Diplomarbeit im Studiengang
Medieninformatik

Einsatz von XMT und MPEG-4 zur Erstellung von RichMedia

vorgelegt von

Markus Brenner
Matrikelnummer 12340

an der Fachhochschule Stuttgart - Hochschule der Medien

am 8. März 2004

1 Prüfer: Prof. Dr. Fridtjof Toenniessen
2 Prüfer: Dipl.-Wi.-Inf.-Umw. wiss. Sibylle Wahl

Fraunhofer-Institut Stuttgart
(IAO New Media Communication Center)

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbstständig und ohne Benutzung anderer als der erwähnten Quellen und Hilfsmittel angefertigt habe. Die Arbeit hat in dieser oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ich gestatte der Hochschule der Medien Stuttgart, vorliegende Arbeit unter Beachtung datenschutz- und wettbewerbsrechtlicher Vorschriften für Forschung und Lehre zu nutzen, sofern ein expliziter Verweis auf diese Diplomarbeit und dessen Autor gewahrt bleiben.

Stuttgart, 8. März 2004

Markus Brenner

Abstrakt

Diese Diplomarbeit beschäftigt sich mit dem Einsatz von XMT[1] und MPEG-4 im Allgemeinen und unter spezieller Betrachtung im RichMedia-Umfeld. Es wird versucht aufzuzeigen, welche Vorteile und Nachteile die Realisierung solcher Anwendungen mittels MPEG-4 in Verbindung mit XMT bringen können.

Die Arbeit beginnt in den ersten zwei Kapiteln mit einem generellen Überblick über MPEG-4 und XMT. Dieser Teil vermittelt dem Leser - unabhängig von der später folgenden Betrachtung bezogen auf RichMedia - allgemein die Materie und Technologie, welche sich hinter diesen beiden Schlagwörtern versteckt. Auf einzelne technische Details wird, sofern sie keine besondere Bedeutung im Rahmen dieser Diplomarbeit darstellen, nicht näher eingegangen und bleiben den jeweiligen Spezifikationen vorbehalten.

Im dritten Teil wird das Thema RichMedia zunächst unter technischem Aspekt aufgegriffen und versucht, ein Prototyp einer RichMedia-Anwendung praktisch umzusetzen. Schwerpunkte sind dabei unter anderem die Erstellung und Konvertierung von Inhalten sowie deren Distribution und Konsum.

Der vierte Abschnitt versucht die beiden Technologien mit anderen, bereits bestehenden Standards und Lösungsansätzen zu vergleichen und gegenüber zustellen.

Das letzte Kapitel bietet schließlich eine gesamtheitliche Zusammenfassung der Technologien XMT und MPEG-4 und ein Fazit hinsichtlich der Verwendung dieser beiden Technologien für RichMedia sowie einen Ausblick.

Stuttgart, 8. März 2004

1 eXtensible MPEG-4 Textual Format

Innerhalb der Computergemeinschaft lebt man nach
der Grundregel, die Gegenwart sei ein Programmfehler,
der in der nächsten Ausgabe behoben sein wird.

Clifford Stoll

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

2D.....zweidimensional
3D.....dreidimensional
3GPP.....3rd Generation Partnership Project
AAC..... Advanced Audio Coding
AFX..... Animation Framework eXtension
API..... Application Programming Interface
ASP..... Advanced Simple Profile
ATM.....Asynchronous Transfer Mode
ATSC..... Advanced Television Systems Committee
AV..... Audio Visual
AVC..... Advanced Video Coding
AVI.....Audio Visual Interleave
BIFS..... Binary Format for Scenes
CD.....Compact Disc
CELP..... Code Excited Linear Prediction
CPU..... Central Processing Unit
DAI.....DMIF Application Interface
DCT..... Diskrete Cosinus Transformation
DMIF.....Delivery Multimedia Integration Framework
DRM..... Digital Rights Management
DRM..... Digital Radio Mondiale
DSL..... Digital Subscriber Line
DVB.....Digital Video Broadcasting
DVD.....Digital Versatile Disc
EBU..... European Broadcasting Union
EDV..... Elektronische Datenverarbeitung
ER..... Error Resilience
ES..... Elementary Stream
FBA..... Face and Body Animation
FCD..... Final Committe Draft
FDIS..... Final Drafts of International Standard
fps.....frames per second
GMC..... Global Motion Compensation
HD.....High Definition
HE.....High-Efficiency
HiFi.....High Fidelity
HILN.....Harmonic and Individual Lines plus Noise
HTML..... HyperText Markup Language
HTTP.....HyperText Transfer Protocol
HVXC..... Harmonic Vector Excitation Coding
IAO.....Institut für Arbeitswirtschaft und Organisation

IBM..... International Business Machines
IEC.....International Electrotechnical Commission
IETF.....Internet Engineering Task Force
IP.....Internet Protocol
IPMP.....Intellectual Property Management and Protection
IS.....International Standard
ISDN.....Integrated Services Digital Network
ISMA..... Internet Streaming Media Alliance
ISO.....International Organization for Standardization
ITU.....International Telecommunication Union
ITU-T.....ITU – Telecommunications Standardization Sector
JVM..... Java Virtual Machine
JVT.....Joint Video Team
LAN..... Local Area Network
LC..... Low Complexity
LD..... Low Delay
LKW.....Lastkraftwagen
MIDI.....Musical Instrument Digital Interface
MP3.....MPEG-1 Layer 3
MP4.....MPEG-4 Dateiformat
MPEG.....Moving Picture Experts Group
MUW.....Multi User Worlds
OCI.....Object Content Info
OD..... Object Descriptor
PC..... Personal Computer
PCM.....Pulse Code Modulation
PDF..... Portable Document Format
PVR..... Personal Video Recorder
QCIF.....Quarter Common Intermediate Format
QoS.....Quality of Service
RFC..... Request For Comments
ROM..... Read Only Memory
RTP..... Real-Time Transport Protocol
RTSP.....Real Time Streaming Protocol
SBR.....Spectral Band Replication
SDK.....Software Development Kit
SMIL.....Synchronized Multimedia Integration Language
SNHC.....Synthetic/Natural Hybrid Coding
SSR.....Scalable Sampling Rate
SVG.....Scalable Vector Graphics
TCP..... Transmission Control Protocol
TR..... Technical Report
TS..... Transport Stream

TTS..... Text-To-Speech
UDP..... User Datagram Protocol
UML..... Unified Modeling Language
UMTS..... Universal Mobile Telecommunications Systeme
URL..... Universal Resource Locator
VBR..... Variable Bit Rate
VfW..... Video for Windows
VHDL..... Very High Speed Integrated Circuit Hardware Description Language
VOP..... Video Object Plane
VRML..... Virtual Reality Modeling Language
W3C..... World Wide Web Consortium
WD..... Working Draft
WMF..... Windows Metafile Format
WWW..... World Wide Web
WYSIWYG..... What You See Is What You Get
X3D..... eXtensible 3D
XHTML..... eXtensible HyperText Markup Language
XML..... eXtensible Markup Language
XMT..... eXtensible MPEG-4 Textual Format
XSLT..... eXtensible Stylesheet Language Transformation

Einleitung

In Zeiten von breitbandigen Internetanschlüssen boomt geradezu der Tausch von Audio und Video über das Internet. Eine wesentliche Voraussetzung dabei sind effiziente Algorithmen und Techniken zur Kompression von Daten, um so solche Medien in der Datengröße entscheidend verkleinern und sie damit überhaupt in verträglichen Zeiten übertragen zu können.

Im Audiobereich ist MPEG-1 Layer 3, besser bekannt unter dem Namen MP3, schon seit Jahren in aller Munde und im Computer- und Unterhaltungsbereich nicht mehr wegzudenken.

Bei bewegten Bildern hat sich im Laufe der letzten Jahre besonders im PC-Umfeld der internationale Standard MPEG-4 – fälschlicherweise oftmals auch mit DivX[1] gleichgesetzt – etabliert. Dieser gewinnt in jüngster Zeit an zusätzlicher Popularität durch die Einführung von ersten Unterhaltungsgeräten wie DVD-Player mit derartiger Abspielfunktion. Es wird somit möglich, ganze Spielfilme in DVD-Qualität vom Internet herunterzuladen, auf eine einzige CD zu brennen und anschließend im Wohnzimmer anzusehen.

Im Unterhaltungsbereich angetrieben, werden diese Komprimier- und Medienstandards – herausstellend sei hier der allumfassende Standard MPEG-4 genannt – auch zunehmend in anderen Bereichen von Bedeutung gewinnen.

Im Kommunikationssektor ist hier besonders Videotelefonie und -streaming zu nennen. Die Einführung der neuen, breitbandigen 3G-Mobilfunknetze wie UMTS wird dies auf breiter Nutzung möglich machen - MPEG-4 wird dabei eine zentrale Rolle zukommen.

Aber auch in fremden Bereichen wie im Service- und Schulungssektor eröffnen sich dadurch neue Möglichkeiten. Das viel umworbene eLearning wird sich erst durch die Aufwertung mit interaktiven Inhalten und einer einfachen Nutzung sinnvoll einsetzen lassen.

Durch die Aussicht auf einen schnellen Einstieg in diesen lukrativen Markt sind mittlerweile viele Einzellösungen von diversen Firmen hervorgegangen. Dies ist natürlich nicht gerade förderlich für eine breite Einführung und Nutzung - zudem müssen sich solche Anwendungen wirtschaftlich erstellen und betreiben lassen. Interoperabilität und Kompatibilität werden somit zur Voraussetzung. Dies kann am Besten durch internationale und nichtproprietäre Standards erreicht werden. Der von der ISO/IEC zertifizierte MPEG-4 Standard scheint dabei alle Vorteile zu vereinen.

Das Fraunhofer-Institut (IAO New Media Communication Center) in Stuttgart arbeitet ebenfalls an einer RichMedia[2]-Lösung für kleine und mittelständische Un-

1 DivX ist ein zu MPEG-4 Visual ASP (fast) kompatibler Videokomprimieralgorithmus und damit nur ein kleiner Teilbereich des umfassenden Multimediastandards MPEG-4.

2 engl. für "angereicherte Medien"

ternehmen. Hierbei wird momentan auf aktuell verfügbare Technologien und Standards zurückgegriffen. Dies sind vor allem das proprietäre RealMedia-Format für Audio beziehungsweise Video und SMIL als Beschreibungssprache.

Diese Diplomarbeit versucht aufzuzeigen, in wie fern sich die beiden "neuen" Technologien MPEG-4 und XMT[1] – der Beschreibungssprache von MPEG-4 - für den Einsatz in RichMedia-Anwendungen eignen.

Viele der in diesem Dokument zum Einsatz kommenden Fachbegriffe wurden bewusst nicht aus dem Englischen ins Deutsche übersetzt. Dies geschieht vor allem deshalb, da der überwiegende Teil der frei zur Verfügung stehenden Abhandlungen zum Thema MPEG-4 und XMT in Englisch erhältlich ist. Um nicht unnötig Verwirrung beim Leser in Bezug auf einschlägige Spezialbegriffe zu stiften, wurde daher weitestgehend auf eine Eindeutschung der Fachtermini verzichtet.

Die Bedeutungen aller nicht erläuterten Abkürzungen können im Abkürzungsverzeichnis nachgeschlagen werden. Weniger geläufige Fachbegriffe sind im Glossar erklärt.

1 eXtensible MPEG-4 Textual Format

1 MPEG-4 im Überblick

MPEG-4 ist eine Gruppe von Standards für multimediale Anwendungen und genormt als ISO/IEC 14496.

Die Informationen in folgendem Kapitel stützen sich, falls nicht zusätzlich auf weitere Quellen verwiesen wird, auf [TMPEG4B], [MPEG4JS] und den unter [MPEG] zu findenden Internetseiten.

1.1 Geschichte und Entwicklung

Das "Projekt" MPEG-4 wurde 1993 von der MPEG (Moving Picture Experts Group) [MPEG] ins Leben gerufen.

Später[1] ist man zu der formellen Bezeichnung Coding of Audio-Visual Objects übergegangen, um auch den Neuerungen wie synthetische Objekte[2], welche es in den beiden Vorgängern MPEG-1 (ISO/IEC 11172) und MPEG-2 (ISO/IEC 13818) noch nicht gab, gerecht zu werden. Daraus resultierte auch eine wesentliche Voraussetzung von MPEG-4, nämlich verschiedene Objekte aller Art unabhängig voneinander verwalten und darstellen zu können. Vorausgegangen war das so genannte SNHC (Synthetic/Natural Hybrid Coding), dessen Entwicklung bis in das Jahr 1995 zurückreicht.

Waren MPEG-1 und MPEG-2 noch auf CD-ROM basierte Videoanwendungen beziehungsweise digitales Fernsehen ausgerichtet, so war spätestens mit Beginn des Massenphänomens Internet klar, dass ein für alle Datenübertragungswege[3] geeignetes Rahmenwerk gebraucht würde. 1996 wurde sodann mit der Entwicklung des DMIF (Delivery Multimedia Integration Framework) begonnen.

Mit VRML (Virtual Reality Modeling Language) in leicht veränderter Form wurde 1997 ein bereits etablierter Standard übernommen, um den Aufbau und die Anordnung von Objekten beschreiben zu können. Mit MPEG-J wurde zur gleichen Zeit die Definition einer Java-Schnittstelle angestoßen.

Eine der letzten bedeutenden Entwicklungen innerhalb MPEG-4 waren das MP4-Dateiformat[4], der Transport von MPEG-4 Inhalten über MPEG-2 TS (Transport Stream) beziehungsweise IP-Rechnernetze sowie XMT (eXtensible MPEG-4 Textual Format), einer textbasierten Beschreibungssprache für MPEG-4 Inhalte.

XMT ist mitunter Themenschwerpunkt dieser Arbeit und wird in Kapitel 2 näher erläutert.

1 1996

2 Bei synthetischen Objekten wird versucht, diese anhand parametrisierbarer Informationen zu beschreiben. Typische Vertreter in MPEG-4 sind zum Beispiel die Sprachausgabe von Text, Vektorgrafik oder FBA (Face and Body Animation).

3 zum Beispiel ATM (Asynchronous Transfer Mode) oder DAB (Digital Video Broadcasting), aber auch all die verschiedenen Möglichkeiten, Daten via IP zu übertragen

4 in wesentlichen Teilen von Apples QuickTime-Dateiformat abgeleitet

1998 wurde MPEG-4 letztlich von der ISO/IEC als Standard verabschiedet. Zwei Jahre später erfolgte eine abwärtskompatible Erweiterung namens MPEG-4 Version 2. [N4668]

1.2 Aufbau

Der MPEG-4 Standard besteht aus mehreren Teilen, die zwar aufeinander aufbauen, aber dennoch klar erkennbare Zuständigkeiten aufweisen. Die Teile können individuell[1] implementiert oder kombiniert werden.

Folgend sind alle bisher verabschiedeten sowie teils noch im Entwicklungsprozess befindlichen Teile – die so genannten Parts (engl.) - aufgelistet und erläutert:

- **Part 1: Systems**
  Spezifiziert eine Szenenbeschreibung[2] unter dem Namen BIFS (Binary Format for Scenes), Multiplexing und Synchronisation von Daten und Objekten, Speichermanagement (unter anderem das MP4-Dateiformat) sowie Rechtemanagement und Schutz von Inhalten (IPMP[3]). Später wurden in diesem Teil ebenfalls XMT, AFX (Animation Framework eXtension) und MUW (Multi User Worlds) festgelegt.
- **Part 2: Visual**
  Definiert die Kodierung[4] von natürlichen und synthetischen visuellen Objekten.
- **Part 3: Audio**
  Beschreibt natürliche und synthetische Darstellungsmöglichkeiten für Audio.
- **Part 4: Conformance Testing**
  Legt Möglichkeiten fest, um MPEG-4 Implementierungen gegenüber dem Standard auf Konformität testen zu können.
- **Part 5: Reference Software**
  Beinhaltet Software zu den meisten Parts des MPEG-4 Standards und dient zur Anschauung und Verdeutlichung der mitunter komplexen Algorithmen sowie als Basis für eigene Entwicklungen.
- **Part 6: Delivery Multimedia Integration Framework (DMIF)**
  Definiert ein Sitzungsprotokoll für den Transport und das Management von multimedialen Inhalten über ein generisches Übertragungsmedium.

1 zum Beispiel kann MPEG-4 Audio "standalone", das heißt unabhängig eingesetzt werden
2 Möglichkeit, Beziehungen zwischen den einzelnen audio-visuellen Objekten in einer Szene beschreiben zu können
3 Intellectual Property Management and Protection
4 Streng genommen ist nicht der Kodiervorgang selbst, sondern vielmehr die Repräsentation von kodiertem Inhalt festgelegt. Somit bleibt den Herstellern von MPEG-4 Implementierungen Spielraum, sich von Mitbewerbern abheben zu können.

- **Part 7: Optimised Software for MPEG-4 Tools**
  Beinhaltet optimierte Software speziell für den visuellen Teil des Standards. Im Gegensatz zu anderen Teilen des MPEG-4 Standards ist dies nur ein technischer Bericht[1] und keine Spezifikation im eigentlichen Sinne.
- **Part 8: 4 on IP Framework**
  Definiert die Übertragung von MPEG-4 mittels IP-basierten Protokollen.
- **Part 9: Reference Hardware Description**
  Dieser Teil soll Hardware-Referenzimplementierungen in Form von VHDL (Very High Speed Integrated Circuit Hardware Description Language) beinhalten.
- **Part 10: Advanced Video Coding (AVC)**
  Erweiterung der MPEG-4 Video Kompressionsalgorithmen hinsichtlich Darstellungsqualität und Effizienz. AVC ist auch unter den Namen H.264, H.26L und JVT Codec bekannt.

In der jüngsten Zeit kamen weitere Entwicklungen[2], auf welche im Rahmen dieser Diplomarbeit nicht näher eingegangen werden soll, hinzu. Dies sind:

- **Part 11: Scene Description and Application Engine**
  Die Szenenbeschreibung soll von Part 1: Systems in einen eigenen Teil ausgegliedert werden.
- **Part 12: ISO Base Media File Format**
- **Part 13: IPMP Extensions**
  Erweiterungen im Management von digitalen Rechten sowie dem Schutz von Inhalten.
- **Part 14: MP4 File Format**
  Basiert zu Teilen auf Part 12.
- **Part 15: AVC File Format**
  Basiert zu Teilen auf Part 12.
- **Part 16: Animation Framework eXtension (AFX)**
  Beinhaltet unter anderem Multi User Worlds (MUW).
- **Part 17: Streaming Text Format**
- **Part 18: Font Compression and Streaming**

[MPEGIF]

1 Technical Report (TR)
2 liegen teilweise bisher nur als Entwurf vor

1.3 Zielsetzungen

Die Zielsetzungen von MPEG-4 sind sehr vielschichtig. Die wichtigsten können folgendermaßen zusammengefasst werden:

- MPEG-4 soll eine Sammlung an verschiedenen Technologien bereitstellen, welche von allen Arten von multimedialen Diensten und Netzwerken genutzt werden können. Der Austausch von Inhalten zwischen diesen verschiedenen Diensten sollte dabei "fließend" sein.
- Verbesserung der Benutzerfreundlichkeit sowie Möglichkeiten, dem Anwender audio-visuelle Inhalte mit einer dem World Wide Web (WWW) vergleichbaren Interaktivität bieten zu können.
- Integration von zusätzlichen Informationen (RichMedia), wobei diese sowohl von Autoren als auch Endanwendern leicht erstellt und genutzt werden können sollen. Die Informationen können natürliche wie auch künstliche Medien beinhalten.

1.4 Funktionalitäten

Entsprechend den Zielgruppen – Fernsehen/Film/Entertainment, EDV und Telekommunikation - lassen sich die (angestrebten) Funktionalitäten in drei Teilbereiche einteilen:

1. **Inhaltsbezogene Interaktivität**
   - ◇ Inhaltsbasierter Zugriff auf multimediale Daten
   - ◇ Manipulation von einzelnen Inhalten beziehungsweise Objekten einer Komposition
   - ◇ Verknüpfung von natürlichen und künstlichen Inhalten zu einer Szene
   - ◇ Effizienter Zugriff auf eine beliebige Stelle innerhalb einer audio-visuellen Sequenz
2. **Kompression**
   - ◇ Bessere und effizientere Kompression als bisherige Standards
   - ◇ Kodierung von mehreren, gleichzeitigen Datenströmen[1]
3. **Universeller Zugriff**
   - ◇ Fehlertolerant
   - ◇ Skalierbarkeit in räumlicher und zeitlicher Richtung

1 zum Beispiel mehrere Tonspuren oder Blickrichtungen

1.5 Architektur

Dieser Abschnitt vermittelt die hinter MPEG-4 stehende technische und ideologische Architektur.

1.5.1 Objektorientierung

Der Hauptunterschied von MPEG-4 zu vorhergegangen audio-visuellen Kodierstandards ist neben der Einführung von neuen Funktionalitäten vor allem der objektbasierte Ansatz bei der Repräsentation von Inhalten.

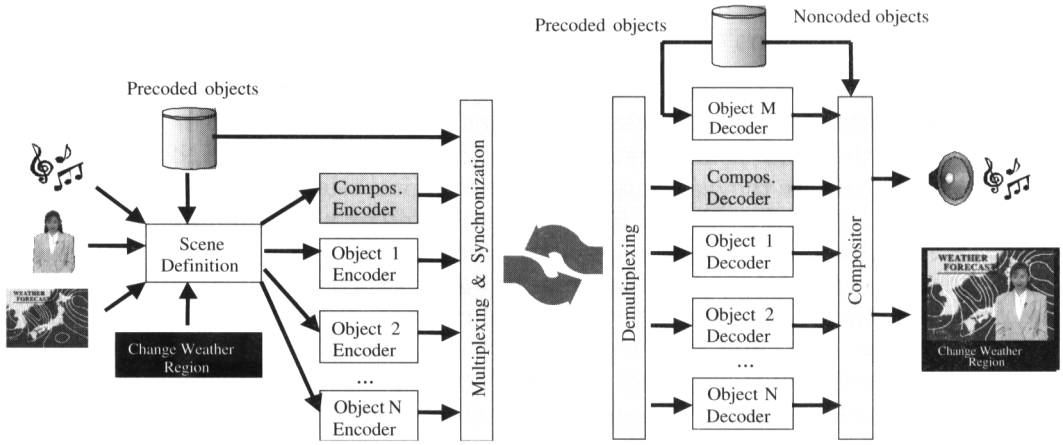

Abbildung 1: Objektbasierte Architektur in MPEG-4

Eine objektbasierte Szene kann aus verschiedenen, in Zeit und Raum unabhängigen Objekten aufgebaut werden. Dies bringt unmittelbare Vorteile mit sich:

- Einzelne Objekte können unterschiedlich kodiert und komprimiert werden, um so jeweils eine optimale Qualität und Effizienz zu erzielen. Ein Untertitel könnte somit beispielsweise als Text und das dahinter liegende Video in Form von binären Pixeln repräsentiert werden.
- Es erlaubt die harmonische Integration von Inhalten und Daten unterschiedlichsten Typs in einer Szene – im Besonderen solche mit natürlichem und künstlichem Ursprung wie zum Beispiel ein realer Mensch in einer virtuellen 3D Welt.
- Interaktion und Verknüpfung von Objekten ist nun möglich.

Da die Anwendungsbereiche dieses neuen Konzeptes sehr vielseitig sind, wurde MPEG-4 von Anfang mit dem Ziel entwickelt, kein Standard im klassischen Sinne sondern eine Gruppe von Werkzeugen und Rahmenbedingungen zu sein.

1.5.2 Systemarchitektur

In nachfolgendem Bild ist die allgemeine Systemarchitektur von MPEG-4 abgebildet.

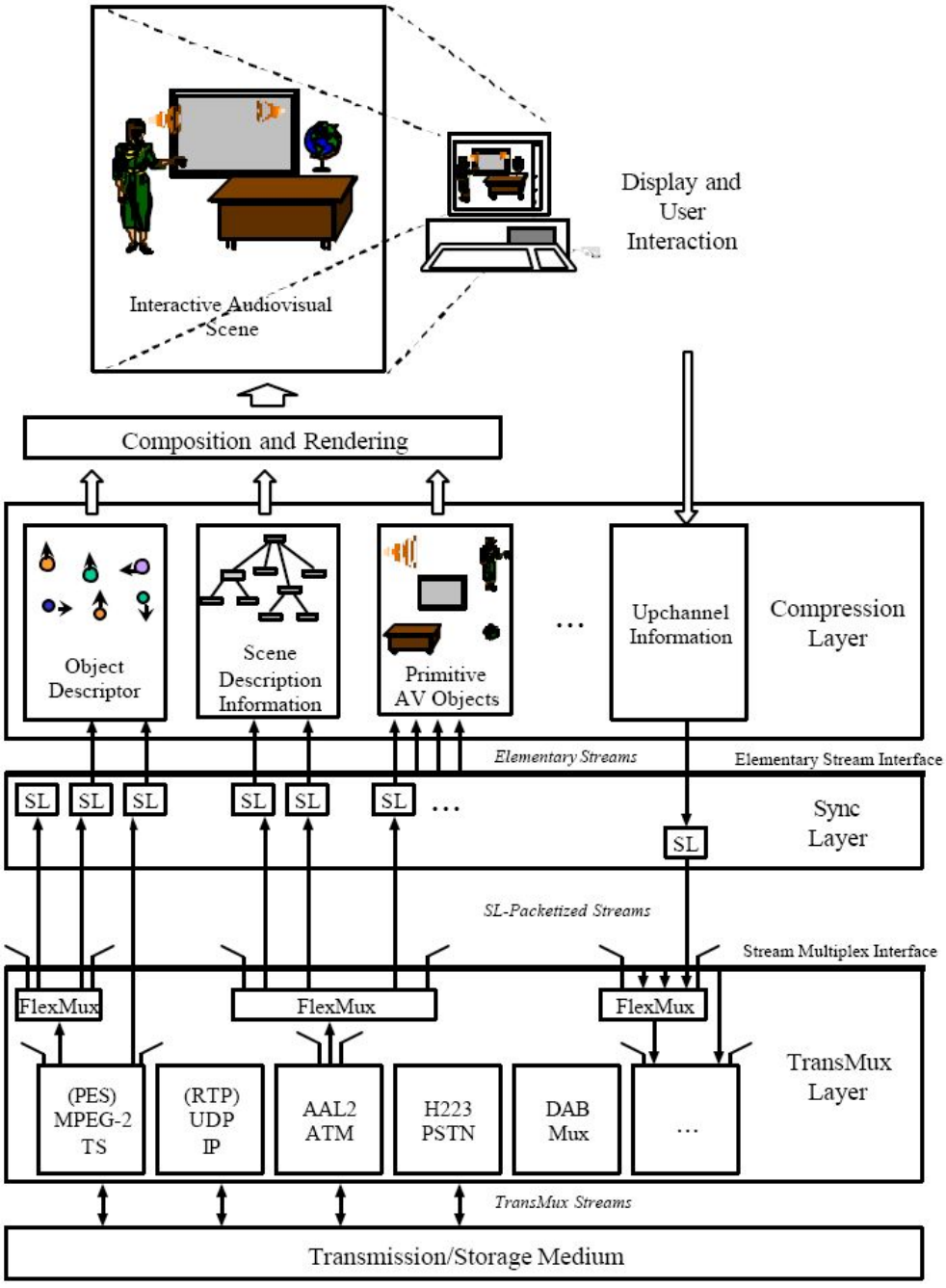

Abbildung 2: Systemarchitektur

Auf der vorangegangenen Abbildung ist zu erkennen, dass das System einem Drei-Schichtenmodell folgt. Die einzelnen Schichten sind:

- **Kompressionsschicht**
  Hier wird der logische Inhalt festgelegt. Daten, Beziehungen und Verhalten (Interaktivität) werden komprimiert, kodiert und definiert.
- **Synchronisationsschicht**
  Diese Schicht sorgt für eine einheitliche Synchronisation zwischen den Datenströmen der einzelnen Objekte - den so genannten Elementary Streams (ESs) - indem es diese mit Zeitstempeln versieht.
- **Transportschicht**
  Die Transportschicht führt mehrere ESs schließlich mittels Multiplex-Verfahren zu einem geeigneten Datenstrom zusammen, um diesen dann über Netzwerke übertragen oder in Dateien vorhalten zu können.

Das Schichtenmodell legt keine bestimmte Richtung vor, das heißt die Kompressionsschicht kann genauso Decoder beziehungsweise Encoder oder die Transportschicht Empfänger beziehungsweise Sender darstellen.

1.5.3 Ende-zu-Ende Wertschöpfungskette

Obwohl der MPEG-4 Standard für keine spezielle Anwendung alleine, sondern für ein möglichst breites Anwendungsgebiet konzipiert wurde, so ergeben sich bei Erstellung, Distribution und Konsum von MPEG-4 Inhalten doch Gemeinsamkeiten.

Solch ein Beispielszenario ist in der nachfolgenden Abbildung dargestellt und zeigt die zusammenwirkenden Elemente und deren Funktion in einer Wertschöpfungskette.

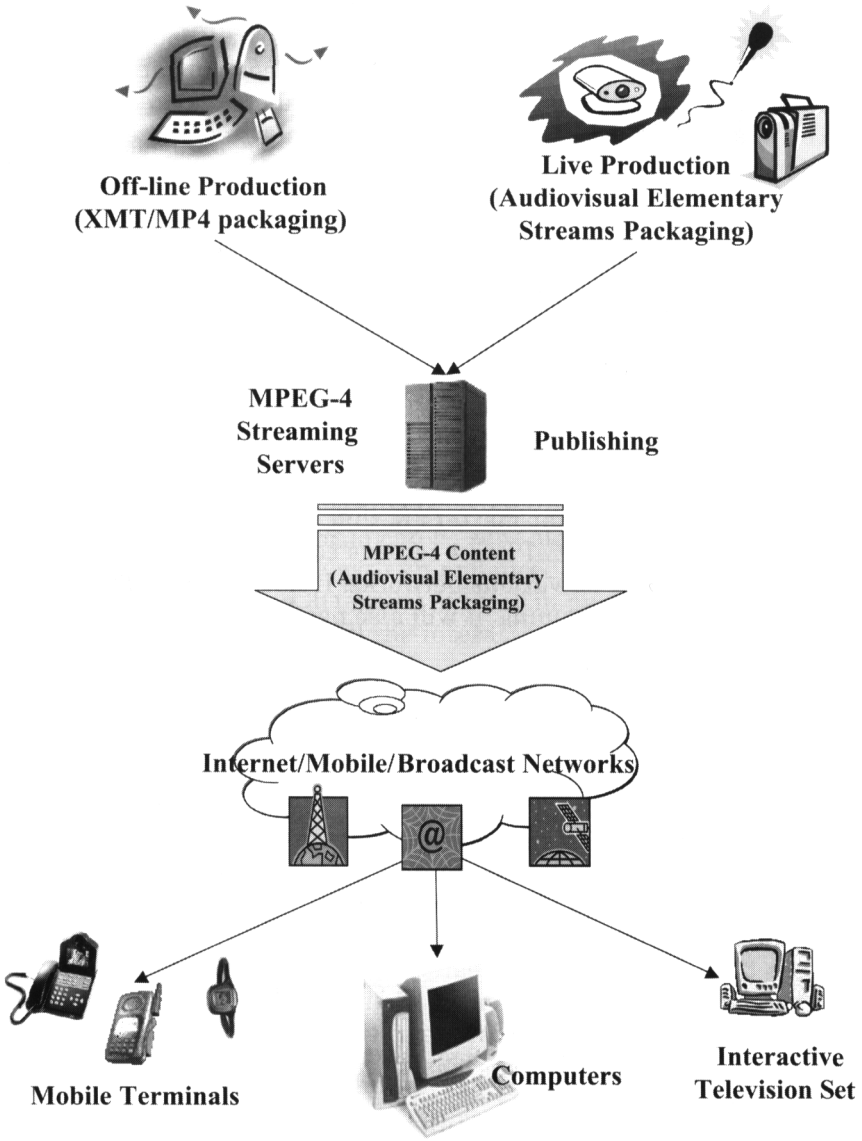

Abbildung 3: Ende-zu-Ende Wertschöpfungskette

Am Anfang dieser Kette stehen Autoren, welche audio-visuelle Inhalte mittels unterschiedlichsten Werkzeugen und Verfahren erstellen - live oder auch auf "normalem" Wege. Dabei unterscheidet man zwischen:

- "Authoring" (engl.) - der Produktion beziehungsweise Aufnahme von Medien
- dem Zusammenfügen von Medien mittels einer Szenenbeschreibung und Anreicherung mit Interaktivität
- der Verteilung der dabei entstandenen Inhalte über Netzwerke oder Datenträger

Die Verteilung von Inhalten kann in Form von binären Formaten (Streaming, MP4-Dateien, usw.) oder XMT, einer auf XML basierten Sprache zur Beschreibung

von Inhalten geschehen. Letzteres Format bietet den nächsten an der Kette stehenden Elementen mehr Interoperabilität und Flexibilität für weitere Manipulationen und Anpassungen der Daten. XMT ist deshalb gerade für den Austausch zwischen Erstellern von Inhalten geeignet.

MPEG-4 Daten müssen aber zwangsläufig nicht immer in MP4- oder XMT-Dateien gespeichert werden. So können Inhalte zum Beispiel auf Servern in ganz verschiedenen Formaten gehalten werden – unter der Voraussetzung, dass die einzelnen Inhalte zum Standard kompatibel bleiben, denn was letztlich vom Server zum Empfänger gesendet wird, sind nur so genannte Elementary Streams (ESs) – zum Beispiel die einzelnen Objekte einer Szene.

Diese einzelnen Streams können auch über mehrere, verschiedene Netzwerke versendet und erst beim Empfänger wieder zu einem synchronen Datenstrom zusammengefügt werden.

Durch Abstraktion von Inhalt und Übertragungsmedium mit Hilfe des DMIF[1] können diese Streams über jedes beliebige Medium (zum Beispiel dem Internet mittels IP oder über Satellit/MPEG-2 TS) transportiert werden.

Zusammengefasst wird mit MPEG-4 der Ansatz "create once, run everywhere" verfolgt.

1.6 MPEG-4 Technologien

Dieses Kapitel bietet einen groben Überblick über die wichtigsten Technologien in MPEG-4. Zum Zeitpunkt der Erstellung dieser Arbeit waren dies Systems, Visual, Audio und das Delivery Multimedia Integration Framework (DMIF).

Seitdem eingebrachte Erweiterungen wie Animation Framework eXtension (AFX) werden nicht behandelt.

1.6.1 Systems

MPEG-4 definiert unter anderem verschiedene Algorithmen und Technologien zur Kompression von Audio und Video. Die resultierenden Datenströme - fortan Elementary Streams (ESs) genannt - können dabei getrennt gespeichert oder übertragen werden.

Wie diese anschließend wieder auf Empfängerseite zu einem Objekt oder einer Szene zusammengeführt werden und welche Möglichkeiten es dabei gibt, ist unter anderem in MPEG-4 Part 1: Systems [MPEG4-1] festgelegt.

Die Technologien aus diesem Teil des MPEG-4 Standards werden nachfolgend zusammengefasst.

1 Delivery Multimedia Integration Framework – mehr dazu in Kapitel 1.6.4

1.6.1.1 Object Descriptor (OD)

OD definiert ein Rahmenwerk, mit welchem die Beziehungen zwischen den individuellen ESs und den Medienobjekten in einer Szene beschrieben werden können.

ODs bieten darüber hinaus zusätzliche Informationen wie Speicherort[1] der ESs, benötigte Voraussetzungen für einen Decoder oder Daten für den Schutz von Inhalten sowie die Verwaltung und Wahrung von Rechten (DRM[2]). Letzteres ist in MPEG-4 unter der Abkürzung IPMP (Intellectual Property Management and Protection) bekannt.

1.6.1.2 Binary Format for Scenes (BIFS)

Mit Hilfe der binären Szenenbeschreibungssprache BIFS, welche an VRML angelehnt ist, gibt es eine Möglichkeit, die räumliche (2D und 3D) sowie zeitliche Anordnung von Objekten innerhalb einer Szene zu beschreiben.

Neben Primitiven für Text und Grafik bietet BIFS auch Möglichkeiten zur Gruppierung und Verknüpfung von Objekten. Des weiteren wird Interaktivität in Form einer Ereignisbehandlung - ausgelöst von Betrachter, der Szene selbst oder durch Interaktion zwischen Client und Server – unterstützt.

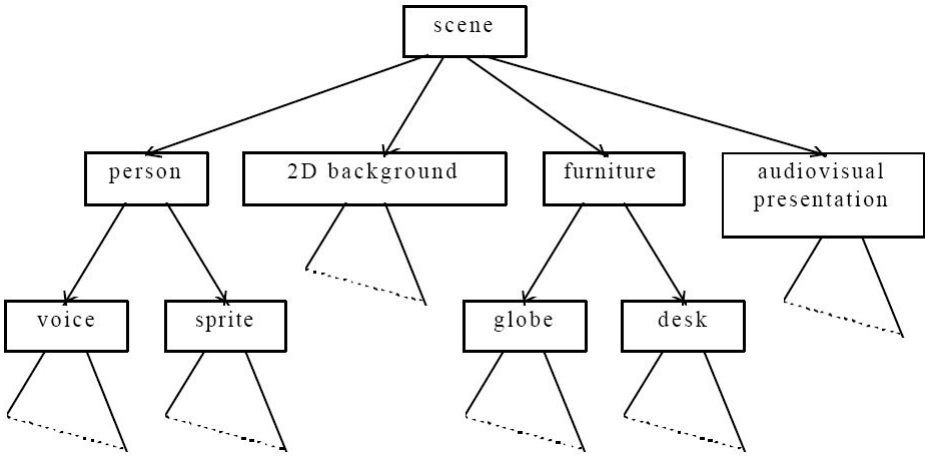

Abbildung 4: Beispiel für einen Szenengraphen in MPEG-4

Eine Szene ist ein gerichteter azyklischer Baumgraf, dessen Knoten die einzelnen Medienobjekte darstellen. Jedes Medienobjekt in MPEG-4 hat sowohl eine räumliche als auch eine zeitliche Ausdehnung und Position. Es wird in beiden Koordinaten relativ zu seinem Elternknoten platziert. [TMM]

1 zum Beispiel in Form einer URL (Universal Resource Locator) oder einem Verweis auf einen Track innerhalb einer MP4-Datei

2 Digital Rights Management

1.6.1.3 MPEG-J

MPEG-J spezifiziert eine Umgebung und ein API[1] für die Programmiersprache Java innerhalb des MPEG-4 Standards. Dadurch wird es unter anderem möglich, Inhalte programmgesteuert mit Logik zu erweitern sowie mit dem Endgerät/-benutzer zu interagieren.

Werden diese Programme zusammen mit anderen multimedialen Inhalten übertragen ("gestreamed"), so spricht man von MPEGlets. MPEG-J definiert auch einen Mechanismus für den getrennten Transport von MPEGlets und Java-Klassen und -Objekten.

Abbildung 5 verdeutlicht den Aufbau und die Integration von MPEG-J.

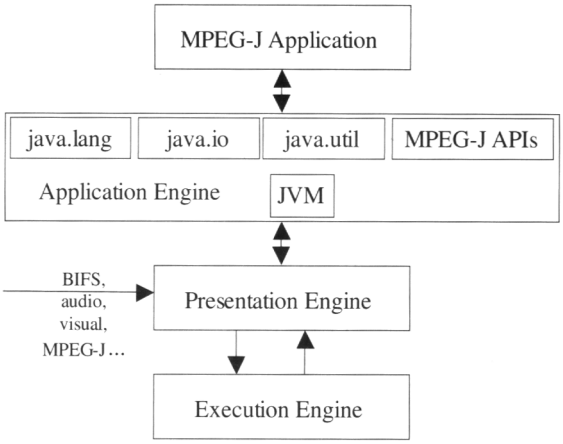

Abbildung 5: MPEG-J Application Engine

Nachfolgendes UML[2]-Diagramm zeigt die Interfaces und Klassen im Paket `org.iso.mpeg.mpegj`. Sie zusammen stellen das Terminal API dar.

1 Application Programming Interface
2 Unified Modeling Language

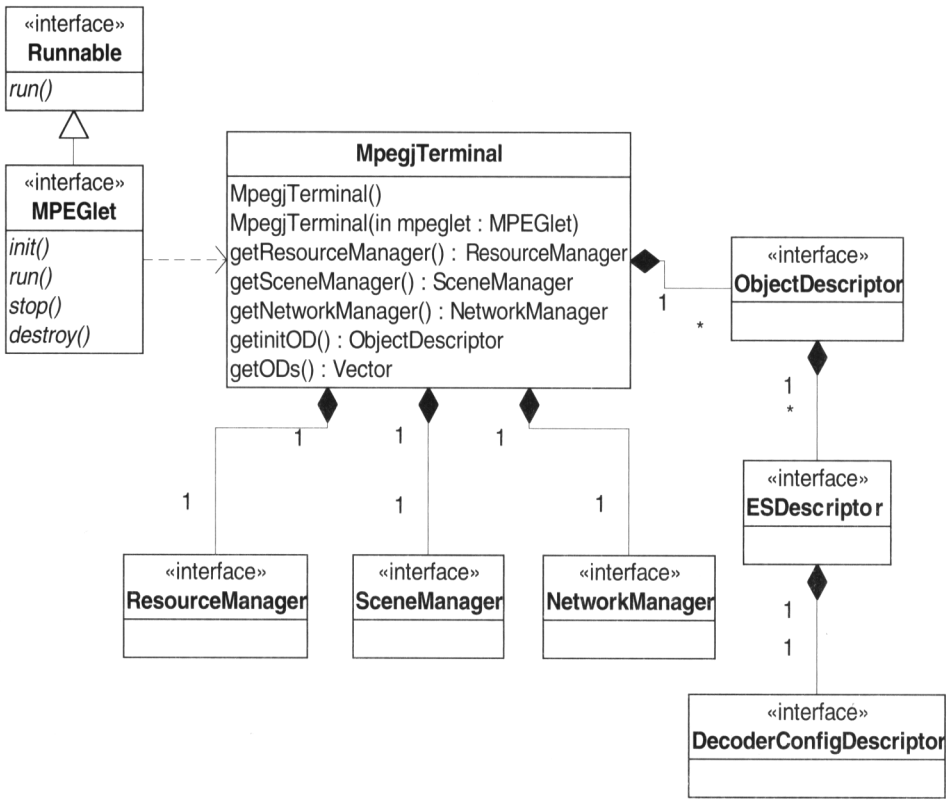

Abbildung 6: UML-Diagramm des Terminal API

1.6.1.4 eXtensible MPEG-4 Textual Format (XMT)

XMT beschreibt eine textuelle, auf XML basierende Variante der binären MPEG-4 Repräsentation. XMT ermöglicht damit den einfachen Austausch von Inhalten zwischen Autoren, verschiedenen Softwarewerkzeugen oder auch Anbietern solcher Inhalte.

Durch die Unterstützung von X3D, einer erst kürzlich verabschiedeten XML-Variante von VRML[1] und der Synchronized Multimedia Integration Language (SMIL) wird zudem die Interoperabilität zu bestehenden Formaten gewahrt und damit die einfache Nutzung durch bereits etablierte Technologien ermöglicht.

XMT ist mitunter Themenschwerpunkt dieser Arbeit und wird in Kapitel 2 im Detail erläutert.

1.6.1.5 Transport- und Speichermechanismen

Obwohl Transport und Speicherung von Multimedia-Daten nicht den Schwerpunkt des MPEG-4 Standards bilden, wurden hier dennoch zwei Werkzeuge definiert:

- **MP4-Dateiformat**
  Das MP4-Dateiformat baut auf das QuickTime-Dateiformat von Apple auf und bildet eine Art Container für verschiedenste Inhalte. Metadaten, wie

1 Virtual Reality Modeling Language

etwa Zeitinformation oder Eigenschaften zu einzelnen Inhalten, werden dabei innerhalb der Datei getrennt von den Mediendaten gespeichert. Dies bildet eine der Grundlagen für ein hochflexibles, gut erweiterbares Dateiformat für jede Art von Einsatz (zum Beispiel Live-Aufnahme, Erstellung, Editierung, Archivierung, Streaming) und allerlei Medien wie Video, Audio, Grafik, Text, usw.

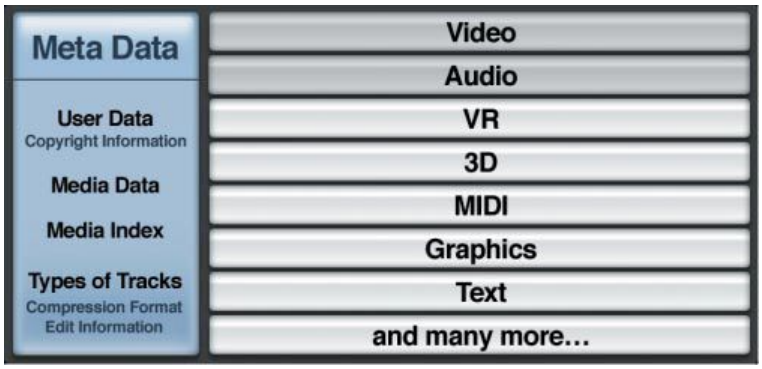

Abbildung 7: Das MP4-Dateiformat

- **FlexMux**
  Mit FlexMux wird eine Technik beschrieben, wie mehrere Datenströme inklusive Zeitinformationen zu einem einzigen Datenstrom zusammengefasst werden können. Bei der Entwicklung von FlexMux stand eine große Flexibilität im Vordergrund, um so den verschiedenen Einsatzgebieten von MPEG-4 gerecht zu werden.

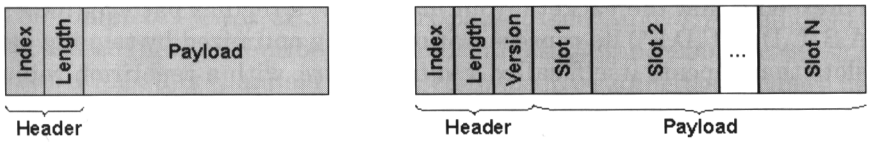

Abbildung 8: FlexMux (Simple-Mode und MuxCode-Mode)

- Wichtige Rahmenbedingungen, die mit FlexMux abgedeckt werden, sind:
  - ◇ Geringer Daten-Overhead
  - ◇ Geringe zeitliche Verzögerung
  - ◇ Variable Paketgröße

1.6.1.6 IPMP-Schnittstelle

Ein Teil der MPEG-4 Systems Spezifikation widmet sich der Benutzung von IPMP-Systemen (Intellectual Property Management and Protection). Dazu wurde eine API spezifiziert, deren Ziel es ist, solche Systeme einheitlich identifizieren und nutzen zu können.

Mit dieser Schnittstelle können das Management von Rechten - so genanntes DRM (Digital Rights Management) - und der Schutz von Inhalten verwirklicht werden.

1.6.2 Visual

MPEG-4 Visual erlaubt die hybride Kodierung und Kompression von natürlichen Bildern und Video, also basierend auf Pixeln, zusammen mit künstlichen, computergenerierten Inhalten. Letzteres beinhaltet neben 2D auch 3D Grafik.

Es soll hier nur ein Überblick über die Möglichkeiten der in diesem Teil des MPEG-4 Standards verwendeten Technologien gegeben werden. Weiterreichende Informationen und Erklärungen der Technologien im Detail finden sich in der MPEG-4 Spezifikation Teil 2 [MPEG4-2].

1.6.2.1 Videokomprimierung

Die eingesetzten Algorithmen bieten die effiziente Kompression von Video bei Bitraten zwischen 5 kBit/s und 1 GBit/s. Die Auflösungen bewegen sich hierbei zwischen QCIF (176 x 144 Pixel) und darunter bis zu Studioqualität mit 4000 x 4000 Bildpunkten.

Es werden dabei sowohl Voll- als auch Halbbilder unterstützt – im Englischen auch bekannt als Progressive und Interlaced. Texturen in 2D und 3D Szenen können ebenfalls in kompakter Weise kodiert werden. Der wahlweise Zugriff auf beliebige Stellen in einem Video bietet Möglichkeiten wie sie zum schnellen Vor- oder Zurückspulen in einem Video gebraucht werden.

1.6.2.2 Skalierung

MPEG-4 Visual unterstützt verschiedene Arten der Skalierung. Dies sind im Wesentlichen:

- Skalierung der Komplexität auf Seiten des Encoders und Decoders. Einem weniger leistungsstarken Decoder wird somit ermöglicht, nur bestimmte Teile eines Bitstreams[1] zu dekodieren. Die rekonstruierte Qualität steht im Allgemeinen in Beziehung zu der bei der Kodierung genutzten Komplexität.
- Räumliche Skalierung erlaubt einem Decoder, Bilder, Texturen und Videos mit einer geringeren Auflösung als der des Quellmaterials zu dekodieren. Dabei wird nur die dafür relevante Untermenge des Bitstreams dekodiert. Es werden maximal 11 Abstufungen unterstützt.
- Zeitliche Skalierung ermöglicht die Dekodierung mit einer geringeren zeitlichen Auflösung, wobei wie bei der räumlichen Skalierung, dazu nicht der komplette Datenstrom dekodiert werden muss. Drei Stufen werden unterstützt.

1 Datenstrom, eine Folge von digitalen Informationen

- Skalierung der Qualität erlaubt die Zerlegung eines Bitstreams in mehrere Streams mit unterschiedlichen Bitraten. Die Zerlegung kann bereits während der Übertragung oder erst im Decoder geschehen. Die rekonstruierte Qualität ist im Normalfall relativ zur Anzahl der für die Dekodierung genutzten Streams beziehungsweise Schichten.

1.6.2.3 Transparenz und willkürliche Formen

Mittels eines Alpha-Kanals in Form einer Graustufenabbildung ist es möglich, eine bestimmte Transparenz eines jeden Bildpunktes festzulegen. Durch die Möglichkeit, die Stärke der Transparenz nun innerhalb eines visuellen Objekts zu variieren, können zum Beispiel weiche Übergänge an Kanten erzeugt werden.

Mit Hilfe einer binären Abbildung kann zudem festgelegt werden, ob ein bestimmter Bildpunkt zu einem Objekt gehört oder nicht, also ob ein Bildpunkt dazugehört oder nicht. Dadurch sind nicht nur rechteckige sondern auch willkürliche Formen möglich.

1.6.2.4 Fehlertoleranz

Um Bilder und Video auch über fehleranfällige Medien transportieren beziehungsweise auf solchen speichern zu können, werden Mechanismen zur sicheren Übertragung beziehungsweise Speicherung sowie Erkennung von Fehlern unterstützt. Spezielle Erweiterungen adressieren zudem zusätzlich auftretende Probleme bei niedrigen Bandbreiten, wie etwa bei mobilen Anwendungen.

1.6.2.5 Animation von Gesicht und Körper

MPEG-4 spezifiziert mit FBA (Face and Body Animation) bestimmte Parameter, um künstlich generierte Gesichter und Körper definieren, kalibrieren und animieren zu können. Die 3D Modelle selbst sind dabei nicht standardisiert.

1.6.2.6 Kodierung von 2D und 3D Modellen

Verschiedene Werkzeuge und Definitionen ermöglichen die Kodierung von 2D und 3D Modellen. Neben der Beschreibung der Modelle in Form von Drahtgitter-Polygonen werden zudem Bewegung und Animation solcher Modelle sowie das dynamische Laden von Texturen spezifiziert. Die Parameter wie Bewegungsvektoren, Normalen, Farben, usw. können dabei platzsparend und komprimiert abgebildet werden. [N4668]

1.6.3 Audio

MPEG-4 Audio [MPEG4-3] deckt eine breite Schicht an Anwendungen ab – von der Sprache bis zu hoch qualitativen Mehrkanalton und von natürlichen bis zu synthetischen Tönen.

Zusammengefasst wird die hoch effiziente Repräsentation von Audio in verschiedenen Bereichen unterstützt. Die wichtigsten Anwendungsfelder sind nachfolgend aufgeführt.

1.6.3.1 Audio allgemein

Die Unterstützung in diesem Anwendungsbereich reicht von sehr geringen Bitraten[1] bis zu solchen mit hohen[2] oder höchsten Qualitätsansprüchen. Die niedrigsten Bitraten beginnen dabei bei 6 kBit/s pro Kanal und einer Bandbreite von unter 4 kHz.

Viele Komprimieralgorithmen bauen im Wesentlichen auf die schon von MPEG-2 her bekannte Technologie Advanced Audio Coding (AAC) auf, erweitern diese jedoch mit dem Ziel einer noch besseren Effizienz der Kompression und neuen Funktionalitäten wie Skalierung oder einem Modus für eine möglichst geringe Latenz während der Komprimierung.

Des weiteren existieren so genannte parametrische Komprimieralgorithmen wie HILN (Harmonic and Individual Lines plus Noise), welche das Eingangssignal in seine ausgeprägten Einzelkomponenten (harmonischer Ton, individuelle Sinusschwingungen und Rauschen) zerlegen. Dabei sind Bitraten von unter 4 kBit/s möglich.

1.6.3.2 Sprache

MPEG-4 Audio unterstützt Sprachkomprimierung bei Bitraten von 2 kBit/s bis zu 24 kBit/s. Zum Einsatz kommen dabei zwei Technologien: Code Excited Linear Prediction (CELP) und ein parametrischer Kodieralgorithmus mit dem Namen Harmonic Vector Excitation Coding (HVXC).

Mit Einsatz von variablen[3] Bitraten lassen sich Durchschnittswerte bis hinunter zu 1,2 kBit/s erreichen. Geringe zeitliche Verzögerungen der Codecs zielen speziell auf Anwendungen im Kommunikationsbereich ab. HVXC erlaubt dem Anwender zudem die Änderung von Schnelligkeit und Tonhöhe während dem Abspielen von Sprache.

1 Mit Bitrate wird die zur Repräsentation der digitalen Informationen benötigte Datenmenge pro Zeiteinheit (meistens pro Sekunde) beschrieben.

2 Das sehr weit verbreitete MP3-Format (MPEG-1 Layer 3) bietet bei einer Bitrate von 128 kBit/s CD-Qualität. High-Efficiency (HE) AAC, eine Erweiterung um SBR (Spectral Band Replication), erreicht dieses im Vergleich schon bei 48 kBit/s. [MPEGIF]

3 Bei einer variablen Bitrate kann die Datenmenge innerhalb eines bestimmten Zeitraums unterschiedlich verteilt werden. Eine Sprechpause wird zum Beispiel stärker komprimiert und die dadurch gesparte Datenmenge für das Gehör beziehungsweise Verständnis relevanteren Teile aufgewendet. Das Ziel ist eine insgesamt subjektiv bessere Qualität.

1.6.3.3 Künstliche Töne

Mit Hilfe einer strukturierten Sprache ist es möglich, Töne und Musik anhand von Instrumenten und Geräuschen in Verbindung mit Noten zu beschreiben. Diese Technologie entspricht dem populären MIDI-Standard (Musical Instrument Digital Interface), welcher ebenfalls unterstützt wird.

1.6.3.4 Künstliche Sprache

So genannte TTS-Werkzeuge (Text-To-Speech) ermöglichen es, aus Text eine künstliche Sprachausgabe zu erzeugen. Diese kann anhand von verschiedensten Parametern in Tonhöhe, Stimme, Dauer eines Lautes, Alter und Geschlecht des Sprechers, usw. variiert werden.

Die Bitraten von skalierbarer TTS-Kompression reichen dabei von 200 Bit/s bis zu 1,2 kBit/s. Es werden zudem internationale Sprachen, Phoneme und Dialekte sowie die Synchronisation mit FBA (Face and Body Animation) unterstützt. [N4668]

1.6.4 DMIF

Mit dem so genannten DMIF (Delivery Multimedia Integration Framework), definiert in Teil 6 [MPEG4-6] der MPEG-4 Spezifikation, wurde eine Schnittstelle zwischen Anwendungen auf der einen und Netzwerken auf der anderen Seite definiert.

Die eigentliche Schnittstelle, das DMIF Application Interface (DAI), wird dabei vom DMIF Signaling Protocol, welches zur Konfiguration der Beteiligten oder der Einhaltung von Quality of Service (QoS) genutzt wird, unterstützt.

Die Anwendung und der Transport von MPEG-4 Daten sind somit flexibel an verschiedene Übertragungstechnologien anpassbar. Damit wird angestrebt, dass einmal erstellte Inhalte auf allen beliebigen Speicher- und Übertragungsmedien abgespielt und genutzt werden können.

Das MPEG-4 DMIF unterstützt dabei folgende Funktionalitäten:

- Eine transparente Schnittstelle zu Anwendungen, um so eine Abhängigkeit von der Herkunft der Daten zu vermeiden.
- Möglichkeiten zur Kontrolle der dabei eventuell verwendeten Kanäle
- Benutzung von homogenen Netzwerken zwischen interaktiven Endpunkten wie IP, ATM[1], ISDN[2], etc.
- Unterstützung von mobilen Netzwerken
- Benutzerkommandos inklusive Benachrichtigung zur Bestätigung
- Management des MPEG-4 Sync Layer (Synchronisationsschicht)

1 Asynchronous Transfer Mode
2 Integrated Services Digital Network

Folgende drei "Hauptkategorien" des Medientransports werden unter anderem abgedeckt:

- **Rundfunk**
  - Kabel
  - Satellit
  - Terrestrisch
- **Interaktive Netzwerke**
  - Internet (IP)
  - ATM
  - etc.
- **Lokale Datenspeicher**
  - Festplatte
  - DVD/CD
  - etc.

Um MPEG-4 bereits in existierenden Umgebungen einsetzen zu können, wurden in Anlehnung an vorangegangene Kategorien jeweils ein netzwerkspezifischer Transportmechanismus definiert:

- Übertragung von MPEG-4 Inhalten in einem MPEG-2 Transport-Stream
- Übertragung von MPEG-4 Inhalten über das Internet
- Speicherung von MPEG-4 Inhalten in MP4-Dateien

[N4668]

1.7 Profile und Levels

Um die Interoperabilität zwischen verschiedenen MPEG-4 Implementierungen zu gewährleisten, werden wie auch schon bei den beiden Vorgängern MPEG-1 und MPEG-2, Profile und Levels eingesetzt.

Profile und Levels ermöglichen Anwendungen für nur spezielle Einsatzgebiete, ohne dass diese Anwendungen den gesamten MPEG-4 Standard in all seinen Möglichkeiten beherrschen müssen. Ein Audioplayer muss so zum Beispiel nicht wissen, wie er Videodaten zu behandeln hat.

Profile und Levels spezifizieren Punkte wie:

- Unterstützung der verschiedenen Tools (Technologien)
- Bitrate
- Bildgröße
- Anzahl der Objekte

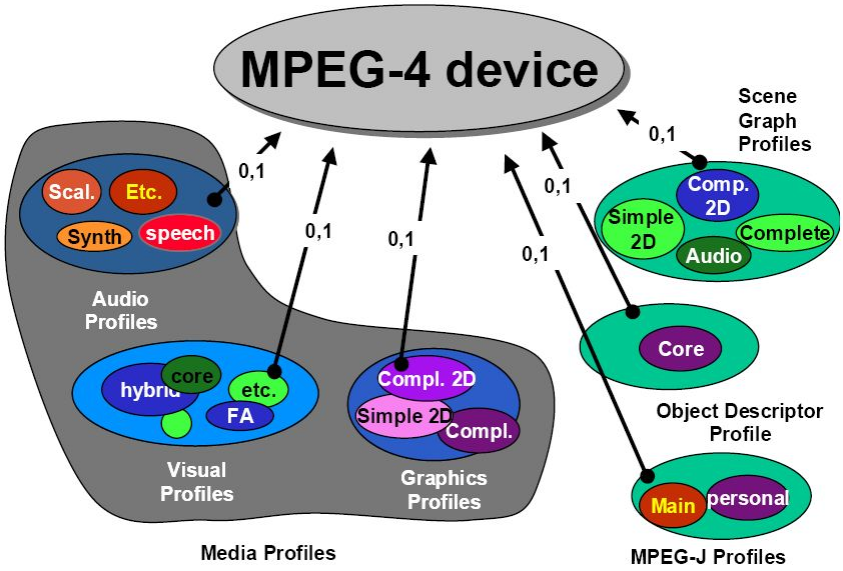

Abbildung 9: Profile und Levels

Profile bilden funktionelle Gruppen aus den verschiedenen MPEG-4 Tools innerhalb einer Kategorie, wie zum Beispiel Visual oder Audio. Zum jetzigen Zeitpunkt existieren Profile für folgende Kategorien:

- Visual
- Audio
- Graphics
- Scene Graph
- Object Descriptors (OD)
- MPEG-J

Im Gegensatz zu Profilen legen Levels die Komplexität, wie zum Beispiel Auflösung oder Bitrate, innerhalb eines Profils fest.

Außer der Interoperabilität bilden Profile und Levels die Grundlage für Konformitätstests, wobei Profile und Levels immer gemeinsam betrachtet werden. Endgeräte, welche ein bestimmtes Profil bei einem bestimmten Level unterstützen, sollten sich daher konform zueinander verhalten.

Das Beispiel Simple Profile @ Level 1 legt unter anderem ein

- fehlertolerantes, rechteckiges und natürliches Videoobjekt mit Unterstützung von I (Intra) und P (Predicted) Video Object Planes (VOPs)

sowie

- einer maximalen Anzahl von 99 Makroblöcken[1] (entspricht zum Beispiel 176 x 144 Pixel = QCIF[2])

fest. [MPEGIF]

1.8 Versionen

Mit MPEG-4 wurde ein weitgehend flexibler Standard geschaffen, welcher auch in Zukunft um Erweiterungen ergänzt werden kann.

Profile spielen hierbei eine wichtige Rolle, da sie klare Abgrenzungen zwischen den verschiedenen Funktionalitäten beziehungsweise Technologien schaffen. Entscheidend ist also weniger die Unterscheidung von Versionen, welche nur eine Art Zusammenfassung von Profilen darstellen, sondern vielmehr die der Profile selbst.

Bestehende Funktionalitäten und Profile werden dabei niemals durch nachfolgende Versionen geändert oder ersetzt, sondern immer nur durch neue Technologien in Form von neuen Profilen ergänzt.

Die nachfolgende Abbildung zeigt den Zusammenhang zwischen den Versionen. Version 2 ist dabei rückwärts kompatibel zu Version 1, und Version 3 zu Version 2 – und so weiter.

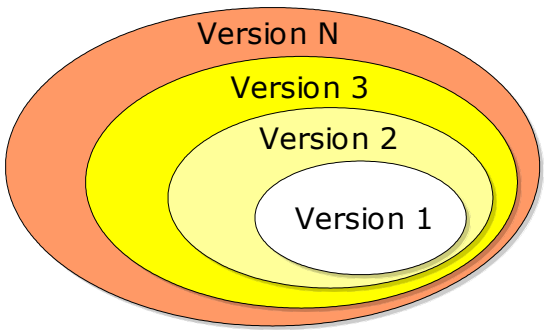

Abbildung 10: Zusammenhang zwischen Versionen in MPEG-4

MPEG-4 Version 1 wurde 1998 von der MPEG verabschiedet. Aktuell liegt Version 2 vor. [N4668]

1 16 x 16 Pixel
2 Quarter Common Intermediate Format

1.9 MPEG-4 in der Praxis

In diesem Kapitel soll ein Überblick über MPEG-4 hinsichtlich praxisnaher Aspekte wie Anwendungsbereiche oder Kosten gegeben werden.

1.9.1 Anwendungsfelder

Mit MPEG-4 wird versucht, eine möglichst breite Schicht an Anwendungen zu adressieren.

Neben den bereits bekannten Anwendungen werden auch viele neue, teils noch unbekannte Anwendungsbereiche angestrebt - angetrieben von den zahlreichen Neuerungen und Verbesserungen in MPEG-4.

Anders als MPEG-2, wo die "Killeranwendung" digitales Fernsehen mit Vertretern wie DVD und DVB war und somit nicht mehr als eine digitale Umsetzung vom bereits vorhandenen analogen Fernsehen darstellt, zielt MPEG-4 nicht auf eine spezielle Klientel von Anwendungen und Bereichen ab, sondern öffnet sich für alle Anwendungsfelder gleichermaßen.

Mögliche Anwendungen können sich zum Beispiel erstrecken über:

- Inhaltsbasierte Archivierung (besonders in Zusammenspiel mit MPEG-7)
- Computerspiele
- Mit Zusatzinformationen und Interaktion aufgewertetes digitales Fernsehen (DVB, DVD, Set-Top-Boxen)
- (Fern-) Überwachung
- Persönliche Kommunikation (Videotelefonie)
- Mobile Multimedia-Anwendungen
- Virtuelle Einrichtungen und Treffen (Avatare, interaktiver Chat)
- Streaming von Inhalten über das Internet
- Video-on-Demand, Video-over-DSL/IP
- Infotainment
- Postproduction in Studio und Fernsehen

[N2724]

1.9.2 Kosten und Lizenzierung

Um MPEG-4 in der Praxis einsetzen zu dürfen, müssen Lizenzgebühren bezahlt werden.

Da MPEG-4 von vielen unterschiedlichen Firmen und Institutionen entwickelt worden ist, wurden verschiedene Organisationen geschaffen oder beauftragt, die

Gebühren der jeweils verwendeten MPEG-4 Teile beziehungsweise Technologien einzufordern und entsprechend den Firmen und Institutionen, welche mitunter Patente auf im MPEG-4 Standard verwendete Technologien besitzen, zu verteilen.

Organisationen, die solche Lizenzen verwalten, sind zum Beispiel:

- MPEG LA [MPEGLA] für MPEG-4 Systems, MPEG-4 Visual und DRM/IPMP
- Via Licensing Corporation [VIA] für MPEG-4 Audio

Da die Lizenzierungsbedingungen von den jeweils verwendeten Technologien (zum Beispiel MPEG-4 AAC) und den jeweils angestrebten Einsatzbereichen abhängig und daher sehr komplex sind, kann hier nur ein kleiner Überblick über die zu entrichtenden Gebühren gegeben werden. Nähere Informationen finden sich bei den vorher genannten Organisationen.

- Endanwender müssen im Allgemeinen keine Gebühren für die Nutzung von MPEG-4 Anwendungen und Inhalten entrichten.
- Hersteller und Entwickler von MPEG-4 Anwendungen oder Teilen davon (zum Beispiel Encoder oder Decoder), welche für den Endnutzer bestimmt sind, müssen dagegen Lizenzgebühren entrichten. Diese hängen mitunter von verschiedenen Faktoren wie Anzahl der Kanäle bei Audio, professioneller Einsatz (zum Beispiel im Rundfunk) oder nur Einsatz beim Konsumenten, ab. Meist muss zu einer einmalig zu entrichtenden Gebühr noch eine jährliche Pauschale gezahlt werden. Die eigentliche Lizenzgebühr ist oftmals im Preis der verkauften Produkte nach gestaffelt. Am Beispiel von MPEG-4 AAC sind dies:
  - $15.000 einmalig
  - $0.50 für jeweils Encoder beziehungsweise Decoder bei einem Volumen von 1 bis 100.000. Bei einem Volumen von über 10 Millionen müssen nur noch $0.12 entrichtet werden. Professionelle Encoder und Decoder kosten unabhängig vom Volumen $20.00 beziehungsweise $2.00. Die Preise richten sich hierbei immer pro Kanal. Handelt es sich um eine PC-Software, sind die Preise wiederum anders gestaffelt – außerdem ist dort eine maximal zu entrichtende Gebühr festgelegt. Im Vergleich dazu liegen die standardmäßig zu entrichtenden Gebühren für Encoder/Decoder von MPEG-4 Visual und MPEG-4 Systems bei $0.25/$0.25 beziehungsweise $0.25/$0.15.
- Für Anbieter von kommerziellen Downloads, Video-on-Demand, Pay-per-View, usw. fallen in der Regel (zum Beispiel bei MPEG-4 Visual) Lizenzgebühren je nach Länge des angebotenen Inhalts an. Videoportale und Rundfunksender zahlen meist eine jährliche Pauschale. Jedoch gibt es auch hier Ausnahmen und Staffelungen. [MPEGIF] [MPEGLA] [VIA]

2 XMT: eXtensible MPEG-4 Textual Format

XMT – das eXtensible MPEG-4 Textual Format – stellt ein Rahmenwerk zur Darstellung von MPEG-4 Systems Inhalten und deren Verknüpfung mit audio-visuellen Medien auf Basis einer XML-basierten Syntax dar.

Die hohe Abstraktion und textuelle Darstellung von XMT erleichtert gegenüber BIFS (Binary Format for Scenes) - der binär codierten Szenenbeschreibung von MPEG-4 – und dem OD-Rahmenwerk (Object Descriptor) die Erstellung und Pflege von solchen Multimedia-Inhalten unter Beibehaltung der Semantik.

XMT ist daher als bevorzugtes Format zum Austausch von Inhalten zwischen Autoren und Autorenwerkzeugen anzusehen.

Alle Informationen des Kapitels sind, sofern nicht zusätzlich angegeben, [TMPEG4B], [MPEG4JS] und der zugrunde liegenden Spezifikation von MPEG-4 Systems [MPEG4-1] entnommen.

2.1 Cross-Standard Interoperabilität

Neben der autorenfreundlichen Abstraktion der hinter MPEG-4 liegenden Technologien stand auch die Überlegung, Autoren, welche bisher mit anderen gängigen Multimedia-Standards wie XML, HTML, VRML, SMIL oder X3D vertraut waren, entgegen zu kommen. Dies fördert einen schnellen Einstieg in die MPEG-4 Terminologie durch die weitere Anwendung bekannter Praktiken und Fertigkeiten.

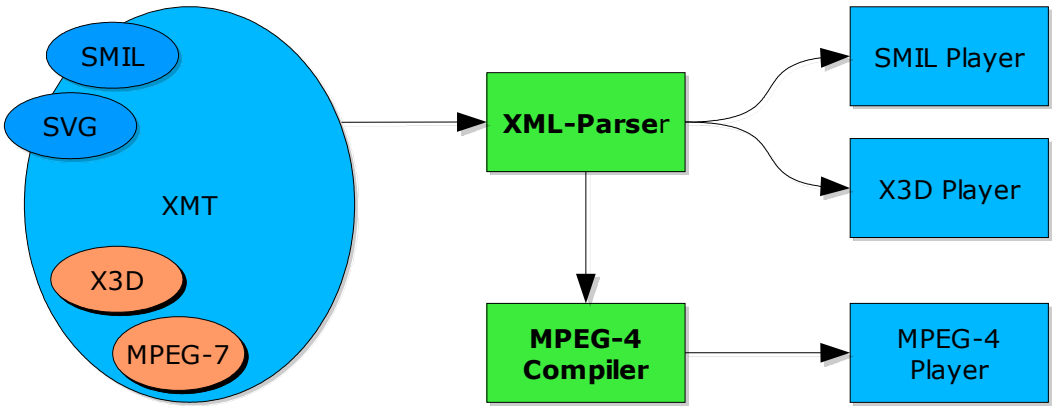

Abbildung 11: Beziehungen von XMT

Wie vorangegangene Abbildung zeigt, versucht XMT darüber hinaus so weit wie möglich zu bestehenden Formaten (X3D, SMIL, SVG) interoperatibel zu bleiben. Dies erlaubt Erstellern von Inhalten bereits vorhandenes Material in MPEG-4 (es werden nicht alle Möglichkeiten von SMIL und SVG unterstützt) wiederzuverwenden.

Darüber hinaus werden mit XMT neue Funktionalitäten im Vergleich zu bereits bestehenden Formaten eingeführt:

- gemeinsame Darstellung von 2D und 3D
- Ereignismodell (abgeleitet von VRML)

Diese Interoperabilität ermöglicht, dass Inhalte ohne größere Neuimplementierungen sowohl in SMIL, VRML/X3D und MPEG-4 Playern abgespielt werden können.

Nicht zuletzt trägt auch die Verwendung "des" Standardformats XML zu einer schnellen Akzeptanz bei. Inhalte auf Basis von XMT, lassen sich so leicht durch einfach zu erstellende Skripte – beispielsweise mittels XSLT (eXtensible Stylesheet Language Transformation) – generieren beziehungsweise durch Transformation aus vorhandenem Material erstellen.

Daneben können sowohl neue als auch bestehende Softwareentwicklungen leicht - falls nicht schon implementiert - mit einer Vielzahl an erprobten und standardisierten XML-Bibliotheken aufgewertet werden, um so eine Schnittstelle für den Export oder Import von XMT-Inhalten zu bieten.

2.2 Zwei-Schichten-Architektur

XMT wurde in einer Zwei-Schichten-Architektur entworfen. Die zwei einzelnen Schichten werden mit den Namen XMT-Ω und XMT-A bezeichnet.

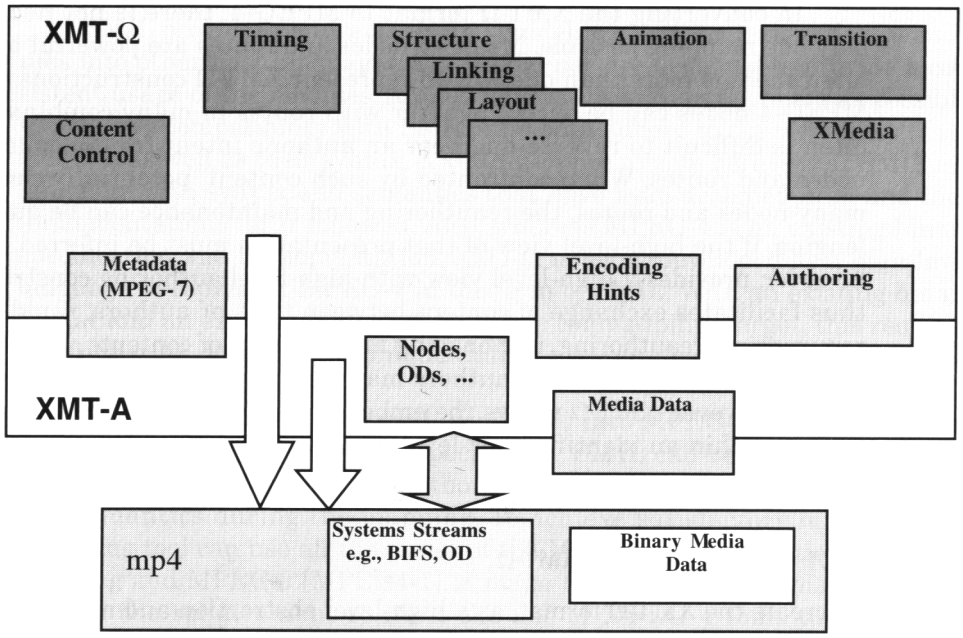

Abbildung 12: Zwei-Schichten-Architektur von XMT

XMT-Ω ist ein Format mit hoher Abstraktion und basiert auf der etablierten Multimedia-Sprache SMIL 2.0[1].

XMT-A ist lediglich eine direkte XML-Repräsentation des binären MPEG-4 Systems Formats. Da MPEG-4 Systems wiederum zu großen Teilen auf VRML basiert, hat auch das XMT-A Format viel mit X3D, dem ebenfalls in XML-Syntax gehaltenen VRML-Nachfolger, gemeinsam.

2.3 XMT-Ω Format

Das Ziel des XMT-Ω Formats ist die einfache Erstellung von Inhalten und deren Austausch zwischen Autoren und Werkzeugen sowie die Interoperabilität mit SMIL 2.0[2].

In MPEG-4 werden Objekte in einem Szenengraph als Knoten repräsentiert, wobei solche Knoten auch Verbindungen zu audio-visuellen ESs (Elementary Streams), also nicht textuell darstellbaren Medien wie Audio oder Video, beinhalten können. Die Interaktivität zwischen diesen Knoten kann in Form eines Mechanismus für ein- und ausgehende Ereignisse beschrieben werden.

Dem XMT-Ω Format liegt dabei ein sehr hohes Abstraktionsniveau zugrunde, was Autoren erlaubt, ihre Intention direkt mittels Medien-, Zeit- und Animationsinformationen auszudrücken, anstatt auf "low-level"-Ebene dies wie bei XMT-A mit Knotenverbindungen tun zu müssen – dazu später mehr im Kapitel XMT-A Format.

XMT-Ω kann mittels Compilern und Encodern zu binären MPEG-4 Datenströmen wie BIFS, ODs, audio-visuellen Medien usw. umgewandelt werden, um sie so effizienter verteilen und abspielen beziehungsweise ihre Inhalte vor etwaigen Manipulationen besser schützen zu können.

Wie eine Szenenbeschreibung basierend auf XMT-Ω letztlich zu XMT-A oder einem binären MPEG-4 Datenstrom (BIFS) kodiert wird, ist dabei nicht festgelegt, da es mehrere Möglichkeiten gibt, die XMT-Ω Konstrukte umzusetzen (zum Beispiel mittels MPEG-4 Replace Field BIFS Update-Kommando oder MPEG-4 TimeSensors).

Auf die Informatik und Programmierung bezogen, könnte man XMT-Ω als eine Hochsprache wie C ansehen, welche mittels einem Compiler erst zu Assemblercode (XMT-A) und dann zu binärem Maschinencode (BIFS/OD) verarbeitet wird. Assemblercode lässt sich dabei jederzeit wieder aus Maschinencode generieren und umgekehrt, nicht aber der ursprüngliche Code der Hochsprache und deren Semantik.

1 SMIL bezieht sich im Rahmen dieser Diplomarbeit soweit nicht anders angegeben immer auf SMIL 2.0

2 Die Synchronized Multimedia Integration Language (SMIL) ist eine Sprache zur Beschreibung von interaktiven 2D Multimedia-Präsentationen.

XMT-Ω bietet die Möglichkeit, Knoten und Routen auf "low-level"-Ebene mittels eingebetteter XMT-A Knoten und Routendefinition direkt mit einzubeziehen, um so spezielle Medienkonstrukte zu ermöglichen.

2.3.1 Dokumentstruktur

Die allgemeine Dokumentstruktur besteht aus einem Kopf und einem Hauptteil, in welchem der eigentliche Inhalt steht beziehungsweise spezifiziert ist.

Im Kopf können globale Definitionen wie Layout, Metainformationen[1], häufig verwendete Konstanten wie Schriftarten oder Makros (im Sinne von Templates[2]) festgelegt werden.

Der genaue Aufbau wird anhand eines kompletten Beispiels in Kapitel 3.1.1 Schritt für Schritt erklärt.

2.3.2 Primitiven

XMT-Ω unterstützt eine Vielzahl grundlegender Primitiven wie:

- `<points>` um Punkte zeichnen zu können
- `<lines>` zur Darstellung von Linien
- `<rectangle>` zur Darstellung von Rechtecken
- `<circle>` um Kreise darzustellen
- `<polygons>` zur Anzeige willkürlich geformter gerader Flächen
- `<curve>` zur Darstellung kurviger Linien und Flächen
- `<text>` zur Einbindung und Darstellung externer Textdateien
- `<string>` zur Anzeige von Text
- `<img>` für Bilder
- `<video>` um Videos darzustellen
- `<audio>` zur Ausgabe von Audio

Im 3D Raum stehen außerdem unter anderem die Primitiven `<box>`, `<cone>`, `<cylinder>`, `<sphere>` und `<mesh>`[3] zur Verfügung. Auf deren Eigenschaften und 3D im Allgemeinen wird im Rahmen dieser Arbeit nicht weiter eingegangen.

2.3.3 Eigenschaften grafischer Elemente

Das Aussehen eines fast jeden grafischen Objektes lässt sich mit den Elementen `<material>`, `<outline>`, `<chromakey>` und `<texture>` beeinflussen beziehungsweise festlegen.

1 beispielsweise Titel, Autor oder Copyright
2 parametrisierbare Schablonen
3 genutzt zur Darstellung willkürlich geformter, gerader Flächen im 3D Raum

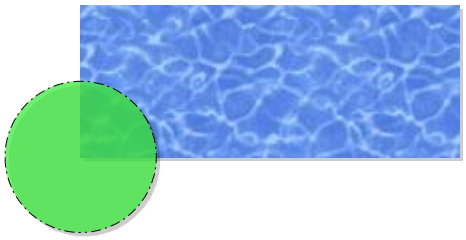

Abbildung 13: Rechteck und Kreis

```
<rectangle size="500 200">
  <transformation tranlation="250 100">
  <texture src="WaterTexture.jpg"
           translation="50 50" rotation="45grad"/>
</rectangle>

<circle radius="200">
  <material color="green" filled="true" transparency="0.25"/>
  <outline color="black" width="3" style="dash-dot"/>
</circle>
```

Vorangegangener Auszug stellt zwei Objekte dar. Einen grün ausgefüllten, leicht transparenten Kreis. Dieser ist von einer schwarzen Strich-Punkt-Linie umrundet. Das Rechteck – also das erste Objekt – ist mit dem Bild einer Wassertextur ausgefüllt. Die Textur ist um jeweils 50 Pixel nach rechts oben versetzt und um 45 gedreht. Es ist auch möglich, ein Video als Textur zu verwenden.

Mit dem Element `<chromakey>` lässt sich eine Farbmaske über ein Bild oder Video legen, um bestimmte Farbbereiche auszublenden beziehungsweise transparent erscheinen zu lassen. Der "aktive" Bereich, in welchem beispielsweise ein Mausklick registriert werden würde, ändert sich dabei dementsprechend mit.

```
<img src="FlowerOnBlueBackground.jpg">
  <chromakey keyColor="blue" lowThreshold="0.1"/>
</img>
```

2.3.4 Erweiterbare Medienobjekte (xMedia)

Erweiterbare Medienobjekte (xMedia-Objekte) sind Objekte wie `<img>` oder `<rectangle>`, welche durch untergeordnete Elemente[1] mit erweiterten, in SMIL nicht vorhandenen Eigenschaften – beispielsweise `<material>` - näher beschrieben werden können.

Das kommt daher, dass MPEG-4 im Gegensatz zu SMIL in diesem Bereich viel mehr Möglichkeiten bietet, man aber bei bereits bestehenden Objekten und deren Attributen keine Veränderungen mehr vollziehen möchte, um somit zu SMIL kompatibel zu bleiben.

1 bei XML spricht man auch von Kind-Knoten

2.3.5 Zeitsteuerung und Synchronisation

Elemente in XMT-Ω können mit Hilfe der aus SMIL bekannten Timing and Synchronisation Module zeitlich arrangiert werden. Dabei stehen sowohl Elemente wie auch Attribute zur Steuerung und Synchronisation von Medienobjekten, Animationen, usw. bezüglich deren Darstellung zur Verfügung.

Drei Elemente zur Synchronisation fungieren dabei als eine Art Zeitcontainer, indem sie die Zeitsteuerung von untergruppierten Elementen erlauben. Dies sind:

Element	Beschreibung
`<par>`	Spielt eine oder mehrere untergeordnete Elemente parallel ab.
`<seq>`	Untergeordnete Elemente werden sequenziell abgespielt.
`<excl>`	Spielt immer nur ein untergeordnetes Element gleichzeitig ab – die Reihenfolge ist dabei nicht von Bedeutung.

Tabelle 1: Elemente zur Synchronisation

Die Timing Module definieren Attribute, mit welchem sich das zeitliche Verhalten von Elementen definieren lässt. Die wichtigsten Attribute sind dabei:

Attribut	Beschreibung
`dur`	Legt die Dauer eines Elements fest.
`begin`	Spezifiziert die Startzeit eines Elements in einer Vielfalt an Möglichkeiten, zum Beispiel Sekunden oder Ereignissen wie ein Mausklick.
`end`	Definiert das Ende in ähnlicher Weise wie `begin` bezüglich den Möglichkeiten.
`min`	Legt die minimale aktive Dauer eines Elements fest.
`max`	Legt die maximale aktive Dauer fest.

Tabelle 2: Wesentliche Attribute zur Synchronisation

Nachstehend ist die Benutzung der Zeitcontainer `<par>` und `<seq>` sowie der Attribute `dur` und `begin` verdeutlicht:

```
<par begin="5s">
  <rectangle size="10 10" dur="5s"/>
  <img src="my.jpg" begin="1s" dur=6s"/>
  <seq>
    <polygons coord="10 10; 34 45; 23 12" dur="10s"/>
    <circle radius="50" dur="2s"/>
  </seq>
</par>
```

Das Element `<par>` beginnt bei t=5s. Dabei werden auch das Rechteckobjekt sowie das Polygonobjekt innerhalb des `<seq>` Knotens angefangen abzuspielen. Bei t=6s, also t=1s relativ zu `<par>`, startet das Objekt `<img>` – zu diesem Zeitpunkt werden also alle 3 Objekte parallel abgespielt. Bei t=10s endet das Rechteck, bei t=12s das `<img>` Objekt. Nach t=15s endet `<polygons>` und das nächste Element in der Sequenz - in diesem Falle der Kreis – wird für 2 Sekunden angezeigt. Bei t=17s endet dann sowohl der Kreis als auch das umschließende `<par>` Element. Nachfolgende Abbildung skizziert den zeitlichen Verlauf auf:

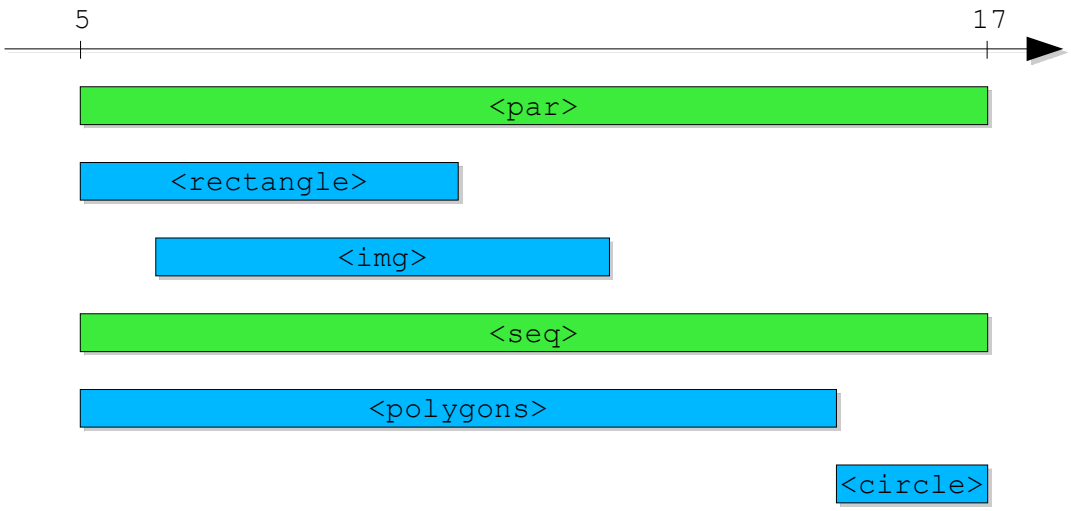

Abbildung 14: Zeitlicher Verlauf

XMT-Ω bietet außer den in Tabelle 2 gezeigten Attributen noch weitere, teilweise MPEG-4 spezifische Attribute, die hier aber nicht weiter erklärt werden sollen. Dies sind: `repeatDur` und `repeatCount`, `fill`, `endsync`, `restart` und `restartDefault`, `syncBehaviour` und `syncBehaviorDefault` sowie `SyncMaster`.

2.3.5.1 Ereignisbehandlung

Wie schon bei dem Attribut `begin` angedeutet, kann die zeitliche Steuerung auch ereignisbasiert sein. Dies ist einer der Grundlagen für interaktiven Inhalt.

Folgendes XMT-Ω Fragment stellt einen Kreis dar, nachdem das Objekt `myButton` angeklickt wurde.

```
<circle radius="50" begin="myButton.click">
  <material color="green"/>
</circle>
```

Neben `click` gibt es noch einige weitere Ereignisse, auf welche reagiert werden kann. Dies sind unter anderem:

Ereignis	Beschreibung
`click`	Mausklick
`mouseup`	Maustaste drücken
`mousedown`	Maustaste loslassen
`mouseout`	Objekt mit Maus verlassen
`mouseover`	Objekt mit Maus berühren
`mousedrag`	Objekte mit Maus ziehen
`viewable`	Sichtbarkeit von Objekten feststellen
`near`	Annäherung von Objekten detektieren
`collide`	Kollision von Objekten detektieren

Tabelle 3: Ereignisse in XMT-Ω

Bis auf `viewable`, `near` und `collide` sind eben vorgestellte Ereignisse sowohl im 2D als auch 3D Raum definiert. Letztere werden nur im 3D Raum unterstützt.

Nachstehendes Beispiel spielt eine Audiodatei ab, sobald die zwei 3D Objekte `<sphere>` und `<box>` zusammenstoßen.

```
<head>
  <layout metrics="pixel">
    <topLayout width="640" height="480"/>
  </layout>
</head>
<body>
  <group id="ballBox" collide="true">
    <sphere id="ball" radius="8" dur="20s"/>
      <transformation tranlation="-100 50 30">
        <animateMotion to="0 0 0" dur="5s"/>
      </transformation>
    </sphere>
    <box id="crate" size="20 20 20" dur="20s"/>
  </group>
  <audio src="crash.wav" begin="ballBox.collide"/>
</body>
```

Ereignisse wie `click` beziehen sich auf die ganze Fläche eines Objektes. Um den sensitiven Bereich einzuschränken, können so genannte Hotspots definiert werden. Deren Form und zeitlicher Verlauf kann dabei dynamisch geändert werden, um so beispielsweise Objekte in einem Video verfolgen zu können.

2.3.5.2 Steuerung von Geschwindigkeit und Zeit

XMT-Ω bietet mit `speed`, `accelerate`, `decelerate` und `autoReverse` Möglichkeiten, die Abspielgeschwindigkeit und -folge zu steuern.

In folgendem Beispiel wird die Abspielgeschwindigkeit verdoppelt. Die gesamte Sequenz dauert somit nur 14 Sekunden.

```
<seq speed="2.0">
  <video dur="10s" ...>
  <video dur="18s" ...>
</seq>
```

Die Attribute `accelerate` und `decelerate` können überall dort angegeben werden, wo sich auch Angaben zur Dauer finden. In nachstehendem Auszug wird das Bild in den ersten drei Sekunden zuerst vom Stillstand aus auf die maximale Geschwindigkeit beschleunigt. Nach sechs Sekunden wird das Bild wieder bis zum Stillstand abgebremst. Solch ein "softer" Bewegungsablauf wird oftmals als "ease in, ease out" bezeichnet und führt zu realistischerer Darstellung.

```
<img ...>
  <animateMotion dur="12s" accelerate=".25" decelerate=".25" .../>
</img>
```

Mit `autoReverse` können schließlich Effekte wie Pendel oder Schwingungen dargestellt werden. Die Zeit "fließt" dabei vor und zurück.

2.3.5.3 FlexTime-Unterstützung

Das MPEG-4 FlexTime Modul wird innerhalb XMT-Ω durch `flexBehavior` und `flexBehaviorDefault` unterstützt. Damit können bei nicht vorherzusehenden Verzögerungen (zum Beispiel beim Streaming) Objekte in deren zeitlichen Verlauf angepasst werden. Wie dies geschieht, kann durch verschiedene Attribute festgelegt werden. Im einfachsten Fall - wie bei Bildern - könnte die Anzeigedauer entsprechend verlängert werden. Video oder Audio könnten hingegen schneller beziehungsweise langsamer abgespielt oder gar gekürzt werden. Bei einer Sprachaufzeichnung könnten wiederum die Pausen zwischen den einzelnen Wörtern angepasst werden.

2.3.6 Layout

Um Objekte räumlich anzuordnen und auszurichten, bietet XMT-Ω Elemente wie `<layout>`, `<topLayout>` oder `<region>`, welche zwar den Layout Modulen aus SMIL ähneln, aber zu diesen nicht kompatibel sind.

Dabei stehen die beiden Einheiten `pixel` und `meter` zur Verfügung. Letztere ist relativ und bezieht sich auf die Größe des Fensters, in welchem die Präsentation auf einem Endgerät abgespielt wird.

Mittels `<region>` lässt sich das Layout in mehrere Bereiche beziehungsweise Ebenen einteilen.

Das Element `<transformation>` bietet beispielsweise die Möglichkeit, Objekte zu positionieren, drehen, skalieren oder deren Ursprung festzulegen. In folgendem Beispiel wird das Rechteck um 100 Einheiten nach oben versetzt.

```
<rectangle size="50 70">
  <transformation translation="0 100"/>
</rectangle>
```

Objekte können darüber hinaus mit dem Element `<group>` gruppiert werden, um so allgemeine Manipulationen wie Positionierung oder Skalierung auf mehreren Objekten gleichzeitig vorzunehmen. Ein `<group>` Element verhält sich dabei wie der Zeitcontainer `<par>` und kann ineinander verschachtelt werden.

2.3.7 Animationen

Mit dem aus SMIL bekannten Animation Module, welches in XMT-Ω übernommen wurde, lassen sich Attribute von xMedia-Objekten animieren und so dynamische Inhalte realisieren.

Eines der einfacheren Kommandos ist dabei `<set>` – komplexere Anweisungen sind `<animate>`, `<animateColor>` oder `<animiateMotion>`.

Folgendes Beispiel zeigt einen roten Kreis, bei welchem die Farbe zuerst auf Orange gesetzt wird. Später wird die Farbe linear über die Farben Gelb, Blau und Grün interpoliert.

```
<circle radius="160" begin="6s" dur="35s">
  <material color="red">
    <set attributeName="color" to="orange" begin="3s" dur="6s"/>
    <animateColor attributeName="color" values="yellow; blue;
                  green" keyTimes="0; 0.4; 1" calcMode="linear"
                  begin="15s" dur="12s"/>
  </material>
</circle>
```

2.3.8 Transistionen

Über das Element `<transistion>` lassen sich sehr einfach verschiedene Überblendeffekte bei grafischen Objekten realisieren.

Folgendes verdeutlicht, wie ein Bild von rechts "hereingeschoben" wird:

```
<img src="HdMLogo.png" dur="indefinite">
  <transitionFilter type="slideWipe" subtype="fromRight"
                    dur="2s" mode="in"/>
</img>
```

Alternativ kann eine Überblendung im Header[1] global definiert werden und einem grafischen Objekt dann über die Attribute `transIn` und `transOut` zugewiesen werden.

Es werden die SMIL-Module BasicTransitions und InlineTransitions, nicht aber TransitionModifiers, unterstützt. Es ist zu beachten, dass viele Effekte wie bei SMIL optional sind und so eventuell nicht von jedem MPEG-4 Player abgespielt werden können.

2.3.9 Inhaltssteuerung

XMT-Ω unterstützt die aus SMIL bekannten Module BasicContentControl, CustomTestAttributes, PrefetchControl und SkipContentControl zur Steuerung des Inhalts.

Mittels eines `<switch>` Elements lassen sich so unter anderem statische oder dynamische Ausdrücke (beispielsweise Benutzereingaben oder Systemattribute wie Bildschirmgröße oder aktuelle CPU-Belastung des Endgerätes) auswerten, um dadurch den weiteren Verlauf einer Präsentation beeinflussen zu können.

`<prefetch>` erlaubt es Autoren, Objekte schon vor ihrem eigentlichen "In-Kraft-Treten" in einer Szene "manuell" zu laden und vorzubereiten, um diese beispielsweise bei nicht deterministischen Ereignissen sofort abspielen zu können. Dies ist gerade beim Streaming über langsame Transportverbindungen sehr nützlich.

2.3.10 Linking

Es wird das aus SMIL und HTML bekannte Konstrukt `<a>` unterstützt, um auf externe MPEG-4 Präsentationen verweisen zu können.

Wird in folgendem Beispiel auf den Kreis oder das Rechteck geklickt, so wird die aktuelle Präsentation beendet und die Datei `foo.mp4` abgespielt.

```
<a href="foo.mp4">
  <circle radius="25" .../>
  <rectangle size="100" .../>
</a>
```

2.4 XMT-A Format

Das XMT-A Format stellt eine direkte Abbildung von MPEG-4 Systems in XML-Syntax dar.

Die Szenenbeschreibung in Form von Knoten ist dabei weitgehend kompatibel mit eXtensible 3D (X3D), dem vom Web3D-Konsortium entwickelten Quasi-Nachfolger von VRML. Mit Hilfe der Szenenbeschreibung kann die räumliche und zeitliche Darstellung von Objekten beschrieben werden.

1 engl. für Kopf

Object Descriptors (ODs), ein weiterer wichtiger Bestandteil von MPEG-4 Systems, lassen sich ebenso in Form von Text darstellen. Sie spezifizieren ein Rahmenwerk zur Identifikation, Beschreibung und Assoziation von Elementary Streams (ESs). ESs können dabei entweder untereinander oder mit Objekten in einer Szenenbeschreibung assoziiert sein.

2.4.1 Dokumentstruktur

Ein XMT-A Dokument beginnt mit einem optionalen `<Header>` Element, gefolgt von einem `<Body>` Element, in welchem die eigentlichen Daten festgelegt werden. Ersteres kann unter anderem Metainformationen in Form von `<meta>` Elementen sowie `<InitialObjectDescriptor>` Elemente, das sind MPEG-4 spezifische Daten zur Definition von Profil und Level, beinhalten.

Die Dokumentstruktur lehnt sich stark an die von X3D an, um so die Interoperabilität zwischen XMT-A und X3D zu unterstützen.

2.4.2 Zeitsteuerung

XMT-A nutzt das bereits aus SMIL und XMT-Ω bekannte `<par>` Element zur Steuerung der zeitlichen Abfolge. Dieses kann gruppiert werden, um mehrere Kommandos, Ereignisse oder Nachrichten gleichzeitig zu verarbeiten.

Die zeitliche Repräsentation von Objekten hängt dabei alleinig von der relativen Position innerhalb einer Schachtelung von `<par>` Elementen ab, das heißt es werden bis auf das äußerte `<par>` Element, welches implizit `<par begin="0s">` ist, keine absoluten Zeiten verwendet. Das `beginn` Attribut steht dabei im Gegensatz zu XMT-Ω nur in den `<par>` Elementen. Folgendes Beispiel verdeutlicht dies:

```
<Body>
  ...
  <par begin="5s">
    ...
    <par begin="4s">    <!-- beginnt bei 9 Sekunden -->
    ...
    </par>
  </par>
</Body>
```

Ein `<par>` Element kann folgende Elemente beinhalten:

- `<par>`
- BIFS-Kommandos
- BIFS-Animationen
- OD-Kommandos
- IPMP-Nachrichten

- OCI-Ereignisse
- MPEG-J Stream Header

2.4.3 Szenenbeschreibung

Mit XMT-A ist es möglich, BIFS[1] in Form von Text darzustellen. Dies beinhaltet die Knoten einer Szenenbeschreibung, Verknüpfungen, mit welchen die einzelnen Knoten verbunden werden können und BIFS-Kommandos zum Anzeigen und Ändern einer Szene.

2.4.3.1 Knoten

Es gibt für jeden Knoten innerhalb BIFS ein Pendant in XMT-A, das heißt einen XML-Knoten mit dem Namen des jeweiligen BIFS-Knotens. Für die Attribute dieser Knoten gilt das gleiche.

Optional hinzugefügte Elemente und Attribute können Anweisungen und Informationen für eine spätere binäre Kodierung beinhalten. Der Inhalt bleibt dabei kompatibel.

XMT-A folgt bei der Umsetzung der Knoten den gleichen Regeln wie X3D für die XML-Darstellung von VRML. XMT-A ist zu X3D kompatibel, da MPEG-4 BIFS selbst wiederum von VRML abgeleitet wurde.

2.4.3.2 Verknüpfungen

Das Element `<ROUTE>` stellt das XMT-A Pendant zu MPEG-4 Systems ROUTE dar. Mit diesem Element können Felder miteinander verbunden werden.

Mit dem optionalen Attribut `ID` kann einer Verknüpfung ein Name zugewiesen werden. Dadurch kann eine Verknüpfung später referenziert werden, um so zum Beispiel eine Verknüpfung durch eine andere Verknüpfung zu löschen oder zu ersetzen.

In folgendem Beispiel wird das Feld `emissiveColor` eines Knoten (vom Typ Material2D) mit einem anderen Knoten verknüpft. Wird die Farbe des Knotens `MyRectangleMaterial` geändert, so ändert sich die Farbe des verknüpften Knotens `MyCircleMaterial` mit.

```
<ROUTE fromNode="MyRectangleMaterial" fromField="emissiveColor"
       toNode="MyCircleMaterial" toField="emissiveColor"/>
```

2.4.3.3 BIFS-Kommandos

XMT-A bietet folgende Basisbefehle, um Knoten, (indizierte) Werte und Verknüpfungen einfügen, löschen oder ändern zu können:

- `<Insert>`

1 binäre Szenenbeschreibungssprache, festgelegt in MPEG-4 Systems

- `<Delete>`
- `<Replace>`

Nachstehende Beispiele veranschaulichen die Benutzung dieser drei Befehle.

Im ersten Beispiel werden zwei Figuren und eine Verknüpfung in dem Knoten `OrderedGroupX` eingefügt. Das Rechteck wird dabei am Anfang und der Kreis am Ende der Liste der untergeordneten Elemente eingefügt. Der Kreis sowie das übergeordnete `<Shape>` Element besitzen eine DEF-ID, sodass eine spätere Referenzierung möglich ist.

```
<Insert atNode="OrderedGroupX" position"BEGIN END">
  <Shape>
    <geometry>
      <Rectangle size="120 80"/>
    </geometry>
  </Shape>
  <Shape DEF="MyCircle">
    <geometry>
      <Circle radius="25" DEF="Circle25"/>
    </geometry>
  </Shape>
  <ROUTE fromNode="Mat2D-A" fromField="filled"
         toNode="Mat2D-B" toField="filled"/>
</Insert>
```

Nachfolgend ist zu sehen, wie der Kreis wieder gelöscht werden könnte.

```
<Delete atNode="MyCircle">
```

Das dritte Beispiel ändert zuerst den Radius des Kreises auf einen Wert von 40 Einheiten und sodann dessen Geometrie zu einem Rechteck mit den Maßen 25 und 25.

```
<Replace atNode="Circle25" atField="radius" value="40"/>
<Replace atNode="MyCircle" atField="geometry">
  <Rectangle size="25 25"/>
</Replace>
```

2.4.4 Object Descriptor (OD) Repräsentation

Außer der Möglichkeit, die Szenenbeschreibung in Form von Text (XML) darzustellen, dient XMT-A auch dazu, MPEG-4 Object Descriptors in XML-Syntax beschreiben zu können.

Die textbasierte Repräsentation umfasst dabei die Deskriptoren selbst, Kommandos zum Ändern dieser und eine Darstellungsmöglichkeit für ESs.

2.4.4.1 Deskriptoren

Object Descriptors identifizieren und beschreiben elementare Datenströme und können diese mit einem Objekt in einer Szene verknüpfen. Sie stellen somit einen Zugang zu MPEG-4 Daten her.

Ein Deskriptor ist selbst ein elementarer Stream und bietet Informationen wie Typ (zum Beispiel Audio oder MPEG-J) oder Profil eines Streams, Daten zur Konfiguration des Decoders, QoS[1]-Anforderungen oder Informationen für das Rechtemanagement[2].

Nachstehend ist die Grundform aller Deskriptoren in XMT-A Syntax abgebildet:

```
<InitialObjectDescriptor
  ObjectDescriptorID=""
  includeInlineProfileLevelFlag="">
  <URL URLstring=""/>
  <Profiles
    ODProfileLevelIndication=""
    sceneProfileLevelIndication=""
    audioProfileLevelIndication=""
    visualProfileLevelIndication=""
    graphicsProfileLevelIndication="">
  <Descr>
    <esDescr>       <!-- 1 bis 255 -->  </esDescr>
    <ociDescr>      <!-- 0 bis 255 -->  </ociDescr>
    <ipmpDescrPtr>  <!-- 0 bis 255 -->  </ipmpDescrPtr>
  </Descr>
  <extDescr> ... </extDescr>
</InitialObjectDescriptor>
```

2.4.4.2 Kommandos

Mit XMT-A können Deskriptoren nicht nur definiert, sondern auch manipuliert werden. XMT-A beinhaltet dazu Descriptor-Kommandos, um Deskriptoren wie ObjectDescriptor, ES_Descriptor oder IPMP_Descriptor ändern oder löschen zu können.

2.4.4.3 Elementary Streams

Um auf einen ES verweisen zu können, bietet XMT-A das Descriptor-Element `<StreamSource>`. Damit können auch externe Datenströme oder auch solche wie natürliche Video- oder Audiomedien, welche nicht wie BIFS, OD, OCI, IPMP oder MPEG-J (nur Header) in Form von Text dargestellt werden können, mittels eines Verweises mit einbezogen werden.

1 Quality of Service
2 wird in MPEG-4 allgemein als IPMP (Intellectual Property Management and Protection) bezeichnet

2.5 Beispiel: XMT-Ω zu XMT-A

Folgendes Beispiel demonstriert, wie eine Szenenbeschreibung aufbauend auf das XMT-Ω Format in das XMT-A Format umgesetzt werden könnte.

Im dem Beispiel wird ein ausgefülltes Rechteck mit den Maßen 50 und 50 angezeigt. Wird auf dieses geklickt, so wird dessen Farbe in einem Zeitraum von sechs Sekunden über drei verschiedene Farben linear hinweginterpoliert.

Zunächst die Beschreibung im XMT-Ω Format:

```
<rectangle id="Square" size="50 50">
  <transformation visibility="true" translation="40 75"/>
  <material color="#ee0000" filled="true">
    <animateColor attributeName="color" dur="6s"
                  begin="Square.click"
                  values="#ee0000; #ffcc45; #ffffff"
                  keyTimes="0; 0.3; 1" calcMode="linear"/>
  </material>
</rectangle>
```

Nun die gleiche Szene, nur im XMT-A Format:

```
<Replace>
  <Scene>
    <OrderedGroup>
      <children>
        <Switch whichchoice="0">
          <choice>
            <Transform2D translation="40 75">
              <children>
                <Shape>
                  <appearance>
                    <Appearance>
                      <material>
                        <Material2D DEF="SquareMat"
                         emissiveColor="0.93 0.0 0.0"
                         filled="TRUE"/>
                      </material>
                    </Appearance>
                  </appearance>
                  <geometry>
                    <Rectangle size="50 50"/>
                  </geometry>
                </Shape>
                <TouchSensor DEF="Touch"/>
                <TimeSensor DEF="Timer" cycleInterval="6"/>
                <ColorInterpolator DEF="Coloring"
                 key="0.0 0.3 1.0" keyValue="0.93 0.0 0.0,
                 1.0 0.93 0.27, 1.0 1.0 1.0"/>
```

```
            </children>
          </Transform2D>
        </choice>
      </Switch>
    </children>
  </OrderedGroup>
  <ROUTE fromNode="Touch" fromField="touchTime"
         toNode="Timer" toField="startTime"/>
  <ROUTE fromNode="Timer" fromField="fraction_changed"
         toNode="Coloring" toField="set_fraction/>
  <ROUTE fromNode="Coloring" fromField="value_Changed"
         toNode="SquareMat" toField="emissiveColor"/>
 </Scene>
</Replace>
```

Es wird angemerkt, dass vorangegangenes Beispiel nur eines von mehreren Möglichkeiten darstellt, die "Hochsprache" XMT-Ω nach XMT-A umzusetzen.

2.6 Interoperabilität zu bestehenden Formaten

Diese Kapitel erklärt Zusammenhänge von bestehenden, bereits etablierten Formaten mit XMT. Des weiteren wird versucht aufzuzeigen, in wie fern sich in diesen Formaten bestehende Inhalte wiederverwenden beziehungsweise in das XMT-Format konvertieren lassen.

2.6.1 SMIL

Die Synchronized Multimedia Integration Language (SMIL) dient zur Beschreibung von interaktiven 2D Multimedia-Präsentationen. Sie basiert auf XML und wurde vom World Wide Web Consortium (W3C) standardisiert. Teile von SMIL 2.0 werden auch in XHTML[1] und SVG zur Zeitsteuerung eingesetzt.

Im Gegensatz zu SMIL, welches speziell als Abspielformat entworfen worden ist, stellt XMT vor allem eine Möglichkeit dar, MPEG-4 Inhalte generell textbasiert anstatt binär darzustellen.

Da XMT-Ω stark an SMIL angelehnt ist, sind viele Konstrukte und Module[2], sofern sie innerhalb des Querschnitts beider Standards liegen, austauschbar.

Folgende Tabelle bietet einen generellen Überblick über die Funktionalitäten in SMIL und deren Wiederverwendung in XMT-Ω.

1 eXtensible HyperText Markup Language – XML-Variante von HTML
2 SMIL wurde in funktionelle Bereiche unterteilt, die so genannten Module

SMIL-Module	Unterstützung	XMT-Ω Module
Animation	ja	XMT-Ω unterstützt die SMIL Animation Module, welche das dynamische Ändern von Attributen definieren. Im Gegensatz zu SMIL, wo es ein Hauptattribut mit dazugehörigem Wert gibt, wird nur ein Attributwert unterstützt.
Content Control	ja	Die Auswahl und Selektion von Inhalten anhand von Ausdrücken und Testattributen (zum Beispiel die Auswahl einer bestimmten Sprache durch einen Übergabeparameter bei der Kompilierung oder dynamisch durch eine Eingabeaufforderung) wird wie in SMIL übernommen und unterstützt.
Layout	nein	XMT-Ω definiert ein eigenes Layoutmodul, welches an die hierarchischen Strukturen und räumlichen Anordnungsmöglichkeiten in MPEG-4 angepasst ist.
Linking	nein	XMT-Ω bietet die Möglichkeit, Verknüpfungen mittels BIFS Anchor Knoten darzustellen.
Media	ja	Über SMIL hinaus werden neue, zusätzliche MPEG-4 spezifische Elemente definiert.
Metainformation	ja	XMT-Ω bietet zusätzlich die Einbettung von MPEG-7 Metainformationen.
Structure	nein	XMT-Ω definiert ein kompatibles Strukturmodul, welches nötige Transformationen für einen Austausch mit SMIL auf ein Minimum reduziert.
Timing and Synchronisation	beinahe vollständig	SMIL bietet in diesem Bereich umfassende Möglichkeiten. Die meisten dieser Möglichkeiten werden von XMT-Ω unterstützt.
Time Manipulations	ja	XMT-Ω definiert zusätzlich ein neues und flexibles Modul zur Repräsentation von MPEG-4 FlexTime.

SMIL-Module	Unterstützung	XMT-Ω Module
Transitions	teilweise	Es werden nur einige der in SMIL (und SMPTE) definierten Übergänge unterstützt.

Tabelle 4: Beziehungen zwischen SMIL und XMT-Ω Modulen

[SMIL]

2.6.2 SVG

SVG (Scalable Vector Graphics) ist eine auf XML-Syntax basierende Sprache zur Beschreibung von 2D Grafiken und ist vom W3C verabschiedet worden. Dabei werden drei Arten von Grafiken unterschieden:

- Vektorgrafiken
- Bilder
- Text

Grafische Objekte können gruppiert, transformiert und mit Stilen versehen werden. Weiterhin werden unter anderem Clipping, Alpha-Masken (Transparenz) und Filter unterstützt. [SVG]

2.6.3 VRML

Die Virtual Reality Modeling Language (VRML) ist eine Sprache zur Beschreibung von 3D Objekten und Welten und als ISO/IEC 14772 international verabschiedet worden. Es wurde als universelles Austauschformat für 3D Modelle und Multimedia-Inhalte entworfen.

BIFS, die Szenenbeschreibung von MPEG-4, basiert auf VRML, wurde aber um einige Funktionalitäten wie 2D, einem Update und Animations Protokoll, Zeitsteuerung, Kompression oder verbesserten Audioeigenschaften erweitert. [VRML]

2.6.4 X3D

eXtensible 3D (X3D) ist der designierte Nachfolger von VRML (Virtual Reality Modeling Language), dem "originalen" ISO-Standard für Web 3D Grafik und Multimedia und erweitert und verbessert diesen in einigen Punkten, zum Beispiel durch die Verwendung der XML-Syntax.

XMT-A stellt in dieser Hinsicht das MPEG-4 Pendant zu X3D dar. Folgende Tabelle vergleicht die Darstellung beider Formate miteinander und versucht die minimalen Unterschiede bei einer Konversation von einem in das andere Format aufzuzeigen.

XMT-A	X3D
<Header> <meta> </meta> <InitialObjectDescriptor .../> </Header> <Body> <Replace> <Scene> <!-- Szeneninhalt --> </Scene> </Replace> </Body>	<Header> <meta> </meta> </Header> <Scene> <!-- Szeneninhalt --> </Scene>

Tabelle 5: Vergleich zwischen XMT-A und X3D

[X3D]

2.6.5 MPEG-7

Weder MPEG-7 noch MPEG-21[1] spezifizieren (wie MPEG-4) Technologien für die Kompression von audio-visuellen Informationen.

Bei MPEG-7 geht es um die Beschreibung und Klassifizierung von Medien-Inhalten mittels Metadaten. Diese Metadaten können manuell oder durch automatisierte Extraktion festgelegt oder erhoben werden.

XMT unterstützt dabei die Einbindung solcher MPEG-7 Metadaten. Siehe nachfolgendes Beispiel:

```
<img id="Markus_GoldenGate.jpg" dur=60s>
  <StructuredAnnotation>
    <Who>Markus Brenner</Who>
    <When>Spring 2003</When>
    <TextAnnotation xml:lang='en-us'>
      Markus Brenner in front of the Golden Gate Bridge.
    </TextAnnotation>
  </StructuredAnnotation>
</img>
```

Obiger Auszug wurde zwecks einer besseren Lesbarkeit vereinfacht, indem die Namespaces entfernt wurden. [N5525]

1 Mit MPEG-21 wird versucht, ein komplettes Rahmenwerk für das Management und den Gebrauch von digitalen Daten, festzulegen.

3 Praktische Anwendung

3.1 Erstellung eines RichMedia-Prototyps

In folgendem Abschnitt soll ein Prototyp einer RichMedia-Anwendung auf Basis von XMT-Ω und MPEG-4 erstellt werden. Als Vorlage dazu dient eine bereits vom Fraunhofer-Institut (IAO) für die MAN AG erstellte RichMedia-Anleitung zur Endabnahme von LKWs. Diese Anwendung ist auf den Real-Player ausgerichtet und basiert auf einer Kombination aus dem RealMedia-Format für Audio und Video, SMIL als Szenenbeschreibung und Bildern für den eigentlichen Inhalt.

3.1.1 XMT-Ω Dokument

Die folgenden Kapitel zeigen die wichtigsten Schritte zur Erstellung der zugrunde liegenden XMT-Ω Beschreibung in Auszügen auf. Das komplette Beispiel ist dem Anhang zu entnehmen.

3.1.1.1 Grundaufbau

Der Grundaufbau einer jeden XMT-Ω Beschreibung besteht aus einer XML-Datei – in diesem Fall zum Beispiel MAN.xml – mit folgendem Inhalt:

```
<?xml version="1.0" encoding="UTF-8"?>
<XMT-O xmlns="urn:mpeg:mpeg4:xmto:schema:2002"
       xsi:schemaLocation="urn:mpeg:mpeg4:xmto:schema:2002
       schemas/xmt-o/xmt-o.xsd"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xlink="http://www.w3.org/1999/xlink">
  <head> ... </head>
  <body> ... </body>
</XMT-O>
```

Erste Zeile definiert, dass der Inhalt dieser Datei dem XML-Syntax entspricht. `encoding` legt dabei die Zeichenkodierung[1] der Datei fest.

Die Basiselemente sind `<XMT-O>` und die darin enthaltenen `<head>` und `<body>` Elemente. Die Attribute innerhalb von `<XMT-O>` definieren unter anderem den Namespace für das Schema, welches zur automatisierten Validierung des Dokuments herangezogen werden kann.

Innerhalb des `<head>` Elements werden für die gesamte Beschreibung allgemein gültige Definitionen[2] festgelegt.

1 damit Sonderzeichen wie © auch richtig kodiert werden, muss das Dokument im entsprechenden Kodierformat abgespeichert werden

2 Neben Metainformation und Layout können unter anderem auch Konstanten oder Makros – vergleichbar mit Templates – definiert werden.

```
<head>
  <meta name="title" content="XMT/MPEG-4 Demo: MAN LKW Endabnahme"/>
  <meta name="author" content="Markus Brenner"/>
  <meta name="copyright" content="Copyright© 2004 Markus Brenner,
        FhG IAO Stuttgart, HdM Stuttgart"/>
  <layout metrics="pixel" type="xmt/xmt-basic-layout">
    <topLayout backgroundColor="black" width="734" height="418"/>
  </layout>
</head>
```

Dies sind unter anderem Metainformationen wie beispielsweise Titel oder Autor sowie Angaben zum Layout einer Präsentation.

Vorangegangener Auszug definiert ein 734 Pixel x 418 Pixel großes Layout mit standardmäßig schwarzem Hintergrund. Farben können dabei außer mit ihrem Namen[1] auch in Form von RGB-Werten – beispielsweise `#0000FF`[2] oder `0 0 1`[3] für Blau – angeben werden.

Neben der metrischen Einheit `pixel` existiert außerdem die Einheit `meter`. Letztere ist eine relative Einheit und entspricht der halben Breite einer Präsentation – 0.1 entspricht beispielsweise 1/20 der Breite. Die Position und Größe eines Objektes würde somit mit der Größe einer Präsentation skalieren.

3.1.1.2 Layout und Regionen

Um später nicht für jedes Objekt die Position neu "berechnen" und festlegen zu müssen, ist es möglich, das Layout in Regionen einzuteilen. Die Position eines einer Region zugewiesenen Objektes verhält sich dann relativ zu dessen Position.

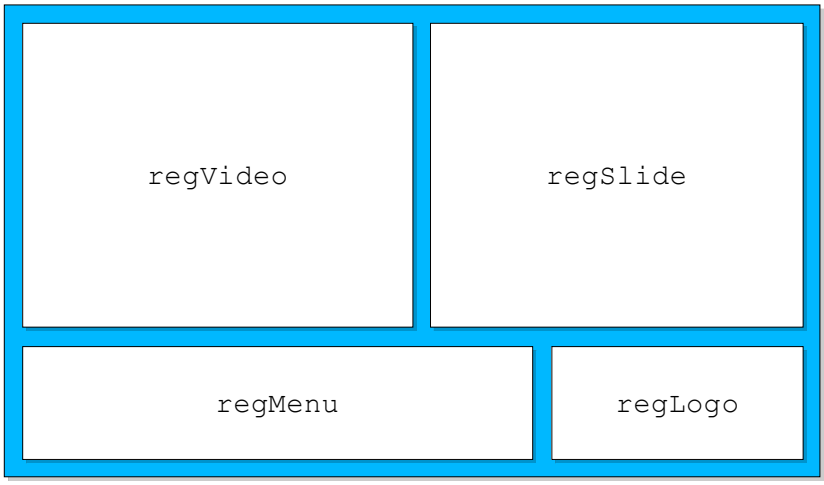

Abbildung 15: Layout

1 es sind nur für die gängigsten Farben Namen definiert
2 hexadezimal (00 bis FF je Farbkomponente)
3 dezimal (0 bis 1 je Farbkomponente)

```
<topLayout backgroundColor="black" width="734" height="418">
  <region id="regVideo" translation="-181 55" size="352 288"/>
  <region id="regSlide" translation="181 55" size="352 288"/>
  <region id="regMenu" translation="-107 -148" size="500 100"/>
  <region id="regLogo" translation="245 -148" size="204 100"/>
</topLayout>
```

Regionen werden innerhalb des `<topLayout>` Elements, welches selbst `<layout>` und `<head>` untergeordnet ist, festgelegt. Das `size` Attribut definiert dabei ein Ausschnittrechteck, das heißt alles was über dessen Grenzen geht, wird nicht dargestellt. Bei überlappenden Regionen ist es sinnvoll eine Ebenenposition in Form des Attributs `z-index` zu definieren.

Um nun später ein grafisches Objekt – beispielsweise ein Logo - innerhalb der dazugehörigen Region darzustellen, müsste dem jeweiligen Element nur einfach dessen Referenz hinzugefügt werden:

```
<img src="Media/Logo.png" dur="indefinite" region="regLogo"/>
```

Wird keine Region spezifiziert, so wird auf das `<topLayout>` Bezug genommen.

3.1.1.3 Zeitlicher Ablauf

Inhalte wie Bilder, Video, Text oder Primitiven selbst sind dem `<body>` Element untergeordnet. In welcher Reihenfolge solche Objekte zeitlich dargestellt werden, wird mittels `<par>`, `<seq>` oder `<excl>` gesteuert.

Elemente innerhalb `<par>` werden dabei gleichzeitig, innerhalb `<seq>` nacheinander abgespielt beziehungsweise angezeigt – nähere Details dazu finden sich in Kapitel 2.3.5.

In dem vorgestellten RichMedia-Prototyp sollen nun einige Folien nacheinander abgespielt werden. Das rechts oben befindliche Video, das darunter liegende Menü und das rechts unter den Folien liegende Logo sollen dabei ständig sichtbar sein. Weiterhin soll die Präsentation mit einem Intro eingeleitet und mit einem Outro beendet werden. Folgender Auszug zeigt, wie das Vorhaben realisiert werden könnte:

```
<body>
  <par>
    <!-- Hintergrundbild -->
    <seq>
      <!-- Intro -->
      <par>
        <!-- Menü -->
        <!-- Logo -->
        <!-- Video -->
        <seq>
          <!-- Folie 1 -->
```

```
          <!-- Folie 2 -->
          <!-- Folie N -->
        </seq>
      </par>
      <!-- Outro -->
    </seq>
  </par>
</body>
```

Wie zu erkennen ist, wird das Hintergrundbild die gesamte Zeit über angezeigt.

3.1.1.4 Hintergrundbild und Logo

Das Hintergrundbild und das Logo lassen sich mit dem `<img>` Element wie folgt realisieren:

```
<body>
  <par>
    <img id="imgBackground" src="Media/Background.jpg"
         dur="indefinite"/>
    ...
    <seq>
      ...
      <par>
        <img id="imgLogo" src="Media/Logo.png"
             dur="indefinite" region="regLogo"/>
        ...
      </par>
      ...
    </seq>
  </par>
</body>
```

Es ist sinnvoll allen Objekten einen eindeutigen Namen in Form des `<id>` Attributs zu geben. Objekte lassen sich dadurch später referenzieren.

`<dur>` ist ebenso wie `<id>` optional und spezifiziert die Dauer der Anzeige. Da diese aber bei allen grafischen Objekten, falls nicht von einem übergeordneten Element mit deterministischen Zeitangaben umschlossen, standardmäßig 0 ist, sollte immer explizit ein Wert angegeben werden.

Grafische Objekte werden, falls nicht weiter definiert, zentriert dargestellt, das heißt ihr Ursprung (= Zentrum) ist definiert durch die Mitte des Layouts beziehungsweise der ihr zugewiesenen Region. Um nun eine andere Position festzulegen, genügt es ein `<transformation>` Element dem Objekt hinzuzufügen. In nachfolgendem Beispiel wird das Bild um 200 Einheiten nach oben verschoben:

```
<img ...>
  <transformation translation="0 200"/>
</img>
```

3.1.1.5 Video

Videoobjekte sind im Gegensatz zu Bildern keine diskreten Elemente, das heißt die Abspieldauer der Elemente ist durch das Video bereits selbst definiert, ansonsten folgen sie den gleichen Prinzipien wie Bilder. Soll ein Video komplett abgespielt werden, so ist das Attribut `dur` auf `media` zu setzen.

```
<video id="video" src="Media/Video.mp4" dur="media"
       region="regVideo"/>
```

XMT-Ω verlangt im Gegensatz zu SMIL bereits zur Compile-Zeit eine klare Auftrennung der Medien. Deshalb müssen Audio und Video mittels den Elementen `<audio>` und `<video>` getrennt angegeben werden. Befinden sich in einer MP4-Datei mehrere Spuren, so kann in folgender Weise darauf Bezug genommen werden:

```
<video id="video" src="Media/Video.mp4#video" dur="media"
       region="regVideo"/>
<audio id="audio" src="Media/Video.mp4#audio" dur="media"/>
```

Über das Präfix `#` wird der jeweils erste auf das angegebene Format zutreffende Track ausgewählt.

3.1.1.6 Hervorhebung von Menü und Folien

Um den Inhalt in den Regionen `regSlide` und `regMenu` etwas vom Hintergrund hervorzuheben, wird jeweils ein blau gefülltes Rechteck, welches zudem transparent erscheinen soll, entsprechend den Maßen der jeweiligen Region gezeichnet.

```
<rectangle size="500 100" region="regMenu">
  <material color="blue" filled="true" transparency="0.9"/>
</rectangle>
...
<rectangle size="352 288" region="regSlide">
  <material color="blue" filled="true" transparency="0.9"/>
</rectangle>
```

Das `<material>` Element ist dabei nicht nur auf Primitiven beschränkt. Beispielsweise kann es ebenso auf Videos oder Bilder angewendet werden.

3.1.1.7 Inhalt von Menü und Folien

Der Inhalt des Menüs und der Folien soll aus Text bestehen. Folgendes verdeutlicht das an einem Menüpunkt mit dem Text "1. Demontage vorbereiten":

```
<string id="menu1"
        textLines="&amp;quot;1. Demontage vorbereiten&amp;quot;"
        dur="indefinite" region="regMenu">
  <transformation translation="-230 20"/>
```

```
  <material color="white" filled="true"/>
  <fontStyle justify="MIDDLE; MIDDLE" size="16"
             style="BOLD" family="&quot;SANS&quot;"/>
</string>
```

Der angezeigte Text wird im Attribut `textLines` spezifiziert. Der Text selbst muss dabei in zusätzlichen Anführungszeichen stehen. Wie bei XML-Syntax allgemein üblich, ist hier darauf zu achten, dass keine Sonderzeichen verwendet werden, das heißt die Anführungszeichen müssen mit der "Escape"-Sequenz `&quot;` kodiert werden. Sollen mehrere Zeilen Text ausgegeben werden, so müssen die Zeilen in gesonderten Anführungszeichen und von Semikolons getrennt stehen.

Die Farbe der Schrift wird durch das Element `<material>` festgelegt. Schriftattribute wie Größe, Ausrichtung oder Schriftart werden durch das Element `<fontStyle>` definiert.

Die Ausrichtung des Textes kann eine Kombination aus `FIRST`, `BEGIN`, `MIDDLE` und `END` sein. Folgende Abbildung verdeutlicht die Anwendung dieser Werte:

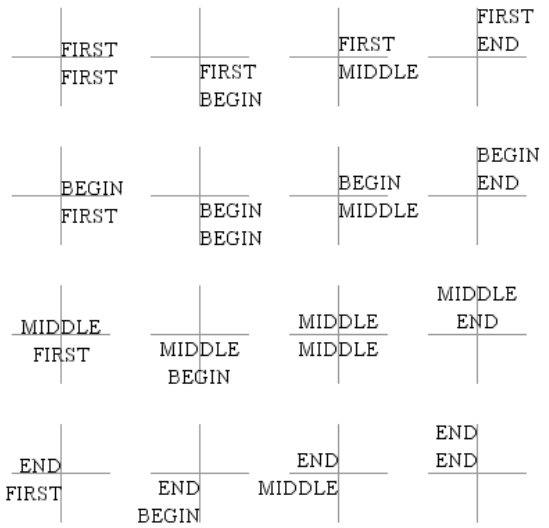

Abbildung 16: Schriftausrichtung

Das Attribut `family` – also die Schriftart – kann mehrere Schriftnamen beinhalten. Die erste Schrift, welche auf dem MPEG-4 Endgerät gefunden wird, wird zur Anzeige benutzt. Standardmäßig werden die Schriften `SANS`, `SERIF` und `TYPEWRITER` unterstützt.

Um nun bestimmte, immer wiederverwendete Definitionen wie die Schriftart nicht mehrmals angeben zu müssen, können diese Definitionen global im `<head>` Element festgelegt werden. Gewünschte Änderungen müssen dann nur noch an einer Stelle vorgenommen werden. Siehe folgendes Beispiel:

```
<defs>
  <fontStyle id="defMenuFont" justify="MIDDLE; MIDDLE" size="16"
             style="BOLD"family="&amp;quot;SANS&amp;quot;"/>
  <material id="defMenuFontMat" color="white" filled="true"/>
</defs>
```

Das ursprüngliche Codekonstrukt kann nun wie folgt umgeschrieben werden, um die eben vorgestellten globalen Definitionen zu referenzieren:

```
<string id="menu1" ...>
  <transformation translation="-230 20"/>
  <use xlink:href="#defMenuFont"/>
  <use xlink:href="#defMenuFontMat"/>
</string>
```

3.1.1.8 Gruppierung zu Folien

Um den Inhalt einer Folie in der Region `regSlide` thematisch, aber auch vor allem aus Sicht der Programmierung (beispielsweise gemeinsames Ein- und Ausblenden aller Objekte einer Folie), zusammenfassen zu können, werden diese Objekte in das Element `<group>` eingeschlossen.

Das `<group>` Element fungiert als Container, der mehrere Elemente (auch `<group>` Elemente) beinhalten kann und auf den auch die von Medienobjekten bekannten Operationen und Definitionen (zum Beispiel `region`, `begin`, `size` oder `translation` – Letzteres allerdings hier als Attribut) vorgenommen werden können.

Bezüglich des Layouts verhält sich ein `<group>` Element wie eine Region. Nachfolgendes Beispiel demonstriert die Anwendung des `<group>` Elements:

```
<group id="slide1" region="regSlide">
  <string id="item1_1" ...>
    <transformation translation="-166 134"/>
    <use xlink:href="#defSlideFont"/>
    <use xlink:href="#defSlideFontMat"/>
  </string>
  <string id="item1_2" ...>
    <transformation translation="-166 104"/>
    <use xlink:href="#defSlideFont"/>
    <use xlink:href="#defSlideFontMat"/>
  </string>
  ...
</group>
```

3.1.1.9 Aktuelle Position hervorheben

Um die aktuelle Videoposition auch innerhalb des Menüs und der jeweiligen Folien selbst widerzuspiegeln, werden die entsprechenden Punkte im Menü und auf den Folien hervorgehoben.

Hinsichtlich der Gestaltung bieten sich dafür mehrere Möglichkeiten an: zum Beispiel ändern der Schriftfarbe oder Markierung durch einen Pfeil. Eine weitere ist, den Text in seiner Sichtbarkeit, genauer gesagt seiner Transparenz, zu variieren.

In folgendem Beispiel wird der Text einer Folie standardmäßig leicht transparent und nur zu einer bestimmten Zeit vollständig opaque dargestellt.

```
<string id="item1_4" textLines="&quot;Die Räder der
        Arbeitsbühne arretieren&quot;" dur="indefinite">
  <transformation translation="-166 44"/>
  <use xlink:href="#defSlideFont"/>
  <material color="white" filled="true"    transparency="0.4">
    <set attributeName="transparency" to="0" begin="69s"/>
    <set attributeName="transparency" to="0.4" begin="78s"/>
  </material>
</string>
```

Um Zeitwerte nicht zweimal setzen zu müssen, wird bei Menüpunkten Bezug auf den Anfang beziehungsweise das Ende der entsprechend zugeordneten Folien genommen. Im Beispiel der Menüpunkt 2:

```
<string id="menu2" ...>
  ...
  <material color="white" filled="true" transparency="0.4">
    <set attributeName="transparency" to="0" begin="slide2.begin"/>
    <set attributeName="transparency" to="0.4" begin="slide2.end"/>
  </material>
</string>
```

3.1.1.10 Intro, Outro und Blende

Die Präsentation soll mit einem Intro eingeleitet und mit einem Outro beendet werden.

Es soll jeweils ein kurzer Text – hinterlegt von einem leicht transparenten Rechteck – für kurze Zeit ein- und wieder ausgeblendet werden. Dabei wird der Text zusammen mit dem Rechteck von oben "hereingeschoben" und nach unten hin wieder "hinausgeschoben".

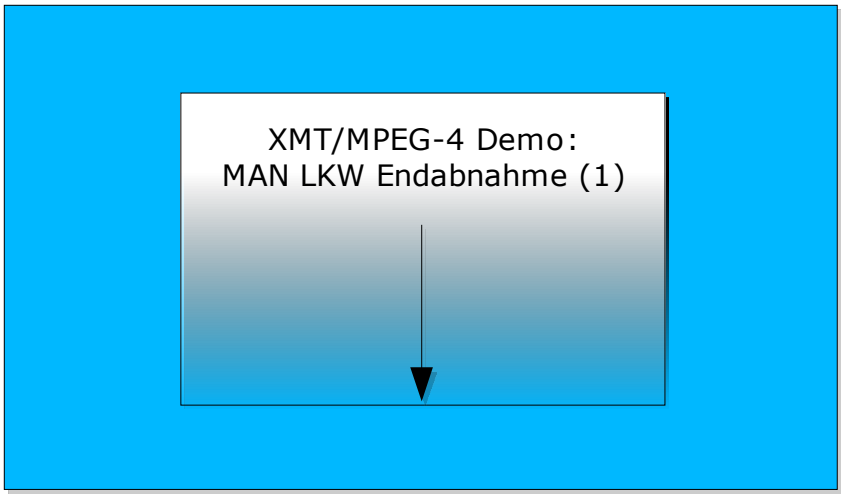

Abbildung 17: Intro-Einblende

Nachstehender Auszug demonstriert die Umsetzung am Beispiel des Intros.

```
<group id="intro" dur="6s"
       transIn="transSlideWipeTop"
       transOut="transSlideWipeTop">
  <rectangle size="367 209">
    <material color="blue" filled="true" transparency="0.9"/>
  </rectangle>
  <string textLines="&quot;XMT/MPEG-4 Demo:&quot;;
          &quot;MAN LKW Endabnahme (1)&quot;">
    <use xlink:href="#defIntroOutroFont"/>
    <use xlink:href="#defIntroOutroFontMat"/>
  </string>
</group>
```

Beide Objekte, also der Text und das Rechteck, werden mittels einem `<group>` Element gruppiert. Diesem werden dann über die Attribute `transIn` und `transOut` jeweils eine Transistion für die Ein- und Ausblende zugewiesen.

Die Transistion selbst wurde unter dem `<head>` Element definiert:

```
<transition id="transSlideWipeTop" type="slideWipe"
            subtype="fromTop" dur="2s"/>
```

Wie auch bei SMIL, werden bei MPEG-4 nur einige Blendeneffekte standardmäßig von jedem Endgerät unterstützt. Werden optionale, nicht fest vorgeschriebene Effekte benutzt, so wird eine Transistion bei fehlender Unterstützung seitens des Endgerätes ignoriert und das jeweilige Objekt ohne Transistion dargestellt.

Die vier Hauptobjekte der Präsentation – also das Video, das Logo sowie die beiden zur Hervorhebung des Menüs und der Folien verwendeten Rechtecke – werden ebenfalls mit dem gleichen Wischeffekt von links und rechts hinein- beziehungsweise nach dem Ende der Präsentation hinausgeschoben.

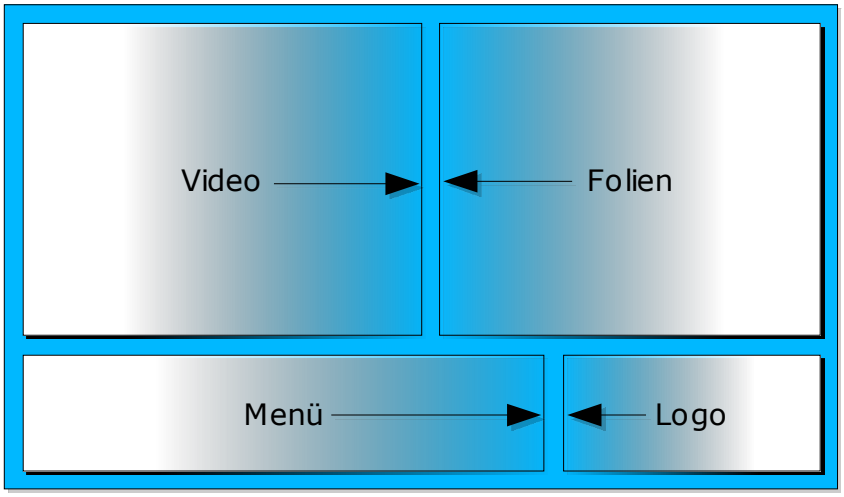

Abbildung 18: Einblende nach Intro

Die jeweiligen Definitionen sehen wie folgt aus:

```
<transition id="transSlideWipeRight" type="slideWipe"
            subtype="fromRight" dur="2s"/>
<transition id="transSlideWipeLeft" type="slideWipe"
            subtype="fromLeft" dur="2s"/>
```

Die beiden Attribute `transIn` und `transOut` wurden entsprechend dem ersten Beispiel den vier zuvor erwähnten Objekten hinzugefügt. Dabei ist das Menü ähnlich dem Intro zu einer Gruppe zusammengefasst.

3.1.1.11 Interaktion und Ablaufsteuerung

Zuletzt soll dem Benutzer die Möglichkeit gegeben werden, in die bisher nur selbstständig ablaufende Präsentation einwirken zu können und somit der multimedialen Anwendung Interaktivität verleiht werden.

Über das Menü, also durch einen Mausklick auf einen Menüpunkt, soll zur entsprechend thematischen Stelle gesprungen werden können. Sowohl das Video als auch die Folien sollen an anderer Stelle fortgesetzt werden.

Dieses Vorhaben gestaltet sich in vorliegendem Fall aus zwei verschiedenen Gründen als unerwartet schwierig und ist in der jetzigen Situation nicht (zufrieden stellend) lösbar:

- Objekt-Linking in der Form `<a href="#slide2">` wird im Gegensatz zu SMIL nicht unterstützt, das heißt es ist nicht möglich, von einem Menüpunkt auf eine Folie zu verweisen. Stattdessen muss die Ablaufsteuerung der Folien in diesen selbst mittels dem Attribut `begin` festgelegt werden.

  Um eine Folie nun zu einer bestimmten Zeit (bei automatischem Ablauf der Präsentation ohne direkten Eingriff des Benutzers über das Menü) oder aber

bei direkter Anwahl des entsprechend zugeordneten Menüpunktes anzuzeigen, müsste im Prinzip nur Folgendes – im Beispiel Folie 2 – definiert werden:

```
<par>
  <!-- slide 1 -->
  <group id="slide2" dur="101s" region="regSlide"
         begin="slide1.end; menu2.click">
  </group>
  <!-- slide 3 und weitere -->
</par>
```

Die Definition von mehreren Werten für die Attribute `begin` und `end` – so genanntes MultiArcTiming – wird aber (noch) nicht von allen XMT-Ω Compilern[1] voll unterstützt.

Die Folien innerhalb eines `<seq>` Elements statt wie in vorhergehendem Beispiel innerhalb eines `<par>` Elements zu definieren und das Attribut `begin="slide1.end; menu2.click"` auf den einzelnen Wert `menu2.click` zu reduzieren[2] würde nicht funktionieren, da die Folien zwar automatisch nacheinander abgespielt werden würden, aber jeweils erst nach Klick auf den entsprechenden Menüpunkt.

- Da nicht auf andere Objekte verweisen und somit im zeitlichen Ablauf "gesprungen" werden kann, müssten die Abspielpositionen von parallel ablaufenden Audio- oder Videoelementen bei einer Menüauswahl "manuell" geändert werden. Die zugrunde liegende Spezifikation MPEG-4 Part 1: Systems schreibt aber keine solche Möglichkeit vor.

Laut Jeff Boston [JEFF], einem Mitarbeiter des IBM T.J. Watson Research Center, ist momentan die einzige Möglichkeit diesen "Umstand" in XMT-Ω zu umgehen, die Präsentation inklusive Video in verschiedene MP4-Dateien entsprechend aufzuteilen. Würde nun auf einen Menüpunkt, welcher mittels einem `<a>` Element auf die nächste Datei verweist, geklickt werden, so würde die aktuelle Präsentation beendet und im selben Abspielfenster mit dem nächsten Teil beziehungsweise mit der nächsten Datei fortgefahren werden. Prinzipiell funktioniert dies zwar, es entstehen jedoch "unschöne" Pausen beim Wechsel und der Vorteil, eine komplette Präsentation inklusive aller Medien wie Audio, Video oder Bilder in einer einzigen Datei halten zu können, ist damit auch nicht mehr zu verwirklichen.

1 so zum Beispiel der Compiler aus dem IBM Toolkit for MPEG-4, das in dieser Arbeit eingesetzt wurde und bis dato den einzig erhältlichen XMT-Ω Compiler enthält

2 auf das Ereignis `slide1.end` müsste durch den nun sequenziellen Ablauf nicht mehr reagiert werden

3.1.2 Audio und Video Komprimierung

Die in einem XMT-Ω referenzierten audio-visuellen Medien werden im nächsten Schritt, der Kompilierung, zu einer Datei zusammengefasst.

Grundlage dafür ist, dass alle diese Medien bereits in einem von MPEG-4 direkt unterstützten Format vorliegen. Deshalb muss das unkomprimierte Audio- und Videoeingangsmaterial komprimiert werden. Folgender Ablauf stellt einen von vielen[1] Möglichkeiten dar.

Das Video wurde mittels VirtualDub und XviD komprimiert. Es wurde im Rahmen dieser Arbeit darauf geachtet, das Simple Profile[2] einzuhalten.

Das Audiomaterial wurde mit Nero Digital nach AAC komprimiert. Da viele Abspielgeräte HE AAC noch nicht unterstützen, wurde darauf verzichtet.

Die jeweils erhaltene Audio- und Videodatei wurde mittels dem im IBM Toolkit for MPEG-4 [IBMM4] enthaltenen MPEG-4 Audio Video Content Generator zu einer MP4-Datei "umgewandelt" und zusammengefasst. Dieser Schritt ist nicht für AAC notwendig, aber für die von VirtualDub erhaltene AVI-Datei, da dieses Dateiformat von MPEG-4 nicht unterstützt wird.

Nachfolgend ist ein Überblick über verschiedene Parameter des Eingangs- und Ausgangsmaterials zu sehen:

	Eingang	**Ausgang**
Audio		
Länge	5:41 min	5:41 min
Format	PCM 16 Bit	AAC LC (MPEG-4 Main @ L2)[3]
Bitrate	1.410 kBit/s	45 kBit/s
Kanäle	2	1
Samplerate	44.100 Hz	24.000 Hz
Datengröße	58.756 KB	1.921 KB
Kompression	30	1
Video		
Länge	5:42 min	5:42 min
Format	IYUV (4:2:0)	MPEG-4 Simple @ L3[4]
Bitrate	30.348 kBit/s	185 kBit/s

1 Kapitel 3.2.1 bietet eine Auflistung verschiedener Encoder
2 das Simple Profile legt bestimmte Kriterien wie maximale Videogröße, Datenrate oder Framerate fest, sodass Inhalte speziell noch auf kleinen, leistungsschwachen Endgeräten abgespielt werden zu können
3 VBR (Variable Bit Rate)
4 komprimiert in einem Durchgang (Single-Pass) bei einem Quantizer von 7

	Eingang	Ausgang
Framerate	25 fps	15 fps
Größe	352 x 288	352 x 288
Datengröße	824.371 KB	7.850 KB
Kompression	105	1
Gesamt		
Datengröße	883.127 KB	9.771 KB
Kompression	90	1

Tabelle 6: Überblick der Audio- und Videokomprimierung

3.1.3 Kompilierung

Im letzten Schritt zur Erstellung des RichMedia-Prototyps wird die XMT-Ω Szenenbeschreibung kompiliert und die darin referenzierten Medien wie Bilder und Video zu einer MP4-Datei zusammengefasst.

Für die Kompilierung von XMT-Ω bietet sich bisher nur eine Software, nämlich der MPEG-4 XMT Batch Converter aus dem IBM Toolkit for MPEG-4, an. Dies liegt daran, dass XMT erst vor kurzer Zeit verabschiedet wurde und IBM – wie auch schon bei SMIL - bei der Entwicklung und Standardisierung von XMT entscheidend beigetragen hat.

Das Toolkit, welches auch andere Programme wie einen MPEG-4 Player oder ein Programm zum Generieren von MP4-Dateien beinhaltet, ist frei verfügbar. Möchte man die Software außer für Forschung, Lehre und privaten Gebrauch auch kommerziell einsetzen, muss diese von IBM lizensiert werden.

Alle in dem Toolkit enthaltenen Programme sind komplett in Java gehalten und so auf jedem Betriebssystem[1] ohne Installation oder desgleichen sofort einsetzbar. Des weiteren liegt auch ein Java-SDK[2] bei, sodass die verwendete Technologie leicht in eigene Programme eingebunden und implementiert werden kann.

1 installierte Java Virtual Machine vorausgesetzt
2 Software Development Kit

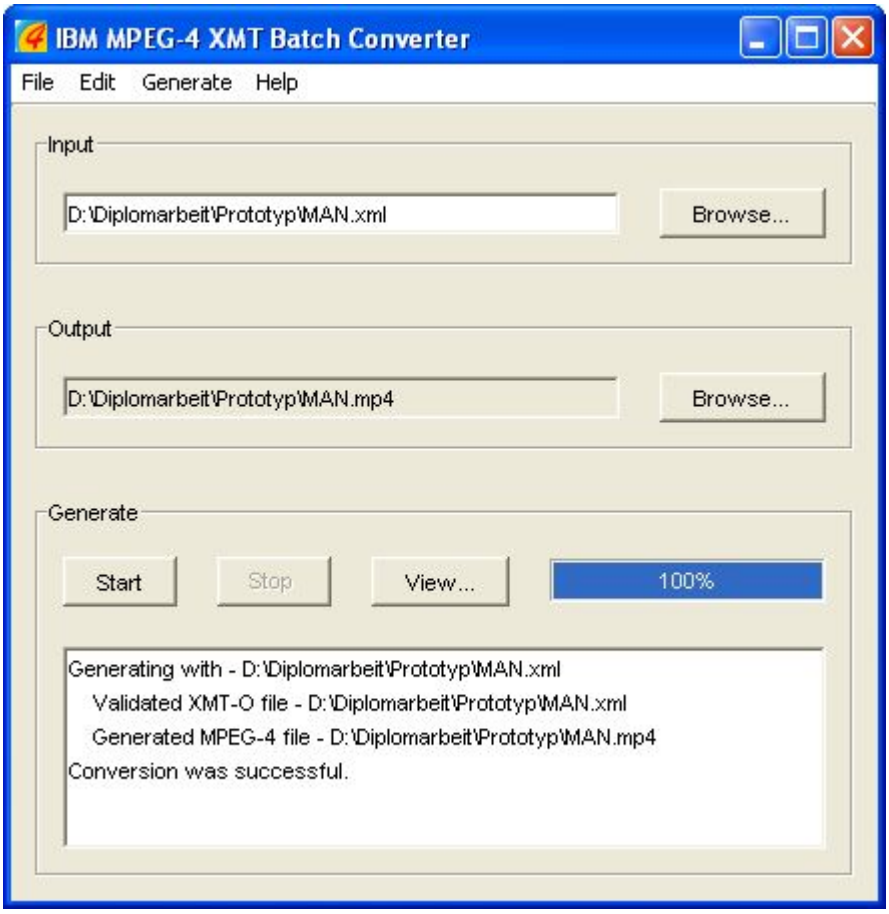

Abbildung 19: IBM MPEG-4 XMT Batch Converter

Mit dem MPEG-4 XMT Batch Converter ist es möglich, XMT-Ω zu XMT-A oder direkt zu einem binären MPEG-4 Stream zu kompilieren. Daneben kann auch XMT-A zu seinen binären Pendants transformiert werden und umgekehrt.

Wird als Ausgangsformat MP4 gewählt, so werden alle in einem XMT-Ω oder XMT-A Dokument referenzierten Medien zu einer einzigen MP4-Datei zusammengefasst[1]. Diese Medien müssen dabei in einem schon von MPEG-4 unterstützten Format vorliegen, das heißt Audio muss beispielsweise bereits in MP3 oder AAC komprimiert sein.

Eine Ausnahme bilden Vektorgrafiken, die in MPEG-4 mittels den in bereits in Kapitel 3.1.1 vorgestellten Primitiven abgebildet werden können und daher als eigenständiges Format nicht unterstützt werden. Der MPEG-4 XMT Batch Converter transformiert daher Vektorgrafiken bei der Kompilierung von XMT-Ω nach XMT-A oder MP4 direkt in solche Primitiven um. Bisher wird nur das weit verbreitete WMF[2]-Format unterstützt.

1 man spricht hierbei auch von "linken" (engl.)
2 Windows Metafile Format

Nachstehende Grafik zeigt grob den gesamten Arbeitsablauf und die Zusammenhänge aller beteiligten Dateien des Prototyps auf.

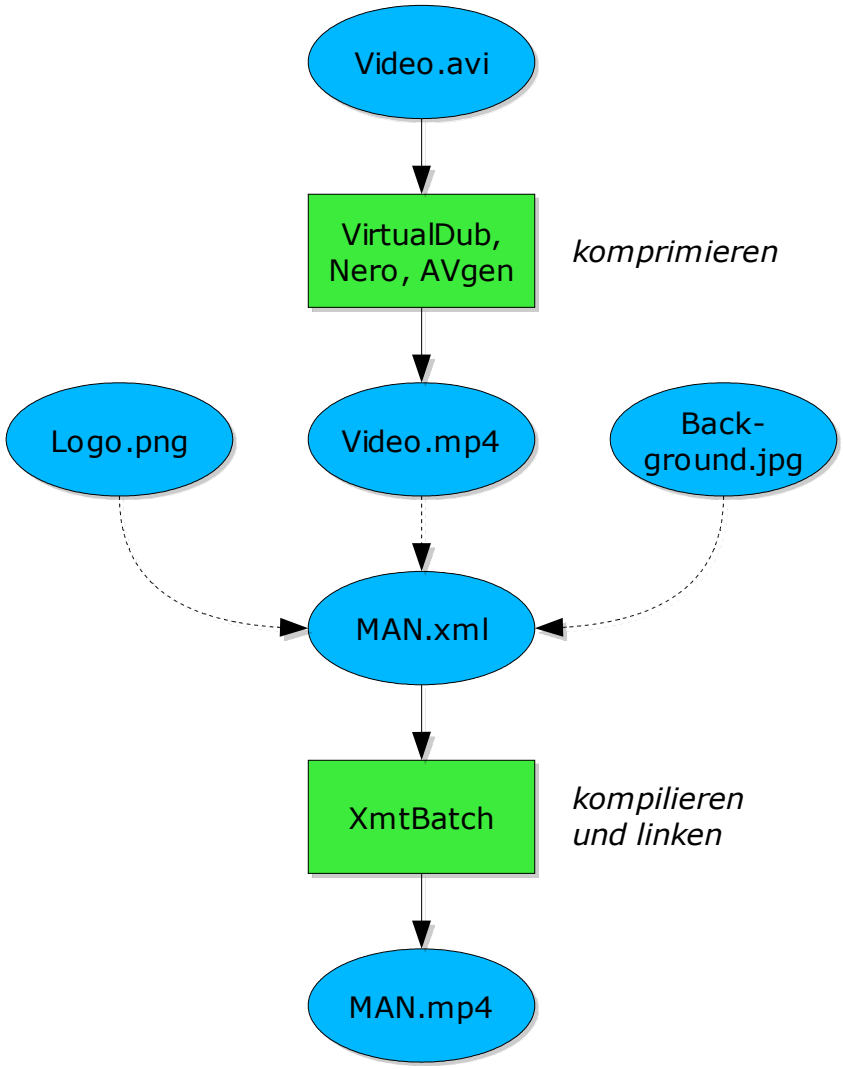

Abbildung 20: Arbeitsablauf

Wie die fertige RichMedia-Präsentation aussieht, zeigt der folgende Bildschirmausschnitt:

Abbildung 21: Bildschirmausschnitt des fertigen Prototyps

3.2 Konvertierung von bestehenden Inhalten

Folgender Abschnitt befasst sich mit der Frage, in wie fern sich bestehende und bereits auf Basis von RealMedia erstellte RichMedia-Anwendungen[1] des Fraunhofer-Instituts (IAO) nach MPEG-4 unter Verwendung von XMT konvertieren lassen können.

3.2.1 Audio und Video

Audio-visuelle Medien lassen sich in der Regel ohne Probleme in ein anderes Format konvertieren, sofern das Quellmaterial in einem gängigen Format vorliegt.

Dabei muss das Format einerseits in Komprimierungsformat und andererseits in Container- beziehungsweise Dateiformat – beispielsweise AVI – unterschieden werden.

Speziell das RealMedia-Format wird in beiden Gesichtspunkten nur sehr wenig seitens Konvertierungs- und Autorensoftware unterstützt.

1 bestehend aus dem proprietären RealMedia-Format (Audio und Video), SMIL (Szenenbeschreibung) und Bildern (Inhalt)

3.2.1.1 VirtualDub

Liegt das Quellmaterial wie im Fall des Projekts, in dem der Prototyp erstellt wurde, als unkomprimiertes AVI vor, so kann bereits mit kostenloser und frei verfügbarer Software – beispielsweise mit VirtualDub [VDUB] – der Inhalt nach MPEG-4 komprimiert werden.

Bei Verwendung von VirtualDub, das als Ausgabedateiformat nur AVI unterstützt, muss das Ergebnis anschließend in ein von MPEG-4 unterstütztes Dateiformat[1] umgewandelt werden. Dazu eignen sich unter anderem AVgen aus dem IBM Toolkit for MPEG-4 oder mp4UI[2] [MP4UI].

Ein weiterer Nachteil von VirtualDub ist die fehlende Unterstützung von MPEG-4 Audio (zum Beispiel AAC), das mittels einer weiteren Software in einem zweiten Schritt hinzugefügt werden müsste. MP3 wird zwar unterstützt, kann aber bei bestimmten Situationen[3] zu Problemen führen.

Die Vorteile von VirtualDub liegen in einer mächtigen Skriptsteuerung (bestimmte Vorgänge lassen sich dadurch über viele Dateien hinweg automatisieren) und der freien Wahl des Video- und sofern MP3 verwendet wird auch Audioencoders.

Populäre "standalone[4]" MPEG-4 Encoder für Video sind nachfolgend aufgelistet. Alle Encoder unterstützen dabei das effiziente Advanced Simple Profile inklusive B-Frames und 2-pass Encoding.

Encoder	Beschreibung
XviD [XVID]	kostenlos und frei verfügbar als Open Source[5], sehr vielfältige Einstellungsmöglichkeiten (Profile und Level sindunter anderem wählbar)
DivX von DivXNetworks [DIVX]	weite Verbreitung und "Unterstützung" von Endgeräten, übersichtliche Einstellungsmöglichkeiten, nur Pro-Version kostenpflichtig
3ivX von 3ivX Technologies [3IVX]	bisher weniger verbreitet, benutzerfreundlich, kostenpflichtig

Tabelle 7: Videoencoder

1 beispielsweise MP3 oder AAC für Audio beziehungsweise M4V oder CMP für Video (ohne Ton) – das Hauptdateiformat MP4, das Audio und Video gemeinsam beinhalten kann, wird meist ebenso unterstützt
2 entwickelt von Markus Brenner, dem Auto dieser Arbeit
3 das AVI-Dateiformat ist nicht für variable Audiobitraten ausgelegt – beim Einsatz könnte es zu asynchronen Verhalten von Audio und Video kommen
4 Encoder, welche sich nur über ein Dritt-Software nutzen lassen (unter Windows meist unter Einbeziehung der DirectShow- oder VfW-Schnittstelle)
5 für den kommerziellen Einsatz gelten besondere Bestimmungen

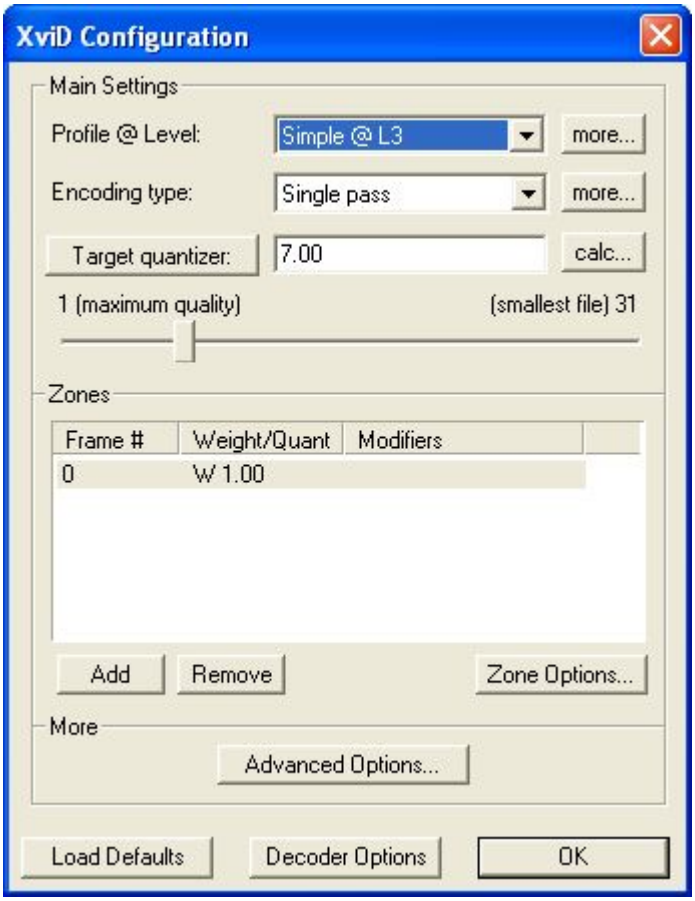

Abbildung 22: XviD Einstellungen

Nachstehend sind Audioencoder für das AAC-Format aufgelistet:

Encoder	**Beschreibung**
Nero Digital von Ahead [NERO]	Unterstützung von HE AAC, keine[1] Nutzung durch externe Software möglich, im weit verbreiteten Brennprogramm Nero[2] bereits enthalten
FAAC [FAAC]	kostenlos und frei verfügbar als Open Source[3], keine Unterstützung von HE AAC
compaact! von dicas digital image coding [MPGABL]	benutzerfreundlich (live mithören möglich), keine Unterstützung von HE AAC, kostenpflichtig

Tabelle 8: Audioencoder

1 es kursieren dennoch einige "Zusatz"-Tools wie NEncode, EziiiEncode oder OagMachine im Internet, welche die Beschränkung der Nutzung nur unter Nero "umgehen"
2 ab Version 6
3 für den kommerziellen Einsatz gelten besondere Bestimmungen

AAC, der Quasi-Nachfolger von MP3, ist aufgrund der besseren Effizienz, Qualität und Funktionalität (beispielsweise Mehrkanalton) gegenüber MP3 zu bevorzugen. Sollte MP3 zum Einsatz kommen, so können die Audiomedien beispielsweise mit dem kostenlosen und als Open Source[1] frei verfügbaren Lame Encoder [LAME] komprimiert werden.

Bei Encodern mit Unterstützung der Windows DirectShow-Schnittstelle kann auch das eigentlich für Programmentwickler gedachte Tool GraphEdit herangezogen werden. Es ist Teil des DirectX SDK. [DOOM9] [ATEST]

3.2.1.2 QuickTime Pro

Eine komfortablere Lösung zur Komprimierung von Medien bietet das kostenpflichtige QuickTime Pro von Apple.

Dieses Programm bietet eine beachtliche Anzahl an Formaten für den Import (beispielsweise DV[2]) und Export. Das MP4-Dateiformat wird auch unterstützt.

Die in QuickTime Pro bereits enthaltenen Encoder unterstützen aber nicht die effizienteren Formate Advanced Simple Profile (inklusive GMC und B-Frames) für Video und HE AAC für Audio. Externe Encoder, wie beispielsweise in VirtualDub möglich, lassen sich nur bedingt einsetzen.

Auch die Einstellungsmöglichkeiten sind begrenzt.

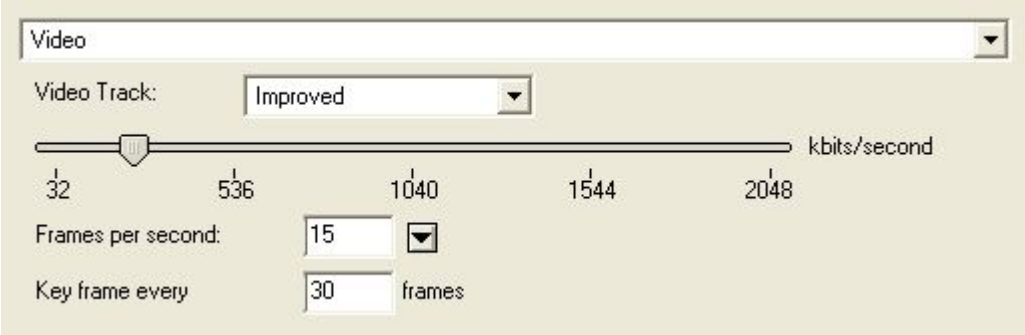

Abbildung 23: QuickTime MPEG-4 Video Codec Einstellungen

Einer im Rahmen dieser Arbeit durchgeführten Evaluierung zufolge ist die Kombination aus XviD/DivX (Video) und Nero Digital (Audio) in Bezug auf Effizienz und Qualität besser angebracht. [QUICKTIME]

3.2.1.3 mpegable X4 live

mpegable X4 live von dicas digital image coding bietet wie QuickTime Pro eine "rund-um" integrierte Softwareumgebung zur Konvertierung von Mediendateien nach MPEG-4 Standard.

1 es gelten besondere Bestimmungen für den kommerziellen Einsatz
2 häufig verwendetes Format für digitale Filmkameras

mpegable X4 live unterstützt dabei aber nicht so viele Formate für den Import und Export wie QuickTime Pro - das MP4-Dateiformat wird jedoch unterstützt. Dafür lassen sich mehr Einstellungen vornehmen. [MPGABL]

3.2.1.4 Studio Encode

Studio Encode von iVAST orientiert sich vom Funktionsumfang her an mpegable X4 live.

Neben Simple und Advanced Simple werden auch das Main Profile für die Komprimierung von Video unterstützt. Audio kann zudem in AAC (LC[1]) und CELP[2] kodiert werden.

Studio Encode bietet außerdem eine Echtzeitvorschau während dem Komprimiervorgang sowie den Import von in iVast Studio Author erstellten Projekten. Es lassen sich somit neue, anders komprimierte MP4-Dateien unter Beibehaltung von Szenenbeschreibung, Bildern, usw. erstellen. [IVAST]

3.2.2 Szenenbeschreibung und Inhalt

Im Gegensatz zu Audio und Video, die sich ohne größere Probleme zu MPEG-4 komprimieren lassen, ist die Konvertierung der Szenenbeschreibung selbst, das heißt SMIL nach XMT-Ω, nicht ohne weiteres möglich.

Der Grund hierfür ist, dass XMT-Ω sich zwar sehr stark an SMIL anlehnt, aber dennoch nicht in allen Bereichen und Details mit SMIL übereinstimmt.

XMT wurde mit dem Ziel entwickelt, MPEG-4 - insbesondere BIFS und OD – in Form von Text anstatt binären Daten beschreiben zu können. Um alle Vorteile und Möglichkeiten von MPEG-4 voll ausschöpfen zu können, sind deshalb zusätzliche Erweiterungen oder an MPEG-4 angepasste Ausdrücke in der Beschreibungssprache unabdingbar gewesen. Des weiteren waren zum Zeitpunkt der Entwicklung von XMT bereits wichtige Teile von MPEG-4 verabschiedet, sodass mit XMT auf den bestehenden MPEG-4 Standard Rücksicht genommen werden musste. Andere Funktionalitäten und Module aus SMIL waren somit überflüssig oder passten nicht in das Konzept von MPEG-4. Kapitel 2.6.1 bietet einen Überblick über die von MPEG-4 unterstützten SMIL-Module.

Es gibt bisher noch keine Autorensoftware mit (vollständiger) Unterstützung von sowohl SMIL als auch MPEG-4, mit dem der Import und Export von einem in das jeweils andere Format möglich wäre.

Ein Skript (zum Beispiel mittels XSLT) oder Programm für die Transformation von SMIL nach XMT-Ω selbst zu schreiben ist sehr umfangreich, da beide Standards

1 Low Complexity
2 Code Excited Linear Prediction - spezieller Audiocodec zur Komprimierung von Sprache

sehr komplex sind und es deshalb viele Besonderheiten zu beachten gibt. Sollen nur Dokumente mit wenigen enthaltenen Elementtypen konvertiert werden, so fällt dieser Ansatz entsprechend weniger umfangreich aus.

Bestimmte Codepassagen durch andere zu ersetzen oder ergänzen (beispielsweise die Attribute `width` und `height` zu `size` zusammenzufassen oder das Element `root-layout` in `topLayout` umzubenennen) würden demnach keine großen Hindernisse darstellen, vorausgesetzt es sollen nur "wenige spezielle" Dokumente, die nicht den SMIL-Standard in all seiner Komplexität ausnutzen, konvertiert werden.

Weitaus schwieriger stellt sich der Ansatz dar, in XMT und MPEG-4 fehlende Elemente in ihren Funktionalitäten nachzubilden. Als Beispiel soll hier die in XMT-Ω fehlende Möglichkeit, Objekte mittels `<a>` zu "verlinken", genannt werden. Mit dem Element `<a>` kann in XMT-Ω nur auf externe MPEG-4 Präsentationen verwiesen werden.

In folgendem SMIL-Beispiel werden ein Kreis sowie 3 Rechtecke für jeweils zehn Sekunden nacheinander dargestellt. Wird auf den Kreis geklickt, so wird unverzüglich das letzte der drei Rechtecke angezeigt, das heißt zeitlich gesehen zum letzten Rechteck "gesprungen".

```
<seq>
  <a href="#rect3">
    <circle id="circ" radius="100" dur="10s"/>
  </a>
  <rectangle id="rect1" size="20 20" dur="10s"/>
  <rectangle id="rect2" size="40 40" dur="10s"/>
  <rectangle id="rect3" size="60 60" dur="10s"/>
</seq>
```

Möchte man nun dieses Beispiel mit XMT-Ω umsetzen, kann man mehrere Möglichkeiten nutzen. Eine davon ist, alle Rechtecke über den gesamten Zeitraum parallel abzuspielen, wobei diese standardmäßig unsichtbar sind und nur zur jeweiligen Zeit sichtbar gemacht werden. Bei einem möglichen Klick auf den Kreis müssen darum das erste Rechteck unsichtbar und das letzte Rechteck sichtbar gemacht werden.

```
<par>
  <circle id="circ" radius="100" dur="10s"/>
  <rectangle id="rect1" size="20 20" dur="indefinite"
             visibility="false">
    <set attributeName="visibility" to="true" begin="circ.end"/>
    <set attributeName="visibility" to="false" begin="10s"/>
    <set attributeName="visibility" to="false" begin="circ.click"/>
  </rectangle>
  <rectangle id="rect2" size="40 40" dur="indefinite"
             visibility="false">
```

```
    <set attributeName="visibility" to="true" begin="rect1.end"/>
    <set attributeName="visibility" to="false" begin="10s"/>
  </rectangle>
  <rectangle id="rect3" size="60 60" dur="indefinite"
             visibility="false">
    <set attributeName="visibility" to="true" begin="rect2.end"/>
    <set attributeName="visibility" to="false" begin="10s"/>
    <set attributeName="visibility" to="true" begin="circ.click"/>
  </rectangle>
</par>
```

Dieses Exempel zeigt, dass aus einfachen Zusammenhängen komplexe Abhängigkeiten zwischen allen Elementen einer Präsentation entstehen können.

Müssen RealMedia-spezifische Erweiterungen wie Verweise auf externe Dokumente (beispielsweise PDF[1]-Dateien) oder Internet-Links eingebunden werden, führt dies zu weiteren "Schwierigkeiten" in der Umsetzung. PDF-Dateien könnten beispielsweise in Form von Bildausschnitten[2] in eine XMT-Anwendung einbezogen werden, nicht aber wie im Real-Player über das Adobe Acrobat PDF-Plugin. Links auf Web-Seiten lassen sich in XMT dagegen nicht[3] realisieren.

3.3 Erstellung von Inhalten

Da MPEG-4 ein "relativ" neuer Standard ist, gibt es bisher nur wenige Werkzeuge zum komfortablen Erstellen von MPEG-4 Inhalten über grafische Oberflächen.

Verfügbare Anwendungen decken bislang oft immer nur ein spezielles Gebiet ab, wie das Komprimieren von Audio oder Video. "Komplette" Autorensoftwareumgebungen[4], mit welchen sich auch Inhalte unter Ausnutzung der "neuen" RichMedia-Möglichkeiten wie in MPEG-4 Systems beschrieben erstellen lassen, sind bislang im professionellen und kommerziellen Umfeld noch wenig vertreten.

Einige Autorenprogramme, welche teilweise schon Funktionalitäten von MPEG-4 - insbesondere von XMT/BIFS und damit MPEG-4 Systems - unterstützen, sind:

- iVAST Studio Author [IVAST]
- 4Mation von Envivio [ENVIVIO]
- VeonStudio von Philips [VEON]
- IBM MPEG-4 XMT Authoring Tool [IMXET]

1 Portable Document Format
2 die Schwierigkeit liegt hierbei in der automatisierten Konvertierung eines PDF-Dokuments in Bilder
3 XMT/MPEG-4 ist "allgemein" gehalten und nicht wie SMIL speziell auf die Bereiche PC und Internet ausgerichtet
4 mit grafischer (Timeline, Drag & Drop) und codebasierter (XMT) Erstellung der Szenenbeschreibung, Komprimierung von Audio und Video, Import und Export von Fremdformaten, Debugger, integrierter Player, usw.

Da SMIL nun schon länger als Standard verabschiedet wurde, gibt es folglich einige ausgereifte Werkzeuge zur Erstellung von Inhalten auf SMIL-Basis. Populäre Vertreter von SMIL-Editoren sind beispielsweise:

- GRINS[1] Pro Editor for SMIL 2.0 von Oratrix [GRINS]
- Fluition von Confluent Technologies [FLUI]

Einfache Anwendungen, welche nicht die besonderen Fähigkeiten von MPEG-4 ausnutzen, lassen sich über den Umweg einer "Teil"-Konvertierung – beispielsweise über eine XSLT-Transformation - von SMIL nach XMT-Ω und MPEG-4 BIFS erstellen.

XMT-Ω ist zu SMIL nicht "100 %" kompatibel. Wie in Kapitel 3.2.2 bereits erläutert, lassen sich deshalb nicht alle SMIL-Konstrukte "1:1" konvertieren. Werden von XMT-Ω nicht unterstützte SMIL-Konstrukte verwendet, so ist eine manuelle Nachbearbeitung und Anpassung erforderlich.

Um weniger komplexe, aber umfangreiche Inhalte leicht und schnell mittels grafischen WYSIWYG[2]-Editoren zu erstellen, könnte diese Methode bis zum Erscheinen von vollwertigen XMT/MPEG-4 Autorenwerkzeugen zumindest eine "Ausweichmöglichkeit" darstellen.

3.4 Distribution von Inhalten

Da mit dem MPEG-4 Systems Standard flexible Mechanismen für den Transport sowie die Speicherung von Daten (siehe Kapitel 1.6.1.5) und speziell darauf zugeschnittenen Rahmenwerken (DMIF – mehr dazu in Kapitel 1.6.4) festgelegt sind, bieten sich vielfältige Möglichkeiten zur Distribution von Inhalten an.

Die Verteilung auf Datenträger und das Streaming über Netzwerke – also dem Herunterladen von Daten[3] im Moment des Abspielens - werden im Folgenden näher erläutert.

3.4.1 Datenträger

Sollen Inhalte - beispielsweise RichMedia-Anwendungen – auf Datenträgern (CD, DVD, Memory-Stick usw.) gespeichert oder an Dritte über solche weitergeben werden, kann dies mit dem dafür in MPEG-4 vorhergesehenen binären MP4-Dateiformat geschehen.

Dieses Dateiformat ist sehr flexibel entwickelt worden – gerade auch im Hinblick auf zukünftige MPEG-4 Technologien. Das heißt es ist nicht nur - wie beispielsweise MP3-Dateien - auf nur ein Format (eben das MP3-Format) und einen

1 Graphical Interface for SMIL
2 What You See Is What You Get
3 es werden nur Datenfragmente heruntergeladen, verarbeitet und danach wieder verworfen

einzigen Inhalt (Titel) festgelegt, sondern fungiert als Container für verschiedenste Formate, wobei sich auch mehrere Inhalte (Audio, Video, Text, Bilder, usw.) in Form von Tracks[1] darin speichern und untereinander verknüpfen lassen.

Aufgrund seines internen Aufbaus können auch zur eigenen Verwendung bestimmte Informationen (beispielsweise spezielle Metadaten wie Angaben zum Produktionsablauf oder ein privates, proprietäres Format) untergebracht werden.

Werden solche Dateien auf "fremden" Produkten verarbeitet, so werden diese Informationen zwar ignoriert, aber trotzdem beibehalten. Zusammenfassend kann das MP4-Dateiformat als das XML-Format für binäre Multimedia-Daten angesehen werden. [MPEG4-1]

3.4.2 Streaming

Sollen Inhalte "komfortabel" über das Internet angeboten werden, so bietet sich der Einsatz von Streaming-Technologie an.

Nutzer müssen dabei die Inhalte nicht erst als Datei (man stelle sich GB-große Videos vor) herunterladen, sondern können diese quasi "live" und in "realtime" auf ihren Endgeräten nutzen. Der Inhalt kann beim Streaming automatisch und dynamisch in Qualität, Auflösung, usw. der zur Verfügung stehenden Netzwerk-Bandbreite angepasst werden. MPEG-4 bietet in diesem Zusammenhang durch seine starke Skalierbarkeit[2] besondere Vorteile.

Die Inhalte selbst können in Form von MP4-Dateien auf dem Server abgelegt werden. Damit ein jeder Server diese Dateien auch "streamen" kann, müssen die Dateien mit so genannten Hint[3]-Tracks versehen sein. Solche Tracks können auch im Nachhinein hinzugefügt werden und beinhalten streaming-spezifische Information[4] über den Inhalt der MP4-Dateien.

Der Vorteil der zusätzlichen Hint-Tracks liegt darin, dass Streaming-Server die Formate der Inhalte selbst nicht verstehen und verarbeiten können müssen, da alle für das Streaming nötigen Informationen in diesen Hint-Tracks stehen. Im Gegensatz zu den proprietären Streaming-Formaten von RealNetworks und Microsoft wird dadurch die Implementierung und der Einsatz solcher Server einfacher, was einerseits zu mehr Interoperabilität und andererseits zu einer größeren Auswahl an Server-Software und dadurch mehr Wettbewerb führt.

1 engl. für Spuren
2 einen Einblick gewährt das Kapitel 1.6.2.2
3 engl. für Hinweis, Rat
4 zum Beispiel die maximale Größe eines darin vorkommenden Paketes – durch diese Information kann der Versand der Daten über das Übertragungsmedium/Netzwerk angepasst und optimiert werden, sodass beispielsweise keine zeitlichen Verzögerungen durch Fragmentierung entstehen

Im Folgenden werden populäre Streaming-Server, welche MPEG-4 unterstützen, vorgestellt:

Produkt	**Beschreibung**
Darwin Streaming Server [DARWIN]	kostenlos und frei verfügbar als Open Source, wenig Einstellungsmöglichkeiten, Konfiguration über Web-Interface, leicht einzurichten, keine "besonderen" Systemanforderungen nötig
QuickTime Streaming Server von Apple [QSERVER]	für den kommerziellen Einsatz gedachte Version des Darwin Streaming Server mit erweiterten Funktionalitäten und Einstellungsmöglichkeiten sowie einer grafischer Oberfläche, nur für das Macintosh-Betriebssystem erhältlich, kostenpflichtig
Helix Universal Server von RealNetworks [REAL]	sehr viele Funktionalitäten, leistungsfähig, bietet "Enterprise"-Merkmale wie "fail-over", teuer in Einkauf (ab $2.000) und Unterhaltung

Tabelle 9: MPEG-4 fähige Streaming-Server

Eben vorgestellte Produkte unterstützen alle das Streaming über die etablierten und offen gelegten Protokolle RTP[1] (Real-Time Transport Protocol) [RTP] in Verbindung mit RTSP[2] (Real Time Streaming Protocol) [RTSP]. Ersteres Protokoll gewährleistet die Echtzeitzustellung von Multimediadaten über Unicast[3] oder Multicast[4] Netzwerk-Services. RTSP hingegen ist ein Anwendungsprotokoll, mit welchem "Realtime"-Inhalte gesteuert[5] und geregelt werden können.

Außer dem Streaming über RTP/RTSP unterstützt MPEG-4 mittels dem so genannten Multiplex-Format[6] auch speziell das progressive Herunterladen und Abspielen von Inhalten über das HTTP[7]-Protokoll. M4X-Dateien lassen sich ohne besondere Einrichtung eines Streaming-Servers über das Internet empfangen. [MPEG4-1]

1 festgehalten als RFC (Request For Comments) 1889
2 niedergeschrieben als RFC 2326
3 1:1-Beziehung – Interaktion mit dem Server möglich
4 gleichzeitige "Ausstrahlung" an mehrere bestimmte Endgeräte – der Empfänger kann nur "mithören", nicht aber mit dem Server interagieren
5 beispielsweise das Ändern der Abspielposition
6 Dateiendung M4X
7 HyperText Transfer Protocol

3.5 Abspielen von Inhalten

Die Möglichkeit, Filme sehr kompakt komprimieren zu können, um sie somit auf CD zu brennen oder über das Internet auszutauschen, führte in den letzten Jahren zu einer enormen Verbreitung von DivX und somit MPEG-4 und ist vergleichbar mit dem Siegeszug von MP3 im Audiobereich.

Diese Verbreitung beschränkte sich aber lediglich auf die Videokomprimierung – die anderen Teile von MPEG-4 wie Audio oder Systems wurden nicht unterstützt.

So gibt es viele Programme, die meist nur in AVI eingebettete, MPEG-4 kodierte Videos mit MP3 als Audio abspielen können.

Das MP4-Dateiformat, das Grundlage anderer MPEG-4 Technologien ist, wird erst von wenigen Anwendungen unterstützt.

Seltener sind Player, welche BIFS oder andere Technologien, die in MPEG-4 Systems spezifiziert sind, unterstützen. Grund ist die Lizenzgebühr und Komplexität und der vor allem erst später entwickelten Technologien wie XMT.

Nachfolgend sind Programme aufgelistet, welche bereits in der Lage sind, RichMedia-Anwendungen aufbauend auf MPEG-4 Systems abzuspielen:

Programm	**Beschreibung**
M4Play aus dem IBM Toolkit for MPEG-4 [IBMM4]	betriebssystemunabhängig durch Implementierung in Java, keine Installation erforderlich bei Einbettung in eine Internetseite (Applet), Streaming-Unterstützung, nur 2D, nur "Play, Stop, Pause"
iVAST Experience Player [IVAST]	nur 2D, Streaming/DMIF-Unterstützung (auch Multi-Home[1]), als Browser-Plugin verfügbar, JavaScript-Support[2]
EnvivioTV von Envivio [ENVIVIO]	Plugin[3] für Real, QuickTime und Windows Media Player, Unterstützung von MPEG-J und Streaming, nur 2D
Octagon Player [OCTAGA]	ausgerichtet auf 2D und 3D - weniger auf Audio und Video, Unterstützung von Streaming, VRML, X3D und MUW[4], als Applet verfügbar, teuer (€520)

1 Referenzen auf externe Medien von verschiedenen Servern
2 Unterstützung (engl.)
3 zusätzliche Software, um ein Programm in seiner Funktionalität zu erweitern
4 Multi User Worlds

Programm	Beschreibung
BS Contact MPEG-4 von Bitmanagement [BITMT]	2D und 3D, Unterstützung von X3D, als Applet verfügbar, viele Funktionalitäten, gute Erweiterbarkeit (Plugins kompatibel zur IM1 MPEG-4 Reference Software)
IM1 Player [MPEG4-5]	Teil der MPEG-4 Reference Software, wie der Name andeutet, nur als Referenz und Prototyp für Entwickler gedacht
GPAC[1]/Osmo [GPAC]	kostenlos und frei verfügbar als Open Source[2], für Entwickler gedacht, Streaming-Unterstützung, nur 2D

Tabelle 10: MPEG-4 Player

Viele dieser Software-Player unterstützen den Standard MPEG-4 Systems nicht vollständig in all seinen Funktionalitäten und Möglichkeiten. Im Rahmen dieser Diplomarbeit wirkte sich dies dadurch aus, dass Inhalte nicht auf allen Playern abgespielt werden konnten. Der Prototyp konnte beispielsweise über M4Play aus dem IBM Toolkit for MPEG-4 abgespielt werden, nicht aber auf dem GPAC/Osmo-Player.

1 GPAC Project on Advanced Content
2 für den kommerziellen Einsatz gelten besondere Bestimmungen

4 Bewertung und Gegenüberstellung

Folgendes Kapitel bietet eine Bewertung und Gegenüberstellung von MPEG-4 mit anderen Technologien.

4.1 Komprimierungseffizienz

MPEG-4 vereint viele Vorteile gegenüber anderen am Markt befindlichen Standards. Ein wesentlicher Punkt ist die weit überlegenere Kompression von Medien hinsichtlich Qualität und Datengröße. Folgend wird Audio und Video anhand MPEG-4 HE AAC und MPEG-4 AVC, welche jeweils die fortschrittlichste Technologie in ihrem Bereich darstellen, erläutert und mit anderen Formaten verglichen.

4.1.1 Audio: MPEG-4 HE AAC

MPEG-4 HE AAC[1] ist laut der European Broadcasting Union (EBU) momentan die fortschrittlichste und hochwertigste Technologie zur allgemeinen Audiokompression weltweit. Dieses Audioformat ist ausgerichtet auf natürliche Audiosignale – der Einsatz reicht von Musik bis zum Mehrkanalton für Fernsehen und Kino.

MPEG-4 HE AAC ist eine abwärtskompatible Erweiterung um SBR (Spectral Band Replication) zu dem herkömmlichen AAC-Standard.

Gegenüber dem populärem MP3-Format (MPEG-1 Layer 3), welches CD-Qualität bei 128 kBit/s bietet, schafft dies AAC HE bereits bei nur 48 kBit/s. 5.1 Mehrkanalton wird im Vergleich dazu schon bei 128 kBit/s und gute Stereoqualität bei 32 kBit/s erreicht.

Die (technischen) Merkmale sind:

- Skalierbare Architektur: ermöglicht die dynamische Anpassung an bestimmte Gegebenheiten wie zur Verfügung stehende Bandbreite, Rechenleistung, usw.
- Sehr flexibel: 12 mögliche Sampleraten von 8 – 96 kHz, bis zu 256 kBit/s pro Kanal, bis 48 Kanäle
- Unterstützung von verschiedenen Profilen, zum Beispiel LD (Low Delay) mit Verzögerungen kleiner 20 ms für Mobil- und Internettelefonie oder LC (Low Complexity) und SSR (Scalable Sampling Rate) für leistungsschwache Decoder
- Spezielle Unterstützung von Fehlertoleranzmechanismen durch ER-Profile (Error Resilience)

Mehrere unabhängige Hörtests belegen, dass MPEG-4 HE AAC eine bessere Komprimierqualität und - effizienz gegenüber zurzeit am Mark erhältlichen Formaten bietet.

1 High-Efficiency Advanced Audio Coding

Ein 2002 von der European Broadcasting Union durchgeführter Test (48 kBit/s, Stereo) hebt hervor, dass die Erweiterung SBR, welche auch bei MP3 Pro zum Einsatz kommt, hierbei einen entscheidenden Anteil des Qualität/Bitraten-Verhältnisses hat. Das nachfolgende Diagramm des EBU-Hörtests zeigt den subjektiven Höreindruck der jeweiligen Formate im Verhältnis zum unkomprimierten "Original".

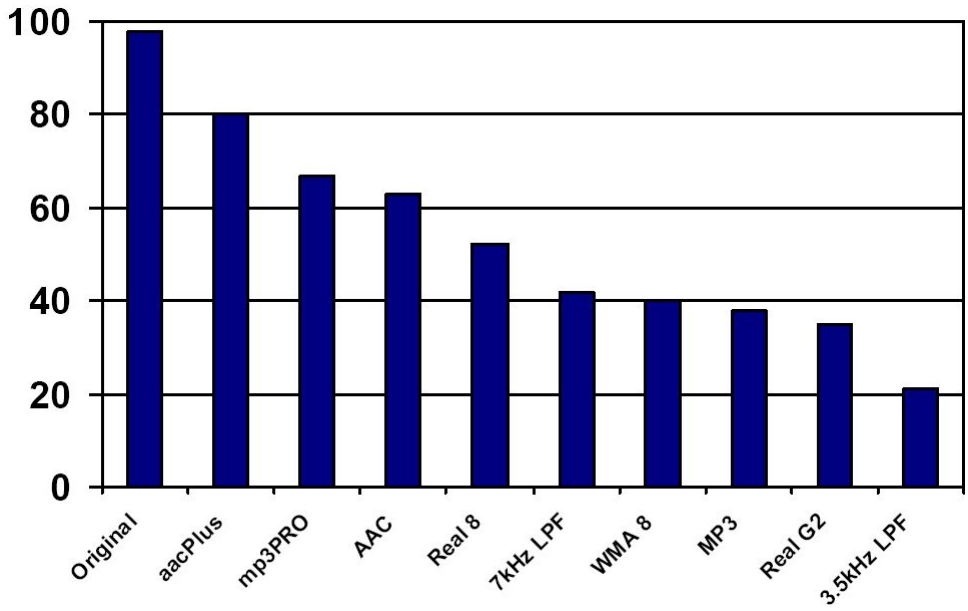

Abbildung 24: EBU Hörtest bei 48 kBit/s

Ein von Roberto Amorim im Internet [ATEST] privat durchgeführter Hörtest, bei dem sich jeder Internetnutzer beteiligen und seine Wertung abgeben konnte, vergleicht aktuelle Audiokompressionsformate bei einer Bitrate von 64 kBit/s.

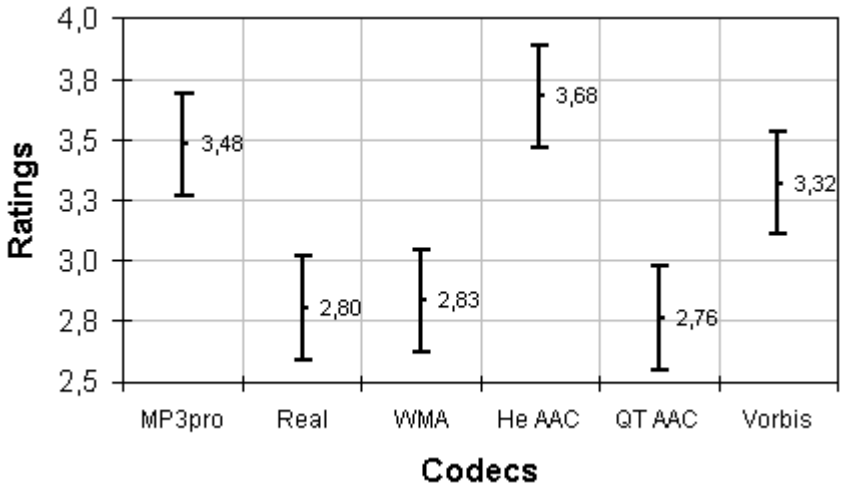

Abbildung 25: Vergleich von Audioformaten bei 64 kBit/s

Die Testmuster wurden mit den jeweils populärsten Implementierungen der jeweiligen Standards erstellt. MPEG-4 HE AAC schneidet hierbei am Besten ab.

Obwohl HE AAC erst im März 2003 offiziell in den MPEG-Standard mit aufgenommen wurde, stehen bereits "produktreife" Encoder sowie Decoder für HE AAC zur Verfügung. Anwendung findet der Standard unter anderem in XM Satellite Radio oder DRM[1] sowie in diversen Softwareprodukten (beispielsweise Nero von Ahead). [HEAAC] [MPEG4-3] [TMPEG4B] [ACODING] [CTECH]

4.1.2 Video: MPEG-4 AVC

MPEG-4 AVC[2] bietet eine ungefähr 50 Prozent höhere Kompression als die besten bisher erhältlichen Standards für die Komprimierung von Video.

Der Standard wurde in Zusammenarbeit von ITU-T und ISO/IEC entwickelt und adressiert die gesamte Bandbreite an Videoanwendungen wie Fernsehen, DVD, Kino, mobile Videotelefonie, usw.

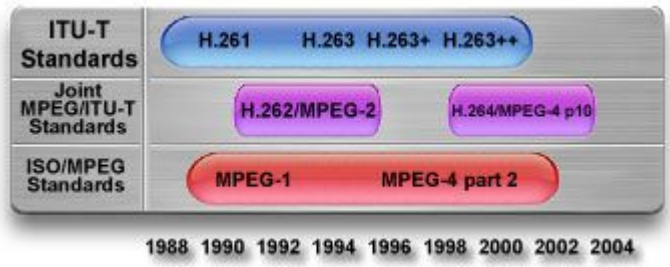

Abbildung 26: Videokomprimierstandard-Historie

MPEG-2 ist der bisher bedeutendste Videokomprimierstandard und heute in Massenmärkten wie DVD, digitales Fernsehen (DVB, ATSC[3]) und Set-Top-Boxen vertreten. MPEG-4 AVC stellt die größte einzelne Steigerung in Qualität und Effizienz seit der Einführung von MPEG-2 dar. Es wird erwartet, dass MPEG-4 AVC MPEG-2 und MPEG-4 ASP[4] in vielen Anwendungen ablösen und zudem neue Bereiche wie Video-over-IP/DSL erschließen wird.

	MPEG-4 ASP	**H.263**	**MPEG-2**
MPEG-4 AVC	39 %	49 %	64 %

Tabelle 11: Durchschnittlicher Bitratengewinn von MPEG-4 AVC

1 Digital Radio Mondiale (DRM) ist die digitale Umsetzung des analogen AM-Radios (SW-, LW- und MW-Bänder)
2 auch bekannt unter den Namen H.264, H.26L, JVT und MPEG-4 Part 10
3 ATSC (Advanced Television Systems Committee) ist ein amerikanischer, dem europäischen DVB (Digital Video Broadcasting) ähnelndem Standard für digitales Fernsehen
4 Advanced Simple Profile

Wichtige Neuerungen in MPEG-4 AVC sind nachfolgend aufgelistet:

- stark verbesserte Bewegungsschätzung und -vorhersage
- 4 x 4 Integer-Transformation (anstatt 8 x 8 DCT wie bei MPEG-2 und MPEG-4 ASP) führt zu keinen Rundungsfehlern mehr und weniger Artefakten
- Adaptiver In-Loop Deblocking Filter
- Verbesserte Entropie-Kodierung

AVC ist, wie viele andere Technologien in MPEG-4 auch, in Profile eingeteilt. Momentan sind dies das Baseline, Extended und Main Profile. Bei Letzterem ist im Vergleich zu MPEG-2 die Komplexität beim Kodiervorgang bis zu acht Mal und beim Dekodiervorgang bis zu vier Mal so groß.

AVC wurde als Standard bereits verabschiedet, ist im Gegensatz zu HE AAC aber noch nicht so ausgereift[1] (optimiert auf Geschwindigkeit und Qualität) und deshalb nur in sehr wenigen Produkten zu finden. In verschiedenen Schlüsselbereichen wie

- digitales Fernsehen (DVB 2.0)
- DVD (Nachfolger mit Unterstützung von HD[2])
- Video-over-IP (Internet Streaming Media Alliance)
- UMTS-Videotelefonie (3rd Generation Partnership Project)

wird aber schon an eine Einführung gedacht oder bereits daran entwickelt.

Laut [HARM] stellen AVC (Main Profile) zusammen mit HE AAC momentan die fortschrittlichsten und effizientesten Technologien in ihrem Bereich dar und werden schon in "naher Zukunft" in die Fußstapfen des überaus erfolgreichen MPEG-2 treten. [CSVT] [LSI] [BROENG]

4.2 Vergleich mit Internet-Multimedia-Standards

Herstellerunabhängige Lösungen wie MPEG-4 garantieren eine weite, erfolgreiche Verbreitung im Markt eher als solche, die nur von wenigen, einzelnen Firmen entwickelt und vermarktet werden. Proprietäre Lösungen können nur erfolgreich sein, wenn sie in allen Bereichen eine große Marktdurchdringung erlangen.

Die technischen Unterschiede zwischen MPEG-4 und anderen populären Internet-Multimedia-Standards zeigt folgende Tabelle auf:

1 wie die jeweiligen Standards implementiert werden, ist im MPEG-Standard nicht vorgeschrieben – Anbieter solcher Codecs können sich dadurch voneinander absetzen

2 High Definition (HD) bietet im Gegensatz zum "herkömmlichen" Fernsehformat Auflösungen bis zu 1920 x 1080 Pixel

	MPEG-4	**Windows Media**	**Real**	**Flash**
Audio/Video Codec	herstellerun-abhängig	proprietär	proprietär, MPEG-4 über Plugin	proprietär
Interaktivität	hohe Interaktivi-tät	begrenzt	ja, mittels SMIL	hohe Interak-tivität
Digitale Rech-teverwaltung	MPEG-4/21, offene Schnitt-stelle zu DRM-Systemen	Microsoft DRM	Zugriffsbe-rechtigung	nein
Echtzeit-Streaming	ja	ja	ja	nein
Synchronisa-tion	alle Objekte (hohe Genauig-keit)	nur Audio und Video	nur Audio und Video	keine Syn-chronisation zwischen Szene und Streams
Rund-funktauglich-keit	ja, inklusive in-teraktiver Inhalte	nur Audio und Video	Szene muss Unicast sein	nein
Objekt-orientierung	Audio/Video, 2D/3D, DRM, Grafik	nur Audio und Video	Audio/Video und weitere Medien (kein Streaming) mittels SMIL	Audio/Video und weitere Medien mittels proprietärem Protokoll
Grafikobjekte	ja	nein	nein	ja
Transport	HTTP, UDP, RTP/RTSP, MPEG-2 TS, mo-bile Netze	HTTP, UDP, RTP/RTSP, mobile Netze	HTTP, RTP/RTSP, mobile Netze	HTTP
PC, Set-Top-Boxen, mobile Anwendunge n	ja	ja	ja	nein

Tabelle 12: Vergleich von MPEG-4 mit Internet-Multimedia-Standards

[MPEGIF]

4.3 Erstellung, Übertragung und Konsum

Nachstehende Tabelle vergleicht MPEG-4 hinsichtlich Erstellung, Übertragung und Konsum mit der bisherigen Situation "aktueller" Technologien in diesen Bereichen:

	Bisher	**MPEG-4**
Erstellung	◆ meist 2D ◆ Natürlicher Inhalt, erstellt mit Kamera und Mikrophon ◆ Komposition von Objekten in der Produktionsstufe	◆ 2D und 3D ◆ Inhalt kann natürlich, computergeneriert (synthetisch) oder gemischt sein ◆ explizite Kodierung und Komposition von Objekten
Übertragung	◆ wenige Netzwerke übertragen AV-Informationen (Satellit, LAN, ISDN) ◆ viele Netzwerke haben spezielle Formate für die Inhaltsbeschreibung ◆ Übertragung über homogene Netzwerke	◆ fast jedes Netzwerk kann die audio-visuellen Informationen übertragen ◆ gemeinsames Format für die Inhaltsbeschreibung ◆ Übertragung über heterogene Netzwerke (mehrere Netzwerktypen)
Konsum	◆ Inhalt bereits geschnitten ◆ meist passiver Inhalt	◆ Komposition beim Endanwender ◆ Informationen können interaktiv gelesen, gehört und gesehen werden

Tabelle 13: Erstellung, Übertragung und Konsum

[MPEG4JS] [TMPEG4B]

4.4 Rechtemanagement und Schutz von Inhalten

MPEG-4 spezifiziert Möglichkeiten, die es dem Urheber erlauben, Inhalte vor Zugriff oder Änderungen zu schützen und mittels dynamischer und flexibler Rechte deren Nutzung zu kontrollieren.

Unter dieses so genannte DRM (Digital Rights Management) fallen auch:

- die Identifikation von Inhalten
- das automatische "Tracking[1]" von Werken

1 Weiterverfolgung (engl.)

- die Buchführung von Manipulationen
- die Unterstützung von Transaktionen zwischen Anbietern von Medien und den Inhabern der Rechte dieser Medien

Alle diese Forderungen[1] lassen sich in MPEG-4 über die so genannte IPMP-Schnittstelle (Intellectual Property Management and Protection) realisieren.

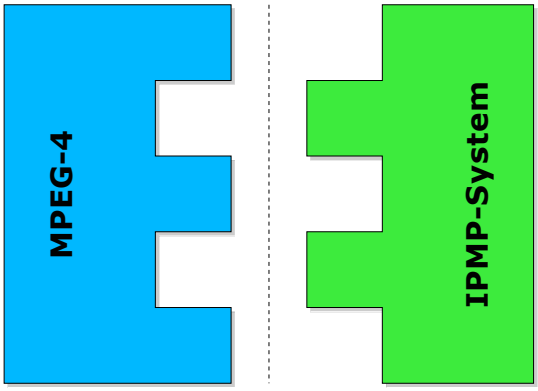

Abbildung 27: Architektur von IPMP

Über diese Schnittstelle, die Teil von MPEG-4 Systems ist, können externe DRM-Systeme eingebunden werden. MPEG-4 definiert also vielmehr nur die Benutzung und Einbindung solcher Systeme, nicht aber die Systeme selbst. Vertreter solcher DRM-Systeme sind beispielsweise ISMACryp von der gleichnamigen Internet Streaming Media Alliance [ISMA] oder OpenIPMP [OIPMP].

Dieser Ansatz führt im Gegensatz zu inkompatiblen und proprietären Technologien, wie sie bei RealNetworks oder Microsoft zu finden sind, zu folgenden Vorteilen:

- Interoperabilität
- Sicherheit
- Flexibilität
- Kompatibilität

Neben der Möglichkeit, Inhalte durch die eben angesprochenen DRM-Systeme explizit schützen zu können, so bietet - gerade bei RichMedia-Anwendungen mit textueller Codebasis – alleine schon ein binäres Format (BIFS, MP4-Dateiformat) ein gewisses Hindernis, dass Inhalte von Dritten "einfach" manipuliert werden können.

1 1997 von der MPEG in dem Dokument Call for Proposals for the Identification and Protection of Content niedergeschrieben

Im Falle von XMT-Ω kann zudem die ursprüngliche Beschreibung (Source Code) nicht mehr "1:1" aus den binären Daten zurückgewonnen werden – das hohe Abstraktionsniveau des ursprünglichen Quellcodes lässt sich so nicht mehr für umfassende Manipulationen nutzen. [N3943]

5 Zusammenfassung, Fazit und Ausblick

Dieses Kapitel bietet eine gesamtheitliche Zusammenfassung der Technologien XMT und MPEG-4 und ein Fazit hinsichtlich der Verwendung dieser Technologien für RichMedia sowie einen Ausblick.

5.1 Zusammenfassung der Technologien

Folgender Abschnitt fasst die Materie und Technologie, die sich hinter den beiden Schlagwörtern XMT und MPEG-4 versteckt, zusammen.

MPEG-4 ist eine Gruppe von Standards für multimediale Anwendungen und spezifiziert:

- Technologien zur effizienten Kompression und Darstellung von audio-visuellen Inhalten in natürlicher sowohl auch künstlicher[1] Form
- Möglichkeiten zur Beschreibung multimedialer und interaktiver Präsentationen (RichMedia) in 2D und 3D
- Protokolle und Datenformate zur Distribution und Speicherung von Inhalten

MPEG-4 adressiert kein spezielles Anwendungsfeld, sondern ist in allen (digitalen) Bereichen einsetzbar. Beispiele sind:

- digitales Fernsehen und Radio
- interaktiv aufbereitete RichMedia-Anwendungen im Internet
- Videotelefonie
- eLearning

Bei XMT handelt es sich um eine Beschreibungssprache für MPEG-4 Inhalte. Die Sprache wird in XML-Syntax ausgedrückt und folgt einer Zwei-Schichten-Architektur:

- XMT-Ω lehnt sich an SMIL 2.0 an und dient zur Beschreibung von RichMedia auf hohem Abstraktionsniveau
- XMT-A baut auf X3D (VRML) auf und dient zur Repräsentation von (binären) MPEG-4 Inhalten auf niedrigem Niveau

XMT-Ω kann zu XMT-A oder einem[2] binären MPEG-4 Datenstrom kompiliert werden.

1 computergeneriert – beispielsweise Vektorgrafiken oder Text-To-Speech

2 in XMT referenzierte (externe) Medien wie Audio, Video oder Bilder können zu einer einzigen Datei zusammengefasst werden

5.2 Fazit

Der Einsatz von XMT und MPEG-4 zur Erstellung von RichMedia wird in folgendem Abschnitt zusammenfassend als Fazit bewertet.

Die beiden Technologien XMT und MPEG-4 bieten viele Vorteile hinsichtlich der der Verwendung in RichMedia-Anwendungen. Nachfolgend sind die Vorteile einer solchen Verwendung aufgelistet:

- XMT und MPEG-4 sind herstellerunabhängige und nichtproprietäre Technologien und als internationale Standards verabschiedet. Sie garantieren eine weite und erfolgreiche Verbreitung, Akzeptanz und Unterstützung (in der Zukunft) eher als solche Technologien, die nur von wenigen, einzelnen Firmen entwickelt und vermarktet werden.
- Beide Technologien sind für den Einsatz in RichMedia-Anwendungen "ausgereift", das heißt dafür nötige Schlüsseleigenschaften wie Interaktion oder die gleichzeitige Darstellung von verschiedenen audio-visuellen Informationen werden abgedeckt.
- XMT ist aufgrund der Verwendung der XML-Syntax leicht zu lesen, generieren und zu ändern sowie einfach in eigene Produkte implementierbar[1]. Durch die Anlehnung an bestehende Standards (SMIL, X3D) ist die Sprache XMT zudem leicht von Anwendern erlernbar. XMT führt so zu einer schnelleren Etablierung am Markt als völlig neu entwickelte oder nicht offen gelegte Formate und Standards.
- MPEG-4 spezifiziert eine Vielfalt an Möglichkeiten für die Repräsentation von audio-visuellen Inhalten. Medien können dadurch in einer möglichst optimalen Weise - ausgerichtet auf die jeweiligen Einsatzzwecke (beispielsweise Sprache im Vergleich zu HiFi-Mehrkanalton) - kodiert und komprimiert werden. Dabei werden sowohl natürliche als auch synthetische Medien unterstützt.
- Für die Kompression von audio-visuellen Medien bietet MPEG-4 gegenüber aktuell verfügbaren Technologien effizientere Formate und Verfahren. Im Audiobereich ist dies herausstellend HE AAC und im Videobereich AVC.
- Inhalte in MPEG-4 sind skalierbar hinsichtlich Komplexität, Qualität, usw. und dadurch mehrmals in verschiedenen Anwendungsfeldern (PC, mobile Geräte, etc.) einsetzbar. Inhalte müssen so nur einmal erstellt und als Daten vorgehalten werden, um Anwendungen mit unterschiedlichen Anforderungen wie zur Verfügung stehende Bandbreite oder Rechenleistung gleichzeitig bedienen zu können.
- Für die Distribution und Speicherung von Daten stehen flexible Mechanismen (DMIF, Streaming) und Formate (MP4-Dateiformat) zur Verfügung. In

1 beispielsweise eine Exportfunktion

MPEG-4 erstellte Inhalte sind dadurch nicht auf spezielle Transport- oder Speichermedien angewiesen. Die Möglichkeiten reichen von Rundfunk über Internet bis zu Datenträgern.

- MPEG-4 bietet eine offene Schnittstelle (IPMP) zur Einbindung von DRM[1]-Systemen, um so Inhalte vor Zugriff oder Änderungen schützen und mittels dynamischer und flexibler Rechte deren Nutzung kontrollieren zu können.
- Profile und Levels stufen die Funktionalität und Komplexität der verschiedenen MPEG-4 Technologien in unterschiedliche Anforderungskriterien seitens einer Anwendung ein. Anwendungen für nur spezielle Einsatzzwecke müssen daher nicht den gesamten MPEG-4 Standard in all seinen Ausprägungen beherrschen. Versionen stellen zudem die Kompatibilität zu zukünftigen Entwicklungen in MPEG-4 sicher.

Gegen einen Einsatz der Technologien XMT und MPEG-4 im RichMedia-Umfeld sprechen zum jetzigen Zeitpunkt[2] folgende Kriterien:

- MPEG-4 und besonders XMT sind "relativ" neue Standards. Es sind folglich bisher nur wenige "komfortable" Produkte zum Erstellen und Abspielen von Inhalten vorhanden. Verfügbare Anwendungen decken oftmals immer nur ein Teilgebiet (zum Beispiel Audio- oder Videokomprimierung) ab.
- Die Szenenbeschreibung in Form von BIFS und XMT sowie weitere Teile von MPEG-4 Part: 1 Systems wie das MP4-Dateiformat werden in der Praxis bislang nur wenig eingesetzt und unzureichend[3] unterstützt. Verfügbare Produkte sind in diesen Bereichen oft noch nicht "ausgereift", das heißt sie sind inkompatibel, wenig benutzerfreundlich oder weisen Programmfehler auf.
- Produkte und Inhalte sind bislang oftmals noch nicht vollständig interoperatibel und funktionieren deshalb oft nur unter einer Geräte- oder Produktplattform eines Herstellers oder Anbieters. Das in MPEG-4 oft zitierte Motto "create once, run everywhere" hat sich deshalb bis jetzt noch nicht bestätigt.
- Die Szenenbeschreibungssprache XMT-Ω lehnt sich zwar an SMIL an, ist zu diesem Standard aber nicht in allen Punkten kompatibel. Vorhandene, auf SMIL basierende Inhalte und Produkte lassen sich aus diesem Grund nicht direkt wiederverwenden oder -nutzen. Die Konvertierung von bestehenden SMIL-Inhalten ist schwierig und komplex.
- Viele Firmen und Institutionen waren und sind an der Entwicklung des MPEG-4 Standards beteiligt. Von diesen Firmen und Institutionen eingebrachte Technologien sind teilweise patentiert. Um MPEG-4 in der Praxis einsetzen zu dürfen, müssen deshalb Lizenzgebühren bezahlt werden.

1 Digital Rights Management
2 8. März 2004
3 es werden nicht alle Möglichkeiten der jeweiligen Technologien implementiert

Wie zu erkennen ist, überwiegen – vor allem auf längere[1] Sicht hin bezogen - deutlich die Vorteile von XMT und MPEG-4 zur Erstellung von RichMedia und es ist davon auszugehen, dass die beiden Technologien sich deshalb im Bereich des RichMedia-Umfeldes erfolgreich am Markt etablieren werden.

5.3 Ausblick

Die nächsten zwei Unterkapitel bieten einen Einblick in zukünftige MPEG-Standards sowie einen Ausblick über die weitere Marktentwicklung des Multimedia-Standards MPEG-4.

5.3.1 Zukünftige MPEG-Standards

MPEG-7 und MPEG-21 - die vermeintlichen "Nachfolger" von MPEG-4 - stellen keine klassische Weiterentwicklung im Sinne von MPEG-4, das heißt im Speziellen der Kompression von audio-visuellen Informationen, dar. Beide Standards adressieren völlig neue Anwendungsfelder.

5.3.1.1 MPEG-7

Mit MPEG-7 können Medien anhand von Metadaten[2] beschrieben, klassifiziert und gefiltert werden.

Dabei geht es um mehr als das Hinterlegen von Informationen in einer Datenbank. Die zugrunde liegende Idee ist das automatisierte Extrahieren von Informationen aus den Inhalten, beispielsweise die Erkennung und Zuordnung von Personen in einem Video oder einem Gesprächsmitschnitt.

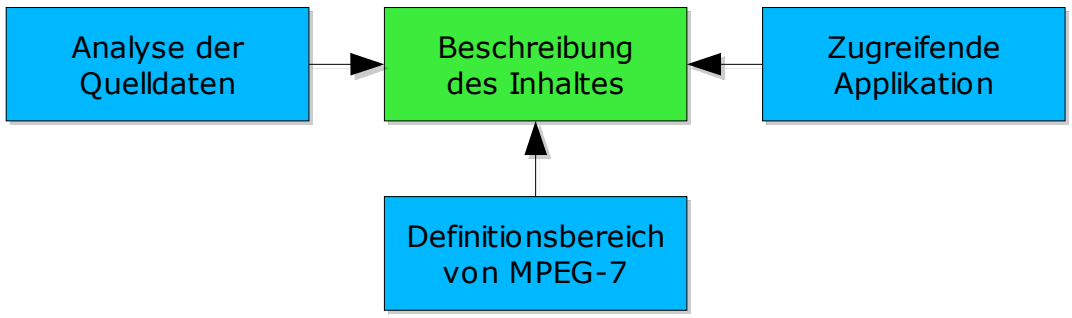

Abbildung 28: Definitionsbereich von MPEG-7

1 bis die anfänglichen "Anlaufschwierigkeiten" wie geringe Verfügbarkeit an Produkten, oder solchen, die sich noch in der Entwicklung befinden, "überwunden" sind

2 beispielsweise Informationen wie Autor oder Erstellungsjahr aber auch abstrakte Attribute wie die Anzahl der in einem Video gesprochenen Wörter

Die Medien (Video, Bilder, Musik, Sprache, usw.) werden hierbei durch eine Hierarchie von Deskriptoren, den Beschreibungselementen von MPEG-7, dargestellt. So kann zum Beispiel die Farbverteilung eines Bildes durch wenige Bits mit einem entsprechenden Farbdeskriptor beschrieben werden.

Durch die sehr kompakte Datenrepräsentation fungiert MPEG-7 in praktischen Anwendungen etwa als Vorfilterung, um nach bestimmten Suchkriterien eine Untermenge aus Millionen von Daten zu extrahieren.

Mit XMT wurde bereits innerhalb von MPEG-4 die Einbindung solcher MPEG-7 Daten berücksichtigt. MPEG-7 Version 1 wurde im September 2001 als internationaler Standard (ISO/IEC 15938) verabschiedet. Die bisherige Entwicklung reicht dabei bis in das Jahr 1996 zurück. [N5525] [MPEG]

5.3.1.2 MPEG-21

MPEG-21 wird unter der Bezeichnung ISO/IEC 21000 mit dem Ziel, ein einheitliches Rahmenwerk für das Management und die Nutzung von digitalen Daten zu schaffen, entwickelt.

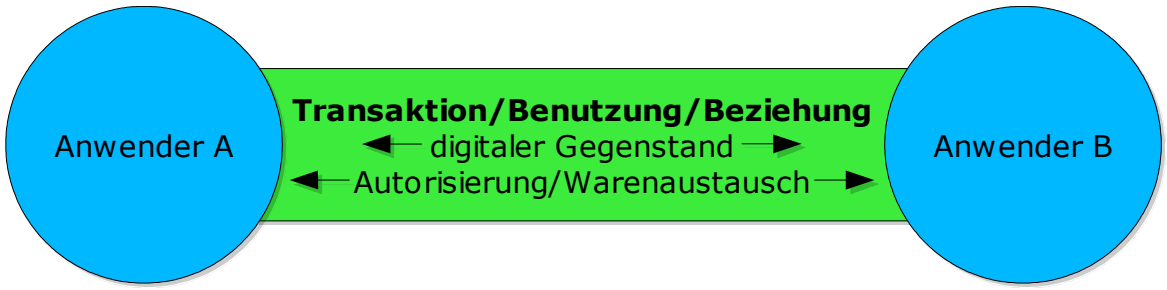

Abbildung 29: Rollenbasierte Beziehung in MPEG-21

Der Standard beschäftigt sich mit der Technologie, die gebraucht wird, um den Endanwender im Zugang, der Anwendung und im Tausch digitaler Daten zu unterstützen. Die Daten selbst sollen dabei auf eine möglichst effiziente, transparente und interoperatible Weise zur Verfügung gestellt werden.

MPEG-21 integriert Technologien zur:

- Erstellung/Produktion, Distribution/Archivierung und Konsum von Inhalten
- Verwaltung digitaler Rechte und zum Schutz von Inhalten
- Identifikation, Beschreibung und Deklaration von Inhalten
- Abstraktion von Terminals[1] und Netzwerken
- Benachrichtigung bei Ereignissen

1 Endgeräten (engl.)

Der Standard MPEG-21 befindet sich in einem frühen "Entwicklungsstadium" – viele Teile liegen deshalb bisher nur als Entwurf oder Empfehlung vor. [N5231] [MPEG]

5.3.2 Marktentwicklung

MPEG-4 und vor allem XMT sind "relativ" neue Standards, zu denen es nicht viele marktwirtschaftliche Informationen und Untersuchungen gibt. Folgender Abschnitt versucht deshalb eine eigene persönliche Einschätzung von XMT und MPEG-4 hinsichtlich der weiteren Marktentwicklung aufzuzeigen.

MPEG-4 hat sich bisher am *stärksten* im PC-Umfeld behauptet. Der populäre DivX[1]-Videocodec avancierte in den letzten Jahren zum Pendant zu MP3 und sorgt im Internet und in Tauschbörsen (zum Leidwesen von Rechteinhabern) für eine enorme Verbreitung von (tausenden illegalen) Videofilmen.

Die Verbreitung von solchen Filmen und damit von MPEG-4 *könnte* durch die Einführung von weiteren Unterhaltungsgeräten[2] (DVD-Player, PVR[3], Streaming-Boxen, AV-Receiver, usw.) mit derartiger Abspielfunktion rapide zunehmen. In MPEG-4 komprimierte Filme lassen sich dadurch zum ersten Mal von jedermann in "Wohnzimmerumgebung" nutzen und sind nicht mehr nur versierten Computerfreaks vorbehalten.

MPEG-4 Audio (AAC) erfährt *erstmalig* durch das populäre Online-Musikportal iTunes weite Verbreitung im Unterhaltungsbereich. Immer mehr tragbare Abspielgeräte unterstützen neben MP3 nunmehr auch AAC. Der Konzern Apple setzt neben iTunes auch in seiner QuickTime-Produktfamilie verstärkt auf MPEG-4. So werden Musik und Videofilme im MP4-Dateiformat angeboten.

Das erst kürzlich verabschiedete effiziente Audioformat HE AAC findet bereits ebenfalls Anwendung im digitalen Rundfunk[4]. Des weiteren *könnte* es MP3 als Audioformat bei MPEG-4 komprimierten Videofilmen auf längere Sicht hin ablösen, da sich damit Filme mit 5.1 Mehrkanalton realisieren lassen, die auch noch auf einer CD Platz finden. Der Anbieter Nero hat Anfang 2004 die erste "rundum" integrierte Software zum Transkodieren von DVD nach MPEG-4[5] vorgestellt. Unterhaltungsgeräte mit Unterstützung für die Formate (HE) AAC und MP4 werden *vermutlich* in den nächsten Monaten folgen.

MPEG-4 AVC *könnte* speziell in (mobilen) Anwendungen mit niedriger Bandbreite oder bei Fernsehen oder Filmen mit hoher Auflösung (High Definition) Anwendung finden. Videotelefonie über die neuen UMTS-Netze *soll* sich damit zur

1 MPEG-4 ASP Implementierung
2 Sigma Designs präsentierte bereits 2001 den ersten MPEG-4 Video-Chipsatz für Unterhaltungsgeräte [SIG]
3 Personal Video Recorder
4 beispielsweise Digital Radio Mondiale oder XM Satellite Radio
5 es werden ASP für Video, HE AAC für Audio (Mehrkanalton) und das MP4-Dateiformat (Metainformationen, Untertitel) unterstützt

"Killerapplikation" entwickeln. Fernsehen oder Video-on-Demand über IP/DSL *könnte* damit auch Realität werden. Zusammen mit HE AAC wird AVC bereits bei den Nachfolgern von DVD und DVB als das mögliche Format für Audio beziehungsweise Video gehandelt.

Die Technologien BIFS und XMT sind bisher nur in wenigen kommerziellen Softwareprodukten oder gar Hardware-Geräten zu finden. BIFS *könnte* jedoch beispielsweise im Nachfolger der DVD oder in zukünftigen Set-Top-Boxen zur Darstellung von interaktiven Inhalten (beispielsweise bei der Navigation durch das Menü einer DVD) Anwendung finden. RichMedia-Anwendungen wie eLearning wären damit nicht *nur mehr* auf den PC beschränkt.

Da XMT vor allem eine textbasierte Variante von BIFS ist, *dürfte* die Anwendung und der Einsatz von XMT besonders von der weiteren Unterstützung von BIFS seitens Abspielanwendungen und -geräten abhängig sein. XMT *wird* hauptsächlich für die Erstellung und den Austausch von Inhalten zwischen Autoren eingesetzt werden und weniger als Format selbst Unterstützung auf Playern finden.

Einige in MPEG-4 spezifizierte Technologien wie MPEG-J, synthetische[1] Audioformate oder FBA (Face and Body Animation) *werden* in nächster Zukunft *wohl weniger* eingesetzt werden oder populär sein. Ein Faktor dafür *könnte* die Komplexität ihrer Implementierungen im Verhältnis zu deren Vorteil zu bereits existierenden "Lösungen" oder Standards sein.

1 künstlich, computergeneriert

Anhang A: XMT-Ω Dokument

Nachfolgend ist der Inhalt der Datei MAN.xml – Teil des Prototyps aus Kapitel 3.1.1 - abgebildet. Zur besseren Lesbarkeit wurden Zeilenumbrüche manuell gesetzt.

```
<?xml version="1.0" encoding="UTF-8"?>
<XMT-O xmlns="urn:mpeg:mpeg4:xmto:schema:2002"
       xsi:schemaLocation="urn:mpeg:mpeg4:xmto:schema:2002
       schemas/xmt-o/xmt-o.xsd"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <!-- meta stuff -->
    <meta name="title" content="XMT/MPEG-4 Demo:
          MAN LKW Endabnahme"/>
    <meta name="author" content="Markus Brenner"/>
    <meta name="copyright" content="Copyright© 2004
          Markus Brenner, FhG IAO Stuttgart, HdM Stuttgart"/>
    <!-- layout/regions -->
    <layout metrics="pixel" type="xmt/xmt-basic-layout">
      <topLayout backgroundColor="black" width="734" height="418">
        <region id="regVideo"
                translation="-181 55" size="352 288"/>
        <region id="regSlide"
                translation="181 55" size="352 288"/>
        <region id="regMenu"
                translation="-107 -148" size="500 100"/>
        <region id="regLogo"
                translation="255 -148" size="204 100"/>
      </topLayout>
    </layout>
    <!-- transitions -->
    <transition id="transSlideWipeTop" type="slideWipe"
                subtype="fromTop" dur="2s"/>
    <transition id="transSlideWipeRight" type="slideWipe"
                subtype="fromRight" dur="2s"/>
    <transition id="transSlideWipeLeft" type="slideWipe"
                subtype="fromLeft" dur="2s"/>
    <!-- definitions -->
    <defs>
      <!-- intro/outro font -->
      <fontStyle id="defIntroOutroFont"
                 justify="MIDDLE; MIDDLE" size="32"
                 style="BOLD" family="&amp;quot;SANS&amp;quot;"/>
      <material id="defIntroOutroFontMat"
                color="white" filled="true"/>
      <!-- menu font -->
```

```
    <fontStyle id="defMenuFont"
               justify="BEGIN; MIDDLE" size="16"
               style="BOLD" family="&quot;SANS&quot;"/>
    <material id="defMenuFontMat"
              color="white" filled="true"/>
    <!-- slide font -->
    <fontStyle id="defSlideFont"
               justify="BEGIN; BEGIN" size="17"
               style="BOLD" family="&quot;SANS&quot;"/>
  </defs>
</head>
<body>
  <par>
    <!-- main background -->
    <img id="imgBackground" src="Media/Background.jpg"/>
    <seq>
      <!-- intro -->
      <group id="intro" dur="6s"
             transIn="transSlideWipeTop"
             transOut="transSlideWipeTop">
        <rectangle size="367 209">
          <material color="blue" filled="true"
                    transparency="0.9"/>
        </rectangle>
        <string textLines="&quot;XMT/MPEG-4 Demo:&quot;;
                &quot;MAN LKW Endabnahme (1)&quot;">
          <use xlink:href="#defIntroOutroFont"/>
          <use xlink:href="#defIntroOutroFontMat"/>
        </string>
      </group>
      <!-- slide show -->
      <par end="video.end">
        <!-- video incl. audio -->
        <video id="video" src="Media/Video.mp4#video"
               dur="media" region="regVideo"
               transIn="transSlideWipeLeft"
               transOut="transSlideWipeRight"/>
        <audio id="audio" src="Media/Video.mp4#audio"
               dur="media"/>
        <!-- logo -->
        <img id="imgLogo" src="Media/Logo.png"
             region="regLogo"
             transIn="transSlideWipeRight"
             transOut="transSlideWipeLeft"/>
        <!-- menu -->
        <group id="menu" region="regMenu"
               transIn="transSlideWipeLeft"
               transOut="transSlideWipeRight">
```

```
<rectangle size="500 100">
  <material color="blue" filled="true"
            transparency="0.9"/>
</rectangle>
<string id="menu1" textLines="&quot;1. Demontage
        vorbereiten&quot;" dur="indefinite">
  <transformation translation="-230 20"/>
  <use xlink:href="#defMenuFont"/>
  <material color="white" filled="true"
            transparency="0.4">
    <set attributeName="transparency" to="0"
         begin="slide1.begin"/>
    <set attributeName="transparency" to="0.4"
         begin="slide1.end"/>
  </material>
</string>
<string id="menu2" textLines="&quot;2. Ansatzteile
        demontieren&quot;" dur="indefinite">
  <transformation translation="-70 20"/>
  <use xlink:href="#defMenuFont"/>
  <material color="white" filled="true"
            transparency="0.4">
    <set attributeName="transparency" to="0"
         begin="slide2.begin"/>
    <set attributeName="transparency" to="0.4"
         begin="slide2.end"/>
  </material>
</string>
<string id="menu3" textLines="&quot;3. Spoiler
        demontieren&quot;" dur="indefinite">
  <transformation translation="100 20"/>
  <use xlink:href="#defMenuFont"/>
  <material color="white" filled="true"
            transparency="0.4">
    <set attributeName="transparency" to="0"
         begin="slide3.begin"/>
    <set attributeName="transparency" to="0.4"
         begin="slide3.end"/>
  </material>
</string>
<string id="menu4" textLines="&quot;4. Spoiler
        versetzen&quot;" dur="indefinite">
  <transformation translation="-230 -20"/>
  <use xlink:href="#defMenuFont"/>
  <material color="white" filled="true"
            transparency="0.4">
    <set attributeName="transparency" to="0"
         begin="slide4.begin"/>
```

```
            <set attributeName="transparency" to="0.4"
                 begin="slide4.end"/>
          </material>
        </string>
        <string id="menu5" textLines="&quot;5. Spoiler
                versetzen&quot;" dur="indefinite">
          <transformation translation="-70 -20"/>
          <use xlink:href="#defMenuFont"/>
          <material color="white" filled="true"
                    transparency="0.4">
            <set attributeName="transparency" to="0"
                 begin="slide5.begin"/>
            <set attributeName="transparency" to="0.4"
                 begin="slide5.end"/>
          </material>
        </string>
        <string id="menu6" textLines="&quot;6. Gurte
                entfernen&quot;" dur="indefinite">
          <transformation translation="100 -20"/>
          <use xlink:href="#defMenuFont"/>
          <material color="white" filled="true"
                    transparency="0.4">
            <set attributeName="transparency" to="0"
                 begin="slide6.begin"/>
          </material>
        </string>
      </group>
      <!-- slide rectangle -->
      <rectangle size="352 288" end="video.end"
                 region="regSlide"
                 transIn="transSlideWipeRight"
                 transOut="transSlideWipeLeft">
        <material color="blue" filled="true"
                  transparency="0.9"/>
      </rectangle>
      <!-- slides -->
      <seq end="video.end">
        <!-- slide 1 -->
        <group id="slide1" begin="2s" dur="96s"
               region="regSlide">
          <string id="item1_1" textLines="&quot;Fahrzeug
                  vom Abstellplatz zum Montageplatz
                  fahren&quot;" dur="indefinite">
            <transformation translation="-166 134"/>
            <use xlink:href="#defSlideFont"/>
            <material color="white" filled="true"
                      transparency="0.4">
```

```
            <set attributeName="transparency" to="0"
                 begin="18s"/>
            <set attributeName="transparency" to="0.4"
                 begin="33s"/>
          </material>
        </string>
        <string id="item1_2" textLines="&quot;Kontrolle
                des Fahrzeugabnahmeprotokolls&quot;"
                dur="indefinite">
          <transformation translation="-166 104"/>
          <use xlink:href="#defSlideFont"/>
          <material color="white" filled="true"
                    transparency="0.4">
            <set attributeName="transparency" to="0"
                 begin="33s"/>
            <set attributeName="transparency" to="0.4"
                 begin="54s"/>
          </material>
        </string>
        <string id="item1_3" textLines="&quot;
                Arbeitsbühne ans Fahrzeug
                heranfahren&quot;" dur="indefinite">
          <transformation translation="-166 74"/>
          <use xlink:href="#defSlideFont"/>
          <material color="white" filled="true"
                    transparency="0.4">
            <set attributeName="transparency" to="0"
                 begin="54s"/>
            <set attributeName="transparency" to="0.4"
                 begin="69s"/>
          </material>
        </string>
        <string id="item1_4" textLines="&quot;Die Räder
                der Arbeitsbühne arretieren&quot;"
                dur="indefinite">
          <transformation translation="-166 44"/>
          <use xlink:href="#defSlideFont"/>
          <material color="white" filled="true"
                    transparency="0.4">
            <set attributeName="transparency" to="0"
                 begin="69s"/>
            <set attributeName="transparency" to="0.4"
                 begin="78s"/>
          </material>
        </string>
        <string id="item1_5" textLines="&quot;
                Durchluftzuleitung an der Arbeitsbühne
                anschließen&quot;" dur="indefinite">
```

```
            <transformation translation="-166 14"/>
            <use xlink:href="#defSlideFont"/>
            <material color="white" filled="true"
                      transparency="0.4">
              <set attributeName="transparency" to="0"
                   begin="78s"/>
              <set attributeName="transparency" to="0.4"
                   begin="87s"/>
            </material>
          </string>
          <string id="item1_6" textLines="&quot;Leiter
                 einhängen&quot;" dur="indefinite">
            <transformation translation="-166 -16"/>
            <use xlink:href="#defSlideFont"/>
            <material color="white" filled="true"
                      transparency="0.4">
              <set attributeName="transparency" to="0"
                   begin="87s"/>
            </material>
          </string>
        </group>
        <!-- slide 2 -->
        <group id="slide2" dur="98s" region="regSlide">
          <string id="item2_1" textLines="&quot;
                  Dachspoiler ausfahren&quot;"
                  dur="indefinite">
            <transformation translation="-166 134"/>
            <use xlink:href="#defSlideFont"/>
            <material color="white" filled="true"
                      transparency="0.4">
              <set attributeName="transparency" to="0"
                   begin="3s"/>
              <set attributeName="transparency" to="0.4"
                   begin="21s"/>
            </material>
          </string>
          <string id="item2_2" textLines="&quot;Untere
                  Verschraubung wird gelöst&quot;"
                  dur="indefinite">
            <transformation translation="-166 104"/>
            <use xlink:href="#defSlideFont"/>
            <material color="white" filled="true"
                      transparency="0.4">
              <set attributeName="transparency" to="0"
                   begin="21s"/>
              <set attributeName="transparency" to="0.4"
                   begin="34s"/>
            </material>
```

```
      </string>
      <string id="item2_3" textLines="&quot;Obere
              Verschraubung wird gelöst&quot;"
              dur="indefinite">
        <transformation translation="-166 74"/>
        <use xlink:href="#defSlideFont"/>
        <material color="white" filled="true"
                  transparency="0.4">
          <set attributeName="transparency" to="0"
               begin="34s"/>
          <set attributeName="transparency" to="0.4"
               begin="52s"/>
        </material>
      </string>
      <string id="item2_4" textLines="&quot;Ansatzteil
              wird abgenommen und abgelegt&quot;"
              dur="indefinite">
        <transformation translation="-166 44"/>
        <use xlink:href="#defSlideFont"/>
        <material color="white" filled="true"
                  transparency="0.4">
          <set attributeName="transparency" to="0"
               begin="52s"/>
          <set attributeName="transparency" to="0.4"
               begin="61s"/>
        </material>
      </string>
      <string id="item2_5" textLines="&quot;
              Wiederholung des Vorganges mit dem 2.
              Ansatzteil&quot;" dur="indefinite">
        <transformation translation="-166 14"/>
        <use xlink:href="#defSlideFont"/>
        <material color="white" filled="true"
                  transparency="0.4">
          <set attributeName="transparency" to="0"
               begin="61s"/>
        </material>
      </string>
    </group>
    <!-- slide 3 -->
    <group id="slide3" dur="50s" region="regSlide">
      <string id="item3_1" textLines="&quot;Lösen der
              vier Verschraubungen des Spoilers&quot;"
              dur="indefinite">
        <transformation translation="-166 134"/>
        <use xlink:href="#defSlideFont"/>
        <material color="white" filled="true"
                  transparency="0.4">
```

```
            <set attributeName="transparency" to="0"
                 begin="5s"/>
            <set attributeName="transparency" to="0.4"
                 begin="36s"/>
          </material>
        </string>
        <string id="item3_2" textLines="&quot;
                Währendessen fährt der 2. Arbeiter den
                Portalkran&quot;;&quot;in Position
                &quot;" dur="indefinite">
          <transformation translation="-166 104"/>
          <use xlink:href="#defSlideFont"/>
          <material color="white" filled="true"
                    transparency="0.4">
            <set attributeName="transparency" to="0"
                 begin="36s"/>
          </material>
        </string>
      </group>
      <!-- slide 4 -->
      <group id="slide4" dur="61s" region="regSlide">
        <string id="item4_1" textLines="&quot;Gurte am
                Spoiler befestigen&quot;"
                dur="indefinite">
          <transformation translation="-166 134"/>
          <use xlink:href="#defSlideFont"/>
          <material color="white" filled="true"
                    transparency="0.4">
            <set attributeName="transparency" to="0"
                 begin="3s"/>
            <set attributeName="transparency" to="0.4"
                 begin="28s"/>
          </material>
        </string>
        <string id="item4_2" textLines="&quot;Spoiler
                wird angehoben&quot;" dur="indefinite">
          <transformation translation="-166 104"/>
          <use xlink:href="#defSlideFont"/>
          <material color="white" filled="true"
                    transparency="0.4">
            <set attributeName="transparency" to="0"
                 begin="28s"/>
            <set attributeName="transparency" to="0.4"
                 begin="40s"/>
          </material>
        </string>
```

```
          <string id="item4_3" textLines="&quot;
                  Schraublöcher werden mit
                  Kunststofftropfen verschlossen&quot;"
                  dur="indefinite">
            <transformation translation="-166 74"/>
            <use xlink:href="#defSlideFont"/>
            <material color="white" filled="true"
                      transparency="0.4">
              <set attributeName="transparency" to="0"
                   begin="40s"/>
            </material>
          </string>
        </group>
        <!-- slide 5 -->
        <group id="slide5" dur="15s" region="regSlide">
          <string id="item5_1" textLines="&quot;Spoiler
                  ablegen&quot;" dur="indefinite">
            <transformation translation="-166 134"/>
            <use xlink:href="#defSlideFont"/>
            <material color="white" filled="true"
                      transparency="0.4">
              <set attributeName="transparency" to="0"
                   begin="6s"/>
            </material>
          </string>
        </group>
        <!-- slide 6 -->
        <group id="slide6" dur="18s" region="regSlide">
          <string id="item6_1" textLines="&quot;Gurte
                  entfernen&quot;" dur="indefinite">
            <transformation translation="-166 134"/>
            <use xlink:href="#defSlideFont"/>
            <material color="white" filled="true"
                      transparency="0.4">
              <set attributeName="transparency" to="0"
                   begin="0s"/>
            </material>
          </string>
        </group>
      </seq>
    </par>
    <!-- outro -->
    <group id="outro" dur="6s"
           transIn="transSlideWipeTop"
           transOut="transSlideWipeTop">
      <rectangle size="367 209">
        <material color="blue" filled="true"
                  transparency="0.9"/>
```

```
          </rectangle>
          <string textLines="&quot;Copyright© 2004&quot;;
                  &quot;Markus Brenner&quot;">
            <use xlink:href="#defIntroOutroFont"/>
            <use xlink:href="#defIntroOutroFontMat"/>
          </string>
        </group>
      </seq>
    </par>
  </body>
</XMT-O>
```

Anhang B: CD-ROM

Beiliegende[1] CD-ROM enthält eine Kopie dieser Diplomarbeit in elektronischer Form (Datei `Diplomarbeit - Einsatz von XMT und MPEG-4 zur Erstellung von RichMedia.pdf`) sowie den RichMedia-Prototyp aus Kapitel 3.1.

Der Anhang des Prototyps – zu finden in gleichnamigem Verzeichnis – ist in zwei Verzeichnisse aufgeteilt. Das Verzeichnis `Input` enthält alle zur Erstellung des Prototyps benötigten Dateien. Im Verzeichnis `Output` ist der fertige Prototyp (`MAN.mp4` beziehungsweise `MAN.m4x`) in drei verschiedenen Varianten abgelegt:

Im Verzeichnis `Lokal` befindet sich eine lokal abspielbare Variante des Prototyps. Ein Player ist mitgeliefert. Um den Prototyp über diesen Player abzuspielen, muss die Datei `M4Play.bat` gestartet werden.

`Internet (HTTP)` und `Streaming (RTP)` demonstrieren die Einbettung eines MPEG-4 Inhalts in eine Web-Seite. Beide Beispiele können nicht lokal, sondern nur unter Zuhilfenahme eines HTTP- beziehungsweise Streaming-Servers abgespielt werden.

In allen drei Beispielvarianten ist eine auf dem Betriebssystem installierte Java Virtual Machine (JVM) Voraussetzung.

1 auf der Innen-/Rückseite des Einbands

Glossar

A

Algorithmus Vorgang, der nach einem bestimmten (sich wiederholenden) Schema abläuft

B

Bandbreite Kapazität einer Netzwerkverbindung hinsichtlich der Übertragung von Daten – gemessen in Bit oder Byte pro Sekunde

BIFS Binäres Format zur Beschreibung von Szenen (Präsentationen)

Bit Kleinste Informationseinheit mit einem Wert von 0 oder 1

Bitrate Geschwindigkeit, mit welcher Bits über ein Netzwerk übertragen werden – gewöhnlich pro Sekunde

Broadcast Gleichzeitige Übertragung von Daten an ein gesamtes Netzwerk

Byte 8 Bits

C

CD Medium zur Speicherung von Daten

Client Nutzer-/Empfängerseite einer Software oder eines Computers

Clipping Überlappung, Ausschnitt,

Codec Technologie für das Komprimieren oder Dekomprimieren von Daten

D

Datenrate Anzahl der Informationen pro Sekunde

Decoder Anwendung oder Gerät zum Dekodieren (Dekomprimieren) von Daten

Distribution Verteilung

DivX Populäre Software zur Komprimierung von Video nach dem MPEG-4 (ASP) Standard

DRM Technologie zur Verwaltung von geistigem Eigentum

E

eLearning Vermittlung und Nutzung von Lerninhalten durch elektronische Medien

Encoder Anwendung oder Gerät zum Kodieren (Komprimieren) von Daten

F

Frame Einzelnes Bild in einem Film oder einer Sequenz aus Bildern

Framerate Anzahl der Frames pro Sekunde

Framework Rahmenwerk (= Umgebung mit definierten Schnittstellen)

H

Hardware Gesamtheit der technisch-physikalischen Teile einer Datenverarbeitungsanlage

HTML Sprache zur Herstellung von Internetseiten

HTTP Protokoll, das für das Browsen durch das World Wide Web im Internet benutzt wird

I

Internet Weltweiter Verbund von Computersystem in einem Netzwerk

Interoperabilität Austauschbarkeit, Fähigkeit verschiedener Systeme zusammenarbeiten und kommunizieren zu können

Intro Einleitung/Vorspann einer Präsentation

IP Verbindungsloses Protokoll zum Übertragen von Daten über ein Netzwerk (zum Beispiel des Internets)

IPMP Technologie zur Verwaltung und zum Schutz von geistigem Eigentum

ISO Organisation, die internationale Qualitäts- und Produktionsstandards entwickelt

J

Java Programmiersprache

K

Kompatibilität Vereinbarkeit/Verträglichkeit verschiedener Systeme

Kompilierung Prozess des Verknüpfens und Zusammenfügens von Daten oder Befehlen

L

Layout Anordnung

Level Stufe, Niveau, Ebene

Linking verbinden, verweisen

M

Metadaten Informationen, die Daten näher beschreiben

MIDI Standard zur Datenübertragung zwischen elektronischen Musikinstrumenten und dem Computer

MP3 Populäres Format zur Komprimierung von Musik

MPEG-4 International verabschiedeter Multimedia-Standard

Multicast Gleichzeitige Übertragung von Daten an mehrere bestimmte Teilnehmer in einem Netzwerk

Multimedia Gleichzeitige Anwendung von verschiedenen Medien (Text, Grafik, Audio, usw.) bei der Darstellung von Informationen

O

opaque undurchsichtig

Open Source Offenlegung von Programmquellen einer Software und gemeinsame Entwicklung an dieser über das Internet

Outro Abspann einer Präsentation

Overhead Zusätzlicher Verwaltungsaufwand oder Daten, die nicht zu den eigentlichen Nutzdaten gehören

P

Pixel Einzelner Punkt in einem Bild mit einem bestimmten Farb- und Helligkeitswert

Player Anwendung oder Gerät zum Abspielen von Inhalten

Protokoll Standard, nach dem sich die Kommunikation und der Datenaustausch zwischen verschiedenen Teilnehmern richtet

R

Rahmenwerk Umgebung mit definierten Schnittstellen

RichMedia Mit Multimedia und Interaktion "angereicherte" Medien

RTP Netzwerkprotokoll zur Übertragung von Multimedia-Daten in Echtzeit

RTSP Protokoll zur Steuerung von Multimedia-Inhalten

S

Samplerate Anzahl der Abtastungen eines Tons pro Sekunde

Server Software oder Computer, welche/r Dienste anbietet

SMIL Sprache zur Beschreibung von interaktiven 2D Multimedia-Präsentationen

Software Zum Betrieb einer Datenverarbeitungsanlage erforderliche nicht apparative Funktionsbestandteile (Einsatzanweisungen, Programme, usw.)

Streaming Übertragung von Daten in Echtzeit über ein Netzwerk – die Daten werden nach dem Empfang und der Verarbeitung meist sofort wieder verworfen

Syntax Korrekte Anordnung und Verknüpfung von Wörtern, Elementen oder Befehlen

T

Track Repräsentation eines einzelnen Titels/Inhalts in einer MP4-Datei

Transistion Übergang

U

Unicast Übertragung von Daten zu genau einem Teilnehmer innerhalb eines Netzwerkes

V

VRML Sprache zur Beschreibung von dreidimensionalen Objekten und Welten

X

XML Flexible und erweiterbare Sprache zur Beschreibung von textbasierten Inhalten

XMT Eine auf XML aufbauende Sprache zur Beschreibung von MPEG-4 Inhalten

Literatur- und Quellenverzeichnis

[3IVX] MPEG-4 Audio and Video Compression, http://www.3ivx.com

[ACODING] AudioCodingWiki, http://www.audiocoding.com/wiki/

[ATEST] Roberto's public listening tests page, http://www.rjamorim.com/test/index.html

[BITMT] Bitmanagement Software, http://www.bitmanagement.de

[BROENG] MPEG-4 AVC, http://broadcastengineering.com/ar/broadcasting_mpeg_avc/

[CSVT] Overview of the H.264 / AVC Video Coding Standard, http://www.dspr.com/www/products/video/compression/pci_encoder/csvt_overview.pdf

[CTECH] Coding Technologies, http://www.codingtechnologies.com

[DARWIN] Darwin Streaming Server, http://developer.apple.com/darwin/projects/streaming/

[DIVX] The Official Site of DivX Video, http://www.divx.com

[DOOM9] Doom9 Codec comparisons, http://www.doom9.org/codec-comparisons.htm

[ENVIVIO] Envivio, http://www.envivio.com

[FAAC] AudioCoding.com, http://www.audiocoding.com

[FLUI] Fluition, http://www.fluition.com

[GPAC] GPAC Project on Advanced Content, http://gpac.sourceforge.net/

[GRINS] GRiNS Pro Editor for SMIL 2.0, http://www.oratrix.com/Products/G2E/

[HARM] AVC+AAC The Next Generation of Compression, , 2003

[HEAAC] New MPEG-4 High-Efficiency AAC Audio, Dok. HE AAC White Paper, MPEG Industry Forum, April 2003

[IBMM4] IBM MPEG-4 Technologies, http://www.research.ibm.com/mpeg4/

[IMXET] IBM MPEG-4 XMT Authoring Tool, http://www.research.ibm.com/mpeg4/Projects/authoring.htm

[ISMA] Internet Streaming Media Alliance, http://www.isma.tv

[IVAST] iVAST, http://www.ivast.com

[JEFF] Kommunikation mit Jeff Boston über eMail und Forum, http://www.alphaworks.ibm.com/forum/tk4mpeg4.nsf/

[LAME] The LAME Project, http://lame.sourceforge.net/

[LSI] H.264/MPEG-4 AVC - Video Compression Tutorial, http://www.lsilogic.com/products/islands/h264/H.264_MPEG4_Tutorial.pdf

[MP4UI] mp4UI - MP4 file tool, http://mp4ui.sourceforge.net

[MPEG] The MPEG Home Page, http://www.chiariglione.org/mpeg/

[MPEG4-1] ISO/IEC 14496-1:2002. Coding of Audio-Visual Objects - Part 1: Systems, ISO/IEC JTC1/SC29/WG11, 2002

[MPEG4-2] ISO/IEC 14496-2:2001. Coding of Audio-Visual Objects - Part 2:

Visual, ISO/IEC JTC1/SC29/WG11, 2001
[MPEG4-3] ISO/IEC 14496-3:2001. Coding of Audio-Visual Objects - Part 3: Audio, ISO/IEC JTC1/SC29/WG11, 2001
[MPEG4-5] ISO/IEC 14496-5:2001. Coding of Audio-Visual Objects - Part 5: Reference Software, ISO/IEC JTC1/SC29/WG11, 2001
[MPEG4-6] ISO/IEC 14496-6:2000. Coding of Audio-Visual Objects - Part 6: Delivery Multimedia Integration Framework (DMIF), ISO/IEC JTC1/SC29/WG11, 2000
[MPEG4JS] Walsh, Aaron E./Bourges-Sévenier, Mikael, MPEG-4 Jump-Start, Prentice Hall PTR, 2002
[MPEGIF] MPEG-4 – The Media Standard, Dok. MPEG-4 White Paper, MPEG Industry Forum, November 2002
[MPEGLA] MPEG LA, http://www.mpegla.com/
[MPGABL] mpegable, http://www.mpegable.com
[N2724] ISO/MPEG N2724. MPEG-4 Applications, ISO/IEC JTC1/SC29/WG11, März 1999
[N3943] ISO/MPEG N3943. Intellectual Property Management and Protection in MPEG Standards, ISO/IEC JTC1/SC29/WG11, Januar 2001
[N4668] ISO/MPEG N4668. MPEG-4 Overview, ISO/IEC JTC1/SC29/WG11, März 2002
[N5231] ISO/MPEG N5231. MPEG-21 Overview, ISO/IEC JTC1/SC29/WG11, Oktober 2002
[N5525] ISO/MPEG N5525. MPEG-7 Overview, ISO/IEC JTC1/SC29/WG11, März 2003
[NERO] Nero Digital, http://www.nerodigital.com
[OCTAGA] Octaga, http://www.octaga.com
[OIPMP] OpenIPMP - Open-Source Rights Management, http://openipmp.com
[QSERVER] QuickTime Streaming Server, http://www.apple.com/quicktime/products/qtss/
[QUICKTIME] QuickTime Pro, http://www.apple.com/quicktime/upgrade/
[REAL] Helix Universal Server, http://www.realnetworks.com/products/server/
[RTP] RFC 1889 - RTP, http://www.ietf.org/rfc/rfc1889.txt
[RTSP] RFC 2326 - RTSP, www.ietf.org/rfc/rfc2326.txt
[SIG] Sigma Designs provides first Full-Resolution MPEG-4 Decoder, http://www.sigmadesigns.com/news/press_releases/011112.htm
[SMIL] Synchronized Multimedia Integration Language (SMIL) - Version 2.0, W3C, August 2001
[SVG] Scalable Vector Graphics (SVG) - Version 1.1, W3C, Januar 2003
[TMM] Henning, Peter A., Taschenbuch Multimedia (3. Auflage), Fachbuchverlag Leipzig, 2003

[TMPEG4B] Pereira, Fernando/Ebrahimi, Touradj, The MPEG-4 Book, Prentice Hall PTR, 2002

[VDUB] VirtualDub, http://www.virtualdub.org

[VEON] VeonStudio - The MPEG-4 Authoring Solution, http://www.software.philips.com/InformationCenter/Global/FArticleSummary.asp?lNodeId=771&channel=771&channelId=N771A2182

[VIA] Via Licensing Corporation, http://www.vialicensing.com/

[VRML] ISO/IEC 14772-1:1997 - Virtual Reality Modeling Language (VRML), Web3D Consortium, 1997

[X3D] ISO/IEC 19775:200x. Extensible 3D (X3D), Web3D Consortium, 2001

[XVID] Home of the XviD codec, http://www.xvid.org

Stichwortverzeichnis

Z