Ein Ansatz zur Verbesserung des Entwicklungsprozesses von Multimedia-Anwendungen

Ralph Depke, Gregor Engels, Katharina Mehner, Stefan Sauer, Annika Wagner
Universität-GH Paderborn, FB 17 – Mathematik/Informatik, 33095 Paderborn
{depke | engels | mehner | sauer | awa}@uni-paderborn.de

Zusammenfassung

Multimedia-Anwendungen sind interaktive Softwaresysteme, in denen Multimedia-Dokumente eingesetzt werden, d.h. Dokumente, in denen verschiedene Medientypen wie z.B. Text, Bild, Video und Audio auftreten. Heutzutage werden Multimedia-Anwendungen in der Regel mit Hilfe von Autorensystemen entwickelt. Diese unterstützen eine Ad-hoc-Entwicklung auf Implementierungsniveau ohne eine vorausgehende Modellierungsphase. Dies führt zu den in der Softwaretechnik bekannten Problemen wie fehlendem Entwurf und fehlender Dokumentation und damit auch zu unvermeidlichen Problemen bei der Wartung und Erweiterung eines solchen Systems.

Wir stellen in diesem Beitrag ein Vorgehensmodell vor, bei dem zwischen einer Entwurfs- und Implementierungsphase für Multimedia-Anwendungen unterschieden wird und systematisch aus einem Entwurf die Implementierung mit einem Autorensystem abgeleitet werden kann.

Keywords: Multimedia, Vorgehensmodell, Modellierung, Autorensystem

1 Einleitung

Multimedia-Anwendungen (MM-Anwendungen) sind interaktive Softwaresysteme, in denen Multimedia-Dokumente eingesetzt werden. MM-Dokumente wiederum sind Dokumente, die Objekte verschiedener Medientypen wie Text, Bild, Bewegtbild (Animation oder Video) und Audio zusammen verwenden. Diese Eigenschaft besitzen beispielsweise Textdokumente, in die Bilder eingebunden sind. Üblicherweise bezeichnet man aber eher Dokumente als multimedial, in die immanent zeitabhängige Medientypen wie Bewegtbild und Audio eingebunden sind. MM-Anwendungen zeichnen sich durch eine komplexe Struktur der MM-Dokumente und deren adäquate Präsentation aus. Die Komposition verschiedener Medienobjekte in MM-Dokumenten wird durch die inhaltliche Ausrichtung der Anwendung bestimmt.

Typische MM-Anwendungen sind Kiosksysteme zur Präsentation von Hinweisen und Informationen, Lehrprogramme zum computerunterstützten Lernen (*Computer Based Training*) und Komponenten von Electronic Commerce-Systemen, mit denen heutzutage im Internet Verkaufsgeschäfte durchgeführt werden.

MM-Anwendungen werden in der Regel mit Autorensystemen erstellt. Marktführer bei diesen Systemen sind Director [Mac97b], Authorware [Mac97a] und Toolbook [Asy97]. Hierbei handelt es sich um interaktive Werkzeugumgebungen, die die Erstellung von MM-Anwendungen mit Hilfe von Drag&Drop-Techniken ermöglichen. Falls die gewünschte Funktionalität der Anwendung die im Werkzeug vordefinierten Möglichkeiten übersteigt, steht in der Regel eine einfache Skriptsprache zur Verfügung, mit der die fehlende Funktionalität ausprogrammiert werden muß. Die wesentliche Idee von Autorensystemen besteht darin, die Ad-hoc-Entwicklung von MM-Anwendungen zu ermöglichen. Stellt man die Bildschirmgestaltung in den Vordergrund, sind Drag&Drop-Techniken die geradezu ideale Art, MM-Dokumente zu komponieren. Die Benutzung der vordefinierten Funktionalität für die Anwendungslogik erlaubt es dann, auf einfache Art zu einer Animation zu gelangen.

Dieser Vorteil von Autorensystemen ist gleichzeitig auch ihr größter Nachteil. Denn ein solches Vorgehen verleitet eher zu einem prototyphaften Ausprobieren als der gezielten Implementierung eines durchdachten Entwurfs. Ohne einen Entwurf entsteht jedoch keine ausreichende Dokumentation, insbesondere da ein Autorensystem auch die Implementierung nicht dokumentiert. Dies führt zu den aus der Softwaretechnik bekannten Problemen bei der Wartung von Anwendungen. Darüber hinaus befindet sich die vordefinierte Funktionalität auf dem niedrigen Abstraktionsniveau primitiver Programmierkonstrukte. Konzepte zur funktionalen Abstraktion, Datenabstraktion oder zur hierarchischen Strukturierung fehlen. Hierdurch wird der oben beschriebene Effekt einer schweren Wartbarkeit noch verstärkt.

Da interaktive MM-Anwendungen in der Zukunft eine weit verbreitete Form von Softwaresystemen darstellen werden, sollte es eine vorrangige Aufgabe der Softwaretechnik sein, verbesserte Konzepte, Notationen, Methoden und Werkzeuge für die Realisierung von MM-Anwendungen zu entwickeln [Müh96]. Ansätze dazu existieren bereits mit verschiedenen Schwerpunkten. Dabei liegt der Schwerpunkt von [Ker97] auf der Integration didaktischer Konzepte in den Softwareentwicklungsprozeß, während es [MF96] um die Integration beteiligter Designer geht. [MG96] betrachtet die Besonderheiten der Entwicklung verteilter MM-Anwendungen. [PVS98] beschreibt ein Vorgehensmodell für die Entwicklung von Hypermedia-Applikationen mit Domänen-, Instanzen- und Repräsentationsmodell.

Es ist das Ziel dieses Beitrags, einen Ansatz vorzustellen, wie mit Hilfe bekannter Softwaretechnikkonzepte der Entwicklungsprozeß für MM-Anwendungen auf der Grundlage existierender Autorensysteme verbessert werden kann. Wir gehen von der Vorstellung aus, daß der Multimedia-Entwicklungsprozeß im wesentlichen dem Prozeß der Softwareentwicklung gleicht, also auch aus den Phasen Analyse, Entwurf, Implementierung etc. besteht. Dabei liegt unser Hauptaugenmerk auf dem Entwurf und der Implementierung von MM-Anwendungen mit Autorensystemen. Während der Entwurfsphase wird ausgehend von einem Anwendungsentwurf und einem Modell der Realisierungsmöglichkeiten mit einem konkreten Autorensystem systematisch eine Implementierung für das Autorensystem abgeleitet.

Wir werden diese Idee eines verbesserten Entwicklungsprozesses für MM-Anwendungen auf der Basis des Autorensystems Director detailliert erläutern. Dabei ist wichtig zu betonen, daß unser Ziel ein verbesserter Entwicklungsprozeß ist und nicht, ein umfassendes Referenzmodell für MM-Anwendungen oder ein umfassendes Modell für das Autorensystem Director vorzustellen.

Der Beitrag ist wie folgt aufgebaut: Die Entwicklung von MM-Anwendungen mit dem Autorensystem Director erläutern wir in Kapitel 2. In Kapitel 3 stellen wir unser Vorgehensmodell vor. Die folgenden Kapitel behandeln einzelne Schritte des Vorgehensmodells im Detail. Das anwendungsneutrale Modell der Funktionalität von Director wird in Kapitel 4 beschrieben. In Kapitel 5 skizzieren wir den Aufbau eines realisierungsunabhängigen Modells von MM-Anwendungen. Die Instanziierung dieses Modells wird anhand einer Beispielanwendung in Kapitel 6 angegeben. Kapitel 7 erläutert, wie die Grundlage der Realisierung in Director systematisch aus den zuvor entwickelten Modellen abgeleitet wird. Der Beitrag schließt in Kapitel 8 mit einer Zusammenfassung und einem Ausblick.

2 Realisierung von Multimedia-Anwendungen mit Director

In diesem Kapitel sollen die Möglichkeiten der Realisierung von MM-Anwendungen mit dem Autorensystem Macromedia Director [Mac97b] vorgestellt werden. Dabei werden wesentliche Charakteristika der Arbeit mit Autorensystemen im allgemeinen herausgearbeitet. Diese Charakteristika dienen später der Motivation unseres Vorgehensmodells, das unabhängig von der Verwendung eines konkreten Autorensystems einsetzbar ist. Zur Illustration verwenden wir das Beispiel eines einfachen Lehrprogramms.

Abbildung 1 zeigt mehrere Screenshots dieser Beispielanwendung. Dabei soll einem lernenden Kind das Zählen und die Funktionsweise von Zahnrädern vermittelt werden. Nach der Auswahl des Menüpunkts *Zahlen* erscheint auf dem Bildschirm die Zahl 1, die durch eine von oben in die Szene schwebende Figur (Bär) illustriert wird. Die nächste Zahl und damit die nächste Bildschirmseite wird durch Klick auf einen Button erreicht. Zur Demonstration der Zahl 2 schweben zwei Bären ein. Der Lernende kann jederzeit zum Auswahlmenü gelangen, um mit einer anderen Lehreinheit fortzufahren. Als Alternative kann er im Beispiel die Funktionsweise von Zahnrädern anhand eines kurzen Videofilms demonstriert bekommen.

Das Autorensystem Director benutzt die Metapher der Filmproduktion. Man arbeitet mit Besetzungen bestehend aus Darstellern, einem Drehbuch und einer Filmleinwand, die hier Bühne heißt. Die von Director erzeugte MM-Anwendung wird Film genannt. Jeder Darsteller ist genau einer Besetzung zugeordnet und wird basierend auf vordefinierten Medientypen (Text, Grafik, Video, Audio) erzeugt. Im Unterschied zum üblichen Kino- oder Fernsehfilm kann ein Director-Film vom Betrachter interaktiv gesteuert werden. Der Film hat dabei nicht nur eine lineare Struktur, sondern auch Verzweigungen, und der Betrachter bestimmt interaktiv einen der möglichen Pfade durch den Film. Um solche Interaktion zu ermöglichen, existieren besondere Darsteller, die über Steuerelemente verfügen.

Betrachten wir konkret die Besetzung unseres Beispielfilms. Dazu gehören unter anderem die Textdarsteller *Lerneinheiten, Zahlen* und *Zahnräder*, sowie ein Graphikdarsteller *Bär*. Alle Buttons, die wir für die Navigation durch den Film verwenden, sind durch Schaltflächendarsteller vertreten. Wichtig ist an dieser Stelle die Beobachtung, daß die Besetzung *vollständig* und *konkret* angegeben wird. Es gibt jedoch nur einen Darsteller für gleich aussehende Buttons. Darsteller sind unabhängig von der Verwendung im Film.

Wie Darsteller für den Film eingesetzt werden, wird im Drehbuch beschrieben. Der Auftritt eines Darstellers (in Director als Sprite bezeichnet) umfaßt dabei neben dem Zeitpunkt auch die Dauer und die Position seines Auftretens. Ein Darsteller kann mehrmals im gleichen Film auftreten. Das Drehbuch basiert auf der Vorstellung, daß ein Film aus einer Folge von Einzelbildern besteht, hier Frames genannt. Demzufolge besteht die Zeitachse des Drehbuchs aus aufeinanderfolgenden, numerierten Frames. Soll ein Darsteller zu einem bestimmten Zeitpunkt auftreten, so wird er im Drehbuch bei der betreffenden Framenummer eingetragen. Dann wird dieser Auftritt auf die gewünschte Länge, d.h. die Anzahl von Frames, verlängert.

Im Beispiel (siehe Abb. 2) beginnt der Film mit der

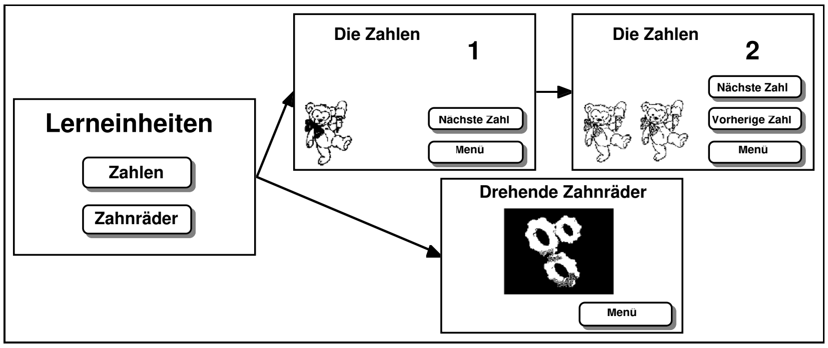

Abbildung 1: Beispielanwendung eines Lehrprogramms

Anzeige des Titels und der beiden möglichen Themen. Die entsprechenden Darsteller werden untereinander im ersten Frame eingetragen. Sie werden während der nächsten 9 Frames angezeigt. Die Länge ihres Auftrittes wird durch eine horizontale Linie unterhalb der Leiste mit den Framenummern visualisiert. Später treten dann andere Darsteller auf. Zur Erklärung der Zahl 2 tritt der Darsteller *Bär* gleich zweimal zur selben Zeit auf (vgl. Frame 25). Wichtig ist auch hier wieder die Beobachtung, daß das Drehbuch wie die Besetzung *konkret* und *vollständig* angegeben wird.

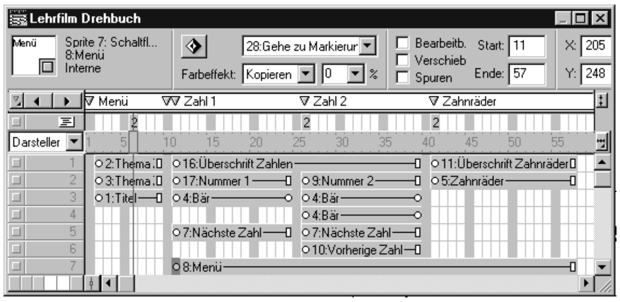

Abbildung 2: Drehbuch einer Director-Anwendung

Die Interaktion zwischen MM-Anwendung und Benutzer wird dadurch bestimmt, wie steuernde Darsteller während ihres Auftrittes auf Benutzereingaben reagieren. Insbesondere kann so die Navigation durch einen Film unterstützt werden. Dazu müssen Markierungen an bestimmten Frames angebracht werden, die dann als Sprungziele in den steuernden Darstellern angegeben werden können.

Im Beispiel (siehe Abb. 2) existieren die Markierungen *Menü*, *Zahl1*, usw. Sie befinden sich in der Leiste direkt über den Framenummern. Die Darsteller *Nächste Zahl* und *Vorherige Zahl* sind Steuerdarsteller, genauer gesagt Schaltflächen. Sie reagieren auf das Benutzerereignis *Maustaste gedrückt* mit einem Sprung zur Framenummer des bei ihnen definierten Sprungziels, also z.B. *Zahl1*.

3 Vorgehensmodell

In diesem Kapitel stellen wir das von uns vorgeschlagene Vorgehensmodell für die Entwicklung von MM-Anwendungen vor. Die Grundidee besteht darin, einen nach softwaretechnischen Prinzipien gestalteten Anwendungsentwurf mit Hilfe des Modells eines Autorensystems in ein Instanzmodell zu transformieren, das den Programmierkonstrukten des Autorensystems entspricht. Dieses Instanzmodell kann dann direkt in dem Autorensystem implementiert werden. Der wesentliche Vorteil dieses Vorgehens besteht darin, daß es die systematische Gestaltung einer Anwendung fördert und der Dokumentation und Wartbarkeit dient.

Im Gegensatz zu dem Entwurf eines herkömmlichen Softwaresystems, der weitestgehend auf der Typebene stattfindet und eine Instanzmodellierung nur zur exemplarischen Beschreibung von Szenarien verwendet, beinhaltet unsere Methode eine vollständige Modellierung auf Instanzniveau. Dies begründet sich darauf, daß die zur Realisierung eingesetzten Autorensysteme eine Beschreibung der MM-Anwendung auf Instanzebene verlangen, die wir in der Entwurfsphase spezifizieren müssen. Wie wir in Kapitel 2 gesehen haben, müssen in Director alle Darsteller und Sprites konkret als Instanzen angegeben werden, verallgemeinernde Beschreibungen für Klassen von Objekten sind nicht möglich.

Außerdem stellt ein Autorensystem eine gewisse Basisfunktionalität zur Verfügung, die für die Implementierung genutzt werden soll. Deshalb muß ein autorensystemspezifischer Entwurf angefertigt werden, der auf dieser Basisfunktionalität aufbaut. Das ist ähnlich zur Entwicklung von Datenbankanwendungen, bei denen zuerst ein konzeptionelles Schema der Anwendung erstellt wird, das von dem verwendeten logischen Datenmodell unabhängig ist. Anschließend wird ein Datenbankschema in dem logischen Datenmodell entworfen, das dem verwendeten Datenbanksystem zugrunde liegt. Schließlich erfolgt die Realisierung in dem physischen Schema des konkreten Datenbanksystems.

In unserem Vorgehensmodell abstrahieren wir also erstens von einer konkreten Anwendung durch die Einführung einer Typebene und zweitens von einem konkreten Autorensystem durch einen realisierungsunabhängigen Anwendungsentwurf.

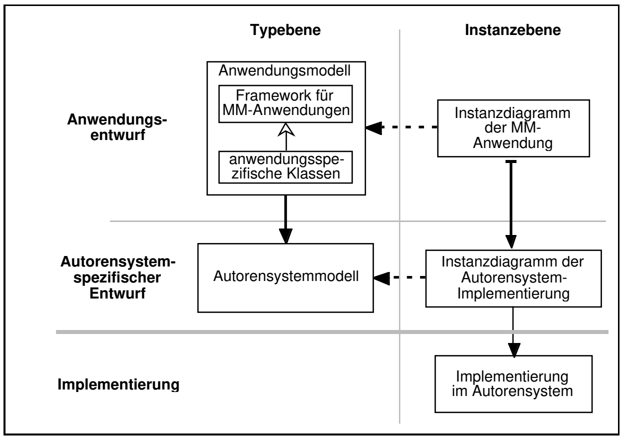

Abbildung 3: Vorgehensmodell für MM-Anwendungen

Abbildung 3 veranschaulicht unser Vorgehensmodell. Wir unterscheiden zwischen *Typ-* und *Instanzebene*. Orthogonal hierzu unterscheiden wir die zwei Entwurfsperspektiven realisierungsunabhängiger *Anwendungsentwurf* und anwendungsneutraler, *autorensystemspezifischer Entwurf*. Beide Perspektiven starten im Vorgehensmodell auf der Typebene.

Autorensystemmodell. Auf der Seite des autorensystemspezifischen Entwurfs, der das Modell der Implementierung liefert, beginnen wir mit einem Autorensystemmodell. Es soll dem Programmiermodell des Autorensystems möglichst genau entsprechen. Es nimmt damit eine ähnliche Funktion wie das Framework von Gibbs und Tsichritzis [GT96] ein, das eine abstrakte Schnittstellenbeschreibung zu dem System, auf dem eine MM-Anwendung abläuft, angibt. Dieses Modell muß nur einmalig für eine Autorensystemversion erstellt werden und kann für spätere Anwendungen, die mit Hilfe desselben Autorensystems realisiert werden sollen, wiederverwendet werden.

Anwendungsmodell. Beim realisierungsunabhängigen Anwendungsentwurf wird zuerst und ebenfalls einmalig ein allgemeines Framework für MM-Anwendungen erstellt. Dieses wird anschließend für die konkrete Anwendungsdomäne spezialisiert, wobei Klassen des Frameworks weiter verfeinert und abstrakte Klassen implementiert werden. Framework und anwendungsspezifische Klassen bilden gemeinsam das Anwendungsmodell. Die Entwicklung des Anwendungsmodells und die Erstellung des Autorensystemmodells sind unabhängig voneinander.

Relation und Abbildung auf Typebene. Auf der Typebene wird nun eine Relation zwischen dem Anwendungsmodell und dem Autorensystemmodell hergestellt. Diese setzt die Elemente der beiden Modelle, soweit geeignete Entsprechungen auf logischer Ebene erkennbar sind, zueinander in Beziehung. Aus dieser Entsprechung wird im folgenden, soweit möglich, eine Abbildungsvorschrift hergeleitet, die Elemente (Klassen und Beziehungen) des Anwendungsmodells auf Elemente des Autorensystemmodells abbildet.

Instanzdiagramm der MM-Anwendung. Auf der Seite des realisierungsunabhängigen Anwendungsentwurfs wird das Anwendungsmodell für die konkrete zu erstellende MM-Anwendung instanziiert. Dieser Schritt liefert ein Instanzdiagramm der Anwendung. Er kann parallel zur Bestimmung der Abbildungsvorschrift erfolgen.

Instanzdiagramm der Implementierung. Auf das so entstandene Instanzdiagramm der MM-Anwendung kann nun die Abbildungsvorschrift vom Anwendungsmodell auf das Autorensystemmodell angewendet werden. Diese Anwendung liefert schließlich als Ergebnis das Instanzdiagramm der Implementierung in dem Autorensystem, das als Grundlage der Realisierung dient. Weil diese Abbildung nicht surjektiv ist, müssen ggf. Elemente, die für die Implementierung notwendig sind, manuell ergänzt werden.

Implementierung im Autorensystem. Das Instanzdiagramm der Autorensystemimplementierung wird in das Autorensystem eingegeben und erforderlichenfalls ergänzt.

In dieser Arbeit ist das Vorgehen nur für den statischen Teil der Modellierung ausgearbeitet. In den folgenden Kapiteln werden wir Klassendiagramme für das Autorensystemmodell von Director und ein MM-Anwendungsmodell in der Unified Modeling Language (UML; [Rat97]) angeben. Aspekte des dynamischen Verhaltens werden nicht im Klassendiagramm berücksichtigt.

4 Modell des Autorensystems Director

Das Autorensystemmodell von Director soll in diesem Kapitel vorgestellt werden. Wir werden das interne Datenmodell von Director mit Hilfe eines Klassendiagramms erfassen. Das im folgenden definierte Autorensystemmodell erhebt nicht den Anspruch, die Möglichkeiten von Director vollständig wiederzugeben. Es dient lediglich der Illustration des Vorgehensmodells.

Den Ausgangspunkt im Diagramm bildet die Klasse Darsteller. Den Darstellern wird bei ihrer Instanziierung ein Medientyp zugeordnet. Zu den in Abbildung 4 nicht weiter spezialisierten Medientypen gehören Text, Bild und kontinuierliche Medientypen wie Video und Animation. Darsteller werden zu einer Gruppe, der Besetzung, aggregiert. Bei der Plazierung auf der Bühne wird zu dem gewünschten Darsteller ein Sprite erzeugt. Seine Position ist durch Attribute und seine Existenz während eines bestimmten Zeitabschnitts ist durch assoziierte Framenummern für Start und Ende festgelegt. Ein Sprite kann auf mehreren Darstellern basieren. Alle Sprites und Framenummern werden im Drehbuch zusammengefaßt. Die Verknüpfung der Framenummern durch die Assoziation *nächste Framenummer* legt die Reihenfolge der Frames und damit die zeitliche Abfolge der Sprites fest. Die Abarbeitung der Frames erle-

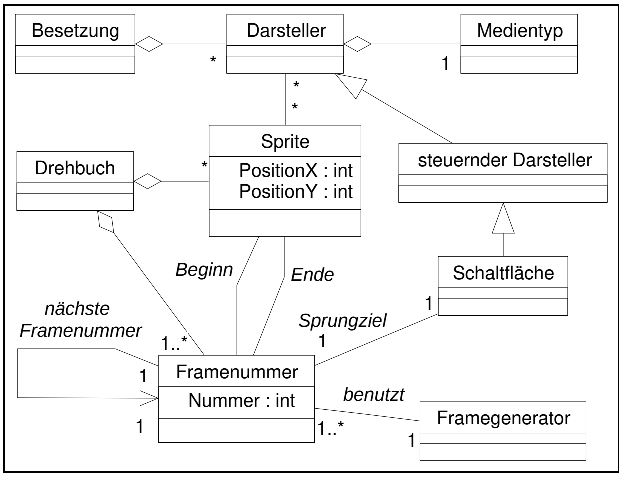

Abbildung 4: Internes Datenmodell von Director

digt ein Framegenerator, der die Framenummern in der definierten Reihenfolge durchläuft und die Sprites über die aktuelle Framenummer benachrichtigt. Die steuernden Darsteller bilden eine spezielle Darstellerklasse, die für Kontrollfunktionen und die Ablaufsteuerung eines Director-Films zuständig sind. Innerhalb eines Sprite darf höchstens ein steuernder Darsteller existieren. Beispielsweise realisiert die Klasse Schaltfläche Sprünge innerhalb des Films dadurch, daß sie über die Assoziation *Sprungziel* mit der Klasse Framenummer verbunden ist.

5 Ein einfaches Anwendungsmodell

Wie aus dem Vorgehensmodell ersichtlich, besteht das in dieser Arbeit vorgestellte Anwendungsmodell aus zwei Teilen:

- einem allgemeinen Framework für MM-Anwendungen
- anwendungsspezifischen Klassen, die durch Spezialisierung aus abstrakten Klassen des Frameworks abgeleitet oder dem Framework hinzugefügt werden.

Das vorgestellte Modell erhebt nicht den Anspruch, ein endgültiges Referenzmodell für MM-Anwendungen zu sein. Dazu wären weitere Spezialisierungen und Einschränkungen von Klassen und Assoziationen notwendig.

Im Gegensatz zum Multimedia-Framework in [GT96] betrachten wir ein Framework für MM-*Anwendungen*. Das Framework von Gibbs und Tsichritzis beschreibt ein abstraktes Application Programing Interface (API), welches durch konkrete Klassen für verschiedene Plattformen spezialisiert wird. Eine Anwendung verwendet dieses API als homogene Schnittstelle zu heterogenen Plattformen. Das hier vorgestellte Framework ist keine Schnittstelle zum System, sondern ein Anwendungsframework.

MM-Anwendungen sind interaktiv und haben einen wesentlichen Schwerpunkt auf der multimedialen Benutzungsschnittstelle. Das Framework ähnelt deshalb eher Software-Architekturen aus dem Bereich Modellierung von Benutzungsschnittstellen. Die bereits aus diesen Modellen bekannte Trennung von logischem Modell der Anwendung, Kontrollfunktion und Präsentation (vgl. z.B. Seeheim-Modell [Pfa85], MVC [KP88] oder PAC [Cou97]) haben wir aufgegriffen und um eine weitere Dimension zur Darstellung von Multimedia-Objekten der unterschiedlichen diskreten und kontinuierlichen Medientypen erweitert.

Das Framework enthält Klassen für Präsentation, Kontrolle und Medientypen. Bei der logischen Dimension gehen wir im Gegensatz zu den zitierten Modellen über die für das User Interface relevante Schnittstelle der Anwendung hinaus und ordnen hier die Anwendungsfunktionalität insgesamt ein. Deshalb haben wir eine Spezialisierung des MM-Anwendungsframeworks für das konkrete Anwendungsgebiet. Wir geben in dieser Arbeit nur sehr allgemeine anwendungsspezifische Klassen für ein Lehr-/Lernsystem an, die ihrerseits für komplexe Anwendungen weiter spezialisiert werden können.

Die vier genannten Dimensionen bilden die horizontale und vertikale Achse des in Abbildung 5 gezeigten Klassendiagramms. Oben im Diagramm befindet sich die Präsentationsdimension, rechts die Medientyphierarchie, links die Kontrolle und unten das logische Modell der Anwendung.

Kontrolle. Die Dimension der Kontrolle ist für die Ablaufsteuerung und somit das zeitliche Verhalten der Anwendung und Präsentation verantwortlich. In der Kontrolldimension unterscheiden wir lediglich System- und interaktive Benutzerkontrolle, da das statische Klassendiagramm ungeeignet zur Spezifizierung der tatsächlichen Steuerungsabläufe ist.

Medienhierarchie. Das wesentliche Kriterium zur Differenzierung der Medientypen ist deren Zeitabhängigkeit. Von den zeitunabhängigen (diskreten) Typen wie Text und Bild unterscheiden sich kontinuierliche Medientypen wie Bewegtbilder und Audio dadurch, daß ihre Präsentation sich im Lauf der Zeit ändert. Die Dimension der Medientypen basiert auf einer Generalisierungshierarchie verschiedener diskreter und kontinuierlicher Medientypen, wie sie beispielsweise im Framework von Gibbs und Tsichritzis zu finden ist (vgl. [GT96]). Diese wird erweitert um eine hierarchische Komposition von komplexen Medientypen aus atomaren und anderen komplexen Medientypen, so daß Medienobjekte einfache Objekte eines Medientyps oder strukturierte Instanzen diverser kombinierter Medientypen sein können, z.B. ein Video mit separatem Ton.

Präsentation. Die Dimension der Präsentation wird im Modell in die visuelle Darstellung und den Audiokanal differenziert. Im folgenden werden wir exemplarisch die Visualisierung betrachten, d.h. die Abbildung der räumlichen Anordnung der Medienobjekte. Der Präsentation, die das logische Gegenstück

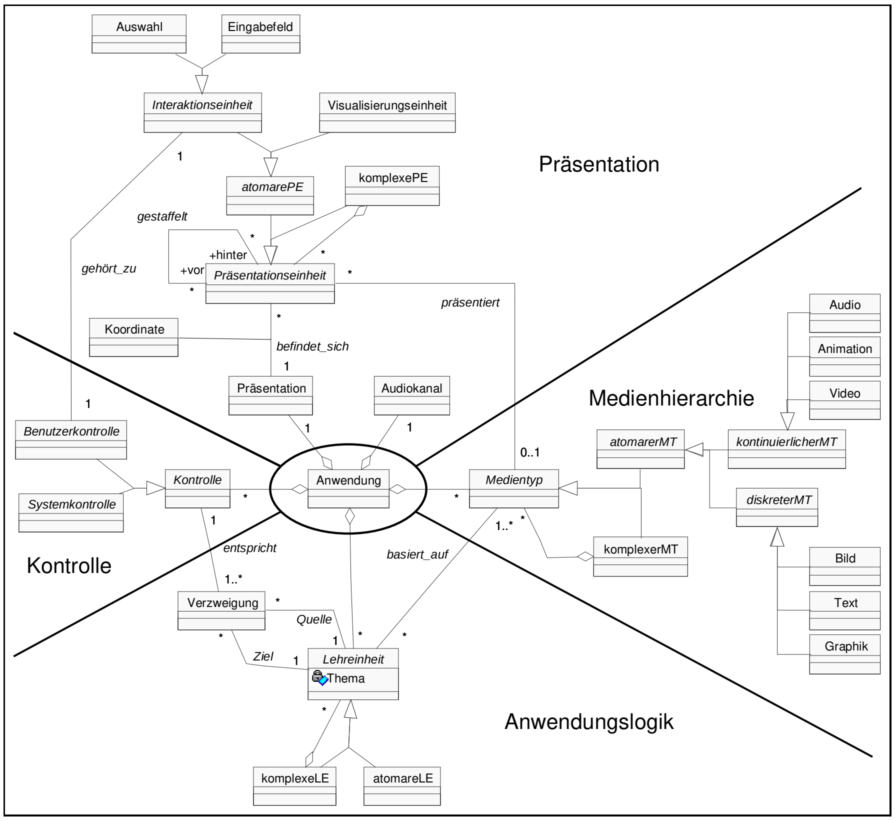

Abbildung 5: Multimedia-Anwendungsmodell

beispielsweise eines physikalischen Bildschirms darstellt, werden die verschiedenen Präsentationseinheiten einer Anwendung zugeordnet, wobei diese Beziehung über die Assoziationsklasse Koordinate weiter charakterisiert wird. Neben dieser zweidimensionalen Zuordnung geben wir eine räumliche Tiefenstaffelung, die sich auf einer zweidimensionalen Oberfläche als Überdeckung präsentieren kann, mit Hilfe der Assoziation *gestaffelt* an. Auch für Präsentationseinheiten sehen wir die Möglichkeit der hierarchischen Komposition vor, z.B. ein Fenster mit verschiedenen Schaltflächen, Eingabefeldern und Anzeigen. Atomare Präsentationseinheiten spezialisieren wir in Interaktionseinheiten und Visualisierungseinheiten. Instanzen der letzteren Klasse dienen ausschließlich der Anzeige von graphischen Objekten und lassen keine Benutzerinteraktion zu. Dieses ermöglichen die Instanzen der Klasse Interaktionseinheit, welche ihrerseits eine Generalisierung von Auswahl und Eingabefeld ist.

Anwendungslogik. Die logische Struktur der zu vermittelnden Information stellt einen eigenständigen Aspekt dar und ist i.a. der Ausgangspunkt bei der Entwicklung einer MM-Anwendung. Sie ist insbesondere aus softwaretechnischer Sicht bedeutsam. Die logische Dimension dient zur Darstellung des Wissens aus Sicht eines Experten und ist vom konkreten Anwendungsgebiet abhängig. Das bedeutet, daß insbesondere hier eine Spezialisierung des MM-Anwendungsframeworks erforderlich ist. Wir illustrieren diese inhaltliche Perspektive am Beispiel von Lehr-/Lernanwendungen, die sich aus verschiedenen Lehreinheiten zu vorgegebenen Themen zusammensetzten. Als allgemeine Strukturierungmöglichkeit verwenden wir die hierarchische Komposition von Lehreinheiten wie sie beispielsweise in der Kapitel- und Abschnittstruktur von Büchern zu finden ist. Zusätzlich sehen wir die explizite Angabe von Verzweigungen vor, um zu jeder Lehreinheit eine Menge von erlaubten Vorgänger- und Nachfolgerobjekten derselben Klasse spezifizieren zu können. So wird die Angabe verschiedener Navigationsstrukturen zu den Lehreinheiten möglich.

Beziehungen zwischen den Dimensionen. Präsentationseinheiten werden zu Medientypen in Beziehung gesetzt, wobei ein Medienobjekt durch verschiedene Präsentationseinheiten präsentiert werden kann. Lehreinheiten des logischen Modells basieren ihrerseits auf einem oder mehreren Medienobjekten, die auch in Beziehung zu verschiedenen Lehreinheiten stehen können. Die Benutzerkontrolle erfolgt über Interaktionseinheiten; Kontrolle erfolgt allgemein über die Verzweigungen im logischen Modell.

6 Instanziierung des Anwendungsmodells für die Beispielanwendung

Nachdem wir im letzten Kapitel ein allgemeines Modell für MM-Anwendungen vorgestellt haben, erläutern wir nun dessen Instanziierung für die Beispielanwendung. Wir beginnen mit der logischen Ebene. Hier ist zuerst der Aufbau der Lehreinheit zu betrachten und dann mögliche Bearbeitungsreihenfolgen. Wie man in Abbildung 1 erkennt, ist die gesamte Lehreinheit mit dem Thema *Lernen für Kleinkinder* in zwei Teile unterteilt. Das Lernen der Zahlen ist der erste Teil und bildet selbst wieder eine komplexe Lehreinheit, während das Lernen der Funktionsweise von Zahnrädern eine atomare Lehreinheit darstellt. Die erwünschten Bearbeitungsreihenfolgen wurden in Kapitel 2 erläutert und werden durch Objekte des Typs Verzweigung realisiert.

Die zu verwendenden Medientypen erläutern wir exemplarisch für die Lehreinheit *Zahl 2*. Sie basiert auf einem komplexen Medienobjekt, bestehend aus einer Animation (*Bär*) und einem Text als atomare Medienobjekte. Auf Präsentationsebene sehen wir für diese beiden atomaren Medienobjekte eigene Visualisierungseiheiten vor, die angeben, wo (mittels Koordinatenobjekten) und wie sie präsentiert werden. Dem Medienobjekt *Bär* werden zwei Visualisierungsobjekte zugeordnet. Darüber hinaus ist es wesentlich, diese atomaren Präsentationseinheiten zu komplexen zusammenzufassen, sobald sie gemeinsam angezeigt werden sollen. Daher gehören neben den Visualisierungsobjekten, die die Medienobjekte repräsentieren, auch noch die Präsentationen der Menüpunkte für die Steuerung des Ablaufs zur gleichen komplexen Präsentationseinheit. Bei *Zahl 2* sind das *Nächste Zahl*, *Vorherige Zahl* und *Menü*. Man beachte, daß ein Menüpunkt jeweils aus seiner Visualisierung sowie einem Auswahlobjekt für die entsprechende Interaktion komponiert wird. Diese Interaktionsobjekte gehören zur Benutzerkontrolle, die die Auswahl der auf *Zahl 2* folgenden Lehreinheit übernimmt.

7 Realisierung einer konkreten MM-Anwendung mit Director

Im folgenden konkretisieren wir unser Vorgehensmodell, indem wir genauer die einzelnen Schritte erläutern, die zur Entwicklung des Instanzdiagrammes für die Implementierung einer MM-Anwendung im Autorensystem notwendig sind. Dazu analysieren wir zuerst die Zusammenhänge zwischen dem Anwendungsmodell und dem Autorensystemmodell. Diese Analyse liefert als Ergebnis eine Relation der Modellelemente, die in Tabelle 1 zusammengefaßt ist. Aus dieser Relation leiten wir eine Abbildung des Anwendungsmodells auf das Director-Modell ab. Die Anwendung dieser Abbildung präsentieren wir anhand des Ausschnitts des Instanzdiagramms für die Beispielanwendung, wie er im letzten Kapitel erläutert wurde.

Die größten Parallelen zwischen beiden Modellen findet man bei der Medienhierarchie. Hier haben atomare Medientypen des Anwendungsmodells eine direkte Entsprechung in Medientypen des Director-Modells. Die Hierarchisierung von Medientypen, mit

Anwendungsmodell	**Autorensystemmodell**
atomarer Medientyp	Medientyp
komplexer Medientyp	− − −
atomare Präsentationseinheit	Sprite
komplexe Präsentationseinheit	− − −
Visualisierungseinheit	Sprite
Interaktionseinheit	Sprite
Lehreinheit	Besetzung
komplexe Lehreinheit	− − −
Verzweigung	Sprungzielkante bzw. Verkettung der Framenummern
− − −	Darsteller

Tabelle 1: Abbildung des Anwendungsmodells auf das interne Modell von Director

deren Hilfe man z.B. ein Video und begleitenden Ton zu einem komplexen Medienobjekt zusammenfassen konnte, fehlt allerdings in Director. Entsprechungen für die logische Ebene des Anwendungsmodells fehlen in Director fast vollständig. Die einzige Möglichkeit, etwas ähnliches zu formulieren, bietet die Besetzung. Somit setzen wir sie mit der Lehreinheit gleich. Damit kommen wir zum ersten konkreten Abbildungsschritt bei der Entwicklung des Instanzdiagramms für die Implementierung im Autorensystem:

- Zu jedem verwendeten atomaren Medienobjekt füge einen Darsteller in die Besetzung ein. Dabei wird für Medienobjekte mit einer Entsprechung auf logischer Ebene ein Mediendarsteller erzeugt. Gibt es diese Entsprechung nicht, so ist ein Steuerdarsteller zu verwenden.

  Beispiel: Die primitive Lehreinheit mit dem Thema Zahl 2 basiert auf einem komplexen Medienobjekt bestehend aus Animation und Text. Für diese beiden Medienobjekte sehen wir jeweils einen Darsteller in der Besetzung vor (vgl. Darsteller *Bär* und *Nummer 2*).

Wie und wo Medienobjekte auf dem Bildschirm dargestellt werden, ist im Anwendungsmodell durch eine dem Medientyp zugeordnete Präsentationseinheit repräsentiert. Eine Entsprechung haben sie in den Sprites des Datenmodells von Director. Wiederum gibt es jedoch keine Hierarchisierung, die erlauben würde, eine stets gleichzeitige Präsentation zu beschreiben. Außerdem entfällt die Unterscheidung zwischen Interaktions- und Visualisierungseinheiten in Director. Damit kommen wir zum zweiten konkreten Schritt:

- Für die dem Medienobjekt zugeordnete Visualisierungseinheit erzeuge einen Sprite für den entsprechenden Darsteller. Dabei ergeben sich die Werte der Attribute des Sprites aus dem Kontext der Visualisierungseinheit.

Beispiel: Zu den Darstellern *Bär* und *Nummer 2* wird je ein Sprite in das Drehbuch eingefügt und durch Angabe der Koordinaten positioniert.

Schließlich müssen wir uns noch der Realisierung des Ablaufs zuwenden. Mögliche Abarbeitungsreihenfolgen, wie sie im Anwendungsmodell durch die Verzweigungen zwischen Lehreinheiten beschrieben werden, werden im Director-Modell durch Sprungzielkanten zwischen Schaltflächen und Framenummern bzw. direkt durch die Verkettung der Framenummern realisiert.

- Im nächsten Schritt werden daher den verschiedenen Sprites zunächst irgendwelche Framenummern zugeordnet. Dabei ist lediglich darauf zu achten, daß komplexe Präsentationseinheiten gleichzeitig dargestellt werden, d.h. daß die entsprechenden Sprites die gleiche Framenummer erhalten.

  Beispiel: Die Visualisierungseinheiten der Medienobjekte Animation und Text gehören zur gleichen komplexen Präsentationseinheit. In der Beispielrealisierung erhalten die jeweiligen Sprites Framenummer 25 gemeinsam mit den Sprites für die Visualisierungseinheiten der Auswahlen *Nächste Zahl* und *Vorherige Zahl* (vgl. Abb. 2).

- Als letztes müssen die Framenummern gemäß der logischen Struktur der Verzweigungen zwischen den Lehreinheiten verzeigert werden. Wir unterscheiden hierzu zwei Fälle: Im ersten Fall betrachten wir eine Verzweigung, für die die Kontrolle mit einem Auswahlobjekt verbunden ist. Dieses Auswahlobjekt gehört zur gleichen komplexen Präsentationseinheit wie eine Visualisierungseinheit, der wiederum ein Sprite in Director entspricht. Da jeder Sprite nur maximal einen Steuerdarsteller enthalten kann, erhalten wir so die Schaltfläche, für die ein Sprungziel definiert werden muß. Bilden wir auf gleiche Art und Weise das Ziel der Verzweigung auf einen Sprite ab, so erhalten wir die gesuchte Framenummer als dessen Start.

  Im zweiten Fall ist nicht die Visualisierungseinheit der Auswahl der Ausgangspunkt, sondern die der Lehreinheit, von der die Verzweigung ausgeht. Das Vorgehen ist analog. Ist die Lehreinheit nicht visualisiert, so ist nichts zu tun. Andernfalls wird die Verzeigerung der erhaltenen Endeframenummer des Sprites wie oben auf die Startframenummer des Sprites der Verzweigung gesetzt.

  Beispiel: Die Verzweigungen der primitiven Lehreinheit *Zahl 2* sind Beispiele für den ersten Fall. So ergibt sich u.a. eine Sprungzielkante zur Framenummer der primitiven Lehreinheit *Zahl 1*. In der Realisierung ist das Framenummer 10 (vgl. Abb. 2). Die Sprungzielkante selbst ist im Drehbuch nicht zu sehen.

8 Zusammenfassung und Ausblick

In diesem Papier haben wir ein Vorgehensmodell vorgestellt, was den Entwicklungsprozeß komplexer MM-Anwendungen aus softwaretechnischer Sicht verbessern soll. Im Mittelpunkt steht dabei die Einbeziehung einer Modellierungsphase, in der insbesondere auch ein logisches Modell der Anwendung erstellt wird. Diese Modellierung dient einerseits der Dokumentation, um so spätere Änderungen vorzubereiten. Andererseits haben wir aber auch ein Verfahren vorgeschlagen, wie auf Grundlage eines solchen allgemeinen Modells eine Realisierung der Anwendung mit Hilfe eines marktüblichen Autorensystems durchgeführt werden kann.

Der vorgestellte Ansatz wird zur Zeit von den Autoren im Rahmen des NRW Universitätsverbundprojektes *Multimedia in der wirtschaftswissenschaftlichen Lehre* im praktischen Einsatz getestet. Außerdem wird das Vorgehensmodell aktuell in einer Diplomarbeit zur Entwicklung eines multimedialen Tutorials für UML eingesetzt. In einer studentischen Projektgruppe werden gegenwärtig Entwurfswerkzeuge entwickelt, die das vorgestellte Vorgehensmodell unterstützen sollen.

Literatur

[Asy97] Asymetrix Toolbook II, Version 6.0. Asymetrix Corp., Bellevue, WA, USA, 1997.

[Cou97] J. Coutaz. PAC-ing the architecture of your user interface. In: *Proc. 4th Eurographics Workshop on Design, Specification and Verification of Interactive Systems*, Springer-Verlag, 1997, pp. 15-32.

[GT96] S. J. Gibbs und D. C. Tsichritzis. *Multimedia Programming: Objects, Environments and Frameworks.* ACM Press, Wokingham, England, 1995.

[Ker97] M. Kerres. *Multimediale und telemediale Lernumgebungen.* Oldenbourg Verlag, München, 1997.

[KP88] G. E. Krasner und S. T. Pope. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *Journal of Object-oriented Programming*, Aug./Sept. 1988, 1(3): 26-49.

[Mac97a] Macromedia Authorware, Version 4.0. Macromedia Inc., San Francisco, CA, USA, 1997.

[Mac97b] Macromedia Director, Version 6.0. Macromedia Inc., San Francisco, CA, USA, 1997.

[MF96] S. J. Morris und A. C. W. Finkelstein. Integrating design and development in the production of multimedia documents. In: *Proc. Intl. Workshop on Multimedia Software Development*, IEEE, 1996, pp. 98-109.

[MG96] M. Mühlhäuser und J. Gecsei. Services, frameworks, and paradigms for distributed multimedia applications. *IEEE Multimedia*, Fall 1996, pp. 48-61.

[Müh96] M. Mühlhäuser. Issues in multimedia software development. In: *Proc. Intl. Workshop on Multimedia Software Development*, IEEE, 1996, pp. 2-9.

[Pfa85] G. E. Pfaff (Hrsg.). *User Interface Management Systems.* Springer-Verlag, Heidelberg, 1985.

[PVS98] P. Pauen, J. Voss und H.-W. Six. Modellierung von Hypermedia-Applikationen mit HyDev. In: *Proc. Modellierung'98*, pp. 69-73.

[Rat97] UML Notation Guide, Version 1.1. Rational, 1997.