Multimedia-Systeme – ein neues Anwendungsgebiet für die Software-Technik

Helmut Balzert
Lehrstuhl für Software-Technik
Ruhr-Universität Bochum
Universitätsstr.150, D-44780 Bochum, Germany
Tel: +49 234-700-6880
E-mail: hb@swt.ruhr-uni-bochum.de

Christian Weidauer
Lehrstuhl für Software-Technik
Ruhr-Universität Bochum
Universitätsstr.150, D-44780 Bochum, Germany
Tel: +49 234-700-6794
E-mail: weidauer@swt.ruhr-uni-bochum.de

Charakteristika von Multimedia-Systemen und -Entwicklungen

Multimedia-Systeme haben in den letzten Jahren eine zunehmende Bedeutung erlangt. Es entsteht eine neue Branche. Sie ist noch stärker als traditionelle Software-Entwicklungen – wie kaufmännische und technische Anwendungen – durch interdisziplinäre Teamarbeit geprägt. Grafiker, Pädagogen und Software-Ingenieure – oft ergänzt durch Audio- und Video-Spezialisten – erstellen Lehr- und Lernsysteme, Präsentations- und Informationssysteme, Unterhaltungs- und Spielsysteme /Müller 95, S. 256/. Das Ergebnis jeder Multimedia-Entwicklung ist ein Software-Produkt und damit ein neues Anwendungsgebiet für die Software-Technik.

Gegenüber traditionellen Anwendungssystemen der Software-Technik unterscheiden sich Multimedia-Systeme im Wesentlichen in folgenden Punkten:

- Andere Ziele
  Traditionelle Anwendungen haben meist das Ziel, Routineaufgaben zu erledigen, z.B. Aufträge abwickeln oder eine Produktion steuern. Multimedia-Systeme haben meist andere Ziele, z.B. die Vermittlung von Wissen bei Lehr- und Lernsystemen, das Wecken von Kaufinteresse bei Präsentationssystemen /Schifmann, Heinrich, Heinrich 97, S. 9/. Diese anderen Ziele erfordern andere Benutzungsoberflächen, da nicht mehr ein Benutzer über längere Zeit hinweg ein System benutzt.
- Andere Benutzer
  Bedingt durch die anderen Ziele werden Multimedia-Systeme auch von anderen Benutzern eingesetzt, z.B. Studenten nutzen Lehr- und Lernsysteme, Kaufinteressenten sehen sich eine Produktpräsentation an oder suchen in einem multimedialen Informationssystem.
- Es passiert mehr auf der Benutzungsoberfläche.
  Multimedia-Systeme nutzen natürlich Möglichkeiten wie Grafik, Animation, Film, Ton und Sprache, während diese Gestaltungsmöglichkeiten bei traditionellen Anwendungen keine oder nur eine untergeordnete Bedeutung besitzen.
- Es werden mehr Sinne angesprochen.
  Bedingt durch die anderen Ziele und die anderen Benutzer und durch die neuen technischen Möglichkeiten werden der Sehsinn intensiver beansprucht und der Hörsinn mit einbezogen.
- Jeder Benutzer verwendet das System nur einmal oder selten.
  Bedingt durch die anderen Ziele werden Multimedia-Systeme meist nicht regelmäßig durch einen Benutzer bedient, sondern oft nur einmal oder sporadisch benutzt. Dies hat insbesondere Auswirkungen auf die Gestaltung der Benutzungsoberfläche. Die Erlernbarkeit erhält eine höhere Priorität als die Bedienbarkeit.
- Das Anwendungsspektrum ist vielfältiger.
  Durch die technischen Multimedia-Möglichkeiten ergibt sich ein vielfältiges Anwendungsspektrum. Die bereits bekannten Multimedia-Anwendungen stellen sicher erst den Anfang des Spektrums dar.

Diese wenigen Punkte zeigen bereits deutlich, daß sich Multimedia-Systeme wesentlich von traditionellen Software-Anwendungen unterscheiden.

Betrachtet man aufgrund dieser Unterschiede eine Multimedia-Softwareentwicklung im Vergleich zu einer traditionellen Software-Entwicklung, dann sind folgende Gesichtspunkte wesentlich:

- Die Erstellung einer Multimedia-Anwendung ist hochgradig interdisziplinär.
  Obwohl auch traditionelle Anwendungen zunehmend von interdisziplinären Teams entwickelt werden, erfordert eine Multimedia-Anwendung doch wesentlich mehr Spezialisten, die zudem noch aus völlig verschiedenen Ausbildungsbereichen kommen.
- Eine Zeitsynchronisation ist erforderlich.
  Der Einsatz von Animation, Film, Ton und Sprache erfordert eine sorgfältige zeitliche Synchronisation, die durch Software-Werkzeuge geeignet unterstützt werden muß.
- Es gibt weniger Vorbilder.
  Da die Entwicklung von Multimedia-Systemen noch ein relativ neues Anwendungsgebiet der Software-Technik ist, gibt es für die Entwicklung noch wenig Vorbilder. Empirische Untersuchungen und veröffentlichte Erfahrungsberichte sind noch selten.
- Stärkeres Gewicht der Methodik und Didaktik

Viele Multimedia-Anwendungen – insbesondere Lehr- und Lernsysteme – erfordern eine ausgefeilte Methodik und Didaktik, um eine erfolgreiche Anwendung durch den Benutzer sicherzustellen.

- Oft einmalige Entwicklungen
  Durch das vielfältige Anwendungsspektrum sind manche Entwicklungen einmalig und eignen sich daher nicht als Vorbilder für weitere Entwicklungen. Methodische Erkenntnisse sind nur bedingt übertragbar, implementierte Teile nicht wiederverwendbar.
- Oft hochgradig innovative und kreative Entwicklungen
  Während der Innovationsgrad bei traditionellen Software-Entwicklungen meist gering ist, sind neuartige Multimedia-Anwendungen oft hochgradig innovativ. Ein hohes Maß an Kreativität ist erforderlich. Für diese Anwendungen müssen oft neue Wege in der Entwicklung gefunden und gegangen werden.
- Es gibt noch keine Standards.
  Für Multimedia-Entwicklungen gibt es noch keine Produktstandards, keine Prozeßstandards und keine allgemeingültigen *style-guides*. Da es bei Multimedia-Entwicklungen neue und zusätzliche Aufgaben zu erledigen gibt, ist es wissenswert, wie diese in ein Prozeßmodell integriert werden und inwieweit die Tätigkeiten voneinander abhängen bzw. parallelisierbar sind. Für die Gestaltung der Interaktion mit dem Benutzer gibt es teilweise noch keine Regeln. Beispielsweise ist unklar, wie eine Sprachausgabe ergonomisch zu gestalten ist: Sollen angezeigte Informationen noch einmal vorgelesen werden? Sollen alle gesprochenen Informationen wiederholbar sein? Wieviel gesprochene Sprache verträgt der Benutzer im Verhältnis zu angezeigten Informationen?
- Viele Hardware- und Software-Schnittstellen
  Charakteristisch für viele Multimedia-Systeme ist die Ansteuerung und Einbindung vieler Hardware- und Software-Schnittstellen, z.B. Video- und Audiosysteme.
- Oft Einsatz vieler Software-Werkzeuge
  Um die einzelnen Teile einer Multimedia-Anwendung zu erstellen, sind oft viele verschiedene Werkzeuge nötig, z.B. Grafikeditor, MIDI-System, Video-Schnittsystem usw. Dementsprechend entstehen auch eine Vielzahl von Dateien, die alle verwaltet und zu Konfigurationen zusammengefaßt werden müssen.
- Schwierig zu testen
  Bedingt durch die vielfältigen Interaktionsmöglichkeiten mit den verschiedenen Medien und der oft zeitbehafteten Präsentation ist es schwierig und zeitaufwendig, einen ausreichenden Test – z.B. eine 100 %ige Zweigüberdeckung – durchzuführen.

Defizite der Software-Technik

Vergleicht man die Charakteristika von Multimedia-Anwendungen und die Charakteristika von Multimedia-Entwicklungen mit traditionellen Anwendungen und Entwicklungen, dann stellt sich die Frage, inwieweit die Software-Technik auf diese neuen Charakteristika eingestellt ist.

Betrachtet man die Multimedia-Entwicklung, dann wird der Kern einer Multimedia-Anwendung in der Regel mit einem Autorensystem erstellt. Bei Autorensystemen handelt es sich in der Regel um Entwicklungsumgebungen, die es erlauben, die Benutzungsoberfläche grafisch interaktiv zu gestalten, Grafiken, Bilder und Ton zu integrieren sowie Zeitabläufe einzugeben. Die eigentliche Programmierung erfolgt meist durch proprietäre Skriptsprachen.

Vergleicht man diese Entwicklungsumgebungen mit Werkzeugen, die bei traditionellen Entwicklungen eingesetzt werden, dann handelt es sich bei diesen Entwicklungsumgebungen um Programmierumgebungen. Eine moderne Software-Entwicklung kennt heute aber mindestens die Entwicklungsphasen Definition, Entwurf und Implementierung (Programmierung). Objektorientierte Software-Entwicklungen starten mit einem objektorientierten Analysemodell (OOA) und gelangen über den objektorientierten Entwurf (OOD) zur objektorientierten Programmierung (OOP). OOA- und OOD-Modelle werden mit CASE-Werkzeugen erstellt und aus diesen Informationen Code-Rahmen für die Programmierumgebungen generiert.

Vergleicht man diese Situation mit einer Multimedia-Entwicklung, dann stellt man fest, daß es für Multimedia-Entwicklungen vor der Programmierung weder Werkzeuge noch geeignete Beschreibungskonzepte gibt.

Die Software-Technik hat in einem schmerzhaften Prozeß verbale Pflichtenhefte überwunden und formale Beschreibungskonzepte für Software-Anforderungen gefunden, wie SA, SA/RT und OOA. Parallel zur Entwicklung dieser Beschreibungskonzepte wurden entsprechende Werkzeuge erstellt, um die Modelle auf Konsistenz, Vollständigkeit und Eindeutigkeit zu überprüfen. Zusätzlich wurden Methoden entwickelt, um systematisch Modelle zu erstellen und zu überprüfen.

Im Multimedia-Bereich befindet man sich heute noch auf dem Stand der verbalen Pflichtenhefte, auch wenn sie dort Drehbücher, *storyboards* oder ähnlich heißen /Schifmann, Heinrich, Heinrich 97, S. 107 ff./. Damit hat man alle Nachteile verbaler Beschreibungen: mehrdeutig, unvollständig, inkonsistent, auf unterschiedlichem Abstraktionsniveau, schwer vergleichbar, nicht standardisiert, maschinell nicht auswertbar, für Generierungen nicht zu verwenden. Als Werkzeuge werden – wenn überhaupt – Textsysteme eingesetzt. Damit befindet man sich bei einer Multimedia-Entwicklung auf dem Stand Ende der 70er Jahre. 1976 wurde das *Entity-Relationship*-Konzept und 1979 die

strukturierte Analyse erfunden. Auch wenn manche Pflichtenhefte um Flußdiagramme und Bildschirmabzüge ergänzt sind, ändert dies jedoch die oben beschriebene Situation nicht.

Der Entwicklungsrückstand, bezogen auf Konzepte und Werkzeuge, beträgt im Multimedia-Bereich also ca. 20 Jahre! Hinzu kommt noch der methodische Nachholbedarf. Erfahrungsgemäß entwickelt sich eine gute Methode erst dann, wenn die Konzepte konsolidiert sind.

Was ist zu tun?

Die aufgeführten Defizite lassen sich nicht von heute auf morgen beheben. 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 der Benutzerinteraktion – sollten genutzt werden.

Insbesondere muß die Entwicklung von Multimedia-Systemen als ein wahrzunehmender und bedeutender Anwendungsbereich der Software-Technik begriffen werden. Multimedia ist Software!

Ein Beispiel

Am Lehrstuhl für Software-Technik der Ruhr-Universität Bochum werden seit fünf Jahren multimediale Lehr- und Lernsysteme für den Bereich Objektorientierung entwickelt /Object Lab 97/. Erfahrungen bei der Entwicklung dieser Systeme haben folgendes gezeigt:

- 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 Neuprogrammierung.
- Ein Lehr- und Lernsystem enthält gewisse, immer ähnlich wiederkehrende Elemente, die es nahelegen, dafür einen allgemeinen Rahmen bereitzustellen.

Aufgrund dieser Erfahrungen und um Transparenz in den Aufbau und die Struktur eines Lehr- und Lernsystems zu bekommen, wurde ein OOA-Modell für ein Lehr- und Lernsystem erstellt (Abb. 1 und Abb. 2).

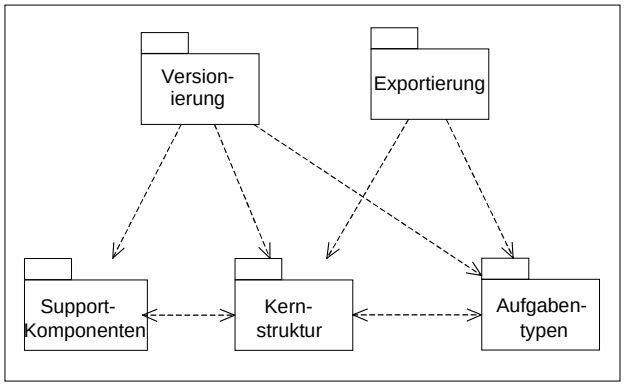

Abb. 1: Subsysteme eines Erfassungssystems für multimediale Lehr- und Lernsysteme

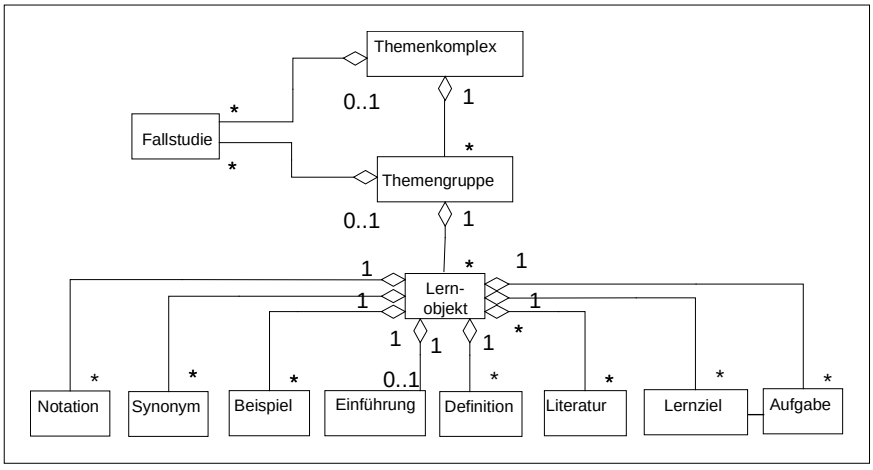

Abb. 2: OOA-Modell der Kernstruktur eines multimedialen Lehr- und Lernsystems

Das Modell zeigt die verschiedenen Komponenten in Form von Klassen und ihre Abhängigkeiten voneinander. Um das Problem der Verwaltung, Versionierung und Konfigurierung zu lösen, wurde mit dem Janus-Generatorsystem /Balzert, Hofmann, Kruschinski, Niemann 96/ ein Verwaltungssystem erzeugt. Das Janus-Generatorsystem erlaubt es, ausgehend von einem OOA-Modell ein lauffähiges Verwaltungssystem zu generieren. Das generierte Multimedia-Verwaltungssystem für das modellierte Lehr- und Lernsystem, im folgenden kurz MM-Verwaltungssystem genannt, erlaubt es dem Autor, seine Lehr- und Lerninhalte komfortabel

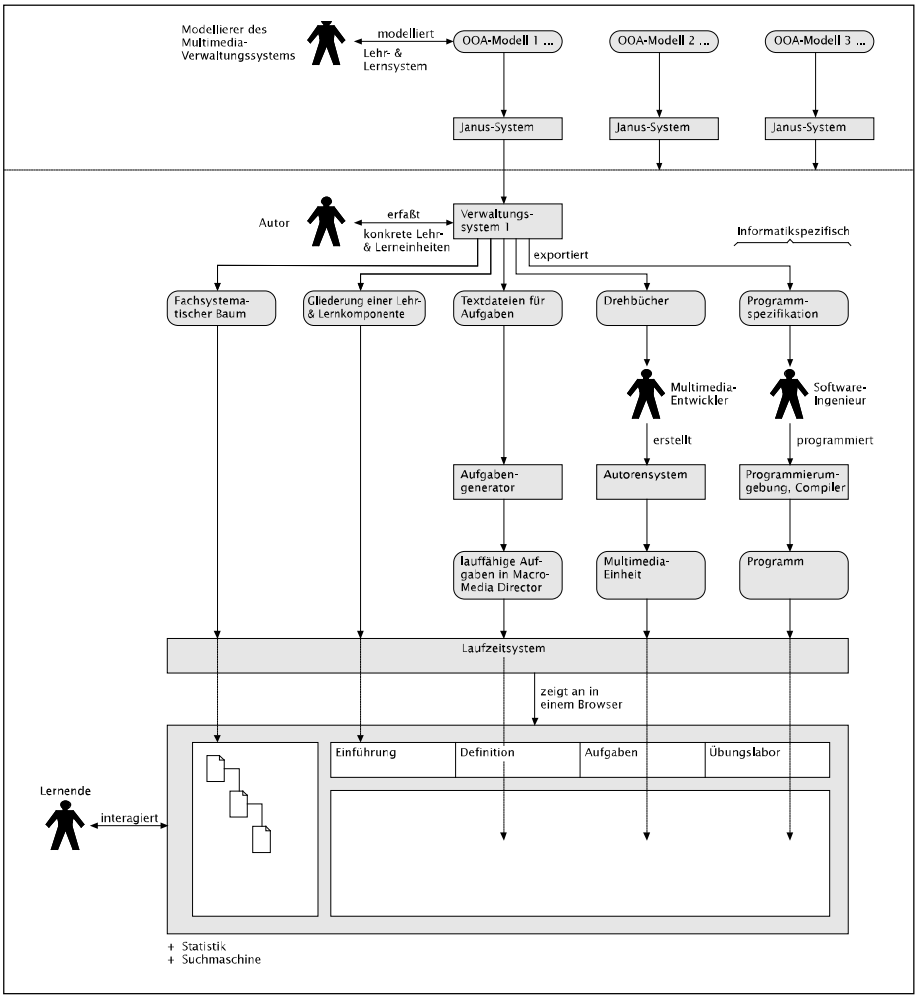

Abb. 3: Erstellung eines multimedialen Lehr- und Lernsystems unter Verwendung eines Verwaltungs- und Laufzeitsystems

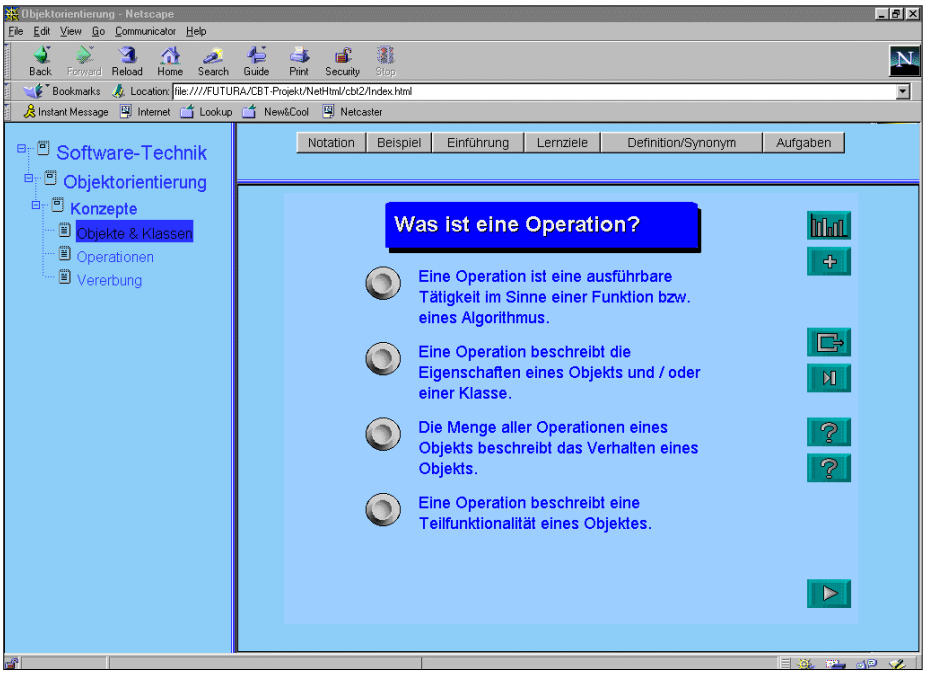

Abb. 4: Benutzungsoberfläche des Laufzeitsystems

und herstellerunabhängig zu erfassen. Die Benutzungsoberfläche entspricht dem Standard heutiger kaufmännischer Systeme.

Die Abb. 3 zeigt im oberen Teil den Modellierer eines MM-Verwaltungssystems. Er erstellt für das jeweilige Lehr- und Lernsystem ein OOA-Modell, das die spezifischen Anforderungen berücksichtigt. Beispielsweise unterscheidet sich ein MM-Verwaltungssystem für den Informatikbereich von einem MM-Verwaltungssystem für den Medizinbereich. Aus dem jeweiligen OOA-Modell wird mit Hilfe des Janus-Generatorsystems ein MM-Verwaltungssystem generiert.

Im generierten Verwaltungssystem werden von dem oder den Autoren folgende Informationen erfaßt und verwaltet:

- Fachsystematischer Baum, der das Lehr- und Lerngebiet hierarchisch strukturiert.
- Gliederung einer Lehr- und Lernkomponente, z.B. in Einführung, Definitionen & Notationen, Beispiele, Lernziele, Literatur, Aufgaben, Übungslabor.
- Aufgabentexte mit Sprechertexten, Bild- und Grafikdatei-Angaben.
- Drehbücher, *story boards* usw. als freier oder strukturierter Text.
- Für den Informatikbereich werden u.U. Programm-spezifikationen vorgegeben.

Aus dem Verwaltungssystem heraus können andere Anwendungen aufgerufen werden, z.B. Grafik-, Zeichen- und Bildsysteme, deren Dateien dann mitverwendet werden.

Der mittlere Teil der Abb. 3 zeigt, daß aus dem Verwaltungssystem eine Reihe von Dateien exportiert werden.

Die Aufgabentexte für Ja/Nein-, *Single Choice*- und *Multiple Choice*-Aufgaben werden in einen Aufgabengenerator importiert,

der daraus lauffähige Filme für den Macromedia-Director generiert, so daß dem Autor diese Arbeit erspart bleibt.

Sollen die Aufgaben später in Java implementiert werden – anstelle des Macromedia-Directors, dann muß nur der Aufgabengenerator, aber nicht die Aufgabenspezifikation im Verwaltungssystem, modifiziert werden.

Drehbücher müssen von einem Multimedia-Entwickler mit Hilfe eines Autorensystems umgesetzt werden, Programmspezifikationen von einem Software-Ingenieur mit Hilfe einer Programmierumgebung in ein Programm.

Der untere Teile der Abb. 3 zeigt das entwickelte Laufzeitsystem. Es integriert die verschiedenen Teile und ergänzt sie um zusätzliche Funktionalität wie Suchmaschine und Statistikauswertung und -anzeige. Generell erfolgt die Anzeige in einem WWW-Browser. Die Benutzungsoberfläche gliedert sich in drei Teile (siehe auch Abb. 4):

- Anzeige der fachsystematischen Gliederung in Form eines Baums (wie im Explorer) (links angeordnet). Hier wählt der Lernende die jeweils gewünschte Lehr-/Lernkomponente.
- Anzeige der Binnengliederung einer Lehr-/Lernkomponente (oben angeordnet). Hier wählt der Lernende die Lernform aus, die er innerhalb der Komponente bearbeiten will.
- Anzeige des Arbeitsbereichs (in der Mitte). Hier läuft die Präsentation ab, oder es werden die Aufgaben angezeigt usw.

Das Laufzeitsystem kann die Inhalte von CD-ROM, von der Festplatte oder einem Server laden, d.h. es ist eine *online*- und *offline*-Nutzung (auch kombiniert) möglich. Eine »kleine« Suchmaschine durchsucht das Internet automatisch oder nur auf Wunsch nach aktuellen oder kontextnahen Lehr- und Lerneinheiten (Kontextnähe einstellbar). Das Suchergebnis wird eingeblendet und der Benutzer kann zu weiteren Lehr- und Lerneinheiten auf Wunsch verzweigen. Die Suche erfolgt auf voreinstellbaren Servern.

Dieses Beispiel zeigt einen Weg, den Werkzeugeinsatz bei der Entwicklung von multimedialen Lehr- und Lernsystemen zu intensivieren, die Herstellerabhängigkeit von einzelnen Werkzeugen zu reduzieren und Teilbereiche wie die Aufgabenimplementierung zu automatisieren.

Das generierte und damit für den jeweiligen Anwendungsbereich leicht adaptierbare MM-Verwaltungssystem bringt folgenden Nutzen:

- Es kann von Autoren verwendet werden, um eigene Lehr- und Lerninhalte multimedial zu vermitteln. Die Erstellung geht wesentlich **schneller**, da das Verwaltungs- und Laufzeitsystem herstellerunabhängig zur Verfügung steht und Aufgaben teilweise generativ erstellt werden.
- Die **Qualität** ist wesentlich höher, da ein lauffähiges und bereits mehrfach eingesetztes System verwendet wird. Es stellt die Konsistenz des generierten Lehrangebots sicher und hält die Abhängigkeiten zwischen den Lehrinhalten bei Änderungen und Ergänzungen nach.
- Wegen der ersten beiden Punkte können **umfangreiche Lehr- und Lerninhalte** vermittelt werden. Dadurch können insbesondere Studenten selbstnavigierend und entsprechend dem eigenen Lerntempo lernen, Vorlesungen vor- oder nachbereiten, Lerndefizite ausgleichen und zeitlich und räumlich unabhängig lernen.
- Die Autoren müssen **keine Spezialisten** in Software, Pädagogik und Gestaltung mehr sein, sondern können sich auf ihr Fachgebiet konzentrieren. Dadurch können auch Wissenschaftsdisziplinen solche Systeme erstellen, denen bisher dieses Spezialwissen fehlte.

Ungelöst ist noch der methodische Bereich: Wie modelliert man die Anforderungen an Multimedia-Systeme in formaler Form? Ist eine objektorientierte Modellierung analog zu OOA-Modellen sinnvoll und möglich?

Erste Ideen zur formalen Anforderungsspezifikation

Multimedia setzt im Gegensatz zu den klassischen kaufmännischen Anwendungen weitere Medien ein und fügt den Anwendungen Dynamik hinzu. Die klassischen Anwendungen stellen somit eigentlich eine Teilmenge der Multimedia-Anwendungen dar. Sie sind »Multimedia-Anwendungen ohne Multimedia und Dynamik«.

Für Multimedia-Anwendungen ist daher hauptsächlich eine Erweiterung der statischen Benutzungsoberfläche (UI) und des statischen Fachkonzeptes um die dynamische Komponente erforderlich.

Die Multimedia-Anwendung stellt einen dynamischen multidimensionalen Informationsraum I(t) dar. Das UI(t) besteht im wesentlichen aus einem dreidimensionalen grafischen Informationsraum GI(t), der auf eine zweidimensionale Oberfläche GUI(t) transformiert wird, mehreren Tonkanälen sowie den Möglichkeiten zur Interaktion.

Der Informationsraum I einer multimedialen Anwendung wird über die Benutzungsoberfläche dem Benutzer dargeboten und von dem Benutzer über die von ihm ausgelösten Ereignisse E beeinflußt: $I = f(t,E)$.

$$I(t,E) = I_{statisch} + I_{dynamisch}(t,E)$$
$$I_{dynamisch}(t,E) = I_{unbeeinflußbar}(t) + I_{beeinflußbar}(t,E)$$
$$I_{beeinflußbar}(t,E) = I_{zeitunabhängige\ Reaktion}(E) + I_{zeitabhängige\ Reaktion}(t,E)$$

Bei der klassischen kaufmännischen Anwendung reduziert sich der Informationsraum auf
$I(E) = I_{statisch} + I_{zeitunabhängige\ Reaktion}(E)$.

Die Dynamik der Multimedia-UI-Elemente (MUI-Elemente) besteht in der Zeitabhängigkeit des Ortes (Translation), der Ausrichtung (Rotation), der Ausdehnung (Dimension) und des Zustandes (Metamorphose). Der Ton mit seiner Zeitbehaftung kommt noch hinzu.

Schon die Analyse- und Entwurfsphase müssen daher Konzepte und Methoden enthalten, die die Beschreibung von dynamischem zeitabhängigem Verhalten unterstützen. Aus der Regelungstechnik sind Methoden zur Beschreibung dynamischer Systeme bekannt, die hier Verwendung finden können. Für die Definition der Reaktionen auf die weiterhin vorhanden Benutzerinteraktionen können z.B. zeitabhängige Entscheidungstabellen dienen. Zustandsautomaten lassen sich auch verwenden.
Auch für Multimedia-Anwendungen gilt, daß eine möglichst gute Entkopplung von Fachkonzept, Benutzungoberfläche und Ereignisbehandlung vorliegen soll. Eine an die Erfordernisse von MM-Anwendungen angepaßte *Model-View-Control* (MVC)-Architektur zeigt Abb. 5.

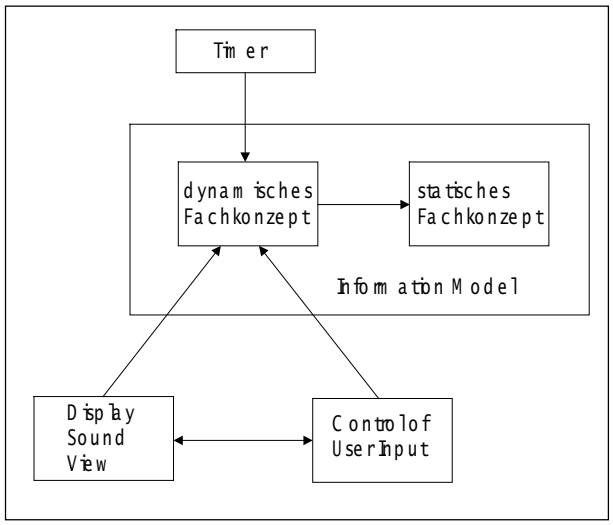

Abb. 5: MVC-Architektur für Multimedia-Anwendungen

Wie sieht dieses konkret in einem Beispiel aus? Angenommen ein Student soll auf dem Bildschirm als „Männchen“ dargestellt werden, das über eine Straße geht, auf Mausklick stehenbleibt und »Guten Tag!« sagt. In einer Sprechblase werden Name und Matrikelnummer angezeigt. Zehn Sekunden später geht der Student weiter.
Matrikelnummer und Name sind Attribute des klassischen statischen Fachkonzeptes. Das dynamische Fachkonzept enthält den aktuellen Ort, die Momentangeschwindigkeit und die Momentanbeschleunigung des Studenten. Da der Student geht und dieser Vorgang grafisch dargestellt werden soll, ist es ferner erforderlich, die aktuelle Bewegungsphase (z.B. »stehend«, »linkes Bein vorne«, »rechtes Bein vorne«, »Beine auf gleicher Höhe«) zu bestimmen. Die dynamische Änderung dieser Attribute wird durch die Objektoperationen bewirkt, die regelmäßig über den *Timer* aktiviert werden. Die Ereignisse, die der Benutzer erzeugt (z.B. Mausklick) werden auch an das dynamische Fachkonzept weitergeleitet und können einen abrupten zeit- und zustandsabhängigen Zustandswechsel bewirken (Stehenbleiben des Studenten). Ferner sind die Zustände »sprechend« und »Informationen anzeigend« zu berücksichtigen.
Die Benutzungsoberfläche aktualisiert sich, indem sie den aktuellen Zustand von dem dynamischen Fachkonzept erfragt und anzeigt (Tonabspielen ist hier impliziert). Das MUI-Element definiert wie der jeweilige Zustand, in dem sich das zu repräsentierende Objekt befindet, angezeigt wird. Hier sind beispielsweise die grafischen Darstellungen für die einzelnen Bewegungsphasen und der zu sprechende Text abgelegt. Mit jedem Wechsel in den Zustand »linkes Bein vorne« oder »rechtes Bein vorne« kann zusätzlich das Auftretgeräusch der Schuhe ertönen, ohne daß hierzu auf das Fachkonzept zurückgegriffen werden müßte.
Indem das Verhalten sowie die Verhaltensänderungen der Objekte beschrieben werden, erreichen die dargestellten Objekte eine Autonomie und müssen nicht für jeden Zeitpunkt der Anwendung festgelegt werden. Es ist ausreichend den Startzustand des Systems festzulegen und von da an »lebt« das System und damit die Anwendung. Die Objekte können aufeinander innerhalb des dynamischen Fachkonzeptes reagieren und ihr Verhalten dementsprechend ändern. Dynamisches Standardverhalten kann den Objekten durch Vererbung gegeben werden und durch Überschreiben spezialisiert werden. Durch diesen Ansatz lassen sich leicht multimediale Welten erschaffen und beschreiben. Beispielsweise kann man die Dynamik eines Balles beschreiben. Erzeugt man ein Objekt Ball, wird dieses zunächst vom System in freien Fall beschleunigt, trifft auf den Fußboden und springt wieder hoch im Moment des Auftreffens kann das Auftreffgeräusch ertönen. Das Aussehen des Balles wird durch das MUI-Element festgelegt und kann unabhängig von der Dynamik-Komponente gewählt werden.
Innerhalb der Anwendung werden Objekte erzeugt, die aufeinander reagieren und auch durch den Benutzer beeinflußt werden können. Damit lassen sich dynamische komponentenbasierte multimediale Anwendungen zusammenbauen.

Literaturverzeichnis

/Balzert, Hofmann, Kruschinski, Niemann 96/
Balzert H., Hofmann F., Kruschinski V., 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)

/Müller 95/
Müller, W.: Interaktive Medien im professionellen Einsatz: Elektronische Kataloge, Infoterminals, CBT, Videokonferenzen – Bonn, Paris: Addison-Wesley, 1995

/Object Lab 97/
Object Lab light, CD-ROM, Interaktives multimediales Lernsystem zur objektorientierten Softwareentwicklung, Lehrstuhl für Software-Technik, Ruhr-Universität Bochum: Spektrum Akademischer Verlag 1997

/Schifmann, Heinrich, Heinrich 97/
Schifmann, R. S.; Heinrich, Y.; Heinrich, G.: Multimedia Design interaktiv! Von der Idee zum Produkt - Berlin, Heidelberg: Springer-Verlag 1997