# Grundlagen eines Modells zur Archivierung elektronischer Unterlagen*

Karl-Ernst Lupprian und Rodrigo Readi Nasser
Generaldirektion der Staatlichen Archive Bayerns†

19.02.2003

Die Grundlagen eines Modells zur Archivierung elektronischer Unterlagen innerhalb eines Datenträgers werden vorgestellt. Zu archivierende elektronische Unterlagen sollen nach ihm mit Metadatendateien hierarchisch geordnet und beschrieben. Diese Metadatendateien ordnen jeder Unterlage eine Bezeichnung zum Auffinden zu, die bei der Erschließung in den Findmitteln benutzt werden kann. Da als zu archivierende Unterlage eine Datei mit einem Verweis auf eine archivierte physische Unterlage möglich ist, ist dieses Modell auch für hybride Akten geeignet. Das Modell setzt eine Vereinbarung mit der Behörde über die Form der Abgabe voraus. Ein in der Generaldirektion der Staatlichen Archive Bayerns erstellter Prototyp eines Programms zum Lesen der archivierten elektronischen Unterlagen wird beschrieben.

## §1
### Träger und Format

Daten werden auf einem **Datenträger** (Träger) in einem bestimmten **Format** gespeichert. Die möglichen Formate sind vom Träger abhängig. Papier, Film und Magnetband sind Beispiele von Trägern. Die Schriftart auf Papier oder Film ist ein Beispiel von Format, ASCII und Bitmap sind Beispiele von Formaten für Magnetbänder. In diesem Aufsatz beschäftigen wir uns nur mit Trägern für elektronische Dateien und mit entsprechenden Formaten. Man muß zwischen der Speicherung von Dateien auf einem Träger (Gerät) und der Archivierung des Trägers, eines physischen Objekts, unterscheiden. Diese letzte Aufgabe wird einem heutigen Archiv sicher nicht fremd sein; sie hat aber nur einen Sinn, wenn die auf dem Träger gespeicherten Dateien lesbar bleiben, wenn sie erschlossen werden können.

Eine Datei besteht aus einer Folge von **Bits**. Jeder Bit hat entweder den Wert `0` oder den Wert `1`. Nur wenn das Format bekannt ist, kann man den Inhalt der Datei erkennen, vielleicht als einen Text oder ein Bild, vielleicht als eine Aneinanderreihung mehrerer Dateien, die wiederum Inhalte verschiedener Art in verschiedenen Formaten enthalten. Das Format wird von Technikern bestimmt, wie das Verhalten eines bestimmten Trägers als Gerätetyp. Die Lesbarkeit von heute gespeicherten Daten in der Zukunft ist ebenso von physischen Änderungen des Trägers durch Alterung wie vom Vergessen des Formats oder des Verhaltens des Gerätes negativ beinflußt. Deswegen reden wir sowohl für Träger wie für Formate von **Stabilität** und von **Volatilität** der gespeicherten Daten. Die Stabilität bei längerfristiger Speicherung ist vom technischen Vermögen von Menschen, denen die benutzte Speicherungstechnik fremd erscheinen kann, abhängig und wird deswegen durch **Einfachheit** der Formate und des Gerätes erhöht.

*Diese Arbeit entstand im Rahmen eines von der DFG geförderten Projekts.
†Schönfeldstr. 5, 80539-München. iuk@gda.bayern.de

Die Übertragung in zeitgemäße Formate und Träger (Migration) kann Stabilität fördern, ist aber zu riskant und teuer, ein Teil der Information kann verloren gehen. Allein die ursprünglichen Träger und Formate sagen etwas über die Daten aus: davon lebt die Paläographie. Es ist nicht nur die abstrakte Information wichtig: das Aussehen eines Dokuments hat auch eine Wirkung. Das Aneinanderreihen von Programmen, die wie in einem Fließband ein Format in ein anderes übertragen (Emulation), ist nicht nur teuer und ineffizient, sondern auch eine schlechte Programmierpraxis, besonders wenn neue Programmschichten von Menschen geschrieben werden, die die älteren nicht verstehen.

Zeitgemäße Träger und Formate sind günstiger und leichter zu handhaben, aber aus den oben erwähnten Gründen für die langfristige Speicherung oder Archivierung nicht unbedingt geeignet. Die Wahl von Träger und Format sollte also davon abhängig sein, ob der Zweck allein mit der gewöhnlichen Erstellung und dem problemlosen Lesen der Daten erfüllt ist oder ob auch die Speicherung eine wichtige Rolle spielt, im letzten Fall sollte man zwischen kurz-, mittel- und langfristiger Speicherung unterscheiden. Einerseits werden Daten, die erstellt und zunächst kurzfristig gespeichert werden, eventuell später längerfristig gespeichert; andererseits werden längerfristig gespeicherte Daten nicht nur gespeichert, sondern auch gelegentlich gelesen, am liebsten mit zeitgemäßen Mitteln. Aus diesem Grund sollte man bei der Wahl der Träger und Formate für die verschiedenen Zwecke die Möglichkeit der Übertragung vorsehen. **Übertragung** heißt hier nicht, daß die Daten im ursprünglichen Format und Träger verloren gehen, sondern daß sie in ein anderes Format und einen anderen Träger kopiert werden, um zum Beispiel die Lesbarkeit für eine Zeitspanne zu erleichtern, ohne auf die zu schonenden Träger für die Langzeitarchivierung ständig zugreifen zu müssen.

Zur Zeit scheint für die Langzeitarchivierung das beste Format für Bilder das Bild selbst zu sein, der einfachste, physisch stabilste und günstigste Träger dafür Mikrofilm. Die Lesbarkeit der Daten wäre dann allein durch ihre Vergrößerung garantiert, was technisch sehr einfach ist. Filme können gescannt werden, ihr Inhalt in ein zeitgemäßes Format und auf einen elektronischen Träger übertragen werden. Als Bilder in Film kann man auch ASCII-Zeichen und Bits darstellen, nach dem Scannen und der Übertragung in ein elektronisches Format können sie wieder in ihre ursprüngliche Form mit **OCR-Methoden** gebracht werden, wenn die gewählte Bilddarstellung dafür geeignet ist.

## §2
### Dateisysteme für die Speicherung von Dokumenten

Betriebssysteme verwalten **Dateisysteme** zur Speicherung von Dateien. Dateien werden gewöhnlich in **Verzeichnissen** hierarchisch geordnet. Jeder Datei und jedem Verzeichnis entsprechen **technische Metadaten** zum Auffinden, Identifizieren, Lesen und Ändern. Dateisysteme von Betriebssystemen sind für den Zweck der Archivierung einerseits zu kompliziert, andererseits zu einfach, und deswegen nicht geeignet. Sie sind zu kompliziert, weil sie für die Verwaltung des Speicherplatzes auf Datenträgern, nicht für die Archivierung, entworfen sind. Bei der Archivierung setzt man voraus, daß die Datei sich nicht mehr ändert, was das Betriebssystem nicht voraussetzen kann. Deswegen kann es eine Datei in mehreren Teilen des Datenträgers zerstreut speichern. Sie sind zu einfach für unseren Zweck, weil wir jede Datei und jedes Verzeichnis **mit beliebig vielen Metadaten** versehen wollen, nicht nur mit den vorgesehenen technischen Metadaten.

Ein altes Format für Archivierung ist das **TAR-Format** (Tape Archiving), es kann eigentlich auch als ein Dateisystem betrachtet werden. Mit ihm können mehrere in Verzeichnissen geordnete Dateien als nur eine Datei dargestellt werden, diese Datei kann dann unabhängig vom Betriebssystem auf einem Datenträger gespeichert werden, um mit ihr zu einem späteren

Zeitpunkt die Dateien und Verzeichnisse wiederherstellen zu können, sogar in einem anderen Betriebssystem als dem ursprünglichen. TAR ist ein sehr einfaches Format. Die Dateien und Verzeichnisse werden in diesem Format in ihrer natürlichen Ordnung aneinandergereiht, vor jeder Datei und jedem Verzeichnis werden die technischen Metadaten gespeichert, dieser Teil heißt **Kopf** (header) und hat eine feste, von der Datei oder vom Verzeichnis unabhängige Größe. Es gibt mehrere Versionen von TAR, die sich in der Struktur der Köpfe unterscheiden; die neueren erlauben das Speichern von mehr technischen Metadaten, die ältere (UNIX V7) ist allgemeiner lesbar und deswegen zu bevorzugen. Da die Metadaten zur Auffindung von Dateien und Verzeichnissen in der ganzen TAR-Datei zerstreut sind, zusammen mit der entsprechenden Datei oder Verzeichnis, sind bei Beschädigung eines Teiles des Datenträgers die Dateien und Verzeichnisse im nicht beschädigten Teil leicht wiederherzustellen, was bei Dateisystemen von Betriebssystemen nicht unbedingt der Fall ist. TAR ist nicht unmittelbar für unseren Zweck zu benutzen, weil es außer einigen technischen Metadaten andere nicht vorsieht.

Wenn wir in zusätzlichen, auch zu speichernden Dateien unsere Metadaten, einschließlich der hierarchischen Verzeichnisstruktur, kodieren, dann genügt für unseren Zweck ein Dateisystem, das eine Reihe von Dateien als eine Datei darstellt und nur den Dateinamen als Metadatum speichert. Die Dateinamen sollten zur Auffindung der Datei dienen und in den Metadatendateien für die Verweise auf andere Dateien benutzt werden. Aus dem Dateinamen muß erkennbar sein, ob eine Datei eine Metadatendatei ist. Es genügt also das alte TAR-Format, sogar ein viel einfacheres Format. Um die Lesbarkeit der archivierten Unterlagen ohne besondere Programme zu garantieren, könnten die Metadatendateien aus lesbaren ASCII-Zeichen (7 Bit) bestehen und inhaltlich von Menschen leicht verständlich sein. Am besten in Zeilen von nicht mehr als 80 Zeichen, möglichst nicht mehr als 72.

Diese Reihe von Dateien, die auch Metadatendateien enthält, könnte auf Mikrofilm aufgenommen werden, in derselben Ordnung der Reihe. Eventuell muß eine Datei in mehreren Bildern dargestellt werden. In jedem Bild einer Datei muß der Name der Datei, die Anzahl der Bilder in der Datei und die Position des Bildes in der Datei so stehen, daß sie nicht nur von Menschen, sondern auch automatisch beim Scannen des Filmes leicht erkennbar seien. Man will mit dem Film die Reihe von Dateien wiederherstellen können, jede Datei mit dem ursprünglichen Namen. Nur das Format der Bilddatei darf sich dabei ändern, jedoch nicht bei Metadatendateien, die wieder in ihre ursprüngliche Form (ASCII) gebracht werden sollen. Deswegen sollten die Metadatendateien mit einer OCR-lesbaren Schrift aufgenommen werden. Die zuverlässige Anwendung von OCR sollte ein Maßstab für die Qualität der Filmaufnahme sein.

## §3
### Allgemeine Struktur der Metadatendateien

Die Metadatendateien sollen einerseits für Menschen leicht zu verstehen, andererseits mit einem Rechner leicht zu verarbeiten sein. Diese Metadatendateien sollen die Dateien in einer hierarchischen Verzeichnisstruktur ordnen und den Dateien und Verzeichnissen Metadaten zuordnen. Für diesen Zweck ist **SGML** (Standard Generalized Markup Language, siehe [1]), sogar eine eingeschränkte Form dieser Beschreibungssprache, sehr angemessen: SGML ist eine einfache Konvention, um strukturierte ASCII-Texte zu schreiben und sie mit allgemeinen Programmen bearbeiten zu können. Die **Marken** (Tags) und ihr Gebrauch werden durch eine **DTD** (Document Type Definition, siehe [1]) bestimmt, die entweder in der Metadatendatei selbst oder in einer zusätzlichen ASCII-Datei steht, die hier auch als Metadatendatei zu betrachten ist.

Wir teilen die Marken in **einführende** und **beschreibende** ein, und entsprechend die mit ihnen gebildeten Elemente. Ein einführendes Element führt entweder ein Verzeichnis für ein Objekt oder einen Verweis auf eine Datei ein. Im letzten Falle reden wir auch über ein **verweisendes** Element, diese können auch auf Metadatendateien der hier beschriebenen Art verweisen. Bei Verwaltungsakten können diese Objekte zum Beispiel Akten, Vorgänge und Dokumente sein; das Verzeichnis für ein Dokument kann zum Beispiel Elemente mit Verweisen auf Dateien mit dem Hauptschriftstück, mit Anlagen oder mit dem Geschäftsgang enthalten; das Verzeichnis für einen Vorgang die Verzeichnisse für seine Dokumente. Die beschreibenden Elemente kommen innerhalb einführender Elemente vor und beschreiben die entsprechenden Objekte, sie sind also die eigentlichen Träger der Metadaten.

Jedes einführende Element enthält zuerst einige beschreibende Elemente, danach einige (oder keine) einführende Elemente für seine Bestandteile. Die beschreibenden Elemente eines einführenden Elementes sind: zuerst ein beschreibendes Element des Typs `TXT`, dessen Inhalt eine von Menschen lesbare Bezeichnung des Objekts ist; danach ein beschreibendes Element des Typs `TYP`, dessen Inhalt ein einteilendes Stichwort (eventuell eine leere Zeichenkette) ist; danach andere beschreibende Elemente, deren Typen von der DTD und dem Inhalt vom Element `TYP` abhängig sind; und endlich beliebig viele (oder keine) Elemente des Typs `ANM` mit Anmerkungen. Ein Element `TYP` kommt nur im Inhalt einführender Elemente vor. Alle einführenden Elemente können das Attribut `N` für einen im übergeordneten Verzeichnis eindeutigen **Kurznamen**, `DT` für das **Entstehungsdatum**, `ZR` für **Zugriffsrechte** und `ST` für **Status** haben. Das Attribut `ST` kommt nur bei einführenden Marken vor: durch `TYP` und `ST` erkennt man diese Marken. Mit dem Namen einer Metadatendatei und einem Pfad, der mit den Kurznamen der geschachtelten einführenden Elemente gebildet wird, kann man eine Bezeichnung jedes einführenden Elements bilden. Diese Bezeichnung zusammen mit einem Hinweis auf den Träger kann als **Auffindungsmerkmal im Archivfindmittel** dienen.

Ein verweisendes Element enthält nur beschreibende Elemente. Nach `TXT` und `TYP` enthählt es `ID`, `ID1`, `ID2` für die Auffindungsmerkmale der Datei, `GROESSE` für ihre Größe, `FORMAT` für ihr Format. Das Element `FORMAT` enthält das Format als Stichwort und eventuell ein Element `MIT` für einen Hinweis auf die Software, mit der sie in diesem Format erzeugt wurde. In der Regel ist nur eine Marke für Verweise notwendig, wir benutzen die Marke `STUECK` dafür. Wir erlauben in der `DTD` grundsätzlich das Vorkommen von Elementen `STUECK` in jedem nicht verweisenden, einführenden Element, und zwar in jeder Stellung unter seinen einführenden Elementen. Ein verweisendes Element könnte auch für Verweise auf andere Objekte als Dateien benutzt werden, zum Beispiel auf ein einführendes Element in einer anderen Metadatendatei (durch den Dateinamen und den Pfad) oder auf eine physische Unterlage im Archiv. Anstatt unmittelbar auf physische Unterlagen zu verweisen, wäre es vielleicht besser auf eine Datei zu verweisen, deren Inhalt zum Bezeichnen, Auffinden, Identifizieren und Beschreiben der physischen Unterlage dient.

Die Metadatendateien haben einen Typ, der von der obersten (äußersten) Marke der Dokumentinstanz angegeben wird. Diese ist eine einführende Marke, das entsprechende Element kann unmittelbar nach `TXT` und `TYP` auch Elemente des Typs `ID`, `ID1`, `ID2` und `FORMAT` enthalten, um Auffindungsmerkmale (Dateinamen) und das Format der Metadatendatei anzugeben. Das ist nützlich, um Chaos bei Umbenennungen zu vermeiden, und um automatisch die Software aufzufinden, die die Datei im angegebenen Format lesen soll.

Die Metadatendateien, einschließlich der DTD, sollen von Menschen leicht zu lesen und effizient auf Mikrofilm unterzubringen sein: dieser Grundsatz muß besonders beim Entwurf der DTD und beim Formatieren der Metadatendateien gelten. Die Metadatendateien sollten aus lesbaren ASCII-Zeichen (7-Bit), in Zeilen von möglichst nicht mehr als 72, maximal

80, Zeichen bestehen. Die Zeichen des deutschen Alphabets, die keine ASCII Entsprechung haben, können mit den Entitäten "`&Ae;`", "`&Oe;`", "`&Ue;`", "`&ae;`", "`&oe;`", "`&ue;`", "`&sz;`" umschrieben werden. Optionale Elemente und Attributsgleichungen sollten nur dann vorkommen, wenn sie Information enthalten. Die DTD sollte das Weglassen von Marken, Elementen und Attributsgleichungen dort zulassen, wo es die eindeutige Lesbarkeit nicht beeinflußt. Die Marken und Attributsnamen sollten kurz sein, aber den möglichen Inhalt des Elementes oder des Attributswertes andeuten, wenigstens im Kontext der Metadatendatei. Eine DTD, die nicht viele Metadatendateien definieren soll, soll im SGML-Prolog der Datei eingeführt werden, sonst in einer getrennten Datei.

## §4

### Besondere Marken für Metadaten bei der Abgabe von Verwaltungsakten

Wir wollen mit einer einzigen Metadatendatei einen Aktenband beschreiben. Die einführenden Marken sind `AKTENBAND`, `VORGANG`, `DOKUMENT` und `STUECK`. Ein Element `AKTENBAND` kann Elemente des Typs `VORGANG` oder `STUECK` enthalten, ein Element `VORGANG` Elemente des Typs `DOKUMENT` oder `STUECK`, ein Element `DOKUMENT` Elemente des Typs `STUECK`. Der Gebrauch ist selbsterklärend. Ein Element `DOKUMENT` stellt ein Verzeichnis dar, das Verweise auf die **Primärdateien** mit dem Inhalt eines Dokuments enthält. Ein Element `VORGANG` stellt ein Verzeichnis dar, das die Bestandteile eines Vorgangs enthält, und das sind als Elemente `DOKUMENT` dargestellte Dokumente. Ein Element `AKTENBAND` kommt nur einmal vor, als oberstes (äußerstes) Element, und enthält Elemente des Typs `VORGANG`. Das beschreibende Element `TYP` in einem Element `DOKUMENT` kann ein Stichwort wie `EINGANG`, `ENTWURF` oder `AKTENVERMERK` enthalten, das beschreibende Element `TXT` seinen Betreff oder eine andere Bezeichnung. Zusätzliche Verweise mit `STUECK` können Metadaten enthalten.

Ebenfalls wollen wir mit einer einzigen Metadatendatei eine Abgabe mehrerer Aktenbände beschreiben. Diese Metadatendatei hat eine andere DTD als die für Aktenbände. Die einführenden Marken sind `ABGABE`, `GRUPPE`, `AKTENSEGMENT` und `STUECK`. Ein Element `AKTENSEGMENT` kann Elemente des Typs `STUECK` enthalten, ein Element `ABGABE` oder `GRUPPE` Elemente entweder des Typs `GRUPPE`, `AKTENSEGMENT` oder `STUECK`. Ein Element `AKTENSEGMENT` kann Elemente `STUECK` mit Verweisen auf Metadatendateien von abgegebenen Aktenbänden, wie die oben beschriebenen, oder auf ASCII-Dateien mit Metadaten zur Auffindung, Identifizierung und Beschreibung physischer Aktenbände enthalten. Ein Element `AKTENSEGMENT` soll alle Aktenbände einer Abgabe zusammenfassen, die dasselbe Aktenzeichen haben. Ein Element `GRUPPE` entspricht einer Gruppe des Aktenplans und dient einer Strukturierung der Abgabe, es müssen jedoch nicht alle Gruppen des Aktenplans abgebildet werden, im Gegenteil: Überschaulichkeit der Metadatendatei geht der Strukturierung vor. Ein Element `ABGABE` kommt nur einmal vor, als oberstes Element. Die Metadatendatei einer Abgabe sollte genug Information enthalten, um aus ihr automatisch das Findmittel bilden zu können, was von den beschreibenden Elementen in ihr abhängt.

Die in unserem Prototyp benutzten DTD für diese zwei Typen von Metadatendateien werden am Ende dieses Aufsatzes beigefügt, wir erörtern nun den Gebrauch einiger beschreibender Marken.

Ein beschreibendes Element `LFD` enthält ein Kennzeichen, das das eingeführte Objekt einer Gruppe zuordnet, und eine laufende Nummer in seinem Attribut `N`, die die Stellung des Objekts innerhalb der Gruppe angibt. Ein Element `VRG` ist immer optional und enthält die analoge Information des Vorgängers. `LFD` und `VRG` können dazu benutzt werden, Aktensegmente verschiedener Abgaben in einen Kontext zu bringen. Elemente `DATUM` und Attribute `DT` stehen für die Angabe von Datum und/oder Uhrzeit, grundsätzlich in iso-8601. Der Inhalt

von DATUM kann die Angabe eines Zeitpunkts sein; oder Elemente VON und BIS mit solchen Angaben, wenn es um eine Zeitspanne geht. Bei Zeitangaben ist es wichtig, die Abweichung von der 0-Zone (UTC, GMT) anzugeben, um MEZ (+01) von MESZ (+02) zu unterscheiden. Das Jahr soll vierstellig vorkommen. Da der Wert von DT ein MTOKEN ist, sind da die Zeichen ":" und "+" nicht zulässig. Für "+" könnte "--" benutzt werden. Für das Datum ist die Form JJJJ-MM-TT in DATUM und DT zu benutzen. Für die Uhrzeit die Form HH:MM:SS+HH in DATUM, die Form HHMMSS--HH in DT. Wenn beide vorkommen, wird die Angabe von Datum von der Angabe von Uhrzeit mit einem "T" getrennt.

Der Gebrauch eines beschreibenden Elements DATUM sollte aus seinem Kontext selbstverständlich sein. Das Attribut DT kommt optional in den einführenden Elementen für die Angabe des Entstehungsdatums des Objekts (letzte Änderung, "Versiegelungsdatum") und in beschreibenden Elementen mit einem Typ wie BEZUG, ERSTELLER, GEZ, GZ, AZ, EINGANG, AUSLAUF, FEDER oder PLAN vor. Ein Element BEZUG enthält das Geschäftszeichen des Schriftstücks, auf das ein Dokument sich bezieht, sein DT das Datum dieses Schriftstücks. Ein Element ERSTELLER enthält den Ersteller, sein DT das Erstellungsdatum. Jedes Element GEZ einen Zeichner, DT das Zeichnungsdatum. Ein Element GZ das Geschäftszeichen, ein Element AZ das Aktenzeichen, ihre DT das Registrierungsdatum. Ein Element EINGANG die Eingangsform (EMAIL, FAX, POST, u.s.w), sein DT das Eigangsdatum. Analog ein Element AUSGANG. Jedes Element FEDER den Namen des Federführers, DT das Datum der Übernahme der Federführung. Ein Element PLAN enthält Elemente mit Informationen aus dem Aktenplan, sein DT das Datum des Aktenplans. Ein Element GZ oder AZ kann ein Element REG mit einer Bezeichnung der Registratur enthalten. Ein Element FEDER kann ein Element OE mit einer Bezeichnung der Organisationseinheit enthalten. Während das Attribut DT bei EINGANG sehr wichtig sein kann, manchmal wichtiger als der Inhalt des Elements, ist es in anderen Elementen nicht so bedeutend.

## §5

### Aufbau einer elektronischen Abgabe von Verwaltungsakten

Nicht nur die Primärdateien, sondern auch die Metadatendateien sollten von der Behörde erstellt und im Archiv grundsätzlich nicht zu ändern sein. Das Archiv sollte mit der Behörde verhandeln, um alle Dateien in ihrem endgültigen Format zu bekommen. Das Archiv bekommt eine Tar-Datei, die eine Reihe von Dateien enthält, und hier ist eine laufende Nummerierung erwünscht, die nicht unbedingt mit 1 oder 0 anfangen muß, denn sie könnte von einer vorherigen Abgabe abhängig sein. Wir nehmen als Dateiname diese laufende Nummer zusammen mit einer vom Format abhängigen Erweiterung, beide durch einen Punkt getrennt. Da unter diesen Dateien Metadatendateien sind, die diese Namen in Verweisen enthalten, sind diese Namen auch nicht mehr zu ändern.

Diese Tar-Datei enthält am Anfang eine Metadatendatei für die ganze Abgabe, wie oben beschrieben, mit der Erweiterung ".2" und deswegen hier **2-Datei** gennant; sowie zusätzliche Metadatendateien der Abgabe. Es kommt danach für jeden abgegebenen Aktenband eine Metadatendatei mit der Erweiterung ".1", hier **1-Datei** genannt; sowie zusätzliche Metadatendateien für den Band, gefolgt von den Primärdateien des Bandes, hier **0-Dateien** genannt. Die Erweiterung einer 0-Datei ist von ihrem Format abhängig und muß mit dem Archiv vereinbart werden. Eine 1-Datei enthält Verweise auf 0-Dateien in einer Ordnung, die von der Struktur des Aktenbandes bestimmt ist, und in dieser Ordnung sollten die 0-Dateien in der Tar-Datei vorkommen. Die 2-Datei enthält als Anmerkung die Namen (also laufenden Nummern) der ersten und letzten Datei der Abgabe, sowie andere Informationen (z. B. über physische Unterlagen), um die Vollständigkeit der Abgabe überprüfen zu können.

Die als Dateiname benutzte Nummer einer Datei (ohne die Erweiterung) kann man als Wert für das Attribut "N" im Verweis mit einem Element STUECK benutzen. Nicht nur die Dateien einer Abgabe können laufend nummeriert werden, sondern unabhängig auch die Bände, die Vorgänge und die Dokumente einer Abgabe in der selben Weise. Der Nummer eines Bandes kann man "B" als Präfix zufügen, der Nummer eines Vorgangs "V", der Nummer eines Dokuments "D", um einen Kurznamen zu bilden, der als Wert für das Attribut "N" in Elementen AKTENBAND, VORGANG oder DOKUMENT der 1-Dateien benutzt werden kann.

Ein Element ID1 bei einem Verweis auf eine Datei oder bei einer Metadatendatei (Verweis auf sich selbst) könnte die Zeichenkette "DATEI" enthalten, das Element ID2 eine Zeichenkette, die die Abgabe kennzeichnet, zum Beispiel ein Kürzel des aufnehmenden Archivs, ein Kürzel der abgebenden Behörde, das Jahr der Abgabe mit vier Ziffern und eine laufende Nummer der Abgabe, alle mit "/" getrennt. Sie kann zum Beispiel so aussehen: "BayHStA/MF/2002/1". Eine ähnliche Zeichenkette könnte auch bei LFD im Element ABGABE benutzt werden: zuerst das Kürzel der Behörde, danach das des aufnehmenden Archivs und endlich das Jahr, die laufende Nummer aber als Wert des Attributs N. Die Kennzeichen in FORMAT und im optionalen MIT sollen mit dem Archiv vereinbart werden. Für Standardformate genügt das als Erweiterung benutzte Sufix als Inhalt (z. B. "txt", "tif", "pdf", "ps", "dvi", "htm", "eml").

## §6

### Programme zur Darstellung der Akten

Die 2-Datei gibt einen Überblick über abgegebene Akten, die 1-Dateien beschreiben diese Akten, die 0-Dateien geben ihren Inhalt. Die 1- und 2-Dateien sind ASCII-Dateien, deren Inhalt mit etwas Mühe leicht verständlich ist, ebenfalls der Inhalt der 0-Dateien, wenn man die dem Format entsprechende Software hat. Man will trotzdem Programme haben, die das Lesen dieser Dateien so erleichtern, daß auch ein Archivbenutzer sie lesen kann. Dazu haben wir Prototypen erstellt, hier wollen wir das Grundsätzliche ihres Aufbaus beschreiben. Das Ziel der Prototypen ist, das Entwerfen eines Endproduktes zu fördern und zu erleichtern. Die Technik der Programmierung soll einfach sein, um Änderungen und Verbesserungen schnell vornehmen zu können.

Es geht also um die Darstellung einer Metadatendatei, und je nach Anforderung des Benutzers, der Dateien, auf die sie verweist. Diese Darstellung soll die wichtigsten Metadaten in der Datei anschaulich zeigen. Die Information soll von einem Rechner im Netz angeboten werden, von einem oder mehreren gelesen. Das **Client-Server-HTTP-Protokoll** [8] eignet sich sehr gut. Ein **CGI-Script** auf dem Server kann die angeforderte Information aus den Metadatendateien so aufbereiten, daß der Client sie in einer passenden Form darstellen kann. Die kompliziertere Programmierung von graphischen Oberflächen wird in dieser Weise durch **HTML** und eventuell **Javascript** [8] zunächst umgangen.

Der HTTP-Server muß in der Lage sein, CGI-Scripten [8] aufzurufen; wir benutzen **Apache** 1.3.27 [6] in einem x386-Rechner mit dem Betriebsystem **FreeBSD** [2], einer Implementierung des BSD Standards der Universität von Berkeley. Der größte Vorteil dieses alten Unix-Dialekts gegenüber Linux ist, daß der Standard sich nicht nur auf den Kern des Betriebsystems beschränkt und daß es Implementierungen für andere Rechner gibt. Der HTTP-Client soll in der Lage sein, verschiedene Programme zum Lesen der 0-Dateien aufzurufen, **Netscape** und **Lynx** [7] können es tun; das letzte ist sogar in der Lage, selbst CGI-Scripten in seinem Rechner ohne Mitwirkung eines HTTP-Servers aufzurufen, was vorteilhaft ist, wenn man aus Sicherheitsgründen ein Netz vermeiden will. Die Programmiersprache für die CGI-Scripten muß das Schreiben zu STDOUT und das Lesen von STDIN und der Umgebungsvariablen

erlauben, sowie die Behandlung von SGML-Dateien erleichtern. Wir benutzen **Tcl 8.3** [3], eine sehr einfache, leicht zu erlernende und erweiterbare Programmiersprache, zusammen mit **Cost 2** [4], einer Erweiterung zur Behandlung von SGML-Dateien. Cost braucht einen SGML-Parser, wir benutzen **nsgmls** [5].

Für das Lesen der 0-Dateien, 1-Dateien und der 2-Datei haben wir drei Programme `meta0.cgi`, `meta1.cgi` und `meta2.cgi` im Verzeichnis `cgi-bin` eines Serverrechners mit einer IP-Addresse "`host`". Alle drei lesen den Namen einer Datei, die sich in einem bestimmten Verzeichnis "`archiv`" befindet, aus der Umgebungsvariable `PATH_INFO`, und die ersten zwei lesen eine Zeichenkette aus der Umgebungsvariable `QUERY_STRING`. Alle drei schreiben in `STDOUT` das, was der Client zeigen soll. Wie für CGI-Scripten vorgeschrieben, schreiben sie zuerst eine Zeile mit der **MIME**-Bezeichnung eines Formats, danach eine leere Zeile und danach die vom Programm erzeugte Information. Wenn der Client eine Tabelle verwaltet, die MIME-Formatbezeichnungen entsprechende Programme zum Lesen der Formate zuordnet (bei Lynx ist das eine **mailcap**-Datei), dann zeigt der Client die Information mit dem richtigen Programm. Die von `meta1.cgi` und `meta2.cgi` erzeugte Information ist in HTML kodiert.

Das Programm `meta0.cgi` zum Lesen von 0-Dateien ruft man aus dem Client mit einem URL folgender Form auf:

```
http://host/cgi-bin/meta0.cgi/datei?f=format
```

Hier ist "`datei`" der Name einer Datei im Format "`format`", verschlüsselt wie in einem Element `FORMAT` von `STUECK`. Das Programm liest "`datei`" aus der Umgebungsvariable `PATH_INFO` und alles, was nach "`?`" kommt, aus der Umgebungsvariable `QUERY_STRING`. Es übersetzt "`format`" mit Hilfe einer Tabelle in eine MIME-Bezeichnung des Formats, schreibt diese Bezeichnung in `STDOUT`, danach eine neue Zeile und danach den Inhalt der Datei "`datei`".

Das Programm `meta1.cgi` zum Lesen von 1-Dateien ruft man aus dem Client mit einem URL folgender Form auf:

```
http://host/cgi-bin/meta1.cgi/datei.1?pfad
```

Hier ist "`datei.1`" der Name einer 1-Datei und "`pfad`" eine Zeichenkette, die ein in dieser Datei definiertes Objekt bezeichnet, es kann folgende Formen haben:

```
b=BN
b=BN&s=N
b=BN&v=VN
b=BN&v=VN&s=N
b=BN&v=VN&d=DN
b=BN&v=VN&d=DN&s=N
```

Hier ist `BN` der Kurzname des Bandes, `VN` des Vorgangs, `DN` des Dokuments und `N` des Verweises, wie er im Attribut `N` des Elementes vorkommt, das das Objekt einführt. Die Kurznamen `BN`, `VN`, `DN`, `N` dürfen weggelassen werden, wenn die Eindeutigkeit der Bezeichnung darunter nicht leidet. Zum Beispiel "`b=`" genügt zur Bezeichnung des einzigen eingeführten Bandes in der Datei. Das Programm erzeugt mit dem Element in der Datei, das das

bezeichnete Objekt einführt, eine HTML-Darstellung des Objekts mit Verweisen auf seine Bestandteile, die Aufrufe von `meta0.cgi` und `meta1.cgi` anbieten. Es wird auch `datei.1` mit "`meta0.cgi/datei.1?f=txt`" angeboten. Das Programm `meta1.cgi` schreibt nach der Angabe der Mime-Formatbezeichnung und der neuen Zeile die HTML-Darstellung des Objekts. Der Client wird dann diese Darstellung formatiert zeigen.

Das Programm `meta2.cgi` zum Lesen von 2-Dateien ruft man aus dem Client mit einem URL folgender Form auf:

```
http://host/cgi-bin/meta2.cgi/datei.2
```

Hier ist "`datei.2`" der Name einer 2-Datei. Aus ihr erzeugt das Programm eine HTML-Darstellung der Abgabe mit Verweisen auf die abgegebenen Aktenbände mit `meta1.cgi`. Ebenfalls wird "`meta0.cgi/datei.2?f=txt`" angeboten.

## §7

### DTD für Metadatendateien abgegebener Verwaltungsakten

Die von unserem Prototyp zur Zeit benutzte DTD für 2-Dateien zur Beschreibung einer Abgabe sieht wie folgt aus:

```
<!ELEMENT TXT O O (#PCDATA)>
<!ELEMENT TYP - O (#PCDATA)>
<!ELEMENT (ID | ID1 | ID2) - O (#PCDATA)>
<!ELEMENT GROESSE - O (#PCDATA)>
<!ELEMENT ANM - O (#PCDATA)>
<!ELEMENT LFD - O (#PCDATA)>
 <!ATTLIST LFD
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT VRG - O (#PCDATA)>
 <!ATTLIST VRG
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT (VON | BIS | MIT | AUF) - O (#PCDATA)>
<!ELEMENT DATUM - O  (#PCDATA, VON?, BIS?)>
<!ELEMENT AZ  - O (#PCDATA, REG?)>
<!ELEMENT REG - O (#PCDATA)>
<!ELEMENT APZ - O (#PCDATA)>
<!ELEMENT STUECK    - -
 (TXT, TYP?,
  ID, ID1?, ID2?,
  GROESSE?, FORMAT+, ANM*)>
 <!ATTLIST STUECK
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ZR NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT FORMAT - O  (#PCDATA, MIT?)>
<!ELEMENT AKTENSEGMENT - -
 (TXT, TYP?,
  LFD, VRG?, AZ, DATUM, AUF?, ANM*,
  STUECK*)>
```

```
 <!ATTLIST AKTENSEGMENT
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ZR NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT GRUPPE - -
 (TXT, TYP?,
  APZ, DATUM, GROESSE?, ANM*,
  (STUECK | AKTENSEGMENT | GRUPPE)*)>
 <!ATTLIST GRUPPE
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ZR NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT ABGABE - -
 (TXT, TYP?,
  ID, ID1?, ID2?, FORMAT,
  BEHOERDE, ARCHIV, DATUM, LFD, VRG?, ANM*,
  (STUECK | AKTENSEGMENT | GRUPPE)*)>
 <!ATTLIST ABGABE
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ZR NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT BEHOERDE - O (#PCDATA)>
<!ELEMENT ARCHIV   - O (#PCDATA)>
<!ENTITY Ae "&#196;">
<!ENTITY Oe "&#214;">
<!ENTITY Ue "&#220;">
<!ENTITY ae "&#228;">
<!ENTITY oe "&#246;">
<!ENTITY ue "&#252;">
<!ENTITY sz "&#223;">
```

Die von unserem Prototyp zur Zeit benutzte DTD für 1-Dateien zur Beschreibung eines Aktenbandes sieht wie folgt aus:

```
<!ELEMENT (VON | BIS | MIT | UNTER) - O (#PCDATA)>
<!ELEMENT TXT O O (#PCDATA)>
<!ELEMENT TYP - O (#PCDATA)>
<!ELEMENT (ID | ID1 | ID2) - O (#PCDATA)>
<!ELEMENT GROESSE - O (#PCDATA)>
<!ELEMENT ANM - O (#PCDATA)>
<!ELEMENT BETREFF - O  (#PCDATA)>
<!ELEMENT BEZUG   - O  (#PCDATA)>
 <!ATTLIST BEZUG  DT NMTOKEN #IMPLIED>
<!ELEMENT ERSTELLER   - O  (#PCDATA)>
 <!ATTLIST ERSTELLER  DT NMTOKEN #IMPLIED>
<!ELEMENT DATUM - O  (#PCDATA, VON?, BIS?)>
<!ELEMENT GEZ   - O  (#PCDATA)>
 <!ATTLIST GEZ  DT NMTOKEN #IMPLIED>
<!ELEMENT GZ  - O (#PCDATA, REG?)>
 <!ATTLIST GZ DT NMTOKEN #IMPLIED>
<!ELEMENT AZ  - O (#PCDATA, REG?)>
 <!ATTLIST AZ  DT NMTOKEN #IMPLIED>
<!ELEMENT REG - O (#PCDATA)>
<!ELEMENT APZ - O (#PCDATA)>
```

```
<!ELEMENT LFD - O (#PCDATA)>
 <!ATTLIST LFD
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT VRG - O (#PCDATA)>
 <!ATTLIST VRG
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT FGZ        - O  (#PCDATA)>
<!ELEMENT ABSENDER   - O  (#PCDATA)>
<!ELEMENT ADRESSAT   - O  (#PCDATA)>
<!ELEMENT EINGANG    - O  (#PCDATA)>
 <!ATTLIST EINGANG  DT NMTOKEN #IMPLIED>
<!ELEMENT AUSGANG    - O  (#PCDATA)>
 <!ATTLIST AUSGANG  DT NMTOKEN #IMPLIED>
<!ELEMENT BEHOERDE - O  (#PCDATA)>
<!ELEMENT FEDER    - O  (#PCDATA, OE?)>
 <!ATTLIST FEDER   DT NMTOKEN #IMPLIED>
<!ELEMENT OE       - O (#PCDATA)>
<!ELEMENT STUECK    - -
 (TXT, TYP?,
  ID, ID1?, ID2?,
  GROESSE?, FORMAT+, ANM*)>
 <!ATTLIST STUECK
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ZR NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT FORMAT - O  (#PCDATA, MIT?)>
<!ELEMENT DOKUMENT   - -
 (TXT, TYP,
  BETREFF?, BEZUG?, ERSTELLER?, DATUM?, GEZ*, GZ?,
 (FGZ?, ABSENDER, ADRESSAT, EINGANG?, AUSGANG?)?, ANM*,
  STUECK*)>
 <!ATTLIST DOKUMENT
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ZR NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT VORGANG   - -
 (TXT, TYP?,
  BETREFF?, FEDER*, DATUM?, GZ*, ANM*,
  (STUECK | DOKUMENT)*)>
 <!ATTLIST VORGANG
  N  NMTOKEN #IMPLIED
  DT NMTOKEN #IMPLIED
  ZR NMTOKEN #IMPLIED
  ST NMTOKEN #IMPLIED>
<!ELEMENT AKTENBAND   - -
  (TXT, TYP?,
   ID, ID1?, ID2?, FORMAT,
   BEHOERDE, OE?,
   DATUM?, AZ*, LFD?, VRG?,
   PLAN*,
   ANM*,
   (STUECK | VORGANG)*)>
 <!ATTLIST AKTENBAND
```

```
    N  NMTOKEN #REQUIRED
    DT NMTOKEN #IMPLIED
    ZR NMTOKEN #IMPLIED
    ST NMTOKEN #IMPLIED>
  <!ELEMENT PLAN - O
    (TXT?, APZ, BETREFF, UNTER*)>
   <!ATTLIST PLAN DT NMTOKEN #IMPLIED>
```

## Literatur


[1] A Gentle Introduction to SGML (http://www.tei-c.org/P4beta/SG.htm).

[2] The FreeBSD Project (http://www.freebsd.org).

[3] Tcl Developer Site (http://www.tcl.tk).

[4] Cost Home Page (http://www.flightlab.com/cost).

[5] SP (http://www.jclark.com/sp).

[6] The Apache Software Foundation (http://www.apache.org).

[7] Lynx Information (http://lynx.browser.org).

[8] Web Technology Made Really Easy (http://www.jmarshall.com/easy),
JavaScript Ref. (http://developer.netscape.com/docs/manuals/communicator/jsref/index.htm),
Internet Related Technologies (http://www.irt.org/).