Hochschule für Technik und Wirtschaft Dresden (FH)

Fachbereich Informatik/Mathematik

Diplomarbeit

im Studiengang Medieninformatik

Thema: *Entwicklung einer benutzerfreundlichen Anwendung zur Generierung von Navigationsstrukturen für E-Learning-Kurse nach der SCORM 1.3 Spezifikation*

Eingereicht von: Melanie Pietzsch

Eingereicht am: 12.09.2005

Betreuer: Frau Prof. Dr. Teresa Merino
Hochschule für Technik und Wirtschaft Dresden

Herr Dr. Jürgen Haufe
Fraunhofer IIS/EAS Dresden

Inhaltsverzeichnis

Einleitung

Das Internet gewinnt als multimediale Wissensplattform immer mehr an Bedeutung. So kann Wissen in Form von E-Learning-Kursen gewinnbringend für die Aus- und Weiterbildung eingesetzt werden. Das Lernen über das Internet ist flexibel, kostengünstig und besonders in Verbindung mit einem hohen Praxisbezug sowie individuellen Kursabläufen sehr effizient. Ziel des Autors sollte dabei sein, didaktisch hochwertige multimediale Lernanwendungen möglichst effektiv umzusetzen. Für die zeit- und aufwandsreduzierte Kurs-Erstellung spielt die Einhaltung von E-Learning-Standards eine entscheidende Rolle. Diese Standards erhöhen die Interoperabilität sowie die Austausch- und Wiederverwendbarkeit von Lerninhalten. Zwar gibt es derzeit noch keinen internationalen Standard, aber mit dem Sharable Content Reference Model (SCORM) wird eine Spezifikation zur Verfügung gestellt, die mehrere Standards und Spezifikationen in einem Referenzmodel integriert.

Auch die Außenstelle Entwurfsautomatisierung (EAS) des Fraunhofer-Instituts für Integrierte Schaltungen (IIS) in Dresden stellt ihr aktuelles und innovatives Spezialwissen in Form von E-Learning-Kursen für die Aus- und Weiterbildung zur Verfügung. Veröffentlicht werden die Anwendungen auf einer SCORM-konformen Lernplattform.
Als problematisch hat sich dabei erwiesen, dass die Kurs-Entwickler am Institut wissenschaftliche Mitarbeiter und keine ausgebildeten E-Learning-Autoren sind. Bei der Erstellung von Lernanwendungen nach der SCORM 1.2 Spezifikation muss daher auf unterstützende Softwareprodukte wie *HTML*- und SCORM-Editoren zurückgegriffen werden. Für die aktuelle SCORM Version 1.3 stehen derzeit keine Werkzeuge zur Verfügung die eine intuitive Kurserstellung ohne tiefere Kenntnisse über die Spezifikation ermöglichen. Dennoch besteht für die neuen Funktionalitäten, im Besonderen die Generierung individueller Lernpfade, großes Interesse.

Ziel dieser Arbeit ist es, eine Anwendung zu konzipieren und prototypisch umzusetzen, die es ermöglicht, E-Learning-Kurse nach der aktuellen SCORM 1.3 Spezifikation zu entwerfen. Die Anwendung soll besonders intuitiv und ohne Kenntnisse der zugrunde liegenden Spezifikation bedienbar sein. Dafür sollen zunächst die Möglichkeiten von

SCORM 1.3 ausgewertet werden. Darauf aufbauend sind Kursstrukturen zu entwickeln, die im Speziellen den didaktischen E-Learning-Anforderungen des Instituts entsprechen.

Im ersten Kapitel wird ein kurzer Überblick über die Entwicklung von E-Learning an der Fraunhofer Gesellschaft und im Speziellen des Fraunhofer IIS/EAS in Dresden gegeben. Auf zwei Projekte des Institutes wird dabei näher eingegangen.

Das zweite Kapitel befasst sich mit der aktuellen SCORM-Spezifikation. Wichtige Begriffe werden definiert und es folgt eine Gegenüberstellung mit der Vorgängerversion SCORM 1.2. Abschließend wird die Bedeutung der Spezifikation für das Fraunhofer IIS/EAS dargestellt.

Die Erstellung SCORM konformer E-Learning-Kurse wird im dritten Kapitel beschrieben. Nach der Erläuterung des allgemeinen Ablaufes werden verschiedene Werkzeuge vorgestellt, welche die Entwicklung von SCORM-Kursen unterstützen. Besonders ausführlich wird dabei auf die Verwendung des Reload Editor 2004 eingegangen, da er als derzeit einziges Programm für die Erstellung SCORM 1.3 konformer Kurse für das Institut in Frage kommt.

Auf Grund der in Kapitel drei gewonnenen Erkenntnisse, soll ein Konzept für die Entwicklung einer benutzerfreundlichen Anwendung zur Generierung von E-Learning-Kursen nach der SCORM 1.3 Spezifikation entwickelt werden. Im vierten Kapitel wird dieses Konzept detailliert vorgestellt. Hierbei werden die konkrete Aufgabenstellung, die nötigen Voruntersuchungen und Vorbereitungen sowie die Funktionalität der Anwendung und des zu implementierenden Prototyps beschrieben.

Im fünften Kapitel wird die prototypische Implementierung der Anwendung dargestellt. Es wird zunächst auf die verwendete Technologie, Softwarearchitektur und Softwareentwicklung eingegangen. Ein Anwendungsbeispiel soll die intuitive Benutzung des Programms veranschaulichen.

1 E-Learning der Fraunhofer-Gesellschaft

Einführend soll dieses Kapitel einen Überblick über den aktuellen Stand von E-Learning an der Fraunhofer-Gesellschaft und im Speziellen des Fraunhofer-Instituts (FhI) für Integrierte Schaltungen (IIS), Außenstelle Entwurfsautomatisierung (EAS) Dresden geben. Ausgewählte Projekte des Fraunhofer IIS/EAS Dresden, werden vorgestellt.

1.1 Überblick

In rund 80 Forschungseinrichtungen der Fraunhofer-Gesellschaft mit etwa 12.500 Mitarbeitern wird fortwährend innovatives und praxisorientiertes Wissen auf vielen Forschungs- und Entwicklungsfeldern einer modernen Industriegesellschaft aufgebaut. Das trifft insbesondere für die Informations- und Kommunikationstechnik und –technologie, sowie Mikroelektronik, Energie, Produktion, Verkehr und Umwelt zu.
Dieses zunehmend nachgefragte Wissen soll in Form von E-Learning-Kursen für die Aus- und Weiterbildung zur Verfügung gestellt werden. Die Lernprodukte ersetzen dabei nicht das an Universitäten gewonnene Basiswissen, sondern vermitteln darauf aufbauend aktuelles und anwendungsorientiertes Spezialwissen.
Mit dem größer werdenden Angebot steigt auch das Interesse der Initiatoren und Entwickler multimedialer Wissensprodukte an adäquaten Technologien und Umsetzungsmöglichkeiten. Daher beschäftigt sich die Fraunhofer-Gesellschaft als vornehmlicher Content Provider zunehmend auch mit der Entwicklung innovativer Werkzeuge und Lernplattformen für den optimalen Einsatz von E-Learning Produkten.
Für die Vermarktung dieses Angebotes wurde 2002 die Abteilung *eLearning, Aus- und Weiterbildung* eingerichtet. Sie betreut die Institute und externe Kunden bei der Erstellung von Wissensprodukten. Unter dem Label *eQualification* werden die multimedialen Lernanwendungen und E-Learning Technologien von derzeit 17 Fraunhofer-Instituten auf dem Internet-Portal [1] angeboten. Außerdem besteht die Möglichkeit sich auf einer Lernplattform anzumelden und dort an über 60 verfügbaren Seminaren teilzunehmen. Tabelle 1-1 stellt einen Teil der vorhandenen Kurse vor.

Fachgebiet	Kurs	Inhalt	Zielgruppe	FhI
Druck und Medien	Job Messaging Format JMF und JDF Work-flows	Nutzen und Einsatzmöglichkeiten von JMF und JDF in der grafischen Industrie	Entscheider im grafischen Gewerbe	Graphische Datenverarbeitung (IGD)
	JDF Open Source und Schema	Überblick über das JDF Open Source Projekt von CIP4	Entwickler JDF basierter Produkte	Grafische Datenverarbeitung (IGD)
E-Learning	Erfolgreich WBTs erstellen	Entwicklungszyklus für strukturiertes, didaktisch sinnvolles und benutzerfreundliches Erstellen von WBTs	Fachleute aus Industrie, Wissenschaft, Verwaltung, die WBTs erstellen möchten	Integrierte Publikations- und Informationssysteme (IPSI)
Elektrotechnik	Entwurf digitaler elektronischer Systeme	Spezifikation, Verifikation und Entwurf digitaler elektroni-scher Schaltkreise und Systeme	Entwicklungsingenieure, technische Führungskräfte, Studenten, Berufsanfänger	IIS/EAS in Dresden
	EMV Trainingssystem	Einstieg in die Problematik der Elektromagnetischen Verträglichkeit (EMV)	Entwurfsingenieure, Berufsanfänger und Studenten	Institut für Zuverlässigkeit und Mikrointegration (IZM)
Handwerk, Industrie, Technik allgemein	Vorbereitungskurs zum DVS-EWF-Kleb-fachkraft-Lehrgang	Vermittlung der physikalischen und chemischen Grundlagen des Klebens	Facharbeiter/in und Industriemeister/in der Fachrichtung Metall, Kunststoff und Kautschuk, Chemielaboranten /innen, Chemotechniker/in, Anwendungstechniker/in	Fertigungstechnik und Angewandte Materialforschung (IFAM), Klebtechnisches Zentrum (KTZ)
Informationstechnologie	Sicherheit im elektronischen Geschäftsverkehr	Überblick und praktische Tipps zu Problemen der Sicherheit am Computer und im Umgang mit dem Internet	Benutzer von Arbeitsplatzrechnern, Nutzer von Internetanwendungen	Institut für Sichere Informationstechnologie (SIT)
	UML interaktiv für Entwurfsingenieure	Einführung in UML	Entwurfsingenieure in der Software-Industrie	Institut für Experimentelles Software Engineering (IESE)
Maschinen und Anlagenbau	DIN-gerechte CNC-Programmierung	Erstellung von CNC-Programmen	Technische Fachkräfte (Facharbeiter)	Produktionsanlagen und Konstruktionstechnik (IPK)

Tabelle 1-1: Kurse der Fraunhofer-Gesellschaft

Da sich die Fraunhofer-Gesellschaft zu einem Großteil (ca. 60%) von Industrieaufträgen finanziert, ist eine Präsenz auf Messen und Kongressen unumgänglich. Hier können zukunftsrelevante Forschungsprojekte präsentiert und intensive Kontakte zu Kunden und potentiellen Partnern gepflegt und aufgebaut werden.

So stellten Anfang des Jahres neun verschiedene Institute ihre aktuellen E-Learning-Projekte und Konzepte auf der *LEARNTEC (europäischer Kongress und Fachmesse für Bildungs- und Informationstechnologie)* 2005 in Karlsruhe vor.

Darunter das Projekt *Concert Chat* des Fraunhofer-Instituts für Integrierte Publikations- und Informationssysteme (IPSI), welches den Einsatz von Internet- und Intranet-Chat auf professioneller Ebene und somit auch für kooperative Lernübungen ermöglichen soll.

Einen entscheidenden Vorteil verschafft die übersichtliche Strukturierung und Protokollierung der Beiträge, was sowohl eine synchrone als auch eine asynchrone Kommunikation und Bearbeitung ermöglicht. Das Diskutieren komplexer Sachverhalte wird durch grafische Bezüge zwischen Bildern, Dokumenten und Nachrichten erleichtert. Ein integriertes Autoren-Tool ermöglicht die Erstellung von eigenen Gruppenübungen und Rollenspielen.

Der Einsatz des Concert Chat in Lernmanagementsystemen ermöglicht so eine intensive Betreuung der Studierenden und verbessert die Kommunikation untereinander. Abbildung 1-1 zeigt einen Screenshot der Software Concert Chat.

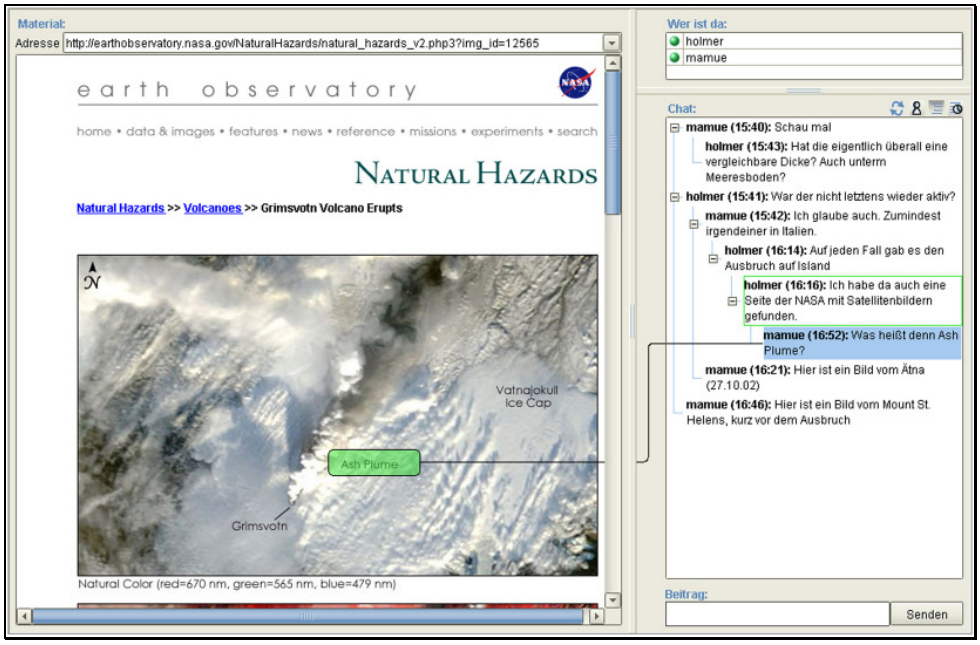


Abbildung 1-1 Concert Chat Screenshot

Das Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO) hat mit der Entwicklung von *VITERO (Virtual Team Room)* einen virtuellen Schulungsraum geschaffen. Besonderer Wert wurde auf eine einfache Bedienbarkeit und eine hohe

Kommunikationseffizienz gelegt. Entstanden ist eine neue Dimension virtuellen Lernens, in der ähnlich wie in der Realität agiert und Wissen ausgetauscht werden kann.
Auch die Mobilität beim Lernen gewinnt immer mehr an Bedeutung. So wurde von dem Fraunhofer-Institut für angewandte Informationstechnik (FIT) ein intelligentes Content Service System entwickelt, mit dem Aus- und Weiterbildung nun auch mit Hilfe von *PDAs (Personal Digital Assistant)* und Mobiltelefonen ermöglicht wird.

1.2 E-Learning am Fraunhofer IIS/EAS Dresden

Das Fraunhofer IIS/EAS Dresden führt Forschungs- und Entwicklungsarbeiten für den rechnergestützten Entwurf von elektronischen und heterogenen Systemen durch.
Es werden Methoden und Werkzeuge für Modellierung, Simulation, Synthese und Optimierung, Verifikation und Testgenerierung entwickelt. Diese Werkzeuge werden in applikationsspezifische Entwurfsabläufe integriert oder dienen der Ergänzung vorhandener Entwurfssysteme. Darüber hinaus werden prototypisch Hardware-Software-Systeme erstellt, welche zurzeit hauptsächlich in der Telekommunikations-technik und im digitalen Rundfunk eingesetzt werden.
Die auf diesen Arbeitsgebieten erzielten Forschungs- und Entwicklungsergebnisse sind außerordentlich innovativ und daher im wachsenden Maße für die betriebliche Weiterbildung von Interesse. Das so aufgebaute Wissen soll in Form von E-Learning-Kursen aufbereitet werden, und damit eine weitgehend orts- und zeitunabhängige sowie kostensparende Aneignung zu ermöglichen.
Passend zur Fraunhofer-Mission des Technologie- und Wissenstransfers in die Wirtschaft, wird bei Aus- und Weiterbildungsvorhaben der Fraunhofer-Gesellschaft Wert darauf gelegt, die theoretische Wissensvermittlung durch einen hohen Praxisanteil zu unterstützen. Ermöglicht wird dies unter anderem durch die Integration von Online-Werkzeugen wie Simulatoren, mit denen der Lernende sein neu erworbenes Wissen anhand von praxisorientierten Experimenten überprüfen und anwenden kann. Damit unterscheiden sich die Wissensprodukte des EAS maßgeblich von üblichen E-Learning-Anwendungen. Diese veranschaulichen zwar komplexe Lerninhalte durch multimediale Interaktionen, aber sie ermöglichen in der Regel nicht das Sammeln eigener Erfahrungen, welche für die erfolgreiche Aufnahme anwendungsbereiten Wissens entscheidend sind.

Die Autoren sind ausschließlich Wissenschaftler des technischen Bereiches, was zu einem Defizit bei der medientechnischen und mediendidaktischen Aufbereitung der Lernanwendungen führt. Um dem entgegen zu wirken, ist eine Kooperation mit Studenten und Mitarbeitern von Universitäten sowie Fachhochschulen mit der nötigen Ausbildung unverzichtbar.

Die Anwendungen sind plattformunabhängig und können so problemlos in die Lernplattform der Fraunhofer-Gesellschaft und in die Systeme der Kunden integriert werden. Mit Hilfe von Pilotanwendern werden die entstandenen Kurse regelmäßig getestet und weiterentwickelt.

Projekt	Inhalt	Zielgruppe	Partner	Finanzierung
DynLAB	Lehrgang zur Dynamik multi-disziplinärer und gesteuerter Systeme, durchgeführt in einem virtuellen Labor	Studenten, Ingenieure, Lehrer	TU Prague, CZ	TU Prague, CZ EU
LIMA	Multimediale Lernumgebung bestehend aus drei Kursen: Vermittlung von hoch qualifizier-tem Wissen im Bereich der Mikroelektronik, praxisnahe Designbeispiele	Schaltungs-entwickler, Wissenschaft-ler, Studenten	U Linz, A FhI IZM, D Nokia, SF Infineon, D Zuken, D, UK s.team, D CISC, A EADS Matra, F DICE, A	IIS/EAS EU
MILEON	Multimediale Lernumgebung bestehend aus fünf Kursen: Vermittlung von Basis- und Spezialwissen zu Funktion, Entwurf und Fertigung von elektronischen Systemen und Mikrosystemen	Entwicklungs-ingenieure, technische Führungskräf-te	TU Dresden	IIS/EAS Land Sachsen

Tabelle 1-2: EAS E-Learning-Kurse

Unter diesen Gesichtspunkten ist bereits eine Reihe an E-Learning Produkte entstanden. Dabei müssen reine Content-Produkte die der Aneignung von Lerninhalten dienen und E-Learning Technologien für eine optimale Erstellung dieser Wissensprodukte unterschieden werden. In den Tabelle 1-2 und 1-3 werden diese Projekte vorgestellt werden. Anzumerken ist, dass dabei meist mit Universitäten und Vertragspartnern der freien Wirtschaft zusammengearbeitet wird und die Finanzierung überwiegend durch öffentliche Gelder unterstützt bzw. übernommen wird.

Projekt	Inhalt	Zielgruppe	Partner	Finanzierung
MOSCITO Tool Integrator	Programm zur Integration von interaktiven Werkzeugen in Onlinekurse	Hersteller von Inhalten und Lernumgebungen für E-Learning		IIS/EAS
VILAB	Das Virtuelle Labor bietet eine Testumgebung für die Simulation von integrierten Schaltkreisen	Hersteller von Inhalten und Lernumgebungen für E-Learning	TU Tallinn, EST Warzaw UT, PL IET Bratislava, SK Slovak UT, SK TU Darmstadt, D U Linkoeping, S	IIS/EAS EU

Tabelle 1-3: EAS E-Learning Technologien

1.3 MOSCITO Tool Integrator

In Zusammenarbeit mit dem Institut für Informatik in Bratislava und der Technischen Universität von Tallinn, entwickelte das Fraunhofer IIS/EAS das Programmsystem MOSCITO.

Diese leistungsfähige Integrationsplattform wurde für die internetbasierte Nutzung von Programmen und Entwurfswerkzeugen konzipiert. Unter dem Namen MOSCITO Tool Integrator, wurde dieses Projekt weiterentwickelt und um Komponenten für die Anbindung an Online-Lernsysteme erweitert.

Mit diesem Tool können Programme wie Simulatoren vom Server des Lernportals gestartet und ausgeführt werden. Der Lernende nimmt dabei die Interaktionen über eine, dem Lernprogramm grafisch angepasste Schnittstelle vor. So sind weder die benötigten Werkzeuge auf dem Rechner zu installieren, noch müssen diese bedient werden können. Auch der einheitliche Datenaustausch zwischen verschiedenen Anwendungen wird von MOSCITO unterstützt.

1.4 Multimediale Lernplattform MILEON

MILEON steht für *Microsystems Technology and Microelectronics Lectures Online* und ist ein im Jahr 1999 gestartetes Gemeinschaftsprojekt des Instituts für Halbleiter- und Mikrosystemtechnik (IHM) der Technischen Universität (TU) Dresden und dem Fraunhofer-Institut IIS/EAS Dresden.

Dieses E-Learning-Projekt besteht aus derzeit fünf Lernmodulen die der Weiterbildung und Ergänzung von Lehrveranstaltungen auf dem Gebiet der Mikrosystemtechnik und

Mikroelektronik dienen. Die Bausteine vermitteln Grundlagen und aktuelles Spezialwissen zu Entwurf, Technologie und Fertigung von elektronischen Systemen und Mikrosystemen. Durch den Einsatz von realen Entwurfswerkzeugen wird eine besonders praxisnahe und experimentelle Arbeitsumgebung für den Wissensaufbau geschaffen. Einen Überblick über die einzelnen Bausteine gibt die nachstehende Tabelle geben.

Lernmodul	Inhalt
Grundlagen der Mikrosystemtechnik	Lehrgang zur Dynamik multidisziplinärer und gesteuerter Systeme, durchgeführt in einem virtuellen Labor
Entwurf von mikromechanischen Sensoren und Aktoren (MEMS)	Vermittlung theoretischer Grundlagen über Entwurf, Modellbildung und Simulation von Micro-Electronical-Mechanical-Systems (MEMS); Einführung in die Hardwarebeschreibungssprache VHDL-AMS Anwendungsbeispiel: intelligenter Drucksensor
Entwurf digitaler elektronischer Systeme	Erläuterung von Spezifikation, Verifikation und Entwurf digitaler elektronischer Schaltkreise und Systeme; Vorstellung dafür verfügbarer Entwurfswerkzeuge (können online gestartet werden); Modellierung und Simulation mit VHDL-AMS
Modellierung und Simulation für den HF-Systementwurf	Grundlagen der Modellierung und Simulation von Hochfrequenz-systemen; Vorstellung eines Designflows und der zugehörigen Simulationswerkzeuge; Erstellung und Anwendung von analogen Verhaltensmodellen mit VHDL-AMS und Verilog-A; Komplexes Anwendungsbeispiel
Einführung in VHDL-AMS	Wiederholung der Hardwarebeschreibungssprache VHDL; Einführung in VHDL-AMS (Erweiterung von VHDL zur Beschreibung von analogen und gemischt analog-digitaler Systeme); Übungsbeispiele erfolgen mit den Programmen ADVance MS und SystemVision der Firma Mentor Graphics

Tabelle 1-4: Lernmodule MILEON

Zielsetzung

Die MILEON-Kurse sollen eine Zeit und Kosten sparende Vermittlung von Grundlagen und innovativem Spezialwissen auf dem Gebiet der Mikrosystemtechnik und Mikroelektronik ermöglichen. Sie dienen der betrieblichen Aus- und Weiterbildung von Entwicklungsingenieuren und technischen Führungskräften, sowie der Ergänzung universitärer Lehrveranstaltungen.

Dabei steht das Schaffen einer besonders praxisnahen Lernumgebung im Vordergrund. Neben herkömmlichen multimedialen Hilfsmitteln wie Texten, Bildern, Animationen, Videos, werden auch kommerzielle Entwurfs- und Simulationswerkzeuge aus dem täglichen Arbeitsumfeld des Lernenden eingesetzt. Durch die interaktive Darstellung sind komplexe Zusammenhänge leichter nachvollziehbar und das erlernte Wissen kann sofort überprüft und angewandt werden. Einen Screenshot des Kurses zeigt die folgende Abbildung.

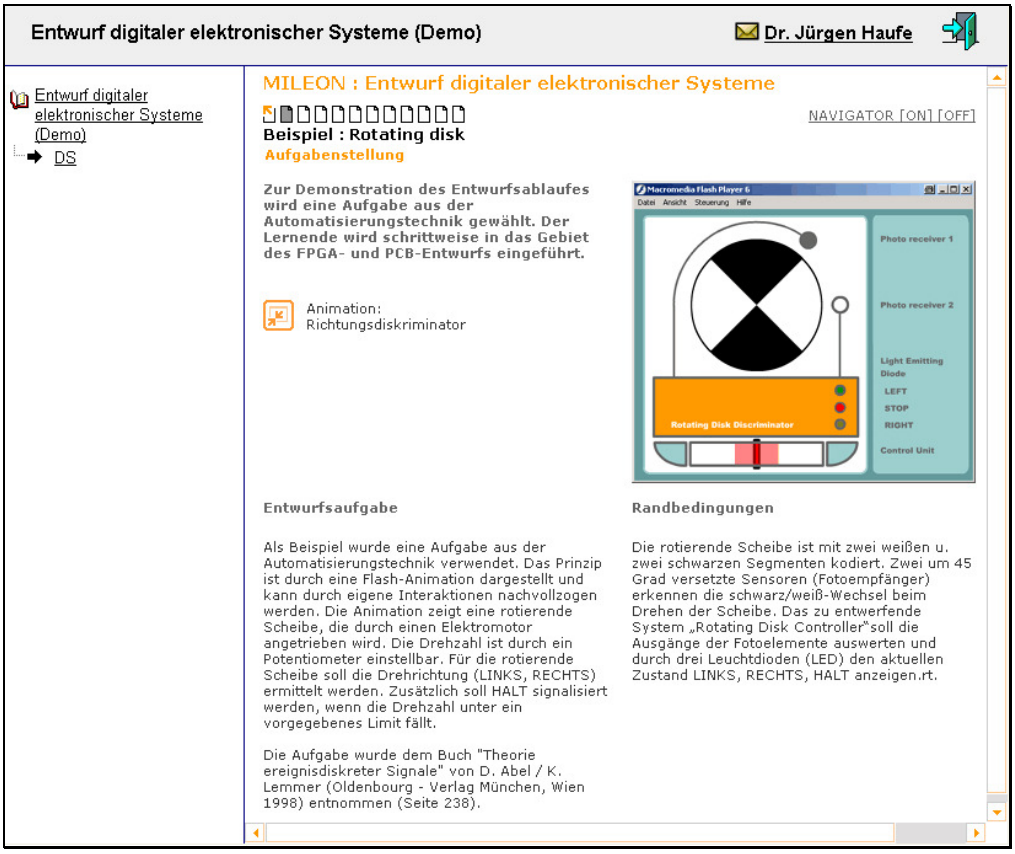


Abbildung 1-2: Screenshot des MILEON-Kurses

Implementierung

Die einzelnen Lernmodule wurden von wissenschaftlichen Mitarbeitern des Fraunhofer-Instituts IIS/EAS Dresden und Angestellten der Technischen Universität Dresden erstellt. Dabei hat die TU das Konzept für das Layout, sowie den Kurs „Grundlagen der Mikrosystemtechnik“ entwickelt. Das Fraunhofer-Institut war unter anderem für die Erstellung der weiteren vier Kurse und die Einbindung der verschiedenen Online-Werkzeuge zuständig.

Die verschiedenen Autoren erstellten zunächst die Lerninhalte in Form von Textdokumenten unter Verwendung von Programmen wie Microsoft Word, Microsoft Powerpoint, Adobe Framemaker und Adobe Acrobat. Für die multimediale Aufbereitung wurden Grafiken unter anderem mit Bildbearbeitungssoftware wie Adobe Photoshop entworfen. Macromedia Flash fand für die Entwicklung von Animationen Verwendung.

Mit Macromedia Dreamweaver wurden Texte, Bilder und Animationen zu einzelnen Lernobjekten im HTML-Format zusammengeführt. Die gesamte Kursstruktur sowie die Navigation wurden mit JavaScript implementiert.

Für die Gewährleistung einer intuitiven Bedienung, wurden eindeutige und wiederkehrende Elemente für die Navigation verwendet. Die MILEON-Bausteine sind plattformunabhängig und so in unterschiedlichen Arbeitsumgebungen ausführbar. Die Einbindung der Online-Werkzeuge wurde mit Hilfe des MOSCITO Tool Integrator realisiert.

Im Rahmen einer Diplomarbeit am Institut IIS/EAS Dresden, wurden die MILEON-Kurse mediendidaktisch und medientechnisch aufbereitet sowie exemplarisch nach SCORM 1.2 Prinzipien umgesetzt. Dabei wurden bald die Grenzen der Individualisierung von Lernprozessen unter Berücksichtigung dieser Spezifikation deutlich. Die neue Version SCORM 1.3 verspricht nun, dies mit Hilfe umfangreicher Sequencing-Anweisungen zu ermöglichen.

2 SCORM

In diesem Kapitel wird zunächst das Sharable Content Object Reference Model (SCORM) vorgestellt werden. Es folgt eine Einführung in die grundlegenden Bestandteile der SCORM Spezifikation sowie ein Vergleich zwischen der aktuellen Version und deren Vorgängern.

2.1 Überblick

Mit wachsender Nachfrage an E-Learning-Angeboten steigt auch der Wunsch der Autoren nach deren effizienter Erstellung. Durch Einhaltung von Standards können system- und plattformunabhängige Lerneinheiten erstellt werden. Sie gewährleisten außerdem die Austausch- und Kombinierbarkeit von Inhalten, wodurch die Kurse zeit- und kostensparender erstellt werden können. Mit dem Ziel, die schwer überschaubare Menge bestehender E-Learning-Standards und -Spezifikationen zu vereinheitlichen, etablierte die *Advanced Distributed Learning Initiative (ADL)* das *Sharable Content Object Reference Model (SCORM)*.

ADL

Die Advanced Distributed Learning Initiative wurde 1997 von dem US-Verteidigungsministerium (United States Department of Defense) gegründet, um eine kosten- und zeiteffiziente Erstellung qualitativ hochwertiger Lerninhalten zu ermöglichen.

Mit SCORM führt ADL Empfehlungen und Richtlinien für Lerntechnologien verschiedener Organisationen wie dem *Institute of Electrical and Electronics Engineers (IEEE)*, dem Global Learning Consortium *IMS* und dem *Aviation Industry Computer-Based Training Committee (AICC)* zusammen.

SCORM

Das Sharable Content Object Reference Model ist eine Spezifikation zur Beschreibung der Strukturierung von Kursmaterialien und deren Verwendung in *Lernmanagement-systemen (LMS)*. Das Modell soll dabei folgende Anforderungen erfüllen:

- **Wiederverwendbarkeit:** Unabhängig von den Entwicklungswerkzeugen können Lerninhalte in unterschiedlichen Applikationen und auf verschiedenen Plattformen weiter bearbeitet und verwendet werden.
- **Zugänglichkeit:** SCORM-konforme Kurse sind für verschiedene Personen an unterschiedlichen Orten jederzeit erreichbar.
- **Interoperabilität:** Erstellte Lerninhalte sind plattform-, software- und betriebssystemunabhängig.
- **Beständigkeit:** Auch nach Weiterentwicklung der Basistechnologie sollen die Kurse möglichst uneingeschränkt verwendbar bleiben.
- **Anpassungsfähigkeit:** Anweisungen können an individuelle und situationsabhängige Bedürfnisse angepasst werden.
- **Erschwinglichkeit:** Kosten- und Arbeitsaufwand sollen gesenkt, aber die Lerneffizienz gesteigert werden.

Bis zur Version 1.2 wurde die SCORM-Spezifikation in drei Büchern herausgegeben: Overview, *Content Aggregation Model (CAM)* und *Run-Time Environment (RTE)*. In der Version 1.3 (SCORM 2004) ist die Bibliothek um das Buch *Sequencing and Navigation (SN)* erweitert wurden. In Abbildung 2-1 sind die einzelnen Bücher dargestellt.

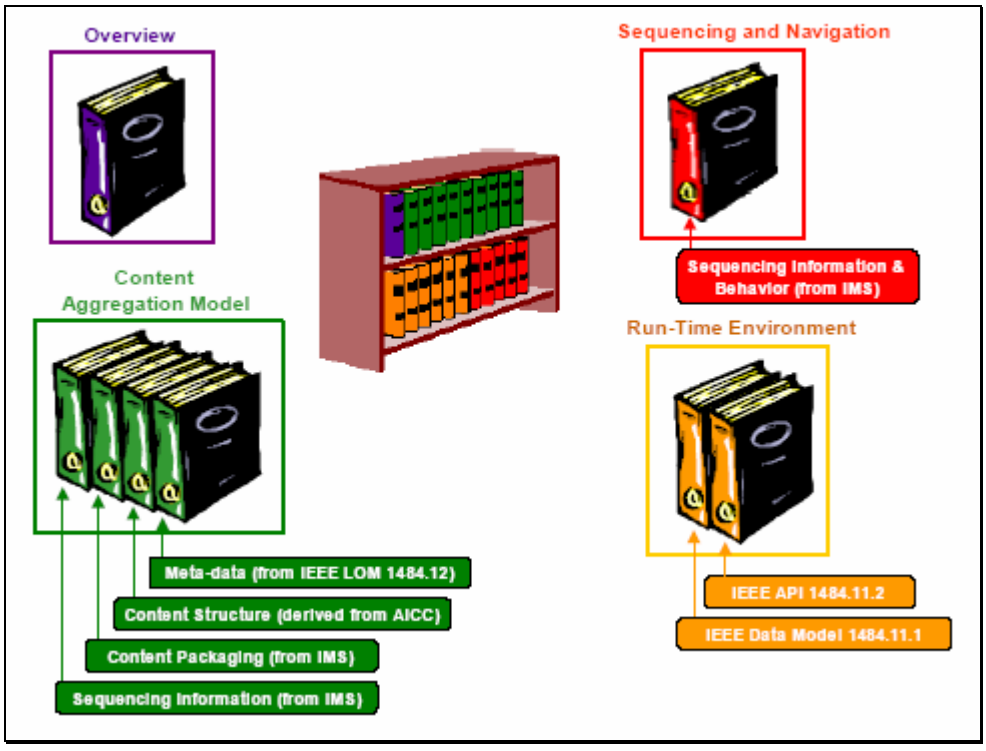


Abbildung 2-1: Überblick über die SCORM 1.3 Bücher

Im Overview Book werden die ADL-Initiative und deren wirtschaftliche Gründe für die Entwicklung von SCORM vorgestellt. Es folgt ein Überblick über die Spezifikation sowie eine Einführung in die folgenden SCORM-Bücher und deren Beziehungen unter einander. Richtlinien für das strukturierte Zusammenstellen von Lerninhalten werden vom Content Aggregation Model beschrieben. Das Run-Time Environment Dokument beinhaltet die Richtlinien für die Aufzeichnung des individuellen Benutzerverhaltens (*User Tracking*) sowie für die Kommunikation und den Datenaustausch zwischen den Lerneinheiten und dem Lernmanagementsystem. Die Ablaufsteuerung der Lerninhalte in Abhängigkeit des Verhaltens oder den Lernergebnissen des Anwenders wird durch das Sequencing and Navigation Book spezifiziert.

SCORM Produktionsablauf

An dieser Stelle wird grob der typische Produktionsablauf zum Erstellen von SCORM-konformen E-Learning-Kursen vorgestellt. Er soll den Bezug zwischen Theorie und Praxis ermöglichen und so einen Einstieg in die begrifflichen Grundlagen bilden. Im Kapitel 3 wird die Erstellung von SCORM-Kursen ausführlich beschrieben.

Zunächst müssen die Lerninhalte als HTML-Dateien vorliegen. Eine HTML-Seite entspricht dabei einer Bildschirmseite des späteren E-Learning-Kurses. Aus didaktischer Sicht sollte eine HTML-Seite einer Lerneinheit entsprechen. Sie wird meist aus mehreren Ressourcen wie Texten, Bildern, Videos, Animationen usw. (Assets, vgl. Kapitel 2.2.1) zusammengestellt.

Soll eine Seite später mit dem LMS kommunizieren (SCO, vgl. Kapitel 2.2.1), was zum Beispiel für das User Tracking oder die Auswertung von Übungsaufgaben nötig ist, müssen mit JavaScript die entsprechenden *API*-Befehle *(Application Programming Interface)* in die HTML-Seite eingebettet werden.

Um die Wiederverwendbarkeit der Lerneinheiten zu gewährleisten, fordert SCORM eine strikte Trennung von Navigation und Inhalt. Die Organisation der Seiten untereinander (Content Organization, vgl. Kapitel 2.2.1) wird daher nicht innerhalb der HTML-Seiten festgelegt, sondern mit Hilfe eines SCORM-standardisiertes XML-Schemas (Manifest, vgl. Kapitel 2.2.1), welches später vom LMS interpretiert wird.

Um die Lerneinheiten mit der definierten Navigationsstruktur in Form eines Kurses in einem SCORM-konformen LMS anzeigen zu können, müssen sie standardisiert gespeichert werden. Dafür werden alle Ressourcen und die Dateien zur Beschreibung

der Strukturierung in einer ZIP-Datei (Content Package, vgl. Kapitel 2.2.1) gepackt. Abbildung 2-2 soll den Produktionsablauf in Form einer Grafik verdeutlichen.

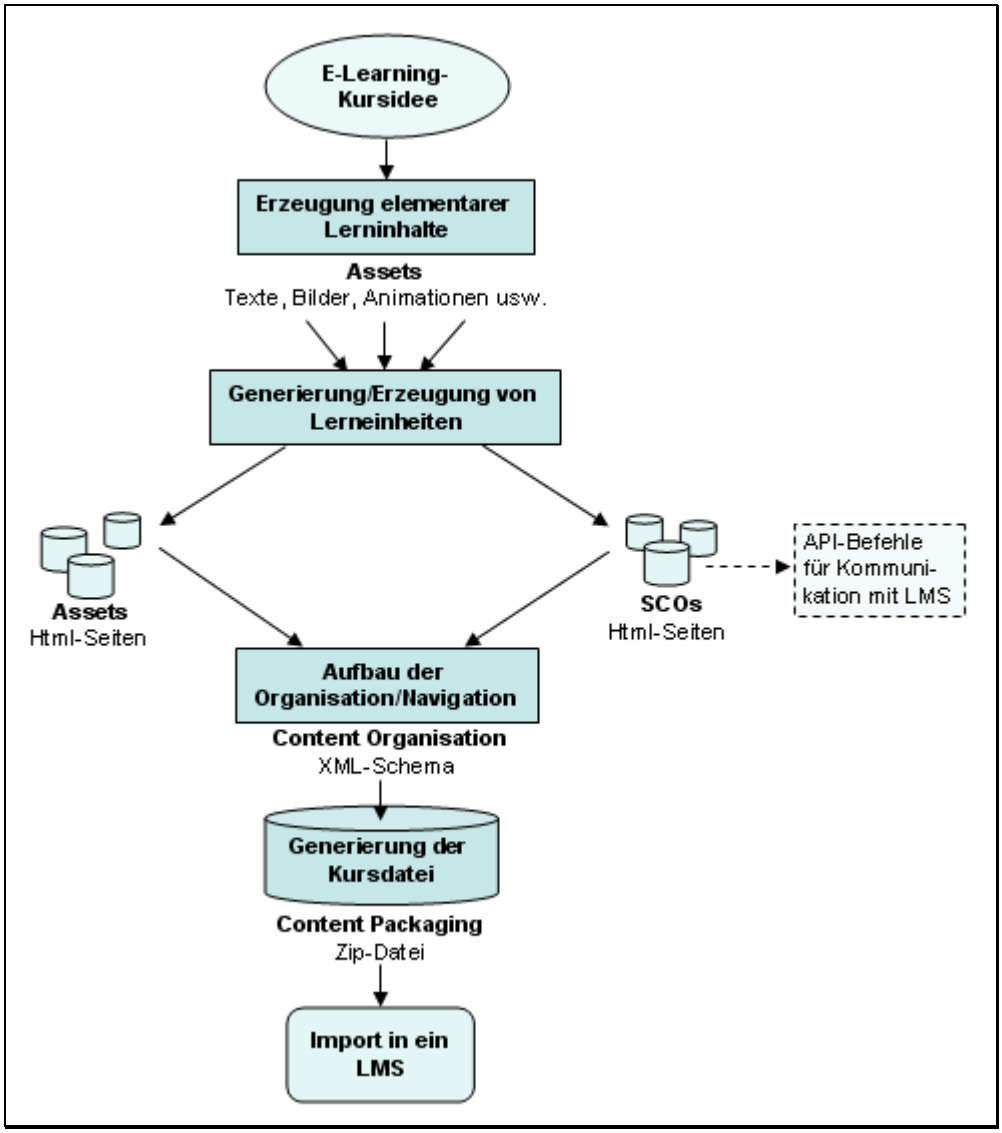


Abbildung 2-2: Typischer Produktionsablauf für die Generierung eines SCORM-konformen E-Learning-Kurses

SCORM-konformes LMS

In einem SCORM-konformen LMS können die nach der SCORM-Spezifikation erstellten E-Learning Kurse verwaltet und dem Lernenden zur Verfügung gestellt werden.

Das LMS wird auf einem zentralen Computer (Server) installiert und ist dann über lokale Software (in der Regel ein Webbrowser) für den Lernenden im Idealfall orts- und zeitunabhängig von jedem Client mit Internet- oder Intranetanschluss ansprechbar.

Um die Lernanwendung über das Netzwerk verfügbar zu machen, muss der im Zip-Format komprimierte Kurs in die Datenbank des SCORM-konformen LMS importiert werden.
Wird der Kurs von einem registrierten Nutzer gestartet, muss das LMS die entsprechende Navigationsstruktur anhand der Manifest-Datei (vgl. Kap 2.2.1) interpretieren und dem Lernenden entsprechend zur Verfügung stellen.
Protokolliert das LMS den individuellen Lernprozess des Studenten, ist es auch möglich, einen persönlichen Lernpfad in Abhängigkeit von Fortschritt und Erfolg des Kursteilnehmers, zu generieren. So könnte zum Beispiel ein Test ausgewertet und dem Lernenden anschließend die nicht bestandenen Lektionen in Form eines Lernpfades angeboten werden. Grundlage hierfür ist die Kommunikation zwischen den Lerninhalten und dem LMS, welche durch das SCORM Run-Time Environment (vgl. Kapitel 2.3) standardisiert ist. Die Kurse sind damit vom eigentlichen LMS unabhängig und können in anderen SCORM-konformen Lernmanagementsystemen eingebunden werden. Inhalte verschiedener Anbieter lassen sich ebenfalls in einem LMS integrieren.
Ein LMS sollte auch eine Suchfunktion zur Verfügung stellen. Die SCORM-Spezifikation stellt dafür einen Datensatz an Metadaten zur Verfügung, mit dem Lerninhalte (vgl. Kapitel 2.2.3) beschrieben und katalogisiert werden können. Damit wird eine effektive Suche in der gesamten Datenbank des LMS ermöglicht.

SCORM Viewer oder Player bieten dem Kursentwickler die Möglichkeit SCORM-Kurse offline, aber mit der vollen Funktionalität eines LMS zu testen. Damit ist es auch für den Lernenden möglich, einen Kurs offline zum Beispiel von CD zu starten. Eine Verfolgung des Bearbeitungsprozesses durch das LMS ist dabei in der Regel nicht möglich.
Abbildung 2-3 erläutert die Aufteilung des Browserfensters, welche bei der Ausführung eines Kurses vom LMS bzw. in einer Testumgebung angezeigt wird. Hierbei wird die Trennung von Inhalt und Navigation innerhalb von SCORM-Kursen deutlich. In den beiden rot markierten Bereichen werden die Navigationselemente und das Menü angezeigt, welche das LMS zur Laufzeit anhand der Manifest-Datei (vgl. Kapitel 2.2.1) generiert. Auf die grafische Oberfläche dieser Frames hat der Kursentwickler keinen Einfluss – sie werden von dem verwendeten LMS/Viewer bestimmt. In dem grün markierten Bereich werden die Lerneinheiten, also die einzelnen HTML-Seiten, angezeigt.

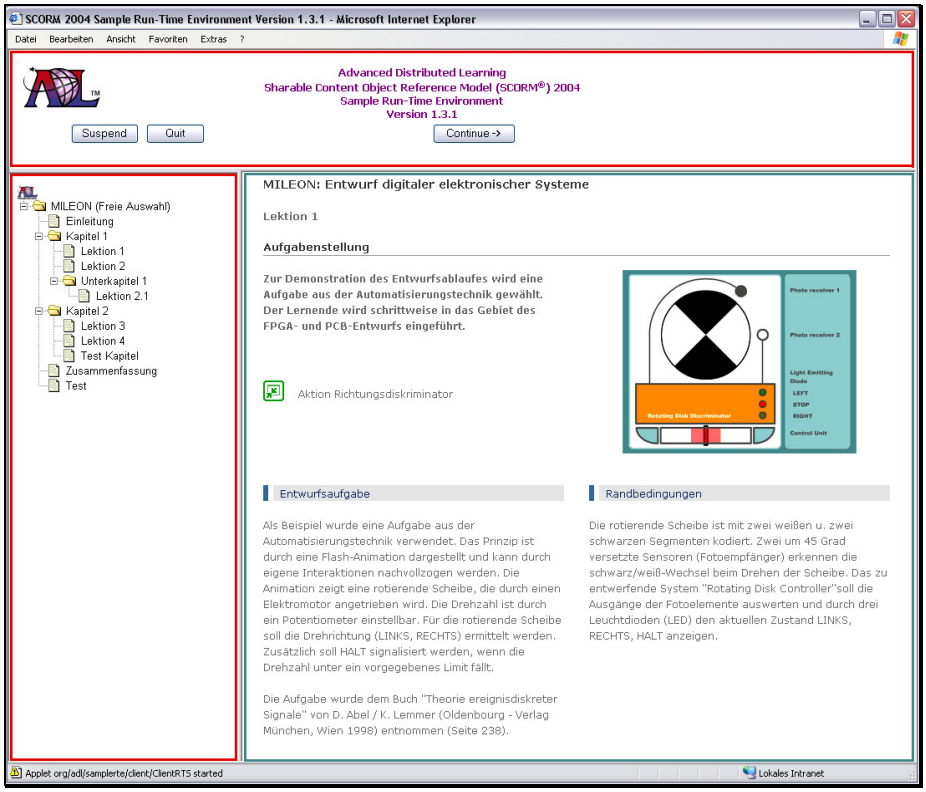


Abbildung 2-3: Screenshot LMS-Testumgebung

2.2 Content Aggregation Model

Das Content Aggregation Model definiert Aufbau, Ablaufsteuerung und Navigation von wiederverwendbaren Lerneinheiten sowie deren Attributierung mit Metadaten. Das Model ist in folgende Teile untergliedert:

- Content Model
- Content Packaging
- Metadaten
- Sequencing and Navigation (ab SCORM 1.3)

2.2.1 Content Model

Das Content Model definiert die elementaren inhaltlichen Komponenten (Assets, SCOs) und deren Aggregation zu komplexeren Einheiten (Content Organization).

Asset

Assets sind die kleinsten, inhaltlich in sich abgeschlossenen, Ressourcen. Sie sind wiederverwendbar und können miteinander zu einem weiteren Asset oder einem SCO kombiniert werden. Beispiele für verschiedene Assets zeigt Abbildung 2-4.

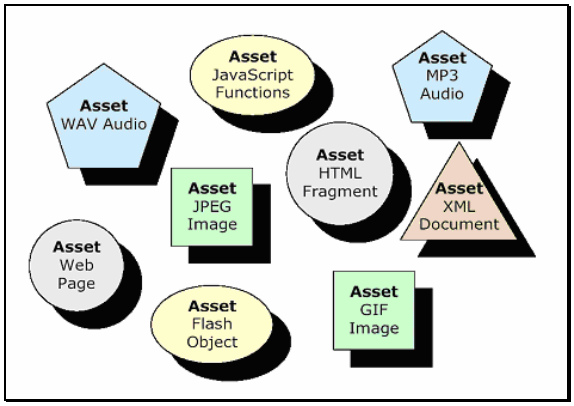


Abbildung 2-4: Beispiele für Assets [6]

SCO

Ein *SCO (Sharable Content Object)* besteht aus einem oder mehreren Assets. Ein SCO kann im Gegensatz zu einem Asset Informationen über den API-Adapter mit dem LMS austauschen. Dafür müssen vom SCO die entsprechenden API-Methoden aufgerufen werden. Diese werden in Kapitel 2.3 vorgestellt. Die Abbildung 2-5 verdeutlicht beispielhaft den Aufbau eines SCO.

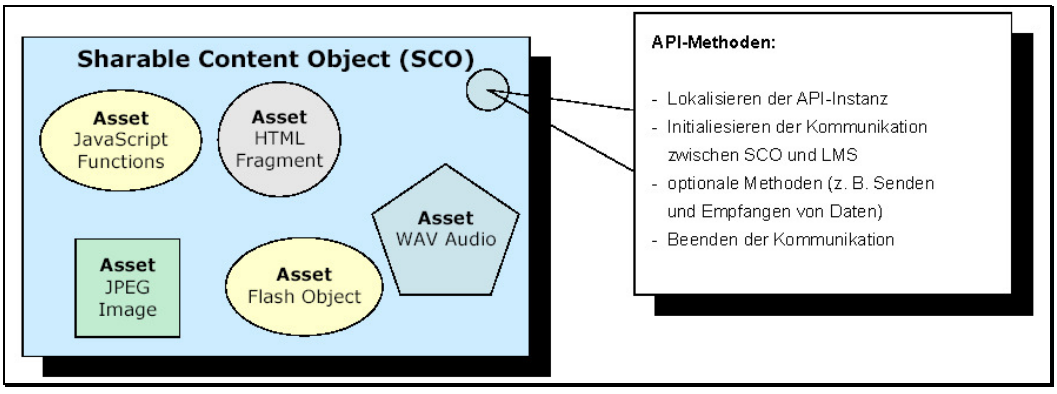


Abbildung 2-5: Aufbau eines SCO [6]

Inhaltsseiten, die zum Beispiel Testergebnisse des Lernenden auswerten, werden als SCOs implementiert. Hierfür müssen sie um die entsprechenden JavaScript API-Methoden erweitert werden.

Reine Informationsseiten können hingegen als Asset dargestellt werden, sofern sie keine User Tracking Daten ermitteln sollen.

Content Organization

Die Content Organization (in SCORM 1.2 Content Aggregation) beschreibt die Ablaufsteuerung und das Navigationsverhalten zwischen den Lerneinheiten. Die Struktur wird baumartig nach dem SCORM standardisierten XML-Schema in der Manifest-Datei abgebildet und zur Laufzeit vom LMS interpretiert. Eine mögliche Strukturierung zeigt Abbildung 2-6.

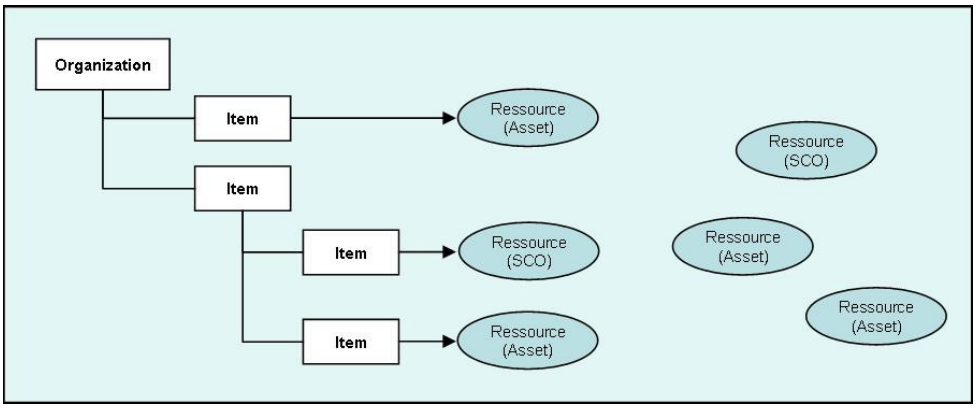


Abbildung 2-6: Beispiel einer Content Organization

Manifest

Im Manifest werden Angaben zur Navigationsstruktur, Metadaten und Verweise auf verwendete Ressourcen in XML abgelegt. Die Daten werden in der Manifest-Datei `imsmanifest.xml` gespeichert. Abbildung 2-7 zeigt vereinfacht den Aufbau dieser Datei.

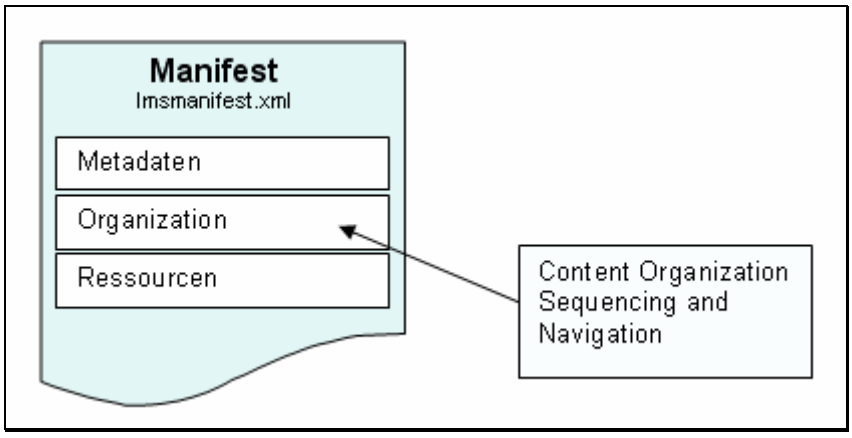


Abbildung 2-7: Aufbau einer Manifest-Datei

Die wichtigsten XML-Elemente der Manifest-Datei sind in Tabelle 2-1 erläutert. Ein Beispiel einer Manifest-Datei wird am Ende des Kapitels 2.2 gezeigt.

Element	Beschreibung	Mögliche Unterelemente
<manifest>	Das <manifest> Element ist das Wurzelelement der Manifest-Datei und kapselt alle Referenzdaten. Es kann ein oder mehrere Sub-Manifeste enthalten.	<metadata> <organizations> <resources> <manifest> <imsss:sequencing Collection>
<metadata>	Dieses Element dient der individuellen Beschreibung des Manifests, der Organization und deren Elemente	
<organizations>	Das <organizations> Element beschreibt eine oder mehrere hierarchisch aufgebaute Strukturen oder Organizations für das Content Package	<organization>
<organization>	In dem <organization> Element wird ein bestimmter Teil der Orga-nization hierarchisch definiert.	<title> <item> <metadata> <imsss:sequencing>
<title>	Dieses Element gibt den Namen der Organization oder eines Items an.	
<item>	Ein <item> Element verweist entweder auf ein weiteres <item> Element, oder wenn es keine weiteren Unterelemente enthält, auf eine Lernressource (SCO, Asset). Die hierarchische Anordnung der einzelnen <item> Elemente bildet die Struktur der Organization. (vgl. Abbildung 2-6)	<title> <item> <metadata> <imsss:sequencing>
<imsss:sequencing>	Das <imsss:sequencing> kapselt alle nötige Sequencing-Informationen (Kapitel 2.2.4)	
<imsss:sequencing Collection>	Mit diesem Element können wieder verwendbare Sequencing-Informationen gekapselt werden.	
<resources>	Hier werden alle verwendeten Res-sourcen aufgelistet.	<resource>
<resource>	Das <resource> Element verweist auf die physischen Dateien der verwendeten Ressourcen	<metadata> <file>

Tabelle 2-1: Aufbau einer Manifest-Datei

2.2.2 Content Packaging

Das Content Packaging beschreibt ein standardisiertes Format für die Strukturierung und Speicherung der Lerneinheiten. Dadurch wird die Wiederverwendbarkeit in verschiedenen LMS, Entwicklungswerkzeugen und Datenbanken sichergestellt. Es

enthält die Manifest-Datei (`imsmanifest.xml`) und die physischen Dateien (Ressourcen) eines eigenständigen Kurses, mehrerer Kurse oder eines Teilkurses. Für den Import in ein LMS, bzw. für den Austausch zwischen verschiedenen Systemen, wird das Content Package als *Package Interchange File (PIF)* in Form einer komprimierten Archivdatei im Zip-Format verpackt, siehe Abbildung 2-8:

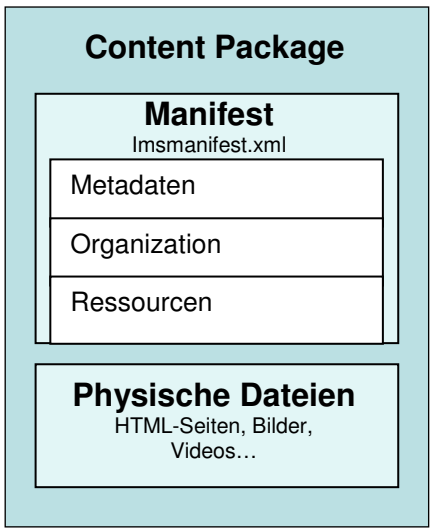


Abbildung 2-8: Aufbau eines Content Package

2.2.3 Metadaten

Die individuellen Eigenschaften des Content Package sowie der Content Model Komponenten (Assets, SCOs, Content Organizations) können durch Metadaten beschrieben werden. Sie erlauben eine eindeutige Identifizierung und Katalogisierung dieser Elemente.

Metadaten ermöglichen zudem eine effektivere Suche nach Lerneinheiten, beispielsweise innerhalb von Datenbanken, denn nicht immer ist eine Volltextidentifizierung zu empfehlen. Zum einen werden nicht alle Lerneinheiten ausreichend durch ihren Inhalt beschrieben und zum anderen kann so die Suche auf Schlüsselbegriffe beschränkt werden, was bei größeren oder stark frequentierten Datenbanken von großem Vorteil ist.

Eine exakte Beschreibung von Lerneinheiten durch Metadaten ermöglicht dem Anwender eine effektive Suche nach geeigneten Lernmaterialien und Kursentwicklern

das Auffinden vorhandener Lerneinheiten für deren Wiederverwendung. Der SCORM-konforme Metadaten-Datensatz ist in folgende neun Kategorien unterteilt:

Kategorie	Beschreibung	Beispiele
General	allgemeine Informationen über die Lernressource	Titel, Sprache, Schlüsselwörter
Lifecycle	Angaben zum Lebenszyklus und dem aktuellen Stand der Ressource sowie deren Verfasser	Version, Autor
Meta-Metadaten	Beschreibung der Metadaten selbst	Sprache, Verfasser
Technical	Enthält technische Daten und Anforderungen der Ressource	Format, Dateigröße, Speicherort, Installationsvoraus-setzungen
Educational	Beschreibung der pädagogischen Eigenschaften	Lernziel, Zielgruppe, Lerndauer
Rights	Informationen zu den Urheberrechten und Nutzungsbestimmungen	Urheberrechte, Kosten, Copyright
Relation	Beschreibung der Beziehungen der Lernressource zu Anderen	Zielressource, Beziehungsart
Annotation	Anmerkungen zum pädagogischen Nutzen der Ressource	Verfasser, Anmerkung, Datum
Classification	Einordnung in ein Klassifizierungssystem	Klassifikation, Zweck

Tabelle 2-2: Metadatenbeschreibung

Wie bereits erwähnt, werden die Metadaten innerhalb der Manifest-Datei den entsprechenden Ressourcen zugeordnet und gespeichert. Es ist auch möglich mit dem `<location>` Element auf Metadaten in einer externen XML-Datei zu verweisen. Abbildung 2-9 veranschaulicht den Aufbau einer Manifest-Datei.

2.2.4 Sequencing and Navigation

Der Sequencing and Navigation Abschnitt des Content Aggregation Models beschreibt wie die verschiedenen Sequencing Strategien mit XML codiert und in die Manifest-Datei eingefügt werden. Dieser Teil des Content Aggregation Models ist neu in der SCORM Version 1.3 enthalten und soll daher in dem Kapitel 2.3.2 Veränderungen im CAM Book näher betrachtet werden.

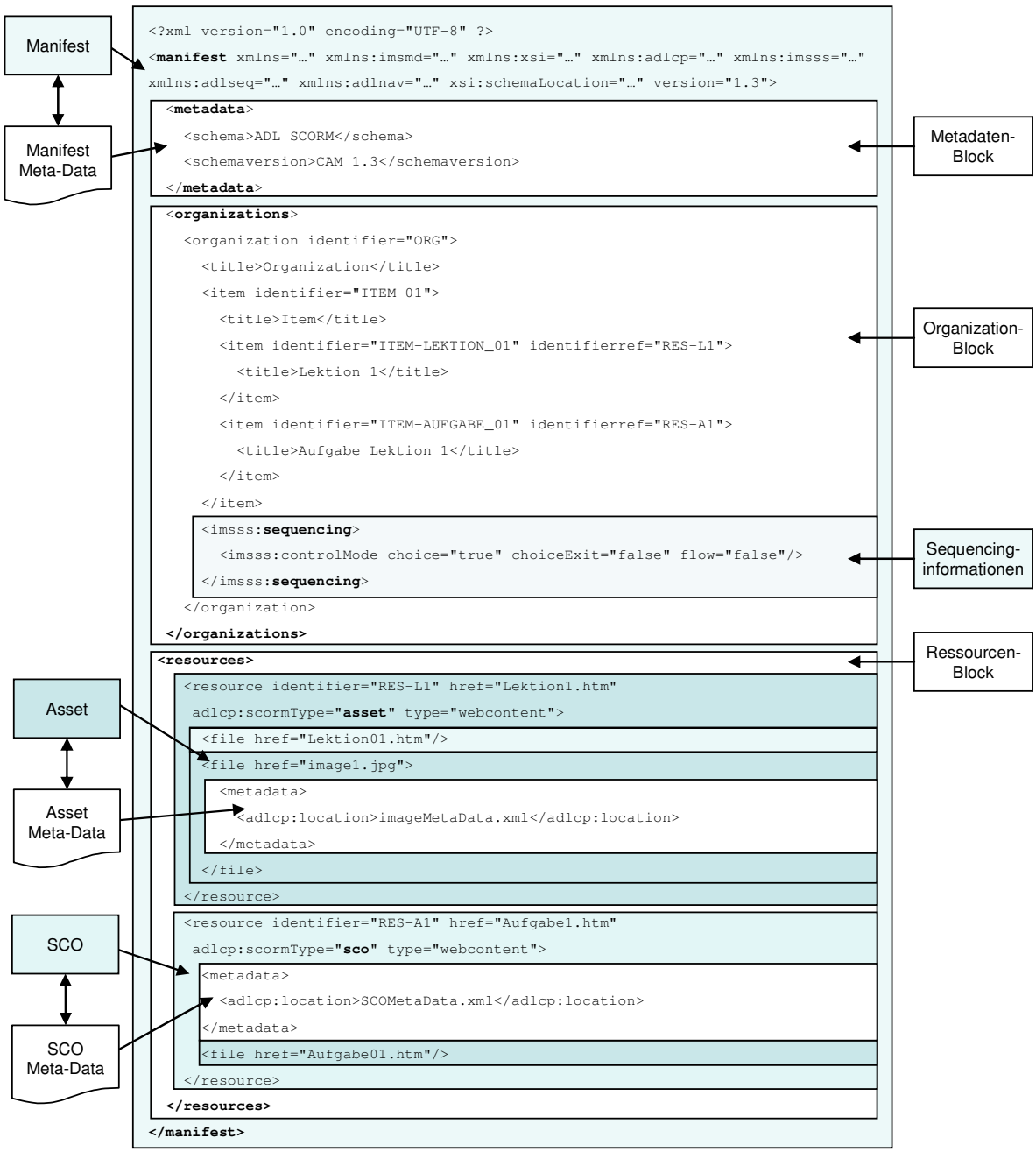


Abbildung 2-9: Manifest-Datei mit allen Bausteinen

2.3 Run-Time Environment

Um die Interoperabilität und die Wiederverwendbarkeit von Lernressourcen zu gewährleisten, muss auch die Kommunikation mit dem LMS standardisiert sein. Das SCORM Run-Time Environment stellt die hierfür nötigen Mechanismen zur Verfügung (siehe Abbildung 2-10).

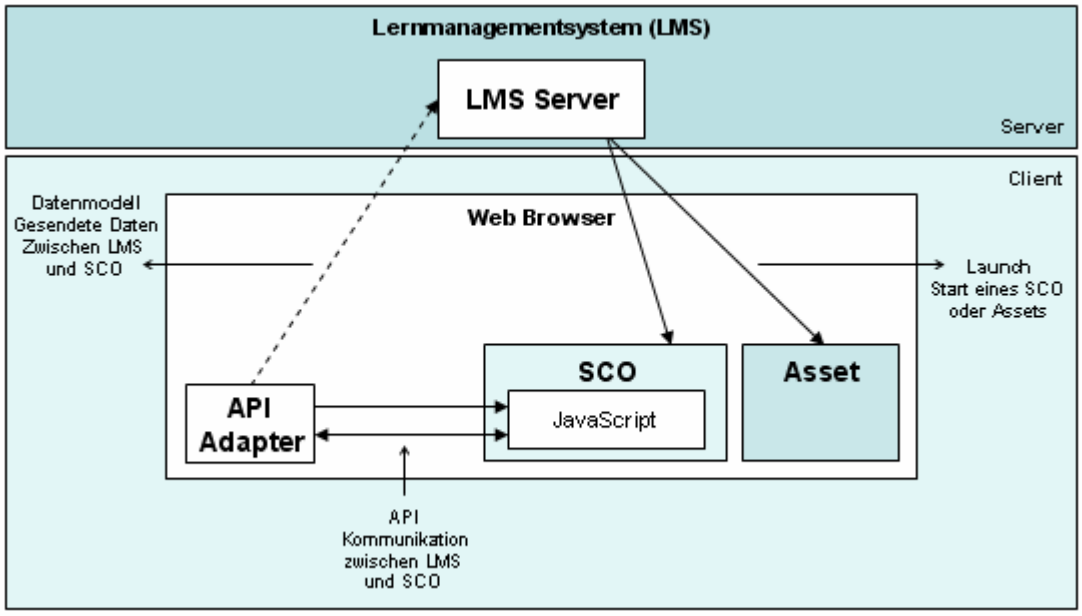


Abbildung 2-10: Funktionsweise der SCORM Run-Time Environment [10]

Launch

Das *SCORM Launch Model* definiert das Starten der Lerneinheiten (SCOs und Assets) vom LMS. Da Assets passive Inhalte darstellen, müssen sie lediglich über das HTTP-Protokoll vom LMS gestartet werden. SCOs hingegen sollen mit dem LMS kommunizieren können. Sie sind daher nach ihrem Start dafür verantwortlich, die Schnittstelle (API) des LMS zu lokalisieren. Hierfür muss das LMS das SCO entweder in einem abhängigen Browserfenster oder in einem Frame des LMS Fensters starten, durch welches es Zugriff auf die API-Instanz erhält. Das Launch Model verlangt außerdem, dass immer nur ein SCO pro Lernendem aktiv ist. Deshalb muss sich ein SCO nach Verwendung durch den entsprechenden API-Aufruf auch wieder beim LMS abmelden, bevor ein neues SCO als „aktiv“ markiert werden kann.

Application Programming Interface

Das Application Programming Interface stellt eine Reihe von Methoden für die standardisierte Kommunikation zwischen SCOs und dem LMS sowie deren Datenaustausch zur Verfügung. Jegliche Kommunikation wird dabei vom SCO über die entsprechenden JavaScript (*ECMAScript)* Funktionen initiiert. Hat das SCO die API-Instanz des LMS lokalisiert, wird die Kommunikations-Session durch den Aufruf der Methode `Initialize()` gestartet. Danach stehen die API-Funktionen `GetValue()`, `SetValue()` und `Commit()` für den Datenaustausch zwischen SCO und LMS zur

Verfügung. Mit der Methode `Terminate()` wird die Kommunikations-Session beendet. Die Fehlerbehandlung erfolgt durch die Funktionen `GetLastError()`, `GetErrorString(error code)` und `GetDiagnostic (value)`.

Data Model

Das Data Model beschreibt den für den Datenaustausch gültigen Wortschatz und definiert den Typ, den diese Elemente annehmen können. Damit wird eine einheitliche Informationsverarbeitung der Lernmanagementsysteme sichergestellt.

Mit dem Aufruf `SetValue("cmi.completion_status", „completed")` würde dem LMS zum Beispiel übermittelt werden, dass ein SCO vom Lernenden vollständig abgearbeitet wurde. Die Bedingungen, die für eine ausreichende Bearbeitung eines SCOs erfüllt sein müssen, werden dabei nicht von SCORM, sondern vom Entwickler des SCOs festgelegt. Eine vollständige Übersicht aller Elemente ist im Anhang A zu finden.

2.4 Sequencing and Navigation

Im Sequencing and Navigation Book werden die umfangreichen Möglichkeiten für die Ablaufsteuerung der Lernressourcen, abhängig vom Verhalten und der Lernfortschritte des Anwenders, beschrieben. Es ist neu in SCORM 1.3 und soll daher im nächsten Kapitel ausführlich beschrieben werden.

2.5 Vergleich zwischen SCORM 1.2 und SCORM 1.3

Drei Jahre nach der Veröffentlichung der SCORM Version 1.2 im Oktober 2001 brachte ADL im Januar 2004 die SCORM 1.3 Spezifikation (SCORM 2004) heraus. Nur wenige Monate später erschien dann im Juli 2004 eine überarbeitete Version. In diesem Kapitel sollen die Unterschiede von SCORM 1.2 und SCORM 1.3 aufgezeigt werden.

2.5.1 Grundlegende Veränderungen

In der Version 1.2 erfolgte erstmalig die Einteilung der Spezifikation in einzelne Bücher (Overview, Content Aggregation Model und Run-Time Environment). In SCORM 1.3 wurde die Bibliothek um ein weiteres Buch, das Sequencing and Navigation Book, erweitert.
Mit dieser Version wurde auch die Versionsgebung von SCORM verändert. Die einzelnen Bücher wurden separiert, womit die Versionsaktualisierung künftig für die einzelnen Dokumente und nicht mehr für die gesamte SCORM Spezifikation erfolgt. Mit der überarbeiteten Version von SCORM 1.3 tragen die Bücher die Versionsnummer 1.3.1.

2.5.2 Veränderungen im CAM Book

Content Model

Das Content Model-Element *Content Aggregation* wurde in SCORM 1.3 in Content *Organization* umbenannt, da es die Organisation der Lernressourcen beschreibt. Content Aggregation beschreibt nun die Zusammenstellung der Inhalte und das Content Packaging. Diese Umbenennungen wirken sich auch auf bestimmte Metadaten der entsprechenden Manifest Elemente aus:

Manifest Element	SCORM 1.2 MetaData Name	SCORM 1.3 MetaData Name
<manifest>	Package-level Meta-data	Content Aggregation Meta-data
<organization>	Content Aggregation Meta-data	Content Organization Meta-data
<item>	Content Aggregation Meta-data	Activity Meta-data

Tabelle 2-3: Umbenennungen der Metadaten Namen

Content Packaging

Mit der Einführung des Sequencing and Navigation sowie der Veränderungen der Content Packaging Spezifikation und des SCORM Run-Time Environment Datenmodells musste auch das Content Package angepasst werden. Die folgende Tabelle gibt einen Überblick über die Änderungen an Manifest-Dateielementen.

Element	SCORM 1.2	SCORM 1.3
<item>	Ein <item> Element, das weitere untergeordnete <item> Elemente enthält (Elternelement), kann eine Lernressource (SCO, Asset) referenzieren.	Ein <item> Elternelement darf keine Lernressource (SCO, Asset) mehr referenzieren, dies ist nur noch Blattitems (<item> Elemente die keine weiteren <item> Elemente enthalten) gestattet.
<adlcp:prerequisites>	Mit dem <adlcp:prerequisites> Element können Vorbedingungen an die <item> Elemente gebunden werden. (z.B. erfolgreicher Abschluss bestimmter Lernressourcen)	Das <adlcp:prerequisites> Element wurde durch das <imss:sequencing> Element ersetzt. Mit <adlnav:presentation> ist es nun auch möglich, Vorschriften für das <organisation> Element zu bestimmen. (siehe auch Kapitel 2.5.4)
<adlcp:maxtimeallowed>	Mit <adlcp:maxtimeallowed> kann eine Zeitbegrenzung für ein <item> Element festgelegt werden.	<adlcp:maxtimeallowed> wurde entfernt und durch die Sequencingangaben (siehe Kapitel 2.5.4) ersetzt.
<adlcp:timelimitaction>	<adlcp:timelimitaction> bestimmt die Aktion, die ausgeführt werden soll, wenn eine bestimmte Zeit überschritten wurde.	<adlcp:timelimitaction> wurde in <adlcp:timeLimitAction> umbenannt.
<adlcp:datafromlms>	Das <adlcp:datafromlms> Element speichert die Initialisierungsdaten der Ressource, die nach dem Launch vom LMS erwartet werden	<adlcp:datafromlms> wurde in <adlcp:dataFromLms> umbenannt.
<adlcp:masteryscore>	Im <adlcp:masteryscore> Element kann die erreichte Punktzahl des Lernenden für ein <item> Element gespeichert werden.	Das <adlcp:masteryscore> Element wurde entfernt und durch die Sequencing-Angaben (vgl. Kapitel 2.5.4) ersetzt.
<metadata>	Das <metadata> Kindelement des <manifest> Elements ist optional.	Das <metadata> Element ist nicht mehr optional, sondern wird verlangt.
<schema>	Das <schema> Kindelement des <metadata> Elements ist optional.	Das <schema> Element ist nicht mehr optional, sondern wird verlangt. Es muss den Wert „ADL SCORM“ enthalten.
<schemaversion>	Das <schema> Kindelement des <metadata> Elements ist optional.	Das <schema> Element ist nicht mehr optional, sondern wird verlangt. Es muss den Wert „CAM 1.3“ enthalten.
<adlcp:scormtype>	Das <adlcp:scormtype> gibt an, welchen Typ die Lernressource hat (SCO oder Asset)	Das <adlcp:scormtype> Element wurde in <adlcp:scormType> umbenannt

Tabelle 2-4: Änderungen an Manifest-Dateielementen

Eine weitere Änderung im Content Package betrifft die PIF-Datei. In SCORM 1.2 konnte sie in einem beliebigen Archivierungsformat (`.zip, .jar, .car, .tar` usw.) gespeichert werden, ab SCORM 1.3 ist nur noch das *ZIP*-Format zulässig.

Metadaten

Auch bei dem Metadaten-Standard sind einige Änderungen vorgenommen worden. So werden seit SCORM 1.3 Namen der Metadatenelemente, die aus mehreren Wörtern bestehen, im so genannten *lower camel case* Format geschrieben, siehe nachstehende Tabelle.

SCORM 1.2	SCORM 1.3
<aggregationlevel>	<aggregationLevel>
<lifecycle>	<lifeCycle>
<datetime>	<dateTime>
<metametadata>	<metaMetadata>
<minimumversion>	<minimumVersion>
<maximumversion>	<maximumVersion>
<installationremarks>	<installationRemarks>
<otherplatformrequirements>	<otherPlatformRequirements>
<interactivitytype>	<interactivityType>
<learningresourcetype>	<learningResourceType>
<interactivitylevel>	<interactivityLevel>
<semanticdesity>	<semanticDesity>
<intendedenduserrole>	<intendedEndUserRole>
<typicalagerange>	<typicalAgeRange>
<typicallearningtime>	<typicalLearningTime>
<copyrightandotherrestrictions>	<copyrightAndOtherRestrictions>
<taxonpath>	<taxonPath>

Tabelle 2-5: Namensänderungen von Metadaten

Änderungen, welche die Struktur des Metadatenschemas betreffen, werden in Tabelle 2-6 dargestellt.

Sequencing and Navigation

Der Abschnitt Sequencing and Navigation ist neu im SCORM CAM 1.3.1 Book. Es wird beschrieben, wie die Steuerung der Lernressourcen in Abhängigkeit des Verhaltens und der Lernergebnisse des Anwenders, in XML umgesetzt und in die Manifest-Datei eingefügt werden kann. Dabei wird auf das Sequencing and Navigation Book verwiesen, welches detaillierte Informationen zu den möglichen Ablauf- und

Navigationssteuerungen enthält. Für das bessere Verständnis sollen die XML-Elemente daher im Kapitel 2.5.4 erläutert werden.

Metadatenelement	SCORM 1.2	SCORM 1.3
General	<identifier> Unterelement ist lediglich reserviert, Unterelement <catalogentry> existiert <general> <identifier> <!-- Reserved --> </identifier> <catalogentry> <catalog>ISBN</catalog> <entry>2-7354</entry> </catalogentry> </general>	<identifier> Element wird verwendet, das <catalogentry> Element wurde gelöscht <general> <identifier> <catalog>ISBN</catalog> <entry>2-7354</entry> </identifier> </general>
Lifecycle	<identifier> Unterelement ist lediglich reserviert, Unterelement <catalogentry> existiert (siehe General Element) Unterelement <centitiy> existiert <centity> <vcard></vcard> </centity>	<identifier> Element wird verwendet, das <catalogentry> Element wurde gelöscht (siehe General Element) <centity> Element wurde durch <entity> ersetzt <entity></entity>
Meta-Metadaten	Unterelement <centitiy> existiert (siehe Lifecyle Element)	<centity> Element wurde durch <entity> ersetzt (siehe Lifecyle)
Technical	<requirement> Unterelement, welches die technischen Vorraussetzungen beschreibt, existiert <technical> <requirement> <minimumversion>0.4 </minimumversion> </requirement> </technical>	Element <requirement> erhält Unterelement <orComposite> für das Beschreiben verschiedener Anforderungen <technical> <requirement> <orComposite> <minimumversion>0.4 </minimumversion> <orComposite> <orComposite> <minimumversion>0.2 </minimumversion> <orComposite> </requirement> </technical>

Metadatenelement	SCORM 1.2	SCORM 1.3
Educational	<typicallearningtime> Unterelement ist vom Datentyp DateTime <typicallearningtime> <datetime>01:30:00</datetime> <description> <langstring>…</langstring> </description> </typicallearningtime>	Element <typicallearningtime> ist vom Datentyp Duration (Zeitdauer) <typicalLearningTime> <duration>PT1H30M</duration> <description> <string>…</string> </description> </typicalLearningTime>
Relation	<identifier> Unterelement ist lediglich reserviert, Unterelement <catalogentry> existiert (siehe General Element)	<identifier> Element wird verwendet das <catalogentry> Element wurde gelöscht (siehe General Element)
Annotation	Unterelement <person> existiert	Unterelement <person> wurde in <entity> umbenannt
Classification	Element ist nur für Content Aggregation und SCO Metadaten verfügbar	Element ist nur für alle Metadaten Anwendungsprofile verfügbar

Tabelle 2-6: Strukturelle Änderungen von Metadaten

2.5.3 Veränderungen im RTE Book

Im RTE Book 1.3.1, in dem die Richtlinien für das User Tracking und der Datenaustausch zwischen SCOs und dem LMS geregelt sind, kam es zu Veränderungen aufgrund der Standardisierung des JavaScript Application Programming Interface (*ECMAScript*) und dem Data Model. In den kommenden Abschnitten sollen diese Erneuerungen vorgestellt werden.

API

In Tabelle 2-7 werden zunächst die Namensänderungen von Methoden gegenübergestellt werden. Diese beeinflussen sowohl die Entwicklung von SCOs als auch die Implementierung durch das LMS.

In SCORM 1.3 ist es nun auch möglich, dass ein SCO die Initialize-Methode mehr als einmal aufruft. Es wird definiert, wie das LMS einen erneuten Aufruf von `Initialize(„")` verarbeiten muss.

SCORM 1.2	SCORM 1.3
`LMSInitialize(„")`	`Initialize(„")`
`LMSFinish(„")`	`Terminate(„")`
`LMSGetValue(parameter)`	`GetValue(parameter)`
`LMSSetValue(parameter_1, parameter_2)`	`SetValue(parameter_1, parameter_2)`
`LMSCommit(„")`	`Commit(„")`
`LMSGetLastError()`	`GetLastError()`
`LMSGetErrorString(parameter)`	`GetErrorString(parameter)`
`LMSGetDiagnostic(parameter)`	`GetDiagnostic(parameter)`

Tabelle 2-7: Namensänderungen der API-Methoden

Auch die Fehlercodebehandlung ist in SCORM 1.3 erweitert worden. Es gibt mehr und speziellere Fehlercodes für verschiedene Situationen. Diese sollten von SCORM 1.3-konformen Lernmanagementsystemen unterstützt werden. Eine Liste aller Fehlercodes, die während der Bearbeitung der verschiedenen API-Methoden generiert werden können, findet sich in Tabelle 2-8.

Data Model

Das Datenmodell des RTE 1.3.1 wurde ebenfalls verändert. Die Data Model Bibliothek wurde um Elemente für die Beschreibung von Sequencing-Informationen erweitert. Einige Datenelemente erhielten eindeutigere Bezeichnungen oder wurden durch spezifischere Elemente ersetzt. Teilweise wurden Elemente auch vollständig aus der Bibliothek entfernt. Eine Gegenüberstellung aller Datenelemente beider SCORM Versionen findet sich im Anhang A.

Während in SCORM 1.2 einige Elemente noch obligatorisch vom LMS implementiert werden konnten, müssen SCORM 1.3 konforme Lernmanagementsysteme alle Datenmodellelemente unterstützen.

SCORM 1.2 Fehlercode	SCORM 1.3 Fehlercode
0 – kein Fehler	0 – kein Fehler
101 – allgemeiner Fehler	101 – allgemeiner Fehler 102 – allgemeiner Initialisierungsfehler 103 – bereits initialisiert 104 – SCO-Kommunikation bereits beendet 111 – allgemeiner Fehler bei Kommunikationsbeendung 112 – Kommunikationsbeendung vor Initialisierung 113 – Kommunikationsbeendung nach erfolgreicher Beendung 122 – Datenabfrage vor Initialisierung 123 – Datenabfrage nach Kommunikationsbeendung 132 – speichern von Daten vor Initialisierung 133 – speichern von Daten nach Kommunikationsbeendung 142 – Commit vor Initialisierung 143 – Commit nach Kommunikationsbeendung
201 – ungültiges Argument	201 – allgemeines Argument
202 – Element kann keine Kinder erhalten 203 – Element ist kein Array; kann nicht gezählt werden	301 – allgemeiner Fehler beim Datenempfang
	351 – allgemeiner Fehler bei Datenübermittlung
	391 – allgemeiner Fehler bei Commit-Aufruf
401 – nicht implementierter Fehler	401 – undefiniertes Data Model Element
401 – nicht implementierter Fehler	402 – nicht implementiertes Data Model Element
301 – nicht initialisiert	403 – Data Model Element Wert nicht initialisiert
403 – Element kann nur gelesen werden	404 – Data Model Element kann nur gelesen werden
404 – Element kann nur geschrieben werden 402 – ungültige Wertzuweisung, Element ist ein Schlüsselwort	405 – Data Model Element kann nur geschrieben werden
405 – inkorrekter Datentyp	406 – Data Model Element falsch zugeordnet
	407 – Data Model Element Wert außerhalb des zulässigen Bereichs
	408 – Data Model Abhängigkeit nicht bekannt

Tabelle 2-8: Fehlercodes SCORM 1.2 – SCORM 1.3

2.5.4 Sequencing and Navigation Book

Das SN Book ist der SCORM 1.3 Spezifikation neu hinzugefügt worden. Es definiert zum einen, welche Sequencing-Regeln auf Lernaktivitäten angewandt werden können, um dem Anwender einen individuellen Lernpfad nach seinen Bedürfnissen zu ermöglichen. Zum anderen beschreibt das Buch wie diese Regeln vom LMS zur

Laufzeit interpretiert werden sollen und wie dem Lernenden die entsprechende Navigationsstruktur über die Benutzerschnittstelle zur Verfügung gestellt werden muss. Ablaufsteuerung und Navigation sind dabei vom Inhalt getrennt, womit die Wiederverwendbarkeit der Lernressourcen gewährleistet wird.
In der Version 1.2 gab es bereits rudimentäre Möglichkeiten, Ablaufsteuerung und Navigation der Kurse zu beeinflussen. Diese wurden aber noch im CAM Book im Abschnitt Content Packaging zusammengefasst.

2.5.4.1 Sequencing und Navigation in SCORM 1.2

Das Sequencing in SCORM 1.2 ist lediglich über das Festlegen von Vorbedingungen möglich, die entscheiden, ob und wann ein SCO dem Lernenden angezeigt werden darf. In der Manifest-Datei wird dafür das `<adlcp:prerequisites>` Element an das entsprechende `<item>` Element angehangen. Über das `<cmi.core.lession_status>` Element des RTE Data Models können sechs verschiedene Zustände einer Lerneinheit definiert und ausgewertet werden:

- passed – die Lerneinheit wurde erfolgreich ausgeführt
- completed – das SCO wurde ausreichend bearbeitet
- browsed – der Anwender hat die Seite überblättert
- failed – die Abarbeitung der Lerneinheit war nicht erfolgreich
- not attempted – nur unwesentliche Nutzung der Lerneinheit
- incomplete – das SCO wurde nicht ausreichend bearbeitet

So könnte zum Beispiel festgelegt werden, dass der Lernende die Lektion B erst starten kann, wenn er die Lektion A erfolgreich bearbeitet (passed) hat.
Es besteht auch die Möglichkeit, dass die Zustände mehrerer SCOs zu einer Vorbedingung verbunden werden. Eine Lerneinheit C soll eventuell nur gestartet werden, wenn die Lerneinheit A und B bereits erfolgreich absolviert wurden. Die hierfür zu verwendenten Operatoren der Prerequisites Scripting Language (*aicc_script*) werden in Tabelle 2-9 vorgestellt.

Operator		Beschreibung
And	&	Und-Verknüpfung: SCO1 & SCO2 SCO1 und SCO2 müssen bearbeitet sein („passed“ oder „completed“
Or	|	Oder-Verknüpfung: SCO1 | SCO2 SCO1 oder SCO2 muss bearbeitet sein
Not	~	Negation: ~ SCO3 SCO3 darf noch nicht (erfolgreich) bearbeitet sein („failed“, „incomplete“, oder „not attempted“)
Equals	=	Genaue Bestimmung: SCO3 = “passed“ SCO3 muss den Status „passed“ besitzen
Not equals	<>	Ungleich: SCO3 <> „completed“ SCO3 darf nicht den Status „completed“ besitzen
Set	{}	Liste von SCOs: {SCO1, SCO2, SCO4} SCO1, SCO2 und SCO4 sind Teile der Liste, durch Komma getrennt
Seperator	,	Trennzeichen in Listen
Minimum	x*	X aus y: 2* {SCO1, SCO2, SCO4} Zwei oder mehr der Elemente der Liste müssen bearbeitet sein
Precedence	( )	Vorrang: SCO1 & (SCO2 | SCO3) SCO1 und SCO2 oder SCO1 und SCO3 müssen bearbeitet sein

Tabelle 2-9: Operatoren der Prerequisites Scripting Language [9]

Mit diesen Vorbedingungen kann also schon, wenn auch nur in sehr einfacher Form, ein individueller Lernpfad erstellt werden. Zu beachten ist dabei, dass das Sequencing nicht von allen Lernmanagementsystemen unterstützt wird.

Die entsprechende Navigation der Lerninhalte wird zur Laufzeit vom LMS anhand der Content Organization, der definierten Ablaufsteuerung interpretiert und dem Lernenden beispielsweise in Form von Menüs und Buttons zur Verfügung gestellt. Diese Navigationselemente werden dabei getrennt von den Lernressourcen, zum Beispiel in einem separaten Frame, dargestellt.

2.5.4.2 Sequencing und Navigation in SCORM 1.3

Das Sequencing erfolgt in SCORM 1.3 nicht mehr über das `<adlc:prerequisites>` Element. Das neue Konzept basiert auf Aktivitätsbäumen mit denen die Ablaufsteuerung und das Navigationsverhalten unabhängig von der Implementierung beschrieben werden können. Dieses Konzept soll im Folgenden näher erläutert werden.

Activity

Aktivitäten oder Activities stellen eine bestimmte Handlung dar, die der Lernende ausführt, während er einen Kurs bearbeitet. Eine Aktivität kann auch auf weitere

Aktivitäten verweisen. Stellt sie ein Blattelement dar, muss sie eine Lernressource referenzieren.

Das Anbinden von so genannten Objectives ermöglicht das Speichern von Tracking Informationen bezüglich der Lernziele einzelner Lernaktivitäten.

Activity Tree

Ein Aktivitätsbaum beschreibt die hierarchische Struktur der einzelnen Lernaktivitäten. Diese basiert auf der Content Organization, in der jedes Item einer Lernaktivität entspricht. Die Erstellung eines Aktivitätsbaumes aus der Inhaltsstruktur zeigt Abbildung 2-11. Auch die an Items gebundenen Sequencing-Regeln können in den Aktivitätsbaum übernommen werden.

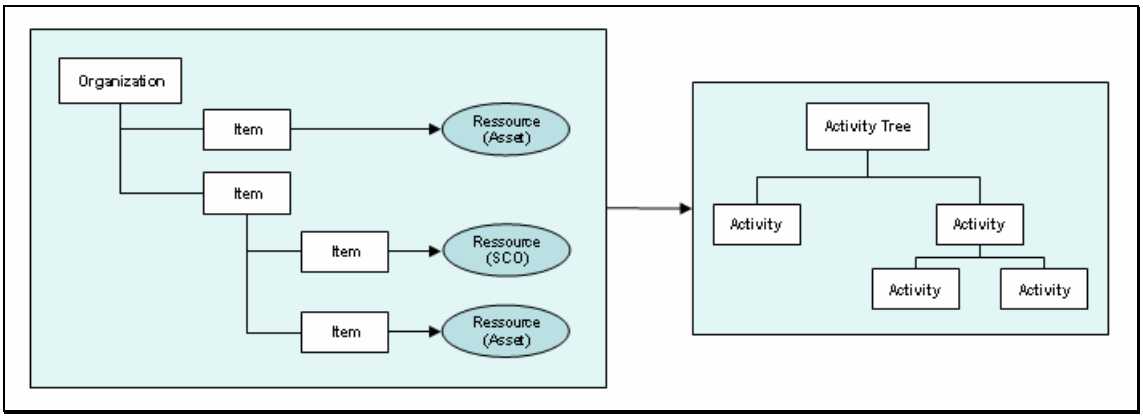


Abbildung 2-11: Erstellung eines Aktivitätsbaums

Sequencing Definition Model

Das Sequencing Definition Model stellt die Elemente zum Beschreiben des Sequencingverhaltens zur Verfügung. SCORM 1.3 konforme Lernmanagementsysteme müssen die Werte dieser Elemente richtig interpretieren um dem Lernenden die entsprechende Navigation bereitstellen zu können. Alle Elemente sowie deren XML Beschreibung werden in der Tabelle 2-10 erläutert.

Das XML Element `<imss:sequencing>` kapselt alle benötigten Sequencing-informationen in der Manifest-Datei. Mit dem `<imss:sequencingCollection>` Element kann eine Menge von Sequencinginformationen zu einer wiederverwendbaren Ablaufstruktur zusammengesetzt werden.

Element	Beschreibung
Sequencing Control Modes <SequencingControlModes>	Definieren das Navigationsmenü, das dem Lernenden angeboten wird z.B. über ein Menü oder Vor- bzw. Zurückbuttons
Constrain Choice Controls <ConstrainChoiceControls>	Einschränkung der freien Auswahl von Aktivitäten (Lernende soll nicht zu tief in die Ebenen der Lernaktivitäten vordringen können) z. B. Kindelemente erst frei wählbar, wenn Elternelement aktiv ist
Sequencing Rules <SequencingRules>	Regeln für das Anzeigen von Aktivitäten; diese Regeln werden von einer oder mehreren Beding-ungen und einer daraus resultierenden Aktion definiert z.B. ob und wann Aktivitäten im Menü angezeigt, übersprungen oder verlassen werden
Limit Conditions <LimitConditions>	Bedingungen unter denen eine Aktivität nicht zur Verfügung gestellt werden darf z. B. wie oft, oder wie lange eine Lernaktivität angezeigt wird
Rollup Rules <RollupRules>	Regeln für die Bestimmung der Tracking Daten einer Aktivität abhängig von ihren Unteraktivitäten z. B. Aktivität gilt nur als vollständig bearbeitet, wenn auch alle ihre Unteraktivitäten vollständig bearbeitet wurden
Objectives <Objectives>	Zuordnen von Lernzielen an eine Aktivität
Selection Controls <SelectionControls>	beschreibt wie bestimmte Kindaktivitäten ausgewählt werden können und z. B. Auswahl von 3 Testfragen (Kindaktivitäten) aus einer Aktivität Test mit 10 Testfragen (Kindern)
Randomization Controls <RandomizationControls>	legt fest, wie die Kindaktivitäten organisiert werden z. B. zufällige Anordnung von Testaufgaben
Delivery Controls <DeliveryControls>	Festlegen von Einschränkungen für das User Tracking z. B. Erfassen von Tracking Informationen verhindern

Tabelle 2-10: Sequencing Elemente

Sequencing Behavior

Das Sequencing Behavior beschreibt, wie das LMS auf die im Sequencing Definition Model definierten Ereignisse reagieren muss. Dieses Verhalten wird aus drei verschiedenen Modellen hergeleitet (siehe Tabelle 2-11).

Navigation Model

Mit dem in SCORM 1.3 neu hinzugefügten Navigation Model ist es neben der Navigation außerhalb der Lernressourcen auch möglich, SCOs aus einem SCO heraus zu starten. Im HTML-Script des SCOs können zum Beispiel Navigationselemente, wie Buttons, implementiert werden, in deren Ereignisbehandlungsroutinen API-Methoden

des SCORM RTE aufgerufen werden. Diese Methoden erhalten als Parameter die Navigation Model Elemente, welche das LMS veranlassen die entsprechende Navigation durchzuführen.

Es kann zum Beispiel der Aufruf `SetValue(„adl.nav.request", „continue")` innerhalb des JavaScripts eines SCOs erfolgen. Damit startet das LMS die im Aktivitätsbaum folgende Lernressource, sobald die aktuelle Kommunikationssession beendet wurde.

Modell	Beschreibung
Tracking Model	Dieses Modell speichert den aktuellen Zustand einer Aktivität. Es ist abhängig vom Verhalten und Fortschritt eines Lernenden und muss daher dynamisch zur Laufzeit angepasst werden.
Activity State Model	In diesem dynamischen Modell wird der Sequencing Zustand jeder Aktivität und der globale Zustand des Aktivitätsbaumes verwaltet.
Sequencing Definition Model	Das Sequencing Definition Model ist ein statisches Modell und beschreibt, wie die verschiedenen Sequencingprozesse die Informationen des Tracking Modells interpretieren müssen, um die definierte Ablaufsteuerung umzusetzen.

Tabelle 2-11: Modelle des Sequencing Behaviors

2.6 Bedeutung von SCORM für das Fraunhofer IIS/EAS

Für das Fraunhofer IIS/EAS spielt die SCORM-Spezifikation bei der Erstellung von E-Learning-Kursen eine bedeutende Rolle. Kurse können zum einen durch die strikte Trennung von Struktur und Inhalt und zum anderen durch die Verwendung von Standards und Spezifikationen wiederverwendbar und austauschbar erstellt werden.

Viele Kurse, wie zum Beispiel im Projekt MILEON, bauen auf dem selben oder ähnlichem Grundlagenwissen auf. Da die Inhalte von der Navigationsstruktur getrennt sind, können einzelne Lerneinheiten einfach in andere Kurse übernommen werden, ohne dass sie dafür zum Beispiel von einem vorhandenen Navigationsmenü getrennt werden müssen.

Um ein effizienteres Auffinden vorhandener Lerneinheiten für die Wiederverwendung zu ermöglichen, erfolgt die Beschreibung aller Lerneinheiten durch SCORM-konforme Metadaten.

Außerdem arbeitet das Fraunhofer Institut sehr oft mit anderen Instituten oder Unternehmen zusammen, womit die Austauschbarkeit von Lerninhalten und Lerneinheiten und die Plattformunabhängigkeit entscheidend sind.

All diese Vorteile ermöglichen zudem eine stark aufwands- und kostenreduzierte Erstellung von Lerneinheiten und E-Learning-Kursen.
Die erstellten SCORM-Kurse werden entweder mit entsprechenden Viewern bzw. Playern auf CD oder auf der SCORM 1.2 Plattform der Fraunhofer *eQualification* veröffentlicht.
Diese Plattform heißt „JCollege“ und wurde 2002 von T-Systems Multimedia Solutions GmbH speziell für die Fraunhofer Gesellschaft entwickelt. Anfänglich mussten die Kurse in einem eigenen Format nach einer spezifischen XML-Beschreibung gepackt werden. Dies stellte sich aber bald als problematisch heraus, da die Fraunhofer Gesellschaft auch Kurse von Fremdanbietern, meist für die eigene Weiterbildung, verwendet. Diese Kurse müssen natürlich ebenfalls in diesem Format erstellt werden, was zusätzliche Kosten verursacht. Deshalb wurde im November 2003 die Plattform auf die Verwendung der SCORM 1.2 Spezifikation umgestellt.
Für eine effektive Erstellung von Kursen, speziell für das Fraunhofer IIS/EAS, sind vom Institut Richtlinien sowie Templates für die Entwicklung von Lerneinheiten (SCOs und Assets) und deren Beschreibung durch Metadaten nach der SCORM 1.2 Spezifikation entwickelt worden. Als nachteilig wurde hierbei angesehen, dass bei der Strukturierung kaum oder gar nicht auf individuellen Erfahrungen und Ergebnissen des Lernenden eingegangen werden konnte. Jeder Kursteilnehmer erhielt die selbe Kursstruktur und die selben Inhalte, unabhängig von seinen Vorkenntnissen, oder seinem persönlichen Lernziel.
Genau diese Einschränkung könnte das Fraunhofer Institut mit der Entwicklung von E-Learning-Kursen nach der aktuellen SCORM 1.3 Spezifikation beheben. Wie ein entsprechender Produktionsablauf für einen Kurs mit individuellem Lernpfad aussieht, wird im folgenden Kapitel beschrieben.

3 Erstellung SCORM 1.3 konformer E-Learning-Kurse

Dieses Kapitel beschreibt, wie das in Kapitel 2 erlangte theoretische Wissen über die SCORM-Spezifikation in der Praxis umgesetzt werden kann. Hierfür wird zunächst der allgemeine Produktionsablauf erläutert. Unter Verwendung eines SCORM Editors (Reload Editor 2004) wird die Erstellung ein konkreten Beispiels für einen E-Learning-Kurs beschrieben.

3.1 Allgemeiner Ablauf

In diesem Abschnitt wird der allgemeine Ablauf bei der Erstellung eines SCORM konformen E-Learning-Kurses und der dafür benötigten Komponenten beschrieben.

3.1.1 Erstellen der Lerninhalte und Lerneinheiten

Assets

Zu Beginn müssen die einzelnen Lerninhalte erstellt werden. Diese sollten den mediendidaktischen und medientechnischen Ansprüchen eines E-Learning-Kurses genügen. Da sich diese Arbeit auf die technische Umsetzung begrenzen soll, wird auf die Diplomarbeit „Mediendidaktische und medientechnische Konzeption eines SCORM-basierten E-Learning-Kurses am Beispiel eines Projektes im Fraunhofer IIS/EAS Dresden [11] verwiesen.

Bei den Lerninhalten handelt es sich in der Regel um Texte, Bilder, Tonaufnahmen, Animation, Simulation, und Videos. Diese werden zunächst mit den entsprechenden Softwareprogrammen, wie Texteditoren, Bildbearbeitungsprogrammen, Videoschnittsoftware usw. erstellt. Im SCORM-Modell entsprechen die entstanden Materialien den Assets, also Lerninhalte, die später nicht mit dem LMS Daten austauschen können.

Danach erfolgt die Anordnung der Materialen in Form von Webseiten. In der Regel werden mehrere Lerninhalte, auf einer Seite zusammengeführt. Der nötige HTML-Code kann mit Hilfe von *WYSIWYG*-HTML-Editoren (*What You See Is What You Get*), wie

Macromedia Dreamweaver, oder manuell erstellt werden. Auch bei diesen HTML-Seiten handelt es sich um Assets.

SCOs

Sollen Lerneinheiten erstellt werden, die mit dem LMS kommunizieren können (SCOs), beispielsweise zur Speicherung und Verarbeitung eines Testergebnisses, müssen die entsprechenden API-Befehle für die Datenübertragung zwischen SCO und LMS implementiert werden. Hinzuzufügen ist außerdem das nötige JavaScript für die Auswertung oder Verarbeitung der Testergebnisse bzw. der Benutzereingaben.

Für die API-Aufrufe sollte die `APIWrapper.js` JavaScript-Datei von ADL als Schnittstelle verwendet werden. In dieser Datei sind alle verfügbaren API-Funktionen gekapselt, so dass sie für alle SCOs für die Datenübertragung zwischen SCO und LMS verwendet werden können. Beim Aufruf der Funktion `DoInitialize()` wird über die Fensterhierarchie die API-Instanz lokalisiert und in einem Handle bis zum Beenden der Kommunikation bereitgestellt. Über dieses Handle können nun die API-Aufrufe `Initialize(„"), GetValue(Data Model Element), SetValue(Data Model Element, Wert), Commit(„") und Terminate(„")` erfolgen. In `DoTerminate()` wird das API-Handle durch den Aufruf von `Terminate()` freigegeben und somit die Kommunikation beendet. Für jeden API-Aufruf erfolgt eine entsprechende Fehlerbehandlung.

Die `APIWrapper.js` Datei von ADL befindet sich im Anhang B. Ein Beispiel für den HTML-Code eines SCO mit einer Testaufgabe ist im Anhang C zu finden.

Soll dem Lernenden eine Rückmeldung über seine Testergebnisse durch das LMS gegeben werden, muss ein weiteres SCO programmiert werden, welches die Antworten auswertet und darstellt. Ein Beispiel für eine solche Auswertungsseite ist im Anhang D zu finden.

3.1.2 Erstellen der Manifest-Datei

Sind alle Lerninhalte und Lerneinheiten erstellt, kann die Manifest-Datei formuliert werden. In diesem XML-Dokument werden die Metadaten und die Content Organization definiert. Die Angaben von Metadaten für die einzelnen Content Model Elemente (Assets, SCOs, Content Organizations) und das Content Package sind

weitgehend optional. Eine ausführliche Beschreibung durch diese Daten ist jedoch für das Auffinden und die Wiederverwendung dieser Einheiten von hohem Nutzen. Die Metadaten werden entsprechend ihrer Zuordnung in verschiedenen Bereichen der Manifest-Datei angelegt:

Metadaten	Zuordnung in der Manifest-Datei	Bemerkung

Content Aggregation Metadaten	Direktes Unterelement des `<manifest>` Knotens	Beschreibung des gesamten Kurses (Content Aggregation/Content Package)
		Achtung: folgende Elemente sind obligatorisch!
		
		`<metadata>`
		`  <schema> ADL SCORM`
		`  </schema>`
		`  <schemaversion> CAM 1.3`
		`  </schemaversion>`
		`</metadata>`

Content Organization Metadaten	Direktes Unterelement des `<organization>` Knotens	Beschreibung von Lerneinheiten und deren Struktur (Content Organization)
Activity Metadaten	Direktes Unterelement des `<item>` Knotens (Unterelement von `<organization>`)	Beschreibung einer Lernaktivität (Activity)
SCO Metadaten	Direktes Unterelement des `<resource>` Knotens (Unterelement von `<resources>`)	Beschreibung von Lerneinheiten (SCO/ Asset)
Asset Metadaten	Direktes Unterelement des `<file>` Knotens (Unterelement von `<resource>`)	Beschreibung von Lerninhalten (Assets)

Tabelle 3-1: Anwendungsbereiche der Metadaten

Neben der Erstellung von Metadaten innerhalb der Manifest-Datei ist es auch möglich, auf eine externe XML-Datei zu verweisen. Hierfür wird das `<location>` Element verwendet:

```
<metadata>
      <adlcp:location>course/metadata/course.xml</adlcp:location>
</metadata>
```

Listing 3-1: Verweis auf eine externe Metadaten-Datei

Die Metadaten können zum Beispiel manuell in einem XML-Editor erzeugt werden. Es gibt aber auch Tools, die die Erstellung von SCORM Metadaten über Formulareingaben ermöglichen.

Die Definition der Content Organization ist prinzipiell auch manuell möglich, allerdings ist die Programmierung des XML-Codes sehr aufwendig. Hier empfiehlt es sich auf ein unterstützendes Tool zurückzugreifen. Eine Möglichkeit der Strukturerstellung ganz ohne XML-Kenntnisse bieten sogenannte SCORM Editoren (dazu mehr im Kapitel 3.2.3), weshalb die Definition der Content Organization auch an Hand eines solchen Tools erläutert werden soll (vgl. Kapitel 3.3).

3.1.3 Generierung des Package Interchange Files

Für den Import in ein LMS müssen alle verwendeten Dateien und Ressourcen (Content Package) zu einem Package Interchange File (PIF) gepackt werden. Dafür werden alle verwendeten Ressourcen der Lerninhalte und Lerneinheiten sowie die Manifest-Datei `imsmanifest.xml` mit ihren zugehörigen XML-Schemata, in einem Zip-Archiv komprimiert. Bei den XML-Schemata handelt es sich um DTD (Document Type Definition) und XSD (XML Schema Definition) Dateien, welche die Dokumentenstruktur der XML-Datei `imsmanifest.xml` definieren. Sie müssen zusammen mit der Manifest-Datei im Wurzelverzeichnis des Zip-Archivs liegen.
Die Zip-Datei kann mit Hilfe eines entsprechenden Packprogramms erstellt werden. Tools wie SCORM Editoren unterstützen ebenfalls die Generierung eines Content Packages.

3.1.4 Testen erstellter SCORM-Kurse

Es empfiehlt sich die SCORM-Konformität und die Funktionalität der fertigen E-Learning-Kurse vor dem Import in ein LMS offline zu testen. Hierfür können sogenannte *SCORM Viewer* (vgl. Kapitel 3.2.4) verwendet werden.

3.1.5 Import des fertigen Kurses in ein SCORM-konformes LMS

Wenn der nach der SCORM-Spezifikation erstellte Kurs dem gewünschten Design und Funktionalität entspricht, kann das Content Package (PIF) des Kurses in ein SCORM-konformes LMS importiert werden. Hierbei ist zu beachten, dass die verwendete SCORM-Version vom LMS unterstützt wird. Außerdem ist zu prüfen, ob alle verwendeten Elemente und Funktionalitäten, die für ein LMS als optional in der Spezifikation angegeben sind, unterstützt werden.

3.2 SCORM Tools

Wie bereits angedeutet, gibt es eine Reihe von verschiedenen Werkzeugen, die die Erstellung von E-Learning-Kursen nach der SCORM Spezifikation unterstützen. In dieser Arbeit werden diese Werkzeuge unter dem Begriff SCORM Tools zusammengefasst.
Das Kapitel soll eine Auswahl dieser Tools vorstellen und prüfen, in wie weit diese bereits die im Juli 2004 herausgegebene SCORM 1.3 Spezifikation unterstützen.
Werkzeuge für die Erstellung von Lerninhalten wie Bilder, Texte, Animationen, Internetseiten usw. sollen nicht weiter aufgeführt werden, da sie zwar für die Entwicklung von Kursen benötigt werden, aber keine SCORM spezifischen Features bieten. Dies trifft auch auf XML-Editoren zu, die für die Erstellung der `imsmanifest.xml` verwendet werden können.

3.2.1 Tool für die Erstellung von SCOs

Diese Tools erzeugen die notwendigen JavaScript- und API-Funktionen, die ein SCO für die Kommunikation mit einem LMS benötigt. Sie sollen daher im folgenden SCO Builder genannt werden. Es ist aber für jedes Werkzeug zu prüfen, in wie weit die automatische Generierung der Kommunikationsbefehle unterstützt wird.
Für Macromedia Dreamweaver MX ist eine SCORM Runtime Wrapper Extension verfügbar, mit der Befehle für die Initialisierung und Terminierung eines SCOs

automatisch in die HTML-Seite generiert werden können. Es ist außerdem möglich, den Status des SCO beim laden und/oder verlassen zu setzen.
Mit Macromedia Flash MX können Animationen produziert und anschließend als SCO exportiert werden, wobei ebenfalls die nötigen Initialisierungs- und Terminierungsbefehle erzeugt werden.

3.2.2 SCORM Editoren

SCORM Editoren ermöglichen eine einfache Strukturerstellung von E-Learning-Kursen. Der XML-Code der Manifest-Datei wird dabei automatisch generiert. Dafür gibt der Kursentwickler die Struktur anhand eines Ressourcenbaumes und die entsprechenden Navigations- oder Sequencing-Informationen mit Hilfe von Eingabefenstern an. Der Entwickler benötigt somit keinerlei Kenntnisse in XML, sollte aber trotzdem mit der SCORM-Spezifikation vertraut sein, um die Struktur korrekt umsetzen zu können. SCORM Editoren sollten außerdem die Eingabe von Metadaten ermöglichen und diese automatisch der Manifest-Datei hinzufügen oder eine externe Datei erzeugen, welche in der Manifest-Datei entsprechend referenziert wird. In der Regel bieten diese Editoren auch die Möglichkeit der Content Package Erstellung. Folgende Kriterien sollten bei der Auswahl eines SCORM Editors berücksichtigt werden [11]:

- Unterstützte SCORM Version
- Einfaches Erstellen von Kursstrukturen
- Verwaltung von Metadaten bzw. Umfang der möglichen Metadaten
- Im- und Export von Metadaten (Wiederverwendung)
- Erkennung aller von einer Lerneinheit benötigten Ressourcen
- Content Package Erstellung

Eine sehr umfangreiche Funktionalität bietet zum Beispiel der Reload Editor (vgl. Kapitel 3.3). Weitere SCORM Editoren sind unter anderem der Saxonia Course Builder sowie der AltEd SCORM Editor.

3.2.3 Autorensysteme für die Erstellung von SCORM-konformen Kursen

Autorensysteme ermöglichen es, einen kompletten E-Learning-Kurs über eine rein grafische Benutzerschnittstelle zu erstellen. Es sind also keinerlei Programmierkenntnisse notwendig. In der Regel unterstützen Autorensysteme neben einer grafischen Eingabe auch die manuelle Programmierung.
Ein Autorensystem, mit dem Kurse nach der SCORM Spezifikation erstellt werden können, sollte also folgende Kriterien erfüllen:

- Unterstützung der gewünschten SCORM-Version
- Erstellung *elementarer* Lerninhalte wie Texte, Tabellen, Grafiken usw.
- Zusammenfassung aller benötigten Lerninhalte (auch Animationen, Filme usw.) zu Lerneinheiten im HTML-Format nach dem *WYSIWYG*-Prinzip, sowie auf Quellcode-Basis
- Automatische Generierung und Erstellung per Hand der nötigen JavaScript- und API-Befehle für die Erzeugung von SCOs
- Erstellung von Testaufgaben mit notwendigen Kommunikationsbefehlen
- Erzeugung der Manifest-Datei (einschließlich Metadaten) über eine grafische Oberfläche und über XML-Code Eingabe
- Erstellung des Content Package
- Kurserstellung auch ohne tiefere SCORM-Kenntnisse

Autorensysteme bieten die einfachste Möglichkeit, einen SCORM-Kurs zu erstellen, da sie den gesamten Produktionsablauf unterstützen und keinerlei Programmierkenntnisse vom Autor abverlangen. Allerdings sind diese Systeme in ihrer Funktionalität oft eingeschränkt. Demnach ist zu prüfen, in wie weit das Autorensystem die benötigte Funktionalität für die Erstellung des gewünschten Kurses unterstützt.
Zu den SCORM unterstützenden Autorensystemen gehören zum Beispiel WBTExpress und author42. Mit Macromedia Captivate können ebenfalls SCORM konforme E-Learning-Anwendungen erstellt werden, allerdings handelt es sich hierbei um ein Screen Capturing Tool. Dabei werden alle Interaktionen mit dem Bildschirm aufgezeichnet und als Filmsequenzen gespeichert. Jede dieser Aufzeichnungen entspricht dann einer Lerneinheit, welche zum Beispiel für Tutorials für

Softwareprogramme, zu einem Kurs zusammengestellt werden können. Es ist nicht möglich, andere Lerneinheiten, wie Bilder, Texte oder Webseiten, zu importieren.

3.2.4 SCORM Viewer

SCORM Viewer ermöglichen einen Offline-Test während oder nach der Entwicklung von Kursen, bevor sie in ein LMS importiert werden. Sie ermöglichen es, die SCORM Konformität sowie die Form und komplette Navigationsstruktur des E-Learning-Kurses zu überprüfen, da sie diese wie ein LMS interpretieren und darstellen. SCORM Viewer sind unter anderem die Sample RTE von ADL, der Microsoft LRN Viewer und der Reload SCORM Player.

3.2.5 Vergleich/Gegenüberstellung

Im Folgenden sollen noch einmal die verschiedenen Merkmale der SCORM Tools in Form von Tabellen gegenübergestellt werden. Tabelle 3-2 zeigt, welcher Abschnitt bei der Produktion eines SCORM konformen E-Learning-Kurses von welchem Tool unterstützt wird.

	Welcher Abschnitt der Kurserstellung nach SCORM wird unterstützt					
SCORM Tool	Asset	SCO (API-Befehle)	Content Organisation (Manifest)	SCORM Metadaten	Content Package	Offline-Test
SCO Builder	-	x	-	-	-	-
SCORM Editor	-	-	x	x	x	x
SCORM Autorensys-teme	x	x	x	x	x	x
SCORM Viewer	-	-	-	-	-	x

Tabelle 3-2: SCORM-Unterstützung verschiedener SCORM Tools

In Tabelle 3-3 wird dargestellt, welche Kenntnisse bei Verwendung des entsprechenden SCORM Tools vom Autor abverlangt werden, um einen kompletten SCORM-Kurs zu erstellen.

	Benötigte Kenntnisse für Gesamtkurserstellung nach SCORM			
SCORM Tool	HTML	JavaScript	XML	SCORM
SCO Builder	x	-	x	x
SCORM Editor	x	x	-	x
SCORM Autorensysteme	-	-	-	-
SCORM Viewer	x	x	x	x

Tabelle 3-3: **Benötigte Kenntnisse für Kurserstellung mit den SCORM Tools**

Da dieses Kapitel die Erstellung SCORM 1.3 konformer Kurse beschreibt, wurde untersucht in wie weit diese Version bereits von ausgewählten SCORM Tools unterstützt wird.

Wie die Tabelle 3-4 verdeutlicht sind derzeit nur sehr wenige Tools erhältlich, die die Kurserstellung nach der SCORM 1.3 Spezifikation unterstützen, erhältlich. Da Macromedia Captivate auf Grund der Einschränkung von Lerneinheiten auf Bildschirmaufzeichnungen für die Kurserstellung des Fraunhofer IIS/EAS nicht eingesetzt werden kann, ist lediglich die Verwendung des Reload Editor 2004 zu untersuchen. Für den Offline-Test der Kurse kann die Sample RTE 1.3.3 von ADL verwendet werden.

SCORM Tool	Tool	Version	Quelle
SCO Builder	Macromedia Flash MX 2004	1.2	www.macromedia.com
	SCORM Runtime Wrapper für Macromedia Dreamweaver MX	1.2	www.macromedia.com
SCORM Editor	Reload Editor 2004	1.2 / 1.3	www.reload.ac.uk
	Saxonia Course Builder 1.0	1.2	www.saxsys.de
SCORM Autorensysteme	WBTExpress4	1.2	www.wbtexpress.com
	Macromedia Captivate (Screen Capturing)	1.2 / 1.3	www.macromedia.com
	author42	1.2	www.bureau42.de
SCORM Viewer	Sample RTE 1.3.3	1.3	www.adlnet.org
	Reload SCORM Player 1.0.1	1.2	www.reload.ac.uk
	Microsoft LRN Viewer	1.2	Nicht mehr erhältlich

Tabelle 3-4: SCORM Versionsunterstützung der ausgewählten SCORM Tools

3.3 RELOAD Editor 2004

RELOAD ist ein Teil des von *JISC (The Joint Information Systems Committee)* finanzierten „Exchange for Learning“ Projektes. Geleitet wird es von der University of Bolten in Großbritannien. Dieses Projekt konzentriert sich auf die Entwicklung von Werkzeugen, die auf aktuellen E-Learning-Spezifikationen basieren. Die Primärziele liegen dabei auf dem einfachen Erzeugen von wiederverwendbaren Lerneinheiten und pädagogisch wertvoller Lernanwendungen basierend auf Lernpfaden [12].
Der Reload Editor ist ein von RELOAD entwickelter Editor, mit dem aus vorhanden Lerneinheiten, Content Packages nach der SCORM Spezifikation erzeugt, importiert, bearbeitet und exportiert sowie Metadaten erstellt werden können. Dieser SCORM Editor ist ein Open Source Projekt und kann unter [12] herunter geladen werden. Da der Reload Editor mittels der Programmiersprache Java entwickelt wurde, ist er plattformunabhängig und läuft somit unter Windows-, Linux-, und Mac-Betriebssystemen.

3.3.1 Erstellung eines Kurses unter Verwendung des Reload Editor 2004

Im Folgenden soll beschrieben werden, wie mit der aktuellsten Version des Reload Editors (Reload Editor 2004 v1.3.2 Beta2_c) ein SCORM 1.3 konformer E-Learning-Kurs erstellt werden kann. Dafür wird zunächst ein möglicher allgemeiner Produktionsablauf vorgestellt. Anschließend wird die Verwendung des Editors anhand eines konkreten Beispiels erläutert.

Im Allgemeinen sind folgende Produktionsschritte abzuarbeiten, wobei die Reihenfolge teilweise variieren kann:

1. Anlegen eines neuen SCORM 2004 Content Package
2. Ressourcen importieren
3. Content Organization definieren
    - Item-Elemente zuweisen und entsprechende Ressourcen zuordnen (Baumstruktur entspricht der Struktur der Lerneinheiten des Kurses)
4. Sequencing-Informationen zuweisen

5. Kurs und Lerneinheiten mit Metadaten beschreiben
6. Content Package zu einer PIF-Datei im Zip-Format packen

Anwendung des Reload Editors am Beispiel

Der in diesem Abschnitt beispielhaft mit dem Reload Editor umgesetzte Kurs, entspricht dem typischen Szenario eines Kurses für das Fraunhofer IIS/EAS. Die hierarchische Struktur des Menüs sieht wie folgt aus:

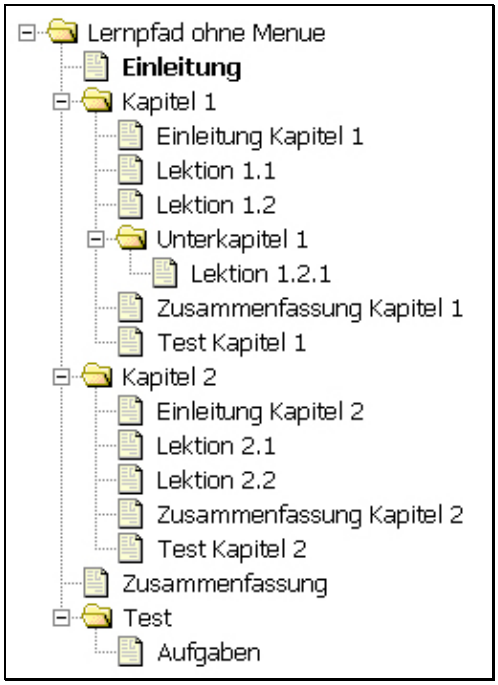


Abbildung 3-1: Menüstruktur des Beispielkurses

Der Kurs beinhaltet eine Einleitung, die einzelnen Kapitel, eine Zusammenfassung und einen Abschlusstest. Jedes Kapitel besteht dabei wiederum aus einer Einleitung, den einzelnen Lektionen, optional Unterkapiteln, einer Zusammenfassung und einem Test, dessen Aufgaben sich auf das aktuelle Kapitel beziehen.

Wird der Abschlusstest ausgewählt, werden die Aufgaben nach einander angezeigt. Während dessen ist nur noch eine Vorwärtsnavigation möglich, damit der Lernende keine Zugriff mehr auf die Lektionen oder voran gegangene Aufgaben hat. Nach Bearbeitung des Tests erscheint die Auswertung, welche ab jetzt auch im Menü auswählbar ist.

Der Test innerhalb eines Kapitels unterscheidet sich vom Abschlusstest dahingehend, dass ein Lernpfad anhand der Testergebnisse erstellt wird, bei dem alle nicht

bestandenen Lektionen wiederholt werden müssen. Der Lernende kann selbst entscheiden, ob er den Lernpfad abarbeiten will, oder sich weiter selbstständig durch das Menü navigiert. Innerhalb des Lernpfades ist nur die seitenweise Vor- und Rückwärtsnavigation möglich, das Menü ist ausgeblendet. Sind die zu wiederholenden Lektionen fertig abgearbeitet, erscheint wieder das vollständige Menü des Kurses, mit welchem der Lernende wieder freien Zugriff auf alle Lerneinheiten erhält.

1. Anlegen eines neuen SCORM 2004 Content Package

Über das Menü *File* und *New* bzw. über den entsprechenden Button wird ein Dialogfenster geöffnet, in dem aus dem Arbeitsplatz ein vorhandener Ordner ausgewählt oder ein neuer Ordner als Speicherort für das Projekt angelegt werden kann. Danach erscheint im Reload Editor folgendes Projektfenster:

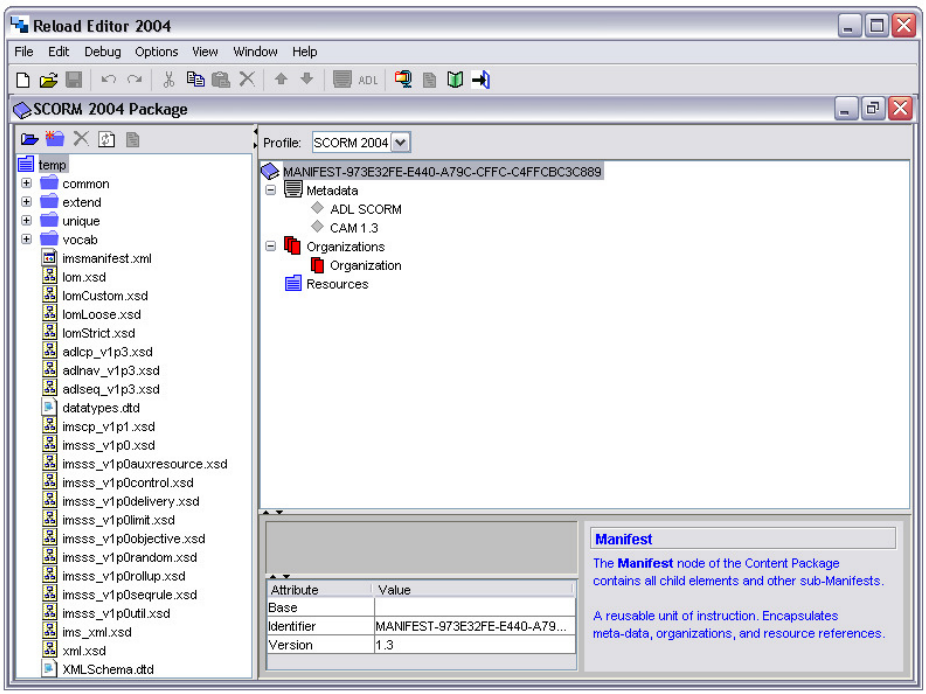


Abbildung 3-2: Projektfenster

Im linken Frame werden die Ressourcen der Projektdatei in einem Treeview angezeigt. Der Reload Editor hat automatisch alle benötigten XML-Schemata sowie die Manifest-Datei (Grundgerüst) erzeugt und im Projektordner gespeichert.

In der rechten Spalte wird die Kursstruktur ebenfalls in einem Treeview angelegt. Auch hier wurde bereits ein Grundgerüst aus folgenden Manifest-Elementen erzeugt:

- die obligatorischen Content Aggregation Metadaten
- der *Organizations*- und ein *Organization*-Knoten, in dem die Content Organization erstellt wird
- *Resources*-Knoten, dem automatisch die verwendeten Ressourcen der Lerninhalte und Lerneinheiten zugeordnet werden

Im unteren Frame können Eingaben erfolgen und ein Hilfetext für das aktuell markierte Element wird angezeigt.

2. Ressourcen importieren

Nachdem ein neues Projekt angelegt wurde, müssen alle benötigten Ressourcen, also die Lerneinheiten sowie die Lerninhalte und alle Dateien, die sie referenzieren (eventuell Stylesheets, die APIWrapper-Datei etc.), importiert werden. Dies geschieht entweder über das Menü *File* → *Import Resources* oder über den entsprechenden Button des Projektfensters. Es wird ein Dialogfenster angezeigt, in dem die benötigten Ressourcen ausgewählt werden können. Diese werden dann in den Projektordner kopiert und im Ressourcen-Treeview des Projektfensters dargestellt.

3. Content Organization definieren

Die importierten Lerneinheiten müssen nun per Drag and Drop dem *Organization*-Element des rechten Treeview zugeordnet werden. Wurde dieses Element nicht automatisch erzeugt, muss dies nachträglich über einen Rechtsklick auf *Organizations* → *Add Organization* geschehen.

Die entstehende Struktur entspricht später der Menüstruktur. Die eingefügten Lerneinheiten stellen die Items der Content Organization dar und referenzieren die entsprechende Webseite, welche automatisch dem Ressourcenknoten zugeordnet wird. Wenn eine Lerneinheit über einem Knoten des Treeview abgelegt wird, an dem ein Einfügen möglich ist, erscheint ein PopUp-Fenster. Hier muss angegeben werden, ob es sich bei der Lerneinheit um ein SCO oder ein Asset handelt. Danach erscheint ein weiteres PopUp, in dem der Typ der auf *webcontent* gesetzt werden muss.

Im unteren Eingabeframe kann der automatisch generierte Name der Lerneinheit verändert werden. Dieser Name entspricht später der Bezeichnung des Menüeintrags.

Ein Item, welches eine Ressource referenziert, können keine Items untergeordnet werden. Dafür müssen „leere“ Items erstellt werden (Rechtsklick auf *Organization* bzw.

auf bereits erstellte *Item*-Elemente und *Add Item*). Der folgende Screenshot verdeutlicht diese Strukturierung. Auch die „leeren“ Items (im Reload Editor durch grüne Rechtecke gekennzeichnet) erscheinen später im Menü und sollten deshalb umbenannt werden.

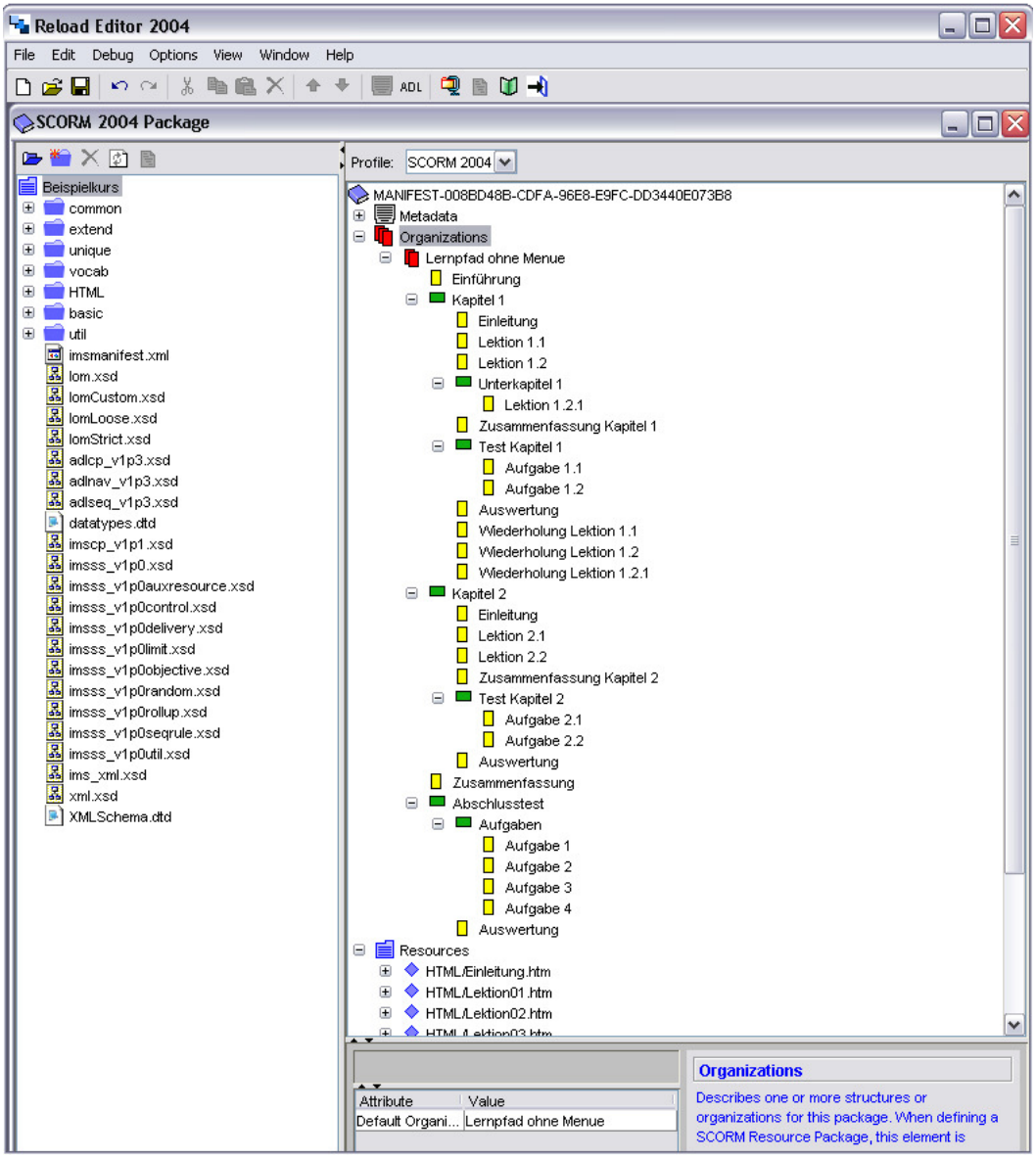


Abbildung 3-3: Strukturierung der Items

4. Sequencing-Informationen zuweisen

Die Lerneinheiten benötigen nun noch die gewünschten Sequencing-Informationen. Das Erstellen dieser Angaben soll im Folgenden für die einzelnen Items erläutert werden.

Zunächst sollen die *Control Modes* gesetzt werden, welche das Layout des Navigationsmenüs beeinflussen. In diesem Kurs sollen alle Lerneinheiten entsprechend

ihrer Struktur im Menü auswählbar sein und über Vor- und Zurück-Buttons abgearbeitet werden können.

Hierfür muss als erstes das Sequencing-Element zugeordnet werden. Dieses Element kann entweder an ein einzelnes Item oder an ein übergeordnetes „leeres“ Item per Rechtsklick und *Add Sequencing* angehangen werden. Einem übergeordneten Item zugeordnet, bestimmt es das Sequencing aller enthaltenden Einheiten der ersten Ebene.

Es muss also je ein Sequencing-Element dem Organization-Knoten, der einzelnen Kapitel und Unterkapitel zugeordnet werden. Mit Rechtsklick auf ein erstelltes *Sequencing*-Element und *Edit Sequencing* erscheint ein PopUp-Fenster, in dem die Registerkarte *Control Mode* ausgewählt wird. Für das gewünschte Layout müssen folgende Angaben gemacht werden:

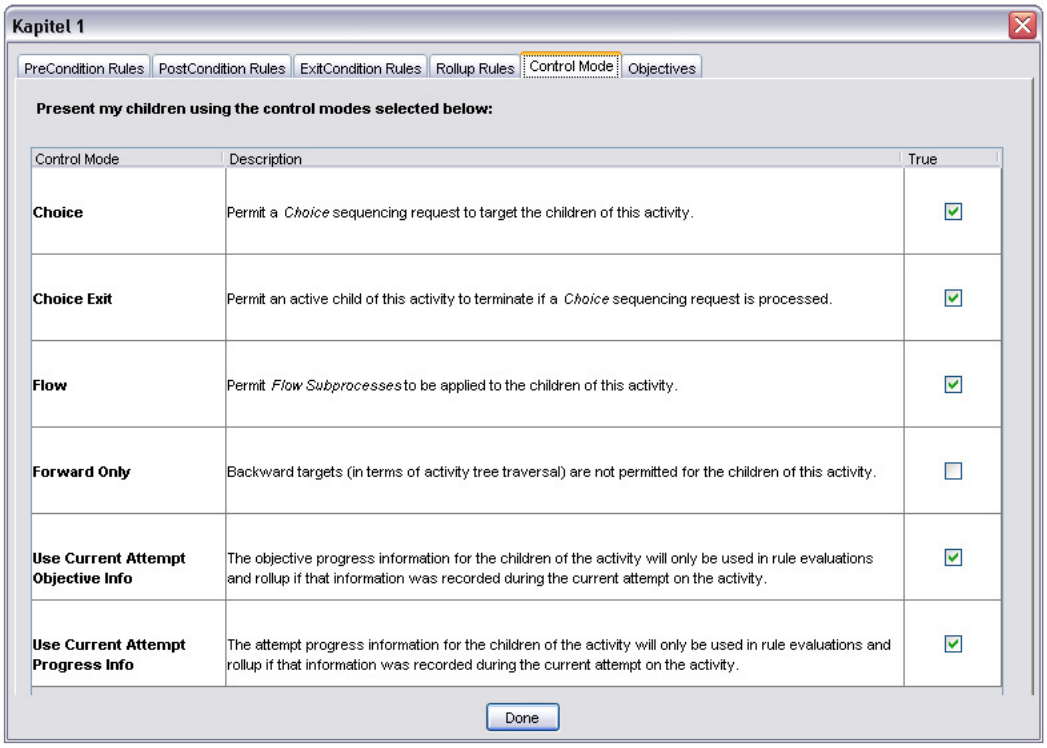


Abbildung 3-4: Control Mode Registerkarte

Nach der Bestätigung schließt sich das Fenster und dem *Sequencing*-Element wurde ein *Control Mode*-Element mit den ausgewählten Attributen zugeordnet. Dieses Element kann nun kopiert und wieder verwendet werden. Der entsprechende XML-Code in der Manifest-Datei sieht wie folgt aus:

```
<imsss:sequencing>
      <imsss:controlMode choice="true" choiceExit="true" flow="true" />
</imsss:sequencing>
```

Listing 3-2: Sequencing-Informationen – Control Modes für Lerneinheiten

Den Test-Items werden ebenfalls Sequencing-Elemente zugefügt, jedoch mit anderen Control Mode Attributen. Da die enthaltenen Testaufgaben nicht im Menü angezeigt werden dürfen und die Rückwärtsnavigation nicht möglich sein soll, wird in der Registerkarte *Choise* und *Choise Exit* deaktiviert, *Flow* und *Forward Only* aktiviert. Im unteren Eingabefenster können diese Attribute jederzeit verändert werden. Folgender Code wird in der Manifest-Datei erzeugt:

```
<imsss:sequencing>
      <imsss:controlMode choice="false" choiceExit="false" flow="true"
            forwardOnly="true" />
</imsss:sequencing>
```

Listing 3-3: Sequencing-Informationen – Control Modes für Test-Items

Etwas komplizierter ist es bei der Auswertungsseite für die Testergebnisse. Diese soll erst dargestellt werden, wenn der Test bereits abgearbeitet wurde. Dafür wird ein *Sequencing*-Element dem Auswertungs-Item zugeordnet und im PopUp die Registerkarte *PreConditionRule* ausgewählt. Hier kann festgelegt werden, dass dieses Item erst im Menü erscheint, wenn es bereits vorher einmal angezeigt wurde. (Nach Beendung des Tests wird die Auswertung durch Vorwärtsnavigation automatischdargestellt.) Folgende Eingaben sind zu vorzunehmen:

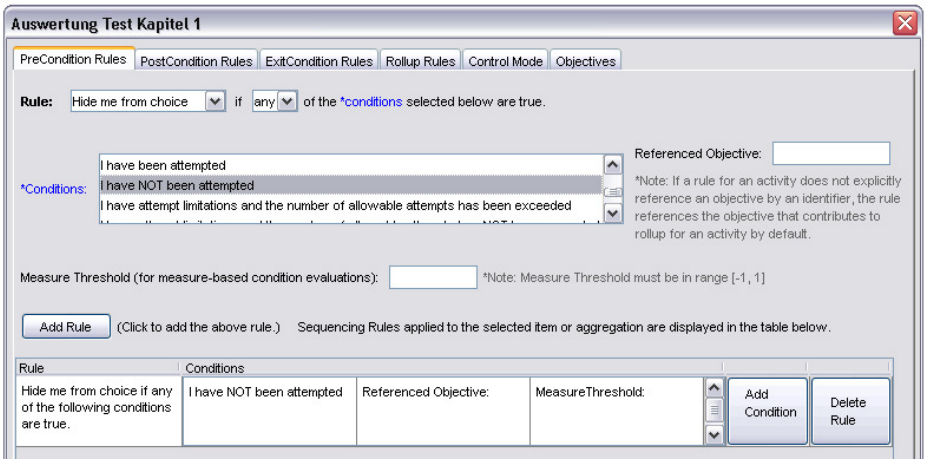


Abbildung 3-5: PreCondition Rule Registerkarte

Bei der Auswertung ist außerdem zu beachten, dass von ihr aus keine Rückwärtsnavigation möglich ist, da sonst die letzte Testaufgabe erscheinen würde. Eigentlich könnte dies durch die Aktivierung des *Control Mode - Forward Only* erreicht werden, aber die Control Mode Angaben des übergeordneten Items (Kapitel-Item) dominieren, weshalb eine andere Lösung notwendig ist. Mit Rechtsklick auf die Auswertung kann das Element *ADL Presentation* hinzugefügt werden. Diesem wird das Element *ADL Navigation Interface* zugeordnet, welches wiederum das *ADL Hide LMSUI* Element mit dem Attribut *previous* erhält, wodurch der Rückwärts-Button (previous) für diese Seite vom LMS nicht dargestellt wird. Diese Elemente können für die weiteren Auswertungen kopiert werden. Der entsprechende XML-Block der Manifest-Datei sieht wie folgt aus:

```
<item identifier="AUSWERTUNG1" identifierref="RES_AUSWERTUNG1">
      <title>Auswertung Test Kapitel 1</title>
      <adlnav:presentation>
            <adlnav:navigationInterface>
                  <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
            </adlnav:navigationInterface>
      </adlnav:presentation>
      <imsss:sequencing>
            <imsss:sequencingRules>
                  <imsss:preConditionRule>
                        <imsss:ruleConditions conditionCombination="any">
                              <imsss:ruleCondition operator="not"
                                    condition="attempted" />
                        </imsss:ruleConditions>
                        <imsss:ruleAction action="hiddenFromChoice" />
                  </imsss:preConditionRule>
            </imsss:sequencingRules>
      </imsss:sequencing>
</item>
```

Listing 3-4: XML-Block für Auswertungs-Item

Der aufwendigste Teil ist jedoch die Generierung des Lernpfades. Hierfür werden zunächst alle Lektionen, die nach diesem Test wiederholt werden können, in der richtigen Reihenfolge hinter der Auswertung eingefügt. Die Lernpfadstruktur für den Test des Kapitels 1 dieses Kurses entspricht der folgenden Abbildung:

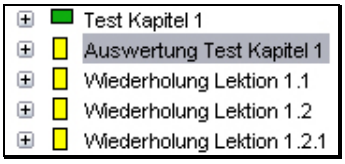


Abbildung 3-6: Screenshot des Reload Editors – Lernpfad

Damit die Testergebnisse ausgewertet und der daraus resultierende Lernpfad ermittelt werden können, muss der Lernerfolg in so genannten Objectives gespeichert werden. Dafür muss zunächst folgende Struktur für eine Testaufgabe durch wiederholtes Rechtsklicken auf das entsprechende Element erstellt werden:

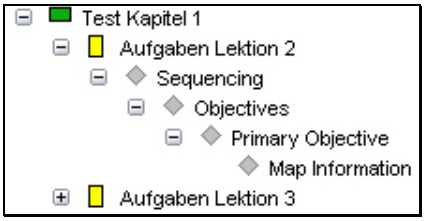


Abbildung 3-7: Hinzufügen von Map Informations der Aufgaben

Für das *Primary Objective* ist im Eingabefenster eine beliebige *Objective ID* zu vergeben. Dies kann leicht vergessen werden, da der Reload Editor keine Eingabe verlangt, sie ist aber notwendig.

Bei der Erstellung des *Map Information* Elementes erscheint ein PopUp, in dem eine eindeutige ID für die zu referenzierende Lektion (die Lektion die bei Nichtbestehen wiederholt werden soll) eingegeben werden muss. Die Werte der Attribute müssen im Eingabefenster wie folgt gesetzt werden:

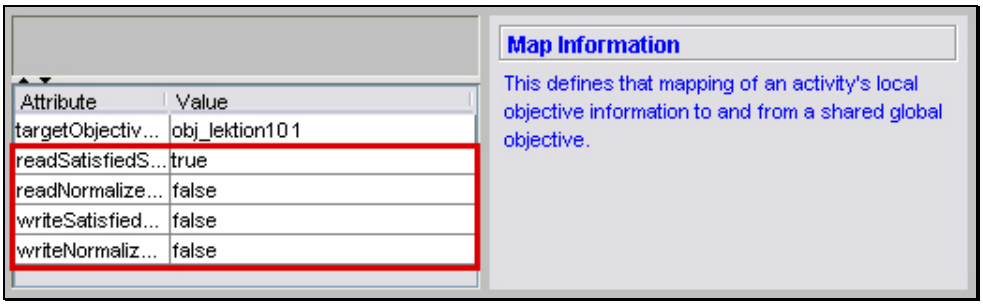


Abbildung 3-8: Map Informations der Aufgaben

Durch diese Angaben wird der Status der Aufgabe in das globale Objective geschrieben. Das Sequencing-Element kann für die weiteren Aufgaben kopiert werden, allerdings muss die ID für die Ziellektion (*targetObjectiveID*) geändert werden. Nachstehend wird der entsprechende XML-Code der Manifest-Datei aufgeführt:

```
<item identifier="AUFGABEN1_01" identifierref="RES_AUFGABEN_01">
      <title>Aufgabe Lektion 1.1</title>
      <imsss:sequencing>
            <imsss:objectives>
                  <imsss:primaryObjective objectiveID="PRIMARY">
                        <imsss:mapInfo targetObjectiveID="obj_lektion101"
                                       readSatisfiedStatus="false"
                                       readNormalizedMeasure="false"
                                       writeSatisfiedStatus="true"
                                       writeNormalizedMeasure="true" />
                  </imsss:primaryObjective>
            </imsss:objectives>
      </imsss:sequencing>
</item>
```

Listing 3-5: XML-Block für Testaufgaben-Item

Sollen bei Nichtbestehen einer Aufgabe mehrere Lektionen wiederholt werden, zum Beispiel wenn die Lektion weitere Unterkapitel enthält, wird einfach für alle dieselbe *targetObjectiveID* gesetzt. In diesem Beispiel wären das die Wiederholungslektion 1.2 und 1.2.1.

Für die Auswertung muss für jede Lektion ein lokales Objective angelegt werden, welches den Status des globalen Objectives dieser Lektion einliest. Folgende Struktur ist zu erstellen:

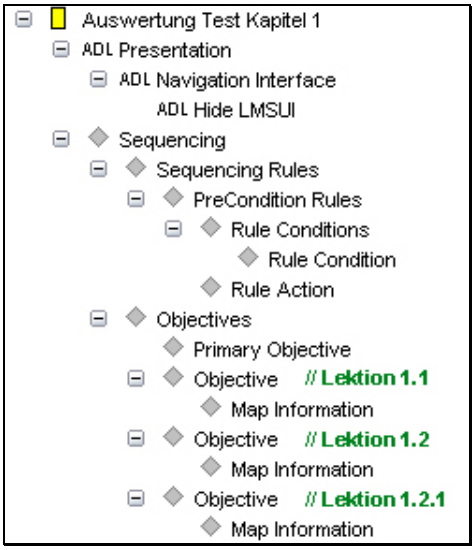


Abbildung 3-9: Objectives der Auswertung

Hierbei ist unbedingt zu beachten, dass die *Objective*-Elemente die *objectiveID* erhalten, die in der HTML-Seite der Auswertung angesprochen werden. Der Aufruf in der verwendeten Beispieldatei für die Auswertung (siehe Anhang C) lautet:

```
function getObjectivesStatus(lektion)
{
      var tempObjective = "local_obj_lektion0" + (lektion+1);
      ...
}
```

Listing 3-6: Funktion `getObjectivesStatus()`

Die *objectiveID* muss demzufolge für die erste Lektion `local_obj_lektion01` lauten. Die *targetObjectiveID* des *Map Information* Elements entspricht der *targetObjectiveID*, die für die entsprechende Ziellektion bei den Aufgaben-Items angegeben wurde. Der XML-Block für das Auswertungs-Item sieht nun wie folgt aus:

```
<item identifier="AUSWERTUNG1" identifierref="RES_AUSWERTUNG1">
      <title>Auswertung Test Kapitel 1</title>
      <adlnav:presentation>
            <adlnav:navigationInterface>
                  <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
            </adlnav:navigationInterface>
      </adlnav:presentation>
      <imsss:sequencing>
            <imsss:sequencingRules>
                  <imsss:preConditionRule>
                        <imsss:ruleConditions conditionCombination="any">
                              <imsss:ruleCondition operator="not"
                                    condition="attempted" />
                        </imsss:ruleConditions>
                        <imsss:ruleAction action="hiddenFromChoice" />
                  </imsss:preConditionRule>
            </imsss:sequencingRules>
            <imsss:objectives>
                  <imsss:primaryObjective />
                  <imsss:objective objectiveID="local_obj_lektion01">
                        <imsss:mapInfo targetObjectiveID="obj_lektion101" />
                  </imsss:objective>
                  <imsss:objective objectiveID="local_obj_lektion02">
                        <imsss:mapInfo targetObjectiveID="obj_lektion102" />
                  </imsss:objective>
            </imsss:objectives>
      </imsss:sequencing>
</item>
```

Listing 3-7: XML-Block für Auswertungs-Item

Anschließend sind den Wiederholungslektionen die *Map Informations* zuzuordnen. Die zu erstellende Struktur der Sequencing-Elemente entspricht den Testaufgaben (siehe Abbildung 3-5). Allerdings sollen diese den Status der Lektion lesen und nicht setzen. Die Attribute der Map Informations müssen also dementsprechend verändert werden:

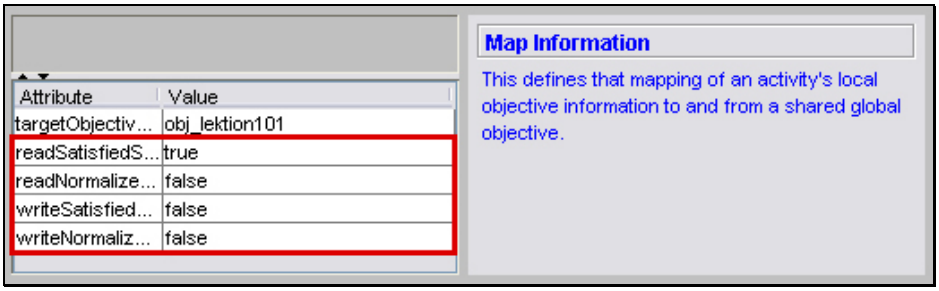


Abbildung 3-10: Map Informations der Wiederholungslektionen

Folgender XML-Code wird in der Manifest-Datei für eine Wiederholungslektion generiert:

```
<item identifier="WIEDERHOLUNG1_01" identifierref="RES_LEKTION_01">
      <title>Wiederholung Lektion 1.1</title>

      // Verweis auf eine Sequencing Collection (siehe im Folgenden)
      <imsss:sequencing IDRef="WIEDERHOLUNG">
            <imsss:objectives>
                  <imsss:primaryObjective objectiveID="PRIMARY">
                        <imsss:mapInfo targetObjectiveID="obj_lektion101"
                                       readSatisfiedStatus="true"
                                       readNormalizedMeasure="false"
                                       writeSatisfiedStatus="false"
                                       writeNormalizedMeasure="false" />
                  </imsss:primaryObjective>
            </imsss:objectives>
      </imsss:sequencing>
</item>
```

Listing 3-8: XML-Block für Wiederholungslektions-Item

Alle weiteren Sequencing-Informationen der Wiederholungslektionen werden in einer Sequencing Kollektion zusammengefasst. Dafür wird das Manifest-Element (Root) markiert und über einen Rechtsklick *Add sequencingCollection* ausgewählt. Dieses Element wird vom Reload Editor oberhalb des Ressourcen-Elements im Treeview eingefügt. Beim Import in ein LMS stellt man allerdings fest, dass dieses Element nach dem Ressourcen-Knoten eingefügt werden muss, damit der Kurs SCORM-konform ist. Daher muss die Datei `imsmanifest.xml` im rechten Ressourcen-Frame durch Rechtsklick und *View File* oder den entsprechenden Button, geöffnet werden. Der XML-Block `<imsss:sequencingCollection>` muss manuell hinter das `<resources>` Element verschoben werden.

Damit während der Abarbeitung des Lernpfades das Menü ausgeblendet wird, ist das Element *Control Mode* einzufügen und das Attribut *flow* auf *true* zusetzen, alle anderen Attribute auf *false*.

Da die Wiederholungslektionen aber immer noch im Menübaum angezeigt werden, ist zusätzlich eine *PreCondition Rule* anzulegen. Hierfür wird dem *Sequencing*-Element das *Sequencing Rules*-Element zugeordnet und diesem wiederum eine *PreCondition Rule*. Im aufgehenden PopUp-Fenster ist für die Rule Action *hiddenFromChoise* auszuwählen. Der *PreCondition Rule* ist nun noch eine *Rule Condition* hinzuzufügen, für die der Wert des Attributes *Condition* im unteren Eingabefenster auf *always* gesetzt werden muss. Die Wiederholungslektionen werden so nie im Menü angezeigt.
Um zu erreichen, dass wirklich nur die Lektionen im Lernpfad angezeigt werden, die nicht bestanden wurden, ist eine weitere PreCondition Rule anzulegen. Die Rule Action erhält den Wert *skip* und die Condition *satisfied*. Eine Wiederholungslektion wird also übersprungen, wenn sie erfolgreich bestanden wurde.
Für das Sequencing-Element ist nun noch eine ID zu vergeben, die den Wiederholungen als Sequencing *IDRef* übergeben wird (siehe oben stehender XML-Code). Abbildung 3-11zeigt die entstandene Struktur der Sequencing Kollektion. Der XML-Code für die beschriebene Sequencing Collection ist im Listing 3-9 zu sehen.

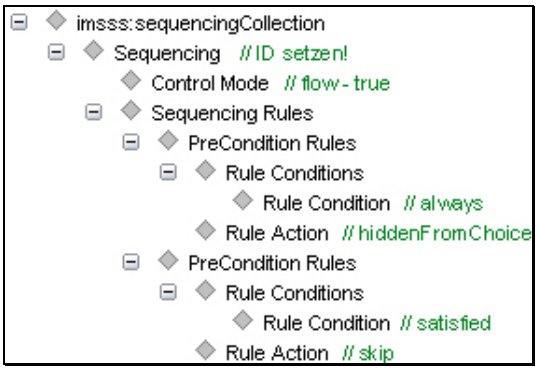


Abbildung 3-11: Sequencing Collection

Die komplette Struktur des fertig erstellten Kurses befindet sich im Anhang E. Die aus diesem Beispiel resultierende Manifest-Datei sowie das komplette Beispiel sind auf der beiliegenden CD zu finden.

```
<imsss:sequencingCollection>
      <imsss:sequencing ID="WIEDERHOLUNG">
            <imsss:controlMode choice="false" choiceExit="false" flow="true"
                  forwardOnly="false" />
            <imsss:sequencingRules>
                  <imsss:preConditionRule>
                        <imsss:ruleConditions>
                              <imsss:ruleCondition condition="satisfied" />
                        </imsss:ruleConditions>
                        <imsss:ruleAction action="skip" />
                  </imsss:preConditionRule>
                  <imsss:preConditionRule>
                        <imsss:ruleConditions>
                              <imsss:ruleCondition condition="always" />
                        </imsss:ruleConditions>
                        <imsss:ruleAction action="hiddenFromChoice" />
                  </imsss:preConditionRule>
            </imsss:sequencingRules>
            <imsss:rollupRules rollupObjectiveSatisfied="false" />
      </imsss:sequencing>
</imsss:sequencingCollection>
```

Listing 3-9: XML-Block für Sequencing Collection

5. Kurs und Lerneinheiten mit Metadaten beschreiben

Nun sollten eigentlich noch die Metadaten erstellt werden. Beim Reload Editor gibt es dabei allerdings folgendes Problem: Die Metadaten-Elemente können zwar hinzugefügt und in einem Eingabeformular editiert werden, allerdings handelt sich dabei um die Metadaten-Elemente aus SCORM 1.2. Da diese nicht mit denen aus der aktuellen Version kompatibel sind, wird ein LMS oder ein SCORM Viewer den Import eines solchen Kurses nicht akzeptieren. RELOAD hätte daher lieber ganz auf den Metadateneditor verzichten sollen. Unter Umständen kann bereits sehr viel Aufwand in die Beschreibung des Kurses durch Metadaten erfolgt sein, bis dieses Problem sichtbar wird.

6. Content Package zu einer PIF-Datei im Zip-Format packen

Um den Kurs in ein Zip-Archiv zu packen, welches dann von einem LMS oder einem Viewer eingelesen werden kann, muss lediglich der entsprechende Button oder im Menü *File* und *Zip Content Package* ausgewählt werden. Es wird ein Dialogfenster geöffnet in dem der Speicherort ausgewählt und der Name der Zip-Datei vergeben wird. Das Content Package kann nun veröffentlicht beziehungsweise getestet werden.

3.3.2 Vor- und Nachteile des Reload Editor 2004

In diesem Abschnitt sollen die Vor- und Nachteile, die sich aus der Arbeit mit dem Reload Editor 2004 ergeben, aufgezeigt werden. Nach einer allgemeinen Gegenüberstellung (siehe Tabelle 3-5) erfolgt eine Untersuchung in wie weit dieses Programm für die Kurserstellung speziell für das Fraunhofer IIS/EAS Verwendung finden kann.

Vorteile	Nachteile
Open Source	durch umfangreiche Funktionalität ist die Bedienung sehr komplex unüberschaubar
plattformunabhängig	für die Erstellung von komplexen Kursen (Sequencing) sind teifgreifende Kenntnisse in SCORM 1.3 erforderlich
benutzerfreundliche Oberfläche - Zuordnung der Ressourcen per Drag & Drop - Logische Elemente (Buttons, Symbole) - Eingabe von Sequencing-Informationen über Formulare	Fehler des Anwenders werden nicht erkannt und so nicht SCORM konforme Manifest-Dateien erzeugt (Fehler werden erst beim Import in ein LMS / Viewer erkannt)
Möglichkeiten für Sequencing und Navigation komplett nach SCORM 1.3 umsetzbar	Noch verwendete Ressourcen können aus dem Ressourcen-Treeview ohne Warnung entfernt werden (Fehlermeldung erst beim Import in ein LMS/Viewer)
keine XML-Kenntnisse erforderlich	Das SequencingCollection-Element wird an falscher Position erstellt (Fehler erst beim Import in ein LMS/Viewer sichtbar)
	Es können keine SCORM 1.3 Metadaten erstellt werden (es werden SCORM 1.2 Metadaten erstellt was zu Konformitäts-problemen in der Manifest-Datei führt)
	Kein Frageneditor
	Keine Hilfe oder Tutorials für die Erstellung von Sequencing-Informationen

Tabelle 3-5: Vor- und Nachteile des Reload Editor 2004

Der Reload Editor ist ein sehr komplexes Programm mit Hilfe dessen sehr anspruchsvolle Kurse erstellt werden können. Hierfür sind allerdings tiefgreifende SCORM 1.3 Kenntnisse, vor allem für das Sequencing, notwendig. Da die Spezifikation überaus umfangreich ist, muss für die Einarbeitung mit einem erheblichen Zeitaufwand gerechnet werden.

Auf Grund der hohen Komplexität des Programms beziehungsweise der umfangreichen SCORM-Spezifikation sind fehlerhafte Eingaben nahezu unvermeidlich. Diese Fehler werden vom Reload Editor leider nicht erkannt, wodurch es zur Generierung von nicht SCORM konformen Kursen kommt.

Metadaten sind zwar für die Erstellung eines Kurses nicht vorgeschrieben, jedoch sind sie, wie schon mehrfach erwähnt, von großer Bedeutung für die Wiederverwendung von Kursen und einzelnen Lerneinheiten. Mit Hilfe des Reload Editors können keine SCORM 1.3 Metadaten erstellt werden. Beim Einfügen von Metadaten werden stattdessen SCORM 1.2 Metadaten ohne Warnung erzeugt.

Einsatz des Reload Editors für die Kurserstellung des Fraunhofer IIS/EAS

Aufgrund der genannten Nachteile kommt die Verwendung des Reload Editors für das Fraunhofer IIS/EAS nicht in Frage. Bei den Autoren handelt es sich meist um wissenschaftliche Mitarbeiter, mit wenig oder keinen Kenntnissen im E-Learning-Bereich. Eine Einarbeitung in die SCORM-Spezifikation ist daher bereits aus Zeitgründen nicht möglich. Außerdem ist zu beachten, dass es keine spezialisierten Mitarbeiter für die Erstellung von E-Learning-Kursen gibt. Daher sollte im Idealfall möglichst jeder Wissenschaftler einen didaktisch anspruchsvollen Kurs ohne hohe Einarbeitungszeit erstellen können.

Erstellung einer erweiterten Anwendung

Trotz umfangreicher Recherchen im Rahmen dieser Diplomarbeit wurden keine Programme gefunden, die eine sehr einfache Kurserstellung auf einem trotzdem hohen didaktischen Niveau, nach der SCORM 1.3 Spezifikation ermöglichen. Deshalb soll eine neue Anwendung erstellt werden, die speziell auf die Anforderungen des Fraunhofer IIS/EAS zugeschnitten ist.

4 Konzeption eines SCORM Editors

In diesem Kapitel wird das Konzept für eine Anwendung vorgestellt, welche die benutzerfreundliche Entwicklung von E-Learning-Kursen nach der SCORM 1.3 Spezifikation ermöglicht. Voruntersuchungen, Vorbereitungen und Funktionalität werden beschrieben und Umfang des im Rahmen der Arbeit zu implementierenden Protyps wird festgelegt.

4.1 Aufgabenstellung

Für die benutzerfreundliche und intuitive Strukturierung von E-Learning-Kursen nach der SCORM 1.3 Spezifikation soll eine Entwicklungsumgebung entwickelt werden, welche die typischen Erfordernisse berücksichtigt, so wie diese am Beispiel des Fraunhofer IIS/EAS vorgefunden werden. Dabei sollen für die Kursentwicklung keinerlei JavaScript- und SCORM-Kenntnisse des Autors vorausgesetzt werden. Die Anwendung muss demnach, im Unterschied zum Reload Editor, die benötigten JavaScript-Befehle für die verschiedenen SCOs (Testaufgaben und Auswertungsseiten) generieren können und die entsprechenden Sequencing- und Navigations-Anweisungen selbstständig in die Manifest-Datei aufnehmen und vor dem Autor verbergen.

4.2 Voruntersuchung

Eine umfangreiche Untersuchung des Marktes hat ergeben, dass derzeit kein Produkt verfügbar ist, welches den Anforderungen des Fraunhofer-Institutes genügt (vgl. Kapitel 3). Daher ist eine erweiterte Anwendung zu entwickeln. Hierfür wird zunächst geprüft, in wie weit der Open Source Code des Reload Editors verwendet und erweitert werden kann, oder ob ein neues Programm erstellt werden muss.

4.2.1 Erweiterung des Reload Editors

Ein entscheidender Vorteil für die Erweiterung des Reload Editors stellt die Wiederverwendung bereits vorhandener Funktionen dar. Dazu gehören die Funktionen

zum Anlegen und Öffnen eines Projektes, zum Importieren von Ressourcen und der Zuordnung per Drag & Drop, sowie die Vorschau des Kurses und das Archivieren des Content Package.
Um die vorhanden Quellen nutzen zu können, ist allerdings eine Einarbeitung in sehr umfangreiche Quellen (umfasst über 350 Java-Klassen) notwendig.

4.2.2 Entwicklung eines neuen SCORM Editors

Trotz der Notwendigkeit der Programmierung aller benötigten Funktionen, sprechen eine Reihe von Vorteile für die Entwicklung einer neuen Anwendung.

- einfache Erweiterbarkeit um neue Funktionen in einem wesentlich kompakteren Quellcode
- Reduzierung des Produktionsaufwandes durch die Verwendung von Templates für das Erzeugen der Manifest-Datei
- durch Austausch der Templates oder Änderung von Attributen wie Sequencing-Informationen ist eine einfache Entwicklung neuer Kurs-Szenarien ohne Programmieraufwand möglich
- es ist anzunehmen, dass durch Austauschen der Templates auch künftige SCORM-Spezifikationen umgesetzt werden können
- Formulare für die Eingabe von SCORM 1.3 Metadaten können ebenfalls auf Basis von austauschbaren Templates implementiert werden
- Programmiersprache für die Implementierung der Anwendung ist frei wählbar

4.2.3 Entscheidung

Auch wenn bei der Erweiterung der Reload Editor Quellen einige Funktionen wieder verwendet werden können, spricht der Aufwand für die Einarbeitung in den umfangreichen Quellcode eher für die Entwicklung einer neuen Anwendung. Auf Grund der Einschränkung auf verschiedene Kurs-Szenarien und die damit verbundene Möglichkeit Templates zu verwenden, ist ein deutlich geringerer Quellcode-Umfang zu implementieren. Ein weiterer Vorteil ist die einfachere und übersichtlichere

Programmwartung und Erweiterung. Unter Auswägung aller Vor- und Nachteile wurde daher im Rahmen dieser Diplomarbeit entschieden, eine eigenständige Anwendung auf Basis von XML-Templates für die Erstellung der Manifest-Datei zu implementieren. Diese Anwendung wird im Weiteren als „KursEditor“ bezeichnet.

4.3 Vorbereitung

Bevor mit der Implementierung der Anwendung begonnen werden kann, müssen zunächst die Kursstruktur und deren relevante Sequencing-Regeln festgelegt werden, die mit Hilfe des KursEditors erstellt werden sollen. Diese basieren auf den in Kapitel 2 gewonnen Erkenntnissen über die Möglichkeiten der SCORM 1.3 Kurserstellung auf didaktischer Ebene, sowie auf Entscheidungen der Autorin sowie den verantwortlichen Mitarbeitern des Fraunhofer IIS/EAS nach Präsentationen und Diskussionsrunden. Die daraus entstandene Kursstruktur wird im Folgenden näher erläutert.

4.3.1 Relevante Kursstruktur für das Fraunhofer IIS/EAS

Grundstruktur eines Kurses

Jeder Kurs soll eine Einleitung, die einzelnen Lektionen, eine Zusammenfassung und einen Abschlusstest und/oder einen Vortest mit einer Auswertung der Lernergebnisse enthalten. Die Lektionen werden in Kapitel unterteilt und können in beliebiger Tiefe in Unterkapitel untergliedert werden.

Ein Kapitel enthält wiederum eine Einleitung, eine Zusammenfassung und einen Test mit Aufgaben zu den Lektionen des Kapitels.

Für jeden Test kann ein individueller Lernpfad generiert werden. Hierfür können den Aufgaben alle Lektionen des Kurses als Wiederholungslektionen zugeordnet werden. Für die Aufgaben, die vom Lernenden nicht bestanden wurden, werden die entsprechenden Wiederholungslektionen in einem Lernpfad angezeigt. Diese Grundstruktur wird in Abbildung 1-1 illustriert.

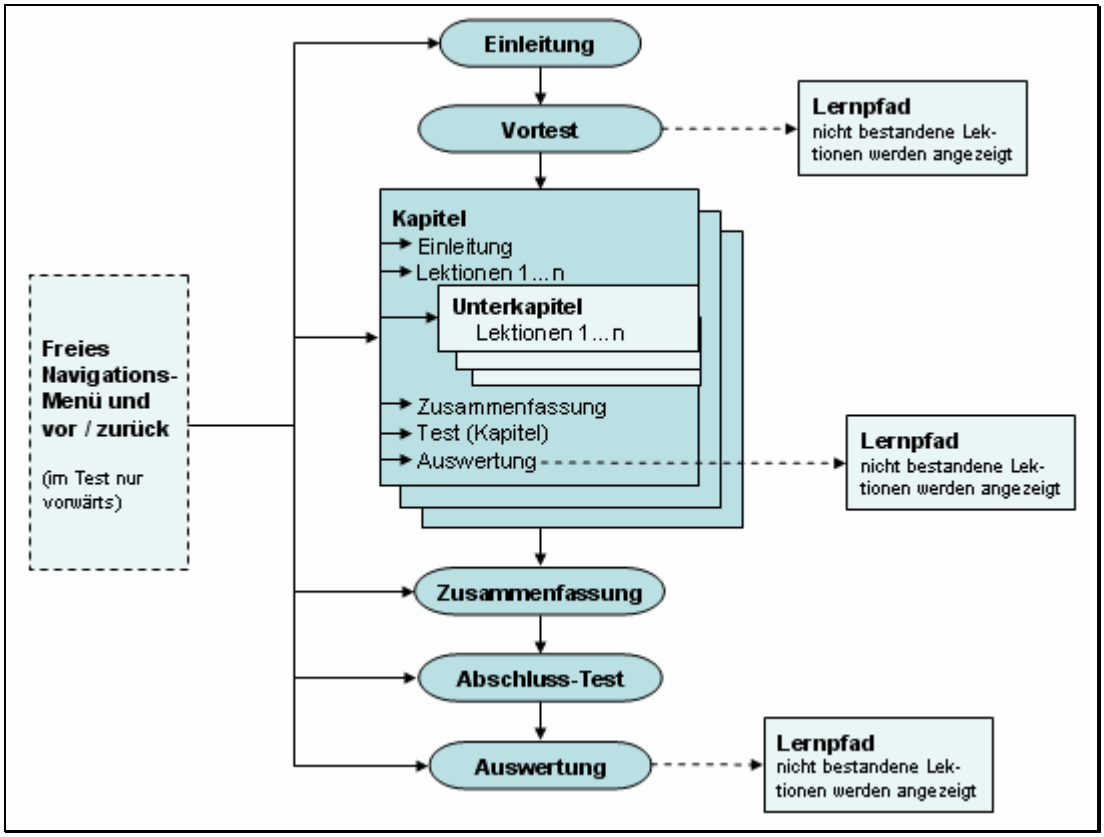


Abbildung 4-1: Grundstruktur eines Kurses

4.3.2 Sequencing-Regeln für die Kursstruktur

Generelle Sequencing-Regeln eines Kurses

Alle Lektionen werden in einem Menü angezeigt und können seitenweise durch Vor- bzw. Rückwärtsnavigation abgearbeitet werden. Innerhalb eines Tests ist kein Zugriff auf das Menü möglich und es besteht lediglich die Möglichkeit der Vorwärtsnavigation. Die Auswertungsseite eines Tests ist im Menü auswählbar, sobald der Test einmal bearbeitet wurde. Von dieser Auswertungsseite kann der Lernpfad gestartet werden.

Sequencing-Regeln für den individuellen Lernpfad

Soll dem Lernenden nach Bearbeitung der Testaufgaben ein Lernpfad präsentiert werden, muss jeder Aufgabenseite eine oder mehrere Wiederholungslektionen zugeordnet werden. Der Autor kann dabei zwischen folgenden Sequencing-Regeln für den Lernpfad wählen:

Lernpfad ohne Menüzugriff

Der Lernpfad kann lediglich durch Vor- und Rückwärtsnavigation abgearbeitet werden. Das Menü ist ausgeblendet. Der Lernpfad kann nur durch Beenden des Kurses unterbrochen werden.

Lernpfad mit Menüzugriff

Der Lernpfad kann ebenfalls nur durch Vor- und Rückwärtsnavigation bearbeitet werden. Das Menü ist jedoch eingeblendet. Der Lernende kann so jederzeit den Lernpfad verlassen.

Generell gilt, dass beim Verlassen des Lernpfades oder beim Beenden des Kurses der aktuelle Lernpfad gesichert wird. Der Lernpfad kann somit jederzeit fortgesetzt werden. Bereits bearbeitete Wiederholungslektionen werden aus dem Lernpfad entfernt.

4.4 Der KursEditor

Mit dem KursEditor soll eine Anwendung entwickelt werden, welche die Umsetzung der oben genannten Kursstruktur intuitiv und ohne Kenntnisse über die SCORM-Spezifikation ermöglicht. Der Editor muss dabei die Navigationsstruktur mit den entsprechenden Sequencing-Regeln erzeugen, einen Frageneditor für das Erstellen SCORM-konformer Testaufgaben und der Auswertungsseite zur Verfügung stellen und die Eingabe von Metadaten ermöglichen. Die Funktionalität sowie die Gestaltung der grafischen Bedienoberfläche werden im Folgenden beschrieben.

4.4.1 Funktionalität des KursEditors

Allgemeine Anforderungen

- Intuitive Bedienung (auch ohne Kenntnisse in SCORM)
- Wiederverwendbarkeit
- Erweiterbarkeit
- Betriebssystem: Microsoft Windows XP

Spezifische Funktionalitätsanforderungen

- Erzeugen von Projektdateien für die Speicherung und Wiederverwendung von Kursen
- Speichern, Schließen und Öffnen von Projektdateien
- Anlegen eines Projektordners für alle benötigten Ressourcen (Schemata, Manifest-Datei, Lerneinheiten)
- Import von Ressourcen (Speicherung im Projektordner)
- Archivierung des Projektordners im Zip-Format für den Import in ein LMS
- Bereitstellung einer Hilfefunktion sowie eines Tutorials

Anforderungen an die Bedienoberfläche

Hauptmenü

- Erstellen, Öffnen, Speichern und Schließen der Projektdatei
- Generierung des Zip-Archivs für den Import in ein LMS/Viewer
- Projekteinstellungen
- Anzeigen der Hilfe oder des Tutorials
- Beenden des Programms

Ressourcen-Treeview

- Treeview, in dem alle importierten Ressourcen angezeigt werden
- Löschen der Ressourcen (Meldung, wenn sie bereits verwendet werden)
- Anzeigen der Ressourcen in dem entsprechenden Programm (Programm, welches unter Windows für den entsprechenden Datei-Typ als Standard-programm eingestellt ist)

Kurs-Treeview

- Treeview, in dem ein automatisch generiertes Grundgerüst dargestellt wird
- Zuordnung der HTML-Seiten aus dem Ressourcen-Treeview per Drag & Drop in den Kurs-Treeview (innerhalb des Kurs-Treeview wird von Lerneinheiten gesprochen)

- Eingabefenster für das Umbenennen von Lerneinheiten beim Drag & Drop in den Kurs-Treeview
- Umbenennen der Lerneinheiten und der im Menü des Kurses erscheinenden Elemente des Grundgerüsts in die späteren Menüeinträge
- Verschieben von Lerneinheiten durch Drag & Drop innerhalb einer Ebene durch entsprechende Buttons
- Löschen von Lerneinheiten (Meldungen bei eventuellen Abhängigkeiten)
- Hinzufügen von Kapiteln und Unterkapiteln
- Zuordnen von Aufgaben-/Auswertungsseiten oder Erzeugen dieser Seiten in einem internen Frageneditor (Formulare für die nötigen Eingaben)
- Kontextmenü für das Element Test, in dem der Typ des Lernpfades gewählt werden kann
- Generierung von Lernpfaden für Kapiteltests durch Zuordnung von Aufgaben zu Lektionen, die bei Nichtbestehen wiederholt werden sollen
- Formulare für die Eingabe von SCORM 1.3 Metadaten

4.4.2 Gestaltung der grafischen Bedienoberfläche

Struktureller Aufbau der Softwareoberfläche

Die Oberfläche der Anwendung ist in fünf Bereiche gegliedert (Abbildung 4-2: Struktur der Softwareoberfläche). Auf der linken Seite wird der Ressourcen-Treeview abgebildet wird. Hier werden alle Lerninhalte und Lerneinheiten entsprechend ihrer hierarchischen Struktur im Projektordner angezeigt. Rechts neben dem Ressourcen-Treeview wird der Kurs-Treeview dargestellt. Er beansprucht den größten Teil der Softwareoberfläche. In diesem Bereich wird das Grundgerüst der Kursstruktur entsprechend des gewählten Szenarios angezeigt. Für die Erstellung eines Kurses müssen den verschiedenen Knoten die entsprechenden Lerneinheiten aus dem Ressourcen-Treeview per Drag & Drop zugeordnet werden.

Jedem Treeview ist eine eigene Menüleiste für alle Treeview spezifischen Befehle zugeordnet. Das Hauptmenü befindet sich am obersten Rand der Anwendung.

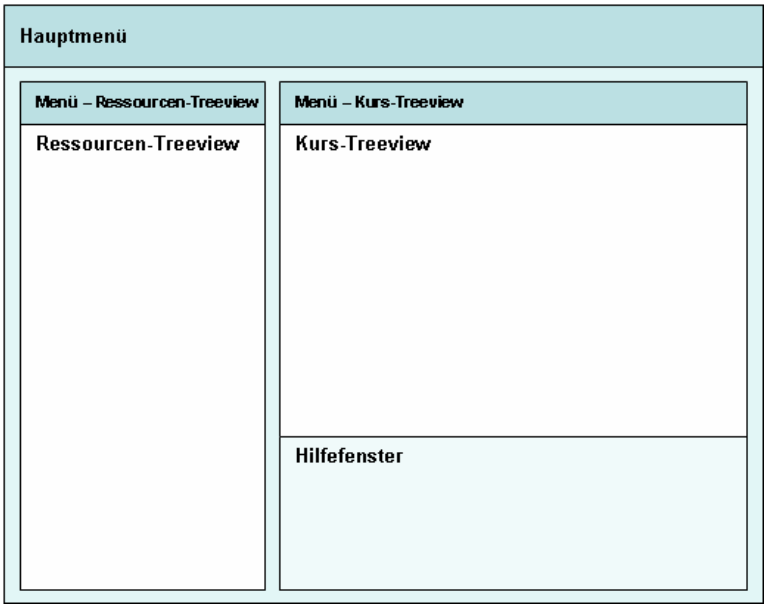


Abbildung 4-2: Struktur der Softwareoberfläche

Unterhalb des Kurs-Treeview wird ein Hilfefenster mit Informationen zu dem ausgewählten Element eines Treeview angezeigt. Der nachfolgende Screenshot zeigt die Umsetzung der strukturellen Oberflächengestaltung.

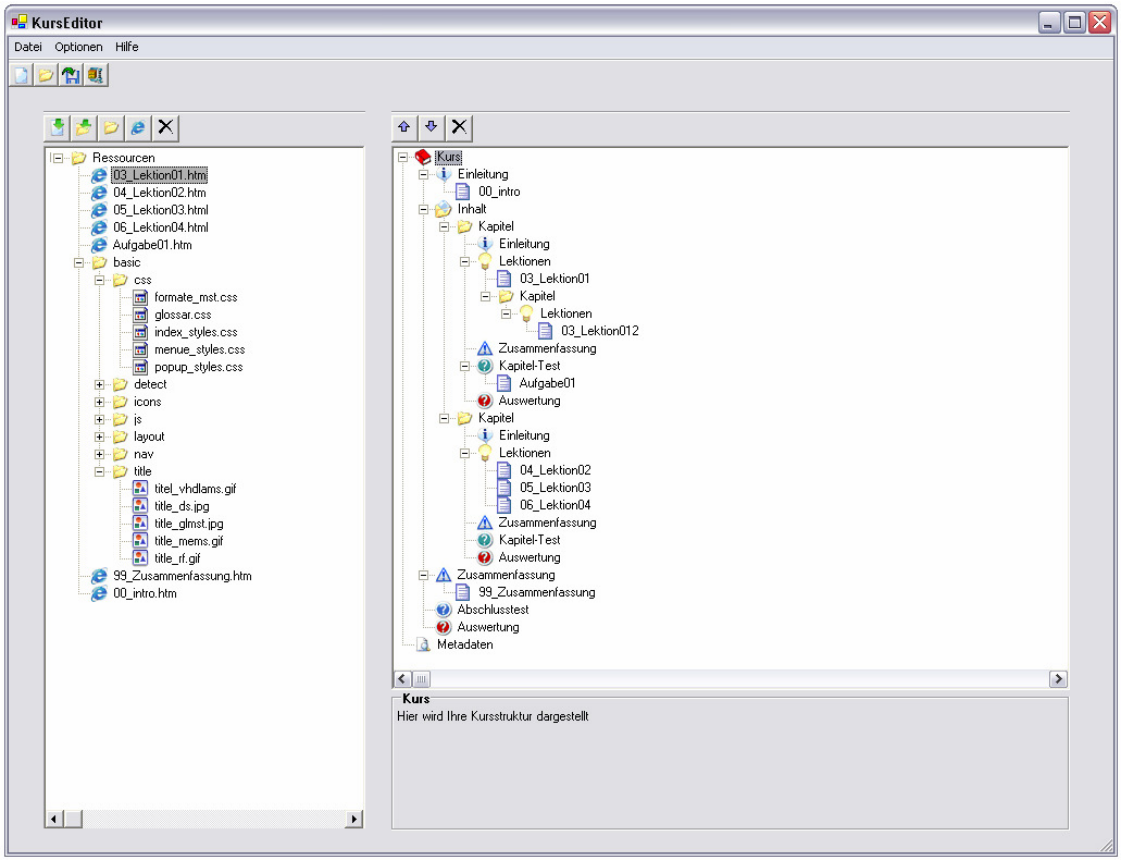


Abbildung 4-3: Screenshot des KursEditors

4.4.3 Funktionalität der grafischen Bedienoberfläche

Hauptmenü

Die Befehle des Hauptmenüs sind sowohl über ein Kontextmenü als auch durch Buttons über eine Symbolleiste (*Toolbar*) ausführbar. Sie können entweder mit der Maus oder über ein Tastenkürzel (*Shortcut*) aufgerufen werden. Der entsprechende Shortcut wird im Menü angezeigt (siehe Abbildung 4-4).

Abbildung 4-4: Kontextmenü des KursEditors

Über den Menüeintrag *Datei* können KursEditor Projekte erstellt, geöffnet, gespeichert und geschlossen werden. Die Befehle *Speichern* und *Schließen* sind deaktiviert bis ein Projekt geöffnet wurde (siehe Abbildung 4-4).

Wird ein neues Projekt erstellt, erscheint ein Dialogfenster in dem der Speicherort sowie der Dateiname gewählt werden muss. Dieser Dialog entspricht dem Windows-Design und ist daher intuitiv bedienbar. Es werden nur Ordner und Projektdateien angezeigt. Da sich in einem Projektordner jeweils nur eine Projektdatei befinden darf, erscheint eine Fehlermeldung, sollte der Anwender in einem Ordner eine Weitere erzeugen wollen. Die Erstellung des neuen Projektes wird daraufhin abgebrochen. War das Anlegen des Projektes erfolgreich, werden dem ausgewählten Ordner alle, für einen SCORM 1.3 konformen Kurs benötigten Schemata und die noch leere Manifest-DateiS hinzugefügt.

Beim Öffnen eines Projektes wird ein Dialogfenster angezeigt, in welchem bereits erstellte Projektdateien ausgewählt und geöffnet werden können. Für eine bessere Übersicht werden nur Projektdateien angezeigt.

Wird ein Projekt geschlossen oder das Programm beendet, erfolgt eine Abfrage, ob die Datei vor dem Schließen gespeichert oder die Aktion abgebrochen werden soll.

Über den *Optionen*-Eintrag kann der Befehl *Zip-Archiv generieren* ausgeführt werden und Projekteinstellungen vorgenommen werden, sobald sich ein Projekt im Bearbeitungsmodus befindet (siehe Abbildung 4-5).

Abbildung 4-5: Optionen-Eintrag des Hauptmenüs

Das Hilfefenster unterhalb des Ressourcen-Treeview lässt sich über die Menüeinträge *Hilfe* und *Hilfefenster* Schließen und Anzeigen. Ist das Hilfefenster aktiviert erscheint ein Häkchen vor dem Menüeintrag. Ein Tutorial kann ebenfalls über den Hilfe-Menüeintrag gestartet werden.

Die wichtigsten Befehle (Projekt erstellen, Öffnen, Speichern und Zip-Archiv generieren) können auch über Buttons ausgeführt werden. Dies ermöglicht einen schnellern Zugriff. Wird mit der Maus über einen Button gefahren, wird eine kurze Beschreibung des Befehls (*Tooltip*) angezeigt (siehe Abbildung 4-6).

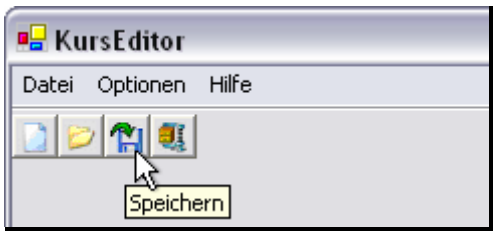


Abbildung 4-6: Toolbar des Mainmenüs

Die Schaltflächen *Speichern* und *Zip-Archiv generieren*, sind im Toolbar erst sichtbar wenn ein Projekt geöffnet wurde.

Ressourcen-Treeview

Im Ressourcen-Treeview werden die Ressourcen eines Projektes, mit Ausnahme der vom Programm automatisch generierten Dateien (Schemata, Manifest- und Projektdatei), angezeigt. Für den Import stehen im Menü des Treeviews zwei Buttons zur Verfügung.

Der Befehl *Dateien importieren*, öffnet ein Dialogfenster in dem eine oder mehrere Dateien ausgewählt werden können. Bestätigt der Anwender diese Auswahl, wird das Dialogfenster geschlossen, die Dateien werden in das Projektverzeichnis kopiert und im Ressourcen-Treeview angezeigt. Für die Kurserstellung wichtigsten Dateitypen, wie HTML-Seiten und Bilder, wird das entsprechende Windows-Icon vor dem Namen der Datei angezeigt. Ist der Typ der Datei nicht bekannt wird das Windows-Icon für unbekannte Dateien zugeordnet. (siehe Abbildung 4-7).

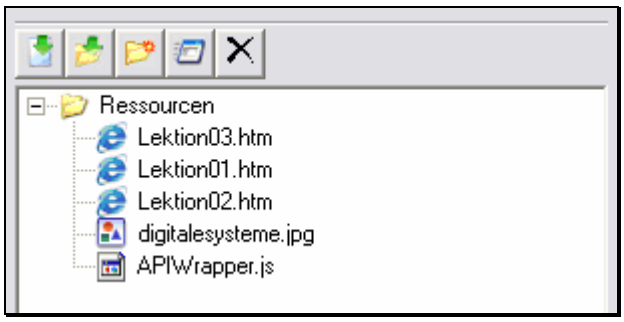


Abbildung 4-7: Ressourcen-Treeview

Die zweite Möglichkeit Ressourcen einzufügen besteht darin ganze Ordner zu importieren. Es wird ein Dialogfenster geöffnet, in dem ein Ordner aus der Verzeichnisstruktur ausgewählt werden kann. Dieser wird beim Import mit allen Unterordnern und Dateien in den Projektpfad kopiert und entsprechend der Struktur im Ressourcen-Treeview dargestellt.

In der Statusleiste wird der Verzeichnispfad der Ressource angezeigt, über der sich der Mauszeiger befindet (siehe Abbildung 4-8). Damit wird es dem Anwender, ermöglicht die Dateien im Verzeichnisbaum einfach und schnell zu lokalisieren.

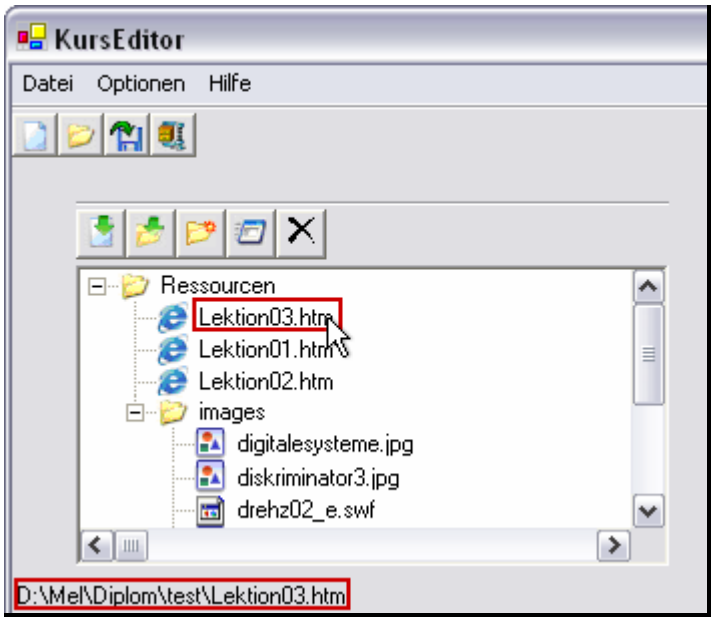


Abbildung 4-8: Verzeichnispfad-Anzeige einer Ressource

Menü des Ressourcen-Treeviews

Über das Menü des Ressourcen-Treeviews können Dateien bzw. Ordner importiert werden. Diese Ressourcen werden im Treeview angezeigt. Über einen weiteren Button können zusätzliche, noch leere, Ordner erzeugt werden, um die Ressourcen beliebig zu strukturieren. Über das Menü können Ressourcen gelöscht werden, wobei dem Anwender eine Meldung angezeigt wird, wenn diese Ressource bereits verwendet wird. Für HTML-Seiten gibt es zusätzlich die Möglichkeit der Anzeige in einem Browser-Fenster. Abbildung 4-9 zeigt das Menü.

Abbildung 4-9: Menü des Ressourcen-Treeviews

Kurs-Treeview

Im Kurs-Treeview wird die Struktur des zu entwickelnden Kurses dargestellt. Wurde ein neues Projekt erstellt, wird zunächst das Grundgerüst für den E-Learning-Kurs angezeigt. Für eine intuitive Benutzung erhalten alle Einträge entsprechend ihrer Bedeutung ein bestimmtes Icon. Dieses Grundgerüst sieht wie folgt aus:

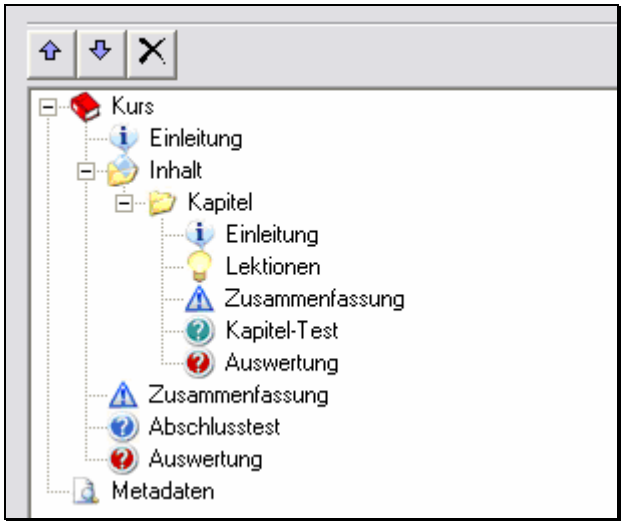


Abbildung 4-10: Grundgerüst eines Kurses im Kurs-Treeview

Per Drag & Drop können den einzelnen Knoten des Kurs-Treeviews die entsprechenden HTML-Seiten aus dem Ressourcen-Treeview zugeordnet werden. Für alle anderen Dateien ist dies nicht möglich, da für einen SCORM-konformen E-Learning-Kurs nur HTML-Seiten als Lerneinheiten zugefügt werden können. Wenn eine Ressource in den Kurs-Treeview kopiert wurde, wird sie in diesem Treeview als eine Lerneinheit dargestellt, welche die Ressource referenziert. Standardmäßig wird beim Verschieben einer HTML-Seite in den Kurs-Treeview ein Eingabefenster geöffnet in dem der Name für die zu erstellende Lerneinheit vergeben werden kann. Wird vom Anwender keine Eingabe vorgenommen, erhält die Ressource den Namen der HTML-Seite ohne deren Dateikennung. Dieses Fenster kann auf Wunsch unterdrückt werden.

Durch einen Rechtsklick auf den Knoten *Inhalt* und *Kapitel* können weitere Kapitel bzw. Unterkapitel hinzugefügt werden. Abbildung 4-11 verdeutlicht dies.

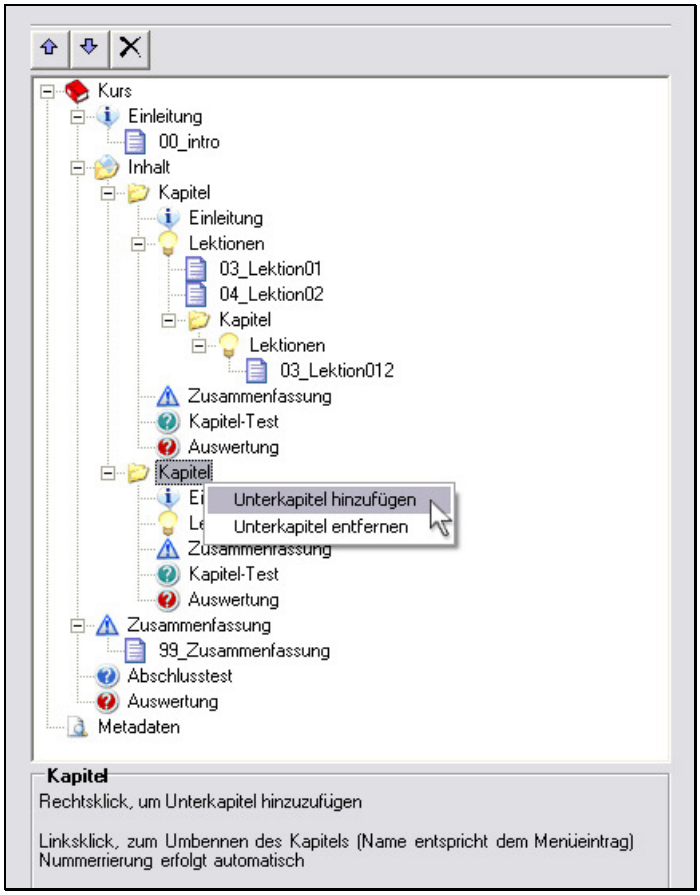


Abbildung 4-11: Beispielhafte Struktur eines Kurs-Treeviews

Eine Besonderheit bilden die Auswertungsseiten und die Lerneinheiten, die einem Test zugeordnet werden. Testaufgaben und deren Auswertung (SCOs) müssen entsprechend der SCORM-Spezifikation mit den nötigen API-Befehlen programmiert sein (Anhang C bzw. Anhang D). Um dem Autor des Kurses diese Arbeit abzunehmen, ist ein Frage-Editor im Programm integriert. Über das Kontextmenü eines Tests kann eine Aufgabe hinzugefügt werden. Es wird ein Formulareingabefenster geöffnet, in dem die nötigen Einstellungen gesetzt und der Fragetext eingegeben werden kann. Die entsprechende Auswertungsseite wird automatisch erzeugt. Da der Frage-Editor im Rahmen der Diplomarbeit nicht implementiert wird, soll nicht näher auf dessen Funktionalität eingegangen werden.

Über das Kontext-Menü eines Tests kann zusätzlich der Typ des gewünschten Lernpfades angegeben werden (vgl. Kapitel 4.3.1). Für die Generierung eines Lernpfades muss jeder Aufgabe eine Lektion zugeordnet werden, welche bei Nichtbestehen in den individuellen Lernpfad eingefügt wird. Mit einem Rechtsklick auf

die Aufgabe lässt sich das entsprechende Kontextmenü öffnen. In einem neuen Dialogfenster wird ein Treeview mit allen Knoten des Kurs-Treeviews abgebildet. Über ein Kontrollkästchen können Lerneinheiten der aktuellen Aufgabe zugewiesen werden. Abbildung 4-12 zeigt einen Screenshot dieses Dialogfensters.

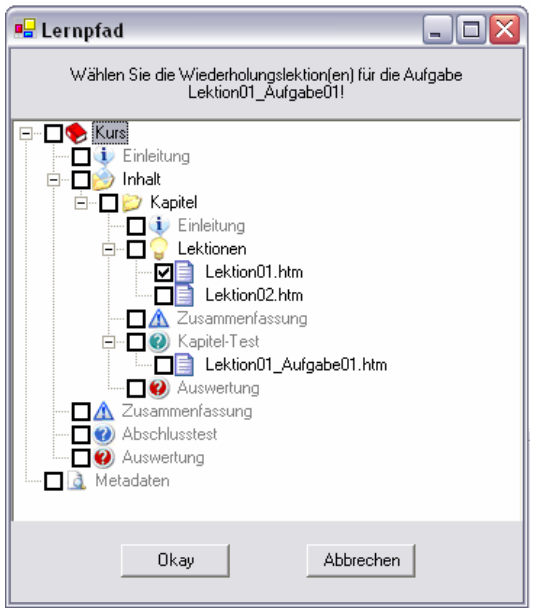


Abbildung 4-12: Dialogfenster für das Zuordnen von Wiederholungslektionen

Menü des Kurs-Treeviews

Das Menü des Kurs-Treeview besteht aus einem Toolbar mit Buttons zum Verschieben und Löschen von Lerneinheiten die dem Kurs-Treeview zugeordnet sind. Abbildung 4-13 zeigt einen Screenshot des Toolbars. Mit Hilfe der Pfeiltasten können die Lerneinheiten innerhalb ihrer Ebene verschoben werden, da dies nicht vom Drag & Drop unterstützt wird. Um ein versehentliches Entfernen einer Lerneinheit zu verhindern, erscheint eine Eingabeaufforderung in welcher der *Löschen*-Vorgang zu bestätigen ist.

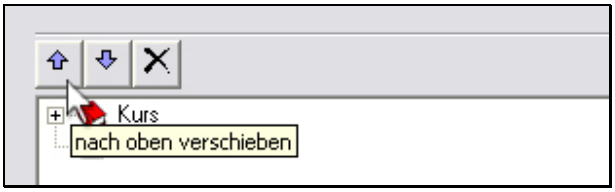


Abbildung 4-13: Toolbar des Menü des Kurs-Treeview

4.5 Prototyp des KursEditors

Im Rahmen dieser Diplomarbeit wird ein Prototyp der vom Fraunhofer IIS/EAS benötigten Anwendung implementiert. Mit diesem Prototyp können E-Learning-Kurse erstellt, gespeichert und weiterverarbeitet werden. Sie entsprechen der SCORM 1.3 Spezifikation und können nach der programminternen Zip-Archivierung in einem entsprechenden LMS/Viewer gestartet werden. Der Metadateneditor wird nur beispielhaft umgesetzt und auf die Entwicklung des Frageneditors wird verzichtet. Die Generierung der Lernpfade ist dennoch möglich, wenn die Aufgaben- und Auswertungs-Seiten bereits die nötigen API-Befehle enthalten. Hierfür werden im Kapitel 3 dieser Diplomarbeit entsprechende Beispiel-Dateien vorgestellt.

Funktionalität des Prototyps

Die Anforderungen an die Funktionalität des Prototyps werden in den folgenden Stichpunkten zusammengefasst:

Spezifische Funktionalitätsanforderungen

- Erzeugen von Projektdateien für die Speicherung und Wiederverwendung von Kursen
- Speichern, Schließen und Öffnen von Projektdateien
- Anlegen eines Projektordners für alle benötigten Ressourcen (Schemata, Manifest-Datei, Lerneinheiten)
- Import von Ressourcen (Speicherung im Projektordner)
- Archivierung des Projektordners im Zip-Format für den Import in ein LMS

Anforderungen an die Bedienoberfläche

Hauptmenü

- Erstellen, Öffnen, Speichern und Schließen der Projektdatei
- Generierung des Zip-Archivs für den Import in ein LMS/Viewer
- Projekteinstellungen
- Anzeigen der Hilfe bzw. des Tutorials
- Beenden des Programms

Ressourcen-Treeview

- Treeview, in dem alle importierten Ressourcen angezeigt werden
- Löschen der Ressourcen (Meldung wenn sie bereits verwendet werden)
- Anzeigen der Ressourcen in dem entsprechenden Programm

Kurs-Treeview

- Treeview, in dem ein automatisch generiertes Grundgerüst dargestellt wird
- Zuordnung der HTML-Seiten aus dem Ressourcen-Treeview per Drag & Drop in den Kurs-Treeview (innerhalb des Kurs-Treeview wird von Lerneinheiten gesprochen)
- Eingabefenster für das Umbenennen von Lerneinheiten beim Drag & Drop in den Kurs-Treeview
- Umbenennen der Lerneinheiten und der im Menü des Kurses erscheinenden Elemente des Grundgerüsts in die späteren Menüeinträge
- Verschieben von Lerneinheiten per Drag & Drop innerhalb einer Ebene durch entsprechende Buttons
- Hinzufügen von Kapiteln und Unterkapiteln
- Kontextmenü für das Element Test, in dem der Typ des Lernpfades gewählt werden kann
- Generierung von Lernpfaden für Kapiteltests durch Zuordnung von Aufgaben zu Lektionen, die bei Nichtbestehen wiederholt werden sollen
- Formulare für die Eingabe von SCORM 1.3 Metadaten

5 Prototypische Implementierung des KursEditors

In diesem Kapitel wird die prototypische Implementierung des KursEditors beschrieben. Nach einem kurzen Überblick über die verwendete Technologie, wird die Softwarearchitektur und Softwareentwicklung vorgestellt. Im Folgenden werden Möglichkeiten für die Weiterentwicklung aufgeführt. Ein Anwendungsbeispiel zeigt abschließend die intuitive Verwendung des KursEditors für die Erstellung SCORM-konformer E-Learning-Kurse mit individuellen Lernpfaden.

5.1 Erstellung der Templates für die Manifest-Datei

Die Implementierung des Kurseditors basiert auf der Verwendung von XML-Templates aus denen die Manifest-Datei generiert wird. Für die Erstellung dieser Templates wurden mit dem Reload Editor Beispielkurse erzeugt, die der gewünschten Kursstruktur entsprechen und die nötigen Sequencing- und Navigationsregeln enthalten. Aus den entstandenen Manifest-Dateien können dann die benötigten Templates entwickelt werden. Basis bildet zunächst folgendes Grundgerüst:

```
<?xml version="1.0" encoding="UTF-8" ?>
<manifest ... version="1.3">
      <metadata>
            <schema>ADL SCORM</schema>
            <schemaversion>CAM 1.3</schemaversion>
      </metadata>
      <organizations default="ORG01">
            <organization identifier="ORG01" adlseq:objectivesGlobalToSystem=
                  "false"> // globale Objectives sind nur für dieses Content
                           // Package gültig
                  <title>Kursname</title>
            </organization>
      </organizations>
      <resources>
      </resources>
</manifest>
```

Listing 5-1: XML-Code des Basis-Templates

Für die Generierung der vollständigen Manifest-Datei müssen die entsprechenden XML-Blöcke (`<manifest>`, `<metadata>`, `<organizaion>`, `<resources>`) um Templates für die verschiedenen Items (Kapitel, Lektionen, Testaufgaben usw.), Sequencing Rules, Ressourcen und Objectives ergänzt werden. Die entwickelten Templates befinden sich im Anhang der Arbeit.

Bestimmte Attribute, wie der Name einer Lektion, oder auch verschiedene Control Modes und Sequencing Rules, müssen durch das Programm gesetzt werden. Es empfiehlt sich, die Generierung der Manifest-Datei auf Basis der Templates zunächst durch manuelles Zusammenstellen zu testen, bevor sie in die Programmstruktur integriert werden.

5.2 Technologie für die Entwicklung

Für die Implementierung des KursEditors soll eine objektorientierte Programmiersprache verwendet werden. Diese sollte die Gestaltung einer anspruchsvollen Programmoberfläche ermöglichen und in hohem Maße unterstützen, da der Kurseditor viele grafische Objekte wie Menüs, Treeviews, Icons usw. enthalten soll. Für die Erstellung der Manifest-Datei aus XML-Templates ist eine umfangreiche XML-Funktionalität wünschenswert.

5.2.1 C# und das .NET Framework

.NET

Microsoft *.NET* wurde im Jahr 2002 von Microsoft veröffentlicht. Bei .NET handelt es sich um eine innovative Technologie für die Entwicklung und Ausführung verteilter Anwendungen wie zum Beispiel Internetbrowser, Handy-Browser, Windows- und PDA-Programme [13].

Bereits Ende der 1990er Jahre wurde mit der Entwicklung der .NET Technologie begonnen, hervorgerufen durch die immer stärker werdende Konkurrenz der Java Technologie [14]. Diese zeichnet sich durch ihre betriebssystem- und programmiersprachenunabhängige Java Plattform aus.

Technisch gesehen sind sich die, für die.NET-Technologie entwickelte .NET Plattform und die Java Plattform erstaunlich ähnlich. So basieren beispielsweise beide auf einer automatischen Speicherverwaltung und einer umfangreichen Klassenbibliothek. Programme werden zunächst in eine Zwischensprache übersetzt und erst zur Laufzeit durch einen Just-In-Time-Compiler in Maschinencode konvertiert.

.NET Framework

Das *.NET Framework* ist Hauptbestandteil des .NET Konzeptes. Die Architektur des Frameworks soll anhand der folgenden Abbildung näher erläutert werden.

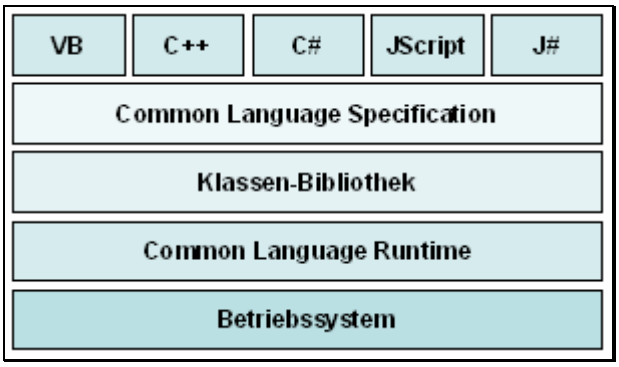


Abbildung 5-1: Architektur des .NET Frameworks

Die Laufzeitumgebung von .NET ist die *Common Language Runtime (CLR)*. Sie setzt auf die Dienste des Betriebssystems auf und ist für die Konvertierung der Zwischensprache in Maschinencode zuständig. Weitere Aufgaben sind unter anderem das Code-Management, die Speicherverwaltung (Garbage Collection) sowie die Erzwingung von Codesicherheit.

Die sehr umfangreiche Klassen-Bibliothek von .NET kann von allen .NET-Sprachen verwendet werden und stellt zum Beispiel Standardfunktionen für die Erstellung des Oberflächendesigns, die Arbeit mit Datenbanken und XML-Dateien zur Verfügung. Die über 4.500 Klassen der Bibliothek sind in so genannte Namensräume (Namespaces) gegliedert und erleichtern so das Auffinden bestimmter Klassen.

Damit die .NET-Plattform von mehreren Programmiersprachen verwendet werden kann, müssen die verschiedenen Sprachen bestimmte Mindestforderungen erfüllen, welche in der *Common Language Specification (CLS)* festgelegt sind.

C#

C# ist eine objektorientierte Programmiersprache die von Microsoft entwickelt wurde um die Funktionalität von .NET voll ausschöpfen zu können. Sie ist eine Kombination aus den Sprachen Java, C++, Visual Basic und Delphi. Besonderer Wert wurde bei der Entwicklung von C# auf eine Vereinfachung der Programmierung gelegt, jedoch ohne den Funktionalitätsumfang einzuschränken.

5.2.2 Fazit

Für die Implementierung des KursEditors soll die Programmiersprache C# und das .NET Framework verwendet werden. Das .NET Framework bietet sehr umfangreiche Bibliotheken für die Erstellung grafischer Bedienoberflächen und für die Arbeit mit XML. Die gestellten Anforderungen an die benötigte Technologie werden somit in hohem Maße erfüllt.
Als Entwicklungsumgebung ist das Microsoft Visual Studio .NET 2003 am Fraunhofer Institut verfügbar. Die darin enthaltenen Werkzeuge, wie der Form-Designer zur einfachen und übersichtlichen Gestaltung anspruchsvoller Bedienoberflächen sowie der integrierte Debugger, erleichtern die Entwicklung der Software in hohem Maße.

5.3 Softwarearchitektur

Mit dem KursEditor soll dem Nutzer eine interaktive Anwendung zur Verfügung gestellt werden, die eine schnelle und intuitive Erstellung von SCORM-konformen E-Learning-Kursen ermöglicht. Die Hauptkomponenten der Softwarearchitektur sind daher eine grafische Bedienoberfläche (GUI) und eine SCORM-Konverter, der entsprechend der Nutzereingaben den gewünschten Kurs erzeugt. Abbildung 5-2 zeigt die Softwarearchitektur des prototypisch umgesetzten KursEditors.

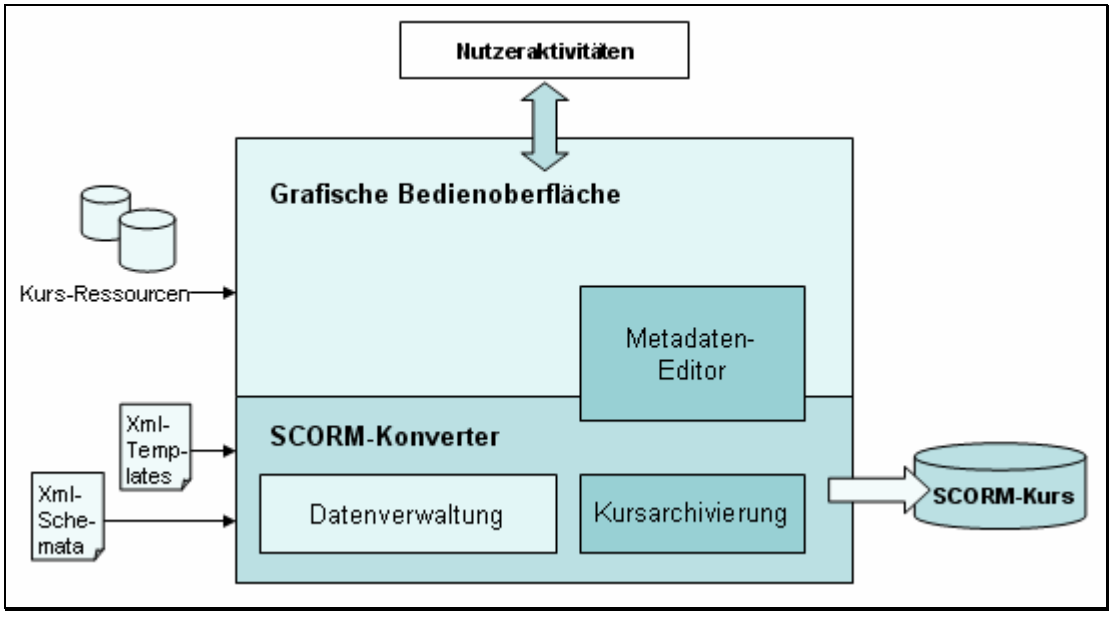


Abbildung 5-2: Softwarearchitektur des KursEditors

Grafische Bedienoberfläche

Über verschiedene Elemente der grafischen Bedienoberfläche kann der Anwender die Struktur und den Inhalt eines Kurses bestimmen. Die Eingaben an den im Treeview dargestellten Kurskomponenten können unter anderem per Drag & Drop über verschiedene Menüs, Kontextmenüs und Eingabefelder vorgenommen werden.

Metadaten-Editor

Der Metadaten-Editor ist ein Eingabeformular, also ein Teil der grafischen Bedienoberfläche. In diesem Formular werden vom Autor die Metadaten über Textfelder eingegeben.

SCORM-Konverter

Der SCORM-Konverter verarbeitet alle Nutzereingaben und erstellt aus den Informationen die entsprechende Manifest-Datei. Vom Autor definierte Metadaten werden in externen Metadaten-Dateien gespeichert. Die Dateien entsprechen der SCORM 1.3 Spezifikation und werden auf Basis von XML-Templates erstellt.

Datenverwaltung

Die Datenverwaltung der Anwendung erzeugt für jedes neue Projekt, dass der Nutzer anlegt, einen Projektordner mit einer Projektdatei und allen benötigten XML-Schematas. Hier werden alle Templates, die für die Erstellung der Manifest- und Metadaten-Dateien benötigt werden, verwaltet. Diese Dateien und die importierten Ressourcen werden ebenfalls im Projektordner abgelegt, so dass sich eine zentrale Stelle im Dateisystem mit allen Kursdateien befindet.

Kursarchivierung

Damit ein erstellter Kurs auch in ein LMS importiert werden kann, muss der Inhalt des Projektordners in einem SCORM-konformen Content Package im Zip-Format archiviert werden. Diese Aufgabe erfolgt durch die Kursarchivierung. Den Speicherort bestimmt der Nutzer über die grafische Bedienoberfläche.

5.4 Softwarenetzwicklung

Die wichtigsten Klassen der Anwendung sollen zunächst in einer Übersicht dargestellt werden (siehe Abbildung 5-3). Die Richtung der Beziehung zwischen den Klassen wird durch eine offene Pfeilspitze gekennzeichnet. Erbt eine Klasse von einer Anderen, wird dies durch einen nicht ausgefüllten Pfeil dargestellt, der von der abgeleiteten Klasse auf die Basisklasse zeigt.

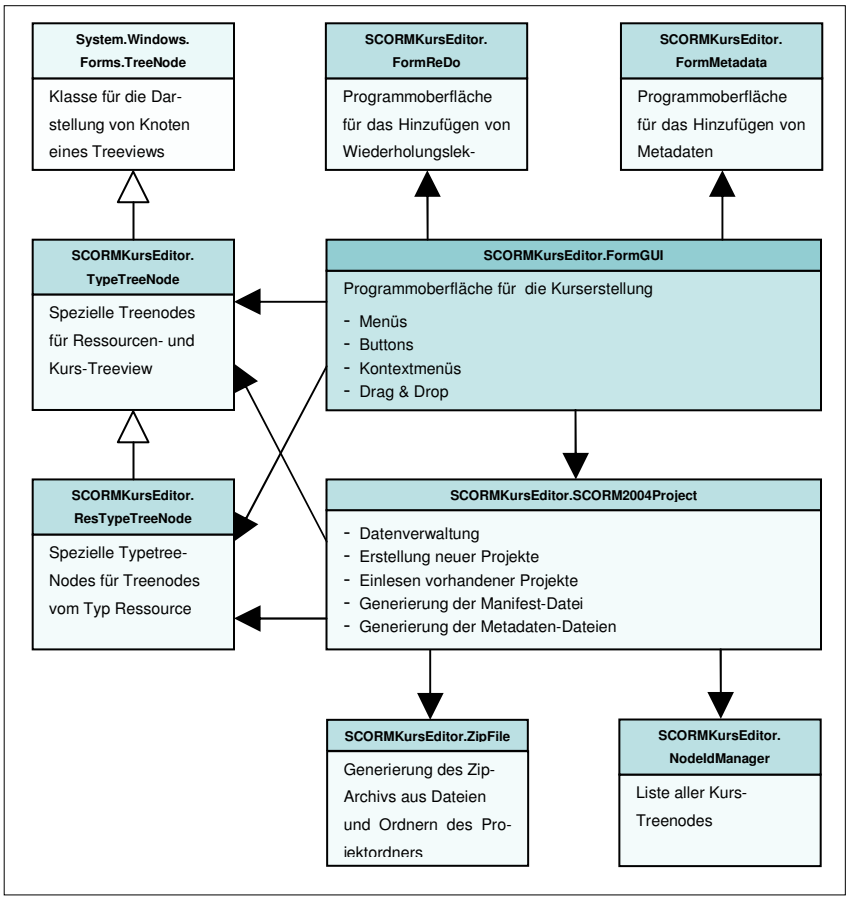


Abbildung 5-3: Struktur der wichtigsten Klassen des KursEditors

SCORMKursEditor.FormGUI

Die Klasse *SCORMKursEditor.FormGUI* repräsentiert das Hauptfenster der Anwendung und beinhaltet den Einstiegspunkt der Applikation. Sie ist der zentrale Container für alle Controls, die dem Bediener für die Erstellung eines Kurses zur Verfügung stehen. Die Datenfelder der Klasse sind somit fast ausschließlich Steuerelemente wie Buttons, Treeviews und Menüs. Die Hauptaufgabe der Klasse ist es, auf die Eingaben des Benutzers zu reagieren und die Daten an die verarbeitenden Klassen weiterzuleiten. Umgekehrt werden die Programmzustände und

Verarbeitungsergebnisse an die Oberfläche zurückgegeben, um sie dem Bediener sichtbar zu machen.

Die Klasse *SCORMKursEditor.FormGUI* ist von der durch das .NET Framework bereitgestellten Klasse *System.Windows.Forms.Form* abgeleitet und beinhaltet somit automatisch alle Methoden und Eigenschaften, die für die Darstellung eines Fensters unter dem Betriebssystem Microsoft Windows benötigt werden.

SCORMKursEditor.FormMetadata

Für die Konfiguration von Kurs-Metadaten stellt die Klasse *SCORMKursEditor.FormMetadata* ein Eingabefenster mit den dafür notwendigen Steuerelementen bereit. Diese Klasse ist ebenfalls von *System.Windows.Forms.Form* abgeleitet und erlaubt dem Nutzer den Zugriff auf die Metadaten des Kurses, wie Titel und Sprache, über Textfelder.

SCORMKursEditor.ReDo

Diese Klasse stellt ein Windows-Formular dar, in dem Wiederholungslektionen für einen individuellen Lernpfad festgelegt werden können. Dieses Fenster wird über das Kontextmenü von Testaufgaben (Elemente unterhalb eines Test-Elementes) geöffnet. Es wird ein Treeview dargestellt, dessen Struktur dem Kurs-Treeview entspricht. Die Elemente dieses Treeviews haben eine Checkbox, über die Wiederholungslektionen festgelegt werden können. Elemente des Treeviews, die nicht in den Lernpfad aufgenommen werden können, sind inaktiv.

SCORMKursEditor.TypeTreeNode

Als Hilfsmittel für die übersichtliche Anzeige und einfache Erstellung und Bearbeitung eines Kurses mit dem Kurseditor dient das grafische Steuerelement *Treeview*. Es eignet sich besonders gut für die Darstellung hierarchischer Strukturen und ist somit ein geeignetes Mittel, die theoretisch unbegrenzte Tiefe von Kapiteln und Unterkapiteln in den Kursen abzubilden. Um die Bearbeitung des Treeviews durch den Benutzer einfach zu gestalten sind Eingaben per Drag & Drop möglich.

Der Treeview wird innerhalb der .NET Framework Klassenbibliothek durch die Klasse *System.Windows.Forms.TreeView* bereitgestellt. Der Typ der Blatt-Elemente des Treeviews ist *System.Windows.Forms.TreeNode* und enthält alle benötigten Basisfunktionen, wie beispielsweise das Zuweisen von Anzeigetexten oder Icons. Für

die Prüfung der Gültigkeit der Zuordnungen des Benutzers innerhalb der Treeviews werden zudem weitere Eigenschaften benötigt. Zum Beispiel soll der Benutzer die Möglichkeit haben Inhalte innerhalb des Treeviews zu verschieben, aber nicht die Struktur so zu verändern, dass der Kurs falsch aufgebaut werden kann.
Um solche Prüfungen zu ermöglichen und den erstellten Treeview exakt in ein XML-Dokument entsprechend der SCORM 2004 Spezifikation überführen zu können, wird die Klasse *SCORMKursEditor.TypeTreeNode* für die Repräsentation der Blätter verwendet. Die Klasse ist von *System.Windows.Form.TreeNode* abgeleitet, ist aber um Eigenschaften wie dem Typ des Nodes, einer Liste für die Verknüpfung von Wiederholungslektion, einem Hilfetext usw., erweitert.

SCORMKursEditor.ResTypeTreeNode

Eine Ableitung der Klasse *SCORMKursEditor.TypeTreeNode* ist die Klasse *SCORMKursEditor.ResTypeTreeNode*. Sie erbt damit alle Methoden und Eigenschaften ihrer Elternklasse, enthält zusätzlich aber Informationen über referenzierte Lerninhaltsdaten wie HTML-Seiten oder Bilder. Damit können Instanzen der Klasse dem Nutzer den Speicherort der Ressource anzeigen und erlauben den Algorithmen zur Erstellung der Manifest-Datei den Verweis auf die zugrunde liegende HTML-Seite.

SCORMKursEditor.SCORM2004Project

Diese Klasse stellt den Kern der Algorithmen zur Kurserstellung bereit. Sie enthält keine Oberflächenelemente sondern verfügt über Methoden und Eigenschaften für den Datenaustausch mit den grafischen Steuerelmenten der Bedienoberfläche. Über entsprechende öffentliche Methoden sorgt die Klasse für das Erzeugen eines Grundgerüsts im Kurs-Treeview für einen neuen Kurs, das Laden und Speichern von Kursen sowie das Archivieren des Content Package bzw. des Projektordners.
Zur Umsetzung der Algorithmen benutzt die Klasse weitere Klassen, z.B. zum Laden der Template-Dateien durch die .NET XML-Klassen, zur Identifizierung von Treeviewnodes mittels der Klasse *ScormKursEditor.NodeIdManager* oder dem Archivieren der Inhalte mit den Klassen der *SharpZipLib*.

Beschreibung wichtiger Funktionen

1. Einlesen der XML-Templates
2. Erstellen der XML-Datei aus dem Kurs-Treeview und der Templates
3. Speichern der Projektdatei
4. Laden der Projektdatei
5. Archivierung des Content Package

Einlesen der XML-Templates

Durch den Modularen Aufbau der Manifest-XML-Datei bietet sich die Arbeit mit XML-Templates für die einzelnen Elemente des Zieldokuments an (siehe Kapitel 5.1). Entsprechend der jeweiligen Kursstruktur werden die einzelnen Elemente aneinandergereiht oder verschachtelt und bilden im Ergebnis das SCORM 2004 kompatible XML-Dokument.

Für das Einlesen der Templates sorgt die Methode `ReadTemplates()` der Klasse *SCORMKursEditor.SCORM2004Project*. Dazu verwendet die Klasse je ein Datenfeld vom Typ *System.Xml.XmlDocument* der .NET Klassenbibliothek (siehe Listing 5-2). Diese Klasse verfügt bereits über Methoden zum Einlesen von XML-Dokumenten und erlaubt die Manipulation der XML-Elemente.

```
#region Fields
...
private XmlDocument docBody     = null;
private XmlDocument docItem     = null;
private XmlDocument docRes      = null;
private XmlDocument docChap     = null;
private XmlDocument docTest     = null;
private XmlDocument docQuest    = null;
private XmlDocument docQuestSeq = null;
private XmlDocument docScore    = null;
private XmlDocument docReDo     = null;
private XmlDocument docObj      = null;
private XmlDocument docOrgSeq   = null;

#endregion Fields
```

Listing 5-2: Einlesen der XML-Templates

Jedes der Datenfelder wird beim Erzeugen eines neuen Projektes instanziiert und liest das jeweilige XML-Template aus dem Pfad der ausführbaren Datei des KursEditor ein. Für die Pflege oder Weiterentwicklung der Templates ist somit zu beachten, dass sich etwaige Änderungen erst mit dem nächsten Erzeugen eines neuen Kurses oder dem

nächsten Laden eines vorhandenen Kurses auswirken. Die Namen der XML-Template-Dateien sind durch Konstanten der Klasse festgelegt:

```
#region const

private const string TEMPLATE_BODY       = @"\Body.xml";
private const string TEMPLATE_ITEM       = @"\Item.xml";
private const string TEMPLATE_RES        = @"\Resource.xml";
private const string TEMPLATE_TEST       = @"\Test.xml";
private const string TEMPLATE_QUEST      = @"\Aufgabe.xml";
private const string TEMPLATE_QUEST_SEQ  = @"\SequencingAufgaben.xml";
private const string TEMPLATE_SCORE      = @"\Auswertung.xml";
private const string TEMPLATE_REDO       = @"\Wiederholung.xml";
private const string TEMPLATE_OBJ        = @"\Objective.xml";
private const string TEMPLATE_CHAP       = @"\Kapitel_mit_Sequencing.xml";
private const string TEMPLATE_SEQ_COL    = @"\SequencingCollection.xml";
private const string TEMPLATE_ORG_SEQ    = @"\SequencingOrganization.xml";

#endregion const
```

Listing 5-3: Konstanten für Namen der XML-Templates

Die Objektinstanzen werden mit jedem Aufruf der privaten Methode neu gebildet (siehe Listing 5-4). Durch die Garbage Collection kann auf das explizite Freigeben zuvor erzeugter Instanzen verzichtet werden.

```
private bool ReadTemplates()
{
      this.docBody = new XmlDocument();
      this.docItem = new XmlDocument();
      this.docRes  = new XmlDocument();
      this.docChap = new XmlDocument();
      this.docTest = new XmlDocument();
      ...

      string exePath=Path.GetDirectoryName(Application.ExecutablePath);

      try
      {
            this.docBody.Load(exePath + TEMPLATE_PATH + TEMPLATE_BODY);
            this.docItem.Load(exePath + TEMPLATE_PATH + TEMPLATE_ITEM);
            this.docRes.Load (exePath + TEMPLATE_PATH + TEMPLATE_RES);
            this.docChap.Load(exePath + TEMPLATE_PATH + TEMPLATE_CHAP);
            this.docTest.Load(exePath + TEMPLATE_PATH + TEMPLATE_TEST);
            ...
      }
      catch (XmlException e)
      {
            return false;
      }
      return true;
}
```

Listing 5-4: Erzeugung der Template-XML Dokumente

Erstellen der XML-Datei aus dem Kurs-Treeview und der Templates

Speichert der Anwender sein Kursprojekt wird auch das Erzeugen der Manifestdatei ausgeführt. Dazu wird der Methode `Save()` der Klasse *SCORMKursEditor.SCORM2004Project* die Liste der Nodes des Kurs-Treeview übergeben. Der Typ der Nodes sowie die Art der Verschachtlungstiefe (zum Beispiel durch Kapitel und Unterkapitel) bestimmen den Aufbau des Kurses und damit den Aufbau des zu erstellenden Manifest-XML-Dokuments. Zunächst wird das Template mit dem Grundgerüst in das Zieldokument importiert:

```
this.docManifest.RemoveAll();
this.idNumber = 0; // ID-Nummer für xml-Identifier zurücksetzten

// einfügen aller XML-Nodes des XML-Dokumentes docBody in das XML-
// Dokument docManifest
foreach (XmlNode xmlNode in this.docBody.ChildNodes)
{
      XmlNode newXmlNode = this.docManifest.ImportNode(xmlNode, true);
      this.docManifest.AppendChild(newXmlNode);
}
```

Listing 5-5: Import des Manifest-Grundgerüsts aus dem docBody-Template

Danach wird die Liste der Kurs-Treenodes durchlaufen und anhand des Typs der Nodes das zugehörige XML-Template ermittelt und in das Zieldokument eingefügt. Das Element unter das die weiteren Templates eingefügt werden heißt *Organization*.

```
XmlNodeList xmlNodeList =
      this.docManifest.GetElementsByTagName(XMLELEMENT_ORGANIZATION);

if (xmlNodeList.Count == 0) // es muss mind. einen Organization-Knoten geben
      return false;

XmlNode orgXmlNode = xmlNodeList[0];

foreach(TypeTreeNode treeNode in treeKursNodes)
{
      if (!BuildManifest(orgXmlNode, treeNode))
      {
            MessageBox.Show("Fehler beim Aufbau der Manifest-Datei!");
            return false;
      }
}
```

Listing 5-6: Einfügen der Templates für Kurselemente

Für jedes Node der obersten Hierarchiestufe des Treeviews wird ein *XML-Child-Element* dem *Organization-Element* durch je einen Aufruf der Methode `BuildManifest()` hinzugefügt. Innerhalb dieser Methode wird immer zunächst der Typ des Kurs-Treenodes ermittelt und danach entschieden welches Template dem

ebenfalls übergebenen XML-Knoten zugeordnet wird. Für die Abbildung der Hierarchie des Treeviews sorgt der rekursive Aufruf am Ende der Methode. Damit kann der Treeview beliebig tief sein und wird dennoch korrekt im Manifest-XML-Dokument abgebildet. Listing 5-5 zeigt einen Quellcode-Auszug der Methode `BuildManifest()`:

```
private bool BuildManifest(XmlNode xmlNode, TypeTreeNode treeNode)
{
      // für nächsten Rekursionsschritt zunächst aktuellen Knoten annehmen
      XmlNode newXmlNode = xmlNode;

      switch (treeNode.NodeType)
      {
            case TypeTreeNode.TreeNodeType.Course :
            {
                  xmlNode["title"].InnerText = treeNode.Text;
                  XmlNode orgSeqXmlNode = this.docManifest.ImportNode
                        (this.docOrgSeq.FirstChild["imsss:sequencing"], true);
                  xmlNode.AppendChild(orgSeqXmlNode);

                  // Anfügen des Sequencing-Knoten für gesamte Organization
                  CreateSequColXmlNode(xmlNode.ParentNode.ParentNode);
            }break;
      ...

      // rekursiver Aufruf für jedes Kind-Treenode
      foreach (TypeTreeNode childTreeNode in treeNode.Nodes)
      {
            if (!this.BuildManifest(newXmlNode, childTreeNode))
                  return false;
      };
      return true;
}
```

Listing 5-7: Rekursives Erzeugen der XML-Knoten aus dem Kurs-Treeview

Speichern der Projektdatei

Erzeugt der Anwender ein neues Projekt, wird im Projektordner durch den Aufruf der Methode `Create()` unter anderem auch eine Projektdatei angelegt. Diese ist ein XML-Dokument und enthält zunächst die folgenden Elemente:

```
<SCORM2004Project>
      <CourseTreeview />
</SCORM2004Project>
```

Listing 5-8: Grundgerüst der Projektdatei

Beim Speichern des Projektes wird die Methode `Save()` der Klasse *SCORMKursEditor.SCORM2004Project* mit einer Liste aller Ressourcennodes und Kurs-Nodes als Parameter aufgerufen (siehe Listing 5-9).

```
public bool Save(TreeNodeCollection resTreeNodes, TreeNodeCollection
      treeKursNodes)
{
      // Erzeugt die Xml-Einträge für den Kurs-Treeview
      if (SaveKursTreeview(treeKursNodes) == false)
            return false;

      // Speichert das Xml-Document in der Projektdatei
      this.xmlDocProject.Save(this.projectFileName);
      ...
      return true;
}
```

Listing 5-9: Speichern der Projektdatei

Zuerst wird die Methode `SaveKursTreeview()` ausgeführt, welche eine Liste des Kurs-Treeview in das XML-Dokument einliest (siehe Listing 5-10). Dafür wird rekursiv für jedes Treeview-Node die Methode `CreateXmlNodeFromTreeNode()` aufgerufen, welche alle zur späteren Wiederherstellung benötigten Eigenschaften als Attribute eines Xml-Nodes dem XML-Dokument hinzufügt.

```
private bool SaveKursTreeview(TreeNodeCollection treeKursNodes)
{
      ...

      // KursTreeviewNodes der ersten Ebene
      foreach(TypeTreeNode courseNode in treeKursNodes)
      {
            // Erzeugt XmlNode für übergebenen TreeNode
            XmlNode xmlNode = CreateXmlNodeFromTreeNode(courseNode);

            // fügt den XmlNode in KursTreeviewXmlNode ein
            courseTreeviewXmlNode.AppendChild(xmlNode);
      }
      return true;
}
```

Listing 5-10: Einlesen der Kurs-Treeview-Elemente in das XML-Dokument

Danach wird die Projektdatei selbst durch die Methode `Save()` der *XmlDocument-Klasse* gespeichert. Die Datei sieht auszugsweise wie folgt aus:

```
<SCORM2004Project>
  <CourseTreeview>
    <TypeTreeNode Name="Kurs" NodeID="118" NodeType="2" Helptext="Hier wird
          Ihre Kursstruktur dargestellt" Editable="True" ImageIndex="8"
          SelectedImageIndex ="8" Resource="false">
     <Childs>
       <TypeTreeNode Name="Einleitung" NodeID="120" NodeType="3"
          Helptext="Ordnen Sie per Drag&amp;&amp;Drop die entsprechende
          Ressourcen zu" Editable="False" ImageIndex="3"
          SelectedImageIndex="3" Resource="false">
         <Childs>
           <TypeTreeNode Name="Einleitung" NodeID="118" NodeType="1"
          Helptext="Linksklick, zum Umbenennen der Ressource (Name entspricht
          dem Menüeintrag)" Editable="True" ImageIndex="15"
```

```
            SelectedImageIndex="15" Resource="true" Filename="Einleitung.htm"
            Path="D:\Kurse\KursA\HTML\Einleitung.htm">
             <Links />
             <Childs />
           </TypeTreeNode>
         </Childs>
       </TypeTreeNode>
        ...
      </Childs>
    </TypeTreeNode>
  </CourseTreeview>
</SCORM2004Project>
```

Listing 5-11: Auszug einer vom KursEditor erstellen Projektdatei

Laden der Projektdatei

Will der Anwender ein bereits vorhandenes Projekt fortsetzen, kann er die entsprechende Projektdatei laden. Hierfür wird die Methode `Load()` der Klasse *SCORMKursEditor. SCORM2004Project* aufgerufen. Zunächst wird die Datei durch die `Load()`-Methode der *XmlDocument-Klasse* eingelesen. Anschließend wird der Kurs-Treeview für den geladenen Kurs von der Methode `LoadKursTreeView()` geladen. Einen Quellcodeauszug zeigt das Listing 5-12.

```
public bool Load(string projectFileName, TreeNodeCollection resTreeNodes,
      TreeNodeCollection treeKursNodes)
{
      ...
      this.xmlDocProject = new XmlDocument();

      try
      {      // Laden der Projekt-Datei
            this.xmlDocProject.Load(projectFileName);
      }
      catch (Exception e)
      {
            MessageBox.Show("Die Projekt-Datei kann nicht geöffnet werden!" +
                e.Message);
            return false;
      }
      ...
      this.LoadKursTreeView(treeKursNodes); // KursTreeview füllen
      return true;
}
```

Listing 5-12: Laden einer Projektdatei

`LoadKursTreeView()` ruft rekursiv für alle XML-Nodes der Projektdatei die Methode `CreateTypeTreeNodeFromXmlNode()` auf. Hier wird für jedes XML-Node das entsprechende TypeTreeNode für den Kurs-Treeview erstellt. Die jeweiligen Eigenschaften werden aus den XML-Attributen der Projektdatei gelesen.

5.5 Praktische Anwendung

Wie intuitiv und einfach SCORM-Kurse mit individuellen Lernpfaden bereits mit dem Prototyp des KursEditors umgesetzt werden können, soll das folgende Anwendungsbeispiel zeigen. Das Ziel, Kurse auch ohne jegliche SCORM-Kenntnisse erzeugen zu können, ermöglicht der Prototyp jedoch noch nicht ganz. Er stellt noch keinen Fragen-Editor zur Verfügung, so dass Testaufgaben und Auswertungsseiten mit den nötigen API-Befehlen zunächst noch manuell erzeugt werden müssen. Beispieldateien befinden sich im Anhang dieser Arbeit und können als Vorlagen verwendet werden.

Der Kurs, der in diesem Anwendungsbeispiel erstellt wird, entspricht dem Kurs der in Kapitel 3.3.1 mit dem Reload Editor erstellt wurde. Auf Begriffe aus der SCORM-Spezifikation wird verzichtet, um zu zeigen, dass Kenntnisse darüber für die Kurs-Erstellung (mit Ausnahme der Test- und Auswertungsseiten) nicht notwendig sind.

Allgemeine Produktionsschritte

1. Erstellen eines neuen Projektes
2. Ressourcen importieren
3. Ressourcen der Kursstruktur zuweisen
4. Lernpfade erstellen
5. Kurs mit Metadaten beschreiben
6. SCORM-Kurs für den Import in ein LMS generieren

1. Erstellen eines neuen Projektes

Nachdem der KursEditor gestartet wurde, kann über das Menü *Datei* und *Neu*, beziehungsweise über den entsprechenden Button (ein Tooltip wird angezeigt), der Speicherort des Projektes in einem Dialogfeld ausgewählt werden.
Ein Treeview für die Ressourcen und ein Treeview mit dem Grundgerüst des Kurses werden angezeigt (siehe Abbildung 5-4).

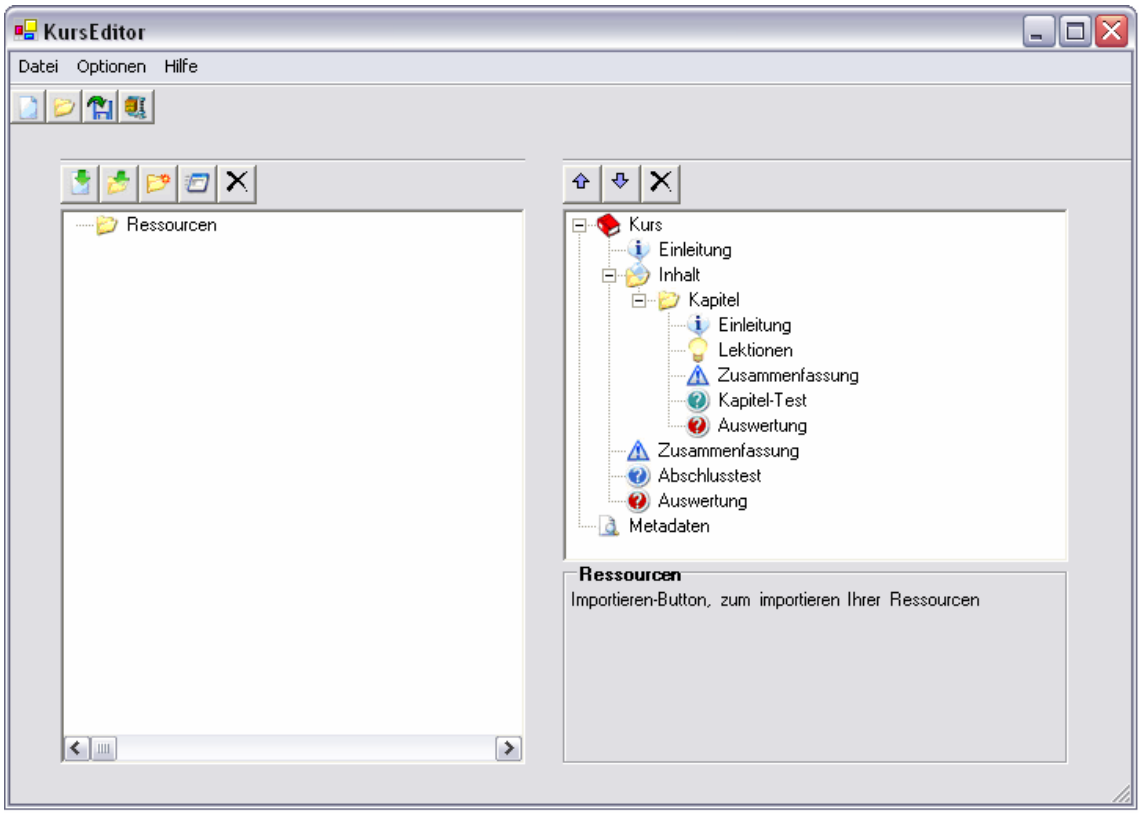


Abbildung 5-4: KursEditor-Shreenshot nach Erstellung eines neuen Projektes

2. Ressourcen importieren

Über Buttons des Ressourcen-Treeview können entweder ganze Ordner oder einzelne Dateien importiert werden. Alle Ressourcen werden entsprechend ihrer Struktur im Dateisystem im Treeview angezeigt (siehe Abbildung 5-5).

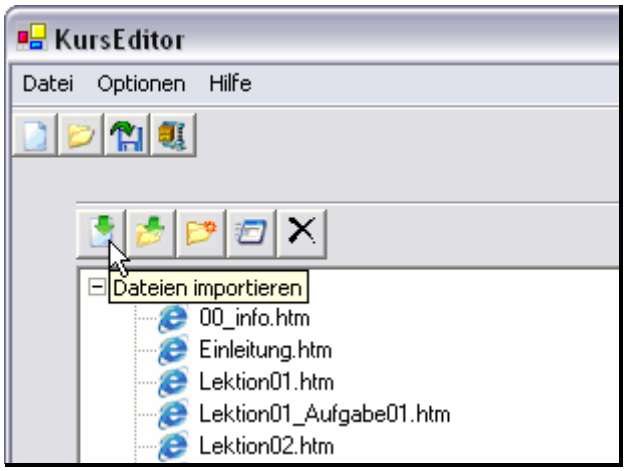


Abbildung 5-5: Ressourcen-Treeview

Über den Button *Ressource öffnen* können die Ressourcen in dem unter Windows standardmäßig festgelegten Programm zur Bearbeitung geöffnet werden.

3. Ressourcen der Kursstruktur zuweisen

Alle HTML-Dateien des Ressourcen-Treeview können nun den entsprechenden Elementen des Kursgrundgerüstes via Drag & Drop zugeordnet werden. Die HTML-Seite mit der Einleitung unter das Element *Einleitung*, Lektionen in gewünschter Reihenfolge unter das Element *Lektionen* usw.

Über das Kontextmenü von *Inhalt* und *Kapitel* können weitere Kapitel beziehungsweise Unterkapitel in das Grundgerüst eingefügt werden (siehe Abbildung 5-6).

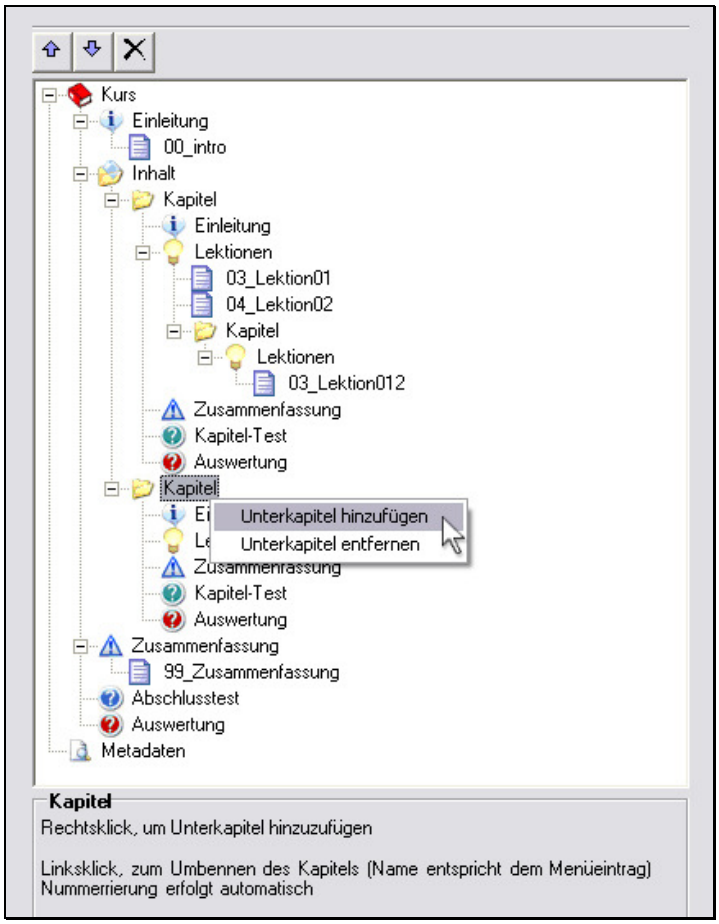


Abbildung 5-6: Beispielhafte Strukturierung des Kurs-Treeviews

4. Lernpfade erstellen

Über das Kontextmenü der einzelnen Test-Elemente kann zunächst festgelegt werden, ob dem Lernenden der Lernpfad mit oder ohne Kursmenü angezeigt werden soll.

Über einen Rechtsklick auf die dem Test zugeordneten Aufgaben wird ein neues Fenster für die Wahl der Wiederholungslektion/en angezeigt (siehe Abbildung 5-7).

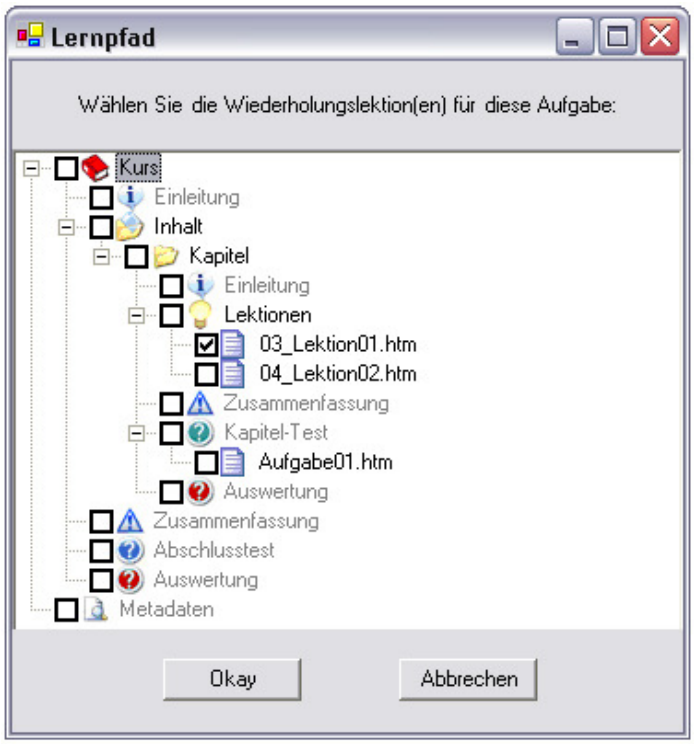


Abbildung 5-7: Auswahlfenster für Wiederholungslektionen

In einem Treeview wird die bisher erstellte Kursstruktur angezeigt. Es können jedoch nur Inhaltsseiten ausgewählt werden, da eine Wiederholungslektion auf eine HTML-Ressource verweisen muss.

5. Kurs mit Metadaten beschreiben

In der prototypisch implementierten Anwendung des KursEditors können lediglich vier Metadaten definiert werden, welche auch nur den gesamten Kurs beschreiben (Content Aggregation Metadaten). Dies ist ausreichend um das Prinzip zu verdeutlichen.

Über das Kontextmenü des Metadaten-Elementes wird ein Formular in einem neuen Fenster geöffnet (siehe Abbildung 5-8). In die Eingabefelder können die entsprechenden Schlüsselwörter eingegeben werden, wobei Felder auch leer bleiben dürfen.

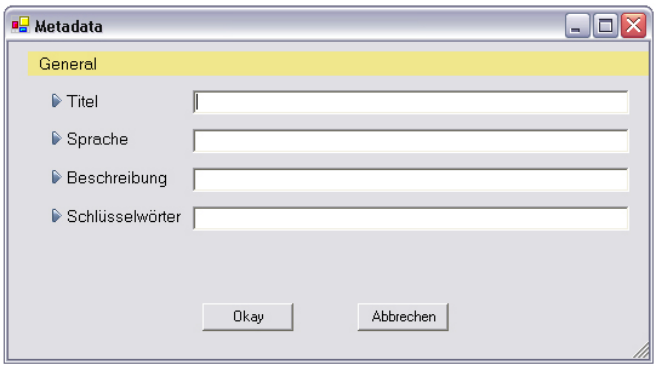


Abbildung 5-8: Eingabeformular für Metadaten

6. SCORM-Kurs für den Import in ein LMS generieren

Für den Import des gewünschten Kurses in ein LMS, muss über den *Zip*-Button das nötige Zip-Archiv generiert. Der Speicherort ist über einen Speichern-Dialog wählbar.

5.6 Ansätze zur Weiterentwicklung des KursEditors

Der in dieser Arbeit entwickelte Prototyp des KursEditors erfüllt bereits einen Großteil der Funktionalität, die für die endgültige Anwendung vorgesehen ist. Im Folgenden sollen Ansätze für die noch zu implementierenden Funktionen erläutert werden.

Metadaten-Editor

Der beispielhaft umgesetzte Metadaten-Editor sollte dahin gehend erweitert werden, dass alle SCORM-Metadaten für alle Bereiche und Items eines Kurses erstellt werden können (Tabelle 3-1). Über ein Kontextmenü könnten den entsprechenden Elementen des Kurs-Treeviews jeweils ein Metadaten-Element hinzugefügt werden. Weiterhin müssten das Template sowie das Eingabeformular entsprechend vervollständigt werden. Die Metadaten werden für jedes Item in einer eigenen XML-Datei gespeichert, auf der in der Manifest-Datei entsprechend verwiesen wird.
Eine weitere Funktion dieses Editors könnte der Im- und Export von Metadaten-Dateien sein.

Fragen-Editor

Wie bereits die Generierung der Manifest- und der Metadaten-Dateien durch die Verwendung von Templates erfolgt, könnte dies auch für den Fragen-Editor umgesetzt werden.

Für die verschiedenen Fragetypen (z.B. Multiple Choice) können vollständige HTML-Seiten mit den benötigten JavaScript- und API-Methoden als Templates dienen. Über Eingabeformulare würden somit vom Anwender die Fragen, Antwortmöglichkeiten, Lösungen etc. definiert werden können. Zu beachten ist hierbei, dass der Autor das Layout dieser Seiten nicht bestimmen kann. Es sollte daher ein möglichst neutrales Design gewählt werden.

Für die Auswertungsseiten könnten ebenfalls Templates eingesetzt werden. Benötigte Variablen, wie die Gesamtanzahl der Aufgaben, können vom Programm ermittelt und gesetzt werden, so das keinerlei Eingaben vom Anwender verlangt werden müssen. Für diese Seite wäre ebenfalls das Layout durch das Template vorbestimmt.

Weitere Funktionen

Während der Entwicklung und durch Tests mit den Endanwendern sind weitere Ideen und Wünsche für das Endprodukt des KursEditors entstanden. Einige sollen im Folgenden stichpunktartig aufgezählt werden.

- Arbeitsschritte Rückgängig machen / Wiederherstellen
- Drag & Drop von Ressourcen innerhalb des Ressourcen-Treeviews
- beim Löschen von Ressourcen Verweise auf Kurs-Treeview-Elemente prüfen und gegebenenfalls Elemente löschen
- Aufgaben denen bereits Wiederholungslektionen zugeordnet wurden, könnten durch ein entsprechendes Icon gekennzeichnet werden
- „Speichern unter“-Dialog
- Menübefehle über Tastatureingaben ermöglichen

Bei der Erweiterung des KursEditors sollte jedoch immer berücksichtigt werden, dass es sich um eine Anwendung handeln soll, die besonders leicht und intuitiv zu bedienen ist. Zu viele Funktionen, Einstellungsmöglichkeiten etc., können ein Programm schnell unübersichtlich gestalten.

Zusammenfassung und Ausblick

Für die Erstellung hochwertiger Lernsoftware gewinnt die Einhaltung von Standards immer mehr an Bedeutung. Sie ermöglichen es Kurse effektiver zu erstellen, weiterzuentwickeln und wieder zu verwenden.
Die Fraunhofer Gesellschaft hält sich dabei zunehmend an die etablierte SCORM-Spezifikation. Es hat sich aber gezeigt, dass diese einige Hindernisse mit sich bringt. Sie ist sehr umfangreich und eine Einarbeitung sehr aufwendig. So genannte SCORM-Tools ermöglichen es jedoch Kurse auch mit weniger hohem Aufwand zu erstellen. Ein bisher ungelöstes Problem war aber eine starke Einschränkung auf mediendidaktischer Ebene. Wichtige Funktionalitäten wie das Auswerten von Testergebnissen und das Generieren individueller Lernpfade, war nur eingeschränkt möglich und musste nicht vom LMS unterstützt werden. Mit der im Jahr 2004 erschienenen SCORM-Version 1.3 sollte sich dies aber ändern. Die Untersuchung der aktuellen Spezifikation war Bestandteil der Arbeit und hat ergeben, dass der Funktionsumfang von SCORM 1.3 im Vergleich zur Vorgängerversion sehr viel größer geworden ist. Daraus ergibt sich jedoch ein noch höherer Aufwand für die Einarbeitung.

Für das Fraunhofer IIS/EAS ist dieser Aufwand nicht vertretbar, zu dem kommt dass es sich bei den Autoren in erster Linie um wissenschaftliche Mitarbeiter und nicht um E-Learning-Spezialisten handelt. Die Verwendung von SCORM-Tools könnte Erleichterung bringen. Eine Marktuntersuchung hat jedoch ergeben, dass derzeit keine Werkzeuge für die Kurserstellung verfügbar sind, die ohne Einarbeitung in die sehr umfangreiche SCORM 1.3 Spezifikation angewendet werden können.
In dieser Arbeit wurde daher ein Konzept für eine Anwendung entwickelt, welches den Anforderungen des Institutes gerecht wird. E-Learning-Kurse sollen intuitiv und ohne SCORM-Kenntnisse entwickelt werden können und dennoch hohen didaktischen Ansprüchen genügen.
Ein Großteil des Konzeptes wurde im Rahmen dieser Arbeit bereits implementiert. Per Drag & Drop können Lerninhalte in eine vordefinierte Kursstruktur eingefügt werden. Über Auswahlmenüs lassen sich Wiederholungslektionen für die Generierung individueller Lernpfade bestimmen. Der nötige SCORM 1.3 konforme XML-Code sowie das für den Import in ein LMS benötigte Kursarchiv wird vom Programm

erzeugt. Metadaten können mit dem Prototypen bereits beispielhaft definiert werden, ein Frageneditor ist zu ergänzen.

Die Generierung der SCORM 1.3 konformen Manifest-Datei basiert auf der Verwendung von Templates. Für die künftige Weiterentwicklung des konzipierten SCORM-Tools, wie beispielsweise die Entwicklung neuer Kurs-Szenarien, können diese problemlos erweitert oder ergänzt werden. Wird für die Auswertung dieser XML-Templates eine neue Programmlogik notwendig, sind entsprechende Funktionen anzupassen.
Aufgrund des strukturierten Aufbaus der Anwendung und der Aufteilung von Funktionseinheiten in verschiedene Klassen, ist eine gute Ausgangsbasis für künftige Programmerweiterungen geschaffen. Durch die Trennung zwischen Daten anzeigenden und Daten verarbeitenden Funktionen ist eine Einbindung des SCORM-Konverters in andere Applikationen möglich.

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

ADL	Advanced Distributed Learning
AICC	Aviation Industry Computer-Based Training Committee
API	Application Programming Interface
CAM	Content Aggregation Model
ECMA	European Computer Manufacturer Association
HTML	Hypertext Markup Language
IEEE	Institute of Electrical and Electronics Engineers
IMS	Instructional Management Systems
JISC	The Joint Information Systems Committee
LEARNTEC	europäischer Kongress und Fachmesse für Bildungs- und Informationstechnologie
LMS	Lernmanagementsystem
MILEON	Microsystems Technology and Microelectronics Lectures Online
PDA	Personal Digital Assistant
PIF	Package Interchange File
RTE	Run-Time Environment
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
SN	Sequencing and Navigation
VITERO	Virtual Team Room
XML	Extensible Markup Language

Literaturverzeichnis

[1] ADL
http://www.adlnet.org/ (verfügbar am 18.05.2005)

[2] SCORM 1.2 Overview Version 1.2.
http://www.adlnet.org/downloads/index.cfm (verfügbar 18.05.2005)

[3] SCORM 1.2 Content Aggregation Model Version 1.2
http://www.adlnet.org/downloads/index.cfm (verfügbar 18.05.2005)

[4] SCORM 1.2 Run-Time Environment Version 1.2
http://www.adlnet.org/downloads/index.cfm (verfügbar 18.05.2005)

[5] SCORM 2004 Overview 2nd Edition
http://www.adlnet.org/downloads/index.cfm (verfügbar 18.05.2005)

[6] SCORM 2004 Content Aggregation Model Version 1.3.1
http://www.adlnet.org/downloads/index.cfm (verfügbar 18.05.2005)

[7] SCORM 2004 Run-Time Environment Version 1.3.1
http://www.adlnet.org/downloads/index.cfm (verfügbar 18.05.2005)

[8] SCORM 2004 Sequencing and Navigation Version 1.3.1
http://www.adlnet.org/downloads/index.cfm (verfügbar 18.05.2005)

[9] Gericke, A.: Untersuchung und Bewertung von SCORM 2004 sowie prototypische Realisierung von Online-Kursmaterial nach dieser Spezifikation. Diplomarbeit Hochschule für Technik und Wirtschaft Dresden, 2004

[10] 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
http://www.cbtware.de/cms_content_view_18_2.html (verfügbar 20.05.2005)

[11] Garten, V.: Mediendidaktische und medientechnische Konzeption eines SCORM-basierten E-Learning-Kurses am Beispiel eines Projektes im Fraunhofer-Institut IIS/EAS. Diplomarbeit Hochschule Mittweida, 2004

[12] RELOAD
http://www.reload.ac.uk/ (verfügbar 15.07.2005)

[13] Gesellschaft für Informatik, Regionalgruppe Bremen Oldenburg
http://www.gi-hb-ol.de/uta/gi-rg/dotnet.ppt (verfügbar 27.08.2005)

[14] Wikipedia – Die freie Enzyklopädie
http://de.wikipedia.org/ (verfügbar 05.09.2005)

Anhang A

Diese Tabelle wurde in Anlehnung an die Tabelle aus Anhang B der Quelle [9] erstellt. Sie beschreibt die Datenelemente der SCORM RTE. Außerdem werden die Veränderungen von der Version 1.2 zu 1.3 gegenüberübergestellt und angegeben, ob das SCO Lese- und/oder Schreibzugriff auf das Element besitzt und ob das Datenelement für das LMS optional oder obligatorisch ist.

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**	**Beschreibung der Datenelemente; Bemerkungen**
`cmi.core._children` `(M,R)`	–	→ liefert die vom LMS unterstützten Kindelemente von `cmi.core`; Kategorie *core* wurde aus dem Datenmodel entfernt
`cmi.core.student_id` `(M,R)`	`cmi.learner_id` `(M,R)`	→ liefert ID des Lernenden; Änderung des Datentyps von CMIIdentifier auf long_identifier_type
`cmi.core.student_name` `(M,R)`	`cmi.learner_name` `(M,R)`	→ liefert Name des Lernenden; Änderung des Datentyps von CMIString255 auf localized_string_type
`cmi.core.lesson_location` `(M,R,W)`	`cmi.location` `(M,R,W)`	→ liefert/setzt Stelle eines SCO an der sich der Lernende zuletzt befunden hat; Änderung des Datentyps von CMIString255 auf characterstring
`cmi.core.credit` `(M,R)`	`cmi.credit` `(M,R)`	→ gibt zurück, ob der Abschluss des SCO den Status des Lernenden beeinflussen soll; Änderung des Datentyps von CMIVocabulary auf state
`cmi.core.lesson_status` `(M,R,W)`	–	→ gibt zurück/ legt fest wie der Lernende ein SCO abgeschlossen hat (z. B. erfolgreich); Element existiert noch im Datenmodell, soll aber in SCORM 1.3 SCOs nicht mehr benutzt werden; LMS sollten dieses Element aber noch unterstützen

	cmi.success_status (M,R,W)	→ gibt zurück/legt fest, ob das SCO vom Lernenden erfolgreich abgeschlossen wurde; neues Element in SCORM 1.3; ersetzt zusammen mit cmi.completion_ status das Element cmi.core.lesson _status
	cmi.completion_status (M,R,W)	→ gibt zurück/setzt, ob SCO vollständig bearbeitet wurde; neues Element in SCORM 1.3; ersetzt zusammen mit cmi.success_status das Element cmi.core.lesson_status
cmi.core.entry (M,R)	cmi.entry (M,R)	→ gibt zurück, ob der Lernende das SCO bereits einmal betreten hat; Änderung des Datentyps von CMIVocabulary auf state
cmi.core.score._child ren (M,R)	cmi.score._children (M,R)	→ liefert eine Liste aller Kinder von *score*; 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)	→ liefert/setzt die Punktzahl des Lernenden; Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real (10,7)
cmi.core.score.max (O,R,W)	cmi.score.max (M,R,W)	→ liefert/setzt die Punktzahl, die maximal erreicht werden kann; Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real(10,7)
cmi.core.score.min (O,R,W)	cmi.score.min (M,R,W)	→ liefert/setzt die Punktzahl, die mindestens erreicht werden kann; Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real (10,7)
	cmi.score.scaled (M,R,W)	→ liefert/setzt die erreichte normalisierte Punkzahl des Lernenden („-1.0“…“1.0“); neues Element in SCORM 1.3
cmi.core.total_time (M,R)	cmi.total_time (M,R)	→ liefert Zeit die Lernende im SCO insgesamt verbracht hat; Änderung des Datentyps von CMITimespan auf time(second,10,2)
cmi.core.lesson_mode (O,R)	cmi.mode (M,R)	→ liefert den Modus in dem das SCO abgespielt werden soll: „browse“ - Lernerdaten werden nicht aufgezeichnet; „normal“ - Lernerdaten werden aufgezeichnet; „review“ - SCO bereits bearbeitet; Änderung des Datentyps von CMIVocabulary auf state

cmi.core.exit (M,W)	cmi.exit (M,W)	→ setzt den Grund/Art und Weise wie das SCO beendet wurde an 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)	→ setzt die Bearbeitungszeit des SCO vom aktuellen Zugriff; Änderung des Datentyps von CMITimespan auf timeinterval (second,10,2)
cmi.suspend_data (M,R,W)	cmi.suspend_data (M,R,W)	→ ermöglicht dem SCO Speicherung/ Abfrage beliebiger Informationen; Änderung des Datentyps von CMIString4096 auf characterstring
cmi.launch_data (M,R)	cmi.launch_data (M,R)	→ liefert dem SCO beliebige Initialisierungsdaten; Änderung des Datentyps von CMIString4096 auf characterstring
cmi.comments (O,R,W)	cmi.comments_from _learner (M,R,W)	→ ermöglicht Speicherung/Abfrage von Lernerkommentaren; 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)	→ SCO kann mit diesem Element Kommentare des LMS abrufen; 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)	→ liefert die vom LMS unterstützten Kindelemente von cmi.objectives; 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)	→ liefert die Anzahl der Einträge in cmi.objectives; Änderung des Datentyps von CMIInteger auf integer
cmi.objectives.n.id (O,R,W)	cmi.objectives.n.id (M,R,W)	→ liefert/setzt ID des Objectives; 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)	→ liefert die vom LMS unterstützten Kindelemente von cmi.objectives.score; 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)	→ liefert/setzt die Punktzahl die für dieses Objective erreicht wurde; 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)	→ liefert/setzt die Punktzahl, die maximal für das Objective erreicht werden kann; 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)	→ liefert die Punktzahl, die mindestens für das Objective erreicht werden kann; Änderung des Datentyps von CMIDecimal bzw. CMIBlank auf real(10,7)
	cmi.objectives.n. score.scaled (M,R,W)	→ liefert/setzt die erreichte normalisierte Punkzahl des Lernenden für dieses Objective („-1.0“…“1.0“); neues Element in SCORM 1.3
cmi.objectives.n. status (O,R,W)	-	→ gibt zurück/legt fest, wie der Lernende das Objective abgeschlossen hat; Element existiert noch im IEEE 1484.11.1 Datenmodell, soll aber in SCORM 1.3 SCOs nicht mehr benutzt werden; LMS sollten dieses Element aber noch unterstützen
	cmi.objectives.n. success_status (M,R,W)	→ gibt zurück/legt fest, wie der Lernende das Objective bestanden hat; neues Element in SCORM 1.3; ersetzt zusammen mit cmi.objectives.n. completion_status das Element cmi.objectives.n.status
	cmi.objectives.n. completion_status (M,R,W)	→ gibt zurück/setzt, ob der lernende das Objective erfolgreich bestanden hat; neues Element in SCORM 1.3; ersetzt zusammen mit cmi.objectives.n. success_status das Element cmi.objectives.n.status
	cmi.objectives.n. description (M,R,W)	→ mit diesem Element kann eine Beschreibung für das Objective gespeichert/abgerufen werden; neues Element in SCORM 1.3

cmi.student_data. _children (O,R)	-	→ liefert die vom LMS unterstützten Kindelemente von cmi.student_data. _children; Kategorie *student_data* wurde aus dem Datenmodel entfernt
cmi.student_data. mastery_score (O,R)	cmi.scaled_passing _score (M,R)	→ gibt den Erfolgsstatus an, der vom Lernenden mindestens für das Bestehen des SCOs erreicht werden muss (Definition in der Manifest-Datei); Änderung des Datentyps von CMIDecimal auf real(10,7); Initialisierung des Daten-elements 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)	→ gibt ein Zeitlimit für ein SCO zurück Änderung des Datentyps von CMITimespan auf timeinterval (second,10,2); Initialisierung des Datenelements mit dem Wert von <imsss:attemptAbsolute DurationLimit> (falls angegeben)
cmi.student_data. time_limit_action (O,R)	cmi.time_limit_action (M,R)	→ gibt an, wie sich das SCO verhalten soll, wenn ein Zeitlimit erreicht wird; Änderung des Datentyps von CMIVocabulary auf state
cmi.student _preference._children (O,R,W)	cmi.learner _preference._children (M,R,W)	→ liefert/setzt die vom LMS unterstützten Kindelemente; Änderung des Datentyps von CMIString255 auf characterstring; Aktualisierung der Liste der Kindelemente mit den geänderten Elementnamen
cmi.student _preference.audio (O,R,W)	cmi.learner _preference.audio _level (M,R,W)	→ liefert/setzt die Lautstärkeeinstellung Änderung des Datentyps von CMISInteger auf real(10,7)
cmi.student _preference.language (O,R,W)	cmi.learner _preference.language (M,R,W)	→ liefert/setzt die vom Lernenden gewählte Sprache Änderung des Datentyps von CMIString255 auf language_type
cmi.student _preference.speed (O,R,W)	cmi.learner _preference.delivery _speed (M,R,W)	→ liefert/setzt die Abspielgeschwindigkeit Änderung des Datentyps von CMISInteger auf real(10,7)
cmi.student _preference.text (O,R,W)	cmi.learner _preference.Audio _captioning (M,R,W)	→ gibt zurück/legt fest ob Begleittexte für Audiodatein angezeigt werden sollen 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)	→ liefert die vom LMS unterstützten Kindelemente von cmi.interactions._children; Änderung des Datentyps von CMIString255 auf characterstring; Aktualisierung der Liste der Kindelemente mit den geänderten Elementnamen
cmi.interactions._count (O,R)	cmi.interactions._count (M,R)	→ liefert die Anzahl der Einträge in cmi.interactions; Änderung des Datentyps von CMIInteger auf integer
cmi.interactions.n.id (O,W)	cmi.interactions.n.id (M,R,W)	→ liefert/setzt ID der Interaktion; Änderung des Datentyps von CMIIdentifier auf long_identifier_type
cmi.interactions.n.objectives._count (O,R)	cmi.interactions.n.objectives._count (M,R)	→ liefert die Anzahl der Einträge in cmi.interactions.n.Objectives. Änderung des Datentyps von CMIInteger auf integer
cmi.interactions.n.objectives.n.id (O,W)	cmi.interactions.n.objectves.n.id (M,R,W)	→ liefert/setzt ID des Objectives das der Interaktion zugeordnet ist; Änderung des Datentyps von CMIIdentifier auf long_identifier_type
cmi.interactions.n.time (O,W)	cmi.interactions.n.timestamp (M,R,W)	→ liefert/setzt den Zeitpunkt dieser Interaktion; Änderung des Datentyps von CMITime auf time(second,10,2)
cmi.interactions.n.type (O,W)	cmi.interactions.n.type (M,R,W)	→ liefert/setzt den Typ der Interaktion 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)	→ liefert die Anzahl der Einträge in cmi.interactions.n.correct_responses (Anzahl der möglichen richtigen Antworten für eine Interaktion); 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)	→ liefert/setzt eine mögliche Antwort für eine Interaktion; 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)	→ liefert/setzt das Gewicht der Interaktion; Änderung des Datentyps von CMIDecimal auf real(10,7)

cmi.interactions.n. student_response (O,W)	cmi.interactions.n. learner_response (M,R,W)	→ liefert/setzt die Antwort eines Lernenden; kein allgemeiner Datentyp mehr festgelegt, jeder Typ (entsprechend cmi.interaction.n.type) hat separate Festlegungen bezüglich Datentyp und Format
cmi.interactions.n. result (O,W)	cmi.interactions.n. result (M,R,W)	→ liefert/setzt die Bewertung der Antwort (z.B. richtig oder falsch); 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)	→ liefert/setzt die Zeit die der Lernende für die Beantwortung benötigte; Änderung des Datentyps von CMITimespan auf timeinterval(second,10,2)
	cmi.interactions.n. description (M,R,W)	→ mit diesem Element kann eine Interaktion gespeichert/abgerufen werden; neues Element in SCORM 1.3,
cmi._version (O,R)	cmi._version (M,R)	→ gibt Versionsnummer des RTE-Datenmodells zurück; repräsentiert die Version des Datenmodells, sollte den Wert "1.0" enthalten

Anhang B

APIWrapper.js

```
// Definieren der Fehlercodes:
var _NoError = 0;
var _GeneralException = 101;
var _GeneralInitializationFailure = 102;
var _AlreadyInitialized = 103;
var _ContentInstanceTerminated = 104;
var _GeneralTerminationFailure = 111;
var _TerminationBeforeInitialization = 112;
var _TerminationAfterTermination = 113;
var _ReceivedDataBeforeInitialization = 122;
var _ReceivedDataAfterTermination = 123;
var _StoreDataBeforeInitialization = 132;
var _StoreDataAfterTermination = 133;
var _CommitBeforeInitialization = 142;
var _CommitAfterTermination = 143;
var _GeneralArgumentError = 201;
var _GeneralGetFailure = 301;
var _GeneralSetFailure = 351;
var _GeneralCommitFailure = 391;
var _UndefinedDataModelElement = 401;
var _UnimplementedDataModelElement = 402;
var _DataModelElementValueNotInitialized = 403;
var _DataModelElementIsReadOnly = 404;
var _DataModelElementIsWriteOnly = 405;
var _DataModelElementTypeMismatch = 406;
var _DataModelElementValueOutOfRange = 407;

// Definieren lokaler Variablen:
var apiHandle = null;
var API = null;
var findAPITries = 0;
var _Debug = false;

function doInitialize()      // Initialisierung der Kommunikation mit dem LMS
{
   var api = getAPIHandle(); // Aufruf der Methode getAPIHandle()
   if (api == null)          // API konnte nicht lokalisiert werden
   {
      alert("Unable to locate the LMS's API Implementation.\nInitialize was not successful.");
      return "false";
   }
   // Aufruf der API-Methode Initialize("") für Initialiesierung der Kommunikation
   var result = api.Initialize("");
   if (result.toString() != "true") // Fehlerbehandlung
   {
      var err = ErrorHandler();
   }
   return result.toString();//Rückgabewert: true - Initialisierung war erfolgreich, false - wenn
nicht
}

function doTerminate() // Beenden der Kommunikation mit dem LMS
{
   var api = getAPIHandle();
   if (api == null)
   {
      alert("Unable to locate the LMS's API Implementation.\nTerminate was not successful.");
      return "false";
   }
   else
   {
      // Aufruf der API-Methode Terminate("") für Beenden der Kommunikation
      var result = api.Terminate("");
      if (result.toString() != "true") // Fehlerbehandlung
      {
         var err = ErrorHandler();
      }
   }
   return result.toString(); // Rückgabewert (true - Termination war erfolgreich, false - wenn nicht)
}

function doGetValue(name)    // Datenanfrage an das LMS
{
   var api = getAPIHandle();
   if (api == null)

   {
      alert("Unable to locate the LMS's API Implementation.\nGetValue was not successful.");
```

```
      return "";
   }
   else // API-Handle enthällt einen gültigen Wert (API-Instanz gefunden)
   {
      // Aufruf der API-Methode GetValue(Data Model Element) für die Anfrage an das LMS
      // 'name' enthällt das Data Model Element, dass vom LMS abgefragt werden soll
      var value = api.GetValue(name);
      var errCode = api.GetLastError().toString(); // Fehlerbehandlung
      if (errCode != _NoError)
      {
         // an error was encountered so display the error description
         var errDescription = api.GetErrorString(errCode);
         alert("GetValue("+name+") failed. \n"+ errDescription);
         return "";
      }
      else
      {
         return value.toString(); // Rückgabe des Wertes des abgefragten Data Model Elements
      }
   }
}

function doSetValue(name, value)  // Transfer von Daten zum LMS
{
   var api = getAPIHandle();
   if (api == null)
   {
      alert("Unable to locate the LMS's API Implementation.\nSetValue was not successful.");
      return;
   }
   else // API-Handle enthällt einen gültigen Wert (API-Instanz gefunden)
   {
      // Aufruf der API-Methode SetValue(Data Model Element, Wert) für den Transfer von Daten zum LMS
      // 'name' enthällt das zu übergebende Data Model Element und
      // 'value' den Wert dieses Elements der übergeben werden soll
      var result = api.SetValue(name, value);

      if (result.toString() != "true") //Fehlerbehandlung
      {
         var err = ErrorHandler();
      }
   }
   return;
}

function doCommit() // Übertragung aller zwischengespeicherten Daten zum LMS
{
   var api = getAPIHandle();
   if (api == null)
   {
      alert("Unable to locate the LMS's API Implementation.\nCommit was not successful.");
      return "false";
   }
   else
   {
      // Aufruf der API-Methode Commit("") für Übertragung aller zwischengespeicherten Daten
      var result = api.Commit("");
      if (result != "true") // Fehlerbehandlung
      {
         var err = ErrorHandler();
      }
   }
   return result.toString();//Rückgabewert: true - Datenübertragung war erfolgreich,false - wenn
nicht
}

function getAPIHandle()      // liefert ein Handle auf die API-Instanz
{
   if (apiHandle == null)    // wenn Handle der API-Instanz noch nicht gesetzt...
   {
      apiHandle = getAPI();  // ... wir getAPI() aufgerufen
   }
   return apiHandle;         // Rückgabe des API-Handle
}

function getAPI() // Lokalisiert die API-Instanz
{
   var theAPI = findAPI(window);
   if ((theAPI == null) && (window.opener != null) && (typeof(window.opener) != "undefined"))
   {
      theAPI = findAPI(window.opener);
   }
   if (theAPI == null)
   {
      alert("Unable to find an API adapter");
   }
   return theAPI
}
```

```
function findAPI(win)
{
   while ((win.API_1484_11 == null) && (win.parent != null) && (win.parent != win))
   {
      findAPITries++;
      if (findAPITries > 500)
      {
         alert("Error finding API -- too deeply nested.");
         return null;
      }
      win = win.parent;
   }
   return win.API_1484_11;
}

// Methoden für die Fehlerbehandlung

function doGetLastError() // Fehlerbehandlung
{
   var api = getAPIHandle();
   if (api == null)
   {
      alert("Unable to locate the LMS's API Implementation.\nGetLastError was not successful.");
      return _GeneralError;
   }

   return api.GetLastError().toString();
}

function doGetErrorString(errorCode) // Fehlerbehandlung
{
   var api = getAPIHandle();
   if (api == null)
   {
      alert("Unable to locate the LMS's API Implementation.\nGetErrorString was not successful.");
   }
   return api.GetErrorString(errorCode).toString();
}

function doGetDiagnostic(errorCode) // Fehlerbehandlung
{
   var api = getAPIHandle();
   if (api == null)
   {
      alert("Unable to locate the LMS's API Implementation.\nGetDiagnostic was not successful.");
   }
   return api.GetDiagnostic(errorCode).toString();
}

function ErrorHandler() // Fehlerbehandlung
{
   var api = getAPIHandle();
   if (api == null)
   {
      alert("Unable to locate the LMS's API Implementation.\nCannot determine LMS error code.");
      return;
   }
   var errCode = api.GetLastError().toString();
   if (errCode != _NoError)
   {
      var errDescription = api.GetErrorString(errCode);
      if (_Debug == true)
      {
         errDescription += "\n";
         errDescription += api.GetDiagnostic(null);
      }
      alert(errDescription);
   }
   return errCode;
}
```

Anhang C

HTML-Code für ein SCO mit Testaufgabe

```
<!-- HTML-Code für ein SCO mit einer Testaufgabe (3 Radio-Button) -->

<html>
        <head>
                <meta http-equiv="expires" content="Tue, 05 DEC 2000 07:00:00 GMT">
                <meta http-equiv="Pragma" content="no-cache">
                <script language="javascript" src="util/Photoshop_APIWrapper.js"></script>
                <script language="javascript">

   // globale Variablen
   var answer = -1;  // in dieser Variable wird die Antwort des Lernenden gespeichert
   var key = 1;      // gibt an welche Antwort die richtige ist (hier erste Antwort)
   var rawScore = 0; // erreichte Punktzahl des Lernenden
   var exitPageStatus = false; // Status mit dem das SCO verlassen wird

   function loadPage()         // Laden des SCO (Launch)
   {
      var result = doInitialize(); // definiert in der APIWrapper.js Datei
      exitPageStatus = false;
   }

   function calcScore() // Auswertung des Testergebnisses
   {
      // Disable den 'Abschicken' Button
      document.examForm.submitButton.disabled = true;

      calcRawScore(); // Aufruf der Methode calcRawScore() zum ermitteln der Punktzahl

      // Transfer der erreichten Punktzahl an das LMS
      doSetValue( "cmi.score.scaled", rawScore );

      if ( rawScore != 1 ) // wenn die Punktzahl von 1 nicht erreicht wurde ...
      {
         // ... wird Erfolgsstatus auf nicht bestanden gesetzt
         doSetValue( "cmi.success_status", "failed" );
      }
      else
      {
         doSetValue( "cmi.success_status", "passed" ); // ... sonst bestanden
      }
      // unabhängig vom Erfolgsstatus wird der Abschlussstatus nach Bearbeitung auf
      // „abgeschlossen“ gesetzt
      doSetValue( "cmi.completion_status", "completed" );
      doSetValue( "cmi.exit", "" );
      exitPageStatus = true;      // SCO wurde normal beendet

      var result = doTerminate(); // Beenden der Kommunikation mit LMS
   }

   function calcRawScore() // Ermittlung der erreichten Punktzahl des Lernenden
   {
      // zurück setzen globaler Variablen
      answer = -1;
      rawScore = 0;

      for (var i=0; i <= 2; i++) // überprüfe alle Radio-Button Werte ...
      {
         if (document.examForm.Q1[i].checked)
         {
            answer = document.examForm.Q1[i].value; //... und speicher den Wert
                 break;
         }
      }

      if (answer == key) // wenn die Antwort der richtigen Antwort entspricht ...
      {
         rawScore++;     // ... erhöhe die Punktzahl um eins
      }
   }
   function unloadPage() // Beenden der Kommunikation
   {
          if (exitPageStatus != true)
          {
             doQuit();
          }
   }
```

```
   function doQuit() // Beenden der Kommunikation
   {
      calcScore();
   }
        </script>
            <body onLoad="loadPage()" onunload="return unloadPage()">
                 <div id="Div1" name="disp"></div>
                      <form name="examForm" ID="Form1"> Was ist eine EPROM?
                         <ol>
                         <li>
                              <input type="Radio" name="Q1" value="1" ID="Radio1">
                                      ein flüchtiger Speicher
                         <li>
                              <input type="Radio" name="Q1" value="2" ID="Radio2">
                              programmierbarer Festwertspeicher
                         <li>
                              <input type="Radio" name="Q1" value="3" ID="Radio3">
                              programmierbarer Festwertspeicher<br>
                         </ol>
                          <input type="button" name="submitButton" value="Abschicken"
                                 onClick="calcScore()" ID="Button1">
                      </form>
                 </body>
</html>
```

Anhang D

Quellcode einer Auswertungsseite

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
        <head>
                <title>Zusammenfassung &Uuml;bungsaufgaben</title>
                <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
                <link rel="stylesheet" type="text/css" href="basic/css/formate_mst.css">
                        <script language="JavaScript" src="file:///H|/Melanie/Tests/
                        Pers%F6nlicher%20Lernweg%20mit%20Pretest/HTML/util/APIWrapper.js"></script>
                        <script language="JavaScript">

   // Globale Variaben
   var exitPageStatus = false;

   // hier muss die Anzahl der Aufgaben insegsamt eingetragen werden
   var anzahlLektionen = 2;

   var objScaledScore = new Array(anzahlLektionen);
   var objSuccessStatus = new Array(anzahlLektionen);

   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);
      if ( objScaledScore[lektion] == null)
      {
         objScaledScore[lektion] = 0;
      }
          if ( objSuccessStatus[lektion] == null)
      {
         objSuccessStatus[lektion] = 0;
      }
   }

   function writeStartPage()
   {
                document.write("<HTML>");
                document.write("<HEAD>");
                document.write("<title>Zusammenfassung &Uuml;bungsaufgaben</title>");
                document.write("<link href=style.css rel=stylesheet type=text/css>");

                document.write("<table cellSpacing=0 cellPadding=0 width=800 border=0>");
                        document.write("<tr>");
                                document.write("<td width=25></td>");
                                document.write("<td vAlign=top colSpan=3>");
                                        document.write("<p class=mst-titel1>MILEON: Entwurf
                                                digitaler elektronischer Systeme</p>");
                                        document.write("<p class=mst-titel2>Lerneinheit: Rotating
                                                Disk <br><br> </p>");
                                document.write("</td>");
                                document.write("<td width=25></td>");
                        document.write("</tr>");
                        document.write("<tr>");
                                document.write("<td width=25></td>");
                                document.write("<td vAlign=top colSpan=3>");
                                document.write("<td width=25></td>");
                        document.write("</tr>");
                        document.write("<tr>");
                                document.write("<td width=25></td>");
                                document.write("<td vAlign=top colSpan=3>");
                                document.write("<p class=mst-titel3>Auswertung der
                                         &Uuml;bungsaufgaben </p>");
                                document.write("</td>");
                                document.write("<td width=25></td>");
                        document.write("</tr>");
                        document.write("<tr>");
                                document.write("<td width=25></td>");
                                document.write("<td id=xdivider vAlign=top colSpan=3>");
                                document.write("<td width=25></td>");
                        document.write("</tr>");
                        document.write("<tr>");
                                document.write("<td width=25></td>");

                                document.write("<td vAlign=top width=757>");
```

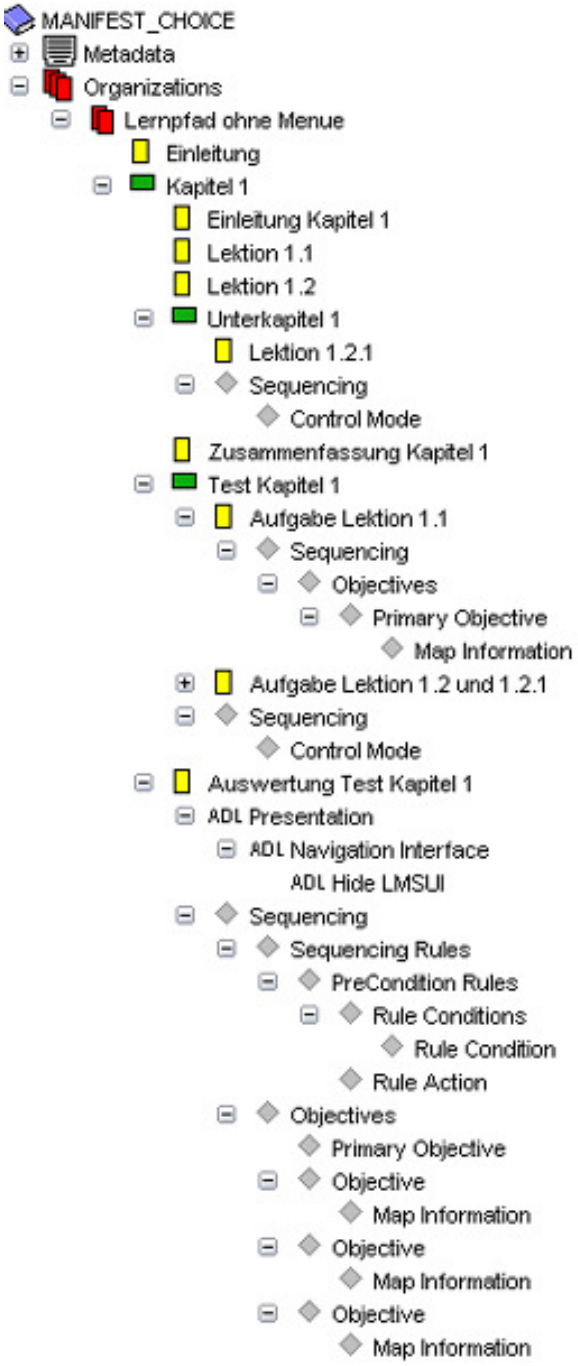
```
                                        document.write("<table height=19 cellSpacing=0
                                                cellPadding=0 width=761 border=0>");
                                        document.write("<tr><br></tr>");
                                        document.write("<tr>");
                                                document.write("<td class=container-orange
                                                        width=6></td>");
                                                document.write("<td width=4></td>");
                                                document.write("<td class=container
                                                        width=650>");
                                                document.write("<p class=mst-
                                                titel4-orange>Ihre        Testergebnisse:
                                                </p>");
                                                document.write("</td>");
                                        document.write("</tr>");
                                document.write("</table>");
                        document.write("</td>");
                document.write("</tr>");
        document.write("</table>");
}

function writeEndPage()
{
        document.write("</td></tr></table>");

        document.write("<table cellSpacing=0 cellPadding=0 width=800 border=0>");
                document.write("<tr>");
                        document.write("<td width=25></td>");
                        document.write("<td vAlign=top colSpan=3>");
        document.write("<p><br><br><b>Klicken Sie [Continue ->] um den ermittelten Lernpfad
                         abzuarbeiten</b>
                        <br>(Lernpfad kann nicht abgebrochen werden!),</p>");
        document.write("<p><b>oder navigieren Sie frei durch das Menü</b></p>");
                        document.write("</td>");
                        document.write("<td width=25></td>");
                document.write("</tr>");
        document.write("</table>");
        document.write("</BODY>");
        document.write("</HTML>");
}

function writeObjectiveStatus()
{
        document.write("<p>");
                document.write("<table cellSpacing=0 cellPadding=0 width=800 border=0>");
                        document.write("<tr>");
                                document.write("<td width=30 ></td>");
                                document.write("<td>");
                                        document.write("<table border=1 hspace=20>");

        for(j=0; j<anzahlLektionen; j++)
        {
                var tempScore = objScaledScore[j] * 100;
                tempScore = tempScore.toString();
                var pos = tempScore.indexOf(".");

                if( pos >= 0) //Punkt enthalten
                {
                        tempScore = tempScore.substring(0,pos);
                        tempScore = tempScore + "%";
                }
                else
                {
                        if(tempScore == 100)
                        {
                                tempScore = tempScore.substring(0,3);
                                tempScore = tempScore + "%";
                        }
                        else
                        {
                                tempScore = tempScore.substring(0,2);
                                tempScore = tempScore + "%";
                        }
                }
                document.write("<tr>");
                document.write("<td class=titel4 width=150>Lektion " + (j+1) + "</td>");
                document.write("<td class=titel4 width=80>" + tempScore + "</td>");
                document.write("<td class=titel4 width=20>");
                                //sonst falsches Bild bei Quit ohne Abschicken/Continue
                if( objSuccessStatus[j] == "passed" && tempScore != "0%")
                {
                        document.write("<img src=Grafik/passed.gif width=20 height=20>");
                }
                else
                {
                        document.write("<img src=Grafik/failed.gif width=20 height=20>");
                }
                document.write("</td>");
                document.write("</tr>");
```

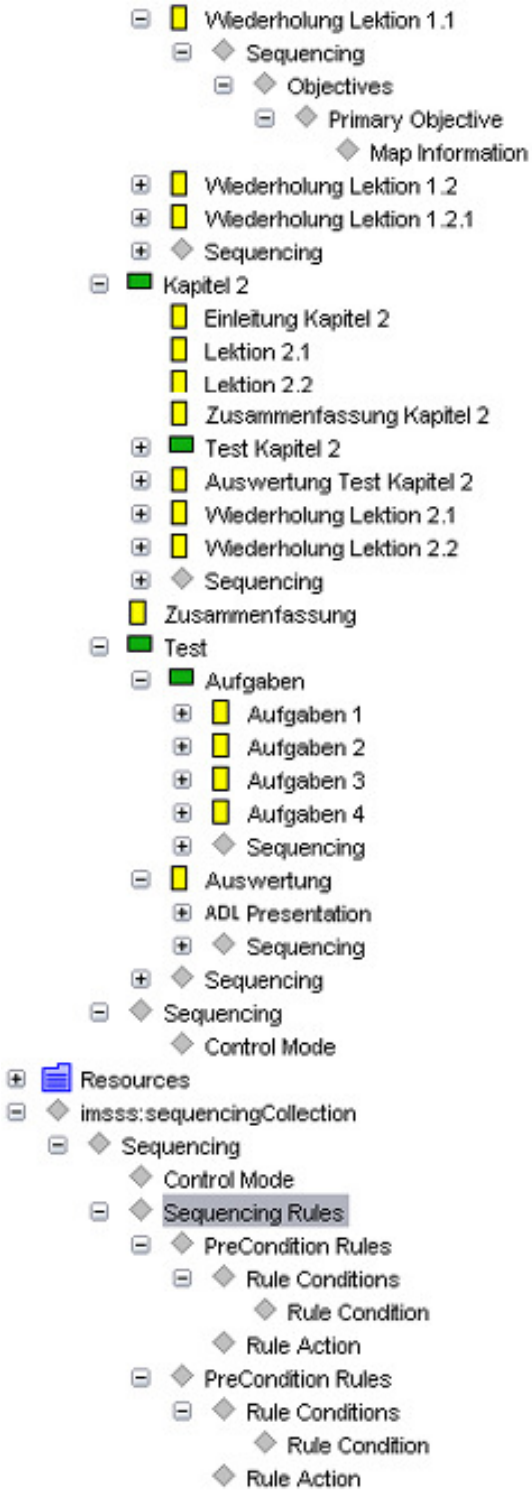
```
                }
                                                document.write("</table>");
                                            document.write("</td>");
                                    document.write("</tr>");
                            document.write("</table>");
                    document.write("</p>");
    }

    function loadPage()
    {
       var result = doInitialize();
       var lektion=0;
       exitPageStatus = false;
       writeStartPage();

       for (i=0; i<anzahlLektionen; i++)
       {
                    getObjectivesStatus(i);
       }
       writeObjectiveStatus();
       writeEndPage();
       doTerminate();
    }

    function findObjective(obj)
    {
             var numOfObj = doGetValue("cmi.objectives._count");
             var objectiveLocation = -1;

             for ( var i=0; i < numOfObj; i++ )
             {
            if ( doGetValue("cmi.objectives." + i + ".id") == obj )
                    {
                            objectiveLocation = i;
                            break;
                    }
             }
       if ( objectiveLocation == -1 )
       {
          objectiveLocation = numOfObj;
          doSetValue( "cmi.objectives." + objectiveLocation + ".id", obj);
       }
             return objectiveLocation;
    }
                            </script>
          </head>
          <body onLoad="loadPage()">
          </body>
</html>
```

Anhang E

Struktur des im 3. Kapitel vorgestellten Beispielkurses

Gleiche Abschnitte wurden teilweise für eine bessere Übersicht zusammengeklappt.




Wiederholung Lektion 1.1
Sequencing
Objectives
Primary Objective
Map Information
Wiederholung Lektion 1.2
Wiederholung Lektion 1.2.1
Sequencing
Kapitel 2
Einleitung Kapitel 2
Lektion 2.1
Lektion 2.2
Zusammenfassung Kapitel 2
Test Kapitel 2
Auswertung Test Kapitel 2
Wiederholung Lektion 2.1
Wiederholung Lektion 2.2
Sequencing
Zusammenfassung
Test
Aufgaben
Aufgaben 1
Aufgaben 2
Aufgaben 3
Aufgaben 4
Sequencing
Auswertung
ADL Presentation
Sequencing
Sequencing
Sequencing
Control Mode
Resources
imsss:sequencingCollection
Sequencing
Control Mode
Sequencing Rules
PreCondition Rules
Rule Conditions
Rule Condition
Rule Action
PreCondition Rules
Rule Conditions
Rule Condition
Rule Action

Anhang F

Systemvorraussetzungen für den prototypisch entwickelten KursEditor

Entwicklungsumgebung

Software

Microsoft Windows XP

Microsoft .NET Framework 1.1 (SP1)

Microsoft Visual Studio .NET 2003

Hardware

512 MB RAM

10 GB Festplatte

1 GHz CPU

Laufzeitumgebung

Software

Microsoft Windows XP

Microsoft .NET Framework 1.1 (SP1)

Hardware

256 MB RAM

5 GB Festplatte

600 MHz CPU

Anlagen

CD mit folgendem Inhalt:

- Programm KursEditor (Prototyp)
- Microsoft .NET Framework 1.1 (SP1)
- Datei *APIWrapper.js* (SCORM 1.3)
- Datei *Aufgabe.htm* (Quellcode für SCO mit Testaufgabe nach SCORM 1.3)
- Datei *Auswertung.htm* (Quellcode für eine Auswertungsseite nach SCORM 1.3)
- Schematas für SCORM 1.3
- XML-Templates für die Generierung der Manifest-Datei mit dem KursEditor
- Beispiel-Manifest für die Generierung aus XML-Templates mit dem KursEditor
- diese Arbeit im PDF und DOC-Format (Microsoft WORD 2002)

Selbständigkeitserklärung

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

Dresden, 12.09.2005

Melanie Pietzsch