UNIVERSITÄT DER BUNDESWEHR MÜNCHEN
FAKULTÄT FÜR INFORMATIK
INSTITUT FÜR SOFTWARETECHNOLOGIE

Langzeitarchivierung bei Medienservern

Entwicklung von Strategien zur Langzeitarchivierung am Beispiel eines Medienservers

Arne Seifert
24.03.2010

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigten Dissertation.

Vorsitzender	Prof. Dr. Ulf Schmerl
Berichterstatter	Prof. Dr. Uwe M. Borghoff (Betreuer)
	Prof. Dr. Johann Schlichter
Prüfer	Prof. Dr. Ulrike Lechner
	Prof. Dr. Gunnar Teege
Tag der mündlichen Prüfung	02.07.2010

Zusammenfassung

Kurzfassung

Der Begriff „Digitales Vergessen“ taucht immer häufiger in den Medien auf. Aktuell werden in großangelegten Projekten mit enormen finanziellen Mitteln Daten digitalisiert, um sie für die Zukunft einfach zugreifbar zu erhalten. Die Originale lagern dabei in sicheren Archiven. Viele bedeutende Arbeiten liegen aber inzwischen nur noch in digitaler Form vor, so dass ein Verlust des Digitalisats gleichbedeutend mit dem Komplettverlust der Information ist.

Eine Vielzahl an Projekten beschäftigt sich mit der Verwaltung digitaler Objekte und Medien im Zusammenhang mit dem einfachen Zugriff auf die Daten, beispielsweise über das Internet. Dabei gibt es beim Übergang einer Sammlung in ein Archiv in vielen Bereichen Probleme, die aufgrund der noch relativ neuen Problemstellung teilweise unzureichend gelöst sind. Betrachtet man die Erhaltung von Metadaten zu Digitalisaten, so gibt es oftmals nur vage Informationen über geeignete Prozesse und Techniken, diese Daten einem langfristigen Erhalt zuzuführen.

In dieser Arbeit werden die Defizite bekannter Medienserversoftware in Bezug auf den langfristigen Erhalt der Daten aufgezeigt. Anhand der gefundenen Problemstellen wird ein Verfahren entwickelt, das sich nahtlos in einen Medienserver integrieren lässt und Softwareunterstützung für die Langzeitarchivierung großer Text- und Bildbestände liefert. Besonderes Augenmerk wird dabei auf den Erhalt der Metadaten zu den Digitalisaten gelegt und die erforderlichen Methoden werden dargestellt und bewertet. Zusätzlich soll Hilfestellung bei der Auswahl geeigneter Dateiformate für die Archivierung gegeben werden.

Abstract

The concept of "digital oblivion" appears more often in the media. A number of large-scale projects aim to digitalize data and make it accessible in the future. The originals are stored in save archives. Even today, many important works only remain in digital form. So loss of the digital data is equivalent to the loss of complete information stored for this object.

Many projects aimed at managing digital objects and media in combination with easy access to data, for example via the internet. However, there are many problematic issues in transferring a collection into an archive for long-term preservation. As these arising issues are new, many are not yet fully solved adequate. For instance, there is only vague information available about processes and techniques of preservation of meta information related to digitalized data.

In this work the deficits of well-known media-server software are pointed out regarding to the long-term preservation of stored data. Basing on the found problems a procedure is developed, which can be smoothly integrated into media-server software to support the long-term preservation of large text and picture collections. Special attention is put thereby on the receipt of the meta data of the digital object. Necessary methods are represented and evaluated. Additionally recommendations are given to the selection of ideal file formats for long-term preservation.

Inhalt

1 Einleitung

Im ersten Kapitel wird die Motivation dargelegt und ein kurzer Überblick gegeben, warum es notwendig ist, im Bereich der Metadatenspeicherung für große Bild- und Publikationsarchive eine neue Methode einzuführen beziehungsweise die Erweiterung bestehender Methoden voranzutreiben.

1.1 Ausgangssituation

Die Produktion digitaler Inhalte in den verschiedensten Formaten hat in der letzten Zeit enorm zugenommen. Leider geht damit auch die Erweiterung um eine Vielzahl an unterschiedlichen und teilweise auch inkompatiblen, proprietären Dateiformaten einher. Die rasante Verbreitung von Digitalkameras und Mobiltelefonen beschleunigen diesen Prozess zusätzlich, da letztere inzwischen nahezu alle immer auch mit Kameras ausgestattet sind. Leider ist es um die Haltbarkeit so erzeugter digitaler Medien nicht immer zum Besten bestellt. Neben den digitalen Medien existieren die Informationen nicht zusätzlich in anderen (analogen) Formaten, deshalb ist der Erhalt der Dateien von besonderer Wichtigkeit.

Nicht nur der Bildbereich erfährt eine Digitalisierung der Informationen, heutzutage stehen immer weniger Datenmaterialien analog zur Verfügung. Hintergrund ist die enorme Verbreitungsgeschwindigkeit und eine immer einfachere Bedienung komplexer Computersysteme. Daraus resultiert oftmals die Umstellung bisheriger Prozesse von analogen hin zu digitalen Formen, aus dem einfachen Brief wird beispielsweise die E-Mail. Mit dieser Umstellung entstehen viele neue Probleme, für die erst geeignete Lösungen gefunden werden müssen.

Neben dem Erhalt der Informationen ist der Zugriff auf die Informationen entscheidend. Die Entwicklung der vergangenen Jahre zeigt, dass vor allem durch den Ausbau schneller Datennetze der Zugriff auf Informationen aus nahezu allen Lebenslagen erwartet wird. Daten stehen universell und jederzeit zur Verfügung, die Verbreitungswege sind dabei vielfältig. Inzwischen reicht es nicht mehr aus, dass Informationen an einer Stelle abgerufen werden können, sie müssen jederzeit parat sein. Aus diesen Gegebenheiten und Forderungen der Anwender hat sich die Situation entwickelt, dass das Internet als zentrale Anlaufstelle für Informationen aller Art genutzt wird. Der Zugriff auf das Internet ist inzwischen von unterschiedlichsten Stellen möglich, vom stationären Computer bis hin zum Mobiltelefon.

Zur Verwaltung und vor allem zur sicheren Aufbewahrung von diesen sehr großen Datenmengen sind Systeme notwendig, die unter Beachtung aktueller Erkenntnisse und im Hinblick auf eine langfristige Verfügbarkeit der Daten die Koordination übernehmen. Im Augenblick steckt dieser Bereich noch in den Anfängen und es werden eine ganze Reihe unterschiedlicher Ansätze verfolgt. Das beginnt bereits mit dem kostenlosen Internetalbum, in das eine Privatperson die eigenen Schnappschüsse einstellen kann. Andererseits bieten kommerzielle Anbieter Software an, die dem langfristigen Erhalt Rechnung tragen sollen. Ein weiterer Bereich sind die Archive, die einen großen Datenbestand meist nicht unbedingt digital vorhalten. Um solche Informationen leichter verbreiten zu können und zugleich die oftmals wertvollen Originale zu schützen, werden immer häufiger Sammlungen unter großem finanziellem und personellem Aufwand retrodigitalisiert. Hier stellt sich die Frage nach einer geeigneten Plattform ebenso wie die nach einem geeigneten Dateiformat. Sinnvoll ist eine Digitalisierung sicherlich erst dann, wenn der Zugriff dauerhaft gewährleistet und damit das kostbare Original geschont werden kann. Die

Verwaltung großer Datenmengen verlangt zusätzliche Angaben zu den verwalteten Objekten, um das schnelle Auffinden der Informationen innerhalb großer Datenmengen zu ermöglichen.

1.2 Abgrenzung

Die nachhaltige Bewahrung von Wissen und Kulturgütern steht im Zentrum vieler Forschungsvorhaben. Daraus hat sich eine Vielzahl wissenschaftlicher Projekte entwickelt, in deren Kern die Langzeitarchivierung digitaler Objekte mit gleichzeitiger Ausweitung der Verfügbarkeit steht. Aus diesen Projekten gehen in der Regel sowohl konzeptionelle als auch softwareseitige Ergebnisse hervor. Diese decken aber meist nur den Grundbedarf an Funktionalität ab oder fordern ganz bestimmte Rahmenbedingungen und bieten daher nur Lösungen von Teilproblemen. Alternativ wird ein Bereich der Thematik um Routinen für spezielle Anforderungen erweitert. Eine Verallgemeinerung der bisher bekannten Verfahren scheitert oft an den Bedingungen, die das Verfahren voraussetzt, um zum Einsatz zu kommen.

Der im Folgenden vorgestellte Ansatz soll die Software eines Medienservers um die zusätzliche Funktionalität der Langzeitarchivierung erweitern. Besonderes Augenmerk wird dabei neben der Flexibilität auf den Erhalt aller Objektinformationen zu den digitalen Daten gelegt. Das gezeigte Verfahren ist dabei so ausgelegt, dass es sich nicht ausschließlich in einer bestimmten Medienserversoftware einsetzen lässt, sondern sich vielmehr als Konzeptgrundlage für die Integration in eine beliebige Software eignet. Aus der parallelen praktischen Umsetzung des Archivierungskonzepts neben der rein theoretischen Entwicklung resultiert ein flexibles Verfahren, das auch in der Praxis Anwendung finden kann.

Durch das noch relativ junge Themenfeld der nachhaltigen Archivierung digitaler Daten fehlen in manchen Bereichen entsprechende Erfahrungen. Erst aus dem produktiven Einsatz oder nach Testläufen zeigen sich Anforderungen, die berücksichtigt werden müssen. Das entwickelte Verfahren minimiert die Vorbedingungen an Software und Hardware und bietet an unterschiedlichen Stellen Erweiterungsmöglichkeiten zur einfachen Integration in bestehende Medienserveranwendungen.

1.3 Überblick über diese Arbeit

Das folgende Kapitel beschreibt die Grundlagen und Anforderungen, die an einen Medienserver gestellt werden. Bestehende Medienserversoftware aus dem Open-Source-Bereich und kommerzieller Anbieter werden betrachtet und dabei Defizite sowohl in der Metadatenverwaltung als auch der Langzeitarchivierung aufgezeigt und gegenübergestellt. Abschließend erfolgt eine kurze Einführung in die Medienserversoftware mediatum, die im weiteren Verlauf für die Testimplementierung herangezogen wird.

Kapitel 3 spezifiziert den Begriff „Metadaten“ und gliedert gängige Metadatenformate in Kategorien ihres Vorkommens und benennt das jeweilige Einsatzgebiet. Zusätzlich werden Besonderheiten der Metadaten bei Medienservern aufgezeigt.

In Kapitel 4 werden unterschiedliche Dateiformate aus dem Umfeld des Medienservers kategorisiert, näher beschrieben und die Eigenheiten herausgearbeitet. Die eingeführte Kategorisierung wird auch für die weitere Betrachtung herangezogen. Abschließend erfolgen eine Gegenüberstellung der Formate hinsichtlich ihrer Eignung beim Einsatz auf Medienservern sowie eine Bewertung unter Kriterien der Nachhaltigkeit und der Langzeitarchivierung.

Den Themenkomplex der Langzeitarchivierung (LZA) erläutert Kapitel 5 näher und zeigt besondere Anforderungen der LZA an digitale Daten auf einem Medienserver. Die Komponenten des OAIS-

Referenzmodells werden eingeführt und auf die Verwendung innerhalb eines Medienservers übertragen. Auf Basis dieser Voraussetzungen erfolgt die Entwicklung eines Prozesses der LZA bei Medienservern. Die erforderlichen Grundüberlegungen und speziellen Anforderungen an ein flexibles Verfahren werden benannt.

Kapitel 6 beschreibt die Entwicklung von einzelnen Komponenten der LZA mit der Anbindung der Funktionalität an einen Medienserver. Dabei werden die Grundvoraussetzungen aus den unterschiedlichen Dateikategorien aufgegriffen und daraus die unterschiedlichen Szenarien konstruiert, die der Archivierungsprozess umfasst.

Die Algorithmen der Dateiverarbeitung und die Erzeugung der Langzeit-Objekte aus den Originaldateien werden in Kapitel 7 dargelegt. Die Erzeugung der erforderlichen Metadaten und die Möglichkeiten der Einbettung werden beschrieben. Zeitkritische Verarbeitungsschritte werden anhand ausgewählter Testdaten analysiert und hinsichtlich der Umsetzbarkeit eingeordnet. Abschließend erfolgen die Analyse der gefundenen Problemstellen des dargestellten Archivierungsverfahrens und die Darstellung weiterer Archivierungsstrategien.

Kapitel 8 fasst die Arbeit zusammen und liefert einen Ausblick über Langzeitarchivierung bei Medienservern.

2 Grundlagen und Szenario – Stand der Dinge

Dieses Kapitel definiert den Begriff des Medienservers und beschreibt Anforderungen, die an einen Medienserver gestellt werden. Es werden existierende Systeme anhand der Anforderungen betrachtet, bewertet und Defizite herausgearbeitet. Zusätzlich wird der Kontext eingeführt, der im Folgenden als Beispiel dienen soll.

2.1 Medienserver

Ein Medienserver ist eine Software, die große Objektmengen unterschiedlicher Herkunft sicher verwalten kann. Die Objekte oder Medien werden mit Metadaten angereichert und kategorisiert gespeichert. Basistypen der Medien bilden Texte oder Bilder, weitere Typen sind aber denkbar. Damit stellt die Forderung nach einem erheblichen Maß an Flexibilität hohe Ansprüche an die Software. Dies liegt auch an dem heterogenen Umfeld, in dem Medienserver generell zum Einsatz kommen.

Zur reinen Verwaltung der Medien kommen noch die Bereiche des Austauschs, Gewährleistung des Zugriffs und der nachhaltigen Sicherung der Informationen. Der Datenaustausch kann mithilfe standardisierter Schnittstellen erfolgen, die auf die jeweiligen Bedürfnisse angepasst werden. Hierfür gibt es individuell angepasste oder speziell entwickelte Protokolle. Die nachhaltige Speicherung der Informationen kann einerseits durch den Betreiber realisiert werden, andererseits können Archivierungssysteme zum Einsatz kommen. Generell muss aber eine klare Unterscheidung zwischen Medienserver und Archivsystem erfolgen. Erst diese Trennung ermöglicht spezialisierte Systeme, die unterstützend eingreifen können.

Medienserver stellen eine Erweiterung zu klassischen Dokumentenservern oder Publikationsservern dar. Während Dokumentenserver als digitale Bibliothek zur Veröffentlichung digitaler Publikationen dienen, können Medienserver mit weiteren Medientypen umgehen und bilden damit eine Obermenge zu Dokumentenservern. Klassische Aufgaben der Dokumentenserver sind:

- Erschließung der Publikationen,
- Bereitstellung der Informationen und
- Archivierung der Daten.

Folgende Themenfelder sind zu betrachten, wenn man die zuvor genannten Aufgaben als Basis weiterer Betrachtungen heranzieht. Dabei werden insbesondere Themen betrachtet, die in direktem Zusammenhang zwischen Medienserver und nachhaltiger Speicherung stehen:

- Langzeitarchivierung,
- Evaluation geeigneter Dateitypen,
- Betrachtung gängiger Metadatenformate und
- Umsetzung für Medienserversoftware.

2.2 Anforderungen an einen Medienserver

Anforderungen an einen Medienserver lassen sich auf zwei Ebenen stellen. Die Basisanforderungen sind zwingend erforderlich, die erweiterten Anforderungen erhöhen entweder die Flexibilität oder den Komfort.

2.2.1 Minimalanforderungen

Für eine spätere Evaluation geeigneter Softwaresysteme werden folgende Minimalanforderungen als Kriterienkatalog definiert:

Objektverwaltung
Eine flexible Objektverwaltung muss das Einspielen und die Wartung der Objekte auf dem Server übernehmen. Metadateninformationen müssen den Objekten konfigurierbar zuzuordnen sein und mittels Kontrollmechanismen den Benutzer bei der Pflege der Daten aktiv unterstützen. Es müssen Anzeigeformen unterschiedlichen Detailierungsgrades vorhanden sein. Eine Hierarchiestruktur zur Gliederung der Objekte ist erforderlich und Objekte können an mehreren Stellen in dieser Struktur erscheinen.

Suchmaschine
Suchfunktionen innerhalb des Datenbestands sollen eine einfache Suche über alle Metadaten inklusive der Volltexte (falls vorhanden) ermöglichen. Zusätzlich muss der Benutzer über konfigurierbare Kriterien Expertensuchen durchführen können, auch die Kombination mehrerer Kriterien soll möglich sein. Dabei sind zur Unterstützung des Nutzers nach Möglichkeit Indexlisten oder Wörterbücher anzubieten. Wichtigstes Kriterium bei der Suche bleibt die kurze Antwortzeit der Suchmaschine auch bei umfangreichem Datenbestand.

Rechtemanagement
Ein flexibles, feingranulares Rechtemanagement muss gewährleisten, dass unterschiedliche Benutzer in abgeschlossenen Bereichen Daten verwalten können. Hierfür sind Lese-, Schreibrechte, sowie eventuell zusätzliche Rechte erforderlich. Nur auf diese Weise kann auch dem Urheberrecht Rechnung getragen werden.

Nachhaltigkeit
Die Nachhaltigkeit und der Erhalt der durch den Medienserver verwalteten Daten muss sichergestellt werden. Entsprechende Vorkehrungen oder Schnittstellen müssen vorhanden sein, entweder innerhalb als eigene Komponente oder als Schnittstelle zu externen Systemen. Der komplette Lebenszyklus eines Objekts auf dem Medienserver sollte dabei abgedeckt werden.

Workflow
Workflows müssen Abläufe steuern können. Änderungen an diesen Abläufen müssen sich auch im Workflow realisieren lassen. Die Datenerfassung muss über Workflows erfolgen können, um Kontrollmechanismen zur Qualitätssicherung zu ermöglichen.

2.2.2 Erweiterte Anforderungen

Wünschenswerte Funktionalitäten werden unter den erweiterten Anforderungen aufgeführt:

Erweiterbarkeit
Über Erweiterungsschnittstellen können zusätzliche Funktionalitäten angebunden werden. Diese Möglichkeit sollte auf unterschiedlichen Ebenen offen sein, um flexibel Anpassungen an persönliche Gegebenheiten zu erlauben. Auf unterster Ebene sind das Erweiterungen um zusätzliche Datentypen.

Anpassbarkeit
Die Anpassbarkeit des Medienservers sollte das System nahtlos in die vorhandene Infrastruktur eingliedern lassen. Dabei sind sowohl Anpassungen auf der Präsentationsebene, als auch auf darunterliegenden Ebenen wünschenswert. Bereits vorhandene Objektsammlungen müssen in das System integrierbar sein.

Benutzerführung
Funktionen des Medienservers sollten den Nutzer über die Oberflächen dergestalt führen, dass weitreichende Konfigurationen ermöglicht werden. Die Benutzerführung soll Fehleingaben oder –konfigurationen möglichst ausschließen. Die Benutzerführung sollte an das Rechtesystem angepasst sein.

Datenaustausch
Schnittstellen zu anderen Programmen sollen entweder bereits vorhanden sein oder über Erweiterungen den Datenaustausch ermöglichen. Dabei sollte sowohl der Export als auch der Import möglich sein und standardisierte Formate unterstützt werden.

2.3 Betrachtung existierender Systeme

Unter den bereits genannten Kriterien [**14**] sind 2005 einige Softwareprodukte auf ihren Funktionsumfang hin bewertet worden [**12**] [**27**]. Darunter fallen Produkte aus dem Open-Source-Bereich, aber auch kommerzielle. Gerade bei den kommerziellen Medienservern ist die Betrachtung relativ schwer, da der Funktionsumfang nur aus den Beschreibungen der Hersteller abgeleitet werden kann. Testinstallationen sind entweder sehr aufwändig oder aufgrund des Lizensierungsmodells des Herstellers nur schwer möglich.

2.3.1 MyCoRe

Zum Zeitpunkt der Betrachtung dient MyCoRe [**59**] in erster Linie für die Verwaltung von textuellen Datentypen als Publikationsserver. Dabei werden die Daten mit dem datenbankgestützten System verwaltet und mit Metadaten angereichert. Als internes Datenformat kommt XML zum Einsatz, das über verschiedene Parser und Transformationen verarbeitet wird. Eine Web-Oberfläche wird ebenso erzeugt. Als Programmiersprache dient dabei Java zusammen mit einigen Open-Source-Komponenten, die beispielsweise die Suche oder Speicherung betreffen (z.B. MySQL als Datenbanksystem). MyCoRe kann sowohl als reine Open-Source-Software eingesetzt werden oder zusammen mit kommerziellen Komponenten, die Basissoftware liegt quelloffen vor. Entwickelt wird MyCoRe von einer Community bestehend aus Mitgliedern mehrerer deutscher Universitäten [**53**].

Die Verwaltung der Objekte umfasst den kompletten Lebenszyklus der Daten. Dabei ist allerdings die Flexibilität bei Änderungen an der Definition der Metadaten eingeschränkt. Definitionen können nur auf dem Server angepasst werden, indem XML-Dateien direkt editiert werden. Eine Suchmaschine, die über konfigurierbare Masken weitreichende Funktionen bietet, ist integriert. Es können auch Suchen über mehrere MyCoRe Installationen hinweg angestoßen werden. Das Rechtemanagement unterscheidet zwischen lesendem und schreibendem Zugriff und kann auf diese Art die Objekte unterschiedlichen Benutzergruppen zugänglich machen. Der Bereich der Archivierung beschränkt sich bei MyCoRe darauf, die verwalteten Daten nativ zu sicher. Die Datenbank mit den Metadaten und die Dateien auf der Festplatte können gesichert werden. Der Sicherungsprozess muss allerdings durch den Betreiber des Systems angestoßen werden, MyCoRe unterstützt hierfür keinerlei automatische Mechanismen. Eine Workflowunterstützung, die rudimentäre Operationen ermöglicht, ist integriert.

Die Erweiterbarkeit ist durch die Quelloffenheit jeder Zeit gegeben, allerdings ist an dieser Stelle Eigenentwicklung notwendig. Die Benutzerführung ist an vielen Stellen nur sehr eingeschränkt vorhanden, oftmals ist das direkte Editieren der Konfigurationsdateien erforderlich. Für den Export ist OAI [**65**] als Austauschschnittstelle vorhanden.

Vor allem durch die starre Struktur der Metadaten und unterstützten Dateitypen, ist MyCoRe an einigen Stellen unflexibel. Zusätzlich ist die Anzahl unterstützter Dateiformate (noch) relativ gering. Das liegt allerdings in erster Linie an der Ausrichtung des Systems als Dokumenten- und Publikationsserver. Zusätzliche Erweiterungen und neuere Versionen wurden entwickelt, um weitere Anwendungsfelder zu erschließen. Die Unterstützung für Bildformate befindet sich gerade im Aufbau.

2.3.2 EPrints

Die Open-Source-Software EPrints [**89**] bietet in erster Linie eine Softwarelösung für Open Access Repositorien zur Verwaltung von wissenschaftlichen Publikationen. Neuere Versionen sind außerdem um die Unterstützung weiterer Dateiformate erweitert worden, aber der Bereich der grafischen Objekte ist noch nicht zufriedenstellend.

Bei der Betrachtung der Entwicklung ist deutlich erkennbar, dass der Funktionsumfang aufgrund bestimmter Nutzeranforderungen erweitert wird. Somit ist es von den Benutzern abhängig, in welcher Richtung die Weiterentwicklung vorangetrieben wird. Eine Einflussnahme ist durch die Teilnahme an der Community des EPrint-Projekts möglich.

Softwareseitig handelt es sich um eine Pearl-basierte Anwendung, die mithilfe des Apache-Webservers die Daten über eine Web-Schnittstelle anbietet. Über zusätzliche Open-Source-Produkte erfolgt die Extraktion der Volltextinformationen. Image-Bibliotheken werden eingesetzt, um Thumbnails oder Vorschaubilder zu erstellen. Zur persistenten Datenablage kommt die Datenbank MySQL zum Einsatz. Verfügbar ist EPrints für unterschiedliche Betriebssysteme, darunter vorgefertigte Pakete für Linux und Windows.

Die mächtige Objektverwaltung unterstützt den Nutzer während der Datenverarbeitung und bietet unter anderem eine Ähnlichkeitsanalyse, um doppelte Objekte zu vermeiden. Objekte können aus unterschiedlichen Dateien oder Unterobjekten aufgebaut sein. Die integrierte Suchmaschine erlaubt sowohl die einfache Suche über alle Daten, als auch eine nach Kriterien einschränkende Expertensuche. Die Suchtreffer können in vielen Exportformaten dargestellt werden und sind damit für die Nutzung in anderen Programmen geeignet. Über ein Rechtemanagement ist die Zugriffssteuerung möglich, die verschiedene Kriterien erlaubt, darunter auch eine Sperrfrist. Flexible Workflows bieten die Grundfunktionen zur Erstellung der Objekte und werden über Konfigurationsdateien gepflegt. Für jedes Objekt werden in einer Historie der Lebenszyklus und Änderungen festgehalten. Eine aktive Unterstützung der Archivierung erfolgt nicht, aber der Datenaustausch kann über standardisierte Austauschformate erfolgen, somit ist auch die Anbindung an externe Systeme möglich.

2.3.3 OPUS

Der Online Publikationsverbund der Universität Stuttgart (OPUS) [**87**] wurde 1998 von der Universität Stuttgart mit der Unterstützung des DFN-Vereins gegründet [**78**]. Ziel war die Schaffung eines Systems, das einen Austausch der elektronischen Quellen der teilnehmenden Partner ermöglicht. Dabei wurde in erster Linie der Austausch elektronischer Prüfungsarbeiten ausgearbeitet. Erst in den letzten Jahren erfolgte durch die kooperative Weiterentwicklung der teilnehmenden Universitäten der Ausbau auch in Richtung Bildobjekte. Aktuell nehmen etwa 70 Universitäten in Deutschland an diesem Softwareverbund teil. Die daraus entstandene Software ist unter der GNU General Public License als Open-Source-Projekt veröffentlicht.

Die Objektverwaltung umfasst die Metadatenvergabe und –pflege. Als Basis dient das Dublin-Core-Metadatenschema. Die Daten werden an Verbundsysteme weitergereicht, dort ergänzt und gegebenenfalls korrigiert. Zentraler Bestandteil von OPUS ist die Such- und Recherchemöglichkeit innerhalb des eigenen Bestands, aber auch in den Beständen anderer Einrichtung, die ebenfalls Opus einsetzen. Ziel dieser Funktion ist die Schaffung eines Netzwerks zur besseren Verfügbarkeit der wissenschaftlichen Arbeiten. Die Suche im Volltext ist auch möglich. Das Rechtemanagement unterstützt Sperrfristen, um auch dem Urheberrecht gerecht zu werden. Ein Modul zur Definition von Nutzerlizenzen erlaubt Autoren unterschiedliche Lizenzmodelle je Objekt zu vergeben. Eindeutige persistente Identifikationen werden in Form des Uniform Resouce Name (URN) (siehe 3.8.2) erzeugt. Die Identifikatoren werden für die nachhaltige Speicherung und Erhaltung des Zugriffs auf die Daten eingesetzt. Anpassbare Workflows fehlen, lediglich der Prozess der Datenerzeugung wird aktiv unterstützt. Die Umsetzung erfolgte in Anlehnung an den Publikationsprozess innerhalb Bibliotheken.

Durch die quelloffene Applikation (Programmiersprache PHP mit Verbindung zu einer relationalen Datenbank) ist die Erweiterungsmöglichkeit gegeben, auch Anpassungen werden damit möglich. Der Datenaustausch kann über standardisierte Schnittstellen erfolgen, beispielsweise OAI zur Datenübertragung in Katalogsysteme.

2.3.4 DSpace

Die erste Version der Open-Source-Lösung DSpace [**55**] wurde 2002 als Gemeinschaftsentwicklung des Massachusetts Institute of Technology (MIT) und den HP Labs veröffentlicht und steht unter der BSD Lizenz [**81**] [**54**]. Die Software bietet Funktionen zum Betrieb eines Medienservers und stellt Werkzeuge zur Verarbeitung, Erfassung und Speicherung digitaler Medien zur Verfügung. Eine mächtige Suchmaschine ist integriert. Zum Einsatz kommt DSpace meist an Universitäten, Forschungseinrichtungen oder Bibliotheken und wird dort als Institutional Repository (IR) [**88**] eingesetzt.

Realisiert ist DSpace in der Programmiersprache Java mit JSP zur Auslieferung der Inhalte über eine Web-Schnittstelle. Die Daten werden in einer relationalen Datenbank vorgehalten. Als Webserver kommt Apache mit Tomcat zum Einsatz. Aufgrund der verwendeten Komponenten eignet sich als Betriebssystem in erster Linie Linux. Die Software selber ist in unterschiedlichen Schichten realisiert, so dass eine klare Trennung und damit auch der Austausch einzelner Komponenten ermöglicht werden. Verwendet wird dabei das 3-Schicht Modell (Application, Business Logic und Storage Layer).

Durch die Entwicklung in Anlehnung an das OAIS-Referenzmodell kann die Software auch für den dauerhaften Erhalt und die Langzeitverfügbarkeit digitaler Ressourcen zum Einsatz kommen. Die Objektverwaltung übernimmt die umfassende Verarbeitung der Informationen und organisiert Objekte auf unterschiedlichen hierarchischen Ebenen. Diese Ebenen spiegeln die Struktur der Organisation wider, von der Organisation auf oberster Ebene bis hin zu Containerobjekten, die auf unterster Ebene die Dateien mit Metadaten enthalten. Die mitgelieferte Suchmaschine (Lucene) erlaubt einfache Suchen über alle Daten und zusätzlich konfigurierbare Expertensuchen mit kombinierbaren Einschränkungskriterien. Ein Rechte- und Rollenkonzept ist integriert, das Benutzer in Gruppen organisiert. In Anhängigkeit der Rollen werden Aktionen auf den Objekten realisiert. Neben der Suche ist eine Browsingfunktion vorhanden.

DSpace klassifiziert Dateiformate anhand des Unterstützungsgrades. Drei Grade kommen zum Tragen unterstützt, bekannt oder unbekannt. Je nach Grad ist die Nachhaltigkeit besser oder schlechter gegeben. Unbekannte Formate werden keiner weiteren Verarbeitung unterzogen und der dauerhafte Zugriff kann nicht durch die Software sichergestellt werden. Unterstützte Formate sind besonders für die Langzeitverfügbarkeit geeignet. Die aktuelle Version nutzt Dublin Core als Metadatenschema,

Erweiterungen sind in Arbeit. Standardisierte Schnittstellen bieten Datenimport und –export und können im Batch-Betrieb genutzt werden.

2.3.5 Kommerzielle Lösungen

Der Zugang zu kommerziellen Systemen für Vergleichszwecke ist aufgrund der anfallenden Kosten oder des Fehlens geeigneter Beispielversionen schwierig. Lediglich die Herstellerangaben können herangezogen werden. DigiTool, eingesetzt im Bibliotheksverbund Bayern (BVB), und die Software DIAS.

DigiTool [**32**] gehört zur Familie der Digital Asset Management Systeme und bietet Unterstützung beim Aufbau digitaler Bibliotheken. Die Anbindung an Bibliothekssysteme ist gegeben. DigiTool kann mit unterschiedlichen Dateiformaten umgehen, darunter Text-. Bild- und auch Multimediaformate, die per Streaming angeboten werden. Differenzierte Such- und Recherchemöglichkeiten ermöglichen den Zugriff auf den Datenbestand. Ein workflowgestützter Datenaustausch für Import und Export ist integriert, Archivierungssysteme können angeschlossen werden.

DIAS (Digital Information Archiving System) [**18**] geht konform mit dem OAIS-Modell und bietet eine Datenablage für digitale Objekte. Zusätzlich zur Ablage mit Zugriffsmöglichkeiten über Schnittstellen (Web, eigene Anwendung) sind Maßnahmen zur Langzeitarchivierung getroffen. Migrationsprozesse können automatisiert angestoßen werden. Als Metadatenschema kommt METS (siehe 3.5.2) zum Einsatz.

2.4 Ergebnis der Betrachtungen

Kommerzielle Medienserverlösungen decken die betrachteten Kriterien zufriedenstellend bis sehr gut ab. Der einzige große Nachteil bleibt die unflexible Erweiterbarkeit. Die Einflussnahme auf kommende Entwicklungen oder angeschlossene Eigenentwicklungen sind aufgrund des Lizenzmodells und der fehlenden Quelloffenheit nicht möglich. Die Realisierung von Spezialanforderungen muss durch den Hersteller der Software erfolgen. Zusätzlich fallen bei den vorgestellten Produkten erhebliche Kosten für die Software an.

Mit Ausnahme von DSpace gehören die betrachteten Medienserver aus dem Open-Source-Bereich in die Kategorie der Dokumentenserver. MyCoRe, EPrints und Opus werden in den aktuellen Versionen um zusätzliche Funktionen ergänzt und so zu Medienservern ausgebaut. Das spiegelt sich auch im Unterstützungsgrad der Systeme für elektronische Publikationen wider. Die Unterstützung für textbasierte Daten ist durchweg gut, weitere Medien werden teilweise nur rudimentär unterstützt.

	MyCoRe	**EPrints**	**Opus**	**DSpace**
Objektverwaltung	**+**	**++**	**+**	**++**
Metadatenschemata	**+**	**-**	**-**	**+**
Suchmaschine	**++**	**+**	**++**	**+**
Rechtemanagement	**+**	**+**	**+**	**++**
Workflow	**+**	**+**	**-**	**+**
Datenaustausch	**+**	**++**	**+**	**++**
Erweiterbarkeit	**+**	**+**	**+**	**+**
Langzeitarchivierung	**--**	**-**	**-**	**+**

Tabelle 2-1: Vergleich der Open-Source-Medienserver

Die Gegenüberstellung der Systeme aus dem Open-Source Bereich in Tabelle 2-1 bewertet die betrachteten Systeme hinsichtlich der untersuchten Kriterien. Die Bewertung erfolgt in Stärken (+) und Schwächen (-).

Auffällig in der Gegenüberstellung sind die Kriterien der Metadatenschemata und der Langzeitarchivierung, die unterdurchschnittlich gelöst sind.

2.4.1 Metadatenschemata

Die Systeme nutzen ein Metadatenschema, das aus einem Basissatz an Feldern besteht. Lediglich MyCoRe bietet zumindest die Erweiterungsmöglichkeit um zusätzliche Felder. Bei den anderen Medienservern handelt es sich um ein starres Schema mit fest vorgegebenen Feldern.

Betrachtet man die unterschiedlichen Medientypen, dann ist ein starres Schema eine erhebliche Einschränkung. Verglichen mit elektronischen Publikationen benötigen vor allem Bildarchive weitere Attribute, da eine Beurteilung unter anderen Gesichtspunkten erfolgt. Bildinformationen sind nicht vollständig automatisch interpretierbar und damit auf Annotationen angewiesen. Weitere Medien aus dem Multimediaumfeld erfordern noch weitreichendere Maßnahmen. Gerade die Größe dieser Medien und die Vielfalt der Ausprägungen bringen neue Probleme.

2.4.2 Langzeitarchivierung

Der nachhaltige Erhalt der Informationen verbunden mit der Langzeitarchivierung bietet im Vergleich der Kriterien die schlechteste Unterstützung durch die Software. In DSpace ist eine grundlegende Einteilung in Eignungsklassen vorhanden, die anderen Systeme behandeln den Aspekt der Langzeitverfügbarkeit nicht explizit. Keines der Systeme kann direkt an Archivierungssoftware angeschlossen werden oder ist dafür vorgesehen. Der Betreiber muss in jedem Fall selbst dafür Sorge tragen, dass die Metadaten und die dazugehörigen Dateien gesichert werden.

2.5 mediaTUM – Medienserver der TUM

Die Universitätsbibliothek der Technischen Universität München (TUM) ist seit 2004 Teilnehmer am DFG-Projekt IntegraTUM. Ziel des auf fünf Jahre angelegten Projekts ist die Verbesserung der Dienstleistungen durch die Überarbeitung organisatorischer Abläufe und technischer Maßnahmen. Als Ergebnis sollen dabei eine stärkere Vernetzung bestehender Angebote und die Erweiterung um neue Angebote realisiert werden, um eine einheitliche, nahtlose und benutzerfreundliche Infrastruktur für die Bereiche Information und Kommunikation zu schaffen [**41**].

Das Teilprojekt mediaTUM der Bibliothek umfasst dabei den Ausbau des Bibliotheksportals und den Aufbau eines institutionellen Repositories (IR) bzw. eines zentralen Medienserver. Beide Systeme sollen an den zentralen, hochschulweiten Verzeichnisdienst angeschlossen werden. Hauptaugenmerk ist die Entwicklung des Medienservers, der den Multimediaeinsatz in Forschung und Lehre sowie die Publikation digitaler Dokumente unterstützen soll. Elementarer Bestandteil zum Erhalt der Daten ist die Langzeitarchivierung, die direkt über den Medienserver steuerbar angeschlossen werden muss.

Ausgehend von der Betrachtung vorhandener Systeme wurde eine MyCoRe-Installation an der TUM zur Publikation elektronischer Prüfungsarbeiten betrieben. Die fehlende Archivierungsschnittstelle und zusätzliche retrodigitalisierte Diabestände führten zu Anforderungen, die MyCoRe nicht abdecken konnte. Es bestand keine Möglichkeit, die fehlenden Bestandteile innerhalb einer kurzen Zeitspanne nachzurüsten. Vor allem der Bereich der Bildarchive stellte neue Anforderungen, die erst in einem pragmatischen Ansatz zusammen mit den Anwendern entwickelt werden mussten. Kommerzielle Produkte schieden aufgrund der Forderung nach einer Open-Source-Lösung komplett aus. Ergebnis war die Entwicklung eines Prototyps für Bildarchive, der durch die flexible Auslegung auch für die Publikation elektronischer Hochschulschriften geeignet war. Seit 2007 betreibt die Bibliothek der TUM einen Medienserver auf der Basis dieser neuen Software (mediaTUM) und löste damit MyCoRe ab.

Die Praxisnähe des Projekts mediaTUM bietet Vorteile, die in die folgenden Betrachtungen mit einfließen. Der große Datenbestand unterschiedlicher Medien erleichtert die Verallgemeinerung der Probleme und ermöglicht eine Bewertung und den Test der gefunden Lösung.

2.6 Stand der Dinge

Die Definition des Begriffs des Medienservers und die Aufstellung notwendiger und wünschenswerter Kriterien machen einen Vergleich bestehender Softwarelösungen möglich. Während des Vergleichs konnten klare Defizite existierender Medienserver ausgemacht werden, die sich insbesondere im Bereich der Metadaten und der Langzeitarchivierung wiederfinden. Gerade die Flexibilität und die Nachhaltigkeit sind kritische Faktoren, die klarer Regelungen bedürfen. Durch den rasant wachsenden Markt und neue Technologien ist es entscheidend, schnell praktikable Lösungen für diese beiden Defizite zu finden.

Metadaten müssen flexibel anpassbar sein. Dadurch ergeben sich neue Probleme, vor allem im Bereich des Datenaustauschs. An dieser Stelle steht die Forderung nach einem einzigen allumfassenden Metadatenschema im Mittelpunkt, das das komplette Spektrum abdecken kann. Nicht nur das Schema selbst, auch die Verbindung zwischen Metadaten und Medium müssen gepflegt werden.

Die nachhaltige Sicherung und die Erhaltung des Zugriffs stellen zusätzliche Anforderungen an einen Medienserver. Dabei ist die Trennung zwischen reiner Sicherung auf physischer Ebene und der Erhaltung der Nutzbarkeit erforderlich. Die reine Sicherung der Daten ist der leichtere Teilaspekt und bereits umfassend gelöst. Die Erhaltung der Nutzbarkeit stellt eine größere Herausforderung dar, weil an dieser Stelle zusätzliche unvorhersehbare Faktoren auftreten, die nicht beeinflusst werden können.

Medienserver bieten Unterstützung für Dateiformate aus unterschiedlichen Bereichen. Die Vielfalt vorhandener Dateiformate birgt die Gefahr, immer neue Formate unterstützen zu müssen und dabei die Probleme bei der angewandten Langzeitarchivierung zu erhöhen.

3 Metadaten

Dieses Kapitel führt die Begriffe Metadaten und Metadatenformat ein und kategorisiert gängige Metadatenstandards hinsichtlich ihrer Definition oder der Verwendung. Abschließend wird die Eignung der Formate für Medienserver bewertet.

3.1 Definition des Begriffs Metadaten

Metadaten oder Metainformationen beschreiben Daten mit zusätzlichen Informationen. Es handelt sind bei Metadaten also um „Daten über Daten“. Diese Zusatzinformationen werden bereits seit Jahrhunderten in Bibliotheken zur Verwaltung großer Datenbestände eingesetzt. Im Computerzeitalter wird der Begriff Metadaten geprägt und beschreibt Dateien näher. Der Direktor des Word Wide Web Consortiums (W3C) definiert den Begriff wie folgt: „Metadaten sind maschinenlesbare Informationen über elektronische Ressourcen oder andere Dinge“ [**30**].

Eine strikte Trennung zwischen Daten und Metadaten ist schwierig, denn es muss immer der Gesichtspunkt der Betrachtung mit berücksichtigt werden. Diese Tatsache erschwert auch die Definition einer allgemeingültigen Menge an Zusatzinformationen. Erst unter der Betrachtung des Zwecks wird eine Unterscheidung zwischen gewöhnlichen Daten und Metadaten möglich. Die Betrachtung des Zwecks lässt damit aus der Summe der verfügbaren Informationen über ein Objekt eine Einteilung in Daten und Metadaten zu. Die Grenzen sind dabei oft fließend.

Standards sind in einem heterogenen Umfeld oder bei der Interaktion unterschiedlicher Systeme entscheidend für die Praktikabilität. Erst dadurch werden unterschiedliche Quellen erschließbar. Standards werden dabei auf unterschiedlichen Ebenen definiert:

- **Syntax**
  Die Syntax integriert die Metainformationen in das Datenmodell
- **Datenmodell**
  Das Datenmodell legt die Struktur der Metadaten fest
- **Semantik**
  Die Semantik definiert die Bedeutung der Metadaten innerhalb des Datenmodells

Diese Ebenen sind einerseits fest miteinander verbunden, da sie aufeinander aufbauen. Andererseits sind sie austauschbar durch andere Repräsentationen. Für jede dieser Schichten existieren Standards, die in unterschiedlichen Kombinationen eingesetzt werden können. Für die Interaktion kommt noch eine eindeutige Identifikation des Objekts hinzu. Identifikatoren bilden eine eigene Säule neben den Ebenen [**72**].

Vorteile bringen Metadaten hinsichtlich unterschiedlicher Kriterien. Metadaten erleichtern das Auffinden von Informationen und Daten. Je genauer die Beschreibung der Daten durch Metadaten ist, desto besser wird eine Selektion relevanter Informationen gewährleistet. Die eindeutige Identifizierung der Daten ermöglicht den Austausch der Informationen, verbessert den Komfort der Nutzung und ermöglicht Verweise innerhalb heterogener Datenbestände.

3.2 Metadatenformate

Die Definition von Kriterien auf einer oder mehrerer der Metadatenebenen erlauben die Festlegung unterschiedlicher Metadatenformate und deren Standardisierung. Frameworks bilden den Rahmen für Entwicklungen und die Nutzung der Formate in unterschiedlichem Kontext.

Unterscheidungen können erfolgen nach:

- Definition:
  - Syntaktische Formate
  - Semantische Formate
- Einsatzgebiet
  - Beschreibende Formate
  - Austauschformate
  - Technische Formate

Eine Abgrenzung der Kategorien und die damit verbundene Zuordnung der Metadatenformate in genau eine dieser Kategorien ist nur bei wenigen Formaten möglich. Diese Formate sind speziell auf einen bestimmten Zweck ausgelegt und bilden damit Speziallösungen. Erweiterungen in der Nutzung sind üblich und so werden Formate in unterschiedlichem Kontext eingesetzt.

Falls keine eindeutige Festlegung erfolgt, nur ein System das Format nutzt oder keine Spezifikation vorhanden ist, spricht man von einem proprietären Format. Trotz der Einstufung als proprietäres Metadatenformat kann sich daraus ein quasi-Standard entwickeln. Gründe dafür können die weite Verbreitung des Dateityps sein, der dieses Format nutzt oder auch eine einfache Handhabung des Formats. Die Ausbildung eines Standards aus einem proprietären Format ist durch eine spätere Definition möglich.

3.3 Syntaktische Formate

Syntaktische Formate verwenden feste Definitionen der Syntax. Zusätzlich werden eindeutige Identifikatoren vorgegeben. Die Semantik bleibt offen, somit sind beliebige Erweiterungen an den Metadatenattributen möglich. Das Datenmodell ist vom jeweiligen Dateityp abhängig, der mit Metadaten ausgestattet werden soll.

Die Extensible Markup Language (XML) ist eine bekannte syntaktische Sprache, um Textdaten strukturiert darstellen zu können. Das World Wide Web Consortium (W3C) hat XML 1998 als Recommendation verabschiedet und 2000 einer Revision unterzogen [**94**].

Über die syntaktische Definition von Tags werden Strukturinformation und eigentliche Daten unterschieden. Die hierarchische Struktur gleicht einem Baum mit einer Wurzel. Mit XML sind einige Vorteile verbunden:

Parser
XML-Parser können Aussagen über eine XML-Datei liefern. Es existieren unterschiedliche Parserklassen, die entweder zum Auslesen oder zur Analyse der Gültigkeit eingesetzt werden (validierende Parser).

Wohlgeformtheit
Für ein wohlgeformtes XML-Dokument müssen Bedingungen erfüllt werden, die sich maschinell abprüfen lassen. Jedes Dokument muss genau ein eindeutig definiertes Wurzelelement aufweisen. Strukturelemente, die Inhalt besitzen, müssen immer von einem öffnenden und einem schließenden Tag umschlossen sein. Nur Elemente ohne Inhalt können in sich geschlossen verwendet werden. Bei

hierarchischer Nutzung der Elemente müssen die Elemente paarweise auf derselben Ebene mit Start- und Endtag verwendet werden. Bei Verwendung attributierter Strukturelemente dürfen keine Doppelungen in den Attributnamen auftreten. Wenn keine dieser Regeln verletzt ist, spricht man von einem wohlgeformten XML-Dokument.

Gültigkeit
Zusätzlich zur Wohlgeformtheit kann ein XML-Dokument eine Strukturdefinition in Form einer Dokumenttypdefinition (DTD) oder eines XML-Schemas besitzen. In der DTD oder dem Schema werden mögliche Elemente und Attribute in ihrer Hierarchie beschrieben. Anhand dieser Grammatik-Definition kann die Gültigkeit automatisiert validiert werden.

Aufgrund der Flexibilität und Beschaffenheit der XML-Spezifikation wird an vielen Stellen XML-Syntax zur Definition von Sprachen oder Protokollen genutzt, beispielsweise in der Datenbeschreibung und im Datenaustausch [**16**]. Einige der im Folgenden beschriebenen Metadatenformate setzen XML zur Darstellung ein.

3.4 Semantische Formate

Semantische Formate nutzen die festgelegte Semantik zur Definition eigener Sprachen. Objektmodell und Syntax bleiben variabel. Eine eindeutige Identifikation ist erforderlich und wird über unterschiedliche Mechanismen erreicht.

3.4.1 Resource Description Framework (RDF)

Das Resource Description Framework (RDF) bezeichnet Standards zur formalen Anreicherung von Objekten um beschreibende Informationen. Eindeutige Bezeichner kennzeichnen Objekte. 1999 verabschiedete das W3C in einer Empfehlung das RDF [**96**], wobei das Rahmenwerk dabei offenlässt, was eine Ressource ist. Verallgemeinert kann RDF alle Dinge beschreiben, die eindeutig zu Identifizieren sind. Als Darstellungsform haben sich RDF-Syntax in XML-Hypertext und als RDF-Modell der RDF-Graph durchgesetzt.

Die Beziehungen zwischen Objekten werden in Sätzen (RDF-Tripel) modelliert, die aus genau drei Elementen bestehen:

Subjekt
RDF-Ausdrücke beschreiben Subjekte oder Ressourcen. Jede Ressource muss dabei eindeutig über eine Bezeichnung definiert werden. Die Definition kann dabei über eine eindeutige URI erfolgen (Ellipse in RDF-Graph-Darstellung).

Prädikat
Prädikate oder auch Eigenschaftselemente sagen etwas über ein Subjekt aus. Sie haben in der sprachlichen Grammatik die Aufgabe, die Ressource näher zu beschreiben (gerichtete benannte Kante in RDF-Graph-Darstellung).

Objekt
Objekte bilden den Wert eines Prädikats (Rechteck in RDF-Graph-Darstellung).

```
<rdf:Description rdf:about="" xmlns:pdf="http://ns.adobe.com/pdf/1.3/">
        <pdf:Producer>Microsoft® Office Word 2007</pdf:Producer>
        <pdf:Keywords>lorem ipsum</pdf:Keywords>
</rdf:Description>
```

Code 3–1: RDF Beispiel als XML Hypertext

Abbildung 3-1: RDF-Beispiel als RDF-Graph

Die Beispiele Abbildung 3-1 und Code 3–1 stellen jeweils denselben Sachverhalt in unterschiedlicher Syntax dar. In beiden Fällen werden Hersteller und Schlüsselworte zu einem Objekt direkt beschrieben.

Zusätzlich zur direkten Beschreibung eines Objekts gibt es indirekte Aussagen. Diese werden nicht über die direkte Verbindung eines Subjekts mit einem Objekt erreicht sondern über weitere RDF-Teilgraphen. Diese Eigenschaft ermöglicht die Definition von Abfragesprachen über RDF und damit eine automatisierte Auswertung und Informationsgewinn. Nach der offiziellen Verabschiedung von SPARQL 2008 durch das W3C hat sich diese Abfragesprache als quasi-Standard durchsetzen können.

```
PREFIX abc: <=http://ns.adobe.com/pdf/1.3/#>
SELECT ?producer ?keywords
WHERE {
  ?x abc:producer ?producer
}
```

Code 3–2: SPARQL Beispiel Query

Die Anwendung der Query aus Code 3–2 auf das RDF-Beispiel aus Code 3–1 liefert als Resultat „lorem ipsum“. Die einfache Anwendung der SPARQL-Queries auch auf sehr große Datenmengen liefert schnelle Ergebnisse und die Vorteile der Metadatenvergabe lassen sich ausnutzen. Vor allem im Bereich des Semantic Web [**26**] kommt inzwischen verstärkt RDF zum Einsatz. Suchmaschinen im World Wide Web nutzen RDF-Informationen als Erweiterung zu normalen textuellen Suchtreffern.

3.4.2 Extensible Metadata Platform (XMP)

Die Extensible Metadata Platform (XMP) ist eine standardisierte Form, um Metadaten in digitale Medien einzubetten. Adobe Inc. veröffentlichte 2001 erstmals den Standard und führte die Nutzung mit dem Acrobat Reader in Version 5 ein [**3**]. XMP basiert auf der formalen Sprache RDF und bettet Metadaten in Binärdateien ein. Ziel von XMP ist eine einheitliche Speicherung von Metadaten und die Nutzung in unterschiedlichen Applikationen nach einem einheitlichen Schema. Adobe unterstützt in allen eigenen Applikationen XMP, für die Integration in weitere Applikationen steht ein XMP-Toolkit-SDK [**2**] zur Verfügung.

Inzwischen unterstützt XMP verschiedene Schemata, die auf die Nutzung innerhalb der Dateitypen optimiert sind. Zugrunde liegt das Dublin Core Schema (siehe Kapitel 3.5.1). Änderungen und Erweiterungen um spezielle Felder werden über die Schemadefinition ermöglicht. Auch die Kombination unterschiedlicher oder mehrerer Schemata ist anhand der XML-Spezifikation möglich. Damit ist XMP eine sehr flexible Möglichkeit, Metadaten innerhalb Binärdateien zu speichern.

3.5 Beschreibende Formate

Beschreibende Metadatenformate liefern Zusatzinformationen zu vorliegenden Objekten hinsichtlich Inhalt oder Entstehung. Sie werden nach ihrem Einsatzgebiet in Kategorien unterteilt. Im Folgenden werden gebräuchliche Vertreter beschrieben, eine umfassende Aufstellung ist aufgrund der Masse an

Formaten kaum möglich. Die beschreibenden Metadaten liefern wichtige Zusatzinformationen, um aus einzelnen Objekten Serien oder Kollektionen zusammenstellen zu können. Für den langfristigen Erhalt der Objekte liefern diese Daten die Grundlage der Kategorisierung und Einordnung (vgl. Kapitel 5).

3.5.1 Dublin Core (DC)

Einen grundlegenden Satz an Metainformationen liefert das Dublin Core Metadatenmodell der Dublin Core Metadata Initiative (DCMI) [**29**], das von verschiedenen Disziplinen aus der Informatik, Wissenschaft und dem Bibliothekswesen entwickelt wurde. Mit Version 1.1 des Dublin Core Metadata Element Sets, ISO Standard 15836, werden grundlegende Attribute spezifiziert, die aus unterschiedlichen Bereichen Informationen liefern.

Das Core-Set ist Bestandteil des DCMI Metadata Terms, das neben weiteren Elementen zusätzlich ein kontrolliertes Vokabular für Objekttypen enthält. Innerhalb des Sets werden lediglich Attribute mit festgelegten Bedeutungen definiert:

Attributname	**Bedeutung**
DC.contributor	Mitwirkender bei der Erstellung der Ressource
DC.coverage	Beschreibung des Geltungsbereichs, z.B. Ortsname
DC.creator	Verantwortlicher für die Erstellung
DC.date	Speicherung von Datums-/Zeitangaben innerhalb der Entwicklung
DC.description	Beschreibung der Ressource
DC.format	Angabe des Formats der Ressource
DC.identifier	Identifikator der Ressource im Kontext
DC.language	Sprache der Ressource
DC.publisher	Verleger/Verantwortlicher für die Verfügbarkeit
DC.relation	Beschreibung verwandter Ressourcen
DC.rights	Informationen über die Rechte der Ressource
DC.source	Verwendete Quellen-Beschreibung
DC.subject	Thema der Ressource
DC.title	Titel der Ressource
DC.type	Typdefinition mittels DCMI-Type-Vokabular-Liste

Tabelle 3-1: Metadaten-Attribute des Dublin-Core Sets

Die Darstellung oder Ablage der Metadaten mit Dublin-Core ist nicht festgelegt, so dass verschiedene Möglichkeiten der Integration von Metadaten bestehen. Die Darstellung erfolgt in der Regel in XML oder RDF und wird von vielen gängigen Formaten unterstützt, darunter auch ODF (siehe Kapitel 4.2.3) oder HTML (siehe Kapitel 4.2.2). Vorteil dieses Verfahrens ist die maschinelle Lesbarkeit der Metadaten und der Austausch der Daten zwischen unterschiedlichen Anwendungen.

3.5.2 Metadata Encoding and Transfer Schema (METS)

Das Beschreibungsformat Metadata Encoding and Transfer Schema (METS) [**50**] liefert Metadaten zu Objekten für digitale Sammlungen. Die Sammlung kann dabei aus mehreren Teilen bestehen oder strukturiert in Hierarchien aufgebaut sein. METS ist definiert als XML-Schema im XML-Format. Das Format der Metadaten ist nicht fest definiert. Unterschiedliche Abschnitte des METS-Schemas erlauben sowohl die Darstellung der inneren Struktur eines digitalen Objekts und die Gruppierung zusammengehöriger Dateien, als auch die Verwendung technischer Metadaten oder Angaben zum Ausgangsmaterial. Zusätzlich zur Beschreibung einer digitalen Sammlung wird METS auch beim Datenaustausch zwischen Systemen verwendet. Innerhalb des OAIS-Referenzmodells (siehe Kapitel 5.2)

kommt METS an unterschiedlichen Stellen als Liefereinheit, Archivierungseinheit oder Bereitstellungseinheit zum Einsatz.

Unterschiedliche Abschnitte innerhalb METS:

METS-Header
Im METS-Header (metsHdr) werden minimale Metadateninformationen über die vorliegende Ressource hinterlegt. Dazu zählen Informationen über Personen mit deren Funktion und der Bearbeitungsstatus der Ressource. Zur besseren Standardisierung werden vordefinierte Attribute und auch Attributwerte verwendet.

```
<mets:metsHdr CREATEDATE="2004-06-05T10:15:00" LASTMODDATE="2004-06-05T10:15:00">
    <mets:agent ROLE="CREATOR" TYPE="ORGANIZATION">
        <mets:name>Library Systems Office, University of California, Berkeley</mets:name>
    </mets:agent>
</mets:metsHdr>
```

Code 3–3: METS-Header in XML-Darstellung

Descriptive Metadata
Der Bereich der Erschließungsangaben besteht aus beliebig vielen internen (mdWrap) oder externen (mdRef) Erschließungsangaben-Bereichen, die in ein dmdSec-Objekt eingehängt werden. Interne Bereiche speichern die Metadaten direkt, externe liefern über einen URI einen Link auf weiterführende Daten.

```
<mets:dmdSec ID="PMDM1">
    <mets:mdRef xlink:href="http://www..." LOCTYPE="URL" MDTYPE="EAD" LABEL="..."/>
</mets:dmdSec>
<mets:dmdSec ID="DMD1A">
    <mets:mdWrap MDTYPE="MODS">
        <mets:xmlData>
            <mods:mods>
                <mods:titleInfo>
                    <mods:title>title</mods:title>
                </mods:titleInfo>
            </mods:mods>
        </mets:xmlData>
    </mets:mdWrap>
</mets:dmdSec>
```

Code 3–4: Erschließungsangaben in METS in XML-Darstellung (intern und extern)

Administrative Metadata
Verwaltungsangaben (amdSec) enthalten Angaben zur administrativen Verwaltung des Objekts bzw. des ursprünglichen Archivobjekts und werden in vier Unterobjekte gegliedert: Technische Metadaten, Urheberrechte und Lizenzinformationen, Angaben zur Quelle und digitale Herkunftsangaben. Diese Unterteilungen folgen dabei der Syntax der Descriptive-Metadata.

```
<mets:amdSec>
    <mets:techMD ID="ADM1">
        <mets:mdWrap MDTYPE="NISOIMG">...</mets:mdWrap>
    </mets:techMD>
    ...
    <mets:techMD ID="ADM n">
        <mets:mdWrap MDTYPE="NISOIMG">...</mets:mdWrap>
    </mets:techMD>
</mets:amdSec>
```

Code 3–5: Administrative Metadaten in METS in XML-Darstellung

File Section
Der Abschnitt zu Dateien (fileSec) enthält mindestens ein fileGrp-Objekt. Dort werden alle digitalen Ausprägungen des Objekts angegeben und über Attribute näher gekennzeichnet.

```
<mets:fileSec>
    <mets:fileGrp VERSDATE="2003-01-22T00:00:00.0" USE="archive image">
        <mets:file ID="FID1A" MIMETYPE="image/tiff" SEQ="1" CREATED="2003…">
        <mets:FLocat xlink:href="..." LOCTYPE="URL"/>
      </mets:file>
    </mets:fileGrp>
    ...
</mets:fileSec>
```

Code 3–6: Dateibeschreibung in METS in XML-Darstellung

Structural Map

Die hierarchische Strukturbeschreibung (structMap) bereitet Informationen über den Aufbau des digitalen Objektes auf und bietet diese dem Benutzer zur Navigation innerhalb des Objekts an.

```
<mets:structMap TYPE="physical">
    <mets:div ORDER="1" TYPE="still image" LABEL="..." DMDID="PMDM1 DMD1A">
      <mets:fptr FILEID="FID1A"/>
      <mets:fptr FILEID="FID2A"/>
      <mets:fptr FILEID="FID3A"/>
      <mets:fptr FILEID="FID4A"/>
    </mets:div>
</mets:structMap>
```

Code 3–7: Strukturbeschreibung in METS im XML-Format

Structural Links

Strukturverknüpfungen (smLink) definieren Links innerhalb eines Objekts, die über die Structural Map angelegt wurden.

```
<mets:smLink from="IMG1" to="P2" xlink:title="Hyperlink from
JPEG Image on Page 1 to Page 2" xlink:show="new"
xlink:actuate="onRequest" />
```

Code 3–8: Verknüpfungsbeispiel in METS im XML-Format

Behavior

Der Verhaltensabschnitt definiert eigene Elemente für Aktionen, die in einem Objekt ausgeführt werden sollen.

```
<mets:behavior ID="DISS1.1" STRUCTID="S1.1" BTYPE="uva-bdef:stdImage"
        CREATED="2002-05-25T08:32:00" LABEL="..."
        GROUPID="DISS1" ADMID="AUDREC1">
    <mets:interfaceDef LABEL="..."
        LOCTYPE="URN" xlink:href="uva-bdef:stdImage"/>
    <mets:mechanism LABEL="..."
        LOCTYPE="URN" xlink:href="uva-bmech:BETTER-imageMech"/>
</mets:behavior>
```

Code 3–9: Verhaltensabschnitt in METS im XML-Format

3.5.3 IPTC/ IIM und IPTC-Core

Den IPTC-NAA-Standard [**19**] zur Speicherung von Textinformationen innerhalb von Bildern gibt es seit 1990. Das International Press Telecommunications Council (IPTC) entwickelte zusammen mit der Newspaper Association of America (NAA) das gleichnamige Format. Es wird bei mehreren Bild-Dateiformaten eingesetzt. Geeignet ist es auch für andere Formate, dort konnte es sich bisher allerdings nicht durchsetzen.

IPTC Code	IPTC-Feld	Beschreibung	Max. Länge
0x05	Object Name	Kurzbezeichnung	64

0x19	Keywords	Schlagworte	je 64 Zeichen, max.64KB
0x6E	Credit	Lieferant	32
0x74	Copyright Notice	Bildrechte	128
0x78	Caption/ Abstract	Beschreibung des Bildinhaltes	2000
0x07	Edit Status	Bearbeitungsstand	64
0x0F	Category	Kategorie	3
0x14	Supplemental Category	Frei wählbare Kategorien (angekündigt)	Je 32 Zeichen, max. 64KB
0x16	Fixture Identifier	Kennzeichner	32
0x28	Special Instructions	Sonstige Hinweise	256
0x37	Date Created	Aufnahmedatum	8
0x3C	Time Created	Aufnahmezeit	11
0x3E	Digital Creation Date	Digitalisierungsdatum	8
0x3F	Digital Creation Time	Digitalisierungszeit	11
0x41	Originating Program	Erzeugendes Programm	32
0x4B	Object Cycle	Bearbeitungszyklus	1
0x50	By-line	Name des Autors/Fotografen.	32
0x55	By-line Title	Titel des Autors/Fotografen. Wird hinter die Autorenangabe gesetzt.	32
0x5A	City	Aufnahmeort: Stadt	32
0x5C	Sublocation	Aufnahmeort: Präzisierung	32
0x5F	Province/ State	Aufnahmeort: Bundesland/Kanton	32
0x64	Country/ Primary Location Code	Aufnahmeort: Staat/Kürzel	3
0x65	Country/ Primary Location Name	Aufnahmeort: Staat	64
0x67	Original Transmission Reference	Übermittlungsort	32
0x69	Headline	Titel	256
0x73	Source	Quelle	32
0x76	Contact	Kontaktperson	128
0x7A	Writer/ Editor	IPTC-Daten Verfasser	128

Tabelle 3-2: IPTC Attribut-Felder

Zusätzlich zu den reinen beschreibenden Metadatenformaten wurde IPTC auch immer als Datenaustauschformat verstanden und als IIM (Information Interchange Model)-Standard bezeichnet.

Seit 2004 gibt es eine Überarbeitung von IPTC (IIM) mit dem Namen IPTC-Core. Ab diesem Zeitpunkt wird IPTC im XMP Format als XML in Dateien abgelegt. Beide Varianten können nebeneinander existieren, bereiten aber auch Probleme, da nicht jedes Programm genau spezifiziert, welche Informationen ausgelesen oder bearbeitet werden.

Tags identifizieren die IPTC-Informationen und dienen beim Auslesen als Identifikator. Tabelle 3-2 listet alle möglichen Tags zusammen mit ihrer Bedeutung und Länge auf. Dabei ist noch zu unterscheiden, ob ein Feld mit fester oder variabler Länge vorliegt. Einfache Datensätze (dunkel unterlegt) besitzen immer eine feste Länge und sind Pflichtfelder. Die restlichen Tags (erweiterte Datensätze) besitzen eine variable Länge, die explizit angegeben werden muss.

Neuste Trends zeigen, dass IPTC zugunsten des XMP Standards immer weiter verdrängt wird. Ein Grund dafür ist die Beschränkung auf einen festen Attributsatz, der nicht immer ausreichende Möglichkeiten der Beschreibung bietet. IPTC-Core liefert ein Schema zur Darstellung in XML.

3.5.4 ID3-Tags

ID3-Tags dienen zur Beschreibung der Metadaten innerhalb von Audiodateien. Das Format ist versioniert und unterstützt abhängig von der verwendeten Version unterschiedliche Metadatenfelder mit fester oder variabler Länge. ID3-Tags kommen in erster Linie bei MP3-Dateien zum Einsatz (siehe auch Kapitel 4.4.2).

ID3v1
Mit der ersten Version der ID3-Tags wurde 1996 ein kleiner Satz an Attributen eingeführt, die zusätzliche Informationen zum vorliegenden Audio-Objekt beinhalten. Im MP3-Format wird das Ende der Audiodaten durch eine Markierung festgelegt. Die Metadaten werden in einem 128 Byte großen Block nach dieser Markierung gespeichert. Die Größe und Position der Attribute innerhalb des Datenblocks ist festgelegt. Tabelle 3-3 listet alle möglichen Tags zusammen mit ihrer Länge und Position auf.

Offset	Länge	Bedeutung
0	3	*TAG* als Kennzeichen des ID3v1-Blocks
3	30	Titel
33	30	Interpret
63	30	Album
93	4	Erscheinungsjahr
97	30	Kommentar
127	1	Genre

Tabelle 3-3: ID3v1 Attribute

Probleme bei der Interpretation des Tags kann es geben, wenn ein Datenblock zufällig auch mit dem einleitenden Wort „TAG“ beginnt. Außerdem ist zu beachten, dass das Genre in einer eigenen Liste mit ids hinterlegt ist und innerhalb der Metadaten als Zahl mit einem einzigen Byte gespeichert wird.

ID3v1.1
Version 1.1 bringt als Neuerung ein neues Datenfeld für die Titelnummer an Byteposition 126 des ID3-Tags mit, was zu einer Überarbeitung des Kommentarfeldes geführt hat, das auf 28 Byte gekürzt wurde. Byte 125 wird auf den festen Wert 0 festgelegt.

ID3v2
Gerade wegen der festgesetzten Länge der einzelnen Attributfelder wurde eine Überarbeitung des ID3-Tags 1998 zu Version 2 durchgeführt. Vor allem für ein schnelles Auslesen und für die Nutzung der Dateien über Streaming wurde die Position der Metadaten an den Anfang der Datei verlegt. Mit einem speziellen Tag „ID3“ mit der zusätzlichen Angabe der Version wird der Metadatenbereich eingeleitet. ID3-Tags erfahren bezüglich ihrer Spezifikation eine ständige Weiterentwicklung, so dass aktuell Version 4 als ID3v2.4 in den meisten Programmen zum Einsatz kommt. Ältere Versionen können teilweise weiterverwendet werden oder parallel vorhanden sein.

Mit Version 2.4 sind nun die unterschiedlichsten Attribute vorgesehen (siehe Anhang A.c), die ein größeres Maß an Flexibilität durch ihre Vielfalt und variable Länge bieten. Das Problem unterschiedlicher Zeichensatzkodierungen der Metadaten konnte ebenfalls zugunsten der Definition nach UTF-8 gelöst werden.

3.6 Austauschformate

Einen wichtigen Platz nehmen Metadatenformate ein, die zum direkten Austausch vorhandener Zusatzinformationen zwischen unterschiedlichen Programmen eingesetzt werden. Für diesen Bereich der Metadaten werden spezielle Anforderungen an die Spezifikation der Formate gestellt, damit ein reibungsloser Austausch ermöglicht werden kann. Vor allem eine Offenlegung der Spezifikation oder Standardisierung ist dabei zwingend erforderlich. Im Bereich der Langzeitarchivierung kommen diese Daten überwiegend zum Einsatz, um Informationen zwischen unterschiedlichen Archivsystemen auszutauschen (vgl. Kapitel 5).

3.6.1 Text Encoding Initiative (TEI)

1987 wurde die Text Encoding Initiative (TEI) gegründet, aus der ein gleichnamiges Datenformat hervorging, das auch die Behandlung von Metadaten beinhaltet [**83**]. Das Datenformat ist dabei in erster Linie als Austauschformat für Texte gedacht und konnte sich in einigen Bereichen der Wissenschaft etablieren. Neben dem reinen Datenaustausch sind auch Richtlinien zum gesamten Publikationsprozess entstanden. Dieser Prozess umfasst die Dokumentanalyse, die Digitalisierung, die Textauszeichnung und die Publikation [**82**]. Grundlage bilden Daten, die über ein XML-Format in der Metasprache TEI definiert werden.

```
<teiCorpus>
    <teiHeader type="corpus">
        <!-- umgebendes Element bei Objektlisten -->
    </teiHeader>
    <TEI>
        <teiHeader type="text">
            <fileDesc>
                <titleStmt>
                    <title>Titel</title>
                </titleStmt>
                <publicationStmt>
                    <!-- Publikationsinformationen -->
                </publicationStmt>
                <sourceDesc>
                    <!-- Bibliographische Informationen -->
                </sourceDesc>
            </fileDesc>
            <encodingDesc>
                <!-- optionale Komponente -->
            </encodingDesc>
            <profileDesc>
                <!-- optionale Komponente -->
            </profileDesc>
        </teiHeader>
        <text>...</text>
    </TEI>
    <TEI>
        <teiHeader>
            <!-- weitere Metadaten -->
        </teiHeader>
        <text>...</text>
    </TEI>
</teiCorpus>
```

Code 3–10: TEI Struktur-Beispiel

Über die teiCorpus Entität können mehrere Einzelobjekte (TEI) zusammengefasst und näher beschrieben werden. Zu Beginn der Beschreibung der Einzelobjekte steht der teiHeader, in dem in der fileDesc-Entität alle wichtigen Informationen über das vorliegende Objekt zu finden sind. Weitere Untersektionen des Headers sind optional. An den Header schließt eine text-Entität an, innerhalb der der eigentliche Inhalt des Einzelobjekts gespeichert wird. Dieses Format eignet sich vor allem zur Speicherung von textuellen Informationen. Aber durch die einfache Erweiterbarkeit über die XML-DTD

können weitere Entitäten eingebunden werden, sodass der Feldumfang an dieser Stelle beliebig erweitert werden kann. Geliefert werden DTDs, die den Basissatz an Elementen beschreiben, spezifische genreabhängige Tags beinhalten und zusätzliche DTDs, die Spezialelemente beinhalten.

Code 3–10 stellt eine Basis-Beispieldefinition in TEI für ein Textobjekt dar. Dabei gliedern sich die unterschiedlichen Metadatenbereiche in unterschiedliche Entitäten. Gerade die Flexibilität der Beschreibung eines Textes und die damit verbundenen Metadaten machen TEI interessant, um Daten langfristig zu erhalten. Unterstützt wird TEI von unterschiedlichen Softwareprodukten, vor allem aus dem Bereich des Bibliothekswesens.

3.6.2 Machine-Readable Cataloguing (MARC)

Das Katalogisierungs- und Austauschformat MARC, Machine-Readable Cataloguing, kommt in erster Linie im Bibliotheksumfeld zum Einsatz, um über ein Protokoll Katalogdaten austauschen und Metadaten weitergeben zu können. Entwickelt wurde MARC in den 1970ern. Die kontinuierliche Weiterentwicklung führte zu ISO Standard 2709, als Format für den Informationsaustausch [**34**].

Aktuell liegen verschiedene Unterformate vor, die sich alle auf diesen Standard abstützen. Unterschiede der einzelnen Unterformate beziehen sich beispielsweise auf die Granularität der gespeicherten Informationen oder die Position, an der die Daten gespeichert werden. Die bedeutendsten Vertreter sind dabei:

UNIMARC
Universal Machine Readable Cataloging (UNIMARC) definiert die Basis-Datenstruktur, die bei MARC zum Einsatz kommt und bildet die Basis der Entwicklung. Hier werden grundlegende Elemente für die Beschreibung definiert. Die Metadaten untergliedern sich in zehn Kategorien, die unter anderem die eindeutige Identifikation erlauben, beschreibende Metadaten und Annotationen liefern und Verweise auf andere Datenbestände ermöglichen.

MARC21
Aus USMARC und CANMARC entwickelte die Library of Congress zusammen mit merhreren Communities das Format MARC21 und integrierte es 2004 in das eigene Bibliothekssystem. Dabei handelt es sich bei den beiden Ursprungsformaten um MARC-Ausprägungen mit spezifischen Erweiterungen für den amerikanischen und kanadischen Sprachraum. MARC21 definiert fünf Formate für den Austausch und die Darstellung für bibliografische, Autoritäts-, Bestands- Klassifikations- und Bürgerinformationsdaten in maschinenlesbarer Form. Die Speicherung der Informationen erfolgt entweder im Klartext oder anhand vorgegebener Kataloge. Entscheidend ist die feste Definition der Codierung der Daten nach ISO 2022 oder UTF-8. Mit diesen beiden Codierungen ist das komplette Spektrum an Zeichen aus den unterschiedlichen Sprachräumen abgedeckt und der Datenaustausch ohne Probleme möglich.

```
00673nam a2200217 a 4504001003300000003000900033O
0500170004200800410005901500190010002000170011903
5001700136040003100153082001600184100001900200245
0062002192600033002813000020003146500060003346500
0310039465500300042 59cbbe7fc3a7346d99c281979d45b6
79cUK-BiTAL20050705133033.0990831s1999 enk
j 000 ||eng|d aGB99Y57412bnb a0747542155 :
a()0747542155 aStDuBDScStDuBDSdUK-BiTAL04a823.
9142211 aRowling, J. K.00aHarry Potter and the pr
isoner of Azkaban /cJ.K. Rowling. aLondon :bBloom
sbury,c1999. a317p. ;c21 cm. 0aPotter, Harry (Fi
ctitious character)vJuvenile fiction. 0aWizardsvJ
uvenile fiction. 7aChildren's stories.2lcsh
```

Abbildung 3-2: Unformatiertes MARC Beispiel

Metadaten werden im MARC-Format hintereinander weg geschrieben (Abbildung 3-2). Durch die Definition von Längen der einzelnen Felder und Identifikatoren kann die rein maschinenlesbare Form aufgebrochen werden und die darin enthaltenen Felder interpretiert werden.

```
LDR 00673nam a2200217 a 4504
001      9cbbe7fc3a7346d99c281979d45b679c
003      UK-BiTAL
005      20050705133033.0
008      990831s1999\\\\enk j\\\\\\000\||eng|d
015 \\$a GB99Y5741$2bnb
020 \\$a 0747542155 :
035 \\$a()0747542155
040 \\$a StDuBDS$cStDuBDS$dUK-BiTAL
082 04$a 823.914$221
100 1\$a Rowling, J. K.
245 00$a Harry Potter and the prisoner of Azkaban /$cJ.K. Rowling.
260 \\$a London :$bBloomsbury,$c1999.
300 \\$a 317p. ;$c21 cm.
650 \0$a Potter, Harry (Fictitious character) $vJuvenile fiction.
650 \0$a Wizards$vJuvenile fiction.
655 \7$a Children's stories.$2lcsh
```

Abbildung 3-3: Formatiertes MARC Beispiel

Die Datensätze aus Abbildung 3-3 sind zeilenweise lesbar und werden über eindeutige Identifikatoren bestimmten Metadatenfeldern zugeordnet. Die Identifikatoren bestehen dabei aus dreistelligen Zahlenkombinationen, die teilweise verpflichtend, teilweise wiederholbar ausgelegt sind. Der Identifikator 100 steht so beispielsweise für den Autor und beinhaltet Unterfelder, die über das $a Attribut angezeigt werden. Der Identifikator 001 hat einen besonderen Stellenwert innerhalb des Datensatzes. Es handelt es sich um den eindeutigen Identifikator des Datensatzes, der Pflichtfeld und nicht wiederholbar ist.

3.6.3 BibTeX

Literaturverzeichnisse können mit dem BibTeX-Format verwaltet werden. Das Format erleichtert mit einer einheitlichen textuellen Darstellung die Verwaltung sehr großer Bestände. Entwickelt wurde das Format ursprünglich, um in LaTeX-Dokumenten Literaturverzeichnisse erstellen zu können. Die feste Definition der verfügbaren Felder für unterschiedliche Publikationstypen führte zu einer weiten Verbreitung des Formats. Neben Pflichtfeldern sind typabhängig optionale Felder vorgesehen, die verarbeitet werden können. Tabelle 3-4 listet verfügbare Typen mit den dazugehörigen Feldern auf. Der Typ *misc* nimmt eine Sonderstellung in der Definition ein. Dieser Typ besteht komplett aus optionalen Feldern und ist damit universell einsetzbar für Beschreibungen, die in keine der anderen Definitionen passen. Werden weitere Felder außerhalb der Definition genutzt, sind diese immer softwareabhängig. Neben der reinen Nutzung für Literaturverzeichnisse wird BibTeX immer häufiger auch als Austauschformat mit eindeutiger Identifikation einer Quelle genutzt. Innerhalb der BibTeX-Definition wird für jeden Datensatz ein eindeutiger Identifikator gebildet (vgl. Code 3–11).

```
@phdthesis {sei2009,
author = "Seifert, Arne",
title = "Langzeitarchivierung bei Medienservern",
year = 2009,
type = "Dissertation",
school = "Universität der Bundeswehr München",
address = "München"
}
```

Code 3–11: BibTeX Beispiel der vorliegenden Dissertation

Verwendet man BibTeX rein als Austauschformat, ist die Formatierung der Darstellung nebensächlich. Als Literaturverzeichnis innerhalb eines Dokuments lässt sich die Formatierung und Auswahl der angezeigten Felder variieren und damit nahtlos in die Umgebung einpassen.

Typ	Pflichtfelder	optionale Felder
article	author, journal, title, year	month, note, number, pages, volume
book	author/editor, publisher, title, year	address, edition, month, note, series, volume/number
booklet	title	address, author, howpublished, month, note, year
conference	author, booktitle, title, year	address, editor, month, note, organization, pages, publisher, series, volume/number
inbook	author/editor, chapter und/oder pages, publisher, title, year	address, edition, month, note, series, type, volume/number
incollection	author, booktitle, publisher, title, year	address, chapter, edition, editor, month, note, pages, series, type, volume/number
inproceedings	author, booktitle, title, year	address, editor, month, note, organization, pages, publisher, volume/number, series
manual	title	address, author, edition, month, note, organization, year
mastersthesis	author, school, title, year	address, month, note, type
misc	-	author, howpublished, month, note, title, year
phdthesis	author, school, title, year	address, month, note, type
proceedings	title, year	address, editor, month, note, organization, publisher, series, volume/number
techreport	author, institution, title, year	address, month, note, number, type
unpublished	author, note, title	month, year

Tabelle 3-4: BibTeX-Typen mit Attributdefinition

3.7 Technische Formate

Technische Metadatenformate beschreiben Informationen über die Datei und das Objekt selbst. Das können einerseits Daten zur Entstehung des Objekts sein, andererseits auch technische Angaben zu Größe oder Auflösung. Die klare Einordnung einzelner Metadatenformate ist schwierig, da neben technischen Angaben meist auch andere Informationen in einer Formatspezifikation enthalten sind. Technische Metadaten sind immer abhängig vom vorliegenden Datenobjekt und Dateityp. Neben den Metadaten, die bereits direkt in der Dateiformatspezifikation enthalten sind, gibt es auch unabhängige Formate. Diese definieren nur die technischen Metadaten, die in unterschiedliche Dateitypen integriert werden können. Die Daten lassen sich in unterschiedliche Bereiche untergliedern:

- Grundlegende Parameter abhängig der vorliegenden Dateitypspezifikation,
- Metadaten zur Erzeugung,
- Metadaten zur Bewahrung der Qualität der Darstellung und
- Metadaten zur Verarbeitungsgeschichte.

Im Bildverarbeitungsbereich sind eigenständige Standards für technische Metadaten etabliert. Im Dokumentbereich besitzen die Dateiformate einen Basissatz an technischen Informationen (vgl. Kapitel 4). Technische Daten des Objekts werden für die Langzeitverfügbarkeit benötigt. Aus diesen Informationen lassen sich entscheidende Angaben zur Entstehungsgeschichte und zur korrekten Anzeige des Objekts gewinnen (vgl. Kapitel 5).

3.7.1 Metadata for Images in XML Schema (NISO MIX)

Die Library of Congress entwickelte zusammen mit dem NISO Technical Metadata for Digital Still Images Standards Committee ein XML-Schema, mit dem technische Metadaten von Bildern und Bildsammlungen abgebildet werden können. Das unter ANSI/NISO Z39.87 [**60**] standardisierte Format liefert Metadatenelemente, die zur Bearbeitung, dem Datenaustausch und der Interpretation von Bildern herangezogen werden können. Grundgedanke war die Nutzung des Formats zwischen Systemen und Services, aber auch den Langzeiterhalt und den dauerhaften Zugriff auf das Bildmaterial zu gewährleisten.

```
<mix xmlns=http://www.loc.gov/mix/v10
        xsi:schemaLocation="http://www.loc.gov/standards/mix/mix10/mix10.xsd"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <BasicDigitalObjectInformation>
        <!-- NISO Section 6 -->
    </BasicDigitalObjectInformation>
    <BasicImageInformation>
        <!-- NISO Section 7 -->
    </BasicImageInformation>
    <ImageCaptureMetadata>
        <!-- NISO Section 8 -->
    </ImageCaptureMetadata>
    <ImageAssessmentMetadata>
        <!-- NISO Section 9 -->
    </ImageAssessmentMetadata>
    <ChangeHistory>
        <!-- NISO Section 10 -->
    </ChangeHistory>
</mix>
```

Code 3–12: MIX-Sektionen

Das Format gliedert die Metadaten in vier Hauptkategorien und deckt damit das komplette Spektrum der technischen Metadaten ab. Code 3–12 zeigt diese Sektionen auf oberster Ebene. In den darunterliegenden Ebenen kommen konkrete Attribute zum Einsatz, die auf das Einsatzgebiet abgestimmt sind.

3.7.2 Exchangeable ImageFile Format (Exif)

Mit dem Exchangeable ImageFile Format (Exif) konnte sich ein Standard durchsetzen, der vor allem im Bereich der digitalen Fotografie eingesetzt wird. Digitalkameras speichern Informationen im Exif-Format innerhalb der Aufnahme, direkt vor den eigentlichen Bilddaten ab. Besonders bei den Formaten JFIF/JPEG und TIFF wird dieses Verfahren angewendet.

Exif in der Version 2.1 [**46**] wurde 1998 herausgegeben und wird seitdem von den meisten Kameraherstellern unterstützt. Version 2.2 stellt seit dem Jahr 2002 Erweiterungen bereit, die vor allem den Druck qualitativ hochwertiger Bilder durch eine verbesserte Kompatibilität zwischen Kamera und Drucker ermöglicht.

Grundsätzlich beginnt der Exif-Datenbereich innerhalb eines Bildes immer mit der Angabe der verwendeten Version. Darauf folgen die belegten Tags mit den Datenwerten. Die Formatierung innerhalb der Tags erfolgt dabei analog der TIFF-Definition für Datenfelder und wird in JPEG-Dateien in

der Regel im APP1-Abschnitt abgelegt. Die Tags können in vier Gruppen unterteilt werden, die neben den reinen Parametern zum Bild auch die verwendete Kamera und deren Charakteristik speichert. Zusätzlich können auch deskriptive Metadaten vergeben werden (vgl. Liste möglicher Tags unter Anhang A.a). Jedem Tag ist ein Datentyp mit einer festen Länge der Informationen zugeordnet. Zur Verkürzung nutzen einige Tags Wertelisten, die Integer-Werte auf fest definierte Wörterbücher abbilden. Vorschaubilder in unterschiedlicher Auflösung können ebenfalls integriert werden.

Bei Exif handelt es sich um keinen ISO-Standard sondern lediglich eine Übereinkunft der Kamerahersteller. Durch die Begrenzung der Anzahl der Tags kommt es dazu, dass manche Tags unterschiedlich genutzt werden und damit Inkompatibilitäten zwischen Herstellern entstehen.

3.8 Persistent Identifier

Die eindeutige Identifikation eines Objektes ist für die Langzeitarchivierung von entscheidender Bedeutung (vgl. Kapitel 5). Durch eine eindeutige Identifikation über einen Persistent Identifier (PI) lässt sich eine Ressource als ein bestimmtes Objekt kennzeichnen. Neben der Eindeutigkeit ist auch die Auffindbarkeit eines Objekts erforderlich. Abbildung 3-4 zeigt die Bestandteile eines PI und die Verbindung zwischen den Komponenten der Identifikation und der Lokation über Resolving-Mechanismen.

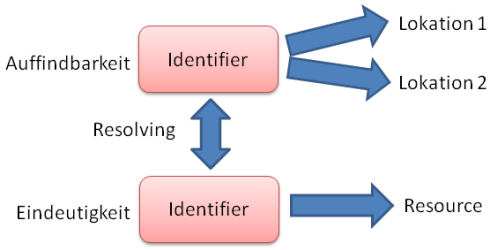

Abbildung 3-4: Bestandteile eines Persistent Identifiers

Erst über die Zuordnung zwischen Objekt und Standort kann ein dauerhafter Zugriff gewährleistet werden. Der zwischengeschaltete Resolver ermöglicht die strikte Trennung zwischen Komponenten der Identifikation einer Resource und der Standortreferenz. Die eindeutige Identifikation einer Ressource erfolgt entweder über das Objekt selbst oder über einen externen Katalog. Eine bekannte Form eines externen Identifiers ist der weltweit eindeutige Uniform Resource Locator (URL). Über die URL wird im Internet der elektronische Standort eines Objekts definiert. Dieser Standort kann über absolute oder relative Adressierung (Links) angesprochen werden. Wird der Standort verändert, ändert sich auch automatisch die URL. Synchronisationsmechanismen fehlen, so dass häufig verwaiste Links auf eine Resource entstehen.

Resolver kennen den aktuellen Standort einer Resource und leiten eingehende Anfragen entsprechend weiter. Damit müssen Hintergrunddienste bereitstehen, die mit technischem und personellem Aufwand für Pflege und Bereitstellung verbunden sind.

3.8.1 Hash-Funktionen

Hash-Funktionen oder Streuwertfunktionen dienen zur Identifikation eines digitalen Objekts. Algorithmen berechnen aus dem Datenstrom einer Datei eine eindeutige Prüfsumme. Diese Prüfsumme ist für jede Datei charakteristisch, lässt dabei aber keine Rückschlüsse auf den eigentlichen Inhalt der Datei zu. Es existieren unterschiedliche Verfahren, die Prüfsummen oder Fingerprints zu berechnen. Weit verbreitet sind:

- Message Digest Algorithmus 5 (MD5) [**70**],
- Secure Hash Algorithm (SHA) oder
- Zyklische Redundanzprüfung (CRC).

In Programmiersprachen ist der MD5-Algorithmus der am weitesten verbreitete, um eindeutige Hash-Werte (128 Bit) zu errechnen. SHA kommt aus dem Bereich der Kryptographie und liefert durch längere Prüfsummen (160 Bit) mehr Schutz gegen Brute-Force-Angriffe als MD5. CRC kommt vor allem bei der Datenübertragung zur Kontrolle des korrekten Datenstroms zum Einsatz. Beispielsweisenutzen CD- oder DVD-Laufwerke dieses Verfahren zur Erkennung von Lese- oder Übertragungsfehlern.

Von einer guten Hash-Funktion spricht man, wenn die Wahrscheinlichkeit von Kollisionen, die Abbildung unterschiedlicher Ausgangswerte auf einen identischen Hashwert, äußerst gering ist und die Hashwerte gleichverteilt sind. Durch Datenreduktion sollte die Länge des Hashwertes deutlich kürzer als der Ausgangswert sein. Die Berechnung der Prüfsumme soll effizient möglich sein, damit auch große Ausgangsdatenströme schnell verarbeitet werden können.

3.8.2 Object Identifier (OID) / Uniform Resource Name (URN)

Object Identifier (OID) sind weltweit eindeutige Bezeichner für ein Informationspaket. Dieser OID ist ein Knoten innerhalb eines Baumes, der über eine Nummernfolge zu erreichen ist. Neue Knoten können jeweils über den Besitzer des übergeordneten Knotens eingebunden werden. Die Verwaltung des Knotens und die Sicherstellung der Eindeutigkeit obliegen dem Besitzer des Knotens. Durch Standardisierung nach ISO/IEC 9834 und DIN 66334 ist die Konsistenz gegeben.

Der Uniform Resource Name (URN) ist ein Uniform Resource Identifier (URI), der über das Schema urn gebildet wird. Ein URN dient als dauerhafter und ortsunabhängiger Bezeichner für eine Resource. Eine Unterteilung in weitere Namensräume, den sog. Namespace Identifier (NID), ist möglich. Ein URN besitzt folgende Form:

```
urn:<NID>:<NID-spezifischer Teil>
```

In einem URN können auch bereits durch andere Mechanismen vergebene Bezeichner zum Einsatz kommen. Vor allem im bibliothekarischen Bereich werden oft URNs gebildet, um wichtige Schriften eindeutig und dauerhaft anbieten zu können.

3.8.3 International Standard Book Number/International Standard Serial Number

Die Internationale Standard Book Number (ISBN) ist ein weiterer Vertreter eindeutiger Identifikatoren. Seit der weiten Verbreitung des Internets und den damit verbundenen Vorteilen der Online-Publikation kam noch die ISSN, International Standard Serial Number, hinzu.

Die ISBN wurde Mitte der 1960er Jahre überwiegend durch das Engagement englischer Vereinigungen zur international eindeutigen Identifikation von Büchern gebildet. 1972 wurde unter der Norm ISO 2108 der offizielle Standard veröffentlicht. Bis 2006 bestand eine ISBN aus einer zehnstelligen Ziffernfolge (einschließlich einer Prüfziffer) (ISBN-10). Die zunehmende Anzahl der Verlage machte eine Erweiterung der bisher verwendeten Zahlenräume auf 13 Ziffern erforderlich, da keine Nummernkreise mehr frei waren (ISBN-13). Seit 2007 ist bei der Vergabe nur noch ISBN-13 in Gebrauch und verbindlich. Vergeben werden die ISBN-Nummern für Verlage jeweils in Zahlenkreisen, dadurch kann man über eine ISBN immer den zugrundeliegenden Verlag ermitteln und Kollisionen vorbeugen. Zusätzlich zur reinen Zahlendarstellung ist eine maschinenlesbare Strichcodedarstellung üblich [**43**].

```
ISBN 123-4-5678-9012-3
ISSN 1234-5678
```

Abbildung 3-5: Beispiel ISBN, ISSN

Um einheitliche Nummernkreise für Zeitschriften und Schriftenreihen zu erzeugen, nutzt man den auf der Norm ISO 3297 basierenden Standard ISSN. Die Nummer besteht dabei aus zwei Nummernpaketen zu je vier Ziffern. Der eigentliche Inhalt ist in den ersten sieben Ziffern gespeichert, die achte dient als die Prüfziffer. Die Berechnung der Prüfziffer erfolgt über einen klar definierten Algorithmus, der über eine Modulo-Differenz der gewichteten Quersumme der einzelnen Stellen die achte Stelle errechnet. Anders als bei der ISBN lassen sich aus der ISSN keine Rückschlüsse über den vertreibenden Verlag ermitteln, die Nummern werden fortlaufend über ISSN-Zentren im jeweiligen Land vergeben [**37**].

3.8.4 Digital Object Identifier (DOI)

Mithilfe des Digital Object Identitier (DOI) ist es möglich, ein digitales Objekt eindeutig zu identifizieren. Das System geht darüber hinaus noch einen Schritt weiter und speichert für jeden vergebenen DOI den aktuellen Ort, über den das Objekt abgerufen werden kann. Dabei wird in einer Datenbank eine Zuordnung geschaffen, dass eine Weiterleitung mittels eines Resolvers, beispielsweise zu einem URI, erfolgen kann.

Gepflegt wird das System von der International DOI Foundation (IDF), die die aktuelle Zuordnung verwaltet. Damit ist es möglich, alleine über die DOI ein digitales Objekt abrufbar zu halten. Zurzeit laufen einige Projekte, die sich mit der Nutzung und Erweiterung dieses Systems befassen. So werden über das Projekt STD-DOI [**77**] beispielsweise wissenschaftliche Primärdaten erfasst, um eine einfache Zitierbarkeit zu erreichen.

3.9 Kategorisierung und Nutzung bei Medienservern

Die in diesem Kapitel dargestellten Metadatenformate sollen einen Überblick über aktuell verwendete Formate geben. Zusätzlich wurde versucht, eine gewisse Strukturierung und Kategorisierung der einzelnen Formate zu erzeugen. Durch den fließenden Übergang in der Nutzung einzelner Formate lassen sich manche Formate in mehr als eine Kategorie einordnen.

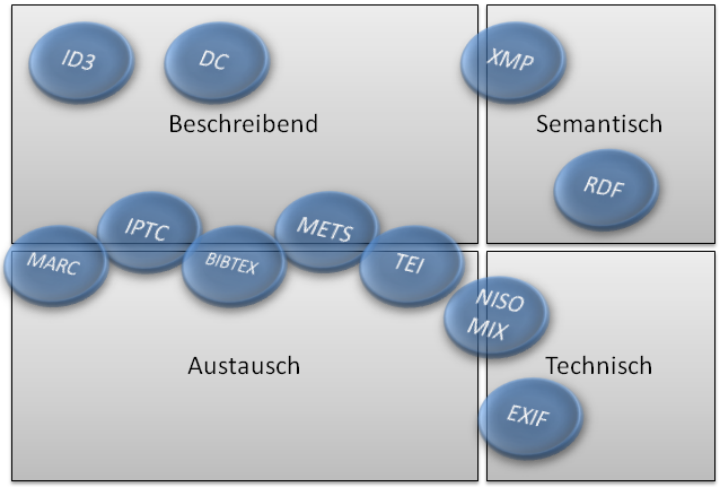

Abbildung 3-6: Kategorisierung der Metadatenformate

Einige der genannten Metadatenformate lassen sich auch parallel nutzen und ergänzen sich gegenseitig. Andere wiederum schließen sich beispielsweise aufgrund ihrer Überschneidung der Speicherposition aus. Somit kann man leider keine generelle Aussage über ein bestimmtes Metadatenformat machen und diese für alle Dateitypen verwenden. Aktuelle Bemühungen zeigen aber, dass von verschiedenen Seiten Entwicklungen vorangetrieben werden, dass eine Vereinheitlichung für den Bereich der Metadaten

erfolgen kann. Zusätzlich wird über die Standardisierung einzelner Dateitypen versucht, die Vielfalt einzudämmen.

Abbildung 3-6 versucht eine Unterteilung in beschreibende, semantische und technische Metadatenformate sowie Austauschformate. Zwischen den beschreibenden Formaten und den Austauschformaten ist der Übergang nahtlos, da für den Datenaustausch die Daten in ähnlicher Form erforderlich sind, wie sie auch bei der Beschreibung verwendet werden können. Gerade die Austauschformate lassen sich oftmals ineinander überführen und unterscheiden sich nur in der verwendeten Syntax.

Medienserver bieten gerade durch die Verarbeitung und Verwaltung von Metadaten zu Objekten entscheidende Vorteile im Umgang mit großen heterogenen Datenbeständen. Für die Verwaltung und zur Kommunikation unterschiedlicher Systeme miteinander nutzen Medienserver alle Bereiche der Metadaten:

- Neben digitalen Objekten werden die beschreibenden Metadaten genutzt, um innerhalb des Bestands Suchfunktionen realisieren zu können.
- Technische Metadatenformate werden eingesetzt, um Objekte und die zugrundeliegenden Dateiformate korrekt darstellen zu können. Eventuell müssen entsprechende Objektinstanzen erzeugt werden, gerade im Grafik- und Bildumfeld.
- Syntaktische und semantische Formate dienen als Container in unterschiedlichen Bereichen der Erzeugung technischer Metadaten oder bei der Kommunikation unterschiedlicher Systeme untereinander.
- Austauschformate und Persistent Identifier bilden die Basis der Interoperation in heterogenen Umgebungen.

Medienserver stellen das Bindeglied zwischen Anwendern und Archiven dar. Es können auch verschiedene Systeme nebeneinander zum Einsatz kommen. In einer heterogenen Umgebung ist es besonders wichtig, dass die Kommunikation und der Informationsfluss vielfältig erfolgen können. Die damit geforderte Flexibilität ist auch im Bereich der Metadaten und der unterstützten Formate entscheidend.

Für jede Anforderung gibt es immer mehr als nur ein einziges ausgezeichnetes Metadatenformat. Gerade dieser Umstand macht anpassbare und erweiterbare Systeme erforderlich. Domänenspezifische Formate sind auf dem Vormarsch, andererseits sind Standardisierungsbemühungen dabei, eine Vereinheitlichung voranzutreiben. Semantische Formate bilden dabei die Basis für den Datenaustausch (METS, XMP). Die Attributdefinition innerhalb dieser Formate ist für unterschiedliche Formate offen (DC, ID3, IPTC).

4 Datenformate/Dateiformate

Dieses Kapitel definiert die Begriffe Datenformat und Dateiformat. Es führt eine Kategorisierung in unterschiedliche Gebiete der Verwendung ein. Zusätzlich werden die gebräuchlichsten Vertreter näher dargestellt und Vorteile und Nachteile herausgearbeitet, die direkten Einfluss auf die Nutzung auf einem Medienserver haben.

4.1 Grundlagen

Computer speichern Informationen in Bitfolgen. Diese Folgen von Nullen und Einsen werden zu Bitströmen zusammengefasst und physisch auf Speichermedien als Datei abgelegt. Die verwendete Syntax und Semantik zur Darstellung der Daten charakterisieren ein Dateiformat. Das Dateiformat stellt eine Abbildung der Information auf den binären Speicher dar. Das Betriebssystem eines Computers stellt anhand des Dateiformats eine Zuordnung zu einer Anwendung her, die die Interpretation der Informationen übernimmt [**15**].

Die Festlegung eines Dateiformats erfolgt in der Regel durch Softwarehersteller oder durch Standardisierungsgremien. Von proprietär spricht man, wenn nur ein einzelner Hersteller das Format unterstützt. Daraus kann sich ein Standard entwickeln, wenn das Format dokumentiert ist und von weiteren Anwendungen aufgegriffen wird. Vor allem im Bereich der Archivierung ist Standardisierung entscheidend für den Erhalt der Lesbarkeit und wird in unterschiedlichen Gremien vorangetrieben. Das Dateiformat kann über unterschiedliche Mechanismen ermittelt werden:

Interpretation des Dateinamens
Eine weit verbreitete aber anfällige Methode der Ermittlung des Dateiformats ist die Interpretation des Dateinamens. Gängige Betriebssysteme nutzen die Dateiendung als Unterscheidungskriterium. Bereits 1973 wurde in CP/M [**80**] die bis heute bekannte 8.3 Konvention eingeführt. Dabei wird der letzte Punkt im Dateinamen als Trennzeichen betrachtet, dem die Dateiendung nachgestellt ist. Die Dateiendung dient als Identifikator zur Identifizierung des Dateityps. Großer Nachteil bei dieser Vorgehensweise ist, dass die Fälschung eines Dateityps sehr einfach ist und nicht erkannt werden kann.

Interpretation des Dateiinhalts
Viele Dateiformate nutzen bestimmte Markierungen innerhalb der Datei zur Charakterisierung des Dateiformats. Dazu müssen allerdings die Datei oder Teile der Datei eingelesen werden und auf gängige Markierungen hin untersucht werden. Diese Markierungen sind über das Betriebssystem mit Anwendungen verknüpft, die die Interpretation übernehmen.

Interpretation der Metadaten
Eine zuverlässige Methode stellt die Auswertung übermittelter Metadaten zu einer Datei dar. Die Metadaten können innerhalb einer Datei liegen oder aber zusammen mit der Datei übertragen werden.

Die Auswertung der Metadaten bietet einen sicheren Schutz vor Fehlinterpretationen. Alle anderen Methoden lassen sich einfach manipulieren. Manipulationen erfolgen gewollt oder ungewollt und richten oftmals großen Schaden an.

Dateiformate können in verschiedene Kategorien zusammengefasst werden. Die Kategorisierung erfolgt durch die Unterscheidung in verschiedenen Schichten. In der physikalischen Schicht dienen die Art der Speicherung der Bitfolgen und der verwendete Zeichensatz als Kriterium. Auf höherer Ebene liefert die Interpretation der Datei ein Unterscheidungskriterium. Aus den vorhandenen Unterscheidungskriterien lassen sich Kategorien bilden, in die die Dateiformate eingeordnet werden können. Viele Dateiformate lassen sich nicht genau einer dieser Kategorie zuordnen. Diese sogenannten Containerformate werden in der Kategorisierung nach ihrem häufigsten Vorkommen einsortiert. Im folgenden Abschnitt werden die weit verbreiteten Dateitypen aus dem Medienserverumfeld betrachtet und gruppiert nach ihrer Kategorisierung auf ihre Eignung hin untersucht.

4.2 Textformate

Textformate sind Dateiformate, die wortbasiert Buchstaben oder Zeichen speichern. Es gibt folgende Unterformate:

- **Unformatierter Text**
  Der Datenstrom besteht rein aus Informationen ohne zusätzliche Steuerzeichen für die formatierte Anzeige.
- **Formatierter Text**
  Zusätzlich zu den Informationen werden innerhalb der Datei Formatierungsregeln gespeichert. Die Formatierung erfolgt über die Interpretation der Datei während der Anzeige mit geeigneter Software.

Bei Medienservern sind Textformate sehr häufig anzutreffen. Die Textdaten kommen aus verschiedenen Bereichen der Forschung und Lehre. Dabei kommt sowohl formatierter Text, als auch unformatierter Text zum Einsatz.

4.2.1 ASCII-Format, Plain-Text

Das ASCII-Format definiert in einem Alphabet den Zeichenvorrat und die dazugehörige binäre Darstellung jedes einzelnen Zeichens des Alphabets. Das einzelne Zeichen wird als binäre Bitfolge kodiert und als Zeichenstrom gespeichert. Durch die 8-Bit Kodierung ist der Zeichenvorrat ASCII-kodierter Datenströme nicht ausreichend, um alle Zeichen gängiger Sprachen abbilden zu können. Neben ASCII gibt es Standardcodes, die deutlich mehr Zeichen aufgrund längerer Bitfolgen abbilden können. Bekannte Vertreter sind EBCDIC, Unicode (ISO/IEC 10646) [**98**] und UCS [**45**]. Mit Unicode, in 16-Bit kodiert, stehen alle Zeichen der wichtigsten lebenden Sprachen zur Verfügung. UCS nutzt eine 32-Bit-Kodierung und kann damit alle bekannten Schriftzeichen abdecken.

So kodierte Dateien werden auch Plain-Text-Dateien genannt, da sie nur die reinen unformatierten Zeichen ohne eine zusätzliche Angabe von Formatierungsvorschriften enthalten. Bekannteste Kodierungen sind an dieser Stelle Kodierungen nach ANSI X3.4-1986 [**10**], ECMA-6 [**31**] oder nach ISO/IEC 646 [**44**].

```
ASCII:    6c 6f 72 65 6d 20 69 70 73 75 6d                                  lorem ipsum
UNICODE: ef bb bf 6c 6f 72 65 6d 20 69 70 73 75 6d                          ï»¿lorem ipsum
UCS:      fe ff 00 6c 00 6f 00 72 00 65 00 6d 00 20 00 69 00 70 00 73 00 75 00 6d   þÿ lorem ipsum
```

Code 4–1: 'lorem ipsum' in unterschiedlichen Kodierungen

Code 4–1 zeigt die Kodierung von ‚lorem ipsum' in unterschiedlichen Standards. Im einfachsten Fall, der ASCII-Kodierung, werden nur die reinen Buchstaben jeweils in 8-Bit kodiert und als Datenstrom gespeichert. Unicode (UTF-8) stellt noch zwei Byte an Steuerinformationen vor den eigentlichen Datenstrom. Die 16-Bit Kodierung von UCS nutzt ebenfalls zwei Byte Steuerzeichen zu Beginn.

Entscheidend für ein korrektes Auslesen des Datenstroms ist ein geeignetes Programm zur Interpretation der Daten. Gerade bei der Verwendung sprachspezifischer Sonderzeichen ist die korrekte Kodierung entscheidend, um Datenverlusten vorbeugen zu können.

4.2.2 Hypertext Markup Language (HTML)

Ein bekanntes standardisiertes textbasiertes Format ist die Hypertext Markup Language (HTML). Im World Wide Web werden Informationen über diese Sprache dargestellt und ausgetauscht. Das World Wide Web Consortium (W3C) entwickelt die Sprache weiter und versioniert sie [**93**].

HTML zeichnet Textteile über spezielle Tags aus und unterscheidet so zwischen Markup und Text. Auf diese Weise erfolgt die Strukturierung des Objekts in HTML-Syntax in verschiedene Elemente. Der Elemente-Satz ist in einer festen Menge definiert, Erweiterungen sind nicht vorgesehen. Elemente können über Attribute näher beschrieben werden. Jedes Element besitzt einen festen Satz Attribute, die optional verwendet werden können. Attributwerte sind einfache Zeichenketten.

HTML verarbeitet neben der reinen Darstellung von Text in speziellen Markups auch externe Datenquellen. Diese Markups können Bilder oder auch Multimediaobjekte sein, die in eigenen Dateien vorliegen. Diese externen Quellen werden über Links in HTML eingebunden. Links ermöglichen zusätzlich Verweise auf andere HTML-Dateien ober bestimmte Stellen innerhalb dieser Dateien. Für die Darstellung liefert HTML Attribute, die eine Formatierung definieren. In Browsern kann die formatierte Version der HTML-Datei angezeigt werden, durch die Einfachheit der Definition ist der HTML-Datenstrom aber auch menschenlesbar und verständlich.

4.2.3 Office-Formate

Im Geschäftsumfeld entstehen täglich große Datenmengen aus unterschiedlichen Bereichen. Drei Basistypen sind dabei weit verbreitet:

- Formatierte Texte oder Dokumente,
- Tabellenkalkulationen und
- Präsentationen.

Besonderes Augenmerk wird bei diesen Formaten auf die Integration der Daten unterschiedlicher Quellen und den Datenaustausch unter den Formaten gelegt. Mehrere Anbieter liefern in sogenannten Office-Suiten Programme, die speziell abgestimmt nahtlos ineinander greifen. Mit diesen Suiten erzeugte Dateien werden häufig ausgetauscht und verbreitet. Gerade durch die Verbreitung ist es notwendig, dass auch die Lesbarkeit und Interpretierbarkeit der Daten sichergestellt wird.

Etabliert haben sich vor allem zwei Anbieter, die nahezu den kompletten Markt bedienen. Das ist einerseits Microsoft mit MS Office und das Open-Source Pendent OpenOffice. Beide Office-Suiten bieten die Möglichkeit, Dateien in XML-Syntax zu speichern und untereinander auszutauschen, MS Office erst seit kurzem. Möglich ist dies durch die Standardisierung und Offenlegung der zugrundeliegenden Formate. Dabei setzen beide Anbieter auf leicht unterschiedliche XML-Dialekte, die nicht komplett ineinander überführbar sind.

Microsoft nutzt Office Open XML (OOXML), das als ISO-Standard 29500 standardisiert werden soll und bereits von der European Computer Manufacturers Association (ECMA) als Standard ECMA 376 anerkannt wurde [**64**]. Dieses Format steht in direkter Konkurrenz zum ISO-Standard 26300, das OpenOffice mit dem Open Document Format (ODF) verwendet. Diese Formate sind flexibel genug, um Daten aus den drei Kernbereichen des Office-Umfelds verarbeiten zu können.

Abbildung 4-1 zeigt die schematische Darstellung des OOXML-Formats. In einem Datei-Container werden in verschiedenen Untercontainern die Bestandteile der Datei gespeichert. Es sind auch Metadaten und Kommentare vorgesehen. Zusätzliche Ressourcen können über einen Link-Mechanismus direkt adressiert werden. Die korrekte Darstellung übernehmen geeignete Interpreter.

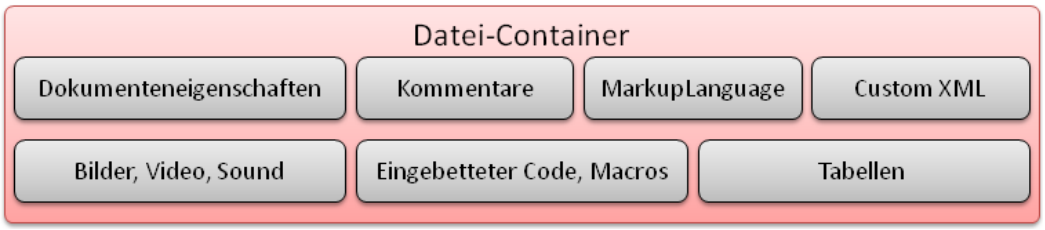

Abbildung 4-1: Office Open XML (OOXML) Format-Architektur

ODF ist in seiner Gesamtstruktur sehr ähnlich aufgebaut. Im direkten Vergleich einzelner Bestandteile fallen die Unterschiede aber klar auf. Beide Office-Suiten bieten über spezielle Erweiterungen die Möglichkeit der Interpretation bzw. Konvertierung des jeweils anderen Formats.

Example document
This has some **bold formatting**, also some *with italics*, a web link

Abbildung 4-2: Beispieltext in formatierter Darstellung

Im Folgenden wird das Beispiel aus Abbildung 4-2 in OOXML und ODF formatiert näher beleuchtet. Es handelt sich um einfache Formatierungsvorschriften, die beide Formate unterschiedlich speichern.

OOXML:
```
<w:p>
    <w:pPr>
        <w:pStyle w:val="Heading1"/>
    </w:pPr>
    <w:r>
        <w:t>
            Example document
        </w:t>
    </w:r>
</w:p>
```

ODF:
```
<text:h text:style-name="P1" text:outline-level="1">
    Example document
</text:h>
```

Code 4–2: Überschriften in OOXML- und ODF-Darstellung im Vergleich

Code 4–2 stellt die Definition einer einfachen Überschrift in beiden Formaten dar. ODF nutzt zur Formatierung verstärkt Attribute innerhalb der XML-Elemente. OOXML bildet die Formatierung über mehrere hierarchische Objekte ab. Die kürzeren Namen in OOXML steigern die Verarbeitungsgeschwindigkeit der Textparser.

Bei komplexer formatiertem Text wird der Unterschied der beiden Formate deutlicher. Code 4–3 nutzt Formatierungsregeln für fett und kursiv und bildet einen Link zu einer externen Ressource ab. Auffällig ist, dass aufgrund der höheren Schachtelungstiefe in OOXML deutlich mehr Objekte und damit auch ein etwas längerer Datenstrom entstehen.

OOXML:
```
<w:p>
    <w:r>
        <w:t>
            This has some
        </w:t>
    </w:r>
    <w:r>
```

ODF:
```
<text:p text:style-name="Standard">
    This has some
    <text:span text:style-name="T1">
        bold formatting
    </text:span>
    , also some
    <text:span text:style-name="T2">
```

```
        <w:rPr>
            <w:b/>
        </w:rPr>
        <w:t>
            bold formatting
        </w:t>
    </w:r>
    <w:r>
        <w:t>
             , also some
        </w:t>
    </w:r>
    <w:r>
        <w:rPr>
            <w:i/>
        </w:rPr>
        <w:t>
            with italics
        </w:t>
    </w:r>
    <w:r>
        <w:t>
            , a
        </w:t>
    </w:r>
    <w:w:hyperlink w:rel="rId4" w:history="1">
        <w:r>
            <w:rPr>
                <w:w:rStyle w:val="Hyperlink"/>
            </w:rPr>
            <w:t>
                web link
            </w:t>
        </w:r>
    </w:hyperlink>
</w:p>
```

```
            with italics
        </text:span>
         , a
        <text:a xlink:type="simple"
              xlink:href="http://www.odfalliance.com">
            <text:span text:style-
                 name="Internet 20 link">
                web link
            </text:span>
        </text:a>
    </text:p>
```

Code 4–3: Formatierter Text in OOXML- und ODF-Darstellung im Vergleich

Für dieses einfache Beispiel ist eine Konvertierung der Formate ineinander möglich, ohne besondere Maßnahmen ergreifen zu müssen. Bei komplexeren Definitionen treten allerdings aufgrund der unterschiedlichen Definition der Formate Probleme auf. Konverter nutzten dafür Routinen, die die Problemstellen erkennen und gesondert verarbeiten. Im Bereich der Tabellenkalkulationen treten vor allem durch unterschiedliche Zahlendefinitionen Probleme auf. Code 4–4 zeigt die Definition eines Wertes in einer Tabellenzelle. Durch die zusätzliche Definition eines Attributes kann der Wert korrekt für Berechnungen herangezogen werden.

OOXML:

```
<c r="C1">
    <v>
        23
    </v>
</c>
```

ODF:

```
<table:table-cell office:value-type="float" office:value="23">
    <text:p>
        23.00
    </text:p>
</table:table-cell>
```

Code 4–4: Zelle C1 mit Wert 23.00

4.2.4 Portable Document Format (PDF)

Seit der Einführung der ersten Version des Portable Document Formats (PDF) 1993 durch Adobe gibt es ein Format speziell für den Datenaustausch. PDF konnte sich aufgrund der Flexibilität und der einfachen Handhabung durchsetzen. Interpreter für alle gängigen Betriebssysteme stehen kostenlos zur Verfügung.

Spätestens seit der ISO Standardisierung 2008 als ISO 32000-1:2008 und der kompletten Offenlegung des Formats kann das PDF-Format als offener Standard betrachtet werden, wenn es um Datenaustausch von Dokumenten, Daten, etc. geht. Die Entwicklung und Erweiterung um neue Funktionalität ist noch nicht abgeschlossen. Diese Erweiterungen sind jeweils in einer Spezifikation festgeschrieben, die in jeder PDF-Datei mit angegeben wird:

PDF 1.0
1992 kommt die erste Version des Portable Document Formats durch das Programm „Carousell", einem Vorgänger von Adobe Acrobat, auf den Markt.

PDF 1.1:
Mit einigen Überarbeitungen und Erweiterungen wird 1993 die Spezifikation 1.1 veröffentlicht. Eine der Neuerungen ist das Rechtesystem, über das Dokumente geschützt werden können. Anmerkungen können direkt mit Farbe, Form und Ersteller versehen, direkt mit abgespeichert werden. TrueType-Fonts lassen sich einbetten und geräteunabhängige Farbräume sind möglich. Auch die ersten Metadaten können hinterlegt werden (z.B. Autor, Titel oder Schlagworte).

PDF 1.2:
1996 wird mit Adobe Acrobat Version 3 die Spezifikation 1.2 veröffentlicht. Darin sind die ersten multimedialen Erweiterungen für Töne und Videosequenzen enthalten. Datenfelder erlauben erstmals die Verbindung von Dokumenten mit externen Daten. Die Beschränkung der Anzahl der Hyperlinks wird aufgehoben.

PDF 1.3:
Mit Erscheinen des Adobe Acrobat in der Version 4 wird im Jahr 2000 die Spezifikation 1.3 veröffentlicht. Vor allem durch die Möglichkeit über ein einfaches Plug-In HTML-Inhalte in das PDF-Format zu transferieren, hat die Anzahl der erzeugten Dateien stark zugenommen. Erstmals können in dieser Version digitale Signaturen erzeugt werden. Bilder in unterschiedlichen Versionen können für verschiedene Zwecke gespeichert werden und der neue Farbraum DeviceN ist integriert. Damit ist die Darstellungen im Hexachrome-Farbraum möglich. Erstmals wird auch JavaScript unterstützt und Seiten werden auf Basis der tatsächlichen Pagina gekennzeichnet [**4**].

PDF 1.4:
Mit der Version 5 des Adobe Acrobat kommt die PDF-Spezifikation 1.4 im Jahr 2001 auf den Markt. Es gibt nur wenige Neuerungen, es sind beispielsweise erstmals Transparenz-Effekte ermöglicht. Diese Funktion bringt allerdings auch neue Herausforderungen mit sich [**5**].

PDF 1.5:
Mit der Spezifikation 1.5 (2003), eingeführt mit Adobe Acrobat in der Version 6, erfolgt die Unterstützung des JPEG2000 Formats für Bilder. Zusätzlich kann man nun auf eine Farbtiefe von 16-Bit zurückgreifen und Layer-Funktionen kommen zum Einsatz, um bei Bedarf beispielsweise Bereiche eines Dokuments ausblenden zu können [**6**].

PDF 1.6:
Mit der Spezifikation 1.6 (2004) gehen größere Änderungen einher (Adobe Acrobat 7). Unter anderem wurde die Seitengrößenbeschränkung aufgehoben und ein PDF-Dokument kann seitdem eine beliebige Seitengröße besitzen. Die Überarbeitung der Schmuckfarben ermöglicht eine Unterscheidung in der Darstellung zwischen CMYK und Schmuckfarbe. Die Unterstützung der OpenType-Fonts ermöglicht die vollständige Einbettung dieser Schriftarten [**7**].

PDF 1.7:
PDF-Dokumente lassen sich seit der Spezifikation 1.7 (2007, ISO 32000), seit Adobe Acrobat 8, zu Paketen zusammenfügen, die aus heterogenen Einzelteilen bestehen können. Zusätzlich wurde die

Steuerung von dreidimensionalen Objekten verbessert und Dateinamen können auf Unicode basieren [**8**].

Neben den verschiedenen Spezifikationen gibt es eine Reihe an Spezialvarianten, die besondere Aufgaben erfüllen:

PDF/X
Hierbei handelt es sich um eine Untermenge an PDF-Dateien, basierend auf dem Standard 1.3 bzw. 1.4, die speziell für die Druckindustrie optimiert wurden. Optimiert meint in diesem Fall, dass gewisse Funktionen innerhalb der PDF-Spezifikation untersagt werden, die ein einheitliches und damit reproduzierbares Ergebnis nach dem Druck eines Dokuments nicht gewährleisten lassen. Das sind beispielsweise multimediale Objekte, die beim Drucken verloren gehen würden oder spezielle Funktionen innerhalb der verwendeten Bildformate. PDF/X liegt inzwischen in unterschiedlichen Versionen vor, die alle nach ISO normiert wurden. Verwendung finden diese speziellen Dokumente vor allem beim Austausch von zu druckenden Daten, z.B. Anzeigen im Zeitungs- und Zeitschriftengewerbe, bei denen es auf eine exakte Wiedergabe ankommt.

Tagged PDF
Die schnelle Anzeige und Navigation innerhalb großer Dokumente ist vor allem bei der Übertragung großer Dokumente sinnvoll. Dafür interessiert meist nicht der komplette Inhalt der Datei, sondern einzelne Bereiche sind von besonderer Bedeutung. Speziell eingeführte Tags (siehe Tabelle 4-1) erleichtern die Navigation innerhalb eines Dokuments. Es handelt sich hierbei um Marken mit direkter Referenz auf den eigentlichen Inhalt, die denen von HTML-Dokumenten sehr ähnlich sind. Über diese Tags kann eine Hierarchie aufgebaut werden, in der eine Navigation möglich ist.

Tag	Name	Beschreibung
<H1> - <H6>	Heading	Überschrift einer bestimmten Ebene
<P>	Paragraph	Textabsatz
<L>	List	Aufzählung
<LI>	List Item	Aufzählungseintrag
<TABLE>, <TR>, <TH>, <TD>	Table, Table Row, Table Header, Table Data	Analog zu HTML
<FIGURE>	Figure	Ähnlich <img> in HTML

Tabelle 4-1: PDF-Tag Auflistung

Linearized PDF
Linearisierte PDF-Dokumente bieten den Vorteil, dass bereits die erste Seite angezeigt werden kann, obwohl das Dokument noch nicht komplett geladen wurde. Gerade bei der Übertragung großer Dokumente über das Internet ist es von Vorteil, dass bereits vor der kompletten Übertragung ein Teil des Inhalts angezeigt wird und der restliche Inhalt kontinuierlich nachgeladen wird. Nach einer Optimierung wird der Inhalt der ersten Seite zu Beginn der Datei gespeichert. In der Spezifikation der PDF-Standards ist nicht sichergestellt, dass die Reihenfolge der Darstellung auch der Reihenfolge der Speicherung der Objektinformationen innerhalb der Datei entspricht, erst bei linearisierten Dokumenten ist das der Fall.

PDF/A
Eine weitere Klasse an PDF-Dokumenten stellen Dokumente konform nach dem PDF/A-Standard dar. Dieser Standard ist speziell für die Langzeitarchivierung entwickelt worden und bildet eine Untermenge der PDF-Spezifikation unter Version 1.4. Nachdem durch Erweiterungen der Spezifikation immer neue Möglichkeiten und flexiblere Einsatzmöglichkeiten von PDF erfolgt sind, haben auch die Probleme bei der Darstellung und vor allem der einheitlichen Darstellung immer mehr zugenommen. Speziell nach den Erweiterungen um Multimediafähigkeiten kann nicht immer sichergestellt werden, dass ein

Dokument auch problemlos den Austausch und die Nutzung an unterschiedlichen Computersystemen erlaubt. In der Zukunft erwartet man gerade an dieser Stelle Probleme. PDF/A existiert im Augenblick in zwei Konformitäts-Ebenen, die jeweils bestimmte Eigenschaften für valide Dokumente garantieren:

PDF/A-1a: Bietet mit Level A sowohl eine eindeutige visuelle Reproduzierbarkeit sowie die Möglichkeit, Text nach Unicode abzubilden und geht auf inhaltliche Strukturen des Dokuments ein.

PDF/A-1b: Bietet mit Level B nur eine eindeutige visuelle Reproduzierbarkeit.

Für die Validierung der Dokumente stehen im Augenblick unterschiedliche Softwarewerkzeuge für die Einordnung in die verschiedenen PDF-Standards zur Verfügung. Aufgrund der Tatsache, dass es sich bei PDF/A um eine Untermenge der Funktionalitäten des aktuellen PDF-Standards handelt, lassen sich nicht alle Dokumente ohne Verluste in das PDF/A-Format umwandeln. Gerade bei neueren Dokumenten treten immer wieder Probleme im Bereich der eingebetteten Bilder oder Grafiken auf. Denn in sehr vielen Fällen werden Transparenzeffekte (z.B. Alphablending) in Bildern verwendet, die bei der Umrechnung entweder verloren gehen oder zu unschönen Farbüberlagerungen führen können. Damit lassen sich zwar einerseits die textuellen Daten lesbar transferieren, andere Informationen können aber verloren gehen.

Aufbau von PDF-Dateien

PDF Dateien sind grundlegend im Aufbau in Untercontainern organisiert (siehe Abbildung 4-3), die über ein Inhaltsverzeichnis zusammengesetzt werden. Die Reihenfolge dieser Container (Header, Body, Cross-Reference Table und Trailer) ist fest definiert.

Abbildung 4-3: PDF-Grobaufbau

Header

Im Header eines PDF-Dokuments werden Informationen über die PDF-Spezifikation gespeichert. Durch die Position zu Beginn an einer fest definierten Stelle können Anzeigeprogramme sehr schnell erkennen, um welchen Dateityp es sich handelt. In Beispiel von Code 4–5 handelt es sich um ein PDF-Dokument in Version 1.4.

```
%PDF-1.4
%ıttttı
```

Code 4–5: Einfacher PDF-Header

Body

Der Body des Dokuments folgt direkt auf den Header und speichert die Daten des Dokuments. Der Body kann weiter in verschiedene Objekte (Sections) unterteilt werden, die z.B. den eigentlichen Inhalt oder Metainformationen enthalten können.

```
693 0 obj
  <<
    /Title(Langzeitarchivierung bei Medienservern)
    /Author(Arne Seifert)
    /Subject(Entwicklung von Strategien zur Langzeitarchivierung am Beispiel eines Medienservers)
    /Keywords(Langzeitarchivierung;LZA;Metadaten;Dateiformat;PDF;TIFF;JPEG;METS;RDF;Semantic Web;PDF/A;)
    /CreationDate(D:20090304083627)
    /ModDate(D:20090304083627)
    ...
  >>
endobj
```

Code 4–6: Beispielobjekt im PDF-Body

Jedes Objekt beginnt mit einer Zahl als eindeutigem Identifikator, der eigentliche Inhalt wird durch obj eingeleitet und mit endobj beendet (siehe Code 4–6). Der Inhalt des Objekts kann variieren. Im Beispiel werden Metainformationen zur PDF-Datei gespeichert.

Cross-Reference Table

Im Anschluss an den Body folgt die Cross-Reference Table (CRT), das Inhaltsverzeichnis der PDF-Datei (siehe Code 4–7). Die Syntax dieses Verzeichnisses ist fest definiert und damit für jedes PDF-Dokument einheitlich. Die Position der CRT speichert der Trailer. Das bietet den Vorteil, dass auf einzelne Objekte innerhalb des Dokuments direkt über ihre Offset-Adresse zugegriffen werden kann, ohne das komplette Dokument bereits zu kennen.

```
xref
0 18
0000000000 65535 f
0000000017 00000 n
0000000339 00000 n
0000000395 00000 n
0000000575 00000 n
0000000792 00000 n
0000000960 00000 n
0000001199 00000 n
0000001435 00000 n
0000001527 00000 n
0000001567 00000 n
0000001618 00000 n
0000001671 00000 n
0000001701 00000 n
0000001916 00000 n
0000049123 00000 n
0000050843 00000 n
0000053516 00000 n
```

Code 4–7: PDF Cross-Reference Table Beispiel

Jede Cross-Reference Table beginnt mit dem Marker xref, gefolgt von einer eigenen Zeile mit der Angabe der Sections innerhalb des Dokuments. Ab der dritten Zeile wird die Offset-Adresse der Section in der Länge von zehn Ziffern angegeben (evtl. mit führenden Nullen), gefolgt von der Generierungs-ID und einer Marke, ob diese Section angezeigt wird oder nicht. Dabei ist jede Zeile exakt 20 Byte lang. Die erste Section stellt einen Sonderfall dar, bei dem die Generierungs-ID immer den Wert 65535 besitzt. Die Reihenfolge der Sections bestimmt die Position innerhalb der CRT.

Trailer
Im Anschluss an die Cross-Reference Table folgt der Trailer. Auch dieser ist in seiner Form fest spezifiziert und beinhaltet die Information, an welcher Offset-Adresse die Cross-Reference Table beginnt.

Der Trailer beginnt immer mit dem Schlüsselwort trailer, gefolgt von Angaben zur Anzahl der Sections, die als Inhalt darzustellen sind. Auf das Schlüsselwort startxref folgt die Offsetadresse der Cross-Reference Table. Beendet werden der Trailer, und damit auch das PDF-Dokument, mit dem Marker EOF (siehe Code 4–8).

```
trailer
<</Size 18/Root 1 0 R/Info 7 0
R/ID[<235D636A74BF0C4B811256479BD2BA02><235D636A74BF0C4B811256479BD2BA02>] >>
startxref
56787
%%EOF
```

Code 4–8: PDF Trailer Beispiel

Die so erhaltene Struktur eines PDF-Dokuments bietet eine ganze Reihe an Vorteilen in der Flexibilität. So müssen beispielsweise die einzelnen Sections des Bodys nicht in der Reihenfolge abgespeichert werden, in der sie auch innerhalb des Dokuments zur Anzeige kommen. Außerdem ist es möglich Sections auszublenden, indem sie nicht im Inhaltsverzeichnis auftauchen. Ein schnelles Laden der Datei wird durch die fest definierten Stellen zu Beginn und am Ende der Datei realisiert, an denen grundlegende Informationen über die vorliegende Datei liegen.

4.3 Grafikformate

Formate zur Speicherung von Bildinformation unterteilen sich im Wesentlichen in drei Hauptkategorien. Diese Kategorien werden aus besonderen Eigenheiten der Grafikformate abgeleitet:

- Vektorformate
- Unkomprimierte Darstellung
- Komprimierte Darstellung.

Den größten Unterschied bilden Vektorformate im Vergleich zu Bitmap oder Pixelgrafiken. Pixelgrafiken produzieren sehr große Datenströme, die über Kompressionsalgorithmen optimiert werden können. Dabei ist eine weitere Unterscheidung nach verlustloser oder verlustbehafteter Speicherung möglich [**57**]. Neben diesen Kriterien spielen noch weitere Faktoren eine Rolle. Das können beispielsweise Patente sein, die eine offene Nutzung verhindern.

Verbreitete Vertreter aus den unterschiedlichen Kategorien werden im Folgenden näher erläutert und Eigenheiten herausgestellt. Die gezeigten Formate kommen häufig im Medienserver-Umfeld zum Einsatz.

4.3.1 Vektorformate

Vektordatenformate stellen eine besondere Klasse der Grafikformate dar, da sie auf spezielle Bedürfnisse einzelner Programme zugeschnitten sind und damit meist proprietär definiert sind. Erst seit der Vernetzung und der flächendeckenden Einführung des Internet haben sich Standardformate herausgebildet, die vor allem für den Datenaustausch optimiert wurden. Am bekanntesten ist dabei SVG, das eine XML-Syntax zur Beschreibung der Bildinformationen nutzt. Da sich SVG-Dateien einerseits im Web-Browser darstellen lassen, andererseits aber auch von Menschen lesbar sind, haben sie sich

verbreiten können. Dennoch bleibt immer noch das Problem, dass Software erforderlich ist, die aus den reinen Textdaten ein Bild darstellen kann.

Bilder basieren auf einfachen Objekten (Pfaden, Kreisen, Punkten, Linien, etc.), die über Koordinaten angeordnet werden können. Vorteil bei der Anzeige ist, dass man diese Bilder ohne Qualitätsverlust auf beliebige Größen vergrößern kann und dabei keine störenden Artefakte oder ähnliches erhält. Im Jahr 2001 wurde das Format durch die W3C in der Version 1.0 als Empfehlung veröffentlicht und ist seitdem offiziell eingeführt [**95**]. Abbildung 4-4 stellt die Grafik als Bild dar, die in Code 4–9 in der Vektorbeschreibung im XML-Format vorliegt.

Abbildung 4-4: SVG in der Bilddarstellung

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" xml:space="preserve"
xmlns:xlink="http://www.w3.org/1999/xlink" width="100%" height="100%" viewBox="1980 352 520 520">

   <title>SVG Logo</title>
 <g>
  <path fill='#000000' d="M1995.59 617.046c0,37.8969 30.7919,68.8247 68.6334,69.0614 -26.5923,26.9246
-26.4966,70.567 0.302128,97.3607 26.8138,26.8138 70.4965,26.8944 97.4161,0.251773 0.176241,37.8617
31.1041,68.6837 69.006,68.6837 37.9271,0 68.87,-30.8674 69.006,-68.7593 26.8944,26.698 70.6022,26.6426
97.4161,-0.176241 26.8038,-26.7987 26.8743,-70.4713 0.22156,-97.3657 37.8818,-0.191348 68.7139,-
31.1343 68.7139,-69.0564 0,-37.9019 -30.7969,-68.8298 -68.6485,-69.0564 26.6074,-26.9196 26.5117,-
70.567 -0.287021,-97.3657 -26.7937,-26.7937 -70.431,-26.8944 -97.3557,-0.302128 -0.236667,-37.8415 -
31.1695,-68.6334 -69.0664,-68.6334 -37.8919,0 -68.8197,30.7818 -69.0664,68.6132 -26.9297,-26.5772 -
70.562,-26.4714 -97.3557,0.32227 -26.7937,26.7937 -26.8994,70.431 -0.312199,97.3607 -37.8365,0.241702
-68.6233,31.1695 -68.6233,69.0614z"/>
…
 </g>
</svg>
```

Code 4–9: Quellcode der SVG-Datei

4.3.2 Bitmap (BMP)/Device Independent Bitmap (DIB)

Das Grafikformat Windows Bitmap (BMP) oder auch device-independent bitmap (DIB) gehört in die Klasse der zweidimensionalen Rastergrafikformate. Entwickelt wurde es vor allem für die Betriebssysteme Microsoft Windows und OS/2. BMP kann in der Version 3, 4 oder 5 vorliegen, am weitesten sind allerdings Dateien der Version 3 verbreitet. Durch das Fehlen einiger gängiger Funktionen modernerer Bildformate, beispielsweise Alphakanäle, Farbkorrektur oder Metadaten, ist der Aufbau der Dateien relativ einfach gehalten. Bilder können in unterschiedlichen Farbtiefen von 1-32 Bit vorliegen und entweder unkomprimiert oder verlustfrei mit RLE-Kompression, einer Lauflängenkompression.

Bei der einfachen RLE-Kompression werden aufeinanderfolgende, gleiche Zeichen, Zahlen oder Buchstaben durch ein einziges Zeichen und die Anzahl der gleichen Zeichen ersetzt. Da bei diesem Verfahren nur unter bestimmten Umständen eine deutliche Kompression erreicht wird, eignet sich das Verfahren in erster Linie für Grafiken oder Bilddateien mit möglichst wenig unterschiedlichen Farben [**57**].

Abbildung 4-5 zeigt den Aufbau einer BMP-Datei. Im Bitmapfileheader werden vier Informationen über die Datei selbst gespeichert, das sind bfType, bfSize, bfReserved und bfOffBits. Die wichtigen Eigenschaften sind bfType und bfOffBits, da sie den Dateityp BMP spezifizieren und die Offsetadresse für den Beginn des Bilddatensegments angeben. In bfSize ist die Dateigröße hinterlegt, dieser Wert ist allerdings nicht immer zuverlässig. Im Bitmapinfoheader (40 Byte) wird eine Reihe an technischen Informationen über das Bild selber abgelegt. Beispielsweise die Bilddimensionen und ob und welcher Kompressionsalgorithmus zum Einsatz kommt. Der Bereich der Color Mask und der Color Palette ist optional und können auch entfallen. Die Größe der Bilddaten lässt sich auf zwei unterschiedliche Arten ermitteln, die von der gewählten Kompression anhängig sind. Bei unkomprimierten Bildern ergibt Höhe*Breite*Farbtiefe/8 die Größe der Pixel Data, alternativ kann der Wert aus dem Bitmapinfoheader herangezogen werden.

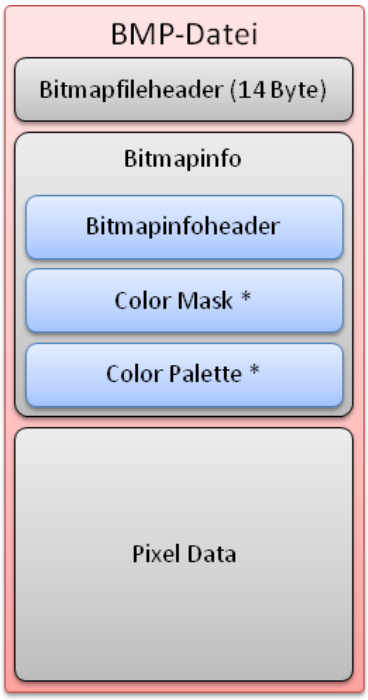

Abbildung 4-5: BMP-Format

4.3.3 Graphics Interchange Format (GIF)

Das Graphics Interchange Format (GIF) ist ein Grafikformat mit verlustfreier Kompression, das auf eine geringe Farbtiefe mit 256 Farben beschränkt ist. Innerhalb einer Datei können mehrere Bilder enthalten sein, die in Webbrowsern auch als Animation genutzt werden können. Das Internet war bei der Einführung von GIF 1987 auch das überwiegende Einsatzgebiet des Formats. Vor allem durch die LZW Kompression [**61**] konnten die Dateigrößen deutlich reduziert werden. Bei dieser Kompressionsmethode werden Zeichenbibliotheken aufgebaut, auf die lediglich verwiesen werden muss. Diese Verweise werden in einem Huffman-kodierten Format gespeichert.

Größter Nachteil ist die Beschränkung auf die indizierte Palette mit maximal 256 Farben. Bei der Betrachtung von Bildern mit höherer Auflösung ist diese Reduktion deutlich zu erkennen. Außerdem ist es bei GIF nicht möglich die Metadaten oder Thumbnails direkt im Objekt zu speichern. Da es inzwischen eine ganze Reihe an moderneren Bildformaten gibt, hat sich auch die Nutzung in Internetseiten stark reduziert.

Dennoch sind drei Vorteile klar hervorzuheben, die zur Verbreitung des Formats gerade im Internet über Webseiten führte:

Interlaced-Funktion: Das Bild wird nicht zeilenweise aufgebaut, sondern es werden verschiedene Schichten nacheinander geladen. Somit ist bereits während des Ladevorgangs der grundlegende Inhalt des Bildes zu erkennen, der bis hin zum endgültigen Bild immer weiter verfeinert dargestellt wird.

Annimationen: Durch die Möglichkeit mehrere Einzelbilder innerhalb einer Datei zu speichern, können einfache Animationen erstellt werden. Die Anzeigedauer jedes einzelnen Bildes kann über ein Zeitintervall gesteuert werden. Bei der Anzeige erfolgt die automatische Weiterschaltung, wahlweise in einer Endlosschleife.

Transparenz: Unter den definierten Farben eines Bildes kann eine Farbe als transparent definiert werden.

Ein großer Nachteil dieses Formats ist die Tatsache, dass die Implementierung des Kompressionsalgorithmus über ein Patent geschützt ist. Es fallen Lizenzgebühren für Programme an, die Dateien im GIF-Format erstellen können [**20**].

4.3.4 Joint Photographic Experts Group (JPEG)

1992 wurde das JPEG Format unter der Norm ISO/IEC 10918-1 vorgestellt. Joint Photographic Experts Group (JPEG) kann sowohl mit verschiedenen Komprimierungs- und Kodierungsmethoden umgehen, als auch verschiedene Farbmodi verarbeiten. In der Norm ist dabei lediglich eine Beschreibung hinterlegt, so dass verlustlose und verlustbehaftete Kompression zum Einsatz kommen kann und sowohl sequenzielle, als auch progressive Farbmodi Verwendung finden können. Am weitesten verbreitet sind allerdings Bilder mit verlustbehafteter Komprimierung und sequenziellen oder progressiven Farbmodi bei 8-Bit-Farbkanälen.

Eine JPEG-Datei ist intern in verschiedene Teilabschnitte untergliedert, die über spezielle Marken gekennzeichnet werden. Dabei folgen auf ein Steuerzeichen (0xFF) als Identifikator die Marke und darauf der Inhalt des eingeleiteten Abschnittes.

Marker	Name	Beschreibung
0xFFC0	SOF0	Baseline DCT
0xFFC1	SOF1	Extended Sequential DCT
0xFFC2	SOF2	Progressive DCT
0xFFC3	SOF3	Spatial lossless
0xFFC4	DHT	Define Huffman table
0xFFC5	SOF5	Differential sequential DCT
0xFFC6	SOF6	Differential progressive DCT
0xFFC7	SOF7	Differential spatial
0xFFC8	JPG	Extension
0xFFC9	SOF9	Extended sequential DCT (AC)
0xFFCA	SOF10	Progressive DCT (AC)
0xFFCB	SOF11	Spatial lossless DCT (AC)
0xFFCC	DAC	Define arithmetic coding conditioning
0xFFCD	SOF13	Differential sequential DCT (AC)
0xFFCE	SOF14	Differential progressive DCT (AC)
0xFFCF	SOF15	Differential spatial (AC)
0xFFD0-0xFFD7	RST0- RST7	Restart 0-7
0xFFD8	SOI	Start of image

0xFFD9	EOI	End of image
0xFFDA	SOS	Start of scan
0xFFDB	DQT	Define quantization table
0xFFDC	DNL	Define number of lines
0xFFDD	DRI	Define restart interval
0xFFDE	DHP	Define hierarchical progression
0xFFDF	EXP	Expand reference component
0xFFE0-0xFFEF	APP0- APP15	Application segment 0-15
0xFFF0-0xFFFD	JPG0- JPG13	Extension 0-13
0xFFFE	COM	Comment

Tabelle 4-2: JPEG-Marker-Übersicht

Tabelle 4-2 führt alle definierten Marker innerhalb der JPEG-Datei auf. Durch die Längenbeschränkung auf zwei Byte ist die Anzahl möglicher Felder begrenzt. Für jeden Marker ist bereits eine Funktion vorhanden, damit werden Erweiterungen schwer. Inzwischen sind bereits Mehrfachbelegungen oder Spezialfunktionen innerhalb einzelner Marker vorhanden. JPEG-Dateien besitzen neben den Markern einen festen Dateiaufbau, der gewisse Marker zwingend voraussetzt (siehe Abbildung 4-6). Vor allem SOI und EOI sind wichtig. Metadaten im Exif-Format werden in der APP1-Sektion standardmäßig abgelegt. Die Sektion DHT zur Speicherung der Quantisierungstabelle ist unbedingt erforderlich, um das Bild für die Darstellung rekonstruieren zu können.

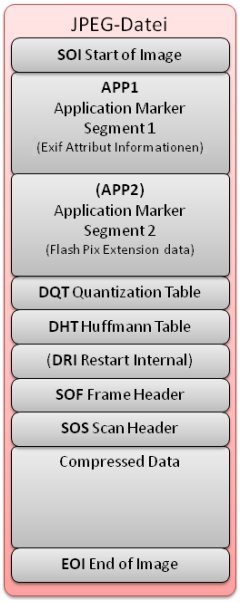

Abbildung 4-6: JPEG Dateistruktur

Das JPEG-Format konnte sich aufgrund des einfachen Kompressionsalgorithmus und dem Gleichgewicht zwischen Dateigröße und Qualität durchsetzen. Die Unterstützung des Formats ist nicht nur auf Computer beschränkt, in vielen anderen Gerätekategorien kommt es standardmäßig zum Einsatz. Digitalkameras nutzen das Format für Fotoaufnahmen.

In der ersten Zeile der Abbildung 4-7 werden die wesentlichen Schritte der Erzeugung einer JPEG-Datei dargestellt. Vor Allem in der Quantisierungsstufe werden die Verluste an Information generiert. Das führt auch dazu, dass bei der Darstellung (untere Zeile) einer JPEG-Datei nicht das Originalbild wiederhergestellt werden kann, sondern nur eine ähnliche Kopie.

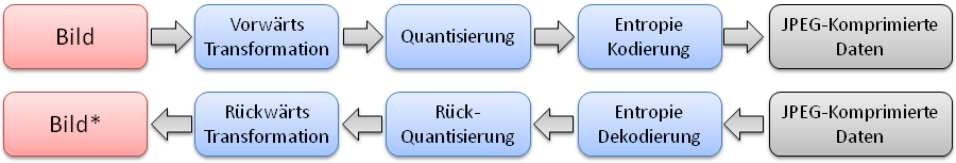

Abbildung 4-7: JPEG-Erzeugung und -Darstellung

JPEG 2000

Bei der Kompression gehen im JPEG-Format Informationen verloren. Durch die Wahl anderer Kompressionsalgorithmen versucht JPEG 2000 (ISO Standard 15444) dem entgegenzuwirken und einige Nachteile von JPEG zu kompensieren:

Die Qualität der Bilder konnte durch eine bessere Komprimierungsrate deutlich gesteigert werden. Der Umstieg auf die Wavelet-Transformation [**76**] [**1**] erzielt diese Verbesserung. Die Einführung zusätzlicher Marker innerhalb des Bildes, die besondere Bereiche als Region of Interrest (ROI) auszeichnet, liefert weitere Qualitätsvorteile. Die ROIs werden gesondert in höherer Auflösung komprimiert. Außerdem führt wiederholtes Öffnen und Abspeichern mit dem gleichen Encoder nicht zu weiterem Qualitätsverlust. Ein komplett neues Feature ist die Unterstützung von Metadaten zusätzlich zu den Exif-Informationen. Metadaten können direkt in das Bildobjekt eingebettet werden. Diese Metadaten werden über ein XML-Schema validiert und können auch einer ROI zugewiesen werden.

4.3.5 Tagged Image File Format (TIFF)

Das Tagged Image File Format (TIFF) wurde 1992 zur Speicherung von Bilddaten entwickelt. TIFF ermöglicht das Ablegen mehrerer Bilder innerhalb einer Daten (Multipage). Deshalb wird das Format auch häufig bei Fax-Software eingesetzt. Die Bilder einer Datei können aber auch unterschiedliche Qualitätsstufen desselben Bildes sein (z.B. Thumbnail, Original, etc.). Die einzelnen Bilder können mit unterschiedlichen Verfahren kodiert sein. TIFF unterstützt sowohl verlustfreie, als auch verlustbehaftete Verfahren und kann beispielsweise auch JPEG-Bilder einschließen. Neben den Bilddaten unterstützt das TIFF-Format die Einbettung von Metadaten im IPTC-Format.

Durch die vielfältigen Möglichkeiten des Formats ist es nicht ohne weiteres möglich, dass alle gültigen TIFF-Dateien auch von allen Grafikprogrammen unterstützt werden. Eine Untermenge bildet deshalb Baseline-TIFF, das auf alle Fälle von Programmen unterstützt werden muss, die als TIFF-fähig gelten. Zusätzliche spezifikationskonforme Bestandteile, werden oftmals nur von Spezialprodukten unterstützt.

Abbildung 4-8 zeigt in schematischer Darstellung die Grundbestandteile einer TIFF-Datei. Der charakteristische 8-Byte große Header (IFH) liefert die Basisinformationen über die TIFF-Datei. Darin enthalten ist der Identifikator „42“ für den Datentyp und die Offset-Adresse des ersten Image File Directories (IFD). Jede TIFF-Datei muss mindestens ein IFD besitzen, das IFD mindestens einen Directory-Eintrag.

Das IFD beginnt mit der Nummer der enthaltenen Einträge, gefolgt von den Offset-Adressen der Directory Entries. Der letzte Wert innerhalb eines IFD ist die Offset-Adresse zum folgenden IFD bzw. „0000“, falls es sich um das letzte Image File Directory handelt. Die Einträge im Directory Entry beginnen mit einem Tag, gefolgt vom Typ des Entries und der Offset-Adresse der eigentlichen Daten zum Eintrag.

Die ersten drei IFDs besitzen besondere Bedeutung. Im ersten sind Primärinformationen über das vorliegende Bild enthalten. Das sind neben der Bildgröße Verweise auf das erste Strip. Strips sind rechteckige Pixelfelder, in die ein Bild zerlegt ist. Die Reihenfolge der einzelnen Strips ist nicht von der Position innerhalb der Datei abhängig. Neben diesen Daten liefert das erste IFD noch die Adresse des Exif- und des GPS-IFD. Im folgenden IFD werden meist die Thumbnail-Informationen abgelegt.

Anhand des Tags kann innerhalb des Directory Entries bestimmt werden, um welche Art von Inhalt es sich handelt. An dieser Stelle erfolgt die Unterscheidung der verschiedenen Levels einer TIFF-Datei. Siehe die Liste der unterstützten Baseline-TIFF-Tags unter Anhang A.b.

Durch die Angabe der Offsetadresse in den Image File Directories, die jeweils mit 32 Bit bei 2 Byte kodiert werden, können maximal Positionen bis zu vier Gigabyte adressiert werden. Diese Einschränkung ist insbesondere bei sehr hochaufgelösten, großformatigen Bildern zu beachten. Immer häufiger erreichen Scans großer Pläne diese Grenze. Die Daten müssen nicht zwingend in linearisierter Form vorliegen, dadurch ist erst nach dem Einlesen der kompletten Datei eine Darstellung möglich. Das ist auch ein Grund dafür, dass das TIFF-Format nicht als Dateiformat für die Bilddarstellung innerhalb von Webseiten geeignet ist oder Verwendung findet.

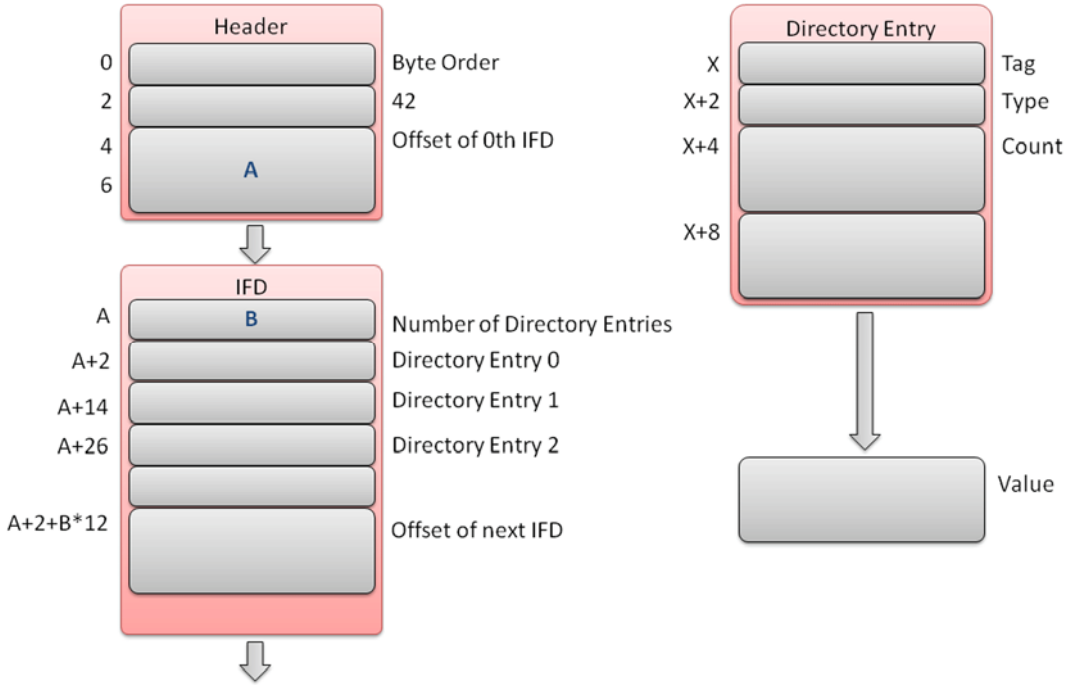

Abbildung 4-8: Schematischer Aufbau des TIFF-Dateiformat

Durch die Möglichkeit von TIFF mit verschiedenen Farbmodellen umgehen zu können, hat sich vor allem im Bereich der Druckvorstufe dieses Format durchsetzen können. Hochaufgelöste und damit druckfähige Bilder werden in der Regel im TIFF-Format angeboten und ausgetauscht. Die Verwendung verlustloser Kompression trägt ein Übriges dazu bei.

Die Einbettung unterschiedlicher Metadaten begünstigt die Verbreitung der Zusatzinformationen direkt mit der Datei. Vor allem im professionellen Umfeld wird diese Funktion benötigt, um beispielsweise Urheberrechte und Ort und Zeitpunkt der Aufnahme zu speichern. Zudem ist die Softwareunterstützung sehr ausgereift und das Format auf die Benutzerbedürfnisse hin optimiert und standardisiert [**9**].

4.4 Audioformate

Audio- und Multimediaformate erfahren neben den reinen Text- und Bildformaten in den letzten Jahren eine starke Erweiterung und Verbreitung. Audioformate bilden das Bindeglied zwischen einem Ton der Musik und der Speicherung als computerlesbares Format. Eine grundlegende Unterscheidung erfolgt in verlustfreie und verlustbehaftete Formate die entweder unkomprimiert oder komprimiert den

Frequenzstrom speichern. Im Medienserverumfeld treten unterschiedliche Audioformate auf. Aufgrund der großen Datenmengen kommen für den täglichen Gebrauch nur komprimierende Formate zum Einsatz, für Archivierungszwecke verlustfreie, unkomprimierte Formate.

4.4.1 WAVE/RIFF

Beim WAVE-Dateiformat handelt es sich um ein Containerformat zur digitalen Speicherung von Audiodaten, basierend auf dem Resource Interchange File Format (RIFF) [**39**], das ursprünglich für das Betriebssystem Windows von Microsoft definiert wurde. Innerhalb des Containers können die unterschiedlichsten Daten gespeichert werden. Ein Identifikator spezifiziert das verwendete Format. Alle möglichen Formate listet Tabelle 4-3 auf, die gebräuchlichsten Vertreter sind dunkel hinterlegt.

ID	Bezeichnung
0x0000	Unknown
0x0001	PCM
0x0002	MS ADPCM
0x0003	IEEE FLOAT
0x0005	IBM CVSD
0x0006	ALAW
0x0007	MULAW
0x0010	OKI ADPCM
0x0011	DVI/IMA ADPCM
0x0012	MEDIASPACE ADPCM
0x0013	SIERRA ADPCM
0x0014	G723 ADPCM
0x0015	DIGISTD
0x0016	DIGIFIX
0x0017	DIALOGIC OKI ADPCM
0x0020	YAMAHA ADPCM
0x0021	SONARC
0x0022	DSPGROUP TRUESPEECH
0x0023	ECHOSC1
0x0024	AUDIOFILE AF36
0x0025	APTX
0x0026	AUDIOFILE AF10
0x0030	DOLBY AC2
0x0031	GSM610
0x0033	ANTEX ADPCME
0x0034	CONTROL RES VQLPC
0x0035	CONTROL RES VQLPC
0x0036	DIGIADPCM
0x0037	CONTROL RES CR10
0x0038	NMS VBXADPCM
0x0039	CS IMAADPCM
0x0040	G721 ADPCM
0x0050	MPEG
0x0069	Xbox ADPCM
0x0200	CREATIVE ADPCM
0x0202	CREATIVE FASTSPEECH8
0x0203	CREATIVE FASTSPEECH10
0x0300	FM TOWNS SND
0x1000	OLIGSM
0x1001	OLIADPCM
0x1002	OLICELP
0x1003	OLISBC
0x1004	OLIOPR
0xFFFF	Experimental

Tabelle 4-3: WAVE-Sample-Formate

WAVE-Dateien nutzen in der Regel die Pulse-Code-Modulation (PCM). Bei diesem Verfahren wird der analoge Signalverlauf mit einer zeitlich konstanten Abtastrate auf eine zeitdiskrete Signalfolge abgebildet. Eine WAVE-Datei besteht aus einer Sequenz von Datenpaketen (Chunks), von denen drei verpflichtend enthalten sein müssen:

RIFF-Chunk:
Dieses Datenpaket kennzeichnet die Datei eindeutig als WAVE-Datei. Enthalten sind dabei die chunkID (RIFF), die Dateigröße und der riffType (WAVE), siehe hierzu auch Code 4–10.

```
52 49 46 46 24 5a 04 00 57 41 56 45            RIFF$Z  WAVE
```

Code 4–10: 12-Byte Riff-Chunk

Format-Chunk:
Auf die 12 Byte des RIFF-Chunks folgt direkt als erstes Sub-Chunk der 24-Byte lange Format-Chunk. Eingeleitet wird der Format-Chunk, siehe Code 4–11, mit der chunkID (fmt), gefolgt von Informationen über die Kompressionsart (1=PCM), die Kanalanzahl (2=Stereo), Abtastrate und weiteren Daten für die Wiedergabe.

```
66 6d 74 20 10 00 00 00 01 00 02 00 44 ac 00 00 10 b1 02 00 04 00 10 00        fmt        D¬   ±
```

Code 4–11: 24-Byte Format-Chunk

Data-Chunk:
Der Data-Chunk wird über die chunkID data eingeleitet. Darauf folgen die Größe des Chunks und die eigentlichen Daten.

Die Unterstützung für Metadaten ist über das Datenpaket „INFO“ möglich. Einige Programme schreiben während der Erzeugung Daten in diesen Chunk, nur gibt es dafür noch keinen Standard. Somit ist nicht sichergestellt, dass Informationen immer an dieselbe Stelle geschrieben werden. Dadurch können Konflikte mit bereits vorhandenen Daten entstehen [**67**].

4.4.2 MPEG-1 Audio Layer3 (MP3)

Das Format MPEG-1 Audio Layer3 (MP3) ist ein Datenformat zur Speicherung digitaler Audiodaten. Dabei kommt ein Kompressionsalgorithmus zum Einsatz, der über die Psychoakustik Audioinformationen aus dem Audiostream entfernt, die für das menschliche Ohr nicht wahrnehmbar sind. Durch diese Methode kann eine relativ hohe Datenreduktion erzielt werden, bei gleichzeitig nur sehr geringer subjektiver Beeinträchtigungen der Audioqualität.

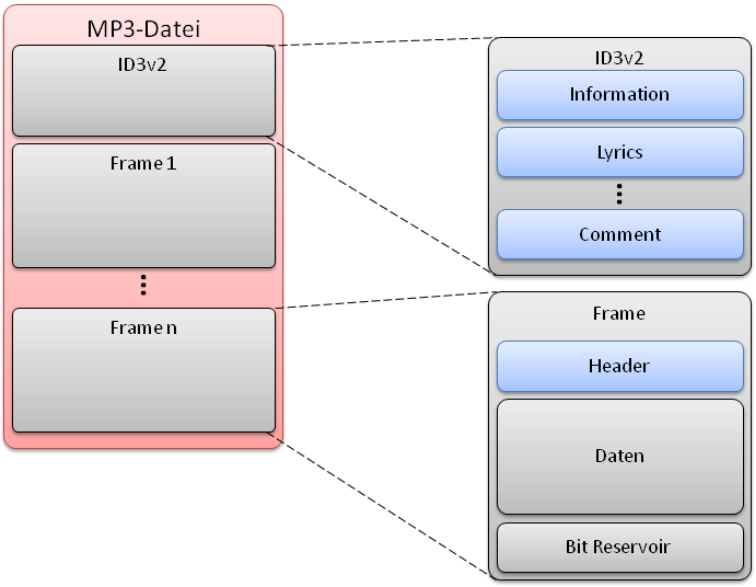

Abbildung 4-9: MP3-Dateistruktur

Entwickelt wurde das Format am Frauenhofer Institut für integrierte Schaltungen zusammen mit der Friedrich-Alexander Universität in Erlangen-Nürnberg ab 1992. Zu Beginn waren keine Möglichkeiten vorgesehen, Metadaten innerhalb der MP3-Datei zu speichern. Erst durch die Erweiterung um Tags (ID3Tags) wurde es ab ID3v1 möglich, eine sehr begrenzte Anzahl von zusätzlichen Metadaten am Beginn der Datei einzubetten. Dabei waren die Einträge auf eine feste Anzahl an Attributen mit einer fest vorgegebenen Länge von 128 Bytes begrenzt (siehe Tabelle 3-3).

Metadaten können ab ID3v2 flexibel am Beginn der Datei abgelegt werden. Durch die Erweiterung um zusätzliche Felder und die Aufhebung der festen Längen der Felder werden alle grundlegenden Informationen unterstützt. Für eine vollständige Liste siehe Anhang A.c. Version 2 ist in weitere Unterversionen gegliedert, die erst ab Version 2.4 eine Kodierung nach UTF-8 nutzen.

Auf die Metadaten folgen in Frames organisiert die eigentlichen Audiodaten (siehe Abbildung 4-9). Die Frames sind in einem Header organisiert, gefolgt von den eigentlichen Daten. Der Frame-Header ist 32 Byte lang und speichert grundlegende Angaben zu Kompressionsverfahren, Bitrate, Samplingfrequenz und die Mode-Extension (z.B. Mono oder Stereo). Im Datenblock sind die komprimierten Informationen über Frequenz und Amplitude gespeichert.

Im Augenblick stehen für die Verwendung von MP3 zwei Encoder zur Verfügung, ein lizenzpflichtiger der Frauenhofergesellschaft [**58**], und der freie LAME-Encoder [**84**], der aus einem Open-Source-Projekt entstanden ist. Zur Dekodierung der MP3-Daten existieren zahlreiche Decoder.

4.5 Videoformate/Videocodecs

Eine Erweiterung des Audioformats stellt das Videoformat dar, bei dem Tonspuren parallel zu bewegten Bildern ablaufen. Für die Darstellung eines Videos ist dabei die Interpretation zeitrelevanter Informationen für eine korrekte Wiedergabe von entscheidender Bedeutung. Gerade durch die Forderung, dass die Zeit als entscheidende Stellgröße mit in die Wiedergabe hineinspielt, ist erst in den letzten Jahren durch geeignete Hardware eine immer bessere Qualität bewegter Bilder möglich.

Die Framerate steuert die Anzahl der Bilder innerhalb eines fest vorgegebenen Intervalls. Nur so ist sichergestellt, dass die Abspielgeschwindigkeit nicht allein von der Leistung des Abspielgeräts abhängt. Angegeben wird die Framerate in Frames Per Second (FPS). Neben den Videodaten ist das synchrone Abspielen der Tonspur zum Bild wichtig. Videoformate nutzen unterschiedliche Verfahren der Synchronisation über Multiplexer.

Als Unterscheidungskriterium der Videoformate dienen die Kategorien:

- Containerformate,
- Komprimierende Formate und
- Unkomprimierte Formate.

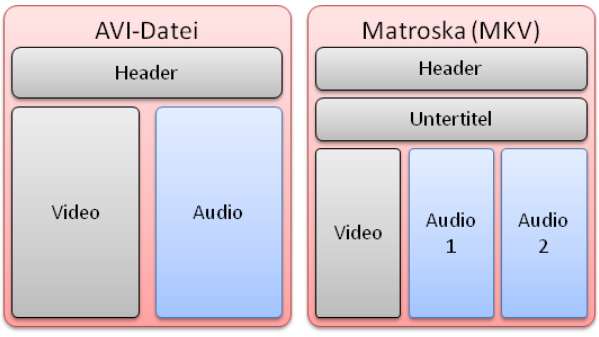

Abbildung 4-10: Schematischer Aufbau von Video-Containerformaten

Moderne Videoformate nutzen Containerformate, unter denen die unterschiedlichen Bestandteile eines Films zusammengefasst werden (siehe Abbildung 4-10). Die Bestandteile der Container bearbeiten unterschiedliche Algorithmen, die sowohl komprimieren können, als auch oder den Originaldatenstrom beibehalten.

Die Bestandteile der Containerformate können in unterschiedlichen Definitionen vorliegen. Nicht jedes Containerformat unterstützt alle gängigen Codierungsverfahren. Tabelle 4-4 liefert eine Gegenüberstellung möglicher Verfahren gegliedert nach Bestandteilen.

	AVI	**Matroska**	**MP4**
MPEG-4 Video			
DivX, Xvid	Ja	Ja	Ja
H.264	Nein	Ja	Ja
Audio			
MP3	Ja	Ja	Ja
Vorbis	Nein	Ja	Nein
AAC	Teils	Ja	Ja
AC-3, DTS	Nein	Ja	Nein
Untertitel			
SubRip	Teils	Ja	Ja
Vobsub	Nein	Ja	Nein

Tabelle 4-4: Unterstützte Formate der Container

Innerhalb des verwendeten Containerformats ist der eingesetzte Codec entscheidend für die Dateigröße und die Qualität des entstehenden Videos. Als Codec wird der Algorithmus bezeichnet, der die Daten kodiert bzw. dekodiert. Typisches Merkmal aktueller Codecs ist die Forderung nach einem schnellen Dekodiervorgang. Der Enkodiervorgang hingegen kann lange dauern, er muss nur einmal ausgeführt werden. Der Enkoder soll über geeignete Kompression die Dateigröße möglichst gering halten (verlustfrei oder verlustbehaftet). Meist werden verlustbehaftete Codecs eingesetzt, die die Qualität der erzeugten Videodaten möglichst hoch halten (siehe Tabelle 4-5).

Format	**Beschreibung**	**Größe**	**Qualität**
RAW	umcodiert, unkomprimiert	--	++
MPEG-1	MPEG-1 kodiert	+	-
MPEG-2	MPEG-2 kodiert	+	+
MPEG-4	XVID kodiert	++	+

Tabelle 4-5: Codecübersicht mit Vergleich Qualität/Größe

Über Medienserver werden immer häufiger Videos angeboten. Daneben ermöglichen zahlreiche Internet-Portale das einfache Veröffentlichen kurzer Videofilme. In diesem Bereich hat in den letzen Jahren ein enormes Wachstum stattgefunden.

4.5.1 Audio Video Interleave (AVI)

Das am weitesten verbreitete Containervideoformat Audio Video Interleave (AVI), wurde von Microsoft entwickelt und vom RIFF-Format abgeleitet (siehe Abbildung 4-11 und Kapitel 4.4.1). In diesem Format sind die Audio und Video-Daten ineinander verzahnt, bei der Einführung war es in erster Linie für kurze Videoclips gedacht. AVI unterstützt unterschiedliche Kodierungsverfahren, das Videoformat wird über den Four Character Code (FourCC) [**91**] spezifiziert, das Audiosignal über TwoCC. 1996 brachte die Firma Matrox die Erweiterung OpenDML [**66**] auf den Markt und damit wurde AVI in der Version 2.0 eingeführt.

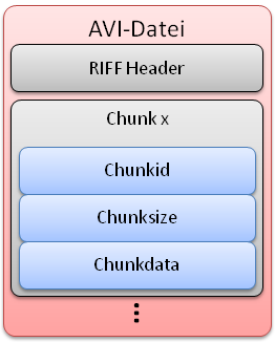

Abbildung 4-11: AVI-Dateiaufbau

```
RIFF ('AVI '
      LIST ('hdrl' ... )
      LIST ('movi' ... )
      ['idx1' (<AVI Index>) ]
     )
```

Code 4–12: AVI-Datei Beispiel

Der Inhalt aus Beispiel Code 4–12 definiert die beiden Pflicht-Chunks, *hdrl* und *movi* die dem Header folgen. In diesen beiden Listen-Chunks wird einerseits das Datenformat definiert und andererseits werden die Daten der AVI-Sequenzen gespeichert. Gefolgt von der Liste *idx1*, die den Index beinhaltet. Die Video und Audio-Daten können mit einer ganzen Reihe unterschiedlicher Encoder kodiert werden. Dabei ist aber nicht jedes Audio-Format geeignet (siehe Tabelle 4-4). Im ersten Sub-Chunk hdrl werden rudimentäre Metadaten zum Video hinterlegt. Dort finden sich Angaben zu Videobreite, -höhe und Framerate. Weitere Metadaten sind bei AVI-Dateien nicht vorgesehen.

Da es sich bei AVI um ein schon relativ altes Format handelt, sind inzwischen aktuellere Container für Videoaufzeichnung vorhanden, beispielsweise Matroska.

4.5.2 Matroska (MKV)

Matroska (MKV) [**22**] ist ein neues Containerformat mit dem Anspruch, das AVI-Format zu ersetzen. Durch eine breite Palette unterstützter Audio-, Video- und Untertitelformate ist es äußerst flexibel einsetzbar. Zusätzlich unterstützt Matroska Metadaten und Attachments und kann mehrere Dateien miteinander verlinken. Nachteilig wirkt sich diese Flexibilität auf die Unterstützung der Abspielhardware aus, die sich im Moment auf den Computer beschränkt.

Die Beschreibung des Containers erfolgt im binären XML-Format EBML. Dadurch sind eine Weiterentwicklung und die Ergänzung weiterer Funktionen möglich, bei gleichzeitiger Erhaltung der Lesbarkeit durch ältere Parser. Matroska sieht vor, dass die Daten in Kapiteln vorliegen und mehrere Tonspuren vorhanden sein können. Die einzelnen Kapitel sind unabhängig voneinander und nur über Links miteinander verbunden. Dadurch sind unterschiedliche Kodierung und Formate möglich. Unterstützt wird auch das Streaming der Daten.

4.5.3 Flash Video (FLV)

Adobe entwickelte mit Flash Video (FLV) ein proprietäres Containerformat, das als Videocodec auf die MPEG-Verfahren H.264 [**47**] oder eine Abwandlung von H.263 aufsetzt [**68**]. Zusätzlich können noch weitere Verfahren zum Einsatz kommen. Im Audiobereich wird in erster Linie MP3 verwendet. Aber auch hier sind andere Kodierungen möglich. Bei den Video-Codecs bestimmt die gewählte Abspielsoftware die unterstützten Formate. Bis Flash7 ist das nur der Sorenson-Codec [**51**], eine H.263-Codec-Variante. Erst ab Flash 8 und 9 kommen VP6 (von On2) und H.264 hinzu.

Vor allem durch die massive Nutzung im Internet erfolgt eine weite Verbreitung des Formats, da es über ein kostenloses Plugin in allen gängigen Browsern abgespielt werden kann.

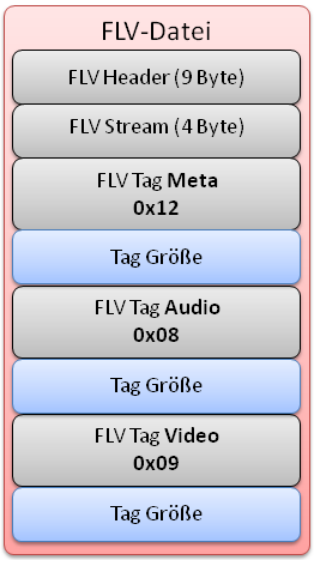

Abbildung 4-12: FLV Datei Aufbau

Die Dateistruktur ist bei FLV klar definiert (siehe Abbildung 4-12). Auf einen 9-Byte großen Header folgen Stream-Informationen und drei Container, in denen die Metadaten (Meta), die Audiodaten (Audio) und die Videodaten (Video) abgelegt werden. Diese Tags besitzen jeweils eine Größenangabe zur einfachen Erkennung durch die Abspielsoftware.

Im Meta-Tag werden folgende Metadaten gespeichert (siehe Tabelle 4-6), die die Abspielsoftware interpretieren kann, ohne das komplette Video lesen zu müssen. Vor allem die videocodecid ist entscheidender Bestandteil, damit eine korrekte Wiedergabe gewährleistet ist.

Tag	**Beschreibung**
audiocodecid	Codec-ID des Audiocodecs (z.B. 0=unkomprimiert)
audiodelay	Zeitversatz des Audiostreams zum Videostream in Sekunden
audiosize	Größe des Audioteils (Audio Tags) in Byte
canSeekToEnd	True, wenn letzter Video-Tag ein Key-Frame ist
creationdate	Erzeugungsdatum
datasize	Größe des Datenteils (Data Tags) in Byte
duration	Länge des FLV-Videos in Sekunden
filesize	Dateigröße absolut in Byte
framerate	Frame-Rate
height	Höhe des Videos in Pixel
lastkeyframetimestamp	TimeStamp des letzten Key-Frame Video-Tags in Sekunden
lasttimestamp	TimeStamp des letzten Tags innerhalb des Videos in Sekunden
metadatacreator	Programm, das die Metadaten erzeugt hat
metadatadate	TimeStamp der Metadatenerzeugung
videocodecid	Codec-ID des Videocodecs (z.B. 2=Sorenson H.263)
videodatarate	Video-Datenrate

videosize	Größe des Videoteils (Video Tags) in Byte
width	Breite des Videos in Pixel
xtradata	Extra-Feld für Metadaten

Tabelle 4-6: FLV Metadaten Tags

4.5.4 MPEG, MP4

Unter der Moving Picture Experts Group (MPEG) haben sich Experten zusammengeschlossen, die sich mit der Standardisierung verschiedener Algorithmen und Etablierung im Multimediabereich befassen. Daraus resultieren einige Standardkompressionsverfahren und die dazugehörigen Containerformate, die auf Audio- und Videodaten angewendet werden.

1990 wurde der H.261 Algorithmus eingeführt und drei Jahre später zu MPEG-1 mit dem dazugehörigen Audioformat MP3 (MPEG-1 Layer 3) standardisiert. Verwendung finden diese Verfahren vor allem bei der Video-Telefonie und Video-CDs.
MPEG-2 (H.262) kommt in den Jahren 1994 und 1995 für Ton- und Videoaufnahmen für DVD sowie DVB zum Einsatz.
Das für HDTV gedachte MPEG-3 wird zwar mit den dazugehörigen Codecs H.263 und Abwandlungen definiert, kommt aber nie zum Einsatz, da MPEG-2 mit kleineren Erweiterungen ausreichend ist.
Zwischen 1998 und 2001 wird MPEG-4 [**42**] entwickelt, das eine deutlich bessere Videokompression beinhaltet und so in einer Reihe von Containerformaten zum Einsatz kommt [**11**][**73**][**90**].

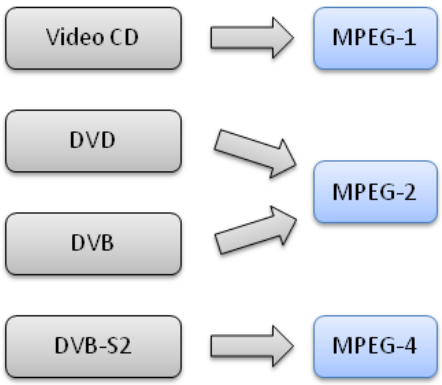

Abbildung 4-13: MPEG-Versionen und Einsatzgebiet

Durch die Definition von Containerformaten und Verfahren zur Audio- und Videospeicherung, können MPEG-kodierte Streams auch innerhalb anderer Containerformate eingebettet werden. Abbildung 4-13 zeigt die Haupteinsatzgebiete der Kodierungsverfahren nach MPEG.

MP4-Dateien nach ISO/IEC 14496-14 untergliedern sich in Objekte, Atome genannt. Das Basisobjekt bildet das movie-Objekt. Darunter liegen track- und media-Objekte, die ihrerseits weiter untergliedert werden (siehe Abbildung 4-14). Jedes Atom besitzt einen eindeutigen Header zur Identifikation und eine Größenangabe, anhand derer innerhalb der Datei navigiert werden kann. Metadaten können in einem eigenen Atom (minfo) abgelegt werden.

In der Spezifikation von MPEG bis Version 4 waren keinerlei Metadatendefinitionen vorgesehen. 2002 wurde mit MPEG-7 eine spezielle Version verabschiedet, die keine Datenkompression beschreibt, sondern eine Metadaten-Definitions-Sprache, mit deren Hilfe Metadaten zu MPEG-Dateien abgelegt werden können [**86**].

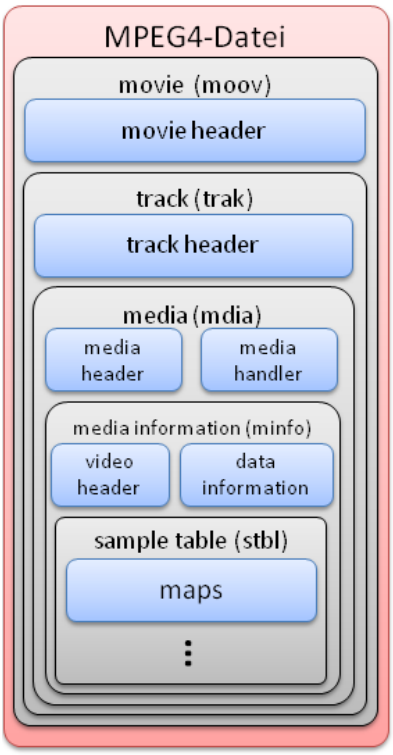

Abbildung 4-14: Schematische Darstellung des MPEG4-Dateiformat

Gerade die Neuerungen von MPEG-4 führten zu einer weiten Verbreitung innerhalb des Videoumfelds:

Media Objects
Der MPEG-4-Standard ist komplett objektbasiert realisiert und so bestehen audiovisuelle Szenen aus Unterobjekten, die in einer hierarchischen Struktur angeordnet werden können:

- Unbewegte Bilder (z.B. als Hintergrund für Szenen),
- Video-Objekte (z.B. bewegte Personen im Vordergrund) und
- Audio-Objekte (z.B. die Stimme der Person).

Sprite Coding

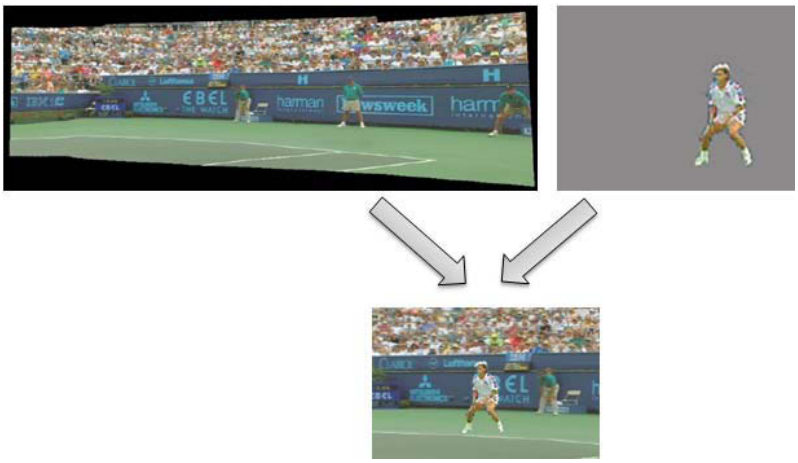

Abbildung 4-15: Funktionsweise Sprite-Coding

Bei der Enkodierung werden Vorder- und Hintergrund getrennt voneinander verarbeitet, so lässt sich auf effiziente Art der Speicherplatz optimieren. Das Hintergrundpanorama (Sprite) wird als erster Frame zum Client übertragen, kann dort im Puffer verbleiben und wird nur mit Kamerasteuerungsinformationen angereichert. Der eigentliche bewegte Teil wird im Vordergrund frameweise eingeblendet und kann separat übertragen werden. Die entstehende Videoszene wird erst beim Empfänger erzeugt [**52**] (siehe Abbildung 4-15).

4.5.5 DIVX/XVID

Digital Video Express (DIVX) [**97**], ist ein Videoformat, das Anfang der 90er Jahre aus dem MPEG-4-Codec entstanden ist. Das von Microsoft entwickelte MPEG-4 wurde von einem französischen Hacker aus dem Windows Media Player extrahiert und um ein paar Funktionen erweitert (Divx3 oder DivX:-)). Dazu erfolgte die Parallelentwicklung durch DivX Networks, Inc die Fertigung des DivX-Codec. Beiden gemein ist ein guter Kompressionsalgorithmus, der den Inhalt einer DVD ohne große Qualitätseinbußen auf eine CD reduzieren lässt.

Aktuell liegt Version 6.8 des DIVX-Codecs vor, der immer noch weiterentwickelt wird. Diese proprietäre Entwicklung liegt nicht quelloffen vor und ist deshalb inzwischen umstritten. Zusätzlich ist noch das integrierbare Rechtemanagement ein Hindernis durch die Reglementierung des Abspielens.

Als Gegenbewegung kann der Videocodec XVID aus der Open-Source-Gemeinde gesehen werden, ein Anagramm zu DIVX. Aus dem zeitweise quelloffenen Softwareprojekt OpenDivX entstand ein eigener Codec, der einerseits quelloffen unter der GPL veröffentlicht wurde und andererseits nahezu die Qualität von DIVX erreichen kann.

Zur Kodierung der Daten nutzen beide Formate den Kompressionsalgorithmus MPEG-4 Part 2 Video Coding [**33**] und kodieren die Inhalte im Advanced Simple Profile (ASP). Dabei werden die Audio-Informationen mit MP3 oder AC-3 für Mehrspuraufnahmen komprimiert. Seit der aktuellen Version wurden in DIVX nun spezielle Metadatentags integriert, die die Speicherung von Metadaten in sog. XTAG-Video-Tags erlauben.

4.6 Gegenüberstellung der Formate

Die zu Beginn dieses Kapitels vorgestellten Datenformate können im Hinblick auf einige Eigenschaften gegenübergestellt werden. Die Stärken (+) und Schwächen (-) der einzelnen Formate lassen sich bewerten und daraus können Rückschlüsse für die geeignetste Nutzung und das Einsatzgebiet gezogen werden. Die betrachteten Kriterien und Eigenschaften orientieren sich an Bedingungen, die ein Medienserver an Dateiformate hinsichtlich der Nutzung stellt. Neben der Nutzung der Formate mit einem Medienserver werden Forderungen der Archivierung (siehe auch Kapitel 5) betrachtet. Nicht alle Kriterien können gleich gewichtet werden, manche Forderungen sind sinnvoll, andere unabdingbar. Für alle Formate kann lediglich die Betrachtung der Unterstützung für Metadaten und die freie Verfügbarkeit oder Standardisierung als entscheidend angesehen werden. Die Betrachtung der Dateiformate erfolgt in der in diesem Kapitel eingeführten Unterscheidung nach Text-, Bild-, Audio- und Videoformaten, die das komplette Spektrum eines Medienservers abdecken.

Textformate
Beim Vergleich verschiedener Textformate fällt auf, dass PDF bei den verwendeten Kriterien eindeutig am besten abschneidet (vergleiche Tabelle 4-7). Da es in der Regel nicht als Datenformat für die Bearbeitung gedacht ist, sondern meist nur als finales Datenformat zum Einsatz kommt, fällt an dieser Stelle auch die negative Beurteilung bei der Editierbarkeit nicht sonderlich ins Gewicht. Die meisten anderen Textformate lassen sich nach PDF transferieren. Vor allem die offengelegte Spezifikation des

Formats und die Unterstützung von Metadaten liefern entscheidende Vorteile gegenüber den anderen Formaten.

Eigenschaft	Plain-Text-Formate	HTML	Office-Formate	PDF
Dateigröße	+	+	-	++
Kompression	-	+	+	++
Editierbarkeit	++	++	+	-
Standardisierung	-	+	+	++
Metadaten	-	+	+	+

Tabelle 4-7: Textformate im Vergleich

Bildformate

Bei Grafikformaten sind besonders die Eigenschaften der Standardisierung und der Metadaten von Bedeutung, die Dateigröße wird eher als sekundär angesehen. Im Bereich der Archivierung kommt es in erster Linie auf die Möglichkeit an, Daten verlustlos zu erhalten und die Dateien über Metainformationen eindeutig zuordnen zu können, da Bildinhalte schwieriger zu erfassen sind, als beispielsweise Textdaten.

Eigenschaft	Vektorformate	BMP	GIF	JPEG	TIFF
Dateigröße	-	-	++	++	-
Kompression	++	++	+	+	+
Verlustfrei	++	++	--	-	+
Standardisierung	-	+	+	+	++
Metadaten	Nicht vergleichbar	--	-	+	++
Eindeutige Darstellung	-	++	++	-	+

Tabelle 4-8: Bildformate im Vergleich

Durch die Vielzahl der vorhandenen Bildformate konnte nur eine sehr kleine Auswahl der gebräuchlichsten miteinander verglichen werden. Im Vergleich bringen vor allem das TIFF- und das JPEG-Format die meisten positiven Eigenschaften mit. Bei JPEG ist die Dateigröße verglichen mit der möglichen Qualität besonders hervorzuheben, neben der Standardisierung und der Unterstützung von Metadaten. Lediglich die eindeutige Darstellung ist nicht gegeben. TIFF bringt mit der großen Flexibilität viele Vorteile mit, die sich positiv auf den Datenerhalt auswirken. Verlustfreie Kompression liefert eine eindeutige Darstellung. Nur die Dateigröße ist aufgrund der verwendeten Algorithmen nicht unerheblich. Metadaten werden ebenfalls unterstützt. Gerade diese Eigenschaften lassen die beiden Grafikformate sehr universell einsetzbar werden (siehe Tabelle 4-8).

Audioformate

Eigenschaft	WAVE	MP3
Dateigröße	-	+
Kompression	+	+
Verlustfrei	++	-
Qualität	++	+
Standardisierung	++	+
Eindeutige Wiedergabe	++	-
Metadaten	+	++

Tabelle 4-9: Audioformate im Vergleich

Bis auf die Dateigröße, bei WAVE-Dateien deutlich größer als bei MP3 Dateien, bietet das WAVE-Format einige Vorteilen gegenüber dem MP3-Format. Gerade beim Erhalt der Originalinformation ist WAVE eindeutig die bessere Wahl. Bei der Betrachtung wurden andere Formate (beispielsweise OGG) außen vor gelassen, weil ihre Verbreitung im speziellen Umfeld erfolgt. In der Unterstützung von Metadaten

bietet MP3 mit der Definition fester Metadatenfelder eine sehr hohe Flexibilität und umfassende Unterstützung. Bei WAVE ist diese Unterstützung sehr viel rudimentärer. Die Bewertung beider Formate hinsichtlich der resultierenden Qualität ist aufgrund unterschiedlicher Kompressionsverfahren und Abtastraten nicht eindeutig möglich. Bei MP3 steht die Qualität in direktem Zusammenhang mit der Dateigröße, je höher die Qualität, desto größer die Datei. Insgesamt überwiegen aber die Vorteile des WAVE-Formats, gerade durch die Möglichkeit verlustlos Audioinformationen über variable Algorithmen kodieren zu können (siehe Tabelle 4-9).

Videoformate

Ähnlich den Audioformaten handelt es sich auch bei den Videoformaten um eine noch relativ junge Sparte innerhalb der Informatik. Bei Videoformaten erkennt man das vor allem daran, dass Formate mit den unterschiedlichsten positiven und damit teilweise negativ verbundenen Eigenschaften existieren, die ständig optimiert und weiterentwickelt werden. Dennoch bringen einige Formate gewisse Allround-Fähigkeiten mit, die sie von der Konkurrenz abheben. Für eine Einordnung erfolgt zusätzlich eine Unterscheidung nach Containerformat und Codec.

Eigenschaft	AVI	Matroska	MP4	FLV
Dateigröße	-	++	+	++
Kompression	+	++	++	++
Qualität	-	++	++	-
Standardisierung	+	+	++	+
Metadaten	-	++	++	+

Tabelle 4-10: Videocontainerformate im Vergleich

Bei den Containerformaten (siehe Tabelle 4-10) schneidet das MP4 Format am besten in der Gegenüberstellung ab, da die Standardisierung gegenüber der Dateigröße wichtiger einzustufen ist. Das ältere AVI-Format bietet viel weniger Funktionsumfang und Flexibilität, als die übrigen Formate. Das FLV-Format bildet eine Ausnahme, weil es für einen ganz anderen Zweck entwickelt wurde. FLV ist für die Verbreitung über das Internet im Browser aufgrund der sehr kleinen Dateigröße in Bezug auf die Qualität besonders geeignet. Bei online-Filmsammlungen (beispielsweise YouTube oder MyVideo) findet man ausschließlich FLV-Videos.

Bei den Videoformaten in Tabelle 4-11 können sich die unterschiedlichen Codecs der MPEG-Versionen gegenüber DIVX/XVID behaupten. Die Unterstützung von DIVX bei Geräten zur Wiedergabe der Videos ist nicht immer gegeben. Aufgrund der fehlenden Standardisierung und des Fehlens offenen Quellcodes können Probleme auftreten, die die korrekte Interpretation des Videomaterials verhindern können.

Eigenschaft	MPEG	DIVX/XVID
Dateigröße	+	+
Kompression	++	++
Qualität	++	++
Standardisierung	++	+
Metadaten	+	+

Tabelle 4-11: Videoformate im Vergleich

Bei der Betrachtung und Bewertung der gezeigten Formate ist für jede Kategorie mindestens ein Dateiformat besonders geeignet und liefert alle notwendigen Kriterien, um im Bereich der Medienserver eingesetzt zu werden. Es existieren Dateiformate, die auf bestimmte Anforderungen abgestimmt sind. Damit sind die einfache Verbreitung der Daten ebenso gegeben, wie die sichere Aufbewahrung und der Erhalt der Informationen.

5 Langzeitarchivierung

Dieses Kapitel bietet eine grundlegende Einführung in die Thematik der Langzeitarchivierung. Begrifflichkeiten werden erklärt, die in diesem Zusammenhang gebräuchlich sind. Zusätzlich werden Techniken beschrieben und Verfahren sowie Vorgehensweisen dargestellt.

Anschließend werden die Grundsätze aus dem Bereich der Langzeitarchivierung auf den langfristigen erhalt der Informationen eines Medienservers übertragen, eine Einordnung in Problemfelder vorgenommen und mögliche Lösungswege aufgezeigt.

5.1 Grundlagen

Unter dem Begriff Langzeitarchivierung (LZA) werden verschiedene Forderungen an den Erhalt von Informationen zusammengefasst. So werden im Einzelnen die Erfassung, die langfristige Aufbewahrung und der Erhalt der dauerhaften Verfügbarkeit der Informationen an diesen Begriff gekoppelt. Schwierigkeiten ergeben sich dabei bei der Begrifflichkeit „Langzeit“, denn es handelt sich hierbei nicht um einen klar umrissenen Zeitraum, sondern um einen in der Zukunft liegenden Zeitpunkt, der aber nicht näher spezifiziert ist. Bei physisch greifbaren Objekten haben sich im Laufe der Zeit unterschiedliche Verfahrensweisen entwickelt, die den langfristigen Erhalt sicherstellen. Bei jeder Entwicklung treten innerhalb der Prozesse häufig Fehler auf, die im Fall der Archivierung zu Daten- oder Informationsverlust führen. Diese Fehler müssen rechtzeitig erkannt und umgangen werden [**13**].

Überträgt man die Archivierung vorhandender Bestände auf die Archivierung digital vorliegender Informationen, treten neue Probleme auf. Für diese neuen Probleme müssen erst geeignete Methoden entwickelt werden. Die direkte Übertragung bekannter Verfahren aus dem klassischen Archivwesen ist nicht möglich, da die Gegebenheiten zu unterschiedlich sind. Zur reinen Konservierung des Trägermediums kommt die Aufrechterhaltung der Interpretierbarkeit als weiterer Aspekt hinzu. Digital vorliegende Daten müssen einerseits in ihrem Bit-Strom erhalten werden, zusätzlich ist es notwendig, die Formatdefinition zu kennen, um diese lesbar zu erhalten. Menschen können Bücher ohne weitere Hilfsmittel lesen und verstehen. Für digital vorliegende Informationen ist immer ein Hilfsmittel erforderlich. Im Folgenden werden die auftretenden Probleme näher erläutert.

5.1.1 Trägermedien

Daten oder Bit-Ströme werden auf einem Medium gespeichert. Bei der klassischen Form des Buchs werden die Worte oder Darstellungen auf Papier gedruckt. Digitale Daten werden auf einem Trägermedium gespeichert. Jedes Trägermedium ist einem Alterungsprozess unterworfen, der die Lesbarkeit der Information beeinträchtigen kann. Der Alterungsprozess kann entweder zu schleichendem Datenverlust oder zum sofortigen Verlust führen. Genau diese Problematik macht die Wahl eines geeigneten Mediums maßgeblich für die spätere Lesbarkeit der Informationen. Derzeit ist kein Trägermedium bekannt, das gegen Alterung „immun“ ist.

Für die Wahl eines passenden Trägermediums müssen bereits im Vorfeld die verschiedenen Einflusskriterien und die Anfälligkeit des Mediums selbst bekannt sein. Die Anfälligkeit kann über aufwändige Simulationen ermittelt werden. So gewonnene Werte sind aber immer nur Anhaltspunkte, da immer gewisse Toleranzen vorhanden sind, die sich nicht ausschließen lassen. In den Simulationen

werden einerseits mechanische Beanspruchungen, andererseits physikalische Faktoren (Temperatur, Feuchtigkeit, Licht) getestet. Die Ergebnisse erlauben eine Einordnung und ungefähre Voraussagen über die Lebensdauer eines Mediums.

Trägermedien können in folgende Kategorien untergliedert werden:

- **Optisch**
  Optische Datenträger (CD oder DVD) kommen vorrangig bei privaten Sicherungen zum Einsatz. Hauptgrund sind die preisgünstigen Medien und die schnelle und zugleich einfache Handhabung. Dabei haben aber gerade diese Medien nur eine sehr kurze Haltbarkeit von nur wenigen Jahren. Unterscheiden muss man an der Stelle noch zwischen Medien, die gebrannt oder gepresst werden. Gepresste CD- oder DVD-Medien kommen auf etwa die doppelte Lebenszeit verglichen mit gebrannten (ca. 8 Jahre zu ca. 4 Jahre). Dieser Unterschied begründet sich auf unterschiedliche Herstellungsverfahren der Rohlinge. Der große Vorteil der optischen Medien besteht darin, dass keine (mechanische) Abnutzung durch den Datenzugriff entsteht. Die Daten werden rein optisch, beispielsweise über Laser, berührungsfrei gelesen.
- **Magnetisch**
  Magnetische Speichermedien sind in erster Linie Festplatten oder Magnetbänder. Im täglichen Gebrauch hat sich gezeigt, dass Festplatten und Magnetbänder ein sehr zuverlässiges Medium für die Datensicherung darstellen. In den meisten Rechenzentren werden Datensicherungen auf Bändern oder Platten realisiert. Die Lebensdauer beträgt dabei in der Regel über 10 Jahre. Die Alterung magnetischer Medien beruht entweder auf Störungen an der Mechanik (Festplatten) oder durch Alterung des Materials, aus dem das Speichermedium hergestellt wurde. Problematisch sind starke Magnetfelder, die die Daten direkt zerstören können, das Medium selbst aber nicht beeinflussen.
- **Mischformen**
  Zusätzlich zu den optischen und magnetischen Medien gibt es auch Mischformen, beispielsweise die Magneto Optical Disk. MO-Medien werden magnetisch beschrieben und optisch ausgelesen. Magnetfelder führen bei diesem Medium zu Problemen, vergleichbar der rein magnetischen.
- **Spezialformen**
  Immer häufiger werden auch die bisherigen Verfahren der Speicherung (optisch, magnetisch) auf neuartige Materialien übertragen. Hiermit sollen die Anfälligkeit unterschiedlicher Materialien bekannter Medien durch haltbarere ersetzt werden, um damit die Lebensdauer des Mediums zu erhöhen.

Bei diesem Vergleich zeigt sich, dass es das perfekte Speichermedium für die Langzeitarchivierung derzeit nicht gibt. Es laufen aber bereits intensive Forschungen an neuen Speichermedien, die vor allem im Bezug auf ihre Haltbarkeit hin optimiert werden. Bis zur Serienreife dieser Medien müssen allerdings weiterhin Rechenzentren die Aufgabe der physischen Datensicherheit übernehmen und in regelmäßigen Abständen den Zustand des Mediums kontrollieren. Redundante Datenhaltung bietet eine weitere Möglichkeit, die Datensicherheit zu erhöhen.

5.1.2 Abspielumgebung

Die Abspielumgebung liefert die notwendigen Komponenten, dass die Datenströme zur Anzeige aufbereitet werden. Dabei sind folgende Grundkomponenten erforderlich:

- **Hardware**
  Zur Hardwareschicht gehören die Komponenten, die einen Computer ausmachen. Das sind unteranderem Prozessor, Arbeitsspeicher, Ein- und Ausgabegeräte, verschiedene

Datenspeicher und die entsprechenden Bus-Systeme zur Vernetzung der Komponenten untereinander.

- **Systemsoftware**
  Die Schicht der Systemsoftware besteht in erster Linie aus einem Betriebssystem, das für den Nutzer eine Abstraktion der Hardware in aufbereiteter Form liefert. Über ein Dateisystem werden Daten in elektronischer Form aus den reinen Bitströmen zu Dateien abstrahiert.
- **Präsentationssoftware**
  Die oberste Schicht bildet die Präsentationssoftware, die Dateien menschenlesbar aufbereitet. Für jeden Dateityp ist auch eine geeignete Anzeigesoftware erforderlich, die die Eigenheiten der Datei interpretieren kann und korrekt darstellt.

Die vielen ineinandergreifenden Teilkomponenten der Abspielumgebung sind stetigen Aktualisierungen unterworfen. Die Aktualisierungen können auf den unterschiedlichen Ebenen erfolgen und müssen aufeinander abgestimmt werden. Zusätzlich erfordern manche Dateien mehr als nur eine Präsentationssoftware. Gerade im Multimediabereich (Ton oder Film) tritt dieses Problem häufiger auf.

5.1.3 Verfahren

Zwei favorisierte Verfahren haben sich innerhalb der Langzeitarchivierung ausgebildet, die den Erhalt der Daten sicherstellen sollen. Diese Verfahren unterscheiden sich grundlegend im Ansatz, können aber auch in Kombination angewandt werden. Als nicht praktikabel haben sich Hardware-Museen erwiesen, die originale Hardware lückenlos betriebsbereit erhalten. Die Vielfalt entwickelter Computersysteme ist zu groß und die Lebensdauer begrenzt.

Eines der verbreiteten Verfahren ist die Emulation. Die Emulation bildet die Laufzeitumgebung zum Zeitpunkt der Erstellung der Daten nach. Es wird ein Emulator für ein bestimmtes System erstellt, das in Funktion und Laufzeitverhalten einem anderen System entspricht. Nachdem es eine Vielzahl an Betriebssystemen und auch Hardware gibt, ist es nicht einfach das komplette Spektrum abzubilden. Je nachdem, bis in welche Tiefe eine Emulation erfolgt, spricht man beispielsweise bei der kompletten Emulation auch der Hardware (inklusive CPU) auch von einer virtuellen Maschine. Hierbei werden alle Aspekte einer Abspielumgebung durch Software nachempfunden und das darauf ausgeführte Programm findet das Ursprungssystem zum Zeitpunkt der Erzeugung vor. Oftmals reicht aber auch nur eine teilweise Nachbildung des Ausgangssystems mittels Emulatoren aus. Hierbei kommt es in erster Linie darauf an, wie komplex die Anwendung in das System eingebunden werden soll und welche Schichten der Abspielumgebung betroffen sind. Viele Programme benötigen keine Simulation der CPU, eine angepasste Präsentationssoftware ist vollkommen ausreichend. Vorteil bei der Emulation ist, dass das erzeugte Datenobjekt keinerlei Änderungen unterzogen werden muss, sondern in der einmal erstellten Form weiterhin bestehen bleiben kann. Nachteilig wirken sich die immer komplexer werdenden Systeme aus, deren Nachbildung erheblichen Aufwand verursachen.

Vorteile:

- Aufwand je Objekt gering
- Objekt-Authentizität sichergestellt
- Kosten abhängig von Zugriffshäufigkeit
- Emulatoren wiederverwendbar

Nachteile:

- Abspielumgebung muss bekannt sein
- Emulationsprogramm aufwändig
- Redundanzen bei ähnlichen Objekttypen

Ein weiteres Verfahren ist die Migration. Hierbei wird nicht die Abspielumgebung an das Datenobjekt angepasst, sondern das Datenobjekt an das aktuelle System. Problematisch ist an der Stelle, dass bei einem Migrationsschritt (die Transformation in ein lesbares Format) sichergestellt werden muss, dass die Informationen aus dem ursprünglichen Datenobjekt behalten werden. Es muss beachtet werden, dass nicht durch eine Reihe aufeinanderfolgender Migrationsschritte das Ausgangsobjekt zerstört wird oder der Informationsgehalt verfälscht wird. Genau dieses Delta, das der Anpassung an das neue System entspricht, muss möglichst klein gehalten werden und darf zu keinem inhaltlichen Verlust führen. Allerdings ist an diesem Verfahren vorteilhaft, dass in der Regel viele Datenobjekte mit der gleichen Laufzeitumgebung erstellt wurden. Somit können diese Objekte mit denselben Verfahren migriert werden.

Vorteile:

- Bekannte Probleme
- Objekt stets aktuell
- Qualitätsverbesserungen möglich

Nachteile:

- Verfälschung des Originals
- Automatisierung schwierig

Abbildung 5-1 zeigt das Antibeispiel einer Migration mit zwei Schritten. Zwei unterschiedliche Ausgangsobjekte können sich einander durch die Migration annähern, es darf aber nicht passieren, dass A‘‘==B‘‘ entsteht. Dieser Effekt darf nur auftreten, wenn die Ausgangsobjekte bereits sehr ähnlich sind und Migrationsschritte genau diese Stellen angleichen. Generell ist darauf zu achten, dass die Eigenständigkeit erhalten bleibt.

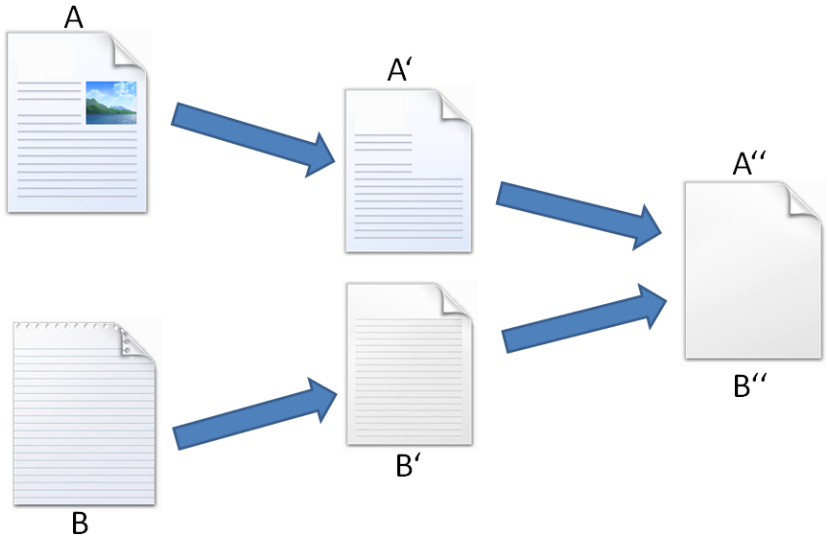

Abbildung 5-1: Migration eines Objekts

5.1.4 Formatüberlegungen

Unabhängig ist die Forderung nach der Standardisierung des Datenformats und somit die Offenlegung und Spezifikation des Funktionsumfangs. Zudem muss die Anzahl der Formate begrenzt sein, um Redundanzen und damit verbundene Probleme gering zu halten. Grundvoraussetzungen sind:

Vollständigkeit der Daten
Innerhalb einer Datei sollen keine externen Ressourcen eingebunden werden. Beispielsweise sind OPI-Kommentare (Open Prepress Interface) zu vermeiden. Alle verwendeten Schriftarten und Vorschaubilder sollen direkt in der Datei gespeichert werden.

Eindeutigkeit der Daten
Benutzte Farben sollen exakt gekennzeichnet sein und auch Fonts dürfen nur die Standardkodierung aufweisen, möglichst in Unicode. Alternative Kodierung bei Bildern oder Texten ist nicht zulässig.

Zugänglichkeit der Daten
Verschlüsselung ist untersagt und auch Zusatzkomponenten sind nicht zugelassen (häufig in Programmiersprachen vorhanden). Auch externe Aktionen sind nicht erlaubt, etwa das Starten externer Programme, Tondokumente oder Filme. Bei der Kompression sind nicht alle Arten möglich. Die LZW-Komprimierung ist aus patentrechtlichen Gründen untersagt, da dadurch die spätere Lesbarkeit nicht garantiert werden kann.

Erschließung der Daten
Die digitalen Objekte müssen eine Reihe von Metadaten enthalten, die den Ursprung der Daten nachvollziehbar machen. Bei Textformaten ist beispielsweise die Angabe der Sprache ein Pflichtkriterium.

Die Langzeitarchivierung stellt je nach gewähltem Verfahren unterschiedliche Anforderungen an den Dateityp, die grundlegenden Forderungen sind verfahrensunabhängig und gelten somit für alle denkbaren Formen als Basis.

Die Emulation versucht die originale Abspielumgebung lauffähig zu erhalten, mit der die Daten erzeugt wurden. Die Daten selber bleiben unangetastet. Für jedes unterstützte Datenformat muss die Abspielumgebung bekannt sein. Damit sich der Aufwand der Erstellung eines Emulators lohnt, ist eine Analyse der Verbreitung des Dateiformats sinnvoll. Kommt ein Format nur sehr selten vor, ist die Emulation nicht die geeignete Methode der Archivierung. Zusätzlich ist zu ermitteln, welche Komponenten für einen bestimmten Dateityp emuliert werden müssen. Die reine Emulation der Darstellung in einem geeigneten Anzeigeprogramm ist dabei die einfachste Form. Weitreichendere Emulatoren bis hin zu virtuellen Maschinen fordern einen deutlich höheren Aufwand. Auf dem Markt gibt es viele ähnliche Formate. Gerade dadurch gibt es Redundanzen bei der Erstellung der Emulatoren.

Für die Migration kommen andere Faktoren zum Tragen. Es spielt meist keine Rolle, ob die Original-Software noch lauffähig vorliegt, wenn die Daten nach einer bekannten Spezifikation erzeugt wurden. Ist eine Langzeitarchivierung vorgesehen, wird bereits während der Erzeugung der Daten auf ein geeignetes Format geachtet. Es eigenen sich besonders Formate, die weit verbreitet sind und gleichzeitig standardisiert sind. Bei Migrationsschritten ist auch die Zusammenlegung verschiedener Formate denkbar. Die Zusammenlegung erfolgt in Formate, die entweder speziell für die LZA entwickelt worden sind, oder sich als besonders geeignet aufgrund ihrer Eigenschaften erwiesen haben.

Durch die gegenseitige Abhängigkeit zwischen Dateiformat und Archivierungsverfahren ist eine genaue Analyse des Bestandes notwendig. Liegen die Daten bereits digital vor, ist die Wahl des passenden Verfahrens entscheidend. Kann auf die Digitalisierung analoger Daten Einfluss genommen werden, ist vor allem die Wahl des passenden Dateiformats entscheidend.

5.2 Open Archival Information System (OAIS)

Die Problemstellung der Archivierung von Repositorien ist noch ein relativ junges Forschungsgebiet. Dennoch gibt es mit dem Reference Model for an Open Archival Information System (OAIS) [**21**] bereits

ein Referenzmodell, in dem theoretische Überlegungen und mögliche Vorgehensweisen erläutert werden. Außerdem werden wichtige Grundbegriffe geprägt, die allgemeine Verwendung finden. Das Modell zeigt Wege auf, wie Prozesse und Abläufe standardisiert erfolgen können ohne dabei speziell auf bestimmte Systeme einzugehen, die bereits softwareseitig realisiert sind.

Die stark vereinfachte Abbildung 5-2 zeigt die grundlegenden Anforderungen, die ein Archivsystem erfüllen muss. Es müssen auf der einen Seite die Inhalte durch Autoren oder Ersteller eingestellt werden können und auf der anderen Seite die Anfragen verarbeitet werden können, die von Nutzern an das System gestellt werden. Die eigentliche Arbeit erfolgt innerhalb der Software und ist in der Regel nicht nach außen sichtbar, außer für Administratoren, die komplexe Vorgänge an die Bedürfnisse anpassen können.

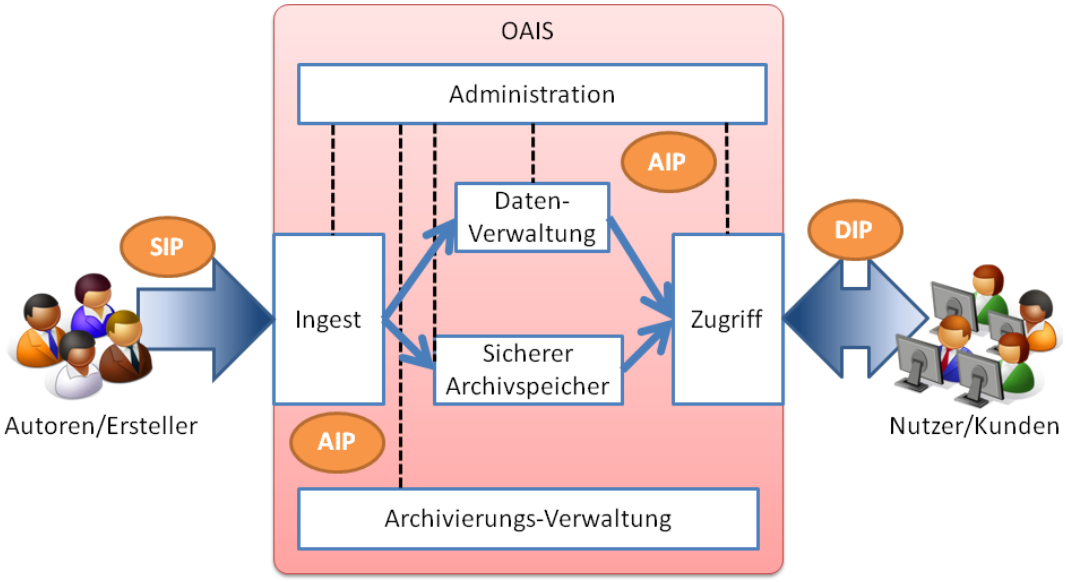

Abbildung 5-2: OAIS-Modell

Das OAIS-Modell zeichnet einige Begrifflichkeiten und Basiskomponenten näher aus, die im Zusammenspiel das Archivsystem bilden:

5.2.1 Begriffe des OAIS-Referenz-Modells

Liefereinheit (SIP)
Die Liefereinheit oder das Submission Information Package (SIP) bezeichnet Informationen, die in das System eingestellt werden sollen. Diese Liefereinheit beinhaltet Daten, die zum einen ein Objekt beschreiben können, andererseits auch das Digitalisat selber. Somit kann man ein SIP wiederum in Untereinheiten zu Content Information und Preservation Description Information (PDI) untergliedern.

Archivierungseinheit (AIP)
Die Archivierungseinheit, Archival Information Package (AIP), besteht aus den Informationen aus der SIP und beinhaltet die PDI. Daraus wird ein archivierungsfähiges Format erstellt, das intern vom OAIS verwaltet werden kann. Dabei ist das eigentliche interne Format nicht fest vorgegeben sondern abhängig vom eingesetzten OAIS.

Bereitstellungseinheit (DIP)
Mit der Bereitstellungseinheit, Dissemination Information Package (DIP), liefert das OAIS dem Nutzer Daten in aufbereiteter Form aus. Dabei können SIPs enthalten sein und auch alle oder nur ein Teil der

PDI mitgeliefert werden. Das ist wiederum abhängig vom OAIS. Grundlegend ist allerdings, dass die Informationen aus den Daten der AIP erzeugt werden.

5.2.2 Basiskomponenten des OAIS-Referenz-Modells

Ingest
Die Ingest-Komponente stellt die Schnittstelle des Erstellers der Daten zum OAIS dar. An dieser Stelle werden die SIPs entgegen genommen und zu AIPs umgewandelt und dabei gleichzeitig die entsprechenden Stellen der Administration und Daten- bzw. Archivierungsverwaltung mit einbezogen. Bei der Umwandlung der SIPs zu AIPs wird außerdem das systeminterne Format erzeugt.

Sicherer Archivspeicher
Der sichere Archivspeicher nimmt vom Ingest die AIPs entgegen und verteilt und verwaltet die Daten. Dabei wird diese Komponente als Black-Box betrachtet und davon ausgegangen, dass die komplette Verwaltung einschließlich des Datenerhalts auf dem Speichersystem an dieser Stelle mit angebunden ist. Diese Komponente ist auf der Hardwareebene anzusiedeln.

Datenverwaltung
Die Datenverwaltung stellt administrative Tools bereit, mit deren Hilfe die Archivdaten verwaltet und gewartet werden können, die in das OAIS eingestellt wurden. Darunter fallen alle Aufgaben, die dem Erhalt und der Kontrolle der Informationen aus den AIPs dienen.

Administration
Die Administration steht innerhalb eines OAIS über allen Operationen und überwacht diese. Dabei ist ein entscheidender Part die Kontrolle der eingespielten Daten bereits beim Einstellen in das OAIS. Hier soll vor allem eine Kontrolle der Daten hinsichtlich gültiger Standards erfolgen. Aber auch die Konfiguration bei Zugriffen auf Daten über die Zugriffskomponente kann über die Administration gesteuert werden.

Archivierungsverwaltung
Unter die Aufgaben der Archivierungsverwaltung fallen Funktionalitäten, die den Langzeitzugriff auf die OAIS-Informationen gewährleisten sollen. An dieser Stelle erfolgt die Verwaltung der AIPs mit allen dazu notwendigen Vorkehrungen und Überarbeitungen, falls beispielsweise für den Datenbestand die Ausführungsumgebung nicht mehr vorhanden ist und eine Migration erforderlich wird.

Zugriff
Die Zugriffskomponente ist die Schnittstelle für Nutzer des OAIS, die Anfragen stellen und Daten geliefert bekommen. Dabei regelt die Zugriffskomponente Beschränkungen, die Daten vor der Verteilung filtert, wenn entsprechende Definitionen vorliegen.

5.2.3 OAIS-Komponenten in Medienservern

Ein Medienserver kann einen Großteil der OAIS-Komponenten abdecken, die vorangegangen beschrieben wurden. Einige Komponenten des OAIS bilden Basiskomponenten des Medienservers, andere können über kleine Erweiterungen den geforderten Funktionsumfang bieten.

Beim Ingest fehlt einem Medienserver die Kontrolle auf Langzeitarchivierungs-Kriterien. Rüstet man die zusätzliche Kontrolle nach, können aus den SIPs die AIPs und PDIs erzeugt werden. Interne Verwaltungs- und Administrationsfunktionen benötigt auch der Medienserver. Nach der Erweiterung um die Anbindung an die Archivierungsverwaltung unterstützt der Medienserver den geforderten Umfang eines OAIS. Medienserver benötigen große Datenspeichersysteme, die schnellen Zugriff auf die Daten

gewährleisten. Die Kombination geeigneter, schneller Speichersysteme mit zusätzlichem, besonders sicherem Speicher für die Archivierung, liefert die fehlende Komponente für die Archivierung.

Die Erweiterung des Medienservers um die spezifischen Komponenten der Archivierung aus dem OAIS-Modell macht aus einem Medianserver ein vollwertiges Archivierungssystem. Neben der reinen Umsetzung der Komponenten ist vor allem die Anpassung der Prozesse entscheidend. Dieser Bereich erfordert größeren Planungs- und Umsetzungsaufwand.

5.3 LZA bei Medienservern

Die Unterstützung der Langzeitarchivierungskomponenten aus dem OAIS-Modell muss bereits bei den Planungen für einen Medienserver mit einbezogen werden. Dabei stellt die Prozessorganisation besondere Anforderungen, die grundlegende Entscheidungen vorsehen. Aus diesen Vorgaben ergeben sich Richtlinien und Abläufe, die den kompletten Medienserverbetrieb beeinflussen. Ein „Nachrüsten“ der LZA-Funktion für bestehende Systeme großer Sammlungen ist entweder nur mit sehr hohem Aufwand oder überhaupt nicht möglich. Für den Prozessablauf sind einige Kriterien zu beachten:

5.3.1 Erhalt des Originals

Nur während des Ingests liegen die Originaldaten des Objekts vor. An dieser Stelle ist die Entscheidung darüber zu treffen, ob diese Informationen gespeichert oder verworfen werden. Medienserver nutzen intern verschiedene Instanzen der digitalen Daten zur geeigneten Verarbeitung und Darstellung. Diese Instanzen sind abhängig von der eingesetzten Software und können über ihre Funktion klassifiziert werden. Neben ihrer Funktion ist auch der zugrundeliegende Dateityp verantwortlich für die Art und den Umfang der Instanzen.

Verwendet der Medienserver intern ein anderes Datenformat als das des Originals, ist für die reine Nutzung der Daten im Medienserver das Originalobjekt nicht erforderlich. Kommt für die LZA die Migration zur Anwendung, können bereits für den Ingest erste Migrationsschritte erforderlich werden. Erhält man das Original, ist die Rekonstruktion möglich und eventuellem Datenverlust bestmöglich vorgebeugt.

Nutzt der Medienserver die Emulation als LZA-Verfahren, bleibt das Original in jedem Fall erhalten. Die Emulation bietet sich allerdings nicht vorranging für den täglichen Betrieb und die Nutzung der Daten über einen Medienserver an. Auch hier wird die Erstellung unterschiedlicher Instanzen des Originals erforderlich.

5.3.2 Langzeitobjekt

Bei der LZA der Medienserverdaten können sowohl das Original, als auch eine speziell angepasste Version des Originals für die Langzeitarchivierung genutzt werden. Die Entscheidung darüber muss im Einzelnen getroffen werden, denn beide Varianten haben Vor- und Nachteile.

Ist das Langzeitobjekt gleich dem Original, können nur die Informationen über das Objekt gespeichert werden, die bereits in der Originaldatei enthalten sind. Weitere Metadaten müssen getrennt davon in einer separaten Datei abgelegt werden, die synchron zum Original vorliegen muss. Bei der Entscheidung zu einem eigenen Langzeit-Objekt sind Migrationsschritte bereits beim Ingest möglich, die eine Anreicherung um Metadaten oder die Konvertierung in ein bestimmtes Datei-Format vornehmen. Dabei erfolgen aber immer Änderungen am Original, die nicht immer rückgängig gemacht werden können.

Migrationsschritte beim Ingest erübrigen sich, falls der Medienserver auf fest definierte Formate begrenzt ist, die die notwendigen Eigenschaften für die LZA bereits mitbringen. Eine Auswahl an

Dateiformaten wurde im vorangegangenen Kapitel näher beschrieben. Wendet man die Kriterien der LZA auf diese Formate an, erhält man aus jeder Dateikategorie mindestens ein geeignetes Dateiformat. Neben dem Dateiformat ist die Art der Speicherung der Metadaten für ein Objekt entscheidend. Durch die Verwendung umkehrbarer Methoden kann aus dem überarbeiteten Objekt das Original wiederhergestellt werden. Es existieren Mechanismen der Verifikation, um Aussagen über die Bit-Gleichheit von Dateien treffen zu können.

5.3.3 Notwendige Informationen

Notwendige Metainformationen zu digitalen Objekten stammen aus den in Kapitel 3 definierten Formaten. Im Medienserverumfeld kommen unterschiedliche Formate zum Einsatz, die für die LZA teilweise Erweiterungen erfordern. Durch die Unterstützung der Basisfunktionen ist bereits der größte Teil der Anforderungen der LZA abgedeckt. Der Bereich der Archiv-Metadaten kommt hinzu. Die Kombination der Metadaten aus den unterschiedlichen Bereichen decken die Forderungen des OAIS-Modells ab. Die Daten des SIP stammen aus den deskriptiven- und den technischen Metadaten. Das DIP kann über die vorliegenden Daten erzeugt werden und wird ebenso standardmäßig von Medienservern unterstützt. Für das AIP ist eine Kombination vorhandenere Daten mit der Erweiterung um zusätzliche Informationen Voraussetzung.

Deskriptive Metainformationen
Die deskriptiven Metadaten bilden den Basissatz an Zusatzinformationen, auf denen auch die Verwaltung der Medienserver aufbaut. Je nach Dateiformat unterscheiden sich die mitgelieferten Metadaten, eine Vereinheitlichung schafft Abhilfe. Besonders geeignet sind das Dublin-Core-Schema oder das komplexere METS-Format. Medienserver müssen nur mit diesen Formaten bei Im- und Export umgehen können, zur internen Verwaltung sind keine weiteren Vorschriften zu beachten. Die interne Verwaltung ist unabhängig.

Diese beiden Metadatenformate liefern nur einen Basissatz an Informationen. Detailliertere Angaben oder spezielle Attribute zu den Objekten selbst können meist problemlos von Medienservern verwaltet werden, aber die Nachnutzung stellt ein Problem dar. Eigene Zusatzinformationen passen nicht zwangsläufig auf die Attribute der standardisierten Formate. Eine geeignete Abbildung ist erforderlich. Bei dieser Abbildung können Kollisionen auftreten, die zu Datenverlust führen, wenn sie nicht rechtzeitig erkannt werden.

Administrative Metainformationen
Administrative Metadaten enthalten Informationen über das Objekt, die für die Verwaltung und Sicherung des Zugriffs genutzt werden. An dieser Stelle erfolgen die Einhaltung des Urheberrechts und die zentrale Zugriffssteuerung auf die Digitalisate oder bestimmte Instanzen. Diese administrativen Daten müssen besonders sicher gelagert werden und vor Missbrauch gesichert werden.

Einige standardisierte Metadatenmodelle sehen spezielle Attribute vor. Im METS-Format ist die Speicherung in einer eigenen Sektion vorgesehen. Die Daten liegen aber getrennt vom Digitalisat vor. Die Verwaltungssoftware muss den Zusammenhang herstellen. Bei IPTC ist der Vermerk über den Autor und das Rechtemodell vorgesehen und direkt in der Datei enthalten. Neben diesen Angaben nutzen Medienserver meist weitere Attribute zur Zugriffssteuerung und Verwaltung, die nicht nach außen weitergegeben werden.

Technische Metainformationen
Ein Teil der technischen Informationen über das digitale Objekt können direkt aus der Datei extrahiert werden. Die betrachteten Dateiformate aus Kapitel 4 beinhalten Daten zu Entstehung und zum Dateiformat. Im Detaillierungsgrad unterscheiden sich die Formate stark voneinander. Textformate

liefern meist nur Hinweise mit der Angabe der Version. Weitere Daten können über das Objekt selbst erzeugt werden.

Grafikformate sehen weitreichendere technische Informationen vor und definieren eigene Formate (beispielsweise Exif). Die Zusatzinformationen sind durch die unterschiedliche Erzeugung dieser Dateien für eine korrekte Darstellung erforderlich. Durch das Fehlen eines formatübergreifenden einheitlichen Metadatenformats und unterschiedliche Kriterien, verwalten Medienserver die technischen Metadaten formatabhängig.

Neben den rein aus der Datei gewonnenen Metadaten verwenden Medienserver weitere technische Informationen zum Objekt. Das kann die genaue Spezifikation der Abspielumgebung sein oder auch Gegebenheiten zum Erzeugungszeitpunkt, die nicht in der Datei hinterlegt sind.

Archivmetainformationen
Das OAIS-Referenzmodell sieht Archivmetainformationen vor. Diese Daten beinhalten Informationen zur Ablauforganisation und für die Erhaltung des Zugriffs auf das Objekt sowie die Erhaltung des Objekts selbst.

Die notwendigen Daten kommen aus den bisher genannten Bereichen der Metadaten und müssen ergänzt werden. Vor allem eine möglichst genaue Beschreibung der vorliegenden Datei in Struktur und Inhalt ist erforderlich. Außerdem sind Daten über die Abspielumgebung notwendig. Das eingesetzte LZA-Verfahren bringt zusätzliche Daten mit. Bei der Emulation ist vor allem die Ausgangsumgebung zu spezifizieren. Bei der Migration müssen für jeden einzelnen Migrationsschritt Angaben vorhanden sein.

Archivmetadaten bringen ein großes Problem mit, da heute nicht alle Einflussfaktoren bekannt sind, die vielleicht in der Zukunft den Zugriff verhindern können. Eine ständige Wartung und Überarbeitung der Daten bringt die größtmögliche Sicherheit.

Strukturelle Metainformationen
Strukturelle Metadaten sind eine Eigenheit eingesetzter Software bei der Nutzung großer Datenmengen. Programmspezifische Informationen kommen zum Tragen, wenn nicht nur ein einzelnes Objekt betrachtet wird sondern eine Sammlung großer Datenmengen, die in Beziehung zueinander stehen. Zum Erhalt dieser Beziehung kann einerseits die Auswertung bestimmter Metadatenattribute erfolgen oder ein direkter Verweis auf ein anderes Objekt gespeichert werden.

Medienserver können mit heterogenen Sammlungen umgehen und benötigen eindeutige Identifikatoren zur Verlinkung und dem Aufbau der Struktur. Das Metadatenformat METS bietet Möglichkeiten der Abbildung dieser Struktur.

5.3.4 Speicherung der Metainformationen

Zwei Grundüberlegungen sind entscheidend für die Speicherung der Metainformationen. Zum einen muss das Format festgelegt werden und zum anderen die Position.

Das Format muss einfach aufgebaut und leicht lesbar sein. Eine weite Verbreitung und die Standardisierung erleichtern die Unterstützung unterschiedlicher Softwaresysteme und fördern damit die Interoperabilität. Als syntaktisches Format bietet sich XML an. Durch die hohe Flexibilität in Kombination mit der Möglichkeit, strukturiert Daten zu verwalten, verwenden viele Standards bereits die Syntax von XML. Das Format ist neben der guten Unterstützung durch Parser auch menschenlesbar. Gerade für die Archivierung ist das ein sinnvolles Kriterium, falls geeignete Parser einmal fehlen. Als Semantik bieten sich die standardisierten Formate RDF und XMP an (siehe Kapitel 3.4.1 bzw. 3.4.2). Beide bieten umfassende Möglichkeiten der flexiblen Metadatenattributverwaltung. Die Unterstützung für RDF bringt für die Nutzung über das Internet weitere Vorteile mit.

Neben Syntax und Semantik der Metadaten ist der Speicherort der Daten zu definieren. Die Daten können entweder direkt innerhalb der Datei abgelegt werden oder extern als eigenständige XML-Datei. Die Kombination ist ebenfalls denkbar. Beide Varianten bringen Vor- und Nachteile mit. Abbildung 5-3 zeigt schematisiert die Möglichkeiten:

Abbildung 5-3: Metadatenspeicherung

Bei der gesonderten Speicherung der Metadaten besteht die Herausforderung darin, die Verbindung zwischen Metadaten und Datei aufrecht zu erhalten. Geht die Verbindung verloren, hat das weitreichende Folgen für die LZA. Ohne das AIP aus den Metadaten sind der Zugriff und die Interpretierbarkeit nicht sichergestellt. Gerade bei der Migration der Daten zwischen Systemen und dem Datenaustausch kommt es immer wieder zur Zerstörung der Verbindung. Neben der Zerstörung kann auch die Fehlzuordnung problematisch sein. Werden fehlerhafte Zuordnungen nicht rechtzeitig erkannt, kann Datenverlust infolge falscher Migrationsschritte resultieren. Zur Vermeidung sind passende Mechanismen über eindeutige Prüfsummen/Identifikatoren oder das Verpacken der Daten in geeignete Containerformate möglich, die allerdings zusätzlichen Aufwand bedeuten. Zudem muss das Containerformat bekannt und ebenfalls standardisiert oder selbsterklärend sein.

Die zweite Möglichkeit ist die Einbettung der Metadaten direkt in die Datei. Die Metadaten werden dabei an definierter Stelle direkt im digitalen Objekt gespeichert. Die Speicherung einer Verbindung zwischen Metadaten und Datei entfallen und somit werden Fehlzuordnungen ausgeschlossen. Aber nicht jedes Dateiformat eignet sich für diese Einbettung. Ursache kann entweder die Definition des Formats selbst sein oder eine Kollisionen mit bereits vorhandenen Metadaten. Somit muss für jedes Dateiformat explizit ermittelt werden, ob eine Einbettung überhaupt möglich oder sinnvoll ist. Die Änderungen an der Datei müssen zudem innerhalb der Spezifikation konform sein und in gültigen Dateien resultieren. Im Bereich der Medienserver kommt nur eine begrenzte Anzahl an Dateiformaten vor. Diese Voraussetzung begünstigt die Nutzung der Einbettung aufgrund der geringen Formatanzahl.

5.3.5 Erzeugung der Metadaten

Die Generierung der Metadaten erfolgt über unterschiedliche Vorgehensweisen und Verfahren, abhängig von ihrer Art und Herkunft:

Deskriptive Metadaten
Die Eingabe der deskriptiven Metadaten muss explizit über geeignete Personen erfolgen, um die Datenqualität ausreichend hoch zu erhalten. Viele Dateiformate bieten Unterstützung für Metadaten. Nach der Extraktion aus der Datei kann nicht mit Sicherheit gesagt werden, ob die Daten automatisiert über Programme oder explizit über Personen eingeben wurden. Bei automatisch eingefügten Metadaten zum Inhalt ist die Fehleranfälligkeit sehr hoch, ein Verlass auf die so gewonnen Informationen somit nicht sinnvoll. Zudem sind die Angaben innerhalb der Dateiformate freiwillige, meist nicht standardisierte Werte.

Für qualitativ hochwertige deskriptive Metadaten wird nach der automatischen Erfassung aus dem Dateiinhalt eine Kontrollinstanz nachgeschaltet, die die Daten beurteilen und gegebenenfalls korrigieren kann. Diese Kontrolle erfolgt in der Regel durch fachkundige Personen und ist sehr zeitaufwändig. Nach der Korrektur können die so gewonnenen Informationen einerseits im internen Verwaltungssystem genutzt werden oder über Exportformate neu zusammengestellt und ausgetauscht werden. Für die Originaldatei ist die grundsätzliche Frage zu beantworten, ob die korrigierten Metadaten wieder in die Datei zurückfließen oder die Originaldatei erhalten bleiben soll.

Die zu erwartende Datenqualität ist abhängig vom zugrundeliegenden Dateiformat. Bei Multimediaformaten werden die deskriptiven Metadaten generell besser gepflegt, als beispielsweise bei Grafikformaten.

Administrative Metainformationen
Die administrativen Metainformationen können größtenteils aus dem System heraus erzeugt werden, in dem das digitale Objekt eingestellt ist. Für den Endbenutzer sind diese Informationen bis auf wenige Ausnahmen nicht maßgebend, beispielsweise Angaben zum Urheberrecht. Eine genaue Rekonstruktion des Ausgangssystems oder die Migration auf ein anderes System sind ohne die administrativen Daten nicht möglich.

Technische Metadaten
Der größte Teil der technischen Metadaten kann direkt aus dem digitalen Objekt gewonnen werden. Die Extraktion ist dabei automatisierbar und es werden nur geringe Eingriffe erforderlich, wenn spezielle Informationen erzeugt werden sollen. Der Umfang der Metadaten ist vom Dateityp abhängig. Das reicht von der Minimalangabe des Dateiformats bis hin zu detaillierten Angaben zur Erzeugung der Datei.

Archivierungsinformationen
Für die Erzeugung der Archivierungsinformationen werden unterschiedliche Verfahren eingesetzt. Einen Teil der Informationen kann man direkt aus dem System über die technischen Metadaten gewinnen, ein anderer Teil muss explizit eingeben werden. An dieser Stelle gibt es keinen einheitlichen Standard. Gerade die Angabe der Informationen über die Abspielumgebung erfordert einen manuellen Eingriff.

Strukturelle Metadaten
Strukturelle Metadaten werden komplett automatisiert über das Verwaltungssystem erstellt. Nur der Zeitpunkt der Erfassung der Daten muss entsprechend geeignet gewählt werden. Es macht keinen Sinn, strukturelle Daten direkt nach dem Einspielen des Digitalisats zu erheben. Erst nach einer vollständigen Katalogisierung und Klassifizierung stehen alle Informationen über das Objekt zur Verfügung.

5.3.6 Sicherer Archivspeicher

Sichere Archivsysteme sind auf zuverlässigen Datenspeicher angewiesen, die die Daten langfristig erhalten kann. Die Art des Speichers ist für den Medienserver unerheblich. Lediglich gewisse Eigenschaften sind entscheidend:

Datensicherheit
Das Speichersystem muss zuverlässig die Daten vorhalten und vor Datenverlust aufgrund von Hardwarefehlern schützen. Gerade die begrenzte Lebensdauer verwendeter Speichermedien muss über das Speichersystem geregelt sein.

Anbindung
Über Schnittstellen muss der Speicher an einen Medienserver angebunden werden können. Dabei ist die Kommunikation mit dem Speicher in beide Richtungen erforderlich, lesend und schreibend. Daten müssen vom Medienserver auf das Archivsystem ausgelagert werden können und auch wieder

zurückgelesen werden können, um notwendige Wartungsarbeiten durchzuführen. Nach der Bearbeitung der Daten werden diese wieder an den Archivspeicher übergeben.

Zugriff
Die Zeit zwischen Anforderung der Daten durch den Medienserver bis zur Auslieferung durch den Archivspeicher stellt ein wichtiges Kriterium dar, das der Medienserver nicht beeinflussen kann. Entsprechende Funktionen innerhalb des Medienservers müssen dafür sorgen, dass die Archivdaten ausgeliefert werden können und gegebenenfalls asynchrone Mechanismen mit Benachrichtigung vorhanden sind.

5.3.7 Zusätzliche Verwaltungsstrukturen

Die Organisation der zusätzlichen Komponenten für die LZA erfordert geeignete Verwaltungsstrukturen, die auf die Anforderungen des Medienservers und die Erfordernisse des OAIS-Modells eingehen. Die Archivverwaltung kann als zusätzliche Komponente der Verwaltung angesehen werden und muss im kompletten Lebenszyklus eines Objekts berücksichtigt werden. Die erforderlichen Parameter für die Ablaufsteuerung der Archivierung können in Form von Metadaten auf Objektebene erfolgen. Schwierigkeiten bereitet die Tatsache, dass bereits während des Ingests definiert sein muss, ob eine LZA erfolgen soll. Denn bereits an dieser Stelle sind geeignete Vorkehrungen zu berücksichtigen.

5.4 Problemanalyse

Die bisher beschriebenen Rahmenbedingungen der Langzeitarchivierung bei Medienservern lassen sich in Teilbereiche untergliedert. Es werden nur die charakteristischen Bestandteile beschrieben. Die Bedingungen lassen sich in hardware- und softwareseitige Überlegungen, administrative und allgemeine Bestandteile zerlegen. Eine funktionsfähige Lösung erfordert die Umsetzung aller Komponenten:

5.4.1 Speichermedien

Für die unterschiedlichen Anforderungen gibt es spezielle Speichermedien, die Vorzüge in bestimmten Bereichen bieten. Nur die Nutzung verschiedener Medien deckt alle Anforderungen des Medienservers und die der Archivierung ab:

Medienserver fordern die schnelle Datenlieferung an den Benutzer, die Archivierung einen besonders sicheren und zuverlässigen Speicher [**17**]. Eine Verbindung beider Forderungen in einem einzigen Speicher ist nur mit hohem Kostenaufwand realisierbar. Schneller Speicher in großen Mengen ist sehr teuer und nicht optimal für den langfristigen Erhalt und die Einlagerung großer Datenmengen. Besonders sicherer Archivspeicher ist für den langfristigen Datenerhalt ausgelegt und eignet sich aufgrund langer Zugriffszeiten nicht zur primären Nutzung als Datenspeicher für Medienserver. Nur eine heterogene Umgebung mit der Kombination aus schnellem und sicherem Speicher erfüllt beide Anforderungen. Für die tägliche Nutzung der Medienserverdaten kommt der schnelle direkt angebundene Speicher zum Einsatz, zusätzlich steht sicherer Speicher für Archivierungszwecke zur Verfügung.

Die Kontrolle beider Ablageorte übernimmt die Archivierungsverwaltung des Medienservers. Hier erfolgt die Entscheidung, welche Daten sinnvollerweise auf welchem Speicher abgelegt werden. Für die Auslieferung der Archivdaten direkt über den Medienserver sind Transformationen der Archivdaten erforderlich. Zur optimalen Sicherung der Medienserverdaten im Langzeitspeicher kommen verschiedene Archivsysteme infrage. Der Medienserver kann an mehrere Archivsysteme gleichzeitig angebunden werden, die auf die Anforderungen abgestimmt sind. Die Implementierung der Protokolle der Archivsysteme und die Verbindung mit dem Medienserver über konfigurierbare Schnittstellen

schafft die erforderliche Flexibilität. Neben der Speicherung der Archivdaten im entsprechenden Speichersystem ist die Zwischenspeicherung häufig genutzter Archivdaten in schnellem Zwischenspeicher sinnvoll. Damit ist der Übergang zwischen den eingesetzten Systemen fließend und die mehrfache Ablage von Teilen der Daten, auch auf mehreren Speichermedien, möglich und erforderlich.

5.4.2 Laufzeitumgebungen

Die Laufzeitumgebung des Medienservers ist ein flexibles System, das den Nutzer und Autor bei seinen Operationen unterstützt. Dem Nutzer werden die angeforderten Daten in möglichst kurzer Zeit ausgeliefert. Dabei stehen Kategorisierungs- und Suchfunktionen zur Verfügung. Autoren benötigen bei der Publikation der Daten umfassende Benutzerführung. Vor allem die Benutzerführung unterschiedlicher Anwender fordert den Medienserver.

Eine einfache Oberfläche mit Hilfestellung für mögliche Nutzerinteraktion gewährleistet die Akzeptanz der Nutzer. Neben der Akzeptanz ist aber gerade die Unterstützung beim Ingest-Prozess entscheidend für die Datenqualität. Denn nur durch Maßnahmen der Qualitätssicherung kann ein langfristiger Erhalt der Daten auch gewährleistet werden. Die Qualitätssicherung umfasst die Bereiche des Dateityps, der Metadaten und die Einordnung und Verortung der Daten innerhalb des Bestandes eines Medienservers. Dadurch ist das Wiederfinden über Suchmechanismen und die Beschreibung der Daten für die LZA sichergestellt.

Die angemessene Performanz des Medienservers trägt zur Akzeptanz der Benutzer bei. Liegen die Antwortzeiten in vertretbarem Rahmen, steigt die Anzahl der Nutzer. Die Definition des vertretbaren Rahmens ist nur subjektiv möglich und hängt stark von den Erwartungen des jeweiligen Nutzers ab. Durch die Gewöhnung an sofortigen Zugriff auf Informationen erwartet der Internetbenutzer eine Reaktion des Systems innerhalb eines einstelligen Sekundenbereichs. Inzwischen sind unterschiedliche Mechanismen im Einsatz, die Engpässe der Performanz für den Nutzer auf ein erträgliches Maß reduzieren. Die Begrenzung des Datenvolumens während des Transfers oder das Aufteilen einzelner Anfragen in mehrere Teile bringen Verbesserung (beispielsweise AJAX [**35**]).

Die Archivierungsverwaltung erfordert eine einfache Integration der Steuerung direkt aus dem Medienserver heraus. Die Verankerung der LZA benötigt an unterschiedlichen Stellen zusätzliche Mechanismen, angefangen beim Ingest über die Verwaltung bis zur Auslieferung der Daten.

5.4.3 Datenformate

Die Anzahl der Datenformate in aktuell genutzten Softwareprodukten steigt nahezu täglich. Nur eine geringe Zahl an Formaten ist offengelegt und auch standardisiert. Die restlichen Formate sind proprietärer Art, die jeweils auf eine spezielle Anwendung zugeschnitten ist. Genau an dieser Stelle liegt das Problem für die Langzeitarchivierung. Nichtstandardisierte Formate sind für eine nachhaltige Speicherung ungeeignet, da sie jeder Zeit Änderungen unterworfen sein können. Beispiele aus der Vergangenheit zeigen, dass Quasi-Standards innerhalb kürzester Zeit in Vergessenheit geraten können und durch andere Formate ersetzt werden. Gibt es keine geeigneten Migrationstools, kann das zu Datenverlust führen. Neben den Dateiformaten sind auch Betriebssysteme einer kontinuierlichen Weiterentwicklung unterworfen. Formate für spezielle Betriebssysteme eignen sich nicht für einen langfristigen Erhalt.

Geeignete Dateiformate können nur nach heutigen Gesichtspunkten analysiert und durch Erfahrungen aus der Vergangenheit ergänzt werden. Die daraus resultierende Einschätzung über eine Eignung bietet keine hundertprozentige Sicherheit. Diese Problematik führt zur Entwicklung spezieller Dateiformate,

die besonders auf die LZA abgestimmt sind. Es gibt diese Archivformate inzwischen für die verschiedensten Bereiche.

5.4.4 Objektkategorien

Bereits in Kapitel 4.6 konnten für verschiedene Objektkategorien geeignete Dateiformate gefunden werden, die im Zusammenhang mit der LZA bei Medienservern besonders positive Eigenschaften mitbringen. Eine Verallgemeinerung der Objektkategorien liefert zwei sich grundlegend unterscheidende Typen, einerseits textbasierte Formate, andererseits Multimediaformate.

Textobjekte
Die Flexibilität des PDF-Formats als De-fakto-Standard für textbeschreibende Objekte ermöglicht es, dass nahezu jedes textbasierte Dateiformat nach PDF migriert werden kann. Bei diesem Migrationsschritt ist darauf zu achten, dass nach Möglichkeit PDF/A zum Einsatz kommt. Damit sind eine einheitliche Darstellung und die einfache Verbreitung mit gleichzeitiger Wahrung des Zugriffs gewährleistet. Inzwischen existieren neuere Versionen von PDF, die erweiterte Möglichkeiten bieten und zu PDF/A nicht verlustfrei kompatibel sind. Wird das bereits bei der Erzeugung der Ausgangsdatei berücksichtigt, führt dies nur zu minimalen Einschränkungen.

Nicht alle textbasierten Dateien erfordern eine Migration nach PDF. Handelt es sich um reine ASCII-Formate (siehe Kapitel 4.2.1), bei denen die Darstellung unerheblich ist, kann die Datei im Rahmen der LZA direkt erhalten werden. Es besteht die Möglichkeit, diese Dateien nach XML zu migrieren, um eine Vereinheitlichung zu erreichen.

Aus Sicht der LZA decken die beiden Formate PDF und XML den kompletten Bereich der textbasierten Dateitypen ab und eignen sich damit als Migrationsziele.

Mediaobjekte
Mediaobjekte erfordern unbedingt eine korrekte Interpretation der Daten. Darunter fallen Bilder, Ton- und Filmdaten. Im Bereich der Bilddaten ist das TIFF-Format sehr gut geeignet, um für die LZA zum Einsatz zu kommen (siehe Kapitel 4.6). Das Containerformat erfüllt die Kriterien am besten. Die aktuelle Entwicklung in der Bilderzeugung nutzt überwiegend das JPEG-Format. Dieses Format kann problemlos nach TIFF migriert werden. Allerdings ist die Tatsache zu beachten, dass bereits mit der Erzeugung der JPEG-Dateien Datenverlust einhergeht. Für Retrodigitalisierungsprojekte ist daher darauf zu achten, dass bereits für die Erzeugung ein verlustloses Format zum Einsatz kommt.

Bei Ton- und Videodateien konnte sich bisher kein Standard durchsetzen, der als alleiniges Format für die LZA eingesetzt wird. In der Gegenüberstellung der Formate im vorangegangenen Kapitel zeigt sich diese Tatsache darin, dass jedes Format immer auch einen gravierenden Nachteil mitbringt. Bei der Betrachtung der Formate in Bezug auf die Nutzung in einem Medienserver ist dieser Nachteil die Dateigröße. Die Übertragung großer Datenmengen erfordert eine hohe Bandbreite. Wird dieser Faktor vernachlässigt, finden sich mit dem WAVE-Dateiformat für Audiodaten und dem MP4-Format für Videos geeignete Vertreter, die für die LZA zum Einsatz kommen können. Für die Nutzung innerhalb des Medienservers sind dabei weitere Instanzen der Daten erforderlich, die besonders auf ihre Größe hin optimiert sind.

Für Mediaobjekte deckt das TIFF-Format den Bildbereich, das WAVE-Format den Audiobereich und das MP4-Format den Videobereich am besten in Bezug auf eine LZA ab. Diese Formate eignen sich am besten als Migrationsziele für Mediaobjekte.

5.4.5 Rechtliche Einschränkungen

Neben den Dateiformaten stellen rechtliche Einschränkungen ein großes Problem bei der LZA dar. Besonders das Urheberrecht verlangt genaue Kenntnisse über die Erzeugung des Objekts und das damit verbundene Eigentum. Über die Dauer der Archivierung kann der gesicherte Zugriff nicht gewährleistet werden. Entweder können die Daten aufgrund interner Authentifizierungsmethoden nicht angezeigt werden oder es kann der Schutz vor unberechtigtem Lesen nicht garantiert werden. Beide Varianten bereiten Probleme, da nicht sichergestellt werden kann, dass Passwörter oder eine eindeutige Identifikation des Nutzers auch in der Zukunft möglich ist.

Zur sicheren Aufbewahrung ist es unbedingt notwendig, dass die Daten zum Zeitpunkt der Einbringung in das Archiv frei zugänglich sind und keine dateiinternen Schutzmaßnahmen genutzt werden. Die beschriebenen Probleme bringen neue Lizenzmodelle mit, die die uneingeschränkte Veröffentlichung der Informationen ermöglichen. Gerade für die LZA bringt das den Vorteil, dass Informationen frei zugänglich werden und keine Kontrolle der Zugriffsrechte erfolgen muss. Weit verbreitete Lizenzmodelle sind Creative Commons [**24**] und GNU GPL in unterschiedlichen Versionen [**36**].

Problematisch bleibt aus heutiger Sicht weiterhin die Ungewissheit in der Entwicklung der rechtlichen Vorgaben. Heute ist die uneingeschränkte Nutzung vieler Daten gegeben. Wie die Zukunft dieser Daten aussieht, bleibt abzuwarten. Mit diesem Hintergrund kämpft nicht nur die Archivierung, vor allem Systeme der Verwaltung (Medienserver) können die Daten zwar erhalten, nur die Steuerung des Zugriffs bleibt zu klären. Aktuell ist ein Trend hin zur Öffnung anfangs verschlossener Daten zu beobachten, mit dem damit verbundenen Ansteigen der Zitationsrate [**38**]. Diesem Trend folgend erfahren großangelegte Digitalisierungsprojekte weitreichende Unterstützung. Wird nicht nur das nationale Recht betrachtet, sind weitere Unterschiede durch Gesetze und Regelungen unterschiedlicher Länder zu erwarten [**69**].

5.4.6 Vorhersehbarkeit

Ein großes Problem stellt die Vorhersehbarkeit dar. Voraussetzungen und Parameter für die Archivierung definieren sich aus dem aktuellen Wissen über Problemstellen und der geforderten Sicherheit der Daten. Die so erzeugten Bedingungen sind so ausgelegt, dass keine Hindernisse zu erwarten sind. Auf dieser Grundlage baut der Archivierungsprozess auf. Gerade die Definition geeigneter Kriterien bereitet Schwierigkeiten, da nur der aktuelle Trend mit berücksichtigt werden kann. An einigen Stellen ist nicht abzusehen, ob und welche Probleme sich ergeben können. Eine Gewissheit lässt sich aus diesen Erkenntnissen nicht ableiten.

Für die Vorhersehbarkeit im Bereich der LZA kann man deshalb immer nur vom aktuellen Stand der Technik ausgehen und versuchen, die Einschränkungen so gering wie möglich zu halten. Wird beispielsweise die Entwicklung der Formatvielfalt in bisheriger Geschwindigkeit fortgeführt, erfordert das vor allem eine Kontrolle der Kompatibilität zu bereits genutzten Archivdateiformaten. Ist die Kompatibilität zu bekannten Formaten nicht sichergestellt, erschwert das die Etablierung standardisierter Formate für die LZA. An dieser Stelle handelt es sich um ein bereits erkanntes Problem, somit sind ein aktives Gegenwirken und die Entwicklung geeigneter Maßnahmen möglich. Als aktives Gegenwirken bestehen Empfehlungen zu Dateiformaten, die für die LZA besonders passend sind. Diese Reduktion der Dateiformate erleichtert die Planung für eine LZA erheblich. Ein weiter Bereich in Frage kommender Daten für einen Medienserver ist darin enthalten. Für den nichtenthaltenen Bereich ist durch Anwendung der Kriterien der LZA die Definition geeigneter Formate möglich (siehe Kapitel 4.6).

Auf Systemebene treten weitere Problemstellungen auf. Im Augenblick ist die Interpretation älterer Dateiformate auf aktuellen Computersystemen unproblematisch, von einigen Ausnahmen abgesehen. Notfalls kann über den Weg der virtuellen Maschine Hardware simuliert werden, um Zugriff auf die Daten zu erlangen. Die Interpretation von Maschinencode bedarf einigen Aufwands, falls die Ursprungshardware nicht mehr vorhanden ist. Dabei ist der Quellcode für den Menschen zwar lesbar,

für Computersysteme sind aber geeignete Compiler für die Interpretation erforderlich. Das Lesen der Daten erfordert Betriebssysteme, die die verschiedenen Hardwarekomponenten ansprechen können, um die Datenströme zu verarbeiten.

Die unterste Ebene stellt die Rechnerhardware dar. In den vergangenen 20 Jahren haben sich bereits die Trägermedien komplett ausgetauscht. Schnittstellen und standardisierte Anschlüsse wurden immer wieder überarbeitet und durch andere ersetzt, oftmals mit fehlender Abwärtskompatibilität. So wurden Disketten inzwischen komplett durch kleine USB-Flashspeicher oder CD/DVD ersetzt. Das Lesen der Disketten bereitet inzwischen aufgrund der fehlenden Hardware Schwierigkeiten. Neue Diskettenlaufwerke sind kaum noch erhältlich. Der Trend zu immer größeren und schnelleren Speichermedien forciert die Entwicklung neuer Hardware. Für die LZA tritt dabei das Problem auf, dass die Wahl eines geeigneten Mediums mit langfristiger Hardwareunterstützung immer schwerer wird. Entweder muss die benötigte Hardware zusammen mit den Daten funktionsfähig erhalten werden (Computermuseum) oder eine ständige Migration der Speichermedien auf aktuell vorkommende Varianten erfolgen. Beide Varianten erfordern ein hohes Maß an Aufwand.

Vorhersehbarkeit lässt sich nicht allumfassend behandeln. Nach der Identifikation der Problemfelder helfen entsprechende Gegenmaßnahmen, den Datenverlust möglichst gering zu halten. Eine hundertprozentige Sicherheit ist dabei nicht gegeben. Es bleibt nur die genaue Beobachtung der Trends und die ständige Kontrolle der Kompatibilität. Diese Maßnahmen sind sehr zeit- und damit kostenintensiv. Als erster Schritt und damit Basis hilft das schärfen des Problembewusstseins, dass eine funktionierende LZA ermöglicht wird und zu erwartende Schwierigkeiten benannt werden.

6 Entwicklung der LZA Funktionalität

Dieses Kapitel beschreibt die Entwicklung geeigneter LZA-Komponenten für einen Medienserver hinsichtlich der bisher genannten Kriterien zu Dateiformaten, Metadatenformaten und der Anforderungen der LZA.

6.1 Funktionsweise

Für die LZA werden spezielle Objekte benötigt, die über geeignete Algorithmen erzeugt werden können. Zur Vereinfachung ist es sinnvoll, diese Algorithmen zu Bibliotheken zusammenzufassen und direkt in den Medienserver zu integrieren. Zwei Mechanismen werden dabei benötigt, die das LZA-Objekt erzeugen können und anschließend den Zugriff auf das Objekt auf dem Medienserver garantieren.

6.1.1 Erstellen des Langzeit-Objekts

Um ein Langzeit-Objekt erstellen zu können, ist zuallererst eine Analyse des vorliegenden Ausgangsprodukts erforderlich. Denn bereits das Ausgangsobjekt muss ein passendes Format aufweisen und darin auch konform erzeugt worden sein. Bei PDF-Dokumenten kann man an dieser Stelle bereits auf verschiedene Probleme treffen:

Version der Datei
Das PDF-Format ist in sich versioniert und jede Version unterstützt einen unterschiedlichen Funktionsumfang. Die Versionen sind abwärtskompatibel, so dass zumindest die neueste Viewer-Generation alle Features anzeigen kann. Für die Archivierungsvariante PDF/A wird allerdings nicht auf die zurzeit neueste PDF-Version 1.7 zurückgegriffen, sondern einige Funktionen ausgelassen und basierend auf Version 1.4 ein Langzeitformat entwickelt. Dieser „Rückschritt“ ist dadurch zu erklären, dass die PDF-Spezifikation immer neue Funktionalitäten vorsieht, die sich nicht mit der LZA vereinbaren lassen. Beispielsweise bereitet das Einbetten von Multimediaobjekten in ein PDF Schwierigkeiten, da der passende Codec zur Anzeige zur Verfügung stehen muss. Weitere grafische Elemente werden ebenso nicht unterstützt (beispielsweise Alpha-Blending).

Unterschiedliche, externe Schrifttypen
Ein weiteres Beispiel ist die Kodierung der verwendeten Schrifttypen. Diese Fonts können sich sehr schnell ändern und stellen eine zusätzliche Anforderung an die Abspielumgebung, die für die Zukunft nicht garantiert werden kann. Aus diesem Grund müssen beispielsweise Schriftarten direkt im Dokument hinterlegt werden, dass sie nicht „verloren“ gehen können.

Bei Bild- oder Multimedia-Objekten kommen andere Kriterien zum Tragen, die beachtet werden müssen:

Farbkodierung
Anders als bei Textobjekten kann bei Bildern die kleinste Veränderung der Farbe des Originals fatale Folgen haben und zu Sinnverfälschungen führen. So ist es beispielsweise bei der Farbgebung wichtig, dass auch die korrekte Anzeige der Farben nachvollzogen werden kann. Denn die Farbe Weiß ist nicht gleich der Farbe Weiß. Besonders bei der Archivierung von Druckvorlagen ist die korrekte Wiedergabe von Farben enorm wichtig.

Datenverlust durch Kompression
Bildformate der LZA unterstützen unterschiedliche Kompressionsverfahren. Darunter befinden sich reversible und irreversible. Für die Archivierung mit der Maßgabe des Erhalts der Daten, ist nach Möglichkeit immer die verlustlose Variante zu wählen. Unterstützt ein Dateiformat nur verlustbehaftete Kompression, ist eine Migration des Ausgangsmaterials in ein Format mit reversibler Kompression sinnvoll.

Unter Beachtung dieser maßgeblichen Kriterien kann aus dem Ausgangsdatenmaterial ein Langzeit-Objekt erzeugt werden, das für die Archivierung die besten Voraussetzungen mitbringt und einfach handzuhaben ist. Einen Großteil des Einsatzgebiets eines Medienservers decken Text- und Bildformate ab. Die dort auftretenden Formate können für die LZA auf zwei bis drei Formate reduziert werden, für die eine Migration möglich ist. Im Multimedia-Bereich ist eine Reduktion auf jeweils ein Format für Audio und Video möglich. Die Erzeugung der Formate selbst kann automatisiert erfolgen und direkt in einem Medienserver integriert werden. Innerhalb des Medienservers ist die Kontrolle der so erzeugten Objektinstanzen realisierbar. Werden weitere Datentypen erforderlich, kann erst nach der Analyse möglicher Dateiformate eine konkrete Umsetzung erfolgen.

6.1.2 Zugriff auf Langzeit-Objekte

Für den langfristigen Erhalt der Daten sind auf der Seite der Speicherlösung geeignete Mechanismen erforderlich, die verschiedene Schritte auf unterschiedlichen Ebenen realisieren. Auf Hardwareebene kommt ein spezieller Datenspeicher zum Einsatz, der einerseits eine lange Lebensdauer mitbringt und andererseits diese Lebensdauer auch mit hoher Sicherheit erreicht beziehungsweise garantiert. Daneben ist das frühzeitige Erkennen sich anbahnender Probleme erforderlich, um Gegenmaßnahmen einzuleiten. In der Praxis hat sich herausgestellt, dass adäquate Mittel beispielsweise Datenspeicherung auf Magnetbändern oder magnetooptischen Medien sind, die in speziell angepassten Hardwaresystemen eingebunden werden. Dort wird über Redundanzen und Kontrollmechanismen sichergestellt, dass die physische Verfügbarkeit gewährleistet bleibt. In der Regel handelt es sich dabei um Robotersysteme, die die benötigten Datenträger aus einer Art Ablage der Kommunikationsschnittstelle zuführen.

Aufgrund der Architektur dieser Speichersysteme ist der Zugriff auf die Daten nicht im Millisekunden-Zeitraum realisierbar. Für die Anbindung der Daten an einem Medienserver bedeutet das zusätzlichen Aufwand, der folgende Kontrollmechanismen erfordert:

Steuerung des Zugriffs
Der Zugriff auf die Daten erfolgt asynchron, auf direkte Anforderung durch den Medienserver. Der Medienserver stößt einen Prozess auf dem Archivsystem an, der die Daten ausliefert. Im Zeitraum zwischen Anfrage und Datenlieferung ist auf der Seite des Medienservers ein eigener Prozess zur Information des Benutzers über das Vorliegen der Daten erforderlich. Werden bestimmte Daten häufig abgerufen, reduziert ein schneller Zwischenspeicher die Wartezeiten für den Benutzer. Auf diesem Zwischenspeicher lagern die Archivdaten redundant für die direkte Auslieferung über den Medienserver.

Verwaltung der Daten
Neben der Auslieferung der Daten ist die Kontrolle der Daten erforderlich. Im Medienserver sind die Informationen vorhanden, die zur Kontrolle der Archivdaten herangezogen werden. Auswertungen der Daten definieren den Zeitpunkt einer notwendigen Datenmigration der Archivdaten. Zusätzlich ist die Steuerung der Zwischenspeicherinformationen über den Medienserver zu realisieren. Daraus leitet sich ab, dass der sichere Archivspeicher rein zur Datenablage zum Einsatz kommt. Die Steuerung des Datenbestands erfolgt über den Medienserver.

Für die Archivierung und den langfristigen Erhalt der Daten eines Medienservers sind neben dem sicheren Archivspeicher und dem schnellen Medienspeicher ein Zwischenspeicher erforderlich. Die drei Speichersysteme unterschieden sich in Größe, Geschwindigkeit und Einsatzzweck und sind daraufhin optimiert. Die Verwaltung der Speicherlösungen erfolgt über den Medienserver.

6.2 Validierung notwendiger unterstützter Datentypen

Für die Anbindung und Umsetzung der LZA an einen Medienserver ist die Validierung notwendiger und sinnvoller Dateiformate Grundvoraussetzung. In Kapitel 4 erfolgt die Kategorisierung verschiedener Dateiformate und die abschließende Bewertung der Eignung in Bezug auf die LZA.

Durch die Kategorisierung ist eine Eingrenzung der Formatvielfalt auf einen Basissatz an Dateiformaten je Kategorie möglich. Zur Validierung ist es sinnvoll, die bisher verwendete Unterteilung in die verschiedenen Teilgebiete weiterhin zu verwenden.

6.2.1 Textformate

Im Medienserverumfeld können Textformate auf ein Dateiformat reduziert werden, das sowohl für die Anzeige im Medienserver, als auch für die Archivierung zum Einsatz kommen kann. Bereits Untersuchungen des Kompetenznetzwerk Langzeitarchivierung nestor [**62**] zeigten, dass das PDF/A-Format ausreichend ist. Neben PDF/A liefert XML weitere Funktionen, um eine umfassende Unterstützung für Textformate bieten zu können. Für die LZA werden nur diese beiden Formate betrachtet.

Die Metadatenspeicherung erlauben beide Formate innerhalb der Dateistruktur. Einige Metadaten liefert PDF/A bereits mit, weitere lassen sich integrieren. Bei XML ist die Integration aufgrund der Flexibilität ebenfalls problemlos möglich. Weitere textbasierte Formate sind nach den augenblicklichen Erkenntnissen nicht notwendig. Sollten die gewählten dennoch nicht ausreichen, muss entweder die Möglichkeit der Konvertierung in ein bereits genutztes Dateiformat geprüft werden oder als letzte Möglichkeit müssen weitere Formate aufgenommen werden. Bei der Neuaufnahme sind vor allem die Kriterien der Langzeitarchivierung (siehe Kapitel 5) zu berücksichtigen.

6.2.2 Grafikformate

Bei Grafikformaten fällt die eindeutige Entscheidung hin zu einem bestimmten Format schwieriger, als bei textuellen Formaten. Durch die große Vielfalt einerseits und die häufig anzutreffenden Formate andererseits ist eine Einschränkung auf eine kleine Auswahl an Formaten möglich. Darunter befinden sich auch solche, die für die LZA geeignet sind (siehe Kapitel 4.6).

Nach Meinung von Experten (z.B. nestor), eignet sich das TIFF-Format am besten für die LZA. In diesem Format werden bereits eine ganze Reihe an Maßnahmen für den langfristigen Erhalt unterstützt (siehe dazu auch Kapitel 4.3.5). Im Bereich der Retrodigitalisierung, dem nachträglichen Digitalisieren von vorhandenen analogen Quellen, ist die Forderung nach TIFF-Dateien einfach durchsetzbar. Leider reicht diese Forderung in der Realität nicht aus, um eine komplette Abdeckung zu erzielen. Denn spätestens seit der großen Verbreitung der Digitalkameras hat sich die Nutzung des JPEG-Formats noch einmal verstärkt. Da bei diesem Dateityp bereits während der Erzeugung ein Teil der Bildinformation der Kompression zum Opfer fällt, ist dieser Verlust nicht wieder zu rekonstruieren. Die spätere Verwaltung der Dateien im Archivierungsumfeld stellt kein Problem dar, da über die Offenlegung des Formats jederzeit geeignete Interpreter zur Verfügung gestellt werden können. Wie aber bereits in Kapitel 4.3.4 geschrieben, bringt das JPEG-Format eine Reihe an Nachteilen für die LZA mit, beispielsweise die nichteindeutige Darstellung.

6.2.3 Multimediaformate

Bei Multimediaformaten ist eine generelle Abgrenzung auf ein einziges Dateiformat für die LZA noch um ein vielfaches schwieriger [**23**]. Hier bleibt zuallererst noch eine grundlegende Unterscheidung in Audiomaterial und Videomaterial.

Audiodateien erfahren durch die weite Verbreitung tragbarer Musikspieler große Beliebtheit. Überwiegend kommt dabei das MP3-Format zum Einsatz. Für die Archivierung ist MP3 aufgrund der verlustbehafteten Kompressionsverfahren nicht optimal. Aus Sicht der LZA ist WAVE deutlich besser geeignet. Es besteht die Möglichkeit, aus MP3-Dateien WAVE-Dateien verlustfrei zu transkodieren. Für die Nutzung innerhalb eines Medienservers eigenen sich MP3-Dateien aufgrund ihrer geringeren Größe besser als WAVE-Dateien. Beide Formate werden aber problemlos unterstützt.

Bei Videoquellen ist die Unterscheidung zwischen analog erzeugten und bereits digital vorliegenden Medien notwendig. Bei den analogen Videokassetten im VHS-Format ist eine Retrodigitalisierung erforderlich. Hier kann bei der Digitalisierung ein geeignetes Format zum Einsatz kommen. Als Standardcontainer hat sich MPEG durchsetzen können mit MP4 als Kompressionsalgorithmus. Für die LZA ist dieses Format sehr gut geeignet (siehe Kapitel 4.6). Liegt das Videomaterial bereits digital vor, ist entweder eine Transkodierung erforderlich oder das Material ist bereits im MPEG-Format. Bei der Transkodierung kommt es zu Datenverlust, da sich die unterschiedlichen Videoformate nicht verlustfrei ineinander überführen lassen. Nachdem allerdings bereits während der Erzeugung des Videomaterials aufgrund der großen Datenmengen eine Kompression erfolgt, kann über die Nutzung passender Parameter der Verlust durch die Transkodierung gering gehalten werden.

Medienserver mit Datenaustausch über das Internet nutzen eigene Formate, die besonders auf die Reduktion der Datengröße hin optimiert sind. Gerade die Echtzeit-Übertragung von Video- und Audioinformationen erfordert komprimierende Formate. Über diese Kompression gehen einerseits Informationen verloren, andererseits halten diese Formate in Anlehnung an Vorschaubilder im Grafikbereich genug Informationen für eine Vorschau bereit. Mit dieser doppelten Datenhaltung wird einerseits das Original erhalten, andererseits ein schneller Zugriff auf eine reduzierte Variante bereitgehalten. Diese reduzierten Formate sind MP3 für Audio- und FLV für Videodaten.

6.3 Grundsätzliche Überlegungen zur Erzeugung des LZA-Formats

Neben den bereits festgelegten Kriterien (die teilweise nicht beeinflussbar sind), ergeben sich weitere, die entweder für eine sinnvolle LZA notwendig sind oder eine größere Flexibilität für die Zukunft ermöglichen.

6.3.1 Formaterweiterbarkeit

Die bisher diskutierten Dateiformate stellen den aktuellen Stand der Entwicklung im Bezug auf LZA und die Nutzung innerhalb der Medienserver dar. Für die Umsetzung bedarf es der Klärung, ob generell nur ein Dateiformat je Kategorie eingesetzt wird oder auch mehrere denkbar sind. Nachdem sich nicht alle Bereiche sinnvoll auf ein einziges Format reduzieren lassen, erfordert die Umsetzung der Archivierungsschnittstelle die Möglichkeit der Erweiterung um zusätzliche Dateiformate. Nur so ist es möglich, auf die Weiterentwicklung im Bereich der Medienserver und zusätzliche Kategorien reagieren zu können.

Für bereits bekannte Kategorien ist mindestens jeweils ein Dateiformat besonders gut für die LZA geeignet und gleichzeitig auch im Medienserverumfeld etabliert. Der Betreiber des Medienservers entscheidet über die Inhalte, die über den Server angeboten werden. Ebenso liegt es in der

Verantwortung des Betreibers, das passende Archivformat für diese Daten zu ermitteln und auch einzusetzen.

6.3.2 Metadaten

Die Integration der Metadaten und die Speicherung zusammen mit dem Digitalisat werfen grundlegende Fragen auf:

- Existieren bereits Standards für diesen Anwendungsbereich?
- Welche Standards kommen im Umfeld der eigenen Nutzung zum Einsatz?
- Welcher Standard passt am besten zur eigenen Anwendung oder zum Kontext?
- Welcher Standard entspricht am besten den eigenen Anforderungen?

Basis der Überlegung ist die Suche nach einem geeigneten Standard für die LZA und den Medienserver. Dieser Punkt ist aufgrund der vorliegenden Formate aus unterschiedlichen Einsatzgebieten nicht eindeutig zu beantworten. Denn das heterogene Dateiumfeld bringt eine Reihe an Metainformationen bereits mit, deren Schnittmenge aus den verschiedenen Kategorien sehr gering ausfällt. Standards liefern nur einen Basissatz an Information, der gerade im Bereich der Grafik unzureichend für eine genaue Beschreibung und Auswertung ist (siehe Kapitel 3.9).

Ausweg bietet die Kombination verschiedener Metadatenformate. Nachdem meist nur Attribute mit Werten als Tupel betrachtet werden, reicht die Auswertung der Attributnamen aus. Als Basissatz unterstützen alle Dateitypen das Dublin-Core Schema (siehe Kapitel 3.5.1). Zusätzlich ist die Extraktion technischer Metadaten aus den Dateiformaten möglich.

Neben diesen Informationen kommen im Medienserver deutlich mehr Attribute zum Einsatz, deren Nutzung auch für die LZA möglich ist. Häufig werden Beziehungen der Objekte untereinander über Hierarchien hinterlegt. Diese Informationen bringen neue Möglichkeiten für die LZA, da sie Ähnlichkeiten der Objekte dokumentieren. Liegen bereits weitreichende Daten über Objekte vor, ist deren Erhalt unbedingt erforderlich. Durch den Einsatz flexibler semantischer Metadatenformate (RDF oder XML) lassen sich auch diese Daten problemlos in den Metadatenbestand integrieren.

Für die Interoperation unterschiedlicher Systeme stellen Medienserver verschiedene Formate bereit (siehe Kapitel 3.6). Bei geschickter Kombination bringen die Medienserver damit bereits alle notwendigen Voraussetzungen mit, dass auch Metadatenformate für die LZA erzeugt werden können. Durch die Flexibilität der Medienserver lassen sich damit die weiteren Fragen zum geeigneten Metadatenstandard beantworten: Die Kombination aus Medienserver als Konfigurationswerkzeug mit gängigen Metadatenstandards liefert die geforderten Funktionen für die LZA.

6.3.3 Änderungen auf Byte-Ebene

Bei der Einbettung der Metadaten in eine Datei verursachen unterschiedliche Verfahren geringe oder sehr große Änderungen an der Ausgangsdatei. Je nach Position der Metadaten innerhalb der Datei reicht das vom Anhängen der Daten am Ende der Ursprungsdatei bis hin zur kompletten Überarbeitung. Gerade diese Stelle birgt große Gefahren, da aufgrund der Änderungen Daten verloren gehen oder Dateien zerstört werden können. Ebenso muss das Ergebnis der Überarbeitung innerhalb der Dateiformat-Spezifikation gültig sein. Nur so ist das spätere Interpretieren der Daten sichergestellt.

Eine Reduktion der Gefahr ist zu erzielen, wenn die Änderungen auf Dateiebene möglichst gering sind. Bei LZA-Formaten ist zu beobachten, dass es sich meist um Containerformate handelt, die Erweiterungen um zusätzliche Container problemlos ermöglichen. Die Formalisierung des Erzeugungsverfahrens erlaubt nach Abschluss der Änderungen auf Dateiebene eine einfache Kontrolle

des Ergebnisses. Neben der Kontrolle ist die Definition der Korrektheit des Prozesses möglich. Vergleichsoperationen zwischen Original und überarbeitetem Objekt reichen dafür aus.

6.3.4 Umkehrbarkeit des Verfahrens

Neben der korrekten Erzeugung des Archivformats ist die spätere Reproduktion des Originals erforderlich. Falls das Original aufgrund fehlgelaufener Migrationsschritte oder bisher unbekannter Gründe angefordert wird, muss das Archivsystem in der Lage sein, das Original wiedererstellen zu können. Die bitgenaue Kopie des Originals lässt sich über die Kontrolle der Dateiprüfsumme sicherstellen. Die Prüfsumme selbst ist als Metadatum abzuspeichern.

Die Umkehrbarkeit des Archivierungsverfahrens stellt damit sicher, dass keine Information verloren gehen kann, die zum Zeitpunkt der Aufnahme in das Archiv über das Objekt vorlag. Damit ist der Weg zurück zum Original offen und die Gegenreaktion bei Problemen mit dem Objekt gegeben.

6.4 Kodierung der Metadaten

Neben dem digitalen Objekt sind die Metadaten von entscheidender Wichtigkeit für die Archivierung. In den vorangegangenen Kapiteln wurde bereits ein Überblick über gängige standardisierte Formate gegeben, die in unterschiedlichen Bereichen zum Einsatz kommen. Durch eine geschickte Kombination und Auswahl geeigneter Definitionen lassen sich die Vorteile einzelner Standards miteinander kombinieren und somit ein geeignetes Format erzielen.

6.4.1 Verfahren und Formate

Durch die Einführung des Semantic Web bieten sich neue Möglichkeiten, Metainformationen maschinenlesbar zu hinterlegen. Dafür kommt das RDF-Format zum Einsatz (siehe Kapitel 3.4.1). Dadurch, dass es sich bei RDF lediglich um ein semantisches Format handelt, muss ein geeigneter Satz an Metainformationen gefunden werden, der für die Archivierung herangezogen wird. In der Kombination mit Dublin Core, einem weit verbreitetem beschreibenden Format, kann damit der Grundbedarf an Informationen abgedeckt werden. Zusätzlich entscheidet der Administrator eines Archives, welche Metadaten für ein Digitalisat entscheidend sind und kann diese für die Langzeitarchivierung mit einbinden. Es zeigt sich, dass oftmals bei den Metadaten auch Attribute gepflegt werden, die eher für die tägliche Verwaltung, als für die Charakterisierung eines Objektes herangezogen werden. Daraus resultiert, dass nicht alle Metainformationen sinnvollerweise der LZA zugeführt werden müssen.

Als Ausgangspunkt ist davon auszugehen, dass zu einem Digitalisat unterschiedliche Metaattribute zur Verfügung stehen. Diese Attribute kommen dabei aus den unterschiedlichen Bereichen, die es für Metadaten gibt. Bei Medienservern werden Tupel, bestehend aus einem Attributnamen und dessen Wert, gespeichert und können zur Weiterverarbeitung herangezogen werden. Dafür kommen spezielle Methoden zum Einsatz, die alle Metadaten eines Objektes liefern können.

6.4.2 Erzeugung des Formats

Über eine Schemadefinition erhalten Objekte Zusatzinformationen. Damit wird über das Schema festgelegt, welche Metadatenattribute gepflegt werden können oder müssen (Pflichtfelder). Jedes dieser Metadatenattribute besitzt über die Definition eine Reihe an Definitions-Informationen, die grundlegende Eigenschaften festlegen. Darunter ist unter Anderem die Definition des Metadatentyps. Anhand dieses Typs kann das System die korrekte Darstellung des Metadatenfeldes in der jeweiligen

Ausgabe formatieren. Zusätzlich ist in der Definition beispielsweise der Wertebereich der Auswahllisten definiert, die für die Eingabe herangezogen werden.

Die unterschiedlichen Darstellungsformen der Metadaten eines Schemas definieren Masken. Masken legen eine Art Filter über die Summe der zur Verfügung stehenden Attribute eines Objekts und bestimmen damit die Reihenfolge der anzuzeigenden Metadatenattribute. Masken existieren für unterschiedliche Zwecke. Es gibt Masken für verschiedene Anzeigeformen (Kurzanzeige, Vollanzeige etc.) und Spezialmasken für den Export. Masken „kennen" die Art und Weise, auf die der Inhalt darzustellen ist und liefern einen korrekten Stream im passenden Format, der dann wiederum innerhalb des Medienservers weiterverwendet wird.

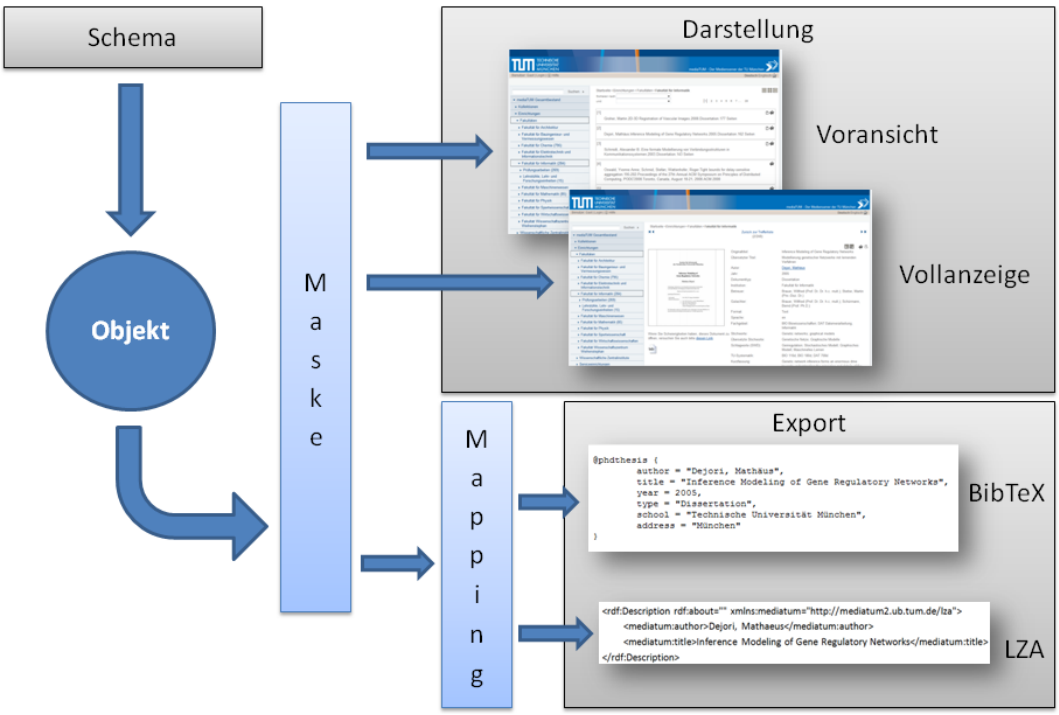

Abbildung 6-1: Schematische Darstellung der Anzeige/Export-Möglichkeiten

Eine besondere Funktion erfüllen Mappings. Über die Mappingdefinitionen können zusätzliche Angaben über die Formatierung hinterlegt werden. Dadurch wird es möglich, Attribute eines Objekts in XML-Form zu bringen und Kopf- und Fußzeile zu erzeugen, so dass am Ende formatspezifisch korrekte Streams erzeugt werden können. Gerade im Bereich der Exportfunktionalität kann mithilfe der Mappings der Metadatenbestand eines Objekts in nahezu jede beliebige Form gebracht werden. Zusätzlich bietet diese Art der Erzeugung den Vorteil, dass dynamisch durch Konfiguration zusätzliche Formate erzeugt werden können. Abbildung 6-1 zeigt schematisch die Zusammenhänge der einzelnen Komponenten und mögliche Ausgabeformate.

6.4.3 Extraktion der Metadaten aus dem LZA-Format

In einer weiteren Funktion kann der Medienserver beim Einspielen eines Objekts erkennen, ob Zusatzinformationen in der Datei enthalten sind. Die Erkennung läuft über einen zweistufigen Algorithmus, der zuerst die infrage kommenden Stellen innerhalb einer Datei untersucht und in einem zweiten Schritt die Metadaten extrahiert. Bei der Extraktion der Metadaten unterscheidet der Algorithmus zwischen binär und textuell gespeicherten.

Sind Metadaten vorhanden, werden sie der Weiterverarbeitung zugeführt. Der umgekehrte Weg der Erzeugung kann das leisten. Ergebnis ist ein Objekt und alle dazu verfügbaren Metainformationen. Problematisch ist an dieser Stelle, dass die gefundenen Attribute korrekt der ursprünglichen Definition zugeordnet werden müssen. Änderungen an der Schemadefinition im Medienserver wirken sich negativ aus, falls die Änderungen nicht auch in den Archivdaten nachvollzogen wurden. In so einem Fall können unterschiedliche Wege eingeschlagen werden:

- Es werden nur Metadatenattribute importiert, die auch komplett auf die aktuelle Schemadefinition passen. Alle anderen Werte werden verworfen und gehen somit für den Medienserver verloren. Innerhalb des Digitalisats bleiben die Informationen solange erhalten, bis eine neue LZA-Variante erzeugt wird. Dabei werden dann allerdings alle alten Metadaten überschrieben und dadurch auch die nichtimportierten Werte gelöscht.
- Es werden alle Metadatenattribute importiert. Dieses Verfahren kann angewendet werden, weil der Medienserver dann zwar auf der einen Seite über die Masken nur Attribute anzeigen kann, die auch in der aktuellen Schemadefinition vorgesehen sind. Andererseits kann ein flexibles System nichtgenutzte Werte im Hintergrund verwalten, auch wenn keine Anzeige erfolgt. Zusätzliche Kontrollmechanismen helfen bei der Identifikation solcher Werte und unterstützen bei der Verarbeitung und ggf. bei der Korrektur.

Die geeignetere Variante ist die zweite, da die Gefahr des Datenverlusts hier deutlich geringer ist. Damit ist auch für die Umsetzung der zweite Weg der bessere für den Einsatz eines Medienservers mit angebundener LZA.

6.5 Szenarien

In den verschiedensten Szenarien kommen Teilkomponenten der Langzeitarchivierung zum Tragen. Das reicht vom Einspielen und der Bearbeitung der Daten, die interne Verwaltung der Daten mit der Anbindung an die Archivierungsschnittstelle und die anschließende Ausgabe an den Benutzer.

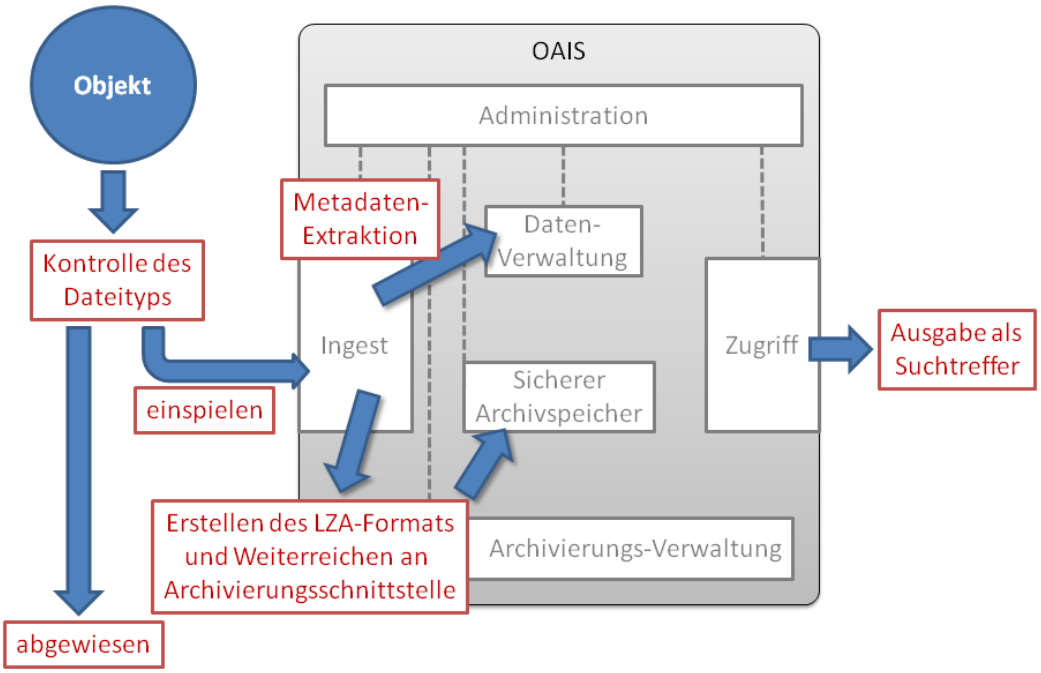

Abbildung 6-2: Notwendige Eingriffe am Beispiel des OAIS-Modells

Die so genutzten Teilkomponenten erfordern unterschiedliche Schnittstellen, um mit der Medienserversoftware kommunizieren zu können. Diese Schnittstellen müssen jeweils auf beiden Seiten abgestimmt werden und flexibel Erweiterungen oder Änderungen am Szenario erlauben. Abbildung 6-2 zeigt schematisch den Prozess des Einspielens eines Objekts in den Medienserver und die angeschlossenen Kontrollmechanismen, zusätzlich das Abrufen der Daten. Die unterschiedlichen Szenarien werden nachfolgend näher beschrieben.

6.5.1 Einspielen der Daten

Für jeden vom System unterstützten Dateityp ist eine Kontrolle des Dateiformats vorgesehen. Bei dieser Kontrolle werden zugleich noch notwendige Constraints kontrolliert, die zum jeweiligen Dateityp gehören. Dabei können beispielsweise die Versionsnummer des Typs oder gewisse zusätzliche Forderungen validiert werden, die eine Datei erfüllen muss. Die genauen Forderungen werden entweder vom jeweiligen Betreiber vorgegeben oder sind in der Grundeinstellung des Medienservers definiert. Grundeinstellung meint dabei Maßnahmen für ein korrektes Arbeiten und das Eingehen auf bekannte Probleme bestimmter Dateiformate. Falls die Kontrolle negativ ausfällt, wird die Datei abgewiesen.

Bereits während des Einspielens erfolgt die Vorbereitung der LZA-Datei mit der automatisiert möglichen Extraktion der technischen Metadaten aus der Datei. Zusätzlich werden weitere Verarbeitungsschritte mit angestoßen. Die genaue Funktion liegt an der Stelle immer beim vorliegenden Dateityp und ist alleine von diesem und der Objektkategorie abhängig. Die endgültige Erzeugung des LZA-Objekts ist an dieser Stelle noch nicht möglich, da die Metadaten nicht vollständig vorliegen. Erst nach der manuellen Nachbearbeitung ist das möglich (siehe auch Abbildung 6-2).

6.5.2 Interne Weiterverarbeitung

Nach einem erfolgreichen Einspielen kommen die internen Abläufe zum Tragen, die die Verwaltung der Originaldaten, der LZA-Variante und der Arbeitskopie(n) übernehmen. Die Änderungen an den Metadaten des Objekts auf dem Medienserver müssen in regelmäßigen Abständen an die LZA-Variante weitergegeben werden. Neben diesen Operationen sind weitere Funktionen erforderlich, die im folgenden Kapitel näher beschrieben werden.

6.5.3 Ausgabe der Daten

Der Medienserver unterscheidet mehrere Versionen oder Varianten eines Objekts. Je nach angeforderter Instanz sind unterschiedliche Verarbeitungsschritte erforderlich. Eine grundlegende Unterscheidung ist möglich:

Anforderung der Arbeitskopie
Die Arbeitskopie eines Objekts kann eine Vorschauansicht oder der Volltext sein. Diese Daten liegen direkt im Medienserver vor und können ohne Umwege und ohne Zugriff auf die Archivierungskomponenten direkt ausgeliefert werden.

Anforderung des Originals
Die Original-Daten eines Objekts liegen im Archivspeicher. Der Medienserver stößt auf dem Archivsystem eine Anfrage nach dem Original an. Der Zeitraum zwischen der Anfrage und der Auslieferung der Daten ist aufgrund des eingesetzten Speichers deutlich zu lang für eine synchrone Abarbeitung. Der Nutzer des Medienservers erhält vom Medienserver Informationen über den aktuellen Stand der Anfrage. Liefert das Archivsystem die Daten, werden diese an den Nutzer weitergereicht. Zur Verkürzung der Wartezeiten erfolgt die Zwischenspeicherung der Archivdaten auf schnellem Speicher, der direkt an den Medienserver angebunden ist. Damit ist die direkte Auslieferung oft angefragter Archivdaten möglich. Die Auslieferung der Daten über das Internet bringt weitere Probleme mit sich, die

über das Zwischenspeichern gelöst werden können. Verbindungsabbrüche und daraus resultierende erneute Anfragen lassen sich auf diese Weise effizient vermeiden.

6.5.4 Migration eines Objekts

Neben dem Zugriff auf die Daten erfordern Migrationsschritte der Objekte und die Aktualisierung der LZA-Daten definierte Abläufe. Der eigentliche Migrationsschritt kann nicht vollautomatisiert vom Medienserver angestoßen werden. An dieser Stelle ist der Eingriff des Betreibers des Servers erforderlich, der die Datei in das entsprechende Format bringen muss. Liegt die Datei in der korrekten Form vor, ist der restliche Ablauf automatisierbar, die Metadateneinbettung in die Datei und die Übertragung der Daten an das Archivsystem.

Wichtig ist auch das Erkennen des Zeitpunkts, an dem ein Migrationsschritt erforderlich wird. Eine Kategorisierung der Daten auf dem Medienserver ermöglicht es, ähnliche und damit ebenfalls für eine Migration erforderliche Daten zu ermitteln und diese der Migration zuzuführen.

7 Umsetzung der LZA-Funktionalität

Dieses Kapitel beschreibt die Umsetzung der von mir entwickelten und beschriebenen Langzeitarchivierungskomponente für ausgewählte Dateiformate und bewertet die gefundene Lösung, unter anderem hinsichtlich der Umsetzbarkeit und der Performanz. Außerdem werden Probleme und mögliche Erweiterungen aufgezeigt.

7.1 Grundlegendes zur Umsetzung

In Kapitel 6 werden unterschiedliche Möglichkeiten und notwendige Überlegungen für die Anbindung einer LZA-Komponente an einen Medienserver dargestellt. Ausgehend von diesen Möglichkeiten ist die beispielhafte Umsetzung im Medienserverprojekt mediatum der TUM zu finden [**74**]. Die produzierten Algorithmen und Methoden sind Bestandteil der Software und direkt integriert. Dabei werden die im Augenblick am häufigsten anzutreffenden Themenfelder berücksichtigt und eine Umsetzung für Text- und Grafikformate realisiert.

Generell ist die Architektur der LZA-Format-Erzeugung derart ausgelegt, dass zu jeder Zeit weitere Dateitypen mit in das Archivierungskonzept aufgenommen werden können. Erreicht werden kann diese Forderung über eine offene API, die nur die grundlegenden Methoden nach außen weiterreicht. Der Kernbestandteil variiert von Dateityp zu Dateityp und kann sehr unterschiedlich sein. Vorteil einer derartigen Modellierung ist, dass der eingesetzte Medienserver keine Kenntnisse über die genauen Abläufe besitzen muss und lediglich die Nutzerszenarien (siehe Kapitel 6.5) erfüllen muss. Dadurch ist die Anbindung unterschiedlicher Archivsysteme möglich, die angepasst auf den Dateityp LZA-Objekte erzeugen und den Zugriff gewähren.

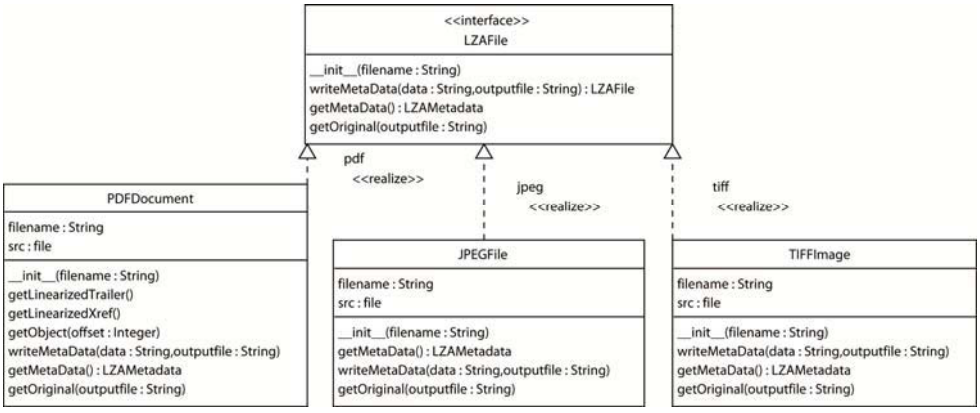

Abbildung 7-1: Klassendiagramm der LZA-Erzeugung

Abbildung 7-1 zeigt beispielhaft das Klassendiagramm mit der Interfaceklasse (LZAFile) als Kernkomponente. Implementierungen der Hauptmethoden finden sich für die unterschiedlichen Dateitypen wieder. Die Hauptmethoden für das Erzeugen der Archiv-Variante (writeMetaData), Auslesen vorhandener Metadaten (getMetaData) und Rückerzeugung des Originals (getOriginal) werden direkt über die API angesprochen. Alle weiteren Methoden dienen der internen Verwaltung.

7.2 Erzeugung für Textformate

Wie bereits vorangegangen beschrieben, genügt im Bereich der Textformate die Umsetzung für die Dateitypen XML und PDF, um eine umfangreiche Abdeckung für Medienserver zu erreichen.

7.2.1 XML

Das hierarchisch angelegte XML-Format definiert eine eindeutige Wurzel als oberstes Element innerhalb der Struktur. Das Hinzufügen weiterer Informationen ist am einfachsten über folgende Vorgehensweise zu erreichen:

- Definition eines neuen Wurzelknotens,
- Eingliederung der originalen Wurzel unterhalb der neuen Wurzel,
- Schaffung eines zusätzlichen Metadatenknotens unterhalb der neuen Wurzel.

Mit dieser einfachen Umformung kann beliebiger Metadateninhalt in ein vorliegendes XML-Objekt eingebettet werden (siehe auch Abbildung 7-2).

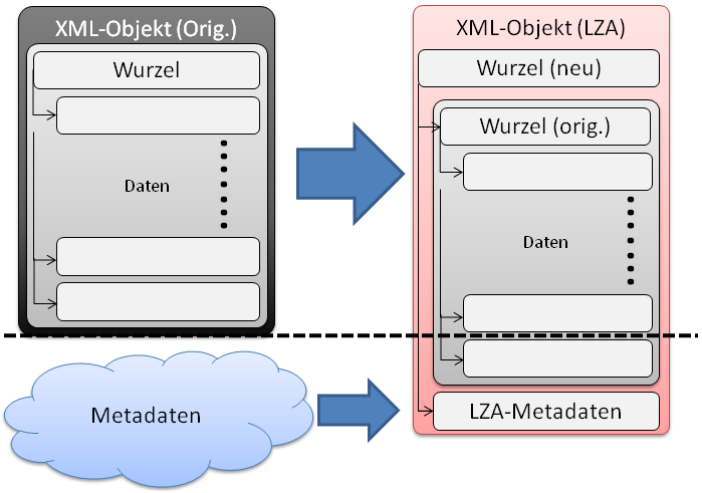

Abbildung 7-2: Schematische Darstellung LZA-Erzeugung (XML)

7.2.2 PDF

Bei der Verarbeitung von PDF-Dateien ist in erster Linie der Unterschied zwischen linearisierten und „normalen“ PDF-Dateien zu beachten (siehe Kapitel 4.2.4). Ausgehend von dieser Unterscheidung kann ein Algorithmus die entsprechenden Dateien mit den eingebundenen Metadaten erzeugen. Die Unterscheidung selbst kann über die Suche nach dem Schlüsselwort „linearized“ innerhalb der PDF-Datei erfolgen. Dabei sieht der PDF-Standard vor, dass das Schlüsselwort zu Beginn einer Datei in einem speziellen Dictionary zu finden ist. Tests mit vorliegenden Dateien haben ergeben, dass das Lesen der ersten 500 Zeichen ausreicht, um eine sichere Unterscheidung vornehmen zu können.

Standard-PDF
Bei einem Standard-PDF kann direkt an den alten Trailer ein neuer Bereich mit den zusätzlichen Daten angefügt werden. Dabei wird die Ausgangsdatei bitgenau übernommen. Abbildung 7-3 zeigt schematisch den Übergang von der Ausgangsdatei zur LZA-Variante.

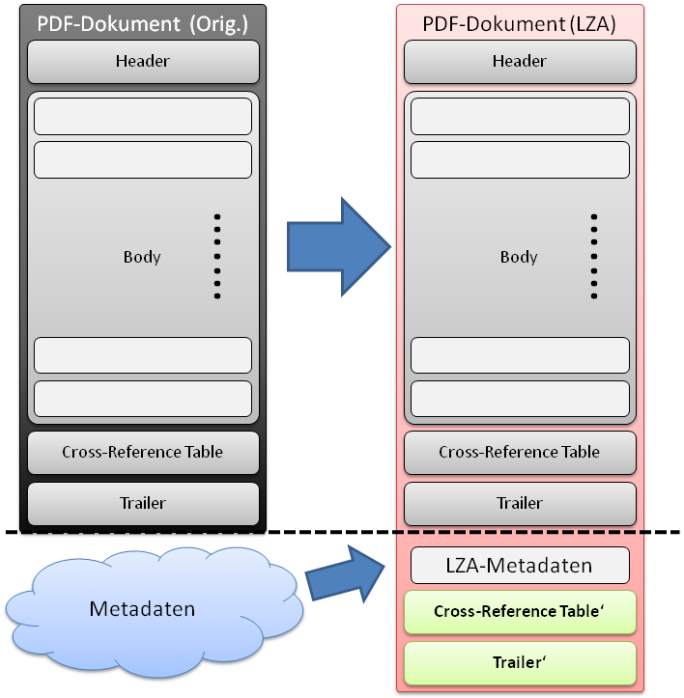

Abbildung 7-3: Schematische Darstellung LZA-Erzeugung (PDF)

Die eigentlichen Zusatzinformationen werden in Form der LZA-Metadaten in einen eigenen Datencontainer verpackt. Dieser Container ist analog aufgebaut zu den Containern im Body der Datei. Code 7–1 zeigt einen Beispiel-Container der ID 18 mit dem zusätzlichen Attribut Length, der Länge des Containers, und den eigentlichen Inhaltsdaten.

```
18 0 obj
stream
<< /Type/Metadata/Subtype/XML/Length 227>>
        <?xpacket begin="i»¿" id="mediatum metadata"?>
                <lza:data>
                    [Metadaten des Objekts]
                </lza:data>
        <?xpacket end="w"?>
endstream
endobj
```

Code 7–1: Beispiel LZA-Metadaten-Container

An der Stelle ist es entscheidend, dass einige Angaben zum Objektcontainer korrekt gesetzt werden. Das sind unteranderem die Objektid und die Länge des Inhalts. Dabei lässt sich die ID aus der alten Cross-Reference Table erzeugen und die Länge der Daten ist auch bekannt.

Zusätzlich zu den einzubettenden Metadaten im Container muss noch die PDF-Dateistruktur wiederhergestellt werden. Dafür muss noch eine neue Cross-Reference Table und ein neuer Trailer erzeugt werden. Die Cross-Reference Table kann analog zur bereits vorhandenen generiert werden und die absoluten Positionsangaben des neuen Objekts können neu errechnet werden (siehe Code 7–2).

```
xref
18 1
0000057303 00000 n
```

Code 7–2: Beispiel Cross-Reference Table nach der Überarbeitung

Für den Trailer sind noch zusätzliche Informationen aus der ursprünglichen PDF-Datei zu erzeugen. Es handelt sich hierbei überwiegend um absolute Sprungadressen zum Aufbau der internen Dateistruktur. Code 7–3 zeigt den überarbeiteten Trailer mit Attributen zur Anzahl der Container, der eindeutigen ID, der Sprungadresse zum vorhergehenden Trailer und die Position der aktuellen Cross-Reference Table.

```
trailer
<< /Size 19 /Root 1 0 R /Info 7 0 R
/ID[<235D636A74BF0C4B811256479BD2BA02><235D636A74BF0C4B811256479BD2BA02>] /Prev 56787>>
startxref
57617
%%EOF
```

Code 7–3: Beispiel-Trailer nach der Überarbeitung

Von großer Bedeutung sind dabei die Angaben zur Cross-Reference Table und zum vorangegangenen, ursprünglichen Trailer. Erst dadurch wird eine korrekte Darstellung des PDF-Dokuments möglich.

Linearized-PDF
Die grundsätzliche Vorgehensweise bei linearisierten PDF-Dateien ist in weiten Teilen analog zur normalen PDF-Datei auszuführen. Allerdings ist zu beachten, dass bei der speziellen Variante die Position der Cross-Reference Table und des Trailers an einer anderen Stelle in der Ausgangsdatei zu finden sind. Aufgrund der durchgeführten Optimierung sind diese beiden Strukturobjekte nicht am Ende der Datei, sondern zu Beginn zu finden.

Beim Einbetten des neuen Objektcontainers ist außerdem die korrekte ID-Vergabe zu berücksichtigen, die strukturbedingt höher als bei normalen PDF-Dateien ausfällt. Die Erzeugung selbst erfolgt analog zur Standard-PDF-Datei.

7.3 Erzeugung für Grafikformate

Aus der Vielzahl der verfügbaren Grafikformate eignet sich das TIFF-Format am besten für die LZA. Aufgrund der weiten Verbreitung und massiven Nutzung auf dem Medienserver ist außerdem als weiteres Format JPEG erforderlich.

7.3.1 TIFF

Die interne Dateistruktur einer TIFF-Datei ist über verkettete Listen aufgebaut, den Image File Directories (IFD). Dabei besitzt ein IFD eine Referenz auf den direkten Nachfolger. Der letzte Eintrag nutzt 0000 als Zeichen dafür, dass keine weiteren Directories mehr folgen.

Beim Einbetten von Zusatzinformationen können diese Informationen an das Ende der Datei angehängt werden. Damit eine gültige TIFF-Datei wiederhergestellt werden kann, muss der Zeiger des bisher letzten IFD auf das neue letzte Directory verweisen. Zusätzlich muss noch ein entsprechendes neues IFD erzeugt werden, das wieder mit dem Terminator 0000 gekennzeichnet ist (siehe Abbildung 7-4).

Das neu erzeugte IFD beinhaltet als einzigen Directory Entry den Stream aus den Metadaten und wird über die ID (83BB) gekennzeichnet. Auch bei TIFF-Dateien werden alle Sprungadressen (z.B. in den Directories) zu den jeweiligen Daten mit absoluten Bitangaben abgelegt. Die neuen Sprungadressen für den Directory Entry mit den Metadaten kann man relativ einfach aus den vorhandenen Informationen erzeugen und an das Ende der bisherigen Datei anhängen.

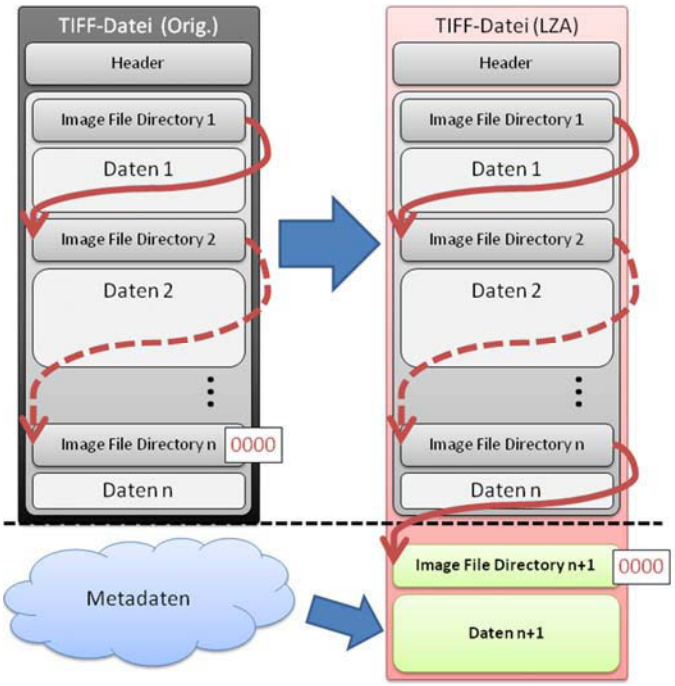

Abbildung 7-4: Schematische Darstellung LZA-Erzeugung (TIFF)

7.3.2 JPEG

Die interne Struktur einer JPEG-Datei ist über Marken definiert. Die Anzahl möglicher Marken ist aufgrund der Länge von zwei Byte begrenzt und jeder Marke ist eine bestimmte Funktion zugeordnet. Zusätzliche Daten können nur unter Zuhilfenahme einer bekannten Marke mit eventueller Umwidmung eingebettet werden.

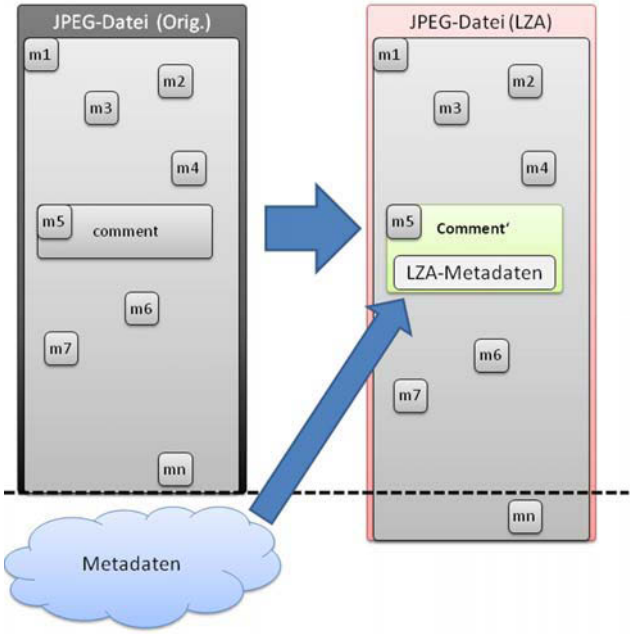

Abbildung 7-5: Schematische Darstellung LZA-Erzeugung (JPEG)

Zur Speicherung der Metadaten für die Archivierung wird deshalb der Comment-Marker mit herangezogen. Hierbei ist dann eine Unterscheidung erforderlich, ob dieser Marker bereits im Original existiert hat, oder neu erzeugt werden muss (siehe Abbildung 7-5). Der einfachere Fall ist die Erzeugung eines neuen Comment-Markers. Ein Stream aus dem Marker (0xFFFE) gefolgt von den Metadaten kann direkt in die Datei eingebettet werden. Existiert bereits ein Comment-Marker, muss dieser überarbeitet werden und die Metainformationen in den Marker am Ende der bisherigen Daten des Comments eingefügt werden.

7.4 Erzeugung für Audioformate

Die Einbettung der Metadaten in Audioformate ist ebenfalls möglich. Auch hier werden wieder geeignete Vertreter aus dem Medienserverumfeld und der LZA betrachtet (siehe Kapitel 4.4).

7.4.1 WAVE

Bei Audiodateien im WAVE/RIFF Format ist die Einbettung zusätzlicher Metadaten über einen neuen sub chunk vom Typ INFO möglich. In Abbildung 7-6 wird schematisch dargestellt, wie die Einbettung abläuft. Die Metadaten werden entsprechend kodiert in einen eigenen Container eingebettet und am Ende der Datei in einem zusätzlichen Datenpaket angehängt.

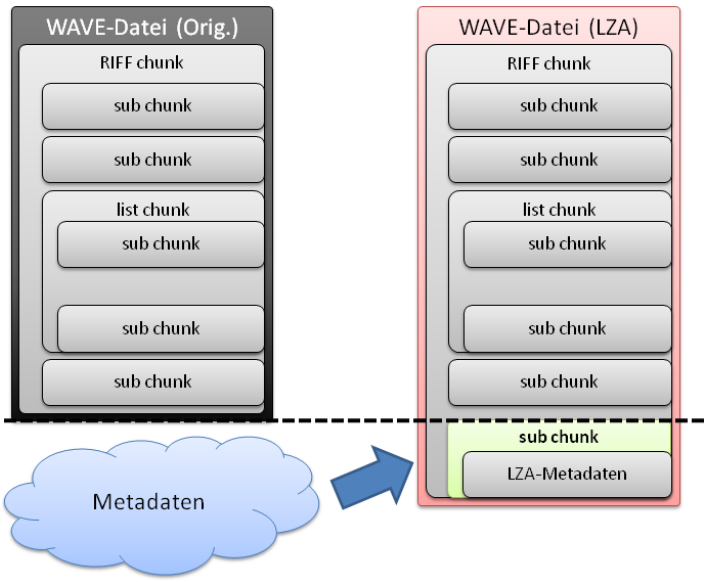

Abbildung 7-6: Schematische Darstellung LZA-Erzeugung (WAVE)

Bei WAVE/RIFF handelt es sich um ein weit verbreitetes Dateiformat, für das bereits andere Softwaretools Metadaten in der Datei einbetten (z.B. Winamp, iTunes, WindowsMediaPlayer). Eine Standardisierung ist noch nicht erfolgt. Somit muss bei der Einbettung unterschieden werden, ob bereits ein INFO-Datenpaket existiert oder nicht. Falls ein INFO-Paket gefunden wird, werden die Metadaten für die LZA ergänzt und zusammen mit bereits vorhandenen Daten abgespeichert. Ist kein INFO-Paket vorhanden, wird am Ende der Datei ein entsprechender chunk mit den Metadaten angelegt.

7.4.2 MP3

Durch die weite Verbreitung des MP3-Formats ist die Unterstützung der LZA für dieses Dateiformat sinnvoll. Ähnlich der JPEG-Dateien besitzt auch das MP3-Format eine Stelle, an der Metainformationen hinterlegt werden können. Diese Daten im ID3v2-Format werden in einem eigenen Frame vom Typ Comment eingebettet. Dabei ist wieder darauf zu achten, dass bereits vorhandene Daten nicht verloren gehen, da der Comment-Frame auch von anderen Anwendungen verwendet wird.

Der Comment-Frame ist notwendig, da zwar bereits Metadaten in den Tags gespeichert werden können, die Attribute aber fest definiert sind. Spezielle Anforderungen benötigen besondere Felder, für die es keine direkte Unterstützung innerhalb der Tags gibt. Daraus ergibt sich die Forderung nach zusätzlicher Flexibilität, den der Comment-Frame mit Freitexteingabe bietet.

Für die Erzeugung muss als erstes die korrekte Position für den Comment-Frame gefunden werden. Gleichzeitig muss erkannt werden, ob bereits von anderen Anwendungen Kommentare in diesem Frame vorhanden sind. Falls der Frame nicht existiert, wird ein Neuer am Ende des ID3v2-Tags erzeugt und die entsprechenden Daten eingehängt. Wenn der Frame bereits mit Daten vorhanden ist, müssen die vorhandenen Daten mit den zu ergänzenden LZA-Daten kombiniert in einem entsprechend angepassten Frame gespeichert werden (siehe Abbildung 7-7).

Vorteilhaft an dieser Methode ist, dass die kompletten Metainformationen im ID3v2-Frame hinterlegt sind. Die Audioinformationen bleiben unberührt und somit auch bei Änderungen an den Metadaten im Original erhalten.

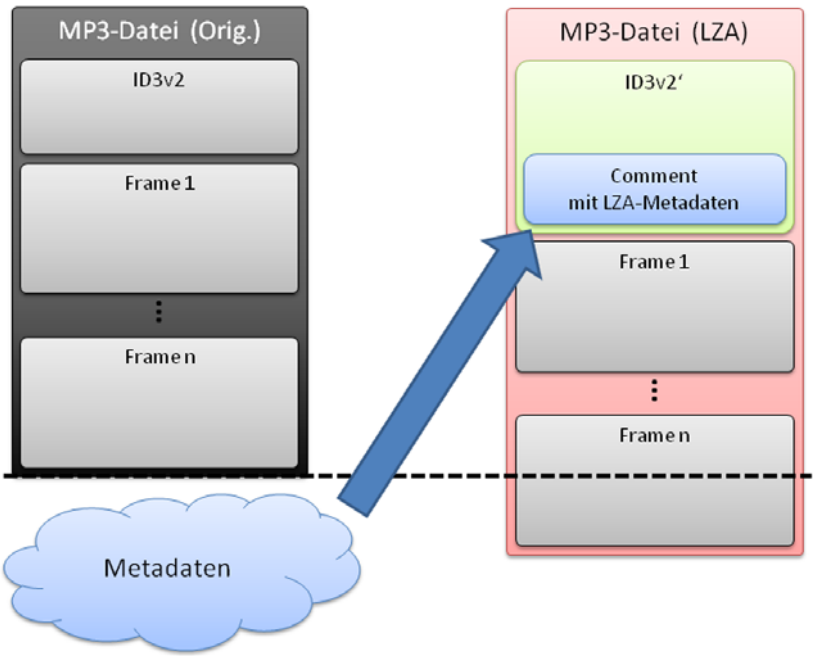

Abbildung 7-7: Schematische Darstellung LZA-Erzeugung (MP3)

7.5 Erzeugung für Videoformate

Die Einbettung der Metadaten in Videoformate ist ebenfalls möglich. Auch hier werden wieder geeignete Vertreter aus dem Medienserverumfeld und der LZA betrachtet (siehe Kapitel 4.5).

7.5.1 MPEG4

Innerhalb des MPEG4-Dateiformates sind generell keine Metadaten vorgesehen. Da das Format über eine hierarchische Struktur aufgebaut ist, lassen sich an dieser Stelle weitere Objekte (Atome) einfügen, die zur Aufnahme von Metadaten verwendet werden können. Dieses Verfahren nutzen bereits andere Hersteller, beispielsweise Adobe, um ein paar wenige Informationen über das vorliegende Videoobjekt anzeigen zu können. Hierfür wird in den Baum der MPEG4-Datei ein neuer Zweig eingebunden, der hierarchisch für Metadaten verwendet wird. Analog zu diesem Verfahren lassen sich auch weitere Metadaten einfügen (siehe Abbildung 7-8).

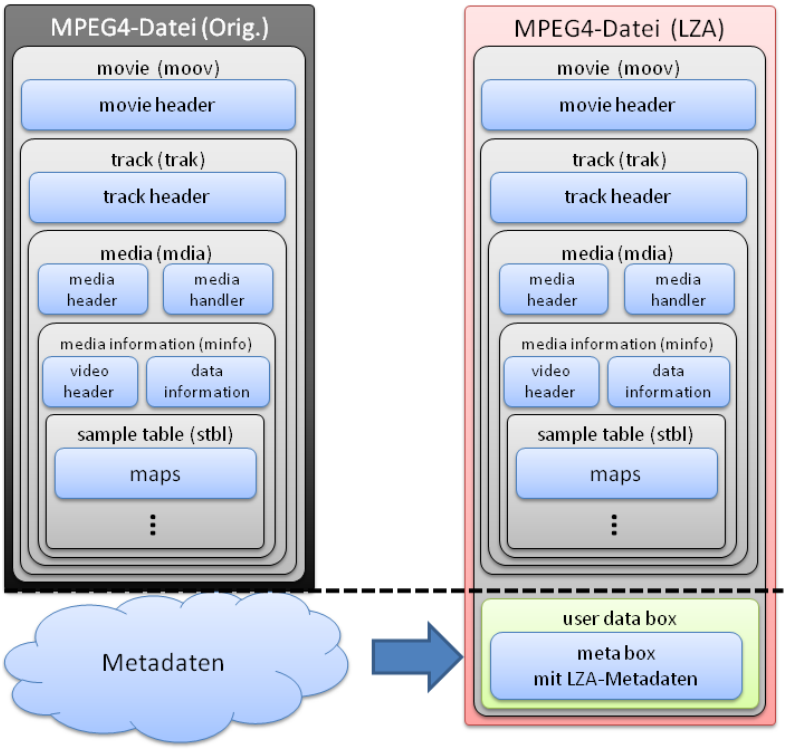

Abbildung 7-8: Schematische Darstellung LZA-Erzeugung (MPEG4)

Unterhalb des Knotens moov wird eine user data box erzeugt und darunter dann die eigentlichen Metadaten eingehängt. An dieser Stelle können die Metadaten nicht in XML-Syntax verwendet werden, hier müssen Binärdaten erzeugt werden (Atome). Diese Atome verwenden folgende Syntax:

- 4 Bytes für die Längenangabe des Atoms,
- 4 Byte für den Namen des Atoms,
- gefolgt vom eigentlichen Inhalt.

Daraus lässt sich der Metadaten-Stream erzeugen und eine beliebige Anzahl an Attributen hintereinander hängen. Falls bereits die Originaldatei einen user data box-Container enthält, muss dieser über die geeignete Wahl der hinzuzufügenden Attribute erhalten werden und um die zusätzlichen Metadatenattribute ergänzt werden.

7.5.2 FLV

FLV bietet formatseitig laut Spezifikation bereits einige wenige Metadatenfelder zu Beginn der Datei. Durch die festen Felder sind Erweiterungen nur im Feld xtradata möglich. Für die Einbettung erfolgen

eine Suche nach den Metadaten (Marker 0x12) innerhalb der FLV-Datei und die Zerlegung in die möglichen Felder. Sind bereits Daten im xtradata-Feld vorhanden, müssen diese erhalten werden und um die zusätzlichen Metainformationen des Medienservers ergänzt werden. Liegen alle Informationen vor, kann daraus der neue Metadatenbereich erzeugt werden und zu Beginn der Datei gespeichert werden. Anschließend ist die Korrektur der Längenangabe für dieses Objekt innerhalb der FLV-Datei zu korrigieren, um eine spezifikationskonforme FLV-Datei zu erhalten (siehe Abbildung 7-9).

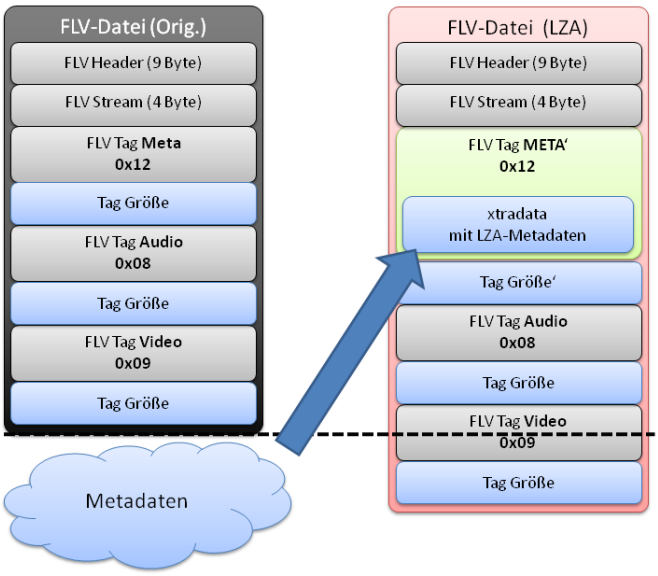

Abbildung 7-9: Schematische Darstellung LZA-Erzeugung (FLV)

Da das xtradata-Feld bereits von anderen Anwendungen mit Informationen belegt sein kann, ist es an dieser Stelle, analog zu MP3 oder JPEG, besonders wichtig, die ursprünglichen Informationen zu erhalten und nur um zusätzliche Daten zu ergänzen.

7.6 Erzeugung des Originals aus LZA-Objekt

Nachdem aus einer Reihe von Dateiformaten die LZA-Varianten erzeugt wurden, muss auch der umgekehrte Prozess ermöglicht werden, die Erzeugung des Originals mit gleichzeitiger Extraktion der gespeicherten Metainformationen. Das gefundene Verfahren erlaubt es, dass aus allen erzeugten Dateien auch wieder die Umkehrung hergeleitet werden kann. Auffällig ist, dass bei Formaten, die sich besonders für die LZA eigenen, auch die Rückerzeugung einfacher möglich ist.

7.6.1 Extraktion für Textformate

Die Erzeugung des Originals aus Textformaten ist abhängig vom vorliegenden Dateiformat.

XML

Bei XML-Dateien müssen zwei Schritte nacheinander ablaufen, um das Original wiedererstellen zu können. Begonnen wird mit der Suche nach dem Wurzelknoten des XML-Baums und die Kontrolle, ob dieser Knoten genau zwei Kinder besitzt. Trifft das zu, muss der zweite Knoten strukturell analysiert werden, ob es sich hierbei um einen Metadatenknoten handelt. Ist dies der Fall, wird aus dem ersten Knoten eine neue XML-Datei erzeugt, die Daten des zweiten Knotens beinhalten alle gespeicherten Metadaten (siehe Abbildung 7-10).

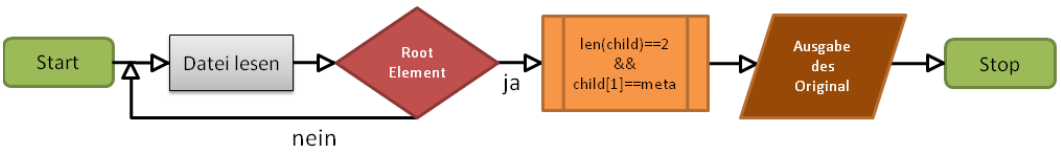

Abbildung 7-10: Programmablauf der Rückerzeugung (XML)

Ergibt die Analyse des zweiten Knotens keinen Metadatenknoten oder besitzt die Wurzel nicht genau zwei Kinder, sind in der vorliegenden Datei keine Metainformationen eingebettet worden und die Ausgangsdatei ist bereits die Originaldatei.

PDF

Der Schritt zurück zum Original ist bei PDF-Dateien sehr elegant realisierbar. Während der Erzeugung werden keine Änderungen an der Originaldatei vollzogen. Es werden lediglich am Ende der Datei weitere Daten angehängt. Damit ist es vollkommen ausreichend, wenn man die Original-Dateigröße kennt und kann daraus, ohne weitere Informationen über die LZA-Datei zu besitzen, das Original rekonstruieren. Dazu muss nur von Beginn der Datei an der Bitstream rekonstruiert werden, bis die Länge der alten Datei erreicht ist. Als Ergebnis erhält man dann eine exakte Kopie der Ursprungsdatei.

Falls die Länge der Ausgangsdatei nicht bekannt ist, kann man über die interne Datenstruktur gehen und die Position der letzten Cross-Reference Table suchen. Direkt zuvor war das alte Dateiende mit dem originalen Trailer zu finden. Wenn diese Position ermittelt ist, kann man das Original über dasselbe zuvor beschriebene Verfahren erzeugen (siehe Abbildung 7-11).

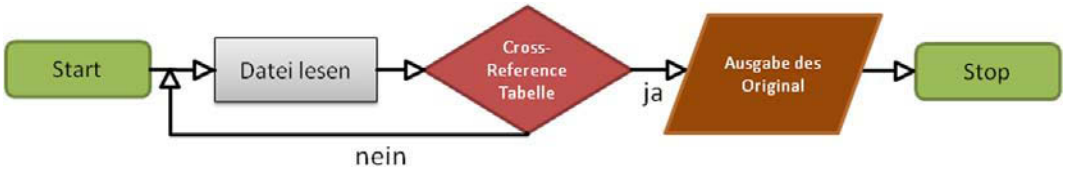

Abbildung 7-11: Programmablauf der Rückerzeugung (PDF)

Die eingebetteten Metadaten können auch einfach aus der LZA-Version extrahiert werden, indem man das letzte Objekt innerhalb der Datei betrachtet. Handelt es sich hierbei um ein Metadatenobjekt, können die gespeicherten Informationen für die spätere Verarbeitung ausgelesen werden und einer weiteren Verarbeitung zugeführt werden.

7.6.2 Extraktion aus Grafikformaten

Für Grafikformate ist die Rückgewinnung des bitgenauen Originals getrennt zu betrachten und abhängig vom Dateityp.

TIFF

Bei TIFF-Dateien kann man das Original gewinnen, indem man zwei Schritte nacheinander ausführt. Als erstes liest man den Bitstream bis zur Länge der alten Datei aus. Dann sind zusätzlich in der internen Datenstruktur der Datei noch der letzte IFD-Eintrag zu suchen und der Zeiger zu korrigieren, dass dieser IFD-Eintrag wieder als letzter erkannt wird und mit 0000 terminiert.

Falls man die originale Dateigröße nicht kennt, kann diese aus der LZA-Version ermittelt werden. Dazu muss die interne Datenstruktur analysiert werden und das vorletzte IFD gefunden werden. Dieses IFD beinhaltet die bitgenaue Sprungadresse, die das ehemalige Ende der Datei darstellte und damit die Ausgangsdateigröße. Anschließend können wieder die zuvor beschriebenen Schritte für die TIFF-Datei ausgeführt werden (siehe Abbildung 7-12).

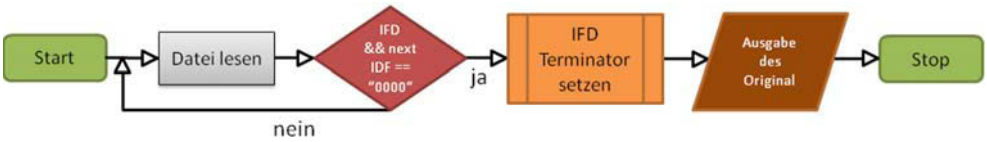

Abbildung 7-12: Programmablauf der Rückerzeugung (TIFF)

Die Metadaten befinden sich im letzten Directory Entry der Datei. Um die genaue Position zu ermitteln, muss in der internen Struktur der TIFF-Datei die Position des letzten Directory Entries gefunden werden und dieser Entry ausgelesen werden. Die interne Struktur bietet über die verkettete Liste aller IFDs eine einfache und schnelle Möglichkeit, die Metadaten zu ermitteln.

JPEG

Bereits bei der Erzeugung des LZA-Formats ist JPEG aufwändiger. Der umgekehrte Prozess ist ebenso nur in mehreren Schritten möglich. Zuallererst muss in der Marker-Struktur der Datei der Comment-Marker gefunden werden. Anschließend müssen die auf den Marker folgenden Daten ausgelesen werden, bis zum Beginn des nächsten Markers innerhalb der Datei.

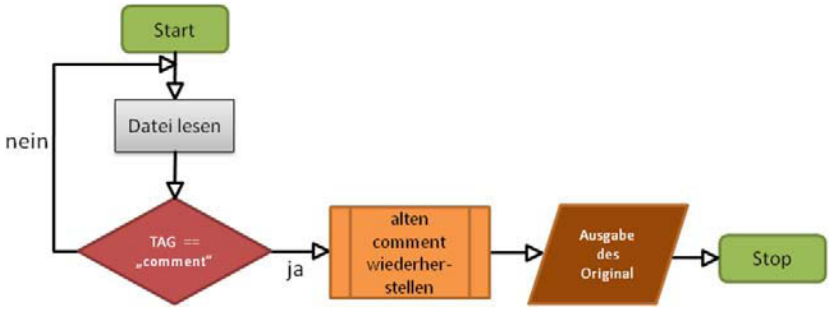

Abbildung 7-13: Programmablauf der Rückerzeugung (JPEG)

Damit hat man nun die Metadaten extrahiert und kann sie weiterverarbeiten. Um allerdings die Originaldatei erzeugen zu können, müssen die Metadaten analysiert werden. Beinhalten sie Informationen über einen original vorhandenen Comment, müssen die Daten des Original-Comments wieder in die Datei eingefügt werden und anschließend ab dem folgenden Marker die Datei bis an das Dateiende weiter geschrieben werden. Damit kann eine bitgenaue Kopie des Originals erreicht werden. Falls die Ausgangsdatei keinen Comment-Marker besitzt, muss lediglich der komplette Comment-Marker aus der LZA Variante entfernt werden, um das Original rekonstruiert zu haben (siehe Abbildung 7-13).

7.6.3 Extraktion aus Audioformaten

Bei Audioformaten ist die Rückgewinnung des Originals teilweise recht aufwändig. Das kann an unterschiedlichen Punkten liegen, wie beispielsweise der Dateigröße oder der internen Dateistruktur.

WAVE/RIFF

Beim WAVE/RIFF-Format ist die Rückgewinnung durch einen Algorithmus realisierbar, der das Ende der Datei ab einer bestimmten Position abschneidet. Die Position ist ermittelbar, indem man den letzten verfügbaren sub chunk wählt und diesen nach der Kontrolle auf den korrekten Typ für Metadaten entfernt (siehe Abbildung 7-14).

Als einzige Schwierigkeit bleibt die klare Einordnung, ob es sich bei diesem letzten Chunk um einen aus dem LZA Umfeld handelt. Andernfalls ist die Verarbeitung der WAVE-Dateien unkompliziert.

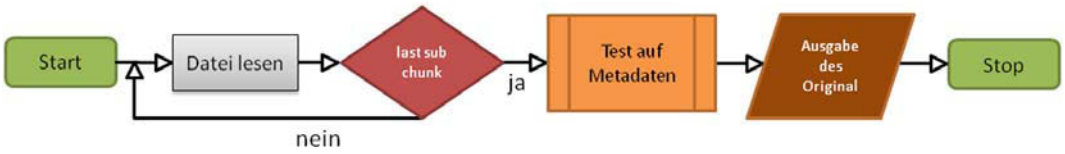

Abbildung 7-14: Programmablauf der Rückerzeugung (WAVE)

MP3

Bei MP3-Dateien werden die LZA-Metadaten innerhalb des ID3v2-Tags abgelegt. Zur Rückgewinnung wird die Datei nach diesem Tag durchsucht und der Wert des Comment-Feldes kontrolliert. Eine Kontrolle ist erforderlich, weil bereits vor der Erzeugung der LZA-Variante Metadaten innerhalb des Comment-Feldes vorhanden sein können. Falls diese vorhanden waren, muss eine Rekonstruktion des alten Inhalts erfolgen und anschließend die Datei wieder zurückgeliefert werden (siehe Abbildung 7-15).

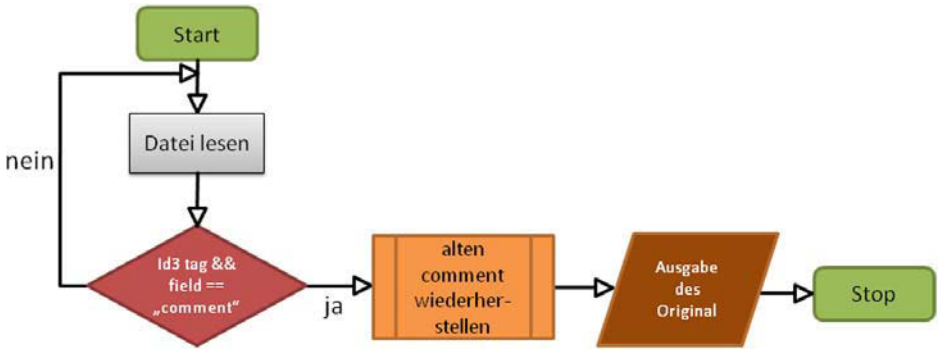

Abbildung 7-15: Programmablauf der Rückerzeugung (MP3)

Die Metadaten, die aus dem Comment-Tag gewonnen werden, müssen anschließend wieder über den Stream rekonstruiert werden. Eine genaue Beschreibung findet sich im nächsten Abschnitt.

7.6.4 Extraktion aus Videoformaten

Bei gängigen Videodateiformaten der LZA ist die Betrachtung der Rückgewinnung ebenfalls formatabhängig vorzunehmen.

MPEG4

Bei MPEG4-Dateien ist es für eine Rückgewinnung der Originaldatei aus der LZA-Variante erforderlich, den internen Objektbaum zu kontrollieren. Befindet sich dort ein user data box-Container am Ende der Datei, muss dieser Container aus der internen Struktur der Datei entfernt werden, falls bei der Kontrolle des Containers außer den LZA-Daten keine weiteren Daten gefunden wurden. Sind zusätzliche Informationen vorhanden, müssen diese rekonstruiert im user data box-Container erhalten werden. Eine so gewonnene Datei entspricht dem Original, aus dem die LZA-Variante erzeugt wurde. Die extrahierten Metadaten liegen nicht in XML-Struktur vor, sondern als Stream mit Größenbeschreibungen. Hier muss zuerst eine Konvertierung erfolgen, dass die Metadaten weiterverarbeitet werden können (siehe Abbildung 7-16).

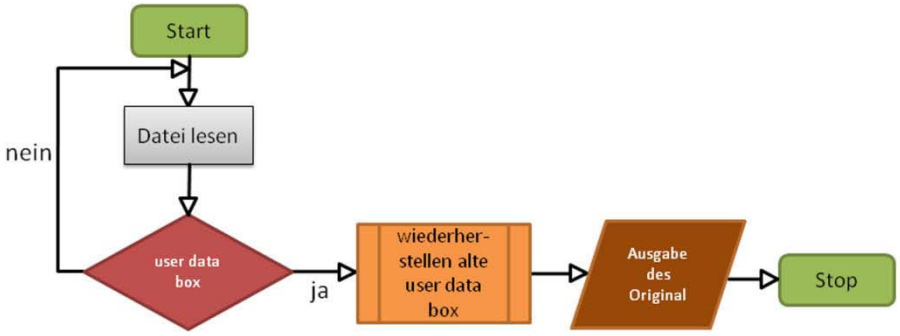

Abbildung 7-16: Programmablauf der Rückerzeugung (MPEG4)

FLV

Das Original einer FLV-Datei kann aus der LZA-Datei erzeugt werden, indem innerhalb der Datei nach dem Tag 0x22 für Metadaten gesucht wird. Anschließend erfolgt eine Zerlegung in die Attribute und die Kontrolle des xtradata-Feldes (siehe Abbildung 7-17). Liefert diese Kontrolle weitere Metadaten neben den LZA-Daten, muss der ursprüngliche Feldinhalt wiederhergestellt werden. Sind die LZA-Daten die einzigen Metainformationen, kann das xtradata-Attribut entfallen und die Datei ausgeliefert werden.

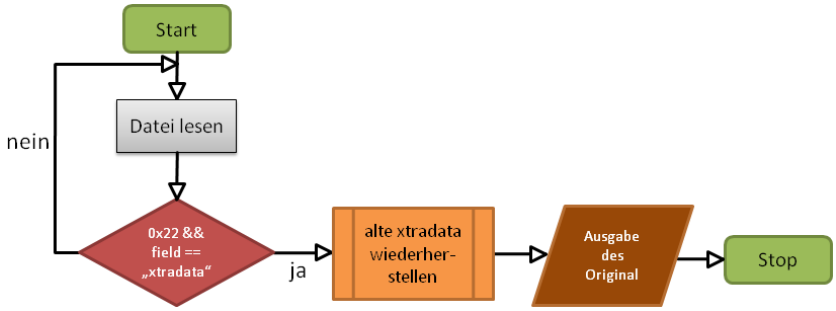

Abbildung 7-17: Programmablauf der Rückerzeugung (FLV)

7.7 Erzeugung der notwendigen Metadaten

Das Archivierungssystem verwaltet zu jedem Objekt verschiedene Attribute. Diese Attribute bilden den kompletten Informationsbestand zu einem Objekt und kommen aus den unterschiedlichsten Metadatenkategorien. Zur Archivierung werden in der Regel nicht alle Attribute herangezogen, die in einem Medienserver vorhanden sind. Denn beispielsweise bei Textdokumenten im PDF-Format ist auch der Volltext ein Metadatenattribut.

Bedingt durch das heterogene Umfeld mit unterschiedlichen Datendefinitionen (Schemata) und Metadatenattributen ist keine automatisierte Erzeugung der Metadaten für die Archivierung möglich. Über Metadaten-Masken und eine nachgeschaltete Formatierung (Mapping) ist die Auswahl der Attribute in der korrekten Formatierung für die LZA gegeben. Die zugrundeliegenden Formate erzeugen die Mappings direkt aus der Definition der Metadatenattribute. Durch den per Definition festgelegten Umfang an Metadaten, ist eine Vereinheitlichung gegeben, die Kontrolle dieser Werte steigert die Datenqualität.

7.7.1 Metadaten-Maske

Metadaten-Masken definieren den Umfang und die Reihenfolge notwendiger Metadatenattribute. Sie bilden Untermengen der verfügbaren Metadaten, die auf einen ganz bestimmten Einsatzzweck hin abgestimmt sind. Masken kommen an unterschiedlichen Stellen innerhalb eines Medienservers zum Einsatz. Ihre Eigenschaften ermöglichen die Nutzung sowohl bei der Anzeige, als auch bei Import und Export. Grundlegend ist immer, dass Masken die eigene Darstellung über ihre Typdefinition kennen und somit unabhängig vom Kontext eingesetzt werden können.

Die Definition der komplexen Funktionalität der Masken ist unbedingt über geeignete Editoren zu realisieren, nur dann ist die Kontrolle gegenüber Fehlern gegeben. Abbildung 7-18 zeigt das Zusammenspiel der unterschiedlichen Komponenten, die beim Erzeugen der Darstellung beteiligt sind. Besonders wird an dieser Stelle die Konfiguration der Metadaten für die LZA über Masken hervorgehoben. Jedes Objekt besitzt Metadaten, die in Masken ähnlich einem Filter zusammengestellt werden. Vor allem für die Darstellung der Metadaten in unterschiedlichen Anzeigeszenarien spielt die Steuerung der Reihenfolge durch Masken und die daraus resultierende Priorisierung der Metadaten eine Rolle. Für die LZA ist die Reihenfolge der Felder unerheblich.

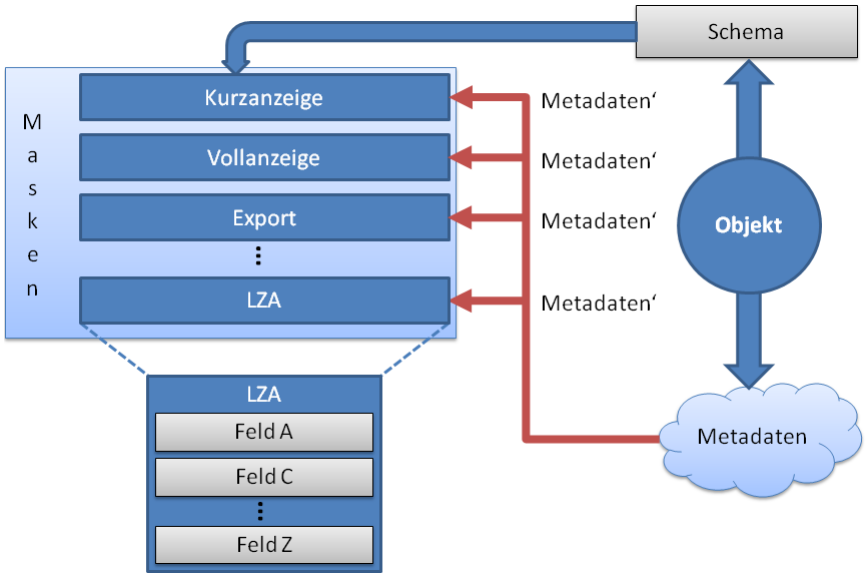

Abbildung 7-18: Datenfluss bei Metadaten-Masken

7.7.2 Metadaten-Mapping

Masken übernehmen aufgrund ihrer Definition die Darstellung von Objekten innerhalb des Medienservers. Verlangt ein Format für die Darstellung eine bestimmte Formatierung, können Abbildungen (Mappings) die per Maske definierten Metadatenattribute weiterverarbeiten und in die gewünschte Form bringen. Über diesen Mechanismus des Mappings kann das Attribut-Werte-Tupel in nahezu jede beliebige Form gebracht werden. Vor allem für den Datenexport ist damit die notwendige Flexibilität vorhanden, dass ein reibungsloser Datenaustausch zwischen verschiedenen Anwendungen gewährleistet werden kann. Über diesen Mechanismus der Abbildung ist die Erzeugung aller möglichen Standards gegeben, von XML oder BibTeX für den Datenexport bis hin zu speziellen Formaten für die Archivierung.

7.8 Anbindung an eine Archivierungshardware

Von einer Archivierungshardware wird erwartet, dass die übergebenen Daten dauerhaft sicher gespeichert werden und dabei zugreifbar bleiben. Der Anwender sieht nur eine Oberfläche oder Schnittstelle, die zum Datenaustausch dient. Auf dem Markt existieren verschiedene Systeme, die diesem Anspruch gerecht werden. Das reicht von reinem Speicherplatz auf speziellen Festplatten, bis zu kommerziellen Soft- und Hardwarelösungen.

Abstrakt betrachtet muss ein Archivsystem Daten aufnehmen können und diese in einer Black-Box sicher aufbewahren, bis ein Abrufen der Information erfolgt. Der Zeitraum zwischen diesen beiden Ereignissen ist beliebig. Für den Nutzer ist die Art der Datenspeicherung uninteressant, solange die Daten mit Sicherheit erhalten bleiben. Damit ist eine Schnittstelle für den Nutzer mit den beiden Methoden zum Übergeben der Daten und Auslesen der Daten ausreichend (siehe Abbildung 7-19).

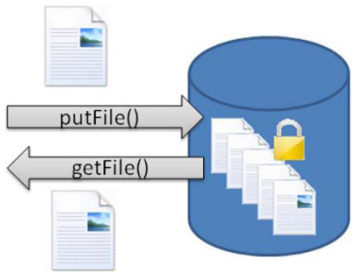

Abbildung 7-19: Vereinfachte Darstellung der Schnittstelle zum Archivierungssystem

Jeder Datenknoten des Medienservers besitzt Statusinformationen über den aktuellen Stand der Archivierung als eigenes Metadatum. Dieser Status kontrolliert das Verhalten des Medienservers, falls Informationen aus dem Archivsystem angefragt werden. Vorgesehen sind folgende Statuswerte:

Status 1:
Die Originaldaten des Objekts liegen im Archivsystem vor. Zugriff auf die Daten ist nur über das Archivsystem möglich.

Status 2:
Die Originaldaten wurden bereits aus dem Archivsystem geholt und liegen an der Originalposition innerhalb des Medienservers vor. Dieser Status speichert die Daten einen konfigurierbaren Zeitraum lang, falls sich Zugriffe auf gewisse Originale häufen. Nach Ablauf des Zeitraums werden die Daten wieder vom Medienserver entfernt.

Status 3:
Die Originaldaten sollen in das Archivsystem übertragen werden. Dabei ist zu unterscheiden, ob es sich um eine Erstaufnahme in das Archiv handelt oder die Daten im Archiv aktualisiert werden sollen.

Innerhalb eines Medienservers ist nicht nur die Nutzung einer einzigen Archivierungslösung denkbar. Für diesen Zweck kann für jedes Objekt separat definiert werden, welchem Archivierungsprozess die Daten zugeführt werden sollen. Aufgrund der unterschiedlichen Sammlungen auf einem Server ist eine angepasste Archivlösung sinnvoll, die auf die Objekte abgestimmt ist. Das kann die besonders sichere Speicherung der Daten innerhalb einer Archivierungshardware sein oder auch nur das Umkopieren auf einen anderen Datenträger. Die einfache Schnittstelle zwischen Medienserver und anzuwendender Archivierung bietet die notwendige Flexibilität.

7.8.1 Umsetzung der Funktionalität innerhalb der Medienserversoftware

Der Medienserver bietet eine konfigurierbare Schnittstelle, die mit unterschiedlichen Archivierungskomponenten umgehen kann. Ein Archiv-Manager übernimmt die Steuerung parallel zum laufenden Medienserverbetrieb. In einem eigenen Prozess kontrolliert der Archiv-Manager in regelmäßigen Zeitabständen, ob sich bei den Daten des Medienservers Änderungen ergeben haben, die eine Aktualisierung der Archivdaten erforderlich machen. Erkennungsmerkmal für diese Änderungen sind die zuvor beschriebenen Stati. Neben der Kontrolle der Stati erfolgt im Archiv-Manager auch die Zuordnung der Archivierungsimplementierung zum Objekt innerhalb des Medienservers.

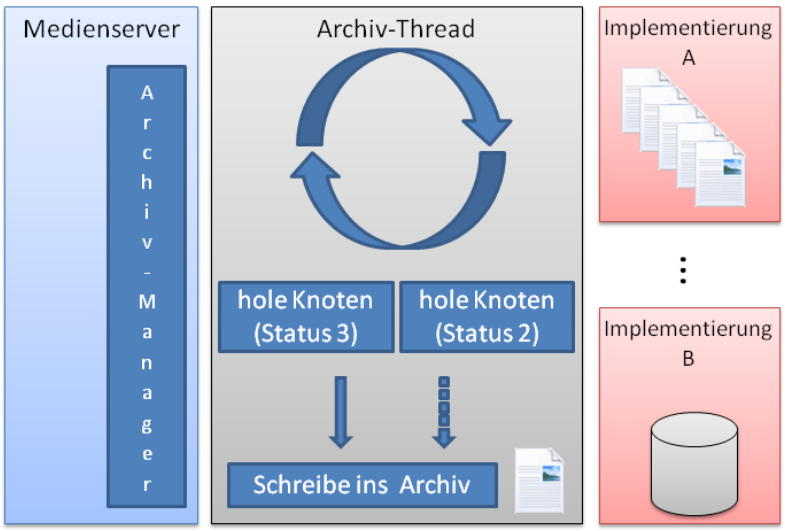

Abbildung 7-20: Schematische Darstellung des Archiv-Managers

Abbildung 7-20 zeigt die schematische Darstellung des Archiv-Managers, der als Teil des Medienservers direkt mit diesem verbunden ist oder als eigenständige Komponente ausgeführt sein kann, falls der Medienserver die Integration nicht erlaubt. Die eigenständige Komponente des Archiv-Thread besitzt die Aufgabe, die notwendigen Daten zu ermitteln, die durch die angebundenen Implementierungen verarbeitet werden sollen. Dabei ist es von der jeweiligen Implementierung abhängig, welche Dateitypen unterstützt werden und welches Archivierungssystem eingebunden ist. Der Archiv-Thread kontrolliert an dieser Stelle die Knoten auf den Status zwei und drei. Knoten mit Status drei werden über die definierte Implementierung direkt an das Archiv übergeben, anschließend wird für den Knoten der Status auf eins gesetzt. Knoten mit Status zwei stehen über den Archivmanager direkt dem Medienserver zur Verfügung. Implementierungsabhängig ist nach einem definierten Zeitraum das Löschen der Originaldaten aus dem Speicher des Medienservers erforderlich oder auch ein Zurückspielen der Daten in das Archiv. Das genaue Verhalten ist implementierungsabhängig. Die Implementierungen bilden damit die Verbindung zwischen den Daten des Medienservers und des Archivierungssystems über einen Connector.

Neben der Prozessverwaltung der Archivanbindung ist innerhalb des Medienservers eine Benutzerschnittstelle erforderlich, die die Daten ausliefert. Zwischen der Anforderung der Daten und der Auslieferung über den Archiv-Thread ergibt sich eine Zeitspanne, die die Benutzeroberfläche des Medienservers überbrücken muss. Bei der Verwendung einer Web-Schnittstelle bietet sich AJAX-Technologie an, die geeignete Funktionen für asynchrone Prozesse mitbringt. Aus den Gegebenheiten ergibt sich folgendes Szenario:

Ein Nutzer fordert die Archivdaten zu einem Objekt über die Oberfläche des Medienservers an. Dazu präsentiert der Mediensever einen Link an entsprechender Stelle. Die Anforderung bearbeitet der

Archiv-Manager mit einer Kontrolle des Status des Objekts. Liegen die Daten bereits im Zwischenspeicher des Medienservers vor, erfolgt eine direkte Auslieferung an den Nutzer. Ist das nicht der Fall, stößt der Archiv-Manager den Übertragungsprozess aus dem Archiv an. Die Daten stehen dem Nutzer nach der Abarbeitung der Anforderung für einen konfigurierbaren Zeitraum zur Verfügung. Ist diese Zeitspanne abgelaufen, kontrolliert der Archiv-Thread das Objekt und leitet das Löschen vom Zwischenspeicher ein.

7.8.2 Implementierung und Test der Schnittstelle

Zu Testzwecken erfolgt eine Implementierung der Archivierungsschnittstelle innerhalb von mediatum für den Tivoli-Storage-Manager (TSM) der Firma IBM [**40**]. Dieses System wird vom LRZ zur Verfügung gestellt und an der TUM an verschiedenen Stellen eingesetzt. Anhand dieser Umsetzung erfolgt die Kontrolle der bisher gefundenen Lösung hinsichtlich Machbarkeit und Leistung. Die Datensicherheit garantiert das System bzw. das LRZ, somit erfolgt keine gesonderte Betrachtung dieses Faktors.

Die Anbindung der API des TSM erfolgt über die Realisierung eines Python-Moduls für den Medienserver. Dieses Modul kapselt in einer Black-Box die spezifische Funktionalität des TSM und stellt dem Archiv-Thread die geforderten Funktionen zur Verfügung. Die Kommunikation mit TSM erfolgt innerhalb einer Session, über die die Authentifizierung der Systeme abgearbeitet wird. Die Session bietet sowohl lesenden, als auch schreibenden Zugriff auf die Daten des Archivs. Für den Archiv-Thread werden diese Operationen über eine Getter- und Setter-Methode realisiert. Als Besonderheit des TSM steht ein Lock-Attribut auf den Archivdateien zur Verfügung, das einen zusätzlichen Schutz gegen unbeabsichtigtes Löschen bietet (siehe Abbildung 7-21).

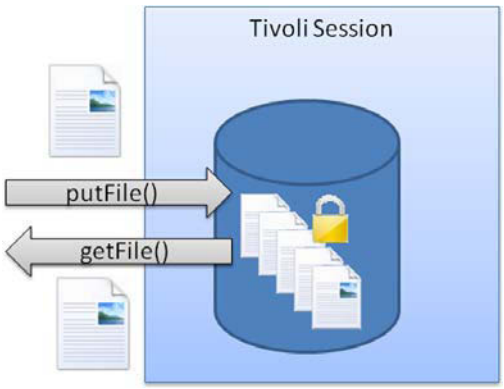

Abbildung 7-21: Schematische Darstellung eines Zugriffs über pytsm

Zusätzlich zur Implementierung für TSM erfolgt die Realisierung einer einfachen Kopierfunktion. Die Archivdaten werden dabei nur an eine definierbare Stelle kopiert, ohne weitere Operationen darauf auszuführen. Die Lieferung der Archivdaten erfolgt über den umgekehrten Prozess. Damit stehen zwei unterschiedliche Verfahren zur Verfügung, die die einfachste Möglichkeit (Kopieren) und eine komplexe, kommerzielle Komponente (TSM) abdecken. Eine Gegenüberstellung beider Archivierungsimplementierungen erfolgt unter folgenden Kriterien:

Notwendiger Funktionsumfang
Die Kapselung der Implementierung ermöglicht die Funktionsreduktion auf ein Minimum. Die realisierten Verfahren zeigen, dass für die Anbindung an den Medienserver lediglich Methoden für das Lesen der Archivdaten und das Schreiben der Daten erforderlich sind. Weitere Details sind abhängig von der gewählten Archivierungsmethode und können im Inneren der Black-Box realisiert werden.

Sicherheit
Für die Sicherheit der Verfahren sind alleine die Implementierungen verantwortlich. Der Medienserver besitzt keine Bewertungskriterien, die eine Bewertung der Sicherheit ermöglichen. An dieser Stelle ist der Betreiber des Systems in der Verantwortung, ein geeignetes Verfahren auszuwählen.

Performanz
Die Performanz der Archivierung ist abhängig von den eingesetzten Verfahren und Software bzw. Hardware. Das Kopieren der Dateien stellt das schnellste Verfahren dar. Andere Methoden können deutlich mehr Zeit in Anspruch nehmen. Auf der Seite des Medienservers sind verfahrensunabhängige Vorkehrungen notwendig, die die Zeitspanne bei der Auslieferung überbrücken.

Die Testszenarien mit den beschriebenen Implementierungen haben ergeben, dass das gezeigte Verfahren in unterschiedlichen Ausprägungen für die LZA einsetzbar ist. Sowohl für das einfachste denkbare Verfahren des Kopierens, als auch die Anbindung einer komplexen Archivierungshardware sind realisierbar. Die Steuerung erfolgt direkt aus dem Medienserver heraus und verlangt Komponenten der Verwaltung und Erzeugung der notwendigen Daten.

7.9 Import von LZA-Objekten in das System

Der Import von Daten in den Medienserver ist mit einem Zusatzmechanismus versehen, der direkt beim Anlegen des Objekts überprüft, ob innerhalb des Digitalisats bereits LZA-Informationen enthalten sind. Es handelt sich hierbei um eine Erweiterung der sonst üblichen Mechanismen, die technische Metadaten aus Objekten auslesen kann.

Durch die Art der Implementierung erfolgt die Ermittlung der LZA-Daten für jeden Dateityp gesondert. Die charakteristische Position der Daten ist dateitypabhängig, die Kontrolle wird automatisiert vom Medienserver angestoßen und erfolgt performant direkt während des Datenimports. Werden LZA-Informationen gefunden, werden diese automatisch extrahiert und weiterverarbeitet. Die Auswahl eines Metadatenschemas zur Charakterisierung des Objekts erleichtert die Zuordnung der Daten zum jeweiligen Attribut. Falls keine LZA-Metadaten gefunden werden, sind keine weiteren Maßnahmen beim Anlegen eines Objekts einzuleiten (siehe Abbildung 7-22).

Beim Erzeugen des Objektes werden dabei folgende Operationen ausgeführt:

- Test auf LZA-Metadaten: Anhand eines Tests auf LZA-Metadaten wird entschieden, ob das Anlegen eines reinen Medienserver Objekts ausreichend ist oder weitere Schritte erforderlich sind. Wenn keine LZA-Daten gefunden werden, ist der Import mit der Erzeugung des Objekts abgeschlossen. Sind LZA-Metadaten enthalten, erfolgt eine Extraktion der Daten zusammen mit den dateiformatspezifischen Metadaten als Attribut-Werte-Tupel.
- Auslesen des passenden Mappings: Falls während des Imports LZA-Metadaten gefunden wurden, werden diese als Attributtupel gespeichert. Über das entsprechende Mapping, ausgehend vom gewählten Metadatenschema, werden diese Tupel den Attributen des Schemas zugeordnet und zusammen mit dem erzeugten Objekt gespeichert.

Dieses automatisierte Verfahren bietet sich für einen Import der LZA-Metadaten an, da an dieser Stelle der größtmögliche Komfort für den Nutzer gegeben ist. Manuelle Verfahren sind ebenfalls denkbar, die beispielsweise nur die gefundenen Daten auflisten und eine manuelle Zuordnung zu Metadatenattributen erfordern. Des Weiteren sind auch Mischformen denkbar. Wie der genaue Ablauf des Imports erfolgen soll, bleibt dem jeweiligen Betreiber des Medienservers überlassen. Auffällig ist an dieser Stelle, dass ein höherer Grad an Automatisierung nicht unbedingt eine höhere Datenqualität zur Folge hat. Die Datenqualität ist in erster Linie von der Sorgfalt während der Erzeugung der Daten abhängig.

Der beschriebene Ansatz bildet den Mittelweg zwischen Automatisierung und Datenqualität. Auf der einen Seite werden die Metadaten automatisch extrahiert, auf der anderen Seite kann nur eine manuelle Kontrolle auch sicherstellen, dass die automatisch zugeordneten Metadaten auch sinnvollen Inhalt aufweisen. Somit eignet sich der gezeigte Ansatz, um mit der Automatisierung beim Import und der nachgeschalteten Einordnung der Metadaten zu Objekten eine praktikable Lösung für die Nutzung einer LZA-Komponente aus einem Medienserver heraus zu realisieren.

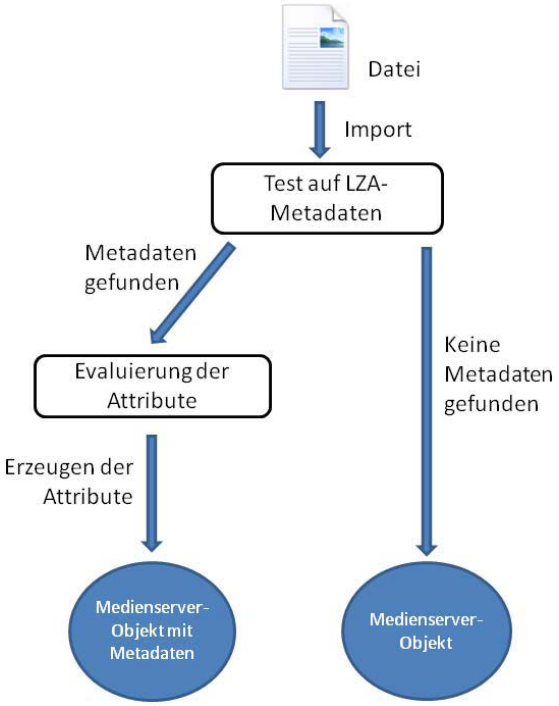

Abbildung 7-22: Vollautomatischer Ablauf eines Dateiimports

7.10 Laufzeitmessungen

Für das gefundene Verfahren sind an unterschiedlichen Stellen Laufzeitmessungen erforderlich, um Aussagen über die Umsetzbarkeit der Lösung treffen zu können. Der vorherige Abschnitt befasst sich mit der Dauer der Datenübermittlung von und zum Archivsystem. Eine weitere kritische Stelle ist die Dauer der Erzeugung der Langzeitobjekte und die Rückgewinnung der Originale.

Für die Tests wurde der Datenbestand des Medienservers der TUM herangezogen, auf dem sich Dateien aus den unterschiedlichen Kategorien befinden. Ein spezielles Skript (siehe Anhang B) durchläuft den Datenbestand und erzeugt für jede verfügbare Datei ein LZA-Objekt. Zur besseren Vergleichbarkeit der Erzeugungsdauer wird dabei immer derselbe Satz an Metainformationen im XML-Format eingebettet, die Daten liegen bereits als Stream vor. Ist die Generierung des LZA-Objekts erfolgreich verlaufen, erzeugt das Skript anschließend aus dem LZA-Objekt wieder das Original und vergleicht das Resultat mit der Ausgangsdatei. Als Kontrolle des Ergebnisses dient der Vergleich der ursprünglich in die Datei eingebetteten LZA-Metadaten mit den aus dem LZA-Objekt extrahierten Daten. Zusätzlich geben die Dateigrößen Aufschluss über ein korrektes Arbeiten des Algorithmus. In die Zeitmessung wurden nur die notwendigen Operationen während der Bearbeitung mit einbezogen. Hintergrundoperationen werden nicht berücksichtigt, um das Ergebnis nicht zu verfälschen. Mit den so gewonnen Zahlen wurden jeweils drei unterschiedliche Vorgänge ausgewertet:

- Die absolute Erzeugungsdauer der LZA-Variante in Sekunden im Verhältnis zur Dateigröße.
- Die Verarbeitungsgeschwindigkeit während der Erzeugung in MB/Sekunde, wieder in Abhängigkeit zur Dateigröße. Zusätzlich kann noch mit einer Trendlinie die Festplattenperformanz im Mittel dargestellt werden.
- Das Verhältnis zwischen Dauer der Erzeugung der LZA-Variante zur Dauer der Rückerzeugung des Originals. Der Quotient lässt Rückschlüsse auf die Erzeugung zu. Dabei bedeuten alle Werte größer als eins, dass die Rückerzeugung des Originals schneller von Statten geht, als die Erzeugung der LZA-Variante.

Die Messungen erfolgten getrennt in den unterschiedlichen Dateitypen auf derselben Hardware auf der der Medienserver betrieben wird.

7.10.1 Messungen für PDF-Dateien

Der Algorithmus hat für den aktuellen Bestand an PDF-Dokumenten (5273 Dokumente mit insgesamt 31,5 GB) jeweils eine LZA-Version erzeugt und anschließend aus der LZA-Version wieder das Original reproduziert. Die genutzten Dateien stammen dabei aus verschiedenen Bereichen und umfassen neben den Elektronischen Prüfungsarbeiten (Dissertationen und Habilitationsschriften) auch eine Reihe an Veröffentlichungen verschiedener Lehrstühle der TUM. Um ein realistisches Szenario zu gewährleisten, wurden die Messungen während des laufenden Betriebs unter der Woche am späten Vormittag durchgeführt.

Erzeugung LZA-Version
Abbildung 7-23 zeigt den Zusammenhang zwischen Dateigröße und Verarbeitungsdauer für die Erzeugung des LZA-Objekts. Dabei ist auffällig, dass sich ein linearer Zusammenhang zwischen Dateigröße und Verarbeitungsdauer herleiten lässt. Es kommt somit in erster Linie auf die Leistungsfähigkeit des Computersystems an, das die Verarbeitung ausführt. Die Dateigröße ist wiederum abhängig vom Anwendungsbereich des PDF-Formats. Sind viele Grafiken eingebunden, steigt auch die Dateigröße der PDF-Datei und damit die Verarbeitungsdauer der LZA-Versionserzeugung.

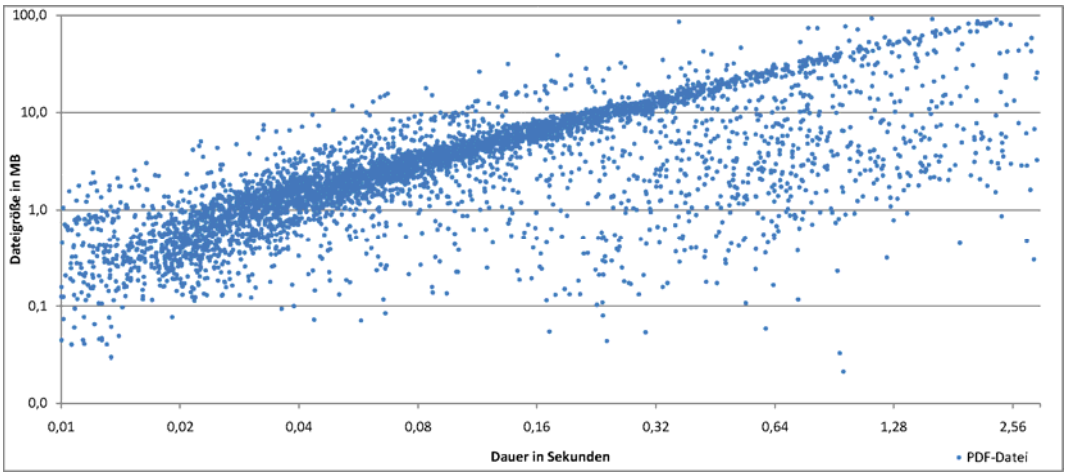

Abbildung 7-23: Erzeugungsdauer LZA-Objekt bei PDF-Dateien

Ähnliches lässt sich in Abbildung 7-24 ablesen. Auch hier ist erkennbar, dass nicht in erster Linie die absolute Dateigröße ausschlaggebend für die Verarbeitungsleistung der Dateien ist, sondern das System, auf dem die Dateien liegen. Bei PDF-Dateien ist generell die Dateigröße eher im unteren Bereich anzusiedeln. Gerade bei Bilddateitypen sind die Dateien in der Regel deutlich größer.

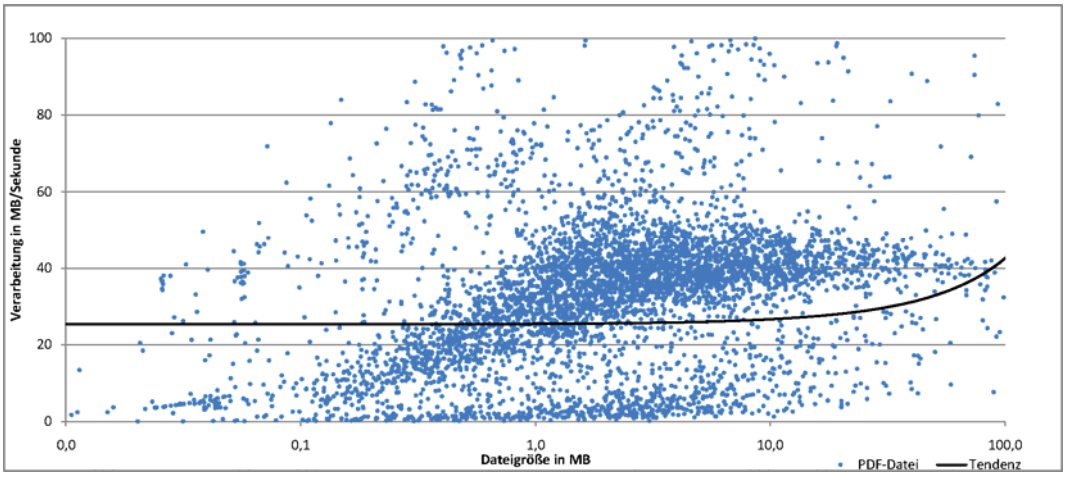

Abbildung 7-24: Verarbeitung in MB/Sekunde bei der Erzeugung (PDF)

Auffallend ist in dem Zusammenhang, dass die beiden unterschiedlichen Festplatten RAID-Verbunde den maßgeblichen Anteil an der Verarbeitungsleistung beitragen. Der Großteil der Dateien über 1 MB Größe liegt dabei auf einem RAID-Verbund, der deutlich über 30 MB/Sekunde verarbeiten kann, im Mittel etwa 37 MB/Sekunde. Eine Reihe an Dateien, die auf einem langsameren Festplattenlaufwerk liegen, erreichen nur unterdurchschnittliche Werte. Das liegt unter anderem auch daran, dass das langsamere RAID-Laufwerk deutlich weniger Speicherkapazität bietet. Es wurde deshalb darauf geachtet, dass die größeren Dateien auch auf dem größeren und zugleich schnelleren Laufwerk abgelegt werden.

Vergleich Erzeugung LZA/Original

Aus den gewonnenen Messdaten lässt sich auch eine Gegenüberstellung der Erzeugung des LZA-Objekts zur Erzeugung des Originals aus den Archivdaten herleiten. Dabei ist gut zu erkennen, dass die Rückerzeugung des Originals im Mittel fünfmal schneller erfolgen kann, als die Erzeugung des LZA-Objekts. Diese Zahl lässt klar erkennen, dass die gefundene Lösung für PDF-Dateien ein geeignetes Verfahren im Bezug auf die Laufzeit und Datensicherheit für die LZA darstellt. Das Ergebnis war zu erwarten, da beim Rückerzeugen der PDF-Dateien nur sehr geringe Änderungen an der Ausgangsdatei erfolgen müssen. Diese Änderungen sind vergleichbar mit dem Kopieren der Daten und dem Abschneiden des Endes.

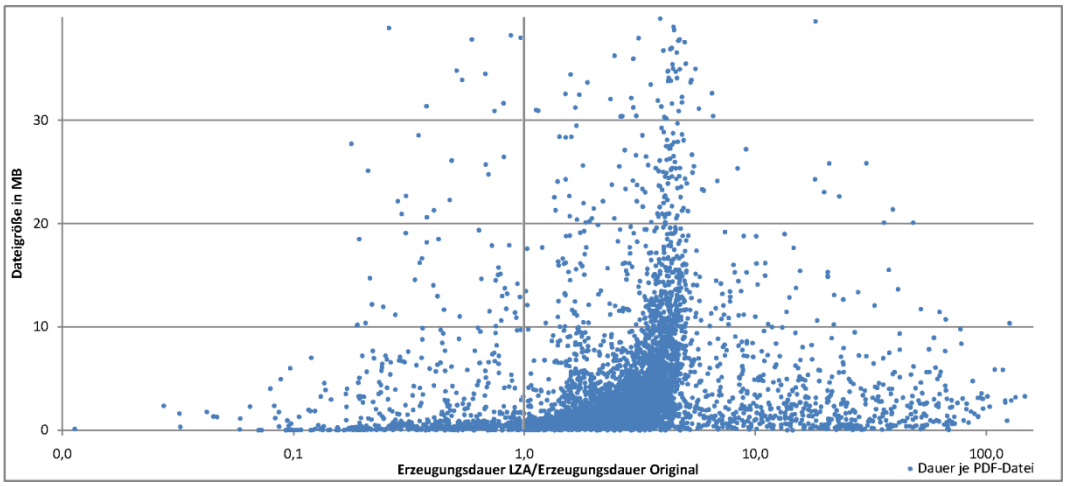

Abbildung 7-25: Vergleich Erzeugung LZA-Objekt/Original (PDF)

Abbildung 7-25 zeigt außerdem, dass sich über den Verlauf der Messungen die Werte und vor allem die Störungen auch widerspiegeln. Störungen resultieren in erster Linie aus den unterschiedlichen Festplatten-Konfigurationen und der damit verbundenen Leistung. Zusätzlich erfolgen der Betrieb des Medienservers und das Abarbeiten der Benutzeranfragen.

7.10.2 Messungen für TIFF-Dateien

Analog zu den Messungen für PDF-Dateien erfolgen die Messungen für TIFF-Bilddateien. Hierfür wurden 7545 Dateien mit einer Gesamtgröße von 215,8 GB herangezogen. Durch die Verwendung von TIFF-Dateien innerhalb einiger retrodigitalisierter Bildarchive hat sich eine Vielzahl an Dateien mit nahezu identischer Dateigröße ausgebildet. In Abbildung 7-26 lässt sich deutlich ablesen, dass die Streuung der Dateigrößen deutlich geringer ist, als bei den Textdateien im PDF-Format. Im Augenblick existiert eine Reihe an Dateien, die im Bereich zwischen 50 und 60 MB Größe liegen.

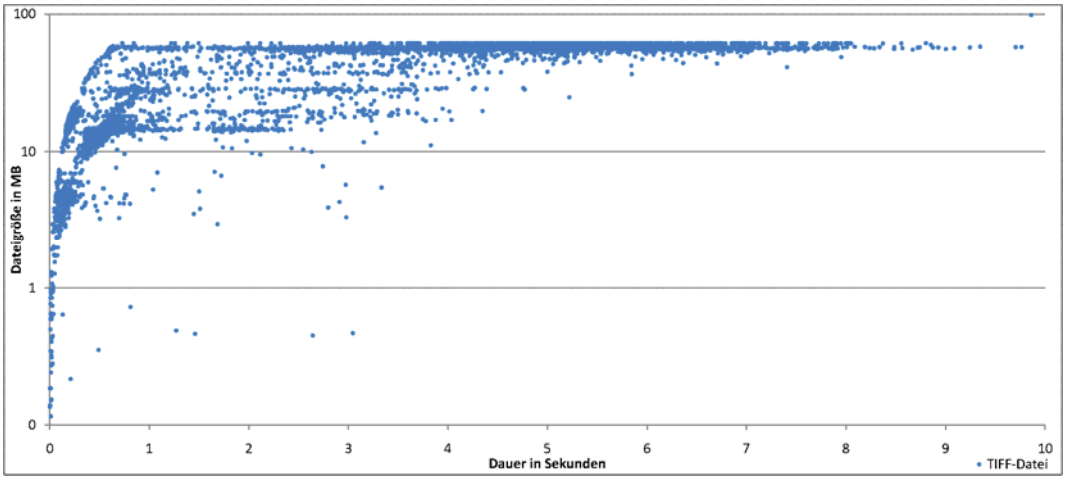

Abbildung 7-26: Erzeugungsdauer LZA-Objekt bei TIFF-Dateien

Da auch die Tests mit den TIFF-Dateien im laufenden Betrieb des Servers stattgefunden haben, ergibt sich an dieser Stelle auch wieder die vorliegende Streuung im Ergebnis, bezogen auf die Erzeugungsdauer. Da die Erzeugung aufgrund der größeren Dateigrößen bezogen beispielsweise auf die der PDF-Dateien auch länger gedauert hat, sind auch die Streuungen der Ergebnisse zu erklären. Je größer die Datei, desto leichter können andere Prozesse, die auf demselben Server laufen, die Erzeugungsdauer beeinflussen.

Die Tests für die Erzeugung der TIFF-Dateien wurden am späten Nachmittag bis in den Abend hinein durchgeführt. Somit sind auch die Störungen zuordenbar, die von anderen Prozessen auf dem Server ausgehen. In Abbildung 7-27 beeinflusst zusätzlich noch die Lage der Dateien auf unterschiedlichen Festplatten das Ergebnis, bezogen auf die Dauer der Erzeugung. Die Abbildung stellt einen Zusammenhang zwischen Dateigröße und Erzeugungsdauer her. Dabei ist auffällig, dass die Dateigröße einen gewissen Einfluss auf die Schreibgeschwindigkeit auf die Festplatte hat. Im Mittel werden dabei beim Schreiben etwa 25 MB/Sekunde erreicht. Erst ab einer Dateigröße jenseits der 30-MB-Marke bricht die Geschwindigkeit auf einen Wert zwischen 10-30 MB ein.

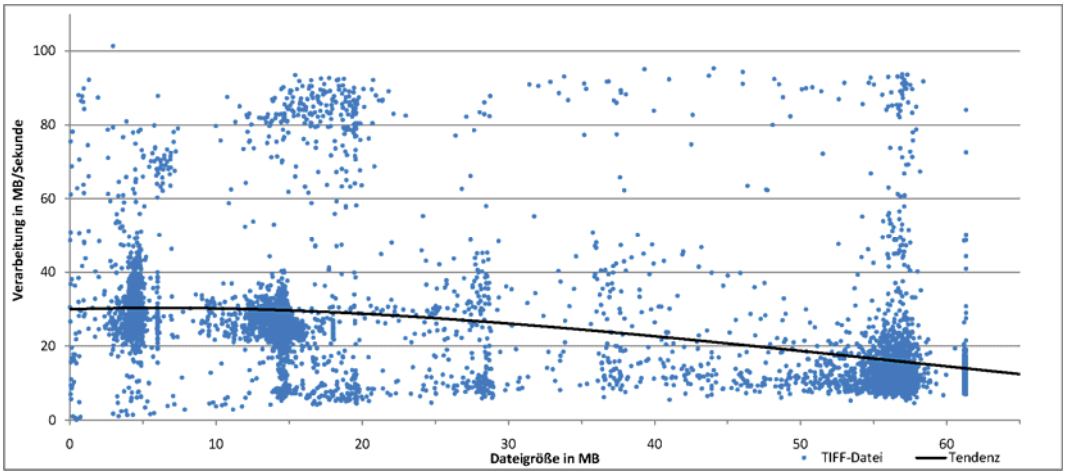

Abbildung 7-27: Verarbeitung in MB/Sekunde bei der Erzeugung (TIFF)

Dass gerade im Bereich einer Dateigösse zwischen 50-60 MB dieser deutliche Einbruch zu erkennen ist lässt sich damit erklären, dass diese Dateien auf einer langsameren Festplattenkonfiguration abgelegt sind. Sie stammen aus der Retrodigitalisierung eines Diaarchivs und wurden in einem Durchlauf auf den Medienserver abgelegt, damit sind ihre Dateigröße und die Abmessungen nahezu identisch. Die Häufung zwischen 5MB und 15MB erklärt sich analog, auch diese Daten stammen aus der Digitalisierung einer kompletten Diasammlung. Andere Bildkollektionen wachsen kontinuierlich und verteilen dadurch ihre Daten über unterschiedliche Datenspeicher mit unterschiedlicher Größe und Leistung hinweg.

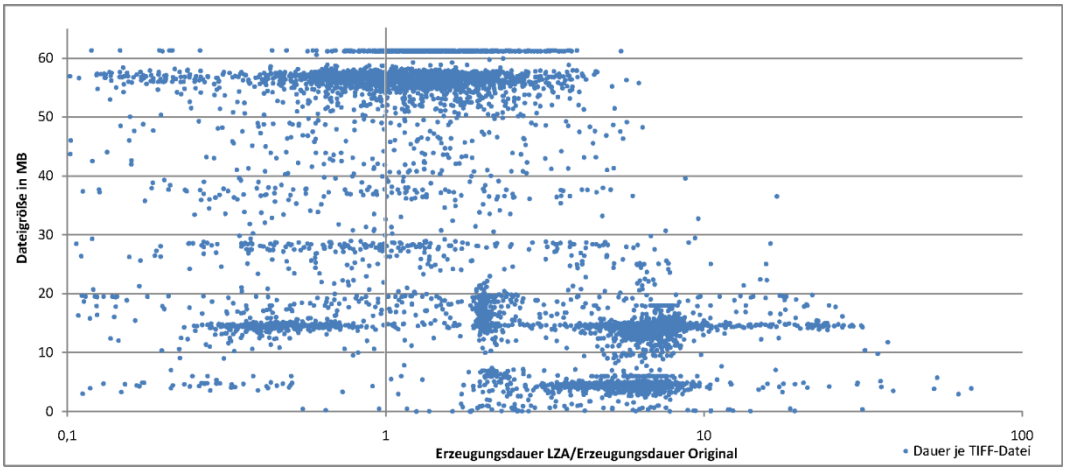

Abbildung 7-28: Vergleich Erzeugung LZA-Objekt/Original (TIFF)

In Abbildung 7-28 soll die Leistung der Implementierung bezüglich des Verhältnisses der Erzeugungssauer des Originals gegenüber der Dauer der Erzeugung der LZA-Variante bewertet werden. Dabei ist erkennbar, dass sich für den größten Teil der Dateien in deutlich kürzerer Zeit das Original rekonstruieren lässt, als die Erzeugung der LZA-Variante gedauert hat. Im Mittel kann man sagen, dass die Rekonstruktion viermal schneller ist, als die Erzeugung der LZA-Variante. Dieses Verhalten ist notwendig, da häufiger auf das Original zurückgegriffen werden muss, als eine neue LZA-Variante erzeugt wird.

Auch an dieser Stelle ist wieder zu beobachten, dass die Festplattenkonfiguration mit den großen TIFF-Dateien langsamer Daten verarbeiten kann, als die der übrigen Daten. Damit bleibt zu überlegen, ob

eine Umschichtung der Daten eventuell sinnvoll ist, um etwaig allein durch die Lage der Daten entstehende Performanz-Probleme zu beheben. Bei kleineren Dateigrößen fällt dieses Verhalten nicht so stark ins Gewicht als bei großen.

7.10.3 Messungen für JPEG-Dateien

JPEG-Dateien stellen den größten Anteil an Daten dar, die zu Messungen herangezogen werden konnten. Mit 19.451 getesteten Dateien und etwa 22,5 GB Gesamtgröße. Für die Messungen wurde das bereits erwähnte Skript verwendet und jeweils dieselben Metadaten eingebettet. Aus Abbildung 7-29 kann man sehr gut erkennen, dass ein lineares Verhalten zu beobachten ist. Somit steht die Dateigröße in direkter Proportion zur Erzeugungsdauer der Langzeitarchivierungs-Version.

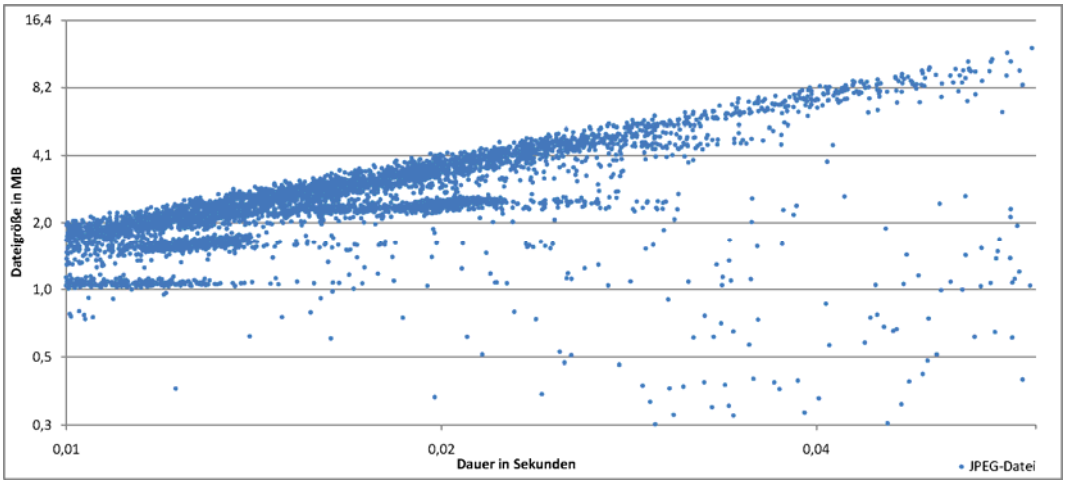

Abbildung 7-29: Erzeugungsdauer LZA-Objekt bei JPEG-Dateien

Zusätzlich ist zu erkennen, dass die Art und Weise, auf die die Metadaten eingebettet werden, etwas komplexer ist, als bei den für die anderen Dateitypen gezeigten Verfahren. Das liegt unter anderem daran, dass die Position für die Metadaten nicht bei jeder Datei an derselben Stelle zu finden ist. Somit muss die Datei im schlechtesten Fall zuerst komplett gelesen werden, um die entsprechende Position für die Einbettung zu finden. Positiv wirkt sich an dieser Stelle aus, dass JPEG-Dateien relativ klein sind und damit ein byteweises Lesen immer noch sehr schnell vonstattengeht.

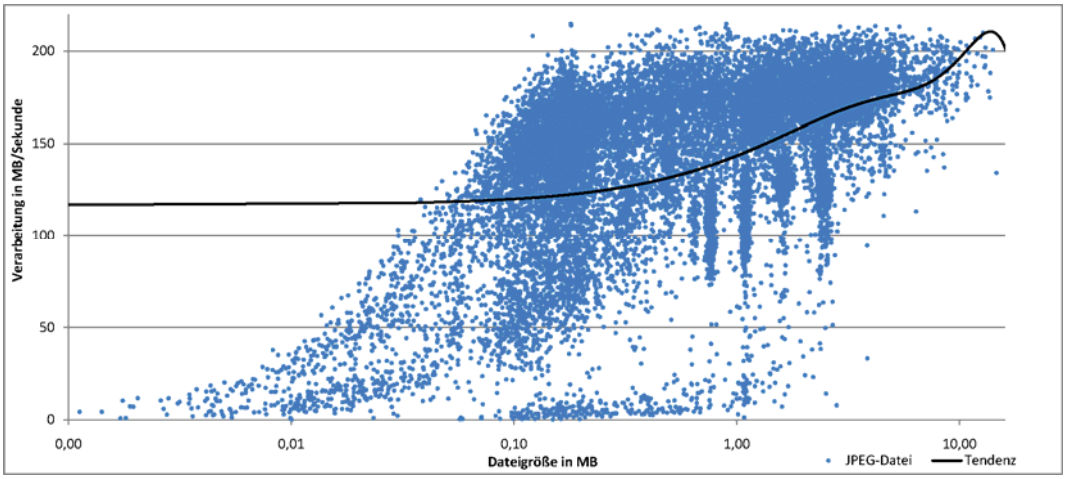

Abbildung 7-30: Verarbeitung in MB/Sekunde bei der Erzeugung (JPEG)

Vor allem durch die Dateigröße, die bei den Beispieldaten meist um eine Größe von etwa 1 MB liegt, kann die hohe Verarbeitungsgeschwindigkeit erklärt werden, die deutlich höher ausfällt, als bei vergleichbaren Dateien anderen Formats (vgl. Abbildung 7-30). Dabei wirkt sich positiv aus, dass das Betriebssystem kleinere Dateien komplett in den Arbeitsspeicher laden kann und zusätzlich die Dateien auf der etwas schnelleren Festplattenkonfiguration abgelegt wurden.

Einen regelrechten Einbruch der Performanz kann man beim Rückerzeugen der Originaldateien aus den LZA-Varianten erkennen. Das Rückerzeugen ist gut 450mal langsamer, als die Erzeugung der LZA-Variante (vgl. Abbildung 7-31). Hier gibt es keine einzige Datei, die schneller wiederhergestellt werden konnte, als die Erzeugung der Archiv-Version gedauert hatte. Damit ist klar, dass sich das JPEG-Format nicht wirklich für die Nutzung als Archivformat eignet.

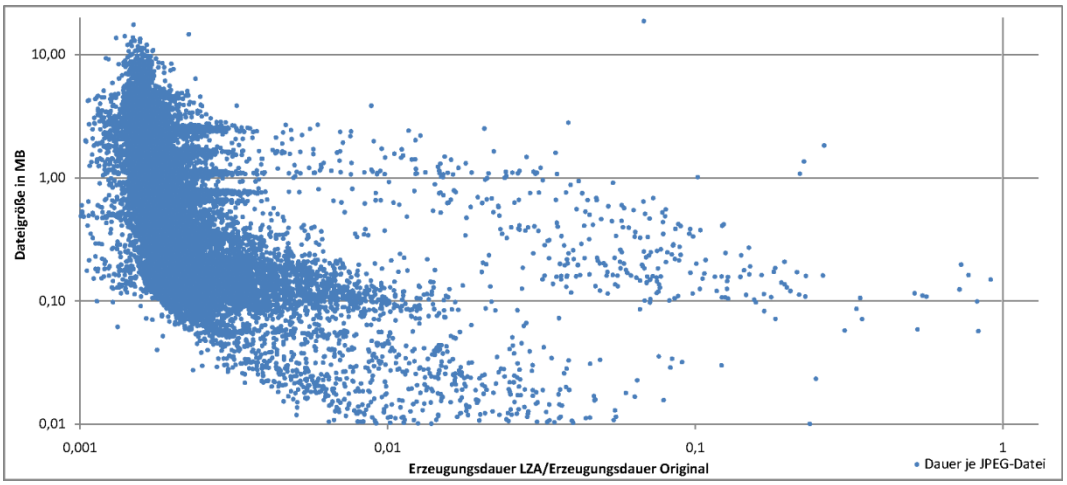

Abbildung 7-31: Vergleich Erzeugung LZA-Objekt/Original (JPEG)

7.10.4 Messungen für Multimedia-Dateien

Da sich aktuell nur sehr wenige Multimedia-Dateien auf dem Server befinden, ist auch eine sinnvolle Leistungsmessung nicht möglich. Auf dem Medienserver sind nur wenige Videodateien verfügbar, die im angepassten FLV-Format vorliegen, das Originalmaterial ist nicht vorhanden. Für ein verwertbares Messergebnis ist einerseits eine gewisse Anzahl an Dateien, aber vor allem die Originaldatei in einem geeigneten Dateiformat unbedingt erforderlich.

Audiodaten sind aktuell überhaupt nicht angefragt und im vorgestellten Verfahren lediglich vorgesehen, aber nicht umgesetzt. An dieser Stelle bieten sich Ansatzpunkte für zukünftige Arbeiten.

7.11 Besonderheiten und Probleme

Während der Umsetzung und Tests konnten an unterschiedlichen Stellen Probleme des LZA-Verfahrens beobachtet werden. Die gefundenen Stellen ließen sich klar benennen und über geeignete Gegenmaßnahmen konnte ihnen entgegengewirkt werden.

7.11.1 Erzeugung der LZA-Datei

Die Erzeugung der LZA-Datei aus dem Original bringt aufgrund der unterschiedlichen Dateiformate und Dateiversionen Schwierigkeiten in der Formalisierung mit. Ausnahmebehandlungen für diese Fälle schaffen Abhilfe.

Besonders im Bereich der PDF-Dokumente sind Probleme aufgrund der Vielfalt an Versionen und Spezialvarianten zu beobachten. Auch die Vielzahl an Programmen zur PDF-Erzeugung trägt zu den gefundenen Schwierigkeiten bei. In den zur Verfügung stehenden Testdaten war zu erkennen, dass maßgeblich das Erzeugungs-Programm der Ausgangsdatei verantwortlich für die Probleme war. Die untersuchten 5273 Dateien wurden mit 449 unterschiedlichen Programmen bzw. Versionen der Programme erzeugt. Zu Beobachten war, dass die Programme großer Hersteller (z.B. Adobe Acrobat) Kontroll- und teilweise sogar Korrekturmechanismen für korrupte PDF-Dateien mitbringen.

Bei den anderen umgesetzten Dateitypen ist festzustellen, dass die Probleme deutlich geringer sind und auch in der Umsetzung der Konvertierungsalgorithmen weniger Spezialfälle betrachten werden müssen.

7.11.2 Ursachenanalyse der Probleme bei PDF-Dateien

Bei den durchgeführten Tests wurden einige PDF-Dokumente gefunden, die nach der Umwandlung zur LZA-Version als fehlerbehaftete Datei vorlagen. Bei der Analyse der auftretenden Probleme konnten gleich mehrere Ursachen dafür aufgedeckt werden:

Unterschiedliche Dateikodierung:
Durch das Vorliegen der PDF-Dateien quasi im Plain-Text können unterschiedliche Kodierungen der Datei (z.B. Windows oder Linux-konform) zu Problemfällen bei der Bearbeitung führen. Bei der Umsetzung des Konvertierungsalgorithmus wurden zunächst als Basis korrekte PDF-Dateien herangezogen. Die Korrektheit wurde dabei mit Validierungstools sichergestellt. Beim Parsen der Ausgangsdateien konnte beobachtet werden, dass ein wesentlicher Punkt die Korrektheit des Inhaltsverzeichnisses (Cross-Reference Table) innerhalb der PDF-Datei darstellt. Falls bei diesem Parsen Probleme auftreten, werden die konvertierten Dateien immer fehlerbehaftet sein. An der Stelle muss also ein robuster Algorithmus sicherstellen, dass mögliche Probleme erkannt und abgefangen werden können.

Es stellte sich heraus, dass die Zeilenumbrüche der einzelnen Einträge der Cross-Reference Table teilweise unterschiedlich erzeugt wurden, so dass es sowohl die windowskonforme, als auch die linuxkonforme Variante geben kann. In der Spezifikation des PDF-Dateiformates ist dabei keine Vorschrift vorhanden, in welcher Form der Zeilenumbruch zu realisieren ist. Durch geeignete Unterscheidungsmechanismen konnten diese Probleme umgangen werden und korrekte LZA-Varianten der Ausgangsdateien erzeugt werden.

```
endobj xref
0 12
0000000000 65535 f
0000002133 00000 n
0000000000 65536 n
0000002010 00000 n
0000001838 00000 n
0000000608 00000 n
0000000015 00000 n
0000000212 00000 n
0000001785 00000 n
0000001899 00000 n
0000002192 00000 n
0000002244 00000 n
TrailerCode
```

Code 7–4: Beispiel Cross-Reference Table

Das Beispiel aus Code 7–4 zeigt den Aufbau einer gültigen Cross-Reference Table. In der PDF-Spezifikation ist verankert, dass die Objekte der Datei zeilenweise mit einer Länge von 20 Byte beschrieben werden. Der Parser-Algorithmus unterscheidet zwischen Carriage Return und Line Feed und wertet daraus einen korrekten Umbruch aus.

Verschiedene PDF-Versionen und Varianten:
Das PDF-Format kann in unterschiedlichen Versionen und Varianten vorliegen (siehe Kapitel 4.2.4). Gerade die möglichen Varianten bilden dabei an einigen Stellen problematische Unterschiede aus, so dass bei der Umwandlung in das LZA-Format hierauf explizit eingegangen werden muss. Vor allem die Optimierung für eine schnelle Anzeige (linearisiertes PDF) muss als Sonderfall betrachtet werden. Erforderlich macht das die geänderte Position des internen Inhaltsverzeichnisses.

Neben den verschiedenen Varianten bereiten die neueren PDF-Versionen immer wieder Probleme. So kann in Version 1.6 die komplette Cross-Reference Table, die in allen bisherigen Versionen obligatorisch war, durch eine Variation ersetzt werden. Die einzelnen Objekte innerhalb der PDF-Datei werden hintereinander angegeben und die Referenzen auf das jeweilige Folgeobjekt werden nicht absolut in XREF angegeben, sondern relativ im vorangegangenen Objekt kodiert.

Dateiversion	Vorkommen
1.0	0,07%
1.1	0,73%
1.2	6,24%
1.3	32,19%
1.4	39,73%
1.5	7,46%
1.6	13,11%
1.7	0,45%

Tabelle 7-1: Versionsverteilung der PDF-Dateien der Testdaten

Die Versionsverteilung in Tabelle 7-1 zeigt, dass lediglich knapp 14% der PDF-Dateien aufgrund des Funktionsumfangs der Version eine Sonderbehandlung benötigen. In diesem Fall bleibt zu überprüfen, ob eine Transkodierung in eine niedrigere Version das Problem beheben kann. Denn nicht nur das Inhaltsverzeichnis, auch die zusätzlichen Möglichkeiten der Versionen ab 1.5 bereiten für die LZA Nachteile (siehe Kapitel 4.2.4).

Lösungsmöglichkeiten
Durch den großen Datenvorrat für die Tests war auch gleichzeitig die Vielfalt der anzutreffenden Versionen und Variationen breit gestreut. Es war erkennbar, dass ein fehlerfreies Abarbeiten der vorliegenden Daten auch eine robuste Lösung für die Erzeugung darstellt und der Erzeugungsalgorithmus für die LZA-Version der PDF-Dateien damit auch mit allen gängigen korrekten PDF-Dateien zurechtkommt.

7.11.3 Archivierungsverwaltung

Die direkte Anbindung der Archivierungsverwaltung an den Medienserver bringt Möglichkeiten der Steuerung der Ausgabe mit. Dem Benutzer kann beispielsweise direkt über den Medienserver die Archivdatei zur Verfügung gestellt werden. Zusätzlich erkennt die Verwaltung der Metadaten Änderungen, die eine Aktualisierung der LZA-Daten erforderlich werden lassen. Der nahtlose Betrieb der Archivierung mit der Nutzung auch mehrerer Archivsysteme zur LZA erhöht den Komfort erheblich.

7.12 Ergebnisse und Bewertung des entwickelten Verfahrens

Das gezeigte Verfahren der LZA erfordert zusätzliche Komponenten, um eine sichere Langzeitverfügbarkeit der Daten eines Medienservers zu garantieren. Die einzelnen Komponenten müssen dabei aufeinander abgestimmt sein und ineinandergreifen. Folgende Komponenten sind innerhalb des Medienservers zu realisieren:

- Archivierungsverwaltung,
- Angepasste Metadatenverwaltung und
- Dateitypabhängiger Konvertierungsalgorithmus.

Für jede dieser Komponenten treten in den unterschiedlichen Szenarien Schwierigkeiten auf, die beachtet werden müssen.

7.12.1 Ergebnisse der Tests

Die Einbettung der Metadaten in unterschiedlichen Dateiformaten zur Langzeitarchivierung nach dem hier gezeigten Verfahren ist für die anzutreffenden Dateiformate auf einem Medienserver positiv zu bewerten. Für die in diesem Kapitel gezeigten Dateiformate ist das problemlos umsetzbar, weitere Formate sind denkbar, falls dies erforderlich wird. Dennoch erfordert die LZA bei Medienservern einen Zusatzaufwand. Die Bewertung erfolgt gegliedert in die Komponenten.

Archivierungsverwaltung
Die Steuerung der LZA erfolgt direkt aus dem Medienserver heraus. Nur so kann an den erforderlichen Stellen Einfluss auf das Gesamtsystem genommen werden. Vor allem die Steuerung des Datenflusses und die Benutzerführung sind an dieser Stelle von entscheidender Wichtigkeit. Zusätzlich spielt die Wahl des Archivierungssystems für die Prozesssteuerung eine entscheidende Rolle. Das entwickelte Verfahren beschränkt sich an der Stelle nicht auf eine bestimmte Archivierungshardware oder Software, sondern liefert grundlegende Konzepte zur Umsetzung.

In der für die Tests eingesetzten Implementierung zeigte sich, dass die Umsetzung der herausgearbeiteten Anforderungen an das Steuerungssystem unabhängig zum Medienserver zu sehen ist. Somit sind die gefundenen Lösungen immer auch auf andere Systeme übertragbar.

Angepasste Metadatenverwaltung
Ein weiterer Kernbestandteil für eine praktikable LZA stellt die Wahl der Metainformationen zu einem Objekt dar. Auch hier ist wieder in erster Linie das Konzept der Auswahl eines Metadatenstandards entscheidend. Die vorgestellten Standards unterstützten entsprechend ihres Einsatzzwecks bestmöglich. Innerhalb eines Medienservers ist an dieser Stelle die Führung des Nutzers entscheidend, um für die LZA brauchbare Ergebnisse zu erzielen. Dabei steht die Flexibilität im Vordergrund, eine Kombination aus unterschiedlichen Standards situationsabhängig zu erzeugen.

Konvertierungsalgorithmus
Die Aufbereitung der Dateien mit der gemeinsamen Speicherung der Metadaten und des digitalen Objekts bietet die bereits benannten Vorteile. Durch die Anwendbarkeit dieses Verfahrens für eine Vielzahl unterschiedlicher Dateiformate ergibt sich die notwendige Flexibilität. Für die meistgenutzten Formate eines Medienservers ist die Realisierung der Einbettung der Metadaten in das digitale Objekt problemlos möglich. Weitere Formate können unabhängig bei Bedarf ergänzt werden.

7.12.2 Übertragbarkeit der Ergebnisse und Bewertung des Verfahrens

Die Ergebnisse der durchgeführten Tests liefern einen Anhaltspunkt für die Machbarkeit des vorgestellten Archivierungsverfahrens. Die gemessenen Werte sind dabei immer nur im Verhältnis zueinander zu sehen und absolut gesehen abhängig von der eingesetzten Implementierung und Hardware. Im relativen Vergleich ist positiv zu bewerten, dass häufig genutzte Prozesse deutlich schneller abgearbeitet werden können, als einmalig erforderliche Methoden. Zudem ist die Erweiterungsmöglichkeit an vielen Stellen gegeben, sei es um weitere Dateitypen oder in der Anbindung unterschiedlicher Archivierungssysteme. Das vorgestellte Verfahren ist nicht auf eine bestimmte Medienserver-Software begrenzt, sondern lässt sich auf andere Systeme übertragen. Dafür eignen sich

die verschiedenen Softwaresysteme auch unterschiedlich gut. Die betrachteten Open-Source-Medienserver aus Kapitel 2.3 können beispielsweise nach der Integration der Kernbestandteile des Verfahrens ebenfalls auf diese Art der LZA zurückgreifen, falls dies erforderlich ist.

Neben der hier gezeigten Methode der LZA bei Medienservern sind auch andere Möglichkeiten denkbar. Bei einer reinen Archivierung des Datenbestandes ist vor allem die Anbindung des Archivsystems an die Daten erforderlich. Das vorliegende Umfeld beeinflusst maßgeblich die Entscheidungen zu bestimmten Verfahren. Basis bildet die Art der Archivierung, entweder Migration, Emulation oder Mischformen. Desweiteren ist zu klären, ob der Medienserver die Kontrolle über die Archivierung übernehmen soll oder externe Programme die Überwachung übernehmen. Für die Speicherung der Daten wird die Entscheidung hin zu einem Format notwendig. Entweder kommen Containerformate zum Einsatz, die Metadaten und Dateien aufnehmen können oder es erfolgt die Einbettung der Metadaten in einem Objekt zusammen mit dem Digitalisat.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

In dieser Arbeit wurde die Problematik der dauerhaften Vorhaltung und nachhaltige Sicherung großer Datenbestände als Datengrundlage eines Medienservers untersucht. Dabei wurden verschiedene Problemstellen anhand bestehender Systeme analysiert und die Grundlagen der unterschiedlichen betroffenen Themenkreise detaillierter beleuchtet und versucht, eine Erweiterung zu erarbeiten, die sowohl in der Theorie, als auch in der Praxis zum Einsatz kommen kann, um große Datenmengen unterschiedlicher digitaler Objekte zu verwalten.

Dabei stellt die in Kapitel 6 vorgestellte Methode der Einbettung eine Erweiterung der bisher üblichen Metadatenspeicherung dar. Durch die Eignung des Verfahrens für unterschiedliche Datentypen, kann auf diese Weise für die verschiedensten Datenformate eine einheitliche Speicherung erfolgen, die einerseits einmal vergebene Metainformationen sicher innerhalb der Datei aufbewahren lässt, andererseits aber auch das Ursprungsobjekt exakt wiederherstellen lässt. Diese positiven Eigenschaften des Verfahrens wurden in einer Implementierung getestet und haben sich als praktikabel erwiesen.

Kapitel 7 beschreibt das Verfahren der Einbettung für die unterschiedlichen Dateiformate, die bereits in Kapitel 4 vorgestellt wurden. Die Auswahl der Dateiformate erfolgte unter dem Hintergrund des Vorkommens auf Medienservern und der Eignung der Formate anhand der Gegenüberstellung in Kapitel 4.6. Neben der reinen Implementierung des gezeigten Verfahrens, erfolgt in Kapitel 7.10 eine Leistungsmessung und Analyse der gewonnenen Ergebnisse.

Die Algorithmen dieses Verfahrens sind dabei immer auf Performanz und Sicherheit ausgelegt, dass beispielsweise das originale digitale Objekt im inneren Datensegment nicht verändert werden muss. Durch die Übertragbarkeit und die zum Einsatz kommende Technik können für diese Methode auch andere Datentypen realisiert werden, falls das erforderlich ist. Im Kern wurde darauf geachtet, dass alle gängigen Datentypen unterstützt werden, die in der Langzeitarchivierung eine Rolle spielen. Eine Erweiterung um zusätzliche Formate ist aber jederzeit gegeben.

Zusätzlich wurde großer Wert darauf gelegt, dass bereits vorhandene Metainformationen keine Auswirkungen auf die für die Langzeitarchivierung vergebenen Metadaten haben. Vorhandene Metadaten werden entweder ergänzt oder bleiben aufgrund ihrer Lage an anderer Stelle innerhalb der Datei unangetastet. Damit ist sichergestellt, dass auch diese Informationen weiterhin erhalten bleiben.

8.2 Beitrag der Arbeit

Diese Arbeit soll einen Beitrag dazu leisten, Probleme und Schwierigkeiten zu beleuchten, die im Bereich der LZA der Medienserverdaten auftreten können. Durch eine Analyse bestehender Medienserversoftware mit unterschiedlicher Ausrichtung (als reine Publikationsplattform für Texte oder Bildarchive) konnten die Defizite gerade im Bereich der LZA benannt werden. Anhand der gefundenen Probleme wird ein Verfahren vorgestellt, dass neben der theoretischen Entwicklung auch in der Praxis zum Einsatz kommen kann. Dieses Verfahren eignet sich dabei nicht ausschließlich für die gezeigte Softwareumgebung, sondern kann in bestehende Systeme integriert oder angeschlossen werden.

Meine persönliche Leistung muss in zwei Abschnitte unterteilt werden. Das ist einerseits die Sichtung vorhandener Systeme und Methoden mit einer anschließenden Entwicklung und Modellierung der Archivierungsverfahren, die dann im zweiten Abschnitt auch umgesetzt worden sind und dabei auf ihre Praktikabilität hin untersucht werden konnten. Bereits in der Entwurfsphase hat sich dabei herausgestellt, dass eine ganze Reihe an Entwicklungen erforderlich ist, damit eine Langzeitarchivierung mit sinnvollen Sicherungsaspekten an einen Medienserver angebunden werden kann. Genau diese Komponenten, die auch von mir entwickelt wurden, möchte ich noch einmal kurz darstellen:

Vorbereitung der Daten
Bereits bei der Auswahl der nutzbaren Formate kommen von Beginn an meine Betrachtungen mit ins Spiel und führen zur Realisierung der unterstützten Formate. Dabei wurde auf die Speicherung der Metainformationen besonderer Wert gelegt, so dass die Anforderungen an den Konfigurationsbereich innerhalb der Medienserversoftware definiert werden konnten. Die betrachteten Gesichtspunkte unterstützen dabei den Nutzer sowie den Administrator gleichermaßen in der Konfiguration und Zusammenstellung notwendiger Metadaten und Dateiformate.

Verwaltung der Daten
Die Datenverwaltung des Medienservers benötigt an einigen Stellen geeignete Schnittstellen, die konfigurierbar den Datenaustausch mit Fremdsystemen garantieren. Diese Schnittstellen wurden durch meine Überlegungen so ausgelegt, dass sie über flexible Konfigurationsmöglichkeiten einen weiten Bereich an Anforderungen abdecken können. Über die Zwischenschaltung einer zusätzlichen Schicht zur Formatierung der Metadaten können diese in beliebige Formate exportiert bzw. importiert werden.

Realisierung der Archivierungsschnittstelle
Die erforderlichen Funktionen im Bereich der Schnittstelle zur Archivierung stellen eine Entwicklung nach den von mir erarbeiteten Kriterien dar. Vor allem durch die generische Auslegung mit der Möglichkeit unterschiedliche Archivierungskomponenten (auch gleichzeitig) an einen Medienserver anbinden zu können, kann eine ebenso flexible wie sichere Archivierung der Daten des Medienservers realisiert werden. Durch die Umsetzung für die Medienserversoftware der TUM mit der Anbindung an die Archivierungsmechanismen des LRZ konnten Laufzeittests durchgeführt werden und das Verfahren evaluiert werden.

Theoretische Arbeit
Zusätzlich zu der praktischen Erarbeitung und Umsetzung einer Langzeitarchivierungskomponente im Bereich der Medienserversoftware umfasst diese Arbeit im theoretischen Teil die Aspekte, die für die spätere Umsetzung als Voraussetzungen und Leitlinie herangezogen wurden. Dabei wurden durch mich mit der Betrachtung der Ausgangssituation, dem Vergleich anderer Applikationen auf dem Markt und die Bewertung verschiedener Dateiformate die Grundlagen dafür geschaffen, dass eine praktische Umsetzung mit sinnvollen Aspekten erst realisieren werden konnte. Anschließend an die Realisierung habe ich mithilfe einer Leistungsmessung eine Bewertung der gefundenen Umsetzung angeschlossen.

8.3 Ausblick

Im Rahmen dieser Arbeit erfolgte für einige der hier vorgestellten Dateitypen bereits die konkrete Umsetzung des Einbettungsalgorithmus für die LZA. Umgesetzt wurden die am häufigsten angetroffenen Dateitypen auf Medienservern (PDF, TIFF, JPEG). Für alle anderen, vor allem aus dem Multimediabereich, sind weitere Arbeiten erforderlich. Auch die Anbindung an die Archivierungshardware kann nicht umfassend realisiert werden, es existieren zu viele unterschiedliche Konzepte und Systeme. Dennoch ist bei allen Teilkomponenten stets darauf geachtet worden, dass sich die Verfahren und Schnittstellen nicht nur auf ein einziges System anwenden lassen. Vielmehr soll ein

universeller Weg der Archivierung über die Einbettung der Metadaten in die Datei aufgezeigt werden, der sich an die verschiedensten Systeme angliedern lässt.

Ein weiteres, sehr weites Feld stellt die Auswahl geeigneter Metadaten zur Archivierung dar. An dieser Stelle kann sicherlich nur eine Sichtung unterschiedlicher Standards erfolgen und eine Empfehlung hin zu einem Format ausgesprochen werden. Nachdem allerdings bestehende Systeme nicht immer alle geforderten Daten liefern können, kann nur eine Empfehlung erfolgen. Erst beim Anlegen neuer Archive ist die gezielte Auswahl zu beachten.

Insgesamt kann in dieser Arbeit nur ein Weg aufgezeigt werden, wie eine LZA der Mediaserverdaten erfolgen kann. Aufgrund unterschiedlicher Anforderungen und Umgebungen können weitere Maßnahmen erforderlich werden, die neu zu erarbeiten sind. Gerade der schnell wachsende Bereich der Multimediadaten stellt neue Herausforderungen in Bezug auf die LZA mit sich rasch ändernden Dateiformaten und zunehmenden Dateigrößen dar. Aber bereits das Erkennen der Problematik mit einem aktiven Entgegenwirken kann dem Grundproblem des drohenden Datenverlusts entgegenwirken. Werden aktiv Gegenmaßnahmen in Form einer sinnvollen LZA ergriffen, bleibt auch heutiges Kulturgut für die Nachfahren erhalten.

Mit genau dieser Problematik befassen sich zahlreiche Projekte, die aber in der Regel eher theoretisch ausgelegt sind. In Deutschland resultieren bereits aus dem Kompetenznetzwerk Langzeitarchivierung Leitlinien zu unterschiedlichen Teilproblemen der LZA. Eine umfassende allgemeingültige Lösung für die Probleme der LZA ist noch im Aufbau begriffen und aufgrund der unterschiedlichen Anwendungsgebiete bisher nur auf konzeptioneller Ebene realisiert.

Abbildungsverzeichnis

Literaturverzeichnis

[1] M.D. Adams and F. Kossentini, "Reversible Integer-to-Integer Wavelet Transforms for Image Compression: Performance Evaluation and Analysis," *IEEE Transactions on Image Processing*, vol. 9, no. 6, 2000.

[2] Adobe Systems Inc., "Adobe XMP Toolkit SDK Overview," Version 4.4 http://www.adobe.com/devnet/xmp/pdfs/XMP-Toolkit-SDK-Overview.pdf,.

[3] Adobe Systems Inc., "Extensible Metadata Platform (XMP)," http://www.adobe.com/products/xmp/, 2001.

[4] Adobe Systems, Inc., "PDF Reference, Version 1.3," Boston, http://www.adobe.com/devnet/pdf/pdfs/PDFReference13.pdf, 2000.

[5] Adobe Systems, Inc., "PDF Reference, Version 1.4," Boston, http://www.adobe.com/devnet/pdf/pdfs/PDFReference.pdf, 2001.

[6] Adobe Systems, Inc., "PDF Reference, Version 1.5," Boston, http://www.adobe.com/devnet/pdf/pdfs/PDFReference15_v6.pdf, 2003.

[7] Adobe Systems, Inc., "PDF Reference, Version 1.6," Boston, http://www.adobe.com/devnet/pdf/pdfs/PDFReference16.pdf, 2004.

[8] Adobe Systems, Inc., "PDF Reference, Version 1.7," ISO 32000 http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf, 2006.

[9] Adobe Systems, Inc., "TIFF, Revision 6.0," http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf, 1992.

[10] ANSI, X3.4-1986, "Information Systems -- Coded Character Sets 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)," 30.12.1986.

[11] S. Battista, F. Casalino, and C. Lande, "MPEG-4: a multimedia standard for the third millennium. 1," in *Multimedia, IEEE*, 1999, pp. 74-83.

[12] Uwe M. Borghoff, Peter Röding, Jan Scheffczyk, and Franz Schmalhofer, "Bewertung von Softwaresystemen zur Langzeitarchivierung digitaler Objekte," *ZfBB: Zeitschrift für Bibliothekswesen und Bibliographie 52. Jahrgang*, 2005.

[13] Uwe M. Borghoff, Peter Röding, Jan Scheffczyk, and Lothar Schmitz, *Langzeitarchivierung*. Heidelberg: dpunkt.verlag, 2003.

[14] Uwe M. Borghoff, Peter Röding, Jan Scheffczyk, and Lothar Schmitz, "Langzeitarchivierung digitaler Dokumente: Technische Grundlagen, Initiativen und Projekte," *Auskunft: Zeitschrift für Bibliothek, Archiv und Information in Norddeutschland*, 2005.

[15] Günter Born, *Referenzhandbuch Dateiformate: Grafik, Text, Datenbanken, Tabellenkalkulationen*. Bonn; München: Addison-Wesley, 1990.

[16] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, *Extensible Markup Language (XML) 1.0.*, 1998, W3C Recommendation.

[17] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson, "RAID: high-performance, reliable secondary storage," *ACM Computing Surveys (C SUR)*, pp. 145-185, June 1994.

[18] J. H. Christiansen, "A flexible object-based software framework for modeling complex systems with interacting natural and societal processes.," , Banff, Alberta, 2000.

[19] Comité International des Télécommunications de Presse, "IPTC-NAA Information Interchange Model Version 4," http://www.iptc.org/std/IIM/4.1/specification/IIMV4.1.pdf, 1999.

[20] Compuserve Incorporated, "GIF graphics interchange format: a standard defining a mechanism for the storage and transmission of raster-based graphics information," Columbus, OH, 1987.

[21] Consultative Committee for Space Data Systems. (2002, Januar) Reference Model for an Open Archival Information System (OAIS). http://public.ccsds.org/publications/archive/650x0b1.pdf.

[22] CoreCodec, Inc. (2005) Matroska. http://www.matroska.org.

[23] Prof. Dr. Wolfgang Coy, "nestor - materialien 5 - Perspektiven der Langzeitarchivierung multimedialer Objekte," nestor c/o Die Deutsche Bibliothek, Frankfurt am Main, 2006.

[24] Creative Commons-Webseite. [Online]. http://www.creativecommons.org

[25] Morgan Cundiff, "NISO Metadata for Images in XML (NISO MIX)," http://www.loc.gov/standards/mix/mix.xsd, 2003.

[26] S. Decker et al., "The Semantic Web: the role of XML and RDF," *Internet Computing, IEEE*, pp. 63-73, 2000.

[27] Susanne Dobratz, "Open-Source-Software zur Realisierung von Institutionellen Repositories - Überblick," *Zeitschrift für Bibliothekswesen und Bibliographie*, vol. 4/5, pp. 199-206, 2007.

[28] DOMEA, "Dokumenten-Management und elektronische Archivierung 2.1," http://www.verwaltung-innovativ.de/cln_047/nn_684678/DE/Organisation/domea__konzept/domea__konzept__node.html?_nnn=true, 2005.

[29] Dublin Core Metadata Initiative, "DCMI Metadata Terms," http://dublincore.org/documents/dcmi-terms/,.

[30] Erik Duval, Wayne Hodgins, Stuart Sutton, and Stuart L. Weibel, "Metadata Principles and Practicalities," *D-Lib Magazine*, vol. 8, no. 4, April 2002, ISSN 1082-9873.

[31] ECMA-6. (December 1991) 7-Bit coded Character Set, 6th ed. http://www.ecma-international.org/publications/files/ecma-st/Ecma-006.pdf.

[32] Ex Libris. DigiTool. http://www.exlibrisgroup.com/category/DigiToolOverview.

[33] Frauenhofer IIS MPEG-4 Video Software, "MPEG-4 video encoders and decoders on various platforms," http://www.iis.frauenhofer.de/amm,.

[34] B. Furrie, "Understanding MARC Bibliographic: Machine-Readable Cataloging," http://www.loc.gov/marc/, 2003.

[35] Jesse James Garret, "Ajax: A New Approach to Web Applications," http://www.robertspahr.com/courses/cwd/ajax_web_applications.pdf, 2005.

[36] GNU GPL Version 3. [Online]. http://www.gnu.org/licenses/gpl-3.0.html

[37] Elizabeth Groot, "Unique identifiers for serials: an annotated, comprehensive bibliography," *The Serials Librarian*, vol. 1, no. 1, pp. 51-75, 1976.

[38] Steve Hitchcock, "The effect of open access and downloads ('hits') on citation impact: a bibliography of studies," http://opcit.eprints.org/oacitation-biblio.html#most-recent, 2007.

[39] IBM Corp., Microsoft Corp., "Multimedia Programming Interface and Data Specifications 1.0," http://www.tactilemedia.com/info/MCI_Control_Info.html, 1991.

[40] IBM. Tivoli Storage Manager. [Online]. http://www-01.ibm.com/software/tivoli/products/storage-mgr/

[41] IntegraTUM. http://portal.mytum.de/iuk/integratum/index_html/dokumente/Flyer.pdf.

[42] International Organisation of Standardisation, "Overview of the MPEG-4 Standard," , ISO/IEC JTC1/SC29/WG11 N4668, 2002.

[43] ISBN Agentur für die Bundesrepublik Deutschland, *ISBN Handbuch*. Frankfurt am Main, 2005.

[44] ISO/IEC 646, "Information technology -- ISO 7-bit coded character set for information interchange," 1991.

[45] ISO/IEC, "Information Technology—Universal Multiple-Octet Coded Character Set (UCS)," DIS 10646-1.2 1991.

[46] Japan Electronic Industry Development Association, "Digital Still Camera Image File Format Standard," http://www.exif.org/Exif2-1.PDF, 1998.

[47] N. Kamaci and Y. Altunbasak, "Performance comparison of the emerging H.264 video coding standard with the existing standards," *In Proceedings of the IEEE International Conference on Multimedia and Expo ICME'03. July 6-9 Baltimore*, 2003.

[48] Christian Keitel, "Elektronische Archivierung in Deutschland. Eine Bestandsaufnahme," in *78. Deutscher Archivtag 2008*, Erfurt.

[49] Christian Keitel, Rolf Lang, and Kai Naumann, "Konzeption und Aufbau eines Digitalen Archivs: von der Skizze zum Prototypen," in *Ernst, Katharina (Hg.): Erfahrungen mit der Übernahme digitaler Daten. Elfte Tagung des AK „Archivierung von Unterlagen aus digitalen Systemen"*, Stuttgart, 2007, pp. 36-41.

[50] Library of Congress, "Metadata Encoding & Transmission Standard," http://www.loc.gov/standards/mets/,.

[51] Library of Congress. (2007) Sorenson Video Codec, Version 3. [Online]. http://www.digitalpreservation.gov/formats/fdd/fdd000066.shtml

[52] Yan Lu, Wen Gao, and Feng Wu, "Sprite generation for frame-fased video coding," Thessaloniki, 2001.

[53] Frank Lützenkirchen, "MyCoRe und MILESS - Architektur und Technik," http://duepublico.uni-duisburg-essen.de/servlets/DocumentServlet?id=11029, 2003.

[54] Smith MacKanzie, "DSpace : An Institutional Repository from the MIT Libraries and Hewlett Packard Laboratories," in *Lecture Notes on Computer Science*., 2002, vol. 2458, pp. 543-549.

[55] Massachusetts Institute of Technology (MIT) und den HP Labs. DSpace. http://www.dspace.org.

[56] Metadaten für die Archivierung digitaler Unterlagen, http://www.landesarchiv-bw.de/sixcms/media.php/25/konzeption_metadaten10.pdf, 2008.

[57] Miano, John, *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP*.: Addison-Wesley, ISBN 0201604434, 9780201604436, 1999.

[58] "mp3: MPEG Audio Layer 3," [Online] http://www.iis.fraunhofer.de/bf/amm/projects/mp3/index.jsp,.

[59] MyCoRe. System zur Entwicklung von Dokumenten- und Publikationsservern, Archivanwendungen, Sammlungen von Digitalisaten oder vergleichbaren Repositorien. http://www.mycore.org.

[60] National Information Standards Organistation, "Z39.87-2006: Data Dictionary - Technical Metadata forDigital Still Images," ISSN 1041-5653, 2006.

[61] Mark Nelson, "LZW Data Compression," *Dr. Dobb's Journal*, 1989.

[62] nestor, "Kompetenznetzwerk Langzeitarchivierung," http://www.nestor.de, 2003.

[63] Nestor-Arbeitsgruppe, Vertrauenswürdige Archive, "nestor - materialien 8 - nestor-Kriterien - Kriterienkatalog vertrauenswürdige digitale Langzeitarchive Version II," Frankfurt am Main, 2008.

[64] T. Ngo, "Office Open XML Overview," http://www.ecmainternational.org/news/TC45_current_work/OpenXML%20White%20Paper.pdf Retrieved April 8, 2007, 2006.

[65] Open Archives Initiative. Open Archives Initiative. [Online]. http://www.openarchives.org/

[66] OpenDML AVI M-JPEG File Format Subcommittee, "OpenDML AVI File Format Extensions, Version 1.02," http://www.the-labs.com/Video/odmlff2-avidef.pdf, 1997.

[67] K. Petermichl,. Berlin Heidelberg: Springer, 2008, pp. 687-718.

[68] Florian Plag and Roland Riempp, *Interaktives Video im Internet mit Flash*.: Springer-Verlag Berlin Heidelberg NewYork, 2006.

[69] Rechtsanwälte Goebel und Scheller, "nestor - material 1 - Digitale Langzeitarchivierung und Recht," Bad Homburg, 2004.

[70] R. Rivest, "The MD5 Message-Digest Algorithm," RFC-Editor, 1992.

[71] Jeff Rothenberg, "Ensuring the Longevity of Digital Information," *Scientific American*, no. 272, pp. 42-47, 1999.

[72] Ingrid Schmidt, "Modellierung von Metadaten," in *Texttechnologie. Perspektive und Anwendungen*. Tübingen: Stauffenburg, 2004, pp. 143-164.

[73] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable H.264/MPEG4-AVC Extension," in *IEEE International Conference on Image Processing*, Atlanta, 2006, pp. 161-164.

[74] Arne Seifert, "mediaTUM: digital collection management at TUM," in *EUNIS Congress VISION IT - Visions for use of IT in higher education*, Århus, 2008.

[75] Thomas Severiens and Eberhard R. Hilf, "nestor - materialien 7 - Zur Entwicklung eines Beschreibungsprofils für die nationale Langzeit-Archibierungs-Strategie," Frankfurt am Main, 2006.

[76] A.N. Skodras, C. A. Christopoulos, and T. Ebrahimi, "JPEG2000: The upcoming still image compression standard," *Pattern Recognition Letters*, vol. 22, no. 12, 2001.

[77] STD-DOI. Creating access to scientific Data. http://www.std-doi.de/.

[78] Werner Stephan and Frank Scholze, "Online Publikationsverbund: Erfassung und Organisation elektronischer Hochschulschriften," *Bibliotheksdienst*, no. 1, pp. 92-102, 1999.

[79] T. Schärli et al., "Gesamtschweizerische Strategie zur dauerhaften Archivierung von Unterlagen aus elektronischen Systemen," http://www.vsa-aas.org/index.php?id=110&L=0, 2002.

[80] Andrew S. Tanenbaum and Robert van Renesse, "Distributed operating systems," *ACM Comput. Surv.*, vol. 4, 1985.

[81] Robert Tansley et al., "The DSpace institutional digital repository system: current functionality," *International Conference on Digital Libraries*, 2003.

[82] Text Encoding Initiative Consortium, *Guidelines for Electronic Text Encoding and Interchange*, 4th ed. University of Virginia Press, 2002, http://www.upress.virginia.edu/books/tei.html.

[83] Text Encoding Initiative. (2009) Text Encoding Initiative. [Online]. http://www.tei-c.org

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

[85] Kenneth Thibodeau, "Overview of Technological Approaches to Digital Preservation and Challenges in Coming Years," in *Council on Library and Information Resources: The State of Digital Preservation: An International Perspective*, 2002.

[86] S. Tsekeridou, "Metadata and user profile model for personalized enhanced DTV multi-service access," in *2005. ICIP 2005. IEEE International Conference on Image Processing*, 2005.

[87] Universität Stuttgart. Open-Source-Software OPUS. http://opusdev.bsz-bw.de/trac.

[88] University of Nottingham. (2008) The Directory of Open Access Repositories - OpenDOAR. [Online]. http://www.opendoar.org

[89] University of Southampton School of Electronics and Computer Science. Open Access and Institutional Repositories with EPrints. http://www.eprints.org/.

[90] Li Weiping, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard," in *IEEE Transactions On Circuits And Systems For Videoo Technology*, vol. II, 2001.

[91] D. Wilson, "FOURCC.org - your source for video codec and pixel format information," [Online] http://www.fourcc.org/,.

[92] Ian H. Witten, Alistair Moffat, and Timothy C. Bell, *Managing Gigabytes: Compression and Indexing Documents and Images (second edition)*. San Francisco: Morgan Kaufmann Publishing, 1999.

[94] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Second Edition).W3C Recommendation," http:/www.w3.org/TR/REC-xml, 2000.

[93] World Wide Web Consortium, "HTML 4.01 Specification. W3C Recommendation," http://www.w3.org/TR/html401/, 1999.

[96] World Wide Web Consortium W3C, http://www.w3.org/TR/rdf-syntax-grammar/, February 10, 2004.

[95] World Wide Web Consortium W3C, "Scalable Vector Graphics (SVG) 1.1 Specification," http://www.w3.org/TR/2003/REC-SVG11-20030114/, 2003.

[97] "XviD - core library sorce code," [Online] http://www.xvid.org,.

[98] Francois Yergeau, "UTF-8, a transformation format of ISO 10646," RFC Editor, United States, 1998.

Index

O

P

R

S

T

U

V

W

X

Abkürzungen

IIM	Information Interchange Model
IPTC	International Press Telecommunication Council
LZA	Langzeitarchivierungs
METS	Metadata Encoding and Transfer Schema
MODS	Metadata Object Description Language
NAA	Newspaper Association of America
OAIS	Open Archival Information System
ODF	Open Document Format
OOXML	OpenOffice XML
PDF	Portable Document Format
RDF	Resource Descriptive Framework
TEI	Text Encoding Initiative
TSM	Tivoli Storage Manager
TUM	Technische Universität München

Anhang

A. Definitionen

a. Vollständige Liste der Exif-Tags mit Namen

Tag	Name
0x0100	ImageWidth
0x0101	ImageLength
0x0102	BitsPerSample
0x0103	Compression
0x0106	PhotometricInterpretation
0x010A	FillOrder
0x010D	DocumentName
0x010E	ImageDescription
0x010F	Make
0x0110	Model
0x0111	StripOffsets
0x0112	Orientation
0x0115	SamplesPerPixel
0x0116	RowsPerStrip
0x0117	StripByteCounts
0x011A	XResolution
0x011B	YResolution
0x011C	PlanarConfiguration
0x0128	ResolutionUnit
0x012D	TransferFunction
0x0131	Software
0x0132	DateTime
0x013B	Artist
0x013E	WhitePoint
0x013F	PrimaryChromaticities
0x0156	TransferRange
0x0200	JPEGProc
0x0201	JPEGInterchangeFormat
0x0202	JPEGInterchangeFormatLength
0x0211	YCbCrCoefficients
0x0212	YCbCrSubSampling
0x0213	YCbCrPositioning
0x0214	ReferenceBlackWhite
0x828D	CFARepeatPatternDim
0x828E	CFAPattern
0x828F	BatteryLevel
0x8298	Copyright
0x829A	ExposureTime
0x829D	FNumber
0x83BB	IPTC/NAA
0x8769	ExifOffset
0x8773	InterColorProfile
0x8822	ExposureProgram
0x8824	SpectralSensitivity
0x8825	GPSInfo
0x8827	ISOSpeedRatings
0x8828	OECF
0x9000	ExifVersion
0x9003	DateTimeOriginal
0x9004	DateTimeDigitized
0x9101	ComponentsConfiguration
0x9102	CompressedBitsPerPixel
0x9201	ShutterSpeedValue
0x9202	ApertureValue
0x9203	BrightnessValue
0x9204	ExposureBiasValue
0x9205	MaxApertureValue
0x9206	SubjectDistance
0x9207	MeteringMode
0x9208	LightSource
0x9209	Flash
0x920A	FocalLength
0x927C	MakerNote
0x9286	UserComment
0x9290	SubSecTime
0x9291	SubSecTimeOriginal
0x9292	SubSecTimeDigitized

0xA000	FlashPixVersion
0xA001	ColorSpace
0xA002	ExifImageWidth
0xA003	ExifImageLength
0xA005	InteroperabilityOffset
0xA20B	FlashEnergy
0xA20C	SpatialFrequencyResponse
0xA20E	FocalPlaneXResolution

0xA20F	FocalPlaneYResolution
0xA210	FocalPlaneResolutionUnit
0xA214	SubjectLocation
0xA215	ExposureIndex
0xA217	SensingMethod
0xA300	FileSource
0xA301	SceneType
0xA302	CVAPattern

b. Vollständige Liste der Baseline-TIFF Tags

Code (HEX)	Name	Beschreibung
00FE	NewSubfileType	Beschreibung Subfile
00FF	SubfileType	Beschreibung Subfile
0100	ImageWidth	Bildbreite
0101	ImageLength	Bildlänge
0102	BitsPerSample	Bits pro Komponente
0103	Compression	Kompressionsschema
0106	PhotometricInterpretation	Farbraum der Bilddaten
0107	Thresholding	Schwellenwert (s/w Bild)
0108	CellWidth	Breite Ditherings/Halbtonmatrix
010A	FillOrder	Logische Bitreihenfolge innerhalb des Bytes
010E	ImageDescription	Inhaltsbeschreibung (Metadaten)
010F	Make	Scanner Hersteller (Erzeuger)
0110	Model	Scanner Modell
0111	StripOffsets	Offset eines Streifens
0112	Orientation	Bildausrichtung
0115	SamplesPerPixel	Eintrag pro Pixel
0116	RowsPerStrip	Anzahl Zeilen je Streifen
0117	StripByteCounts	Anzahl Bytes nach Kompression
0118	MinSampleValue	Min. benutzter Komponentenwert
0119	MaxSampleValue	Max. benutzter Komponentenwert
011A	XResolution	Auflösung (Bildbreite)
011B	YResolution	Auflösung (Bildhöhe)
011C	PlanarConfiguration	Speicherart
0120	FreeOffsets	Byte Offset von Strings
0121	FreeByteCounts	Anzahl Bytes in Strings
0122	GrayResponseUnit	Auflösung der GrayResponseCurve.
0123	GrayResponseCurve	Pixelschwärzung (s/w Bild)
0128	ResolutionUnit	Auflösungseinheit
0131	Software	Softwarename
0132	DateTime	Erzeugungsdatum
013B	Artist	Erzeuger (Autor)

013C	HostComputer	Computer-System der Erzeugung
0140	ColorMap	Farbpalette
0152	ExtraSamples	Beschreibung der Extrakomponenten
8298	Copyright	Copyright Bemerkung

c. Vollständige Liste der Metadatenfelder nach ID3v2

Tag	**Originalbeschreibung**
AENC	Audio encryption
APIC	Attached picture
COMM	Comments
COMR	Commercial frame
ENCR	Encryption method registration
EQUA	Equalization
ETCO	Event timing codes
GEOB	General encapsulated object
GRID	Group identification registration
IPLS	Involved people list
LINK	Linked information
MCDI	Music CD identifier
MLLT	MPEG location lookup table
OWNE	Ownership frame
PCNT	Play counter
POPM	Popularimeter
POSS	Position synchronisation frame
PRIV	Private frame
RBUF	Recommended buffer size
RVAD	Relative volume adjustment
RVRB	Reverb
SYLT	Synchronized lyric/text
SYTC	Synchronized tempo codes
TALB	Album/Movie/Show title
TBPM	BPM (beats per minute)
TCOM	Composer
TCON	Content type
TCOP	Copyright message
TDAT	Date
TDLY	Playlist delay
TENC	Encoded by
TEXT	Lyricist/Text writer
TFLT	File type
TIME	Time

TIT1	Content group description
TIT2	Title/songname/content description
TIT3	Subtitle/Description refinement
TKEY	Initial key
TLAN	Language(s)
TLEN	Length
TMED	Media type
TOAL	Original album/movie/show title
TOFN	Original filename
TOLY	Original lyricist(s)/text writer(s)
TOPE	Original artist(s)/performer(s)
TORY	Original release year
TOWN	File owner/licensee
TPE1	Lead performer(s)/Soloist(s)
TPE2	Band/orchestra/accompaniment
TPE3	Conductor/performer refinement
TPE4	Interpreted, remixed, or otherwise modified by
TPOS	Part of a set
TPUB	Publisher
TRCK	Track number/Position in set
TRDA	Recording dates
TRSN	Internet radio station name
TRSO	Internet radio station owner
TSIZ	Size
TSRC	ISRC (international standard recording code)
TSSE	Software/Hardware and settings used for encoding
TXXX	User defined text

	information frame
TYER	Year
UFID	Unique file identifier
USER	Terms of use
USLT	Unsychronized lyric/text transcription
WCOM	Commercial information
WCOP	Copyright/Legal information
WOAF	Official audio file webpage

WOAR	Official artist/performer webpage
WOAS	Official audio source webpage
WORS	Official internet radio station homepage
WPAY	Payment
WPUB	Publishers official webpage
WXXX	User defined URL link frame

B. Testskript

Skript stat.py zur Erzeugung der statistischen Werte für die LZA-Dateiverarbeitung. Im Beispiel ist die Version für PDF-Verarbeitung dargestellt. Die anderen Datentypen werden analog verarbeitet.

```
import sys
sys.path += ["../", "."]
import os.path
import core
import core.config as config
import lib.lza.lza as l
import datetime

data = open("lib/data.txt", "r")
data = data.read()
input = open("lib/input.txt", "r")
output = open("lib/output.log", "w")

output.write("filename;filesize;starttime;endtime;timediff;outputsize;startorig;endorig;origdiff;origs
         ize\n")

s time = ""
e time = ""
s_time_remove = ""
e time remove = ""

for line in input:
    line = line.strip()
    filename = config.get("paths.datadir")+line

    if os.path.exists(filename):
         lza = l.LZA(filename)

        s time = datetime.datetime.now()
         lza.writeMediatumData(data, "lib/out.pdf")
        e time = datetime.datetime.now()

         lza = l.LZA("lib/out.pdf")

        s_time_remove = datetime.datetime.now()
        _lza.getOriginal("lib/orig.pdf")
        e time remove = datetime.datetime.now()

        output.write(line + ";" + str(os.path.getsize(filename)) + ";" + str(s time) + ";" +
             str(e time) + ";" + str(e time-s time) + ";" + str(os.path.getsize("lib/out.pdf")) + ";"
             + str(s_time_remove) + ";" + str(e_time_remove) + ";" + str(e_time_remove-s_time_remove)
             + ";" + str(os.path.getsize("lib/orig.pdf")) + "\n")
input.close()
output.close()
```

C. Serverkonfiguration Livesystem (mediatum2)

Hersteller:

DELL Poweredge 2800 Server

Prozessor:

2x Intel Intel(R) Xeon(TM) CPU 3,8 GHz

Festplatte:

SCSI RAID1 mit 69GB (Systempartition)
bestehend aus 2 x 79GB Einzelfestplatten
SCSI RAID5 mit 572GB (Datenpartition) /data/hd2
Bestehend aus 2 x 320GB Einzelfestplatten
SCSI RAID5 mit 858GB (Datenpartition) /data
Bestehend aus 3 x 320GB Einzelfestplatten

Arbeitsspeicher:

4GB