



Cooperation & Management
Institut für Telematik
Fakultät für Informatik
Universität Karlsruhe (TH)



MODELLIERUNG UND IMPLEMENTIERUNG EINES AUSKUNFTDIENSTES ÜBER SCHULUNGSMATERIALIEN FÜR DIE AUS- UND WEITERBILDUNG

Diplomarbeit
Matthias Bonn

Verantwortlicher Betreuer: Prof. Dr. Sebastian Abeck
Betreuender Mitarbeiter: Dipl. Inform. Dirk Feuerhelm

01.11.2001 – 01.05.2002

Ehrenwörtliche Erklärung

Ich erkläre hiermit, die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Kronau, 1. Mai 2002

Matthias Bonn

Danksagung

Allen Mitgliedern der Arbeitsgruppe Cooperation & Management möchte ich an dieser Stelle herzlich für ihre Unterstützung danken.

Erwähnen möchte ich besonders Herrn Feuerhelm, dessen Anregungen und Tipps maßgeblich zum Gelingen der Arbeit beitrugen, und Herrn Professor Abeck, der das nötige Vertrauen in die entstandene Software hatte und sie trotz ihres Prototypenstatus in seinen Lehrveranstaltungen einsetzt.

INHALT

1	EINFÜHRUNG	1
1.1	MOTIVATION.....	1
1.2	PROBLEMSTELLUNG.....	2
1.3	ERGEBNISSE.....	3
1.4	ÜBERBLICK.....	4
2	STAND DER TECHNIK	5
2.1	ARCHITEKTUREN.....	5
2.1.1	<i>Komponentenbasierte Softwareentwicklung</i>	5
2.1.2	<i>Schichtenarchitektur und Verteilte Systeme</i>	6
2.2	ELEKTRONISCHE TAFEL.....	7
2.2.1	<i>Plasmabildschirm</i>	8
2.2.2	<i>Graphiktablett</i>	8
2.3	E-LEARNING-UMGEBUNGEN.....	9
2.3.1	<i>E-Kreide</i>	9
2.3.2	<i>Cisco e-Learning Architektur</i>	10
2.3.3	<i>Wertung</i>	11
3	KONZEPTION	12
3.1	EINORDNUNG IN ED.TEC.....	12
3.1.1	<i>Erstellung und Aufbereitung</i>	13
3.1.2	<i>Anpassung und Bereitstellung</i>	13
3.1.3	<i>Vermittlung</i>	14
3.1.4	<i>Aneignung</i>	15
3.2	ANFORDERUNGEN.....	16
3.2.1	<i>Einheitliche und zentrale Speicherung der Schulungsmaterialien</i>	16
3.2.2	<i>Transparente Unterstützung der eingesetzten Technologien</i>	17
3.2.3	<i>Bereitstellung einer graphischen Dozentenoberfläche</i>	17
3.2.4	<i>Durchführen von Präsenz- und Onlineveranstaltungen</i>	18
3.3	ENTWURF.....	18
4	ARCHITEKTUR	24
4.1	DIE DATENHALTUNGS-SCHICHT.....	24
4.1.1	<i>Verzeichnisse</i>	25
4.1.2	<i>Dateien</i>	25
4.1.3	<i>Metadaten</i>	26
4.2	DIE GESCHÄFTSLOGIK-SCHICHT.....	27
4.3	DIE PRÄSENTATIONSLOGIK-SCHICHT.....	31
4.4	ZUSAMMENSPIEL DER SCHICHTEN.....	34
5	BETRIEBSKONZEPT.....	36
5.1	ÜBERSICHTLICHE VERWALTUNG DER SCHULUNGSMATERIALIEN.....	37
5.2	TRANSPARENTER NUTZUNG DER PRÄSENTATIONSWERKZEUGE.....	38
5.3	PERMANENTER ÜBERBLICK ÜBER DEN ABLAUF DER SCHULUNG.....	40
5.4	UNTERSTÜTZUNG DER REMOTE-VERANSTALTUNG.....	41
6	IMPLEMENTIERUNG	42
6.1	DER CONTENT INFORMATION SERVER (CIS).....	42
6.1.1	<i>Schnittstellenbeschreibung</i>	43
6.1.2	<i>Arbeitsweise und Datenmodell</i>	47
6.1.3	<i>Verwendung des CIS</i>	50
6.2	DER CONTENT INFORMATION BROWSER (CIB).....	51
6.2.1	<i>Schnittstellenbeschreibung</i>	52
6.2.2	<i>Arbeitsweise und Verwendung</i>	55
6.3	DIE DOZENTENUMGEBUNG.....	56
6.3.1	<i>frmButtons.frm</i>	57
6.3.2	<i>frmEdDocSup.frm</i>	57
6.3.3	<i>Docking.bas</i>	58

6.3.4	<i>modBrowseForFolder.bas</i>	58
6.3.5	<i>modMain.bas</i>	58
6.3.6	<i>clsCIB.cls</i>	58
6.3.7	<i>clsStyle.cls</i>	58
7	AUSBLICK	59
7.1	FUNKTIONALITÄT.....	59
7.2	MICROSOFT .NET	59
8	ANHANG	61
8.1	LITERATUR.....	61
8.2	MANAGEMENT SUMMARY	62

ABBILDUNGEN UND TABELLEN

Abbildung 1.1: Rollen, Komponenten und Prozesse in der Ausbildung	1
Abbildung 2.1: Beispiel eines Plasmabildschirms	8
Abbildung 2.2: Beispiel eines Graphiktablets.....	8
Abbildung 2.3: E-Kreide	9
Abbildung 2.4: E-Learning Architektur von Cisco.....	10
Abbildung 3.1: Aufgabenbereiche der internetbasierten Wissensvermittlung.....	12
Abbildung 3.2: Internetbasierte Wissensvermittlung	16
Abbildung 3.3: UML Anwendungsfall „Schulung durchführen“	18
Abbildung 3.4: UML Anwendungsfall „Material vorbereiten“.....	19
Abbildung 3.5: UML Anwendungsfall „Material präsentieren“	19
Abbildung 3.6: Abstraktes UML Klassendiagramm.....	20
Abbildung 3.7: UML Sequenzdiagramm des Anwendungsfalls „Material sichten“.....	20
Abbildung 3.8: Zustandsübergänge der Klasse „Dozentenumgebung“	21
Abbildung 3.9: UML Aktivitätsdiagramm der Dozentenumgebung.....	23
Abbildung 4.1: 3-Schichten-Architektur	24
Abbildung 4.2: Ausdrucksmöglichkeiten im Dateipräfix.....	26
Abbildung 4.3: Metadaten für Verzeichnisse.....	26
Abbildung 4.4: Dateinamen und Metadateien.....	27
Abbildung 4.5: Architektur	27
Abbildung 4.6: UML Klassendiagramm des CIS.....	30
Abbildung 4.7: Architektur mit Content Information Browser	31
Abbildung 4.8: Graphische Darstellung der Schulungsmaterialablage	32
Abbildung 4.9: Demonstration des Content Information Browsers	32
Abbildung 4.10: UML Klassendiagramm des CIB	33
Abbildung 4.11: Überblick über die Schichten	34
Abbildung 4.12: Zusammenspiel der Schichten.....	35
Abbildung 5.1: Überblick über die eingesetzten Systeme und Protokolle	36
Abbildung 5.2: Verwaltung der Materialien.....	38
Abbildung 5.3: Dozentenumgebung.....	39
Abbildung 5.4: Präsentationsleiste	39
Abbildung 5.5: Kontextmenü der Präsentationsleiste.....	40
Abbildung 5.6: Präsentationsleiste im Betrieb.....	41
Abbildung 5.7: Indexgenerierung während der Veranstaltung	41
Abbildung 6.1: Pakete und Klassen des CIS	42
Abbildung 6.2: Setzen der benötigten Referenzen in Visual Studio	51
Abbildung 6.3: UML Sequenzdiagramm, Zusammenspiel von CIB, CIS, WSH und benutzender Anwendung	55
Abbildung 6.4: Komponenten und Referenzen der Dozentenumgebung.....	57

1 EINFÜHRUNG

1.1 MOTIVATION

Die Aus- und Weiterbildung unterliegt, wie viele andere Bereiche unserer Gesellschaft, dem immer weiter zunehmenden Einfluss des Informationszeitalters und der damit verbundenen Informations- und Kommunikationstechnologie. Jedoch gestalten sich die Nutzung und der sinnvolle, unterstützende Einsatz dieser Technologien für die beteiligten Personen oft als zu kompliziert und wenig effizient. Ersichtlich wird dies oft in Veranstaltungen zur Aus- und Weiterbildung – im universitären Umfeld Vorlesungen und Übungen – in denen der Dozent bestenfalls ein handgeschriebenes Manuskript präsentiert. Oder die Materialien liegen zwar digital vor, zur Präsentation werden sie dann aber auf Folie ausgedruckt und mittels Overhead-Projektor dargestellt. Nicht selten sind die Materialien dabei – optisch ersichtlich – aus verschiedensten Quellen zusammengestellt, und in Verbindung mit dem dadurch häufig verbundenen Wechsel der Präsentationstechnologie erscheint die Veranstaltung für die Lernenden sehr inhomogen. Diese Situation zu verbessern ist die Aufgabe des internetbasierten Wissenstransfers.

Die Grundlage des Wissenstransfers bilden strukturierte und in kleine Teile, so genannte Einheiten, aufgeteilte Lehrinhalte, die durch Autoren erstellt, verwaltet und gepflegt, durch den Dozenten für eine Veranstaltung zusammengestellt und präsentiert, und schließlich durch den Lernenden konsumiert werden.

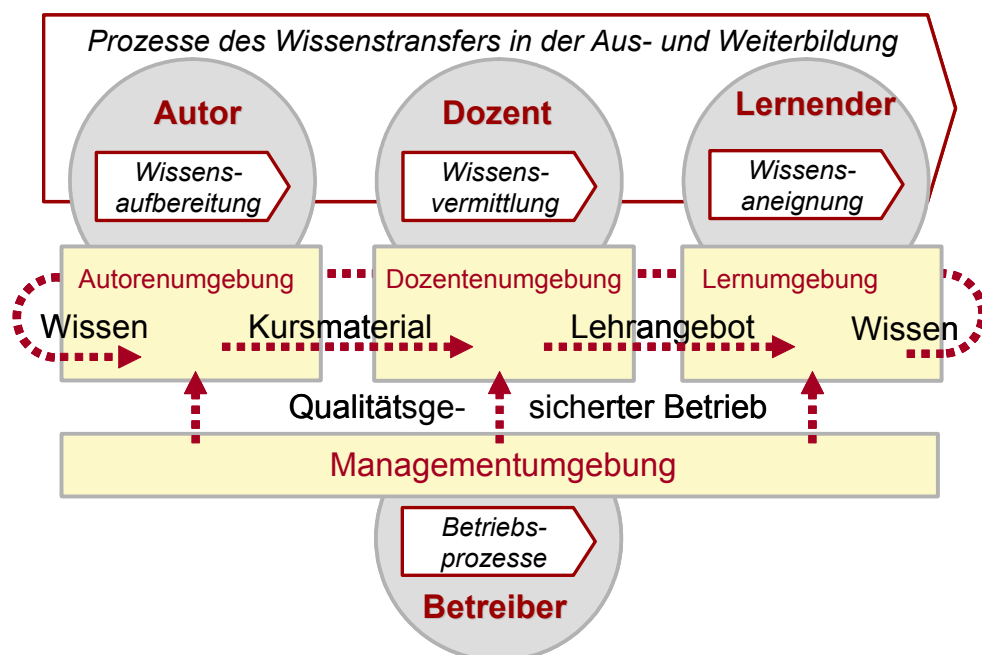


Abbildung 1.1: Rollen, Komponenten und Prozesse in der Ausbildung

Das Zusammenspiel der Rollen und der hiermit verbundene Austausch von Wissen werden als Dienstleistungen aufgefasst: Autoren sind Dienstleister gegenüber dem Dozenten, indem Lehrmaterialien zur Wissensvermittlung zur Verfügung gestellt werden. Der Dozent wiederum bietet seine Dienste in Form einer Lehrveranstaltung gegenüber den Lernenden an. Diese Dienste durch den Einsatz von Informations- und Kommuni-

kationstechnologie zu ermöglichen und deren zuverlässiges Funktionieren sicherzustellen ist Aufgabe des Betreibers. Abbildung 1.1 zeigt den Zusammenhang zwischen den beteiligten Rollen, Komponenten und Prozessen.

Dabei ist es wichtig, sich nicht nur auf die Prozesse des Lernenden zu konzentrieren, sondern auch die Werkzeuge und Vorgänge, die den Dozenten und seine Aufgabe betreffen, zu berücksichtigen. Denn für ein effizientes und erfolgreiches Lernen sind nicht nur die eigentlichen Lehrmaterialien bzw. deren Inhalt verantwortlich, sondern auch die Art und Weise, wie diese vermittelt werden. Eine gute Schulungsveranstaltung zeichnet sich zwar in erster Linie durch eine hohe Kompetenz des Dozenten und inhaltlich geeignete Materialien aus, aber auch durch den sinnvollen und zuverlässigen Einsatz von Informationstechnologie, um den Dozenten bei seiner Arbeit zu unterstützen und eine angenehme Schulungsatmosphäre zu gewährleisten. Die Basis der Wissensvermittlung – die Erstellung, Speicherung und Aufbereitung der Lehrmaterialien – findet im Autorenprozess statt. Auch in diesem Bereich kann eine gute Informationsinfrastruktur unterstützend wirken bzw. die Verwaltung und Wiederverwendung der Inhalte erleichtern.

Die Voraussetzungen für eine effiziente Aufbereitung von Lehrmaterialien, eine gelungene Vermittlung der Inhalte durch den Dozenten und ein erfolgreiches Aneignen durch den Lernenden sind eine geeignete Ablage der Lehrmaterialien und eine grundlegende Architektur zum Aufbau von Werkzeugen, die die Prozesse der Aus- und Weiterbildung unterstützen.

1.2 PROBLEMSTELLUNG

Ziel der Arbeit ist es, eine Architektur und Werkzeuge zu entwerfen und zu realisieren, welche den Anforderungen, die die Lehre und Ausbildung stellt, gerecht wird. Dies betrifft sowohl die Ablage, Speicherung und Verwaltung der Lehrmaterialien als auch die eigentliche Vermittlung des Stoffes im Hörsaal. Hierzu gehört das vorherige Sichten, Zusammenstellen und Auswählen der für eine bestimmte Veranstaltung benötigten Schulungsmaterialien. Die Architektur bzw. deren Realisierung soll einen möglichst einfachen und einheitlichen Zugang zur Schulungsmaterialablage ermöglichen. Dabei soll nicht nur der Nutzer, sondern auch der Entwickler unterstützender Werkzeuge durch einheitliche Schnittstellen zu dieser Ablage gefördert werden.

Grundgedanke ist, auf Basis einer Mehrschichten-Architektur sämtliche Operationen auf Datenebene – wie zum Beispiel das Anlegen einer neuen Datei – zu verdecken und eine einheitliche Datenzugriffsschicht zu realisieren, auf deren Basis dann weitere Werkzeuge aufsetzen können. Weiterhin soll eine einheitliche Sicht und graphische Repräsentation der Datenablage entwickelt werden, die unabhängig von der tatsächlich eingesetzten Speichertechnologie eine übersichtliche Darstellung der vorhandenen Lehrmaterialien bietet.

Der Entwurf und die Modellierung des Werkzeugs werden dabei in der *Unified Modelling Language* **UML** erfolgen. Das objektorientierte Konzept soll modularen Aufbau, Trennung von Darstellung, Anwendungslogik, internen Datenstrukturen und realem Dateisystem sowie die einfache und problemlose Wiederverwendbarkeit der einzelnen Programmkomponenten in weiteren Werkzeugen gewährleisten.

Schließlich soll auf Basis dieser Architektur und ihrer Komponenten ein Werkzeug entstehen, das den Dozenten sowohl bei der Vorbereitung als auch bei der Durchführung einer Lehrveranstaltung in geeigneter Weise unterstützt. Diese graphische Dozentenumgebung hat zum Ziel, die zur Verwaltung der strukturierten Schulungsmaterialablage

nötigen Funktionen zu integrieren und Mechanismen bereitzustellen, mit deren Hilfe ein reibungsloser und homogener Ablauf von elektronischen Präsentationen in Schulungsveranstaltungen gewährleistet werden kann. Dabei sollen vorhandene und bewährte Standardtechnologien nicht etwa ersetzt, sondern vielmehr deren Nutzung und Einsatz optimiert bzw. ergänzt werden.

1.3 ERGEBNISSE

Auf Basis der oben genannten Aufgabenstellung und Grundideen wurde im Rahmen dieser Arbeit eine Mehrschichten-Architektur entworfen und realisiert, die Speicherung, Verwaltung und Präsentation von Lehrmaterialien im universitären Umfeld erleichtert, Entwicklern einheitliche und leicht anzusprechende Schnittstellen bietet und die spätere Wiederverwendung der einzelnen Komponenten in weiteren Werkzeugen ermöglicht.

Die **strukturierte Schulungsmaterialablage** realisiert die Speicherung der digitalen Lehrmittel und bietet die Möglichkeit, einzelne Kurse in Kapitel und Unterkapitel zu strukturieren und mit Reihenfolgenvorgaben zu versehen. Die eigentlichen Lehrmaterialien in Form von Dateien lassen sich nach Wichtigkeit priorisieren und mit Attributen versehen, die festlegen, in welchem Kontext die jeweilige Datei erscheinen soll bzw. welchem Verwendungszweck sie letztlich dient. Darüber hinaus lassen sich die einzelnen Materialien noch mit Zusatzinformationen, den so genannten Meta-Daten, belegen. Diese Daten können zum Beispiel Informationen über den Autor, den Schwierigkeitsgrad oder technische Details wie Formatangaben, enthalten.

Der Zugang zu dieser Ablage wird realisiert vom so genannten **Content Information Server**. Er bildet die Datenablage in ein handhabbares Zwischenformat ab, das in den verschiedenen Umgebungen, die die Ablage nutzen, verwendet wird. Er bietet Entwicklern überdies eine einheitliche Schnittstelle zu dieser Ablage, um die dortigen Strukturen wie Dateiattribute oder Klassifizierung zu ändern. Dies geschieht für die benutzenden Werkzeuge vollkommen transparent. Das heißt, der Programmierer muss sich nicht mehr um die tatsächlich vorhandenen Datenstrukturen kümmern und man erreicht eine gewisse Flexibilität bzw. Unabhängigkeit gegenüber Änderungen an der Schulungsmaterialablage.

Will ein Anwender – wie zum Beispiel ein Autor oder Dozent – einen Überblick über die in der Datenablage vorhandenen Lehrmaterialien erhalten, erweist sich der direkte Zugang zu dieser Ablage als unübersichtlich und zu technisch. Die Schulungsmaterialablage muss graphisch aufbereitet, übersichtlich und dem jeweiligen Einsatzzweck angepasst, dargestellt werden. Diese Funktionalität leistet der im Rahmen dieser Arbeit ebenfalls realisierte **Content Information Browser**. Dieses Softwaremodul benutzt die vom Content Information Server angebotene Informationsstruktur und stellt mit deren Hilfe die Datenablage übersichtlich in Baumstruktur dar. Der Browser lässt sich über eine umfangreiche Schnittstelle vielfältig an die Gegebenheiten anpassen, die graphische Darstellung ist ebenfalls konfigurierbar. Das Modul stellt kein eigenständiges Programm dar, sondern lässt sich ohne großen Aufwand in vorhandene Werkzeuge integrieren.

Schließlich wurde mit der **graphischen Dozentenumgebung** ein Werkzeug entwickelt, das zweierlei leistet: Die Verwaltung der Schulungsmaterialablage und die Präsentation der Materialien. Damit erhält der Dozent einen graphischen Überblick über den Materialbestand sowie die Möglichkeit, einzelne Materialien in andere Formate umzuwandeln, Dateiattribute zu ändern und die Dateien aus einer integrierten Umgebung heraus während einer Veranstaltung zu präsentieren, ohne sich um technische Details – wie benötigte Werkzeuge zur Darstellung der Materialien – kümmern zu müssen. Die Dozenten-

umgebung benutzt dabei die von Content Information Server und Browser angebotene Funktionalität.

1.4 ÜBERBLICK

Nach einem kurzen Überblick über die Basisprinzipien von wiederverwendbarer Software, Schichtenarchitekturen und modernen Präsentationstechnologien werden exemplarisch zwei typische e-Learning Umgebungen vorgestellt. Kapitel 3 zeigt die Anforderungen und Aufgabenbereiche auf, die sich bei der internetbasierten Wissensvermittlung ergeben, analysiert diese und spezifiziert die Architektur und die Software, die diese Anforderungen erfüllen soll.

Kapitel 4 erläutert detailliert die realisierte Architektur, das ihr zugrunde liegende Schichtenmodell, die zentralen Aufgabenbereiche der einzelnen Schichten, ihre jeweilige Funktionalität und das Zusammenspiel der Komponenten untereinander. Anschließend illustriert Abschnitt 5 die Nutzung der graphischen Dozentenumgebung sowie deren Einsatz im universitären Lehrbetrieb.

Detaillierte Informationen für Entwickler, die genaue Schnittstellenbeschreibung der einzelnen Komponenten sowie Erläuterungen zu deren Einsatz bzw. Verwendung und Integration vermittelt der daran anschließende Abschnitt „Implementierung“. Hier werden auch die interne Arbeitsweise der einzelnen Komponenten und die verschiedenen Datenstrukturen erläutert. Abschließend folgt noch ein kurzer Ausblick über die Weiterentwicklung und das Ausbaupotenzial der Module. Im Anhang sind im Laufe der Arbeit entstandene Folien beigefügt.

2 STAND DER TECHNIK

2.1 ARCHITEKTUREN

2.1.1 KOMPONENTENBASIERTE SOFTWAREENTWICKLUNG

Typisches Merkmal früherer Software-Projekte war der sehr geringe Wiederverwendungsgrad von bereits vorhandener Software. Die vielfach praktizierte Trennung von Algorithmen und Daten führte häufig dazu, dass man vorhandenen Code nicht einfach in neue Anwendungen übernehmen konnte und oft „das Rad neu erfinden“ musste. Vorangetrieben und unterstützt von der zunehmenden Objektorientierung stieg allmählich der Wiederverwendungsgrad von Software, indem man versuchte, einzelne Bausteine mit einer in sich abgeschlossenen Funktionalität und fest definierten Eigenschaften zu realisieren. Der Vorteil solcher Objekte besteht in der inneren Einheit von Daten und Funktionen: Entwickler, die die Funktionalität nutzen möchten, müssen lediglich wissen, welche Funktionalität ein Objekt erbringt und wie man diese anspricht. Änderungen an der inneren Struktur brauchen sie nicht zu interessieren und haben keinerlei Auswirkung, solange sich die Schnittstelle nicht ändert.

Softwarekomponenten sind also speziell für die Wiederverwendung in anderen Anwendungen vorgesehene Programmteile, deren Einbindung in vorhandene Anwendungen so einfach wie möglich erfolgen können sollte. Verschiedene Komponenten können dabei miteinander kombiniert, angepasst oder verändert werden, der Anwender muss nur die Funktionalität und die Schnittstelle der Softwarekomponente kennen.

Vorteile von komponentenbasierter Softwareentwicklung:

- Durch den verringerten Programmieraufwand lassen sich Anwendungen schneller realisieren.
- Eine Komponente, die oft verwendet wird, ist im Allgemeinen gut getestet. Dies kommt der Qualität zugute.
- Kleine Komponenten mit genau spezifiziertem Verhalten dienen der Übersichtlichkeit und verringern oftmals die Komplexität von Programmcode.
- Durch die Möglichkeit, eine Komponente intern zu ändern und an sich ändernde äußere Gegebenheiten anzupassen, ohne die Schnittstelle zur benutzenden Anwendung zu ändern, sinkt der Wartungsaufwand.

Bei der Entwicklung von Softwarekomponenten sollte darauf geachtet werden, dass der Wiederverwendungsgrad möglichst hoch ausfällt, indem die Funktionalität so allgemein oder umfangreich wie möglich gehalten wird. Des Weiteren wünschenswert sind einfache Erweiterbarkeit, gute Anpassbarkeit auf sich ändernde Gegebenheiten, die Möglichkeit die Funktionalität zu testen und letztlich das Sicherstellen von Korrektheit und Fehlerfreiheit.

Es existieren verschiedene standardisierte Architekturen zur Realisierung der Kommunikation einzelner Softwarekomponenten untereinander:

- Die von der *Object Management Group* (OMG) standardisierte *Common Object Request Broker Architecture* (CORBA) [1]. Damit sind plattformunabhängige Verwaltung und Einsatz von Objekten möglich.

- Microsoft's Standard *Distributed Common Object Model* (COM/DCOM). Auch diese Architektur stellt einen Standard zur Verfügung, durch den Softwarekomponenten miteinander kommunizieren können [2].

2.1.2 SCHICHTENARCHITEKTUR UND VERTEILTE SYSTEME

Typischerweise lassen sich die einzelnen Komponenten eines Softwaresystems in drei Gruppen einteilen. Man spricht in diesem Falle von einer so genannten 3-Schichten-Architektur [3]:

1. **Präsentationsschicht:** Diese Komponenten realisieren die Schnittstelle zum Benutzer und sind zuständig für die meist graphische Aufbereitung der Daten für den Nutzer und das Entgegennehmen von Benutzereingaben.
2. **Anwendungsschicht:** Sie repräsentiert typische anwendungsspezifische Funktionen wie Algorithmen, interne Datenstrukturen oder Kommunikation mit anderen Anwendungskomponenten.
3. **Datenhaltungsschicht:** Die zu dieser Schicht gehörenden Komponenten bieten Zugriff auf die eigentlichen Daten und leisten deren Verwaltung in Form von Dateisystemen oder Datenbanken auf Basis der zugrunde liegenden Speichertechnologie.

Der Vorteil einer solchen Schichtenarchitektur besteht in der klaren Trennung von Nutzerschnittstelle, Anwendungslogik und Datenrepräsentation und der damit verbundenen Möglichkeit, eine Schicht zu ändern, ohne dass sich das an anderer Stelle auswirkt. Es müssen lediglich die Schnittstellen der einzelnen Schichten zueinander stabil bleiben. So könnte die Datenhaltung geändert werden, die Schnittstelle für die Anwendungsschicht jedoch unverändert bleiben.

Ein Beispiel einer solchen Schichtenarchitektur ist das Kommunikationsmodell des Internet: Das Transportprotokoll TCP/IP bietet eine einheitliche Schnittstelle für Anwendungen, die darunter liegenden Systemtreiber bilden die Schnittstelle zur eingesetzten physikalischen Übertragungstechnik. Somit können beliebige Anwendungen unabhängig von der Kommunikationstechnologie miteinander kommunizieren.

Bei der Art und Weise, wie die drei oben genannten Schichten *Präsentation*, *Anwendung* und *Datenhaltung* auf verschiedene, räumlich getrennte Recheneinheiten verteilt werden können, unterscheidet man zwischen drei Architekturmodellen:

1. **Single-Tier-Architektur:** Sämtliche Schichten sind in einem einzigen System konzentriert bzw. in einer einzigen Anwendung realisiert. Dies erleichtert die Wartung und Überwachung des Systems, jedoch entstehen Probleme, wenn dessen Leistung nicht mehr ausreicht und Überlastung eintritt: Durch die Konzentration sämtlicher Komponenten in einem einzigen Rechner muss dieser zwangsläufig aufgerüstet werden, was nicht immer ohne weiteres möglich ist. Vertreter der Single-Tier-Architektur sind die großen Mainframe-Systeme, auf die mittels kleiner und einfacher Terminals zugegriffen wird.
2. **2-Tier-Architektur:** Diese Architektur trennt die Datenhaltung von den Anwendungskomponenten und der Nutzerschnittstelle. Die Daten und ihre Verwaltung befinden sich auf einem Server, auf Client-Workstations sind die Anwendungsprogramme und die Präsentation realisiert. Dadurch können die Server einfacher und kostengünstiger realisiert werden. Probleme ergeben sich allerdings, wenn die Anwendungskomponente geändert werden muss: An jedem Client sind, auch individuelle, Anpassungen notwendig, was sich je nach Benut-

zerzahl als schwierig und umständlich erweisen kann. Ein weiterer Nachteil besteht bei sicherheitskritischen Anwendungen. Je weniger Verantwortlichkeit bei den Clients liegt, desto weniger potentielle Sicherheitsprobleme können entstehen.

- 3. 3-/Multi-Tier-Architektur:** Dieser Ansatz verteilt die drei Komponenten entsprechend ihrer Aufgabe auf drei Systemkomponenten: Daten(-bank)server, Anwendungsserver und die Client-Rechner, deren Funktionalität auf das bloße Darstellen der Daten reduziert ist. Dabei können die Datenkomponenten und die Anwendungsserver jeweils wiederum auf mehrere Rechner verteilt sein (Multi-Tier). Die Anwendungslogik ist in den so genannten Middleware Servern realisiert, sie leiten die Anfragen der Clients an die Datenhaltungskomponenten weiter bzw. verteilen die Anfragen auf diese. Diese logische Trennung bietet den Vorteil der leichten Erweiterbarkeit und Weiterentwicklung der einzelnen Komponenten unabhängig voneinander sowie die Anpassung und Wiederverwendung der einzelnen Komponenten. Allerdings werden weitere Maßnahmen zur Lastverteilung und im Bereich des Managements notwendig, zudem ist der Entwicklungs- und Konzeptionsaufwand einer solchen Architektur im Vergleich zu weniger stark verteilten Architekturen recht hoch.

2.2 ELEKTRONISCHE TAFEL

Elektronische Tafeln dienen dazu, zwei grundverschiedene Ansätze der Lehrmitteldarstellung in einer Schulungsveranstaltung zu vereinen. Da wäre einerseits die „klassische“ Präsentation mittels Tafel und Kreide und andererseits die elektronische Präsentation mittels Notebook und LCD-Projektor. Beide haben ihre typischen Vor- und Nachteile:

Tafel und Kreide:

- + spontanes Entwickeln von Tafelbildern während der Präsentation
- + gute Lesbarkeit auch bei hellem Tageslicht
- + Unterrichtstempo wird nicht zu schnell, da auf Schreibgeschwindigkeit des Vortragenden beschränkt
- schlechte Wiederverwendbarkeit der Inhalte
- Lernende müssen mitschreiben

Notebook und Projektor:

- + Aufwerten der Präsentation mit beliebigen auf dem Notebook darstellbaren Inhalten möglich
- + sehr gute Wiederverwendbarkeit der Materialien
- + gute Verteilmöglichkeit der Materialien erspart den Lernenden das Mitschreiben
- Erweiterung der Inhalte während der Präsentation praktisch nicht möglich (nur durch „Ausweichen“ auf die Tafel)

Ziel der so genannten „elektronischen Tafeln“ ist es nun, diese beiden Techniken zu vereinen, bzw. die Vorteile der klassischen Methode der elektronischen Präsentation zur Verfügung zu stellen, ohne dabei einen Medienwechsel vornehmen zu müssen. Hier sollen zwei Verfahren kurz vorgestellt werden: Eingabesensitive Plasmabildschirme und Graphiktablets.

2.2.1 PLASMABILDSCHIRM

Plasmabildschirme sind Flachbildschirme, die bis zu einer Größe von 50 Zoll sichtbarer Bilddiagonale im Handel angeboten werden. Das Bild wird dabei durch leuchtendes Gas erzeugt. In Verbindung mit spezieller Software und einer auf das Display montierten druckempfindlichen Oberfläche wird eine elektronische Tafel realisiert. Der Dozent kann so während der Präsentation seine Folien handschriftlich annotieren. Die mitgelieferte Software erlaubt es dabei, die Annotationen zusammen mit den vorbereiteten Folien zu sichern, so dass diese nicht verloren gehen und den Lernenden auch nach der Veranstaltung zu Verfügung stehen. Betrachtet wurde das von der Firma Smart-Technologies entwickelte und in Abbildung 2.1 zu sehende System „*SMART Board for Plasma Displays*“ [4].



Abbildung 2.1: Beispiel eines Plasmabildschirms

Vorteil dieser Systeme ist die Größe und Leuchtkraft der Plasmadisplays. In kleinen Schulungsräumen kann somit auf einen Projektor verzichtet werden. Nachteilig wirken sich die durch das hohe Gewicht eingeschränkte Mobilität und der noch recht hohe Preis eines solchen Displays aus.

2.2.2 GRAPHIKTABLETT

Graphiktablets liegt das gleiche Prinzip zugrunde, das auch bei der oben beschriebenen Technik verwendet wird: Über einem selbstleuchtenden Display wird eine eingabeempfindliche – im Falle des betrachteten WACOM-Modells durch ein elektromagnetisches Prinzip sogar berührungsfreie – Oberfläche angebracht, über die der Dozent direkt auf der Präsentation schreiben kann, als würde diese in Papierform vorliegen (Abbildung 2.2).



Abbildung 2.2: Beispiel eines Graphiktablets

Im Vergleich zum oben beschriebenen Plasmadisplay-System erweist sich ein Graphiktablett als wesentlich flexibler, da durch die verwendete LCD-Leuchttechnik viel Gewicht gespart wird. Durch die kleine Größe des Displays (entsprechend gängigen TFT-Flachbildschirmen meist 15 Zoll) ist aber der Einsatz eines Projektors unumgänglich. Ein Beispiel eines solchen Tablett findet sich unter [5].

2.3 E-LEARNING-UMGEBUNGEN

2.3.1 E-KREIDE

E-Kreide ist ein sehr interessantes System. Es versucht die Vorteile der klassischen Tafel mit denen einer digitalen Präsentation und eines Teleteaching-Systems zu vereinen. Der Dozent kann zusätzlich zur Funktion, ein elektronisches Tafelbild zu erzeugen, interaktive Java-Programme und Bilder direkt aus dem WWW in die Präsentation integrieren.

Der Lernende kann das System von jedem beliebigen Browser mit Java-Fähigkeit aus benutzen. Drei Datenströme übertragen live während der Veranstaltung das eigentliche elektronische Tafelbild, das auch die Anwesenden im Hörsaal sehen, das Audiosignal aus dem Hörsaal sowie ein Video des Dozenten (Abbildung 2.3).

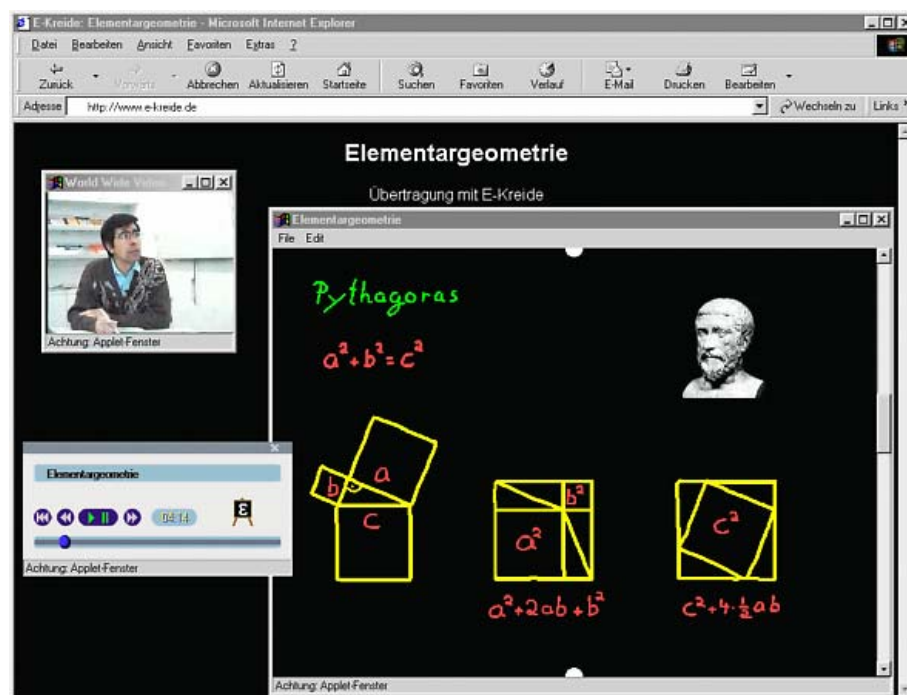


Abbildung 2.3: E-Kreide

Während einer Schulungsveranstaltung laufen auf dem Vortragsrechner parallel drei Server, die jeweils Tafelbild, Audio und Videostrom im Internet zur Verfügung stellen, welche beim Client über je ein separates Applet empfangen und dargestellt werden können. Je nach verfügbarer Bandbreite können die Datenströme einzeln deaktiviert werden. Gleichzeitig werden die Vorlesungen archiviert, so dass sie auch zu einem späteren Zeitpunkt abrufbar sind.

Als Eingabemedium im Vortragsraum kommt neben einem Notebook die oben (Abschnitt 2.2) beschriebene Technologie eines Plasmaschirms oder Graphiktablets zum Einsatz.

Zusätzlich zur Nutzung als elektronische Tafel erlaubt E-Kreide noch die Einbindung von weiteren Elementen:

- Direkte Anzeige von Bildern aus dem Internet,
- Ausführen von Java-Applets direkt „in der Tafel“,
- Auswertung mathematischer Ausdrücke, und
- Graphische Darstellung von Funktionen.

E-Kreide erweist sich insgesamt als ein sehr gutes System, falls der Dozent gewohnt ist, seine Schulungen handschriftlich zu gestalten. Ebenso vorteilhaft ist die enge Integration einer Übertragung zur entfernten Teilnahme an der Veranstaltung.

Informationen zu E-Kreide finden sich unter [6].

2.3.2 CISCO E-LEARNING ARCHITEKTUR

Cisco Systems hat ein sehr umfangreiches System zur unternehmensweiten Aus- und Fortbildung entwickelt, das auf große, weltweit agierende Unternehmen ausgelegt ist. Basierend auf einer 3-Tier-Architektur beinhaltet das Cisco-Modell drei Schichten: Zugriff-, Anwendungs- und die darunter liegende Netzinfrastruktur-Schicht (Abbildung 2.4).

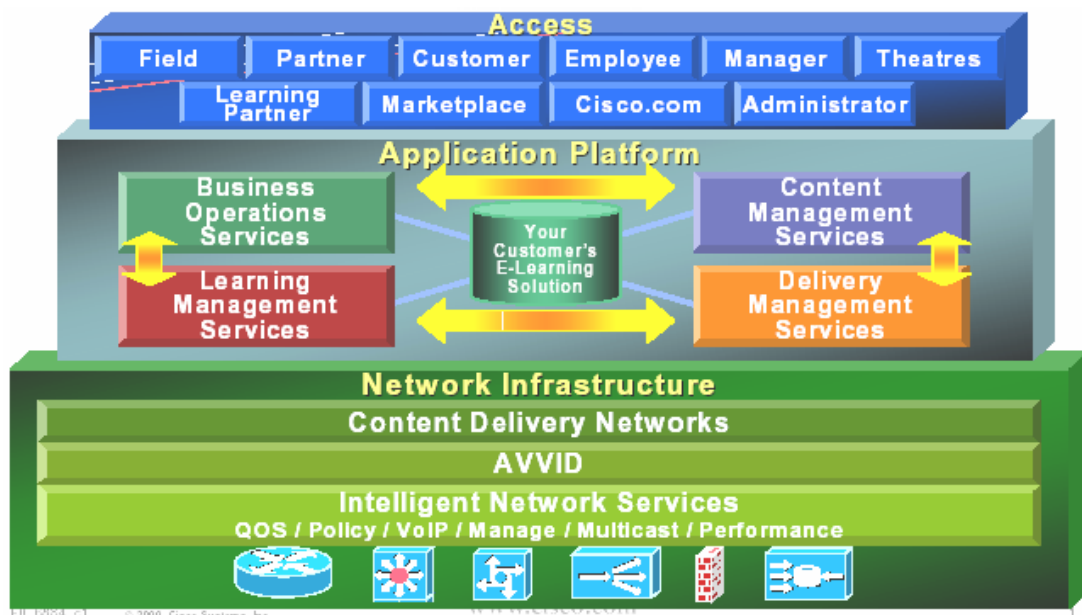


Abbildung 2.4: E-Learning Architektur von Cisco

Der Zugang zur e-Learning-Lösung erfolgt dabei für alle Beteiligten grundsätzlich von der Zugriffsschicht aus, welche sich flexibel an die jeweiligen Bedürfnisse anpassen lässt. Die Anwendungsplattform ist in vier Bereiche unterteilt, die jeweils verschiedenste Dienste zur Verwaltung, Entwicklung, Konzeption und Auslieferung der Lerninhalte

anbieten. Nicht näher betrachtet werden soll die Netzinfrastrukturschicht. Sie ist verantwortlich für Sicherheitsaspekte, Verteilung der Inhalte, Verfügbarkeit und Leistung.

Die *Business Operation Services* vereinigen und beinhalten Dienste, Anwendungen und Werkzeuge, die von allen Teilorganisationen und -bereichen benötigt werden. Dazu gehören Support, Online-Hilfe, Benutzerverwaltung, Benutzerfeedback, Protokollierung und eine zentrale Verwaltung der verschiedenen Anforderungen der einzelnen Ausbildungsbereiche.

Content Management Services ermöglichen die Einstellung, Kategorisierung, Zusammenstellung, Verwaltung und Veröffentlichung der eigentlichen Lehrinhalte. Neben bekannten Autorenwerkzeugen wie Dreamweaver und Powerpoint stehen auch eine Reihe von internen Werkzeugen zur Verfügung: Die Registrierungs-Dienste verwalten zu jedem Inhaltsobjekt den zugehörigen Speicherort sowie beschreibende und strukturierende Metadaten. Weiterhin existieren Suchdienste, Versionskontrollfunktionen und Vorlagen als Basis für den so genannten „Assembler“, mit dessen Hilfe einzelne Lernobjekte zusammengestellt werden können.

Soll nach dem Autoren-Prozess der Inhalt dann veröffentlicht werden, kommen die *Delivery Management Services* zum Zuge. Sie können den Inhalt auf das jeweilige Nutzerprofil und die Anforderungen des Lernenden dynamisch anpassen. Dies betrifft nicht nur inhaltliche Aspekte, sondern auch technische wie z.B. die Art der Auslieferung oder das Format der Daten. Natürlich können auch vorgefertigte, fest vorgegebene Inhaltspakete nutzerunabhängig angeboten und verteilt werden. Weiterhin werden die Aktivitäten eines Lernenden aufgezeichnet und den *Learning Management Services* übermittelt.

Diese *Learning Management Services* bieten neben Registrierung und Zugang zur Lernumgebung personalisierte Führungen, die auf den jeweiligen Nutzer zugeschnitten sind. Dazu gehören auch die Verwaltung der Interaktion mit dem Nutzer, Navigation, Auswahl der Lehrangebote und die Verbindung zum *Delivery Management Service*, um die Materialien letztlich auszuliefern. Die Dienste vereinigen viele Funktionen zur Personalisierung, der Erstellung von Nutzerprofilen, Nutzerregistrierung sowie Suchen und Stöbern in den Materialien. Weiterhin besteht die Möglichkeit, das Nutzerverhalten und dessen Lernfortschritt aufzuzeichnen, eigene Kurse und Lehrpläne zu erstellen oder ein e-Commerce System aufzusetzen.

Informationen über Cisco e-Learning finden sich unter [7].

2.3.3 WERTUNG

E-Kreide zeigt sich zu unflexibel im Zusammenspiel mit vorhandenen Lehrmaterialien, denn es ist nicht möglich, beliebige bereits erstellte digitale Materialien in die Präsentation einzubinden. Falls der Dozent andere Materialien als Java-Applets oder Bilder aus dem WWW in seine Darbietung aufnehmen möchte (hier seien als Beispiel die weit verbreiteten Powerpoint-Präsentationen oder auch Videos und Flash-Animationen genannt), ist ein Wechsel zu anderen Darstellungswerkzeugen notwendig, wodurch eine unerwünschte Unterbrechung der Veranstaltung stattfindet.

Die Cisco-Architektur ist aufgrund der Auslegung auf weltweit agierende Großunternehmen zu umfangreich und zu komplex für einen Einsatz im universitären Lehrbetrieb.

Deshalb wurde für diese Arbeit an anderes Konzept erstellt.

3 KONZEPTION

3.1 EINORDNUNG IN ED.TEC

Die Arbeitsgruppe **ed.tec** des Lehrstuhls „Cooperation & Management“ an der Fakultät für Informatik hat es sich zur Aufgabe gemacht, eine Architektur und Werkzeuge zu entwerfen bzw. zu realisieren, die den Anforderungen des internetbasierten Wissenstransfers in der Aus- und Weiterbildung gerecht werden. Ziel ist die einfache und effektive Unterstützung der drei Teilbereiche Wissensaufbereitung, Wissensvermittlung und Wissensaneignung für die in diesen Teilbereichen involvierten Personen, die Autoren, Dozenten und Lernenden [9, 20].

Einen Überblick über die einzelnen Aufgabenbereiche, die beteiligten Rollen und Vorgänge zeigt die Abbildung 3.1

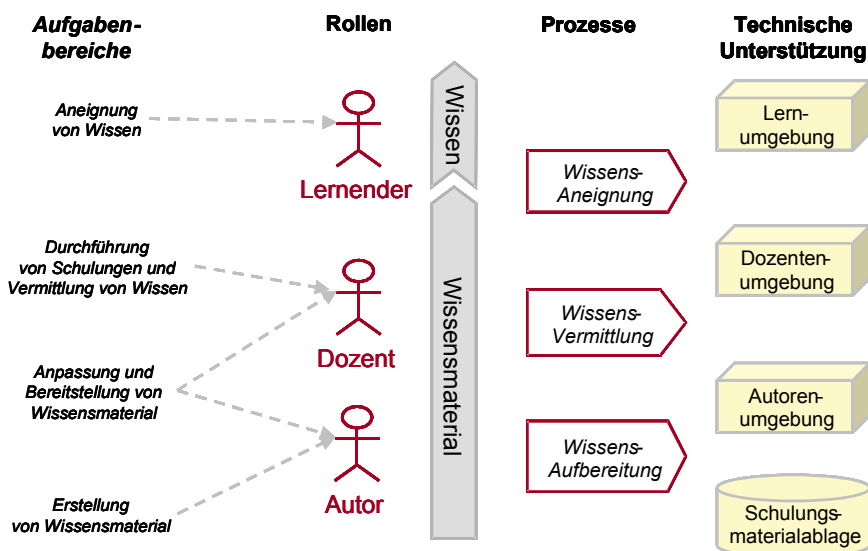


Abbildung 3.1: Aufgabenbereiche der internetbasierten Wissensvermittlung

Dabei wird zwischen Autor und Dozent unterschieden, weil es nicht immer gegeben ist, dass der Verfasser von Material dieses auch in Veranstaltungen vermittelt, sondern andere die Inhalte wieder verwenden. Die Materialien stehen modularisiert – also in kleine Einheiten unterteilt – über die strukturierte Schulungsmaterialablage (siehe auch Abschnitt 4.1) zur Verfügung. Diese Einheiten werden bei der Wissensaufbereitung kategorisiert und mit Zusatzinformationen versehen, was dem Dozenten das Zusammenstellen der für einen Kurs benötigten Materialien erleichtert. Unterstützt werden diese Prozesse der Aufbereitung und Vermittlung durch die Autoren- und Dozentenumgebung (vgl. Kapitel 5). Dies alles dient dazu, die Wissensaneignung für den Lernenden effektiver zu gestalten, was wiederum technisch durch die so genannte Lernumgebung unterstützt wird. Auf die verschiedenen Aufgabenbereiche Erstellung, Anpassung, Vermittlung und Aneignung soll im Folgenden detaillierter eingegangen werden.

3.1.1 ERSTELLUNG UND AUFBEREITUNG

Die Schulungsmaterialien sind das zentrale Element in der Aus- und Weiterbildung. Sie bilden das Grundgerüst einer Schulungsveranstaltung, enthalten das zu vermittelnde Wissen und helfen dem Lernenden, sich das neue Wissen anzueignen.

Die zu einem bestimmten Themengebiet vorhandenen Materialien werden im Regelfall für unterschiedlichste Anforderungen erstellt, genügen verschiedensten Ansprüchen und müssen deshalb üblicherweise neu strukturiert, aufbereitet oder vielleicht sogar komplett neu erstellt werden. Gründe dafür können zum Beispiel ein für die Zielgruppe unangemessener Schwierigkeitsgrad, eine veränderte Betrachtungsweise des Lehrthemas oder auch technische Gesichtspunkte wie ein ungeeignetes Präsentationsformat oder nicht zufriedenstellende Darstellungsqualität der Materialien sein.

Verfolgte Ziele der Aufbereitung und Erstellung sind:

- Zielgruppengerechte Darstellung des Wissens,
- Darstellung in einem dem Umfang und der Dauer eines Kurses angepassten Detaillierungsgrad, und
- Verwendung eines für den Dozenten adäquaten, didaktischen Konzepts.

Technische Unterstützung dieser Prozesse findet sich in Form der strukturierten Schulungsmaterialablage sowie der Autorenumgebung. Das Schulungsmaterial wird ggf. zunächst inhaltlich angepasst und danach, in einer festgelegten Art und Weise strukturiert, über die Autorenumgebung in der Schulungsmaterialablage abgelegt.

3.1.2 ANPASSUNG UND BEREITSTELLUNG

Das vorhandene Schulungsmaterial – sei es nun selbst erstellt oder wieder verwendet – ist zunächst strukturiert und modularisiert (damit ist die Aufteilung in einzelne kleinere Teile gemeint) in der Schulungsmaterialablage abzulegen. Die vorgegebene Struktur erleichtert anderen Autoren das Durchsuchen nach bestimmten Materialien zur Wiederverwendung. Dozenten können aus der Struktur heraus einzelne Module präsentieren und Lernende werden sich Wissen anhand der vorgegebenen Struktur und Module aneignen können. Folgende Mechanismen stehen den Dozenten und Autoren bei der Strukturierung und Anpassung der Materialien zur Verfügung:

- Definition eines Kurses durch den Aufbau einer hierarchischen Ordnung beliebiger Tiefe und Zuordnung der modularisierten Schulungsmaterialien durch Gliederung und Nummerierung,
- Klassifizierung der Schulungsmaterialien durch Unterscheidung von Haupt- bzw. Zusatzmaterial,
- Bestimmung der Zielgruppe (Autor, Dozent und Lernender) bzgl. der Verwendung der Schulungsmaterialien durch Attributvergabe (Beispiel: Der Autor verfasst die Materialien typischerweise in Powerpoint, der Dozent verwendet zur Präsentation aber ein spezielles Whiteboard-Format, während der Lernende eher eine papiersparende 4-auf-1-Darstellung der Folien als PDF-Datei zur Druckausgabe bevorzugen wird),
- Versehen der einzelnen Materialien mit Zusatzinformationen mittels Metadaten-Dateien, um die Recherche zu unterstützen.

Die Mechanismen unterliegen einer vorgegebenen Syntax, die sowohl für die Nutzer verständlich erscheint als auch vom Rechner automatisch verarbeitet werden kann. Die Anpassung und Bereitstellung der Schulungsmaterialien verfolgen dabei nachstehende Ziele:

- Strukturiertes Ablegen von Schulungsmaterialien für den einheitlichen Zugriff durch Mensch und Maschine (siehe auch Abschnitt 4.2) auf die Schulungsmaterialablage,
- Anpassen der Schulungsmaterialien auf Formate, die von Autoren, Dozenten und Lernenden verwendet werden können und deren unterschiedliche Anforderungen abdecken,
- Aufteilen der Schulungsmaterialien in unterschiedliche Ausprägungen wie Präsentationsfolien, Skriptum, Übungen, Animationen, Webadressen usw.,
- Zusätzliche Beschreibung der Materialdateien und deren Inhalte in Bezug auf pädagogische und technische Aspekte durch Vergabe bestimmter Attribute, wie beispielsweise Zielgruppe, Schwierigkeitsgrad, Dauer, Format und benötigtes Präsentationswerkzeug, in den Metadaten.

3.1.3 VERMITTLUNG

Der Dozent hat die Aufgabe mit Hilfe der vorbereiteten Materialien das Wissen zu vermitteln. Dabei sollten ihn sämtliche eingesetzten Kommunikations-, Informations-, und Präsentationstechnologien unterstützen, ohne jedoch hinderlich zu sein oder den Dozenten durch ihre technischen Eigenheiten zu überfordern. Die Schulungsveranstaltung kann dabei auf verschiedene Arten abgehalten werden [11].

Die **Präsenzveranstaltung** bezeichnet die Form, die jeder von der Schule oder von Vorlesungen her kennt. Die Teilnehmer sind bei der Veranstaltung anwesend und nehmen direkt daran teil. Die Präsentation der Inhalte kann dabei auf unterschiedlichste Art und Weise erfolgen: Klassisch mit Tafel und Kreide, mit ausgedruckten Folien und Overhead-Projektor oder per Notebook, Leinwand und einem so genannten „Beamer“. Durch die Rechnerunterstützung müssen die digitalen Schulungsmaterialien im Vorfeld nur noch an das einzusetzende Präsentationswerkzeug angepasst bzw. in ein präsentationsgerechtes Format umgewandelt werden.

Der Dozent möchte die Materialien mit möglichst wenig Aufwand präsentieren: eine Unterstützung, die den unkomplizierten Aufruf der Schulungsmaterialien ermöglicht, wäre wünschenswert. Dies ist vor allem dann der Fall, wenn sich die Veranstaltung über sehr viele einzelne Einheiten erstreckt und für die Anzeige dieser Dateien womöglich noch unterschiedliche Präsentationswerkzeuge notwendig sind. Der Aufruf der einzelnen digitalen Schulungsmaterialien geschieht über die graphische Dozentenoberfläche. Sie zeichnet sich durch die Anordnung aller Schulungsmaterialien wie sie vom Autor in der Schulungsmaterialablage abgelegt wurden aus (siehe auch Kapitel 5). Hierbei wird die Priorität und der Verwendungszweck (hier nur Präsentation durch den Dozenten) der Schulungsmaterialien berücksichtigt. Somit kann der Dozent im Vorfeld die Schulung seinen Vorstellungen entsprechend vorbereiten und sich während der Veranstaltung über die graphische Dozentenoberfläche seinen „roten Faden“ anzeigen lassen.

Die **Live-Veranstaltung** zeichnet sich durch die entfernte Teilnahme der einzelnen Lernenden an der Veranstaltung aus. Für die Teilnehmer entfällt damit die Notwendigkeit, zum Veranstaltungszeitpunkt einen bestimmten Ort aufsuchen zu müssen, denn sie

können die Veranstaltung von jedem beliebigen Rechner mit Internetzugang aus verfolgen.

Diese Art der Veranstaltung setzt zwingend den Einsatz von Digitaltechnologie wie Notebook und Projektor voraus. Weiterhin benötigt werden ein Mikrofon zur Aufnahme der vom Dozenten gesprochenen Kommentare sowie ein Rechner, der die Präsentation aufzeichnet und – als Video kodiert – per Live-Stream im Internet zur Verfügung stellt. Bei dieser Art von Veranstaltung erweisen sich weitergehende Technologien zur digitalen Annotation der Folien während der Präsentation, wie zum Beispiel ein Graphiktablett (vgl. Abschnitt 2.2.2), als sehr nützlich.

Die letzte Ausbaustufe ist die **Remote-Veranstaltung**, mit Hilfe derer der Lernende nun zeit- und ortonabhängig die Schulungsveranstaltung über das Internet abrufen kann. Hierfür sind die technischen Mittel der Live-Veranstaltung Voraussetzung: Es wird ein Video der Präsentation aufgezeichnet. Zusätzlich wird noch ein Web-Server, der die aufgezeichneten Veranstaltungen zum Abruf bereithält, benötigt. Der Lernende kann nun jederzeit die Veranstaltung abrufen. Die graphische Benutzeroberfläche unterstützt den Dozenten auch hierbei, indem bei jedem Wechsel des Schulungsmaterials automatisch eine entsprechende Marke im aufgezeichneten Video gesetzt wird (siehe auch Abschnitt 5.4). Der Lernende kann dadurch nun wahlweise innerhalb des Videos auf verschiedene Vorlesungsabschnitte zugreifen.

Die technische Unterstützung des Dozenten wird durch die Dozentenumgebung transparent erbracht. Die Ziele der Durchführung von Schulungsveranstaltungen können wie folgt definiert werden:

- Transparente Unterstützung des Dozenten durch Informations- und Kommunikationstechnologien,
- Bereitstellung einer graphischen Dozentenoberfläche zum effizienten Aufrufen und Präsentieren von Schulungsmaterialien,
- Durchführen von Präsenz- und Onlineveranstaltungen für die synchrone und asynchrone Teilnahme des Lernenden.

3.1.4 ANEIGNUNG

Die vom Dozenten bzw. Autor vorgenommene Strukturierung der Schulungsmaterialien bildet die Grundlage für einen webbasierten Zugriff auf diese Materialien. Die Struktur der Materialien (ihre Gliederung, Klassifikation, Format) wird dabei automatisch in HTML-Seiten abgebildet und über einen Webserver zugänglich gemacht. Dabei kann im Vorfeld bereits festgelegt werden, welche Materialien über das System zu veröffentlichen sind, indem betreffende Materialien für die Zielgruppe des Lernenden spezifiziert (vgl. Abschnitt 4.1.2) werden. Der Lernende kann über die bereitgestellte Struktur beliebig navigieren und sich alle darin vorkommenden Inhalte direkt anzeigen lassen.

Die Ziele der Wissensaneignung im Rahmen der internetbasierten Wissensvermittlung sind:

- Automatische Bereitstellung vorhandener Schulungsmaterialien über das World Wide Web und
- Beliebige Navigation im Kurs und beliebiger Abruf der bereitgestellten Schulungsmaterialien über das Internet.

Das gesamte Szenario der internetbasierten Wissensvermittlung verdeutlicht nochmals Abbildung 3.2.

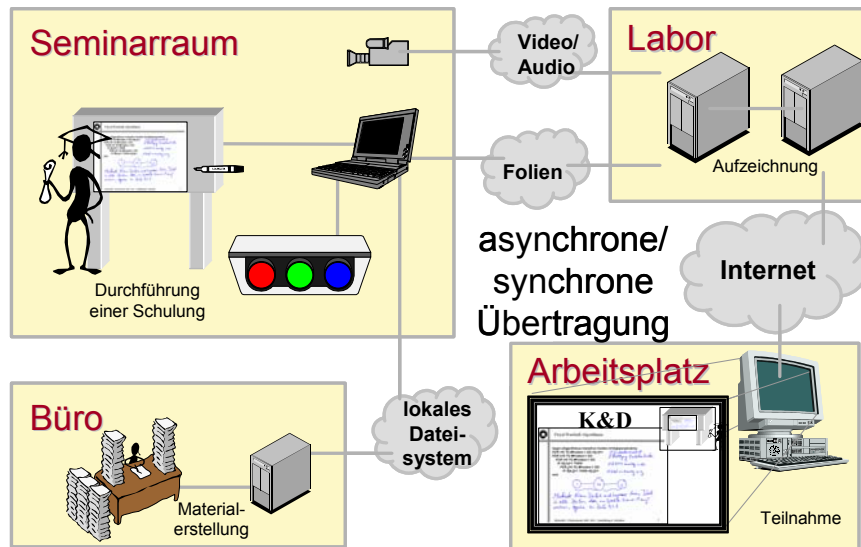


Abbildung 3.2: Internetbasierte Wissensvermittlung

Sowohl vom Büro des Autors als auch vom Seminarraum aus besteht Zugang zum lokalen Dateisystem, welches die Schulungsmaterialien beinhaltet. Während der mit Projektor und Whiteboard-Technik (vgl. Abschnitt 2.2.1) durchgeführten Schulung werden die Folien, und gegebenenfalls auch ein Videobild vom Dozenten, ins Labor übermittelt, dort aufgezeichnet und in Form eines Videos ins Internet übertragen. Damit hat der Lernende über seinen am weltweiten Netz angeschlossenen PC synchron zur Veranstaltung Zugriff auf die Videos [18]. Diese werden archiviert, so dass ein späterer Zugriff (asynchron zur Schulung) ebenfalls gewährleistet ist.

3.2 ANFORDERUNGEN

Der internetbasierte Wissenstransfer mit seinen Teilprozessen stellt eine Vielzahl von Anforderungen an eine unterstützende Architektur. Die Anforderungen des Dozenten, Autors, Entwicklers und Lernenden ergeben sich aus den verfolgten Zielen der Wissensvermittlung und aus der Entwicklung unterstützender Werkzeuge.

3.2.1 EINHEITLICHE UND ZENTRALE SPEICHERUNG DER SCHULUNGSMATERIALIEN

Eine Umgebung, in der Schulungsmaterialien von unterschiedlichen Stellen aus bearbeitet und in sehr verschiedenen Szenarien dargestellt werden müssen, benötigt eine einheitliche, homogene Datenablage. Diese Schulungsmaterialablage sollte so strukturiert und angelegt sein, dass ein einfacher und einheitlicher Zugriff möglich ist. Das betrifft nicht nur die Darstellung während der Präsentation durch den Dozenten oder die Bereitstellung in einem webbasierten Lernsystem, sondern auch einen möglichst einfachen Zugriff auf die Ablage durch Software-Schnittstellen für Entwickler, die unterstützende Werkzeuge realisieren wollen. Solche Werkzeuge könnten zum Beispiel eine graphische Dozentenumgebung sein oder Programme, welche einen schnellen und dem Einsatzszenario angepassten Überblick über die vorhandenen Materialien bieten und weitere, so genannte Meta-Informationen über diese liefern.

Eine solche einheitliche Sicht sollte zudem robust gegenüber Änderungen an der Ablage sein, das heißt, der Zugriff auf die Ablage muss möglichst transparent und unabhängig von der eingesetzten Speichertechnologie erfolgen können.

3.2.2 TRANSPARENTE UNTERSTÜTZUNG DER EINGESETZTEN TECHNOLOGIEN

Viele Informatiksysteme in der Aus- und Weiterbildung, speziell für die Unterstützung des Dozenten setzen, bei diesem ein hohes technisches Verständnis und genügend Hintergrundwissen voraus, um die benutzten Systeme richtig zu bedienen. Beim Einsatz von Informations- und Kommunikationstechnologien ist den Dozenten die Komplexität meist nicht bewusst. Dies führt oft zu Pannen bei der Veranstaltung, wie zum Beispiel nicht funktionierende Präsentationen, weil die Darstellungssoftware in einer veralteten Version vorliegt oder ein zum Abspielen benötigter Mediaplayer nicht installiert ist. Die Dozenten ärgern sich über die ihnen zur Verfügung stehenden Technologien, Gerätschaften und Apparaturen, anstatt sich auf ihre eigentliche Tätigkeit der Wissensvermittlung zu konzentrieren. So sollte vor einer Veranstaltung für den Dozenten so wenig wie möglich zu tun sein, er muss sich auf funktionierende Computertechnik und korrekt konfigurierte Software verlassen können. Aus diesen vielfach in der Praxis beobachteten Missständen [16] lassen sich folgende Anforderungen für die transparente Unterstützung der verwendeten Technologien identifizieren:

- Einmalige Konfiguration aller technischen Komponenten, das betrifft vor allem die bei einer elektronischen Präsentation eingesetzte Computertechnik sowie
- Schnelle Anpassung der Konfiguration an veränderte Bedingungen (z.B. Präsentation verlangt Internetzugang, aber wider Erwarten fällt dieser im verwendeten Seminarraum aus).

3.2.3 BEREITSTELLUNG EINER GRAPHISCHEN DOZENTENOBERVERFLÄCHE

Neben der Komplexität der eingesetzten Hardware kommt zumeist noch eine unüberschaubare Ansammlung von Anwendungen auf den Dozenten zu, falls dieser seine Schulung multimedial aufbereiten möchte. Der Umgang mit unterschiedlichsten Anwendungen erfordert wiederum ein gutes Know-how bezüglich der Bedienung der Werkzeuge bzw. das Wissen, welche Anwendung für eine bestimmte Art von digitalem Lehrmaterial benötigt wird. Trotzdem ist immer wieder auch bei „Profis“ zu beobachten, dass der Umgang mit den Werkzeugen oder der Aufruf von Dokumenten fehlerhaft verläuft. Es wäre für den Dozenten angenehmer, wenn er während der Veranstaltung die benötigten Materialien aus einer einfach zu bedienenden Umgebung heraus direkt aufrufen könnte, ohne wissen zu müssen, welches Präsentationswerkzeuge für das Material benötigt wird. Ein häufiger Wechsel zwischen den einzelnen Präsentationswerkzeugen wirkt sich zudem störend auf den homogenen Ablauf der Veranstaltung aus. Damit der Dozent während der Präsentation nicht den Überblick verliert und immer Bescheid weiß, welche Materialien schon aufgerufen wurden, wäre eine optische Anzeige, die dieses gewährleistet wünschenswert. Trotzdem kann es natürlich vorkommen, dass eine bereits aufgerufene Materialeinheit erneut gezeigt werden soll oder aber eine eingeplante Einheit übersprungen werden muss. Eine integrierte Umgebung zur Unterstützung einer elektronischen Präsentation sollte hier keinerlei Einschränkungen verlangen, sondern dem Dozenten diese Freiheiten einräumen. Folgende Anforderungen bezüglich der eingesetzten Software sind deshalb zu erfüllen:

- Transparente Nutzung der Präsentationswerkzeuge,

- Permanenter Überblick über den Ablauf der Schulung,
- Einfacher Aufruf der Schulungsmaterialien,
- Flexible Präsentationsabfolge,
- Wiederholender Aufruf der Schulungsmaterialien.

3.2.4 DURCHFÜHREN VON PRÄSENZ- UND ONLINEVERANSTALTUNGEN

Der Unterschied zwischen einer Präsenz- und Onlineveranstaltung sollte dem Dozenten während der Durchführung der Schulung nicht bewusst werden, das heißt die Übertragung bzw. Aufzeichnung der Schulung müsste transparent für den Dozenten sein. Zusätzliche Werkzeuge oder technische Komponenten, die für die Online-Veranstaltung zwingend erforderlich sind – wie zum Beispiel Software, die die Übertragung der Präsentation in ein Multimedialabor bewerkstelligt (siehe Abbildung 3.2) – und mit denen der Dozent in direkten Kontakt tritt, sollten ihn bei seiner eigentlichen Arbeit nicht behindern und möglichst intuitiv bedienbar sein. Falls eventuell aufgezeichnetes Videomaterial nach der Veranstaltung noch nachbearbeitet, z.B. geschnitten werden muss, wäre eine Vorbereitung dieser Schritte bereits während der Veranstaltung hilfreich:

- Transparente Übertragung bzw. Aufzeichnung der Schulung,
- Einsatz intuitiver Werkzeuge für die Onlineveranstaltung,
- Unterstützung der Nachbereitung des aufgezeichneten Materials.

3.3 ENTWURF

Ausgehend von den oben aufgezeigten Anforderungen gilt es nun, diese genauer zu analysieren, eine Architektur und unterstützende Werkzeuge zu modellieren und diese prototypisch zu implementieren. Zur Unterstützung von Analyse und Software-Entwurf wird die „Unified Modelling Language“, **UML**, verwendet. Die Notation der verschiedenen Diagramme richtet sich dabei weitgehend nach [8].

Betrachtet man eine Schulungsveranstaltung aus der Perspektive des Dozenten, so liegt es nahe, Anwendungsfälle zu erstellen, die diese Sicht darstellen. Bei genauerer Betrachtung zeigt sich, dass die Durchführung einer Schulungsveranstaltung immer zwei Teile umfasst: Zum einen die eigentliche Veranstaltung, in der der Dozent die verschiedenen Schulungsmaterialien präsentiert, und zum anderen natürlich auch die Vorbereitung der jeweiligen Schulung (siehe Abbildung 3.3).

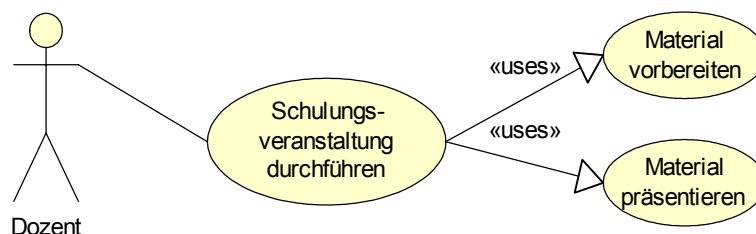


Abbildung 3.3: UML Anwendungsfall „Schulung durchführen“

Geht man davon aus, dass die Materialien bereits erstellt sind (z.B. weil der gleiche Kurs zu einem früheren Zeitpunkt in ähnlicher oder identischer Form schon einmal abgehalten wurde), so bleibt dem Dozenten trotzdem noch die Aufgabe, sich zu entscheiden,

welche Materialien er in der betreffenden Veranstaltung präsentieren möchte. Dies kann vom Wissensstand der Teilnehmer, von zeitlichen oder auch örtlichen bzw. technischen Gegebenheiten abhängen. Der Anwendungsfall „Material vorbereiten“ lässt sich also auf die beiden Fälle „Material sichten“ und „Material auswählen“ zurückführen. Bei der Betrachtung der vorhandenen Materialien und deren Auswahl soll der Dozent bereits von der Dozentenumgebung unterstützt werden (Abbildung 3.4)

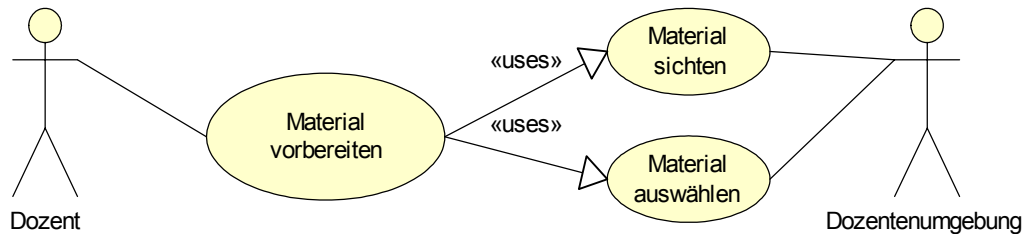


Abbildung 3.4: UML Anwendungsfall „Material vorbereiten“

Hat der Dozent sich für bestimmte Lehrmaterialien entschieden und will seine Veranstaltung halten, lässt sich dies in zwei weitere Anwendungsfälle untergliedern: Die Bereitstellung des Materials auf dem Präsentationsrechner und die dafür notwendigen Anwendungen sowie das eigentliche Aufrufen der Inhalte (Abbildung 3.5). Auch hierbei soll die Dozentenumgebung unterstützend mitwirken.

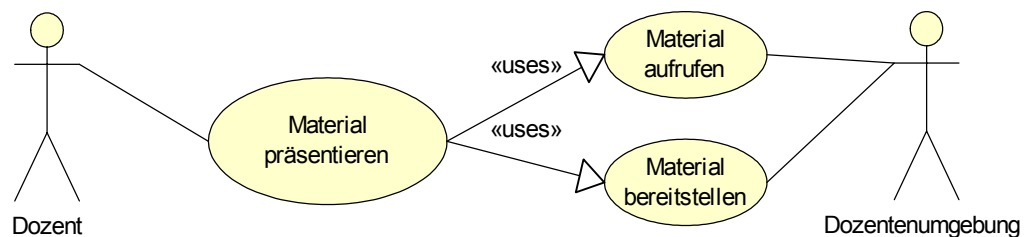


Abbildung 3.5: UML Anwendungsfall „Material präsentieren“

Ausgehend von diesen Anwendungsfällen werden einzelne, noch konzeptionelle Klassen herausgearbeitet, welche die Aufgabe einer den Dozenten unterstützenden graphischen Umgebung auf abstrakter Ebene modellieren sollen. Konzeptionelle Klassen haben eine natürliche Beziehung zu den implementierenden Klassen, es gibt aber keine direkte Abbildung, das heißt, die eigentliche Softwarestruktur ist an dieser Stelle noch nicht festgelegt [8]. Neben dem Akteur „Dozentenumgebung“ sind dies auch die im Kapitel 3.2.1 vorgestellte Schulungsmaterialablage und die graphische Visualisierung dieser Ablage bzw. eines ausgewählten Kurses. Die Klasse „Kursstruktur“ stellt dabei den betrachteten Ausschnitt aus der Schulungsmaterialablage dar. Die zu diesem Zeitpunkt noch sehr oberflächlich und implementierungsfern modellierten Klassen und deren Zusammenhang untereinander zeigt Abbildung 3.6.

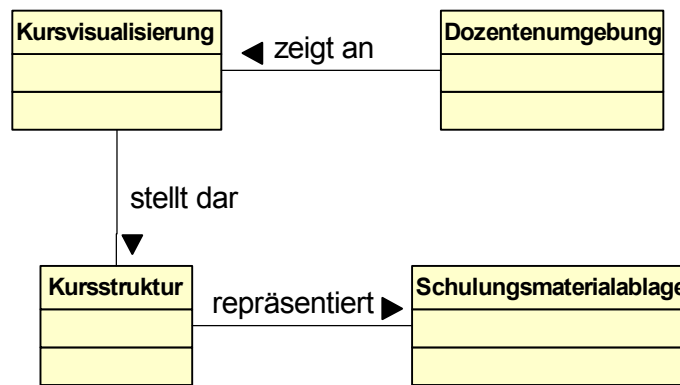


Abbildung 3.6: Abstraktes UML Klassendiagramm

Die Kursstruktur repräsentiert dabei die ganze oder auch nur einen Teil der Schulungsmaterialablage, z.B. einen einzelnen Kurs oder ein einzelnes Kapitel eines Kurses. Diese Datenstruktur wird schließlich graphisch durch die Klasse „Kursvisualisierung“ dargestellt und dem Dozenten in der Dozentenumgebung präsentiert.

Die zeitliche Abfolge der Interaktion der einzelnen Klassen ist als UML Sequenzdiagramm in Abbildung 3.7 zu sehen.

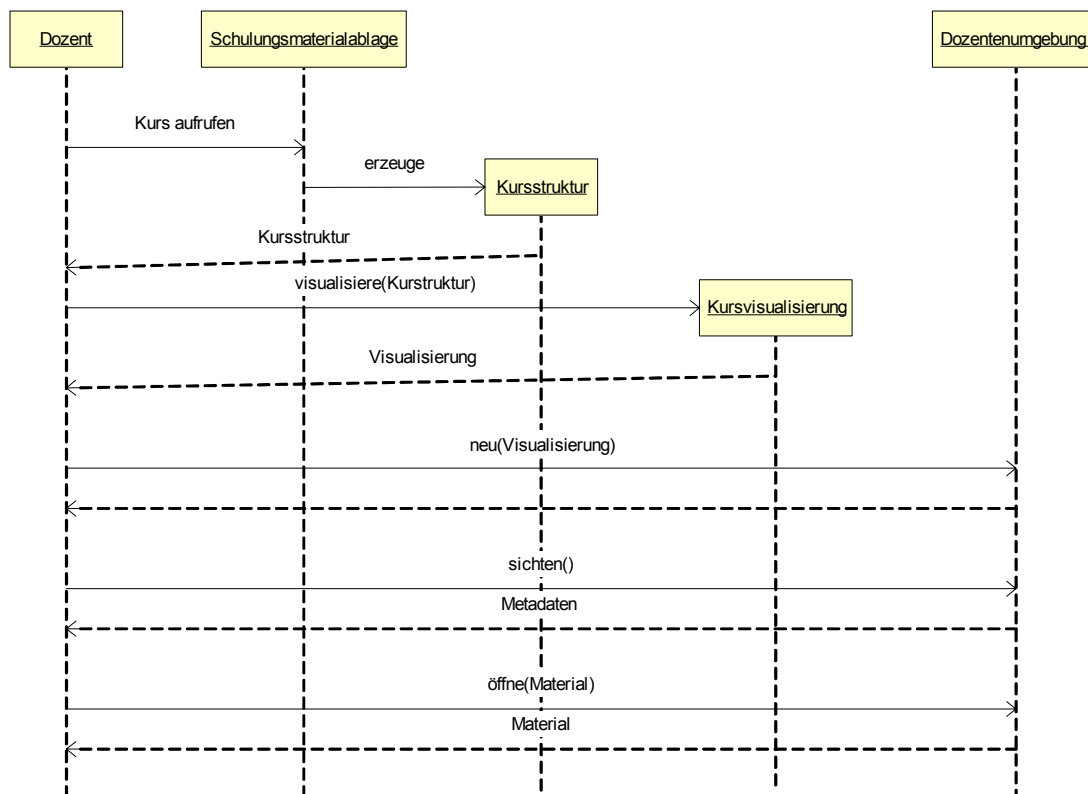


Abbildung 3.7: UML Sequenzdiagramm des Anwendungsfalls „Material sichten“

Das Diagramm geht noch ein wenig weiter: Es zeigt bereits die Interaktion des Dozenten mit der ihn unterstützenden Umgebung bei der – durch den Anwendungsfall „Material sichten“ modellierten – Auswahl einzelner Inhalte. Dazu muss sich der Dozent zunächst für einen Kurs entscheiden und die Kursstruktur generieren lassen. Anschließend bereitet die Klasse „Kursvisualisierung“ diese Struktur graphisch auf, so dass sie dann in die Dozentenumgebung integriert werden kann. Dabei muss in jedem Falle die

Möglichkeit geboten werden, in die jeweiligen Materialien „hineinzuschauen“. Das heißt, der Dozent soll nicht nur die Möglichkeit haben, sich einen Überblick über die vorhandenen Materialien zu verschaffen, sondern sich auch vergewissern können, welche konkreten Inhalte sich hinter den jeweiligen Lehrelementen verbergen. Dies ermöglichen die Aufrufe „sichten()“ und „öffne(Material)“, welche die Dozentenumgebung dem Nutzer zur Verfügung stellen soll.

Bedingt durch die Tatsache, dass viele Lehrmaterialien oft in mehrfacher Ausprägung vorhanden sind (z.B. im Format Microsoft Powerpoint für die Präsentation und im Postscript-Format als Druckvorlage), während der Veranstaltung dagegen oft nur eine bestimmte Version benötigt wird, stellt sich eine weitere benötigte Funktionalität heraus: Die Dozentenumgebung soll zwar nur die vom Dozenten ausgewählten Materialien darstellen, jedoch zusätzlich die Funktion bieten, Materialien die noch nicht für die Verwendung in einer Veranstaltung vorgesehen sind, für genau diesen Zweck zu „veröffentlichen“. Das folgende Zustandsübergangsdiagramm (Abbildung 3.8) zeigt, welche Funktionalität die Dozentenumgebung haben könnte und welche Zustände dabei auftreten.

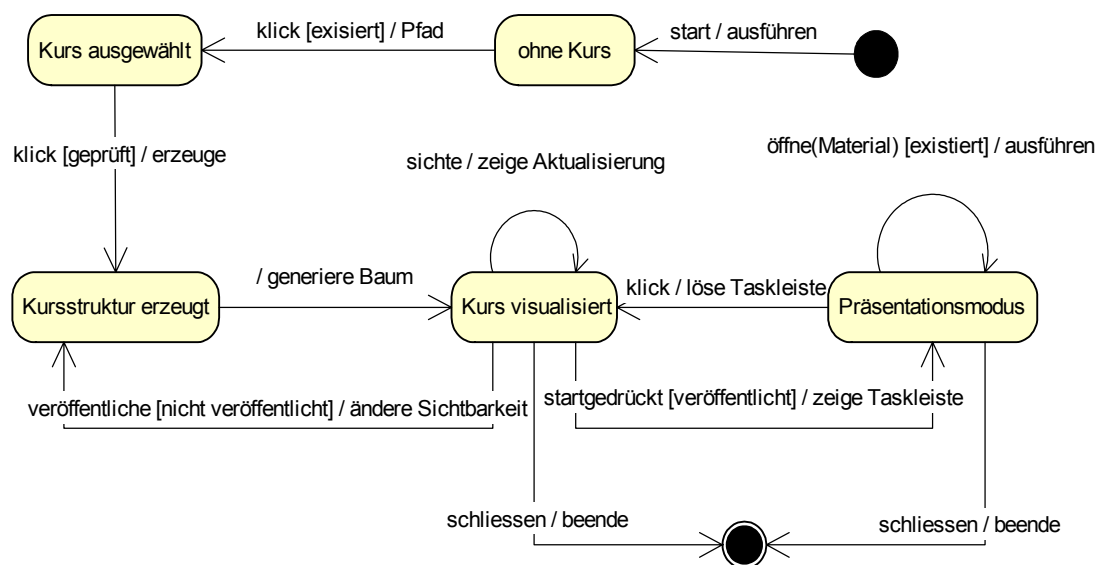


Abbildung 3.8: Zustandsübergänge der Klasse „Dozentenumgebung“

Unmittelbar nach dem Start der Umgebung muss zunächst einmal bestimmt werden, welcher Kurs benötigt wird. Nach der erfolgreichen Auswahl eines Kurses (in Form eines Dateisystempfades) soll dieser zunächst visualisiert werden. Das ist jedoch nur dann möglich, wenn die Speicherung des Kurses einer festgelegten Konvention entspricht und deren Einhaltung vorher erfolgreich geprüft wurde. Eine genaue Beschreibung dieser Konventionen findet sich in Abschnitt 4.1. Ist die Kursstruktur erzeugt, kann sie sofort visualisiert werden. Hierzu wird eine baumartige Darstellungsform gewählt, wie man sie von gängigen Dateisystembrowsern her kennt. In dieser Darstellung soll der Dozent die Möglichkeit haben, durch die einzelnen Inhalte zu navigieren und sich die für die Präsentation notwendigen Materialien zusammenzustellen. Dabei muss die Baumdarstellung ständig aktualisiert werden. Hat der Dozent eine Materialeinheit gefunden, die noch nicht für die Verwendung in einer Vorlesung gedacht ist und die aber trotzdem für die betreffende Veranstaltung benötigt wird, so müssen deren Sichtbarkeitsattribute geändert werden. Diese Tätigkeit wird im Zustandsübergangsdiagramm mit dem Ereignis „veröffentliche“ modelliert. Ausgehend von der Annahme, dass sich

bei dem Vorgang die Kursstruktur ändert, wird diese neu aufgebaut und die Visualisierung aktualisiert.

Der Zustand „Präsentationsmodus“ repräsentiert die Situation, in welcher der Dozent die für die Veranstaltung veröffentlichten Materialien aus einer Art Taskleiste heraus direkt starten kann. Die Taskleiste soll dabei nur die für die jeweilige Veranstaltung ausgewählten Lehrelemente anzeigen. Der Dozent soll dabei die Möglichkeit haben, die Taskleiste beliebig ein- und ausblenden, also die Betriebsmodi wechseln zu können. Weiterhin soll es möglich sein, das Werkzeug von beiden Modi aus beenden zu können.

Aus dem Präsentationsmodus heraus werden dann die Materialien gestartet. Voraussetzung ist natürlich, dass das Material existiert und angezeigt werden kann. Denn um ein möglichst breites Spektrum an verwendbaren Materialformaten realisieren zu können, sollen für das eigentliche Anzeigen der Materialien die dafür gängigen Programme und Autorenwerkzeuge zuständig sein. Die Dozentenumgebung übernimmt lediglich das automatisierte Aufrufen dieser Werkzeuge: Sie bestimmt den Typ des zu startenden Materials (z.B. PPT oder HTML), ermittelt die zuständige Anwendung („Powerpoint“ bzw. „Internet Explorer“), startet diese und übergibt ihr dann das Material. Die Dozentenumgebung ersetzt also nicht die üblichen Präsentationswerkzeuge, sondern bringt sie „unter einen Hut“, macht deren Nutzung also für den Dozenten transparent.

Diese Vorgänge sind nochmals als Aktivitätsdiagramm in Abbildung 3.9 dargestellt: Nach dem Start der Umgebung und der Auswahl eines vorhandenen Kurses wird die Kursstruktur generiert. Sollte der Kurs nicht geprüft sein – also die Vorschriften der Materialablage nicht erfüllen – kehrt die Umgebung wieder in den Ausgangszustand zurück. Ist der Kurs jedoch syntaktisch in Ordnung, können die Baumvisualisierung und die Taskleiste zum direkten Zugriff auf die veröffentlichten Materialien erzeugt werden. Da beide Vorgänge nur auf der Kursstruktur arbeiten und nicht voneinander abhängen, kann dies parallel geschehen.

Nun besteht die Möglichkeit, noch nicht veröffentlichtes Material für die Verwendung in der Dozentenumgebung zu aktivieren, was wiederum ein erneutes Erzeugen der Kursstruktur, der Visualisierung und der Taskleiste zur Folge hat. Parallel dazu können die Materialien gestartet werden.

Letzte Aktivität der Dozentenumgebung ist in jedem Falle das ordentliche Beenden der Software.

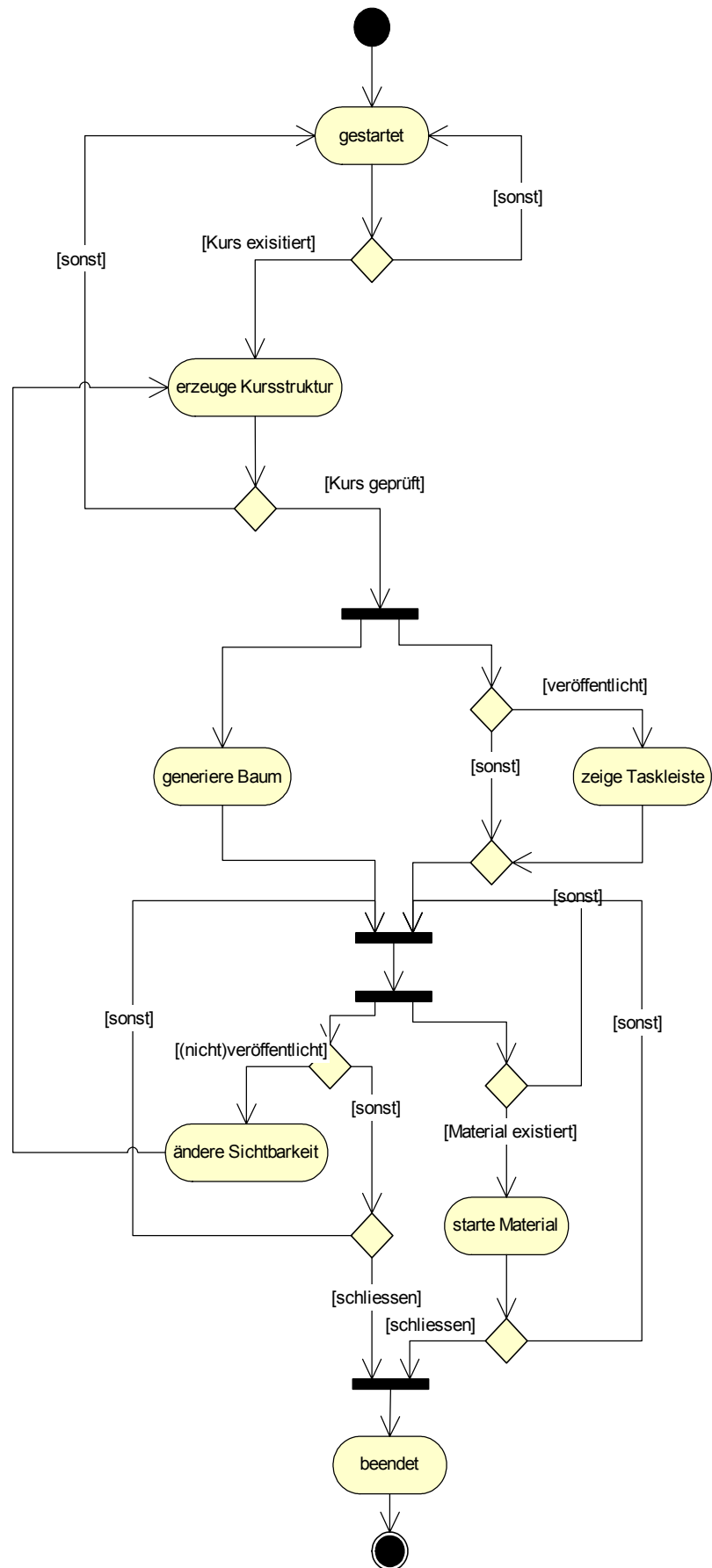


Abbildung 3.9: UML Aktivitätsdiagramm der Dozentenumgebung

4 ARCHITEKTUR

Die Erstellung, Aufbereitung und Vermittlung von Wissensmaterialien in verschiedenen Szenarien wie Präsenzveranstaltung, Live-Veranstaltung und Remote-Veranstaltung stellt sehr unterschiedliche Anforderungen an die Materialien und deren Speicherung. Um mit einer zentralen strukturierten Ablage arbeiten zu können, ist es unumgänglich, diese in einer genau festgelegten Form zu halten wie im folgenden Kapitel beschrieben. Da aber Autor, Dozent und Lernender an die Autoren-, Dozenten- und Lernumgebung sehr unterschiedliche Anforderungen stellen, eine jeweils andere Sicht auf die Materialien benötigen und sich der direkte maschinelle Zugriff auf die strukturierte Schulungsmaterialablage in Form eines Dateisystems als unhandlich und wenig flexibel erweist, wurde eine Mehr-Schichten-Architektur entwickelt, die vom Dateisystem abstrahiert und einen einfacher und flexibler nutzbaren Zugang zur Schulungsmaterialablage bietet. Die drei Schichten sind in Abbildung 4.1 dargestellt und unterteilen sich in:

1. **Datenhaltung:** Die unterste Schicht, sie speichert die Daten und bietet einen datei- und betriebssystemspezifischen Zugriff auf die Materialablage.
2. **Geschäftslogik:** Sie abstrahiert von der Datenhaltung und bietet eine – von der Datenhaltung unabhängige – einheitliche Sicht auf die Schulungsmaterialien.
3. **Präsentation:** Diese anwenderorientierte Schicht stellt die von der Geschäftslogik angebotene Sicht auf die Ablage graphisch dar.

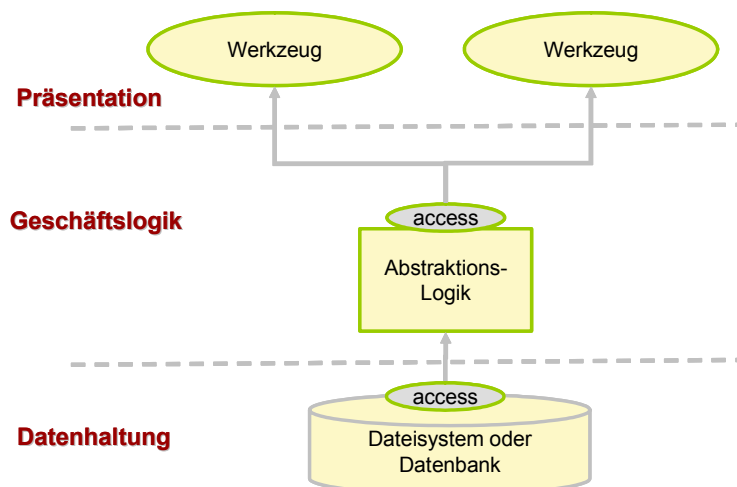


Abbildung 4.1: 3-Schichten-Architektur

4.1 DIE DATENHALTUNGS-SCHICHT

Die Datenablage bildet die Basis jedes Informationssystems. Unabhängig davon, ob es sich dabei um ein datenbankbasiertes Ablagesystem oder ein „einfaches“ Dateisystem handelt, ist es von großer Bedeutung, diese Ablage strukturiert, sinnvoll gegliedert und dem Verwendungszweck angepasst zu gestalten.

Zum Einsatz kommt das bekannte hierarchisch organisierte Dateisystem Microsoft NTFS 5. Hierarchisch heißt, dass das Dateisystem die einzelnen Dateien in einer baumartigen Struktur speichert, wobei Dateien in Verzeichnissen zusammengefasst werden,

welche wiederum in übergeordneten Verzeichnissen hierarchisch abgelegt werden können. Das bedeutet, dass sich die Zugehörigkeit der einzelnen Lehrmaterialien – also einzelner Dateien – zu einem Kurs schon auf Dateisystemebene realisieren lässt, indem alle einem Kurs zugeordneten Dateien im selben Verzeichnis abgelegt werden. Weiterhin ist es leicht möglich, eine Datei von einer beliebigen Stelle im Verzeichnisbaum aus zu referenzieren. Mit diesen als „Verknüpfung“ bezeichneten Verweisen lassen sich Dateien mehrfach verwenden, müssen aber nur an einer Stelle gepflegt werden. NTFS bietet sich noch aus anderen Gründen für eine zentrale Materialablage an: es realisiert neben Benutzerverwaltung, Vergabe von Zugriffsrechten, Komprimierung und Verschlüsselung auch Journaling auf Dateisystemebene. Diese Transaktionsorientiertheit ist in einer verteilten Umgebung sehr wichtig, denn sie verhindert, dass die Datenablage durch gleichzeitigen Zugriff mehrerer Benutzer korrumpiert wird. Einen kurzen Überblick über NTFS geben [10] und [12].

Um einen automatisierbaren Zugriffsmechanismus auf die Materialablage realisieren zu können, ist es unumgänglich, die einzelnen Verzeichnisse und Dateien in maschinenlesbarer Form zu halten. Das heißt, die Namensgebung muss zwingend den vorgegebenen Konventionen entsprechen. Auf den ersten Blick mag das eine Einschränkung sein, aber damit ist es möglich, bereits auf Dateisystemebene allein durch die Namensgebung die in Abschnitt 4.1.2 vorgestellte Klassifikation der Materialien und deren Sichtbarkeit auszudrücken.

4.1.1 VERZEICHNISSE

Verzeichnisse realisieren die Strukturierung der Ablage in einzelne Kurse, Kapitel, Unterkapitel und Reihenfolge. Ein Verzeichnisname muss den folgenden Konventionen entsprechen:

Nummerierung.Name

Die Nummerierung entspricht dabei folgendem Schema:

Kapitel-Unterkapitel-...

Die Nummern müssen genau zweistellig sein, eventuell vorhandene einstellige Nummern erhalten eine vorangestellte Null. Trennzeichen der einzelnen Nummern ist in jedem Fall der Bindestrich „-“.

Der Name ist frei wählbar, jedoch mit folgenden Einschränkungen:

- Nur Kleinbuchstaben,
- Keine Umlaute, keine Sonderzeichen außer Bindestrich (-) und Unterstrich (_),
- Leerzeichen werden durch Unterstriche ersetzt,
- Punkte werden nur zur Trennung der Namensteile verwendet, nicht aber in den Namen selbst.

4.1.2 DATEIEN

Die Dateien enthalten die eigentlichen Lehrmaterialien und liegen je nach Verwendung meist in den gängigen Formaten „ppt“, „pdf“, „html“ oder „doc“ vor. Um die automatische Verwaltung zu ermöglichen, müssen auch sie bestimmte Namensvorschriften einhalten. Sie haben die folgende Form:

Präfix.Name.Format

Das Präfix beschreibt die Klassifikation der Datei in Haupt- oder Zusatzmaterial und die Sichtbarkeit, das heißt, in welcher Umgebung die Datei dargestellt werden soll. Hierbei gibt es drei Sichtbarkeitsgruppen: „Learner“, „Docent“ und „Author“. Ebenfalls ist es möglich auszudrücken, dass die Datei keiner der drei Gruppen zugehört, ihre Sichtbarkeit also unspezifiziert ist.

Die Ausdrucksmächtigkeit des Dateipräfixes lässt sich durch einen logischen Ausdruck darstellen:

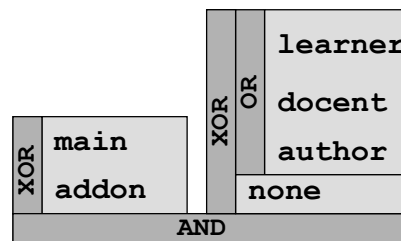


Abbildung 4.2: Ausdrucksmöglichkeiten im Dateipräfix

Die Klassifikation in „main“ wird durch einen vorangestellten Unterstrich im Dateinamen ausgedrückt. Fehlt dieser, so gilt die Datei als Zusatzmaterial. Die Zugehörigkeit zu einer Sichtbarkeitsklasse wird durch die Kleinbuchstaben „a“, „d“ und „l“ dargestellt. Diese dürfen in beliebiger Kombination und Reihenfolge dort auftreten. Soll die Zugehörigkeit einer Datei noch unspezifiziert bleiben, ist anstelle der „a-d-l-Kombination“ ein „n“ einzutragen. So ergibt sich für das Dateipräfix die Vorschrift formuliert als regulärer Ausdruck:

$$\text{Präfix} = (_)^{0,1} ((a, d, l)^{1,3}, n)^1$$

Für den Namen gelten die gleichen Vorschriften wie bei Verzeichnissen, hinter dem Format steckt die unter Windows übliche Dateierweiterung, die den Typ des Inhalts beschreibt.

4.1.3 METADATEN

Metadaten liegen als XML-Dateien [14] vor und haben die Aufgabe, das jeweils zugehörige Verzeichnis bzw. die zugehörige Materialdatei zu beschreiben. Dazu gehören Informationen über Titel, Autor, Sprache, Dauer, Format und charakterisierende Schlüsselbegriffe („Keywords“) um die Recherche zu erleichtern. Eine typische Metadatenfile ist in Kapitel 6.1.2 dargestellt.

Metadaten für Verzeichnisse werden innerhalb des zu beschreibenden Verzeichnisses abgelegt, der Name wird gebildet aus dem Namen des Verzeichnisses, zwei vorangestellten Unterstrichen und der Erweiterung „.meta.xml“:

`__Name.meta.xml`

Ein Beispiel zu den Verzeichnisnamen und der zugehörigen Metadatei zeigt Abbildung 4.3.

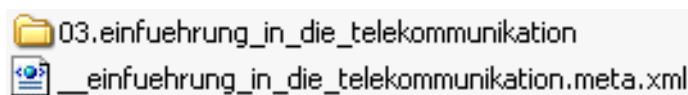


Abbildung 4.3: Metadaten für Verzeichnisse

Die Metadaten für Dateien liegen im gleichen XML-Format vor wie diejenigen für Verzeichnisse, der Name einer Metadatenfile bildet sich aus dem kompletten Namen der zu beschreibenden Datei und einem angehängten „.meta.xml“:

Präfix.Name.Format.meta.xml

Ein Beispiel hierzu zeigt Abbildung 4.4.

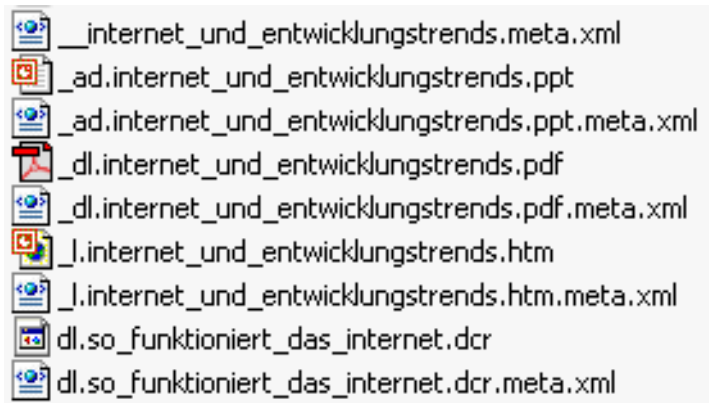


Abbildung 4.4: Dateinamen und Metadateien

4.2 DIE GESCHÄFTSLOGIK-SCHICHT

Aufgabe dieser Schicht ist die Realisierung des Zugangs zur Datenablage. Dazu wird die gesamte Schulungsmaterialablage maschinell erfasst und alle Informationen, die durch ihre Verzeichnisstruktur und den Namen der einzelnen Materialien kodiert sind, in einen **Information Container** mittels XML abgebildet. Dieses XML-Dokument wird vom **Content Information Server (CIS)** erzeugt und allen auf dieser Geschäftslogikschicht aufsetzenden Werkzeugen zur Verfügung gestellt. Die Architektur ist in Abbildung 4.5 dargestellt.

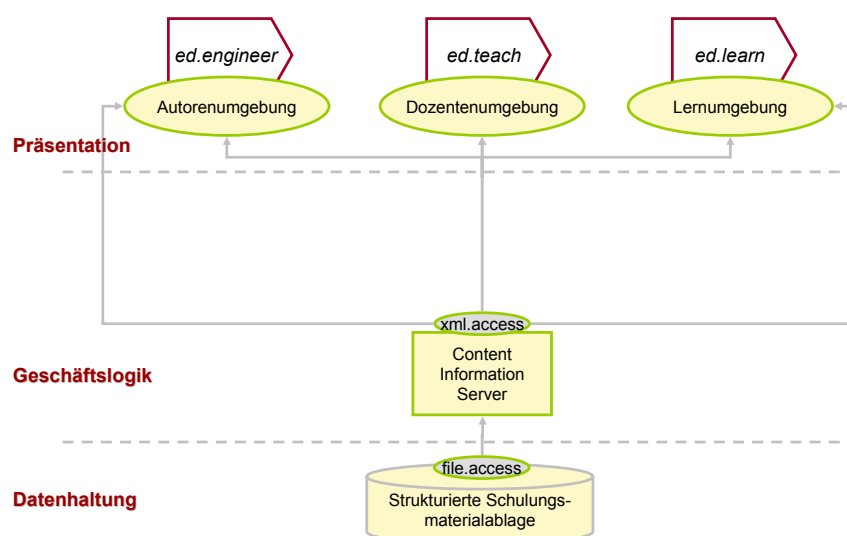


Abbildung 4.5: Architektur

Die Darstellung der strukturierten Schulungsmaterialablage in Form des *Information Containers* hat für die zugreifenden Werkzeuge Autoren-, Dozenten- und Lernumgebung einige Vorteile. Zum einen ist die zugrunde liegende Datenstruktur mit ihren Eigenschaften wie Verzeichnisstruktur, Dateinamen, Dateityp u.ä. nun vollkommen transparent für die aufsetzenden Werkzeuge. Dies hat den Vorteil, dass sich eine eventuelle Änderung an der Struktur der Schulungsmaterialablage nur an einer Stelle auswirkt, nämlich im *Content Information Server*, die Präsentationswerkzeuge jedoch davon unberührt bleiben, solange sich die Schnittstelle zur Ablage (hier mit *xml.access* bezeichnet) nicht ändert. Der zweite große Vorteil dieses Ansatzes ist technischer Natur und betrifft die wesentlich vielfältigeren Möglichkeiten was den Zugriff und die Handhabung des *Information Container* betrifft. Leistungsfähige Werkzeuge zum Durchlaufen und Suchen in XML-Daten sowie mächtige Filterwerkzeuge stehen zur Verfügung, die es den Entwicklern der Präsentationswerkzeuge erleichtern, auf Basis der XML-Datenstruktur ihre Werkzeuge zu implementieren [13].

Die Aufgabe des Content Information Servers besteht also darin, die im Dateisystem vorliegende strukturierte Schulungsmaterialablage in eine für Entwickler handhabbarere Form zu überführen. Das Dateisystem enthält durch die Festlegung von Konventionen über die Namensgebung von Verzeichnissen, Dateien und Metadateien vielfältige Informationen, die aber in der Form, wie sie im Dateisystem gespeichert sind, für Mensch und Maschine recht unhandlich sind.

Diese Informationen werden in Form der durch das World Wide Web Consortium (W3C) standardisierten Extensible Markup Language XML dargestellt [14]. Aufgabe des Content Information Servers ist es, dieses XML-Dokument anzulegen.

Hauptziel beim Entwurf des Content Information Servers war die Realisierung in einer Art und Weise, die es erlaubt, die Funktionalität möglichst einfach und ohne großen Aufwand in anderen Programmen zu verwenden. Neben der Hauptaufgabe, das Dateisystem in XML abzubilden, bietet der Content Information Server noch weitere Funktionen, die den Zugriff auf das Dateisystem vereinfachen sollen. Dabei sollte das System so modular wie möglich aufgebaut werden, um möglichst flexibel auf neue Anforderungen oder Änderungen reagieren zu können.

So wurde der CIS in drei Bereiche unterteilt:

1. Den eigentlichen Information Server, der das Dateisystem in einer Tiefensuche durchläuft und eine Hierarchie in XML erzeugt (Klasse `cis_cacher.cacher`).
 2. Die Funktionen, die die Dateien betreffen (Klassen `coursematerial.*`):
 - Prüfen, ob eine Datei die Namensrichtlinien einhält (Klasse `validate`),
 - Funktionen zum Prüfen auf Vorhandensein von Metadaten (Klasse `metadata`),
 - Funktionen zum Erstellen und Löschen von Dateien (Klasse `materialgateway`),
 - Routinen, mit denen sich leicht abfragen lässt, zu welchem Sichtbarkeitsbereich eine Datei gehört und welche Klassifizierung sie besitzt (Klassen `visibility` und `classification`), sowie
 - Funktionen, diese Sichtbarkeit und Klassifizierung zu ändern (Klasse `materialgateway`).
-

3. Der Bereich, der mit Verzeichnissen arbeitet (Klassen `coursestructure.*`). Dies beinhaltet Funktionalität zum Prüfen,
 - ob ein Verzeichnis Metadaten enthält (Klasse `metadata`),
 - ob die Namenskonventionen eingehalten wurden (Klasse `validate`), sowie
 - Prozeduren zum Erstellen und Löschen von Verzeichnissen (Klasse `structuregateway`).

Das komplette Klassendiagramm des Content Information Server einschließlich aller öffentlich zugänglicher Methoden und Parameter sowie der Zusammenhang der einzelnen Klassen untereinander ist in der folgenden Abbildung 4.6 zu sehen. Der Zugriff des CIS auf das eigentliche Dateisystem erfolgt mit Hilfe des Windows Scripting Host (WSH), dessen Fähigkeit mit Regulären Ausdrücken umgehen zu können, genutzt wird, um die Dateinamenskonformität der Materialien sicherzustellen. Sämtliche Materialien oder Verzeichnisse, die nicht den im vorhergehenden Kapitel vorgestellten Konventionen entsprechen, werden ignoriert.

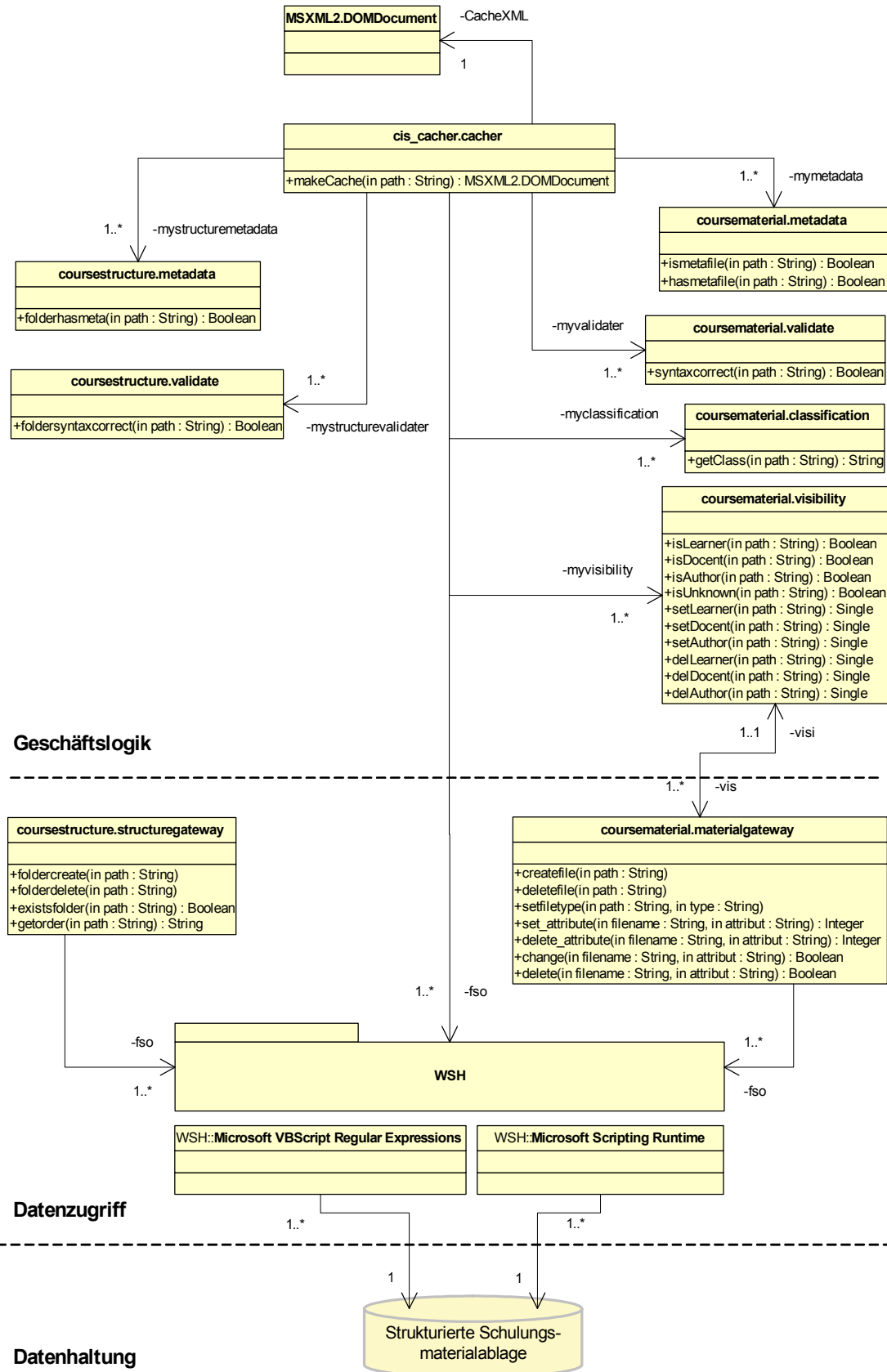


Abbildung 4.6: UML Klassendiagramm des CIS

4.3 DIE PRÄSENTATIONSLOGIK-SCHICHT

Die Praxis hat gezeigt, dass die verschiedenen Werkzeuge wie z.B. Autorenumgebung und Dozentenumgebung eine weitere Anforderung stellen: die übersichtliche Darstellung der Schulungsmaterialablage. Jedoch ist die direkte Ansicht der Ablage – so, wie sie im Dateisystem vorliegt – ungeeignet, weil zu technisch. Möchte sich zum Beispiel der Dozent einen Überblick verschaffen, welche Materialien für eine Veranstaltung zur Verfügung stehen, in welchen Formaten sie vorliegen und welche Prioritäten sie besitzen, so ist es umständlich, schwierig und ermüdend, in der Schulungsmaterialablage die entsprechenden Dateien anhand ihrer im Dateinamen kodierten Eigenschaften zu erfassen und sich einen Überblick zu bilden.

Aus diesem Grund wurde zwischen der Geschäftslogik und der Präsentation eine weitere Schicht eingeführt. Die Präsentationslogikschicht mit der zentralen Komponente **Content Information Browser (CIB)** hat die Aufgabe, die in der Schulungsmaterialablage inhärenten Informationen zu visualisieren und sich dabei leicht in vorhandene oder noch zu entwickelnde Werkzeuge integrieren zu lassen. Der Browser setzt dabei vollständig auf die vom *Content Information Server* bereitgestellten *Information Container* auf, d.h. auch er ist vollkommen unabhängig von Änderungen an der Struktur der Materialablage und des ihr zugrunde liegenden Dateisystems. Diese Architektur soll nochmals mit Hilfe von Abbildung 4.7 verdeutlicht werden.

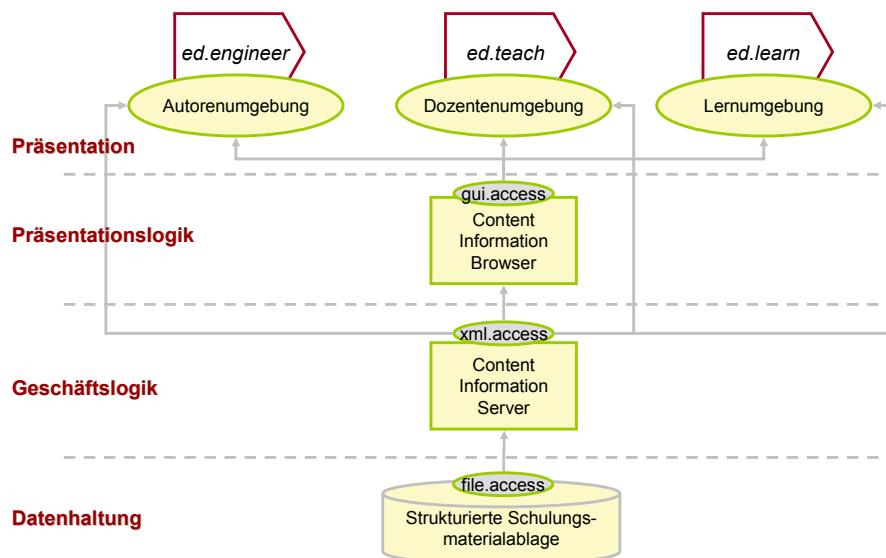


Abbildung 4.7: Architektur mit Content Information Browser

Einem Präsentationswerkzeug stehen somit neben einer leistungsfähigen Datenstruktur, dem *Information Container*, auch eine Präsentationslogik zur Verfügung, die es dem Entwickler leicht ermöglicht, die unterschiedlichen Informationen welche die Schulungsmaterialablage bietet, in seinem Werkzeug graphisch ansprechend und übersichtlich darzustellen.

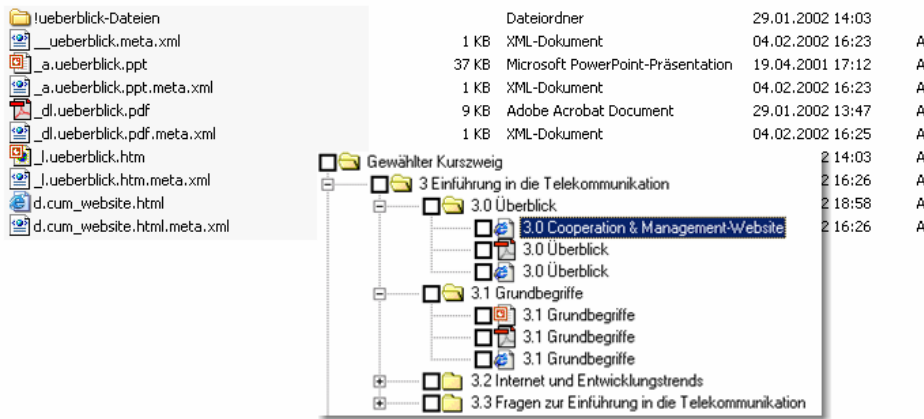


Abbildung 4.8: Graphische Darstellung der Schulungsmaterialablage

Abbildung 4.8 verdeutlicht den Unterschied zwischen der direkten Darstellung einer Schulungsmaterialablage durch das Dateisystem und einer ansprechenden graphischen Darstellung durch den Content Information Browser.

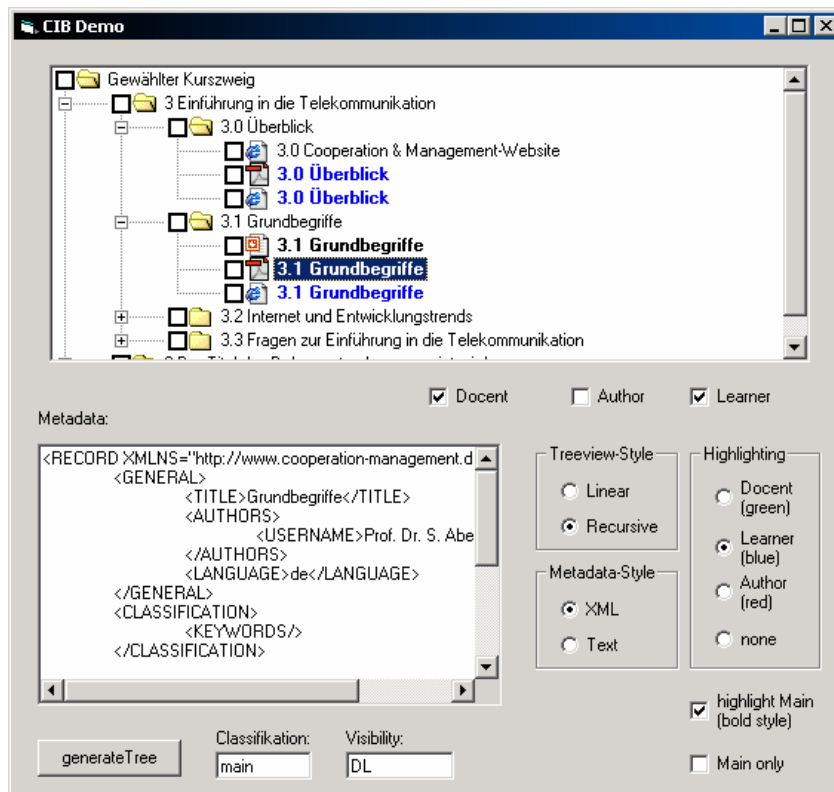


Abbildung 4.9: Demonstration des Content Information Browsers

Abbildung 4.9 zeigt beispielhaft, wie eine Nutzung des CIB aussehen könnte und welche Möglichkeiten geboten werden:

- Darstellung des Kurses auf zwei verschiedene Arten: Entweder werden sämtliche Verzeichnisse direkt unterhalb der Baumwurzel als lineare Folge platziert, oder aber die originale Verzeichnisstruktur wird komplett hierarchisch abgebildet,

- Anzeigen von Material entsprechend seiner Zielgruppe, z.B. könnte ausschließlich Dozentenmaterial dargestellt werden,
- Auslesen der jedem Material zugeordneten Zielgruppenkodierung und graphische Hervorhebung der gewünschten Zielgruppe,
- Ausfiltern von Material entsprechend seiner Priorität und Hervorhebung von höher priorisiertem Material,
- Auslesen der jedem Material zugeordneten Metadaten.

Der Content Information Browser hat die Aufgabe, die vom im vorangegangenen Kapitel vorgestellten CIS erzeugte hierarchische XML-Struktur graphisch als Baum darzustellen.

Dabei sollen verschiedene Darstellungsweisen sowie Filterung möglich sein. Da die strukturierte Schulungsmaterialablage drei verschiedene Sichtbarkeitsklassen (Dozent, Autor und Lernender) bietet sowie eine Unterscheidung in Haupt- und Zusatzmaterial realisiert, soll der Entwickler, der den CIB nutzen möchte, die Möglichkeit haben, entsprechend diesen Kriterien zu filtern. Als Beispiel wäre die in den Kapiteln 5 und 6.3 vorgestellte Dozentenumgebung zu nennen: Dort ist nur das für den Dozenten freigegebene Material von Interesse und so wird man eine Darstellung bevorzugen, in der auch nur eben dieses Material angezeigt wird.

Abbildung 4.10 zeigt den Entwurf des CIB als UML Klassendiagramm mit den Methoden, welche die im vorherigen Abschnitt geschilderten Nutzungsmöglichkeiten realisieren sowie dessen Assoziationen zu den zusätzlich benötigten Klassen `TreeView`, `ImageList` und den Komponenten eines `TreeViews`, dem Knotenobjekt `MSComctlLib.node`. Sie sind für die eigentliche graphische Darstellung notwendig und werden von der Entwicklungsumgebung bereitgestellt.

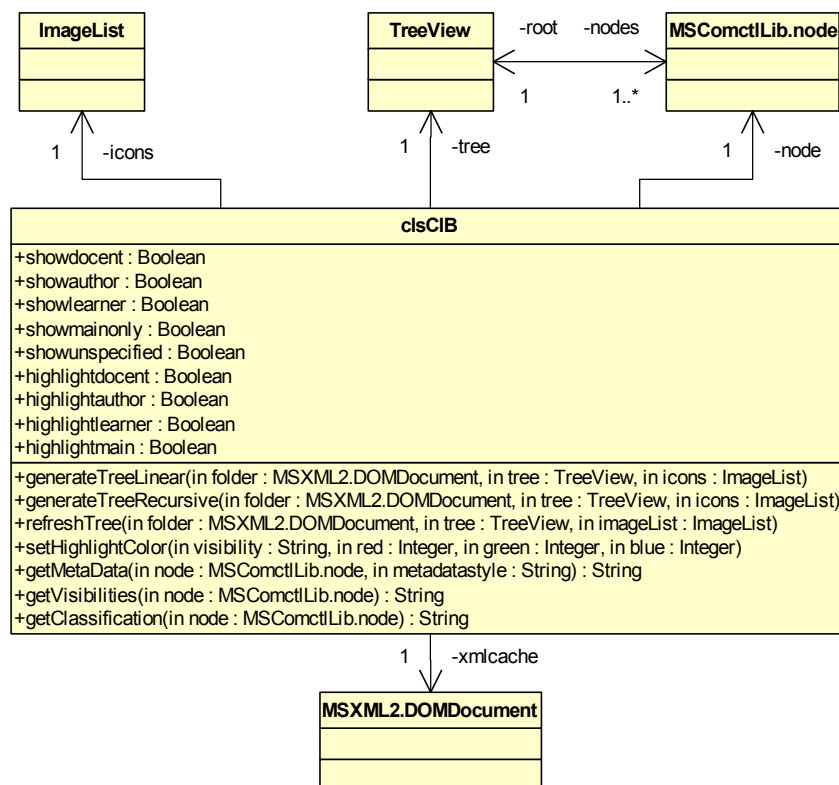


Abbildung 4.10: UML Klassendiagramm des CIB

Ein `TreeView` ist eine Klasse, die unter Windows zur Verfügung steht, wenn der Entwickler graphische baumartige Strukturen realisieren will, wie man sie aus vielen Anwendungen und Dateisystem-Werkzeugen wie zum Beispiel dem „Windows-Explorer“ kennt. Diese Klasse enthält beliebig viele Referenzen auf ihre Knotenobjekte, die `MSComctlLib.node`-Klasse. Über zahlreiche Funktionen kann der Entwickler einem solchen Baum an jeder beliebigen Stelle Knoten hinzufügen, diesen Knoten eindeutig bezeichnen und ihm ein Icon zuweisen. Zu diesem Zweck stellt die Entwicklungsumgebung die Klasse der `ImageList` zur Verfügung. Beim Einfügen eines Knotens in den Baum kann so mit einer Referenz auf einen Index der Bildliste diesem Knoten ein Icon zugefügt werden.

Eine genaue Beschreibung, wie der Entwickler den CIB nutzen kann sowie Codebeispiele finden sich im Abschnitt 6.2.2.

4.4 ZUSAMMENSPIEL DER SCHICHTEN

Der Content Information Browser arbeitet dabei nur auf der vom Content Information Server angebotenen XML-Struktur: Die benutzende Anwendung muss also zunächst ein XML-DOMDocument vom CIS generieren lassen und dieses dem CIB-Objekt beim Aufruf von dessen Funktionen als Parameter übergeben. Hierdurch werden die vorgestellte Schichtenarchitektur (Abbildung 4.11) und die Verdeckung des eigentlichen Dateisystems realisiert.

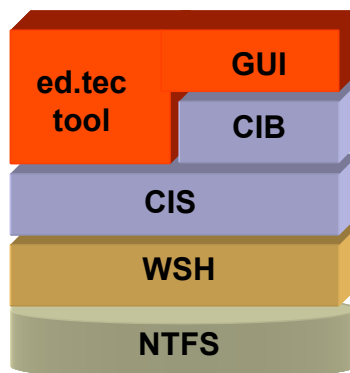


Abbildung 4.11: Überblick über die Schichten

Weiterhin muss die aufrufende Anwendung sicherstellen, dass ein Formular mit den GUI-Komponenten `TreeView` und `ImageList` existiert. Die Namen des `TreeView`s und der Bildliste werden dem CIB-Objekt ebenfalls als Parameter übergeben. Aus diesen generiert der Browser dann die einzelnen Knoten des Baums und füllt den `TreeView` mit diesen Knoten. Die `ImageList` wird dabei verwendet, um die einzelnen Knoten neben dem Text zusätzlich noch mit einem Icon zu versehen.

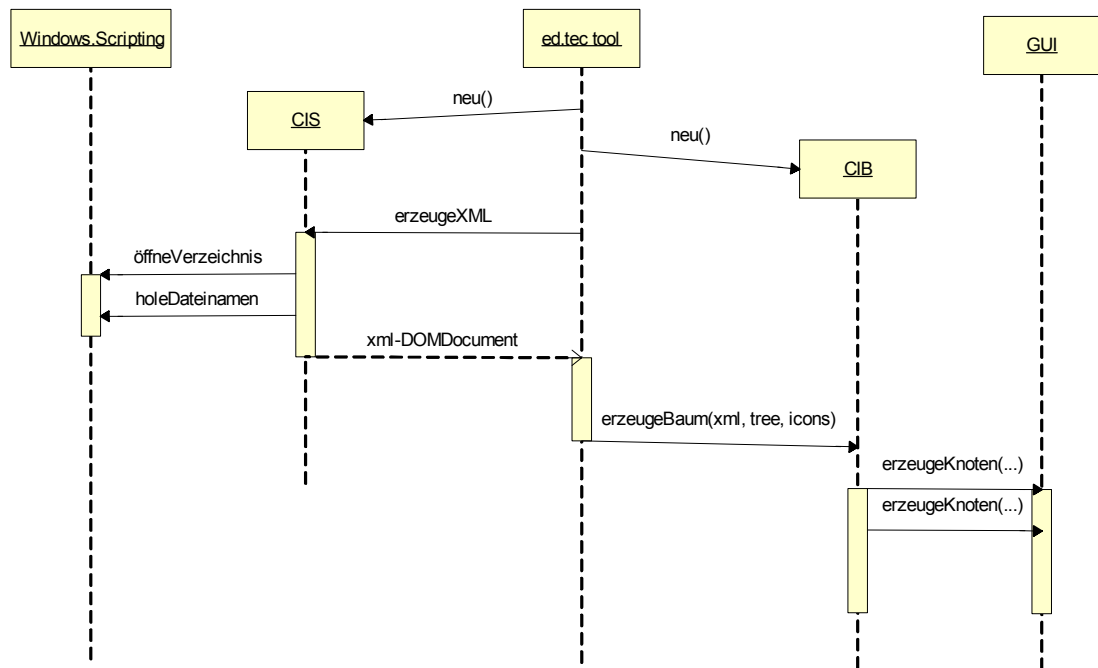


Abbildung 4.12: Zusammenspiel der Schichten

Die Abbildung 4.12 zeigt dabei noch einmal deutlich, wie die einzelnen Schichten zusammenspielen, welche Objekte der Entwickler selbst erzeugen muss und welche Funktionen aufzurufen sind:

1. Erstellen eines Formulars mit oben genannten GUI-Komponenten `TreeView` und `ImageList`, diese mit den Konfigurationswerkzeugen der Entwicklungsumgebung den eigenen Wünschen anpassen (z.B. optischer Stil des Baumes)
2. Je ein neues CIS- und CIB-Objekt anlegen
3. Mittels CIS-Funktionsaufruf die XML-Struktur des gewünschten Kursverzeichnisses generieren lassen
4. Mit den verschiedenen Attributen des CIB-Objektes eventuell gewünschte optische Hervorhebungen konfigurieren
5. Die vom CIS zurück gelieferte XML-Struktur zusammen mit den Namen des noch leeren Baumes und der Bildliste dem CIB als Parameter übergeben. Der CIB liefert keinen Wert zurück, sondern der Baum füllt sich als Seiteneffekt mit den einzelnen Knoten.

Genauere Beschreibungen der einzelnen Funktionsaufrufe und Codebeispiele finden sich in Abschnitt 6.1.3 ff.

5 BETRIEBSKONZEPT

Der internetbasierte Wissenstransfer gliedert sich – wie schon im Abschnitt 3.1 dargestellt – in vielfältige und unterschiedliche Programme und Systeme. Abbildung 5.1 gibt einen Überblick über die in ed.tec eingesetzten Werkzeuge, die verwendeten Kommunikationsprotokolle und über die eingesetzte Präsentationstechnologie. Ersichtlich ist auch der weite Einsatz des Content Information Servers auf Seiten der Autorentools, bei der Verteilung der Lehrmaterialien über das WWW und – in Verbindung mit dem Information Browser – zur Unterstützung der Präsenzveranstaltung im Hörsaal.

- Auf Seiten des Autors wird der CIS dazu benutzt, einen lesenden Zugang zur Schulungsmaterialablage zu erhalten, das betrifft hauptsächlich die in der Ablage enthaltenen Zusatzinformationen, also die Metadaten. Protokollseitig wird der Zugang zu Schulungsmaterialablage über das in der Windows-Welt verbreitete SMB-Protokoll realisiert.
- Der File- und Webserver greift ebenfalls auf die vom CIS erzeugte XML-Struktur zurück, um dynamische Webseiten zu erzeugen, die Zugang zu den in der Schulungsmaterialablage für den Lernenden vorgesehenen Materialien bieten. Der Lernende kann so mit einem Webbrowser per HTTP-Protokoll die Materialien online sichten bzw. herunterladen um sie z.B. auszudrucken.

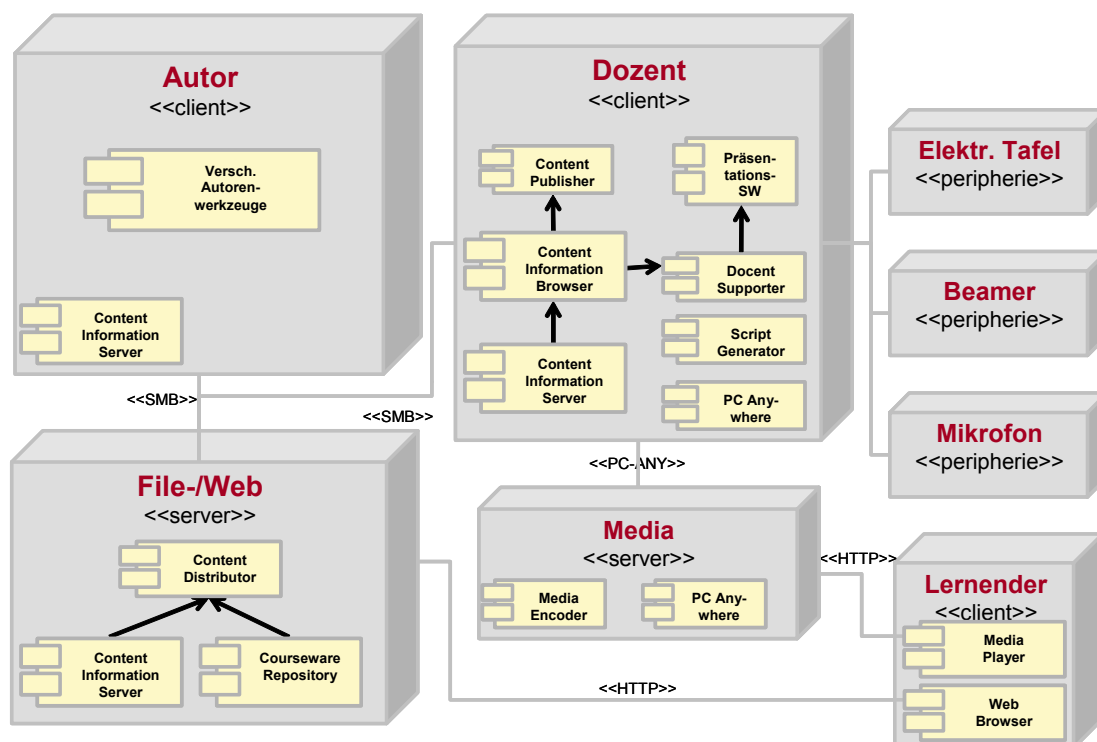


Abbildung 5.1: Überblick über die eingesetzten Systeme und Protokolle

- Der Dozent nutzt eine ganze Reihe von Werkzeugen sowohl zur Vorbereitung als auch bei der Durchführung von Schulungen. Zum Beispiel steht mit dem „Script-Generator“ ein Werkzeug zur Verfügung, das Powerpoint-Präsentationen in Form von Handzetteln ausdruckt, welche vom Dozenten

während der Veranstaltung als Merkzettel verwendet werden. Das kommerzielle Fernwartungssystem „PCanywhere“ dient dazu, die Präsentation vom Notebook des Dozenten in ein Multimedialabor zu übertragen, wo es vom „Media-Encoder“ in ein Video gewandelt, als Live-Stream im Internet zur Verfügung gestellt und archiviert wird (vgl. hierzu auch Abbildung 3.2).

- Weiterhin zum Einsatz kommen Präsentationstechnologien wie elektronische Tafeln (siehe Abschnitt 2.2) in Form eines Graphiktablets, LCD-Projektor und verschiedene Tontechnik.
- Die beiden Dozentenwerkzeuge „Content Publisher“ und „Docent Supporter“ bilden die in den Abschnitten 3.2 und 3.3 entworfene graphische Dozentenumgebung. Sie soll in diesem Kapitel näher vorgestellt werden.

Die graphische Dozentenumgebung besteht aus zwei auf die im Kapitel 4 vorgestellte Architektur aufsetzenden Werkzeugen. Das heißt, auch sie nutzen den CIS zum Zugriff und den CIB zur Visualisierung der Schulungsmaterialablage. Die beiden Programme wurden im Rahmen dieser Diplomarbeit entwickelt und implementiert, und unterstützen den Dozenten bei der Verwaltung der Materialien und den Vortragenden während der Präsenzveranstaltungen auf vielfache Art und Weise:

5.1 ÜBERSICHTLICHE VERWALTUNG DER SCHULUNGSMATERIALIEN

Bereits im Vorfeld einer Schulung hat der Dozent die Möglichkeit, sich einen Überblick über die vorhandenen Schulungsmaterialien zu verschaffen und diese zu verwalten. Die Dozentenumgebung bietet dazu einige Funktionen:

- **Veröffentlichen der Materialien für das WWW-Lernsystem und die Präsentationsumgebung.** Dabei wird die gesamte Kursstruktur mittels der in Abschnitt 4.3 vorgestellten Darstellungsschicht visualisiert, und mit Hilfe der in der Geschäftslogik-Schicht realisierten Funktionen werden die Sichtbarkeits-Attribute in den zugehörigen Dateinamen in der Schulungsmaterialablage entsprechend abgeändert. Ebenso ist es möglich, die Materialien für die jeweilige Veröffentlichungsart zu sperren, d.h. die entsprechenden Attribute werden aus dem zugehörigen Dateinamen entfernt. Diese Änderungen am Dateisystem werden für den Benutzer vollkommen transparent durchgeführt.
- **Konvertieren der Dateiformate.** Üblicherweise liegen Schulungsmaterialien in den Autorenformaten *Microsoft Powerpoint* oder *Word* vor. Diese Formate eignen sich gut zur Präsentation in der Veranstaltung, jedoch nicht für eine Verteilung mittels WWW-Lernumgebung. Die Dozentenumgebung bietet eine Funktion, die diese Formate automatisch in das weit verbreitete und auf vielen Plattformen lesbare *Portable Document Format PDF* konvertiert. Voraussetzung ist allerdings die Installation der Programmpakete *Microsoft Office* und *Adobe Acrobat*.
- **Anzeigen der Dateien.** Sämtliche Schulungsmaterialien können direkt angezeigt werden, somit kann sich der Dozent vor der Veranstaltung noch einmal über deren Inhalt vergewissern. Voraussetzung ist auch hier die Installation der zur Anzeige benötigten Programme.
- **Offline HTML generieren.** Durch die Verwaltung der kompletten Schulungsmaterialablage in XML wird es möglich, aus dieser Datei mittels eines XSL-Stylesheets einen so genannten *offline-Kurs* zu erzeugen. Er liegt in HTML vor und kann beispielsweise genutzt werden, den Kurs per CD-ROM zu verteilen.

Abbildung 5.2 zeigt diesen Teil der Dozentenumgebung.

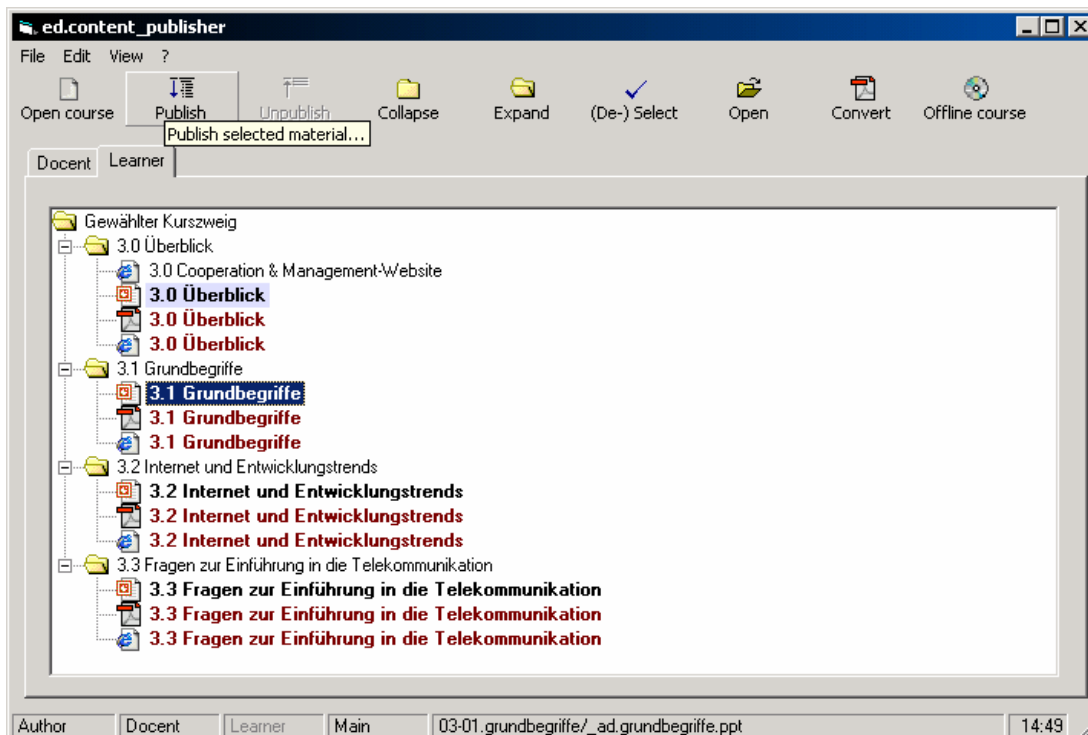


Abbildung 5.2: Verwaltung der Materialien

5.2 TRANSPARENTE NUTZUNG DER PRÄSENTATIONSWERKZEUGE

Der Dozent muss lediglich festlegen, welches Kapitel (in der strukturierten Schulungsmaterialablage durch ein Verzeichnis repräsentiert) er in der Vorlesung behandeln möchte. Die Schulungsmaterialablage darf sich dabei auch auf entfernten Speicherorten wie Netzlaufwerken befinden.

Die benötigten Dateien werden dann in ein – ebenfalls vom Benutzer zu spezifizierendes – schnell zugreifbares lokales Laufwerk kopiert und übersichtlich als Baum dargestellt. Die Umgebung benutzt hierbei ebenso die in Kapitel 3 vorgestellte Architektur, indem sie die benötigten Informationen über die Schulungsmaterialablage aus dem vom *Content Information Server* angebotenen *Information Container* bezieht und die graphische Darstellung der Materialien des *Content Information Browsers* übernimmt. Hier hat der Dozent nochmals die Möglichkeit, einzelne Materialien vor der Schulung auszublenden, indem er in der Baumansicht das betreffende Element anklickt. Ausgeblendete Einheiten werden farbig hinterlegt. Abbildung 5.3 zeigt ein Beispiel eines Kurses und die Felder „Quellpfad“ bzw. „Zielpfad“, in die die Verzeichnisse einzutragen sind.

Der vom *Content Information Server* bereitgestellte *Information Container* wird von der Dozentenumgebung mehrfach verwendet, deshalb hat man die Möglichkeit, diesen lokal abzuspeichern und zu einem späteren Zeitpunkt erneut zu laden. Wenn zum Beispiel die Schulungsmaterialablage während der Vorlesung wegen fehlendem Netzwerkzugang nicht zu Verfügung steht, kann dennoch auf den *Information Container* zugegriffen werden. Der Name der zu speichernden XML-Datei ist auf „content.xml“ voreingestellt, er kann jedoch beliebig geändert werden. Ablageort dieser Datei ist immer das im Feld

„Zielpfad“ angegebene Verzeichnis. Das erneute Laden einer solchen Konfiguration erledigt man im Menüpunkt „Datei“.

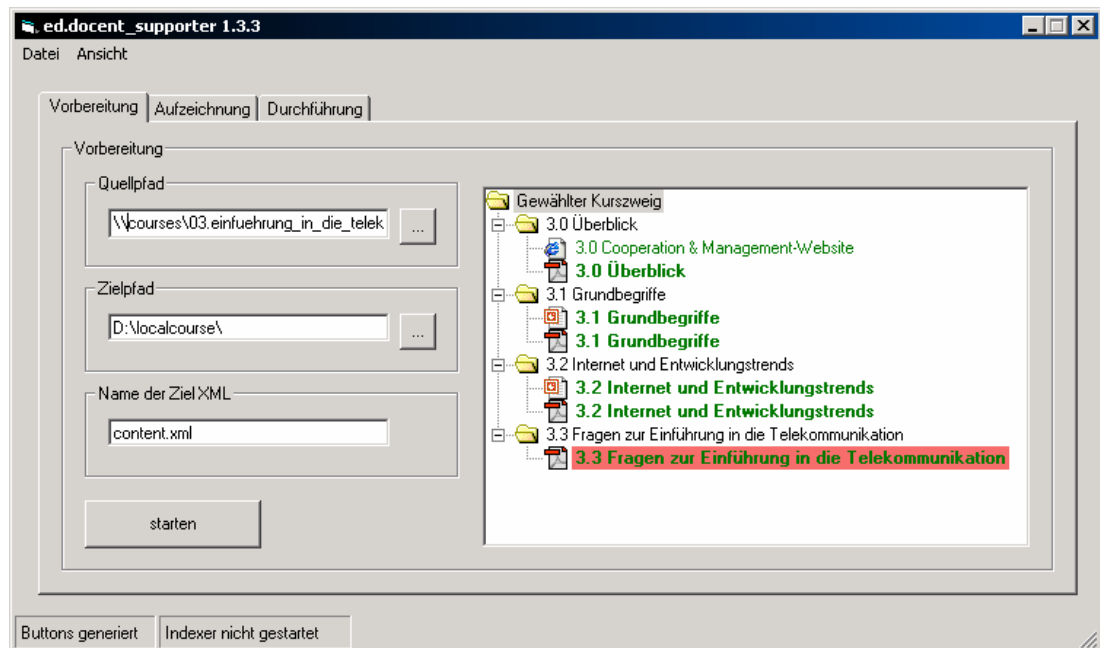


Abbildung 5.3: Dozentenumgebung

Die Umgebung stellt dem Dozenten dann sämtliche Materialien, die die Zielgruppe „Dozent“ besitzen, transparent zur Verfügung. Das heißt, während der Vorlesung besteht Zugriff auf die einzelnen Dateien, ohne dass der Vortragende wissen muss, wo sich diese befinden oder mit welchem Präsentationswerkzeug die einzelnen Materialien darzustellen sind.

Zunächst generiert die Umgebung eine Schaltflächenleiste (im Register „Durchführung“ den Punkt „Präsentation starten“ anwählen) und wechselt in den Präsentationsmodus. Dazu wird an den oberen Rand des Desktops eine Art Taskleiste mit Schaltflächen angeheftet. Jede der Schaltflächen repräsentiert eine Untereinheit eines Kapitels und ermöglicht den direkten Zugriff auf die entsprechende Datei, deren Format durch ein gängiges Icon angezeigt wird. So ist schon beim ersten Blick auf die Leiste klar, was für eine Art von Datei sich hinter dem jeweiligen Knopf verbirgt. Der Dozent muss lediglich die gewünschte Schaltfläche anklicken, um die Datei zu öffnen (Abbildung 5.4).



Abbildung 5.4: Präsentationsleiste

Dazu bestimmt die Umgebung aus der Dateinamenserweiterung die zugehörige Anwendung, startet diese und übergibt der Anwendung dann den Pfad der zu öffnenden Datei. Voraussetzung für ein korrektes Funktionieren ist, dass zu jedem verwendeten Dateityp eine zugehörige Anwendung im Betriebssystem registriert ist. Sollte dies nicht der Fall sein, quittiert die Dozentenumgebung dies mit einer Fehlermeldung. Eine Sonderstellung nehmen Powerpoint-Dateien ein: Sie werden direkt im Präsentationsmodus geöffnet, beim Wechsel zu einer anderen Datei mit einem Zeitstempel versehen und unter einem neuen Namen gespeichert. Der neue Name wird intern vermerkt, so dass beim

erneuten Aufrufen der Datei die zuletzt gespeicherte Version zur Verfügung steht. Diese Technik ermöglicht es, die Präsentationen mit Hilfe einer elektronischen Tafel zu kommentieren und beim Wechseln zwischen den einzelnen Materialien diese Annotationen nicht zu verlieren. Sollte zu einer Einheit Zusatzmaterial existieren, so besteht über ein Ausklappenmenü unter der entsprechenden Schaltfläche ebenfalls direkter Zugriff darauf. Die in Klammer stehende Zahl auf jedem Button (dargestellt in Abbildung 5.5) zeigt die Anzahl dieser Zusatzmaterialien an. Beim Beenden des Programms wird zu jeder vorhandenen Powerpoint-Datei die jeweils zuletzt gespeicherte Version mit dem Namenszusatz „-annotiert“ versehen und eine auf den Dateinamen passende Metadaten-Vorlage erzeugt, so dass die kommentierte Version gleich in die Datenablage eingepflegt werden kann.

In der Taskleiste ist ebenfalls ein Steuermenü integriert, das dem Dozenten während der Präsentation alle notwendigen Funktionen bereitstellt (Abbildung 5.5):

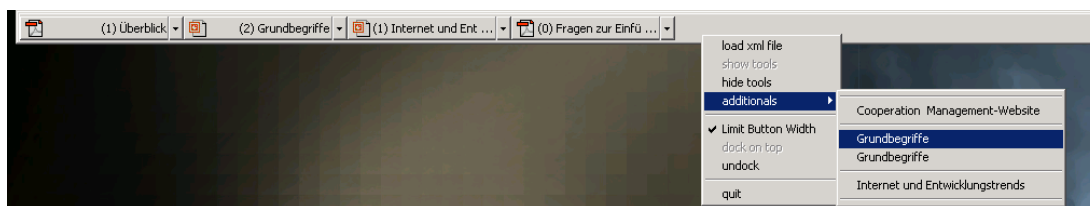


Abbildung 5.5: Kontextmenü der Präsentationsleiste

- **load xml file:** lädt eine vorher gesicherte Konfigurationsdatei.
- **show/hide tools:** dienen zum Ein- und Ausblenden des in Abbildung 5.3 beschriebenen Hauptfensters.
- **additional:** zeigt alle zur Verfügung stehenden Zusatzmaterialien übersichtlich gruppiert an. Per Klick ist das Zusatzmaterial ebenfalls aus dem Menü direkt aufrufbar.
- **limit button width:** schränkt die Breite der Schaltflächen auf einen maximalen Wert ein. Dadurch können mehr Schaltflächen gleichzeitig dargestellt werden. Die Funktion ist von Vorteil, wenn die Schaltflächen aufgrund langer Titel zu breit werden oder wegen einer geringen Bildschirmauflösung wenig Platz zur Verfügung steht.
- **dock on top/undock:** dienen dazu, die Taskleiste vom oberen Bildschirmrand zu lösen bzw. sie wieder dort anzuheften.
- **quit:** beendet das Programm, eventuell noch offene Powerpoint-Präsentationen werden gespeichert.

5.3 PERMANENTER ÜBERBLICK ÜBER DEN ABLAUF DER SCHULUNG

Mittels dieser Leiste hat der Dozent jederzeit Überblick darüber, welche Materialien er bereits gestartet hat: Wurde eine Schaltfläche einmal angeklickt, ändert sich deren optische Darstellung (Abbildung 5.6), so dass sich der Dozent mit einem Blick vergewissern kann, in welcher Phase seiner Vorlesung er sich befindet, welche Materialien schon geöffnet wurden, und welche noch zu präsentieren sind.

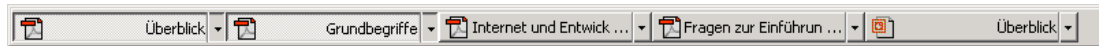


Abbildung 5.6: Präsentationsleiste im Betrieb

Dabei ist der Dozent völlig frei in der Reihenfolge seiner Aktionen, die Anordnung der Schaltflächen in der Leiste spiegelt lediglich die Nummerierung der einzelnen Materialien in der Schulungsmaterialablage wider. Ebenso ist es möglich, bereits präsentiertes Material nochmals aufzurufen, indem einfach die zugehörige Schaltfläche erneut angeklickt wird.

5.4 UNTERSTÜTZUNG DER REMOTE-VERANSTALTUNG

Falls die Präsentation als Video aufgezeichnet wird, besteht die Möglichkeit, von der Dozentenumgebung Indexdateien generieren zu lassen. Das sind Dateien, mit denen sich das Video in einzelne Teile zerlegen lässt. Sollte die Funktion aktiviert sein, wird automatisch bei jedem Klick des Dozenten auf eine Schaltfläche in der Präsentationsleiste ein solcher Index geschrieben. Damit erhält man mehrere Videos, die jeweils ein Kapitel einer Vorlesung repräsentieren. Dies läuft für den Dozenten unsichtbar im Hintergrund ab. Abbildung 5.7 zeigt den in die Präsentationsumgebung integrierten Dialog, mit dem verschiedene Einstellungen zu den Indexdateien vorgenommen werden können. Das Menü wird im Hauptfenster der Dozentenumgebung (Abbildung 5.3) über die Registerkarte „Aufzeichnung“ aufgerufen.

Näheres zu den Indexdateien und wie sich damit Videos zerschneiden lassen, findet sich unter [19].

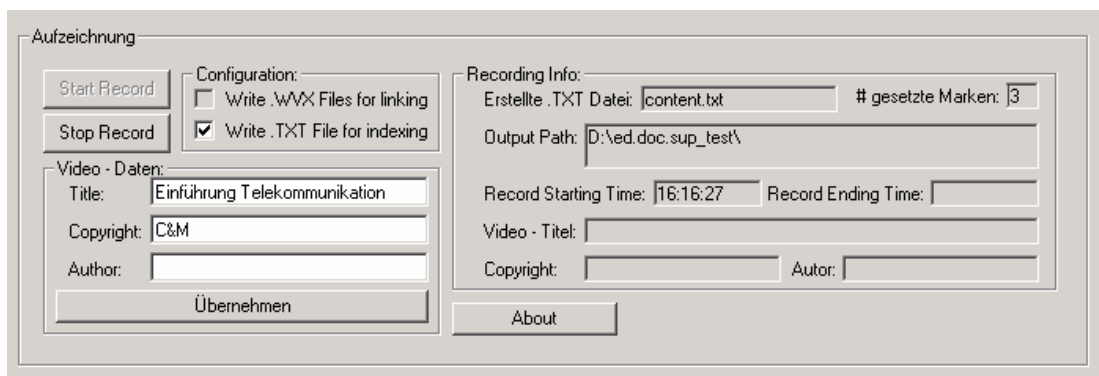


Abbildung 5.7: Indexgenerierung während der Veranstaltung

6 IMPLEMENTIERUNG

Das Kapitel dient der Illustration von Nutzungsprofilen der implementierten Komponenten. Vorgestellt werden ihr interner Aufbau, ihre Arbeitsweise und die verwendeten Datenmodelle.

Des Weiteren sind einige Informationen für Entwickler beigefügt:

- Schnittstellenbeschreibung von CIS und CIB,
- Auflistung der zusätzlich benötigten Komponenten,
- Beispielcode, und
- Illustrationen zur Nutzung der Entwicklungsumgebung am Beispiel von Visual Studio 6.

6.1 DER CONTENT INFORMATION SERVER (CIS)

Wie schon in Kapitel 4.2 beschrieben, wurde der Information Server in drei Funktionsbereiche unterteilt, innerhalb derer die einzelnen Funktionen ja nach Aufgabe wiederum in verschiedenen Klassen zusammengefasst und gruppiert wurden. Über diese Klassen lassen sich die gewünschten Funktionen ansprechen. Jeder der Aufgabenbereiche wurde mit Visual Basic als *ActiveX-dll* realisiert und lässt sich somit sehr einfach in vorhandenen Code einbinden. Intern sind die dlls in semantisch zusammengehörende Klassen unterteilt, so dass sich die einzelnen Funktionen mit den üblichen objektorientierten Prinzipien ansprechen lassen. Abbildung 6.1 zeigt diese Aufteilung in verschiedene Klassen.

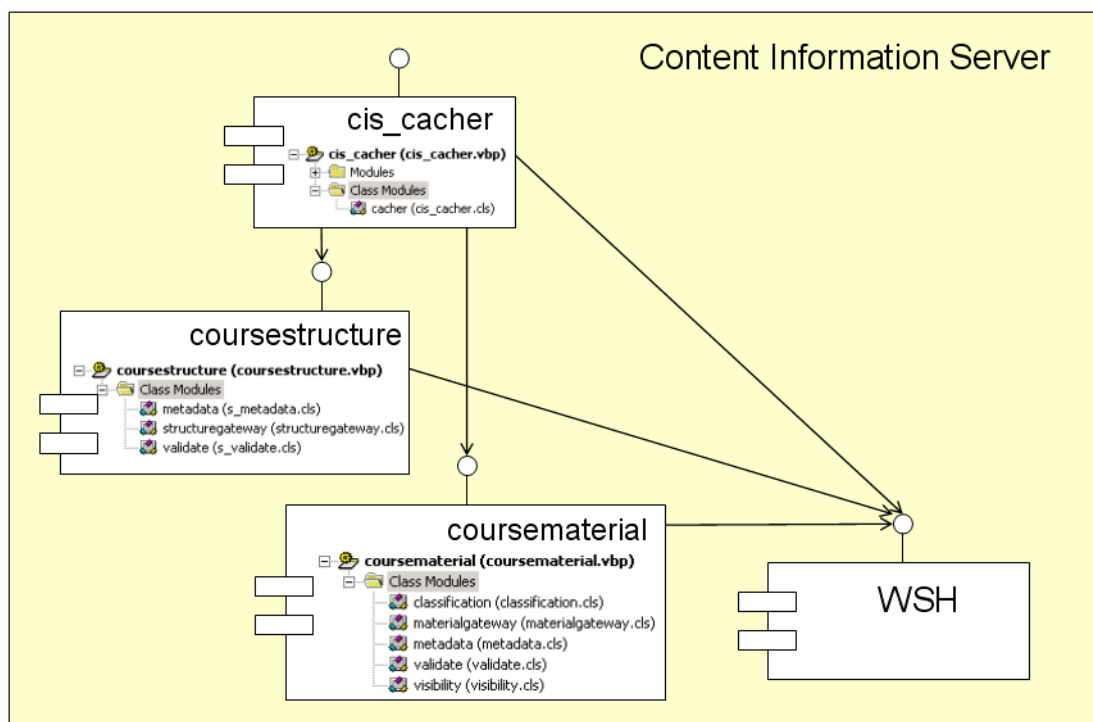


Abbildung 6.1: Pakete und Klassen des CIS

Jedes Paket repräsentiert eine ActiveX-dll, diese heißen `cis_cacher.dll`, `coursestructure.dll` und `coursematerial.dll`. In der Abbildung ist ebenfalls ersichtlich, wie jede einzelne dll intern noch einmal in verschiedene Klassen unterteilt wurde und dass die `cis_cacher.dll` zur Erbringung ihrer Funktionalität auf die beiden anderen zurückgreift. Ein weiterer Bestandteil des CIS besteht in der Verwendung des *Windows Scripting Host* WSH, über dessen Klassen auf das eigentliche Dateisystem zugegriffen wird. Der Windows Scripting Host ist Bestandteil jedes neueren Windows Systems. Vor einer Verwendung des CIS sollte der Entwickler die Aktivität des WSH überprüfen und gegebenenfalls diesen in der Systemsteuerung unter „Software“ → „Windows-Komponenten“ aktivieren. Um die dlls benutzen zu können, ist noch das Registrieren dieser in der Klassenverwaltung von Windows notwendig. Dazu führt man den Befehl

```
Regsvr32 <dllname.dll>
```

in der Kommandozeile von Windows aus.

6.1.1 SCHNITTSTELLENBESCHREIBUNG

Die ActiveX-dlls sind intern in einer oder mehreren Klassen organisiert, die jeweils verschiedene Funktionen beinhalten. Zur Benutzung der Funktionen muss die entsprechende dll zunächst referenziert werden. Dann erstellt man sich ein Objekt der Klasse, die die gewünschte Funktionalität anbietet:

```
Dim meinObject as <dll>.<Klasse>
Set meinObject = New <dll>.<Klasse>
```

Der Aufruf einer Funktion geschieht dann mittels

```
Call meinObject.<Funktion>(<parameter>)
```

(Anstelle einer Referenzierung der benötigten dll gibt es die Möglichkeit, das benutzte Klassenmodul direkt in den Code einzubinden. Der Aufruf der Funktionen erfolgt jedoch auf die gleiche Weise.)

cis_cacher.dll (cis_cacher.cls)

Diese dll erstellt aus einer Verzeichnisstruktur – die den Konventionen entsprechend vorliegt – den Information Container in Form eines MSXML2.DOMDocument.

Voraussetzungen:

- coursematerial.dll muss im System registriert sein
- coursestructure.dll muss im System registriert sein
- Microsoft XML-Parser v3.0 muss installiert sein (ist Bestandteil vom Internet Explorer ab Version 6)
- Microsoft Scripting Runtime aktiv

Funktionen:

Name	Cacher.MakeCache
Aufruf	MakeCache (Folder as String) as MSXML2.DOMDocument

Parameter	Folder Das Basisverzeichnis des Kurses aus dem das XML Dokument erstellt werden soll
Rückgabe	MSXML2.DOMDocument Das XML Dokument, das Format dieses Dokumentes ist im Kapitel 6.1.2 beschrieben
Beschreibung	Diese Funktion erstellt aus einer Verzeichnisstruktur die den Konventionen entsprechend vorliegt ein XML Dokument

coursematerial.dll (classification.cls, materialgateway.cls, metadata.cls, visibility.cls, validate.cls)

Diese dll stellt Klassen bereit, deren Funktionen den Umgang mit dateinamen Operationen erleichtern.

Voraussetzungen:

- Microsoft Scripting Runtime aktiv

Funktionen:

Name	Visibility.isLearner
Aufruf	isLearner (File as String) as boolean
Parameter	File Der Pfad der Datei, die überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion überprüft, ob eine Datei im Lernsystem dargestellt werden soll

Name	Visibility.isAuthor
Aufruf	isAuthor (File as String) as boolean
Parameter	File Der Pfad der Datei, die überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion überprüft, ob eine Datei in der Autorenumgebung dargestellt werden soll

Name	Visibility.isDocent
Aufruf	isDocent (File as String) as boolean
Parameter	File Der Pfad der Datei, die überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion überprüft, ob eine Datei in der Dozentenumgebung dargestellt werden soll

Name	Visibility.isUnknown
Aufruf	isUnknown (File as String) as boolean
Parameter	File Der Pfad der Datei, die überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion überprüft, ob die Darstellung einer Datei noch un-spezifiziert ist

Name	Visibility.setDocent, Visibility.setLearner, Visibility.setAuthor
Aufruf	setDocent, setLearner, setAuthor (File as String) as integer
Parameter	File Der Pfad der Datei, die geändert werden soll
Rückgabe	Integer Fehlercode: 1 = keine Metadaten vorhanden
Beschreibung	Schaltet eine Datei für die Dozenten- (Lern-, Autoren-) Umgebung frei

Name	Visibility.delDocent, Visibility.delLearner, Visibility.delAuthor
Aufruf	delDocent, delLearner, delAuthor (File as String) as integer
Parameter	File Der Pfad der Datei, die geändert werden soll
Rückgabe	Integer Fehlercode: 1 = keine Metadaten vorhanden
Beschreibung	Sperrt eine Datei für die Dozenten- (Lern-, Autoren-) Umgebung

Name	Validate.syntaxcorrect
Aufruf	syntaxcorrect (File as String) as boolean
Parameter	File Der Pfad der Datei, die überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion überprüft, ob eine Datei die Namenskonventionen einhält

Name	Metadata.ismetafile
Aufruf	ismetafile (File as String) as boolean
Parameter	File Der Pfad der Datei, die überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion überprüft, ob es sich bei einer Datei um ein Metadatenfile handelt

Name	Metadata.hasmetafile
Aufruf	hasmetafile (File as String) as boolean
Parameter	File Der Pfad der Datei, die überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion überprüft, ob zu einer Datei Metadaten vorliegen

Name	classification.getclass
Aufruf	ismetafile (File as String) as String
Parameter	File Der Pfad der Datei, die überprüft werden soll
Rückgabe	String „main“ oder „addon“
Beschreibung	Diese Funktion prüft, ob eine Datei als Hauptmaterial oder Zusatzmaterial gekennzeichnet ist

Name	materialgateway.createfile
Aufruf	createfile (File as String)
Parameter	File Der Pfad der Datei, die erstellt werden soll
Rückgabe	-
Beschreibung	Diese Funktion erstellt eine leere Datei

Name	materialgateway.deletefile
Aufruf	deletefile (File as String)
Parameter	File Der Pfad der Datei, die gelöscht werden soll
Rückgabe	-
Beschreibung	Diese Funktion löscht eine Datei

Name	materialgateway.setfiletype
Aufruf	setfiletype (File as String, Type as String)
Parameter	File Der Pfad der Datei, die geändert werden soll Type Gewünschter Typ der Datei
Rückgabe	-
Beschreibung	Diese Funktion ändert den Typ (Dateiextension) einer Datei

coursestructure.dll (s_metadata.cls, structuregateway.cls, s_validate.cls)

Diese dll stellt Klassen bereit, deren Funktionen den Umgang mit verzeichnisnahen Operationen erleichtern.

Voraussetzungen:

- Microsoft Scripting Runtime aktiv

Funktionen:

Name	Validate.foldersyntaxcorrect
Aufruf	foldersyntaxcorrect (Folder as String) as boolean
Parameter	Folder Der Pfad des Verzeichnisses, das überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion überprüft, ob ein Verzeichnis die Namenskonventionen einhält

Name	Metadata.folderhasmeta
Aufruf	folderhasmeta (File as String) as boolean
Parameter	File Der Pfad des Verzeichnisses, das überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion überprüft, ob zu einem Verzeichnis Metadaten vorliegen

Name	structuregateway.foldercreate
------	-------------------------------

Aufruf	foldercreate (Folder as String)
Parameter	Folder Der Pfad des Verzeichnisses, das erstellt werden soll
Rückgabe	-
Beschreibung	Diese Funktion erstellt ein leeres Verzeichnis

Name	structuregateway.folderdelete
Aufruf	Folderdelete (Folder as String)
Parameter	Folder Der Pfad des Verzeichnisses, das gelöscht werden soll
Rückgabe	-
Beschreibung	Diese Funktion löscht ein Verzeichnis

Name	structuregateway.existsfolder
Aufruf	existsfolder (Folder as String) as Boolean
Parameter	Folder Der Pfad des Verzeichnisses, das überprüft werden soll
Rückgabe	Boolean
Beschreibung	Diese Funktion testet, ob ein Verzeichnis existiert

Name	structuregateway.getorder
Aufruf	Getorder (Folder as String) as String
Parameter	Folder Der Pfad des Verzeichnisses, das geprüft werden soll
Rückgabe	Ein String, der die Kapitelnummer des Verzeichnisses darstellt
Beschreibung	Liefert die Nummer des Kapitels eines Verzeichnisses

6.1.2 ARBEITSWEISE UND DATENMODELL

Mit Hilfe des Windows Scripting Host (WSH) wird das Dateisystem rekursiv durchlaufen. Der Einstiegspunkt ist dabei frei wählbar. Sämtliche Verzeichnisse, die den Namenskonventionen entsprechen, werden in den XML-Baum eingehängt. Dabei werden die Position, der Pfad des Verzeichnisses, die Rekursionstiefe und die Nummer des Verzeichnisses (sie repräsentiert ein Kapitel) festgehalten sowie vorhandene Metadaten über das Verzeichnis in den XML-Baum eingetragen.

Enthält ein Verzeichnis einzelne Dateien, wird pro Datei ein neuer XML-Knoten angelegt und Informationen über die Datei festgehalten. Dazu gehören neben dem Pfad der Datei auch die Klassifizierung (legt fest, ob eine Datei Haupt- oder Zusatzmaterial enthält) und die Sichtbarkeitsinformationen. Sie beschreiben, ob eine Datei für den Dozenten während der Vorlesung, für das webbasierte Lernsystem oder nur zu Autorenzwecken bestimmt ist. Falls für eine Datei Metainformationen (sie liegen bereits in XML-Form vor) existieren, werden auch sie dem XML-Baum zugefügt.

Die Zwischenspeicherung der Verzeichnisstruktur in einer XML Datei bietet neben den in Kapitel 4.1 beschriebenen architektonischen Vorteilen einer Abstraktion vom Verzeichnissystem zusätzlich auch technische Vorteile im Zugriff auf die darin gespeicherten Objekte. So können alle Möglichkeiten des verwendeten XML Parsers benutzt werden, z.B. Filterung anhand von Attributwerten oder die Erstellung einer linearen Liste. Eine Beschreibung des Parsers findet sich unter [13].

Das MSXML2.DOMDocument

Das DOMDocument ist Rückgabeparameter der Funktion makeCache und wird durch die ActiveX-dll „cis_cacher.dll“ aus der Dateistruktur und den Metadaten erzeugt. Der Aufbau des XML Dokumentes richtet sich direkt nach der Verzeichnisstruktur.

Das zugehörige XML-Schema lautet wie folgt:

```
<Schema>
  <element name="TREE">
    <complexType name="treeType">
      <all>
        <element name="FOLDER" type="folderType" minOccurs="0" maxOccurs="unbounded">
          <complexType name="folderType">
            <attribute name="LEVEL" type="int" use="required"/>
            <attribute name="ID" type="int" use="required"/>
            <all>
              <element name="ITEM" type="itemType" minOccurs="0" maxOccurs="unbounded">
                <complexType name="itemType">
                  <attribute name="FORMAT" type="string" use="required"/>
                  <attribute name="CLASSIFICATION" type="string" use="required"/>
                  <attribute name="VISIBILITYAUTHOR" type="string" use="required"/>
                  <attribute name="VISIBILITYLEARNER" type="string" use="required"/>
                  <attribute name="VISIBILITYDOCENT" type="string" use="required"/>
                  <all>
                    <element name="NUMBER" type="string" minOccurs="1" maxOccurs="1"/>
                    <element name="LINK" type="string" minOccurs="1" maxOccurs="1"/>
                    <element name="VISIBILITY" type="string" minOccurs="0" maxOccurs="3"/>
                    <element name="RECORD" type="recordType" minOccurs="1" maxOccurs="1"/>
                  </all>
                </complexType>
              </element>
              <element name="RECORD" type="recordType" minOccurs="1" maxOccurs="1">
                <complexType name="recordType">
                  <attribute name="xmlns" type="string" use="value"
value="http://www.cooperation-management.de"/>
                <all>
                  <!-- hier Elementdefinitionen der Metadaten -->
                </all>
              </element>
            </all>
          </complexType>
        </element>
        <element name="FOLDER" type="folderType" minOccurs="0" maxOc-
curs="unbounded"/>
      </all>
    </complexType>
  </element>
</Schema>
```

Eine typische XML-Datei:

```
- <TREE>
- <FOLDER LEVEL="0" ID="0">
  <NUMBER>6.5</NUMBER>
  + <RECORD XMLNS="http://www.cooperation-management.de">
  - <ITEM FORMAT="ppt" CLASSIFICATION="main" VISIBILITYAUTHOR="yes" VISIBILITYLEARNER="no" VISIBILITYDOCENT="no">
    <NUMBER>6.5</NUMBER>
    <LINK>06-05.uebertragungsverfahren_und_umformung/_a.uebertragungsverfahren_und_umformung.ppt</LINK>
    <VISIBILITY>author</VISIBILITY>
    + <RECORD XMLNS="http://www.cooperation-management.de">
    </ITEM>
  </FOLDER>
- <FOLDER LEVEL="0" ID="1">
  <NUMBER>6.4</NUMBER>
  + <RECORD XMLNS="http://www.cooperation-management.de">
  - <ITEM FORMAT="ppt" CLASSIFICATION="main" VISIBILITYAUTHOR="yes" VISIBILITYLEARNER="no" VISIBILITYDOCENT="no">
    <NUMBER>6.4</NUMBER>
    <LINK>06-04.mehrfachnutzung_von_medien/_a.mehrfachnutzung_von_medien.ppt</LINK>
    <VISIBILITY>author</VISIBILITY>
    + <RECORD XMLNS="http://www.cooperation-management.de">
    </ITEM>
  </FOLDER>
+ <FOLDER LEVEL="0" ID="2">
+ <FOLDER LEVEL="0" ID="3">
+ <FOLDER LEVEL="0" ID="4">
+ <FOLDER LEVEL="0" ID="5">
</TREE>
```

Die unter dem Tag <RECORD> zu jedem FOLDER und jedem ITEM eingetragenen Metadateien sehen dabei typischerweise so aus:

```
- <RECORD XMLNS="http://www.cooperation-management.de">
  <!-- ed.tec metadata Template -->
  - <GENERAL>
    <TITLE>Übertragungsverfahren und Umformung</TITLE>
    - <AUTHORS>
      <USERNAME>Prof. Dr. S. Abeck</USERNAME>
    </AUTHORS>
    <LANGUAGE>de</LANGUAGE>
  </GENERAL>
  - <CLASSIFICATION>
    <KEYWORDS>Schlüsselbegriffe, bei Bedarf duplizieren</KEYWORDS>
  </CLASSIFICATION>
  - <PEDAGOGICAL>
    <RESOURCE_TYPE>slide</RESOURCE_TYPE>
    <RESOURCE_TYPE>narrative text</RESOURCE_TYPE>
    <RESOURCE_TYPE>figure</RESOURCE_TYPE>
    <RESOURCE_TYPE>graph</RESOURCE_TYPE>
    <RESOURCE_TYPE>table</RESOURCE_TYPE>
    <DURATION>15 min</DURATION>
    <USAGE_REMARKS />
  </PEDAGOGICAL>
  - <TECHNICALL>
    <FORMAT>Powerpoint 97</FORMAT>
  </TECHNICALL>
</RECORD>
```

Erklärung der einzelnen Elemente und Attribute:

- FOLDER bildet ein Verzeichnis ab, @ID ist eine eindeutige ID, normalerweise numerisch, @LEVEL gibt die Tiefe im Baum an, beginnend bei 0.
 - Zu jedem FOLDER werden die NUMBER (Kapitelnummer), ein LINK (relativer Pfad des Verzeichnisses) und die Metadaten (RECORD) gespeichert.
- ITEM bildet eine Materialdatei ab, @FORMAT ist die Dateiendung, @CLASSIFICATION ist entweder „main“ oder „addon“. @VISIBILITYAUTHOR speichert, ob die Datei in der Autorenumgebung an-

gezeigt werden soll oder nicht („yes“ bzw. „no“). (@VISIBILITYLEARNER und @VISIBILITYDOCENT verhalten sich analog dazu.)

- VISIBILITY ist die Sichtbarkeit, sie kann bis zu dreimal (oder auch gar nicht) vorkommen und die Werte „docent“, „learner“ oder „author“ annehmen. (Die doppelte Speicherung der Sichtbarkeitsinformationen liefert gewisse Freiheiten bei der späteren Verwendung des XML-Dokuments.)
- Es werden zusätzlich noch NUMBER, LINK (Relativer Pfad der Datei) und Metadaten (RECORD) gespeichert.

6.1.3 VERWENDUNG DES CIS

Voraussetzung für ein korrektes Funktionieren der cis_cacher.dll, der coursewarematerial.dll und der coursewarestructure.dll sind:

- Installation von Microsoft XML v3.0 (ist Bestandteil des Internet Explorer 6, kann aber auch separat unter [13] kostenlos bezogen werden)
- Ein aktivierter Windows Scripting Host (wird für Zugriffe auf das Dateisystem und reguläre Ausdrücke zur Überprüfung der Korrektheit der Datei- und Verzeichnisnamen benötigt)

Die cis_cacher.dll benutzt dabei die beiden anderen dlls. Für ein korrektes Funktionieren sind also auch diese beiden notwendig. Bei Aufruf liefert der CIS ein `MSXML2.DOMDocument` zurück, das dann von der aufrufenden Applikation entweder direkt weiterverwendet oder zur späteren Verwendung als XML-Datei gespeichert werden kann.

Um die Funktionalität des CIS zu nutzen, bestehen 2 Möglichkeiten:

1. Das Importieren der Klassenmodule in das benutzende Visual Studio-Projekt.
2. Das Anlegen von Referenzen auf die drei ActiveX-dlls. Diese Vorgehensweise soll am Beispiel von Visual Basic 6 verdeutlicht werden.

Zunächst müssen dem Projekt die benötigten Referenzen zugefügt werden. Dies geschieht im Menü „Project/References“ (siehe Abbildung 6.2). Dabei müssen sich die drei CIS-dlls nicht notwendigerweise im lokalen Systemverzeichnis befinden. Man kann beliebige Ablage-Orte (Button „Browse...“) angeben. Hierbei sind auch entfernte Speicherorte, wie zum Beispiel Netzlaufwerke, möglich.

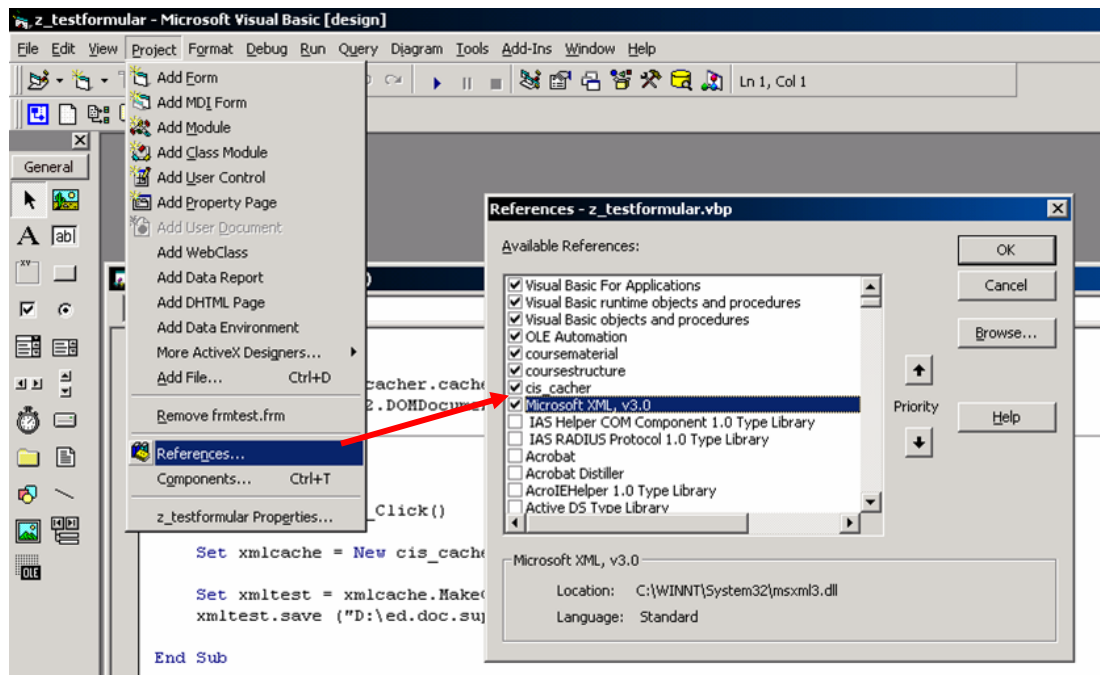


Abbildung 6.2: Setzen der benötigten Referenzen in Visual Studio

Die Benutzung der Funktionalität soll das folgende Codebeispiel verdeutlichen, welches ein XML-DOMDocument des Verzeichnisses "d:\vorlesung\material\" erzeugen und dieses dann unter dem Namen „cache.xml“ in das Verzeichnis "d:\vorlesungen\xml\" abspeichern würde.

```
Dim meinxml as MSXML2.DOMDocument
Dim xmlcache as cis_cacher.cacher
Set xmlcache = New cis_cacher.cacher
Set meinxml = xmlcache.MakeCache("d:\vorlesung\material")
meinxml.save ("d:\vorlesung\xml\cache.xml")
```

Der in der Variable meinxml abgelegte DOM-Baum kann dann mit den umfangreichen Möglichkeiten des Microsoft XML-Parsers weiterverarbeitet werden. Das Abspeichern in einer Datei ist hierzu nicht nötig, sondern nur eine Option, falls das Dokument später wieder verwendet werden soll.

6.2 DER CONTENT INFORMATION BROWSER (CIB)

Ziel bei der Implementierung war es, die in Kapitel 4.3 vorgestellten Filterfunktionen zu realisieren und es dem Entwickler zu ermöglichen, den Content Information Browser möglichst einfach in seine benutzende Anwendung zu integrieren und auf seine Funktionalität zugreifen zu können.

Realisiert wurde der CIB als Visual Basic Klassenmodul, dadurch wird ein Ansprechen der Attribute und Funktionen in objektorientierter Art und Weise gewährleistet.

Die Funktionalität im Überblick:

- Darstellen der Schulungsmaterialablage in flacher Hierarchie. Das heißt, die Verzeichnisstruktur wird nur bis zur ersten Schachtelungstiefe übernommen, die Materialien der weiteren Unterverzeichnisse werden also serialisiert gezeigt,

- Darstellung der Ablage in vollständiger Hierarchie. Die ursprüngliche Verzeichnisstruktur wird dabei komplett dargestellt,
- Auslesen der zu einer Datei zugehörigen Metadaten, wahlweise als Text oder in der originalen XML-Struktur,
- Auslesen der Prioritäts- und Sichtbarkeitskodierung,
- Ein- und Ausblenden der einzelnen Materialien je nach gewünschter Sichtbarkeit und Klassifikation,
- Farbige Hervorhebung der Materialien je nach Sichtbarkeit und Einstellen der Farbe,
- Hervorhebung von als Hauptmaterial gekennzeichneten Dateien durch Fettschrift,
- Neuzeichnen des Baumes, wobei der vorherige Zustand jedes aufklappbaren Knotens wiederhergestellt wird.

6.2.1 SCHNITTSTELLENBESCHREIBUNG

Funktionen:

Name	generateTreeLinear
Aufruf	generateTreeLinear(folder as MSXML2.DOMDocument, tree as TreeView, iconliste as ImageList)
Parameter	folder: Das xml-Dokument, das Grundlage des Baumes ist tree: Der TreeView, der gefüllt werden soll iconliste: Die zu verwendende Bildliste
Rückgabe	-
Beschreibung	Stellt die xml-Struktur als flache Hierarchie im Baum dar, d.h. jedes Verzeichnis wird direkt unterhalb der Baumwurzel dargestellt

Name	generateTreeRecursive
Aufruf	generateTreeRecursive(folder as MSXML2.DOMDocument, tree as TreeView, iconliste as ImageList)
Parameter	folder: Das xml-Dokument, das Grundlage des Baumes ist tree: Der TreeView, der gefüllt werden soll iconliste: Die zu verwendende Bildliste
Rückgabe	-
Beschreibung	Stellt die xml-Struktur als verschachtelte Hierarchie im Baum dar (der originalen Verzeichnisstruktur entsprechend)

Name	getMetaData
Aufruf	getMetaData(node as MSComctlLib.node, metadatastyle as String) as String

Parameter	Node Der Knoten, dessen Metadaten abgefragt werden sollen Metadastyle: bestimmt, wie die Daten dargestellt werden: „xml“ als XML-Datei mit Tags, „text“ als reiner Text ohne Tags
Rückgabe	Die Metadaten als Zeichenkette, entsprechend dem Aufruf formatiert
Beschreibung	Diese Funktion liefert die Metadaten eines Knotens

Name	getVisibilities
Aufruf	getVisibilities(node as MSComctlLib.node) as String
Parameter	Node Der Knoten, dessen Sichtbarkeit abgefragt werden sollen
Rückgabe	Die Sichtbarkeit als Zeichenkette (Folge aus „A“, „D“, „L“)
Beschreibung	Diese Funktion liefert die Sichtbarkeit eines Knotens

Name	getClassification
Aufruf	getClassification(node as MSComctlLib.node) as String
Parameter	Node Der Knoten, dessen Klassifikation abgefragt werden sollen
Rückgabe	Die Klassifikation als Zeichenkette
Beschreibung	Diese Funktion liefert die Klassifikation eines Knotens („main“ oder „addon“)

Name	refreshTree
Aufruf	refreshTree(xmlnew As MSXML2.DOMDocument, tree as MSComctlLib.TreeView, iconliste as MSComctlLib.ImageList)
Parameter	xmlnew: Das xml-Dokument, das Grundlage des Baumes ist tree: Der TreeView, der gefüllt werden soll iconliste: Die zu verwendende Bildliste
Rückgabe	-
Beschreibung	Funktion erstellt den Baum neu. Unterschied zu generateTreeLinear() bzw. generateTreeRecursive() ist der, dass hier die bereits erstellte Hierarchiestruktur übernommen wird und alle Knoten gleich so aufgeklappt werden, wie sie es vorher waren.

Name	setHighlightColor
Aufruf	setHighlightColor(visibility as String, r as Integer, g as Integer, b as Integer)
Parameter	visibililty Die Sichtbarkeit, deren Farbe eingestellt werden soll. Zulässige Werte sind „A“, „D“, „L“ r Rotwert der neuen Farbe g Grünwert der neuen Farbe b Blauwert der neuen Farbe

Rückgabe	-
Beschreibung	Stellt die Farben für die Sichtbarkeiten ein. Grundlage ist das RGB-Farbmodell

Attribute:

Name	Showdocent as Boolean
Beschreibung	Beim Generieren eines Baumes mit einer der beiden obigen Funktionen wird eine Datei dann dargestellt, wenn sie Dozentenlesbarkeit besitzt

Name	Showauthor as Boolean
Beschreibung	Beim Generieren eines Baumes mit einer der beiden obigen Funktionen wird eine Datei dann dargestellt, wenn sie Autorenlesbarkeit besitzt

Name	Showlearner as Boolean
Beschreibung	Beim Generieren eines Baumes mit einer der beiden obigen Funktionen wird eine Datei dann dargestellt, wenn sie Lernsystemlesbarkeit besitzt

Name	Showunspecified as Boolean
Beschreibung	Beim Generieren eines Baumes mit einer der beiden obigen Funktionen wird eine Datei dann dargestellt, wenn sie keine Lesbarkeit besitzt

Dabei hat das Anzeigen einer Datei immer Priorität vor dem Nicht-Anzeigen. Ist zum Beispiel `showdocent = true` und `showlearner = false`, dann wird eine Datei, die beide Sichtbarkeiten besitzt, dargestellt.

Name	Showmainonly as Boolean
Beschreibung	Wenn true werden nur Hauptmaterialien angezeigt

Name	Highlightdocent as Boolean
Beschreibung	Wenn true werden dozentensichtbare Materialien in der eingestellten Farbe dargestellt

Name	Highlightauthor as Boolean
Beschreibung	Wenn true werden autorensichtbare Materialien in der eingestellten Farbe dargestellt

Name	Highlightlearner as Boolean
Beschreibung	Wenn true werden lernumgebungssichtbare Materialien in der eingestellten Farbe dargestellt

Highlightdocent, Highlightauthor und Highlightlearner können dabei nicht kombiniert eingesetzt werden, der Entwickler hat also sicherzustellen, dass jeweils nur eines der drei Attribute „true“ ist.

Name	Highlightmain as Boolean
------	--------------------------

Beschreibung	Wenn true werden main-Materialien in Fettdruck dargestellt (unabhängig von der Sichtbarkeit)
--------------	--

Die Attribute zeigen erst dann Wirkung, wenn ein Baum generiert wird. Schaltet man ein Attribut um, so muss der Baum mittels `generateTreeLinear()`, `generateTreeRecursive()` oder `refresh()` neu erzeugt werden, damit die Wirkung sichtbar wird.

6.2.2 ARBEITSWEISE UND VERWENDUNG

In der folgenden Abbildung 6.3 ist das typische Zusammenwirken von benutzender Anwendung (hier modelliert mittels 2 Klassen „tool“ und „GUI“), Content Information Browser und Content Information Server als UML Sequenzdiagramm dargestellt.

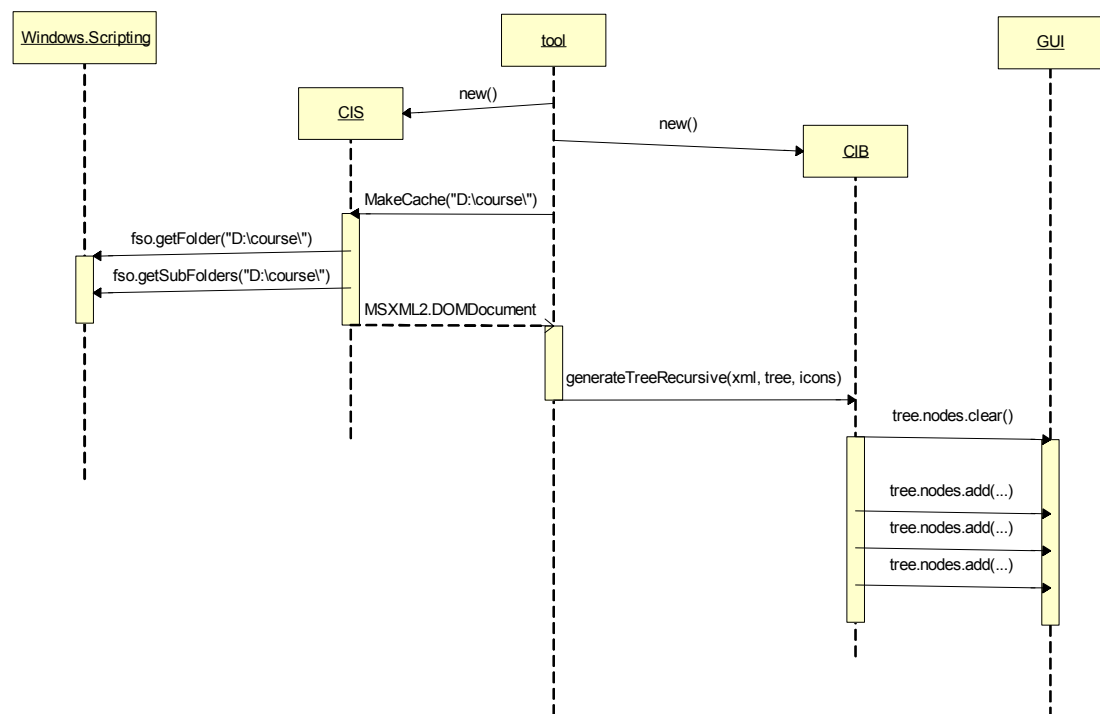


Abbildung 6.3: UML Sequenzdiagramm, Zusammenspiel von CIB, CIS, WSH und benutzender Anwendung

Es bleibt dabei dem Entwickler überlassen, wie er die vielfältigen Eigenschaften des `TreeView` einstellt und auf welche Weise auf die verschiedenen Ereignisse, die dieses Control generieren kann, reagiert wird. Der **CIB** erzeugt lediglich die Baumstruktur des `TreeViews`.

Die Art und Weise, wie der **CIB** benutzt werden kann, soll durch das folgende Codebeispiel verdeutlicht werden: Die aufrufende Anwendung erstellt zunächst je ein **CIS** und ein **CIB**-Objekt. Das **CIS**-Objekt erzeugt das XML-DOMDocument des angegebenen Kurses (siehe Kapitel 6.1) und dem **CIB**-Objekt wird angezeigt, dass nur die für den Dozenten relevanten Hauptmaterialien dargestellt werden sollen.

Anschließend wird das eigentliche Generieren des Baumes aufgerufen. Neben der XML-Struktur erhält das **CIB**-Objekt mit diesem Aufruf auch die wichtige Information darüber, welcher `TreeView` (`frmCIB.Tree`) zur Darstellung benutzt werden soll und welche Icons (`frmCIB.icons`) zur Verfügung stehen.

```

Option Explicit
Dim mycib As clsCIB
Dim xmlcacher As cis_cacher.cacher
Dim meinxml As MSXML2.DOMDocument

Public Sub main()
    FrmCIB.Show
    Set mycib = New clsCIB
    Set xmlcacher = New cis_cacher.cacher
    Set meinxml = xmlcacher.MakeCache("D:\course")
    mycib.showdocent = true
    mycib.showmainonly = true
    Call mycib.generateTreeRecursive(meinxml, _
        frmCIB.tree, frmCIB.icons)
End Sub

```

Der CIB geht dabei davon aus, dass der `TreeView` und die `ImageList` schon existieren. Dieses muss also vom Entwickler, der die CIB-Klasse nutzen möchte, sichergestellt werden.

6.3 DIE DOZENTENUMGEBUNG

Wie in Kapitel 5 beschrieben, wurde mit der graphischen Dozentenumgebung ein Werkzeug entwickelt, welches die Aufgabe hat, den Vortragenden während der Lehrveranstaltung in möglichst vielfältiger Art und Weise zu unterstützen. Dazu gehören:

- Transparente Nutzung der Präsentationswerkzeuge,
- Permanenter Überblick über den Ablauf der Schulung,
- Einfacher Aufruf der Schulungsmaterialien,
- Flexible Präsentationsabfolge,
- Wiederholter Aufruf der Schulungsmaterialien, und
- Unterstützung der Remote-Veranstaltung.

Dabei sollte möglichst auf vorhandene Werkzeuge, Datenstrukturen und Techniken aufgesetzt werden.

Bei der Realisierung wurden die im Kapitel 4 vorgestellten Content Information Server und Browser eingesetzt, um die Dozentenumgebung so unabhängig wie möglich von Verzeichnisstrukturen und Namensvereinbarungen zu halten, gleichzeitig aber mit dem XML-basierten *Information Container* eine mächtige und flexibel nutzbare Datenstruktur nutzen zu können. Zur Visualisierung der Materialien wurde der Content Information Browser in die Dozentenumgebung eingebunden.

Implementiert wurde das Programm in Visual Basic 6. Es besteht aus zwei Formularen, drei statischen Modulen und drei Klassenmodulen sowie einigen Referenzen. Diese zeigen auf die in Kapitel 6.1 vorgestellten ActiveX-dlls des Content Information Servers (CIS), die Windows Scripting Laufzeitumgebung (WSH) und Microsoft XML 3.0 um das vom CIS generierte XML-DOMDocument handhaben zu können (Abbildung 6.4)

Die zusätzlich gesetzten Referenzen sind standardmäßig von Visual Basic eingerichtet.

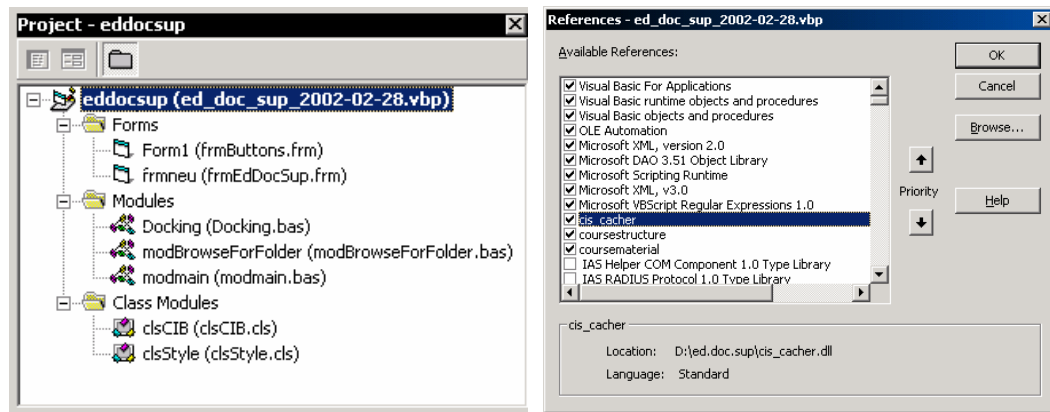


Abbildung 6.4: Komponenten und Referenzen der Dozentenumgebung

Die Komponenten und ihre Aufgaben werden im Folgenden beschrieben.

6.3.1 FRMBUTTONS.FRM

Dieses Formular enthält:

- die Schaltflächenleiste und die Routinen, die diese zur Laufzeit erzeugen, verändern, entfernen sowie auf Benutzerereignisse reagieren,
- ein Kontextmenü und die Funktionen zur Reaktion auf Menüklicks,
- eine Funktion zum nachträglichen Laden einer XML-Datei,
- Aufrufe von Funktionen der Module Docking.bas und clsStyle.cls,
- Aufrufe der Windows-API zum Starten einer Datei nach Benutzerklick auf eine Schaltfläche, und
- einige Array-Listen zur internen Verwaltung der Daten.

6.3.2 FRMEDDOCSUP.FRM

Funktionen und Komponenten:

- sämtliche GUI-Elemente des Hauptfensters, diese sind aufgeteilt in drei so genannte TabStrips,
- die ImageList und der TreeView, aus denen der Content Information Browser die graphische Darstellung des gewählten Verzeichnisses der Schulungsmaterialablage generiert,
- Funktionen, die auf Benutzereingaben an den jeweiligen GUI-Komponenten reagieren,
- Aufruf von Funktionen der Module modBrowseForFolder.bas (zum Festlegen der Verzeichnisse) und modMain.bas, und
- alle Funktionen zur Erzeugung der (in Kapitel 5.4 beschriebenen) Schnittmarkendatei.

6.3.3 DOCKING.BAS

Dieses Modul enthält API-Aufrufe, die das Formular frmButtons.frm als Taskleiste an den oberen Desktoprand anheften.

6.3.4 MODBROWSEFORFOLDER.BAS

Erzeugt einen Dialog, mit dem der Benutzer bequem ein Verzeichnis auswählen kann. Das Modul wird verwendet, um Quell und Zielverzeichnis der Materialien festzulegen.

6.3.5 MODMAIN.BAS

HauptEinstiegspunkt der Dozentenumgebung, startet die beiden graphischen Formulare und enthält Funktionen, die

- den CIS aufrufen, um das XML-DOMDocument generieren zu lassen und dieses zwecks späterer Wiederverwendung speichern,
- die benötigten Materialien vom Quell- in das lokale Zielverzeichnis kopieren,
- das DOMDocument durchlaufen und die benötigten Informationen in den internen Datenstrukturen der Dozentenumgebung speichern,
- die Routine aufrufen, die im Schaltflächenformular die Buttonleiste aufbaut, und
- Funktionen des Content Information Browsers aufrufen, um den Baum im Hauptformular erstellen zu lassen.

6.3.6 CLSCIB.CLS

Bei dieser Klasse handelt es sich um den im Kapitel 6.2 beschriebenen Content Information Browser CIB. Er stellt die Materialien graphisch dar.

6.3.7 CLSSTYLE.CLS

Ändert das Aussehen des Schaltflächenformulars. Die bei Formularen üblichen Titelleisten werden entfernt, so dass das Formular optisch einer Taskleiste entspricht. Hierbei greift die Klasse auf die Windows-API zurück

7 AUSBLICK

Die im Rahmen dieser Arbeit entstandenen Werkzeuge und Softwarekomponenten werden bereits institutsweit eingesetzt: Die Dozentenumgebung wird bei den Lehrveranstaltungen wie Vorlesungen, Übungen und Seminaren des Instituts genutzt, der Content Information Browser findet darüber hinaus Verwendung in einem Autorenwerkzeug und der webbasierten Lernumgebung. Auch aufgrund von Einsatzerfahrungen entstehen immer wieder neue Anforderungen, Ideen und Nutzungsmöglichkeiten bzw. Verbesserungswünsche, die im Rahmen dieser Arbeit nicht mehr realisiert werden konnten und sollten.

7.1 FUNKTIONALITÄT

Die graphische Dozentenumgebung (s. Abschnitt 5.2 ff.) wurde bisher immer in Verbindung mit der in Abschnitt 2.2.1 vorgestellten Smartboard-Technologie eingesetzt. Durch Umstellung der elektronischen Tafel auf die Graphiktablett-Technik kommt nun als Präsentationsformat Powerpoint zum Einsatz, dessen Verwendung aus der Dozentenumgebung heraus andere implementierungstechnische Anforderungen stellt, die im Rahmen der Arbeit zwar noch realisiert wurden, jedoch nicht vollständig auf korrektes Funktionieren bzw. Erfüllen der Anforderungen getestet werden konnten.

Eine weitere Verbesserung betrifft die Verwaltung der Schulungsmaterialien. Der CIS bietet vielfältige Informationen über die Ablage, jedoch sind nur einfache Routinen zur Verwaltung realisiert. Wünschenswert wären zum Beispiel Funktionen zur Strukturierung der Materialien wie Änderung der Reihenfolge oder direktes Editieren der Metadaten, wie etwa die Vergabe von Schlüsselwörtern. Hierbei sollten dem Dozenten und den Autoren Werkzeuge zur Verfügung gestellt werden, die den Einsatz von Dateibrowsern oder speziellen XML-Editoren überflüssig machen.

Das vorhandene System bietet wenig Funktionalität zum Suchen. Um zum Beispiel bestimmte Materialien für eine Vorlesung zusammenzustellen, ist der Dozent darauf angewiesen, zu wissen, welche Datei welchen Inhalt hat. Die Implementierung eines Werkzeuges, welches zum Beispiel einen Index über die in den Metadaten enthaltenen Informationen erstellt und die Suche nach Schlüsselwörtern ermöglicht, wäre gerade beim Sichten der Materialien hilfreich. Alternativ wäre zu überlegen, ob und wie man die betriebssystemeigene Funktion zum Indizieren von Dateien und Suchen nach Inhalten nutzen könnte.

Eine tiefer greifende Veränderung wäre die Umstellung der strukturierten Schulungsmaterialablage: Das Ersetzen des hierarchischen Dateisystems durch eine Datenbank würde komplexere Suchfunktionalität im Bestand vorhandener Lernmaterialien und flexiblere Konfigurierbarkeit von Kursen ermöglichen. Die Folge einer solchen Maßnahme wäre allerdings die komplette Neuprogrammierung von großen Teilen des CIS, da der Zugriff auf die Daten dann nicht mehr mittels Windows Scripting und dessen Dateisystemobjekten, sondern mit einer Datenbankabfragesprache wie z.B. SQL erfolgen müsste.

7.2 MICROSOFT .NET

Während der Entstehung der Arbeit veröffentlichte Microsoft seine neue .NET-Architektur, die auf eine Java-ähnliche Technik setzt: Software besteht nicht mehr aus systemspezifischem Programmcode, sondern wird zur Laufzeit von einer virtuellen Maschine übersetzt. Dies soll die Portierung auf andere Systeme erleichtern. Das Konzept

ist unabhängig von der verwendeten Programmiersprache: Alle Sprachen nutzen eine einheitliche objektorientierte Schnittstelle und können unmittelbar zusammenarbeiten, und zwar wesentlich einfacher als es über Standards wie CORBA oder COM möglich wäre.

Das in .NET integrierte Prinzip der Webservices könnte verwendet werden, anderen Anwendungen die Funktionalität des CIS einfacher zur Verfügung zu stellen. Webservices ermöglichen es Anwendungen, über Rechnergrenzen hinweg gegenseitig Funktionen aufzurufen und Daten auszutauschen. Bei .NET kommt hierbei das Kommunikationsprotokoll HTTP und die Datenaustauschsprache XML zum Einsatz. Die Schnittstellenbeschreibung wird dabei über ein spezielles XML-Format, die *Web Service Description Language* WSDL geregelt. Diese Infrastruktur für verteilte Software-Systeme könnte man nutzen, um die Funktionen des CIS per Web Service anzubieten. Es müssten lediglich Wrapper-Klassen geschrieben werden, die die neue Schnittstelle auf die ActiveX-dlls abbilden. Dies ist technisch machbar, da die .NET-Laufzeitumgebung parallel zu COM verwendet werden kann.

Eine weitere Erweiterung betrifft den CIB: Er könnte zu einem webbasierten Informationsdienst ausgebaut werden, auf den per Webbrowser zugegriffen werden kann. Eine vorhandene Nutzerschnittstelle in ein Web-Frontend umzusetzen gestaltet sich mit .NET einfacher als bisher: Das neue Programmiermodell ASP.NET generiert aus einer herkömmlichen Bedienoberfläche dynamische Webseiten mit eingebetteten Controls.


Informationen zu .NET und der neuen Entwicklungsumgebung *Visual Studio .NET* finden sich unter [15] und [17].

8 ANHANG

8.1 LITERATUR


- [1] Corba-Website der OMG, <http://www.corba.org/>
- [2] Microsoft DCOM/COM Technologies, <http://www.microsoft.com/com/>
- [3] Roy T. Varughese, *Handbuch IT-Management*, MITP-Verlag, 1998
- [4] <http://www.smarttech.com/products/plasma/index.asp>
- [5] <http://www.wacom-europe.com/de/produkte/cintiq/index.asp>
- [6] Elektronische Kreide, eine Java Multimedia-Tafel, <http://www.ekreide.de/>
- [7] Cisco E-Learning,
<http://www.cisco.com/warp/public/10/wwtraining/elearning/>
- [8] M. Fowler, K. Scott, *UML konzentriert*, Addison Wesley, 2. Auflage, 2000
- [9] D. Feuerhelm, O. Mehl, and S. Abeck, *E-Education-Environment*, presented at Elektronische Geschäftsprozesse, Klagenfurt, Österreich, 2001.
- [10] S. Löbeth, *NTFS - das Dateisystem von NT*, <http://wwwbs.informatik.htw-dresden.de/svortrag/ai93/Loebeth/ntfs.htm>
- [11] F. Sturzebecher, *Konzeption einer Methode zur Durchführung einer digitalen Vorlesung* in Institut für Telematik, Cooperation & Management. Karlsruhe: Universität Karlsruhe (TH), 2001.
- [12] Allgemeine Informationen zu Microsoft NTFS,
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fsys_538t.asp?frame=true
- [13] Informationen zu Microsoft XML, <http://www.microsoft.com/xml>
- [14] Allgemeine Informationen zu XML, <http://www.w3.org/XML/>
- [15] Microsoft .NET, <http://www.microsoft.com/germany/themen/net/>
<http://www.microsoft.com/net/>
- [16] F. Imhoff, O. Spaniol, C. Linhoff-Popien, M. Garschhammer, „Aachen-Münchener Teleteaching unter Best-Effort-Bedingungen“, *Praxis der Informationsverarbeitung und Kommunikation*, vol. 23, pp. 156-163, 2000
- [17] Visual Studio .NET,
<http://www.microsoft.com/germany/ms/entwicklerprodukte/visualnet/features.htm>
- [18] M. Bonn, *Broadcast-Übertragung einer digitalen Vorlesung*, Institut für Telematik, Cooperation & Management. Karlsruhe: Universität Karlsruhe (TH), 2001
- [19] Windows Media Format Tools,
<http://www.microsoft.com/windows/windowsmedia/technologies/resource/format.asp>
- [20] S. Abeck, *ed.tec - An Education Technology Environment for Flexible Teaching and Training*, presented at SSGRR: Infrastructure for e-Business, e-Education, and e-Science on the Internet, L'Aquila, Italien, 2001.

8.2 MANAGEMENT SUMMARY



Cooperation & Management


Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck



Management Summary

„Modellierung und Implementierung eines
Auskunftsdiensstes über Schulungsmaterialien
für die Aus- und Weiterbildung“

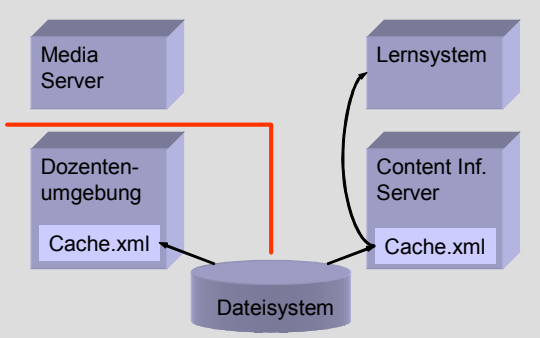
Diplomarbeit Matthias Bonn

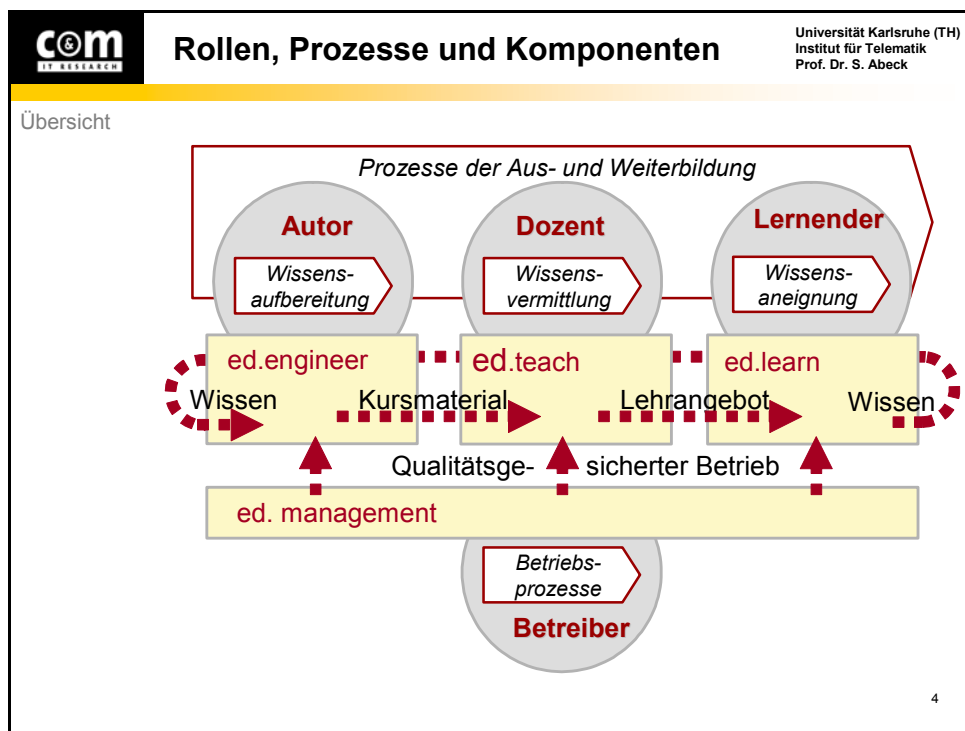
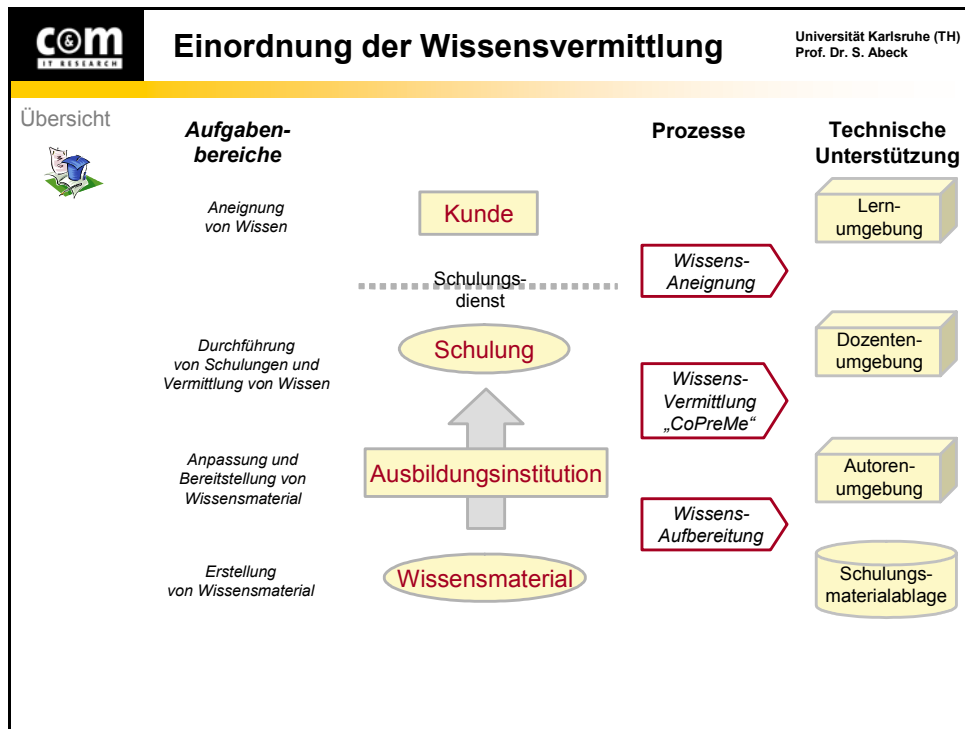


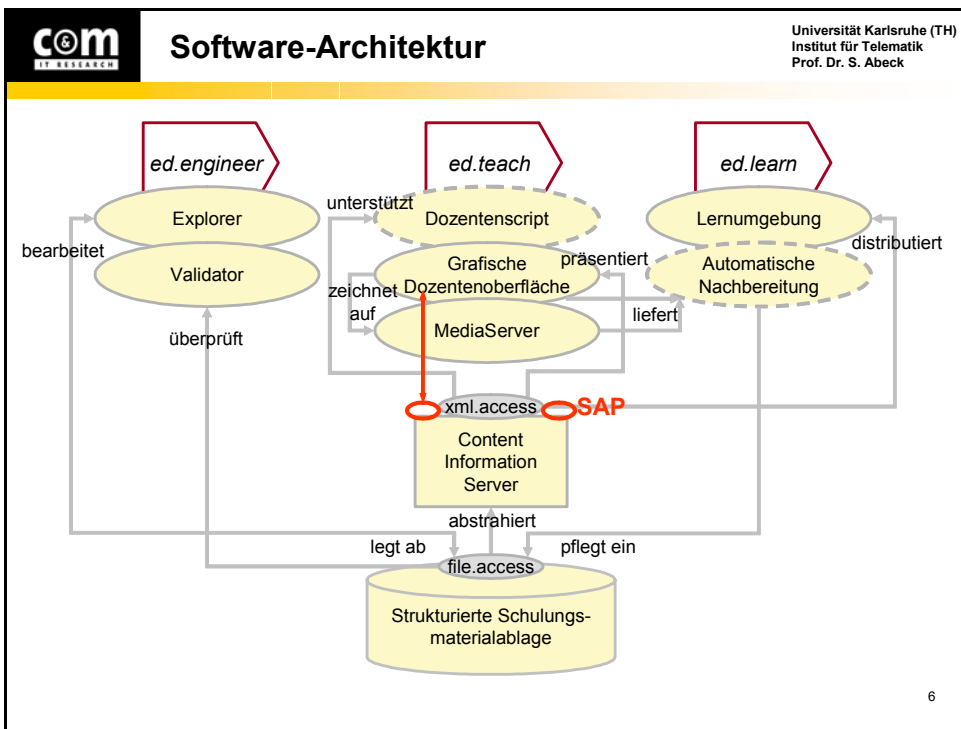
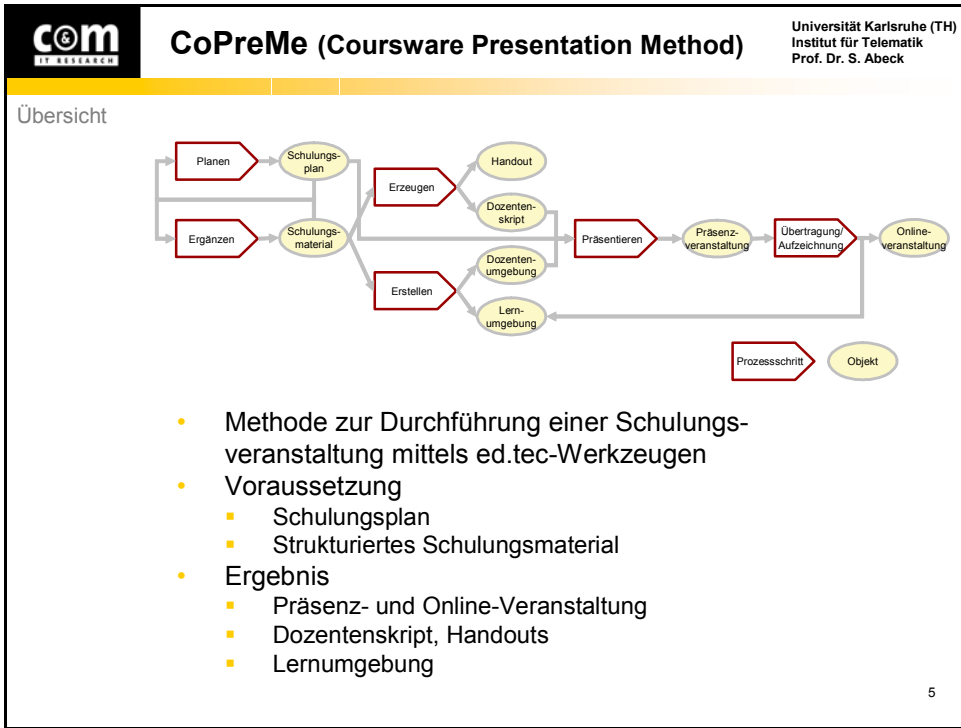
Aktuelle Situation


Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

- drei voneinander stark getrennte Komponenten:
 1. Content Information Server mit Verzeichnisstruktur und cache.xml für's Lernsystem
 2. Dozentenumgebung, die cache.xml selbst erzeugt und benötigte Teile ausliest
 3. Media Server zur Videoerzeugung
- Komponenten kooperieren (nicht) miteinander.
- Bsp.: Sehr viel Funktionen realisiert







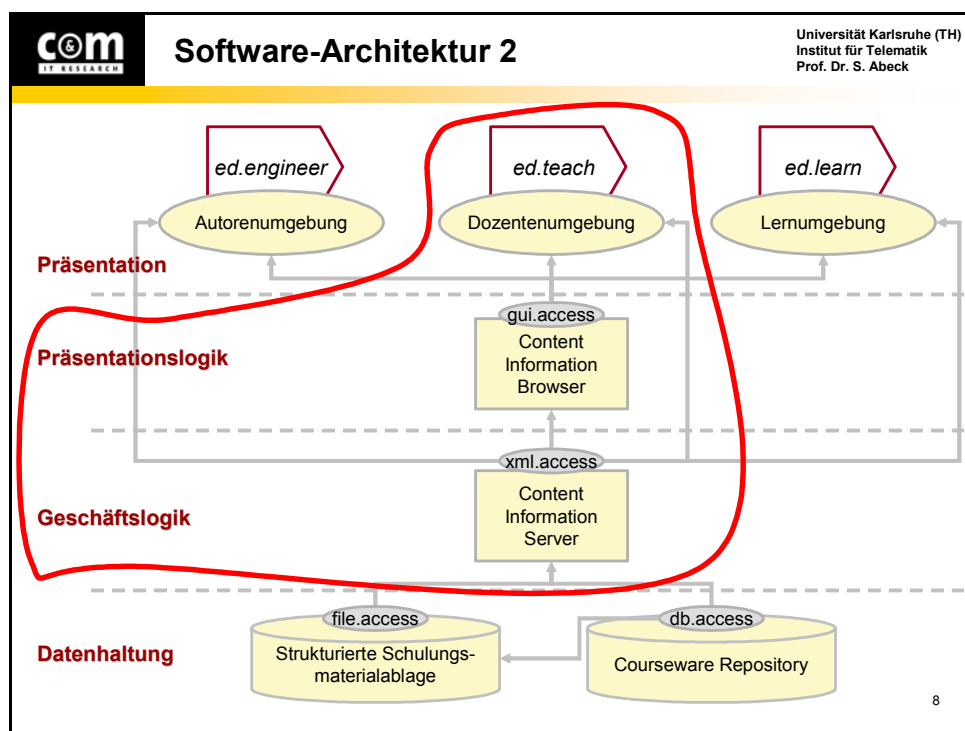


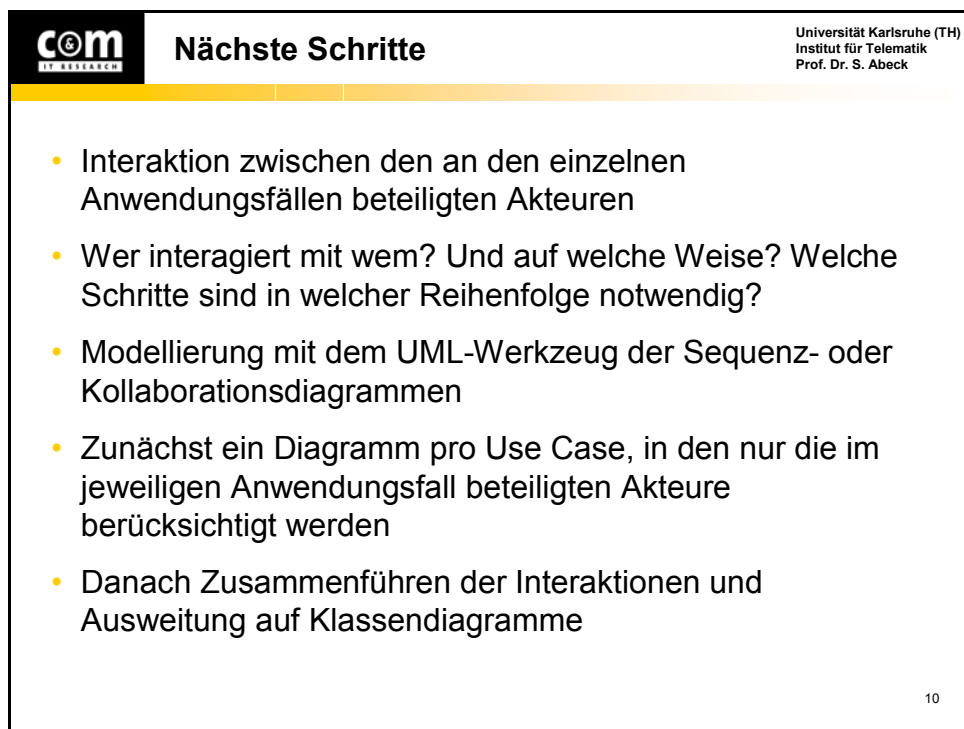
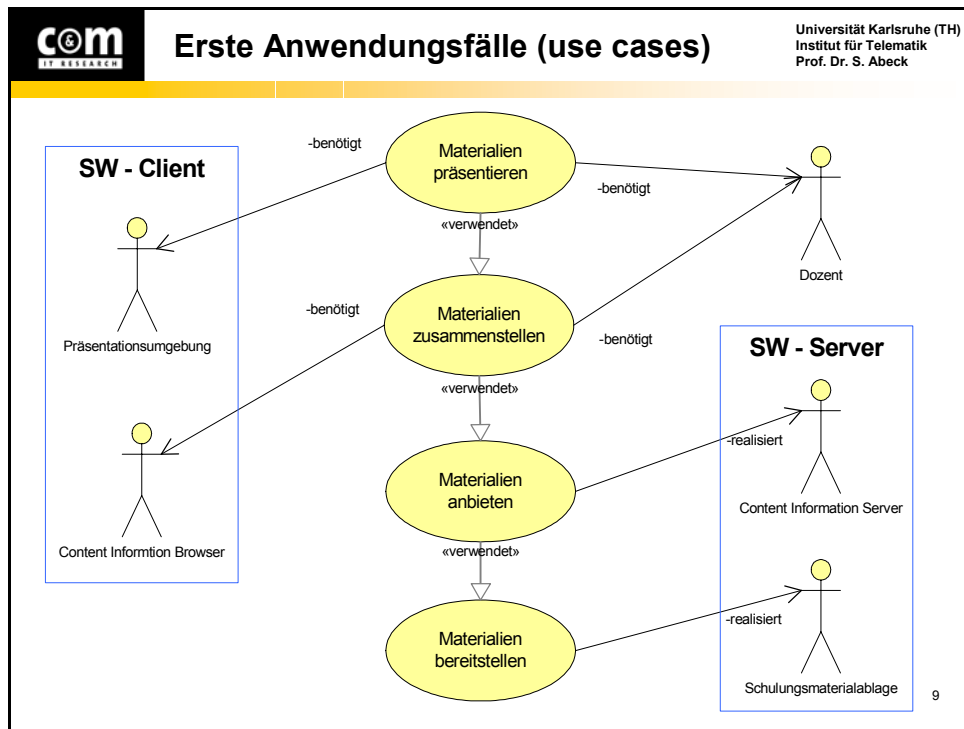
Aufgaben und Ziele

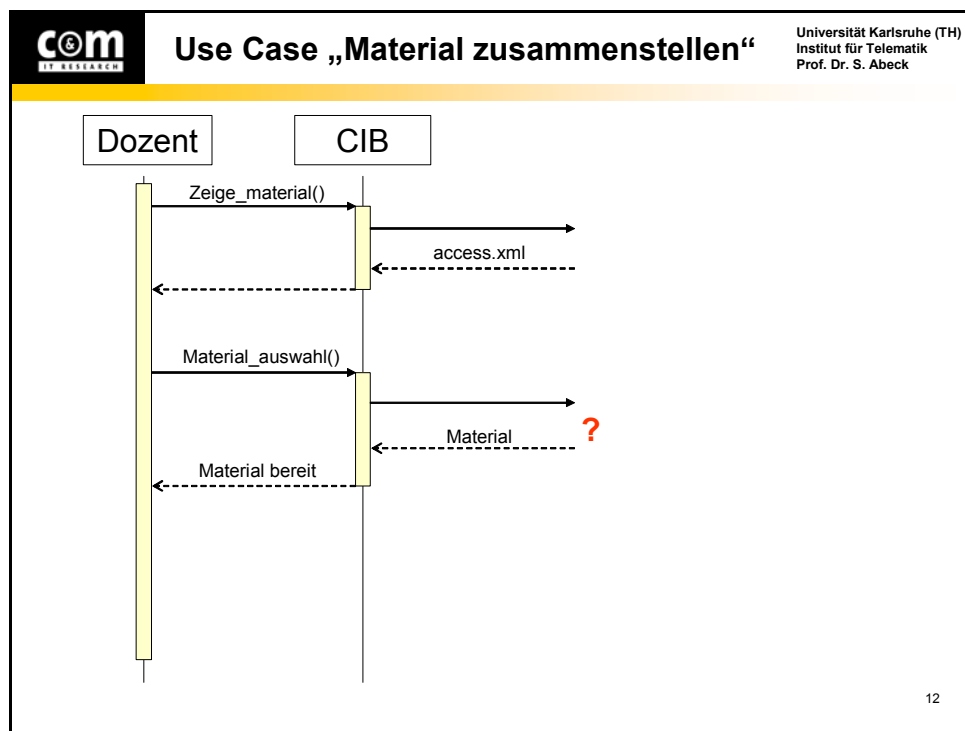
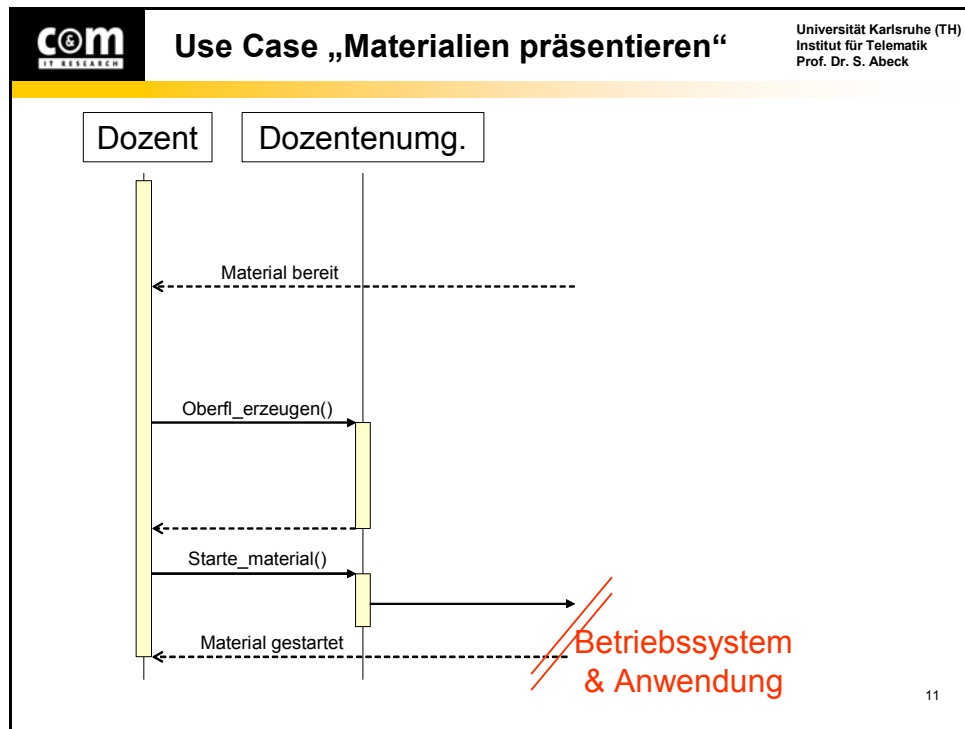
Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

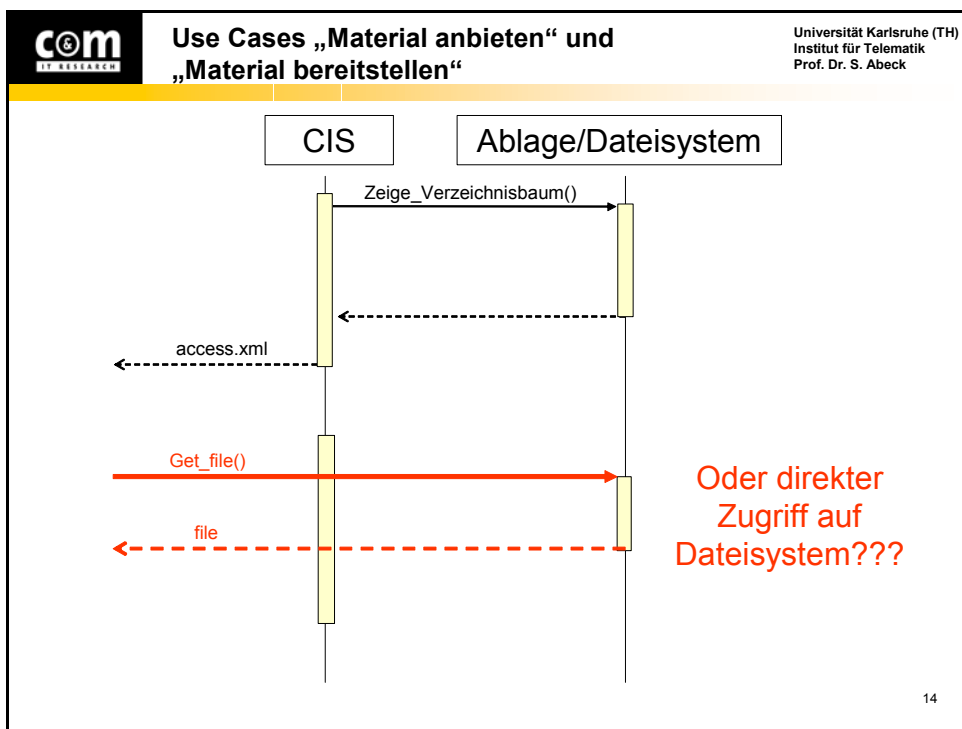
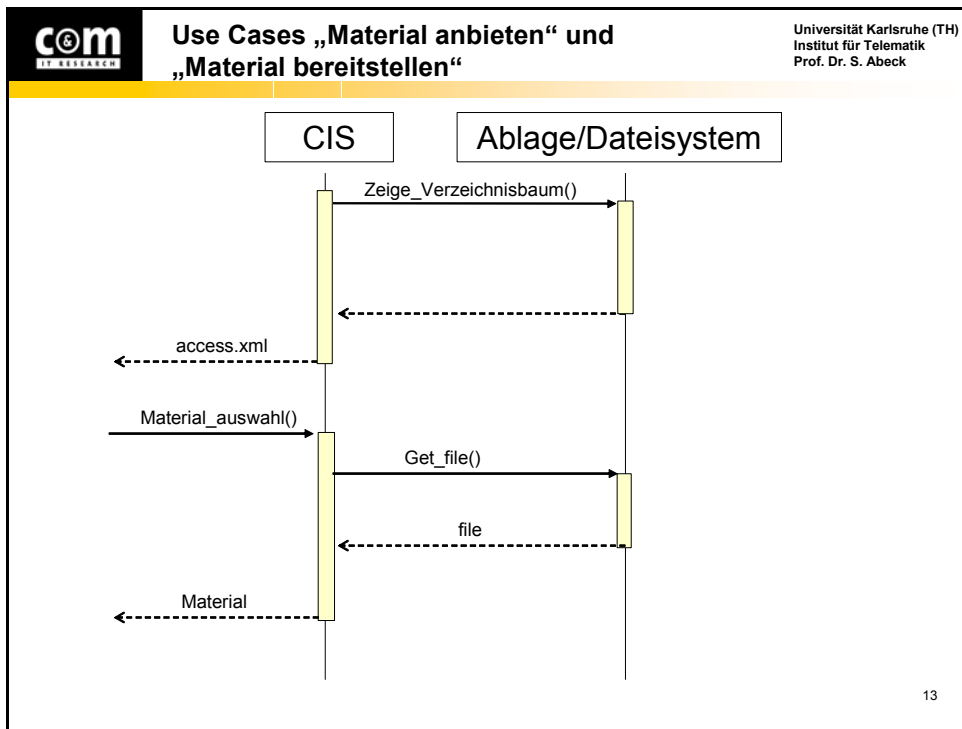
- entfernter Zugriff der Dozentenumgebung auf die Dienstschnittstelle des Content Information Servers (cache.xml soll nicht in Dozentenumgebung nochmals erstellt werden müssen)
- Wie könnte diese Schnittstelle aussehen und welche Dienste könnte sie anbieten? (z.B. Zielgruppenorientierte Auswahl der Materialien)
- Technische Realisierungsmöglichkeiten und (prototypische) Implementierung

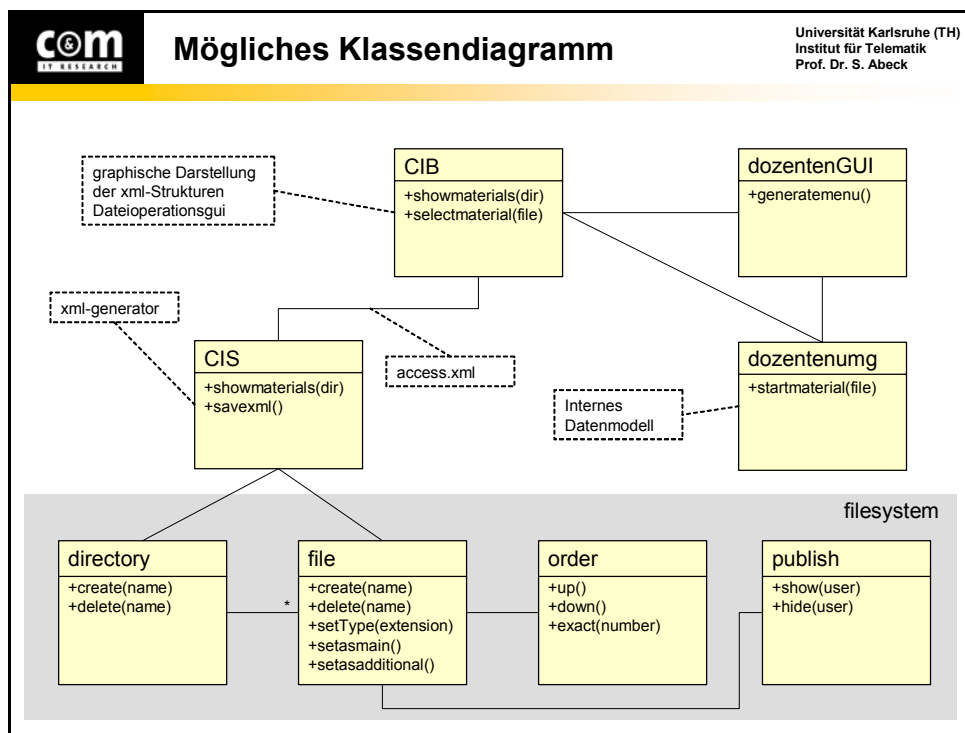
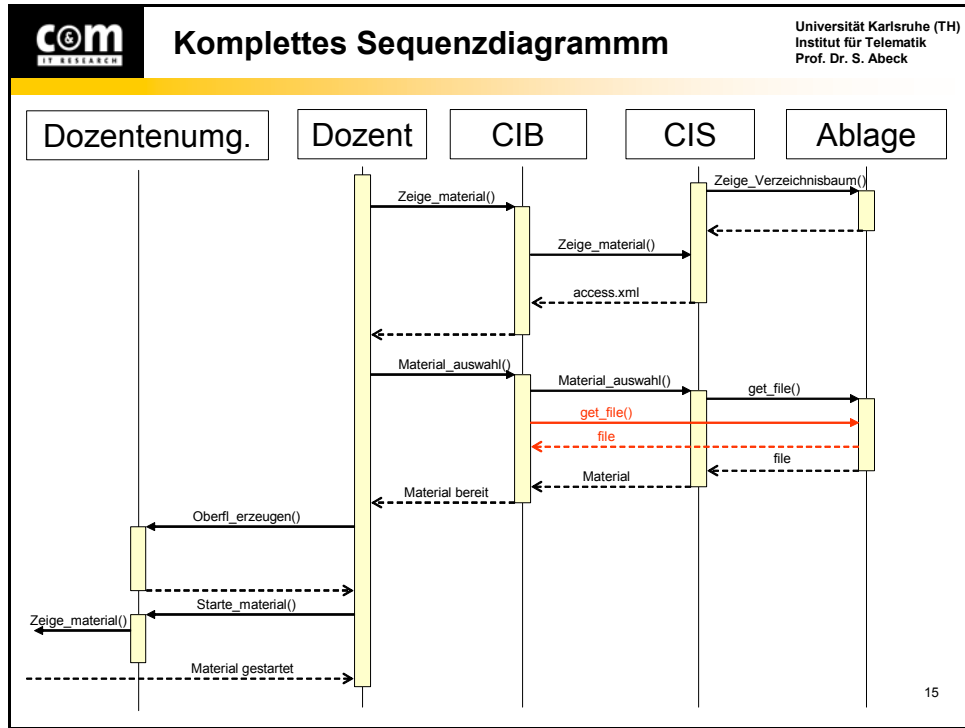
7















Erläuterungen zum Klassendiagramm

Universität Karlsruhe (TH)
 Institut für Telematik
 Prof. Dr. S. Abeck

Ziele:

- Verbesserter modularer Aufbau, Trennung von Darstellung, Anwendungslogik, internen Datenstrukturen und realem Dateisystem, Wiederverwendbarkeit
- Ausgliederung der xml-Erzeugungslogik auf CIS, die Dozentenumgebung soll nur noch vorhandene xml-Dateien parsen, sie aber nicht mehr selber erzeugen
- Zugriff auf das Dateisystem soll über den CIS erfolgen, das Dateisystem wird in einer xml-Datei abstrahiert und mittels dem CIB dargestellt. Er dient auch als Schnittstelle zum Benutzer (zusätzlich zum dozentenGUI)
- Dateisystemoperationen werden durch eigene Klassen verdeckt

17



Die nächsten Schritte

Universität Karlsruhe (TH)
 Institut für Telematik
 Prof. Dr. S. Abeck


- Detaillierterer Entwurf der dateisystemnahen Klassen, Spezifikation der Methoden und Zugriffsparameter
- Detaillierterer Entwurf der Sequenzdiagramme und evtl. von Zustandübergangdiagrammen für komplexere Klassen
- Anpassung des vorhandenen xml-Generators an die neuen Dateinamenskonventionen
- Problem hierbei: Datei bisher entweder „main“, „addon“ oder „hidden“, nun aber vielfältige Kombination zur Klassifizierung möglich:

XOR	main
XOR	addon
XOR	hidden

→

XOR	main
XOR	addon
XOR	learner
OR	docent
OR	author
AND	none

18




Xml-Generator und Dateinamen

Universität Karlsruhe (TH)
 Institut für Telematik
 Prof. Dr. S. Abeck

- Dateiklassifizierungen sind flexibler ⇒ xml-cacher wird komplexer (und cache.xml erweitert sich!)
- Deshalb Aufgliederung in verschiedene Klassen um die verschiedenen Dateisystemoperationen zu realisieren
- Vorteil: Leichtere Anpassung an zukünftige Änderungen der Dateinamenskonventionen (sie werden kommen!) und Verwendbarkeit in anderen Tools (Bsp. Klasse ‚file‘)

file

```
+createfile (filename)
+deletefile (filename)
+setfiletype (filename, format)
+setclass (filename, class)
+getclass (filename)
+setview (filename, visibility)
+delview (filename, visibility)
+islearner (filename) :bool
+isdocent (filename) :bool
+isauthor (filename) :bool
+validate (filename) :pError
```



Der Xml-Generator CIS (1)

Universität Karlsruhe (TH)
 Institut für Telematik
 Prof. Dr. S. Abeck

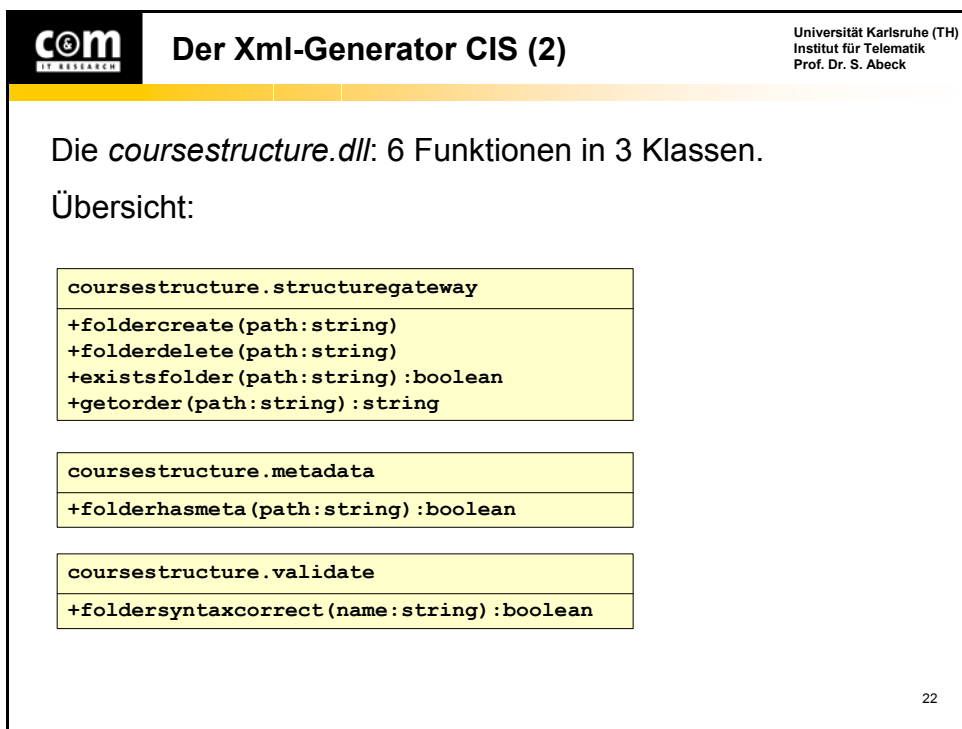
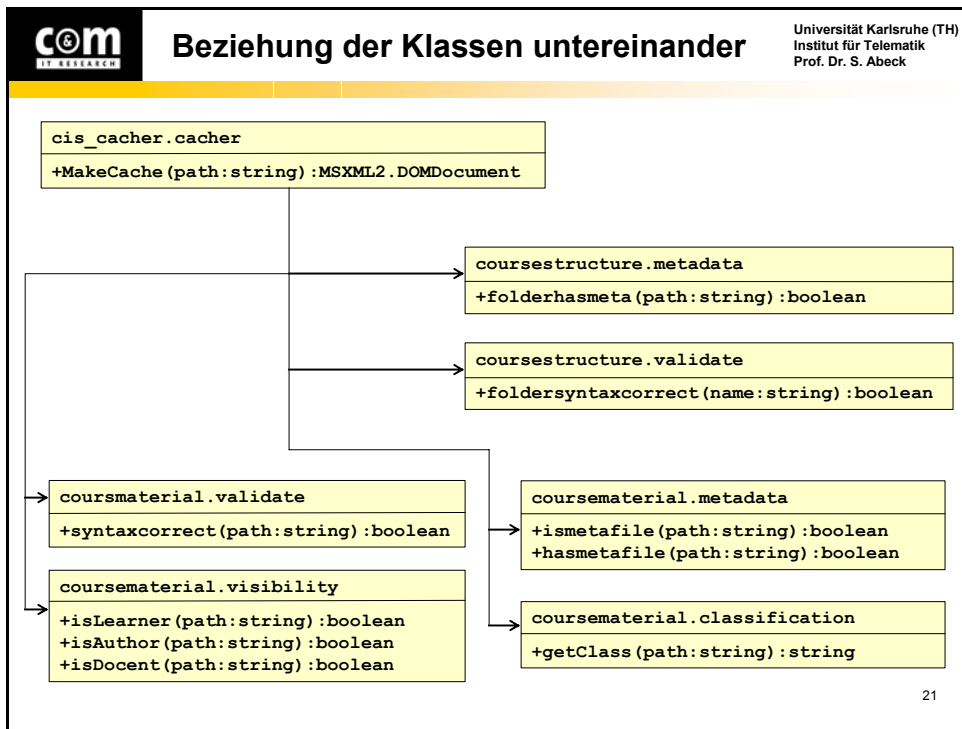
Der Content Information Server besteht aus 3 ActiveX-dlls, (*cis_cacher.dll*, *coursematerial.dll* und *coursestructure.dll*) die „nach aussen“ über 9 Klassen folgende Funktionalität anbieten:


Übersicht über die öffentliche Funktion der Klasse **cis_cacher.cacher**:

```
cis_cacher.cacher
+MakeCache (path: string) :MSXML2.DOMDocument
```

Funktion liefert XML-DOM-Document, das die gesamte Verzeichnisstruktur unterhalb von **path** repräsentiert. Verzeichnisse oder Files, die nicht den Konventionen entsprechen, werden ignoriert

20






Der Xml-Generator CIS (3)

Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

<pre>coursematerial.validate +syntaxcorrect (path:string) :boolean</pre>	<p>Die <i>coursematerial.dll</i>: 17 Funktionen in 5 Klassen.</p>
<pre>coursematerial.materialgateway +createfile (path:string) +deletefile (path:string) +setfiletype (path:string, type:string)</pre>	
<pre>coursematerial.visibility +isLearner (path:string) :boolean +isAuthor (path:string) :boolean +isDocent (path:string) :boolean +isUnknown (path:string) :boolean +setLearner (path:string) :integer +setAuthor (path:string) :integer +setDocent (path:string) :integer +delLearner (path:string) :integer +delAuthor (path:string) :integer +delDocent (path:string) :integer</pre>	
<pre>coursematerial.metadata +ismetafile (path:string) :boolean +hasmetafile (path:string) :boolean</pre>	<pre>coursematerial.classification +getClass (path:string) :string</pre>

23

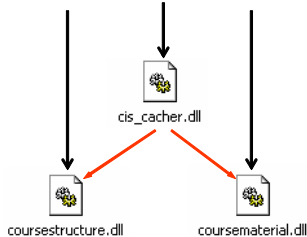


Zusammenspiel der Klassen

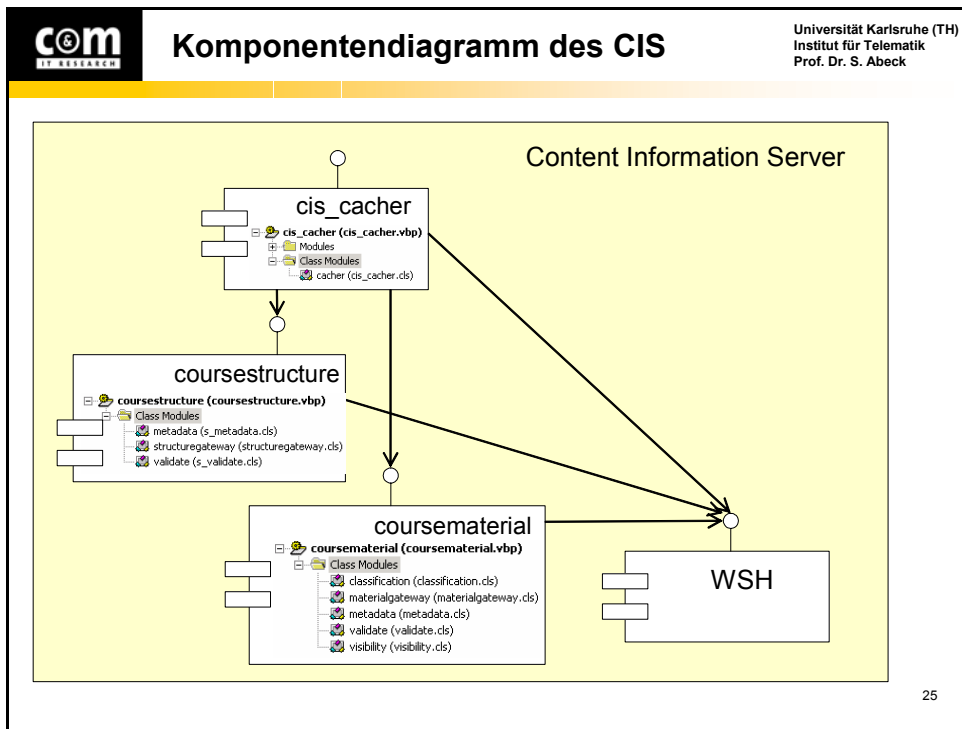
Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

Die Funktionen der Klassen stehen öffentlich zur Verfügung, jedoch ist zu beachten, dass die Klasse **cis_cacher.cacher** einige Funktionen der Klassen in den beiden anderen dlls benutzt, ohne diese also nicht funktionsfähig ist.

coursestructure und **coursematerial** benötigen jeweils keine weiteren dlls, um ihre Funktion zu erbringen.



24




com IT RESEARCH **Realisierung und Benutzung** Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

- Die Klassen stehen als VB-Klassenmodul und als ActiveX-DLL zur Verfügung
- Verwendung erfolgt entweder durch Einkopieren der Klassenmodule in den Code, oder durch eine Referenz auf die benötigten DLLs (in der Entwicklungsumgebung einstellbar)
- Zugriff erfolgt in VB durch folgendes Konstrukt (am Beispiel der cis_cache.dll):

```
Dim meinxml as MSXML2.DOMDocument
Dim xmlcache as cis_cacher.cacher
Set xmlcache = New cis_cacher.cacher
Set meinxml = xmlcache.MakeCache („d:\vorlesung\material“)
```

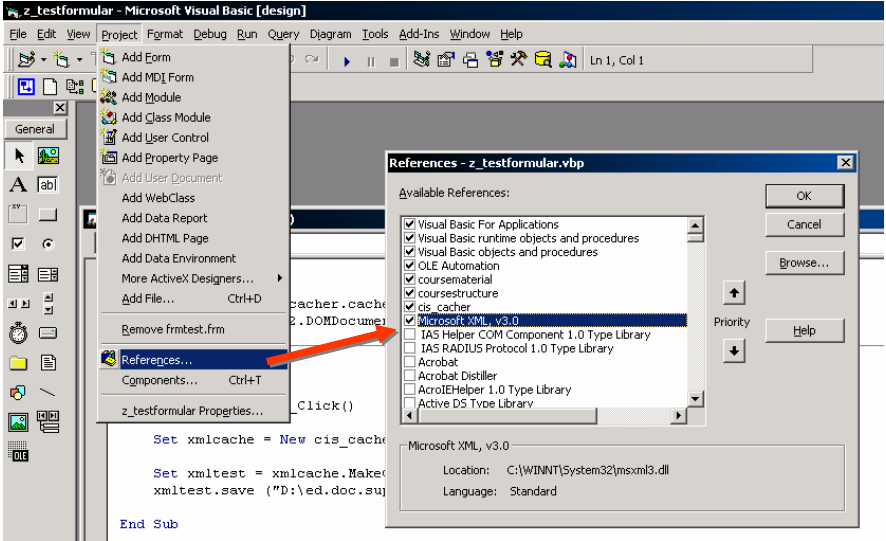
Referenz auf Microsoft XML 3.0 benötigt!

26




Einrichten der Referenzen

Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck



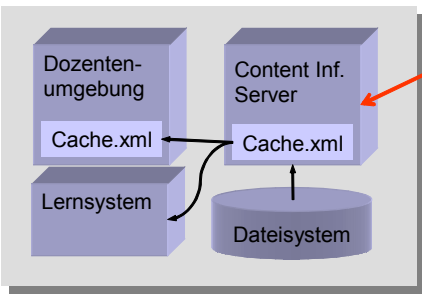
27



Nächste Schritte

Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

CIS-DLLs sollen nach Möglichkeit zentral und über das Netz zugreifbar gespeichert werden




Nur hier sollen Algorithmen zur xml-Erzeugung liegen!
Wie realisierbar?

Idee: Speicherort der dlls auf dem Server als Netzlaufwerk mounten und in Path-Variable eintragen. Dann müsste Win2000 die dlls finden und der Funktionsaufruf auf dem Server möglich sein.

funktioniert so nicht!

Darstellung des xml-caches in Dozenten-umgebung ist zu lösen

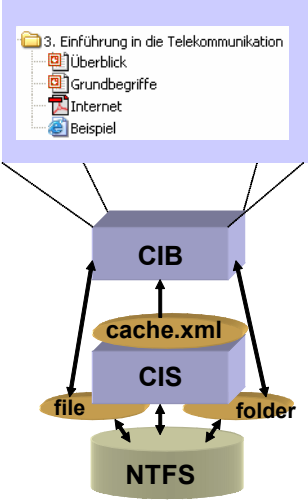
28




Der Content Information Browser (CIB)

Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

- Graphisch brauchbare Darstellung der cache.xml (nur aus ihr wird gelesen!!)
- Filtern der Information entsprechend dem Einsatzzweck (z.B. soll in der Dozentenumgebung nur Material mit Dozentenlesbarkeit angezeigt werden) und ähnliche Funktionen
- Auswahl/Markieren der benötigten Materialien
- Realisierung in VB als (Klassen?)-Modul und Formular (für GUI nötig)
- Dabei wichtig: Leichter „Einbau“ in vorhandene Werkzeuge ermöglichen!



29

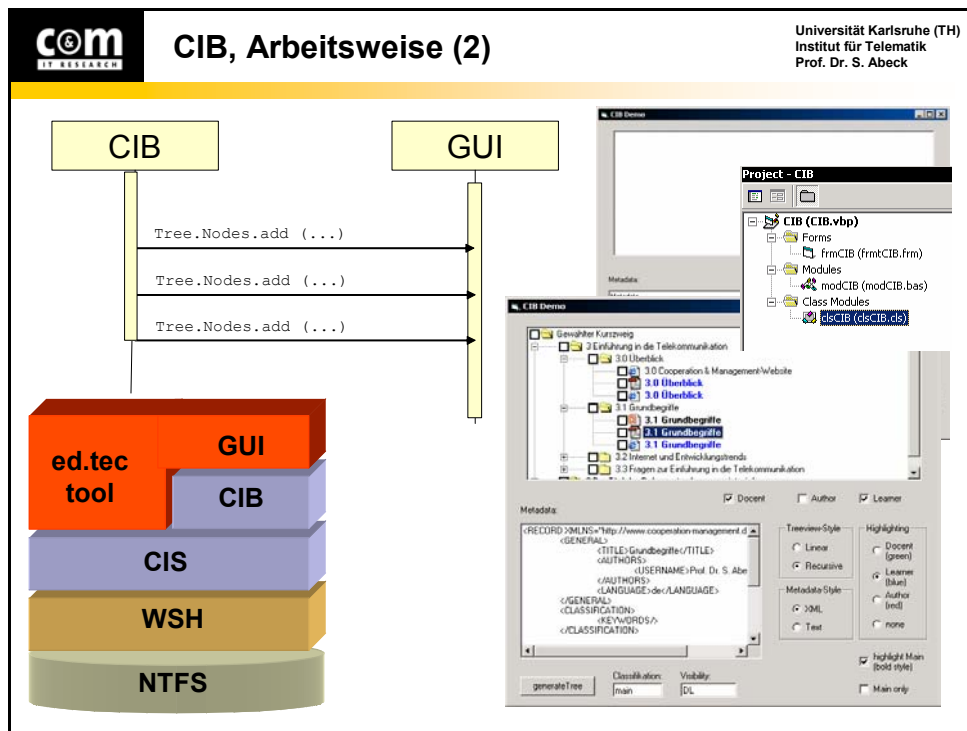
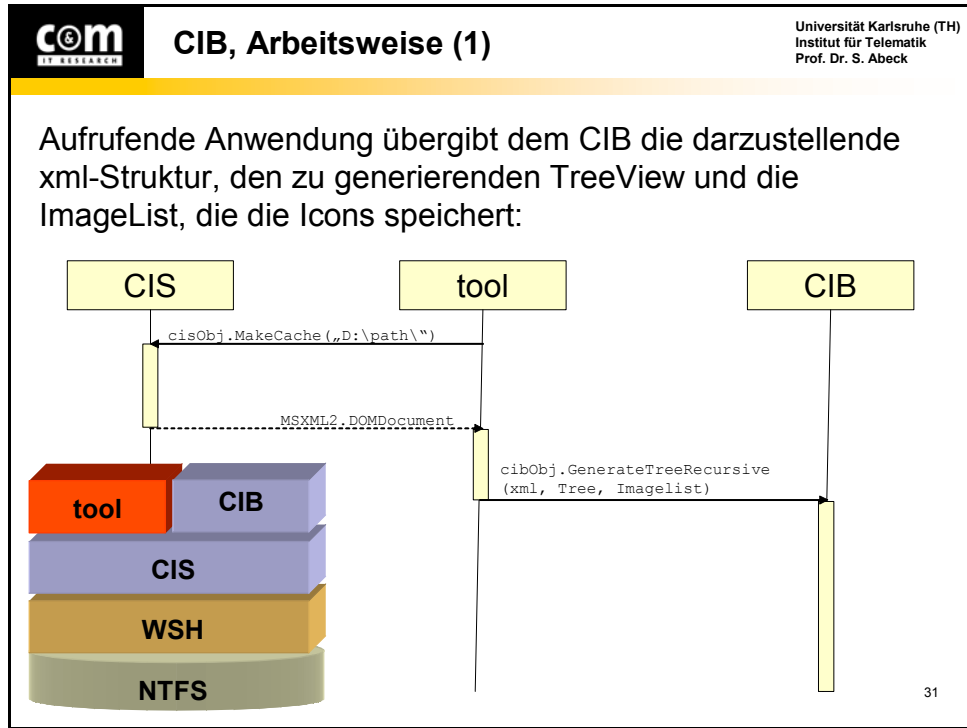



CIB, allgemeines

Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

- Realisierung des Content Information Browsers als VB-Klassenmodul `clsCIB.cls`
- Funktion setzt Formular mit GUI-Komponenten `TreeView` und `ImageList` voraus (Demo-Formular: `frmCIB.frm`)
- CIB bietet verschiedene Funktionalität:
 - Darstellen Materialablage linear
 - Darstellen Materialablage rekursiv
 - Ein und Ausblenden je nach Sichtbarkeit und Klassifikation
 - Optische Hervorhebung je nach Sichtbarkeit und Klassifikation
 - Auslesen Metadaten und Anzeige als xml oder Text
 - Auslesen Klassifikation und Sichtbarkeit als Text
- CIB arbeitet auf XML-Struktur des CIS, d.h. verwendende Anwendung muss zuerst per CIS-Aufruf ein xml-DOMDocument generieren lassen.

30





CIB, vereinfachtes Klassendiagramm

Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

clsCIB


+showdocent: Boolean
+showmainonly: Boolean
+highlightdocent: Boolean
+highlightmain: Boolean

+generateTreeLinear (folder: MSXML2.DOMDocument, tree: TreeView, iconliste: ImageList)
+generateTreeRecursive (folder: MSXML2.DOMDocument, tree: TreeView, iconliste: ImageList)
+getMetaData (node: MSComctlLib.node, metadatastyle: String) : String
+getVisibilities (node: MSComctlLib.node) : String
+getClassification (node: MSComctlLib.node) : String

Attribute zum Einstellen der gewünschten Sichtbarkeiten, Klassifikationen und Hervorhebungen

Liefert die Metadaten eines Baumknotens (Übergabeparameter der tree_NodeClick Methode eines TreeViews)

33



Codebeispiel der aufrufenden Anw.

Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

```

Option Explicit
Dim mycib As clsCIB
Dim xmlcacher As cis_cacher.cacher
Dim meinxml As MSXML2.DOMDocument


Public Sub main()
    frmCIB.Show
    Set mycib = New clsCIB
    Set xmlcacher = New cis_cacher.cacher
    Set meinxml = xmlcacher.MakeCache("D:\ed.doc.sup_test\testkurs2")
    mycib.showdocent = true
    mycib.showmainonly = true
    Call mycib.generateTreeRecursive(meinxml, frmCIB.tree, frmCIB.icons)
End Sub

```

Project - CIB

- CIB (CIB.vbp)
 - Forms
 - frmCIB (frmCIB.frm)
 - Modules
 - modCIB (modCIB.bas)
 - Class Modules
 - clsCIB (clsCIB.cls)

34



Beispiel einer Anwendung für CIS/CIB


Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

Aufgaben einer Dozentenumgebung:

- Transparente Nutzung der Präsentationswerkzeuge
- Permanenter Überblick über den Ablauf der Schulung
- Einfacher Aufruf der Schulungsmaterialien
- Flexible Präsentationsabfolge
- Wiederholender Aufruf der Schulungsmaterialien
- Unterstützung der Remote-Veranstaltung

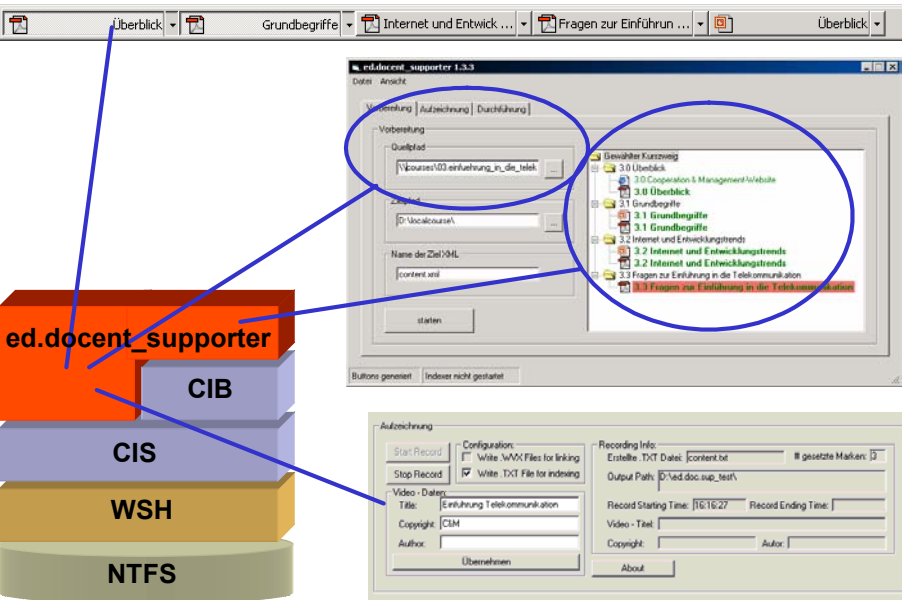
Dabei möglichst auf vorhandene Techniken, Tools und Datenstrukturen aufsetzen

35



Die Dozentenumgebung

Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck




ed.docent_supporter

CIB

CIS

WSH

NTFS



Dozentenumgebung intern

Universität Karlsruhe (TH)
Institut für Telematik
Prof. Dr. S. Abeck

XML als interne Datenstruktur

Benutzung des CIS

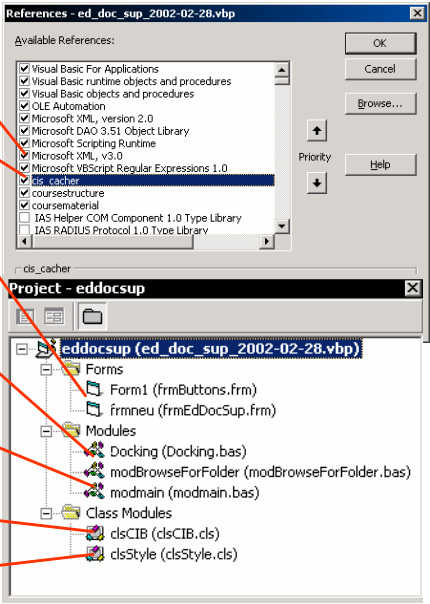
2 GUIs, enthalten Routinen zur
Behandlung der Benutzer-
Eingaben und Ausgabe

Module zur
Buttonleistendarstellung

Hauptmodul mit
Hauptalgorithmen

Zugriff auf CIB

Buttonleistendarstellung



The screenshot shows two windows from the Visual Basic IDE:

- References - ed_doc_sup_2002-02-28.vbp:** A list of available references with checkboxes. The 'cls_cacher' reference is highlighted. Other references include Visual Basic For Applications, Visual Basic runtime objects and procedures, OLE Automation, Microsoft XML, version 2.0, Microsoft DAO 3.51 Object Library, Microsoft Scripting Runtime, Microsoft XML, v3.0, Microsoft VBScript Regular Expressions 1.0, coursestructure, coursematerial, IAS Helper COM Component 1.0 Type Library, and IAS RADIUS Protocol 1.0 Type Library.
- Project - eddocsup:** A tree view showing the project structure. The 'eddocsup (ed_doc_sup_2002-02-28.vbp)' folder is expanded, showing subfolders for Forms, Modules, and Class Modules. Under Forms, there are 'Form1 (frmButtons.frm)' and 'frmneu (frmEdDocSup.frm)'. Under Modules, there are 'Docking (Docking.bas)', 'modBrowseForFolder (modBrowseForFolder.bas)', and 'modmain (modmain.bas)'. Under Class Modules, there are 'clsCIB (clsCIB.cls)' and 'clsStyle (clsStyle.cls)'.

Red arrows indicate the following connections:

- 'XML als interne Datenstruktur' points to 'Microsoft XML, version 2.0' in the References window.
- 'Benutzung des CIS' points to 'cls_cacher' in the References window.
- '2 GUIs, enthalten Routinen zur Behandlung der Benutzer-Eingaben und Ausgabe' points to 'Form1 (frmButtons.frm)' and 'frmneu (frmEdDocSup.frm)' in the Project Explorer.
- 'Module zur Buttonleistendarstellung' points to 'clsStyle (clsStyle.cls)' in the Project Explorer.
- 'Hauptmodul mit Hauptalgorithmen' points to 'modmain (modmain.bas)' in the Project Explorer.
- 'Zugriff auf CIB' points to 'clsCIB (clsCIB.cls)' in the Project Explorer.
- 'Buttonleistendarstellung' points to 'clsStyle (clsStyle.cls)' in the Project Explorer.