  </item>
  <item identifier="LEKTION_02" identifierref="RES_LEKTION_02">
    <title>Lektion 2</title>
    <imsss:sequencing>
      <imsss:deliveryControls tracked = "false"/>
    </imsss:sequencing>
  </item>
  <item identifier="LEKTION_03" identifierref="RES_LEKTION_03">
    <title>Lektion 3</title>
    <imsss:sequencing>
      <imsss:deliveryControls tracked = "false"/>
    </imsss:sequencing>
  </item>
</item>
```

Codebeispiel 17: <deliveryControls>

Das Element `<deliveryControls>` enthält drei Attribute und keine weiteren Unterelemente, siehe Tabelle 39:

Ebene	Element
1	`<deliveryControls>` **Attribute:** • `tracked` • `completionSetByContent` • `objectiveSetByContent`

Tabelle 39: Übersicht <deliveryControls>

[4] Das LMS setzt den *Attempt Completion Status* auf „completed" bzw. den *Objective Satisfied Status* des Primary Objective auf "satisfied", wenn der Zugriff auf die Activity beendet wird und bis dahin der Wert des entsprechenden Elements des RTE-Datenmodells vom SCO nicht gesetzt wurde.

tracked

Dieses Attribut legt fest, ob die Tracking-Informationen der Activity[5] erfasst und für das Rollup zur Cluster-Activity genutzt werden ("true", default). Ist `tracked="false"`, wird die Activity nicht in die Auswertung von Rollup Rules einbezogen (vgl. auch Kap. 3.3.5) und für andere Prozesse (z. B. Prüfung der Bedingungen der Sequencing Rules), für die diese Tracking-Informationen erforderlich sind, wird der default-Wert "unknown" benutzt.

completionSetByContent

Wenn vom Content Object *cmi.completion_status* nicht gesetzt wird, nimmt das LMS an, dass die Activity vollständig bearbeitet ist, wenn der Zugriff beendet wird. Es setzt den *Attempt Completion Status* der Activity auf "completed". Dies ermöglicht, dass auch Assets in die Sequencingstrategie mit einbezogen werden können, obwohl sie nicht mit dem LMS kommunizieren können. Mit `completionSetByContent="true"` wird diese "Annahme" verhindert, der *Attempt Completion Status* bleibt in diesem Fall "unknown", auch wenn der Zugriff beendet wird. Wird `completionSetByContent="true"` also bei einer Activity, die ein Asset referenziert, verwendet, bleibt der *Attempt Completion Status* immer "unknown". Der default-Wert dieses Attributs ist "false". Allerdings werden die vom SCO übertragenen Werte auch in diesem Fall genutzt und nicht ignoriert oder geändert.
`completionSetByContent="true"` wäre also nur sinnvoll, wenn unbedingt verhindert werden soll, dass eine Activity, die ein SCO referenziert, als "completed" gilt, wenn das SCO diesen Wert nicht setzt.
Das Attribut wird nur auf Blatt-Activities angewendet.

objectiveSetByContent

Wenn vom Content Object *cmi.succes_status* nicht gesetzt wird, nimmt das LMS an, dass das Primary Objective der Activity erfolgreich absolviert ist, wenn der Zugriff beendet wird. Es setzt den *Objective Satisfied Status* des Primary Objective auf "satisfied". So können auch Assets in die Sequencingstrategie einbezogen werden. Mit `objectiveSetByContent ="true"` wird diese "Annahme" verhindert, der *Objective Satisfied Status* des Primary Objective bleibt in diesem Fall "unknown", auch wenn der Zugriff beendet wird. Wenn also `objectiveSetByContent ="true"` bei einer Activity verwendet wird, die ein Asset referenziert, bleibt der *Objective Satisfied Status* des Primary Objective immer "unknown".

[5] Das betrifft die Objective Progress Information, die Activity Progress Information und die Attempt Progress Information.

Der default-Wert dieses Attributs ist "false". Allerdings werden die vom SCO übertragenen Werte auch in diesem Fall genutzt und nicht ignoriert oder geändert.
`objectiveSetByContent="true"` wäre also nur sinnvoll, wenn unbedingt verhindert werden soll, dass das Primary Objective einer Activity, die ein SCO referenziert, als "satisfied" gilt, wenn das SCO diesen Wert nicht selbst setzt.
Auch dieses Attribut wird nur auf Blatt-Activities angewendet.

Wie aus der Vielzahl der Elemente ersichtlich ist, sind die Möglichkeiten für den Content-Entwickler, eine Sequencingstrategie in das Content Package zu integrieren, sehr umfangreich. Control Modes sowie Sequencing Rules und Rollup Rules in Verbindung mit Objectives sind sicher die am häufigsten verwendeten Elemente, da sie den größten Einfluss auf die Ablaufsteuerung haben.

Oft wird es vorkommen, dass immer wieder die gleichen Sequencinginformationen für bestimmte Activities verwendet werden. Diese Sequencinginformationen können innerhalb des Elements `<sequencingCollection>` (direktes Unterelement von `<manifest>`) einmal erstellt werden und dann beliebig oft wiederverwendet werden. `<sequencingCollection>` wird mit einer ID versehen und kann dann von jedem `<sequencing>`-Element über das Attribut `IDRef` referenziert werden, siehe Codebeispiel 18:

```
<manifest version="1.3">
  <organizations default="ORG01">
    <organization identifier="ORG01">
      <!-- other manifest data -->
      <item identifier="WIEDERHOLUNG_01" identifierref="RES_LEKTION_01">
        <title>Wiederholung Lektion 1</title>
        <imsss:sequencing IDRef="WIEDERHOLUNG">
          <imsss:controlMode choiceExit="false" />
        </imsss:sequencing>
      </item>
      <item identifier="WIEDERHOLUNG_02" identifierref="RES_LEKTION_02">
        <title>Wiederholung Lektion 2</title>
        <imsss:sequencing IDRef="WIEDERHOLUNG" />
      </item>
      <item identifier="WIEDERHOLUNG_03" identifierref="RES_LEKTION_03">
        <title>Wiederholung Lektion 3</title>
        <imsss:sequencing IDRef="WIEDERHOLUNG" />
      </item>
    </organization>
  </organizations>
  <!-- other manifest data -->
  <imsss:sequencingCollection>
    <imsss:sequencing ID="WIEDERHOLUNG">
      <imsss:controlMode choiceExit="false">
    </imsss:sequencing>
  </imsss:sequencingCollection>
</manifest>
```

Codebeispiel 18: <sequencingCollection>

Es können weiterhin Sequencinginformationen innerhalb des `<sequencing>`-Elements der Activity angegeben werden. Diese Sequencinginformationen überschreiben die referenzierten, falls dort die gleichen Elemente enthalten sind.

3.4 Overall Sequencing Process

Der *Overall Sequencing Process* ist der allumfassende Steuerungsprozess für das Sequencing. Er beinhaltet viele verschiedene Sequencingprozesse, welche alle mit den Elementen des Tracking Models und des Sequencing Definition Models agieren. SCORM beschreibt diese Prozesse mit Pseudocodes (siehe [SCORM, 2004d], Anlage C), welche vom Entwickler eines LMS umgesetzt werden müssen. Es gibt keine näheren Anforderungen an die Implementierung, es muss nur exakt das Verhalten aus dem Pseudocode umgesetzt werden. Da diese Arbeit ihren Fokus auf der Erstellung von Inhalten, die das Sequencing nutzen, hat und nicht auf der Umsetzung des Sequencing im LMS, soll der *Overall Sequencing Process* hier nur kurz erläutert werden.

Der *Overall Sequencing Process* findet während der Sequencing Session statt. Eine Sequencing Session startet mit der Auswahl einer Content Organization (d.h. eines Kurses). Das LMS sendet einen einleitenden Navigation Request und damit wird der *Overall Sequencing Process* gestartet und befindet sich dann in einem "Sequencing Loop". Der *Overall Sequencing Process* wird durchlaufen und wartet dann auf den nächsten Navigation Request. In Abbildung 17 ist der Ablauf des *Overall Sequencing Process* innerhalb der Sequencing Session dargestellt:

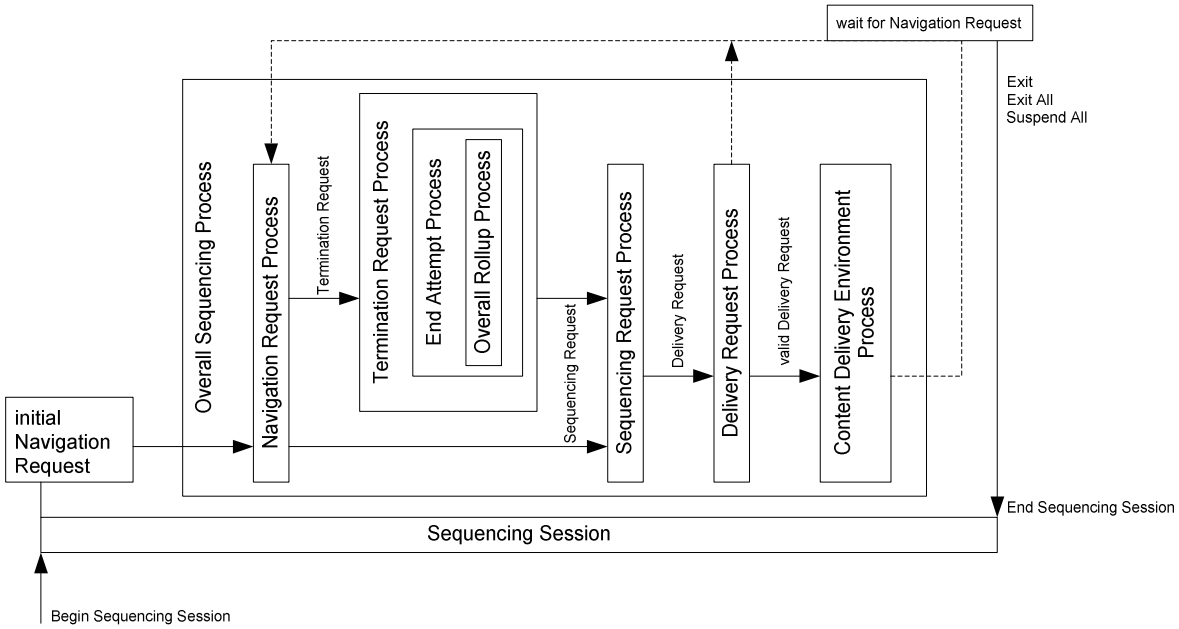


Abbildung 17: Overall Sequencing Process

Der Navigation Request (z. B. *Start*, *Continue*, *Choice*, *Exit All*, vgl. Tabelle 1 im Anhang C) wird im *Navigation Request Process* verarbeitet. Vom *Navigation Request Process* wird ggf. ein Termination Request (z. B. *Suspend All*, vgl. Tabelle 2 im Anhang C) zurückgegeben (wenn der Zugriff auf eine Activity beendet werden muss, beim einleitenden Navigation Request des LMS wird z. B. kein Termination Request weitergegeben) und im *Termination Request Process* verarbeitet. Außerdem wird der Navigation Request in einen entsprechenden Sequencing Request übersetzt und der *Sequencing Request Process* wird gestartet. Basierend auf dem Sequencing Request (z. B. *Start*, *Continue*, *Choice*, vgl. Tabelle 3 im Anhang C) und den Informationen im Tracking Model und Sequencing Definition Model wird die nächste bereitzustellende Activity ermittelt und mit einem Delivery Request an den *Delivery Request Process* weitergegeben. Kann keine nächste Activity ermittelt werden, wird der *Overall Sequencing Process* unterbrochen und wartet auf einen neuen Navigation Request. Der *Delivery Request Process* prüft, ob die Activity bereitgestellt werden kann (abhängig von Sequencing Rules, Limit Conditions etc.). Kann die Activity nicht bereitgestellt werden, wird der *Overall Sequencing Process* unterbrochen und wartet auf einen neuen Navigation Request. Der gültige Delivery Request wird an den *Content Delivery Environment Process* weitergegeben, das mit der Activity verbundene Content Object wird gestartet und die Tracking-Informationen für den neuen Zugriff werden initialisiert. Danach wird der *Content Delivery Environment Process* wird beendet, der Lernende bearbeitet das Content Object und der *Overall Sequencing Process* wartet auf den nächsten Navigation Request. Lautet der Navigation Request entsprechend (*Exit*, *Exit All*, *Suspend All*), wird der "Sequencing Loop" verlassen und die Sequencing Session beendet.

Vom *Termination Request Process* werden weitere wichtige Prozesse aufgerufen, zum einen der *End Attempt Process*, welcher den *Attempt Completion Status* und den *Objective Satisfied Status* aktualisiert und den Zugriff beendet, zum anderen Prozesse, welche Post-Condition-Rules und Exit-Condition-Rules auswerten. Der *End Attempt Process* löst außerdem den *Overall Rollup Process* aus, welcher entsprechend der Rollup Rules die Tracking-Informationen der Cluster-Activities ermittelt.
Sämtliche Sequencingprozesse können auch außerhalb des *Overall Sequencing Process* vom LMS aufgerufen werden. So sind z. B. der *Select Children Process* und der *Randomization Children Process* nicht in den *Overall Sequencing Process* eingebettet, das LMS muss diese Prozesse aufrufen, sobald es erforderlich ist. SCORM macht zu diesen Aufrufen keine weiteren Angaben.

Detaillierte Erläuterungen zu den einzelnen Prozessen findet man im SCORM Sequencing and Navigation Book [SCORM, 2004d] sowie in der IMS Simple Sequencing Spezifikation [IMS, 2003].

3.5 Sequencing in SCORM 1.2

In SCORM 1.2 gab es auch Möglichkeiten, die Ablaufsteuerung der Lerninhalte zu beeinflussen. Allerdings waren diese weniger umfangreich. An jedes SCO konnten "Prerequisites" gebunden werden, das sind Vorbedingungen, die erfüllt sein müssen, bevor auf ein SCO zugegriffen werden kann. Dafür gab es das Element `<adlcp:prerequisites>`, welches dem `<item>`-Element untergeordnet war. Codebeispiel 19 zeigt ein Beispiel für die Verwendung. Für `LESSON_3` gilt, dass sie erst aufgerufen werden kann, wenn `LESSON_1` und `LESSON_2` bearbeitet wurden.

```
<item identifier="MODULE_1">
  <item identifier="LESSON_1" identifierref="RES_LESSON_1">
    <titel>Lesson 1</titel>
  </item>
  <item identifier="LESSON_2" identifierref="RES_LESSON_2">
    <titel>Lesson 2</titel>
  </item>
  <item identifier="LESSON_3" identifierref="RES_LESSON_3">
    <titel>Lesson 3</titel>
    <adlcp:prerequisites type="aicc_script">LESSON_1&LESSON_2</adlcp:prerequisites>
  </item>
</item>
```

Codebeispiel 19: <adlcp:prerequisites>

Anhand des Wertes des RTE-Datenmodell-Elements *cmi.core.lesson_status* ermittelt das LMS, ob die Vorbedingungen erfüllt sind. Wenn im Codebeispiel 19 *cmi.core.lesson_status* für `LESSON_1` und `LESSON_2` den Wert "passed" oder "completed" enthält, ist die Vorbedingung erfüllt.
Wie man im Beispiel sieht, können mehrere SCOs durch Operatoren miteinander verknüpft werden. Diese Operatoren sind Bestandteil der Prerequisites Scripting Language "aicc_script". Tabelle 40 gibt einen Überblick über die Operatoren:

Operator		**Beschreibung**
And	`&`	Und-Verknüpfung: S1 & S2 & S3 S1, S2 und S3 müssen bearbeitet sein ("passed" oder "completed")
Or	`|`	Oder-Verknüpfung: S1| S2 S1 oder S2 muss bearbeitet sein
Not	~	Negation: ~S3 S3 darf noch nicht (erfolgreich) bearbeitet sein ("failed", "incomplete" oder "not attempted")
Equals	=	Genaue Bestimmung: S3="passed" S3 muss den Status "passed" besitzen
Not equals	<>	Ungleich: S3<>"completed" S3 darf nicht den Status "completed" besitzen"
Set	{ }	Liste von SCOs: {S1, S2, S4} S1, S2 und S4 sind Teile der Liste, durch Komma getrennt

Operator		Beschreibung
Separator	,	Trennzeichen in Listen
Minimum	x*	x aus y: 2*{S1, S2, S4} 2 oder mehr der Elemente in der Liste müssen bearbeitet sein
Precedence	( )	Vorrang: S1 & (S2|S3) S1 und S2 oder S1 und S3 müssen bearbeitet sein

Tabelle 40: Prerequisites Scripting Language "aicc_script" [Kaiser, 2001]

Im Prinzip konnte man mit den Prerequisites den Lernpfad von den Interaktionen des Lernenden abhängig machen. Allerdings war das Element `<adlcp:prerequisites>` optional und musste von den Lernmanagementsystemen nicht unterstützt werden. Daher ist das nun eingeführte SCORM Sequencing ein zuverlässigeres Mittel für die Umsetzung von Sequencingstrategien. Durch das komplexe Sequencing Definition Model sind die Möglichkeiten für die Ablaufsteuerung auch sehr viel umfangreicher geworden, in SCORM 1.2 fand die Steuerung nur über das Element *cmi.core.lesson_status* statt. Durch die Objectives können Daten zwischen Activities ausgetauscht werden, was vorher nicht möglich war.

3.6 Navigation Model

Wie im Kapitel 3.4 beschrieben, wird der *Overall Sequencing Process* von Navigation Requests[6] ausgelöst. In der Regel werden diese über Benutzerschnittstellen (z. B. Menü und Buttons) erzeugt, die vom LMS bereitgestellt werden. In manchen Fällen will der Content-Entwickler aber diese Benutzerschnittstellen direkt in die Lerninhalte integrieren und Navigation Requests direkt aus dem SCO erzeugen. Um dies zu realisieren, hat ADL der SCORM-Spezifikation dass Navigation Model hinzugefügt. Es ist kein Bestandteil der IMS-Spezifikation, sondern wurde von ADL zusätzlich erarbeitet.
Genau wie das Sequencing bezieht sich auch das Navigation Model nur auf die Navigation zwischen verschiedenen Activities, die Navigation innerhalb eines Content Objects wird vom LMS nicht registriert oder gesteuert.
Das Modell besteht aus zwei Teilmodellen: über die Elemente des Run-Time Navigation Data Model können Navigation Requests vom SCO zum LMS kommuniziert werden, mit den Elementen des Presentation Information Model kann gesteuert werden, welche Benutzerschnittstellen vom LMS nicht angezeigt werden sollen, während das Content Object angezeigt wird, um redundante Benutzerschnittstellen zu vermeiden. Beide Teilmodelle und deren Anwendung werden in den folgenden Abschnitten erläutert.

[6] Eine Übersicht mit allen Navigation Requests befindet sich im Anhang C.

3.6.1 Run-Time Navigation Data Model

Über die Elemente des Run-Time Navigation Data Model können Navigation Requests vom SCO zum LMS übertragen werden und die Gültigkeit des Navigation Requests kann überprüft werden. Das Modell besteht aus zwei Elementen: *adl.nav.request* und *adl.nav.request_valid*.

adl.nav.request

Über dieses Element kann das SCO den gewünschten Navigation Request zum LMS übertragen. Codebeispiel 20 zeigt ein Anwendungsbeispiel. Die Funktion `doContinue()` (Aufruf z. B. über einen Button) setzt den Wert "continue" für das Element *adl.nav.request*, was später einen *Continue* Navigation Request auslöst. Die Übertragung findet über die API-Instanz statt, die auch für das RTE-Datenmodell genutzt wird.

```
<script>
function doContinue()
{
       doSetValue("adl.nav.request","continue");
       doTerminate();
}
</script>
```

Codebeispiel 20: Anwendung des Elements adl.nav.request

Der Navigation Request wird erst verarbeitet, nachdem die Kommunikation zum LMS beendet wurde. Unmittelbar nach dem Übermitteln des Navigation Requests sollte dann also wie im Beispiel die Methode `Terminate()` aufgerufen werden, damit das LMS den Navigation Request auch wirklich verarbeitet. Vorher hat dieses Element keine Auswirkungen.
Folgende Werte sind möglich:

Wert	Beschreibung
continue previous choice exit exitAll abandon abandonAll	Das LMS soll nach Beendigung der Kommunikation einen entsprechenden Navigation Request (z. B. *Continue* Navigation Request, vgl. Tabelle 1 im Anhang C) verarbeiten.
none	Das LMS soll nach Beendigung der Kommunikation keinen Navigation Request verarbeiten, mit diesem Wert werden eventuell vorher übertragene andere Werte gelöscht.

Tabelle 41: Wertebereich des Elements adl.nav.request

Für einen *Choice* Navigation Request muss zusätzlich das Ziel mit angegeben werden, in der Form `SetValue("adl.nav.request","{target=intro}choice")`, wobei als Ziel die ID der Activity angegeben wird.

adl.nav.request_valid

Für einige Navigation Requests kann das SCO vorab prüfen, ob er gültig ist, z. B. um zu vermeiden, dass Benutzerschnittstellen angezeigt werden, die einen ungültigen Navigation Request erzeugen würden. Im Codebeispiel 21 wird der Continue-Button inaktiv, falls ein *Continue* Navigation Request nicht gültig wäre (z. B. wenn der Control Mode `flow=false` angegeben ist oder keine nachfolgenden Activities im Activity Tree vorhanden sind):

```
if (doGetValue("adl.nav.request_valid.continue") != "true")
{
  disableButton("continue");
}
```

Codebeispiel 21: Anwendung des Elements adl.nav.request_valid

Folgende Navigation Requests können auf ihre Gültigkeit geprüft werden:

Navigation Request	Abfrage
Continue	GetValue("adl.nav.request_valid.continue")
Previous	GetValue("adl.nav.request_valid.previous")
Choice	GetValue("adl.nav.request_valid.choice.{target=intro}")

Tabelle 42: adl.nav.request_valid

Es wird entweder "true", "false" oder "unknown" zurückgegeben.

3.6.2 Presentation Navigation Model

Mit den Elementen des Presentation Information Model kann festgelegt werden, welche Benutzerschnittstellen vom LMS nicht bereitgestellt werden sollen, während das Content Object angezeigt wird. Im Codebeispiel 22 wird angegeben, dass das Navigationselement, welches einen *Continue* Navigation Request auslöst, vom LMS nicht angezeigt werden soll.

```
<item identifier="EINLEITUNG" identifierref="RES_EINLEITUNG">
  <title>Einleitung</title>
  <adlnav:presentation>
    <adlnav:navigationInterface>
      <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
    </adlnav:navigationInterface>
  </adlnav:presentation>
</item>
```

Codebeispiel 22: <adlnav:presentation>

Wie aus dem Codebeispiel 22 ersichtlich ist, werden diese Informationen in der Manifestdatei integriert, mit dem Element `<adlnav:presentation>`, welches ein Unterelement von `<item>` ist. `<adlnav:presentation>` darf aber nur innerhalb von `<item>`-Elementen benutzt werden, die ein SCO referenzieren, demzufolge auch nur an Blatt-Activities. Tabelle 43 zeigt die Unterelemente:

Ebene	Element
1	`<adlnav:presentation>`
1.1	`<navigationInterface>`
1.1.1	`<hideLMSUI>`

Tabelle 43: Übersicht <adlnav:presentation>

Vier Navigationselemente können angegeben werden: `continue`, `previous`, `exit` und `abandon`.

Die Angaben gelten nur für das entsprechende SCO, sobald ein anderes SCO angezeigt wird, sind sie nicht mehr gültig und es werden wieder alle Navigationselemente angezeigt.
Das Presentation Navigation Model soll in Zukunft weiter ausgebaut werden.

In SCORM 1.2 war es nicht möglich, die Navigation zwischen verschiedenen SCOs aus einem SCO heraus zu steuern. Die Einführung des Navigation Models ermöglicht es, dass Navigationselemente (auch ganze Menüs) direkt in die Lerninhalte integriert werden können und die Navigation aber trotzdem über das LMS gesteuert wird. So können Lernangebote kompakter gestaltet werden, die Navigation erfolgt aber trotzdem weiterhin über das LMS. Ob allerdings bei Verwendung des Run-Time Navigation Data Models die Unabhängigkeit der SCOs noch gegeben ist, müsste genauer untersucht werden. Im Prinzip können SCOs jetzt durch die Übertragung von *Choice* Navigation Requests über das Element *adl.nav.request* miteinander "verlinkt" werden, was nicht dem Grundsatz von SCORM entspricht, dass SCOs unabhängig sein sollen vom Gesamtkontext, in den sie eingebettet sind. Diese Untersuchung würde aber den Rahmen der Arbeit sprengen.

4 Prototypische Umsetzung des Sequencing

Ausgehend von den bisherigen Untersuchungen wird in diesem Kapitel an verschiedenen Beispielen die Funktionsweise des Sequencing und speziell die Anwendung des Sequencing Definition Models dargestellt. Für einen Beispielkurs werden verschiedene Sequencingstrategien formuliert und es wird beschrieben, wie die Lerninhalte strukturiert werden müssen (Content Organization) und welche Sequencinginformationen in die Manifest-Datei integriert werden müssen, um die beabsichtigte Sequencingstrategie umzusetzen. Zunächst sollen die verwendeten Werkzeuge vorgestellt werden.

4.1 Verwendete Werkzeuge

Verwendet wurden ein Editor, mit welchem SCORM Content Packages erstellt werden können und eine Beispiel-Implementierung der Laufzeitumgebung von ADL.

4.1.1 Reload Editor

Der Reload[7] Editor ist ein java-basiertes Open-Source-Projekt und unterstützt Content-Entwickler bei der Erstellung von SCORM-konformen Content Packages. Im Rahmen dieser Arbeit wurde der Reload Editor 2004 Beta 1.3.1 verwendet, welcher kostenlos im Internet erhältlich ist[8]. Abbildung 18 zeigt die Benutzeroberfläche:

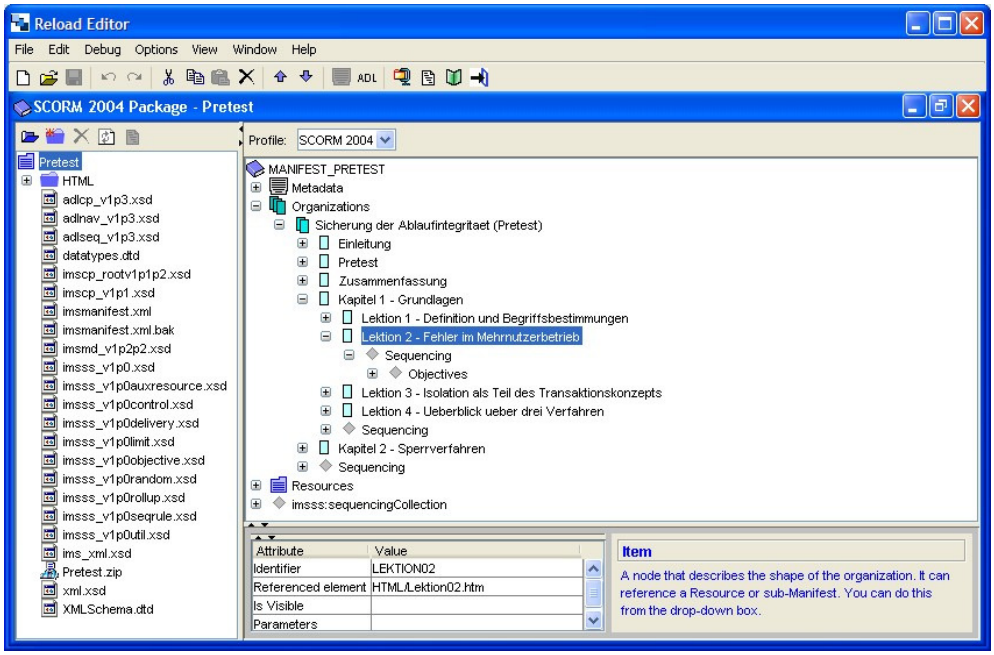


Abbildung 18: Reload Editor

[7] Reusable eLearning Object Authoring & Delivery

[8] http://www.lsal.cmu.edu/adl/scorm/tools/reload/Reload Editor 2004 Beta 1.3.1.zip

Vorteil des Editors ist, dass man keinen XML-Code per Hand bearbeiten muss und so auch Content-Entwickler ohne XML-Kenntnisse eine Manifest-Datei erstellen können. Die Content Organization kann per Drag&Drop erstellt werden und über Eingabemasken können Metadaten und Sequencingangaben hinzugefügt und bearbeitet werden. Mit einem Klick wird zum Schluss das Content Package (zip-Datei) generiert.
Nachteilig ist, dass der Reload Editor meist sämtliche Default-Werte in die Manifest-Datei einträgt, so dass sie etwas unübersichtlich wird. Außerdem entbindet die Nutzung des Editors den Entwickler nicht davon, genau mit den Elementen des Sequencing Definition Models und ihrer Anwendung vertraut zu sein.

4.1.2 ADL Sample Run-Time Environment

Das ADL Sample Run-Time Environment (SampleRTE) ist eine von ADL bereitgestellte Beispielimplementation der in SCORM 2004 beschriebenen Konzepte. Es ist kein vollständiges LMS, sondern zeigt, wie das RTE-Datenmodell und die API sowie das SCORM Sequencing and Navigation in einem LMS umgesetzt werden können.
Das SampleRTE enthält einen API Adapter in Form eines Java Applets, eine vollständige Implementation von RTE-Datenmodell und Navigation Data Model sowie eine Sequencing Engine und ist als webbasierte Client-Server-Applikation realisiert. Abbildung 19 zeigt die Oberfläche des SampleRTE:

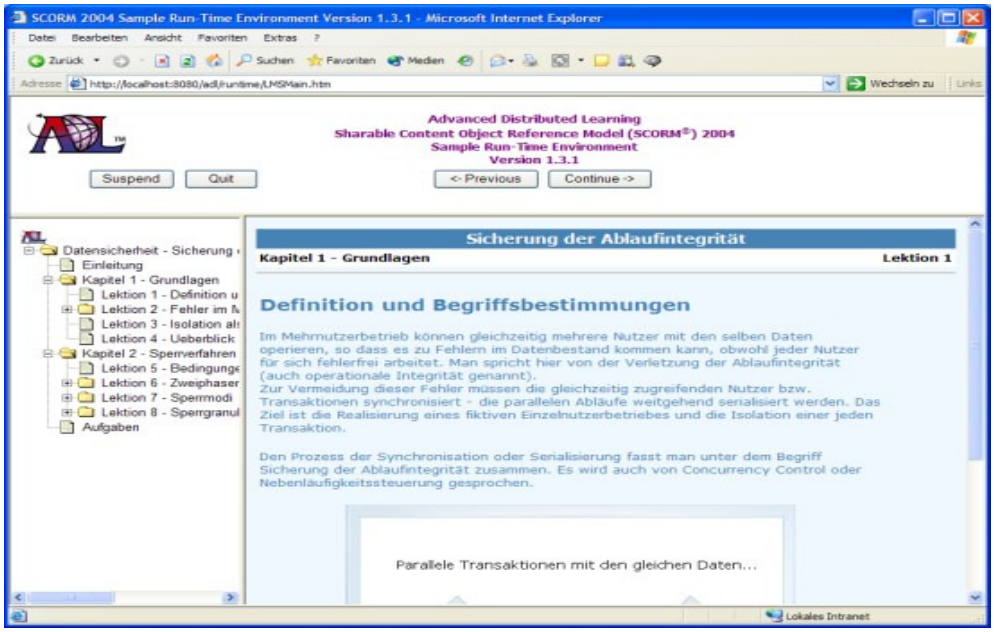


Abbildung 19: ADL Sample Run-Time Environment

Das SampleRTE bietet Funktionen zur Nutzerverwaltung und zum Importieren von Kursen. Nutzer können sich einloggen, sich für Kurse registrieren und Kurse bearbeiten. Administratoren können außerdem den Status der globalen Objectives anzeigen und ändern

und sich den *Objective Satisfied Status* und den *Attempt Progress Status* für die Wurzel-Activity (Element `<organization>`) anzeigen lassen. Die Anwendung ist vor allem in der Benutzeroberflächengestaltung einfach gehalten, sie ist für Content-Entwickler zum Testen von Kursen ausreichend.

Die Implementierung des Sequencing ist besonders seitens des LMS sehr aufwändig, da sämtliche Sequencingprozesse sowie das Tracking Model und Sequencing Definition Model integriert werden müssen. Sicher ist dies ein Grund dafür, dass derzeit noch keine weiteren Laufzeitumgebungen oder Lernmanagementsysteme erhältlich sind, mit denen SCORM 2004 konforme Kurse getestet werden können.

4.2 Kurzbeschreibung des Prototypen

Der Prototyp ist ein Kurs zum Thema "Datensicherheit - Sicherung der Ablaufintegrität in Datenbanken"[9]. Der Kurs gliedert sich in zwei Kapitel mit jeweils vier Lektionen. Zu jeder Lektion gibt es einen Aufgabenblock mit mehreren Multiple-Choice-Aufgaben. Für die Aufgabenblöcke wird ein Punktwert ermittelt und zum LMS übertragen. Anhand des Punktwerts wird auch festgelegt, ob der Aufgabenblock bestanden wurde oder nicht. Außerdem gibt es eine Einleitung und eine Zusammenfassung zu den bearbeiteten Aufgaben. Die grundlegende Struktur ist in Abbildung 20 dargestellt:

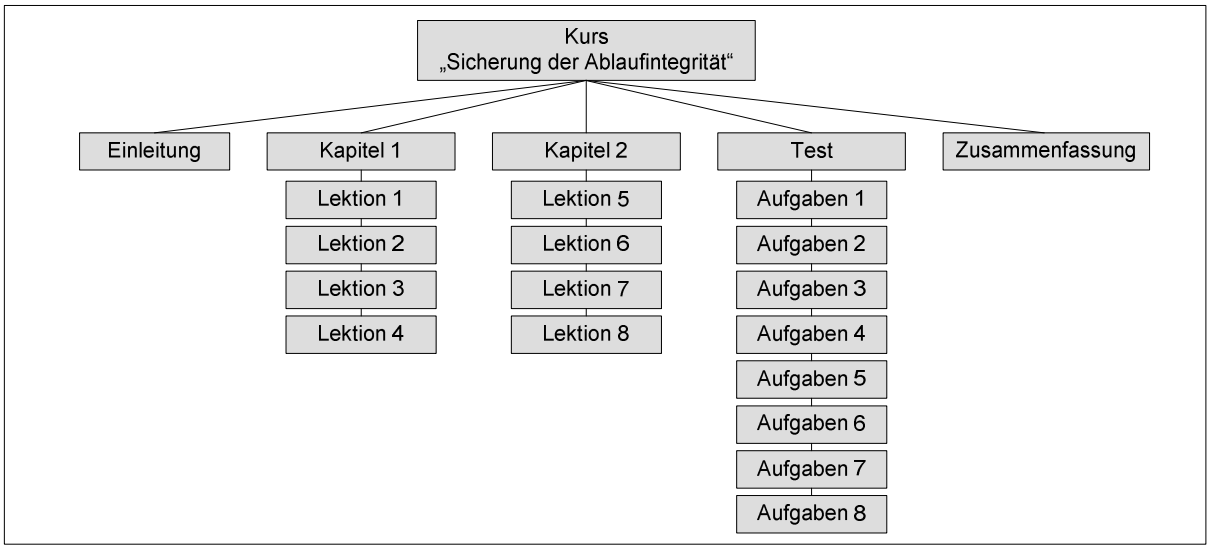


Abbildung 20: Allgemeine Kursstruktur

[9] Die Lerninhalte des Prototypen wurden dem Lernmodul "Datensicherheit und Datenintegrität in Datenbanken" entnommen, welches an der HTW Dresden erstellt wurde und im Bildungsportal Sachsen [Bildungsportal] angeboten wird.

Sämtliche Inhalte sind als HTML-Seiten umgesetzt, wobei jede Lektion und jeder Aufgabenblock aus einem HTML-Dokument besteht. Die nachfolgenden Abbildungen zeigen eine Lektion und einen Aufgabenblock:

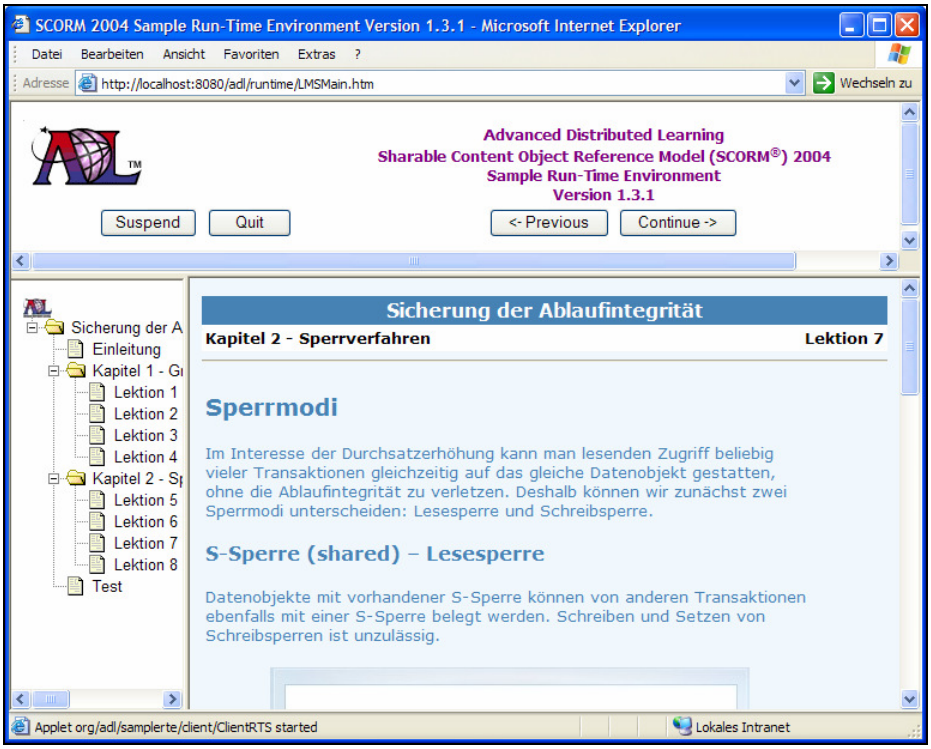


Abbildung 21: Lektion im Prototyp

Abbildung 22: Aufgabenblock im Prototyp

Da der Schwerpunkt auf der technischen Umsetzung des Sequencing liegt, wurden keine didaktischen und gestalterischen Aspekte berücksichtigt. Der Originalkurs enthält zahlreiche Animationen, in diesem Prototypen wurde an deren Stelle nur ein Screenshot der Animation eingefügt.
Die Manifest-Dateien der Beispielkurse enthalten keine Metadaten, da diese keinen Einfluss auf das Sequencing haben und somit für die Funktionsweise nicht von Bedeutung sind.
Es wurden drei verschiedene Sequencingstrategien für diesen Kurs festgelegt, deren Umsetzung in den nächsten Abschnitten beschrieben wird.

4.3 Beispiel "Freie Auswahl mit Wiederholung"

Dieser Kurs beinhaltet die acht Lektionen, die Aufgaben sind als Test zur Wissensüberprüfung zusammengefasst und abhängig vom Wissen des Lernenden werden die Lektionen noch einmal wiederholt.

4.3.1 Sequencingstrategie

Beim Start des Kurses wird die Einleitung angezeigt. Im Menü des LMS sind beide Kapitel mit allen Lektionen und der Test enthalten. Alle Lektionen können in beliebiger Reihenfolge beliebig oft ausgewählt werden. Der Lernende kann sich außerdem über die Navigationselemente des LMS ("Vor"- und "Zurück"-Buttons) in vorgegebener Reihenfolge durch die Lektionen bewegen. Wird der Test ausgewählt (entweder direkt oder durch Vorblättern aus der letzten Lektion), wird dem Lernenden die erste der insgesamt acht Aufgabenseiten angezeigt. Während der Bearbeitung des Tests kann nicht mehr zu den Lektionen gewechselt werden, es wird kein Menü angezeigt. Der Lernende muss die Aufgaben in vorgegebener Reihenfolge abarbeiten, er kann dabei nur vorwärts blättern. Der Test kann insgesamt nur einmal bearbeitet werden. Nach der letzten Aufgabenseite erscheint eine Zusammenfassung der Ergebnisse, wo dem Lernenden zu jeder Lektion angezeigt wird, zu wieviel Prozent er den Aufgabenblock richtig gelöst hat und ob der Aufgabenblock bestanden ist. Sind alle Aufgabenblöcke bestanden, ist der Kurs an dieser Stelle beendet. Ist ein Aufgabenblock nicht bestanden, wird die entsprechende Lektion in die Wiederholung aufgenommen. Die zu wiederholenden Lektionen werden dem Lernenden anschließend in linearer Reihenfolge präsentiert, er kann vor- und zurückblättern, die Lektionen aber nicht in einem Menü auswählen.
Der Kurs gilt als vollständig abgearbeitet und wird beendet, wenn alle Aufgaben und zu wiederholenden Lektionen bearbeitet wurden. Als bestanden gilt der Test und somit der Kurs, wenn mindestens sechs der acht Aufgabenblöcke bestanden wurden.

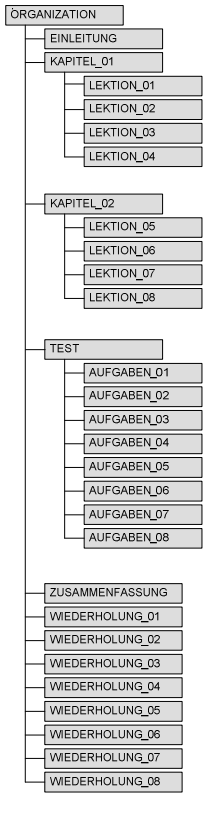
4.3.2 Umsetzung der Struktur

In der Content Organization wird die Struktur des Kurses festgelegt. Sie enthält alle Activities. Aus der oben beschriebenen Sequencingstrategie lässt sich folgende Struktur ableiten:

- Wurzel-Activity "ORGANIZATION" (Element <organization>)
- eine Activity "EINLEITUNG"
- jeweils eine Activity für jedes Kapitel mit den Lektionen als Kind-Activities
- eine Activity "TEST" mit den Aufgabenblöcken als Kind-Activities
- eine Activity "ZUSAMMENFASSUNG"
- eine Activity für jede Wiederholungs-Lektion

Die Wiederholungs-Lektionen könnten theoretisch auch wieder den Kapiteln zugeordnet werden, da sie aber im Menü nicht angezeigt werden, kann darauf auch verzichtet und die Struktur dadurch vereinfacht werden.

Aus dieser Struktur lassen sich nun wie in Abbildung 23 dargestellt die einzelnen <item>-Elemente innerhalb der <organization> ableiten:




```
<organization identifier="ORGANIZATION">
 <item identifier="EINLEITUNG" identifierref="RES_EINLEITUNG"></item>
 <item identifier="KAPITEL_01">
  <item identifier="LEKTION_01" identifierref="RES_LEKTION_01"></item>
  <item identifier="LEKTION_02" identifierref="RES_LEKTION_02"></item>
  <item identifier="LEKTION_03" identifierref="RES_LEKTION_03"></item>
  <item identifier="LEKTION_04" identifierref="RES_LEKTION_04"></item>
 </item>
 <item identifier="KAPITEL_02">
  <item identifier="LEKTION_05" identifierref="RES_LEKTION_05"></item>
  <item identifier="LEKTION_06" identifierref="RES_LEKTION_06"></item>
  <item identifier="LEKTION_07" identifierref="RES_LEKTION_07"></item>
  <item identifier="LEKTION_08" identifierref="RES_LEKTION_08"></item>
 </item>
 <item identifier="TEST">
  <item identifier="AUFGABEN_01" identifierref="RES_AUFGABEN_01"></item>
  <item identifier="AUFGABEN_02" identifierref="RES_AUFGABEN_02"></item>
  <item identifier="AUFGABEN_03" identifierref="RES_AUFGABEN_03"></item>
  <item identifier="AUFGABEN_04" identifierref="RES_AUFGABEN_04"></item>
  <item identifier="AUFGABEN_05" identifierref="RES_AUFGABEN_05"></item>
  <item identifier="AUFGABEN_06" identifierref="RES_AUFGABEN_06"></item>
  <item identifier="AUFGABEN_07" identifierref="RES_AUFGABEN_07"></item>
  <item identifier="AUFGABEN_08" identifierref="RES_AUFGABEN_08"></item>
 </item>
 <item identifier="ZUSAMMENF" identifierref="RES_ZUSAMMENF"></item>
 <item identifier="WIEDERHOLUNG_01" identifierref="RES_LEKTION_01"></item>
 <item identifier="WIEDERHOLUNG_02" identifierref="RES_LEKTION_02"></item>
 <item identifier="WIEDERHOLUNG_03" identifierref="RES_LEKTION_03"></item>
 <item identifier="WIEDERHOLUNG_04" identifierref="RES_LEKTION_04"></item>
 <item identifier="WIEDERHOLUNG_05" identifierref="RES_LEKTION_05"></item>
 <item identifier="WIEDERHOLUNG_06" identifierref="RES_LEKTION_06"></item>
 <item identifier="WIEDERHOLUNG_07" identifierref="RES_LEKTION_07"></item>
 <item identifier="WIEDERHOLUNG_08" identifierref="RES_LEKTION_08"></item>
</organization>
```

Abbildung 23: Content Organization "Freie Auswahl mit Wiederholung"

Die Ressourcen für die Lektionen werden als Asset definiert, da keine Kommunikation zwischen Ressource und LMS notwendig ist. Die Ressourcen für den Test werden als SCO

definiert, da hier Daten zum LMS übertragen werden müssen (*cmi.succes_status* und *cmi.score.scaled*). Für die Zusammenfassung der Testergebnisse wird die Ressource ebenfalls als SCO definiert, da hier die Daten aller globalen Objectives über das RTE-Datenmodell vom LMS abgefragt werden müssen. Mehr dazu im nächsten Abschnitt unter "Objectives".

4.3.3 Sequencingangaben

Vorab wurde die Sequencingstrategie bereits ausformuliert. Nun soll erläutert werden, wie sie umgesetzt werden kann. Hierzu wird für jede Activity aufgelistet, wie die einzelnen Details der Sequencingstrategie mit den Elementen des Sequencing Definition Models realisiert werden. Es werden nur die entsprechenden Attribute der Elemente bzw. die Regeln in Kurzform aufgeführt, um die Übersichtlichkeit zu gewährleisten. Die genaue Struktur der Elemente wurde bereits in Kapitel 3.3 ausführlich beschrieben, die komplette Manifest-Datei zu jedem Beispielkurs befindet sich auf der beiliegenden CD.
Gleichartige Activities wie Lektionen und Aufgaben werden nur einmal aufgeführt (`LEKTION_xx` steht für alle Lektionen usw.), die Angaben müssen aber für jede Activity gemacht werden.

Control Mode

Über die Control Modes werden der Inhalt und die Anzeige des Navigationsmenüs gesteuert. Die Einzelheiten sind in Tabelle 44 aufgeführt:

Activity	Strategie	Control Mode
ORGANIZATION	Einleitung, Kapitel und Test sind im Menü auswählbar	`choice="true"`
	mit den Navigationselementen des LMS kann durch die Inhalte geblättert werden	`flow="true"`
EINLEITUNG	-	–
KAPITEL_xx	alle Lektionen sind im Menü auswählbar	`choice="true"`
	mit den Navigationselementen des LMS kann durch die Lektionen geblättert werden	`flow="true"`
LEKTION_xx	-	–
TEST	während der Bearbeitung des Test wird das Menü nicht mehr angezeigt	`choiceExit="false"`
	die Aufgaben erscheinen nicht im Menü	`choice="false"`
	mit den Navigationselementen des LMS kann durch die Aufgaben geblättert werden	`flow="true"`
	es kann innerhalb des Test nur vorwärts geblättert werden	`forwardOnly="true"`
AUFGABEN_xx	-	–
ZUSAMMENFASSUNG	während der Anzeige der Zusammenfassung wird das Menü nicht mehr angezeigt	`choiceExit="false"`
WIEDERHOLUNG_xx	während der Anzeige der Wiederholungs-Lektionen wird das Menü nicht mehr angezeigt	`choiceExit="false"`

Tabelle 44: Control Modes für den Kurs "Freie Auswahl ..."

Sequencing Rules

Die Sequencing Rules stellen in diesem Fall sicher, dass der Kurs verlassen wird, wenn er vollständig bearbeitet wurde und dass bestimmte Activities nicht im Menü erscheinen. Außerdem wird hier festgelegt, dass der Test nach Beendigung nicht mehr ausgewählt werden kann und dass eine Wiederholungs-Lektion übersprungen wird, wenn der entsprechende Aufgabenblock bestanden wurde.

Activity	Strategie	Sequencing Rules
ORGANIZATION	der Kurs wird beendet, wenn er vollständig bearbeitet wurde	Exit-Condition-Rule: `if completed then exit`
EINLEITUNG	-	–
KAPITEL_xx	-	–
LEKTION_xx	-	–
TEST	wenn der Test vollständig bearbeitet wurde, kann er nicht mehr ausgewählt werden	Pre-Condition-Rule: `if completed then disabled`
AUFGABEN_xx	-	–
ZUSAMMENFASSUNG	die Zusammenfassung soll nicht im Menü erscheinen	Pre-Condition-Rule: `always hiddenFromChoice`
WIEDERHOLUNG_xx	wenn der entsprechende Aufgaben-block bestanden wurde, wird die Lektion übersprungen	Pre-Condition-Rule: `if satisfied then skip`
	die Wiederholungs-Lektionen sollen nicht im Menü erscheinen	Pre-Condition-Rule: `always hiddenFromChoice`

Tabelle 45: Sequencing Rules für den Kurs "Freie Auswahl ..."

Rollup Rules und Rollup Considerations

Um den Erfolgsstatus und den Bearbeitungsstatus für den ganzen Kurs (entspricht der Wurzel-Activity `<organization>`) zu bestimmen, werden Rollup Rules angebracht. Sie "transportieren" den Status von den Blättern zur Wurzel des Activity Trees.

Activity	Strategie	Rollup Rules und Rollup Considerations
ORGANIZATION	wenn der Test bestanden wurde, gilt auch der Kurs als bestanden	`if satisfied TRUE for all then satisfied`
	wenn der Test nicht bestanden wurde, gilt auch der Kurs als nicht bestanden	`if not satisfied TRUE for any then notSatisfied`
	wurde auf den Test und die Wiederholungs-Lektionen zugegriffen, gilt der Kurs als vollständig bearbeitet	`if attempted TRUE for all then completed`
	wurde auf den Test oder eine Wiederholungs-Lektionen nicht zugegriffen, gilt der Kurs als unvollständig bearbeitet	`if not attempted TRUE for any then incomplete`
EINLEITUNG	der Status der Einleitung hat keinen Einfluss auf das Bestehen des Kurses	`rollupObjectiveSatisfied ="false"`
KAPITEL_xx	der Erfolgsstatus der Lektionen bzw. Kapitel hat keinen Einfluss auf das Bestehen des Kurses, dafür ist nur der Status des Tests entscheidend	`rollupObjectiveSatisfied ="false"`
	der Status der Lektionen hat keinen Einfluss auf die vollständige Bearbeitung des Kurses (da sie nicht alle bearbeitet werden müssen)	`rollupProgressCompletion ="false"`
LEKTION_xx	-	–

Activity	Strategie	Rollup Rules und Rollup Considerations
TEST	wenn mindestens 6 Aufgaben bestanden wurden, ist der Test bestanden	`if satisfied TRUE for atLeastCount 6 then satisfied`
	wenn mindestens 3 Aufgaben nicht bestanden wurden, ist der Test nicht bestanden	`if not satisfied TRUE for atLeastCount 3 then notSatisfied`
	wurden alle Aufgaben bearbeitet, ist der Test vollständig bearbeitet	`if attempted TRUE for all then completed`
	solange nicht alle Aufgaben bearbeitet wurden, ist der Test unvollständig	`if not attempted TRUE for any then incomplete`
AUFGABEN_xx	-	-
ZUSAMMENFASSUNG	der Status der Zusammenfassung hat keinen Einfluss auf das Bestehen des Kurses	`rollupObjectiveSatisfied ="false"`
WIEDERHOLUNG_xx	der Status der Wiederholungs-Lektionen hat keinen Einfluss auf das Bestehen des Kurses	`rollupObjectiveSatisfied ="false"`
	nur die nicht bestandenen Lektionen müssen noch einmal bearbeitet werden, damit der Kurs vollständig ist	`requiredForCompleted ="ifNotSkipped"`

Tabelle 46: Rollup Rules und Rollup Considerations für den Kurs "Freie Auswahl ..."

Objectives

Jeder Lektion ist ein Aufgabenblock zugeordnet. Um zu entscheiden, ob eine Wiederholungs-Lektion angezeigt werden soll oder nicht, muss überprüft werden, ob der entsprechende Aufgabenblock bestanden wurde. Mittels eines globalen Objectives, welches für jede Lektion bzw. jeden Aufgabenblock definiert wird, kann der Erfolgsstatus zwischen den Activities ausgetauscht werden. Im Element `<mapInfo>` wird festgelegt, dass die Aufgaben-Activities ihren Status in das globale Objective schreiben und dass die Lektionen-Activities den Status auslesen. Dies ist Abbildung 24 in für Lektion 1 dargestellt und gilt für alle Lektionen:

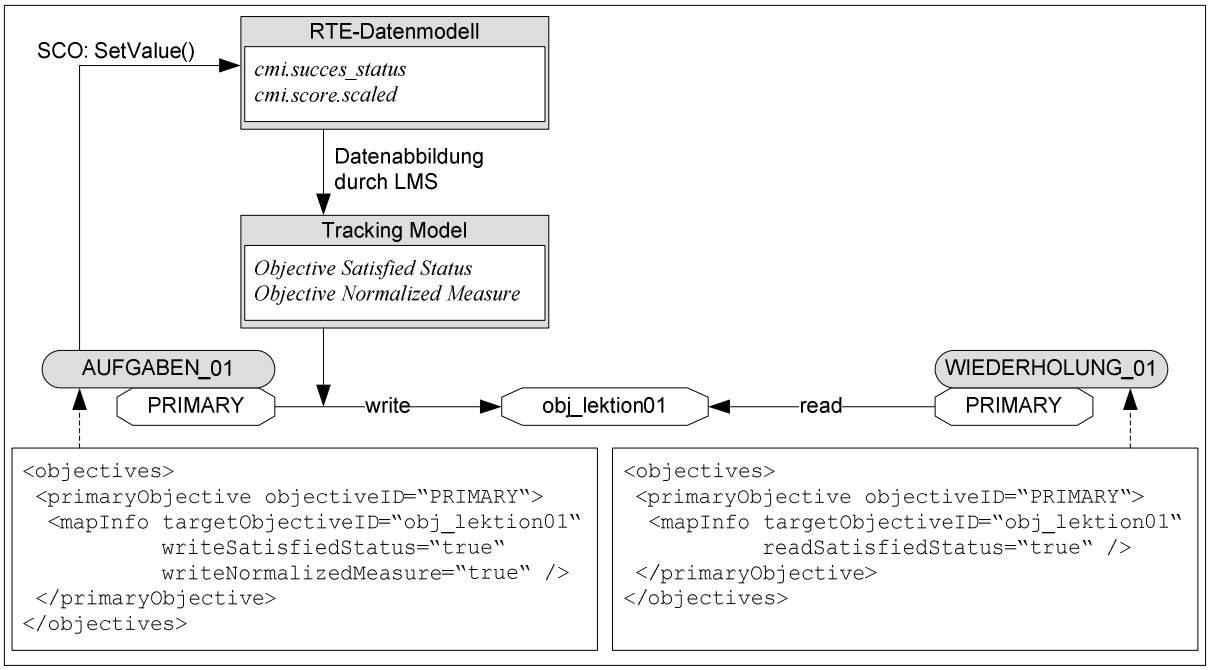


Abbildung 24: Datenaustausch über globale Objectives

Damit die Daten auf das globale Objective abgebildet werden können, müssen sie vorher vom SCO gesetzt werden. Der entsprechende Code ist in Codebeispiel 4 auf Seite 28 bereits dargestellt.

Die Activity `"ZUSAMMENFASSUNG"` benutzt alle globalen Objectives. Für jede Lektion wird ein lokales Objective angelegt. Dieses liest die Werte, wie im Element `<mapInfo>` angegeben, vom entsprechenden globalen Objective. Mit diesen Werten werden dann beim Start des SCOs für jedes lokale Objective die entsprechenden Elemente im RTE-Datenmodell (*cmi.objectives.n.success_status*, *cmi.objectives.n.score.scaled*) initialisiert. Abbildung 25 verdeutlicht diesen Zusammenhang:

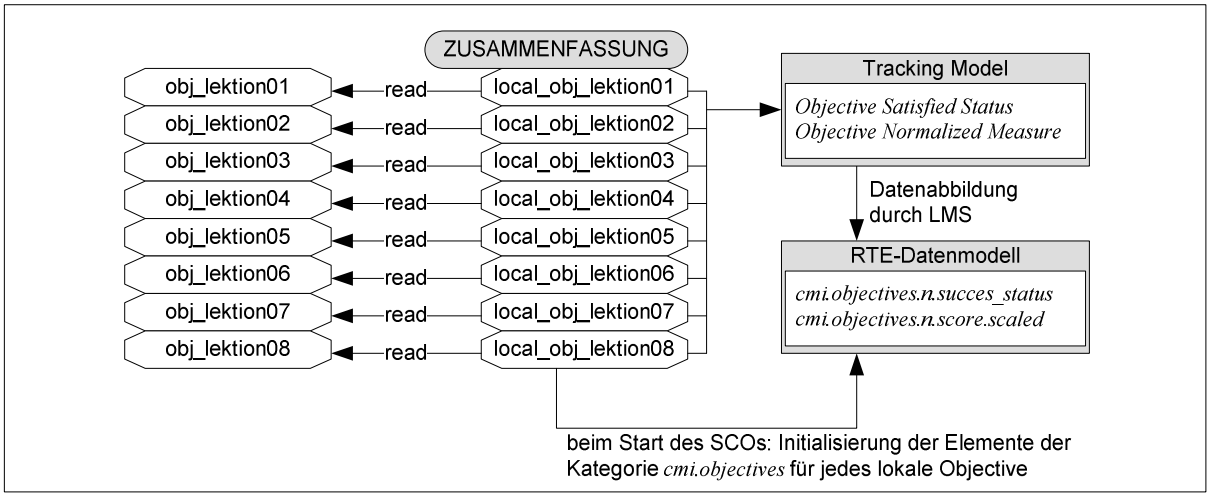


Abbildung 25: Objectives der Activity "Zusammenfassung"

Codebeispiel 23 zeigt die notwendigen Angaben in der Manifest-Datei:

```
<item identifier="ZUSAMMENFASSUNG" identifierref="RES_ZUSAMMENFASSUNG">
  <title>Zusammenfassung Uebungsaufgaben</title>
  <imsss:sequencing>
    <!-- other sequencing data -->
    <imsss:objectives>
      <imsss:primaryObjective />
      <imsss:objective objectiveID="local_obj_lektion01">
        <imsss:mapInfo targetObjectiveID="obj_lektion01" readSatisfiedStatus="true"
                       readNormalizedMeasure="true"/>
      </imsss:objective>
      <imsss:objective objectiveID="local_obj_lektion02">
        <imsss:mapInfo targetObjectiveID="obj_lektion02" readSatisfiedStatus="true"
                       readNormalizedMeasure="true"/>
      </imsss:objective>
      ...
      <!-- other objectives -->
      ...
    </imsss:objectives>
  </imsss:sequencing>
</item>
```

Codebeispiel 23: Objectives der Activity "Zusammenfassung"

Das Element *cmi.objectives.n.id* erhält als Wert die `objectiveID` des lokalen Objectives. Über diese ID kann jedes lokale Objective innerhalb des RTE-Datenmodells identifiziert werden und die Daten können vom SCO abgefragt und verarbeitet werden, wie im

Codebeispiel 24, einem Auszug aus der Ressource der Activity "ZUSAMMENFASSUNG", dargestellt:

```
<html>
  <head>
    <script language="JavaScript">
      var anzahlLektionen = 8;
      var objScaledScore = new Array(anzahlLektionen);
      var objSuccessStatus = new Array(anzahlLektionen);

      function loadPage()
      {
        ...          .
        for (i=0; i<anzahlLektionen; i++)
        {
          getObjectivesStatus(i);
        }
        ...
      }

      function getObjectivesStatus(lektion)
      {
        var tempObjective = "local_obj_lektion0"  + (lektion+1);
        var tempIndex = findObjective(tempObjective);
        var tempObjScore = "cmi.objectives." + tempIndex + ".score.scaled";
        var tempObjStatus = "cmi.objectives." + tempIndex + ".success_status";
        objScaledScore[lektion] = doGetValue(tempObjScore);
        objSuccessStatus[lektion] = doGetValue(tempObjStatus);
      }

      function findObjective(obj)
      {
         var numOfObj = doGetValue("cmi.objectives._count");
         var objectiveLocation;
        for ( var i=0; i < numOfObj; i++ )
        {
           if ( doGetValue("cmi.objectives." + i + ".id") == obj )
          {
             objectiveLocation = i;
             break;
          }
         }
        return objectiveLocation;
      }
  </script>
</head>
<body onLoad="loadPage()">
Inhalt
</body>
</html>
```

Codebeispiel 24: Auszug aus der Ressource der Activity "Zusammenfassung"

Die Werte der Objectives werden in diesem Fall in einer Tabelle verarbeitet, siehe Abbildung 26:

Abbildung 26: Nutzung der Objectives in der Zusammenfassung

Die Aussagen zu den Objectives gelten auch für die Umsetzung der anderen zwei Sequencingstrategien in Kapitel 4.4 und 4.5.

4.4 Beispiel "Lineare Abfolge mit Wiederholung"

Diese Kursvariante ist im Prinzip identisch mit der Variante "Freie Auswahl mit Wiederholung", sie besteht ebenfalls aus den acht Lektionen, einem Test zur Wissenskontrolle und den Wiederholungs-Lektionen. Unterschiedlich ist, dass die Reihenfolge für die Bearbeitung der Lektionen nicht frei wählbar ist.

4.4.1 Sequencingstrategie

Der Kurs beginnt mit der Einleitung. Ein Menü wird nicht angezeigt. Der Lernende kann sich nur über die Navigationselemente des LMS ("Vor"- und "Zurück"-Buttons) in vorgegebener Reihenfolge durch die Lektionen bewegen. Nach den acht Lektionen werden die Aufgabenblöcke bearbeitet. Der Lernende muss die Aufgaben in vorgegebener Reihenfolge abarbeiten, er kann dabei nur vorwärts blättern. Der Test kann insgesamt nur einmal bearbeitet werden. Nach der letzten Aufgabenseite erscheint eine Zusammenfassung der Ergebnisse, wo dem Lernenden zu jeder Lektion angezeigt wird, zu wieviel Prozent er den Aufgabenblock richtig gelöst hat und ob der Aufgabenblock bestanden ist. Sind alle Aufgabenblöcke bestanden, ist der Kurs an dieser Stelle beendet. Ist ein Aufgabenblock nicht bestanden, wird die entsprechende Lektion in die Wiederholung aufgenommen. Die zu wiederholenden Lektionen werden dem Lernenden anschließend in linearer Reihenfolge präsentiert, er kann vor- und zurückblättern, die Lektionen aber nicht in einem Menü auswählen.
Der Kurs gilt als vollständig abgearbeitet und wird beendet, wenn alle Aufgaben und zu wiederholenden Lektionen bearbeitet wurden. Als bestanden gilt der Test und somit der Kurs, wenn mindestens sechs der acht Aufgabenblöcke bestanden wurden.

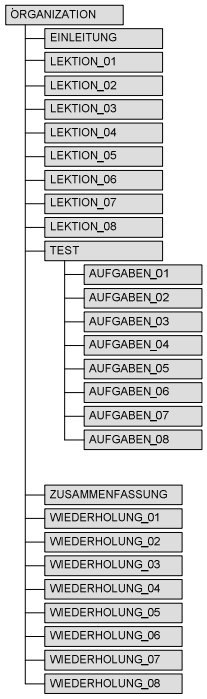
4.4.2 Umsetzung der Struktur

Die Struktur unterscheidet sich vom Kurs in Kapitel 4.3 dahingehend, dass auf eine Untergliederung in Kapitel innerhalb der Content Organization verzichtet wurde, da die Struktur nicht in einem Menü dargestellt wird. Die einzelnen Aufgaben sind aber auch hier der Activity `"TEST"` untergeordnet, um zu ermöglichen, dass zwischen den Aufgaben nur vorwärts geblättert werden kann. Es ergibt sich folgender Aufbau:

- Wurzel-Activity `"ORGANIZATION"` (Element `<organization>`)
- eine Activity `"EINLEITUNG"`

- jeweils eine Activity für jede Lektion
- eine Activity "TEST" mit den Aufgabenblöcken als Kind-Activities
- eine Activity "ZUSAMMENFASSUNG"
- eine Activity für jede Wiederholungs-Lektion

Die entsprechende Anordnung der <item>-Elemente in der Manifest-Datei ist in Abbildung 27 dargestellt:




```
<organization identifier="ORGANIZATION">
 <item identifier="EINLEITUNG" identifierref="RES_EINLEITUNG"></item>
 <item identifier="LEKTION_01" identifierref="RES_LEKTION_01"></item>
 <item identifier="LEKTION_02" identifierref="RES_LEKTION_02"></item>
 <item identifier="LEKTION_03" identifierref="RES_LEKTION_03"></item>
 <item identifier="LEKTION_04" identifierref="RES_LEKTION_04"></item>
 <item identifier="LEKTION_05" identifierref="RES_LEKTION_05"></item>
 <item identifier="LEKTION_06" identifierref="RES_LEKTION_06"></item>
 <item identifier="LEKTION_07" identifierref="RES_LEKTION_07"></item>
 <item identifier="LEKTION_08" identifierref="RES_LEKTION_08"></item>
 <item identifier="TEST">
  <item identifier="AUFGABEN_01" identifierref="RES_AUFGABEN_01"></item>
  <item identifier="AUFGABEN_02" identifierref="RES_AUFGABEN_02"></item>
  <item identifier="AUFGABEN_03" identifierref="RES_AUFGABEN_03"></item>
  <item identifier="AUFGABEN_04" identifierref="RES_AUFGABEN_04"></item>
  <item identifier="AUFGABEN_05" identifierref="RES_AUFGABEN_05"></item>
  <item identifier="AUFGABEN_06" identifierref="RES_AUFGABEN_06"></item>
  <item identifier="AUFGABEN_07" identifierref="RES_AUFGABEN_07"></item>
  <item identifier="AUFGABEN_08" identifierref="RES_AUFGABEN_08"></item>
 </item>
 <item identifier="ZUSAMMENF" identifierref="RES_ZUSAMMENF"></item>
 <item identifier="WIEDERHOLUNG_01" identifierref="RES_LEKTION_01"></item>
 <item identifier="WIEDERHOLUNG_02" identifierref="RES_LEKTION_02"></item>
 <item identifier="WIEDERHOLUNG_03" identifierref="RES_LEKTION_03"></item>
 <item identifier="WIEDERHOLUNG_04" identifierref="RES_LEKTION_04"></item>
 <item identifier="WIEDERHOLUNG_05" identifierref="RES_LEKTION_05"></item>
 <item identifier="WIEDERHOLUNG_06" identifierref="RES_LEKTION_06"></item>
 <item identifier="WIEDERHOLUNG_07" identifierref="RES_LEKTION_07"></item>
 <item identifier="WIEDERHOLUNG_08" identifierref="RES_LEKTION_08"></item>
</organization>
```

Abbildung 27: Content Organization "Lineare Abfolge mit Wiederholung"

Auch hier werden nur die Ressourcen des Tests und der Zusammenfassung als SCO definiert, da nur bei diesen Inhalten die Kommunikation mit dem LMS erforderlich ist.

4.4.3 Sequencingangaben

Im Folgenden wird gezeigt, wie die vorab beschriebene Sequencingstrategie umgesetzt werden kann. Die komplette Manifest-Datei befindet sich auf der beiliegenden CD.

Control Mode

Die hier verwendeten Control Modes verhindern, dass ein Menü angezeigt wird und gewährleisten, dass der Lernende sich über die Navigationselemente des LMS durch die Inhalte bewegen kann. Tabelle 47 zeigt die Angaben im Einzelnen:

Activity	Strategie	Control Mode
ORGANIZATION	es wird kein Menü angezeigt	choice="false"
	mit den Navigationselementen des LMS kann durch die Inhalte geblättert werden	flow="true"

Activity	Strategie	Control Mode
EINLEITUNG	-	–
LEKTION_xx	-	–
TEST	es wird kein Menü angezeigt	choice="false"
	mit den Navigationselementen des LMS kann durch die Aufgaben geblättert werden	flow="true"
	es kann innerhalb des Test nur vorwärts geblättet werden	forwardOnly="true"
AUFGABEN_xx	-	–
ZUSAMMENFASSUNG	-	–
WIEDERHOLUNG_xx	-	–

Tabelle 47: Control Modes für den Kurs "Lineare Abfolge ..."

Sequencing Rules

Mit den Sequencing Rules wird festgelegt, dass der Kurs beendet wird, wenn er vollständig abgearbeitet ist, dass der Test nach Beendigung nicht noch einmal aufgerufen werden kann und dass Wiederholungs-Lektionen übersprungen werden, deren zugehöriger Aufgabenblock bestanden wurde.

Activity	Strategie	Sequencing Rules
ORGANIZATION	der Kurs wird beendet, wenn er vollständig bearbeitet wurde	Exit-Condition-Rule: if completed then exit
EINLEITUNG	-	–
LEKTION_xx	-	–
TEST	wenn der Test vollständig bearbeitet wurde, kann er nicht mehr ausgewählt werden	Pre-Condition-Rule: if completed then disabled
AUFGABEN_xx	-	–
ZUSAMMENFASSUNG	-	–
WIEDERHOLUNG_xx	wenn der entsprechende Aufgaben-block bestanden wurde, wird die Lektion übersprungen	Pre-Condition-Rule: if satisfied then skip

Tabelle 48: Sequencing Rules für den Kurs "Lineare Abfolge ..."

Rollup Rules und Rollup Considerations

Um den Kurs zu beenden, sobald er vollständig bearbeitet wurde und um festzuhalten, ob er bestanden wurde, wird mit Hilfe der Rollup Rules der Status der Activity "ORGANIZATION" ermittelt.

Activity	Strategie	Rollup Rules und Rollup Considerations
ORGANIZATION	wenn der Test bestanden wurde, gilt auch der Kurs als bestanden	if satisfied TRUE for all then satisfied
	wenn der Test nicht bestanden wurde, gilt auch der Kurs als nicht bestanden	if not satisfied TRUE for any then notSatisfied
	wurde auf den Test und die Wiederholungs-Lektionen zugegriffen, gilt der Kurs als vollständig bearbeitet	if attempted TRUE for all then completed
	wurde auf den Test oder eine Wiederholungs-Lektionen nicht zugegriffen, gilt der Kurs als unvollständig bearbeitet	if not attempted TRUE for any then incomplete
EINLEITUNG	der Status der Einleitung hat keinen Einfluss auf das Bestehen des Kurses	rollupObjectiveSatisfied ="false"

Activity	Strategie	Rollup Rules und Rollup Considerations
LEKTION_xx	der Status der Lektionen hat keinen Einfluss auf das Bestehen des Kurses	rollupObjectiveSatisfied ="false"
TEST	wenn mindestens 6 Aufgaben bestanden wurden, ist der Test bestanden	if satisfied TRUE for atLeastCount 6 then satisfied
	wenn mindestens 3 Aufgaben nicht bestanden wurden, ist der Test nicht bestanden	if not satisfied TRUE for atLeastCount 3 then notSatisfied
	wurden alle Aufgaben bearbeitet, ist der Test vollständig bearbeitet	if attempted TRUE for all then completed
	solange nicht alle Aufgaben bearbeitet wurden, ist der Test unvollständig	if not attempted TRUE for any then incomplete
AUFGABEN_xx	-	-
ZUSAMMENFASSUNG	der Status der Zusammenfassung hat keinen Einfluss auf das Bestehen des Kurses	rollupObjectiveSatisfied ="false"
WIEDERHOLUNG_xx	der Status der Wiederholungs-Lektionen hat keinen Einfluss auf das Bestehen des Kurses	rollupObjectiveSatisfied ="false"
	nur die nicht bestandenen Lektionen müssen noch einmal bearbeitet werden, damit der Kurs vollständig ist	requiredForCompleted ="ifNotSkipped"

Tabelle 49: Rollup Rules und Rollup Considerations für den Kurs "Lineare Abfolge ..."

Objectives

Die Angaben zu den Objectives entsprechen den Ausführungen in Kapitel 4.3.3 auf Seite 83 und sollen hier nicht noch einmal erläutert werden.

4.5 Beispiel "Persönlicher Lernweg mit Pretest"

Diese Kursvariante enthält einen Pretest zur Wissensüberprüfung und die acht Lektionen, welche in Abhängigkeit vom Erfolg des Benutzers zu einem persönlichen Lernweg zusammengestellt werden.

4.5.1 Sequencingstrategie

Der Kurs beginnt mit der Einleitung. Es wird kein Menü angezeigt. Innerhalb der Einleitung befindet sich ein Link zum Pretest. Im Pretest muss der Lernende die Aufgaben in vorgegebener Reihenfolge abarbeiten, er kann dabei nur vorwärts blättern. Der Test kann insgesamt nur einmal bearbeitet werden. Nach der letzten Aufgabenseite erscheint eine Zusammenfassung der Ergebnisse, wo dem Lernenden zu jeder Lektion angezeigt wird, zu wieviel Prozent er den Aufgabenblock richtig gelöst hat und ob der Aufgabenblock bestanden ist. Im Menü werden jetzt die Lektionen angezeigt, die der Lernende bearbeiten soll. Er kann die Lektionen in beliebiger Reihenfolge auswählen und außerdem kann er sich über die Navigationselemente des LMS ("Vor"- und "Zurück"-Buttons) in vorgegebener Reihenfolge durch die Lektionen bewegen.

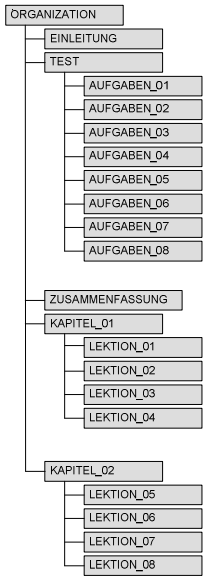
Der Kurs gilt als vollständig abgearbeitet, wenn der Pretest und alle notwendigen Lektionen bearbeitet wurden. Er wird aber nicht automatisch beendet, da die Lektionen beliebig oft bearbeitet werden können. Als bestanden gilt der Kurs, wenn mindestens sechs der acht Aufgabenblöcke erfolgreich absolviert wurden.

4.5.2 Umsetzung der Struktur

Aus der oben beschriebenen Sequencingstrategie ergeben sich folgende Activities:

- Wurzel-Activity `"ORGANIZATION"` (Element `<organization>`)
- eine Activity `"EINLEITUNG"`
- eine Activity `"TEST"` mit den Aufgabenblöcken als Kind-Activities
- eine Activity `"ZUSAMMENFASSUNG"`
- jeweils eine Activity für jedes Kapitel mit den Lektionen als Kind-Activities

Diese werden entsprechend Abbildung 28 in der Content Organization definiert:




```
<organization identifier="ORGANIZATION">
 <item identifier="EINLEITUNG" identifierref="RES_EINLEITUNG"></item>
 <item identifier="TEST">
  <item identifier="AUFGABEN_01" identifierref="RES_AUFGABEN_01"></item>
  <item identifier="AUFGABEN_02" identifierref="RES_AUFGABEN_02"></item>
  <item identifier="AUFGABEN_03" identifierref="RES_AUFGABEN_03"></item>
  <item identifier="AUFGABEN_04" identifierref="RES_AUFGABEN_04"></item>
  <item identifier="AUFGABEN_05" identifierref="RES_AUFGABEN_05"></item>
  <item identifier="AUFGABEN_06" identifierref="RES_AUFGABEN_06"></item>
  <item identifier="AUFGABEN_07" identifierref="RES_AUFGABEN_07"></item>
  <item identifier="AUFGABEN_08" identifierref="RES_AUFGABEN_08"></item>
 </item>
 <item identifier="ZUSAMMENF" identifierref="RES_ZUSAMMENF"></item>
 <item identifier="KAPITEL_01">
  <item identifier="LEKTION_01" identifierref="RES_LEKTION_01"></item>
  <item identifier="LEKTION_02" identifierref="RES_LEKTION_02"></item>
  <item identifier="LEKTION_03" identifierref="RES_LEKTION_03"></item>
  <item identifier="LEKTION_04" identifierref="RES_LEKTION_04"></item>
 </item>
 <item identifier="KAPITEL_02">
  <item identifier="LEKTION_05" identifierref="RES_LEKTION_05"></item>
  <item identifier="LEKTION_06" identifierref="RES_LEKTION_06"></item>
  <item identifier="LEKTION_07" identifierref="RES_LEKTION_07"></item>
  <item identifier="LEKTION_08" identifierref="RES_LEKTION_08"></item>
 </item>
</organization>
```

Abbildung 28: Content Organization "Persönlicher Lernweg mit Pretest"

Wie bei den vorangegangenen Kursbeispielen werden auch hier die Ressourcen der Aufgaben und der Zusammenfassung als SCO definiert. Die Ressource der Einleitung ist ebenfalls ein SCO, da hier Elemente des Navigation Models verwendet werden und dies die Kommunikation mit dem LMS erfordert.

4.5.3 Sequencingangaben

Nachdem die Struktur erstellt wurde, wird nun die vorab beschriebene Sequencingstrategie umgesetzt. Die komplette Manifest-Datei zu diesem Beispiel befindet sich auf der beiliegenden CD.

Control Mode

Die folgenden Control Modes bewirken, dass das Menü erst angezeigt wird, wenn die Lektionen für den persönlichen Lernweg ermittelt wurden.

Activity	Strategie	Control Mode
ORGANIZATION	nach dem Pretest sind die Kapitel im Menü auswählbar	choice="true"
	mit den Navigationselementen des LMS kann durch die Inhalte geblättert werden	flow="true"
EINLEITUNG	während der Einleitung soll kein Menü angezeigt werden	choiceExit="false"
TEST	die Aufgaben können nur über Vorwärtsblättern ausgewählt werden	choice="false"
	es wird weiterhin kein Menü angezeigt	choiceExit="false"
TEST	mit den Navigationselementen des LMS kann durch die Aufgaben geblättert werden	flow="true"
	es kann innerhalb des Test nur vorwärts geblättert werden	forwardOnly="true"
AUFGABEN_xx	-	–
ZUSAMMENFASSUNG	-	–
KAPITEL_xx	alle Lektionen sind im Menü auswählbar	choice="true"
	mit den Navigationselementen des LMS kann durch die Lektionen geblättert werden	flow="true"
LEKTION_xx	-	–

Tabelle 50: Control Modes für den Kurs "Persönlicher Lernweg ..."

Sequencing Rules

Mit den hier verwendeten Sequencing Rules wird das Erscheinungsbild des Menüs gesteuert und gewährleistet, dass der Pretest nur einmal bearbeitet werden kann:

Activity	Strategie	Sequencing Rules
ORGANIZATION	-	–
EINLEITUNG	die Einleitung soll später nicht im Menü erscheinen	Pre-Condition-Rule: if completed then hiddenFromChoice
TEST	der Pretest soll später nicht im Menü erscheinen	Pre-Condition-Rule: if completed then hiddenFromChoice
	wenn der Test vollständig bearbeitet wurde, kann er nicht mehr ausgewählt werden	Pre-Condition-Rule: if completed then disabled
AUFGABEN_xx	-	–
ZUSAMMENFASSUNG	die Zusammenfassung steht erst im Menü, wenn sie das erste Mal angezeigt wird	Pre-Condition-Rule: if not attempted then hiddenFromChoice
KAPITEL_xx	wenn im persönlichen Lernweg keine Lektionen des Kapitels enthalten sind, erscheint das Kapitel nicht im Menü	Pre-Condition-Rule: if satisfied then hiddenFromChoice

Activity	Strategie	Sequencing Rules
LEKTION_xx	wenn der entsprechende Aufgaben-block bestanden wurde, wird die Lektion nicht im Menü angezeigt	Pre-Condition-Rule: `if satisfied then hiddenFromChoice`
	wenn der entsprechende Aufgaben-block bestanden wurde, wird die Lektion übersprungen	Pre-Condition-Rule: `if satisfied then skip`

Tabelle 51: Sequencing Rules für den Kurs "Persönlicher Lernweg ..."

Durch Rollup Rules wird für die Activities `"KAPITEL_01"` bzw. `"KAPITEL_02"` ermittelt, ob sie den Status "satisfied" erhalten (vgl. nächster Abschnitt). Dies ist nötig, um zu entscheiden, ob die Activity im Menü angezeigt wird (Pre-Condition-Rule `if satisfied then hiddenFromChoice`). Beim Test des Kurses mit dem SampleRTE von ADL wird die Activity jedoch auch angezeigt, wenn sie den Status "satisfied" hat. Die naheliegendste Erklärung dafür ist, dass dies ein Fehler des SampleRTE ist, da gleichartige Regeln in anderen Fällen funktionieren (z. B. Pre-Condition-Rule `if completed then hiddenFromChoice` für die Activity `"TEST"`, wobei der Status "completed" ebenfalls über Rollup Rules ermittelt wird).

Rollup Rules und Rollup Considerations

Die in Tabelle 52 aufgeführten Rollup Rules und Rollup Considerations gewährleisten, dass abhängig vom Erfolg beim Pretest und dem Fortschritt bei der Bearbeitung der Aufgaben und Lektionen der Erfolgsstatus und der Bearbeitungsfortschritt für den gesamten Kurs (Activitiy `"ORGANIZATION"`) ermittelt werden können:

Activity	Strategie	Rollup Rules und Rollup Considerations
ORGANIZATION	wenn der Test bestanden wurde, gilt auch der Kurs als bestanden	`if satisfied TRUE for all then satisfied`
	wenn der Test nicht bestanden wurde, gilt auch der Kurs als nicht bestanden	`if not satisfied TRUE for any then notSatisfied`
	wurden Einleitung, Test, Zusammenfassung und die Kapitel vollständig bearbeitet, ist der Kurs vollständig bearbeitet	`if completed TRUE for all then completed`
	wurden Einleitung, Test, Zusammenfassung oder die Kapitel noch nicht vollständig bearbeitet, ist der Kurs nicht vollständig bearbeitet	`if not completed TRUE for any then incomplete`
EINLEITUNG	der Status der Einleitung hat keinen Einfluss auf das Bestehen des Kurses	`rollupObjectiveSatisfied ="false"`
TEST	wenn mindestens 6 Aufgaben bestanden wurden, ist der Test bestanden	`if satisfied TRUE for atLeastCount 6 then satisfied`
	wenn mindestens 3 Aufgaben nicht bestanden wurden, ist der Test nicht bestanden	`if not satisfied TRUE for atLeastCount 3 then notSatisfied`
	wurden alle Aufgaben bearbeitet, ist der Test vollständig bearbeitet	`if attempted TRUE for all then completed`
	solange nicht alle Aufgaben bearbeitet wurden, ist der Test unvollständig	`if not attempted TRUE for any then incomplete`
AUFGABEN_xx	-	–

Activity	Strategie	Rollup Rules und Rollup Considerations
ZUSAMMENFASSUNG	der Status der Zusammenfassung hat keinen Einfluss auf das Bestehen des Kurses	`rollupObjectiveSatisfied ="false"`
KAPITEL_xx	der Status der Kapitel hat keinen Einfluss auf das Bestehen des Kurses	`rollupObjectiveSatisfied ="false"`
	wenn alle Aufgabenblöcke bestanden wurden, ist das Kapitel bestanden (notwendig für den Menüeintrag)	`if satisfied TRUE for all then satisfied`
	wenn mindestens ein Aufgabenblock nicht bestanden wurden, ist das Kapitel nicht bestanden (notwendig für den Menüeintrag)	`if not satisfied TRUE for any then notSatisfied`
	wurden alle Lektionen bearbeitet, ist das Kapitel vollständig bearbeitet	`if attempted TRUE for all then completed`
	solange nicht alle Lektionen bearbeitet wurden, ist das Kapitel unvollständig	`if not attempted TRUE for any then incomplete`
LEKTION_xx	nur die nicht bestandenen Lektionen müssen bearbeitet werden, damit das Kapitel vollständig ist	`requiredForCompleted ="ifNotSkipped"`

Tabelle 52: Rollup Rules und Rollup Considerations für den Kurs "Persönlicher Lernweg ..."

Objectives

Die Angaben zu den Objectives entsprechen den Ausführungen in Kapitel 4.3.3 auf Seite 83 und sollen hier nicht noch einmal erläutert werden.

4.5.4 Einbetten von Navigationselementen

In der Einleitungsseite dieses Beispielkurses befindet sich ein Link, welcher einen *Continue* Navigation Request auslöst, siehe Abbildung 29:

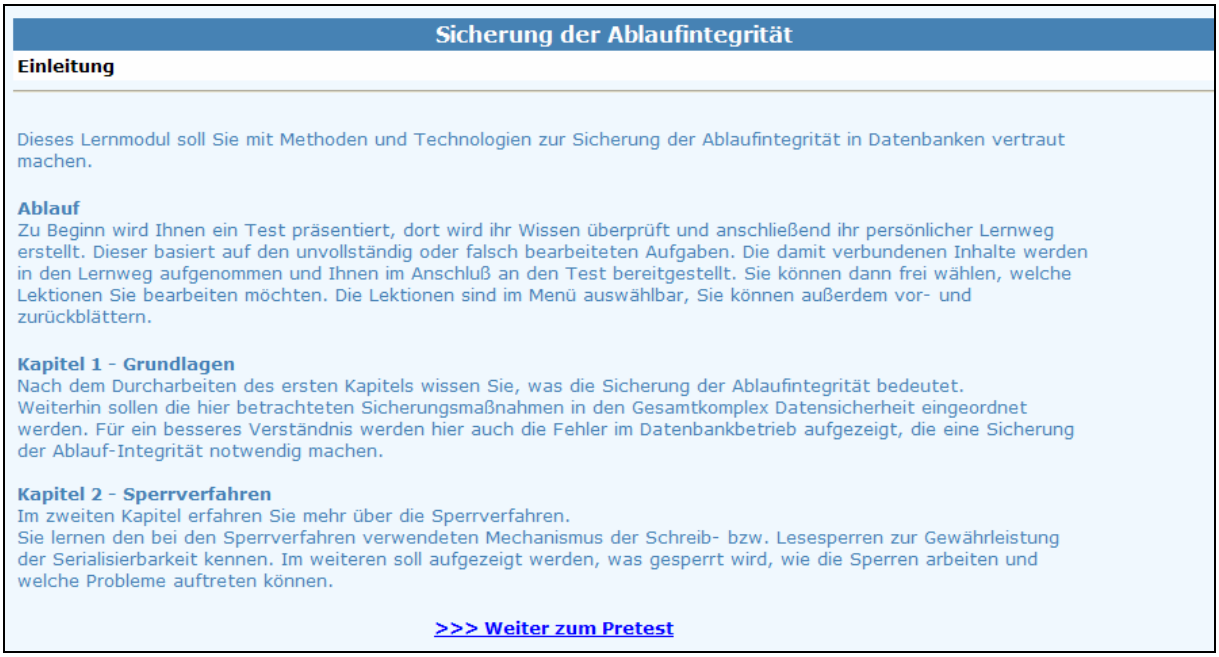


Abbildung 29: Einleitung des Kurses "Persönlicher Lernweg ..."

Mittels einer JavaScript-Funktion wird der Wert des Elements *adl.navigation.request* auf "continue" gesetzt und anschließend die Kommunikation mit dem LMS beendet, da der Navigation Request sonst nicht ausgelöst wird.

```
<script>
function doContinue()
{
        doSetValue("adl.nav.request","continue");
        doTerminate();
}
</script>
```

Codebeispiel 25: Übermittlung eines Navigation Requests aus einem SCO

Damit die Lernumgebung keine redundanten Navigationselemente enthält, wird der "Continue"-Button des LMS ausgeblendet. Codebeispiel 26 zeigt, welche Angaben in der Manifest-Datei dafür nötig sind:

```
<item identifier="EINLEITUNG" identifierref="RES_EINLEITUNG">
  <title>Einleitung</title>
  <adlnav:presentation>
    <adlnav:navigationInterface>
      <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
    </adlnav:navigationInterface>
  </adlnav:presentation>
</item>
```

Codebeispiel 26: Ausblenden von Navigationselementen im Kurs "Persönlicher Lernweg ..."

4.6 Zusammenfassung

Bei der Umsetzung der verschiedenen Sequencingstrategien wurde schnell klar, welches die wichtigsten und am häufigsten verwendeten Elemente des Sequencing Definition Models sind: zum einen die Control Modes und die Sequencing Rules, sie steuern den Inhalt und die Darstellung des Menüs und beeinflussen, auf welche Activities der Lernende zugreifen kann. Weiter zählen dazu die Rollup Rules, welche ermöglichen, dass Regeln, die vom Erfolgsstatus oder vom Abarbeitungsstatus abhängig sind, auf jeder Ebene des Activity Trees angewendet werden können. Allerdings wird die Anwendung von Rollup Rules schnell unübersichtlich, wenn die Activities sehr verschachtelt sind.
Eine wichtige Rolle spielen auch die Objectives, durch welche es erst möglich ist, die Bereitstellung von Activities von der Bearbeitung anderer Activities abhängig zu machen.

Die hier vorgestellten Lösungen sind nicht als ausschließlich anzusehen, sondern stellen lediglich eine Variante der Umsetzung dar. Auch sind die Möglichkeiten für die Anwendung des SCORM Sequencing and Navigation Books mit diesen drei Beispielen längst nicht ausgeschöpft. Interessant wäre zum Beispiel die Anwendung des Navigation Models in Form von in den SCOs eingebetteten Navigationselementen. In der Einleitung des Beispiels

"Persönlicher Lernweg mit Pretest" wurde dies in minimalem Umfang umgesetzt (vgl. 4.5.4), ausführlicher wurde das Navigation Model in dieser Arbeit nicht umgesetzt.

Diese drei Beispiele haben auch gezeigt, dass es durch die Trennung der Lerninhalte von der Struktur und den Sequencingangaben möglich ist, einmal erstellte Lernressourcen für verschiedene Kurse, mit gleichem Inhalt aber anderem Ablauf, zu nutzen. Ein Lernangebot kann durch die verwendete Sequencingstrategie individualisiert werden, die Ressourcen bleiben aber trotzdem wiederverwendbar.

5 Zusammenfassung und Ausblick

Hauptgegenstand dieser Arbeit war die Untersuchung des SCORM Sequencing, welches mit der Version SCORM 2004 erstmals in das Referenzmodell integriert wurde. Die Umsetzung des Sequencing erfordert einerseits die Implementierung von Modellen und Prozessen im Lernmanagementsystem und andererseits die Anwendung in den Lerninhalten. Diese Arbeit beschäftigt sich vorrangig mit der Anwendung des Sequencing bei der Erstellung von Lerninhalten und zeigt diese im Anschluss an die theoretische Ausarbeitung an verschiedenen Beispielkursen.

Zum Einstieg in die Thematik wurde ein Überblick über aktuelle Standardisierungsprojekte aus dem Bereich des E-Learning gegeben, deren Arbeitsergebnisse sich in der SCORM-Spezifikation wiederfinden. Es folgte die Beschreibung des Content Aggregation Models und des Run-Time Environments, wobei unter anderem die Änderungen gegenüber SCORM 1.2 herausgearbeitet wurden. Diese Änderungen betreffen vor allem die Metadaten, wo Elemente umstrukturiert bzw. umbenannt wurden und das RTE-Datenmodell, wo ebenfalls die Umbenennung zahlreicher Elemente stattfand und auch neue Elemente hinzugefügt wurden. Außerdem wurden die Namen der acht API-Funktionen verändert und neue Error-Codes hinzugefügt. Die Umbenennungen und Erweiterungen im CAM und RTE-Datenmodell verfolgen das Ziel, die Strukturierung der Modelle zu optimieren. Damit werden die Zusammenhänge zwischen den einzelnen Elementen besser verdeutlicht, was zu einer besseren Handhabung der Modelle für die Content-Entwickler führt. Speziell die Erweiterungen im RTE-Datenmodell lassen den Autoren einen größeren Spielraum für die Gestaltung dynamischer Kurselemente, da der Informationsaustausch zwischen LMS und Kurs einerseits ein größeres Datenvolumen ermöglicht und andererseits besser strukturiert werden kann.

Das dritte Kapitel ist das umfangreichste Kapitel der Arbeit und beschreibt die verschiedenen Modelle und Prozesse des SCORM Sequencing. Dabei wurde das Sequencing Definition Model besonders hervorgehoben, da es für den Content-Entwickler die Elemente bereitstellt, mit denen die gewünschte Sequencingstrategie umgesetzt werden kann. Mit Codebeispielen, Abbildungen und tabellarischen Übersichten wurde ein anschaulicher Überblick über die Möglichkeiten der Ablaufsteuerung gegeben.

Drei Beispielkurse, die jeweils eine andere Sequencingstrategie verfolgen, zeigten zum Abschluss, wie das Sequencing in der Praxis eingesetzt werden kann. Dabei wurde zu Beginn

die Sequencingstrategie ausformuliert und anschließend Schritt für Schritt mit den Elementen des Sequencing Definition Models umgesetzt.
Die praktische Anwendung zeigte, dass durch die Trennung der Lerninhalte von der Struktur und den Sequencingangaben die Wiederverwendung der Ressourcen in verschiedenen Kursen mit unterschiedlichem Ablauf möglich ist. Es wurde aber auch deutlich, dass bei umfangreichen und tiefer strukturierten Kursen die Anwendung des Sequencing recht kompliziert wird. Etwas Erleichterung bringt hier die Nutzung eines Editors für die Erstellung der Manifest-Datei. Zum jetzigen Zeitpunkt stehen nur sehr wenige Werkzeuge für die Umsetzung und den Test von Kursen nach SCORM 1.3.1 bereit. Auch die Unterstützung seitens der LMS fehlt noch. Diese Situation ist auf den neuen Sequencing-Teil zurückzuführen, welcher für die Hersteller von entsprechenden Modulen für LMS und Autorenwerkzeuge eine komplexe Implementierung dieser zusätzlichen Funktionalitäten erfordert.

Abschließend lässt sich sagen, dass die Nutzung von Standards bei der Entwicklung von Lerninhalten und Lernmanagementsystemen sinnvoll ist, da so die Interoperabilität und Wiederverwendbarkeit erhöht werden kann. SCORM ist einer der wenigen Standards, der sich im E-Learning-Bereich seitens der Entwickler einer breiten Unterstützung erfreut. Grund dafür ist, dass für die Implementierung (insbesondere für Version 1.2) ausreichend Dokumentationen und Werkzeuge bereitstehen und die damit erzeugten Kurse in einer großen Anzahl von LMS importierbar und ausführbar sind. Die Version 1.3.1 enthält viele Optimierungen und Erweiterungen des Content Aggregation Models und des RTE-Datenmodells. Durch das neu eingeführte Sequencing kann ein komplexes Kursdesign mit einer anspruchsvollen Ablaufsteuerung implementiert werden. Insgesamt gesehen führen diese neuen Funktionalitäten für den Content-Entwickler zu einem deutlich breiteren Handlungsspielraum bei der Gestaltung von Kursen, als es in der Version 1.2 der Fall war. Deshalb ist absehbar, dass sich SCORM und insbesondere die aktuelle Version 1.3.1 immer weiter durchsetzen wird.

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Codebeispiele

Abkürzungsverzeichnis

ADL	Advanced Distributed Learning
AICC	Aviation Industry CBT Committee
API	Application Programming Interface
ARIADNE	Alliance of Remote Instructional Authoring & Distribution Networks for Europe
CAM	Content Aggregation Model
CBT	Computer Based Training
CMI	Computer Managed Instruction
DOM	Document Object Model
ECMA	European Computer Manufacturer Association
HTML	Hypertext Markup Language
IEEE	Institute of Electronical and Electronics Engineers
IMS	Instructional Management Systems
LMS	Lernmanagementsystem
LTSC	Learning Technology Standards Committee
PIF	Package Interchange File
RTE	Run-Time Environment
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
XML	Extensible Markup Language

Literaturverzeichnis

Das Datum in Klammern bezeichnet bei Quellen aus dem Internet das Abrufdatum.

[AICC] *Aviation Industry CBT Committee.* http://www.aicc.org (2004-06-17)

[ARIADNE] *ARIADNE : Foundation for the European Knowledge Pool.* http://www.ariadne-eu.org (2004-06-17)

[Bauer, 2001] Bauer, R.; Philippi, T.: *Einstieg ins E-Learning. Die Zukunftschance für beruflichen und privaten Erfolg.* Nürnberg: Bildung und Wissen Verlag, 2001.

[Baumgartner, 2002] Baumgartner, P.; Häfele, H.; Maier-Häfele, K.: *E-Learning Praxishandbuch. Auswahl von Lernplattformen.* Innsbruck: Studienverlag, 2002.

[Bildungsportal] *Bildungsportal Sachsen.* http://www.bildungsportal-sachsen.de

[Collier, 2002] Collier, G.; Robson, R.: *e-Learning Interoperabilty Standards.* Sun Microsystems White Paper. 2002. http://www.sun.com/products-n-solutions/edu/elearning/eLearning_Interoperability_Standards_wp.pdf (2004-05-19)

[Gries, 2003] Gries, V.: *Nutzung von Standards bei der Entwicklung von e-Learning Content.* 2003. http://elearning.anova.de/de/fachw/pdf/Contententwicklung_Standards.pdf (2004-05-18)

[IEEE] *IEEE Learning Technology Standards Committee.* http:// ltsc.ieee.org (2004-06-17)

[IMS] *IMS Global Learning Consortium, Inc.* http://www.imsglobal.org (2004-06-17)

[IMS, 2003] *IMS Simple Sequencing Information and Behavior Model.* 2003. http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_infov1p0.html (2004-05-26)

[IMS, 2004] *IMS Meta-data Best Practice Guide for IEEE 1484.12.1-2002 Standard for Learning Object Metadata.* 2004. http://www.imsglobal.org/metadata/mdv1p3pd/imsmd_bestv1p3pd.html (2004-05-26)

[Kaiser, 2001] Kaiser, R.: *Analyse und Anwendung von Standards für E-Learning-Umgebungen unter besonderer Berücksichtigung des SCORM-Modells.* Diplomarbeit Hochschule für Technik und Wirtschaft Dresden, 2001.

[Kammer, 2003] Kammer, K.: *SCORM-konforme Lernobjekte mit Macromedia Flash MX – Konzeption und Implementierung eines Lernmoduls zum Thema „Heuristik“ für die Lernumgebung SIMLA*. Diplomarbeit Hochschule für Technik und Wirtschaft Dresden, 2003.

[Masie, 2003] The MASIE Center: *Making Sense of Learning Specifications & Standards: A Decision Maker's Guide to their Adoption (2nd edition).* 2003. http://www.masie.com/standards/s3_2nd_edition.pdf (2004-05-17)

[Neubauer, 2002] Neubauer, J.: *Praxistraining eLearning. Hilfe zur Selbsthilfe*. 2002. http://www.elearning-expo.de/head_navi/specials/0/ Praxistraining_eLearning.pdf (2004-05-19)

[SCORM, 2004a] *SCORM 2004 2nd Edition. Overview.* 2004. http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=992 (2004-07-22)

[SCORM, 2004b] *SCORM Content Aggregation Model Version 1.3.1.* 2004. http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=994 (2004-07-22)

[SCORM, 2004c] *SCORM Run-Time Environment Version 1.3.1.* 2004. http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=996 (2004-07-22)

[SCORM, 2004d] *SCORM Sequencing and Navigation Version 1.3.1.* 2004. http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=998 (2004-07-22)

[SCORM, 2004e] *SCORM Version 1.2 to 2004 Changes.* 2004. http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=1106 (2004-07-22)

[Wikipedia] *Wikipedia – Die freie Enzyklopädie.* http://de.wikipedia.org/wiki/Standard (2004-06-10)

Anhang A

Metadaten-Anwendungsprofile SCORM 1.3.1

M – Element ist obligatorisch (mandatory)

O – Element ist optional

Nr.	Element	Content Aggregation	Content Organization, Activity und SCO	Asset
1	<general>	O	M	M
1.1	<identifier>	O	M	M
1.1.1	<catalog>	O	O	O
1.1.2	<entry>	O	M	M
1.2	<title>	O	M	M
1.3	<language>	O	O	O
1.4	<description>	O	M	M
1.5	<keyword>	O	M	O
1.6	<coverage>	O	O	O
1.7	<structure>	O	O	O
1.8	<aggregationLevel>	O	O	O
2	<lifeCycle>	O	M	O
2.1	<version>	O	M	O
2.2	<status>	O	M	O
2.3	<contribute>	O	O	O
2.3.1	<role>	O	O	O
2.3.2	<entity>	O	O	O
2.3.3	<date>	O	O	O
3	<metaMetadata>	M	M	M
3.1	<identifier>	O	M	M
3.1.1	<catalog>	O	O	O
3.1.2	<entry>	O	M	M
3.2	<contribute>	O	O	O
3.2.1	<role>	O	O	O
3.2.2	<entity>	O	O	O
3.2.3	<date>	O	O	O
3.3	<metadataSchema>	M	M	M
3.4	<language>	O	O	O
4	<technical>	O	M	M
4.1	<format>	O	M	M
4.2	<size>	O	O	O
4.3	<location>	O	O	O
4.4	<requirement>	O	O	O
4.4.1	<orComposite>	O	O	O
4.4.1.1	<type>	O	O	O
4.4.1.2	<name>	O	O	O
4.4.1.3	<minimumVersion>	O	O	O
4.4.1.4	<maximumVersion>	O	O	O
4.5	<installationRemarks>	O	O	O
4.6	<otherPlatformRequirements>	O	O	O

Nr.	Element	Content Aggregation	Content Organization, Activity und SCO	Asset
4.7	<duration>	O	O	O
5	<educational>	O	O	O
5.1	<interactivityType>	O	O	O
5.2	<learningResourceType>	O	O	O
5.3	<interactivityLevel>	O	O	O
5.4	<semanticDensity>	O	O	O
5.5	<intendedEndUserRole>	O	O	O
5.6	<context>	O	O	O
5.7	<typicalAgeRange>	O	O	O
5.8	<difficulty>	O	O	O
5.9	<typicalLearningTime>	O	O	O
5.10	<description>	O	O	O
5.11	<language>	O	O	O
6	<rights>	O	M	M
6.1	<cost>	O	M	M
6.2	<copyrightAndOtherRestrictions>	O	M	M
6.3	<description>	O	O	O
7	<relation>	O	O	O
7.1	<kind>	O	O	O
7.2	<resource>	O	O	O
7.2.1	<identifier>	O	O	O
7.2.1.1	<catalog>	O	O	O
7.2.1.2	<entry>	O	O	O
7.2.2	<description>	O	O	O
8	<annotation>	O	O	O
8.1	<entity>	O	O	O
8.2	<date>	O	O	O
8.3	<description>	O	O	O
9	<classification>	O	O	O
9.1	<purpose>	O	O	O
9.2	<taxonPath>	O	O	O
9.2.1	<source>	O	O	O
9.2.2	<taxon>	O	O	O
9.2.2.1	<id>	O	O	O
9.2.2.2	<entry>	O	O	O
9.3	<description>	O	O	O
9.4	<keyword>	O	O	O

Tabelle nach [SCORM, 2004b]

Anhang B

Elemente des RTE-Datenmodells

Diese Tabelle enthält alle Elemente des RTE-Datenmodells in der Version 1.2 und in der Version 1.3.1. Sie zeigt, welche Elemente umbenannt, entfernt oder hinzugefügt wurden und es ist angegeben, welche Elemente in der Version 1.2 noch optional waren und welchen Zugriff das SCO auf das Element hat.

M – Element ist obligatorisch (mandatory) für das LMS

O – Element ist optional für das LMS

R – Read-Zugriff durch SCO

W – Write-Zugriff durch SCO

RTE 1.2	RTE 1.3.1	Bemerkungen
cmi.core._children (M,R)	-	Kategorie *core* wurde aus dem Datenmodel entfernt
cmi.core.student_id (M,R)	cmi.learner_id (M,R)	Änderung des Datentyps von CMIIdentifier auf long_identifier_type
cmi.core.student_name (M,R)	cmi.learner_name (M,R)	Änderung des Datentyps von CMIString255 auf localized_string_type
cmi.core.lesson_location (M,R,W)	cmi.location (M,R,W)	Änderung des Datentyps von CMIString255 auf characterstring
cmi.core.credit (M,R)	cmi.credit (M,R)	Änderung des Datentyps von CMIVocabulary auf state
cmi.core.lesson_status (M,R,W)	-	Element existiert noch im IEEE 1484.11.1 Datenmodell, soll aber in SCORM 2004 SCOs nicht mehr benutzt werden; LMS sollten dieses Element aber noch unterstützen
	cmi.success_status (M,R,W)	neues Element in SCORM 2004; ersetzt zusammen mit cmi.completion_status das Element cmi.core.lesson_status, Beschreibung im Kapitel 2.3.3
	cmi.completion_status (M,R,W)	neues Element in SCORM 2004; ersetzt zusammen mit cmi.success_status das Element cmi.core.lesson_status, Beschreibung im Kapitel 2.3.3
cmi.core.entry (M,R)	cmi.entry (M,R)	Änderung des Datentyps von CMIVocabulary auf state
cmi.core.score._children (M,R)	cmi.score._children (M,R)	Änderung des Datentyps von CMIString255 auf characterstring; Aktualisierung der Liste der Children-Elemente (neues Element *scaled*)
cmi.core.score.raw (M,R,W)	cmi.score.raw (M,R,W)	Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real(10,7)
cmi.core.score.max (O,R,W)	cmi.score.max (M,R,W)	Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real(10,7)
cmi.core.score.min (O,R,W)	cmi.score.min (M,R,W)	Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real(10,7)
	cmi.score.scaled (M,R,W)	neues Element in SCORM 2004, Beschreibung im Kapitel 2.3.3
cmi.core.total_time (M,R)	cmi.total_time (M,R)	Änderung des Datentyps von CMITimespan auf time(second,10,2)

RTE 1.2	RTE 1.3.1	Bemerkungen
cmi.core.lesson_mode (O,R)	cmi.mode (M,R)	Änderung des Datentyps von CMIVocabulary auf state
cmi.core.exit (M,W)	cmi.exit (M,W)	Änderung des Datentyps von CMIVocabulary auf state; der Wert *normal* wurde zum Vokabular hinzugefügt
cmi.core.session_time (M,W)	cmi.session_time (M,W)	Änderung des Datentyps von CMITimespan auf timeinterval(second,10,2)
cmi.suspend_data (M,R,W)	cmi.suspend_data (M,R,W)	Änderung des Datentyps von CMIString4096 auf characterstring
cmi.launch_data (M,R)	cmi.launch_data (M,R)	Änderung des Datentyps von CMIString4096 auf characterstring
cmi.comments (O,R,W)	cmi.comments_from_learner (M,R,W)	Unterelemente *comment*, *location* und *date_time* wurden hinzugefügt; der Inhalt des Elements wird jetzt als Array verwaltet
cmi.comments_from_lms (O,R)	cmi.comments_from_lms (M,R)	Unterelemente *comment*, *location* und *date_time* wurden hinzugefügt; der Inhalt des Elements wird jetzt als Array verwaltet
cmi.objectives._children (O,R)	cmi.objectives._children (M,R)	Änderung des Datentyps von CMIString255 auf characterstring; Aktualisierung der Liste der Children-Elemente (neue Elemente *success_status*, *completion_status* und *description*)
cmi.objectives._count (O,R)	cmi.objectives._count (M,R)	Änderung des Datentyps von CMIInteger auf integer
cmi.objectives.n.id (O,R,W)	cmi.objectives.n.id (M,R,W)	Änderung des Datentyps von CMIIdentifier auf long_identifier_type; sind in der Manifest-Datei Objectives für ein SCO definiert (`<imsss:objectives>` oder `<imsss:primaryObjective>`), sollten deren ID-Attribute für die Initialisierung des Datenelements genutzt werden
cmi.objectives.n.score. _children (O,R)	cmi.objectives.n.score. _children (M,R)	Änderung des Datentyps von CMIString255 auf characterstring; Aktualisierung der Liste der Children-Elemente (neues Element *scaled*)
cmi.objectives.n.score.raw (O,R,W)	cmi.objectives.n.score.raw (M,R,W)	Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real(10,7)
cmi.objectives.n.score.max (O,R,W)	cmi.objectives.n.score.max (M,R,W)	Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real(10,7)
cmi.objectives.n.score.min (O,R,W)	cmi.objectives.n.score.min (M,R,W)	Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real(10,7)
	cmi.objectives.n.score.scaled (M,R,W)	neues Element in SCORM 2004, Beschreibung im Kapitel 2.3.3
cmi.objectives.n.status (O,R,W)	-	Element existiert noch im IEEE 1484.11.1 Datenmodell, soll aber in SCORM 2004 SCOs nicht mehr benutzt werden; LMS sollten dieses Element aber noch unterstützen
	cmi.objectives.n. success_status (M,R,W)	neues Element in SCORM 2004; ersetzt zusammen mit cmi.objectives.n.completion_status das Element cmi.objectives.n.status, Beschreibung im Kapitel 2.3.3
	cmi.objectives.n. completion_status (M,R,W)	neues Element in SCORM 2004; ersetzt zusammen mit cmi.objectives.n.success_status das Element cmi.objectives.n.status, Beschreibung im Kapitel 2.3.3
	cmi.objectives.n.description (M,R,W)	neues Element in SCORM 2004, Beschreibung im Kapitel 2.3.3
cmi.student_data._children (O,R)	-	Kategorie *student_data* wurde aus dem Datenmodel entfernt

RTE 1.2	RTE 1.3.1	Bemerkungen
cmi.student_data.mastery _score (O,R)	cmi.scaled_passing_score (M,R)	Änderung des Datentyps von CMIDecimal auf real(10,7); Initialisierung des Datenelements mit dem Wert von `<imsss:minNormalizedMeasure>` des `<imsss:primaryObjective>`-Elements des SCOs (falls angegeben)
cmi.student_data. max_time_allowed (O,R)	cmi.max_time_allowed (M,R)	Änderung des Datentyps von CMITimespan auf timeinterval(second,10,2); Initialisierung des Datenelements mit dem Wert von `<imsss:attemptAbsoluteDurationLimit> (falls angegeben)`
cmi.student_data. time_limit_action (O,R)	cmi.time_limit_action (M,R)	Änderung des Datentyps von CMIVocabulary auf state
cmi.student_preference. _children (O,R)	cmi.learner_preference. _children (M,R)	Änderung des Datentyps von CMIString255 auf characterstring; Aktualisierung der Liste der Children-Elemente mit den geänderten Elementnamen
cmi.student_preference. audio (O,R,W)	cmi.learner_preference. audio_level (M,R,W)	Änderung des Datentyps von CMISInteger auf real(10,7)
cmi.student_preference. language (O,R,W)	cmi.learner_preference. language (M,R,W)	Änderung des Datentyps von CMIString255 auf language_type
cmi.student_preference. speed (O,R,W)	cmi.learner_preference. delivery_speed (M,R,W)	Änderung des Datentyps von CMISInteger auf real(10,7)
cmi.student_preference.text (O,R,W)	cmi.learner_preference. audio_captioning (M,R,W)	Änderung des Datentyps von CMISInteger auf state mit den Werten *off*, *no_change* und *on*
cmi.interactions._children (O,R)	cmi.interactions._children (M,R)	Änderung des Datentyps von CMIString255 auf characterstring; Aktualisierung der Liste der Children-Elemente mit den geänderten Elementnamen
cmi.interactions._count (O,R)	cmi.interactions._count (M,R)	Änderung des Datentyps von CMIInteger auf integer
cmi.interactions.n.id (O,W)	cmi.interactions.n.id (M,R,W)	Änderung des Datentyps von CMIIdentifier auf long_identifier_type
cmi.interactions.n.objectives ._count (O,R)	cmi.interactions.n. objectives._count (M,R)	Änderung des Datentyps von CMIInteger auf integer
cmi.interactions.n. objectives.n.id (O,W)	cmi.interactions.n. objectves.n.id (M,R,W)	Änderung des Datentyps von CMIIdentifier auf long_identifier_type
cmi.interactions.n. time (O,W)	cmi.interactions.n. timestamp (M,R,W)	Änderung des Datentyps von CMITime auf time(second,10,2)
cmi.interactions.n.type (O,W)	cmi.interactions.n.type (M,R,W)	Änderung des Datentyps von CMISInteger auf state mit den Werten *true_false*, *multiple_choice*, *fill_in*, *long_fill_in*, *matching*, *performance*, *sequencing*, *likert*, *numeric* und *other*
cmi.interactions.n. correct_responses._count (O,R)	cmi.interactions.n. correct_responses._count (M,R)	Änderung des Datentyps von CMIInteger auf integer
cmi.interactions.n. correct_responses.n.pattern (O,W)	cmi.interactions.n. correct_responses.n.pattern (M,R,W)	kein allgemeiner Datentyp mehr festgelegt, jeder Typ (entsprechend cmi.interaction.n.type) hat separate Festlegungen bezüglich Datentyp und Format
cmi.interactions.n.weighting (O,W)	cmi.interactions.n.weighting (M,R,W)	Änderung des Datentyps von CMIDecimal auf real(10,7)
cmi.interactions.n. student_response (O,W)	cmi.interactions.n. learner_response (M,R,W)	kein allgemeiner Datentyp mehr festgelegt, jeder Typ (entsprechend cmi.interaction.n.type) hat separate Festlegungen bezüglich Datentyp und Format

RTE 1.2	**RTE 1.3.1**	**Bemerkungen**
cmi.interactions.n.result (O,W)	cmi.interactions.n.result (M,R,W)	Änderung des Datentyps von CMISInteger auf state mit den Werten *correct*, *incorrect*, *unanticipated*, *neutral* und *real(10,7)*
cmi.interactions.n.latency (O,W)	cmi.interactions.n.latency (M,R,W)	Änderung des Datentyps von CMITimespan auf timeinterval(second,10,2)
	cmi.interactions.n. description (M,R,W)	neues Element in SCORM 2004, Beschreibung im Kapitel 2.3.3
cmi._version (O,R)	cmi._version (M,R)	repräsentiert die Version des Datenmodells, sollte den Wert "1.0" enthalten
	cmi.completion_threshold (M,R)	neues Element in SCORM 2004, Beschreibung im Kapitel 2.3.3
	cmi.progress_measure (M,R,W)	neues Element in SCORM 2004, Beschreibung im Kapitel 2.3.3

Anhang C

Navigation Requests, Termination Requests und Sequencing Requests

Navigation Request	Aktion
Start	Wenn *Current Activity* undefiniert ist, wird ein *Start* Sequencing Request weitergegeben.
Resume All	Wenn *Current Activity* undefiniert und *Suspended Activity* definiert ist, wird ein *Resume All* Sequencing Request weitergegeben.
Continue	Wenn *Activity is Active* für die *Current Activity* "true" ist, wird ein *Exit* Termination Request und ein *Continue* Sequencing Request weitergegeben.
Previous	Wenn *Activity is Active* für die *Current Activity* "true" ist, wird ein *Exit* Termination Request und ein *Previous* Sequencing Request weitergegeben.
Forward	Dieser Navigation Request ist in dieser SCORM-Version nicht definiert.
Backward	Dieser Navigation Request ist in dieser SCORM-Version nicht definiert.
Choice	Wenn *Activity is Active* für die *Current Activity* "true" ist, wird ein *Exit* Termination Request und ein *Choice* Sequencing Request weitergegeben.
Exit	Es wird ein *Exit* Termination Request und ein *Exit* Sequencing Request weitergegeben. Der Zugriff auf die *Current Activity* wird beendet.
Exit All	Es wird ein *Exit All* Termination Request und ein *Exit* Sequencing Request weitergegeben.
Suspend All	Es wird ein *Suspend All* Termination Request und ein *Exit* Sequencing Request weitergegeben. Der Zugriff auf die *Current Activity* und alle Vorfahren wird unterbrochen, gilt aber nicht als beendet. Der Zugriff auf die Activity kann später wieder aufgenommen werden (dies zählt nicht als neuer Zugriff).
Abandon	Es wird ein *Abandon* Termination Request und ein *Exit* Sequencing Request weitergegeben. Der Zugriff auf die *Current Activity* wird beendet, kann aber nicht wieder aufgenommen werden, der Zugriff wird "aufgegeben".
Abandon All	Es wird ein *Abandon All* Termination Request und ein *Exit* Sequencing Request weitergegeben. Der Zugriff auf die *Current Activity* und alle Vorfahren wird beendet, kann aber nicht wieder aufgenommen werden, der Zugriff wird "aufgegeben".

Tabelle 1: Navigation Requests, nach [SCORM, 2004d]

Termination Request	Aktion
Exit	Der Zugriff auf die *Current Activity* wird beendet.
Exit All	Der Zugriff auf alle aktiven Activities wird beendet.
Suspend All	Der Zugriff auf alle aktiven Activities wird beendet und der Zugriff auf die *Current Activity* kann später wieder aufgenommen werden.
Abandon	Der Zugriff auf die *Current Activity* wird beendet und kann nicht wieder aufgenommen werden.
Abandon All	Der Zugriff auf alle aktiven Activities wird beendet und kann später nicht wieder aufgenommen werden

Tabelle 2: Termination Requests, nach [SCORM, 2004d]

Sequencing Request	**Aktion**
Start	Der *Start Sequencing Request Subprocess* wird aufgerufen, dieser startet eine neue Sequencing Session an der Wurzel des Activity Trees.
Resume All	Der *Resume All Sequencing Request Subprocess* wird aufgerufen, dieser nimmt den unterbrochenen Zugriff bei der *Suspended Activity* wieder auf.
Continue	Der *Continue Sequencing Request Subprocess* wird aufgerufen,dieser ermittelt die nächste Activity.
Previous	Der *Previous Sequencing Request Subprocess* wird aufgerufen, dieser ermittelt die vorhergehende Activity.
Choice	Der *Choice Sequencing Request Subprocess* wird aufgerufen und stellt die ausgewählte Activity bereit.
Retry	Der *Retry Sequencing Request Subprocess* wird aufgerufen, dieser beginnt einen neuen Zugriff auf die C*urrent Activity*.
Exit	Der *Exit Sequencing Request Subprocess* wird gestartet, falls die *Current Activity* die Wurzel des Activity Trees ist, wird die Sequencing Session beendet.

Tabelle 3: Sequencing Requests, nach [SCORM, 2004d]

Selbständigkeitserklärung

Ich versichere, dass ich die Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Dresden, 23.12.2004

Anne-Kathrin Gericke