



C# Viewer for CPS-packages for Handhelds

MAGISTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Masterstudium

INFORMATIK

Eingereicht von:

Willibald Hötzingler, 9555566

Angefertigt am:

Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM)

Betreuung:

o.Univ.-Prof. Dr. Jörg R. Mühlbacher

Dipl.Ing. Dr. Susanne Loidl

Pfaffing, März 2007

Kurzbeschreibung

Diese Diplomarbeit hat das Ziel, eine Applikation (im Titel „C# Viewer“ genannt) zu entwickeln, welche CPS-Kurse auch für Handhelds zugänglich macht. Dazu müssen sowohl der Transfer von Kursmaterialien auf den PDA als auch das Abrufen dieser Inhalte auf diesem möglich sein.

Die größte Herausforderung bei der Erstellung dieses Viewers liegt in der Beschaffenheit der Handhelds, die sich von herkömmlichen Desktop-Geräten doch wesentlich unterscheiden. In der Realisierung muss daher eine universell einsetzbare Präsentation der Kurse gefunden werden. Aber vor allem der Datenabgleich (Bereitstellen der Unterlagen auf dem Handheld, Versionsverwaltung, etc.) stellt eine besondere Herausforderung dar. Kurse sind oftmals mehrere hundert Megabyte groß, d.h. dass nicht die gesamte Datenmenge auf dem Handheld verfügbar sein wird bzw. dass es spezieller Komprimierungsalgorithmen bedarf.

Diese Fragen werden in dieser Diplomarbeit geklärt und die Lösung schließlich als Applikation umgesetzt. Als Entwicklungsumgebung dient dabei Microsoft's „.NET“-Technologie, die unter anderem die für Pocket PC-Programmierung geeignete Sprache „C#“ zur Verfügung stellt.

Abstract

This diploma thesis has the goal of developing an application (called "C # Viewer" in the title) which makes CPS courses accessible for handhelds. For this purpose both the transfer of course materials onto the PDA and viewing these contents on this must be possible.

The greatest hurdle in implementing this viewer lies in the condition of handhelds, which substantially differs from conventional desktop devices. Therefore in the realization a universally usable presentation of the courses must be found. But above all the data adjustment represents a special challenge (make the documents available on the handheld, manage different versions of course content, etc.). Courses are often several hundred megabyte large, that means that possibly the entire amount of data can not be present on the handheld and maybe that special compression algorithms are required.

These questions are solved in this diploma thesis and the solution finally is realised as an application. As development environment there acts Microsoft's ".NET" technology which among other things provides the language "C#", which is particularly suitable for pocket PC programming.

Danksagung

An dieser Stelle möchte ich mich bei jenen Personen herzlich bedanken, die mich in den Jahren meines Studiums begleiteten und mir in manch schwerer Stunde hilfreich zur Seite standen. Auch ohne namentliche Erwähnung wissen diese sehr genau, dass ihnen hiermit aus ganzem Herzen gedankt sei.

Für die gute Betreuung und hilfreichen Ratschläge gebührt Frau Dipl.-Ing. Dr. Susanne Loidl ebenfalls besonderer Dank. Vor allem aber dafür, dass sie mich immer wieder motivierte, diese Arbeit abzuschließen und dabei eine Menge Geduld aufbrachte.

Ähnliches gilt für Herrn o.Univ.-Prof. Dr. Jörg R. Mühlbacher. Ihm habe ich zu danken, dass mir diese Arbeit am von Ihm geleiteten Institut anvertraut wurde und er mir bei deren Fertigstellung, die nicht zuletzt unter meinen beruflichen Verpflichtungen litt, einen großen zeitlichen Spielraum ließ.

Vielen Dank!

Inhaltsverzeichnis

Kurzbeschreibung.....	2
Abstract	3
Danksagung.....	4
Inhaltsverzeichnis.....	5
1. Einleitung und Motivation	7
2. eLearning.....	10
3. Web Environment for Learning (WeLearn).....	14
3.1. WeLearn Offline Konverter	14
4. Charakteristik von PDA's	15
5. Grundlagen.....	19
5.1. Extensible Markup Language (XML).....	19
5.2. Strukturbeschreibungen.....	21
5.3. Content Packaging Specification (CPS).....	23
6. CPS Manifest.....	26
6.1. <manifest>.....	26
6.2. <metadata>	27
6.3. <organizations>.....	29
6.4. <resources>	30
6.5. Sub-<manifest>	31
7. Praktischer Teil	33
7.1. Entwicklungsumgebung	34
7.2. Manifest Administrator	35
7.2.1. Manifest Converter.....	37
7.2.2. Manifest Updater.....	51
7.3. Manifest Viewer.....	53
8. Literaturverweise.....	79
9. Abbildungsverzeichnis	84
10. Tabellenverzeichnis.....	87
Eidesstattliche Erklärung.....	88
Curriculum Vitae.....	89
Anhang	91

Manifest Administrator – Benutzungsanleitung	91
▪ Installation	91
▪ Programmbedienung	91
Manifest Viewer – Benutzungsanleitung	97
▪ Installation	97
▪ Programmbedienung	98

1. Einleitung und Motivation

„Bücher werden in unseren Schulen bald überflüssig sein ... Man kann jede Art von menschlichem Wissen mit der neuen Technik lehren.“ Thomas Alva Edison, 1913

Noch vor wenigen Jahren galt „eLearning“ als DIE Bildungsform des 21. Jahrhunderts. Es handelt sich hierbei um eine Form des Lernens, die durch multimediale und telekommunikative Technologien (z.B. CD, Internet) unterstützt wird, wobei Lernende und Lehrende wie beim klassischen Fernlernen räumlich getrennt sein können. Mittlerweile ist man jedoch zu der Erkenntnis gelangt, dass eLearning die traditionellen Bildungsformen nicht ersetzen wird, sondern vielmehr als eine sinnvolle Unterstützung im Lernprozess zu sehen ist, die den Lernenden im Lernen zusätzlich fördern und – im positiven Sinne – fördern kann.

Durch eLearning bzw. Blended Learning, eine Mischform aus realen und virtuellen Lernräumen, wird den Beteiligten mehr Flexibilität eingeräumt, da eine räumliche und zeitliche Unabhängigkeit entsteht. Lernende können nicht nur während der Präsenzphasen Fragen abklären, sondern sich im virtuellen Lernraum jederzeit austauschen. Gerade in Hinblick auf „life-long learning“ sind diese Aspekte nicht zu unterschätzen, denn berufsbegleitendes Lernen erfordert zumeist eine hohe Flexibilität bezüglich Ort und Zeit. Blended Learning ist oftmals auch erfolgreicher, da das Lernmaterial ständig verfügbar und mit Hands-on-Elementen angereichert ist. Lernen kann selbstgesteuert werden und die Betreuung durch den Lehrenden wird verbessert, da sich dieser üblicherweise intensiver mit den Lernenden auseinandersetzt.

Darüber hinaus können die Lerninhalte individuell an die Bedürfnisse des jeweiligen Benutzers angepasst werden, wobei dieser natürlich sein eigenes Tempo bestimmen und den Lernstoff beliebig oft durchgehen kann. Weiters kann dem Lernenden die Möglichkeit geboten werden, sein Verständnis für das Stoffgebiet durch geeignete Testverfahren („Self Assessment“) zu überprüfen.

Die Nutzung von PC und Internet bringt aber noch weitere Vorteile mit sich: Geeignete eLearning-Plattformen erlauben beispielsweise das interaktive Zusammenarbeiten zwischen Lernenden. So können gegenseitig Fragen gestellt und beantwortet werden, Aufgaben gemeinsam gelöst werden usw. Darüber hinaus besteht natürlich die Möglichkeit, Audio- und

Videodokumente in die Kurse einzubinden, was eine neue Dimension im Vermitteln von Lerninhalten darstellt.

Zusätzliche Pluspunkte von eLearning sind die Wiederverwendbarkeit und Weiterentwicklungsmöglichkeiten von Lerninhalten. So können bestehende Kurse zur Gänze oder auch einzelne Teile davon in einen neuen Kurs eingebettet werden, wodurch sich mitunter durchaus beachtliche Synergieeffekte ergeben.

Es herrscht also sicherlich nicht die Absicht vor, Bücher aus den Schulen und Universitäten zu verbannen. Ziel muss es sein, den Lernenden mittels moderner eLearning-Technologien ein Werkzeug zur Verfügung zu stellen, das sie im Lernprozess zusätzlich unterstützt und nach Möglichkeit motivierend auf diese einwirkt. Das elektronische Lernsystem muss eine intelligente, individuelle Anpassung an die Voraussetzungen und Wünsche des Lernenden erlauben.

Am Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM) der Johannes Kepler Universität Linz wurde schon sehr bald das Potential, das im Konzept des eLearnings steckt, erkannt. So wurde neben der Lernplattform „WeLearn“ auch ein Konverter entwickelt (näheres in Kapitel 2), der die Onlinekurse in ein Format konvertiert, welches es ermöglicht, dass diese Kurse auch losgelöst von der webbasierten Lernplattform in offline-Form abgerufen werden können. Die Studenten sind beim Arbeiten mit solchen elektronischen Kursen also nicht zwingend an „WeLearn“ gebunden. Vielmehr reicht bereits der eigene PC und beispielsweise eine CD, die den entsprechenden konvertierten Kurs enthält, um dessen Inhalte abrufen zu können.

Doch nicht nur elektronische Lern-Plattformen erfreuen sich wachsender Beliebtheit: Auch Handhelds gewinnen immer mehr an Bedeutung im Alltag. Was liegt folglich näher, als die Idee des „eLearning“ auch auf diesen Bereich auszuweiten? Man stelle sich eine langweilige Zugfahrt vor – noch 2 Stunden bis zum Zielbahnhof. Und plötzlich packt einen die Lust, sich um die eigene Weiterbildung zu kümmern. Kein Problem! Das mobile Gerät heraus – und schon kann es losgehen! Doch um diese flexible Lernform (mLearning) zu ermöglichen, ist zuerst eine Menge Konzeptions- und Implementierungsarbeit zu leisten. Aus diesem Grund wurde am Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM) diese Magisterarbeit initiiert, mit dem Ziel, ein Softwaresystem zu entwickeln, welches das gezielte

Laden, Anzeigen und Löschen von Kursmaterialien auf Handhelds ermöglicht. Dieses Softwaresystem sollte die Einschränkungen, die PDA's mit sich bringen, berücksichtigen und dem Lernenden eine Lösung anbieten, die es ihm ermöglicht, die Vorzüge des Einsatzes der mobilen Geräte zu nutzen.

2. eLearning

Das Lernen im 21. Jahrhundert sieht sich einem gewissen Paradigmenwechsel gegenüber. Aufgrund unserer heutigen, schnelllebigen Zeit sind die Lernenden angehalten, Kompetenzen zu entwickeln, die es ermöglichen, im Anlassfall Informationen und Wissen schnellstmöglich zu akquirieren. Mit eLearning wird dabei die Hoffnung verbunden, Wissen rasch, zielgerichtet und frei von räumlichen und zeitlichen Barrieren vermitteln zu können und somit berufs- bzw. lebensbegleitendes Lernen zu fördern.

Allgemein betrachtet handelt es sich bei eLearning um eine spezielle Form des computergestützten Lernens, d.h. Lehr- und Bildungsprozesse werden durch technische Hilfsmittel unterstützt. Folgende Aspekte sind dabei charakteristisch für eLearning:

- Ort und Zeitpunkt des Lernens (und Lehrens) sind frei wählbar (just-in-time-learning).
- Individualisierung des Lernens: Ausbildungsziele und -schritte können vom Nutzer selbst bestimmt werden; "Intelligente" Software kann sich an die Lerngeschwindigkeit anpassen.
- Entwicklung und Ermöglichung größerer Interdisziplinarität und Internationalität: unterschiedlichste Fachbereiche verschiedener Länder und Universitäten können leichter und fruchtbringender zusammenarbeiten und von- bzw. miteinander lernen.
- Multimediale Aufbereitung und Verbreitung des Lehrinhaltes: Animationen und Simulationen können komplexe Sachverhalte verständlicher machen und die Motivation des Lernenden fördern.
- Multimediale Techniken erleichtern den Zugriff auf Informationen in Datenbanken und elektronischen Bibliotheken und können zusätzliche Suchfunktionen bieten.
- Wissen kann schneller publiziert und verbreitet werden.
- Neue Formen der Telekooperation zwischen Lehrenden und Lernenden, aber auch zwischen Lernenden bzw. Lehrenden untereinander (in virtuellen Diskussionsforen oder Arbeitsgruppen) können Kreativität beim Lernen freisetzen und Expertenaustausch ermöglichen.

[SCH96]

Nach der Definition der Züricher Hochschule Winterthur ist eLearning durch eine Verschmelzung von Ausbildung und Internet gekennzeichnet, da „Angebot und Vermittlung

von Wissensinhalten unter Einsatz modernen Technologien (v.a. Computern) realisiert wird“. Die Lernumgebungen basieren in erster Linie auf den Diensten des Internet (WWW, eMail, usw.). Studierende und Dozierende können somit räumlich und/oder zeitlich getrennt sein. Um die Lernprozesse leiten und unterstützen zu können, besteht zusätzlich die Möglichkeit der Kommunikation, wobei diese synchron (Chat) und/oder asynchron (eMail, Forum, usw.) stattfinden kann. Als typische Merkmale von eLearning werden genannt:

- Der Zugang zu Kursinhalten ist zeitlich und räumlich nicht beschränkt.
- Die Inhalte sind dynamisch und aktuell.
- Die Inhalte können beliebig vernetzt werden, d.h. Interaktion mit Lehrern und anderen Studierenden ist möglich.
- Die Kurse sind erweiterungsfähig.
- Performance und Lernergebnisse können verfolgt werden.
- Die Anpassung an individuellen Lernstil und Lerngeschwindigkeit ist möglich.
- Auf Benutzerseite kann standardisierte Hardware und Software eingesetzt werden (normalerweise genügt Browser plus Internetanschluss).

[ZHW02]

Lernende haben mittels eLearning die Möglichkeit, in beliebiger Reihenfolge auf die Lernmaterialien zuzugreifen, wodurch sie die die zu vermittelnden Informationen nach eigenem Ermessen logisch strukturieren können. Darüber hinaus können sie beim Erlernen der Kursinhalte wie bereits erwähnt ihr individuelles Tempo wählen und gegebenenfalls Teile beliebig oft wiederholen. Die Möglichkeit zur freien Zeiteinteilung beim Lernen setzt jedoch ein hohes Maß an Selbstdisziplin und Eigenverantwortung voraus. Um die Motivation des Lernenden zu fördern, sollte nach Strzebkowski darauf geachtet werden, dass bei der Entwicklung eines multimedialen Lernproduktes neben der Vermittlung von Wissen auch die Eigenaktivität des Lernenden gefördert wird. Dies kann erreicht werden durch

- die Einbettung des Lerngegenstandes in authentische und komplexe Situationen,
- die Konfrontation mit mehreren Perspektiven und Kontexten eines Sachverhaltes,
- die vorwiegend explorative und assoziative Vorgehensweise bei der Erschließung neuer Informationen,
- die Anregung zum "Learning by doing",
- die Möglichkeit zur Konstruktion eigener Inhalte und Medien-Welten,

- die Möglichkeit der Artikulation und der Selbstreflexion über die eigenen Lern- und Lösungsstrategien,
- die sofortige Anwendung des Gelernten auf lebensnahe Problemsituationen.

[STR97]

Die Web-Plattform „LebensLanges Lernen im Web“ (W3L) nennt zehn „goldene“ Regeln, die bei der Erstellung von online-Kursen berücksichtigt werden sollten, um der Motivation der Lernenden dienlich zu sein:

- Der Lernende kann verschiedene Lernstile wählen und zwischen ihnen beliebig wechseln.
- Der Lernende kann individuell und kooperativ lernen.
- Der Lernende wird durch menschliche Mentoren/Tutoren betreut.
- Der Lernende wird aktiviert und aktiv gehalten.
- Der Lernstoff wird multimedial angepasst an den Inhalt gestaltet.
- Der Lernstoff sollte sinnvoll intern und sparsam extern verlinkt sein.
- Der Lernende muss jederzeit seinen Wissensstand überprüfen können.
- Der Lernende sollte alle 20 bis 30 Minuten ein Erfolgserlebnis haben.
- Der Lernstoff muss aktuell sein.
- Lernende müssen eigene Wissensbausteine erstellen können.

[W3L]

Bei Einhaltung dieser Regeln eröffnen sich durchaus große Chancen durch den Einsatz "neuer" Medien im Bildungsbereich, da neben den reinen technologischen Vorteilen vor allem verbesserte Möglichkeiten im Bereich der Information (Recherchieren, Dokumentieren), der Kommunikation (Interagieren, Kooperieren) sowie der Kognition (Simulieren, Visualisieren, Animieren) geschaffen werden.

Den Vorteilen von eLearning stehen allerdings - zumindest momentan - folgende Nachteile gegenüber:

- Soziale Austauschmöglichkeiten über Datennetze erreichen nicht die Qualität der Diskussion und die Interaktionsangebote in Seminaren trotz IRC, elektronischen Foren und Mailinglisten (fehlender persönlicher Kontakt zum Dozenten, keine Teilhabe am "sozialen Erlebnis Lernen").

- Anonymisierung von Bildungseinrichtungen.
- Übermäßige Standardisierung von Lehrinhalten nicht möglich.
- Zusätzliche Kosten für die Datenübertragung.
- Fehlende oder mangelhafte Kriterien für die Beurteilung virtueller erbrachter Leistungsnachweise.
- Mangelnde soziale und emotionale Akzeptanz von Computermedien in weiten Bevölkerungskreisen bzw. traditionelle kulturpessimistische Haltungen gegenüber dem Einsatz "neuer" Medien in Hochschulkreisen
- Große anfängliche Anforderungen an die Technikausstattung sowie an die Nutzungskompetenz Lehrenden, da Lehrmaterialien für das Internet neu konzipiert und mediengerecht aufbereitet werden müssen.

[SCH96]

Es liegt auf der Hand, dass die Möglichkeit des Einsatzes technischer Hilfsmittel nicht automatisch die Lernprozesse verbessert. Nach einem eLearning-„Hype“ machte sich daher um die Jahrtausendwende eine gewisse Ernüchterung breit. Man musste erkennen, dass viele elektronische Lernsysteme es nicht vermochten, individuell auf die Lernenden einzugehen und sie bei Problemen und auftretenden Fragen zu unterstützen. Folglich entwickelte sich der Trend dahingehend, eLearning mit Präsenzlehre zu kombinieren („Blended Learning“). Dabei werden in der Regel Online-Phasen zur Vor- oder Nachbereitung von Präsenztrainings verwendet. Die Lernenden haben somit beispielsweise die Möglichkeit, sich in Eigenregie ein Grundwissen anzueignen, wodurch die Anlaufphase eines Präsenzkurses, in der etwaige Defizite einzelner Lernender aufgearbeitet werden müssen, entfallen kann. Somit bleibt umso mehr Zeit für Übungen, Diskussionen und Anwendung des Gelernten.

Mit dem Einsatz von eLearning wird also nicht die Abschaffung von Lehrkräften angestrebt. Vielmehr spielen diese in diesem Konzept eine ganz zentrale Rolle – sie begleiten, animieren und motivieren die Lernenden im Lernprozess und stehen diesen bei Problemen unterstützend zur Seite. H. Schauer schreibt in [SCH01]: „Die Kompetenz und Autorität des Lehrers darf nicht auf dessen Wissensvorsprung gegenüber dem Schüler basieren, sondern muss aus der Didaktik, Methodik und Persönlichkeit des Lehrers hervorgehen.“

3. Web Environment for Learning (WeLearn)

Wie oben bereits kurz angesprochen, wurde am Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM) der Johannes Kepler Universität Linz die frei verfügbare Lernplattform „WeLearn“ (Web Environment for Learning“) [WEL] entwickelt, um der Idee des eLearnings gerecht werden zu können. Es handelt sich dabei um eine webbasierte Umgebung, welche kooperatives Lernen und Unterrichten ermöglicht. Kursanbieter stellen dabei den Lernenden die Kursmaterialien online zur Verfügung, wobei diese Kurse aus Gründen der Wiederverwendbarkeit strikt nach der Content Packaging Specification des IMS erstellt werden. Bedient wird die Plattform ausschließlich über Browser, wie z.B. Mozilla Firefox, Microsoft Internet Explorer oder Opera.

Neben dem reinen Sichten von Content haben die Kursteilnehmer aber auch die Möglichkeit, untereinander bzw. mit dem Kursleiter via eigens eingerichteter Foren zu kommunizieren, um über die behandelte Themen oder anfallende Probleme zu diskutieren.

Kurz gesagt zielt das Projekt „WeLearn“ darauf ab, sowohl dem Lehrenden als auch den Lernenden eine Plattform zu bieten, die es vermag, den Lernprozess positiv zu beeinflussen.

3.1. WeLearn Offline Konverter

Um das Lernen von Kursmaterialien auch offline – sprich ohne Internetverbindung – zu ermöglichen, wurde zusätzlich der sogenannte „WeLearn Offline Konverter“ implementiert. Dieser wandelt CPS-konforme Kurse in statische Webseiten um (z.B. DHTML, Java-Applet), welche anschließend ohne weiteres auf CD gebrannt und verteilt werden können.

4. Charakteristik von PDA´s

Bevor mit der Implementierung des Viewers begonnen wurde, musste die Charakteristik der Handhelds genauer unter die Lupe genommen werden. Diese weisen im Vergleich zu herkömmlichen Desktop- bzw. Laptop-Geräten doch fundamentale Unterschiede auf, welche im Zuge der Realisierung möglicherweise zu Einschränkungen und Problemen führen könnten, sofern keine geeigneten Problemlösungskonzepte ausgearbeitet werden. Die wichtigsten Merkmale von Handhelds sind:

- **Geringe Displaygröße, niedrigere Auflösung**

Das Display von PDA´s ist klarer Weise deutlich kleiner als jenes herkömmlicher Rechner. Die beiden von mir verwendeten Geräte weisen beispielsweise eine Displayhöhe von ca. 8 cm und eine Breite von ca. 6 cm auf (siehe Abbildung 1).



Abbildung 1: Compaq iPAQ 3970 und HP iPAQ hx 4700

Aber auch in Punkto Auflösung können Handhelds derzeit noch nicht wirklich mithalten. Während ein durchschnittlicher 17" Monitor ohne Weiteres eine Auflösung von bis zu 1280*1024 Pixel erlaubt, liegt diese bei PDA´s zwischen 160*160 und 480*640 Pixel. Eines der von mir bei meiner Diplomarbeit verwendeten Geräte – der Compaq iPAQ 3970 –

verwendet eine Auflösung von 240*320 Pixel. Das andere Gerät der Serie HP iPAQ hx 4700 wartet bereits mit einer Auflösung von 480*640 Pixel auf.

Neben dem signifikant kleineren Display unterscheidet sich aber auch dessen Form grundlegend von jener der Desktop- und Laptop-Monitore. So sind die Anzeigebereiche der Handhelds – abgesehen neuerer Geräte, die mit dem Betriebssystem Windows Mobile 2003 ausgestattet sind – höher als breit, während ein Desktop-Monitor breiter als hoch ist. Der von mir eingesetzte HP iPAQ hx 4700 läuft auf Windows Mobile 2003 und erlaubt folglich das Umschalten von üblichen „Portrait“- in den querformatigen „Landscape“-Modus (siehe Abbildung 2).



Abbildung 2: Landscape-Format

Die Präsentation der Informationen auf dem PDA ist also ungleich schwieriger als am Desktoprechner. Um die Fülle an Informationen am PDA darstellen zu können, sind die Informationen auf mehrere Fenster aufzuteilen. Dem Benutzer sollte jedoch möglichst wenig Zeit- und Arbeitsaufwand bei Auffinden der gewünschten Funktionen bzw. Inhalte erwachsen, weshalb vom Entwickler eine logische und nachvollziehbare Strukturierung der Informationsmenge gefunden werden muss. Darüber hinaus sollte die Strukturierung aber auch nach Relevanzkriterien, d.h. häufig verwendete Informationen sind einfacher und schneller zu erreichen, erfolgen.

Die Konsequenz aus diesen Überlegungen ist die Reduktion der Funktionalität sowie der Fokus auf das Wesentliche, um sowohl Einfachheit als auch Übersichtlichkeit in der Bedienung gewährleisten zu können.

- **Geringe Akkuleistung**

Auch die doch sehr limitierte Akkuleistung von Handhelds hat Einfluss auf meine konzeptuellen Überlegungen bezüglich Entwicklung des Manifest Viewers. Die sehr rechenintensiven Funktionen bezüglich Konvertierung der eLearning-Kurse werden demnach auf Standgeräten ausgeführt.

- **Umständliche Eingabe**

Im Gegensatz zu Desktoprechnern haben PDA's in der Regel weder eine Maus noch eine Tastatur. Vielmehr sind berührungs-sensitive Displays – so genannte „Touchscreens“ – üblich. Diese können mit einem Stift bedient werden, wobei die Eingabe entweder über eine virtuelle Tastatur, die bei Bedarf ins Display eingeblendet wird, oder über ein Handschrift-Erkennungsverfahren erfolgt.

Die stiftbasierte Eingabe ist jedenfalls langsamer als eine Eingabe per Tastatur, denn schon das präzise Eingeben per Stift ist nicht wirklich einfach. Oftmals werden auch Zeichen nicht bzw. falsch erkannt. Man stelle sich nur eine Busfahrt vor, bei der Benutzer und PDA ständig Erschütterungen ausgesetzt sind. Noch problematischer wird es, falls der Benutzer keine Hand für die Eingabe per Stift frei hat, weil er sich beispielsweise an einem Haltegriff in der Straßenbahn fest hält.

Folglich muss bei der Entwicklung des Viewers Bedacht darauf gelegt werden, so wenig wie möglich Benutzereingaben erforderlich zu machen und Interaktionselemente der GUI ausreichend groß darzustellen.

- **Schwache Prozessorleistung**

Während moderne Desktoprechner eine Taktrate von bereits über 3 GHz erreichen, kommen derzeit handelsübliche Handhelds maximal auf lediglich 400 MHz. Sie sind leistungsmäßig also mit stationären Rechnern von vor sieben bis acht Jahren vergleichbar. Einen wesentlichen Vorteil bringen die relativ bescheidenen Taktraten mit sich: Einen deutlich geringeren Energieverbrauch, was natürlich der Akkulaufzeit zugute kommt.

Angesichts der (noch) schwachen Prozessorleistungen ist bei der Implementierung des Manifest Viewers die Zeitkomplexität bestimmter rechenintensiver Funktionen von besonderer Bedeutung. Dem Benutzer kann schließlich nicht zugemutet werden, bei wichtigen und oftmals auszuführenden Aufgaben allzu lange auf die Ausgabe des Ergebnisses warten zu müssen.

- **Häufig geringe Übertragungsbandbreite**

Ein PDA kann entweder mit einem seriellen oder einem USB-Kabel aber oftmals auch über eine Infrarot-Schnittstelle mit einem Desktoprechner verbunden werden. In den beiden ersteren Fällen wird eine Dockingstation (Cradle) verwendet, in die der PDA eingesteckt wird. Moderne Geräte ermöglichen zusätzlich die Verbindung zu lokalen Netzwerken mittels Bluetooth oder Wireless LAN. Wird über ein serielles Kabel bzw. Infrarot kommuniziert, ist man mit einer sehr geringen Übertragungsgeschwindigkeit konfrontiert. So werden beispielsweise via Infrarot maximal 115 kBit erreicht, was bei großen Datenmengen doch eine entsprechende Portion Geduld des Benutzers erforderlich macht.

Folglich ist aus Entwicklersicht danach zu trachten, das zu übertragende Datenvolumen so gering als möglich zu halten.

- **Geringe Speicherkapazität**

Die Kapazität des Hauptspeichers liegt bei PDA's zwischen 8 und 64 MB. Wichtig anzumerken ist, dass die meisten PDA's keine Festplatte und damit keinen persistenten Speicher haben. Die Daten bleiben also nur so lange im Hauptspeicher, so lange der Akku den Speicher mit Strom versorgt.

Abhilfe schaffen Erweiterungen wie eine CompactFlash-Karte, ein Memory Stick oder ein Microdrive. Diese gehören allerdings nicht zur Standardausstattung eines PDA's, sind eher hochpreisig und erlauben auch nur eine Erweiterung um höchstens 1 Gibabyte. Bedenkt man, dass Festplatten von heutigen Desktop-PC's Speicherkapazitäten von mehreren hundert Gigabytes haben, ergeben sich hier natürlich deutliche Diskrepanzen.

Als Konsequenz sollte der Viewer möglichst wenig Speicher in Beschlag nehmen, was bedeutet, das es auch möglich sein muss, lediglich Teilpakete eines Kurses auf das mobile Gerät zu laden.

- **Fehlen eines CD-Laufwerks**

Da PDA's keine CD-Laufwerke besitzen, können die auf CD's verfügbaren Kurse natürlich nicht direkt auf die mobilen Geräte kopiert werden. Der Viewer muss also eine Übertragung entweder durch serielle bzw. kabellose Kommunikation (insbesondere über WLAN) ermöglichen.

5. Grundlagen

5.1. Extensible Markup Language (XML)

Heutzutage sind Begriffe wie Datenaustausch zwischen Anwendungen und Offenheit zu fremden Applikationen und Plattformen in aller Munde. Dazu ist es notwendig, dass Vereinbarungen über Austauschformate vorliegen, denn nur so lassen sich die Daten in Echtzeit ohne manuellen Eingriff verarbeiten. Ohne solche Standards ist der Austausch von Daten nicht nur zu teuer, sondern auch langsam und überaus fehleranfällig.

Um diese hochgesteckten Ziele erreichen zu können, wurde die offene Auszeichnungssprache „Extensible Markup Language“ (XML) entwickelt, mit deren Hilfe sicher gestellt werden soll, dass auch in ferner Zukunft XML-codierte Informationen noch interpretier- und weiterverarbeitbar sind. Oberstes Gebot der XML-Philosophie ist die strikte Trennung von Inhalt, welcher durch das XML-Vokabular strukturiert wird, und der Darstellung dieses Inhalts (Layout). In diesem Punkt liegt auch ein ganz wesentlicher Unterschied zu altbekannten HTML-Dokumenten, die eine Verschmelzung von Inhalt, Struktur und Präsentation darstellen.

Verantwortlich für die Entwicklung von XML zeichnet sich eine unter der Schirmherrschaft des World Wide Web Consortiums (W3C) stehende Arbeitsgruppe, die 1996 ins Leben gerufen wurde und sich aus SGML-Experten zusammensetzte (u.a. J. Goldfarb) [BRA00]. Das W3C selbst wurde übrigens 1994 gegründet, und zwar mit dem Ziel, Webtechnologien voranzutreiben und offene Standards zu definieren, die im gesamten World Wide Web Gültigkeit haben und somit ein zuverlässiges Zusammenarbeiten fördern. Die Webseiten des W3C sind unter „<http://www.w3.org>“ zu finden, jene des deutschen W3C-Büros unter „<http://www.w3.org/Consortium/Offices/Germany>“.

Die eigens eingesetzte Arbeitsgruppe definierte die Entwurfsziele für XML folgendermaßen:

- XML soll sich im Internet auf einfache Weise nutzen lassen.
- XML soll ein breites Spektrum von Anwendungen unterstützen.
- XML soll zu SGML kompatibel sein.
- Es soll einfach sein, Programme zu schreiben, die XML-Dokumente verarbeiten.

- Die Zahl optionaler Merkmale in XML soll minimal sein, idealer Weise gleich null.
- XML-Dokumente sollten für Menschen lesbar und angemessen verständlich sein.
- Der XML-Entwurf, d.h. die Erstellung der XML-Spezifikation, sollte zügig abgefasst sein.
- Der Entwurf von XML soll formal und präzise sein.
- XML-Dokumente sollen leicht zu erstellen sein.

[XML98]

Als Ergebnis konnte schließlich eine XML-Spezifikation präsentiert werden, die im Februar 1998 vom W3C zu guter Letzt den Status einer Empfehlung („Recommendation“) erhielt. Bis eine W3C-Spezifikation diesen Status zugewiesen bekommt, sind zuvor eine Reihe von Phasen zu durchlaufen. Angefangen vom ersten Arbeitsentwurf („Working Draft“) über verschiedenste Entwurfs- und Empfehlungsstadien bis hin zur W3C-Empfehlung, die nur ausgesprochen wird, falls die im technischen Bericht dargelegten Ideen und Technologien für die Umsetzung der Ziele des W3C förderlich sind und somit vom W3C als verbreitenswert eingestuft werden (siehe Abbildung 3).

[XML02]

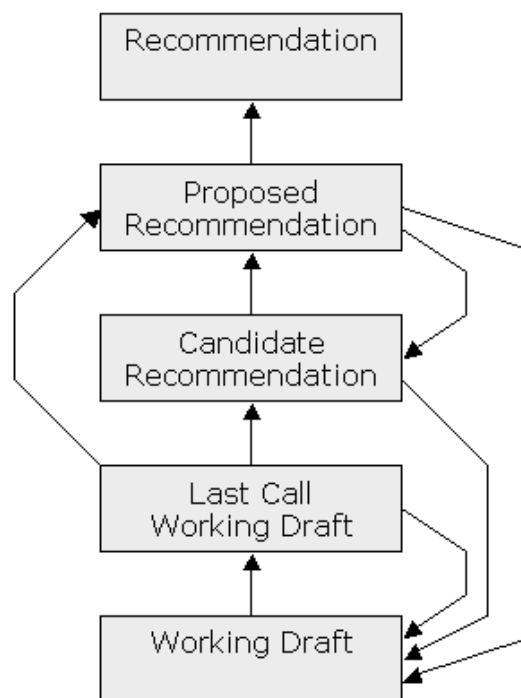


Abbildung 3: Stufen einer Spezifikation bis zur W3C-Empfehlung

Die bei der Erstellung von XML-Dokumenten zu berücksichtigenden Spezifikationen sind frei zugänglich und unter [XML98] abrufbar.

Der – wie oben bereits erwähnt – mit einer W3C-Empfehlung versehene XML-Standard bringt eine Reihe von konkreten Vorteilen mit sich.

- XML ist ein allgemein anerkanntes und erweiterbares Austauschformat.
- Mittels XML können System- und Herstellerunabhängigkeit zu unterschiedlichsten Plattformen und Anwendungen erreicht werden.
- XML ist einfach erlern- und anwendbar.
- XML bietet die Möglichkeit der automatischen Weiterverarbeitung.
- XML ermöglicht Cross Media Publishing (eine Quelle für mehrere Medien).
- XML erlaubt Single-source Publishing (eine Quelle und mehrere Varianten für die Veröffentlichung).
- XML liefert flexible Präsentationsmöglichkeiten der Daten durch Trennung von Inhalt, Struktur und Layout.
- Die Wartung und Pflege der Daten ist relativ einfach.
- Mit XML können verschiedenste Sichten auf Daten genommen werden.
- Durch standardisierte XML-Vokabulare ist die Datencodierung weltweit einheitlich.
- XML bietet die Möglichkeit einer strukturellen Validierung.

[XML02]

5.2. Strukturbeschreibungen

Die Extended Markup Language hat einen wichtigen Beitrag für die Vereinfachung und Verbesserung des Datenaustausches zwischen verschiedensten Systemen geleistet. Es darf jedoch nicht übersehen werden, dass XML selbst eigentlich nur ein „Metastandard“ ist. Für den konkreten Datenaustausch via XML sind im Normalfall Strukturvorschriften notwendig, in welchen die zulässigen Elemente und Attribute sowie hierarchische und quantitative Beziehungen festgelegt werden können. Die Überprüfung, ob ein Dokument die definierten Regeln einhält, kann von einem validierenden Parser vorgenommen werden. Ist das überprüfte Dokument also wohlgeformt und hält die deklarierten Strukturvorschriften ein, so

wird es als „gültig“ bezeichnet. Das XML-Dokument gehört somit zu der durch die Strukturvorschriften definierten Dokumentenklasse.

Strukturvorschriften können entweder in „Document Type Definitions“ (DTD) oder „XML Schema Definitions“ (XSD) erstellt werden, wobei die Verwendung letzterer vom W3C empfohlen wird [MSD].

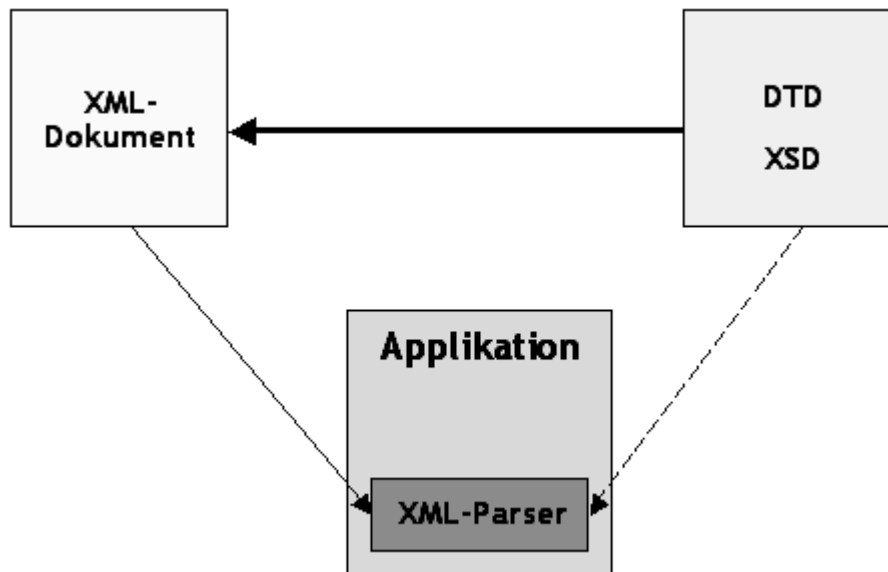


Abbildung 4: Struktur- und Inhaltsbeschreibung

Beide Arten bieten die Möglichkeit, eine Grammatik für XML-Dokumente festzulegen, die eingehalten werden muss, damit das erstellte XML-Dokument gültig ist und damit zur entsprechenden Dokumentenklasse gehört. Sämtliche Personen, die derlei XML-Dokumente modifizieren, haben also darauf zu achten, dass sie das in der Strukturbeschreibung definierte Vokabular verwenden und die darin aufgestellten Vorschriften strikt einhalten. Andernfalls würde das Dokument „ungültig“ werden.

Formal setzt sich eine Strukturbeschreibung aus Deklarationen der im Dokument verwendeten Auszeichnungen (Markup-Deklarationen) zusammen, nämlich aus

- Elementtypdeklarationen,
- Attributlistendeklarationen,
- Entity-Deklarationen und
- Notationsdeklarationen.

Damit Systeme ohne manuelle Eingriffe fehlerlos miteinander kommunizieren können, reicht es jedoch nicht aus, dass sie dieselben Strukturvorschriften benutzen. Denn auch gleiche Datenstrukturen und einheitliche Namen für die Datenfelder verhindern nicht, dass die einzelnen Werte der Datenfelder möglicher Weise unterschiedlich interpretiert werden. Es ist vielmehr ein gemeinsames Verständnis über die Semantik der Ausprägungen einzelner Datenfelder erforderlich, um eine Sprachverwirrung der in Kommunikation stehenden Systeme zu vermeiden. Aus diesem Grund verweisen XML-basierte Austauschformate oftmals auf externe Standards oder Nummernsysteme, die für spezielle Anwendungsdomänen gültig sind und eine eindeutige Zuordnungen von Werten und Bedeutungen bereitstellen.

Für den Bereich „eLearning“ entwickelt und fördert das „Global Learning Consortium“ des IMS (Instructional Management System) eine solche standardisierte XML-basierte Spezifikation („Content Packaging Specification“). Folgende beiden Hauptziele werden von diesem Konsortium verfolgt:

- Technische Spezifikationen sollen definiert werden, um so Interoperabilität zwischen eLearning-Anwendungen und -diensten unterschiedlicher Anbieter zu ermöglichen.
- Die Einbindung der vom IMS verabschiedeten Spezifikationen in Produkte und Dienstleistungen, d.h. die Arbeit der Entwickler von eLearning-Umgebungen, soll weltweit bestmöglich unterstützt werden.

[MOB]

5.3. Content Packaging Specification (CPS)

Damit eLearning-Kurse nicht nur auf einer einzigen Plattform einsetzbar sind, sondern nahezu beliebig („stranded investments“), wird bei deren Erstellung ein international gebräuchlicher De-facto-Standard verwendet. Dieser Standard des „IMS Global Learning Consortiums“ (IMS) – genannt „Content Packaging Specification“ (CPS) – legt exakt die Struktur und den Aufbau einzelner Kurse beziehungsweise ganzer Kurssammlungen fest [IMS].

Wie bereits erwähnt, basiert XML auf dem Konzept der Trennung des Informationsgehalts eines Dokuments von seiner Präsentation. Durch diese Trennung von Inhalt, Struktur und Layout ergibt sich die Möglichkeit, Daten für verschiedene Sichten und Systeme

aufzubereiten („Cross Media Publishing“), da die Datenverarbeitung den jeweiligen Anwendungsprogrammen selbst überlassen bleibt. Da es sich bei XML bekanntlich um eine standardisierte, von Computern und Menschen lesbare Auszeichnungssprache handelt, welche das Definieren beliebiger Elemente und deren Beziehungen zueinander sowie deren automatische Verarbeitung ermöglicht, wurde es als Format für das „Content Packaging“-Datenmodell auserkoren, welches den strukturellen Aufbau von eLearning-Kursen definiert.

Bei einem CPS-Paket („Package“) handelt es sich um ein Verzeichnis, welches neben einer speziell benannten XML-Datei (imsmanifest.xml) auch etwaige XML-Kontrolldokumente (DTD, XSD) sowie eine Menge an Unterverzeichnissen enthält, in welchen sich die Ressourcen des Kurses befinden.

Grob betrachtet besteht ein CPS-Paket – wie aus Abbildung 5 [IMSa] ersichtlich – aus zwei Teilen: Einerseits dem sogenannten „Manifest“, bei welchem es sich um eine XML-Datei handelt, welche den Kurs samt seiner Struktur und dessen verwendeten Ressourcen beschreibt. Und andererseits dem „Content“, also allen Dateien, die den gesamten Kurs im Eigentlichen ausmachen (.html, .jpg, .pdf, ...). Diese Dateien können in beliebige hierarchische Verzeichnisstrukturen gegliedert werden.

Wird ein solches Paket in eine einzige Datei gepackt (z.B.: .zip) – was jedoch nicht zwingend erforderlich ist – so wird diese alles beinhaltende Datei als „Package Interchange File“ (PIF) bezeichnet.

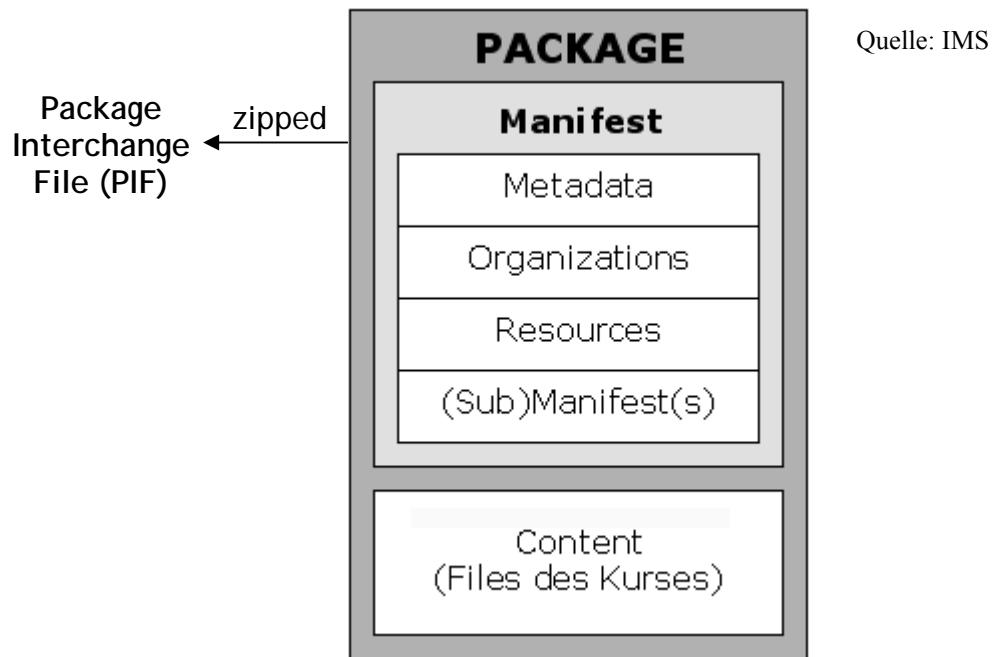
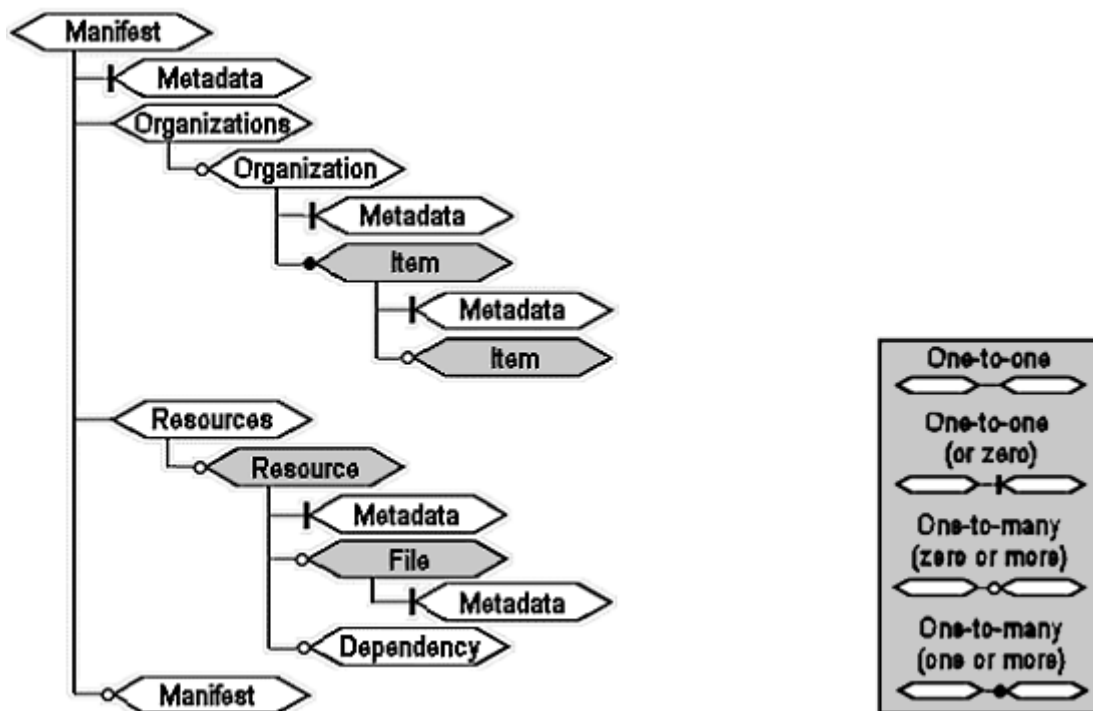


Abbildung 5: Package Interchange File

Abbildung 5 lässt bereits den hierarchischen Aufbau des Content Packaging-Datenmodells erahnen, da manche der darin angeführten Elemente wiederum verschiedenste Subelemente enthalten können. Im nächsten Kapitel wird auf diese Hierarchie näher eingegangen, wobei das Hauptaugenmerk auf dem so genannten „Manifest“ und dessen einzelnen Bestandteilen liegt.

6. CPS Manifest

Ein CPS Manifest legt die Struktur eines elektronischen Kurses und dessen verwendete Ressourcen (Content) fest (siehe Abbildung 6). Dies wird durch Verwendung einer Reihe von XML-Elementen erreicht, deren wichtigste Vertreter im Folgenden angeführt werden.



Quelle: IMS

Abbildung 6: Elemente eines Manifests

Da es sich beim CPS Manifest um eine XML-Datei handelt, haben wir es bei allen Komponenten des Manifests mit XML-Elementen zu tun.

6.1. <manifest>

Der <manifest>-Knoten ist der Startknoten eines CPS Manifests und enthält grundsätzlich drei Hauptkomponenten: <metadata>, <organizations> und <resources>. Darüber hinaus können noch beliebig viele Submanifeste in einem Manifest geschachtelt sein (siehe Abbildung 7).



Abbildung 7: <manifest>-Element

6.2. <metadata>

Metadaten sind nach ISO 11179 Daten, welche andere Daten beschreiben („Daten über Daten“) [ISO79]. Darüber hinaus sind sie ein geeignetes Mittel, um Daten zu archivieren und in weiterer Folge wieder auffindbar zu machen.

Das <metadata>-Element (siehe Abbildung 8) ermöglicht also eine Beschreibung des Manifests. Dabei können unter anderem Titel, Kurzbeschreibung, Schlüsselwörter, pädagogisches Ziel und Zielgruppe des Kurses angegeben werden.

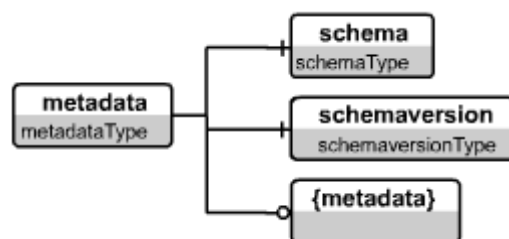


Abbildung 8: <metadata>-Element

Es ist natürlich auch möglich, eigene Elemente einzuführen und zu benutzen. Diese sind dazu in einer DTD- oder XSD-Datei zu definieren. Grundsätzlich müssen Metadaten folgende Anforderungen erfüllen:

- Sie sind für Maschinen les- und interpretierbar.
- Sie eignen sich für den Austausch in heterogenen Netzwerken.
- Sie sind in verschiedenen Anwendungsbereichen (domänenneutral) nutzbar.

- Sie sind von verschiedensten Applikationen nutzbar.
- Bestehende Klassifizierungsschemata sind erweiterbar.

[IMSb]

Das Österreichische Bundesministerium für Bildung, Wissenschaft und Kultur (BMBWK) spezifizierte beispielsweise folgende zwei Elemente, die eine Beschreibung des Manifests mit Blickrichtung auf pädagogische Kriterien erlauben:

- <curriculum>
- <certifications>

Die wichtigsten Vertreter zur Standardisierung der Beschreibung von Lernobjekten sind:

- IMS
- LOM
- AICC
- SCORM

Nähere Informationen zu diesen sind in [LOI03] bzw. [SZÜ03] zu finden.

Die am Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM) der Universität Linz erstellten Kurse werden mit Metadaten, die dem „Learning Object Metadata“-Standard (LOM) entsprechen, versehen [LOM]. Dieser Standard wurde vom LTSC-Gremium der IEEE entwickelt und resultiert u.a. aus Vorarbeiten des IMS- und des ARIADNE-Projektes (EU-Projekt zur Metadatenerfassung) [SCM01].

Der Zweck von LOM wird in [LTS02] wie folgt beschrieben: "The purpose of this multi-part Standard is to facilitate search, evaluation, acquisition, and use of learning objects, for instance by learners or instructors or automated software processes." Ein weiteres Ziel ist die gemeinsame Nutzung und der Austausch von Lernobjekten. Die Lernobjekte können dabei mit vielfältigen Metadaten versehen werden. Die Mehrsprachigkeit wird in diesem Zusammenhang besonders betont.

Das „LOM v1.0 Base Schema“ sieht folgendermaßen aus:

- Jedes Objekt besitzt ein Wurzelement: <lom>

- Es folgen 9 weitere Elemente, unter denen sich die Informationen des Objekts kategorisieren lassen:
 - general: Grundlegende Informationen, die das Lernobjekt beschreiben.
 - lifecycle: Merkmale über die Geschichte und den aktuellen Zustand des Lernobjekts
 - meta-metadata: Informationen über die Metadaten-Instanz an sich
 - technical: Technische Voraussetzungen des Lernobjekts
 - educational: Pädagogische Merkmale des Lernobjekts
 - rights: Urheberrechtliche Informationen des Lernobjekts
 - relation: Beziehungen zwischen dem Lernobjekt und anderen verwandten Lernobjekten
 - annotation: Anmerkungen zum Lernobjekt
 - classification: Einordnung des Lernobjekts in ein Klassifizierungssystem
- Jedes dieser Elemente unterteilt sich in weitere Abschnitte bis hin zu den eigentlichen Metadaten.

[LTS]

6.3. <organizations>

Das CPS-Element <organizations> (siehe Abbildung 9) ermöglicht die Festlegung beliebiger Sichten auf das gesamte Kurspaket. Jede Sicht wird durch einen <organization>-Knoten repräsentiert und kann eine unterschiedliche Kursstruktur mit unterschiedlichen Ressourcen definieren.

Werden mehrere <organization>-Elemente angegeben, so muss eines davon mit Hilfe des „default“-Attributs als Standard deklariert werden. Wie die Auswahl der darzustellenden Sicht in einer entsprechenden Applikation erfolgt bzw. ob generell die Standard-Variante gewählt wird, bleibt dem Entwickler überlassen.

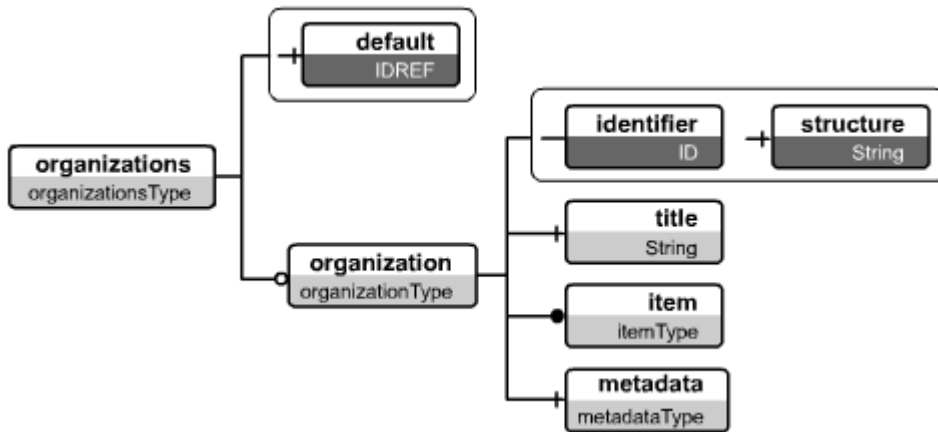


Abbildung 9: <organizations>-Element

Die eigentliche Struktur eines Kurses wird durch die entsprechende Anordnung der <item>-Knoten festgelegt, wobei diese eine Schachtelung in beliebiger Tiefe erlauben (siehe Abbildung 10).

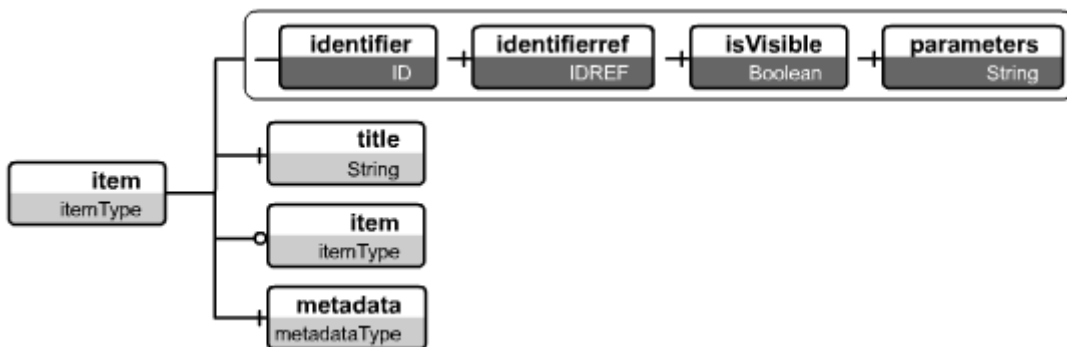


Abbildung 10: <item>-Element

Zusätzlich können sowohl <organization>- als auch <item>-Elemente mittels Metadaten beschrieben werden.

6.4. <resources>

Dieses CPS-Element besteht aus beliebig vielen <resource>-Knoten, die sämtliche für einen Kurs benötigten Ressourcen enthalten (siehe Abbildung 11). Eine Ressource kann grundsätzlich eine beliebige Menge von Dateien umfassen, wobei für jede einzelne Datei ein <file>-Tag verwendet wird, in welchem die Referenz auf diese Datei zu finden ist.

Zusätzlich kann das <resource>-Element wiederum unbeschränkt viele andere <resource>-Elemente einschließen, wobei diese über den Knoten <dependency> referenziert werden. Auch hier ist anzumerken, dass <resource>- und <file>-Knoten zur näheren Beschreibung mit Metadaten versehen werden können.

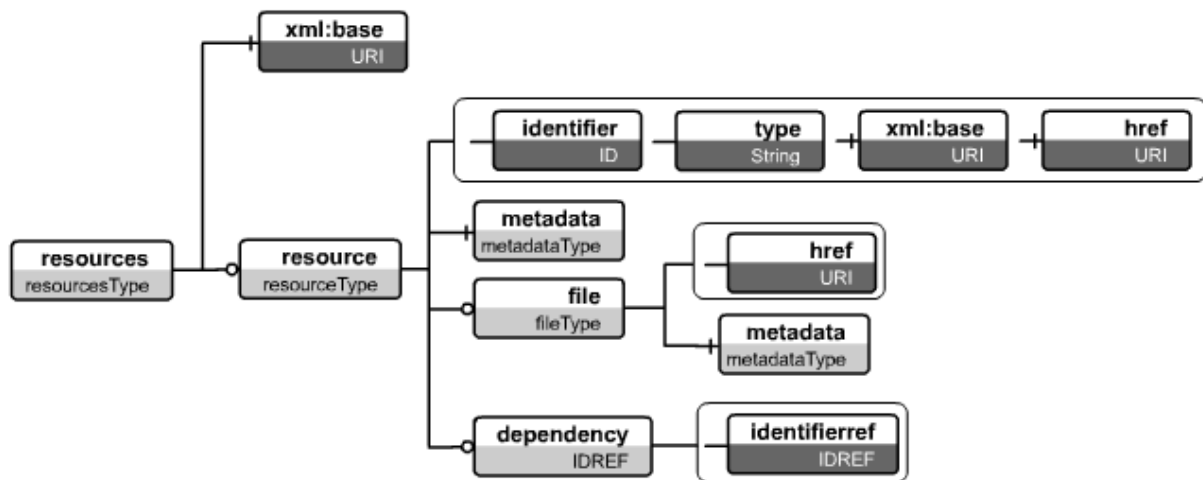


Abbildung 11: <resources>-Element

6.5. Sub-<manifest>

Dieses grundsätzlich bereits bekannte CPS-Element definiert in diesem speziellen Fall ein Submanifest, das in das Hauptmanifest eingebettet wird (siehe Abbildung 12). Da es sich bei einem Submanifest vom Typ her ebenfalls um ein Manifest handelt, hat dieses natürlich die gleiche Struktur wie das alles umfassende Manifest.

Verwendung finden Submanifeste beispielsweise dann, wenn ganze Kurse zu einem Manifest hinzugefügt werden sollen, wobei festzuhalten ist, dass jedes Manifest beliebig viele Submanifeste enthalten kann.



Abbildung 12: Submanifest(e)

Weiterführende Informationen zu den verschiedenen Strukturelementen des IMS Content Packaging Information Models finden Sie unter [IMSa].

7. Praktischer Teil

Nachdem der theoretische Hintergrund behandelt wurde, steht in diesem Abschnitt der praktische Teil meiner Diplomarbeit im Blickpunkt.

Im Wesentlichen besteht das Gesamtsystem aus zwei Komponenten: Dem „Manifest Converter“ und dem „Manifest Viewer“. Der Manifest Converter wird vom Kursanbieter dazu verwendet, CPS-konforme Kurse derart aufzubereiten, um sie in geeigneter Form auf Handhelds verfügbar machen zu können. Die Lernenden andererseits verwenden den Manifest Viewer, um die Inhalte dieser entsprechend modifizierten Kurse auf dem PDA anzeigen zu können.

Abbildung 13 stellt den grundlegenden Ablauf und die Interaktion zwischen den beteiligten Komponenten dar:

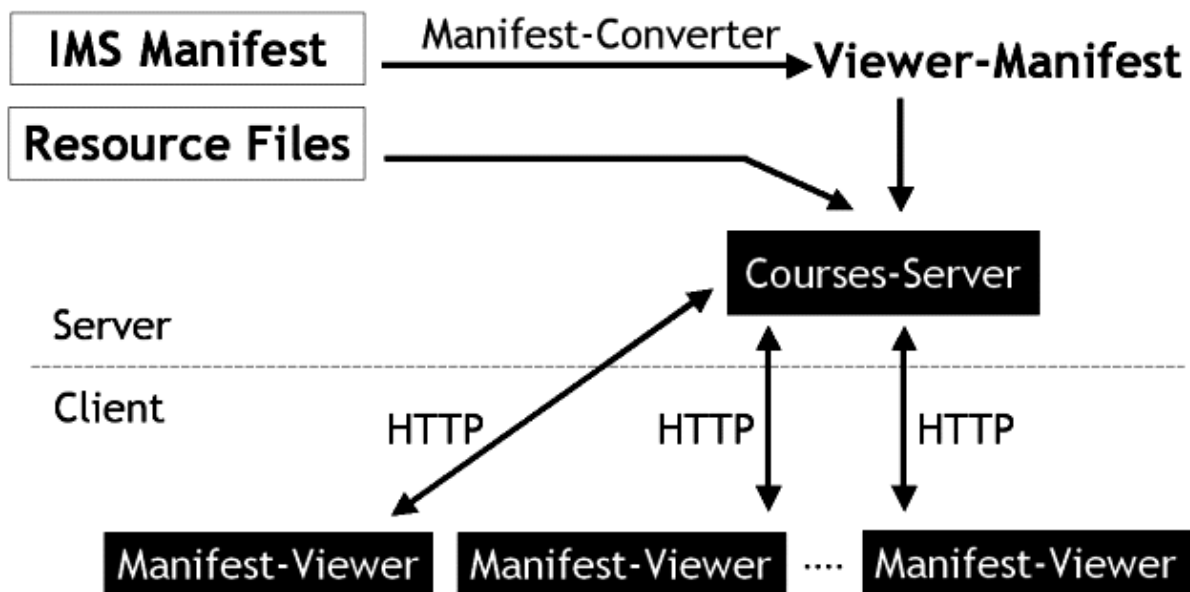


Abbildung 13: Komponenten und Interaktion

Mittels des Manifest Converters werden wie gesagt bestehende CPS-Kursmanifeste speziell für die Anzeige auf einem Handheld adaptiert. Diese modifizierten Manifeste, die im XML-Format vorliegen und neben der Kursstruktur auch Metainformationen über den jeweiligen Kurs enthalten, werden anschließend zusammen mit den für die Kurse benötigten Materialien („Content“) auf einem Webserver veröffentlicht. D.h. der Anbieter eines eLearning-Kurses kopiert all diese benötigten Dateien auf einen über das Internet erreichbaren Server. Der Manifest Viewer gestattet es nun den Benutzern, die publizierten Manifeste und deren

Content herunter zu laden und schließlich auch anzusehen. Die Kommunikation zwischen Viewer und Server erfolgt dabei über das http-Protokoll.

Da neben dem Konvertieren eines CPS-Manifests auch noch weitere Tätigkeiten auf Seiten der Kursanbieter anfallen können, wird der Manifest Converter in ein übergeordnetes Tool eingegliedert, welches diese benötigten Funktionalitäten bereitstellt. Diese Meta-Applikation nennt sich „Manifest Administrator“ und wird im folgenden Kapitel behandelt. Anschließend wird der Fokus auf den Manifest Viewer gelegt, der bezüglich Komplexität und zeitlichem Aufwand den weitaus größten Teil dieser Diplomarbeit ausmacht.

7.1. Entwicklungsumgebung

Erstellt wurde der Manifest Viewer mit Microsoft Visual Studio .NET 2003 in der Programmiersprache C#. Microsoft stellt speziell für den Sektor der mobilen Geräte das sogenannte „.NET Compact Framework“ zur Verfügung, bei dem es sich um eine „abgespeckte“ Version des .NET Frameworks handelt, was sich durch eine deutlich reduzierte Anzahl an zur Verfügung stehenden Klassen und Methoden äußert. Dieses kompakte Framework ist integraler Bestandteil von Visual Studio .NET ab der Version 2003 („Smart Device Extensions“) und ist speziell auf die Entwicklung von Anwendungen für mobile Geräte ausgerichtet. Zusätzlich erlaubt es das Emulieren mobiler Geräte am Computer, was die Entwicklung von mobilen .NET-Anwendungen deutlich erleichtert und fördert.

Ergänzend wurde das OpenNETCF Smart Device Framework verwendet, welches eine Menge an nützlichen Bibliotheken zur Implementierung mobiler Anwendungen bereitstellt. Es handelt sich dabei um ein Open Source-Projekt, das unter Entwicklern von Smart Device-Applikationen mittlerweile große Verbreitung erlangte.

7.2. Manifest Administrator

Wie bereits erwähnt, dient der Manifest-Administrator dazu, Modifikationen am CPS-Manifest anzubringen, die für ein vernünftiges Arbeiten mit eLearning-Kursen auf einem PDA nützlich bzw. notwendig sind.



Abbildung 14: Manifest Administrator

Im Speziellen sind dazu zwei Bausteine des Manifest-Administrators zuständig: Der Manifest Converter und der Manifest Updater (siehe Abbildung 15).

Manifest-Administrator



Abbildung 15: Komponenten des Manifest Administrators

Die wichtigste Funktion kommt zweifelsohne dem Manifest Converter zu. Dieser erweitert nämlich das bestehende Manifest um Informationen, die vom Manifest Viewer benötigt werden, um den speziellen Anforderungen eines Handhelds gerecht werden zu können. In erster Linie heißt das, ein effizientes und benutzerfreundliches Arbeiten mit eLearning-Kursen zu gewährleisten – trotz der bereits bekannten Einschränkungen mobiler Geräte. Auf die nähere Bedeutung von „effizient“ in diesem Zusammenhang wird weiter unten noch eingegangen.

Die zweite Komponente des Manifest Administrators ist der sogenannte „Manifest Updater“. Dessen Aufgabe besteht darin, einen bereits konvertierten Kurs zu aktualisieren, d.h. für alle Dateien des Contents zu überprüfen, ob diese eventuell in neueren Versionen vorliegen. Werden aktuellere Files vorgefunden, sind diese in den publizierten eLearning-Kurs einzubinden.

Die grafische Benutzeroberfläche des Manifest Administrators hat das in Abbildung 16 dargestellte Aussehen:

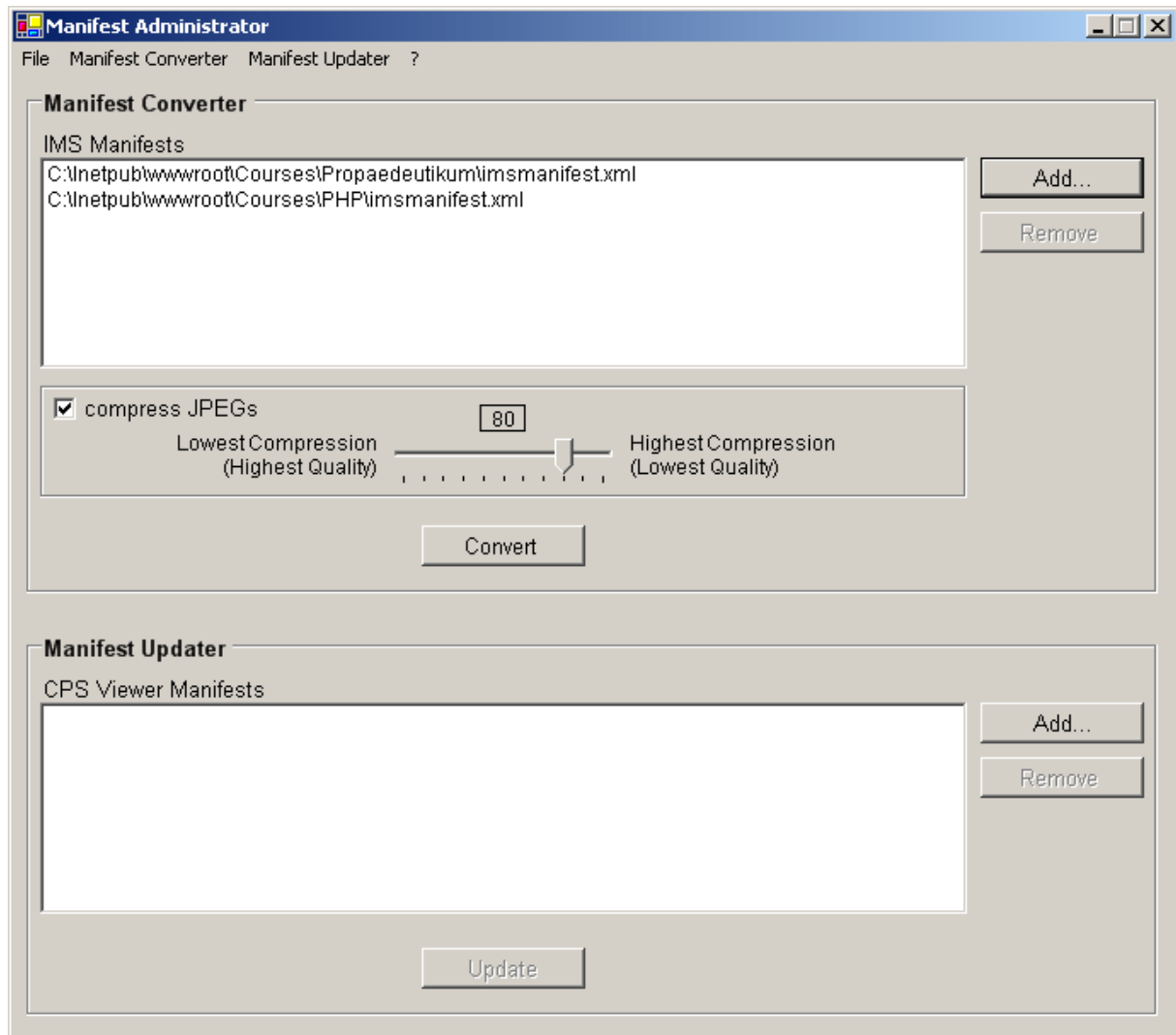


Abbildung 16: Hauptfenster des Manifest Administrators

Aus Abbildung 16 ist ersichtlich, dass sich die beiden wesentlichen Komponenten des Manifest Administrators das Hauptfenster teilen. In der oberen Hälfte befinden sich die Steuerelemente des Manifest Converters, unten jene des Manifest Updaters. Beide werden im den folgenden Abschnitten genau beschrieben.

7.2.1. Manifest Converter

Der Manifest Converter generiert aus einem bestehenden CPS-Manifest eine modifizierte Manifest-Datei, die der Applikation auf dem Handheld („Manifest Viewer“) alle notwendigen Informationen bezüglich eines elektronischen Kurses zur Verfügung stellt. Dazu wird das Original-Manifest („imsmanifest.xml“) in erster Linie um bestimmte – unten näher erläuterte

– Einträge erweitert. Das adaptierte CPS Viewer Manifest („cps_manifest.xml“) ähnelt folglich dem ursprünglichen Manifest sehr stark.



Abbildung 17: Manifestkonvertierung

Der Grund für die Notwendigkeit von Erweiterungen wird anhand des folgenden Beispiels erläutert:

Wird in einen Kurs – d.h. in ein Manifest – ein HTML-Dokument eingebunden, so genügt es üblicherweise, als Referenz im Manifest den Dateinamen der HTML-Seite anzugeben (siehe Abbildung 18).

```
</resource>  
- <resource identifier="MANIFEST01_RESOURCE14" type="webcontent" href="inhalt_information5.2_restau.htm" />  
</resource>  
- <resource identifier="MANIFEST01_RESOURCE15" type="webcontent" href="inhalt_information5.2_restau.htm" />  
</resource>
```

Abbildung 18: Referenz auf HTML-Dokument

Sind auf dieser Seite – wie in Abbildung 19 – andere Dateien (z.B. Bilder, Audio-/Videodateien, etc.) eingebunden, so werden diese (meist) problemlos beim Aufruf der HTML-Seite angezeigt, da man davon ausgehen kann, dass der Benutzer immer auf die gesamten Kursdateien zugreifen kann (entweder auf CD oder online).

```
</p>  
<p class="MsoNormal" align="left">  
  
  
</p>  
<p class="MsoNormal" align="left">  
<img alt="Century Gothic" /><br>
```

Abbildung 19: Referenzen innerhalb eines HTML-Dokuments

Ist ein eLearning-Kurs allerdings auf einem PDA gespeichert, so ist es sehr wahrscheinlich, dass sich nicht der gesamte Kurs, sondern nur Teile davon auf dem mobilen Gerät befinden. Kurse werden schließlich oftmals allein schon aus Speichermangel nicht vollständig auf das Handheld geladen. Angelehnt an das Beispiel aus Abbildung 18 entscheidet sich der Benutzer, lediglich ein Bruchstück des Gesamtkurses – in diesem Fall also die Ressource „MANIFEST01_RESOURCE14“ – auf sein mobiles Gerät zu laden. Welche Situation ergäbe sich, wenn anstatt des mit dem Manifest Converter modifizierten Manifests das Original verwendet würde? Nun, der Manifest Viewer würde die angegebene HTML-Datei herunterladen, jedoch nicht die restlichen Grafiken, die für die vollständige Anzeige der HTML-Seite benötigt werden. Der Manifest Viewer hat schließlich auch keinerlei Informationen, welche bzw. ob überhaupt noch andere Dateien notwendig sind, da im originalen Manifest nur der Filename der HTML-Datei selbst eingebunden ist. Beim Öffnen des HTML-Dokuments würden dann an allen Positionen, an denen sich eigentlich Grafiken befinden sollten, „Platzhalter“ (siehe Abbildung 20) erscheinen, die lediglich das Fehlen von Grafikelementen andeuten.



Abbildung 20: Platzhalter für nicht verfügbare Bilddatei

Es ist daher zwingend erforderlich, dass beim Herunterladen einer Ressource auch sämtliche darin referenzierte Dateien mitgeladen werden. Und dies ist eben nur dann möglich, wenn im Manifest zu jeder Ressource all ihre benötigten Dateien explizit angeführt sind (ausgenommen externe Dateien im WWW). Abbildung 21 zeigt einen entsprechenden Auszug aus einem modifizierten Manifest, bei dem alle notwendigen Dateien der im Beispiel verwendeten Ressource in entsprechenden <file>-Tags hinzugefügt wurden.

```
</resource>
- <resource identifier="MANIFEST01_RESOURCE14" type="web"
  <file href="Html/inhalt_information5.2_restau.htm"
  <file href="Html/pics/speisekarte_englisch.gif"
  <file href="Html/pics/speisekarte_japanisch.gif"
  <file href="Html/pics/prev_blue.gif"
</resource>
- <resource identifier="MANIFEST01_RESOURCE15" type="web"
  <file href="Html/inhalt_information5.2_restau.htm"
  <file href="Html/pics/speisekarte_englisch.gif"
  <file href="Html/pics/speisekarte_japanisch.gif"
  <file href="Html/pics/prev_blue.gif"
</resource>
```

Abbildung 21: Vollständige Auflistung referenzierter Dateien

Aber nicht nur aus der Absicht heraus, das Laden von einzelnen Teilen von Kursen auf das Handheld zu ermöglichen, sind Modifikationen am Original-Manifest nötig. Der Manifest Viewer sollte dem Benutzer auch Auskunft darüber erteilen können, wie viel Speicherplatz bestimmte Kurse bzw. einzelne Kursmaterialien in Beschlag nehmen. Schließlich ist es bisweilen nicht außergewöhnlich, dass der freie Speicher auf einem Handheld äußerst knapp bemessen ist. Der Benutzer muss nun unter Umständen abklären, ob die verfügbare Speicherkapazität noch ausreicht, um die gewünschten Kursteile auf das mobile Gerät laden zu können. Andernfalls bleibt nichts anderes übrig, als entweder vom Download Abstand zu nehmen oder für ausreichend freien Speicher zu sorgen. Letzteres kann durchaus zu einem Löschen von anderen, weniger relevanten Kursmaterialien führen. Damit aber ein zielgerichtetes Auswählen der zu entfernenden Ressourcen möglich ist, sind wiederum Speicherbedarfsinformationen zu diesen Ressourcen notwendig, um die Größe des frei werdenden Speicherplatzes abschätzen zu können.

Unter dem Strich bleibt also die Erkenntnis, dass im modifizierten Manifest neben den benötigten Files auch noch deren jeweiliger Speicherbedarf anzugeben ist. Wie aus Abbildung 22 ersichtlich wird dazu ein neues Attribut („cps_hq“) angelegt, in dem als erster Wert (links vom Komma) die Größe der jeweiligen Datei in Bytes angegeben wird.


```
</resource>
- <resource identifier="MANIFEST01_RESOURCE14" type="webcontent" href="Html/inhalt_informations5.2_restau.htm" cps_hq="2132,6321496992761" />
  <file href="Html/inhalt_informations5.2_restau.htm" cps_hq="2132,6321496992761" />
  <file href="Html/pics/speisekarte_englisch.gif" cps_hq="16444,6321496993207" />
  <file href="Html/pics/speisekarte_japanisch.gif" cps_hq="18482,6321496993512" />
  <file href="Html/pics/prev_blue.gif" cps_hq="161,6321496993516" />
</resource>
- <resource identifier="MANIFEST01_RESOURCE15" type="webcontent" href="Html/inhalt_informations5.2_restau.htm" cps_hq="2132,6321496992761" />
  <file href="Html/inhalt_informations5.2_restau.htm" cps_hq="2132,6321496992761" />
  <file href="Html/pics/speisekarte_englisch.gif" cps_hq="16444,6321496993207" />
  <file href="Html/pics/speisekarte_japanisch.gif" cps_hq="18482,6321496993512" />
  <file href="Html/pics/prev_blue.gif" cps_hq="161,6321496993516" />
  <file href="Html/pics/next_blue.gif" cps_hq="161,6321496993501" />
</resource>
```

Abbildung 22: Zusätzliches File-Attribut - Dateigröße

Der zweite Parameter (rechts vom Komma) dient der Versionierung, welche ein weiterer wichtiger Aspekt für das Modifizieren eines CPS-Manifests ist. Der Viewer muss nämlich erkennen, ob ein Kurs bzw. einzelne Kursmaterialien geändert wurden, d.h. ob auf dem Server eine neuere Kursversion vorhanden ist. Aus diesem Grund sind im Manifest auch entsprechende Informationen bezüglich des Änderungsdatums der einzelnen Dateien mitzuführen, was nun eben über den zweiten Wert im hinzugefügten und zuvor bereits angesprochenen Attribut „cps_hq“ geschieht (siehe Abbildung 23). Konkret steht dieser zweite Wert für die Zahl der Millisekunden, die dem Änderungsdatum des jeweiligen Files entsprechen.

```
</resource>
<resource identifier="MANIFEST01_RESOURCE20" type="webcontent" href="Html/inhalt_informations8_plausi.htm" cps_hq="5960,6321496992761" />
  <file href="Html/inhalt_informations8_plausi.htm" cps_hq="5960,6321496992761" />
  <file href="Html/pics/child.gif" cps_hq="2505,6321496993207" />
  <file href="Html/pics/pendler.gif" cps_hq="38580,6321496993512" />
  <file href="Html/pics/sign_info.gif" cps_hq="1133,6321496993591" />
  <file href="Html/pics/sign_fragezeichen.jpg" cps_hq="7127,6321496993591" />
  <file href="Html/pics/prev_blue.gif" cps_hq="161,6321496993516" />
  <file href="Html/pics/next_blue.gif" cps_hq="161,6321496993501" />
</resource>
<resource identifier="MANIFEST01_RESOURCE21" type="webcontent" href="Html/inhalt_informations8_plausi.htm" cps_hq="5960,6321496992761" />
  <file href="Html/inhalt_informations8_plausi.htm" cps_hq="5960,6321496992761" />
  <file href="Html/pics/child.gif" cps_hq="2505,6321496993207" />
  <file href="Html/pics/pendler.gif" cps_hq="38580,6321496993512" />
  <file href="Html/pics/sign_info.gif" cps_hq="1133,6321496993591" />
  <file href="Html/pics/sign_fragezeichen.jpg" cps_hq="7127,6321496993591" />
  <file href="Html/pics/prev_blue.gif" cps_hq="161,6321496993516" />
  <file href="Html/pics/next_blue.gif" cps_hq="161,6321496993501" />
</resource>
```

Abbildung 23: Zusätzliches File-Attribut - Änderungsdatum

Eine weitere Aufgabe des Manifest-Konverters liegt in der Umstrukturierung jener <item>-Knoten, die in ihrer ursprünglichen Form für den Manifest Viewer ungeeignet sind. Ungeeignet deshalb, da der im Zuge dieser Diplomarbeit entwickelte Viewer eine strenge Trennung zwischen Verzeichnissen (Kapiteln) und Dokumenten vornimmt. Während das Tippen auf einen Verzeichnisknoten entweder dessen Auf- bzw. Zuklappen bewirkt, führt das

Antippen eines Dokumentknotens zum Öffnen des entsprechenden Dokuments – sofern dieses am Handheld verfügbar ist (siehe Abbildung 24).

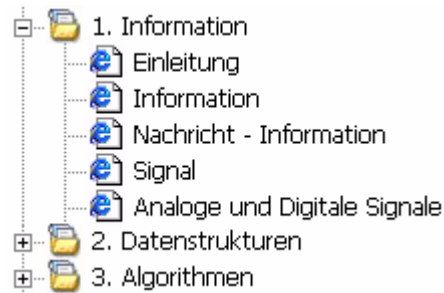


Abbildung 24: Kapitel- und Dokumentstruktur im Manifest Viewer

Im originalen Manifest ist es jedoch ohne weiteres möglich, einem Verzeichnis-Item, welches wiederum Subitems enthält, eine Referenz auf eine Ressource hinzuzufügen, wie in Abbildung 25 ersichtlich ist. Der Ordner „Uebungen“ hat zumindest einen Unterordner (hier: „Aufgaben“) und zusätzlich eine Referenz auf die Ressource „RESOURCE154“.

```

<title>Text Eng...</title>
</item>
<item identifier="MANIFEST01_ITEM79" isvisible="1" parameters="" identifierref="RESOURCE154">
  <title>Uebungen</title>
  <item identifier="MANIFEST01_ITEM115" isvisible="1" parameters="" identifierref="RESOURCE113">
    <title>Aufgaben</title>
  </item>
</item>

```

Abbildung 25: Vater-Item mit Referenz auf Ressource

Um den Programmablauf auf Seiten des PDA bei Erstellung der Baumstruktur mit Ordnern und Blättern nicht unnötig durch zusätzliche Algorithmen zu bremsen, erfolgt die notwendige Umstrukturierung ebenfalls bei der Konvertierung des originalen Manifests. In Abbildung 26 ist zu sehen, dass ein neues Item eingefügt und mit der Referenz, die ursprünglich zum übergeordneten Item gehörte (vgl. Abbildung 25), ausgestattet wurde.

```

<title>Text Eng...</title>
</item>
<item identifier="MANIFEST01_ITEM79">
  <title>Uebungen</title>
  <item identifier="MANIFEST01_ITEM79_GPS" identifierref="RESOURCE154" />
  <item identifier="MANIFEST01_ITEM115" identifierref="RESOURCE113">
    <title>Aufgaben</title>
  </item>
</item>

```

Abbildung 26: Modifikation der Item-Struktur

Dem aufmerksamen Beobachter wird wahrscheinlich auch noch etwas anderes aufgefallen sein: Manche Attribute der <item>-Elemente wurden entfernt. Dies ist zulässig, da es sich in diesem Fall um Attribute handelte, die entweder keinen (parameters="") bzw. einen Standardwert (isvisible="1") zugewiesen hatten, der für die Darstellung im Manifest Viewer keinerlei Relevanz besitzt (siehe Abbildung 27).

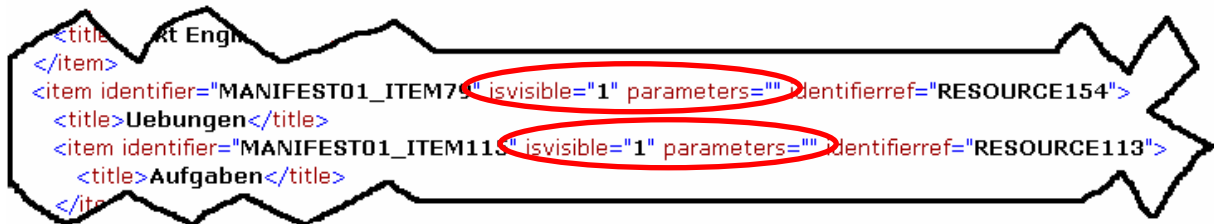


Abbildung 27: Irrelevante Attribute

Zusätzlich werden im Zuge der Konvertierung auch sämtliche <resource>-Knoten beseitigt, die vom Viewer sowieso nie benötigt werden. Bei diesen Knoten handelt es sich nämlich um Resources, die von keinem <item> bzw. keiner anderen <resource> (als Dependency) referenziert werden. In Abbildung 28 ist ein solcher Fall einer Resource-„Leiche“ skizziert, da die „RESOURCE3“ von keinem einzigen in Frage kommenden Knoten referenziert wird.

```

<?xml version="1.0" ?>
- <manifest identifier="MANIFEST1" xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
  xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_v1p2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1.xsd
  http://www.imsglobal.org/xsd/imsmd_v1p2 imsmd_v1p2p2.xsd">
- <organizations default="TOC1">
  - <organization identifier="TOC1" structure="hierarchical">
    <title>Test-Kurs</title>
    - <item identifier="ITEM1">
      <title>Kapitel 1</title>
      - <item identifier="ITEM2" identifierref="RESOURCE1"> ←
        <title>Kapitel 1.1</title>
        </item>
      - <item identifier="ITEM3" identifierref="RESOURCE2"> ←
        <title>Kapitel 1.2</title>
        </item>
      </organization>
    </organizations>
  - <resources>
    <resource identifier="RESOURCE1" type="webcontent" href="docs/page1.htm" />
    <resource identifier="RESOURCE2" type="webcontent" href="docs/page2.htm" />
    <resource identifier="RESOURCE3" type="webcontent" href="docs/page3.htm" />
  </resources>
</manifest>

```

Abbildung 28: Unreferenzierte Resource

Klarerweise würde der Manifest Viewer das Manifest auch verarbeiten können, wenn all diese irrelevanten Attribute und Resources erhalten blieben. Da diese im schlimmsten Fall ein Manifest jedoch gehörig aufblähen können, werden sie vom Manifest Converter eliminiert und finden somit im modifizierten Manifest keine Berücksichtigung mehr.

Als letzte Aktion erzeugt der Konverter noch eine kleine Textdatei („cps_manifest.dat“), die lediglich eine einzige Zeile enthält:

z.B.: 632199157793700000

Diese Zahl stellt den Erstellungszeitpunkt des neuen Manifests (wieder in Millisekunden) dar. Lädt der Viewer ein Manifest, so bekommt er automatisch auch diese Datei mitgeliefert und legt sie im selben Verzeichnis wie das Manifest ab. Somit kann vom Manifest Viewer jederzeit der Erstellungszeitpunkt des lokalen, eventuell bereits veralteten Manifests mit jenem des gewiss auf neuestem Stand befindlichen serverseitigen Manifests verglichen werden. Sollte dies der Fall sein, wird der Benutzer über das Vorhandensein einer aktuelleren Manifest-Version informiert.

Da das Speichervolumen auf PDA's zumeist knapp bemessen ist, gibt der Manifest Converter auch noch eine weitere Möglichkeit, sparsamer mit dem kostbaren Gut umzugehen: Komprimierung der JPEG's. Der Benutzer kann dabei wählen, ob und wie stark er JPEG-Grafiken komprimieren will. Je höher die Komprimierungsrate, desto kleiner wird die entsprechende Datei. Im Gleichklang dazu nimmt jedoch auch die bei JPEG's und MPEG's bekannte Artefaktbildung zu, was der Bildqualität wiederum natürlich abträglich ist. In Abbildung 29 ist zu sehen, dass der Konverter bei JPEG-Dateien, die sowohl in unkomprimierter als auch in komprimierter Form vorliegen, ein zusätzliches Attribut („cps_lq“) einfügt. Bei den beiden Parametern dieses Attributs handelt es sich wie von Attribut „cps_hq“ bereits bekannt um die Dateigröße in Bytes und das Erstellungsdatum in Millisekunden.

```
</resources>
<resource identifier="MANIFEST01_RESOURCE9" type="webcontent" href="Html/inhalt_information2.2
  <file href="Html/inhalt_information2.2_musik.htm" cps_hq="17627,6321496993742" />
  <file href="Html/bob_dylan.jpg" cps_hq="10609,6321496993170" cps_lq="1303,6321874775853" />
  <file href="Html/gunsn_roses.jpg" cps_hq="10207,6321496993301" cps_lq="1172,6321874775858" />
  <file href="Html/mozart.jpg" cps_hq="11088,6321496993493" cps_lq="1236,6321874775862" />
  <file href="Html/pics/handy.gif" cps_hq="4405,6321496993302" />
  <file href="Html/pics/notenblatt.wmf" cps_hq="5218,6321496993502" />
  <file href="Html/...
```

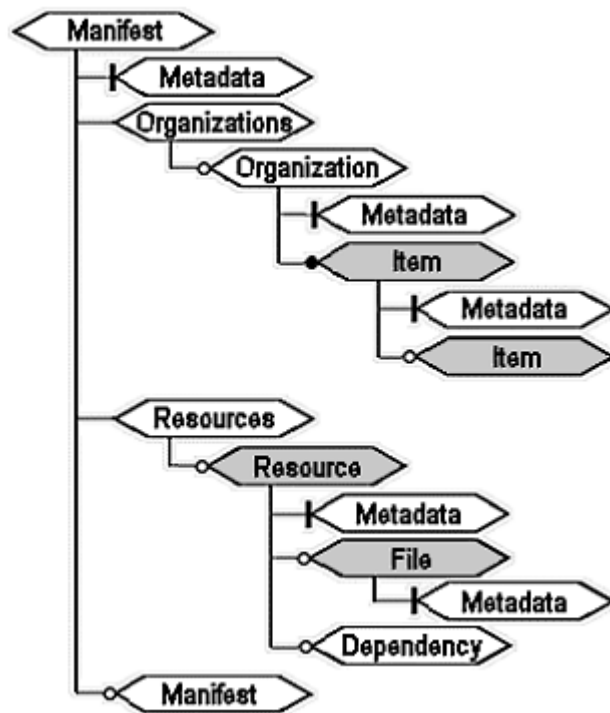
Abbildung 29: Zusätzliches File-Attribut für alternative JPEG-Datei

Die Existenz dieses optionalen Attributs vermittelt dem Manifest Viewer, dass das entsprechende JPEG-File sowohl in hoher als auch in niedriger Qualität auf dem Webserver vorhanden ist. Je nach Wunsch des Benutzers kann nun entweder das größere, qualitativ hochwertigere JPEG oder aber jenes in geringerer Qualität angefordert werden.

Zusammenfassend nimmt der Manifest Converter also folgende Modifikationen vor:

1. Hinzufügen aller aus HTML-Dokumenten referenzierten Dateien in Form von <file>-Tags zum jeweiligen <resource>-Element
2. Erweitern des <file>-Tags um ein Attribut („cps_hq“) zur Angabe von Dateigröße und Änderungsdatum der entsprechenden Datei
3. Gegebenenfalls erweitern des <file>-Tags um ein zusätzliches Attribut („cps_lq“) zur Angabe von Dateigröße und Änderungsdatum einer alternativen JPEG-Datei in niedrigerer Qualität
4. Entfernen irrelevanter <item>-Attribute
5. Entfernen nicht benötigter <resource>-Elemente
6. Gegebenenfalls einfügen von „künstlichen“ <item>-Elementen, um zu gewährleisten, dass <item>-Knoten, die ein Verzeichnis repräsentieren, auf keine Ressourcen referenzieren
7. Generieren einer Textdatei („cps_manifest.dat“) zur Speicherung des Zeitpunktes der Erstellung des modifizierten CPS Viewer Manifests

Sämtliche vom Manifest Converter eventuell modifizierten CPS-Elemente werden in Abbildung 30 durch die graue Hinterlegung nochmals grafisch hervor gehoben.



Quelle: IMS

Abbildung 30: Modifizierte CPS-Elemente

Im Detail besteht der Manifest Converter aus drei Komponenten, welche die zuvor aufgelisteten Konvertierungsaufgaben verrichten (siehe Abbildung 31:): „Manifest Reader/Writer“, „HTML-Parser“ sowie „JPEG-Compressor“.

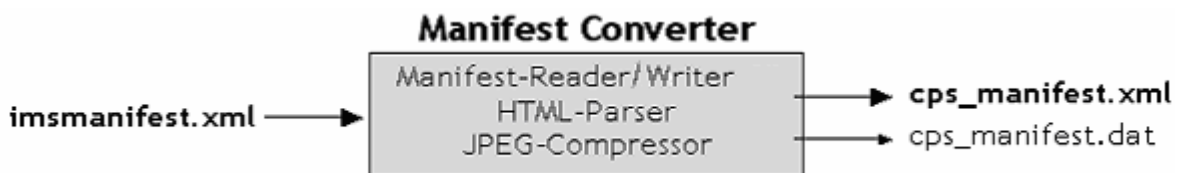


Abbildung 31: An der Konvertierung beteiligte Komponenten

Ersterer dupliziert im ersten Schritt das originale Manifest „imsmanifest.xml“ und nennt es „cps_manifest.xml“. Anschließend wird dieses Duplikat strukturiert durchlaufen, um gegebenenfalls irrelevante Attribute (vgl. Abbildung 27) und Ressourcen (vgl. Abbildung 28) heraus zu filtern und falls nötig zusätzliche <item>-Knoten einzufügen (vgl. Abbildung 26). An dieser Stelle ist anzumerken, dass der Manifest Reader/Writer das Manifest also nicht nur liest, sondern auch Änderungen bzw. Ergänzungen anbringt, weshalb ihm die etwas sonderbare Bezeichnung „Manifest Reader/Writer“ zufällt.

Gleichzeitig stehen aber auch die beiden anderen Komponenten Gewehr bei Fuß, um im Bedarfsfall ihre Arbeit aufnehmen zu können. Wird nämlich beim Parsen des Manifests eine Resource vorgefunden, die auf ein HTML-Dokument verweist bzw. ein solches in einem ihrer `<file>`-Tags beinhaltet, so tritt der HTML-Parser auf den Plan. Wie der Name schon sagt, parst dieser die entsprechende HTML-Datei nach weiteren Dateien, die aus dem HTML-Quellcode heraus referenziert werden. Im in Abbildung 32 dargestellten Beispiel ist der Quellcode einer solchen HTML-Datei abgebildet, aus dem ersichtlich ist, dass dieses HTML-Dokument eine GIF-Grafik beinhaltet.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Metadata</title>
</head>
<body>
<p></p>
</body>
</html>
```

Abbildung 32: HTML-Quellcode mit GIF-Verweis

Der HTML-Parser stöbert all diese Referenzen auf eingebettete Dateien auf und liefert sie dem Manifest Reader/Writer. Der wiederum prüft nun, ob der entsprechende `<resource>`-Knoten bereits sämtliche Verweise auf diese benötigten Dateien in Form von `<file>`-Tags beinhaltet. D.h. für all jene notwendigen Dateien, die bislang noch nicht im Manifest berücksichtigt wurden, werden zusätzliche `<file>`-Tags unterhalb der jeweiligen Resource erstellt (vgl. Abbildung 21).

An dieser Stelle ist jedoch zu erwähnen, dass der HTML-Parser möglicherweise nicht alle eingebetteten Dateien finden kann. Denn wie sein Name schon aussagt, parst er lediglich den HTML-Quellcode. Um jedoch die drei aus der HTML-Datei in Abbildung 33 referenzierten Grafiken ermitteln zu können, müsste der Parser das dafür verantwortliche JavaScript interpretieren können. Dies wäre bei dem abgebildeten und relativ einfachen Beispiel unter Umständen noch vorstellbar. Da der Komplexität von JavaScript-Code jedoch keine Grenzen gesetzt werden kann, wird dies als grundsätzlich nicht machbar betrachtet.



Abbildung 33: HTML-Quellcode mit GIF-Verweisen in JavaScript

Vielmehr ist der Ersteller der Kursmaterialien angehalten, diese Einschränkungen zu berücksichtigen und alle eingebetteten Ressourcen als statische Referenzen in das HTML-Dokument einzubinden (siehe Abbildung 34).

```

<html>

<head>
<title>Test</title>
</head>

<body>

<p></p>
<p></p>
<p></p>

</body>

</html>

```

Abbildung 34: HTML-Quellcode mit statischen Filereferenzen

Hat der Benutzer explizit den Wunsch geäußert, JPEG-Grafiken auch in einer alternativen, niedrigeren Qualität anzubieten (siehe Abbildung 35), so wird der JPEG-Compressor aktiv, sobald der Manifest Reader/Writer beim Abarbeiten des Manifests auf ein JPEG-Bild stößt.

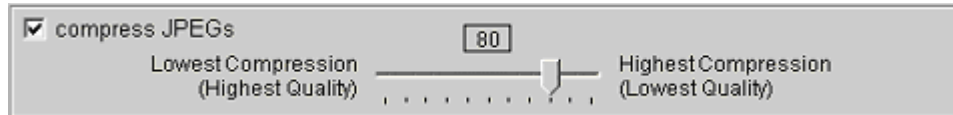


Abbildung 35: Panel zur Einstellung der Komprimierungseigenschaften

Dann nämlich generiert der JPEG-Compressor ausgehend von der vorhandenen JPEG-Datei ein neues JPEG, und zwar mit jener Komprimierungsrate, die vom Benutzer gewählt wurde (vgl. Abbildung 35). Diese neue Datei wird ebenso genannt wie das Original, nur dass vor der Dateiendung das Postfix „_cps_lq_“ eingefügt wird (siehe Abbildung 36).





 beispiel.jpg	65 KB
 beispiel_cps_lq_.jpg	21 KB
 help.jpg	8 KB
 help_cps_lq_.jpg	1 KB

Abbildung 36: JPEG's in alternativen Qualitätsstufen

Der JPEG-Compressor verständigt nun den Manifest Reader/Writer, dass ein aus dem Manifest referenziertes File nun in einer alternativen Form vorliegt. Der Manifest Reader/Writer reagiert auf diese Information, indem er den <file>-Tag um das bereits erwähnte Attribut „cps_lq“ erweitert und in diesem Dateigröße und Änderungsdatum des alternativen JPEG-Bildes festhält.

Nach vollständigem Durchlauf des Manifesthierarchie und Erledigung aller Modifikationen am Manifest bleibt dem Manifest Reader/Writer nur noch die Aufgabe, eine Textdatei mit dem Erstellungsdatum des soeben konvertierten Manifests zu generieren. Somit kann die Konvertierung als abgeschlossen betrachtet werden.

Um diesen Kurs nun auch zu veröffentlichen, hat der Ersteller des neuen Manifests noch zwei Abschlussarbeiten zu erledigen:

- Die beiden neuen Dateien „cps_manifest.xml“ und „cps_manifest.dat“ sowie der gesamte Content des zu veröffentlichenden Kurses müssen auf einen Webserver kopiert werden.

- Jene Datei am Webserver, welche die Titel aller veröffentlichten Kurse sowie deren Pfade beinhaltet („cps_courses.txt“), muss um einen Eintrag für den neuen Kurs ergänzt werden. Sie liegt stets an einem fixen Ort (zum Beispiel im Root-Verzeichnis des Webserver), sodass sie von jedem Viewer, der sich mit dem entsprechenden Server verbindet, gefunden und gelesen werden kann.

Auf dem Server im folgenden Beispiel sind drei Kurse veröffentlicht. Die Textdatei „cps_courses.txt“ hat folgenden Inhalt:

```
Testkurs | Courses/Test  
PHP-Kurs | Courses/PHP  
Propaedeutikum | Courses/Propaedeutikum
```

Der Eintrag vor dem Separator („|“) steht für den Titel des Kurses, jener dahinter gibt den Pfad zum jeweiligen Kurs an, wobei sich der Verzeichnisname durchaus vom Kurstitel unterscheiden kann.

Aus diesen Daten würde sich folgende Verzeichnis- und Dateistruktur auf dem jeweiligen Webserver (hier „www.elearning-kurse.at“) ergeben:

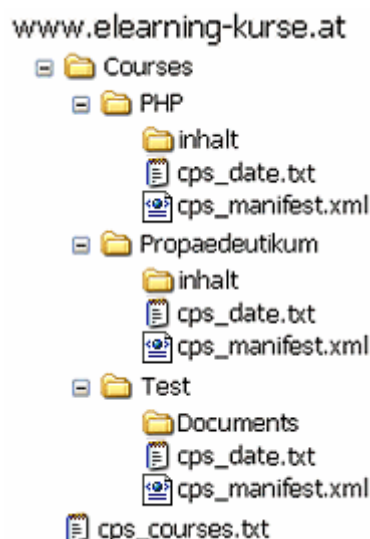


Abbildung 37: Beispiel einer Dateistruktur

Der Content, auf den aus den konvertierten Manifesten heraus referenziert wird, liegt bei zwei Kursen in einem Verzeichnis namens „inhalt“, beim Dritten im Verzeichnis „Documents“.

Somit ist der erste und unbestritten wichtigste Teil in der Beschreibung des Manifest Administrators abgehandelt. Im folgenden Kapitel wird nun dessen zweite Komponente – der Manifest Updater – behandelt.

7.2.2. Manifest Updater

Die Steuerelemente des Manifest Updaters sind in der unteren Hälfte der Benutzeroberfläche des Manifest Administrators angesiedelt (siehe Abbildung 38). Seine Aufgabe besteht darin, Kurse, die mit dem Manifest Converter in ein CPS Viewer Manifest konvertiert wurden, auf den neuesten Stand zu bringen.

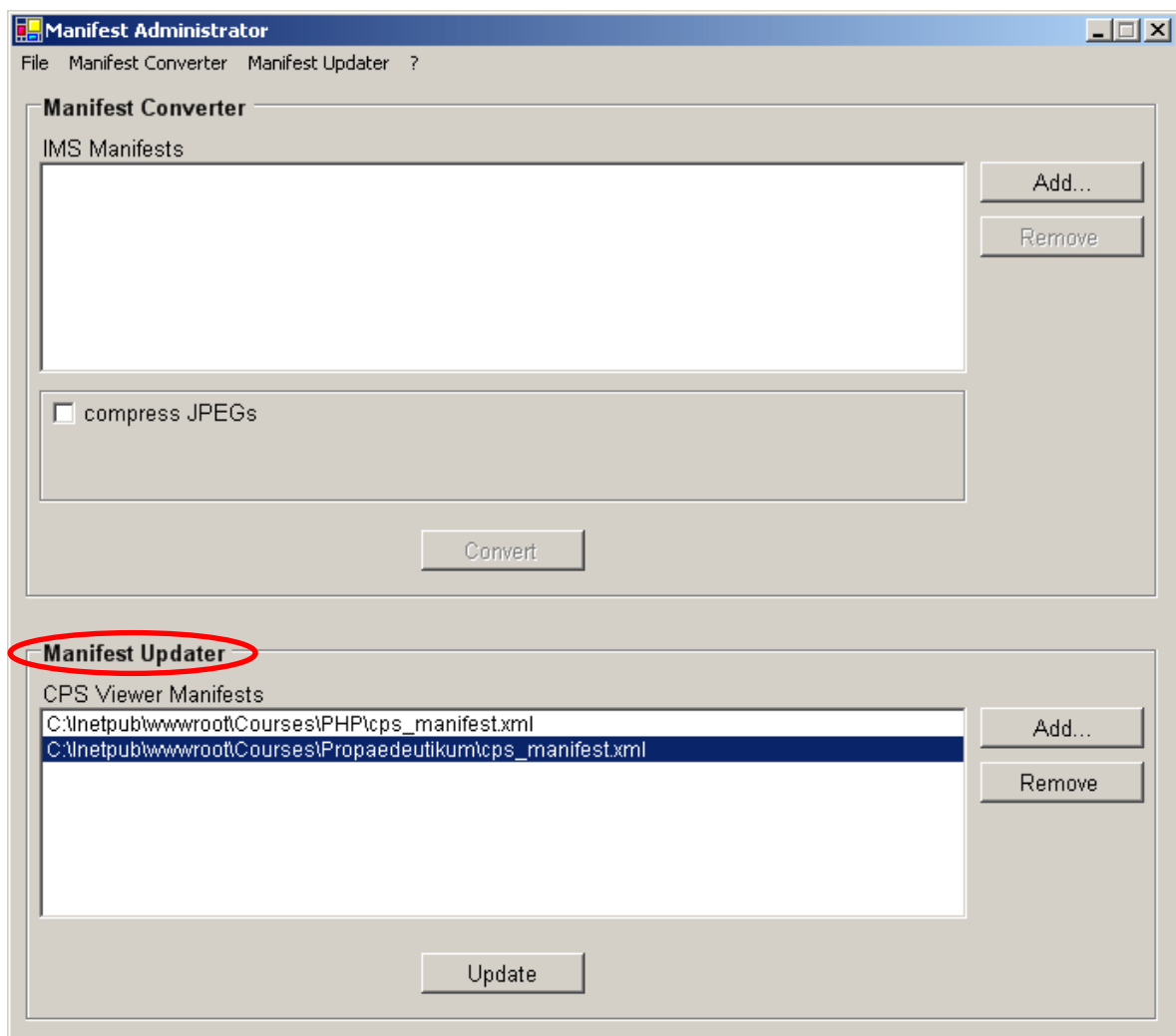


Abbildung 38: Steuerelemente des Manifest Updaters

Doch welche Änderungen können vom Manifest Updater überhaupt erfasst und somit ins aktualisierte Manifest eingebracht werden? Nun, grundsätzlich ist festzuhalten, dass für sämtliche Dateien, die unterhalb der <resource>-Knoten angeführt sind, geprüft wird, ob diese in neueren Versionen vorliegen (siehe Abbildung 39).

```
</resource>
- <resource identifier="MANIFEST01_RESOURCE9" type="webcontent" href="html/inhalt_information2.2_musik.htm">
  <file href="html/inhalt_information2.2_musik.htm" cps_hq="6448,6323200164448" />
  <file href="html/pics/bob_dylan.jpg" cps_hq="10609,6310394685600" cps_lq="1472,6321991940600" />
  <file href="html/pics/guns_n_roses.jpg" cps_hq="10207,6310394697200" cps_lq="1328,6321991940800" />
  <file href="html/pics/mozart.jpg" cps_hq="11088,6310464621200" cps_lq="1399,6321991940800" />
  <file href="html/pics/handy.gif" cps_hq="4405,6310394787400" />
  <file href="html/pics/notenblatt.wmf" cps_hq="5218,6302505342200" />
  <file href="html/pics/champion.gif" cps_hq="3782,6310532462000" />
  <file href="html/pics/handy_musik.jpg" cps_hq="38801,6310394093200" cps_lq="4454,6321991940800" />
  <file href="html/pics/prev_blue.gif" cps_hq="161,6310358866800" />
  <file href="html/knocking_bob.wav" cps_hq="177200,6310394639200" />
  <file href="html/knocking_guns.wav" cps_hq="203888,6310394641800" />
  <file href="html/mozart_1.wav" cps_hq="197900,6310464589200" />
  <file href="html/mozart_1.mp3" cps_hq="803630,6313339610200" />
  <file href="html/champions.mid" cps_hq="549,6310394363800" />
</resource>
```

Abbildung 39: Ressource aus CPS Viewer Manifest

Abbildung 40 zeigt eine Ressource, die aus zwei Files besteht. Im Zuge des Updates des CPS Viewer Manifests wird nun für die Dateien „information.htm“ und „back.gif“ ermittelt, ob diese noch aktuell sind.

```
<resource identifier="RESOURCE1" type="webcontent" href="docs/page1.htm">
  <file href="docs/information.htm" cps_hq="14692,6310703625000" />
  <file href="docs/back.gif" cps_hq="410,6310703625000" />
</resource>
```

Abbildung 40: Resourcefiles vor dem Update

Findet der Manifest Updater neuere Versionen, werden im CPS Viewer-Manifest die Attributwerte bezüglich Dateigröße und Änderungsdatum aktualisiert. Im Beispiel wurde die Datei „information.htm“ durch eine neuere ersetzt, was sich in den beiden neuen Werten des Attributs „cps_hq“ niederschlägt (siehe Abbildung 41). Die GIF-Datei wiederum wurde nicht angetastet, weshalb auch die entsprechenden Attributwerte nicht geändert wurden.

```
<resource identifier="RESOURCE1" type="webcontent" href="docs/page1.htm">
  <file href="docs/information.htm" cps_hq="12281,6320529313800">
  <file href="docs/back.gif" cps_hq="410,6310703625000" />
</resource>
```

Abbildung 41: Ressourcefiles nach dem Update

Enthält ein <file>-Tag auch Informationen über eine Version in niedriger Qualität (cps_lq), so wird natürlich auch für die entsprechende speicherplatzsparende Alternativ-Datei kontrolliert, ob diese durch eine neuere ersetzt wurde. Der <file>-Knoten in Abbildung 42 beispielsweise verweist auf die Grafikdatei „img1.jpg“. Implizit wird jedoch auch noch eine Version dieser Datei in niedriger Qualität referenziert, da das Attribut „cps_lq“ zu finden ist.

```
<file href="pics/img1.jpg" cps_hq="31334,6310703431000" cps_lq="3398,6310703457000" />
```

Abbildung 42: Dateireferenz mit alternativer Darstellung vor Aktualisierung

Der Manifest Updater prüft also bei der Abarbeitung dieses <file>-Tags, ob die Dateien „img1.jpg“ und „img1_cps_lq.jpg“ ausgetauscht wurden. Da für beide Files neuere Versionen vorhanden sind, werden die Werte der beiden Attribute „cps_hq“ und „cps_lq“ entsprechend angepasst (siehe Abbildung 43).

```
<file href="pics/img1.jpg" cps_hq="35408,6310703625000" cps_lq="3511,6310703851000" />
```

Abbildung 43: Dateireferenz mit alternativer Darstellung nach Aktualisierung

Der Manifest Updater aktualisiert also lediglich die Kennwerte für Dateigröße und Änderungsdatum der verwendeten Dateien, ändert aber nichts an der Struktur des CPS Viewer Manifests. D.h. der Aufbau des Kurses bleibt auch nach Aktualisierung gänzlich identisch. Hat der Kursersteller die feste Absicht, die Kursstruktur zu ändern (z.B. Kapitel umreihen, Kapitel hinzufügen, Kapitel entfernen), so bleibt ihm nichts anderes übrig, als den Kurs mit dem Manifest Converter neu zu erstellen.

7.3. Manifest Viewer

Der Manifest-Viewer ist eine für Handhelds entwickelte Applikation, die den Benutzer befähigt, auf diesem elektronische Kurse anzusehen. Diese Kurse müssen entsprechend der

IMS Content Packaging Specification aufgebaut und mit dem oben bereits behandelten Manifest Administrator PDA-gerecht adaptiert worden sein.



Abbildung 44: Splashscreen des Manifest Viewers

Da das Speicherangebot bei Handhelds zumeist knapp bemessen ist, muss ein gezieltes Hochladen einzelner Materialien möglich sein. Aus selbigem Grund muss der Lernende aber auch einzelne Kursteile oder aber einen ganzen Kurs wieder von seinem Handheld entfernen können.

Die folgende Aufzählung gibt einen groben Überblick der Hauptfunktionen des Manifest Viewers:

- Anzeigen der auf dem Server verfügbaren Kurse
- Laden und Löschen von Manifesten
- Anzeigen der Kursstruktur (Kapitel und Dokumente)
- Laden und Löschen von Kursmaterialien
- Ansehen von Kursmaterialien
- Anzeigen von Informationen zu Kursen und einzelnen Kursmaterialien (Metadaten, Speicherbedarf)
- Überprüfen der Aktualität des Kurses (Vergleich mit Referenzkurs am Server)

Bevor die gewünschten Kursinhalte auf dem mobilen Gerät gelernt werden können, muss zuerst eine Verbindung mit einem Server, auf welchem Kurse angeboten werden, hergestellt

werden. Der Benutzer hat dazu im dafür vorgesehenen Login-Dialog die Serveradresse und gegebenenfalls Benutzername und Kennwort anzugeben (siehe Abbildung 45).

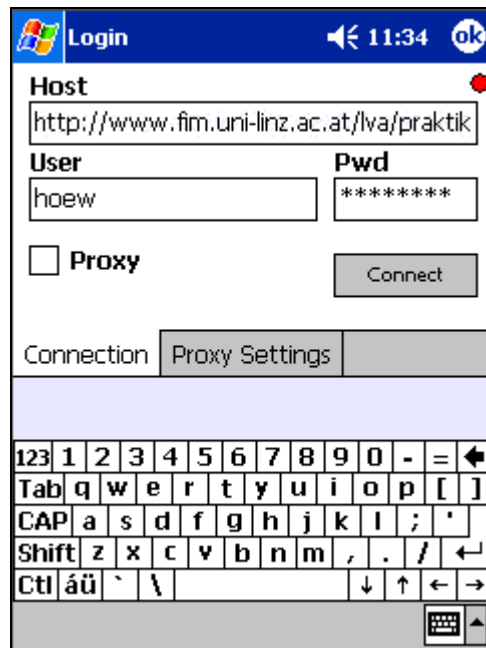


Abbildung 45: Login-Dialog des Manifest Viewers

Läuft die Verbindung über einen Proxy, sind darüber hinaus auch noch dessen Verbindungseinstellungen zu konfigurieren (siehe Abbildung 46).

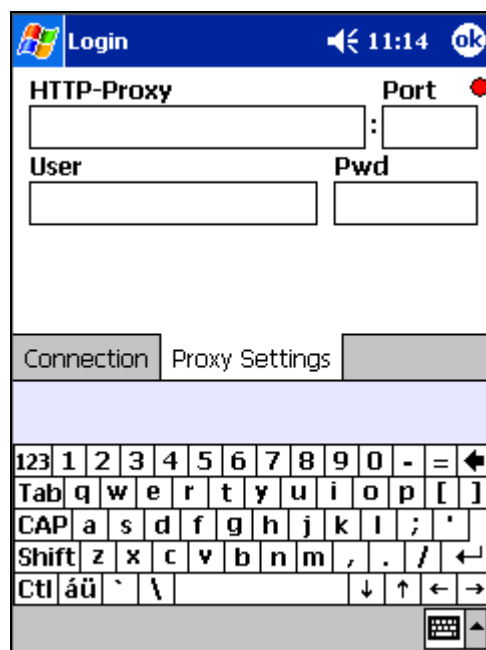


Abbildung 46: Dialog für Proxy-Einstellungen

Nach Bereitstellung aller notwendigen Informationen prüft der Manifest Viewer, ob diese korrekt sind – sprich ob der gewünschte Server damit erreichbar ist. Genau genommen wird versucht, jene Datei, welche Informationen über die am Server veröffentlichten Kurse enthält, auf den PDA zu transferieren. Wie bereits bei der Beschreibung des Manifest Administrators erwähnt, handelt es sich dabei um das File „cps_courses.txt“, welches sich im Root-Verzeichnis des Servers zu befinden hat.

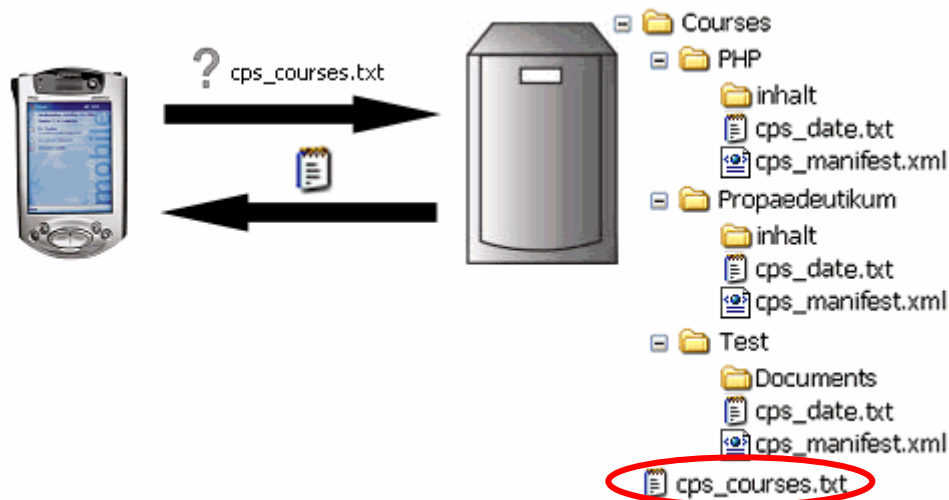


Abbildung 47: Kommunikationsüberprüfung Handheld ↔ Server

Konnte die Datei mit den Kursinformationen auf das Handheld übertragen werden, ist gewährleistet, dass die Verbindung zum Server funktioniert. Dem Benutzer wird dies durch einen grün gefüllten Kreis rechts oben signalisiert. Dieser zeigt nämlich den Verbindungsstatus zu jenem Server, auf dem sich der Anwender eingeloggt hat, an. Grundsätzlich sind folgende Stati aus Tabelle 1 möglich:

- ... Keine Verbindung zum Server hergestellt
- ... Verbindung zum Server hergestellt
- ... Verbindung zum Server unterbrochen
- ... Versuch des Verbindungsaufbaues zum Server

Tabelle 1: Verbindungsstati

Die Bedeutung der ersten beiden Stati dürfte auf der Hand liegen: Der rot gefüllte Kreis signalisiert, dass der Benutzer auf keinem Server eingeloggt ist und folglich auch keine Verbindung bestehen kann.

Grün wird der Kreis, wenn das Login erfolgreich war und eine Verbindung zum Server hergestellt werden kann.

Das dritte Symbol – außen grün, innen rot – weist darauf hin, dass der Benutzer zwar auf einem Server eingeloggt ist, die Verbindung zu diesem jedoch (zwischenzeitlich) nicht mehr besteht. Dieser Fall tritt beispielsweise ein, falls der Server aus welchen Gründen auch immer ausfallen sollte. Es wäre aber auch denkbar, dass der Benutzer mit seinem mobilen Gerät via WLAN kommuniziert und den Wirkungsbereich eines WLAN-Access Points verlässt. Sollte die Verbindung in weiterer Folge wieder möglich sein, färbt sich das Symbol natürlich wieder zur Gänze grün.

Der weiße Kreis weist lediglich darauf hin, dass der Manifest Viewer gerade testet, ob eine Verbindung zum Server besteht. Da das Ergebnis dieser Überprüfung im Normalfall innerhalb kürzester Zeit vorliegt, nimmt der Benutzer dieses Symbol nur als Aufflackern wahr.

Nun zurück zum Szenario des Herunterladens der Datei „cps_courses.txt“. Da die Verbindung zum Server hergestellt werden konnte – ersichtlich am grünen Kreis – zeigt der Manifest Viewer dem Benutzer die verfügbaren Kurse an. Dazu werden die einzelnen Kursnamen aus der übertragenen Datei ausgelesen und anschließend aufgelistet (siehe Abbildung 48).

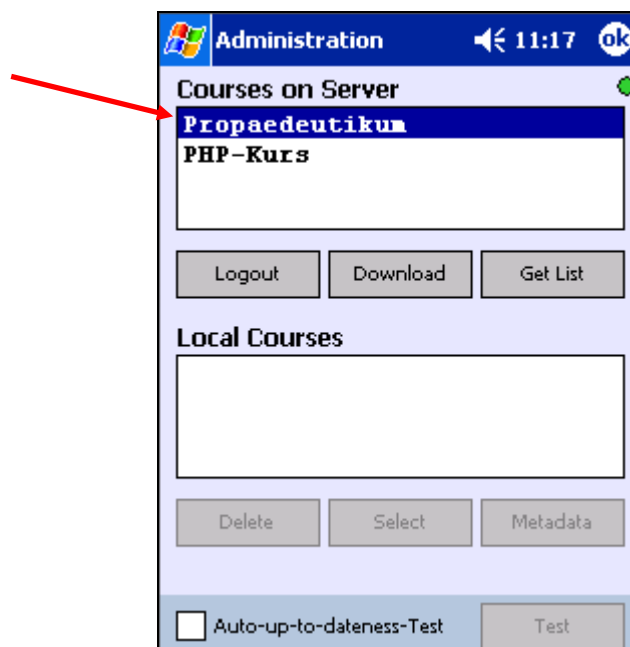


Abbildung 48: Auflistung der serverseitig verfügbaren Kurse

Der Anwender hat nun die Möglichkeit, die Manifeste der veröffentlichten Kurse auf sein mobiles Gerät zu laden. Er selektiert dazu den gewünschten Kurs und tippt auf die Taste

„Download“. Nun wird zuerst das File mit dem Erstellungsdatum des Kurses übertragen („cps_date.txt“). Anhand des darin enthaltenen Zeitstempels kann zukünftig festgestellt werden, ob der lokal auf dem Handheld befindliche Kurs noch aktuell ist, d.h. ob er dem entsprechenden Kurs am Server strukturell gleicht. Konnte auch diese Datei übertragen werden, steht als nächstes der Transfer des Manifests an. Der Manifest Viewer setzt also wiederum ein Web-Request über HTTP ab, um die Datei „cps_manifest.xml“ des gewählten Kurses herunterzuladen (siehe Abbildung 49).

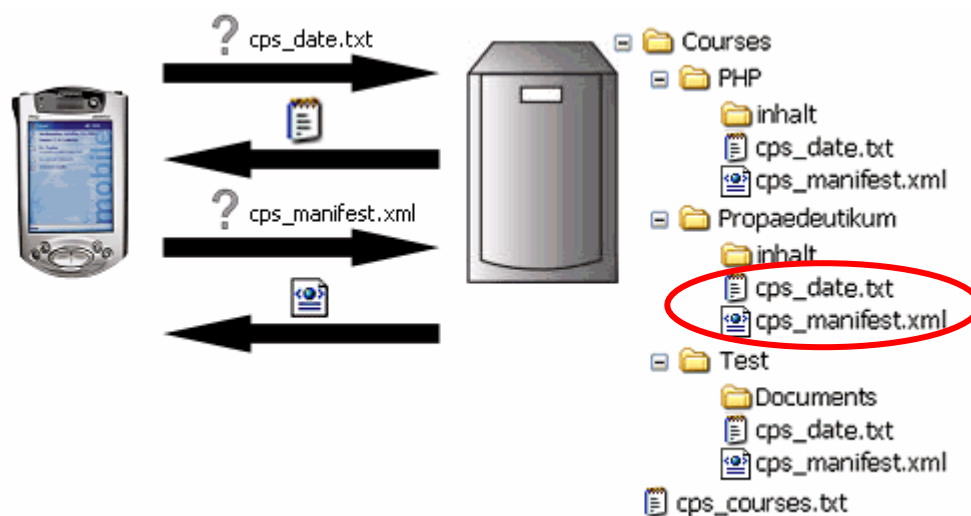


Abbildung 49: Anfordern eines Kurses vom Server

Sobald die Meldung eintrifft, dass das Manifest erfolgreich auf dem mobilen Gerät erstellt werden konnte, wird im lokalen Kursverzeichnis zusätzlich ein Konfigurationsfile („cps_manifest.cfg“) angelegt, welches folgende Informationen über den jeweiligen Kurs beinhaltet:

- Titel
- Pfad des Kursverzeichnisses am Host
- Adresse des Hosts
- ggf. Proxy-Adresse und Proxy-Port
- Zeitstempel (aus „cps_date.txt“)

Diese Datei beinhaltet also jene Daten, die es dem Manifest Viewer ermöglichen, eine Verbindung zum entsprechenden Server aufzubauen und anschließend Materialien des jeweiligen Kurses herunterzuladen.

Zuletzt wird dem Benutzer durch Hinzufügen des Kurstitels zur Auflistung der lokal verfügbaren Kurse noch signalisiert, dass das gewünschte Manifest nun auf seinem mobilen Gerät verfügbar ist. Der Anwender kann nun einen beliebigen Kurs, dessen Manifest auf dem PDA verfügbar ist, öffnen. Er selektiert dazu unter „Local Courses“ den gewünschten Kurs und tippt auf „Select“ (siehe Abbildung 50).

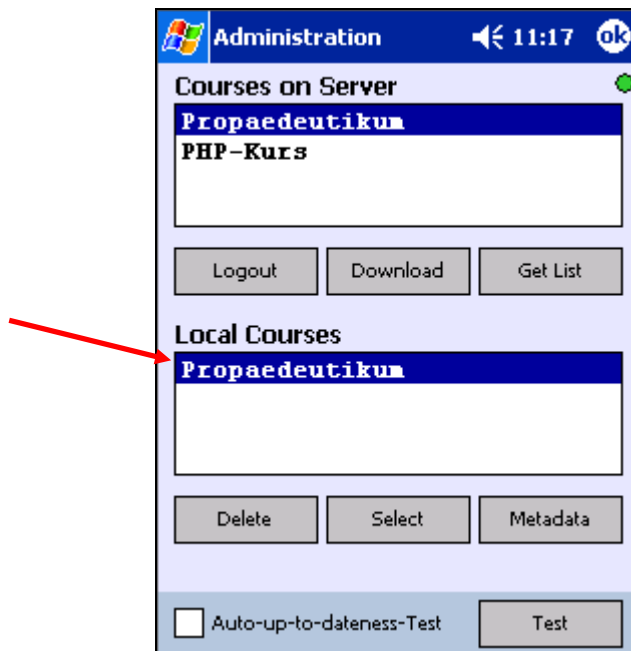


Abbildung 50: Anzeige der lokal verfügbaren Kurse

Nun setzt eine sehr wichtige Komponente des Manifest Viewers mit der Arbeit ein, denn das jeweilige Manifest muss nun eingelesen werden, um in weiterer Folge die Kursstruktur anzeigen zu können. Intern passiert jedoch viel mehr als nur das schöne Einlesen von XML-Knoten und deren Attributen. Der Manifest Viewer instanziiert eine Vielzahl an Objekten, welche miteinander interagieren und einen geordneten Ablauf im Umgang mit dem Manifest Viewer sicherstellen.

Begeben wir uns aus diesem Grund nochmals in die „Niederungen“ eines XML-Manifests. Die Kursstruktur wird durch die Gliederung der <item>-Elemente – unterhalb des Knotens <organization> – festgelegt. Jedes Item kann also wiederum beliebig viele Items beinhalten, d.h. im Grunde repräsentiert es entweder ein Verzeichnis bzw. – falls es keine untergeordneten Items hat – einen Dokument-Knoten (siehe Abbildung 51).

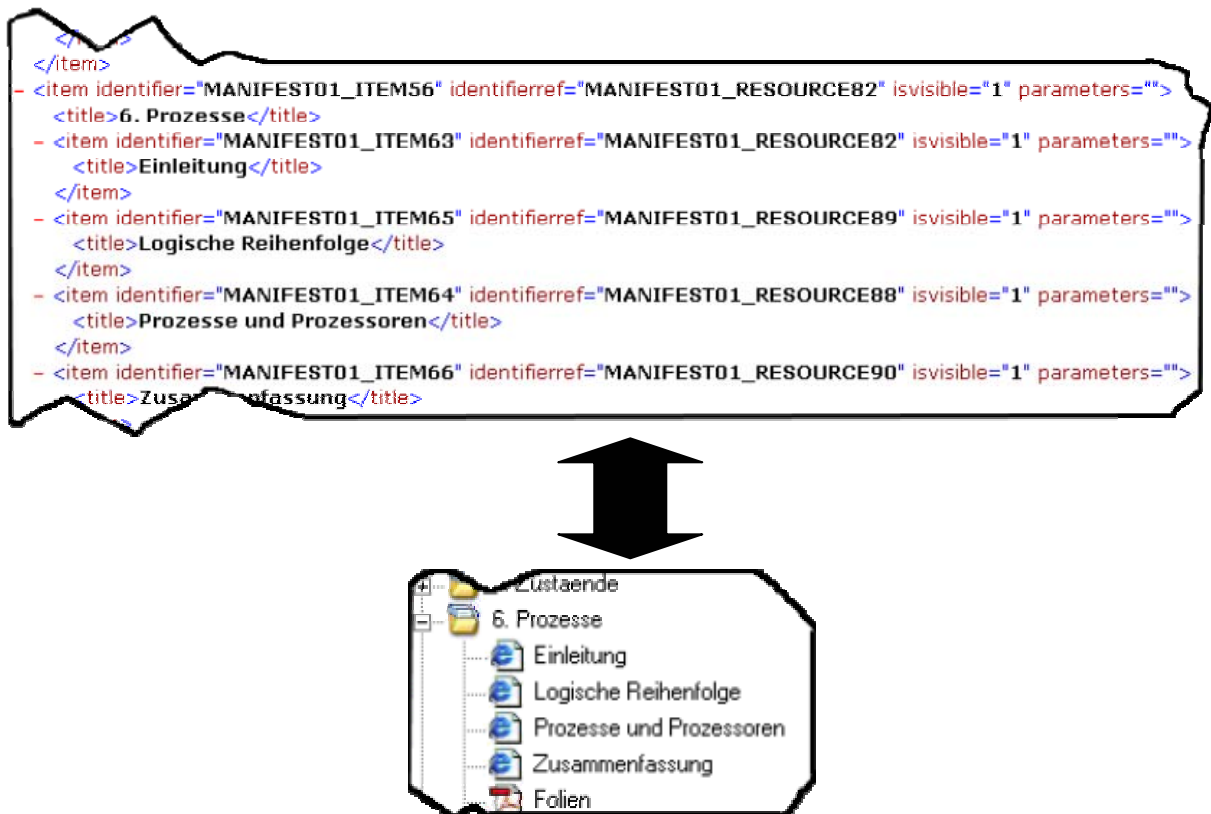


Abbildung 51: Umsetzung der Item-Hierarchie in Manifest Viewer-Struktur

Aus Abbildung 51 ist zudem ersichtlich, dass Items durch das Attribut „identifierref“ auf eine Ressource verweisen können. Hinter diesen Ressourcen verbirgt sich der Content des Kurses, d.h. die Kursmaterialien – von HTML-Seiten, PDF- und Office-Dokumente bis hin zu Bild-, Audio- und Video-Dateien.

Wie bereits in einem der vorigen Kapitel erklärt, können diese <resource>-Knoten ebenfalls eine beliebige Anzahl an XML-Subknoten besitzen. Dabei kann es sich einerseits um <file>-, andererseits um <dependency>-Knoten handeln, wobei letztere selbst wiederum Ressourcen sind.

Zusammenfassend sei nochmals festgehalten, dass Struktur und Inhalt eines Kurses also durch <item>-, <resource>- und <file>-Knoten festgelegt werden (siehe Abbildung 52). Die oben bereits kurz angesprochene Komponente, die das Manifest beim Öffnen einen Kurses verarbeitet, lehnt sich stark an dieses Faktum an und erzeugt Objekte vom Typ „FolderNode“, „LeafNode“, „Resource“ und „ResourceFile“. Die beiden ersteren entstehen aus <item>-Knoten, wobei für Verzeichnis-Items „FolderNodes“, für Dokument-Items „LeafNodes“ angelegt werden. Aus <resource>-Knoten werden – wohl wenig überraschend – „Resource“-Objekte, aus <file>-Knoten Objekte des Typs „ResourceFile“.

```

<organizations default="TOC1">
  <organization identifier="TOC1" structure="hierarchical">
    <title>Test-Kurs</title>
    <item identifier="ITEM1">
      <title>Kapitel 1</title>
      <item identifier="ITEM2" identifierref="RESOURCE1">
        <title>Kapitel 1.1</title>
      </item>
      <item identifier="ITEM3" identifierref="RESOURCE2">
        <title>Kapitel 1.2</title>
      </item>
    </item>
  </organization>
</organizations>
<resources>
  <resource identifier="RESOURCE1" type="webcontent" href="docs/page1.htm">
    <file href="docs/page1.htm" cps_hq="281,6320529313800" />
    <file href="docs/pic1.gif" cps_hq="410,6310703625000" />
    <dependency identifierref="RESOURCE3" />
  </resource>
  <resource identifier="RESOURCE2" type="webcontent" href="docs/page2.htm">
    <file href="docs/page2.htm" cps_hq="58409,6320529723000" />
    <file href="docs/pic1.gif" cps_hq="410,6310703625000" />
    <dependency identifierref="RESOURCE3" />
  </resource>
  <resource identifier="RESOURCE3" type="webcontent">
    <file href="docs/pic2.gif" cps_hq="350,6310703422000" />
  </resource>
</resources>

```

Abbildung 52: Kursstruktur im CPS Viewer Manifest

Genau wie die entsprechenden XML-Elemente im Manifest stehen auch diese Objekte miteinander in Beziehung. Im Klartext bedeutet dies, das FolderNodes eine beliebige Anzahl an LeafNodes enthalten. LeafNodes ihrerseits können ein Resource-Objekt mit sich führen, welche wiederum Behälter für beliebig viele Resources (dependencies) und ResourceFiles sind.

Wie aus der Abbildung oben ersichtlich, wird für mehrfach auftretende <file>-Tags lediglich ein ResourceFile instanziiert. Kommt also ein und dieselbe physische Datei in mehreren <resource>-Knoten vor, so wird die erstellte Instanz dieses ResourceFiles zu allen übergeordneten Resource-Objekten hinzugefügt.

Analog ist die Situation bei <dependency>-Elementen, die Bestandteil mehrerer Resources sind. Auch diese werden lediglich einmal instanziiert, egal in wie vielen Resources auch immer sie enthalten sind.

Für das in Abbildung 52 angeführte Beispiel werden im Zuge der Verarbeitung des Manifests also die in Abbildung 53 dargestellten Instanzen generiert:

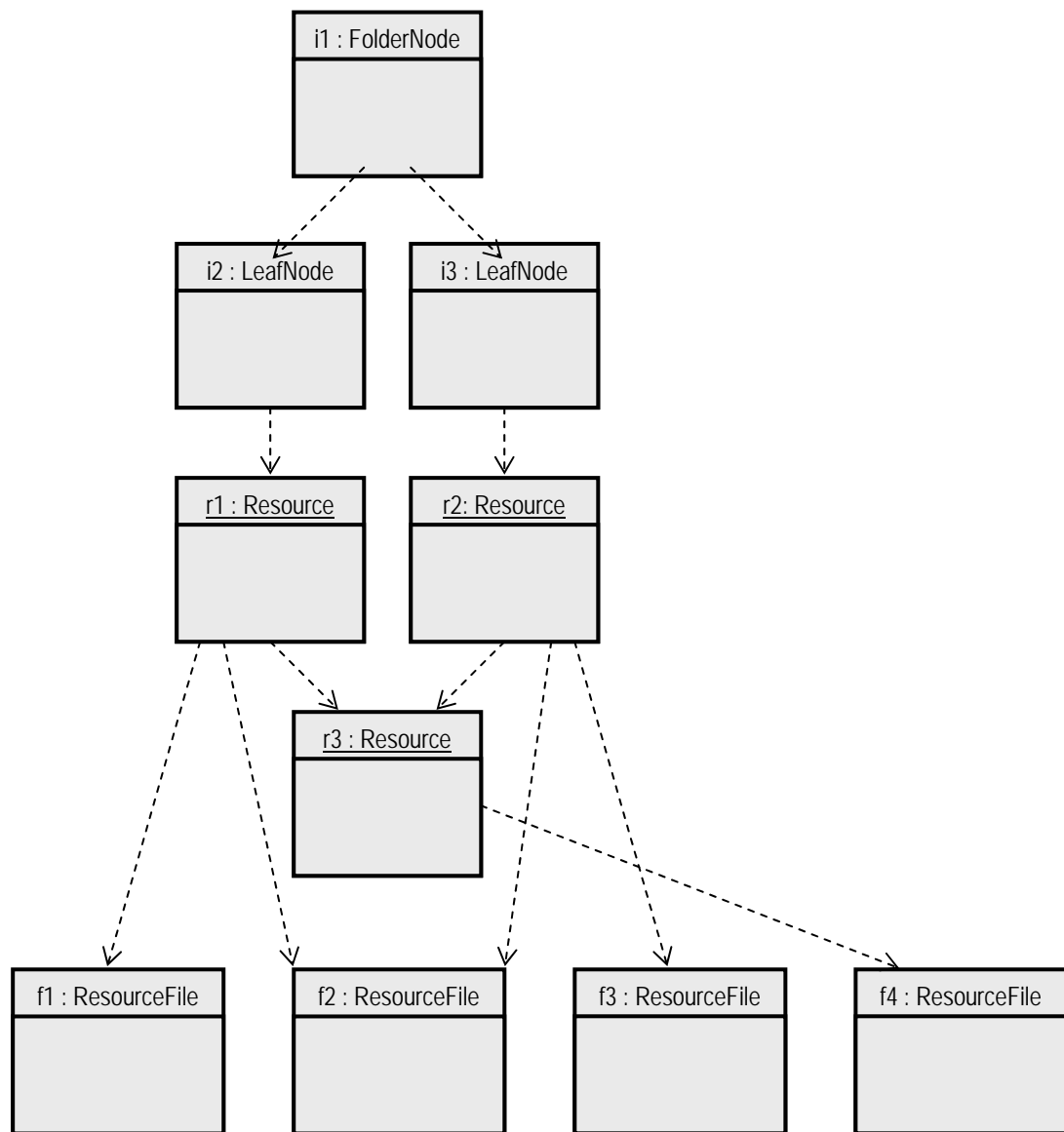


Abbildung 53: Instanzierte Geschäftsobjekte

Die Verbindungslinien zwischen den Objekten deuten an, dass zwischen diesen eine Benutzungsabhängigkeit besteht. Dies bedeutet, dass ein Objekt ein anderes für sein Funktionieren benötigt. Für die Resource `r1` (identifier="RESOURCE1") beispielsweise sind sowohl die beiden ResourceFiles `f1` (href="docs/page1.htm") und `f2` (href="docs/pic1.gif") als auch die Resource `r3` (identifier="RESOURCE3") erforderlich.

Darüber hinaus können Änderungen eines Objekts zu Änderungen abhängiger Objekte führen. Nehmen wir zum Beispiel an, Resource `r1` ist vollständig, d.h. sämtliche Files von `r1`

wurden auf das Handheld heruntergeladen. Ein Löschen der Datei „docs/page1.htm“ von ResourceFile f1 würde nun bedeuten, dass auch die Resource r1 nicht mehr vollständig ist und somit deren Zustand verändert wurde.

Die unterschiedlichen Objekttypen können übrigens eine Reihe von verschiedenen Zuständen einnehmen. Abbildung 54 zeigt, in welche Zustände ResourceFiles durch entsprechende Ereignisse übergeführt werden. Ursprünglich hat ein ResourceFile den Zustand „Unloaded“, d.h. die entsprechende Datei ist nicht lokal auf dem PDA verfügbar. Wird die Datei heruntergeladen, geht das ResourceFile entweder in den Zustand „Download_HQ“ oder „Download_LQ“ über, je nachdem, ob die Datei in hoher oder niedriger Qualität angefordert wurde. Konnte der Download-Vorgang erfolgreich abgeschlossen werden, nimmt das ResourceFile schließlich den Zustand „Downloaded_HQ“ oder „Downloaded_LQ“ ein, welcher erst durch Löschen der jeweiligen Datei wieder verlassen wird. Schlug das Herunterladen fehl oder wird die erfolgreich heruntergeladene Datei zu einem späteren Zeitpunkt wieder gelöscht, wechselt das ResourceFile wieder in den initialen Zustand „Unloaded“.

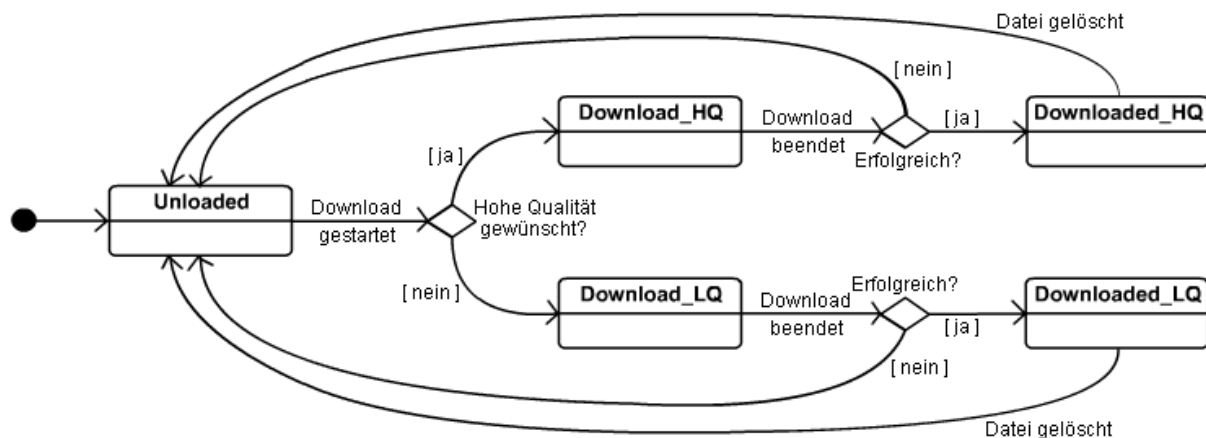


Abbildung 54: Zustandsdiagramm für ResourceFiles

Sehr ähnlich sieht das Zustandsdiagramm der Resources aus (siehe Abbildung 55). Anfangs ist eine Resource „Incomplete“ (unvollständig). Werden nun ResourceFiles dieser Resource heruntergeladen, wechselt der Zustand entweder in „Download_HQ“ oder „Download_LQ“. Ist zumindest eine Datei dieser Resource bereits in niedriger Qualität auf dem mobilen Gerät verfügbar bzw. wird zumindest eine in niedriger Qualität heruntergeladen, so wird der Zustand „Download_LQ“ zugewiesen. Nach Abschluss des Download-Vorgangs wird

geprüft, ob alle ResourceFiles der Resource verfügbar sind. Denn nur dann ist diese vollständig und wechselt in den Zustand „Complete_HQ“ bzw. „Complete_LQ“. Andernfalls wird die Resource wieder der Zustand „Incomplete“ übergeführt.

Ebenfalls in den Zustand „Incomplete“ wird gewechselt, falls zumindest ein ResourceFile einer bis zu diesem Zeitpunkt vollständigen Resource („Complete_HQ“ bzw. „Complete_LQ“) gelöscht wird. Ist eine Resource dann im Zustand „Incomplete“, verbleibt sie natürlich in diesem, auch wenn weitere ResourceFiles dieser Resource gelöscht werden.

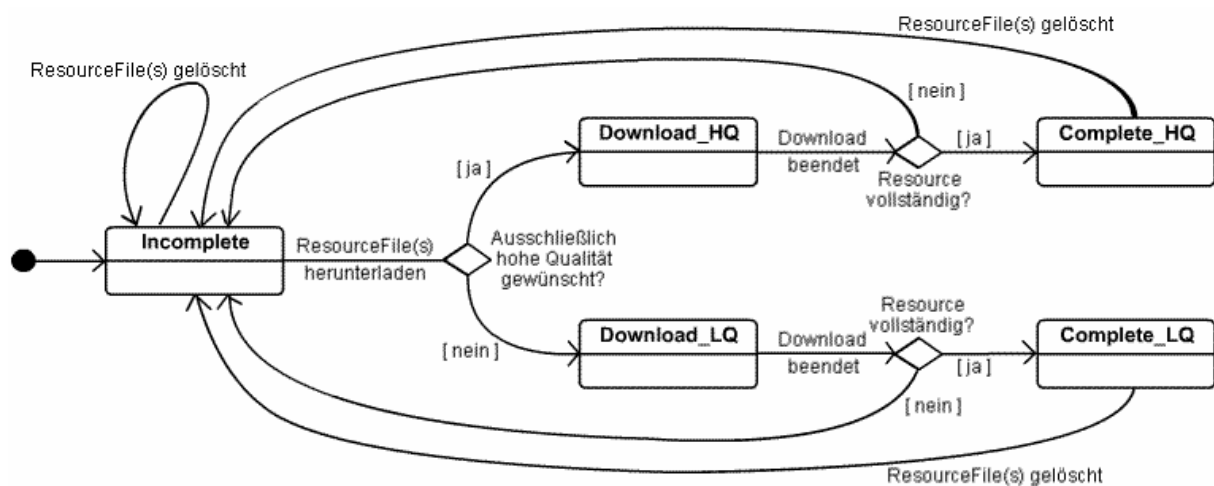


Abbildung 55: Zustandsdiagramm für Resources

Das Zustandsdiagramm aus Abbildung 55 belegt übrigens die oben angesprochene Benutzungsabhängigkeit zwischen ResourceFiles und Resources, und zwar insofern, als ein Zustandswechsel einer Resource durch Herunterladen oder Löschen von ResourceFiles ausgelöst werden kann. Eine Resource muss also auf Zustandsänderungen enthaltener ResourceFiles und Dependencies reagieren, was dadurch erreicht wird, dass sie diese beauftragt, bei Statuswechseln in Kenntnis gesetzt zu werden (.NET-Konzept der Events und Delegates). Wird beispielsweise das ResourceFile f2 in hoher Qualität heruntergeladen, wechselt dessen Zustand von „Unloaded“ in „Download_HQ“. Gleichzeitig setzt das ResourceFile ein Event ab, welches seinen neuen Status verkündet. Die beiden Ressourcen r1 und r2 – beide beinhalten f2 – empfangen dieses Event und ermitteln anschließend, ob dadurch ihr eigener Zustand geändert wird (vgl. Abbildung 53).

Da Resources selbst wiederum Bestandteil übergeordneter Objekte – nämlich von LeafNodes – sind, haben auch sie die Aufgabe, diesen etwaige Statusänderungen mitzuteilen. Da ein

LeafNode genau eine Resource enthält, bewirkt ein Wechsel des Zustands einer Resource stets eine gleichartige Zustandsänderung beim entsprechenden LeafNode-Objekt.

Tabelle 2 zeigt, wie der Manifest Viewer die verschiedenen Zustände von LeafNodes durch eine spezielle Farbgebung kennzeichnet, wobei die Art des Symbols vom Typ der beinhalteten Resource abhängt. Im Beispiel handelt es sich bei der Resource um ein HTML-Dokument, was am verwendeten Icon zu erkennen ist. Das Schema der Farbgebung für die unterschiedlichen Stati lässt sich jedoch auf sämtliche verwendete Dokumenttypen (PDF, DOC, PPT, ...) umlegen.







Zustand des LeafNodes (= Zustand d. Resource)	Icon
Incomplete	 → grau
Download_HQ	 → grün
Download_LQ	 → grün, schwarzes Eselsohr
Complete_HQ	 → „normale“ Farbgebung
Complete_LQ	 → „normale“ Farbgebung, schwarzes Eselsohr
Web	 → normale Farbgebung, blaues Eck

Tabelle 2: Stati von LeafNodes

Einen Sonderfall gilt es noch zu erwähnen: Resources können auch auf ein Dokument im World Wide Web referenzieren, wobei derartige Dokumente nicht auf das mobile Gerät heruntergeladen werden. Vielmehr wird beim Anzeigen stets das WWW-Dokument direkt geöffnet, was natürlich eine bestehende Internetverbindung voraussetzt.

Nachdem nun bereits die verschiedenen Zustände der ResourceFiles, Resources und LeafNodes behandelt wurden, fehlen nur noch jene der FolderNodes. Objekte vom Typ „FolderNode“ sind bekanntlich Behälter einer beliebigen Anzahl an weiteren FolderNodes bzw. LeafNodes. Da eine Zustandsänderung eines untergeordneten FolderNodes bzw. LeafNodes ggf. zu einem Statuswechsel des FolderNodes führen muss, ist der übergeordnete FolderNode darüber entsprechend in Kenntnis zu setzen.

Tabelle 3 zeigt die verschiedenen Stati, die FolderNodes einnehmen können:








Zustand des FolderNodes	Icon
UNLOADED	 Das Verzeichnis enthält keine vollständigen LeafNodes. Derzeit findet auch kein Herunterladen statt. → graues Fach
UNLOADED_PARTDOWNLOAD	 Die LeafNodes des noch leeren Verzeichnisses werden gerade teilweise heruntergeladen. → grünes Fach, ein grünes Blatt
UNLOADED_DOWNLOAD	 Sämtliche LeafNodes des noch leeren Verzeichnisses werden gerade heruntergeladen. → grünes Fach, drei grüne Blätter
PARTDOWNLOADED	 Die LeafNodes des Verzeichnisses sind zum Teil vollständig. Derzeit findet kein Herunterladen statt. → gelbes Fach, ein weißes Blatt
PARTDOWNLOADED_PARTDOWNLOAD	 Die LeafNodes des Verzeichnisses sind zum Teil vollständig. Derzeit wird auch noch ein Teil der bislang nicht vollständigen LeafNodes heruntergeladen. → grünes Fach, ein weißes Blatt
PARTDOWNLOADED_DOWNLOAD	 Die LeafNodes des Verzeichnisses sind zum Teil vollständig. Derzeit werden auch noch alle bislang nicht geladenen LeafNodes heruntergeladen. → grünes Fach, ein weißes und zwei grüne Blätter
COMPLETE	 Sämtliche LeafNodes des Verzeichnisses sind vollständig. → gelbes Fach, drei weiße Blätter

Tabelle 3: Stati von FolderNodes

Nach diesem Ausflug in die Objektwelt des Manifest Viewers kommen wir nun wieder zurück zum Ausgangspunkt. Das Manifest ist mittlerweile verarbeitet, d.h. die laut Manifest benötigten Objekte wurden instanziiert. Nun wird dem Benutzer die Kursstruktur angezeigt, in der er nach Belieben navigieren kann (siehe Abbildung 56).

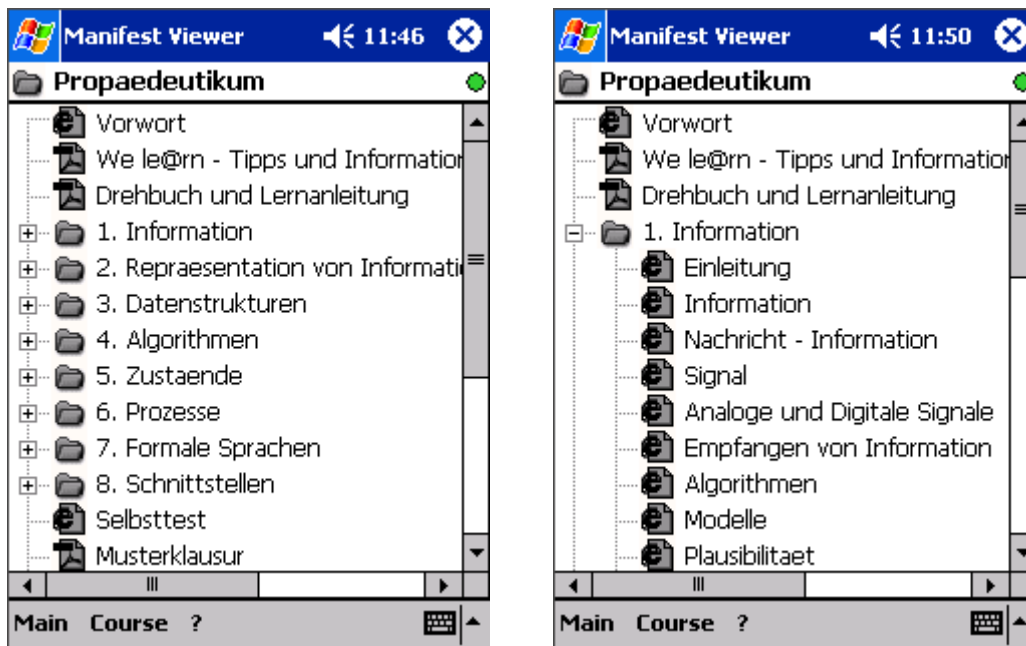


Abbildung 56: Kursstruktur im Manifest Viewer

Wie aus der Abbildung ersichtlich, besteht der Beispielkurs aus einigen Verzeichnissen (FolderNodes) und natürlich Dokumenten (LeafNodes). Durch sogenanntes „tap & hold“ (Stylus etwas länger gedrückt halten) erscheint ein Kontextmenü, das abhängig vom gewählten Baumknoten (FolderNode bzw. LeafNode) und dessen Zustand unterschiedliche Menüpunkte präsentiert (siehe Abbildung 57).



Abbildung 57: Kontextmenü für LeafNode

Da der angetippte Knoten in Abbildung 57 unvollständig ist (durch graues Symbol angezeigt), kann dessen Content bislang nur bei bestehender Internetverbindung angezeigt werden. D.h. das entsprechende HTML-Dokument wird direkt vom Kursserver geöffnet, ohne es lokal auf dem Handheld abzulegen. Will man Kursmaterialien jedoch auch offline sichten, müssen diese zuvor heruntergeladen werden. Dieses Herunterladen erfolgt durch Kommunikation mit dem Host, und zwar in jener Art und Weise, wie sie beim Manifest-Download bereits beschrieben wurde. Der Manifest Viewer fordert das gewünschte File über einen http-WebRequest beim Host an, liest die Bytes des Antwort-Streams und erstellt daraus lokal das entsprechende File.

Da im Falle des Herunterladens von Kursmaterialien mitunter eine große Anzahl an Dateien vom Server auf einen PDA zu transferieren ist, beinhaltet der Manifest Viewer einen Download Manager, der für die reibungslose Abwicklung dieses Download-Prozesses zuständig ist (siehe Abbildung 58).

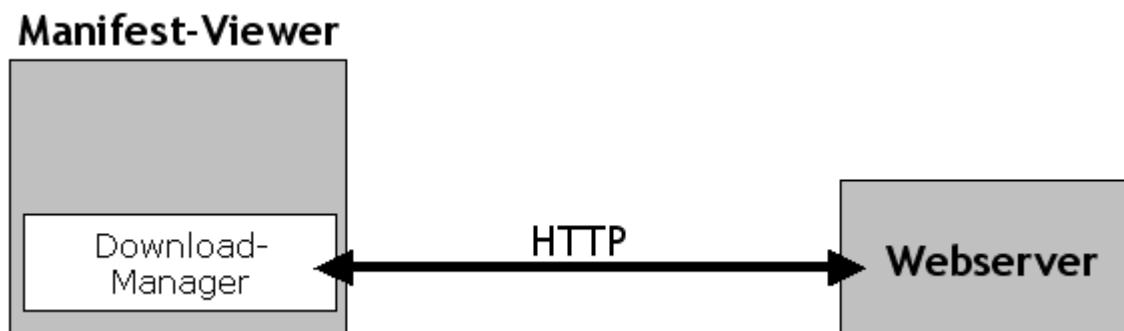


Abbildung 58: Kommunikation Manifest Viewer \leftrightarrow Webserver

Dieser Download Manager besteht einerseits aus einem theoretisch unbegrenztem Puffer, der die herunterzuladenden ResourceFiles aufnimmt. Daneben gibt es einen Download-Komponente, die im Eigentlichen für das Herunterladen zuständig ist. Diese Komponente – in Abbildung 59 als „http-Client“ bezeichnet – lädt der Reihe nach alle Dateien, die durch die ResourceFiles in der Warteschlange gegeben sind, auf das Handheld. Besteht also der Auftrag, ein ResourceFile – sprich die entsprechende Datei – vom Server herunterzuladen, wird dieses an die erste freie Position der Warteschlange gereiht. Sobald eine Datei vollständig heruntergeladen wurde oder aber der Vorgang des Herunterladens fehlschlug (z.B. weil die angeforderte Datei am Server nicht vorhanden ist), teilt der http-Client dies dem Download Manager mit. Dieser rückt nun sämtliche Elemente der Warteliste um eine Position nach

vorne und übergibt das nächste ResourceFile an den http-Client, um die entsprechende Datei herunterzuladen.

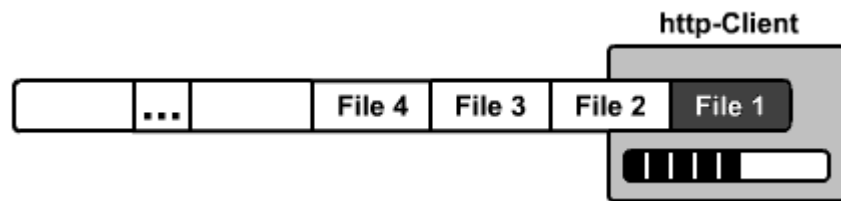


Abbildung 59: Download Manager

Zusätzlich informiert der http-Client den Download Manager darüber, wie viele Bytes der jeweiligen Datei insgesamt heruntergeladen wurden, sodass der Download Manager wiederum feststellen kann, wie viel in Summe bereits heruntergeladen wurde und wie viel noch ausständig ist (siehe Abbildung 60).

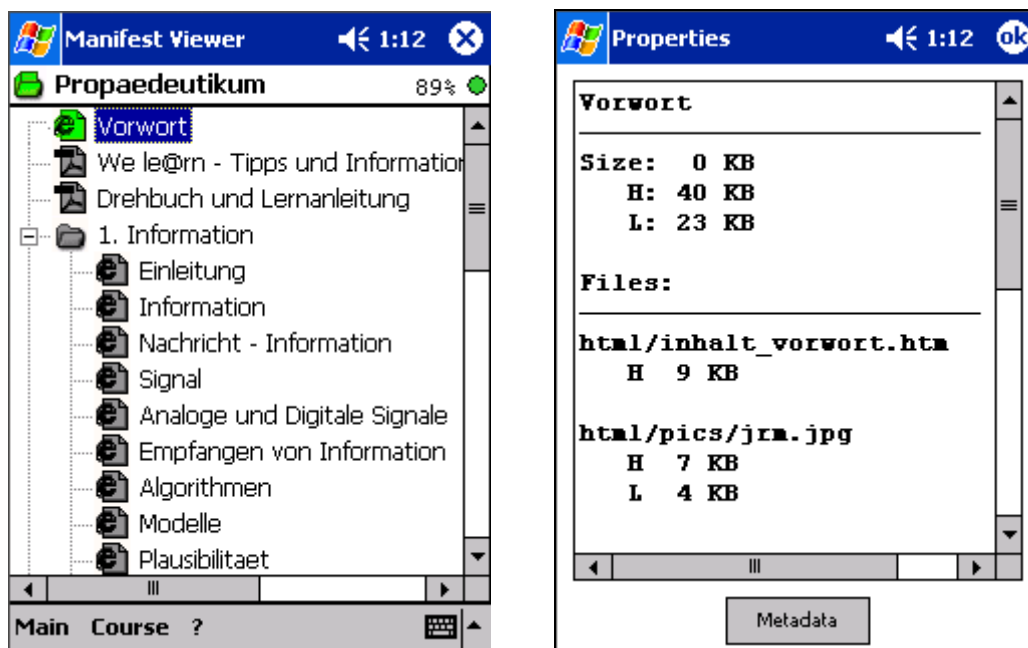


Abbildung 60: Downloadvorgang mit Fortschrittsanzeige und Properties-Dialog

Betrachten wir die Funktionsweise des Download Managers nochmals anhand eines konkreten Beispiels: Der Benutzer will den Knoten „Vorwort“ (vgl. Abbildung 60) herunterladen. Dazu wählt er im Kontextmenü „Download (HQ)“ bzw. „Download (LQ)“. Anschließend werden dem Download Manager sämtliche ResourceFiles des LeafNodes „Vorwort“ übergeben. Es handelt sich dabei um ein HTML-File sowie um jeweils zwei JPEG- und GIF-Grafiken. Da der http-Client derzeit nicht beschäftigt ist, wird der Dateipfad

des ersten ResourceFiles („html/inhalt_vorwort.htm“) sofort an diesen weitergereicht. Die restlichen vier ResourceFiles werden in der Warteschlange geparkt. Der http-Client fordert nun die erste Datei via http-WebRequest vom Server an. Anschließend liest er schrittweise die erhaltenen Bytes aus dem Response-Stream (Antwort des Servers) aus und teilt dem Download Manager unverzüglich mit, wie viel Bytes bereits eingelesen wurden. Das Abfragen dieser Fortschrittsangabe bleibt dem Manifest Viewer überlassen, welcher in regelmäßigen Abständen diese Informationen vom Download Manager bezieht und rechts oben – neben dem Icon für den Verbindungsstatus – anzeigt (siehe Abbildung 60). Nach vollständigem Übertragen der Datei sendet der http-Client an den Download Manager die Mitteilung, dass der Download erfolgreich abgeschlossen wurde. Der Download Manager übergibt daraufhin das erste ResourceFile aus der Warteschlange an den http-client und reiht die restlichen um eine Position nach vor. Darüber hinaus teilt der Download-Manager dem bereits verarbeiteten ResourceFile mit, dass seine Datei erfolgreich heruntergeladen werden konnte. Das entsprechende ResourceFile reagiert auf diese Nachricht dahingehend, dass es seinen Zustand auf „Complete_HQ“ ändert und alle ihre übergeordneten Resources davon in Kenntnis setzt. Dieses Procedere des Herunterladens – gefolgt von kettenreaktionsartigen Benachrichtigungen über Statuswechsel – wird nun so lange fortgesetzt, bis alle ResourceFiles abgearbeitet wurden, d.h. bis die Warteschlange leer ist.

Als wesentliche Aussage muss an dieser Stelle angemerkt werden, dass es möglich ist, nicht nur einen einzelnen LeafNode, sondern auch ganze FolderNodes – sprich eine Menge von LeafNodes – auf einmal herunterzuladen (siehe Abbildung 61). Für den Download Manager macht es nämlich keinerlei Unterschied, wie viele ResourceFiles gleichzeitig angefordert werden bzw. ob diese aus verschiedenen Resources (und folglich LeafNodes) stammen.

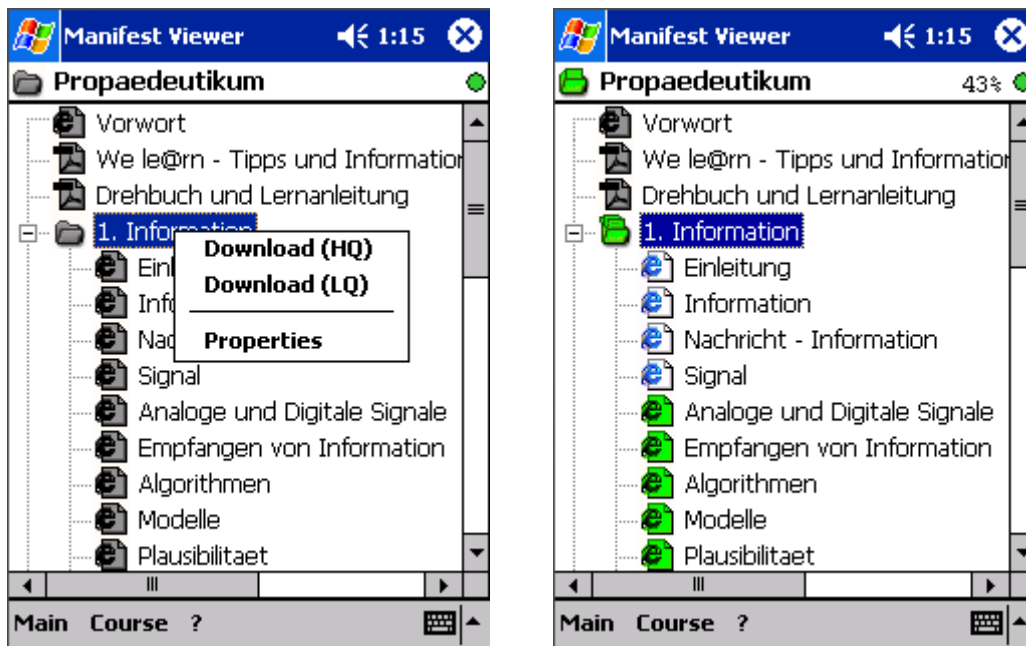


Abbildung 61: Download eines FolderNodes

Nach den Erläuterungen über das Herunterladen von Kursmaterial wird nun das vom Benutzer veranlasste Abbrechen eines Download-Vorgangs behandelt. Er wird beispielsweise davon Gebrauch machen wollen, falls irrtümlich ein falsches Verzeichnis für das Herunterladen gewählt wurde oder der Download aus Zeitmangel nicht mehr vollendet werden kann. Im Kontextmenü ist für diesen Fall der Menüpunkt „Cancel Downloading“ angeführt (siehe Abbildung 62). Wählt der Benutzer diese Option, wird sofort der Download Manager darüber informiert. Dieser fordert nun den http-Client auf, den aktuellen Download unverzüglich abubrechen. Der http-Client seinerseits folgt umgehend dieser Anweisung und stoppt den Download. Anschließend teilt er dem Download Manager mit, dass das Herunterladen der betreffenden Datei abgebrochen wurde. Der Download Manager, welcher inzwischen sämtliche ResourceFiles aus der Wareschlange entfernt hat, verständigt nun all diese ResourceFiles, dass deren Dateien nicht heruntergeladen wurden. Die ResourceFiles ändern darauf ihren Status von „Download_HQ“ bzw. „Download_LQ“ auf „Incomplete“ und setzen ihrerseits wiederum deren übergeordnete Resources davon in Kenntnis. Das bereits bekannte Weiterreichen von Events führt schlussendlich dazu, dass jene LeafNodes, die heruntergeladen werden sollten, vom Zustand „Download_HQ“ (bzw. „Download_LQ“) in den Ausgangszustand „Incomplete“ übergehen und die übergeordneten FolderNodes durch ein Event ihrerseits dazu veranlassen, deren eigenen Zustand neu zu berechnen („Unloaded“ oder „PartDownloaded“).

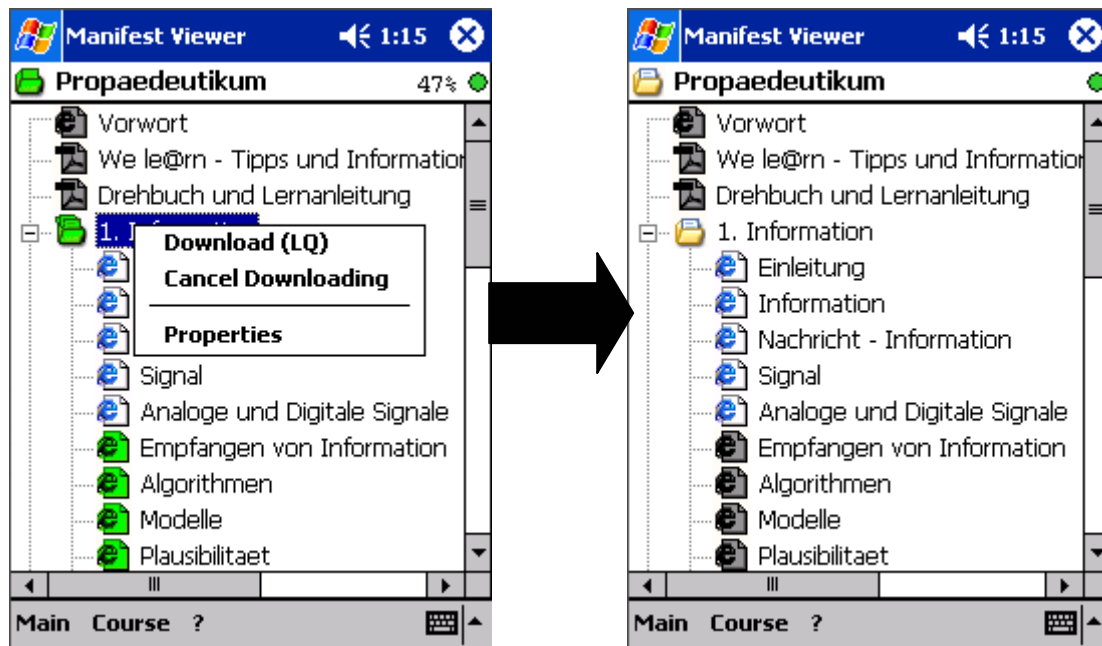


Abbildung 62: Abbruch des Herunterladens

Da auf PDA's – wie bereits mehrfach erwähnt – das Speicherangebot zumeist sehr begrenzt ist, muss der Anwender die Möglichkeit haben, entweder einen ganzen Kurs oder nicht mehr benötigte Kursteile wieder von seinem Gerät zu entfernen.

Das Löschen eines ganzen Kurses beinhaltet neben dem Löschen des Contents auch ein Entfernen des Manifests und des Konfigurationsfiles „cps_manifest.cfg“. Der Kurs wird somit rückstandslos vom mobilen Gerät entfernt. Erreicht wird dies, indem im Administrations-Dialog der zu löschende Kurs unter „Local Courses“ selektiert und anschließend auf die Taste „Delete“ getippt wird (siehe Abbildung 63).

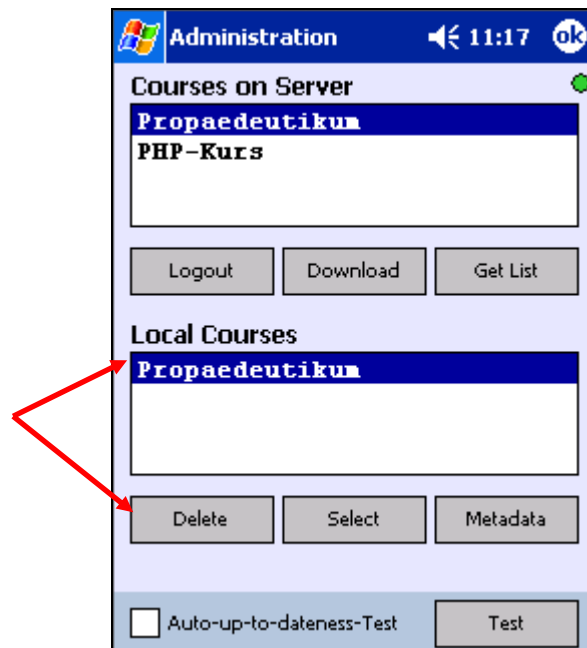


Abbildung 63: Löschen eines Kurses

Will der Benutzer jedoch lediglich einzelne Kursteile entfernen, so regelt er dies über den Strukturbaum (siehe Abbildung 64). Er führt dazu ein „tap & hold“ auf dem Knoten (FolderNode bzw. LeafNode), dessen Content gelöscht werden soll, aus. Danach wählt er den Menüpunkt „Delete“. Der Manifest Viewer löscht nun die nicht mehr länger benötigten Dateien vom Handheld.

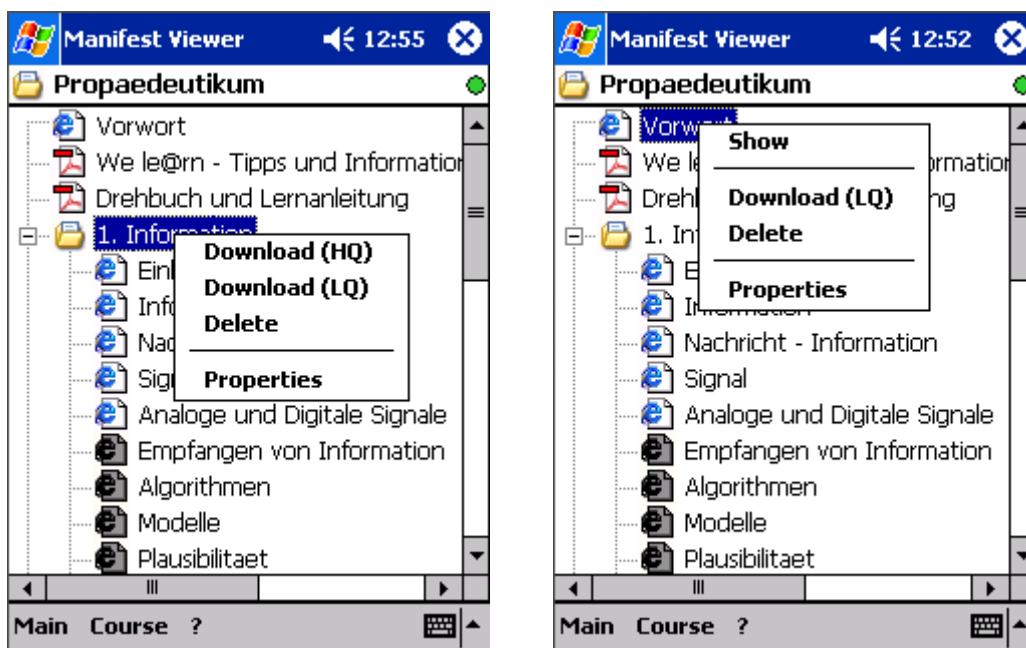


Abbildung 64: Löschen von Kursteilen

Dieses partielle Löschen klingt im ersten Moment einfacher als es in Wahrheit ist. Denn welche Dateien werden wirklich nicht mehr benötigt? Eine Datei kann schließlich ohne weiteres auch von einer anderen Resource benutzt werden. Ginge also der Manifest Viewer derart rigoros vor, einfach alle Dateien eines LeafNodes bzw. FolderNodes zu löschen, würde dies unter Umständen unerwünschte Auswirkungen auf andere LeafNodes haben. Man stelle sich beispielsweise vor, eine Bilddatei wird von zwei verschiedenen HTML-Dokumenten referenziert. Beide HTML-Dateien sind Bestandteil unterschiedlicher Resources und somit auch verschiedener LeafNodes. Wird nun einer der beiden LeafNodes gelöscht, würde durch vollständiges Löschen aller Files auch ein Teil des anderen LeafNodes verloren gehen, was bestimmt nicht im Sinne des Anwenders sein dürfte. Um diesem Problem Herr zu werden, verfolgt der Manifest Viewer die Strategie der „Complete Containers“, welche im folgenden Abschnitt erläutert wird.

Sämtliche ResourceFiles wissen genau, in wie vielen Resources sie enthalten sind. Diese Anzahl entspricht nämlich genau der Zahl jener Resources, die bei Statusänderungen zu benachrichtigen sind. Zur Speicherung dieses Wertes dient das Attribut „containers“. Damit die hier beschriebene Strategie erfolgreich angewandt werden kann, ist es darüber hinaus notwendig zu wissen, wie viele der übergeordneten Resources vollständig geladen sind. Dies wird gewährleistet, indem eine Resource im Falle des Statuswechsels auf „Complete_HQ“ bzw. „Complete_LQ“ all ihre ResourceFiles dazu veranlasst, die Werte ihrer Attribute „completeContainers“, welche die Zahl ihrer vollständigen Behälter (Resources) angeben, um eins zu erhöhen. Im entgegengesetzten Fall, d.h. wenn eine Resource den Zustand „Complete_HQ“ oder „Complete_LQ“ verlässt, werden die Werte der „completeContainers“-Attribute aller untergeordneten ResourceFiles dementsprechend um eins erniedrigt. Alle ResourceFiles können also über das Attribut „containers“ feststellen, in wie vielen Resources sie enthalten sind. Durch ihr Attribut „completeContainers“ lässt sich wiederum ermitteln, wie viele dieser übergeordneten Resources vollständig sind.

Wird nun eine vollständige Resource beauftragt, ihren Content zu löschen, werden die „completeContainers“-Zähler aller untergeordneten ResourceFiles um eins erniedrigt, da diese ab diesem Zeitpunkt in einer vollständigen Resource weniger enthalten sind. Anschließend wird für jedes ResourceFile der zu löschenden Resource überprüft, ob der Wert in „completeContainers“ gleich null ist. D.h. es wird ermittelt, ob es noch eine vollständige Resource gibt, die das jeweilige ResourceFile benötigt.

Ist dies der Fall, bleibt das ResourceFile unangetastet. Ist jedoch keine einzige übergeordnete Resource mehr vollständig, wird das physische File des ResourceFile-Objekts vom mobilen Gerät gelöscht.

Eine weitere wichtige Aufgabe des Manifest Viewers besteht darin, den Benutzer über etwaige serverseitige Änderungen eines Kurses zu informieren. Dazu werden Steuerelemente im unteren Bereich des Formulars „Administration“ zur Verfügung gestellt (siehe Abbildung 65).

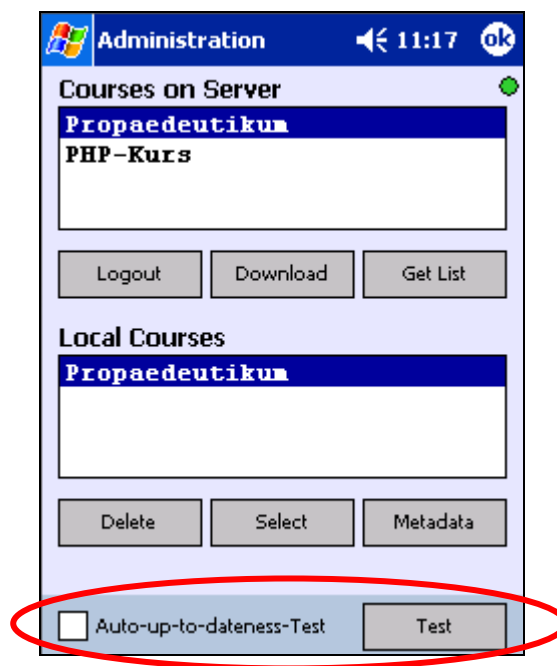


Abbildung 65: Überprüfung auf Aktualität eines Kurses

Ist das Auswahlfeld „Auto-up-to-dateness-Test“ selektiert, wird automatisch bei jedem Login (falls Kurs geöffnet) bzw. beim Öffnen eines Kurses (falls online) ermittelt, ob das Manifest des Kurses, das sich auf dem mobilen Gerät befindet, jenem auf dem Server entspricht. Durch Antippen der Taste „Test“ kann diese Überprüfung zu jedem beliebigen Zeitpunkt durchgeführt werden, sofern unter „Local Courses“ ein Kurs ausgewählt wurde und eine Verbindung zum Server besteht.

Folgende drei Szenarien können bei Überprüfung der Kursaktualität eintreten:

- (1) Die Struktur des Kurses sowie sämtliche Dateien des Contents auf dem Handheld sind aktuell.

- (2) Eine beliebige Anzahl an Dateien des Contents liegt auf dem Server in neueren Versionen vor, die Kursstruktur selbst ist jedoch unverändert.
- (3) Die Kursstruktur auf dem Server wurde geändert.

Szenario 1 besagt, dass die Kursstruktur auf dem Handheld jener auf dem Server gleicht. Darüber hinaus sind auch sämtliche auf das mobile Gerät heruntergeladene Dateien identisch mit jenen, die am Server angeboten werden. Nach erfolgter Aktualitätsprüfung wird dies durch folgenden Dialog aus Abbildung 66 vermittelt:

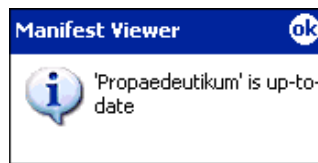


Abbildung 66: Kursstruktur und Content-Dateien sind aktuell

Tritt Szenario 2 ein, d.h. auf dem Server befinden sich Dateien in neueren Versionen, wird der Benutzer durch den Dialog aus Abbildung 67 darüber in Kenntnis gesetzt.

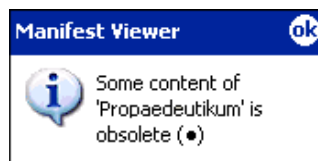


Abbildung 67: Dateien des Contents wurden geändert

Zusätzlich werden jene <item>-Knoten, die nicht mehr auf neuestem Stand befindliche Files enthalten, durch Anzeigen eines schwarzen Punktes in der Baumansicht entsprechend hervorgehoben (siehe Abbildung 68).

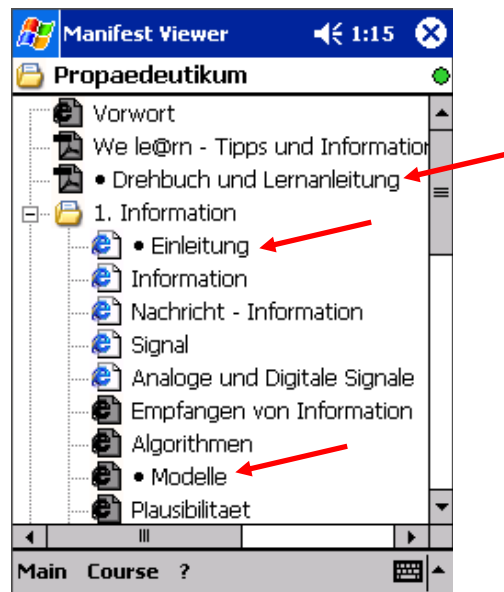


Abbildung 68: Kennzeichnung nicht mehr aktueller Knoten

Der Benutzer hat nun die Möglichkeit, jene gekennzeichneten Kursteile erneut vom Server herunterzuladen, um somit wieder auf Letztstand befindlichen Content auf dem Handheld durcharbeiten zu können. Grundsätzlich wird nach jedem erfolgtem Download im Bedarfsfall das lokale Manifest durch den Viewer entsprechend angepasst. D.h. sollte der Viewer feststellen, dass die Werte für Dateigröße bzw. Erstellungsdatum des übertragenen Files nicht mit jenen im jeweiligen <file>-Tag des lokalen Manifests übereinstimmen, so ändert er diese Attributwerte entsprechend. Bei erneuter Aktualitätsüberprüfung werden folglich neu heruntergeladene <item>-Knoten in der Baumstruktur nicht mehr mit dem Symbol „•“ versehen.

Wichtig dabei ist, dass die Struktur des Kurses – sprich die Anordnung der <item>-Elemente – in diesem Szenario völlig unverändert ist. Andernfalls tritt nämlich Szenario 3 ein, was dem Anwender durch Anzeige des Fensters aus Abbildung 69 mitgeteilt wird.

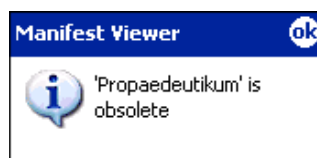


Abbildung 69: Kursstruktur wurde geändert

In diesem Fall kann dem Benutzer nicht mehr garantiert werden, dass alle Dateien, die er vom Server anfordert, auch übertragen werden können. Es ist beispielsweise denkbar, dass eine Ressource im neuen Kurs – sprich auf dem Server – nicht mehr vorhanden ist, da das

entsprechende <item> aus dem Manifest entfernt wurde. Eine weitere Möglichkeit, die das Auffinden einer Datei verhindern würde, ist eine Änderung des Ablageverzeichnisses dieser Datei auf dem Kursserver, was sich ebenfalls im neuen Manifest niederschlägt (<file>-Attribut „href“). Der Manifest Viewer würde in diesem Fall das File fälschlicherweise an der laut altem Manifest festgelegten Position vermuten. Die Konsequenz eines fehlgeschlagenen Download-Versuchs ist die Meldung des Manifest Viewers, dass das gewünschte File nicht übertragen werden konnte. Um dem vorzubeugen, kann es durchaus ratsam sein, den veralteten Kurs komplett vom Handheld zu entfernen und dessen Manifest bzw. den benötigten Content neuerlich herunterzuladen.

8. Literaturverweise

- [BRA00] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C. M., Maler, Eve: Extensible Markup Language (XML) 1.0 (2nd Edition).
WWW: <http://www.w3.org/TR/2000/REC-xml-20001006> (2000)
- [DIV02] Divotkey, R., Mühlbacher, J.R., Reisinger, S., Remplbauer, D.: The WeLearn Distance Teaching Framework. Granada: EDEN (European Distance Education Network), 2002 Eden Annual Conference, Open and Distance Learning in Europe and Beyond Rethinking International Co-operation, Conference Proceedings, 2002, 308-314 (also as SYSPRO 78/02, Mai 2002)
- [IMS] IMS Global Learning Consortium, Inc., Content Packaging Specification.
WWW: <http://www.imsglobal.org/content/packaging/index.html>
- [IMSa] IMS Global Learning Consortium, Inc., IMS Content Packaging Information Model, Version 1.1.3 Final Specification.
WWW:
http://www.imsglobal.org/content/packaging/cpv1p1p3/imscp_infov1p1p3.html
- [IMSB] IMS Global Learning Consortium, Inc., Learning Resource Meta-data Specification.
WWW: <http://www.imsglobal.org/metadata>
- [ISO79] ISO/IEC 11179, Information Technology -- Metadata Registries (MDR) (2006-03-23).
WWW: <http://metadata-standards.org/11179>
- [LED01] Leder, E.: E-Learning in der Wirtschaft am Beispiel von UBS, In Web-basiertes Lernen, Fortbildungsseminar in Informatik, Institut für Informatik, Universität Zürich, 2001
- [LOI03a] Loidl-Reisinger, S., Paramythis, A.: Distance Education - A Battlefield for Standards. In: András Szücs, Erwin Wagner, Costas Tsolakidis: The Quality Dialogue. Integrating Quality Cultures in Flexible, Distance and eLearning; Proceedings of the 2003 EDEN Annual Conference, Rhodes, 16-18.6.2003, 89-94

- [LOI03b] Loidl-Reisinger, S., Sonntag, M.: Using metadata in creating offline views of e-learning content. In: Michael E. Auer, Ursula Auer (Eds.): Interactive Computer Aided Learning. Learning Objects and Reusability of Content. Kassel: Kassel university press 2003 (ISBN 3-89958-029-X)
- [LOI05] Loidl, S., Mühlbacher, J. R., Schauer, H.: Preparatoy Knowledge: Propaedeutic in Informatics; In: Mittermeir, R. T. (Ed.): ISSEP 2005. Berlin: Springer 2005, 104-115 (LNCS 3422) (ISBN 3-540-25336-X)
- [LOM] IEEE Learning Technology Standards Committee (LTSC).
WWW: <http://ieeeltsc.org/wg12LOM>
- [LTS] IEEE Learning Technology Standards Committee (LTSC), LOM v1.0 Base Schema.
WWW: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf
- [LTS02] IEEE Learning Technology Standards Committee (LTSC), Learning Object Metadata Working Group, Draft Standard for Learning Object Metadata.
WWW:
http://ltsc.ieee.org/wg12/files/IEEE_1484_12_02_D01_LOM_11404_binding.doc (2003-06-20)
- [MAN01] Mandl, H.: Auf dem Weg zu einer neuen Kultur des Lehrens und Lernens?, In Web-basiertes Lernen, Fortbildungsseminar in Informatik, Institut für Informatik, Universität Zürich, 2001
- [MOB] MobiLearn.
WWW: <http://www.mobilearn.at/standards>
- [MSD] MSDN XML-Glossar.
WWW: <http://msdn2.microsoft.com/de-de/library/ms256452.aspx>
- [MUE98a] Jörg R.Mühlbacher, P.H.Leng, M.J.Shave, H.Schauer, R.M.Aiken - Interactive Seminars using the WEB: An International Experience. Band: "Teleteaching, Distance Learning, Training and Education. Proceedings of the XV. IFIP World Computer Congress. Wien: Budapest 1998
- [MUE98b] Jörg R. Mühlbacher - Das Ende des Präsenzunterrichts?; ÖZB - Österr. Zeitschrift für Berufsbildung, 4 -97/98, pp8-11; ISBN: 1029-5488 (1998)
- [MUE99] J.R.Mühlbacher, P.H.Leng, M.J.Shave, H.Schauer, R.M.Aiken - An Experiment in multinational collaborative learning and group work using the Internet. Education and Information Technologies. ISBN: 1360-2357

- [MUE02] Mühlbacher, J.R., Mühlbacher, S.C., Reisinger, S.: Learning Arrangements and Settings for Distance Teaching/Coaching/Learning: Best practice report. In: Hofer Christian, Chroust Gerhard (Ed.): IDIMT-2002. 10th Interdisciplinary Information Management Talks. Linz: Universitätsverlag Rudolf Trauner 2002 (also as SYSPRO 79/02, Juni 2002)
- [MUE03] Mühlbacher, J. R., Sonntag, M.: Roadmaps - Navigational Aids and Tools for Reusing Content in Distance Education. In: András Szücs, Erwin Wagner, Costas Tsolakidis: The Quality Dialogue. Integrating Quality Cultures in Flexible, Distance and eLearning. Proceedings of the 2003 EDEN Annual Conference, Rhodes, 16-18.6.2003, 73-78
- [MUE06] Mühlbacher, J.R., Putzinger A. (Hg.): WeLearn 2.3.2 - Premium Edition. (ISBN 3-85499-061-8)
- [PAR03] Paramythis, A., Loidl-Reisinger, S.: Adaptive Learning Environments and e-Learning Standards. In: Williams, R., w.w Associates (Eds.): Proceedings of the 2nd European Conference on e-Learning, Glasgow (Scotland), 6.-7.11.2003. Academic Conferences International Reading 2003, 369-379 (ISBN: 0-9544577-4-9)
- [PAR05] Paramythis, A., Loidl, S., Mühlbacher, J. R., Sonntag, M.: A Framework for Uniformly Visualizing and Interacting with Algorithms in E-Learning; In: IASTED Conference on Education and Technology (ICET) 2005, Calgary, 28 - 33 (ISBN 0-88986-487-X)
- [REI02] Reisinger, S.: Mobile intelligente Agenten als Wegweiser im Distance Teaching/Coaching/Learning. Dissertation. Johannes Kepler University Linz, 2002.
- [SCH01] Schauer, H.: Web-basiertes Lernen in Aus- und Weiterbildung – ein Paradigmenwechsel?, In Web-basiertes Lernen, Fortbildungsseminar in Informatik, Institut für Informatik, Universität Zürich, 2001
- [SCM01] Schulmeister, R.: Virtuelle Universität – Virtuelles Lernen. Mit einem Kapitel von Martin Wessner. – München, Wien: Oldenbourg. 2001
- [SCH96] Schröder, Hartmut & Krempl, Stefan (1996). Telelearning im Internet. WWW: <http://viadrina.euv-frankfurt-o.de/~sk/Virtual-College/frameVirtuCollege.html> (97-03-27)]

- [SON99a] Sonntag, M.: Teleteaching: From unidirectionalism to multidirectionalism. In: Hofer Susanne, Beneder Manfred (Ed.): IDIMT'99. 7th Interdisciplinary Information Management Talks. Linz: Universitätsverlag Rudolf Trauner 1999, 127-141 (ISBN 3-85487-046-9)
- [SON99b] Sonntag, M.: Teleteaching and Teamwork - A contradiction?. In: Online Educa Berlin. 5th International Conference on Technology Supported Learning. Bonn: I.W.H. Wie+Wo Verlag 1999, 142-146 (ISBN 3-92514-409-9)
- [SON03a] Sonntag, M., Loidl-Reisinger, S.: Cooperative Agent-Supported Learning with WeLearn. In: Gerhard Chroust, Christian Hofer (Eds.): Euromicro 2003. New Waves in System Architecture. Proceedings of the 29th Euromicro Conference. Los Alamitos, IEEE Computer Society 2003, 157-164 (ISBN 0-7695-1996-2)
- [SON03b] Sonntag, M.: Online Learning Platforms and E-Government. In: Ferdinand Galindo, Roland Traummüller (Eds.): E-Government: Legal, Technical and Pedagogical Aspects. Zaragoza, Seminario de Informatica y Derecho 2003, 283-299 (ISBN 84-95480-96-4)
- [SON04a] Sonntag, M.: Rechtsprobleme von Online Lernplattformen. Logging, Prüfung und Filterung von Inhalten. In: Schweighofer, E., Liebwald, D., Kreuzbauer, G., Menzel, T.: Informationstechnik in der juristischen Realität. Aktuelle Fragen zur Rechtsinformatik 2004. Wien: Verlag Österreich 2004, 407-414 (ISBN 3-7046-4481-1)
- [SON04b] Sonntag, M.: Metadata in E-Learning Applications: Automatic Extraction and Reuse. In: Hofer, C., Chroust, G. (Eds.): IDIMT-2004. 12th Interdisciplinary Information Management Talks, Budweis (Tschechien). Linz: Universitätsverlag Rudolf Trauner 2004, 219-231 (ISBN 3-85487-665-3)
- [SON04c] Sonntag, M.: Datenschutz im Fernunterricht. In: Plöckinger, O., Duursma, D., Mayrhofer, M. (Hrsg.): Internet-Recht. Wien: Neuer Wissenschaftlicher Verlag 2004, 455-474 (ISBN 3-7083-0169-2)
- [SZÜ03] András Szücs, Erwin Wagner, Costas Tsolakidis: The Quality Dialogue. Integrating Quality Cultures in Flexible, Distance and eLearning. Proceedings of the 2003 EDEN Annual Conference, Rhodes, 15-18.6.2003.

- [STR97] Strzebkowski, R. (1997). Realisierung von Interaktivität und multimedialen Präsentationstechniken (S. 269-304). In L.J. Issing & P. Klimsa (Hrsg.), Information und Lernen mit Multimedia. Weinheim: Psychologie Verlags Union.
- [W3L] Web-Plattform „LebensLanges Lernen im Web“ (W3L).
WWW: <http://www.w3l.de> (03-08-18)]
- [WEL] Web Environment for Learning (WeLearn).
WWW: <http://www.fim.uni-linz.ac.at/Research/WeLearn>
- [XML02] XML|XSL für Buch und Web: 2002, Christine Kränzler, Markt+Technik Verlag
- [XML98a] World Wide Web Consortium (W3C), XML 1.0 Recommendation 1998.
WWW: <http://www.w3.org/TR/1998/REC-xml-19980210> (10.02.98)
- [XML98b] World Wide Web Consortium (W3C), XML 1.0 (Second Edition) Recommendation 2000.
WWW: <http://www.w3.org/TR/2000/REC-xml-20001006>
- [ZHW02] Züricher Hochschule Winterthur (2002). Definition von eLearning.
WWW: <http://elearning.zhwin.ch/pool/definition> (03-05-11)

9. Abbildungsverzeichnis

Abbildung 1: Compac iPAQ 3970 und HP iPAQ hx 4700	15
Abbildung 2: Landscape-Format.....	16
Abbildung 3: Stufen einer Spezifikation bis zur W3C-Empfehlung.....	20
Abbildung 4: Struktur- und Inhaltsbeschreibung	22
Abbildung 5: Package Interchange File	25
Abbildung 6: Elemente eines Manifests.....	26
Abbildung 7: <manifest>-Element.....	27
Abbildung 8: <metadata>-Element	27
Abbildung 9: <organizations>-Element	30
Abbildung 10: <item>-Element	30
Abbildung 11: <resources>-Element	31
Abbildung 12: Submanifest(e)	32
Abbildung 13: Komponenten und Interaktion	33
Abbildung 14: Manifest Administrator	35
Abbildung 15: Komponenten des Manifest Administrators	36
Abbildung 16: Hauptfenster des Manifest Administrators	37
Abbildung 17: Manifestkonvertierung	38
Abbildung 18: Referenz auf HTML-Dokument.....	38
Abbildung 19: Referenzen innerhalb eines HTML-Dokuments	38
Abbildung 20: Platzhalter für nicht verfügbare Bilddatei.....	39
Abbildung 21: Vollständige Auflistung referenzierter Dateien	40
Abbildung 22: Zusätzliches File-Attribut - Dateigröße	41
Abbildung 23: Zusätzliches File-Attribut - Änderungsdatum.....	41
Abbildung 24: Kapitel- und Dokumentstruktur im Manifest Viewer	42
Abbildung 25: Vater-Item mit Referenz auf Ressource.....	42
Abbildung 26: Modifikation der Item-Struktur.....	42
Abbildung 27: Irrelevante Attribute	43
Abbildung 28: Unreferenzierte Resource.....	43
Abbildung 29: Zusätzliches File-Attribut für alternative JPEG-Datei.....	45
Abbildung 30: Modifizierte CPS-Elemente	46
Abbildung 31: An der Konvertierung beteiligte Komponenten.....	46

Abbildung 32: HTML-Quellcode mit GIF-Verweis	47
Abbildung 33: HTML-Quellcode mit GIF-Verweisen in JavaScript.....	48
Abbildung 34: HTML-Quellcode mit statischen Filereferenzen	48
Abbildung 35: Panel zur Einstellung der Komprimierungs-Eigenschaften.....	49
Abbildung 36: JPEG's in alternativen Qualitätsstufen	49
Abbildung 37: Beispiel einer Dateistruktur	50
Abbildung 38: Steuerelemente des Manifest Updaters	51
Abbildung 39: Ressource aus CPS Viewer Manifest.....	52
Abbildung 40: Resourcefiles vor dem Update.....	52
Abbildung 41: Resourcefiles nach dem Update	53
Abbildung 42: Dateireferenz mit alternativer Darstellung vor Aktualisierung.....	53
Abbildung 43: Dateireferenz mit alternativer Darstellung nach Aktualisierung	53
Abbildung 44: Splashscreen des Manifest Viewers.....	54
Abbildung 45: Login-Dialog des Manifest Viewers	55
Abbildung 46: Dialog für Proxy-Einstellungen	55
Abbildung 47: Kommunikationsüberprüfung Handheld \leftrightarrow Server	56
Abbildung 48: Auflistung der serverseitig verfügbaren Kurse	57
Abbildung 49: Anfordern eines Kurses vom Server	58
Abbildung 50: Anzeige der lokal verfügbaren Kurse	59
Abbildung 51: Umsetzung der Item-Hierarchie in Manifest Viewer-Struktur	60
Abbildung 52: Kursstruktur im CPS Viewer Manifest	61
Abbildung 53: Instanzierte Geschäftsobjekte	62
Abbildung 54: Zustandsdiagramm für ResourceFiles.....	63
Abbildung 55: Zustandsdiagramm für Resources.....	64
Abbildung 56: Kursstruktur im Manifest Viewer	67
Abbildung 57: Kontextmenü für LeafNode	67
Abbildung 58: Kommunikation Manifest Viewer \leftrightarrow Webserver	68
Abbildung 59: Download Manager.....	69
Abbildung 60: Downloadvorgang mit Fortschrittsanzeige und Properties-Dialog.....	69
Abbildung 61: Download eines FolderNodes	71
Abbildung 62: Abbruch des Herunterladens.....	72
Abbildung 63: Löschen eines Kurses.....	73
Abbildung 64: Löschen von Kursteilen	73

Abbildung 65: Überprüfung auf Aktualität eines Kurses.....	75
Abbildung 66: Kursstruktur und Content-Dateien sind aktuell	76
Abbildung 67: Dateien des Contents wurden geändert.....	76
Abbildung 68: Kennzeichnung nicht mehr aktueller Knoten.....	77
Abbildung 69: Kursstruktur wurde geändert.....	77

10. Tabellenverzeichnis

Tabelle 1: Verbindungsstati	56
Tabelle 2: Stati von LeafNodes	65
Tabelle 3: Stati von FolderNodes	66

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Pfaffing, am 12. März 2007

Curriculum Vitae

Name: Willibald Hötzingler
Geburtsdatum: 22. Mai 1976
Geburtsort: Vöcklabruck
Staatsbürgerschaft: Österreich
Religion: röm.-katholisch
Familienstand: ledig
Präsenzdienst: abgeleistet



Schulausbildung

1982 – 1986 Volksschule Vöcklamarkt
1986 – 1990 Bundesgymnasium Vöcklabruck
1990 – 1994 Bundesrealgymnasium Vöcklabruck
(Matura erfolgreich abgelegt am 8. Juni 1994)

Hochschulstudium

1995 – dato Johannes Kepler Universität (Studienrichtung Informatik)

Ferienjobs / Praktika / Studienbegleitende Tätigkeiten

1991 Tischlerei Dißlbacher (Neukirchen a. d. Vöckla)
1992 – 2002 Reinhalteverband Vöckla-Redl
2000 Tutorentätigkeit (Universität Linz - Softwareentwicklung)

Berufspraxis

1999 – 2004 Webdesign
2004 – dato Softwareentwicklung (STIWA Fertigungstechnik)

Fähigkeiten

Programmiersprachen: C#, Java, C, C++, Oberon, Pascal
Datenbanken: Oracle, MySQL, MS Access
Webdesign: HTML, Flash, DHTML, Javascript, PHP, ASP
Diverse Software: MS Office (Word, Excel, Powerpoint)
Weitere Kenntnisse: Pocket PC- und Palm-Programmierung

Fremdsprachen: Englisch (in Wort und Schrift)
Französisch (Grundkenntnisse)

Interessen

Sport: Faustball, Fußball, Tennis, Laufen
Lesen: Fachliteratur für Informatik

Anhang

Manifest Administrator – Benutzungsanleitung

▪ **Installation**

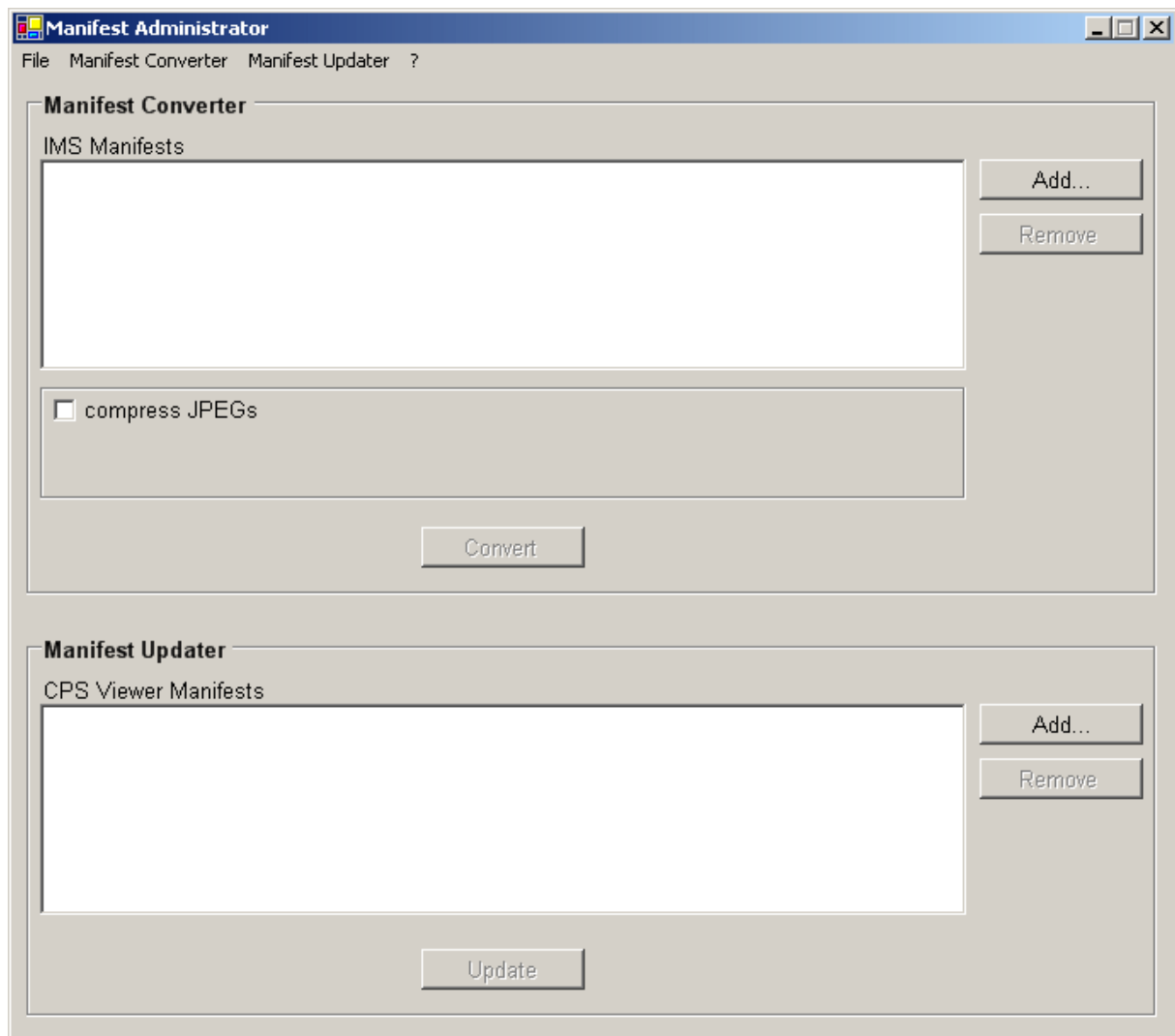
Für den Manifest Administrator ist keine Installation nötig.

▪ **Programmbedienung**

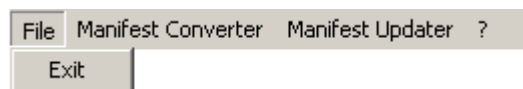
Der Manifest Administrator wird durch Ausführen der Datei „ManifestAdministrator.exe“ gestartet.



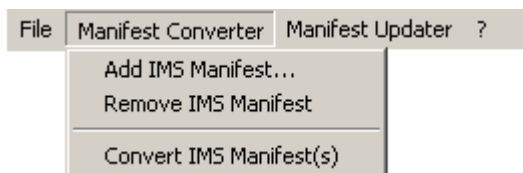
Nach Verschwinden des Splashscreens erscheint das Hauptfenster, welches sich – abgesehen von der Menüleiste – in zwei Bereiche teilt. Die Steuerelemente der oberen Hälfte gehören zum Manifest Converter, jene unten zum Manifest Updater.



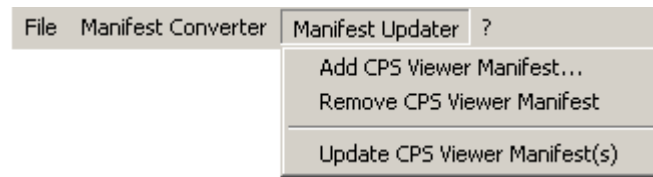
Die Menüleiste enthält die Hauptmenüs „File“, „Manifest Converter“, „Manifest Updater“ und „?“.
 Im Menü „File“ befindet sich lediglich der Menüpunkt „Exit“, durch welchen die Applikation beendet wird.



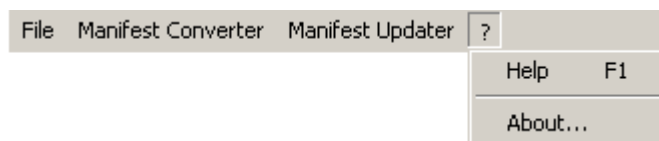
Das Menü „Manifest Converter“ beinhaltet die Menüpunkte „Add CPS Manifest...“, „Remove CPS Manifest“ und „Convert CPS Manifest(s)“.



Ähnlich sieht das Menü „Manifest Updater“ aus, welches die Menüpunkte „Add Viewer Manifest...“, „Remove Viewer Manifest“ sowie „Update Viewer Manifest(s)“ zur Verfügung stellt.



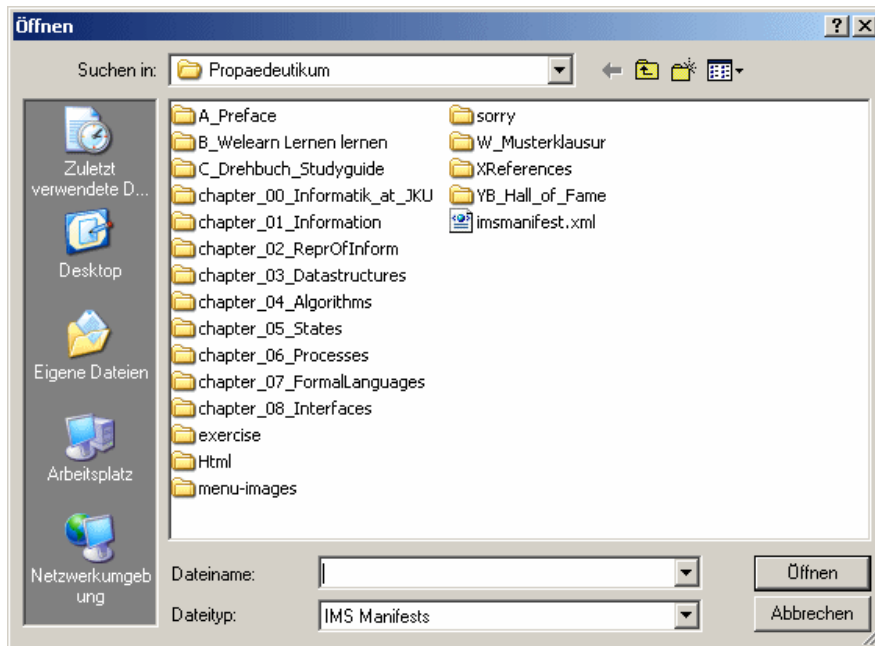
Das letzte Menü („?“) enthält die Menüpunkte „Help“ und „About...“. Über ersteren wird dem Benutzer eine Benutzeranleitung präsentiert, während im „About“-Dialog Informationen bezüglich Ersteller und Version des Softwareproduktes angeführt sind.



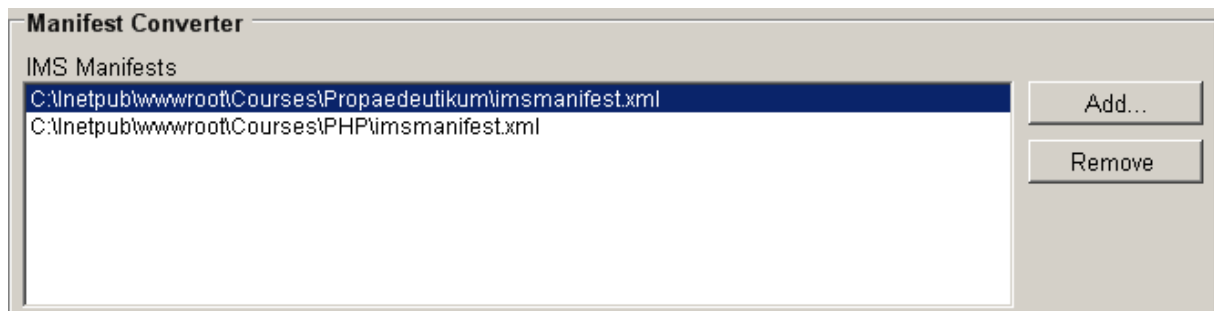
Da die wesentlichen Funktionalitäten in den Hauptmenüs auch über die bereits kurz angesprochenen Bereiche des Manifest Converters und des Manifest Updaters ausführbar sind, werden diese im Zuge der nun folgenden Beschreibung dieser beiden Hauptbereiche erläutert.

Manifest Converter

Der Manifest Converter dient dazu, CPS-Manifeste derart zu modifizieren, dass diese ein ordentliches Arbeiten mit dem Manifest Viewer erlauben. Durch Drücken der Taste „Add...“ bzw. des Menüpunktes „Add CPS-Manifest...“ im Menü „Manifest Converter“ kann eine beliebige Anzahl an CPS-Manifesten über einen Auswahl-Dialog selektiert werden, die in das erforderliche CPS Viewer-Format zu konvertieren sind.



Sämtliche ausgewählte CPS-Manifeste werden anschließend in einer Liste angezeigt. Mittels der Taste „Remove“ bzw. dem Menüpunkt „Remove CPS-Manifest“ kann das jeweils selektierte Manifest bei Bedarf übrigens wieder entfernt werden.



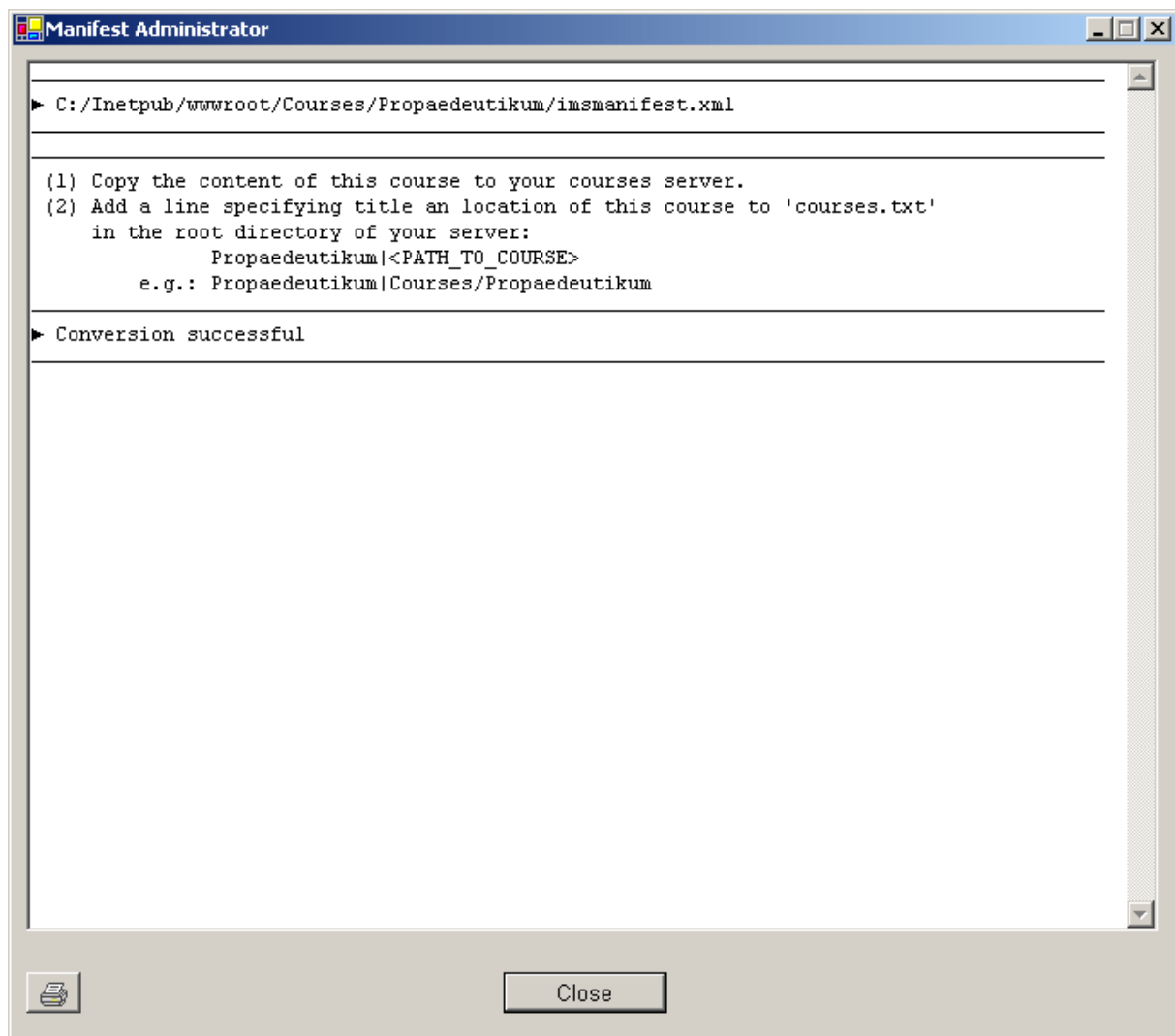
Da eLearning-Kurse oftmals viel Speicherplatz erfordern und dieser auf Handheld zumeist noch sehr knapp bemessen ist, hat der Benutzer die Möglichkeit, die enthaltenen JPEG-Grafiken zu komprimieren und diese neben der ursprünglichen Qualität auch in niedriger Qualität zur Verfügung zu stellen. Über einen entsprechenden Schieberegler kann der Komprimierungsgrad der JPEGs angegeben werden, wobei sich dieser verkehrt proportional zu Bildqualität und Speicherbedarf verhält.



Nachdem nun die zu adaptierenden CPS-Manifeste und gegebenenfalls der Komprimierungsgrad festgelegt wurden, kann die Konvertierung durch Drücken der Taste „Convert“ bzw. des Menüpunktes „Convert CPS-Manifest(s)“ gestartet werden.

Die Konvertierung kann je nach Rechnerleistung, Größe und Anzahl der eLearning-Kurse einige Sekunden benötigen. Anschließend erscheint ein Fenster, welches Auskunft über den Erfolg der Konvertierung eines jeden CPS-Manifests gibt. Darüber hinaus wird der Benutzer angewiesen, den Content der Kurse auf den Webserver zu kopieren und für jeden zu veröffentlichenden Kurs einen Eintrag in die im Root-Verzeichnis des Servers befindliche Datei „courses.txt“ zu machen. Dieser Eintrag enthält einerseits den Kurstitel, andererseits – durch einen senkrechten Strich („|“) getrennt – den Pfad des Kurses auf dem Server. Sind diese beiden Schritte erledigt, ist der Kurs veröffentlicht und kann mittels der Applikation „Manifest Viewer“ vom Server heruntergeladen werden.

Durch die links unten befindliche „Drucken“-Taste kann übrigens das Ergebnis des Konvertierungsprozesses auch ausgedruckt werden.

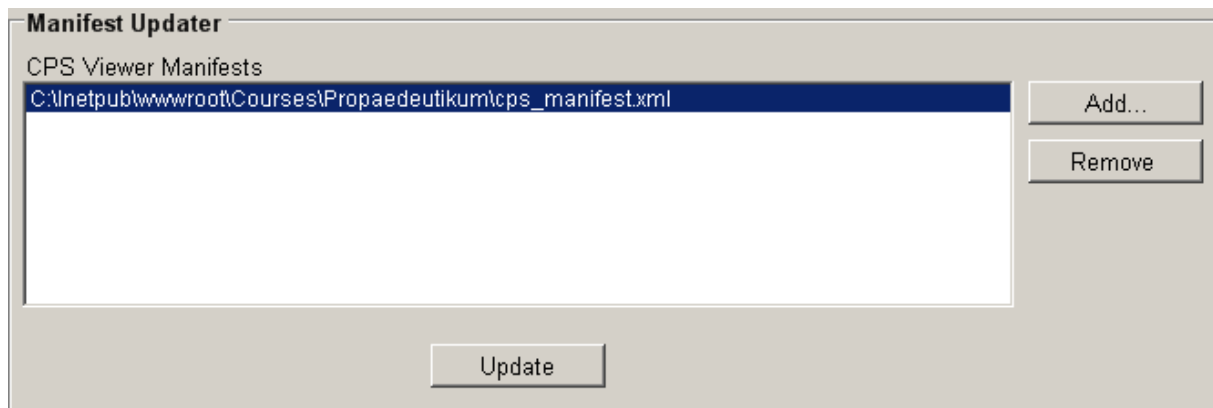


Nachdem das Konvertieren eines eLearning-Kurses behandelt wurde, wird im nächsten Abschnitt auf das Aktualisieren eines bereits konvertierten Kurses eingegangen, was in den Zuständigkeitsbereich des Manifest Updaters fällt.

Manifest Updater

Der Manifest Updater dient dazu, den Content von Manifesten, die mit dem Manifest Converter modifiziert wurden, zu aktualisieren. Der Manifest Updater prüft dabei für die aus dem Kursmanifest referenzierten Dateien, ob eventuell entsprechende neuere Versionen dieser Dateien vorhanden sind. Falls ja, werden die benötigten Fileinformationen (Dateigröße, Änderungsdatum) in das CPS Viewer Manifest übernommen. Wichtig anzumerken ist, dass an der Struktur des bestehenden CPS Viewer Manifests wird nichts verändert wird. Der Manifest Viewer erkennt folglich, dass es sich nicht um ein neues, sondern lediglich um ein aktualisiertes Manifest handelt. Dem Benutzer werden die neueren Ressourcen angezeigt, worauf er diese bei Bedarf auf seinem mobilen Gerät aktualisieren kann.

Wie bei Manifest Converter sind über einen entsprechenden Dateiauswahl-Dialog die auf den neuesten Stand zu bringenden CPS Viewer Manifeste zu wählen. Dieser Dialog kann entweder über die „Add...“-Taste im Bereich des Manifest Updaters bzw. über den Menüpunkt „Add Viewer Manifest...“ im Menü „Manifest Updater“ geöffnet werden. Die gewählten Manifeste werden in eine Liste eingetragen, aus welcher sie mit der Taste „Remove“ oder dem Menüpunkt „Remove Viewer Manifest“ auch wieder entfernt werden können.



Sind sämtliche zu aktualisierende CPS Viewer Manifeste aufgelistet, wird durch Drücken der Taste „Update“ bzw. des Menüpunktes „Update CPS Viewer Manifest“ der Update-Prozess gestartet.

Nach dessen Beendigung werden sämtliche Dateien angezeigt, die in neueren Versionen vorliegen und deshalb zu einer Änderung eines Attributwertes im Manifest führten.

Manifest Viewer – Bedienungsanleitung

▪ **Installation**

1. Setzen Sie den Pocket PC in die Docking Station ein.

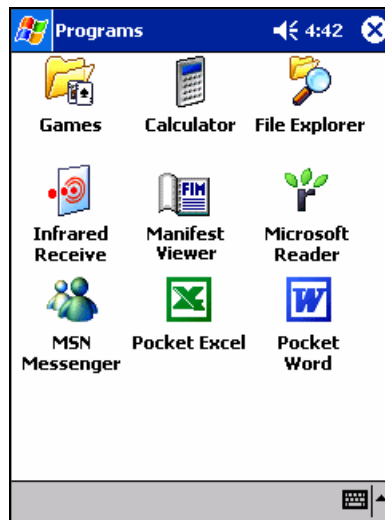
Besitzen Sie ein Handheld mit dem Betriebssystem Windows Mobile 2003, können Sie Schritt 2 überspringen, da .NET Compact Framework bei mobilen Geräten mit Windows Mobile 2003 bereits im ROM integriert ist.

2. Installieren Sie das .NET Compact Framework auf Ihrem PDA. Führen Sie dazu das Installationsfile „NETCFSetup.msi“ auf Ihrem Rechner aus.
3. Installieren Sie den Manifest Viewers mit Hilfe des Installationsfiles „ManifestViewerInstaller.msi“ auf Ihrem Rechner. Anschließend wird automatisch die Installation auf dem mobilen Gerät durchgeführt.
4. Auf Ihrem Handheld erscheint der Hinweis, dass für ein Funktionieren des Manifest Viewers auch noch das OpenNETCF Smart Device Framework erforderlich ist. Kopieren Sie folglich das – abhängig vom Prozessor Ihres mobilen Geräts – benötigte CAB-File mittels ActiveSync auf den PDA. Anschließend starten Sie dort die Installation durch antippen dieser CAB-Datei.

Die Installation ist nun abgeschlossen. Informationen über den Start und die Bedienung des Manifest Viewers erhalten Sie im nächsten Abschnitt.

▪ Programmbedienung

Um den Manifest Viewer zu starten, tippen Sie auf das entsprechende Icon in der Programmliste.

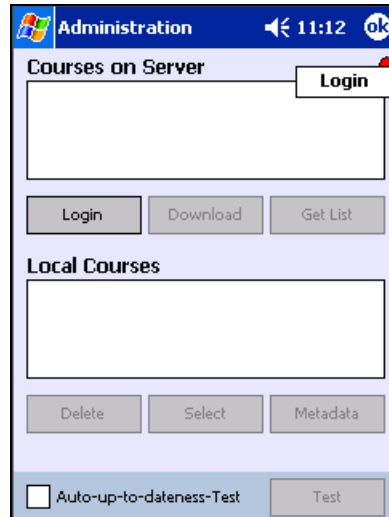


Unmittelbar nach dem Starten der Applikation erscheint ein Splashscreen, der nach kurzer Zeit automatisch verschwindet.



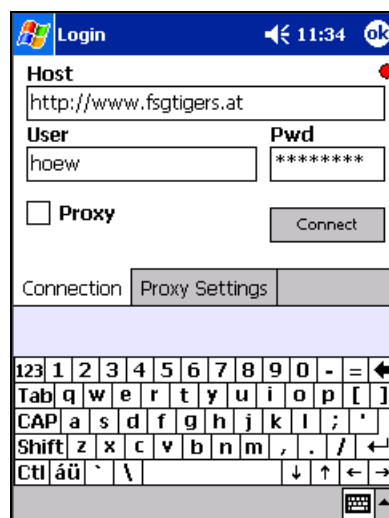
Nun wird der Administrations-Dialog geöffnet. Hier können Sie Kursmanifeste herunterladen, Kursinformationen (Metadaten) einsehen, Kurse öffnen, Kurse auf ihre Aktualität prüfen und diese gegebenenfalls auch wieder von Ihrem mobilen Gerät entfernen.

Zuvor müssen Sie aber erst eine Verbindung zu einem geeigneten Server aufbauen. Tippen Sie dazu entweder auf die „Login“-Taste oder auf den rot gefüllten Kreis rechts oben, der zur Anzeige des Verbindungsstatus dient.







Der „Login“-Dialog dient zum Herstellen einer Verbindung zu einem Server, der Kurse bereitstellt. Geben Sie dazu die benötigten Anmeldeinformationen in die jeweiligen Textfelder ein. Sollte die Verbindung über einen HTTP-Proxy laufen, wechseln Sie in den Reiter „Proxy Settings“ und geben dort die erforderlichen Proxy-Einstellungen an.

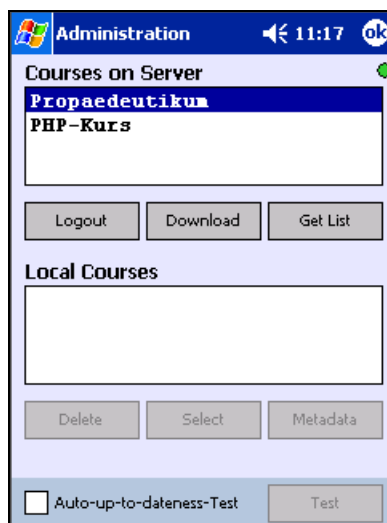
Sobald alle notwendigen Informationen eingegeben wurden, tippen Sie auf die Taste „Connect“. Nach erfolgreichem Verbindungsaufbau wird das „Login“-Fenster automatisch geschlossen und der Kreis zur Anzeige des Verbindungsstatus färbt sich grün.



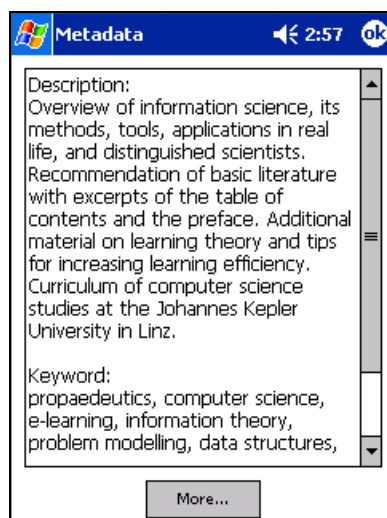
An dieser Stelle werden kurz die verschiedenen Verbindungsstati des Manifest Viewers erläutert:

-  ... Keine Verbindung zum Server hergestellt
-  ... Verbindung zum Server hergestellt
-  ... Verbindung zum Server unterbrochen
-  ... Versuch des Verbindungsaufbaues zum Server

Im oberen Listenfeld bekommen Sie nun sämtliche Kurse aufgelistet, die auf dem von Ihnen gewählten Server angeboten werden. Diese Auflistung kann durch Tippen auf die Taste „Get List“ übrigens jederzeit aktualisiert werden. Nachdem Sie den gewünschten Kurs selektiert haben, wird dessen Manifest durch Tippen auf „Download“ auf das mobile Gerät geladen.



Im unteren Listenfeld stehen alle Kurse, die bereits auf Ihrem mobilen Gerät vorhanden sind. Sie können sich hier zu jedem Kurs dessen Metadaten anzeigen lassen (Taste „Metadata“).

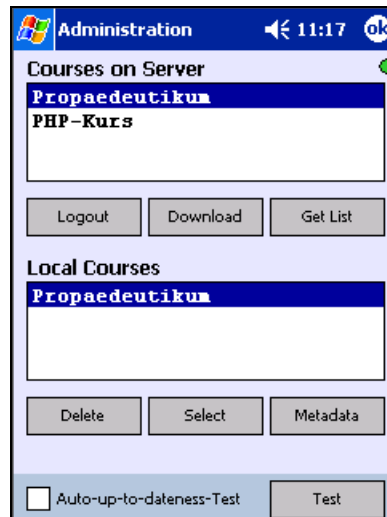


Genauso ist es auch möglich, einen Kurs wieder vollständig (Manifest und Content) von Ihrem Gerät zu entfernen (Taste „Delete“).

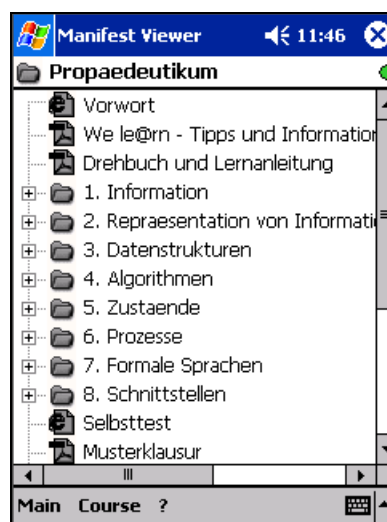
Da eLearning-Kurse oftmals Veränderungen erfahren, können Sie durch Tippen auf die Taste „Test“ feststellen

lassen, ob bzw. inwieweit sich Ihr heruntergeladener Kurs vom entsprechenden Kurs am Server unterscheidet. Selektieren Sie das Auswahlfeld links unten („Auto-up-to-dateness-Test“), wird dieser Aktualitäts-Vergleich automatisch bei jedem Login (falls Kurs geöffnet) bzw. dem Öffnen eines Kurses (falls online) durchgeführt. Dokumentknoten, die nicht mehr auf Letztstand sind, werden im Baum durch einen schwarzen Punkt (●) gekennzeichnet.

Haben Sie den gewünschten Kurs im unteren Listenfeld ausgewählt, können Sie diesen durch Tippen auf die Taste „Select“ öffnen.



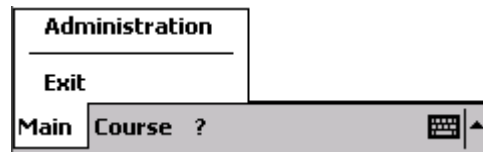
Nach dem Aufbau der Kursstruktur, der abhängig von der Prozessorleistung und der Größe des Kurses doch einige Zeit in Anspruch nehmen kann, haben Sie nun die Möglichkeit, sich verschiedenste Informationen (Speicherbedarf, Metadaten usw.) zu beliebigen Kursteilen anzeigen zu lassen. Sie können aber beispielsweise auch Kursmaterialien Herunterladen bzw. Löschen.



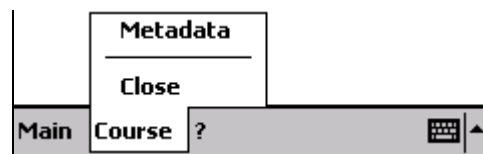
Grundsätzlich werden Aktionen entweder über die Menüleiste (unten) bzw. das Kontextmenü (Stylus länger gedrückt halten) ausgelöst.

Das Hauptmenü besteht aus drei Bereichen: „Main“, „Course“ und „?“.

Unter „Main“ kann einerseits der bereits bekannte Administrations-Dialog („Administration“) geöffnet werden. Andererseits kann hier auch die Applikation beendet werden („Exit“) - und zwar wirklich beendet und nicht nur minimiert, wie es bei Tippen auf das Kreuz rechts oben geschieht.



Der im Menü „Course“ angeführte Menüpunkte „Metadata“ ist im Grunde bereits im Administrations-Dialog enthalten, hier jedoch für den Benutzer schnell und einfach erreichbar. Mit „Close“ kann der aktuell geöffnete Kurs geschlossen werden.



Im letzten Menüpunkt „?“ gelangen sie zur Hilfe („Help“) und zum „About“-Dialog, der Informationen bezüglich Version, Auftraggeber und Ersteller der Applikation bereitstellt.



Neben dem Hauptmenü nimmt wie bereits erwähnt auch das Kontextmenü eine zentrale Rolle im Absetzen von Aktionen ein. Das Kontextmenü erscheint bei längerem Drücken des Stylus auf einen Knoten des Strukturbaums und bietet verschiedenste Menüpunkte an.



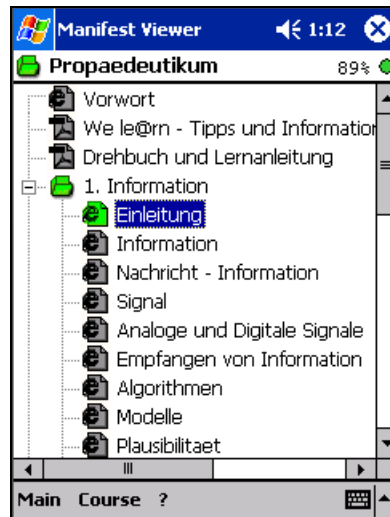
Folgende Punkte können im Kontextmenü je nach Art und Zustand des gewählten Knotens angeführt werden:

- Show: Öffnen des Dokuments
- Download (HQ): Herunterladen der Datei(en) in hoher Qualität
- Download (LQ): Herunterladen der Datei(en) in niedriger Qualität
- Cancel Downloading: Abbrechen des Herunterlade-Vorgangs
- Delete: Löschen der Dateien des Knotens
- Properties: Anzeigen von Knoten-Informationen

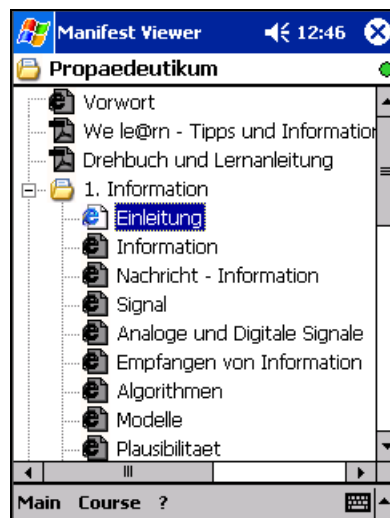
Um die Datei(en) eines einzelnen Dokumentknotens auf Ihr mobiles Gerät zu laden, wählen Sie „Download (HQ)“ (bzw. falls vorhanden „Download (LQ)“) aus dem Kontextmenü dieses Knotens. Dazu muss natürlich eine Verbindung zu dem entsprechenden Server bestehen (grüner Kreis).









Rechts oben sehen Sie nun eine Fortschrittsanzeige des Herunterlade-Vorgangs. Darüber hinaus ist festzustellen, dass sich der jeweilige Knoten und dessen übergeordnete Verzeichnisse grün einfärben, was Auskunft darüber gibt, welche Knoten gerade heruntergeladen werden.



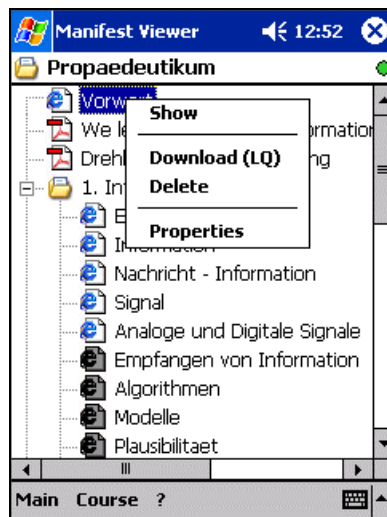
Nach erfolgreicher Beendigung des Herunterladens wird das Bildsymbol des Knotens in bekannter Art farblich dargestellt.



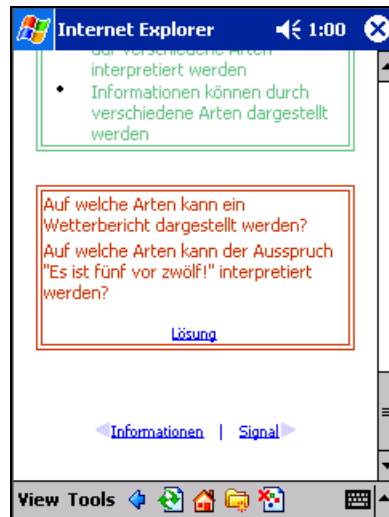
Ein Dokumentknoten kann also verschiedenste Stati einnehmen, welche in folgender Auflistung angeführt sind. Auch wenn zur Veranschaulichung hier nur der HTML-Dokumentknoten verwendet wird, lässt sich das Schema der farblichen Veränderung auch auf alle anderen Dokumenttypen übertragen.

-  Der Dokumentknoten ist nicht heruntergeladen.
-  Der Dokumentknoten wird gerade in hoher Qualität heruntergeladen.
-  Der Dokumentknoten wird gerade in niedriger Qualität heruntergeladen (schwarzes Eck rechts oben).
-  Der Dokumentknoten ist in hoher Qualität heruntergeladen.
-  Der Dokumentknoten ist in niedriger Qualität heruntergeladen (schwarzes Eck rechts oben).
-  Der Dokumentknoten verweist auf ein Dokument im WWW (blaues Eck rechts oben). Das Dokument kann folglich nur bei bestehender Internetverbindung geöffnet werden.

Ist ein Dokumentknoten vollständig geladen (bzw. es handelt sich um ein Dokument im WWW bei bestehender Internetverbindung), kann das entsprechende Dokument entweder über das Kontextmenü („Show“) oder durch einfaches Tippen auf den Knoten geöffnet werden.



Zum Anzeigen des Dokuments wird der standardmäßig für den jeweiligen Dokumenttyp definierte Viewer verwendet. Da es sich in diesem Beispiel um ein HTML-Dokument handelt, wird der Internet Explorer zur Visualisierung herangezogen.



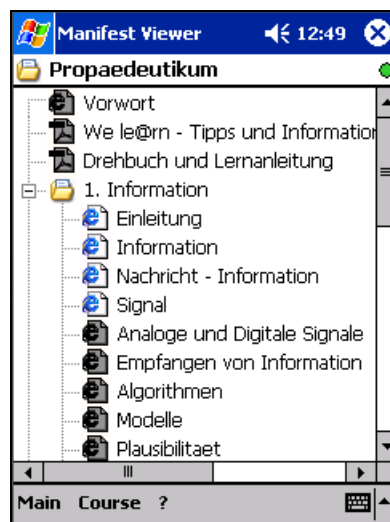
Es ist natürlich auch möglich, Inhalte ganzer Verzeichnisse auf einmal herunterzuladen. Dazu wird wie gehabt über das Kontextmenü des gewünschten Verzeichnisses der Herunterlade-Vorgang in Gang gesetzt.










Wird nun im Zuge des Herunterladens abermals das Kontextmenü aufgerufen, kann festgestellt werden, dass dieses nun teilweise andere Menüpunkte enthält.



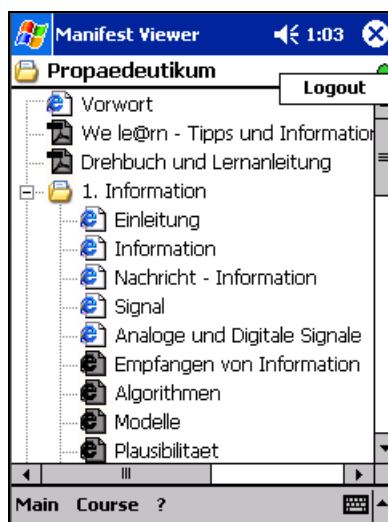
Durch Tippen auf „Cancel Downloading“ kann der Ladevorgang auf der Stelle abgebrochen werden.



Ähnlich der oben bereits behandelten Stati der Dokumentknoten werden nun auch die verschiedenen Stati der Verzeichnis-Knoten angeführt.

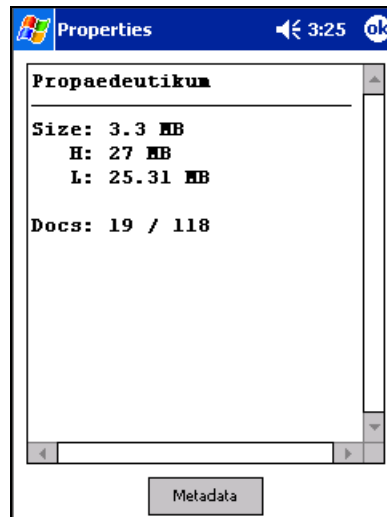
-  Das Verzeichnis enthält keine heruntergeladenen Inhalte, ist also leer. Derzeit findet kein Herunterladen statt (graues Fach).
-  Die Inhalte des noch leeren Verzeichnisses werden gerade teilweise heruntergeladen (grünes Fach, ein grünes Blatt).
-  Die Inhalte des noch leeren Verzeichnisses werden gerade zur Gänze heruntergeladen (grünes Fach, drei grüne Blätter).
-  Die Inhalte des Verzeichnisses sind teilweise heruntergeladen. Derzeit findet kein Herunterladen statt (gelbes Fach, ein weißes Blatt).
-  Die Inhalte des Verzeichnisses sind teilweise heruntergeladen. Derzeit wird auch noch ein Teil der bislang nicht geladenen Inhalte heruntergeladen (grünes Fach, ein weißes Blatt).
-  Die Inhalte des Verzeichnisses sind teilweise heruntergeladen. Derzeit werden auch noch alle bislang nicht geladenen Inhalte heruntergeladen (grünes Fach, ein weißes und zwei grüne Blätter).
-  Sämtliche Inhalte des Verzeichnisses sind heruntergeladen (gelbes Fach, drei weiße Blätter).

Wurden alle gewünschten Inhalte heruntergeladen, kann die Verbindung zum Server beendet werden, um jeglichen unnötigen Datentransfer (Ermitteln des Verbindungsstatus zum Server durch Ping) zu unterbinden. Dies kann wiederum entweder im Administrations-Dialog oder durch Tippen auf die Anzeige des Verbindungsstatus (grüner Kreis) erreicht werden.

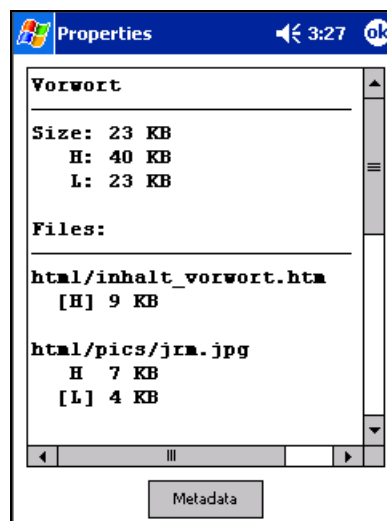


Wie weiter oben bereits erwähnt, können über dem Menüpunkt „Properties“ Informationen zum gewählten Knoten angezeigt werden. Handelt es sich um dabei um ein Verzeichnis, werden folgende Informationen angeführt:

- Size: aktueller Speicherbedarf
- H: Speicherbedarf gesamt, falls sämtliche Inhalte in hoher Qualität vorliegen
- L: Speicherbedarf gesamt, falls sämtliche Inhalte in niedriger Qualität vorliegen
- Docs: Heruntergeladene Dokumentknoten / Dokumentknoten gesamt



Im Falle von Dokumentknoten werden neben den Angaben zum Speicherbedarf auch noch alle Dateien angeführt, die zur Anzeige des Dokuments benötigt werden (bei HTML-Dokument beispielsweise GIF- und JPG-Bilddateien). Darüber hinaus wird für jede solche Datei auch deren Speicherbedarf angezeigt, und – falls diese heruntergeladen ist – durch eine eckige Umklammerung ([H], [L]) angegeben, ob diese in hoher oder niedriger Qualität auf dem mobilen Gerät vorliegt.



Nun sollte der Benutzung des Manifest Viewers nichts mehr im Wege stehen! Deshalb noch viel Spaß und Erfolg beim Lernen auf Ihrem mobilen Gerät!

Sonstige Hinweise

- **Umschalten zwischen Portrait- und Landscape-Modus**

Der Manifest Viewer unterstützt auch den sogenannten „Landscape“-Modus. Das Display des mobilen Geräts wird dabei von Hoch- auf Querformat umgestellt. Sie benötigen dafür jedoch ein Handheld, der mit dem Betriebssystem Windows Mobile 2003 ausgestattet ist.

Setzen Sie die Ausrichtung im Fenster „Touch-screen“, welches über „Start“ - Einstellungen“ – Reiter „System“ – „Touch-screen“ geöffnet wird.

Auf Geräten der Serie HP iPAQ hx4700 können Sie den Anzeigemodus auch durch gleichzeitiges, ca. drei Sekunden langes Drücken der Tasten „Kalender“ und „iTask“ auf der Vorderseite des mobilen Geräts wechseln.

- **Adobe Reader für Pocket PC**

Da Kurse unter anderem auch PDF-Dokumente enthalten, benötigen Sie dafür einen geeigneten Viewer. Unter „<http://www.adobe.com/products/acrobat/readerforppc.html>“ können Sie den Adobe Reader für Pocket PC kostenlos herunterladen.