REIHE INFORMATIK
2/94

MHEG: Ein Austauschformat für interaktive Multimedia-Präsentationen

Thomas Meyer-Boudnik
Universität Mannheim
Seminargebäude A5
68131 Mannheim

MHEG:
Ein Austauschformat für interaktive Multimedia-Präsentationen

Thomas Meyer-Boudnik

Universität Mannheim
Praktische Informatik IV
Postfach 10 34 62
68131 Mannheim
mey@pi4.informatik.uni-mannheim.de

Zusammenfassung

Die logische Struktur von interaktiven Multimedia-Präsentationen wird vergleichbar mit der Struktur eines Textdokumentes durch die Definition von Beziehungen zwischen den einzelnen Inhalten bestimmt. Je nach Anwendungsgebiet kann ein Autor heute aus einer Vielzahl von Sprachen zur Beschreibung dieser Strukturinformationen in Form eines Skriptes wählen. Die Wiedergabe einer Multimedia-Präsentation erfolgt stets elektronisch, d.h. durch die Abarbeitung des Skriptes mit Hilfe einer entsprechenden Laufzeitumgebung. Die verschiedenen Sprachen, die unterschiedlichen Laufzeitumgebungen und die Heterogenität der Multimedia-Systeme behindern somit einen sytemübergreifenden Austausch. Der vorliegende Bericht beschreibt den momentan noch in Entwicklung befindlichen MHEG-Standard. Mit einer systemunabhängigen Kodierung der Strukturinformationen schafft MHEG erstmalig ein standardisiertes Austauschformat für interaktive Multimedia-Präsentationen, das die Speicherung, den Austausch und die Wiedergabe von multimedialen Informationen erleichtern wird. Da dieses Format eine endgültige, d.h. ausführbare Darstellung repräsentiert, ist eine Weiterverarbeitung auf dieser Ebene nicht vorgesehen.

10. Feb. 94

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Ausgangspunkt

Den Ausgangspunkt dieses Artikels bildet die Überlegung, wie können in Zukunft multimediale Informationen in einer heterogenen Umgebung ausgetauscht werden? Multimediale Informationen setzten sich hier im weiteren sowohl aus einzelnen Inhalten, kodiert in verschiedenen zeitlosen (z.B. Text) und zeitbehafteten (z.B. Video) Medien, als auch aus Beziehungen zwischen diesen Inhalten zusammen. Letztere, auch Strukturinformation genannt, sind ein wichtiger Bestandteil einer jeden Multimedia-Präsentation. Man denke z.B. daran, daß die Struktur einer elektronischen Zeitung den redaktionellen Gedanken zum Ausdruck bringt, oder daß durch die Struktur eines Kurses die Lerninhalte didaktisch angeordnet werden.

Dieser Gedanke birgt im Grunde keinen neuen Ansatz, allerdings wird durch die technische Entwicklung eine aktuelle Fragestellung aufgeworfen. Hierzu vergleicht Abbildung 1 den Lebenszyklus eines traditionellen Dokumentes mit dem einer interaktiven Multimedia-Präsentation. Heute editiert ein Autor mit Hilfe eines Texteditors ein Dokument. Dabei bedient er sich für die eigentlichen Inhalte eines vom System unterstützten Zeichensatzes (z.B. ASCII), sowie einer für ihn meist durch den interaktiven Editor verborgenen Sprache zur Strukturbeschreibung (z.B. SGML). Zu diesem Zeitpunkt liegt das Dokument in der sogenannten editierbaren Darstellung vor. Durch einen anschließenden Formatierprozeß wird das Layout des Dokumentes bestimmt. Das Ergebnis ist eine Darstellung des Dokumentes in endgültiger Form. Ein typischer Repräsentant für diese Darstellung ist die Druckerbeschreibungssprache *Postscript*.

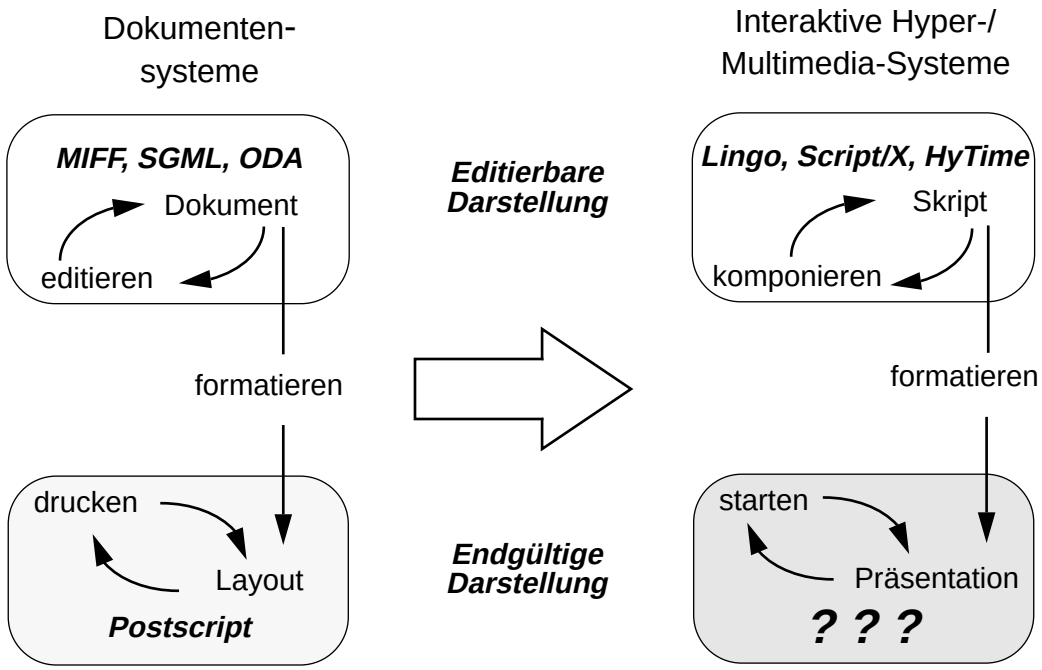

Abbildung 1: Problemstellung

Der Einfluß der Hypertext- und der Multimediatechnik wird das Erscheinungsbild der Dokumente drastisch verändern. Zwar wird die zuvor beschriebene Vorgehensweise bei der Dokumenterstellung bestehen bleiben, allerdings zeichnet sich schon heute ein wesentlicher Unterschied bzgl. der Wiedergabe eines interaktiven Hypermedia-Dokumentes gegenüber einem traditionellen Text-Dokument ab. Diese neuen Dokumente werden überwiegend computergestützt wiedergegeben werden, so daß man auch nicht mehr von einer endgültigen sondern von einer ausführbaren Darstellung der Präsentation spricht. Während auf der Ebene der editierbaren Darstellung schon eine Vielzahl von Formaten entwickelt wurde, ist die Normung einer ausführbaren Darstellung noch nicht abgeschlossen. Die Frage die sich nun stellt ist, wo wird ein solches zukünftiges Format entwickelt und welche Eigenschaften wird es aufweisen.

Auf der Ebene der internationalen Normung hat man erkannt, daß ein solches Format gerade in einer verteilten, heterogenen Systemumgebung von großer Bedeutung ist. Der vorliegende Bericht stellt die diesbezüglich eingeleitete Bestrebung zur Entwicklung eines entsprechenden Austauschformates für interaktive multimediale Präsentationen näher vor.

1.2 Hintergrund

In der ISO *(International Organization for Standardization*) befaßt sich der Unterausschuß SC29 (*Coded Representation of Audio, Picture, Multimedia and Hypermedia Information*) mit der Normung von Austauschformaten und Kodierverfahren für Multimedia-Systeme. Die eigentlichen Standards werden in den vier Arbeitsgruppen WG9-WG12 unter Beteiligung von Wissenschaft und Industrie erarbeitet. Die Verabschiedung erfolgt durch Abstimmungen in den jeweiligen nationalen Normungsgremien.

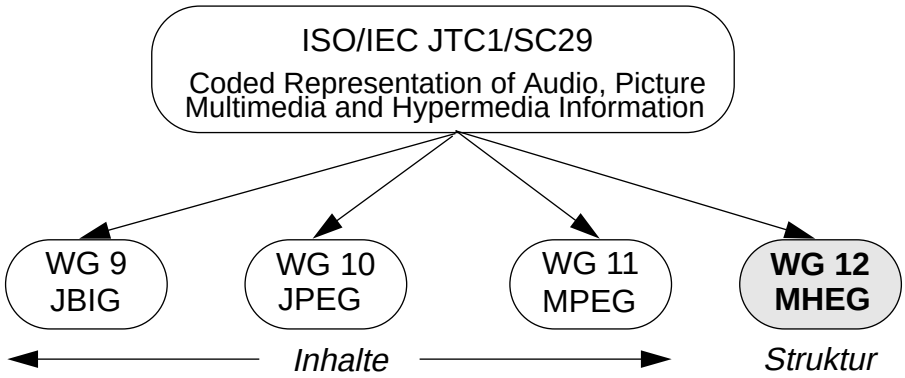

Abbildung 2: Arbeitsgruppen innerhalb des ISO-Unterausschusses SC29

Abbildung 2 zeigt, daß sich die drei inzwischen verabschiedeten Standards [11], [12], [13] mit der Kodierung von Medien befassen. Davon sind im Rahmen der Multimedia-Systeme wohl die Ergebnisse der Arbeitsgruppen: WG10 *(Joint Picture Experts Group,* JPEG) und WG11 (*Motion Picture Experts Group*, MPEG) von besonderer Bedeutung. Der JPEG-Standard definiert ein Kompressionsverfahren für Einzelbilder. In der Praxis wird dieses Kompressionsverfahren gerne angewendet, um Videofilme als eine Sequenz von JPEG-kodierten Einzelbildern zusammenzusetzen. Auch wenn diese Anwendung des JPEG-Standards keiner Normung unterliegt, wird diese häufig als Motion-JPEG bezeichnet. Die Normung eines derartigen Verfahrens ist vielmehr die Aufgabe des MPEG-Standards, der ein entsprechendes Kompressionsverfahren und ein Austauschformat für Video mit assoziertem Audio festlegt.

Die oben genannten Standards beschreiben die Inhalte in Form von Informationsobjekten. Die Struktur einer Multimedia-Präsentation (z.B. zeitlicher Ablauf) ergibt sich erst durch zeitliche und räumliche Beziehungen zwischen diesen Informationsobjekten. Die Normung dieser Strukturbeschreibung ist Gegenstand der Arbeitsgruppe WG12, die unter den Namen *Multimedia and Hypermedia information coding Expert Group* (MHEG) bekannt ist [22]. Der dort derzeit entwickelte Standard heißt *Information Technology - Coded Representation of Multimedia and Hypermedia Information Objects* (MHEG-Objekte), wird aber dennoch häufig mit MHEG abgekürzt. Der endgültige MHEG-Standard wird voraussichtlich in drei Dokumenten beschrieben. Im ersten Teil werden sowohl die Konzepte als auch das Austauschformat ansich behandelt. Der zweite Teil beschreibt eine alternative, semantisch zum ersten Teil isomorphe Syntax des Austauschformates. Im dritten Teil des Standards soll eine Referenzarchitektur für die Anbindung zu den Skriptsprachen dargestellt werden. Die beiden letzteren Dokumente sind erst im Entstehen und können deshalb hier nicht berücksichtigt werden, so daß sich alle weiteren Ausführungen zu MHEG auf die Committee-Draft-Version des ersten Dokumentes [15] beziehen.

Der Artikel gliedert sich wie folgt: Im folgenden Abschnitt werden die wesentlichen Konzepte des MHEG-Standards dargelegt. Hierzu gehören sowohl die Ebenen als auch ein Modell des Datenaustauschs. Desweiteren wird mit der Einführungen eines ersten Szenarios die MHEG-Klassenhierarchie motiviert. Im dritten Abschnitt werden die einzelnen Klassen näher vorgestellt, indem, soweit dem Autor bekannt, auf die Hintergründe des Designs sowie der praktischen Verwendung der einzelnen Klassen näher eingegangen wird. Mit jeweils einer ASN.1-Kodierung eines exemplarischen Objektes wird diese Beschreibung abgerundet. Der vierte Abschnitt diskutiert die Entstehung und den Einsatz von MHEG-Objekten. So wird als erstes aus dem Informationsfluß eine Art “Lebenszyklus” für die MHEG-Objekte abgeleitet. Die Verarbeitung von MHEG-kodierten Präsentationen wird anhand einer Analogie verdeutlicht. Zu diesem Zeitpunkt wird dem Leser spätestens deutlich, daß für die Wiedergabe eine Laufzeitumgebung notwendig wird. Wie eine solche Laufzeitumgebung aussehen kann, welche Anforderungen an sie gestellt werden und für welche Anwendungsbereiche sich diese sinnvoll einsetzen läßt, wird im Anschluß an die Analogie dargelegt. Der Artikel endet mit einem Fazit und der weiteren Entwicklung des zukünftigen Standards.

2 Grundlagen

2.1 Ebenen des Datenaustauschs

Bisher haben sich die Informationsanbieter aus den verschiedenen Branchen der Medienlandschaft ihrer eigenen Technologien bedient (HiFi-Geräte, Telekommunikationsendgeräte, Computersysteme usw.), um ihre Inhalte an den Verbraucher weiterzugeben. Die Integration dieser Technologien in einem Multimedia-System führt in Zukunft zu einer Konzentration des immensen Informationsangebotes. Multimedia-Systeme übernehmen dann die Aufgaben der Speicherung, Verarbeitung und Wiedergabe von Informationen, die in den unterschiedlichsten Medien kodiert sind. Der Erfolg der Multimedia-Systeme wird allerdings mit von dem Zustandekommen eines “offenen”, d.h. eines von vielen Systemen unterstützten Informationsaustausches bestimmt. Technisch bedeutet dies, daß geeignete Austauschformate für multimediale Anwendungen *de jure* festgelegt werden müssen oder sich derartige *de facto* etablieren. Die Informationsanbieter erkennen in dieser Entwicklung die Chance, demnächst neue Anwendungen für den Benutzer zur

Verfügung stellen zu können. Die vielen inzwischen entstandenen "Joint Ventures" im Multimediamarkt (z.B. Kaleida: Apple/IBM, Videotel: Springer/Telekom, TV-Digital: Bell Atlantic/TCI) sind für diese Entwicklung ein eindeutiges Indiz. Dennoch ist im allgemeinen ein noch recht zögerliches Investitionsverhalten der Unternehmen festzustellen. Warum wird z.B. das kürzlich erschienene Buch über die Grundlagen von Multimedia-Systemen [24] nicht in einer CD-ROM-Version angeboten? Zum Durcharbeiten des Buches wäre dann ein Lesegerät notwendig. Technisch bieten sich da einige Alternativen an (z.B. *Compact Disc Interactive* oder *Apple - QuickTime*). Das Buch müßte, um einen breiten Markt ansprechen zu können, für mehrere Technologien mit unterschiedlichen Formaten aufgelegt werden, was wiederum mit immensen Kosten verbunden ist. Der Ausweg wäre ein einheitliches Austauschformat, doch diesbezüglich ist die Situation noch unklar. Neben dem hier diskutierten MHEG-Standard kann dieser offene Informationsaustausch auf vier weiteren, in Abbildung 3 schematisch dargestellten, Ebenen mit unterschiedlicher Auswirkung erreicht werden.

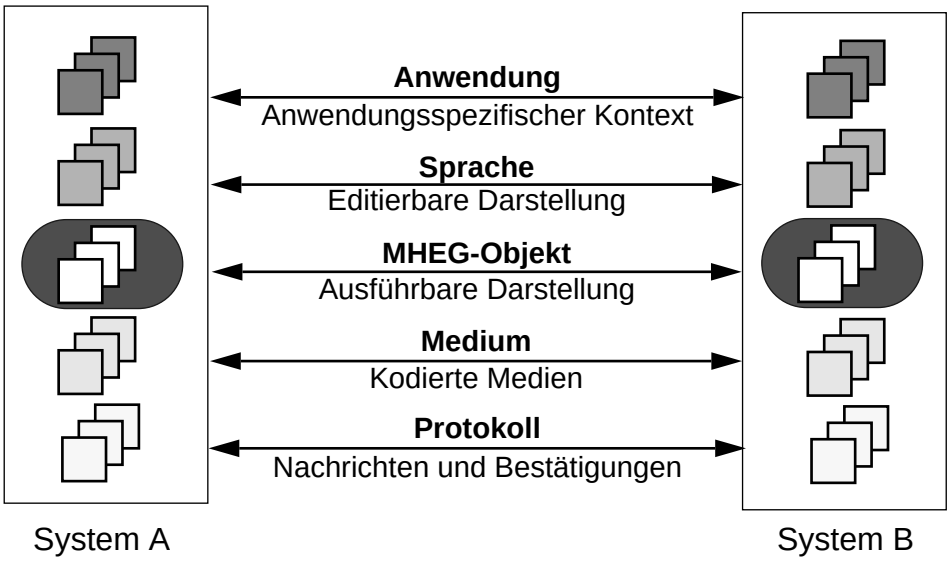

Abbildung 3: Ebenen des Informationsaustauschs

Anwendung

Auf der obersten Ebene erfolgt ein Datenaustausch im Kontext der jeweiligen Anwendung. Jeder Anwendung bleibt es freigestellt, das Format, der zwischen den beteiligten Systemen zu übermittelnden Datenstrukturen und deren Kodierverfahren, selbst zu bestimmen. Eine oftmals fehlende Normung ist charakteristisch für diese Ebene.

Sprache

An dieser Stelle kann man eine Multimedia-Präsentation für einen Augenblick mit dem Dokumentenbegriff gleichsetzen. In Analogie zur Dokumentenarchitektur ODA spricht man auf dieser Ebene von Dokumenten in der sogenannten editierbaren Darstellung (*revisable form*) [2]. Dieses Format wird in der Regel dazu benutzt, um in einem dafür ausgelegten Editor die Struktur des Dokumentes bearbeiten zu können. Für Multimedia-Präsentationen werden diesbezüglich häufig sogenannte Skriptsprachen (z.B. Script/X [17]) verwendet, und aus dem Anwendungsgebiet der Hypermedia-Dokumente ist z.B. der HyTime-Stan-

dard [10], der auf der Beschreibungssprache SGML [8] basiert, zu nennen. Erst durch einen Formatier- bzw. Layoutprozeß, bei dem das Dokument den System- und Benutzeranforderungen angepaßt wird, nimmt es seine endgültige Gestalt an (vgl. Abbildung 1). Ein herkömmliches Dokument wird bekanntlich auf Papier ausgegeben. Statt dessen wird ein Multimedia-Dokument interaktiv auf einem Bildschirm präsentiert, wodurch eine Laufzeitumgebung zur Wiedergabe notwendig wird.

MHEG-Objekte

Aus der Sicht der Multimedia-Systeme ist dieser Ebene bisher kaum Beachtung geschenkt worden, so daß mit dem hier vorgestellten MHEG-Standard ein erster Repräsentant für ein entsprechendes Austauschformat vorgestellt wird. Dieses Austauschformat kann als Schnittstelle für die oben erwähnte Laufzeitumgebung aufgefaßt werden. Der Layoutprozeß erzeugt in diesem Fall systemunabhängigen Pseudo-Code (MHEG-Objekte), der zum Zeitpunkt der Wiedergabe von der Laufzeitumgebung interpretiert wird. Änderung auf dieser Ebene lassen sich im allgemeinen nicht mehr mit dem Editor auf der Sprachebene rekonstruieren. Somit ist MHEG in Analogie zu Postscript eine endgültige bzw. ausführbare Darstellung (*executable* bzw. *final form*) des Dokumentes. Abschließend sei angemerkt, daß innerhalb der ISO ein gemeinsames Verständnis (es bestehen in der ISO formelle Liaisons zwischen den Gruppen) sowohl zwischen MHEG und HyperODA bzw. HyTime [10] besteht, indem derartige Dokumente bei entsprechender Umwandlung durch eine MHEG-Laufzeitumgebung angezeigt werden können [18].

Medium

Auf dieser Ebene werden Austauschformate für die Medien selbst bereitgestellt. Genaugenommen reicht die Festlegung des Typs noch nicht einmal aus, da in beiden Systemen das jeweilige Kodierverfahren bekannt sein muß. Das muß in Zukunft nicht immer gelten, denn eine andere Alternative wäre, daß die Medien durch entsprechende Konverter dem verarbeitenden System angepaßt werden. In der herkömmlichen Datenkommunikation ist das Problem schon öfters auf derartige Weise gelöst worden. In Multimedia-Systemen ist dies, solange man Echtzeitkommunikation voraussetzt, aus Gründen der nicht ausreichenden Systemleistung gerade für Videodaten bisher noch unrealistisch. Sogenannte Mixer, die in Konferenzsystemen mehrere Audioströme zu einem Datenstrom mischen oder Filter, die z.B. einen Audiostrom von PCM 16 Bit auf PCM 8 Bit transformieren, zeigen, daß dies für Audiodaten heute schon möglich ist. Für den Austausch von Einzelbildern sei auch auf das *Image Interchange Facility* [14] verwiesen.

Protokoll

Die unterste Ebene ist den Diensten in Form von Nachrichten und Bestätigungen für den Austausch von kodierten Informationen vorbehalten. Hierzu gehören Kommunikationsprotokolle in ihrem eigentlichen Sinn als Transportprotokolle oder auch Anwendungsprotokolle z.B. in Abfragediensten zum Anfordern von Daten aus entfernten Datenbanken. Meist wird auf dieser Ebene direkt eine binäre Kodierung für die einzelnen Datenfelder einer Protokolldateneinheit angegeben. Anwendungsprotokolle, die z.B. auf der Basis von entfernten Prozeduraufrufen realisiert werden, benutzen oft die im folgenden vorgestellte Technik der abstrakten Datenbeschreibung (z.B. ASN.1).

2.2 Modell des Datenaustauschs

Datenaustauschformate sind in offenen Systemen wegen der unterschiedlichen Darstellung der Daten notwendig. Eine generelle Vorgehensweise, die verhindert, daß die notwendige Konvertierung der Daten für jede Anwendung erneut gelöst werden muß, bietet das aus dem ISO-Referenzsystem für offene Systeme abgeleitete Modell des Datenaustauschs. Die dort genormte Darstellungsschicht hat als Aufgabe, die Beschreibung und Kodierung von Informationen in einer gemeinsamen Sprache zu unterstützen. Die ISO geht davon aus, daß sich zwei Anwendungsinstanzen mit Hilfe der angebotenen Dienste darüber einigen, wie ihre Daten strukturiert sind und welche Datentypen und -werte sie verwenden wollen. In Abbildung 4 sind die einzelnen Anwendungsinstanzen durch die Präsentationskomponenten A und B dargestellt. Dieses Modell wird vom MHEG-Standard angewandt, so daß MHEG-Objekte in den folgenden drei syntaktischen Beschreibungen vorliegen können:

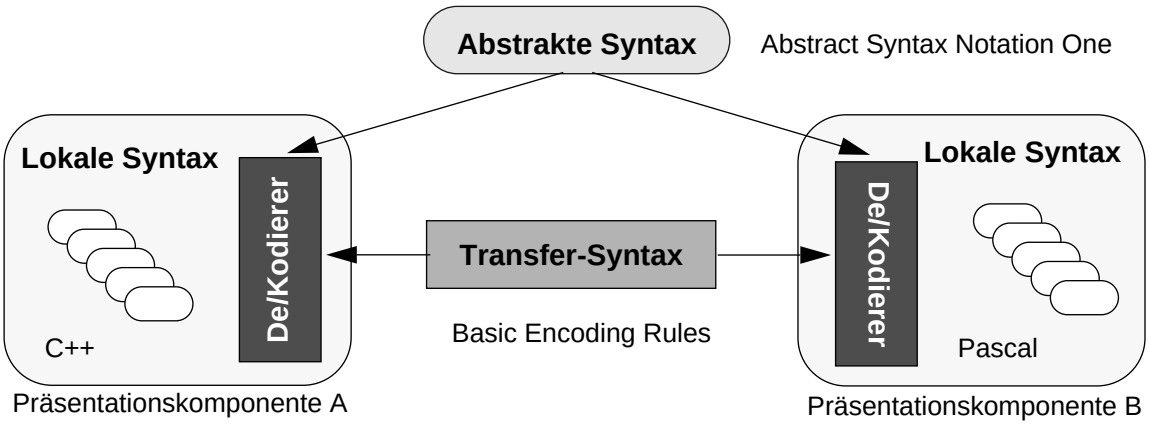

Abbildung 4: MHEG-Objekte in verschiedenen Syntaxen

Abstrakte Syntax

Die Abstrakte Syntax dient der kodierungsunabhängigen Beschreibung von Datenstrukturen und -werten. Von der ISO wurde deshalb im Standard IS 8824 (*Abstract Syntax Notation One*, ASN.1 [5]) eine abstrakte Syntaxnotation für offene Systeme genormt. Im ersten Teil des MHEG-Standards wird neben einer präzisen objektorientierten Definition der Datenstrukturen eine formale ASN.1 Spezifikation angegeben. Außer in der ASN.1 Beschreibung werden MHEG-Objekte im zweiten Teil des Standards zusätzlich in einer isomorphen SGML-Notation angegeben. Diese Form der Beschreibung wurde zur Verdeutlichung der engen Beziehung zu den Dokumentensysteme mit aufgenommen.

Lokale Syntax

Die einzelnen Anwendungsinstanzen verfügen über systemabhängige Datenrepräsentationen. Dies können z.B. für die Präsentationskomponente A C++-Datenstrukturen für einen Intel-Prozessor und für die Präsentationskomponente B Pascal-Datenstrukturen für einen Motorola-Prozessor sein. Die MHEG-Engine in der jeweiligen Präsentationskomponente verarbeitet die MHEG-Objekte auf der Basis dieser Lokalen Syntax. In der Regel werden sogenannte ASN.1-Compiler (vgl. ROSY/PESY aus der ISODE-Entwicklungsumgebung) eingesetzt, um aus der Abstrakten Syntax entsprechende Datenstrukturen für die jeweils eingesetzte Programmiersprache zu erzeugen.

Transfer Syntax

Die Transfer Syntax, auch konkrete Syntax genannt, hat die Aufgabe, eine bitweise Kodierung der abstrakt definierten Datenstrukturen vorzunehmen. Als Ergänzung zur Sprache ASN.1 wurden im ISO-Standard IS 8825 entsprechende Kodierungsregeln (*Basic Encoding Rules* [6]) festgelegt. Bevor MHEG-Objekte zwischen den verschiedenen Systemen ausgetauscht werden können, müssen Sie auf dem sendenden System mit einem entsprechenden Kodierer von der Lokalen Syntax in diese systemunabhängige Transfer Syntax umgewandelt werden. Der Dekodierer des empfangenden Systems stellt die Datenstrukturen in der dort verwendeten Lokalen Syntax wieder her. Kodierer und Dekodierer lassen sich ebenfalls in Form von Quelltext von einem ASN.1-Compiler generieren. Verwendet eine Anwendung als Abstrakte Syntax die alternative SGML-Notation, dann werden die MHEG-Objekte mittels SDIF [7] kodiert.

2.3 Interaktive Multimedia-Präsentationen: Ein Beispiel

Bevor hier im einzelnen auf die MHEG-Objekte eingegangen wird, werden anhand eines Szenarios die einzelnen Elemente einer Präsentation herausgearbeitet. Abbildung 5 stellt zu diesem Zweck als Beispiel eine interaktive Multimedia-Präsentation in Form eines Zeitdiagramms dar. Die Präsentation beginnt mit etwas Musik. Sobald in der Audiosequenz die Stimme einer Sprecherin zu hören ist, soll für ein paar Sekunden eine Grafik auf dem Bildschirm erscheinen. Nach dem Ausblenden der Grafik kann der Betrachter noch einen Text studieren, der mit etwas Hintergrundmusik untermalt wird. Mit der Beendigung der Darstellung des Textes erscheint auf dem Bildschirm ein "Stopknopf" mit dessen Hilfe der Benutzer die Audiosequenz unterbricht. Nun gibt er über ein angezeigtes Eingabefeld den Titel einer gewünschten Videosequenz ein. Diese Videodaten werden im Anschluß an die Modifikation angezeigt.

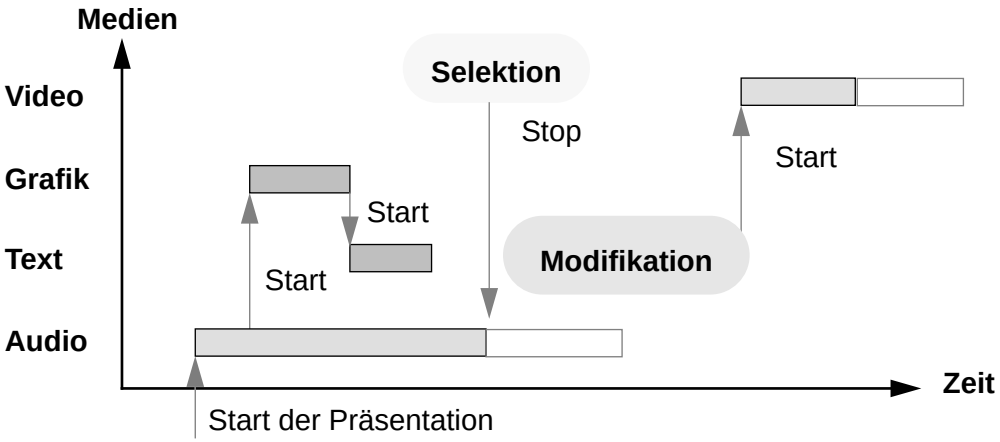

Abbildung 5: Zeitdiagramm einer interaktiven Präsentation

Inhalt

Die Präsentation besteht aus einer Folge von Darstellungen einzelner Informationen. Zur Repräsentation dieser Informationen stehen Medien mit sehr unterschiedlichen Eigenschaften zur Verfügung. Neben den zur Darstellung spezifischen Voraussetzungen ist es aus Gründen der Wiederverwendung sinnvoll, jede Information als ein einzelnes Objekt aufzufassen. Diese sogenannten Inhalte (*Content*, im weiteren beziehen sich kursiv markierte Begriffe auf ASN.1 Datenstrukturen aus dem Standard) sind in dem betrachteten Beispiel die Videosequenz, die Audiosequenz, die Grafik und der Text.

Verhalten

Unter dem Begriff “Verhalten” werden alle Aktivitäten zusammengefaßt, die sowohl die Darstellung der Inhalte betreffen als auch den Ablauf definieren. Ersteres wird über Befehle (*Action*) wie *start, stop, set_-position* usw. gesteuert. Letzteres entsteht durch die Definition zeitlicher, räumlicher und konditionaler Beziehungen (*Link*) zwischen den einzelnen Inhalten. Wann immer sich der Zustand der Darstellung eines Inhaltes ändert, kann dies der Auslöser für weitere Befehle auf andere Objekte sein (z.B. das Löschen der Grafik bewirkte die Anzeige des Textes). Eine weitere Möglichkeit das Verhalten einer Präsentation zu bestimmen, ist mit dem Aufruf externer Programme oder Funktionen (*Script*) gegeben.

Benutzerinteraktion

In dem angegeben Szenario konnte die laufende Animation durch entsprechende Benutzerinteraktionen unterbrochen werden. Bei genauem Betrachten stellt man fest, daß zwei Arten von Benutzerinteraktion unterschieden werden können. Zum einen die einfache Selektion (*Selection*), die durch eine vorgegebene Auswahl den Ablauf der Präsentation steuert (z.B. Drücken des Stopknopfes). Zum anderen die komplexere Modifikation (*Modification*) bei der dem Benutzer über einen längeren Zeitraum die Möglichkeit zur Eingabe von Daten gestattet wird (z.B. Editieren eines Dateneingabefeldes).

Container

Durch das Zusammenfassen der oben genannten Elemente erhält man schließlich eine ablauffähige Präsentation. Damit diese Präsentation als Ganzes zwischen den beteiligten Systemen ausgetauscht werden kann, benötigt man ein Element (*Composite*), das vergleichbar einem Container die zuvor genannten Objekte zu einer Einheit verbindet. Im Sinne eines Hypertext/Hypermedia-Dokuments können derartige Container zu einer komplexen Struktur angeordnet werden, indem diese durch sogenannte Hypertext-Verweise miteinander verknüpft werden.

2.4 Motivation der Klassenhierachie

In Abbildung 6 sind die einzelnen Elemente in der MHEG-Klassenhierarchie zusammengefaßt. Von allen fettgedruckten Klassen (alle Blätter) können Instanzen erzeugt werden. Diese Instanzen werden als MHEG-Objekte bezeichnet. Die kursiv beschrifteten Klassen (alle Knoten einschließlich der Wurzel) sind abstrakte Klassen, d.h. es können keine Instanzen kreiert werden. Lediglich die Wurzel des Baumes vererbt als abstrakte Basisklasse einige Attribute an die Blätter, während die inneren Knoten keine weitere Funktionalität beinhalten. Diese haben lediglich die Aufgabe, einzelne Klassen zu sinnvollen Gruppen zu vereinigen. So werden die Action-, die Link- und die Script-Klasse unter der Behaviour-Klasse zusammengefaßt, da diese das Verhalten in einer Präsentation definieren. Die Interaction-Klasse steht wiederum für die Benutzerinteraktion, die durch die Selection- und die Modification-Klasse modelliert wird. Zusammen mit der Content- und der Composite-Klasse stellen sie die einzelnen Komponenten in der Präsentation und ergeben somit die Component-Klasse. Über die Descriptor-Klasse können einige Eigenschaften der jeweiligen MHEG-Engine abgefragt werden. Die Macro-Klasse dient der Vereinfachung des Zugriffs auf bzw. der Wiederverwendung von Objekten. Beide Klassen spielen hier nur eine untergeordnete Rolle, so daß sie nicht weiter behandelt werden.

Die Entwicklung des MHEG-Standards erfolgt unter Anwendung der Technik des objektorientierten Designs. Hierzu sollte man sich in Erinnerung rufen, daß einem objektorientierten Entwurf zwei zueinander orthogonale Hierarchien zugrunde liegen. Zum einen stehen Klassen über die Vererbung von Verhalten und Struktur in Beziehung ("Ist-ein-Beziehung") und bilden dadurch die Klassenhierarchie. Zum anderen setzen sich Objekte dieser Klassen aus Objekten andere Klassen zusammen ("Hat-ein-Beziehung") und lassen dadurch die Objekthierarchie entstehen.

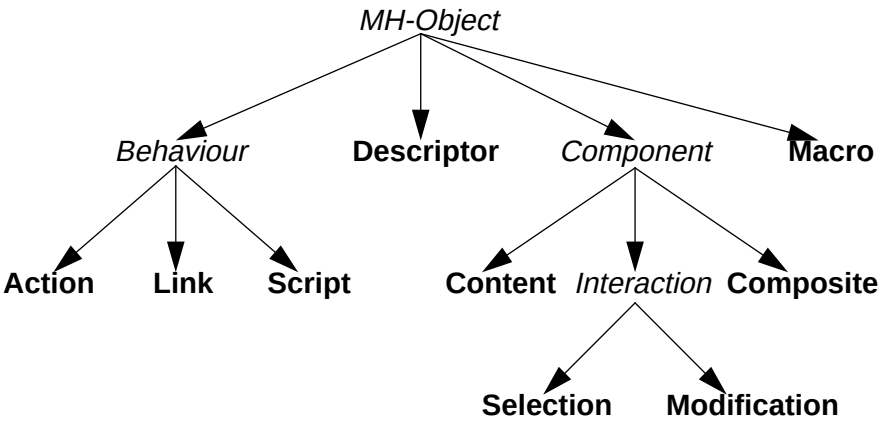

Abbildung 6: Klassenhierarchie der MHEG-Objekte

Obwohl eine Klassenhierarchie oft als Kern eines objektorientierten Designs aufgefaßt wird, muß man bei genauem Betrachten feststellen, daß die MHEG-Klassenhierarchie nicht die Bedeutung hat, die man ihr oft zuspricht. Methoden auf die Klassen werden im Standard nicht angegeben, so daß in der Klassenhierarchie lediglich einige Attribute der MH-Object-Klasse an alle davon abgeleiteten Klassen weitergereicht werden. In diesem Zusammenhang wird ein Problem mit der Verwendung von ASN.1 Syntax zur Beschreibung der MHEG-Klassen sichtbar. Die ASN.1-Spezifikation des MHEG-Standards sieht für jede Klasse ein eigenes Modul vor. Da ASN.1 keine Vererbungsmechanismen unterstützt, behilft man sich letztenendes wieder durch die Aggregation der Attribute aus der MH-Objekt-Klasse in die einzelnen Klassen. Diese wird in ASN.1 durch die Möglichkeit des Imports bzw. Exports einzelner Datentypen zwischen den Modulen unterstützt. Somit kann festgehalten werden, daß die Klassenhierarchie hauptsächlich aus didaktischen Gründen angegeben wurde, denn für die Implementierung einer MHEG-Laufzeitumgebung hat sie keine wesentliche Bedeutung. Für die aus den ASN.1-Datenstrukturen entstehende Objekthierarchie, auf die als solche im Standard nicht explizit eingegangen wird, trifft dies dagegen nicht zu.

In den folgenden beiden Abschnitten werden zuerst die Klassen im einzelnen vorgestellt, um im Anschluß daran einige wesentliche Mechanismen des MHEG-Standards zu diskutieren. Diese Ausführungen ersetzen nicht das eigentliche Standarddokument, sondern es wird auf die Hintergründe des Entwurfs eingegangen. Wenn technisches Details gefragt sind, dann sollte der interessierte Leser das Standarddokument zu Rate ziehen. Aus Platzgründen wurde außerdem auf eine syntaktisch korrekte bzw. vollständige Darstellung der in ASN.1 kodierten Beispiele von MHEG-Objekten verzichtet.

3 Beschreibung der MHEG-Objekte

3.1 MH-Object-Klasse

Die abstrakte MH-Object-Klasse vererbt die folgenden beiden Datenstrukturen an alle abgeleiteten Klassen zum Zwecke der Verwaltung der einzelnen MHEG-Objekte.

MHEG-Identifier

Die Datenstruktur *MHEG-Identifier* dient der Adressierung von MHEG-Objekten. Mit dem Attribut *Application-Identifier* wird eine bestimmte Anwendung durch eine Liste von Integerwerten identifiziert. Etwaige Regeln zur Vergabe dieser Nummer sind im Standard nicht festgelegt worden. In dem folgenden Beispiel wurde z.B. eine hierarchisch aufgebaute Nummer gewählt. Die *Object-Number* ist eine Zahl, die nur innerhalb dieser Anwendung definiert ist. Auf diese Weise erhält jedes MHEG-Objekt einen eindeutigen Schlüssel.

```
MHEG-Identifier {
   Application-Identifier: 9.228.50.7,
   Object-Number: 1
}
```

Description

Es besteht die Möglichkeit jedes MHEG-Objekt für sich durch eine Reihe von Attributen genauer zu charakterisieren. Dies kann z.B. dann sinnvoll sein, wenn eine Präsentation in ihre Bestandteile zerlegt wird und die einzelnen MHEG-Objekte in einer Datenbank abgelegt werden. Ein Autor kann unterstützt durch geeignete Suchfunktionen bestehende MHEG-Objekte wiederverwenden. Alle Attribute dieser Struktur sind optional, d.h. es bleibt der Anwendung überlassen, ob eine Beschreibung der einzelnen Objekte gewünscht ist. Das nachfolgende Beispiel ist selbsterklärend, so daß auch nicht auf die Bedeutung der einzelnen Attribute näher eingegangen wird.

```
Description {
   Name: “The object”,
   Owner: “Thomas & Guido”,
   Version: “1.0”,
   Date: “3/14/94”,
   Copyright: “University of Mannheim”,
   Comment: “no comment”,
   Keywords: no value -- optional
}
```

Aus rein praktischen Überlegungen hat man neben diesen beiden Datenstrukturen noch die Datenstruktur der *Generic-Address* in das Modul der MH-Object-Klasse mit aufgenommen. Diese wird in einigen Objekten als komplexe Datenstruktur aggregiert, aber nicht grundsätzlich an alle von der MH-Object-Klasse abgeleitet Klassen weitervererbt. Da sich hinter dieser Datenstruktur komplexe Adressierungstechniken verbergen, die nicht ohne die Einführung der einzelnen Klassen sinnvoll erklärt werden können, wird erst im Rahmen der Composite-Klasse näher darauf eingegangen. Für das bessere Verständnis der folgenden Ausführungen sei vorweggenommen, daß diese Datenstruktur eine Referenz auf ein anderes in einer Präsentation adressierbares MHEG-Objekt enthält.

3.2 Content-Klasse

Die Content-Klasse unterscheidet sich in sofern von den weiteren Klassen, als diese als einzige einen Bezug zu den eigentlichen Inhalten herstellt. Über die Content-Klasse werden diese Informationen auf sehr flexible und für das unterstützende System offene Art und Weise eingebunden. Jedes Content-Objekt repräsentiert innerhalb einer Präsentation genau eine Information, indem es auf die zur Darstellung notwendigen Daten verweist.

Der Typ des jeweiligen Mediums wird in einem Content-Objekt durch das Attribut *MHEG-Classification* definiert. Die eigentlichen Daten können entweder in dem Objekt enthalten sein oder lediglich über einen eindeutigen Identifikator referenziert werden. Im ersten Fall spricht MHEG von *Included-Data* woraus folgt, daß die Daten ebenso wie das Content-Objekt selbst vor jedem Austausch durch den Kodierer/Dekodierer umgewandelt werden. Dies ist allerdings aus Gründen der Effizienz nur für Inhalte mit geringer Datenmenge, wie diese z.B. bei Texten als Untertitel oder Beschriftungen von Menüs und Knöpfen vorkommen, sinnvoll. Im zweiten Fall der *Referenced-Data* wird lediglich ein Identifikator im Content-Objekt kodiert. Zum Zeitpunkt der Verarbeitung eines derartigen Content-Objektes muß gewährleistet sein, daß die referenzierten Daten über geeignete Anwendungsdienste angefordert werden können. Die Referenzierung ist vorallem dann geeignet, wenn sich mehrere Content-Objekte, unabhängig davon ob sie einer oder mehrerer Präsentationen angehören, auf die gleichen Daten beziehen. Für beide Fälle gilt, daß identifiziert durch den *Hook* eine dem Medium entsprechende Darstellungskomponente zum Anzeigen der Daten aufgerufen wird. Dieser speichert das verwendete Kodierverfahren (*Encoding-Identification*) und etwaige Parameter (*Encoding-Description*). Wird diese Information weggelassen, dann überläßt man es dem ausführenden System, die entsprechende Darstellungskomponente zu initialisieren. Dies kann z.B. dann sinnvoll sein, wenn in einer Datenbank ein Inhalt in verschiedenen Kodierverfahren abgespeichert wurde. Je nach Konfiguration des wiedergebenden Endsystems wird die entsprechend unterstützte Kodierung herangezogen.

Ein Auszug der im Standard vorgesehenen Kodierungen (*MHEG-Catalogue*) ist in Tabelle 1 aufgeführt. Neben diesen vordefinierten Formaten werden auch zukünftige Standards (*Non-MHEG-Standardized-Catalogue*) sowie anwendungsspezifische Kodierverfahren (*Proprietary-Catalogue*) berücksichtigt, indem man der Anwendung bzw. der jeweiligen Implementierung die Definition von eigenen Formaten offen hält. Dies mag zwar zu Inkompatibilitäten führen, erhält dem Standard aber noch eine Rest an Flexibilität, um auch mit zukünftigen Entwicklungen umgehen zu können.

MHEG-Classification	MHEG-Catalogue
Numeric	nicht definiert
Text	ISO 646 + ISO 6429
Graphics	ISO 8632 CGM
Still-picture	MR, MMR & MH, ITU-TS rec. T6 ISO 10918, JPEG ISO 10918, JPEG + JFIF ISO 10918, JPEG + T101 annex F ISO 11544, JBIG

Tabelle 1: Auswahl einiger vorgesehener Medienarten

MHEG-Classification	MHEG-Catalogue
Audio	pcm linear, MIDI ITU-TS rec. G711, G721, G722 & G723 ITU-TS rec. J41 & J42 ISO 11172, MPEG Audio
Video	ISO 11172 MPEG Video ITU-TS rec. H.261
Audio-visual	ISO 11172 MPEG System ITU-TS rec. H.261
Non-media-data	ISO 9069, SGML ISO 10744, HyTime ISO 8613, ODA, PDAM & HyperODA

Tabelle 1: Auswahl einiger vorgesehener Medienarten

Virtuelle Koordinatensysteme

Zusätzlich zum Kodierverfahren können in einem Content-Objekt auch die ursprüngliche Größe (*Original-Size*) und für kontinuierliche Medien die Wiedergabegeschwindigkeit (*Original-Speed*: *yes*, *no*) festgelegt werden. Die dazu notwendigen Maßeinheiten werden aus einem sogenannten *Generic-Space* entnommen. Innerhalb dieser virtuellen Koordinatensysteme können Content-Objekte relativ in Dimension und Anordnung zueinander definiert werden. Das räumliche Koordinatensystem besteht aus drei Achsen: X (Breite), Y (Höhe) und Z (Tiefe). Jeder dieser Achsen ist ein Wertebereich von -32768 bis 32767 zugeordnet. Zur Laufzeit erfolgt dann eine Umrechnung der virtuellen MHEG-Koordinaten in das physische Koordinatensystem der jeweiligen Präsentationsdienste (z.B. die Anzahl der Bildpunkte, die sich aus einem Motif-Fenster ergeben). Zusätzlich gibt es noch ein zeitliches Koordinatensystem, das aus einer Achse T besteht. Der definierte Wertebereich für diese Achse ist ein Intervall von 0 bis unendlich, wobei als Maßeinheit eine Millisekunde angenommen wird, was bezogen auf die Synchronisationsanforderungen in der Praxis ausreichend granular sein dürfte [25].

Im folgenden Beispiel wurde für einen referenzierten MPEG-Video-Inhalt ein entsprechendes Content-Objekt mit einer Größe von 256 x 240 Bildpunkte und originaler Aufnahmegeschwindigkeit kodiert.

```
Content-Class {
   MHEG-Identifier.Object-Number: 1,
   MHEG-Classification: Video,
   Hook {
      Encoding-Identification: ISO-1172-MPEG-Video,
      Encoding-Description: video rate in Kbps
   }
   External-Data.Logical-Name: “The Little Girl”,
   Original-Size : 256 pt, 240 pt, null,
   Original-Speed: yes
}
```

Virtuelle Sichten

Bis jetzt wurde davon ausgegangen, daß die Darstellung der Inhalte genau so erfolgt, wie diese ursprünglich einmal erzeugt wurden. Tatsächlich bietet MHEG eine Reihe von Möglichkeiten, um die Darstellung der Content-Objekte, sei es einen Film in Zeitraffer abzuspielen, die Lautstärke einer Audiosequenz einzustellen oder den Ausschnitt einer Grafik zubestimmen, durch geeignete Parameter zu steuern. Die Manipulation dieser Parameter wird durch entsprechende Befehle (vgl. Action-Klasse) in der Kodierung fest-

gelegt. Wie das folgende Beispiel eines computersimulierten Basketballspiels zeigt, kann die gleiche Spielfigur für jedes Team mehrfach an unterschiedlichen Positionen auf dem Spielfeld vorkommen. Anstatt alle möglichen Kombinationen der Darstellung jeweils als separate Content-Objekte zu speichern, wird die Veränderung der Parameter als Aufruf einer Methode auf das Objekt modelliert. Auf diese Weise entstehen zur Laufzeit sogenannte virtuelle Sichten (*Presentable*) zu einem einzelnen Content-Objekt. Diese virtuellen Sichten werden während der Komposition einer Präsentation durch eindeutige Nummern festgelegt und halten zur Laufzeit die Parameter entsprechend der Darstellung fest. In Abbildung 7 sind ein paar Verwendungsmöglichkeiten virtueller Sichten aufgeführt.

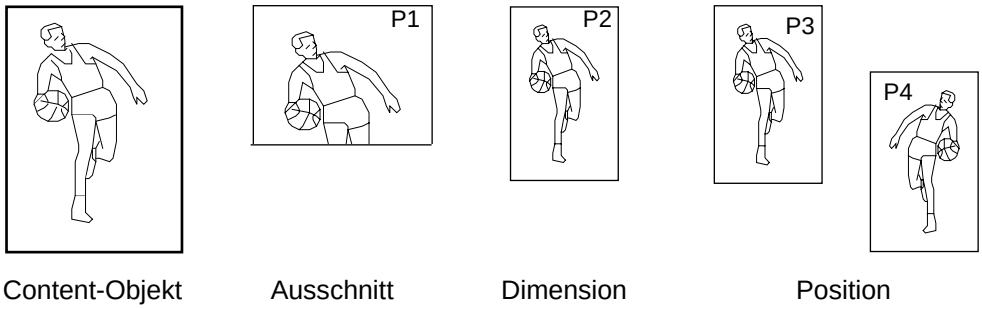

Abbildung 7: Beispiele virtueller Sichten

Derartige Sichten existieren sowohl für alle Component-Klassen als auch für Objekte der Selection-, Modifikation- und Composite-Klasse, wobei diese dort jeweils der Klasse entsprechend eine etwas andere Ausprägung haben. Wird im weiteren von MHEG-Objekten gesprochen, dann sind die virtuellen Sichten mit eingeschlossen. Nur im Falle der unterschiedlichen Behandlung werden diese explizit erwähnt.

3.3 Action-Klasse

Mit der Action-Klasse wird das Verhalten einzelner MHEG-Objekte bestimmt. In der Terminologie der Objektorientierung ist ein Action-Objekt eine Nachricht, die an ein MHEG-Objekt gesendet werden kann. Bei dem Zielobjekt wird dadurch eine entsprechende Methode aufgerufen, die eine Veränderung an diesem herbeiführt. In einem Action-Objekt wird jedoch nicht das Zielobjekt mit angegeben, so daß die Action-Klasse nur die Schnittstelle (Menge aller öffentlichen Methoden) zur Beeinflussung der Darstellung einzelner MHEG-Objekte repräsentiert. Mit der damit verbundenen Kapselung der Spezifika bleibt die Realisierung der einzelnen Methoden im Objekt verborgen. Desweiteren wird häufig die Eigenschaft des Polymorphismus verwendet, d.h. es gibt Aktionen mit gleichen Namen aber mit unterschiedlicher Ausprägung. Insgesamt führt dies zu einer sehr homogenen Schnittstelle (z.B. erfolgt das Anzeigen von Objekten unabhängig vom Medium immer mit der Aktion *run*).

Die einzelnen Aktionen können, je nachdem was für ein Verhalten sie beeinflussen, in verschiedene Gruppen eingeteilt werden. Auch wenn aus einem Action-Objekt nicht der Empfänger desselben hervorgeht kann man hier schon absehen, daß nicht jede Aktion für jedes MHEG-Objekt bzw. virtuelle Sicht einen Sinn ergibt. Abgeleitet aus dieser Tatsache kann aus diesen Gruppen im Falle einer objektorientierten Realisierung für diesen Standard eine weitere Klassenhierarchie entwickelt werden. Einen Überblick über die verschiedenen Aktionen auf MHEG-Objekte gibt Tabelle 3 bzw. auf virtuelle Sichten gibt Tabelle 3.

Gruppe	Aktionen
Availability	prepare*, destroy*
Returnability	return
Script activation	activate*, deactivate*
Multiplexing	multiplex, demultiplex

Tabelle 2: Aktionen auf MHEG-Objekte

Gruppe	Aktionen
Temporal presentability	run*, stop*, set_speed, set_opacity, set_temporal_position, set_timestones
Spatial presentability	set_size, set_box, set_scrolling, set_relative_attachment_point_position, set_aspect_ratio_preserved, set_attachment_point
Multiple generic spaces presentability	set channel
Audible presentability	set_audio_channel_volume
Modification	set_modifiability, set_result, set_initial_modification_presentation
Modification style presentability	set_modification_presentation_style, set_mdification_presentation_details
Selection	set_response, set_initial_value, set_number_of_allowed_selections
Selection style presentability	set_selection_presentation_styles, set_selection_presentation_details, associate_output_presentable
Composition	set composition_status*, set_presentation_piority

Tabelle 3: Aktionen auf virtuelle Sichten

Zustände und Zustandsübergänge

Einige Aktion bewirken an MHEG-Objekten Zustandsänderungen, die von besonderer Bedeutung (in Tabelle 2 und Tabelle 3 jeweils durch * gekennzeichnet) sind. Man erinnere sich an das eingangs vorgestellte Szenario, in dem mit der Beendigung der Anzeige der Grafik ein Text erscheinen sollte. Um eine derartige Beziehung zwischen der Darstellung der Grafik und des Textes herstellen zu können, wird der Zustand, in dem sich die virtuelle Sicht der Grafik befindet (wird angezeigt oder nicht), benötigt. Zu diesem Zweck werden im Standard die für die Definition von Abläufen wichtigen Zustandsänderungen durch Protokollautomaten definiert. Wie anhand der zwei exemplarischen Zustandsübergangsdiagramme aus Abbildung 8 dargestellt, sind diese wegen ihrer geringen Anzahl von Zuständen recht einfach geartet. Der *Preparation status* repräsentiert die Verfügbarkeit eines MHEG-Objektes. Am Beispiel eines Content-Objektes kann dies verdeutlicht werden. Zu Beginn ist das Objekt im Zustand *not_ready*. Mit der Aktion *prepare* werden eventuell referenzierte Daten lokalisiert und das zur Präsentation notwendige Subsystem initialisiert. Konnte dies alles erfolgreich durchgeführt werden, dann wechselt der Zustand auf *ready* und das Objekt kann von einer virtuellen Sicht angezeigt werden. Mit der Aktion *destroy* können die vom Objekt belegten Ressourcen wieder freigegeben werden. Ob diese Objekt anzeigt wird, hängt von dem *Presentation*

status einer zugehörigen virtuellen Sicht ab. Weitere Zustandsübergange sind für den *Composition status*, den *Modification status*, den *Timestone status* und den *Script activation status* definiert. Eine Ausnahme bildet der *Selection status*. Die verschiedenen Stati werden erst mit der Modellierung der einzelnen Benutzerinteraktionen in einer Präsentation bestimmt.

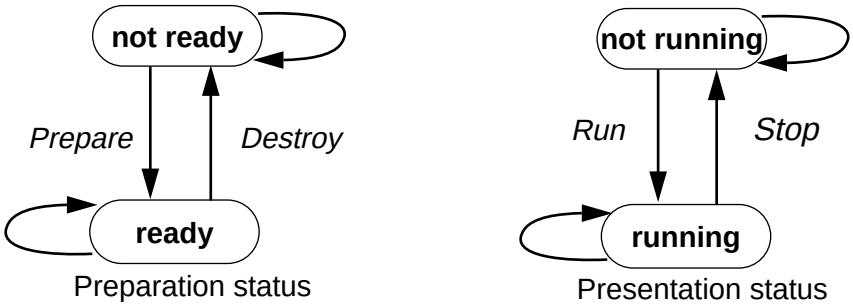

Abbildung 8: Beispiel zweier Zustandsübergangsdiagramme

Die zuvor beschrieben Aktionen sind atomare Operationen auf ein MHEG-Objekt. Der Aufbau eines Action-Objektes sieht aber auch vor, daß komplexe Abläufe durch das Zusammenfassen mehrerer atomarer Aktionen (*Action*) definiert werden können. Je nach Anforderung können die einzelnen Aktionen einer parallelen oder einer sequentiellen Verarbeitung zugeführt werden. Die Datenstruktur der Action-Klasse definiert zu diesem Zweck eine Liste *(Parallel-Action-List)*, deren Elemente sich wiederum aus einer Liste von atomaren Aktionen zusammensetzt (*Delayed-Sequential-Action* definiert als *Sequence of Action*). Die atomaren Aktionen in *Delayed-Sequential-Action* sind grundsätzlich seriell abzuarbeiten. Da der Standard nichts darüber aussagt, ob die einzelnen Aktionen synchron oder asynchron verarbeitet werden, kann die Reihenfolge von atomaren Aktionen mehrdeutig sein. Würde man z.B. nacheinander eine Aktion *run* und eine Aktion *stop* angeben, dann führt die tatsächliche Verarbeitung je nach Prozeßmodell zu unterschiedlichen Ergebnissen. Eine korrekte Kodierung für dieses Beispiel erhält man, wenn beide Aktionen in je einer separaten *Delayed_Sequential_Action* aufgeführt werden, wobei die für die Aktion *stop* ein Verzögerungswert angegeben wird. Ein graphischer Überblick über die Datenstruktur zeigt Abbildung 9.

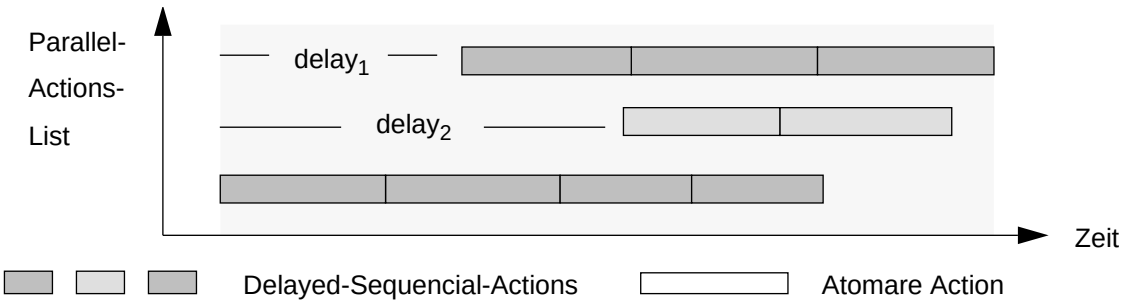

Abbildung 9: Beispiel eines Action-Objektes

Das folgende Action-Objekt setzt sich aus zwei *Delayed-Sequential-Actions* zusammen, die an eine virtuelle Sicht des zuvor kodierten Content-Objekts mit der Objektnummer `1` gesendet werden sollen. Die Erste beinhaltet drei Aktionen: eine zum Setzen der Position, eine zum Setzen der Größe des Videofensters und schließlich eine zum Anzeigen des Videos. Die Zweite stoppt das Video nach 5 Sekunden.

```
Action-Class {
   MHEG-Identifier.Object-Number: 2,
   Parallel-Action-List (
      Parallel-Actions (
         Delay: none
         Delayed-Sequencial-Action (
            Set-Spatial-Position: 250, 175,
            Set-Size: 140, 100,
            Run.Number_Of_Presentations: 1 ) )
      Parallel-Actions (
         Delay: 5 sec,
         Delayed-Sequencial-Action (
            Actions ( Stop ) ) )
}
```

3.4 Link-Klasse

Soweit konnten noch keine Beziehungen, die den Ablauf der einer Präsentation festlegen, zwischen einem Action-Objekt und einem Content-Objekt bzw. einer virtuellen Sicht hergestellt werden. Zwar wäre es prinzipiell möglich diese Beziehungen mittels einer Programmiersprache auszudrücken, allerdings verbunden mit dem Nachteil, daß diese Information ein Teil des jeweiligen Programms ist und somit eine Abhängigkeit zwischen der Präsentation und dem Programm bestehen würde. Die Link-Klasse hat nun zwei Aufgaben zu erfüllen. Zum einen wird definiert welche Aktionen an welche MHEG-Objekte geschickt werden sollen. Zum anderen werden die Bedingungen festgelegt unter denen dies erfolgen soll. Die Link-Klasse spezifiziert somit den geplanten Ablauf. Daraus läßt sich ableiten, daß der Verarbeitung einer MHEG-kodierten Präsentation ein ereignisorientiertes Verarbeitungsmodell zugrunde liegt.

An dieser Stelle erinnere man sich daran, daß in Multimedia-Präsentationen verstärkt Parallelität berücksichtigt werden muß. Skriptsprachen legen Abläufe meist in Form von sequentiellen Befehlsfolgen fest (prozedurale Sprache), d.h. der Autor ist selbst für die Serialisierung verantwortlich. Dies kann in anspruchsvollen Präsentationen mit kontinuierlichen Medien leicht zu einer beliebig komplexen Beschreibung führen. Anstatt den Ablauf entlang einer virtuellen Zeitachse zu definieren, könnte man in Zukunft den Autor durch eine deklarative Sprache von diesem Problem entbinden. Die ereignisorientierte Verarbeitung der Link-Objekte bietet sich hervorragend für die Entwicklung einer Sprache an, die den Ablauf über Regeln (wenn..., dann...) definiert. Die Serialisierung bleibt in diesem Fall dem System überlassen.

Die Semantik eines Link-Objektes ist in Abbildung 10 schematisch dargestellt. Ein Link-Objekt verbindet immer genau ein Quellenobjekt mit einem oder mehreren Zielobjekten. Die Quellen- bzw. Zielobjekte können sowohl MHEG-Objekte als auch virtuelle Sichten sein. Die Ausführung eines Link-Objektes ist immer abhängig von einer Bedingung (*Trigger-Condition*), die durch eine mögliche Zustandsänderung im Quellenobjekt ausgedrückt werden kann. Ist die Bedingungen erfüllt, dann werden die in dem Link-Objekt angegebenen Action-Objekte an alle Zielobjekte gesendet. Bei der Komposition einer Präsentation muß man darauf achten, daß es sich hierbei um eine Liste von Action-Objekten handelt, die laut Standard parallel abgearbeitet werden kann. Obwohl der Standard nichts über die Implementierung aussagt, muß an dieser Stelle davon ausgegangen werden, daß aus Gründen der Effizienz die Bedingung eines Link-Objektes immer genau dann überprüft wird, wenn eine Zustandsänderung an dem jeweiligen Quellenobjekt eingetreten ist. Der *Attachment Point* wird verwendet, um Zielobjekte relativ zum Quellenobjekt zu positionieren, d.h. Koordinaten in einem Action-Objekt beziehen sich auf die Position des Zielobjektes.

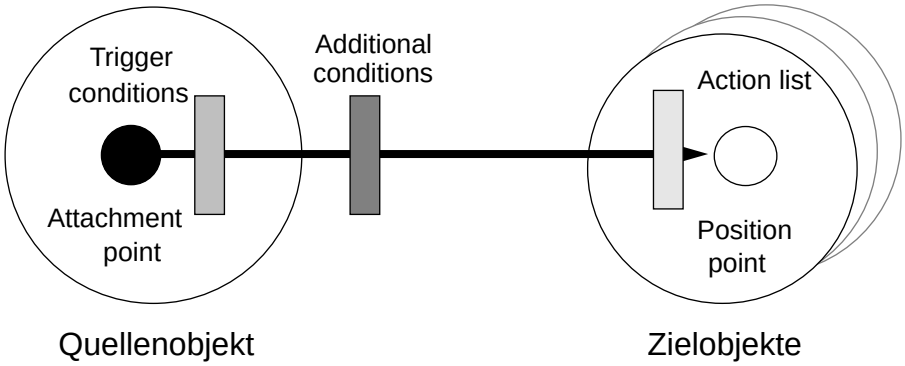

Abbildung 10: Aufbau eines Link-Objektes

Nicht alle Beziehungen lassen sich durch Bedingungen, die an genau ein Quellenobjekt geknüpft sind, ausdrücken. Als Beispiel nehme man an, daß ein Bild angezeigt werden soll, nachdem sowohl eine Audiosequenz als auch die dazu parallel verlaufene Videosequenz beendet wurden. In diesem Fall handelt es sich um eine Konjunktion (logisches "und") zweier Bedingungen, die an verschiedene Quellenobjekte geknüpft sind. Dies kann in einem Link-Objekt, daß ja eigentlich nur genau ein Quellenobjekt kennt, durch die Definition zusätzlicher Bedingungen (*Additional-Condition*s) erreicht werden. Diese werden aber nur geprüft, wenn die auslösende Bedingung an dem eigentlichen Quellenobjekt erfüllt ist. Nun stellt sich die Frage, ob das Audioobjekt zum Quellenobjekt wird und das Videoobjekt in der zusätzlichen Bedingung berücksichtigt wird bzw. umgekehrt. Kann man in dem angenommenen Beispiel nicht vorhersagen, ob die Videosequenz oder die Audiosequenz zuerst beendet sein wird, dann müssen zwei Link-Objekte definiert werden. Das eine verwendet das Audioobjekt für die *Trigger-Condition* und das Videoobjekt in der *Additional-Condition*. Für das andere werden die Rollen gerade vertauscht.

Das folgende Link-Objekt kodiert die Bedingung: "wenn der Presentation-Status im Quellenobjekt 10 von *not-running* auf *running* gewechselt hat, dann sende das Action-Objekt 2 zu den Zielobjekten 11 und 12".

```
Link-Class {
   MHEG-Identifier.Object-Number: 3,
   Trigger-Condition {
      Source-Reference.Object-Number: 10,
      Status-Identifier: presentation_status
      Previous-Evaluation.Presentation-Status: not-running,
      Current-Evaluation.Presentation-Status: running
   }
   Additional-Conditions { ... }
   Action Set {
      Targets: (
        Object-Number: 11, Object-Number: 12 )
      Actions: (Action-Object-Referenced: Object-Number: 2)
   }
}
```

Seit der Abstimmung zum CD-Dokument hat sich ergeben, daß neben der Konjunktion zwischen der *Trigger-Condition* und der *Additional-Condition* auch komplexere boolsche Ausdrücke zugelassen werden müssen. So kann in Zukunft das Feuern eines Links z.B. auch in Abhängigkeit der Lautstärke erfolgen.

3.5 Script-Klasse

Eine weitere Möglichkeit das Verhalten von Objekten oder den Ablauf einer Präsentation zu bestimmen, bietet die Script-Klasse. Diese wurde vorgesehen, um aus einer MHEG-Präsentation gezielt andere Laufzeitumgebungen (z.B. Script/X), externe Programme oder Funktionen aufrufen zu können. Vergleichbar der Content-Klasse werden auch hier verschiedene entweder genormte (*MHEG-Catalogue*) oder proprietäre (*Non-MHEG-Catalogue*) Sprachen unterstützt. Die Script-Klasse ist in der vorliegenden Version des Standards als eine notwendige Funktionalität identifiziert worden. Allerdings sind noch einige Punkte nicht abschließend geklärt (z.B. wie Verhalten sich MHEG-Objekte zu einem Skript, können sie von diesem manipuliert werden oder umgekehrt).

Das folgende Script-Objekt veranlaßt die Ausführung einer Script/X-Präsentation.

```
Script-Class {,
   MHEG-Identifier.Object-Number: 20,
   Scripting-Language {
      Scripting-Language-Identification.MHEG-Catalogue: Script/X
   }
   Script: "representation of a script"
}
```

3.6 Selection-Klasse

Mit den bis jetzt vorgestellten Klassen können Präsentationen zusammengestellt werden, die nach dem Start selbständig ablaufen, d.h. während des Ablaufs besteht keine weitere Möglichkeit des Eingriffs durch einen Betrachter. Interaktive Präsentationen verlangen gerade, daß der Betrachter den Ablauf in einem vordefinierten Rahmen steuern kann. Das einführende Beispiel hat gezeigt, daß im MHEG-Standard zwei Arten der Benutzerinteraktion unterschieden werden. Die Selection-Klasse bietet die Möglichkeit, eine Interaktion als eine Auswahl (Selektion) eines Wertes aus einem vordefinierten Wertebereich zu modellieren. Typische Beispiele sind Verzweigungen im Ablauf “...wählen sie eines der Kapitel...” oder das Vor- und Zurückblättern in aufeinanderfolgenden Seiten.

Die Erläuterungen zur Link-Klasse haben gezeigt, daß der Ablauf einer Präsentation durch das Eintreten von Ereignissen, die vom Standard vorgegeben waren, gesteuert wird. Mit einem Selection-Objekt ist es nun möglich, Benutzerinteraktionen ebenfalls in Form solcher Ereignisse zu berücksichtigen. Die Definition dieser anwendungsdefinierten Ereignisse übernimmt ein Selection-Objekt, indem für die jeweils erwarteten Auswahlmöglichkeiten ein entsprechender Ergebniswert vorgesehen wird. Genauso wie zu einem Content-Objekt gibt es zum Zeitpunkt der Benutzerinteraktion eine virtuelle Sicht auf das Selection-Objekt (*Selection-Presentable*). In dieser wird das Ergebnis der Benutzerinteraktion durch die Zuweisung des vorgesehenen Ergebniswertes in einem dafür vorgesehenen Statusfeld (*selection status)* abgelegt. Die Änderung dieses Statusfeldes sollte dazu führen, daß die Bedingung eines Link-Objektes erfüllt ist und somit Aktionen an andere Objekte gesendet werden.

Das folgende Selection-Objekt definiert eine Selektion mit fünf möglichen Rückgabewerten (*Response*). Desweiteren ist zu erkennen, daß durch die zulässige Rekursion (*Included-Group.Selection-Class*) ohne weiteres auch hierarchische Selektionen (z.B. Menüs) aufgebaut werden können.

```
Selection-Class {
   MHEG-Identifier.Object-Number: 4
   Response: 10
   Elements (
      { Element-Index: 1, Response: 11 },
      { Element-Index: 2,
         Included-Group.Selection-Class {
            Response: 12,
            Elements (
               { Element-Index: 1, Response: 21 },
               { Element-Index: 2, Response: 22 },
               { Element-Index: 3,
                  Included-Group.Selection-Class {
                  Response: 23,
                  Elements (
                     { Element-Index: 1, Response: 31 },
                     { Element-Index: 2, Response: 32 } )
               } } )
   } } )
}
```

Die Veränderung des Statusfeldes, also das Anzeigen und Steuern von geeigneten Primitiven (im Standard vorgesehen sind: *menu*, *pull-down-menu*, *pop-up-menu*, *button_list*, *key_list*, *device_list*, *scrolling_list*, *switch, item*, *button*, *key* und *device*) auf der Benutzeroberfläche, obliegt den sogenannten Präsentationsdiensten (z.B. X-Motif). Mit der Kodierung eines Selection-Objektes wird auch nicht festgelegt, welche Art von Primitiven für die Interaktion verwendet werden sollen. Die Primitive und ihre Eigenschaften (z.B. Beschriftung eines Knopfes) werden über Aktionen (siehe *selection style presentability*) gesetzt, die an die jeweiligen *Selection-Presentables* gesendet werden.

Mit dem folgenden Action-Objekt wird die Darstellung des Selection-Objektes als Pull-Down-Menü bestimmt. Über das Attribut *Associate-Output-Presentable* wird die Gestalt der Menüpunkte unter der Rubrik *Paint* durch entsprechende *Content-Presentables* gesetzt. In diesem Beispiel wurden zwei Pixelgrafiken als Symbol für einen feinen und einen breiten Strich gewählt. Das Ergebnis der abschließenden *run* Aktion ist in Abbildung 11 dargestellt.

```
Action-Class {
   ....
   Actions (
      Selection-Style-Presentability (
         Set-Selection-Presentation-Style: pull-down-menu,
         Set-Selection-Presentation-Details (
            { Item-Identification (0), Orientation: horizontal },
            { Item-Identification (1), Label: ”FILE” },
            { Item-Identification (2), Label: ”EDIT”, Orientation: vertical},
            { Item-Identification (2,1), Label: ”CUT” },
            { Item-Identification (2,2), Label: ”PASTE” },
            { Item-Identification (2,3), Label: ”PAINT”, Orientation: vertical} ),
         Associated-Output-Presentable (
            { Item-Identification (2,3,1), Presentable: 100 },
            { Item-Identification (2,3,2), Presentable: 101 } )
      Selection.Set-Initial-Value (
         { Item-Identification (2), selected }, ... ) )
      Run: Number-Of-Presentations: once )
}
```

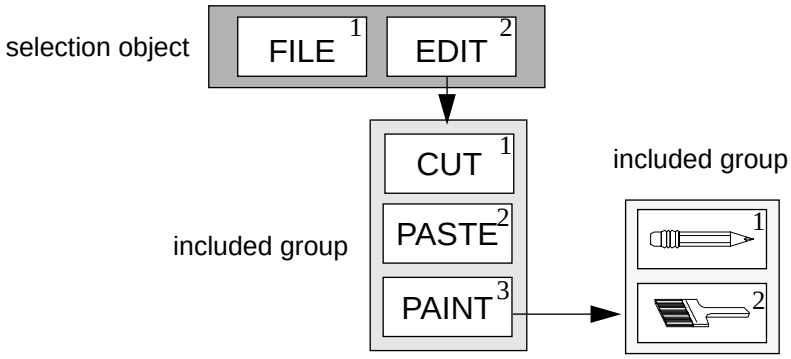

Abbildung 11: Beispiel eines Pull-Down-Menüs

3.7 Modification-Klasse

Die zweite Form der Benutzerinteraktion dient der Eingabe und Manipulation von Daten. Im Gegensatz zu einem Selection-Objekt wird bei einem Modification-Objekt kein Wertebereich vorgegeben sondern das Ergebnis einer Interaktion wird durch ein angegebenes Content-Objekt repräsentiert. Mit der Darstellung eines entsprechenden *Modification-Presentables* wird über bereitgestellte Primitive der Benutzeroberfläche das Content-Objekt manipuliert. Auch in diesen virtuellen Sichten wird in einem Statusfeld (*modification status*) der aktuelle Bearbeitungszustand des Content-Objektes vor (*modifyable*), während (*modifying*) und nach (*modified*) der Modifikation festgehalten. Damit ist es wie bei der Selektion möglich, über ein Link-Objekt das Statusfeld abzufragen und dadurch den Ablauf einer Präsentation zu steuern.

Während des internationalen Abstimmungsprozesses wurde festgestellt, daß die ASN.1-Kodierung im CD-Dokument unvollständig ist. Genaugenommen handelt es sich nämlich bei dem Content-Objekt um eine Variable. Der Wert dieser Variablen kann wiederum als Parameter für eine Aktion dienen (z.B. Lautstärke bei *Set_Volume*). In der vorliegenden ASN.1-Kodierung ist eine Referenzierung dieser Variablen für Action-Objekte nicht möglich. Der Ursprung dieses Problems geht auf Abgrenzungsprobleme zwischen MHEG und anderen Normungsbestrebungen zurück. MHEG beschreibt Präsentationen in endgültiger Form, während die Aktivitäten im Unterausschuß SC18 die editierbare Form betrachten. Dort wird parallel zu MHEG eine Skriptsprache entwickelt. Da man MHEG nicht als Programmiersprache verstanden haben möchte, wird in der MHEG-Terminologie der Begriff “Variable” strikt vermieden. Für die DIS-Version werden deshalb sogenannte *Generic_Values* definiert. Diese können je nach Typ, vorgesehen sind *Boolean*, *Character String*, *Numeric Value*, *Numeric Vector*, *Spatial Vector*, *Temporal* und *Volume*, das Ergebnis einer Modifikation enthalten und von Action-Objekten weiterverwendet werden. In manchen Fällen (z.B. Ausfüllen eines Formulars) wird es notwendig sein, die Werte an die Anwendung zurückzugeben. Dies kann über die Action-Klasse *returnability* gelöst werden.

In dem folgenden Modification-Objekt wird das Content-Objekt mit der Object-Number 7 referenziert. Für dieses Objekt wurde die MHEG-Classification *numeric* gewählt, so daß in diesem Objekt numerische Werte gespeichert werden können. Da der Wert des Attributes *Modifiable* auf *true* gesetzt ist, bedeutet dies hier, daß der im Objekt gespeicherte Wert 4 als Initialwert verwendet wird. Wäre der Wert auf *false* gesetzt, dann würde man ein komplett neues Objekt erzeugen.

```
Modification-Class {
   MHEG-Identifier.Object-Number: 5,
   Modifiable: true,
   Result.Content-Object-Referenced.Object-Number: 7
}
```

Auch für die Modifikation-Klasse gilt, daß die Darstellung ihrer virtuellen Sichten über speziell dafür vorgesehene Aktionen (*modification-style-presentability*) gesteuert wird. Es ist zwar nicht zwingend notwendig, daß das jeweilige Content-Objekt numerische Werte speichert. Allerdings lassen die im Standard vorgesehenen Eingabeprimitive (*slider, joystick, acquisition_area, arrows, numeric_key, alphanumeric_key, designator* und *device*) darauf schließen.

Das folgende erste Action-Objekt wird an das *Content-Presentable* mit der *Object-Number* `100` geschickt. Das zweite Action-Objekt geht an ein *Modification-Presentable* mit dem Ziel die Verbindung zum *Content-Presentable* herzustellen und es als Schieberegler anzuzeigen, wie in Abbildung 12 dargestellt.

```
Action-Class {
   ...
   Actions (
      Moditification-Presentability (
         Set-Modification-Presentation-Style: slider,
         Set-Modification-Presentation-Details {
            Minimum: 1, Maximum: 8, Granularity: 1, Label="Lautstärke"
         } ) )
}

Action-Class {
   ...
   Actions (
      Modification.Set-initial-modification-presentation (
         Object-Number: 100 ),
      Run: Number-Of-Presentations: once )
}
```

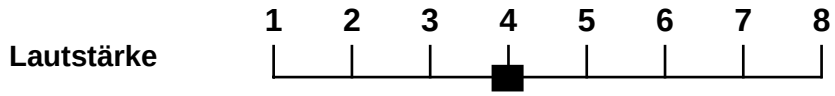

Abbildung 12: Beispiel eines Schiebereglers

3.8 Composite-Klasse

Die Composite-Klasse hat zur Aufgabe, die notwendigen Objekte aus den zuvor beschriebenen Klassen zu einer Präsentation zusammenzufügen. Unabhängig davon, ob die einzelnen Objekte inkludiert oder referenziert werden, verhält sich jedes Composite-Objekt als eine Art “Container”, d.h. es stellt eine in sich abgeschlossene Einheit für den Austausch von Präsentationen zwischen den Systemen dar. Desweiteren bietet die Funktionalität eines Composite-Objektes einige Voraussetzungen für die synchronisierte Wiedergabe desselben durch die MHEG-Engine. Durch Verweise (vgl. Hypertext-Links) zwischen verschiedenen Composite-Objekten untereinander können beliebig komplexe Präsentationen entstehen. An dieser Stelle wird noch einmal die Bedeutung der abstrakten Klassen *Behaviour* und *Component* aus der Klassenhierarchie (siehe Abbildung 6) deutlich. Zusammen mit den hier definierten virtuellen Sichten (*Presentables*) und den beiden Link-Objekten (*Container Start-up Link, Presentation Start-up Link*) bilden diese die in Abbildung 13 grafisch dargestellte Struktur eines Composite-Objektes.

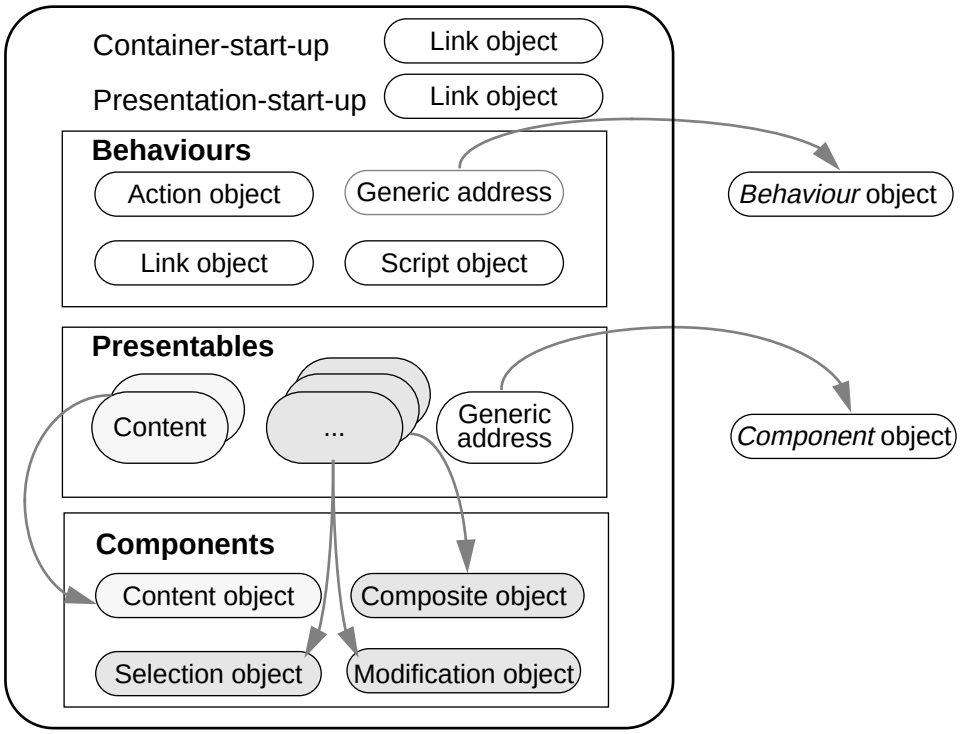

Abbildung 13: Aufbau eines Composite-Objektes

Bevor auf die einzelnen Bestandteile eingegangen wird sei angemerkt, daß die Ausführung jedes Composite-Objektes, wie bei allen anderen MHEG-Objekt, in zwei Phasen abläuft. Die erste Phase, ausgelöst durch die Aktion *prepare*, initialisiert das Composite-Objekt, d.h. es werden das Objekt selbst, die beiden Start-up-Link-Objekte und die enthaltenen virtuellen Sichten registriert. Die eigentliche Ausführung der Präsentation erfolgt mit der Aktion *run* auf ein *Composite-Presentable*. Diese "Zweiphasigkeit" spiegelt sich auch am Aufbau eines Composite-Objektes in den beiden Start-Up-Link-Objekten wieder. Der *Container-start-up-Link* dient der Initialisierung der im Composite-Objekt enthaltenen *Components*, indem eventuell für die Wiedergabe notwendige Ressourcen im ausführenden System aktiviert bzw. reserviert werden, um z.B. Verzögerungen, bedingt durch eventuell notwendiges "Nachladen" ausgelagerter Teile, zu vermieden. Dazu hat dieses Link-Objekt per Definition immer die gleiche Bedingung "wenn der *Preparation-status* des zugehörigen Composite-Objektes von *not ready* auf *ready* wechselt, dann...". Was dann im einzelnen geschehen soll bleibt allerdings in der Verantwortung der Anwendung. Auch der *Presentation-start-up-Link* hat eine vordefinierte Bedingung: "wenn der *Presentation-status* eines *Composite-Presentables* von *not running* auf *running* wechselt, dann...". Diese Bedingung dient der Initiierung der eigentlichen Präsentation, z.B. durch das Starten einer im Composite-Objekt definierten virtuellen Sicht. Wird an dieser Stelle keine Kodierung angegeben, dann wird per Definition eine Aktion *run* an die erste virtuelle Sicht im Composite-Objekt gesendet.

Die Struktur *Behaviour* umfaßt alle MHEG-Objekte, die der Vorbereitung von Component-Objekten, der Definition des Ablaufs und der Beziehungen zwischen den virtuellen Sichten dienen. Im einzelnen werden dazu Action-, Link- und Script-Objekten herangezogen. Die Behaviour-Objekte können sowohl inkludiert (*included*) als auch referenziert (*referenced*) werden.

Die schon mehrfach angesprochenen virtuellen Sichten werden in der Struktur *Presentables* definiert. Aus der Sicht der Kodierung wird jede virtuelle Sicht als ein im Composite-Objekt eindeutiger ganzzahliger Wert und einem Verweis auf ein Component-Objekt dargestellt. In der Laufzeitumgebung wird dieser Wert auf eine systemabhängige Instanz eines entsprechenden Primitivs zur Darstellung des Objektes (z.B. *WidgetId*, *DeviceHandle*) abgebildet. Die Verweise (ebenfalls Indizes) zeigen in der Regel auf Objekte in der Struktur *Components*. Bezieht sich eine virtuelle Sicht auf eine *Generic-Address*, dann handelt es sich hierbei immer um ein referenziertes Component-Objekt. Dies ist z.B. dann sinnvoll, wenn durch die Verknüpfung mehrerer Composite-Objekte eine hypertextartige Struktur aufgebaut werden sollen.

Alle MHEG-Objekte, die in der Struktur *Components* aufgelistet werden, sind in dem Composite-Objekt physisch enthalten und mit einem eindeutigen Index versehen. Verweise zu MHEG-Objekten außerhalb des jeweiligen Composite-Objektes sind aus Gründen der Synchronisation nicht zulässig.

In dem folgenden Composite-Objekt sind ein Content-Objekt und ein weiteres Composite-Objekt enthalten. Das *Presentable* mit dem Index 1 zeigt auf das Content-Objekt, während das Presentable mit dem Index 2 auf das Composite-Objekt zeigt. Desweiteren gehört zu diesem Composite-Objekt ein referenziertes Link-Objekt und ein ebenso referenziertes Action-Objekt.

```
Composite-Class {
   MHEG-Identifier.Object-Number: 6
   Container-Start-Up.Link {...}
   Presentation-Start-Up.Link {...}
   Behaviour {
      Link-Object-Referenced.Object-Number: 2,
      Action-Object-Referenced.Object-Number: 3
   }
   Presentables (
      Presentable {
         Presentable-Index: 1
         Component-Reference.Single-Tail: 10
      }
      Presentable {
         Presentable-Index: 2,
         Component-Reference.Single-Tail: 11
      }
   }
   Components (
      Component {
         Component-Index: 10,
         Included-Component.Content-Class {..}
      }
      Component {
         Component-Index: 11,
         Included-Component.Composite-Class {...}
      } )
}
```

Seitens des MHEG-Standards gibt es keine Voraussetzung bzgl. der Granularität von einzelnen Composite-Objekten. Ein Composite-Objekt kann sich z.B. auf eine einzelne Benutzerinteraktion als Teil einer Anwendung beziehen, oder es kann eine vollständige Animation mit allem was dazugehört beschreiben. Die Granularität wird in der Praxis letztenendes von der jeweiligen Anwendung bestimmt. So ist es denkbar, daß der für eine Kioskanwendung typische seitenorientierte Aufbau des Ablaufs zur Definition von je einem Composite-Objekt pro Seite führt (vgl. Modularisierung).

Im Zusammenhang mit der Beschreibung der Content-Klasse wurde schon angedeutet, daß der MHEG-Standard zwei Möglichkeiten der Referenzierung vorsieht. Zum einen durch die physische Inklusion von Objekten innerhalb des jeweiligen Objektes und zum anderen durch logische Verweise, die zur Laufzeit aufgelöst werden müssen. Im folgenden werden nun die verschiedenen Adressierungstechniken der Klasse *Generic-Address* im einzelnen vorgestellt.

Adressierung über *Head*

Die Adressierung über *Head* dient der Adressierung von MHEG-Objekten, die für sich selbständig existieren können und nicht von anderen MHEG-Objekten abhängig sind. Dazu müssen sie auf eine der folgenden drei Arten adressierbar sein.

Bis jetzt wurde in den vorangegangenen Ausführungen die Möglichkeit über den *MHEG-Identifier* angewendet. Dieser setzt sich bekanntlich aus einem optionalen Anwendungsidentifikator (*application identifier*) und einer innerhalb dieser Anwendung eindeutigen Objektnummer (*object number*) zusammen. Im Gegensatz zu den folgenden beiden Arten kann der MHEG-Identifier schon zum Zeitpunkt der Kodierung eines MHEG-Objektes mit angegeben und in dem dafür vorgesehenen Attribut gespeichert werden.

Logische Namen (*Logical-Name*) sind alphanumerische Zeichenketten, welche als Substitut für ein beliebiges, evtl. systemabhängiges Adressierungsformat herangezogen werden können. Da ein Logischer Name nicht in einem MHEG-Objekt kodiert werden kann, ist es Aufgabe der Anwendung diesen auf das entsprechende MHEG-Objekt abzubilden.

External Identifier sollen im Gegensatz zu den Logischen Namen den Vereinbarungen aus dem ISO 8879 [8] und dem ISO 9070 [9] Standard unterliegen. Beide Angaben sind optional, d.h es kann zwischen einem *Public-Identifier*, einem *System-Identifier* oder einer Kombination von beiden gewählt werden.

Adressierung über *Tail*

Die einzelnen MHEG-Objekte inkludieren bisweilen mehrere andere MHEG-Objekte (z.B. neben dem Composite-Objekt inkludiert kann auch das Link-Objekt Action-Objekte inkludieren). Mit der Adressierung über *Tail* können die inkludierten Objekte in Abhängigkeit zu dem inkludierenden Objekt angesprochen werden. Auch hier werden drei Arten der Adressierung angeboten.

Single Tail stellt einen Pfad in Form einer hierarchisch aufgebauten Liste von einem äußeren Objekt zu einem inkludierten Objekt dar. In einem Composite-Objekt (äußeres Objekt) werden so z.B. die einzelnen inkludierten *Components* und *Presentables* angesprochen, oder im Falle der Selection-Klasse werden die einzelnen *Selection-Items* adressiert. Der zugehörige Pfad für die Adressierung des in Abbildung 14 hellgrau schraffierten Objektes über das dunkelgrau schraffierte Composite-Objekt lautet `1.2`.

Sibling Tail beschreibt einen Pfad von einem äußeren Objekt zu einer Menge von direkten Nachfolgern des adressierten Objektes. Als Beispiel sei auf das Pull-Down-Menü in Abbildung 11 verwiesen. Gibt man dort als *Sibling Tail* `2` an, dann werden die *Selection-Items* `(2,1)`, `(2,2)` und `(2,3)` adressiert.

Descendant Tail beinhaltet einen Pfad von einem äußeren Objekt zu allen Nachfolgern des adressierten Objektes. Angewandt auf das vorherige Beispiel adressiert *Descendant Tail* 2 zusätzlich noch die *Selection-Items* (2,3,1) und (2,3,2).

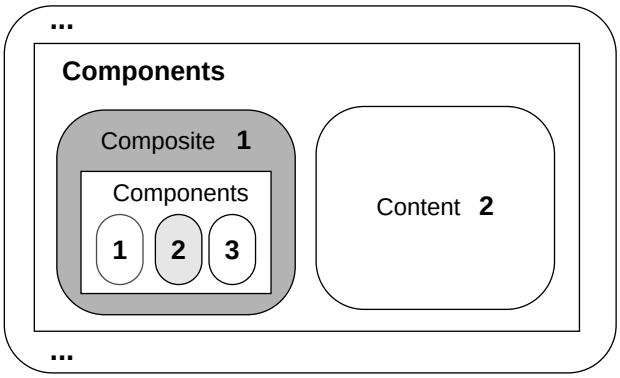

Abbildung 14: Beispiel einer Adressierung über *Single tail*

Schließlich steht auch noch die Möglichkeit ein Objekt aus einer Kombination von *Head* und *Tail* zu adressieren zur Verfügung. Die Möglichkeiten der Adressierung gestalten sich somit recht vielseitig, so daß hier auf die vollständig Aufführung der verschiedenen Kombinationen innerhalb der einzelnen MHEG-Klassen verzichtet wird.

4 MHEG-Laufzeitumgebung

4.1 Lebenszyklus der MHEG-Objekte

Der Lebenszyklus von MHEG-Objekten läßt sich entsprechend Abbildung 15 anhand der am Informationsfluß beteiligten Prozesse darstellen.

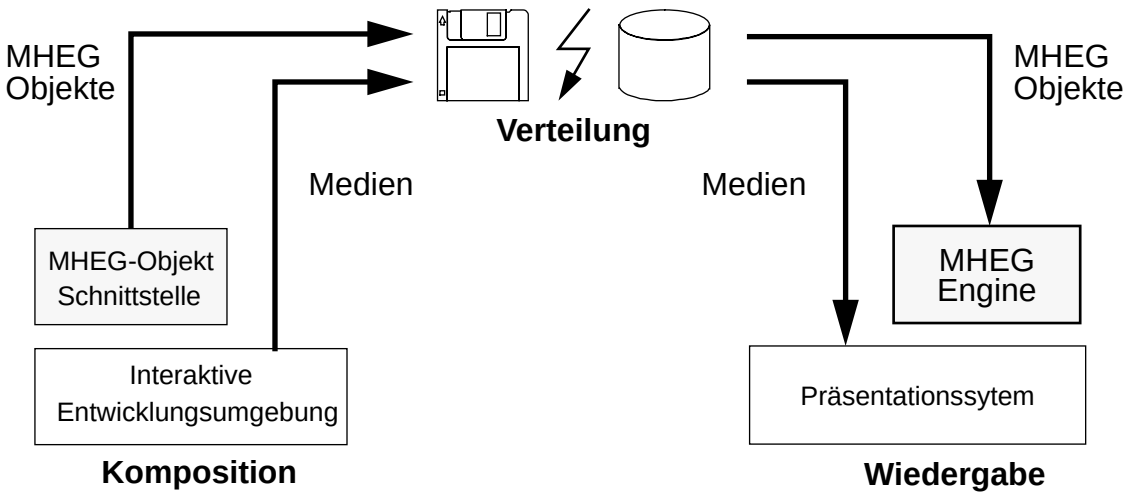

Abbildung 15: Informationsfluß

Komposition

Die Erstellung einer Präsentation durch Kodierung entsprechender MHEG-Objekte wird hier Komposition genannt. Der bevorzugte Weg ergibt sich durch den Einsatz von den schon heute am Markt zahlreich vertretenen Entwicklungswerkzeugen für Autorensysteme (z.B. *Macromind Director*). Diese bedienen sich in der Regel einer Skriptsprache (z.B. Lingo, AVA/2, GEL), die den Anwender durch eine am Anwendungsgebiet orientierte Semantik unterstützten. Auch auf Editoren aus dem Bereich der Hypertext/ Hypermedia-Systeme (z.B. *HyperCard*) und der Textverarbeitungssysteme (z.B. FrameMaker, ODA) kann zurückgegriffen werden. Die Dokumente bzw. Präsentationen, die auf diese Weise erstellt werden, müssen schließlich mit Hilfe eines Konverters oder Übersetzers in entsprechende MHEG-Objekte überführt werden (vgl. Abbildung 1). Eine Komposition auf der Ebene der MHEG-Objekte wird dagegen nicht für sinnvoll erachtet, da die Struktur der MHEG-Klassen zu stark an dem Verarbeitungsmodell orientiert sind. Etwas salopp formuliert würde bei dieser Vorgehensweise ein Autor auf der Ebene eines "Multimedia-Assemblers" arbeiten.

Verteilung

Die Verteilung von MHEG kodierten Informationen kann durch die bekannten Formen des Informationsaustauschs realisiert werden. Zum Beispiel in Form einer Online-Recherche in einer Datenbank, durch den Transport auf Datenträgern (CD-ROM, Disketten) oder als nicht persistente Nachrichten in kooperativen Anwendungen. Für alle Arten der Verteilung spielt MHEG aufgrund der Annahme einer heterogenen Umgebung die Rolle eines gemeinsamen Austauschformates zwischen den Systemen.

Präsentation

Die Präsentation der Informationen erfolgt schließlich mit Hilfe einer geeigneten Laufzeitumgebung (*MHEG-Engine*) in einem Präsentationssystem. Der Benutzer konsumiert die ihm gebotene Information ohne sie im allgemeinen verändern zu können. Die Möglichkeiten der Interaktion gestatten lediglich eine Anpassung der Informationsaufnahme nach individuellen Gesichtspunkten im Rahmen der vorgegebenen Ablaufstruktur. Eine Ausnahme besteht darin, wenn sich der Informationskreislauf hier wieder schließt, d.h. wenn auf dem Endgerät geeignete Werkzeuge vorhanden sind, die eine interaktive Weiterverarbeitung der Präsentation erlauben.

4.2 Analogie der Laufzeitumgebung

Eingangs wurde eine erste Analogie zwischen MHEG und Postscript gezogen. Zum besseren Verständnis der Verarbeitung von MHEG-Objekten bietet sich eine weitere, in Abbildung 16 dargestellte Analogie an. In dieser wird der Prozeß vom Quelltext zum ausführbaren Programm bei Object-Code-Compilern, UCSD-Pascal-Compilern und in Multimedia-Entwicklungsumgebungen verglichen.

Objekt-Code-Übersetzer

Objekt-Code-Übersetzer stellen heute eine weitverbreitete und allseits akzeptierte Technik bereit, die aus Quelltext von Programmiersprachen (C, C++, Cobol,...) ausführbare Programme erzeugen. In einer ersten Phase wird mit Hilfe eines herkömmlichen Editors oder einem interaktiven Entwicklungswerkzeug (Code-Generator) Quelltext erstellt. Dieser Quelltext wird anschließend von einem entsprechenden Übersetzer

(*Compiler*) eingelesen und in aufeinanderfolgenden Schritten lexikalisch, syntaktisch und semantisch analysiert. Als Ergebnis des Übersetzungsvorgangs liegt schließlich sogenannter Objekt-Code (auch *Binär-Code* genannt) aus prozessorabhängigen Maschinenbefehlen vor. Ein Binder (*Linker*) stellt aus dem Objekt-Code und notwendigen Bibliotheken (*Libraries*) ein ausführbares Programm zusammen.

Als Vorteil dieser Technik kann das sehr effiziente Laufzeitverhalten derartig erstellter Programme angeführt werden. Dagegen ist die Abhängigkeit des Objekt-Codes vom Mikroprozessor aus der Sicht der Austauschbarkeit in einer heterogenen Systemumgebung von Nachteil. Ein Austausch der Programme ist somit nur auf der Ebene des Quelltextes sinnvoll. Hierzu sei angemerkt, daß Softwarehersteller normalerweise den Quelltext ihrer Programme nicht der Öffentlichkeit zur Verfügung stellen. Die Anwender wiederum würden die oft beschworene “Binär-Code-Kompatibilität” von Programmen begrüßen.

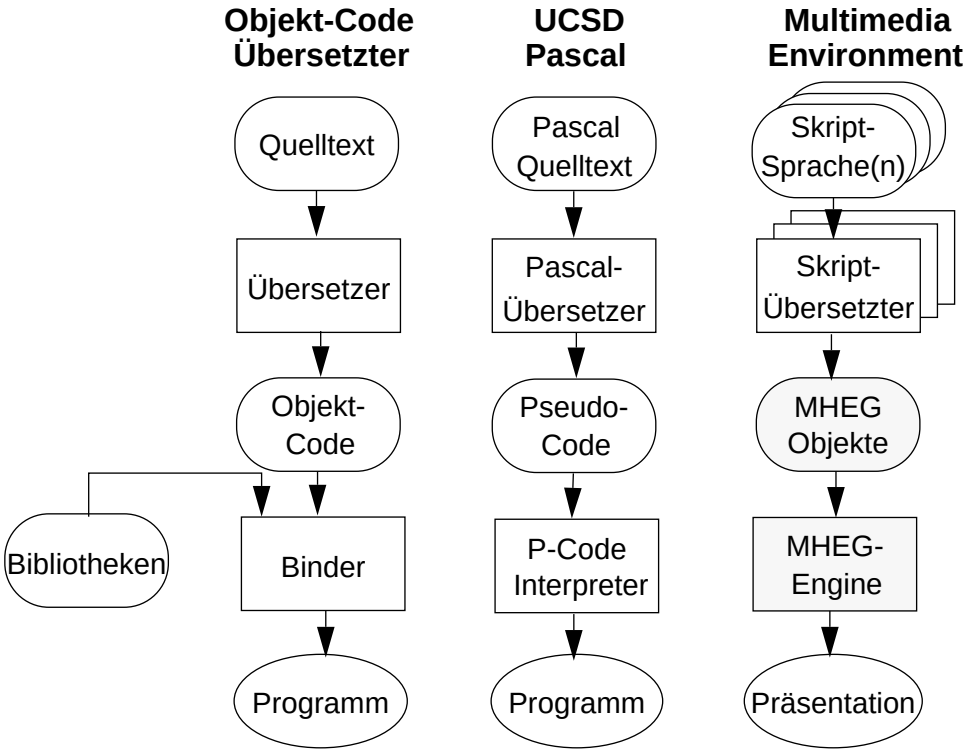

Abbildung 16: Analogie von MHEG zu Pseudo-Code

UCSD-Pascal-Übersetzer

Eine andere Vorgehensweise steht hinter den Programmiersprachen, deren Ausführung durch sogenannte Interpreter erfolgt. So werden z.B. die einzelnen Befehle eines BASIC-Programms erst zur Laufzeit analysiert und anschließend ausgeführt. Der Zwischenschritt über den Objekt-Code entfällt. Eine Verbindung beider Konzepte führt zu dem von Bowles entwickelten UCSD-Pascal-System [4]. Der vom seinem Team realisierte Pseudo-Übersetzter übersetzt einen Pascal-Quelltext anstatt in mikroprozessorabhängigen Objekt-Code in sogenannten Pseudo-Code. Dieser Pseudo-Code setzt sich nicht mehr aus einzelnen direkt ausführbaren Maschineninstruktionen zusammen, sondern aus komplexen Befehlen, die mit Hilfe eines Interpreters (*P-machine*) abgearbeitet werden. Der Interpreter übernimmt die systemspezifischen Aufgaben, während der Übersetzer durch den quasi genormten Pseudo-Code systemunabhängig bleibt.

Die Vorteile der Objekt-Code-Übersetzer bzgl. der einmaligen Quelltextanalyse konnten bei dem UCSD-Pascal beibehalten werden. Auch war es einfach, immer neue Versionen von Pascal zu entwickeln, da der Aufwand für einen neuen Übersetzer recht gering war. Unter Umständen konnte der von dem neuen Übersetzer erzeugte Code auch weiterhin ohne Änderung des Interpreters auf verschiedenen Systemen ausgeführt werden. Wohlgemerkt spielt UCSD-Pascal heute in der Praxis nur noch eine unbedeutende Rolle, denn obwohl Pseudo-Code wesentlich schneller verarbeitet werden kann als z.B. interpretiertes BASIC, sind diese Programme trotzdem deutlich langsamer als das gleiche Programm übersetzt mit einem Objekt-Code-Übersetzer. Dennoch gibt es auch heute noch oder wieder Entwicklungssysteme (z.B. Smalltalk), die sich den Weg über den Pseudo-Code (siehe Smalltalk Byte Code) zunutze machen.

Multimedia-Entwicklungsumgebung

Die Skriptsprachen, die oftmals in Multimedia-Entwicklungsumgebungen eingesetzt werden, sind wesentlich stärker an das jeweilige Anwendungsgebiet angelehnt, als dies bei den im Vergleich dazu sehr allgemeinen prozeduralen Programmiersprachen der Fall ist. Die relativ einfache Ablauflogik der Skriptsprachen führt häufig dazu, daß diese vorrangig durch einen Interpreter verarbeitet werden. Man kann davon ausgehen, daß die meisten Autorensystemen intern mit einem dem Pseudo-Code vergleichbaren Format arbeiten. Ersichtlich wird dies z.B. an den speziellen Dateiformaten, die anstatt den Skriptquelltext in editierbarer Form diesen in einer aufbereiteten und vom Laufzeitsystem direkt ausführbaren Form (*tokenized*) speichern. Diese Schnittstelle bleibt allerdings dem Anwendungsentwickler meist verborgen, so daß diese Kodierung der Präsentation nicht für den Austausch verwendet werden kann.

Durch den Einsatz von MHEG kann diese verborgene Schnittstelle aufgebrochen werden. Der mit der Normung verbundene Vorteil bezieht sich auf die Austauschbarkeit der Strukturinformationen in ausführbarer Form zwischen heterogenen Systemen. Ein weiteres sehr aktuelles Problem ist z.B. die Notwendigkeit, daß neben den Inhalten auch die Struktur einer Präsentationen in einem Datenbanksystem gespeichert werden soll. Einige neuere Datenbankansätze vereinen spezielle (bzgl. Abfragesprache und Zugriffsverfahren) Datenbanksysteme für die einzelnen Medien sowie für die Strukturdaten unter einer gemeinsamen Schale [1], [23]. Auf diese Datenbasis können die unterschiedlichsten Anwendungen auf den Multimedia-Endgeräten zurückgreifen. Wird dies in der editierbaren Form vorgenommen, dann muß vor jeder Wiedergabe erst ein Layoutprozeß ablaufen. Speichert man dagegen die Präsentation in der ausführbaren Form, dann entfällt dieser Schritt und die Verarbeitung erfolgt vergleichbar mit der eines P-Code-Interpreters mit Hilfe einer sogenannten *MHEG-Engine*. Die Erstellung einer Präsentation soll weiterhin mit den Skriptsprachen und den zugehörigen interaktiven Werkzeugen aus den Autorensystemen durchgeführt werden. Geeignete Übersetzer für die jeweiligen Skriptsprachen erzeugen als Ergebnis der Umwandlung die MHEG-Objekte. Ein vergleichbares Konzept wird auch mit Script/X von Kaleida verfolgt [17].

Abschließend sei noch angemerkt, daß mit dieser Analogie nicht der Sprachumfang von MHEG mit dem vom P-Code verglichen werden soll. Denn im Gegensatz zum P-Code ist MHEG keine in sich geschlossene Sprache. MHEG ist ein Austauschformat und so gesehen eher mit *Postscript* vergleichbar. Die Analogie wurde hier hinsichtlich des Verarbeitungsmodells und der Austauschbarkeit gezogen.

4.3 Ausführung einer MHEG-Präsentation

Bis jetzt lag der Schwerpunkt auf der Beschreibung von den einzelnen MHEG-Objekten und wie aus diesen eine interaktive Multimedia-Präsentation aufgebaut werden kann. In diesem Abschnitt wird kurz auf die Wiedergabe einer MHEG-kodierten Präsentation eingegangen. Abbildung 17 zeigt den Fluß von MHEG-Objekten durch die einzelnen Komponenten einer möglichen Realisierung einer MHEG-Engine, die von einer Anwendung bedient wird und durch sogenannte Präsentationsdienste unterstützt wird. Verständlicherweise kann eine derartige Beschreibung nur im Zusammenhang mit einigen Annahmen erfolgen. So werden sowohl der Dekodierer als auch der Kodierer der MHEG-Engine zugeordnet. Eine alternative Schnittstelle zur MHEG-Engine könnte diese Komponenten als Teil der Anwendung vorsehen, um eine Verarbeitung der MHEG-Objekte innerhalb der Anwendung zu gestatten.

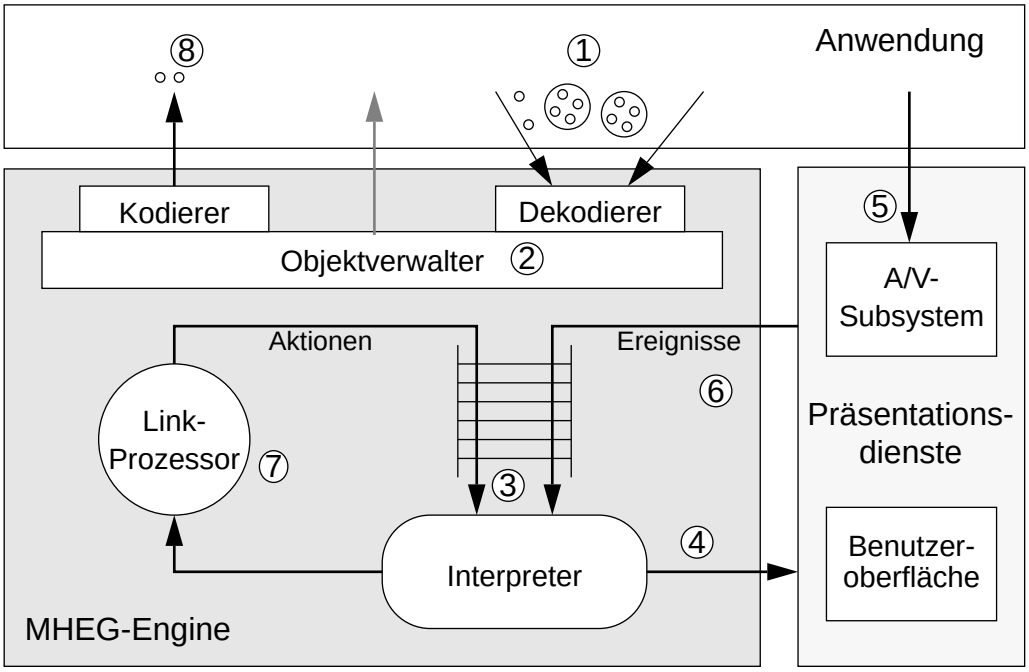

Abbildung 17: Komponenten der MHEG-Engine

1. Im ersten Schritt initialisiert die Anwendung sowohl die MHEG-Engine als auch die notwendigen Präsentationsdienste. Desweiteren hat sie als Aufgabe, die Objekte einer MHEG-kodierten Präsentation bereitzustellen, d.h. in der Regel beginnt eine Präsentation mit der Übergabe eines ersten Composite-Objektes an die MHEG-Engine. Nach dem Start einer Präsentation können weitere MHEG-Objekte nachgereicht werden.

2. Der Objektverwalter stellt eine Art zentrales Verzeichnis aller MHEG-Objekte dar, die im Laufe einer Präsentation in die MHEG-Engine geladen werden. Bevor jedoch MHEG-Objekte verarbeitet werden können, müssen diese von dem Dekodierer aus der Transfer Syntax in die Lokale Syntax überführt werden. Werden in den verschiedenen MHEG-Objekten andere referenziert, dann wenden sich die einzelnen Komponenten an den Objektverwalter, der über die vorgestellten Adressierungstechniken die gesuchten MHEG-Objekte bereitstellt. Hierzu kann es notwendig sein, daß weitere Objekte über die Anwendung angefordert werden müssen (schraffierter Pfeil).

3. Der Interpreter ist die Hauptkomponente in einer MHEG-Engine. Er wird getrieben von Nachrichten, die hier in der Reihenfolge ihres Auftretens nach in einer Warteschlange vorliegen. Nachrichten werden unterschieden in Ereignisse, die von den Präsentationsdiensten an die MHEG-Engine gesendet werden und in Aktionen, die von dem Link-Prozessor aktiviert werden. Beide beziehen sich grundsätzlich auf ein bestimmtes MHEG-Objekt oder eine virtuelle Sicht, so daß sie vom Interpreter entsprechend ausgewertet und verarbeitet werden können. Üblicherweise startet eine Anwendung eine Präsentation mit der Aktion *prepare* auf das zuvor geladene Composite-Objekt.
4. Der Interpreter bedient sich bei der Verarbeitung der Nachrichten der Präsentationsdienste. So muß z.B. eine Aktion *run* auf ein *Selection-Presentable* einen Schieberegler mit Hilfe der jeweiligen Dienste der Benutzeroberfläche anzeigen.
5. Für das Anzeigen von kontinuierlichen Medien bieten sich oft spezielle A/V-Subsysteme an. Die vorangegangenen Ausführungen haben gezeigt, daß die Daten für diese Medien meist nicht im zugehörigen Content-Objekt gespeichert werden, sondern lediglich referenziert werden. In diesem Fall müssen die Daten von der Anwendung bereitgestellt werden.
6. Die Schnittstelle zu den Präsentationsdiensten wird in der Regel sowohl synchron als auch asynchron zu bedienen sein. Es ist z.B. nicht zweckmäßig, auf das Ende eines Videos mit einem synchronen Aufruf zu warten. Deshalb kommt es vor, daß die Präsentationsdienste ihrerseits asynchron Ereignisse an den Interpreter zurückmelden (Ende des Videos). Gleiches gilt auf für die beiden Formen der Benutzerinteraktion.
7. Die einzelnen Nachrichten können Zustandsänderungen in den verschiedenen Statusfeldern (siehe z.B. *preparation*, *presentation*, *selection* und *modification status*) bewirken, so daß in diesen Fällen der Link-Prozessor eingeschaltet wird. Dieser überprüft alle Link-Objekte, bei denen das betroffenen MHEG-Objekt als Quellenobjekt definiert wurde. Sind sowohl die auslösende als auch die zusätzliche Bedingung erfüllt, dann werden die im Link-Objekt definierten Aktionen an die Zielobjekte gesendet, indem diese in die Warteschlange eingestellt werden.
8. Schließlich kann es erforderlich sein, daß MHEG-Objekte aus der MHEG-Engine über den Kodierer an die Anwendung zurückgegeben werden müssen (*returnability*).

Es sei angemerkt, daß der MHEG-Standard weder einzelne Komponenten der MHEG-Engine beschreibt noch eine Schnittstelle zwischen der MHEG-Engine und einer bedienenden Anwendung definiert. Dies kann auch nicht Aufgabe des Standards sein, da verschiedenen Anwendungen sicher unterschiedliche Anforderungen an die MHEG-Engine stellen werden. So werden z. B. derzeit in der ITU-Empfehlung T.170 [16] entsprechende Anforderungen in Form von Schnittstellen und Protokollen aus Sicht eines Abfragedienstes spezifiziert.

4.4 Anforderungen an die Laufzeitumgebung

Der MHEG-Standard führt eine Reihe von Anforderungen auf, die an die ihn gestellt wurden, und die bei der Realisierung einer Laufzeitumgebung zu berücksichtigen sind. Auf die drei wesentlichen Anforderungen: Unterstützung minimaler Ressourcen, Echtzeitfähigkeit und Synchronisation, wird im weiteren etwas näher eingegangen.

Unterstützung minimaler Ressourcen

Bis jetzt wurde stillschweigend davon ausgegangen, daß eine MHEG-Engine jede MHEG-kodierte Präsentation ohne Einschränkung wiedergeben werden kann. Eine realistische Anforderungen könnte sein, daß für die Wiedergabe relative einfache Endgeräte zu berücksichtigen sind (vgl. Videotext). Mögliche Einschränkungen wären z.B. eine Begrenzung der unterstützten Medienarten oder die Anpassung der Qualität ihrer Wiedergabe. Wie kann sich also ein Autor sicher sein, daß die von ihm kodierte Präsentation entsprechend seiner Vorgaben auch tatsächlich auf verschiedenen Präsentationssystemen wiedergegeben werden kann? In der Praxis ist er letztenendes dazu gezwungen, zum Zeitpunkt der Komposition einer Präsentation gewisse Annahmen über die Laufzeitumgebung zu treffen. Im Standard werden diese Annahmen auch als “minimale Ressourcen” bezeichnet. Vom Konzept her ist vorgesehen, daß die Anforderungen an die Ressourcen, die von den einzelnen Objekten einer Präsentation benötigt werden, mit Hilfe eines Descriptor-Objektes zusammengefaßt werden können. Dazu gehört z.B. welche Primitive für die Modifikation verwendet werden oder von welchem darstellbaren Koordinatenbereich ausgegangen wird. Eine Anwendung kann mit dieser Information zur Laufzeit eine MHEG-Engine entsprechend konfigurieren oder im schlechtesten Fall wegen nicht ausreichenden Ressourcen (z.B. keine Audiounterstützung) die Wiedergabe einer Präsentation ablehnen. Neben diesen sehr harten Bedingungen kann der Autor auch die Güte der Wiedergabe steuern (vgl. *Scalablity, Graceful degradation*).

Echtzeitfähigkeit

Die Forderung nach Echtzeitfähigkeit ist aus zwei Voraussetzungen abgeleitet. Zum einen sollen in MHEG zeitbehaftete Medien wie Audio und Video berücksichtigt werden. Deren zeitlich korrekte Wiedergabe ist zwar Aufgabe der zugrundegelegten Präsentationsdienste, dennoch resultiert daraus, daß die MHEG-Engine ein Zeitmodell unterstützen muß (vgl. Zeitachse im virtuellen Koordinatensystem). Zum anderen können vom Autor durch entsprechende Action-Objekte zeitliche Bedingungen formuliert werden (siehe *Delayed-sequential-actions*). Im Gegensatz zu den zeitbehafteten Medien müssen diese Bedingungen von der Laufzeitumgebung erbracht werden.

Synchronisation

Desweiteren trifft der MHEG-Standard einige Annahmen über die ihm gebotenen Synchronisationsmechanismen entsprechend der Einteilung der Multimedia-Synchronisation auf drei Ebenen, wie sie in [20] vorgestellt wurde. Auf der untersten Ebene (Medien-Ebene) wird die Synchronisation einzelner Medien gewährleistet. Auf der nächst höheren Ebene operiert eine Anwendung mit einzelnen Medienströmen und kann diese zum Zwecke der Synchronisation zu einer Gruppe verbinden. Durch die Möglichkeit des statischen und des dynamischen Multiplexens baut der MHEG-Standard auf diesen beiden Ebenen auf. Mit dem statischen Multiplexen können z.B. vor der Wiedergabe eine Audiospur und eine Videosequenz zu einem gemeinsamen Content-Objekt verbunden werden (vgl. MPEG). Bei großen Objekten ist dies nicht immer sinnvoll bzw. im Falle von nicht persistenten Strömen (Videostrom von einer Kamera) ist diese Vorgehensweise nicht möglich, so daß mit dem dynamischen Multiplexen eine logische Gruppe von synchron ablaufenden Strömen definiert werden kann. Die Synchronisation auf der höchsten Ebene (Objekt-Ebene) wird von den Mechanismen einer MHEG-Engine erbracht. Hier müssen komplexe zeitliche Beziehungen zwischen den verschiedenen virtuellen Sichten wiedergegeben werden. Die Definition dieser Beziehungen erfolgt durch die Link-Objekte.

4.5 Anwendungsgebiete für MHEG

Abschließend wird in diesem Abschnitt exemplarisch auf die Verwendung von MHEG in verteilten Multimedia-Systemen eingegangen. Dazu bietet sich eine Einteilung der verschiedenen Anwendungsgebiete anhand der bekannten Klassifikation von Anwendungsdiensten aus der CCITT-Empfehlung I.211 [3] an.

Abfragedienste

Abfragedienste sind ein bevorzugtes Anwendungsgebiet für MHEG. Für die Arbeitsweise der Abfragesysteme ist charakteristisch, daß ein Anwender von einem Endgerät aus eine Abfrage an eine Datenbank stellt und mit ein bißchen Glück als Ergebnis die gewünschte Information angezeigt bekommt (vgl. Online-Recherche). Sieht man einmal von Hypertext/media-Systemen ab, dann können die Anfragen in den seltesten Fällen kontextsensitiv gestellt werden, d.h. der Anwender muß die Anfrage recht gezielt formulieren und es bleibt ihm überlassen, Zusammenhänge zwischen den einzelnen Informationen herzustellen. Speichert man nun die Inhalte und eine Strukturbeschreibung gemeinsam in der Datenbank ab, dann lassen sich durch die Berücksichtigung von Benutzerinteraktion komplexe Abläufe beschreiben. Die Abfragesprache ist nun Teil der Präsentation. Auf diese Weise unterstützt MHEG die Konzepte von Hypertext/ Hypermediasystemen, indem es gestattet, daß ein Anwender interaktiv durch eine Reihe von miteinander verknüpften Präsentationen navigieren kann.

Verteildienste

Ging in Abfragediensten die Informationsbeschaffung vom Anwender aus, so wird in Verteildiensten die Information von einer zentralen Stelle an eine Gruppe von Benutzern verteilt. In den Empfehlungen werden Verteildienste ohne Benutzerinteraktion (vgl. Fernsehen) und Verteildienste mit Benutzerinteraktion (vgl. VideoText) unterschieden. Beiden ist gemeinsam, daß es keinen Rückkanal von der Empfänger- zur Sendestation gibt. Besteht keine Möglichkeit zur Benutzerinteraktion, dann muß die Information so aufgenommen werden, wie sie geliefert wird. Ansonsten bezieht sich die Benutzerinteraktion auf eine lokale Auswahl der übermittelten Informationen durch die Steuerung eines Filters (vgl. Seitennummer im Videotext). In diesem Zusammenhang läßt sich MHEG mit Interactive-TV in Verbindung bringen. Man stelle sich z.B. vor, daß im Rahmen einer Verkaufsshow immer ein Knopf auf dem Fernseher eingeblendet wird, der es dem Zuschauer erlaubt, das gerade angepriesene Produkt zu bestellen. Beim Betätigen des Knopfes erscheint ein entsprechendes Formular, auf dem der Anwender lediglich noch die Menge und eine persönliche Identifikation einträgt. Der auf diese Weise erstellte Datensatz wird über eine im Hintergrund aufgebaute ISDN-Verbindung an den entsprechenden Händler (in der Regel nicht der Fernsehsender) gesendet. In diesem Szenario würden die MHEG-Objekte zur Modellierung des Bestellvorganges zusammen mit dem Fernsehprogramm übertragen.

Konversationsdienste

In einem Konversationsdienst tauschen die einzelnen Kommunikationsteilnehmer die Informationen synchron untereinander aus. Bekannte Dienste sind z.B. das Telefon oder computergestützte Video-Konferenzsysteme. Letztere müssen Mechanismen zum Verwalten von Benutzergruppen, Übertragungskanäle für Audio/Video-Verbindungen und Protokolle zur Synchronisation gemeinsam genutzter Anwendungen bereitstellen. Dies ist sicher nicht die Domäne von MHEG, allerdings kann man sich eine MHEG-Lauf-

zeitumgebung als eine Anwendung in einer Umgebung zum kooperativen Arbeiten vorstellen. Es wäre z.B. ein Szenario denkbar, in dem ein Kunde an einem Bankautomaten der Zukunft, geführt von einem aus der Zentrale zugeschalteten Berater, ein in MHEG kodierten Antrag für einen Sparplan ausfüllt. Einer technischen Realisierung stehen zumindest zwei generelle Alternativen gegenüber. Zum einen als traditionelle Anwendung mit genau einer MHEG-Engine und zum anderen als speziell dafür entwickelte Anwendung mit je einer MHEG-Engine am Endgerät des Kunden bzw. des Beraters. Im ersten Fall ist die Verteilung für die MHEG-Engine transparent gelöst (*non sharing-aware*), z.B. über geeignete Mechanismen in den erwähnten Präsentationsdiensten. Im zweiten Fall erfolgt die Synchronisation beider Sitzungen durch spezielle Dienste über die Anwendung.

Nachrichtentransferdienste

Auch in den Nachrichtentransferdiensten steht MHEG nicht unmittelbar im Vordergrund. Hier kommt es mehr darauf an, daß Dokumente nach dem bekannten *store-and-forward* Prinzip von einem Absender zu einem Adressaten weitergereicht werden. Vergleichbar der Verwendung von ODA in einer X.400-Umgebung könnten multimediale Dokumente, sofern sie nicht weiter editiert werden sollen, in einer MHEG kodierten Form zwischen den Partner ausgetauscht und mit den MHEG-Engines auf den jeweiligen Endgeräten angezeigt werden. Wie man sich die Verwendung von MHEG in solch einem Szenario vorzustellen hat, kann z.B. in [19] nachgelesen werden.

5 Fazit und Ausblick

Der vorliegende Bericht hat einen detaillierten Einblick in die CD-Version des MHEG-Standards gegeben. Dabei ist deutlich geworden, daß MHEG den Austausch von multimedialen Strukturinformationen unterstützt, um in Zukunft interaktive Multimedia-Präsentationen auf heterogenen Systemen wiedergeben zu können. An dem Beispiel der Auswertung von Benutzerinteraktionen ist deutlich geworden, daß der MHEG-Standard auch seine Grenzen hat. Ist eine weitere Auswertung einer Modifikation notwendig (z.B. soll die Eingabe mit Daten aus einer Datenbank in Verbindung gebracht werden), dann kann dies nicht in MHEG kodiert werden. Dazu ist eine Schnittstelle zur Anwendung notwendig mit dem Nachteil, daß hier wieder Systemabhängigkeiten den Datenaustausch auf der Ebene von MHEG behindern können. Eine weitere Einschränkung besteht bzgl. der Weiterverarbeitung einer Präsentation. Diese ist nicht vorgesehen, da MHEG eine endgültige, d.h. ausführbare Darstellung repräsentiert.

Für eine abschließende Bewertung ist weiterhin zu beachten, daß der MHEG-Standard in unmittelbarer Konkurrenz zu Ansätzen, wie z.B. Compact Disk Interactive von Phillips oder Script/X von Kaleida [21] steht. Da sich der Standard noch in der Entwicklung befindet, kann trotz erster Prototypen von Laufzeitumgebungen noch keine Aussage bzgl. der zu erwartenden Marktrelevanz gegeben werden.

Mit dem Übergang zur DIS-Version werden noch einige umfangreiche Änderungen an der hier beschriebenen CD-Version erwartet. So ist heute als Ergebnis der Abstimmung über das CD-Dokument schon bekannt, daß die Link-Klasse überarbeitet, sowie die noch fehlende Kodierung für die Script- und Descriptor-Klasse aufgenommen wird. Für die weitere Normung hat man seitens der SC29/WG12 geplant, den nationalen Normungsgremien das DIS-Dokument in 1994 zur Abstimmung vorzulegen. Nach den

ISO-Direktiven wird dieser Vorgang ein halbes Jahr dauern, so daß aus momentaner Sicht bis zum Ende 1994 ein Ergebnis über die erneute Abstimmung vorliegen wird. Der endgültige Standard wird somit voraussichtlich im Laufe des Jahres 1995 verabschiedet werden.

Abschließend möchte ich mich bei all denen bedanken, die mich bei der Ausarbeitung dieses Berichtes unterstützt haben. Als erstes sei Herr Prof. Dr. W. Effelsberg erwähnt, da er mich zur Anfertigung dieser Arbeit anregte. Desweiteren gilt mein Dank den Mitgliedern des Normenausschusses Informationsverarbeitungssysteme im DIN (NI-29.1, MHEG), ohne deren Zusammenarbeit in der nationalen Normungsarbeit dieser Bericht nicht möglich gewesen wäre. Herr Dr. Ralf Steinmetz vom IBM Europäischen Zentrum für Netzwerkforschung in Heidelberg trug in vielen Gesprächen und mit kritischen Anmerkungen wesentlich zur Qualität des vorliegenden Artikels bei. Schließlich Herr Guido Grassel, er leistete durch Implementierung einer MHEG-Laufzeitumgebung in seiner Diplomarbeit wertvolle Beiträge.

6 Literaturverzeichnis

[1] S. Baldi, R. Cordes, H. Peyn, P. Sokolowsky, *Datenbanken für die Multimedia-Kommunikation*, Theorie und Praxis der Wirtschaftsinformatik, HMD 169, Jan. 1993, S. 39-55.

[2] U. Bormann, C. Bormann, *Offene Bearbeitung multimedialer Dokumente*, Informatik-Spektrum, Springer-Verlag, Bd. 14, 1991, S. 270-280.

[3] CCITT Study Group XVIII, *Draft Recommondation I.211*, International Telegraph and Telephone Consultative Committee, Genf, 23-25 May 1990.

[4] David Fox, Mitchell Waite, *Pascal Primer*, Howard W. Sons & Co Incorporation, Indiana, USA, First Edition Second Printing, 1981.

[5] ISO/IEC IS 8824:1987: *Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)*, 1987.

[6] ISO/IEC IS 8825:1987: *Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) with DAD1*, 1987.

[7] ISO/IEC IS 8869:1988: *Information Processing Systems - SGML support facilities - SGML Document Interchange Format (SDIF)*, 1988.

[8] ISO/IEC IS 8879:1986: *Information Processing Systems - Text and Office Systems - Standard Generalized Markup Language (SGML)*, 1986.

[9] ISO/IEC IS 9070:1991: *Information Processing Systems - SGML support facilities - Registration procedures for public text owner identifiers*, 1986.

[10] ISO/IEC IS 10744:1992: International Standard: Information Technology - Hypermedia/Time-based Structuring Language (HyTime), 1992.

[11] ISO/IEC IS 10918:1992: *Information Technology - Digital Compression and Coding of Continuous-Tone Still Images* (*JPEG*), 1992.

[12] ISO/IEC IS 11172:1992: *Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s (MPEG)*, 1992.

[13] ISO/IEC IS 11544:1992: *Information Technology - Digital Compression and Coding of Bi-level Images ("JBIG")*, 1992.

[14] ISO/IEC IS 12087-3:1992: *Information Technology - Computer Graphics and Image processing - Image processing and Interchange - Part 3: Image Interchange Facility (IIF)*, 1992.

[15] ISO/IEC CD 13552-1:1993: *Information Technology - Coded Representation of Multimedia and Hypermedia Information Objects (MHEG) - Part 1: Base Natotion (ASN.1)*, 15. June 1993.

[16] ITU-T Draft Recommondation T.170: *Audiovisual Interactive (AVI) Systems - General Introduction, Principles, Concepts and Models*, Second Revision, Geneva, CH, 16-25 Nov. 1993.

[17] Kaleida: *Kaleida, Script/X and the Multimedia Market*, Kaleida White Paper, Revision 1, Kaleida Labs Inc. 1945 Charlston Road, Mounten View, CA 94043, USA, 1993.

[18] B. Markey: *Emerging Hypermedia Standards: Hypermedia Marketplace Prepares for HyTime and MHEG*, (U.S.) Assistant Secretary of Defense (Production and Logistics), Washington, DC, PB92-120328, 1991.

[19] E. Moeller, A. Scheller, G. Schürmann: *Der Multimedia-Mail-Teledienst*, W. Effelsberg, K. Rothermel (Hrsg.), Verteilte Multimedia Systeme: Tagungsband GI/ITG-Arbeitstreffen, Stuttgart, 18/19. Feb. 1993, S. 206-226.

[20] T. Meyer-Boudnik, W. Effelsberg, R. Steinmetz, *A Taxonomy on Multimedia-Synchronization*, Proceedings of the Fourth Workshop on Future Trends of Distributed Computing Systems, Lisbon, Portugal, 22-24 Sep. 1993, S. 97-103.

[21] N.N.:, *Kaleida Launches Alliance*, Digital Media: A Seybold Report, Vol. 2, No. 10/11, Mär. - Apr. 1993.

[22] R. Price: *MHEG: An Introduction to the future International Standard for Hypermedia Interchange*, Proceedings of the 1st ACM International Conference on Multimedia, Anaheim, USA (CA), 1-6 Aug. 1993.

[23] T. Rakow, M. Löhr, F. Moser, E. Neuhold, K. Süllow: *Einsatz von objektorientierten Datenbanksystemen für Multimedia-Anwendungen*, R. Oldenburg Verlag, Informationstechnik und Technische Informatik (it+ti), Bd. 35, Nr. 3, Mär. 1993, S. 4-17.

[24] Ralf Steinmetz: *Multimedia-Technologie: Einführung und Grundlagen*, Springer-Verlag Berlin Heidelberg New York, Sep. 1993.

[25] Ralf Steinmetz: Clemens Engler, *Human Perception of Media Synchronization*, IBM Europäisches Zentrum für Netzwerkforschung, Technischer Bericht: TR43.9310, Okt. 1993.