Generatorunterstützte objektorientierte Entwicklung multimedialer Lehr- und Lernsysteme zur Effizienzsteigerung und Qualitätsverbesserung

Christian Weidauer

Lehrstuhl für Software-Technik
Ruhr-Universität Bochum
Universitätsstr.150, D-44780 Bochum, Germany
Tel: +49 234-32-26794
E-Mail: weidauer@swt.ruhr-uni-bochum.de

Zusammenfassung. Die Entwicklung von Multimedia-Anwendungen und multimedialen Lehr- und Lernsystemen (MMLLS) gewinnt zunehmend an Umfang und Bedeutung. Systematische Methoden und Werkzeuge zu ihrer Erstellung sind erforderlich, um Produktivitäts- und Qualitätssteigerungen zu erreichen.

Dieser Artikel stellt eine Entwicklungsmethode für MMLLS vor, die auf objektorientierter Modellierung aufsetzt, und durch Generierung eines Großteils des MMLLS die Produktivität und Qualität verbessert. Die Methode schließt die werkzeugunterstützte Erfassung und Verwaltung der Inhalte ein. Mit dieser Methode sind bereits Projekte erfolgreich realisiert worden. Die gewonnenen Erfahrungen werden ebenfalls dargestellt.

Schlüsselwörter: Entwicklungsmethode für multimediale Lehr- und Lernsysteme, Generierung, CASE, Modellierung von Übungsaufgaben

Abstract. Development of multimedia applications and Computer Based Training (CBT) systems is getting increasingly important. A systematic method and tools are necessary to increase productivity and quality. This paper shows a method to develop CBTs. The method bases on object oriented modelling and improves productivity and quality by generating most parts of CBTs. The method supports both computer-aided acquisition and computer-aided administration of the CBT content. Some projects were already realized successfully by using this method. The gained experiences will be presented as well.

Key words: Method for CBT development, generation, CASE, modelling of exercises

1 Einleitung

1.1 Multimedia-Entwicklung im Kontext der Software-Technik

Wie in der modernen Software-Technik üblich beginnt auch die Entwicklung multimedialer Lehr- und Lernsysteme (MMLLS) mit der Anforderungsanalyse und der Definition des MMLLS. In diesem Artikel wird die Bezeichnung *multimediale* Lehr- und Lernsysteme verwendet, und ist so zu verstehen, dass die Systeme multimedial sein können, aber nicht sein müssen. Die Anforderungsanalyse und die Definition des MMLLS geschieht vor dem Hintergrund, dass die MMLLS-Entwicklung als Software-Entwicklung betrachtet wird. Anhand der Spezifikationen kann das MMLLS realisiert werden. In der Regel geschieht dies mit Hilfe von Autorensystemen. Die Realisierung erfolgt derzeit in erster Linie manuell. Dieses entspricht weitestgehend dem frühen Stand der Software-Technik, als Anwendungen noch ohne Werkzeugunterstützung programmiert wurden. Es ergeben sich vergleichbare Effekte: So treten leicht Fehler bei der Umsetzung auf und die Umsetzung ist sehr aufwendig.

Das Ziel muss es daher sein, den Entwicklungsstand der Multimedia-Entwicklung dem Entwicklungsstand der modernen Software-Technik anzugleichen.

Für die Spezifikation multimedialer Systeme gibt es zahlreiche Ansätze, wie die Relationship Management Methode RMM [8], die Object Oriented Hypermedia Development Methode OOHDM [16] oder auch der HyDev (von Hypermedia Development)-Ansatz [12]. Diese Ansätze befassen sich mit dem authoring-in-the-large (vgl. [6]). Es stellt sich nun die Frage, wie eine Angleichung des Entwicklungsstandes auch bei der Implementierung erreicht werden kann.

In [1] wird dieses Problem aufgegriffen und die folgende Lösungsskizze formuliert: »Ein erster Schritt kann darin bestehen, das Multimedia-Anwendungsspektrum in Anwendungskategorien zu gliedern und die Charakteristika pro Kategorie herauszuarbeiten. Je gleichartiger sich die Anwendungen innerhalb einer Kategorie darstellen, desto einfacher wird es sein, Konzepte, Methoden und Werkzeuge zur Entwicklung zu erstellen. Alle Möglichkeiten der Generierung – insbesondere bei der Benutzerinteraktion – sollten genutzt werden.«
Aus den Erfahrungen bei der Entwicklung eines MMLLS zur objektorientierten Analyse [3] benennen wir in [1] folgende Probleme: »

- Die Programmierung der Benutzerinteraktion durch verschiedene Teams ist **fehleranfällig,** und die Sicherstellung der Einheitlichkeit ist schwierig.
- Der Einsatz eines Autorensystems ist **aufwendig.** Komplexe Interaktionen müssen programmiert werden.
- Für Autoren ohne Programmiererfahrung sind **nur Standardfälle** erstellbar, z. B. einfache Grafiken erstellen, farbige Grafiken importieren, Grafiken und Texte positionieren und animieren, einfache Interaktionen erstellen.
- Umfangreiche Anwendungen sind ohne Software-Ingenieure nicht zu erstellen.
- **Änderungen sind aufwendig** durchzuführen, da teilweise auf Pixelebene gearbeitet wird.

- Problematisch ist die **Verwaltung, Versionierung** und **Konfigurierung** der vielen Komponenten, da die Autorensysteme in der Regel keine Versions- und Konfigurationsverwaltung unterstützen.
- Gefährlich ist die **Abhängigkeit vom Hersteller** des jeweiligen Autorensystems. Jeder Wechsel zu einem anderen Autorensystem oder einer üblichen Programmiersprache wie Java erfordert eine vollständige Neuimplementierung.
- Ein Lehr- und Lernsystem enthält gewisse, immer ähnlich wiederkehrende Elemente, die es nahe legen, dafür einen **allgemeinen Rahmen** bereitzustellen.«
- Darüber hinaus sind die Aspekte der **Wiederverwendung** sowohl von **Inhalten** als auch von **technischen Komponenten** zu berücksichtigen.

Wadsack und Schäfer gelangen in [14] zu einer vergleichbaren Einschätzung und fordern zur Entwicklung von MMLLS: »Um solche komplexen Systeme zu realisieren werden Werkzeuge gebraucht, die den Entwicklungsprozess von der Planung bis hin zur Wartung unterstützen. Werkzeuge erhöhen die Produktivität und verbessern die Qualität von Softwaresystemen, durch den Einsatz von modernen Techniken wie Objektorientierung, graphische Architekturbeschreibung und Generierung von Code. Außerdem werden Werkzeuge immer wichtiger, um die Arbeit im Team zu koordinieren und die Konsistenz des Systems während der Entwicklung und darüber hinaus zu sichern.«

Ein Großteil des Aufwandes bei der Erstellung multimedialer Lehr- und Lernsysteme ist auf die Erstellung der Übungsaufgaben für den Benutzer zurückzuführen. In [15] wird die Gestaltung der Aufgaben und des Feedbacks als die schwierigste Aufgabe für den Lernprogramm-Autor angesehen. Die Aufgabentypen, die in diesen Systemen eingesetzt werden, sind größtenteils standardisiert und lassen sie daher für die Generierung interessant erscheinen, um so den Implementierungsaufwand um Größenordnungen zu reduzieren. Bei der Methodendarstellung wird im Folgenden ausführlich auf die Integration und Modellierung der Aufgaben eingegangen. Die Spezifikation von Aufgaben fällt in den Bereich des authoring-in-the-small (vgl. [6]).

Im Folgenden stelle ich meine Methode vor, die die Generierung eines Großteils eines MMLLS ermöglicht. Exemplarisch wird sie anhand der Entwicklung eines MMLLS dargestellt, das zum Thema »Objektorientierte Analyse« entwickelt wurde,.

Die Methode basiert auf der komponentenbasierten Struktur von MMLLS. Sie greift die dargestellten Probleme auf und löst sie.

Insbesondere werden durch das generatorgestützte Verfahren

- der Aufwand und die Fehleranfälligkeit erheblich reduziert,
- Teamentwicklungen vereinfacht,
- Fachautoren ohne Programmierkenntnisse in die Lage versetzt MMLLS zu erstellen,
- Verwaltung und Wiederwendung von Inhalten unterstützt und
- die partielle Unabhängigkeit von einem einzelnen Hersteller erreicht.

1.2 Die modulare Architektur von MMLLS

MMLLS als Anwendungsrahmen. Der modulare Charakter von MMLLS ergibt sich zum einen aus der Gliederung in die zentralen Bestandteilen eines MMLLS (vgl. [13]):

- Didaktikkomponente,
- Lernermodell,
- Wissensmodell und
- Benutzungsschnittstelle.

Zum anderen stellt auch eine fachlich-inhaltliche Gliederung in Lerneinheiten und Lernobjekte eine Zerlegung in einzelne Lernmodule dar. Somit sind Lernelemente (z. B. Aufgaben, Simulationen, Präsentationen), die sich aus Medienobjekten (Text, Video, Audio etc.) und weiteren Modulen zusammensetzen können, ihrerseits auch Module. Lernelemente lassen sich weiter zu Lerneinheiten zusammenfassen. Die Module eines MMLLS verfügen über unterschiedliche Granularität und Komplexität. MMLLS bestehen somit bei dieser Sichtweise aus einem Anwendungsrahmen, der eine Vielzahl von unterschiedlichen Modulen enthält, verwaltet und zugänglich macht.

Möglichkeiten der Objektorientierung. Module lassen sich mit den Mitteln zur objektorientierten Analyse und Entwurf beschreiben. Da das MMLLS einen Anwendungsrahmen für Lernmodule darstellt, ist es sinnvoll, auch einen technischen Anwendungsrahmen zu verwenden, in dem die Lernmodule technisch als Klassen und Objekte abgebildet werden. Da diese Module Informationen und Ereignisse untereinander bzw. mit dem Anwendungsrahmen austauschen, bietet es sich an, existierende Komponentenarchitekturen (wie z.B. JavaBeans) hierzu zu verwenden. Komponentenarchitekturen definieren u.a. das Zusammenwirken ihrer Komponenten. Die Lernmodule werden dann als technische Komponenten realisiert.

Die Möglichkeiten der Objektorientierung unterstützen die (komponentenbasierte) MMLLS-Entwicklung, indem

- durch Vererbung einzelne Klassen und Komponenten spezialisiert werden können, ohne sie vollständig neu entwickeln zu müssen,
- sich generierte Klassen und Komponenten verwenden lassen,
- vorgefertigte Klassen und Komponenten adaptiert werden,
- Darstellung und Inhalt getrennt werden,
- Komponenten leicht zu kombinieren, zu ergänzen und auszutauschen sind,
- bestehende Anwendungsrahmen wiederverwendbar und
- einzelne kleinere Einheiten entwickelbar, testbar und wartbar sind.

2 Methode zur werkzeuggestützten Generierung von MMLLS

2.1 Überblick

Für die Modellierung multimedialer Systeme gibt es verschiedene Methoden (vgl. Einleitung). Der im Folgenden vorgestellte Generierungsansatz greift zur Inhaltsmodellierung auf das objektorientierte Analysemodell zurück, wie es sich bei OOHDM [16] ergibt. Dieses OOA-Modell stellt den Ausgangspunkt des anschließenden Generierungsprozess dar. Ein Überblick über den gesamten Entwicklungsprozess ist in Abb. 1 dargestellt.

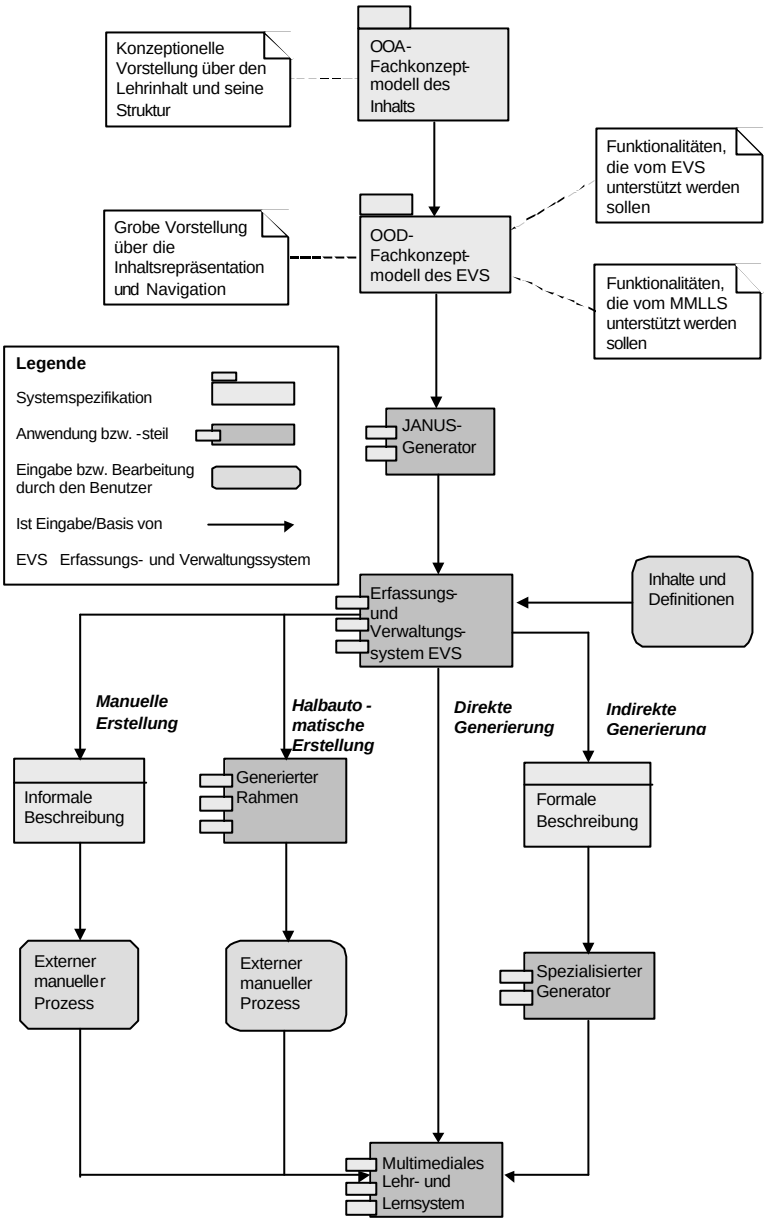

Abb. 1. Überblick über die werkzeuggestützte Entwicklungsmethode

Dieser Prozess beginnt mit der Analyse der Inhalts- und Wissensstruktur, der mit dem MMLLS zu vermittelnden Inhalte. Hierdurch gelangt man zum dem Fachkon-

zept-Modell, wie es von OOHDM her bekannt ist. Die generierbaren Komponenten werden identifiziert. Anschließend werden Unterstützungskomponenten wie Datenbank und Statistik, verschiedene Aufgabentypen und spezielle Generierungsfunktionen hinzugefügt. Diese Komponenten sind individuell anzupassen bzw. zu entwickeln. Die Implementierung des Erfassungs- und Verwaltungssystems für die MMLLS-Inhalte wird durch die Verwendung des JANUS[1]-Generatorsystems [4] unterstützt. Das JANUS-Generatorsystem implementiert aus dem OOD-Modell automatisch das individuelle Erfassungs- und Verwaltungssystem (EVS) für das MMLLS mit Benutzungsoberfläche (s. [7], [9]), Anwendungsserver und relationaler oder objektorientierter Datenbankanbindung. Exportschnittstellen und spezielle Funktionalitäten werden in dem generierten EVS implementiert. Hierbei kann entwickelter Code leicht wiederverwendet und angepasst werden.

Um das MMLLS zu generieren, muss eine Transformation der Wissensfakten und Inhalte in die multimediale Gesamtrepräsentation definiert werden. So muss beispielsweise definiert werden, wie und wo einzeln vorliegende Ton- und Grafikdateien oder Textinformationen in das MMLLS integriert werden. Morris [11] betrachtet den gesamten Entwicklungsprozess von Multimedia-Anwendungen, von denen MMLLS ein Teilbereich sind, als eine Folge von Medien-Transformationen. Teilweise können diese Transformationen automatisch durchgeführt werden. Andere Transformationen müssen manuell bzw. halbautomatisch durchgeführt werden. Ziel ist es, so viele dieser Transformationen wie möglich automatisch durchzuführen, wodurch sich der Aufwand und die Fehleranfälligkeit bei der Implementierung reduzieren. Ein Teil eines MMLLS lässt sich direkt generieren, z. B. die Definitionsdatei für einen Navigationsbaum. Andere Komponenten werden indirekt generiert, d. h. es werden Zwischendefinitionen in einer definierten Sprache in eine Datei geschrieben. Diese Zwischendefinitionen werden dann von speziellen Generatoren verwendet, um die Ziel-Komponenten zu erzeugen, wie z. B. verschiedene Aufgabentypen. Durch die Verwendung dieser Zwischensprache wird es möglich, verschiedene Realisierungen einer Multimedia-Komponente für unterschiedliche Entwicklungsumgebungen (z. B. Java und Macromedia Director) aus einer Spezifikationsdatei zu generieren. Hierdurch wird erreicht, dass man von einer bestimmten Entwicklungsumgebung unabhängig wird, so dass es bei einem Wechsel der Zielplattform nicht notwendig ist, alle Definitionsdateien neu zu schreiben, sondern lediglich ein Generator für die entsprechende Umgebung entwickelt werden muss.

Das Gesamtsystem besteht daher sowohl aus dem Erfassungs- und Verwaltungssystem als auch den Generatoren für die Transformationen in das jeweilige Entwicklungssystem.

Für die verschiedenen Entwicklungssysteme lassen sich generische Standard-Komponenten entwickeln, die für die Erstellung unterschiedlicher MMLLS verwendet werden können.

Die Konzepte, Strukturen und Komponenten von MMLLS sind sich über Fachbereichsgrenzen hinweg ähnlich, so dass die Komponenten, die entwickelt und für die Generierung eingesetzt werden, in der Regel fachbereichsunabhängig eingesetzt werden können. Hierdurch wird ein hoher Grad an Wiederverwendbarkeit erreicht. Die vorgestellte Methode wurde sowohl für die Generierung eines MMLLS für den Bereich Software-Technik als auch für den Bereich Dermatologie erfolgreich angewendet.

Diese Methode ermöglicht eine parallele Eingabe und Entwicklung durch mehrere Benutzer, weil das Erfassungs- und Verwaltungssystem als verteilte Client/Server-Anwendung realisiert werden kann.

Spezielle Informationen für Implementierer, Layout-Designer, Sprecher usw. können individuell aufbereitet exportiert werden. Konfigurations- und Prozess-Management können integriert werden. Die Inhalte können in mehreren Sprachen eingegeben werden, so dass die Entwicklung mehrsprachiger MMLLS, bzw. die Entwicklung eines MMLLS in verschiedenen Sprachen unterstützt wird.

Im Folgenden werden die einzelnen Schritte genauer betrachtet und an einem Beispiel verdeutlicht.

2.2 Modellierung des Fachkonzeptes des Verwaltungs- und Erfassungssystems für das MMLLS

Strukturmodellierung. Bei MMLLS ist eine horizontale und vertikale Strukturierung des Inhalts üblich. Die vertikale Dimension entspricht der fachsystematischen Gliederung wie die Unterteilung eines Fachbuches in Kapitel und Unterkapitel oder der Ontologie des jeweiligen Fachbereichs. Die horizontale Dimension entspricht der Binnengliederung. Diese gibt an, welche Informationsbausteine für jede Kategorie der fachsystematischen Gliederung angeboten werden sollen. Das OOA-Modell des Kerns des MMLLS ist in Abb. 2 dargestellt. Die fachsystematische Strukturierung kann aus der sukzessiven Aggregation von Themenkomplex, Themengruppe und Lernobjekt bestehen. Auf Ebene der Kategorien Themenkomplex und Themengruppe sollen Fallstudien möglich sein. Zu jedem Lernobjekt sollen Synonyme, Notationen, Beispiele, eine Einführung, Definitionen, Literaturhinweise, Aufgaben und Lernziele angeboten werden. Dieses Modell ist das Kernmodell des MMLLS und Ausgangspunkt für den weiteren Entwicklungsprozess. Die Kardinalitäten in dem Modell berücksichtigen, dass bei der späteren Eingabe der MMLLS-Inhalte in das generierte Erfassungssystem nicht die gesamte Struktur auf einmal eingeben werden muss. So ist es beispielsweise möglich zunächst Themenkomplexe ohne Themengruppen zu erfassen und diese erst später hinzuzufügen (Kardinalität 0..*). Im entwickelten System sollten sinnvollerweise jedoch zu jedem Themenkomplex auch Themengruppen existieren (entspräche Kardinalität 1..*).

Identifizierung der generierbaren Komponenten und Spezifikation der Kernklassenattribute. Das Kernmodell ist definiert. Über diesen Kern hinaus muss definiert werden, welche zusätzliche Information benötigt wird, um über die inhaltliche Definition hinaus, die multimediale Präsentation zu spezifizieren. Dieses bedeutet, dass zusätzliche Attribute eingefügt bzw. neue Klassen hinzugefügt werden müssen. Wie die Umsetzung in die Präsentation erfolgen soll, kann beispielsweise im Vorfeld anhand von Skizzen oder Storyboards festgelegt werden. Für Literaturhinweise sollen Autor, Titel, etc. angeboten werden. Es ist notwendig zu entscheiden, wie die Zielkomponenten des MMLLS erzeugt werden sollen. Die Attribute hängen von dieser Entscheidung ab. Hierbei gibt es fünf Möglichkeiten:

- **Direkte Generierung** aus den gespeicherten Informationen. Hierfür ist in dem Erfassungssystem eine detaillierte, vollständige und systematische Definition er-

forderlich. Die Komponente wird direkt im Erfassungs- und Verwaltungssystem festgelegt.

- **Indirekte Generierung** aus den gespeicherten Informationen. Hierfür ist in dem Erfassungssystem eine detaillierte und systematische Definition erforderlich und Exportinformationen müssen zur Verfügung gestellt werden. Das Erfassungssystem definiert lediglich den Inhalt. Das Layout wird über den Generator definiert.
- **Halbautomatische Erstellung** der MMLLS-Komponenten aus den gespeicherten Informationen ermöglicht die Generierung eines Teils der Komponente, die dann aber noch manuell weiter bearbeitet werden muss.
- **Manuelle Erstellung** der MMLLS-Komponenten mittels einer genauen strukturierten informalen internen Beschreibung. Anhand dieser Beschreibung lässt sich die Komponente manuell genau entwickeln.
- Die **ungenaue Beschreibung** der zu erstellenden MMLLS-Komponente erfordert weitere externe Informationen, anhand derer die Komponente entwickelt werden kann.

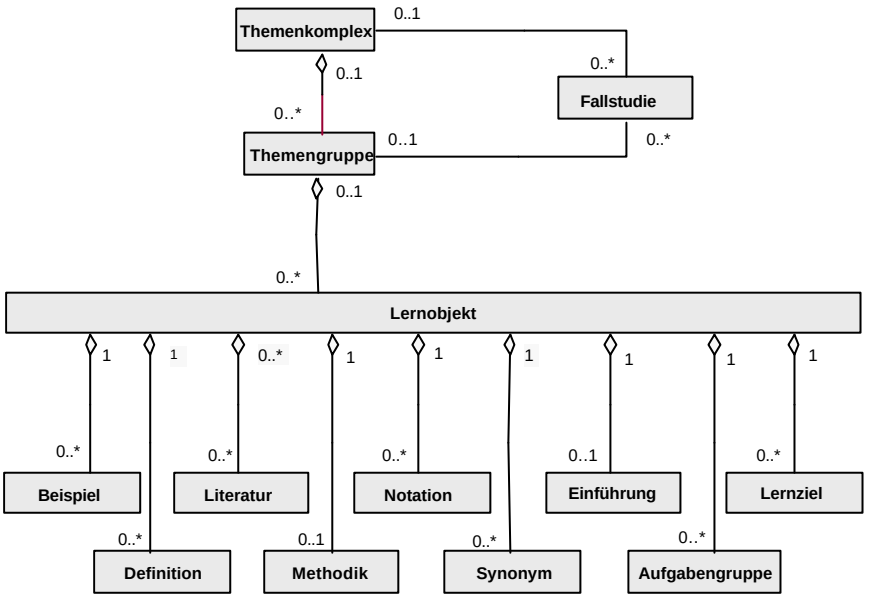

Abb. 2. OO-Modell der Kernstruktur des Fachkonzeptes der Inhaltsstruktur

Ein Beispiel für die direkte Generierung ist die Generierung von Literaturlisten für jedes Lernobjekt. Die Aufgaben werden indirekt generiert, indem sie in eine spezielle Aufgabendefinitionssprache in eine Datei exportiert werden (s. [17]). Diese Dateien werden dann von Generatoren verwendet, um die Aufgaben gemäß ihrer Spezifikation zu generieren. Grafische Notationen lassen sich textuell beschreiben, die Umsetzung in eine Grafik erfolgt aber manuell. Für die Anzeige der Notationen und Beispiele lassen sich die Rahmen generieren, in die die Grafiken noch manuell eingebun-

den werden müssen. Die Einführungsanimationen lassen sich inhaltlich gut, von Ablauf her aber nur ungenau beschreiben. Für eine umfassende Definition ist daher ein externes Drehbuch erforderlich.

Die Entscheidung, ob bzw. auf welche Weise eine Komponente generierbar ist, hängt von ihrem Charakter und der Art und Weise der Informationsrepräsentation ab.

Wenn eine Komponente häufig oder systematisch benutzt wird, signalisiert dies, dass es sinnvoll erscheint, diese Transformation von einem Generator durchführen zu lassen. Eine Animation zur Einführung in ein Thema ist so einzigartig, dass ein generischer Weg sehr kompliziert ist. Diese Transformation kann derzeit noch nicht vollständig automatisiert werden, da es keinen definierten Prozess für diese Transformation von einer Definition über Entwurf bis zur fertigen Animation gibt. Teilaspekte von Animationen lassen sich spezifizieren; so liegen dem Autorensystem FMAD [18] Modelle zugrunde, die die Synchronisation multimedialer Ausgaben und Interaktionen ermöglichen.

Bei diesem Schritt ist es noch nicht notwendig, das komplette Layout-Design des MMLLS festzulegen, aber eine ungefähre Vorstellung ist erforderlich, um über die Generierbarkeit der Komponente und die dafür erforderlichen Attribute entscheiden zu können.

Die Hauptattribute des Kernmodells sind an dieser Stelle definiert und können im weiteren Verlauf noch inkrementell geändert werden. Die Operationen »generiere()« und »exportiere()« müssen noch den jeweiligen Klassen hinzugefügt werden.

2.3 Objektorientierte Modellierung unterschiedlicher Aufgabentypen

Standard-Aufgabentypen sind über vielfältige Anwendungsgebiete hinweg von ihrem Aufbau her unverändert. Diese Aufgabentypen lassen sich daher allgemein modellieren und wiederverwenden. Abb. 3 zeigt die Modellierung der Hierarchie der entsprechenden Klassen. Zunächst werden generierbare und nicht generierbare Aufgaben unterschieden. Aufgaben, die von ihrer Bedienungsart her sehr individuell sind und sich daher nicht allgemein abstrahieren lassen, können lediglich textuell oder grafisch beschrieben werden. Da sie nicht formal spezifiziert werden können, sind sie nicht generierbar. Es ist somit eine manuelle Erstellung dieser Aufgaben notwendig. Die Modellierung nicht generierbarer Aufgaben dient daher ausschließlich der Beschreibung und Verwaltung dieser Aufgaben des MMLLS.

Generierbare Aufgabentypen. Die generierbaren Aufgaben verfügen über gemeinsame Attribute, die für das jeweilige MMLLS vorbelegt werden können (s. Abb. 4). Zu jeder generierbaren Aufgabe lässt sich eine Rückmeldungsgruppe bestimmen, die die Reaktionen auf korrekte und falsche Antworten definiert (s. Abb. 5). Über die Reihenfolge der Rückmeldungen lassen sich die Rückmeldungen je nach Anzahl der unternommenen Lösungsversuche individualisieren.

Die generierbaren Aufgaben werden in die **assoziativen Aufgaben** und die **Auswahlaufgaben** unterteilt.

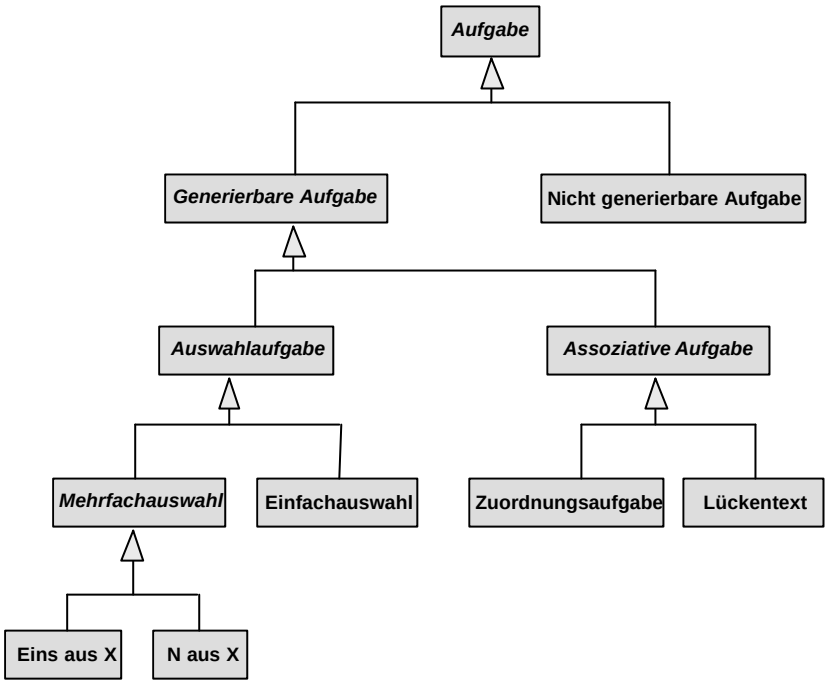

Abb. 3. Klassenhierarchie der unterschiedlichen Aufgabentypen

Auswahlaufgaben. Die Auswahlaufgaben sind dadurch gekennzeichnet, dass die angebotenen Lösungsalternativen entweder richtig oder falsch sein können. Bei der **Einfachauswahlaufgabe** werden die Lösungsalternativen (EFA-Lösungsalternative) voneinander unabhängig zeitlich nach einander angeboten, wobei zu jeder Alternative der Aufgabe auch noch eine individuelle Fragestellung vorliegen kann. Bei den **Mehrfachauswahlaufgaben** werden gleichzeitig mehrere Lösungsalternativen (MFA-Lösungsalternative) angeboten. Die getroffene Auswahl der richtigen Alternativen muss als abgeschlossen bestätigt werden. Hierbei sind **»1 aus X« -** und **»N aus X« – Aufgaben** zu unterscheiden. Bei den ersteren ist lediglich das Auswählen einer Alternative als richtig möglich, wohingegen bei letzteren mehrere oder keine Alternative als richtig ausgewählt werden können. Abb. 6 zeigt das Klassendiagramm der Auswahlaufgaben.

Assoziative Aufgaben. Bei den assoziativen Aufgaben werden Elemente einer Menge definierter Wahlobjekte Elementen einer Menge von Zielobjekten zugeordnet. Dieses ist allgemeiner als die auf richtig und falsch beschränkten Zielobjekte (Lösungsalternativen) der Auswahlaufgaben. Zu den assoziativen Aufgaben zählen die Aufgabentypen **Lückentext** und **Zuordnungsaufgaben** (s. Abb. 7). Bei den Zuordnungsaufgaben werden häufig auch Grafiken verwendet, wohingegen bei Lückentext-

aufgaben die Wahl- und Zielmengen i. d. R. aus Texten bestehen. Es werden Zuordnungen vorgenommen. Bei der Lückentextaufgabe werden die angebotenen Wörter in die Textlücken eingefügt. Bei den Zuordnungsaufgaben können diese Objekte aus Grafik und Text bestehen.

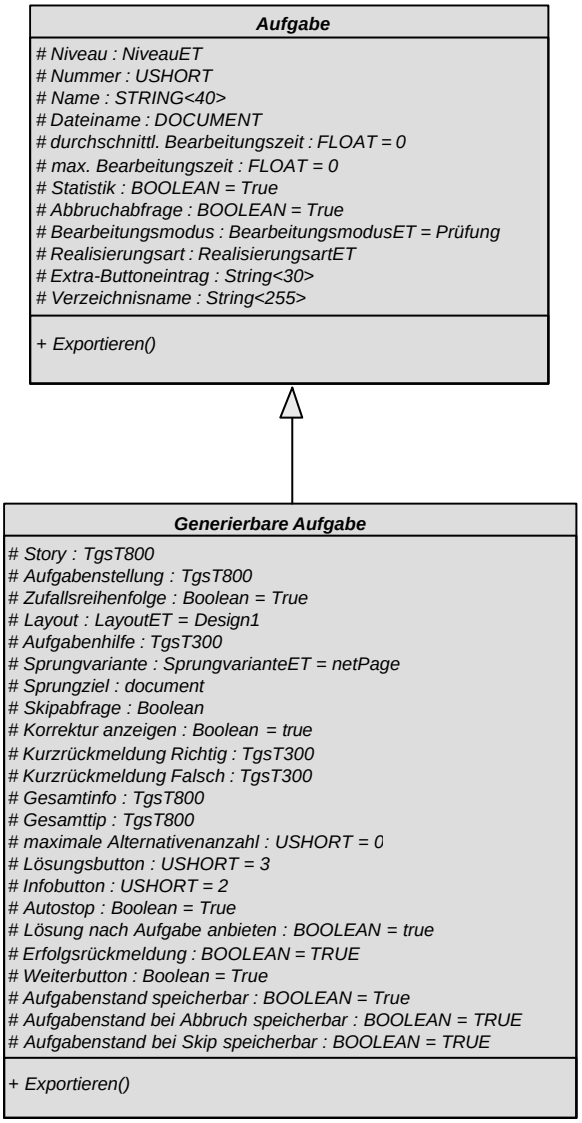

Abb. 4. Die Attribute der generierbaren Aufgabentypen

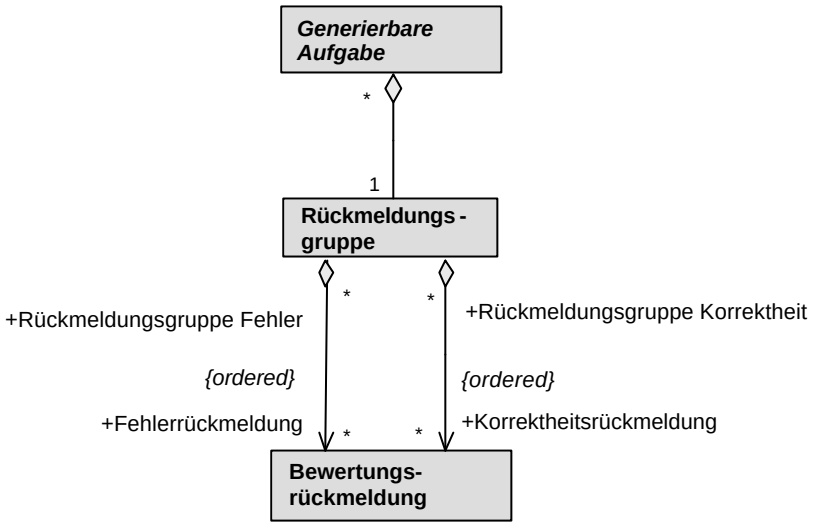

Abb. 5. Die Rückmeldungen generierbarer Aufgaben

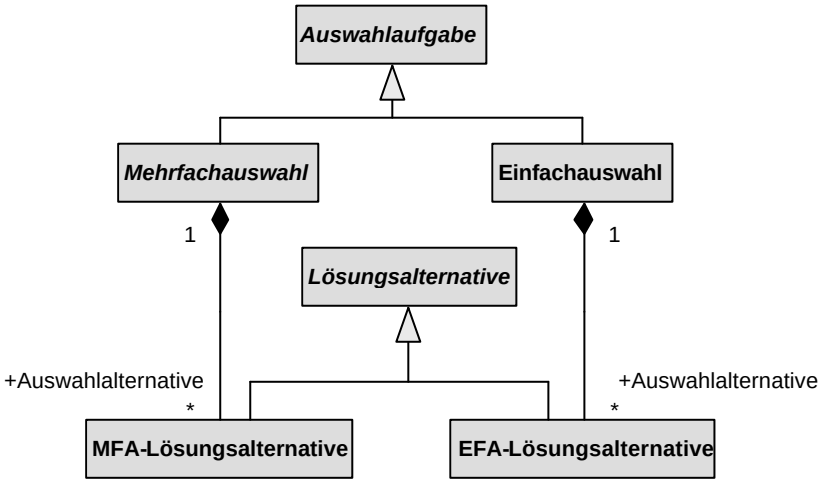

Abb. 6. Auswahlaufgaben und ihre Lösungsalternativen

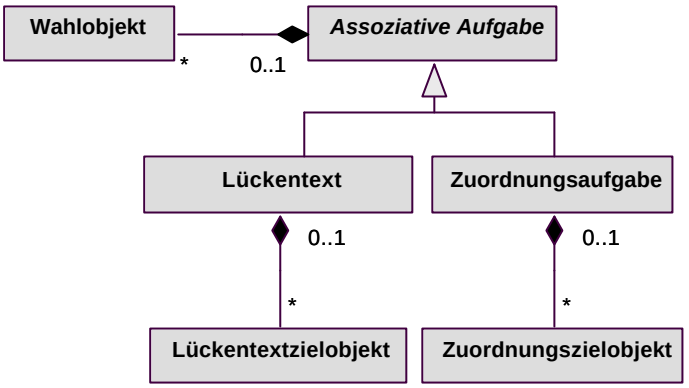

Abb. 7. Assoziative Aufgaben mit Wahl- und Zielobjekten

2.4 Hinzufügen der Systemkomponenten zum Modell

Zu der genauen Beschreibung der Inhaltsstruktur und der Transformation in eine multimediale Form werden konzeptionelle Komponenten hinzugefügt. Diese sind typischerweise

- Navigation,
- Datenbank und Statistik,
- geführte Tour,
- Tutor
- etc.

Der Entwickler muss entscheiden, welche dieser Komponenten in das MMLLS integriert werden sollen, ihre Aufgaben definieren und festlegen, welche Informationen hierfür notwendig sind. Zur Unterstützung dieser Komponenten, die zur Erledigung ihrer Aufgaben auf die Kernklassen zugreifen müssen, werden dem Modell die entsprechenden Klassen hinzugefügt. Auch diese Klassen verfügen über Operationen zur Exportierung und Generierung, um den System-Komponenten die notwendigen Informationen zur Verfügung zu stellen.

2.5 Hinzufügen spezieller Fähigkeiten und Funktionalitäten

Neben der Unterstützung der Komponenten, die das MMLLS betreffen, können weitere Klassen dem Modell hinzugefügt werden, die die Arbeit mit dem Verwaltungs- und Erfassungssystem sowie seine Implementierung unterstützen. In der realisierten Version wurden Elemente zur Versionierung und Prozesskontrolle ergänzt. Darüber hinaus ist die zielgruppenspezifische Exportierung von Informationen möglich oder

auch die Generierung einer Übersicht über die eingegebenen Inhalte. In der Abb. 8 sind alle beteiligten Subsysteme mit ihren Abhängigkeiten dargestellt.

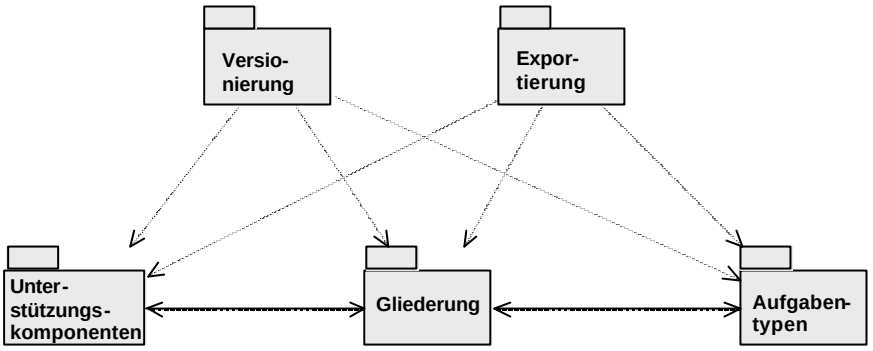

Abb. 8. Die Subsysteme des Erfassungs- und Verwaltungssystems

2.6 Generierung des Inhaltserfassungs- und Verwaltungssystems

Dieses OOD-Modell des Gesamt-Fachkonzepts verwendet der JANUS-Generator, um das Erfassungs- und Verwaltungssystem für das MMLLS zu generieren. Das generierte System ist in Abb. 9 abgebildet. Dieses System kann auch als Client/Server-Anwendung generiert werden. Die Autoren können unmittelbar nach der Generierung mit der Eingabe der Inhalte für das MMLLS in das Erfassungs- und Verwaltungssystem beginnen. Die Exportierungs- und Generierungsfunktionen müssen noch implementiert werden. Dieses kann von Software-Ingenieuren parallel oder später geschehen. Die weiteren Funktionalitäten wie die Versions- und Prozesskontrolle müssen auch noch implementiert werden. Da dieses jedoch Standardprobleme sind, können bereits existierende Lösungen wiederverwendet werden.

2.7 Generierung des MMLLS

Die Abb. 10 zeigt das generierte MMLLS. Im Folgenden wird beschrieben, wie sich die einzelnen Komponenten dieses Systems direkt oder indirekt generieren lassen.

Direkte Generierung. Direktes Generieren bedeutet, dass die Information aus dem Erfassungs- und Verwaltungssystem direkt in die endgültige Form transformiert wird, die im MMLLS benötigt wird. Eine weitere Transformation in der MMLLS-Komponente ist noch möglich, wie es z. B. bei der Transformation des Inhaltsverzeichnisses in die Baumansicht der Fall ist. In dem MMLLS-Beispiel zur Objektorientierung, das in der Abb. 10 dargestellt ist, sind die folgenden Komponenten direkt generiert worden:

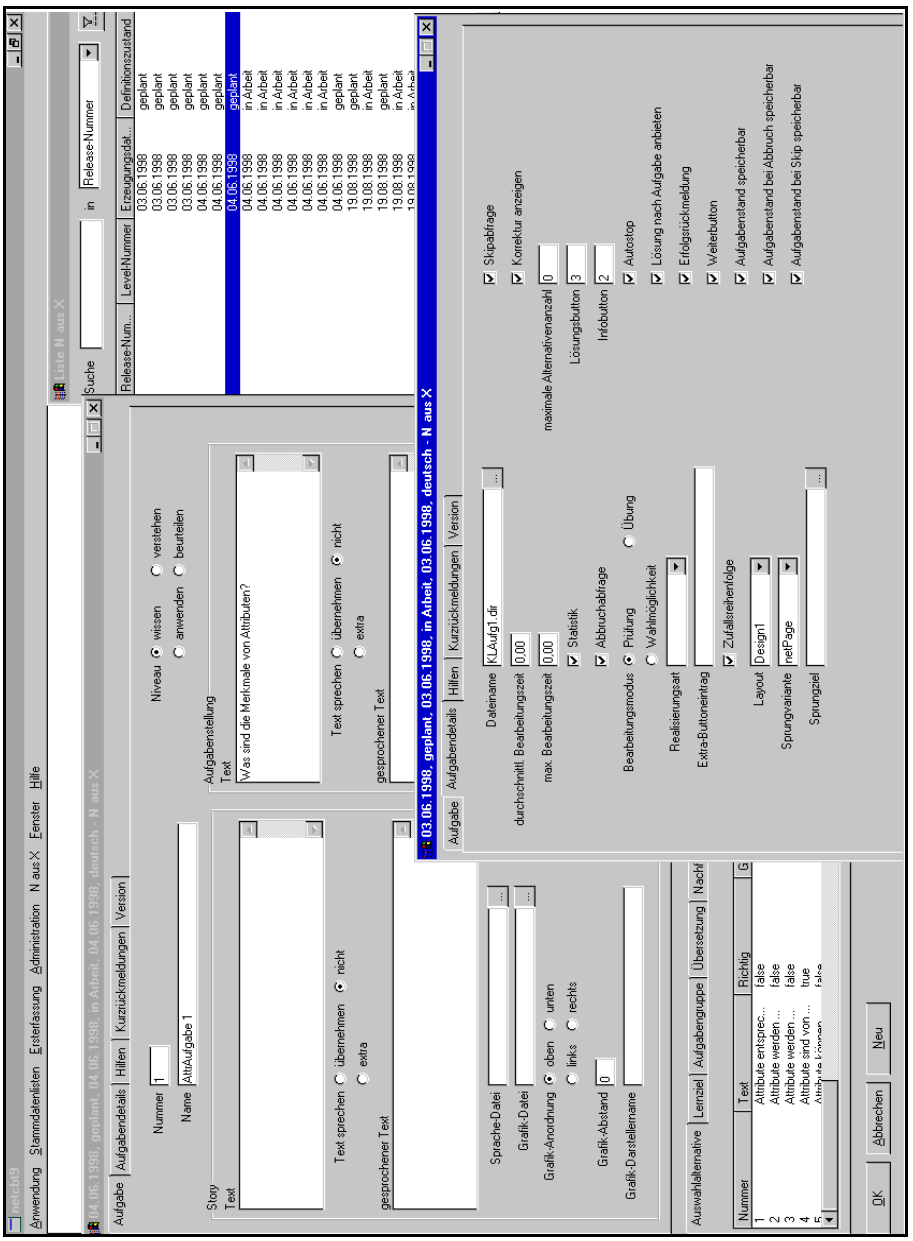

Abb. 9. Das mit dem Janus-Generator erzeugte Erfassungs- und Verwaltungssystem

- Die systematische Dateistruktur des MMLLS
- Das Rahmensystem *(framesets)* in HTML

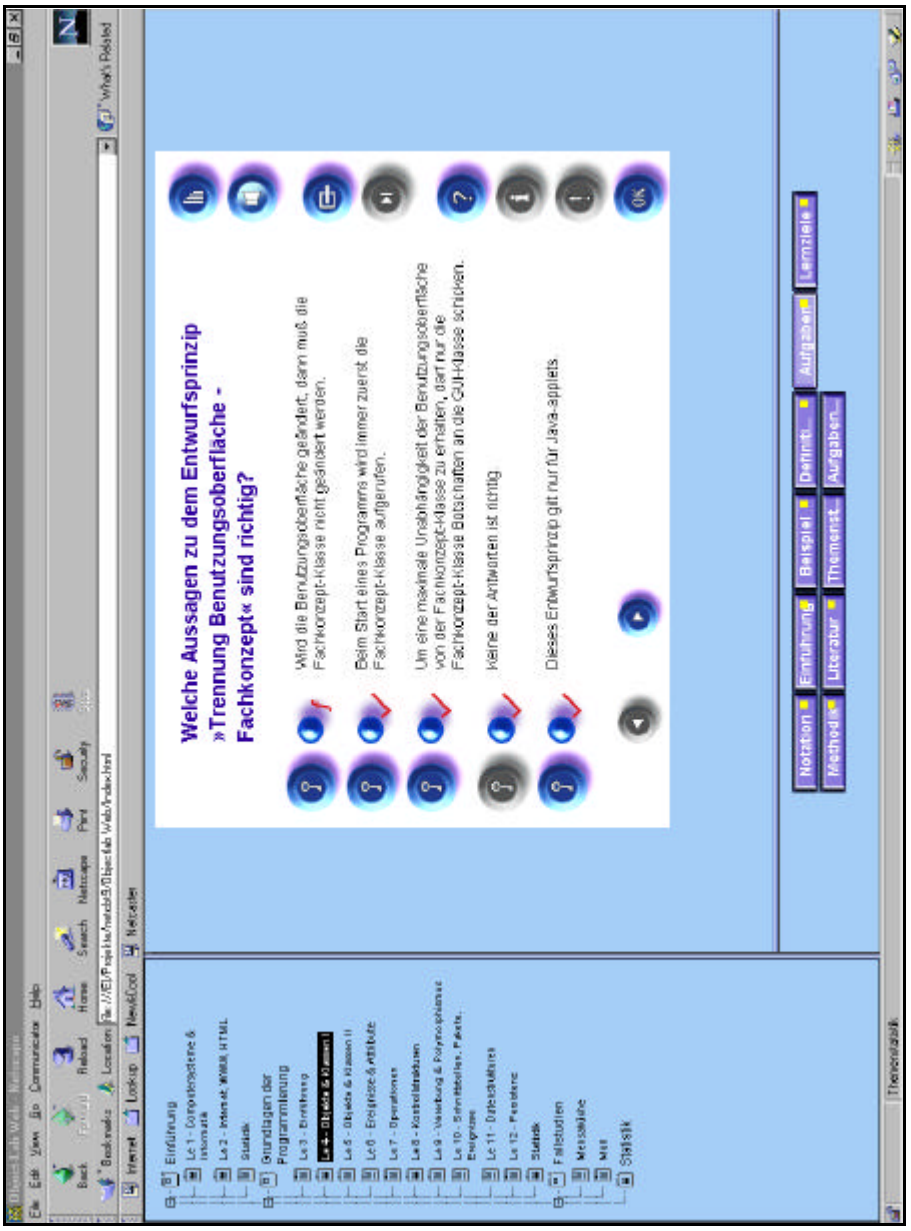

Abb. 10. Das generierte MMLLS »ObjectLab Web«

- Die Definitionsdatei für die Inhaltsbaumansicht in einer proprietären Sprache für das TOC-Applet *(Table of Contents)*

- Die Definitionsdateien für die Auswahlleiste in HTML. Die Leiste ist als generischer Film mit dem Director realisiert worden, der seine Informationen als Shockwave-Film aus der HTML-Datei erhält.
- Die Definition der Statistik-Ansichten. Die Ansichten wurden ebenfalls mit dem Macromedia Director realisiert.
- Das Literaturverzeichnis ist in HTML generiert worden.
- Synonyme, Definitionen und Lernzielkataloge sind ebenfalls in HTML generiert worden.

Die HTML-Seiten, in die die Einführungsanimationen integriert werden, sind auch in HTML generiert worden.

Indirekte Generierung. Indirekte Generierung bedeutet, dass eine Spezifikationsdatei generiert wird, die dann von einem Generator benutzt wird, um die eigentliche MMLLS-Komponente zu generieren. Die Spezifikationsdatei wird von dem endgültigen MMLLS nicht verwendet. Sie kann von verschiedenen Generatoren unabhängig voneinander für unterschiedliche Zielumgebungen benutzt werden. Die Generatoren filtern aus der Datei die relevanten Informationen heraus und generieren die Ziel-Komponente. Alle Komponenten, die man direkt generieren kann, lassen sich auch indirekt generieren. Allerdings benötigt man hierfür ein Exportformat und einen entsprechenden Generator, um die MMLLS-Komponente zu erzeugen. Daher ist die indirekte Generierung aufwendiger als die direkte. Indirekte Generierung kann notwendig sein. In dem gewählten Beispiel-MMLLS sind die folgenden Komponenten indirekt generiert worden:

- Die Generierung des Datenbankschemas und der Tabellen. Die Datenbanken werden für die allgemeinen Informationen über das MMLLS und die benutzerspezifischen Informationen benötigt.
- Die Aufgaben werden indirekt generiert. Die Aufgabeninformationen werden in das EDF (Exercise Definition File)-Format geschrieben, und dienen den Generatoren als Eingabe.

In [14] wird die indirekte Aufgabengenerierung mit dem ExerGen-Konzept (Exercise Generator) und der hierzu entwickelten Exercise Definition Language zur Aufgabendefinition ausführlich dargestellt. Die Abb. 11 zeigt das Vorgehen bei der Aufgabengenerierung: Der Aufgabengenerator liest die Aufgabendefinition aus einer EDF-Datei ein und erzeugt daraus die lauffähige Aufgabe. Eventuell müssen einzelne Grafiken bzw. Bereiche noch manuell angeordnet werden. Bisher sind Generatoren für Java [10] und den Macromedia Director [5] realisiert worden für die Aufgabentypen:

- "1 aus X"-Aufgabe
- "N aus X"-Aufgabe
- "Ja-Nein"-Aufgabe
- "Zuordnung gleichzeitig"-Aufgabe
- "Zuordnung nacheinander"-Aufgabe
- "Reihenfolge"-Aufgabe

In die Aufgaben lassen sich Texte, Grafiken, Ton und Piktogramme einbinden und kombinieren.

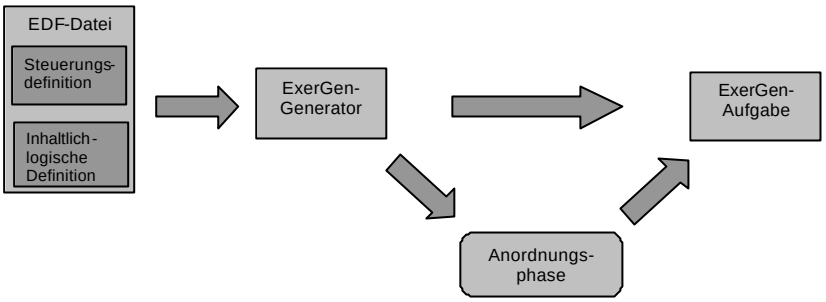

Abb. 11. Die Aufgabengenerierung mit dem ExerGen-Konzept

2.8 Anmerkungen zur Methode

Die Verwendung von HTML, wie in dem Beispiel, ist nur eine Möglichkeit. Die Methode ist unabhängig von HTML und lässt sich mit beliebigen Beschreibungsformaten anwenden.

Bisher werden Animationen nur durch den Weg der ungenauen Beschreibung und manuellen Erstellung unterstützt. Dass die Generierung von Animationen bisher nicht erfolgt, ist ein Spezifikationsproblem. Mit einem entsprechenden formalen Spezifikationsansatz lassen sich auch Animationen und ihre automatische Erstellung direkt in die Methode integrieren.

Kommerzielle Produkte wie beispielsweise Pathware von der Fa. Macromedia bieten eine komfortable Verwaltung von Online-Kursen an. Die eigentliche Erstellung der Inhaltsobjekte muss jedoch manuell mit Autorenwerkzeugen vorgenommen werden. Der in diesem Artikel vorgestellte Ansatz ist weiter gehender, da auch die Erstellung dieser Inhaltsobjekte (zumindest teilweise) automatisiert wird, wie beispielsweise bei der Aufgabenschnittstellengenerierung. Die generierten Aufgaben lassen sich jedoch auch in so eine kommerzielle Online-Kursverwaltung als Kursinhalt integrieren.

3 Mit der Methode realisierte Projekte

Der exemplarisch dargestellte Prototyp ist realisiert worden. Der Bereich der Aufgaben wurde hieraus extrahiert und über 100 Wissens- und Verstehensaufgaben sind begleitend zu dem »Lehrbuch Grundlagen der Informatik« [2] erstellt worden. Sie ergänzen dieses Lehrbuch. Bei dieser umfangreichen Erstellung hat sich die vermutete Effizienz bestätigt. Die Arbeit reduzierte sich im wesentlichen auf die inhaltliche Konzeption der Aufgaben. Insbesondere ist hervorzuheben, dass bei den Auswahlaufgaben zu jeder Lösungsalternative eine individuelle Information (Tipp) und eine individuelle Begründung angeboten werden können. Der Lernende kann sich so gezielt über die einzelnen Alternativen informieren. Zur Unterstützung der Dokumentation und damit der inhaltlichen Korrektheit wurden die Aufgabeninhalte übersichtlich in

HTML exportiert (s. Abb. 12). Die Implementierung der Aufgaben ließ sich wie erwartet in unerheblich geringer Zeit realisieren.

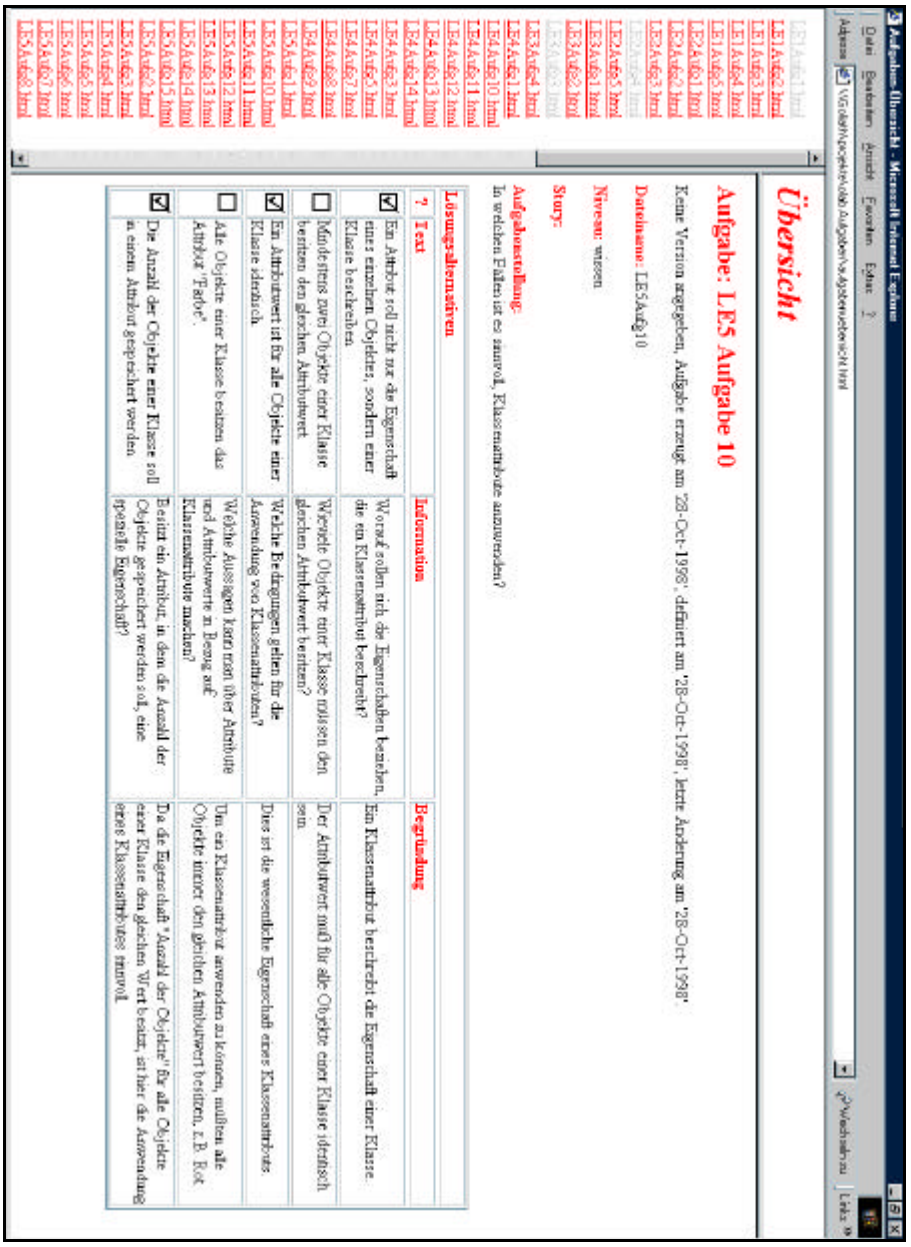

Abb. 12. Übersicht über die Aufgabenspezifikationen

Desweiteren ist die Methode zur Erstellung einer multimedialen Aufgabensammlung zum Thema »Dermatologie« in Zusammenarbeit mit der Universitätsklinik St. Josefs-Hospital in Bochum angewendet worden. Bei den Aufgaben handelt es sich um Aufgaben des sogenannten Medizinertests. Zunächst wurde ein Modell der verwendeten Aufgaben entwickelt, wodurch das Erfassungssystem maßgeschneidert wurde für die konkrete Anwendung. Bei diesen Aufgaben handelt es sich um drei Spezialisierungen einer »1 aus X«-Aufgabe. Es zeigte sich, dass die am Projekt beteiligten Mediziner, obwohl in der Erstellung von MMLLS sie unerfahren waren, direkt problemlos mit den generierten Erfassungssystem arbeiten konnten. Die Exportschnittstellen wurden parallel zur Inhaltseingabe entwickelt. Obwohl die Aufgabentypen bei der Erfassung auf die speziellen Medizinertest-Aufgaben abgestimmt waren, ließ sich der allgemeinere Aufgabengenerator problemlos verwenden.

Zur Verdeutlichung der **Maßschneiderungseffektes** lässt sich der Umfang der allgemeinen Aufgabendefinition, die die Möglichkeit des Generators widerspiegelt, mit dem Umfang der maßgeschneiderten Definition vergleichen:

Für eine allgemeine »1 aus X«-Aufgabe können 67 Attribute definiert werden. Jede Lösungsalternative hierzu kann 57 weitere Attribute enthalten. Eine solche Aufgabe mit fünf Lösungsalternativen kann daher durch bis zu 352 Attribute definiert werden. Für 100 Aufgaben sind daher bis zu 35200 Attributspezifikationen möglich. (Diese Spezifikationen sind i. Allg. nicht vollständig nötig, da es Vorbelegungen gibt). Diese Anzahl zeigt, dass der Einsatz eines Erfassungs- und Verwaltungssystems bei so anspruchsvollen Aufgabenrealisierungen sinnvoll bzw. sogar notwendig ist. Durch das Maßschneidern innerhalb des Medizinerprojektes ließ sich die Attributanzahl für die Aufgaben auf neun und für die Lösungsalternativen auf sieben reduzieren, so dass pro Aufgabe nur noch 44 Attribute zu spezifizieren waren. Die weiteren Attribute waren konstant, so dass sie nicht individuell für jede Aufgabe definiert werden mussten. Dieses Projekt hat bestätigt, dass es mit dieser Methode leicht möglich ist, Fachautoren ohne MMLLS-Erfahrungen in die Lage zu versetzen, ihr Wissen direkt in die MMLLS-Entwicklung einzubringen und Fehler durch die Umsetzung zu vermeiden.

4 Ausblick

Mit der Methode zur Erstellung multimedialer Lehr- und Lernsysteme sollen weitere multimediale Lehr- und Lehrsysteme unterschiedlicher Fachbereichte entwickelt werden. Sowohl das Gesamtsystem als auch die Aufgabengeneratoren sollen weiterentwickelt werden.

Vorhandenes soll hierbei wiederverwendet und ergänzt werden, wie beispielsweise schon durch die Erweiterung um die drei Medizinertest-Aufgabentypen geschehen. Es ist zu erwarten, dass mit zunehmender Realisierungsanzahl der Grad der Wiederverwendung bei der MMLLS-Entwicklung steigt und Neu- bzw. Weiterentwicklungen reduziert werden.

5 Literatur

1. Balzert H., Weidauer C.: Multimedia-Systeme – ein neues Anwendungsgebiet für die Software-Technik, in Softwaretechnik-Trends, Band 18, Heft 4, S. 4-9., Gesellschaft für Informatik, November 1998
2. Balzert H.: Grundlagen der Informatik, Spektrum Akademischer Verlag, Heidelberg, Berlin 1999
3. Balzert H., Balzert H.: Object-lab light – Interaktives multimediales Lernsystem zur objektorientierten Softwareentwicklung, Lehrstuhl für Software-Technik, Ruhr-Universität Bochum, 1997
4. Balzert H., Hofmann F., Kruschinski V. und Niemann C.: Vom Programmieren zum Generieren - Auf dem Weg zur automatisierten Anwendungsentwicklung, in Computer-Aided Design of User-Interfaces (CADUI'96, Namur, 5-7 June 1996)
5. Eismann M.: Generierung verschiedener Aufgabentypen als multimediale Realisierung in Java, Studienarbeit am Lehrstuhl für Software-Technik an der Ruhr-Universität Bochum
6. Garzotto F., Paolini P., Schwabe D.: HDM-A Model Based Approach to Hypermedia Application Design, in ACM Trans. Inf. Syst., Jan. 1993, Volume 11, Number 1
7. Hofmann F.: Grafische Benutzungsoberflächen – Generierung aus OOA-Modellen, Spektrum Akademischer Verlag Berlin, Heidelberg 1998
8. Isakowitz T., Stohr E. A., Balasubramanian P., RMM: A Methology for Stuctured Hypermedia Design, in Communication of the ACM, August 1995-Volume 38, Number 8
9. Kruschinski V.: Layoutgestaltung grafischer Benutzungsoberflächen – Generierung aus OOA-Modellen, Spektrum Akademischer Verlag Berlin, Heidelberg 1999
10. Mittmann L.: Generierung verschiedener Aufgabentypen als multimediale Realisierung im Director, Diplomarbeit 99/02 am Lehrstuhl für Software-Technik an der Ruhr-Universität Bochum
11. Morris S. J.: Media Transformations for the Representation and Communication of Multimedia Production Activities, in Proc. of the IFIP Working Group Conference on Designing Effective and Usable Multimedia Systems (Stuttgart, Germany, Sept. 1998), Kluwer Academic Publishers, pp. 73-87.
12. Pauen P., Voss J., Six H.-W.: Modelling Hypermedia Applications with HyDev, in: Designing effective and usable multimedia systems, Kluver Academic Publishers, 1998
13. Puppe F.: Intelligente Tutorsysteme, in: Informatik-Spektrum 15, S. 195-207, Springer-Verlag 1992
14. Schäfer W. , Wadsack J.: Zusammenfassung, Teil IV, S. 193 f. in : Balzert H., Behle A., Kelter U., Nagl M., Pauen P., Schäfer W., Six H.-W., Voss J., Wadsack J., Weidauer C., Westfechtel B., Studie über Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme der Forschergruppe SofTec NRW, September 1999
15. Schanda F.: Computer-Lernprogramme, Beltz-Verlag, 1995
16. Schwabe D., Rossi G., Barbosa S.: Systematic Hypermedia Application Design with OOHDM, in: Conference Proceedings of ACM Hypertext 1996, ACM Conference Proceedings 1996, pp. 116-128
17. Weidauer C.: Generierung multimedialer Aufgaben mit ExerGen, in: Euler B., Kreutz R., Spitzer K. (Hrsg.) Multimedia in der Medizin, Proceedings zum Workshop, Aachen 27.-28. Oktober 1999
18. Boles D., FMAD - Ein objektorientiertes Autorensystem für interaktive multimediale Anwendungen, in: Proceedings GI-Fachtagung ´95, Braunschweig Oktober 1995, S. 24-34

Anmerkungen

[1] JANUS ist ein Produkt der oTRIs Software AG, Dortmund (www.otris.de).