KOST.07
Archivierung von Unterlagen aus Geschäftsverwaltungssystemen (AUGev)

Analyse von METS, XFDU und EAD als zentrale Metadatenschemata für AIPs

Inhalt

1 Aufgabestellung und Bewertungsschema 2
2 Metadaten nach OAIS 3
3 METS als AIP-Spezifikation 4
3.1 Aufbau des AIP 5
3.2 Beurteilung 5
3.2.1 Logische Struktur 5
3.2.2 Metadaten nach ISAD(G) 6
3.2.3 Datenmigration 6
3.2.4 Zusammenfassung 6
4 XFDU als AIP-Spezifikation 7
4.1 Aufbau des AIP 8
4.2 Beurteilung 9
4.2.1 Logische Struktur 9
4.2.2 Metadaten nach ISAD(G) 10
4.2.3 Datenmigration 10
4.2.4 Zusammenfassung 11
5 EAD als AIP-Spezifikation 12
5.1 Aufbau des AIP 13
5.2 Beurteilung 14
5.2.1 Logische Struktur 14
5.2.2 Metadaten nach ISAD(G) 15
5.2.3 Datenmigration 15
5.2.4 Zusammenfassung 15
6 Anhang: Beispieldaten 16

1 Aufgabestellung und Bewertungsschema

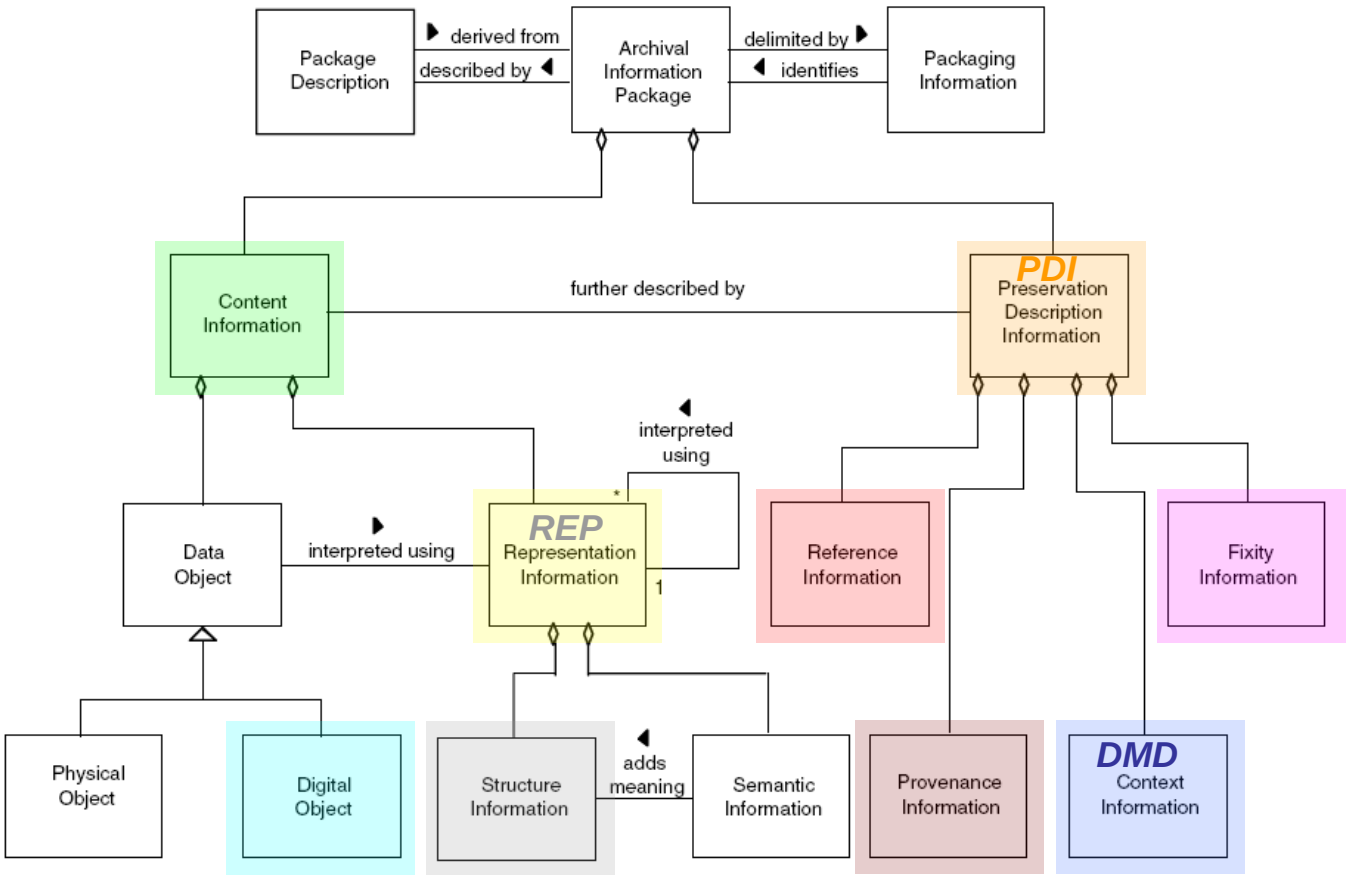
In den verschiedenen AIP-Lösungen, die das AUGev-Projektteam analysiert hat, wird eine Vielzahl von Metadatenstandards verwendet. Diese haben unterschiedliche Funktionen. Neben Standards für deskriptive (Dublin Core, EAD) und für technische Metadaten (LMER, PREMIS; siehe dazu "Vergleich LMER-PREMIS", *Vergleich_LMER-PREMIS.pdf*) werden auch Standards für die Verzeichnung der Ordnungssystemhierarchie und die Beziehungen zwischen den Primärdaten und den einzelnen Metadatenabschnitten benötigt. In den meisten der diskutierten AIP-Varianten wird dafür METS verwendet. Mit dem XFDU-Standard liegt neuerdings ein Konkurrenzentwurf dazu vor. In der KOST-Referenzimplementierung wird auf die Verwendung von METS verzichtet und werden seine Funktionen durch EAD (und teilweise durch JAR) wahrgenommen.

Das vorliegende Dokument vergleicht die drei Standards METS, XFDU und EAD miteinander in Hinblick auf ihre Verwendung in einem AIP. Die Standards werden einzeln vorgestellt und ihr Aufbau analysiert. Speziell beurteilt werden ihre logische Struktur, die Möglichkeit zur Einbettung deskriptiver Metadaten (insbesondere des minimalen Sets der sechs zwingend notwendigen Verzeichnungselemente nach ISAD[G], siehe "Deskriptive Metadaten nach ISAD[G]", *ISAD(G)-Metadaten.pdf*) und die Abbildung von Formatmigrationen. Als gemeinsame Referenz dient das Open Archival Information System OAIS[1].

[1] Consultative Committee on Space Data Systems, Reference Model for an Open Archival Information System (OAIS), CCSDS 650.0-B-1, Blue Book, January 2002. http://public.ccsds.org/publications/archive/650x0b1.pdf

2 Metadaten nach OAIS

Das OAIS bietet ein abstraktes Modell der verschiedenen Metadatenkategorien in einem AIP. Zum besseren Verständnis sind die Metadaten nach dem OAIS-Modell, Figure 4-18: *Archival Information Package (Detailed View)*[2] farblich gegliedert, damit wir sie in denselben Farben in den konkreten Metadaten-Schemata wieder finden können. *REP, DMD*, und *PDI* entstammen der METS-Syntax, werden aber so auch in XFDU verwendet. Zu beachten ist, dass diese Begriffe nicht mit den entsprechenden Metadatengruppierungen in OAIS deckungsgleich sind.

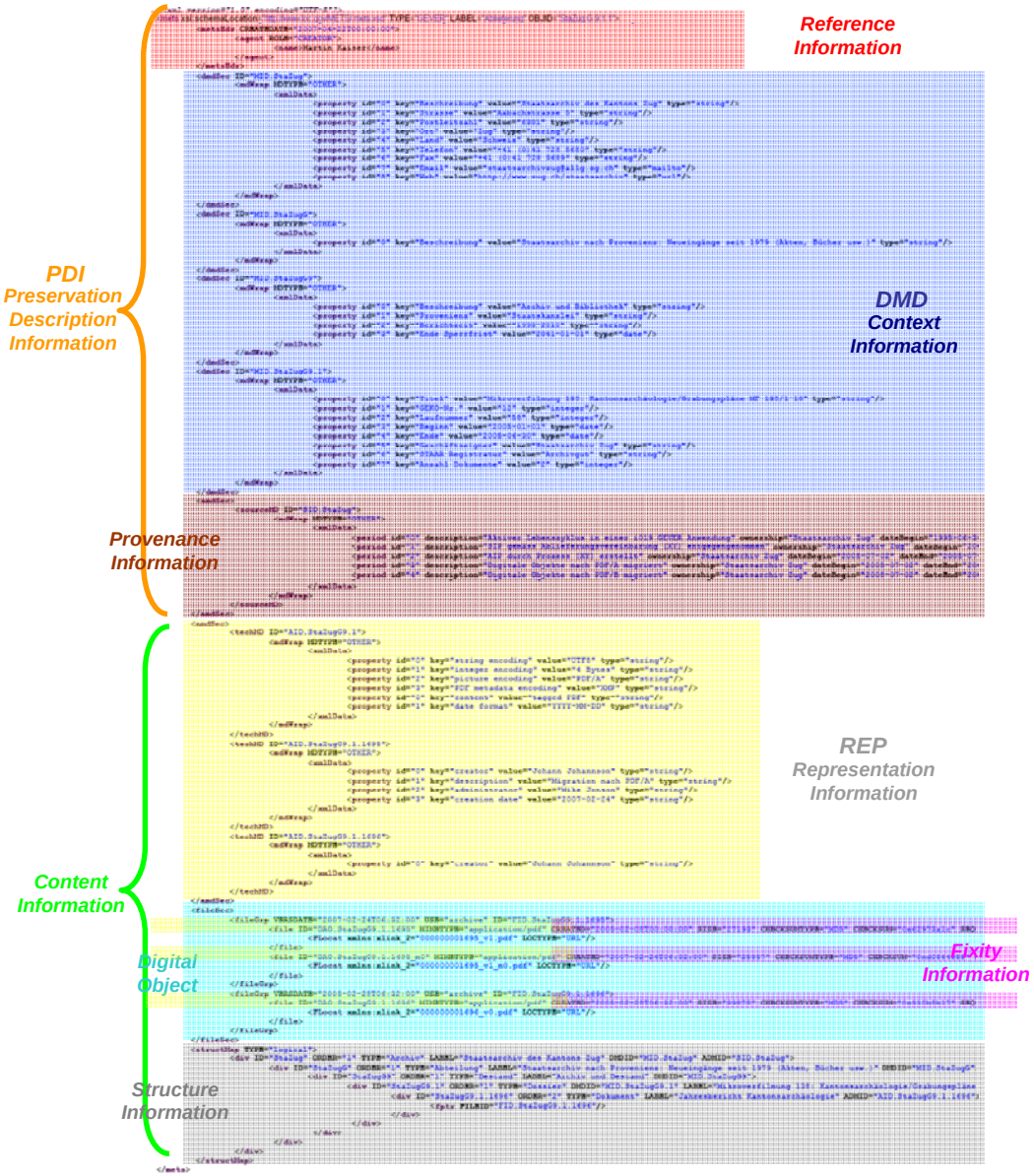


Archival Information Package (Detailed View)

[2] A.a.O., Seite 4-37.

3 METS als AIP-Spezifikation

METS (Metadata Encoding and Transfer Standard)[3] ist entstanden aus der DTD für die digitalen Objekte im "Making of America II"-Projekt (MOA2)[4]. Man stellt fest, dass die Struktur im Wesentlichen die gleiche ist: Eine *Structural Map bildet* die hierarchische Struktur des Objekts ab; dazu kommen eine *File Section* sowie je *Sections* für administrative und deskriptive Metadaten. *File Section (fileSec)*, beschreibende Metadaten (*dmdSec*) und administrative Metadaten (*amdSec*) werden aus der *Structural Map (structMap)* referenziert. Das bedeutet, das Grundprinzip von METS, eine hierarchische Struktur in XML abzubilden, aber den Rest der Metadaten nicht dort zu integrieren, stammt aus MOA2.

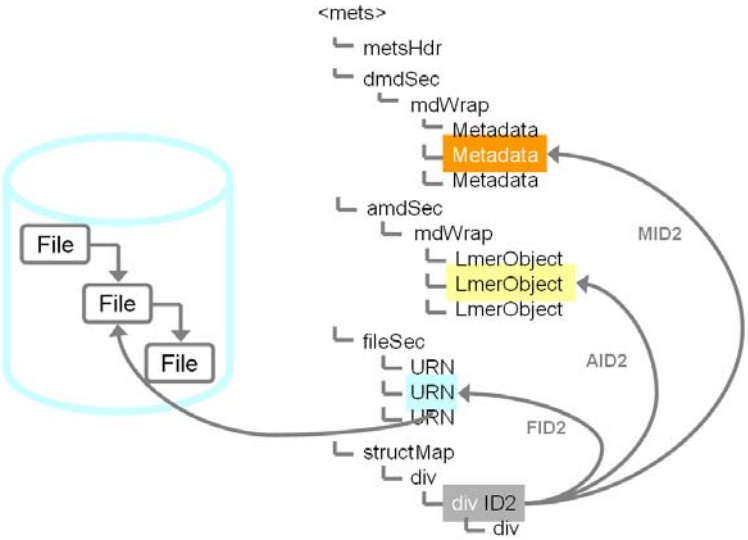


Testdaten-AIP in METS-Schema

[3] http://www.loc.gov/standards/mets/.
[4] http://sunsite.berkeley.edu/moa2/; http://sunsite.berkeley.edu/moa2/papers/moa2dtd2.htm.

3.1 Aufbau des AIP

Die zentrale *structMap* beschreibt die (hierarchische) Struktur der Daten durch Schachtelung von *div*-Verzeichungseinheiten. In der *structMap* werden zudem Verzeichnungsebene *(TYPE)* und Verzeichnungstitel *(LABEL)* festgehalten. Verweise zeigen auf eine *Administrative Metadata Section (amdSec)* und eine *Descriptive Metadata Section (dmdSec).* Eine präzise Abbildung auf das OAIS-Modell ist nicht möglich: *dmdSec* und *amdSec* sind nur Container für spezifische, dem Archivtyp angepasste Metadatenschemata. In dem von uns betrachteten Archivbereich bieten sich für die beschreibenden Metadaten in *dmdSec Dublin Core* oder EAD als Metadatenstandard an, für die technischen Metadaten in *amdSec* stehen PREMIS oder LMER zur Auswahl.



METS: Referenzierungen mit ID, IDREF und IDREFS

3.2 Beurteilung

3.2.1 Logische Struktur

Die Aufteilung in Struktur, Dateiverweis und Metadaten ist nicht so konsequent, wie es im ersten Augenblick erscheint. Eine gewisse Menge beschreibende und technische Metadaten sind bereits in *structMap (TYPE, LABEL)*, und in *fileSec (MIMETYPE, SIZE, CHECKSUM etc.)* definiert. Das Set der definierten Metadaten für technische Metadaten in *fileSec* ist recht umfangreich und lässt den Einsatz eines zusätzlichen Metadatenschemas in diesem Bereich nicht unbedingt als zwingend erscheinen.

```
<fileSec>
        <fileGrp VERSDATE="2007-02-24T06:32:00" USE="archive" ID="FID.StaZugG9.1.1695">
                <file ID="DA0.StaZugG9.1.1695" MIMETYPE="application/pdf" CREATED="2005-02-08T00:00:00" SIZE="27193" CHECKSUMTYPE="MD5" CHECKSUM="0x62973a1c" SEQ
                        <FLocat xmlns:xlink_2="000000001695_v1.pdf" LOCTYPE="URL"/>
                </file>
                <file ID="DA0.StaZugG9.1.1695_m0" MIMETYPE="application/pdf" CREATED="2007-02-24T06:32:00" SIZE="25857" CHECKSUMTYPE="MD5" CHECKSUM="0xd0864558" 
                        <FLocat xmlns:xlink_2="000000001695_v1_m0.pdf" LOCTYPE="URL"/>
                </file>
        </fileGrp>
        <fileGrp VERSDATE="2005-02-28T06:32:00" USE="archive" ID="FID.StaZugG9.1.1696">
                <file ID="DA0.StaZugG9.1.1696" MIMETYPE="application/pdf" CREATED="2005-02-28T06:32:00" SIZE="39875" CHECKSUMTYPE="MD5" CHECKSUM="0x910b5b17" SEQ
                        <FLocat xmlns:xlink_2="000000001696_v0.pdf" LOCTYPE="URL"/>
                </file>
        </fileGrp>
</fileSec>
```

Technische Metadaten in fileSec

Die in METS definierten beschreibenden Metadaten sind jedoch eher bescheiden; ohne zusätzliches Metadatenschema kann eine minimale ISAD(G)-kompatible Abbildung nicht erreicht werden (siehe unten 3.2.2).

Die Metadatencontainer *amdSec, dmdSec, fileSec* sind über Referenzen mit der logischen Datenstruktur in *structMap* verbunden. Der Vorteil dabei ist, dass keine Redun-

danzen bei der Metadatenspeicherung entstehen oder Redundanzen vermieden werden können. Der Nachteil ist die schlechtere XML-"Lesbarkeit".

3.2.2 Metadaten nach ISAD(G)

Die aus archivischer Sicht notwendigen Metadaten können nicht direkt im METS-Schema erfasst werden. Es muss mindestens ein weiteres Metadatenschema (z.B. *Dublin Core*) eingesetzt werden.

	Verzeichnungselement ISAD(G)	METS
1.1	Signatur	ID
1.2	Titel	LABEL
1.3	Entstehungszeitraum / Laufzeit	?
1.4	Verzeichnungsstufe	Position von DIV bzw. ORDER
1.5	Umfang (Menge und Abmessung)	?
2.1	Name der Provenienzstelle	?

ISAD(G)-Verzeichnungselemente und ihre Abbildung in METS

```
<structMap TYPE="logical">
        <div ID="StaZug" ORDER="1" TYPE="Archiv" LABEL="Staatsarchiv des Kantons Zug" DMDID="MID.StaZug" ADMID="SID.StaZug">
                <div ID="StaZugG" ORDER="1" TYPE="Abteilung" LABEL="Staatsarchiv nach Provenienz: Neueingänge seit 1979 (Akten, Bücher usw.)" DMDID="MID.StaZugG":
                        <div ID="StaZugG9" ORDER="1" TYPE="Bestand" LABEL="Archiv und Bestand" DMDID="MID.StaZugG9">
                                <div ID="StaZugG9.1" ORDER="1" TYPE="Dossier" DMDID="MID.StaZugG9.1" LABEL="Mikroverfilmung 138: Kantonsarchäologie/Grabungspläne

                                        <div ID="StaZugG9.1.1696" ORDER="2" TYPE="Dokument" LABEL="Jahresbericht Kantonsarchäologie" ADMID="AID.StaZugG9.1.1696">
                                                <fptr FILEID="FID.StaZugG9.1.1696"/>
                                        </div>
                                </div>
                        </div>
                </div>
        </div>
</structMap>
```

Beschreibende Metadaten in structMap

3.2.3 Datenmigration

Die technischen Metadaten in *fileSec* erlauben keine Abbildung einer Datenmigration. Nur durch geeignete Metadaten in *amdSec* kann das Problem gelöst werden.

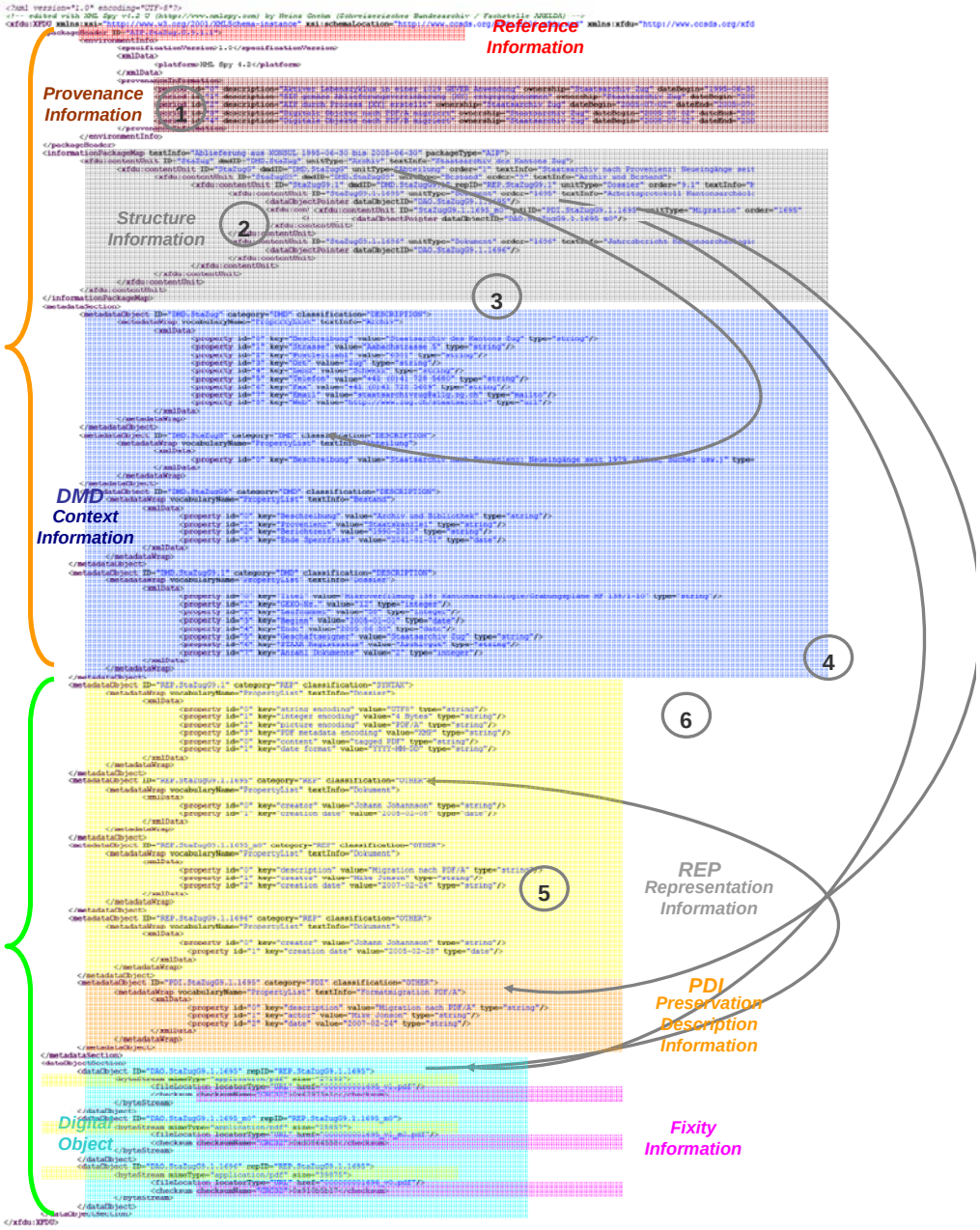
3.2.4 Zusammenfassung

Ein Mangel von METS ist die fehlende Abbildung eines minimalen Metadatensets nach ISAD(G) innerhalb der eigentlichen Schemastruktur. Die Wrapper-Lösung ist zwar äusserst flexibel, erhöht aber die Komplexität und führt zu unterschiedlichen AIPs nach dem METS-Standard, die nicht mehr mit der einheitlichen und einfachen Softwarelösung handhabbar sind.

Die Referenzierung von Struktur zu Metadaten entspricht eher einem relationalen Datenansatz als einem hierarchischen XML-Schema

4 XFDU als AIP-Spezifikation

XFDU (*XML Formatted Data Unit*) wird vom *Consultative Committee for Space Data Systems* entwickelt[5]. Ein Beispiel zeigt, wie unsere Testdaten nach dem XFDU-Schema strukturiert aussehen. Die Word-Dateien sind in diesem Falle nicht übernommen worden. Die drei PDF-Dateien sind extern referenziert und befinden sich im gleichen Datencontainer (z.B. ZIP-Datei) wie die XFDU-Manifest-Datei.

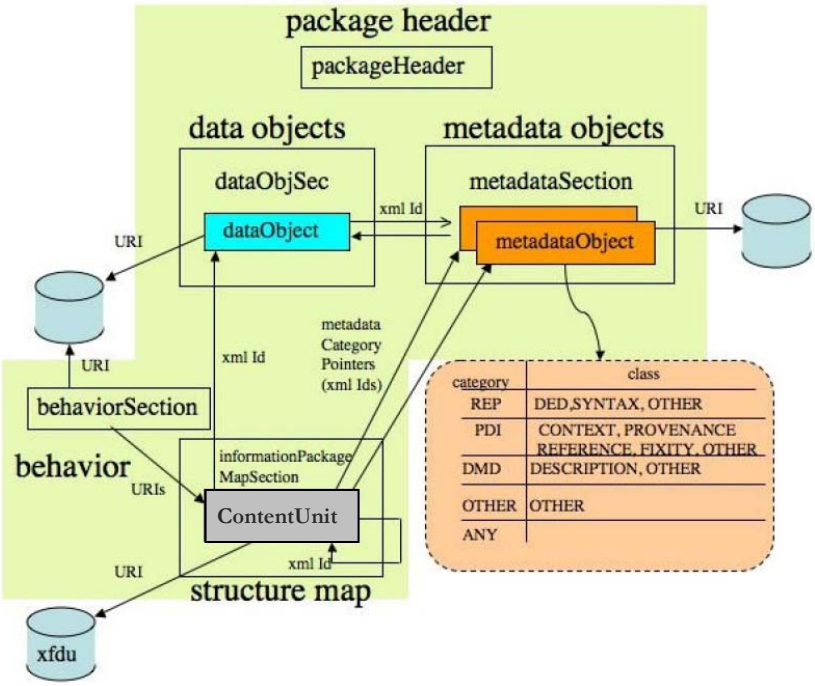


Testdaten-AIP nach XFDU-Schema

[5] Consultative Committee on Space Data Systems, XML Formatted Data Unit (XFDU) Structure and Construction Rules Draft Recommended Standard CCSDS 661.0-R-1 (Blue Book) Issue 1 January 2008. http://sindbad.gsfc.nasa.gov/xfdu/pdfdocs/xfduspec.pdf.

4.1 Aufbau des AIP

Im Zentrum einer XFDU-Datei steht die *informationPackageMap* mit rekursiv verzeichneten *contentUnits* (*Structure Information* nach OAIS). Aus den *contentUnits* wird mit XML-Referenzen auf *dataOjects* (*Digital Objects*) und auf *metadataObjects* (*Preservation Description Information*) verwiesen. Das *dataObject* verweist dann auf die externen Dateien. Ein *dataObject* kann auch auf *metadataObjects* verweisen, aber nur vom Typ *REP (Representation Information).*



XFDU Manifest Logical View[6]

Betrachten wir die Struktur noch einmal Punkt für Punkt:

1. *Provenance information* ist im *packageHeader* in der *environmentInfo* abgelegt[7].
2. Die *informationPackageMap* beschreibt die logische Struktur der Daten als Baum oder Liste. Knoten und Blätter sind dabei die *contentUnit*s, logische Einheiten für Verzeichnisse oder Dokumente.
3. Jede *contentUnit* kann auf eine oder mehrere *DMD (Descriptive Metadata)*, bzw. *Context Information* verweisen. Ein *DMD*-Eintrag kann auch mehrfach referenziert werden. Hier werden die Metadaten zur Struktur abgelegt.
4. Eine *contentUnit* kann auf ein *dataObject* verweisen. Ein *dataObject* ist ein Container für eine Datei. Dieser Container enthält bereits gewisse *Fixity Information* (*mimeType, size, checksum*). Die Datei kann extern verlinkt sein oder in der XML-Datei selber liegen (*fileLocation*).
5. Ein *dataObject* kann auf ein *metadataObject* vom Typ *REP (Representation Information)* und nur von diesem Typ verweisen. Dort können weitere Metadaten zur Datei in einem XML-Wrapper (*xmlData*) zum Beispiel im LMER-Format abgelegt werden.

---

[6] A.a.O., Figure 2-2.
[7] Diese Lösung ist im XFDU-Papier nicht weiter ausgeführt.

6. Eine *contentUnit* kann auch auf ein *metadataObject* vom Typ *PDI* (*Preservation Description Information*) verweisen. Dort können weitere Informationen zu *Reference, Context, Provenance* und *Fixity* abgelegt werden.

Die *metadataSection* mit den *metadataObjects* vom Typ *DMD, REP* und *PDI* ist eigentlich ein Container für Metadaten in beliebiger Struktur innerhalb eines *xmlData* Elementes. Im Beispiel ist dafür eine einfache *Key-Value*-Paar-Darstellung ohne Schema gewählt. Nur relativ wenige Informationen können oder müssen in vom Schema festgelegter Form gespeichert werden, nämlich

– *contentUnit.ID, unitType, order*

– *dataObject.ID, mimeType, size, checksum, fileLocation, locatorTyp, href*

– *metadataOject, category, classification*

Alle weiteren Informationen sind vom Typ *textInfo* (xsd:string).

Gänzlich fehlen Datumsangaben (xsd:date).

4.2 Beurteilung

4.2.1 Logische Struktur

Metadaten sind über Referenzen mit der logischen (in unserem Falle hierarchischen) Datenstruktur verbunden. Der Vorteil dabei ist, dass bei der Metadatenspeicherung Redundanzen vermieden werden können. Der Nachteil ist die schlechtere XML-"Lesbarkeit", denn mit IDREF werden eigentlich relationale Datenbeziehungen dargestellt.

Zwei Beispiele mit den gleichen Daten:

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <XFDU>
  - <informationPackageMap>
    - <contentUnit ID="StaZug" unitType="Archiv" unitTitle="Staatsarchiv des Kantons Zug">
        <contentUnit ID="StaZugG" unitType="Abteilung" unitTitle="Neueingänge seit 1979" />
      </contentUnit>
    </informationPackageMap>
  </XFDU>
```

Metadaten in hierarchischer Baumstruktur

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <XFDU>
  - <informationPackageMap>
    - <contentUnit ID="StaZug" dmdID="DMD.StaZug">
        <contentUnit ID="StaZugG" dmdID="DMD.StaZugG" />
      </contentUnit>
    </informationPackageMap>
  - <metadataSection>
      <metadataObject ID="DMD.StaZug" category="DMD" unitType="Archiv" unitTitle="Staatsarchiv des Kantons Zug" />
      <metadataObject ID="DMD.StaZugG" category="DMD" unitType="Abteilung" unitTitle="Neueingänge seit 1979" />
    </metadataSection>
  </XFDU>
```

Metadaten referenziert

Ein Beispiel mit Redundanz (*access*):

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <XFDU>
  - <informationPackageMap>
    - <contentUnit ID="StaZug" unitType="Archiv" unitTitle="Staatsarchiv des Kantons Zug" access="public">
        <contentUnit ID="StaZugG" unitType="Abteilung" unitTitle="Neueingänge seit 1979" access="public" />
      </contentUnit>
    </informationPackageMap>
  </XFDU>
```

Metadaten redundant

4.2.2 Metadaten nach ISAD(G)

Die aus archivischer Sicht notwendigen Metadaten können nicht in schemakontrollierter Form erfasst werden.

	Verzeichnungselement ISAD(G)	XFDU
1.1	Signatur	contentUnit.ID
1.2	Titel	? (contentUnit.textInfo)
1.3	Entstehungszeitraum / Laufzeit	?
1.4	Verzeichnungsstufe	contentUnit.unitType
1.5	Umfang (Menge und Abmessung)	dataObject.size und checksum
2.1	Name der Provenienzstelle	?

ISAD(G)-Verzeichnungselemente und ihre Abbildung in XFDU

Damit ist die spätere maschinelle Verarbeitung eines XFDU-AIPs stark eingeschränkt, müssen doch zum Schema noch weitere Informationen zur Syntax und Semantik einer Ablieferung festgehalten werden. Auch dem Ablieferer eines entsprechenden SIPs muss neben dem Schema eine Reihe von weiteren Prozessanweisungen mitgegeben werden, damit er syntaktisch einheitliche AIPs erzeugen kann.

Beispielsweise lässt sich die Bedeutung von "2005-02-08" nur wie folgt entschlüsseln:

```
- <metadataObject ID="REP.StaZugG9.1.1695" category="REP" classification="OTHER">
  - <metadataWrap vocabularyName="PropertyList" textInfo="Dokument">
    - <xmlData>
        <property id="0" key="creator" value="Johann Johannson" type="string" />
        <property id="1" key="creation date" value="2005-02-08" type="date" />
      </xmlData>
    </metadataWrap>
  </metadataObject>
```

Metadaten in einem Wrapper

Das Dokument mit der *contentUnit@ID="StaZug9.1.1695"* und der URL *@href="000000001695_v1.pdf"* verweist auf den Metadateneintrag *metadataObjekt@ID="REP.StaZug9.1.1695"* mit dem Inhalt *metadataObjekt/metadataWrap/xmlData/property@value="2005-02-08"* und dem *@key="creation date"*. D.h. 2005-02-08 ist das Erzeugungsdatum der Datei 000000001695_v1.pdf. Dieser ganze Zusammenhang lässt sich nicht aus dem Schema erkennen; es fehlt hier ein Attribut *creationDate* auf Ebene *byteStream*.

4.2.3 Datenmigration

Die Datenmigration wird nicht wirklich unterstützt. Für eine migrierte Datei muss eine neue *contentUnit* angelegt werden, obwohl eigentlich auf Ebene *dataObject* oder sogar *byteStream* die migrierte Datei verzeichnet werden sollte. Die Migrationsinformationen liegen umständlich referenziert in einem *metadataObject* vom Typ PDI:

```
- <metadataObject ID="PDI.StaZugG9.1.1695" category="PDI" classification="OTHER">
  - <metadataWrap vocabularyName="PropertyList" textInfo="Formatmigration PDF/A">
    - <xmlData>
        <property id="0" key="description" value="Migration nach PDF/A" type="string" />
        <property id="1" key="actor" value="Mike Jonson" type="string" />
        <property id="2" key="date" value="2007-02-24" type="string" />
      </xmlData>
    </metadataWrap>
  </metadataObject>
```

Beispiel der Verzeichnung einer Migration

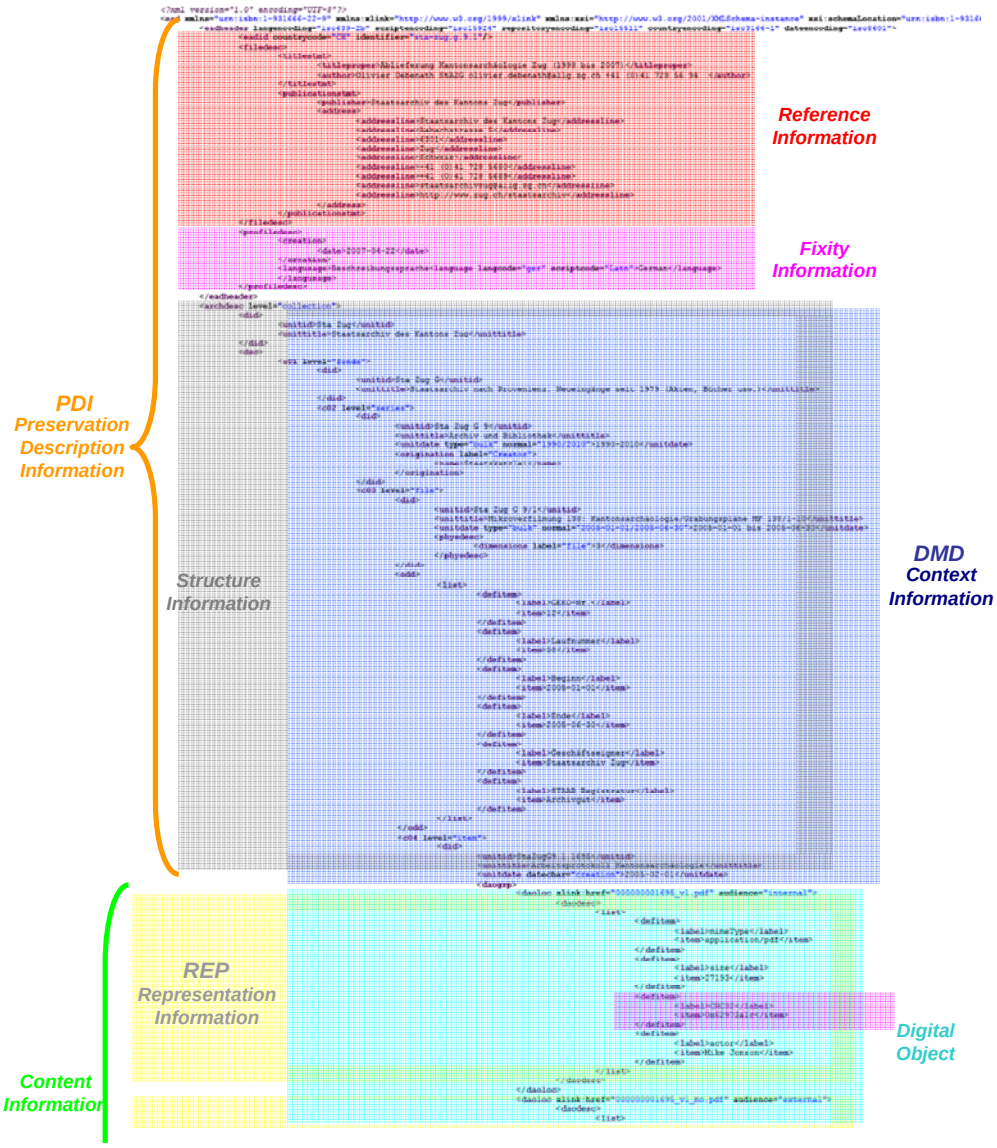
4.2.4 Zusammenfassung

Ein schwerwiegender Mangel von XFDU ist die fehlende Abbildung eines minimalen Metadatensets nach ISAD(G) innerhalb der Schemastruktur. Die Wrapper-Lösung ist zwar äusserst flexibel, aber langfristig ein Problem bei der maschinellen Interpretierbarkeit der Metadaten.

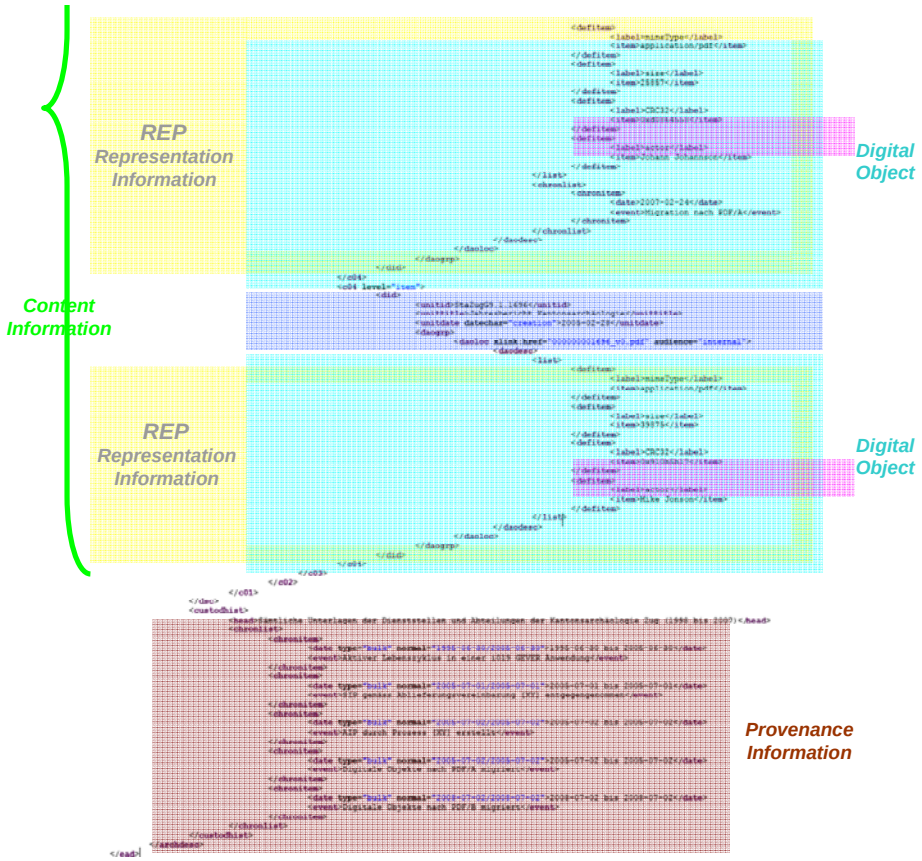
Unschön ist die schlechte Integration von Dateiversionen (Migration) in das Datenobjekt. Auch hier ist zusätzliches Prozesswissen notwendig, das nicht im Schema gespeichert werden kann.

5 EAD als AIP-Spezifikation

Encoded Archival Description (EAD) ist ein dokumentarischer XML-Standard zur Beschreibung von Findbüchern und anderen Findhilfen in Archiven, Museen und Bibliotheken. EAD ist also von seiner Entstehung her kein Containerformat, eignet sich aber hervorragend zur Abbildung von Kontext-Informationen in archivischem Zusammenhang, weil es eigentlich eine XML-Schema-Repräsentation der ISAD(G)-Vorgaben darstellt. Da EAD Version 2002 aber ein Element <dao> *Digital Archival Object* besitzt ("A linking element … to connect the finding aid information to electronic representations of the described materials")[8], drängt sich die Überlegung auf, EAD auch insgesamt als AIP zu verwenden.



[8] Encoded Archival Description Tag Library, Version 2002, EAD Element <dao> Digital Archival Object: http://www.loc.gov/ead/tglib/elements/dao.html.



Testdaten-AIP in EAD Schema

5.1 Aufbau des AIP

Eine EAD-Datei besteht aus den drei Teilen *eadheader*, *arcdesc* und *custodhist*. Der mittlere Teil (*arcdesc)* bildet die Archivstruktur ab.

```
<ead> Outermost wrapper of the finding aid
    <eadheader> Wrapper for description of the electronic finding aid (not of the collection itself)
                                    . . . . .
    <archdesc> Wrapper for the description of the collection itself
                                    . . . . .
    <custodhist> Custodial History
```

Übersicht über das EAD-Template

Die hierarchische Archivstruktur, bestehend aus geschachtelten Verzeichnungs-einheiten, ist in EAD nur teilweise rekursiv aufgebaut. Dedizierte *component levels (c1,* bis *c12*) bietet aber genügend Tiefe für archivische Belange. Die *component levels* sind identisch aufgebaut. EAD nutzt die in XML vorgegebene Verschachtelungsmöglichkeit ohne interne Verweisstruktur.

```
<archdesc level="fonds"> Wrapper for the description of the collection itself
    <did> Descriptive identification: bundles core information about the described materials
        <unitid> Collection number
        <unittitle> Title of component (series, subseries, file, etc.)
        <unitdate> Corresponding dates
        <physdesc> Size of Collection
        <origination> Creator of the collection in catalog authority form
    <dsc> Description of subordinate components
```

```
<c01 level="series">  Component, first level (usually a series)
    <did>  Descriptive identification, required wrapper element
                    . . . . .
    <c02 level="series">  Component 2nd level (e.g. subseries, file, or item)
        <did>  Descriptive identification, required wrapper element
                    . . . . .
        <c03 level="series">  Component 2nd level (e.g. subseries, file, or item)
            <did>  Descriptive identification, required wrapper element
                    . . . . .
            <c04 level="recordgrp">  Component 2nd level (e.g. subseries, file, or item)
                <did>  Descriptive identification, required wrapper element
                    . . . . .
                <c05 level="file">  Component 2nd level (e.g. subseries, file, or item)
                    <did>  Descriptive identification, required wrapper element
                    . . . . .
                        <physdesc>  Size of Collection
                            <physfacet type="format">  information about aspects of the material
                    <odd>  Other Descriptive Data
                        <doagrp>  Digital Archival Object Group
                            <doaloc xlink:hfref>  Digital Archival Object Location
```

Übersicht über das <archdesc>-Element

5.2 Beurteilung

5.2.1 Logische Struktur

Jede Verzeichnungseinheit *(component level)* ist grundsätzlich identisch aufgebaut und in die nächst höhere Verzeichnungseinheit eingebettet. Gemeinsam bilden sie so je nach Betrachtungsweise die Archivtektonik oder den Findmittelbaum ab. Metainformationen werden implizit von einer Verzeichnisebene *(component level)* zur andern vererbt, damit ist EAD grundsätzlich redundanzfrei und einfach zu lesen. Es können aber nur hierarchische Strukturen abgebildet werden.

Das Fehlen von XML-Wrapper-Strukturen erlaubt es nicht, andere (PREMIS, LMER, *Dublin Core* etc.) oder eigene Metadatenschemata in EAD einzubinden. Stattdessen bietet EAD ein Element *odd (other descriptive data)*, das in Form von *Key-Value*-Listen beliebige Metadaten aufnehmen kann.

```
<list>
        <defitem>
                <label>mimeType</label>
                <item>application/pdf</item>
        </defitem>
        <defitem>
                <label>size</label>
                <item>27193</item>
        </defitem>
        <defitem>
                <label>CRC32</label>
                <item>0x62973a1c</item>
        </defitem>
        <defitem>
                <label>actor</label>
                <item>Mike Jonson</item>
        </defitem>
</list>
```

Technische Metadaten als Key-Value-Paare

Es besteht aber die Möglichkeit, aus jeder Verzeichnungseinheit mit *XLink* bzw. *XPointer* auf eine externe Ressource zu verweisen.

5.2.2 Metadaten nach ISAD(G)

EAD ist eigentlich eine XML-Schema-Repräsentation von ISAD(G), damit sind alle relevanten ISAD(G)-Verzeichungselemente abgedeckt und so beim Erstellen und beim Lesen klar definiert.

	Verzeichnungselement ISAD(G)	EAD
1.1	Signatur	unitid
1.2	Titel	unittitle
1.3	Entstehungszeitraum / Laufzeit	unitdate
1.4	Verzeichnungsstufe	level
1.5	Umfang (Menge und Abmessung)	physdesc
2.1	Name der Provenienzstelle	custodhist

ISAD(G)-Verzeichnungselemente und ihre Abbildung in EAD

Technische Metadaten sind in EAD nicht vorgesehen und können nur in Freitext oder *Key-Value*-Paaren erfasst werden.

5.2.3 Datenmigration

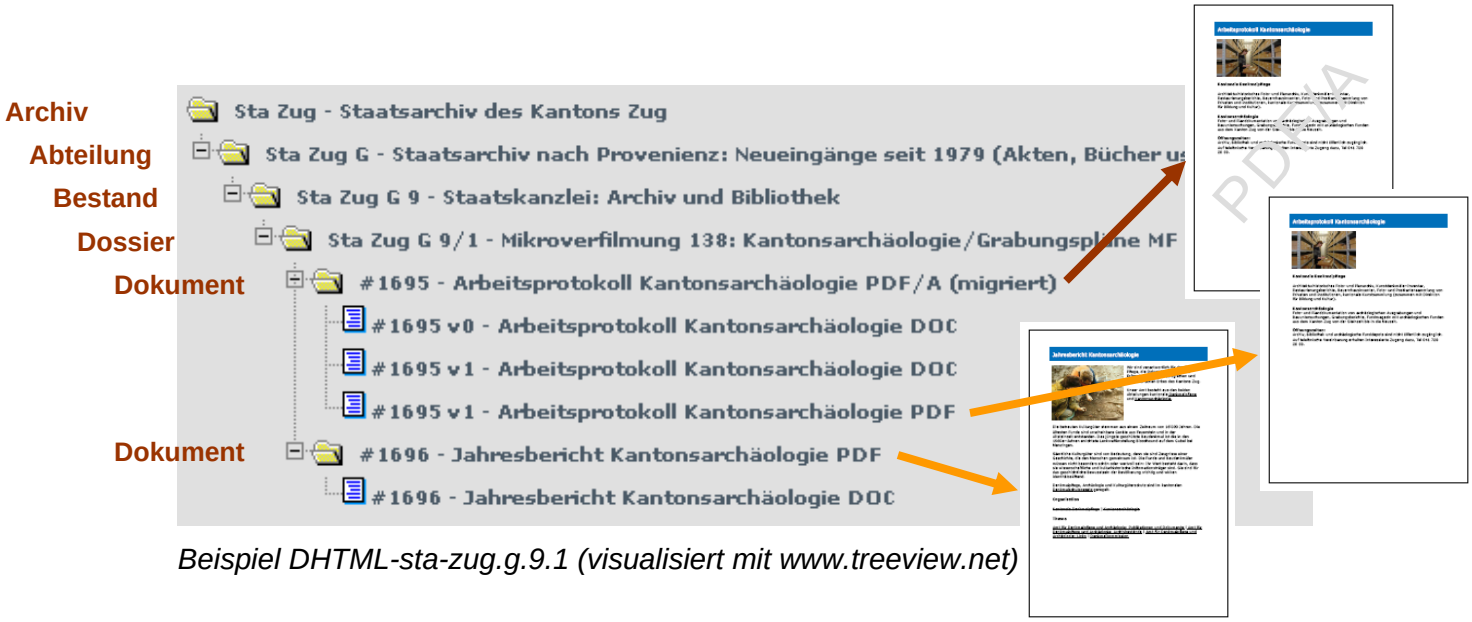
Das Fehlen von technischen Metadaten schliesst die Abbildung einer Datenmigration im EAD-Schema praktisch aus.

5.2.4 Zusammenfassung

Überzeugend sind die den archivischen Belangen angepasste hierarchische Verschachtelung der Verzeichnungsebenen und die vollständige Abbildung von ISAD(G). Hingegen ist die Abbildung technischer Belange von Datenobjekten innerhalb des EAD-Schemas nicht möglich, sie kann nur mit externen Verweisen realisiert werden. Das heisst, eine Beurteilung von EAD als AIP-Lösung ist nur in Zusammenhang mit einem technischen Metadatenschema wie etwa PREMIS oder LMER möglich.

6 Anhang: Beispieldaten

Als Beispieldaten für die verschiedenen Metadatenstandards werden halbfiktive Primär- und Metadaten benutzt, die von Olivier Debenath, StAZG, für das Projekt AUGev entwickelt wurden. Das Beispieldossier "Sta Zug G 9/1" mit Titel "Mikroverfilmung 138: Kantonsarchäologie/Grabungspläne MF 138/1-10" enthält zwei Dokumente, "Arbeitsprotokoll Kantonsarchäologie" und "Jahresbericht Kantonsarchäologie". Beide Dokumente wurden in Word erstellt, von "Arbeitsprotokoll Kantonsarchäologie" gibt es zwei Versionen. Bei Abschluss des Dossiers (oder Dokumentes) wurde vom GEVER-System von der jeweils aktuellen Dokumentversion eine PDF Datei erstellt. Zu einem späteren Zeitpunkt (bei der Übernahme ins Archiv) wurde das PDF #1695 in ein PDF/A konvertiert (hier ein PDF 1.4).



Beispiel DHTML-sta-zug.g.9.1 (visualisiert mit www.treeview.net)

Die Ablieferung (SIP) besteht also aus den GEVER-Metadaten, drei Word-Dateien und zwei PDF-Dateien. Das Beispiel befindet sich auf der AUGev Website unter der URL: http://www.kost-ceco.ch/augev/XML_Schnittstellen/DHTML-sta-zug.g.9.1/sta-zug.g.9.1.html

	GEVER (AMT)			ARCHIV
# 1695	000001695_v0.doc —bearbeiten→	000001695_v1.doc —abschliessen→	000001695_v1.pdf —migrieren→	000001695_v1_m0.pdf
# 1696	000001696_v0.doc —abschliessen→		000001696_v0.pdf —übernehmen→	000001696_v0.pdf

Zustandsdiagramm Dokument # 1695 und # 1696:
"bearbeiten, abschliessen und übernehmen und migrieren"