Hessisches Landesarchiv

HESSEN

Kopplung von Arcinsys und Kitodo

Aufbau einer provisorischen Schnittstelle zum Metadatenimport

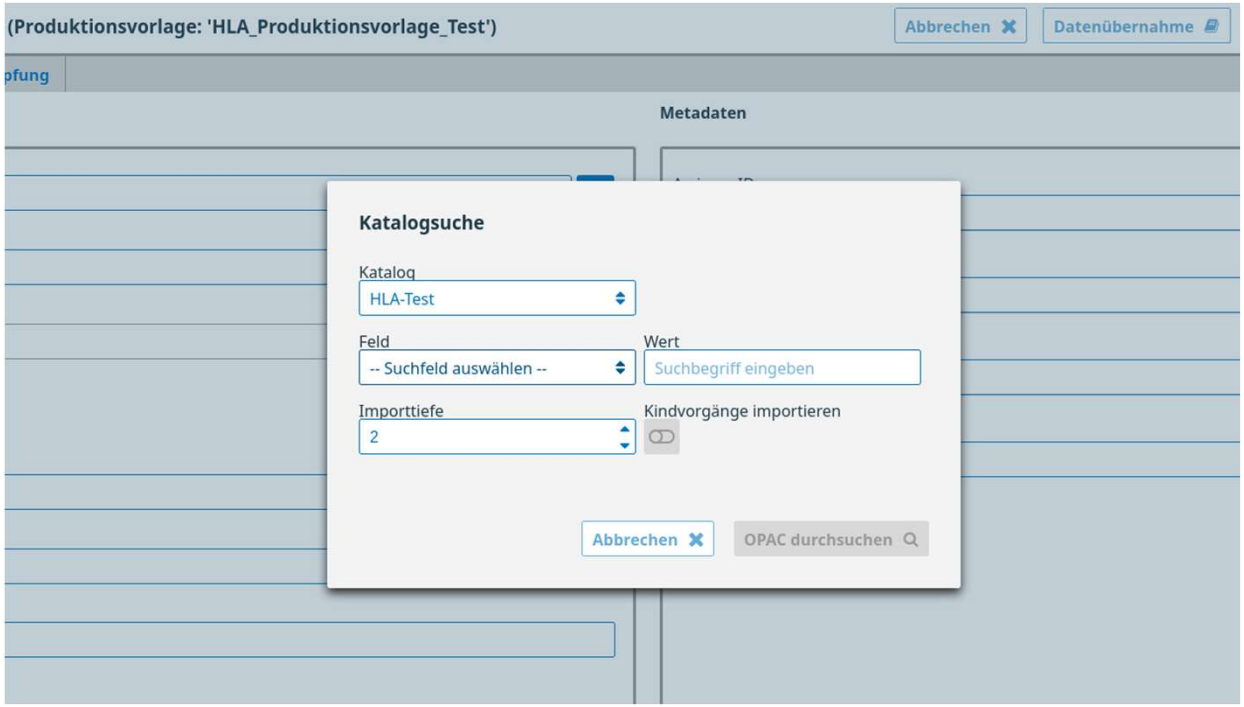
Jörg Bieszczak (HRZ Uni Marburg)
Max Gerlings (HRZ Uni Marburg)
Nils Reichert (Hessisches Landesarchiv)

Berlin, den 07.11.2023

HESSEN

Datenimport

- Der Workflow eines Vorgangs beginnt **unbedingt** mit einem Import der Metadaten (aus einem „Katalogsystem“)
- Anhand eines Identifiers wird ein Datensatz (ggf. mit Eltern) per Schnittstelle geladen
- Alternativ kann per Massenimport eine Liste von Identifiern verarbeitet werden.
- Die momentan verwendete „Datenübernahme“ (manueller Upload) kann mehrere Hierarchieebenen nur schwerlich abbilden



2022

HESSEN

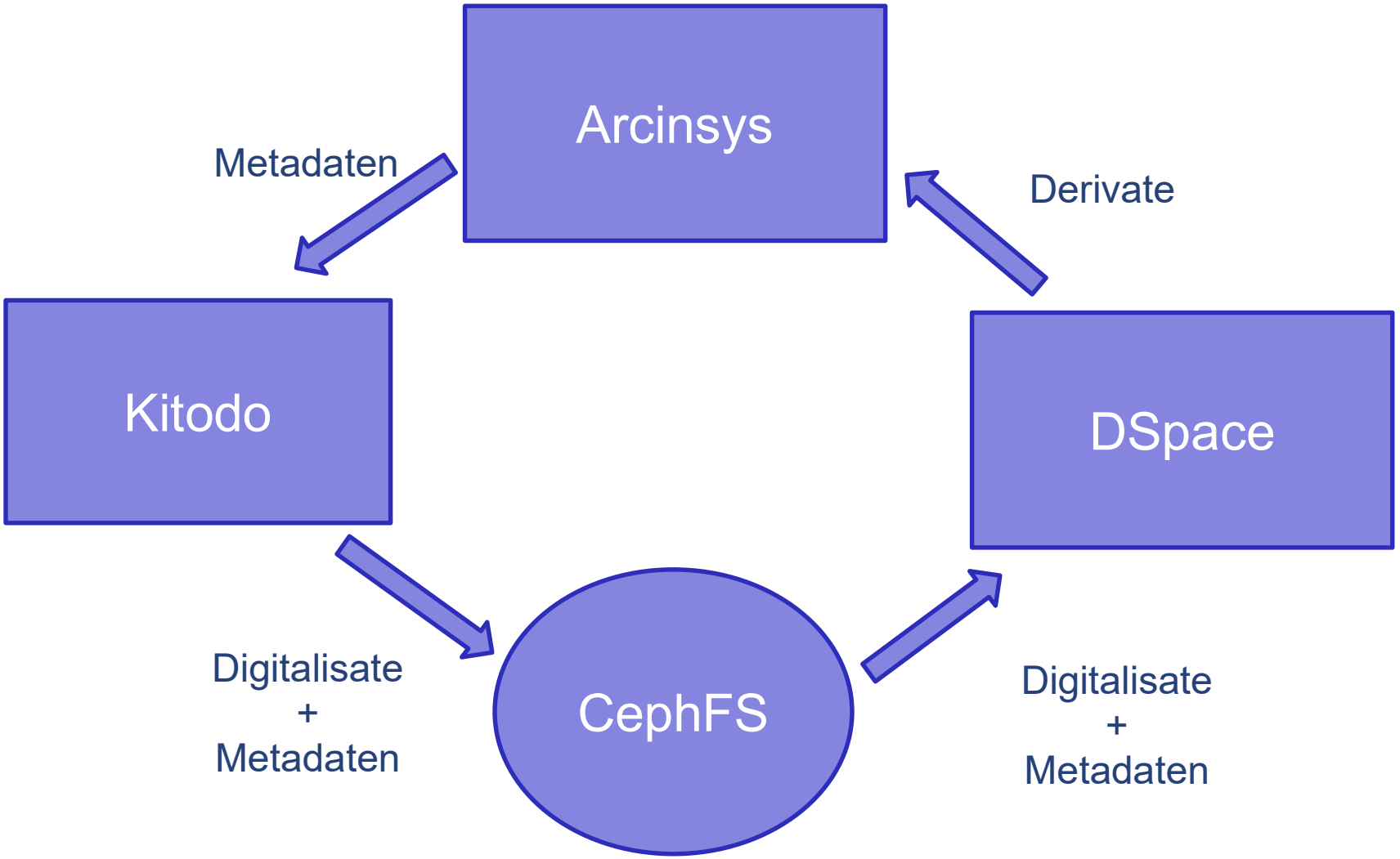
Status Dateningest (Testphase)

- Verarbeitung von EAD-XML durch XSLT-Transformationen
  - Bislang nur einzelne Stücke, keine ganzen Bestände, kein Massenimport
- Zentrale Workflow-Funktion von Kitodo zur Verarbeitung von „Vorgängen“ im Aufbau
  - Potentiell Steuerung von Prozessen im Repo (wenn dieses entsprechende Schnittstellen/Rückgabewerte liefert) möglich
  - Möglichkeit zur automatischen Initiierung von Vorgängen/Workflows (z.B.durch Arcinsys) noch zu klären
    - Nach der Verarbeitung von Kitodo (strukturelle Metadaten, Erstellung Derivate) können Derivate z.B. an Arcinsys ausgeliefert werden.

2022

HESSEN

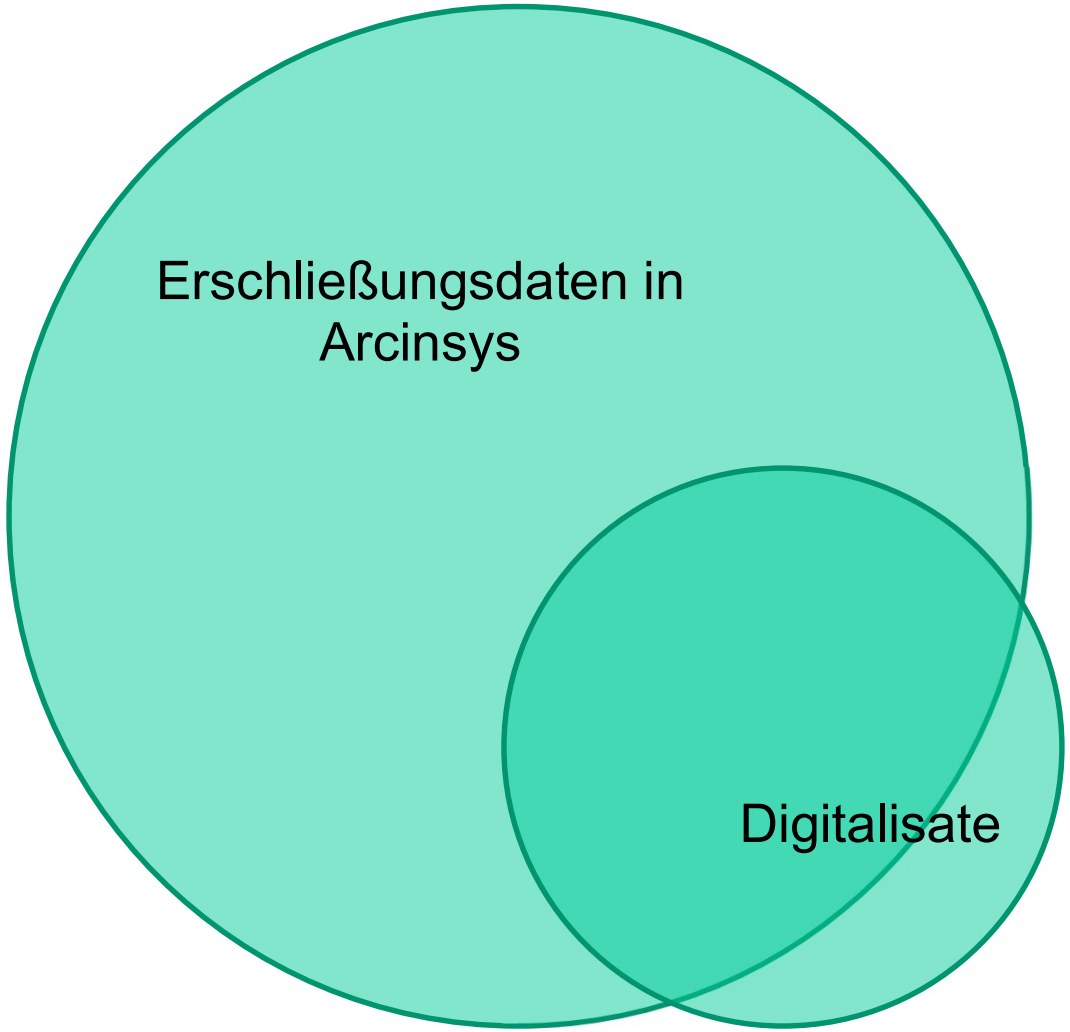
Circle of Life



2022 + 2023

HESSEN

Problem: Nicht übereinstimmende Datenmengen



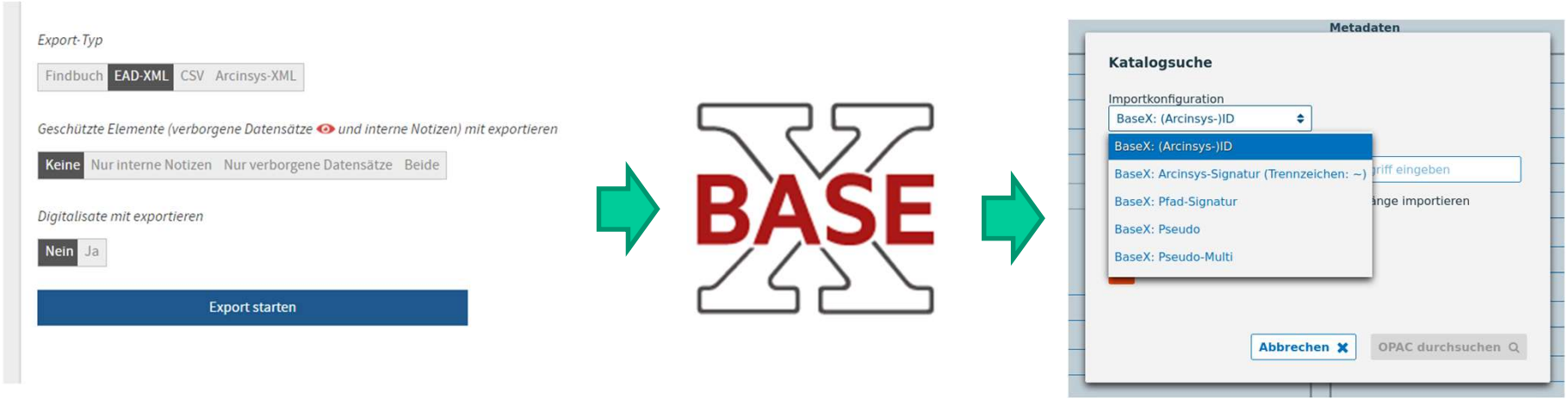
Teil 1: Austausch zwischen Datensystemen

Ziel: Massenhaft die gewünschten u. benötigten Metadaten in Kitodo aufnehmen

HESSEN

Pseudo-API(s) zum Datenimport mit BaseX

- Automatischer Export aller (öffentlichen) Erschließungsdaten aus Arcinsys
  - per cURL
  - über Exportfunktion der Arcinsys-Weboberfläche
  - in EAD-XML (EAD-DDB)
- Konfigurierbare REST-API



HESSEN

Pseudo-API(s) zum Datenimport mit BaseX Teil 2
Bash-Script: Arcinsys2BaseX.sh

```
arcinsys_download_get_JOBID () {
  echo "Start getting JOBID for ${1}..." 1>&3

  START_EXPORT_RESULT=$(
  curl --silent --show-error --fail \
  -H "${JSESSION_COOKIE}" \
    "${BASEURL_ARCINSYS}/system/ajaxStartXmlExport.action?itemid=${1}&visValue=no&exportDigitalisate=no&exportTyp=${2}"
  )

  if [[ "${START_EXPORT_RESULT}" != "" ]]; then
    JOBID=$(echo "${START_EXPORT_RESULT}" | jq .job.id)
    echo "Finished getting JOBID ${1}" 1>&3
  else
    error_exit "curl result is empty, no response from arcinsys! Session still alive?"
  fi

  echo "${JOBID}"
}
```

HESSEN

TAXI-Format

– Token-Index in BaseX möglich

– Abfragegeschwindigkeit um Faktor 50 bis 300 erhöht!

– Anpassung der notwendigen Datenfelder auf wesentlich schlankere und effizientere Struktur. (XML-Lehrbuch ;-))

TAXI

Call it:

Tiny Archive XML Interchange Format

Tiny Archive eXchange Interface

Trivial Archive XML Interchange Format

Trivial Archive eXchange Interface

or "something, that converts supercomplicated archival transport xml-stuff into proper readable and structured xml, like real people do."

```
<?xml version="1.0" encoding="UTF-8"?>
<ead ...
   <eadheader ...
      <eadid mainagencycode="">b7205</eadid>
      ...
   </eadheader>
   <archdesc level="collection" type="Findbuch">
      <did>
      ...
      </did>
      ...
      <dsc>
         <c level="collection" id="b7205">
            <did>
            ...
            </did>
            ...
            <c level="class" id="g257657">
               <did>
               ...
               </did>
               <c level="file" id="v2256588">
                  <did>
                     <unitid>43</unitid>
                     <unittitle>Liederbuch von
...</unittitle>
                     <physdesc>
                        <genreform>Sachakte</genreform>
                     </physdesc>
                     <unitdate normal="1920-01-01/1920-12-
31">
          1920
          </unitdate>
                     <unitid type="Altsignatur">CH
1043</unitid>
                     <abstract type="Enthält">32
Blatt</abstract>
```

```
<?xml version="1.0" encoding="utf-8"?>
<taxi ...
   <object type="genericitem" id="v2256588"
fsig="adjb/ch_2/43" asig="AdJB~CH 2~43">
      <item>
         <id>v2256588</id>
         <signature>
            <arcinsys>
               <current>43</current>
               <previous>CH 1043</previous>
            </arcinsys>
            <path>
               <current>43</current>
               <previous>ch_1043</previous>
            </path>
         </signature>
         <title>Liederbuch von ...</title>

<document_type>ThematicFile</document_type>
         <genre>Sachakte</genre>
         <inclusive_dates>
            <text>1920</text>
            <begin>1920-01-01</begin>
            <end>1920-12-31</end>
            <period>1920-01-01/1920-12-
31</period>
         </inclusive_dates>
         <content>
            <text>32 Blatt</text>
         </content>
      </item>
      <fond>
      ...
      </fond>
      <archive>
      ...
      </archive>
```

HESSEN

Hierarchischer Massenimport

- Regelsätze
- Identifier
- Überordnung

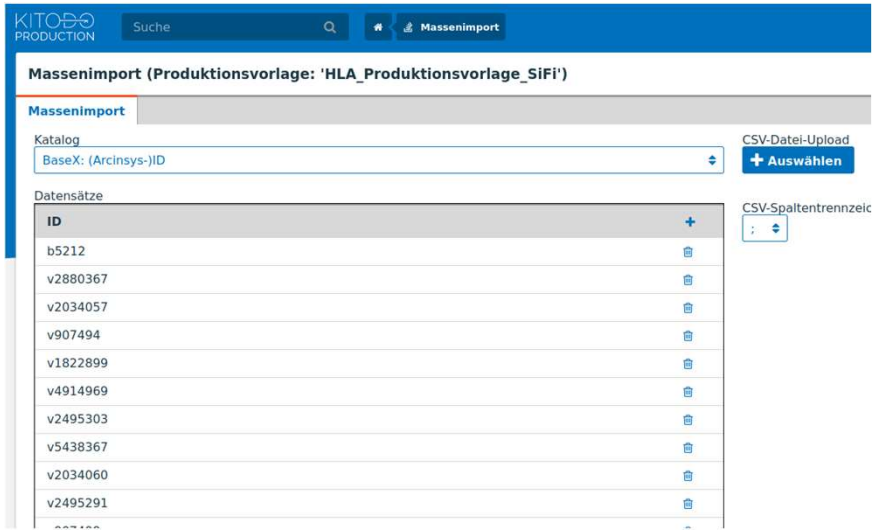
```
<key id="ArcinsysID" use="recordIdentifier">
        <label>Arcinsys ID</label>
</key>
<key id="stockID" use="higherlevelIdentifier">
        <label>Stock-ID</label>
        <label lang="de">Bestands ID</label>
```

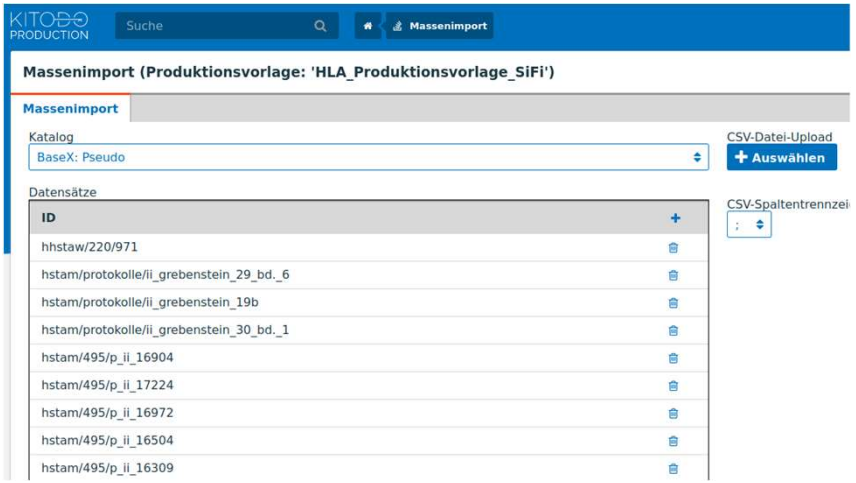
```
-<taxi>
 -<object type="genericitem" fsig="hstam/bestand/stueck">
  -<item>
    <document_type>Unknown</document_type>
   -<signature>
    -<path>
      <current>hstam/bestand/stueck</current>
     </path>
    </signature>
   </item>
  </object>
 </taxi>
```

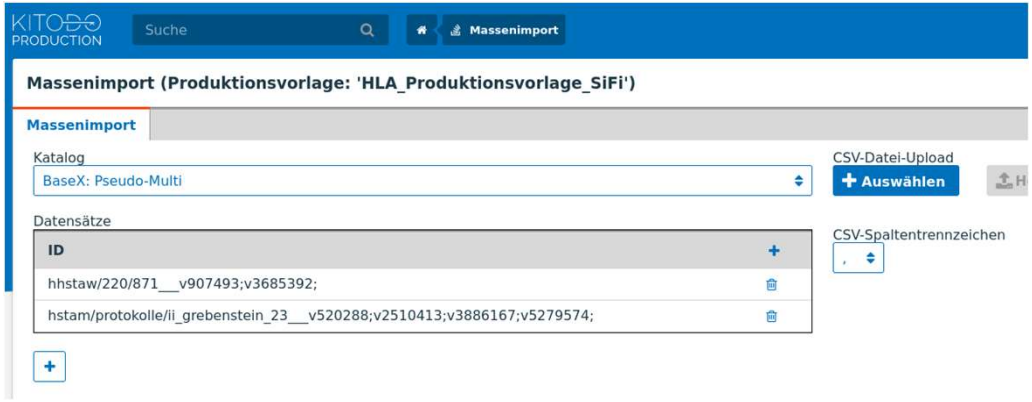
```
<xsl:template match="taxi:object/taxi:fond/taxi:id">
    <kitodo:metadata name="stockID">
            <xsl:value-of select="normalize-space()"/>
    </kitodo:metadata>
</xsl:template>

<xsl:template match="taxi:object[@id]">
    <kitodo:metadata name="ArcinsysID">
            <xsl:value-of select="@id"/>
    </kitodo:metadata>

    <xsl:apply-templates select="@*|node()"/>

</xsl:template>
```

HESSEN

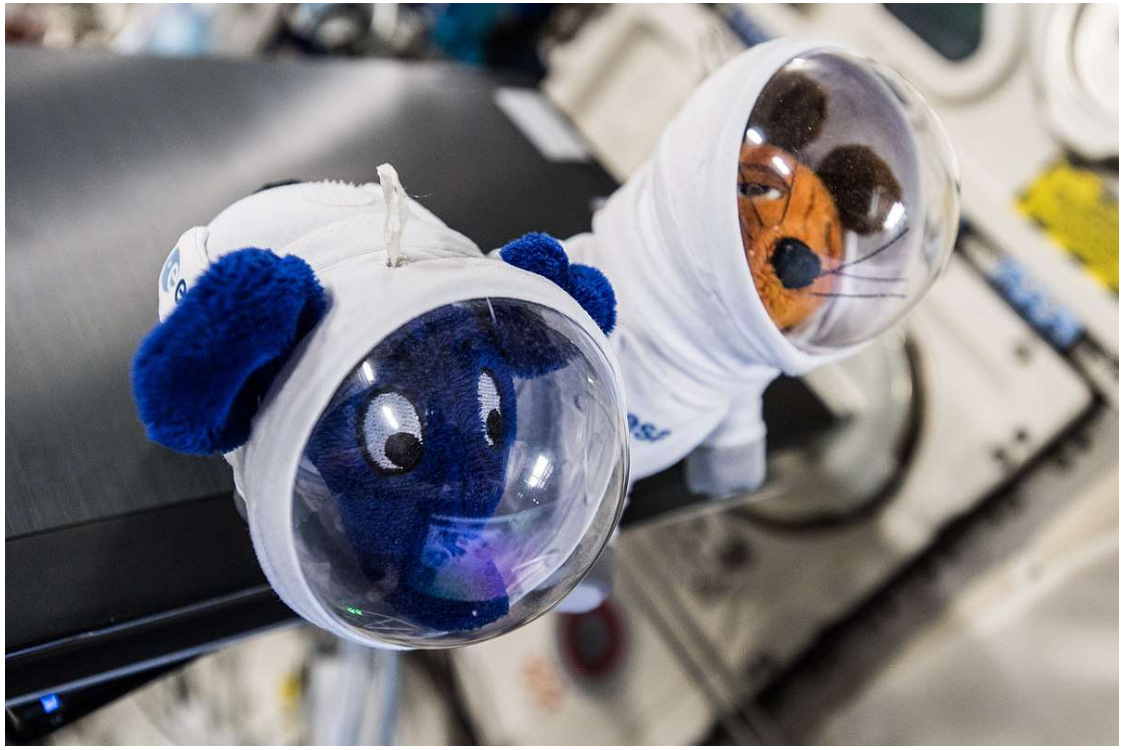
Hierarchischer Massenimport + Pseudoschnittstellen







HESSEN



Teil 2: Prozesse starten

Ziel: Abbildung des einfachsten und regelmäßigsten bisherigen Workflows

(Landesnutzen aus Bundessicherungsverfilmung)

HESSEN

Workflows („Step by Step“)

- Workflows werden oft projektweise definiert
  - bisweilen auch erst nach der Durchführung der Digitalisierung

- Wenige dauerhaft und regelmäßig laufende Workflows
  - Bundessicherungsverfilmung (ca. 2-monatlich, relativ wenig Überraschungen)
  - Reprographieaufträge für Nutzer*innen (halbjährlich, Umfang vorher unbekannt)
  - Längerfristige Kooperationsprojekte

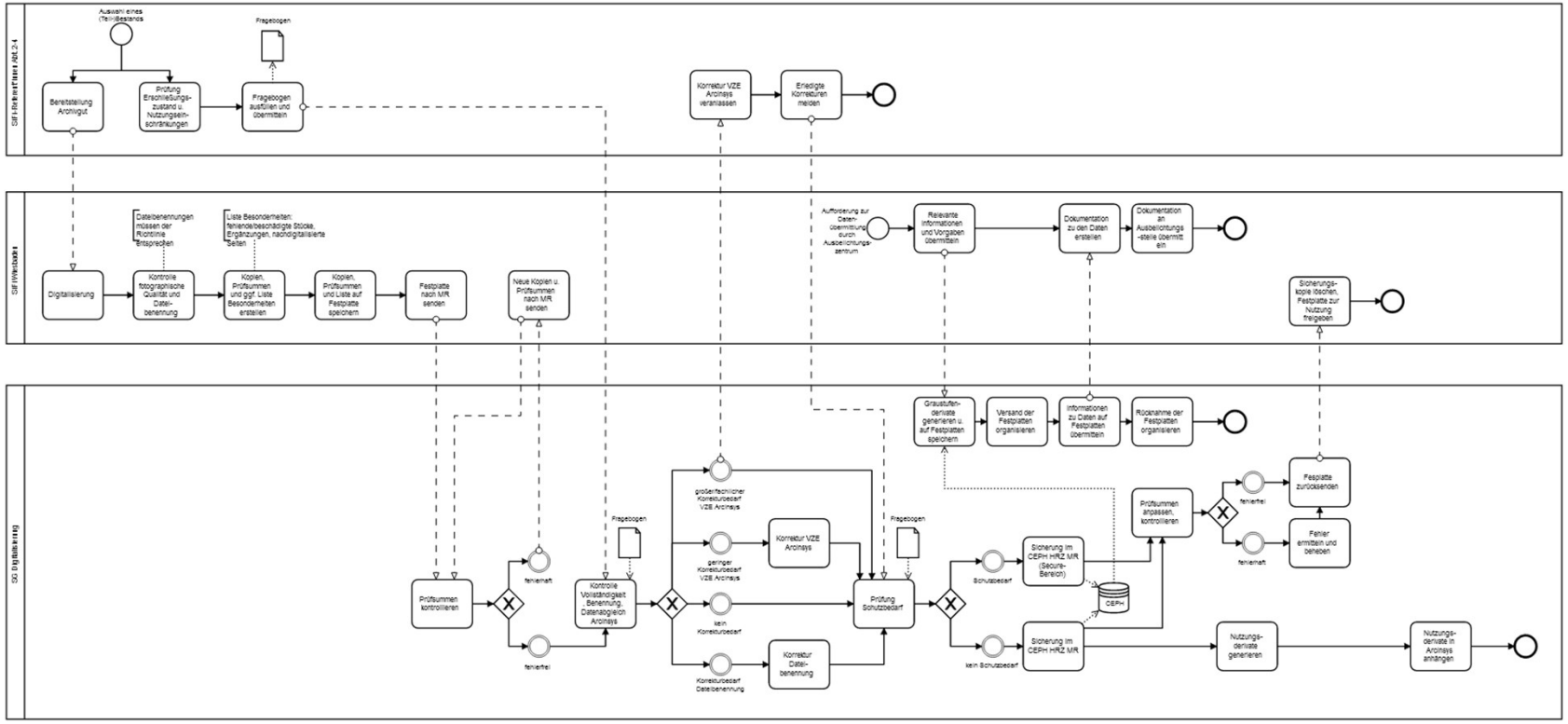
HESSEN

Ein erster Workflow

- Notwendige Anpassung der Benennungsstruktur (Zuordnung zu Metadaten)
- Abgleich Schutzbedarf und differenzierte Behandlung
- Korrekturmeldungen an Erschließung
- Vorbereitung Langzeitspeicherung
- Derivate erzeugen und veröffentlichen
- Festplatten löschen
- An Ausbelichtungsstelle versenden

HESSEN

Der Workflow – wie er war



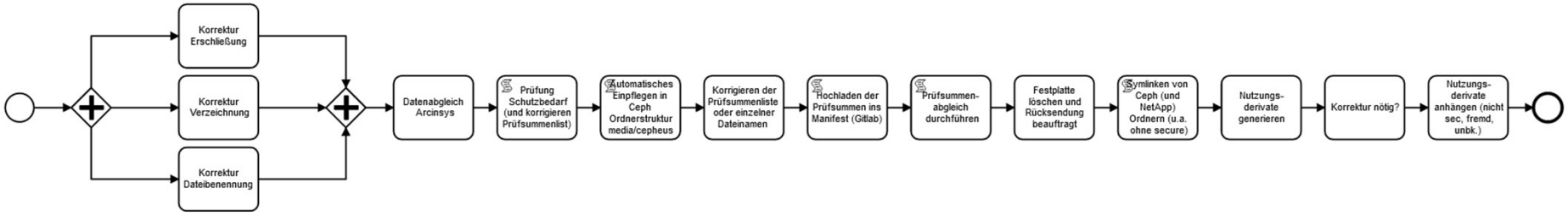
Verzweigungen dienen der Abbildung von:

- Korrektur- und Schutzfällen
- von einander unabhängige weiteren Verarbeitungen

HESSEN

Der Workflow – wie er ist

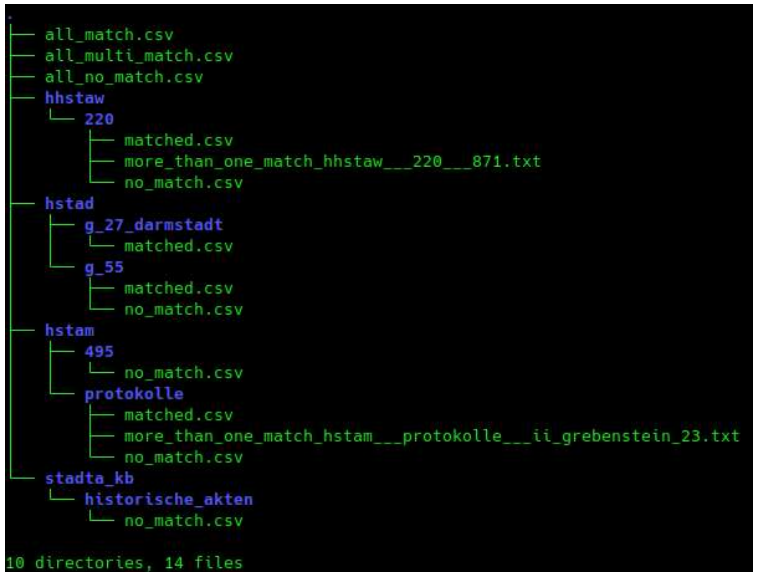
- Keine Verzweigungen möglich (! ! !)
  (siehe https://github.com/kitodo/kitodo-production/issues/5630)
- Kein initialer Abgleich als Teil des Workflows
- Gleicher Workflow für alle Tektonikebenen im gleichen Projekt

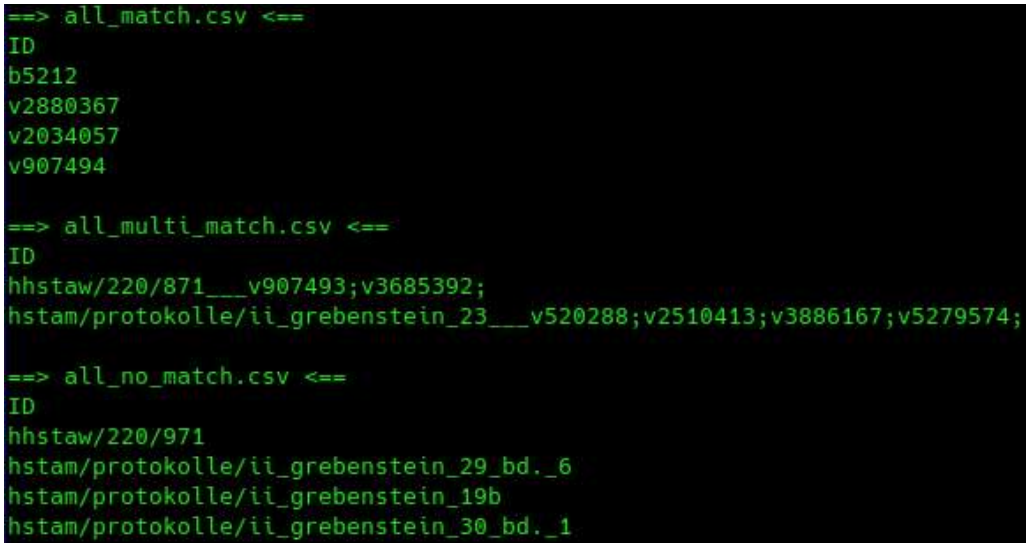


HESSEN

Pre-Ingest Ingest-Workflow

- Anlegen von (sinnvollen) Vorgängen setzt Datenabgleich voraus
- Automatischer Prozess zum Datenabgleich findet ***vor*** dem Workflow statt
- z.T. manuelle Korrekturen ***vor*** dem Pre-Ingest nötig
- ***vor*** eventuellen Korrekturen Upload in Ingest-Speicher





HESSEN

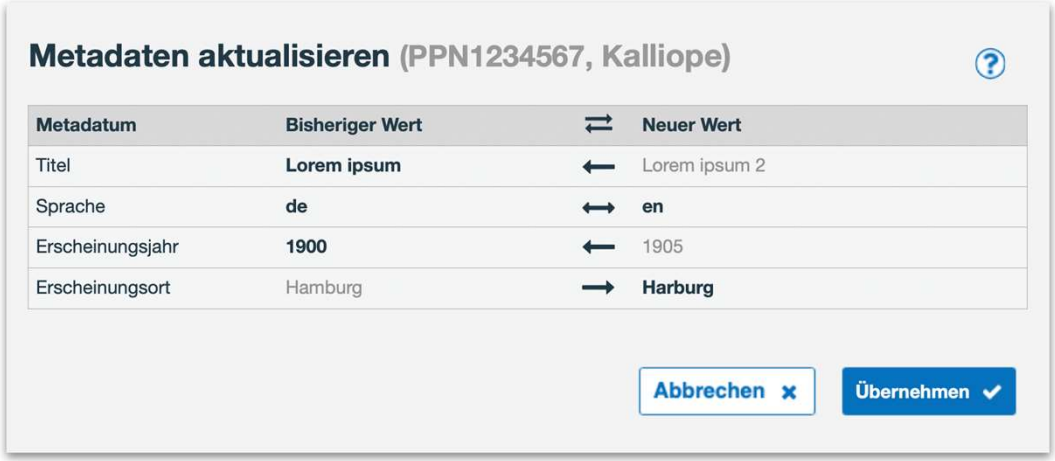
Pre-Ingest Ingest-Workflow Teil 2: Script

```
#extract archive from item
archive=$(echo "${item}" | awk -F / '{print $1}')
#extract fond from item
fond=$(echo "${item}" | awk -F / '{print $2}')
#create archive and fond folder
mkdir -p "${base_path_tmp}/${archive}/${fond}"
#check basex for item by signature
arcid=$(curl --silent --globoff "${url_basex}?${parameter_basex}=${item}")
#convert output to only get ids without header and xml
xmlout=$(echo "${arcid}" | xmlstarlet sel -t -v "/" 2>/dev/null)
#count the occurence of the letter 'v' (every arcinsysid starts with v)
count=$(echo "${xmlout}" | grep -Fo v | wc -l)
#check if more than one arcinsysid
if [[ "${count}" -gt 1 ]]; then
        filename="${item//\//___}"
        echo "More than one matching, tell this Arcinsys! | Signatur: ${item}"
        echo -n "${item}___" > "${base_path_tmp}/${archive}/${fond}/more_than_one_match_${filename}.txt"
        echo "${xmlout}" | tr " " "\n" >> "${base_path_tmp}/${archive}/${fond}/more_than_one_match_${filename}.txt"
        #replace newline by _ and add to all multi match file
        awk -v ORS=";" '1' "${base_path_tmp}/${archive}/${fond}/more_than_one_match_${filename}.txt" >> "${base_path_tmp}/all_multi_match.csv"
        #write newline to all multi match file
        echo "" >> "${base_path_tmp}/all_multi_match.csv"
#check if exactly one matching arcinsysid
elif [[  "${count}" -eq 1 ]]; then
        echo "Exact one matching! | Signatur: ${item} | ArcinsysID: ${xmlout}"
        echo "${xmlout}" >> "${base_path_tmp}/${archive}/${fond}/matched.csv"
#no matching arcinsysid
```

HESSEN

Pre-Ingest Ingest-Workflow: Perspektive

- Ein Massen-Reimport könnte dies ändern:
  1. Anlegen der Vorgänge aus den Pfaden zu den Digitalisaten
  2. Über Kitodo gesteuerter Abgleich mit Arcinsys
- ↦ Durch Reimport Anreichern der Metadaten nach dem Abgleich
  - hierbei wären insbesondere der Doctype und auch der Vorgangstitel wichtig



. . .

Teil 3: Prozesse durchführen

Ziel: Alle Workflow-Komponenten, die keiner menschlichen

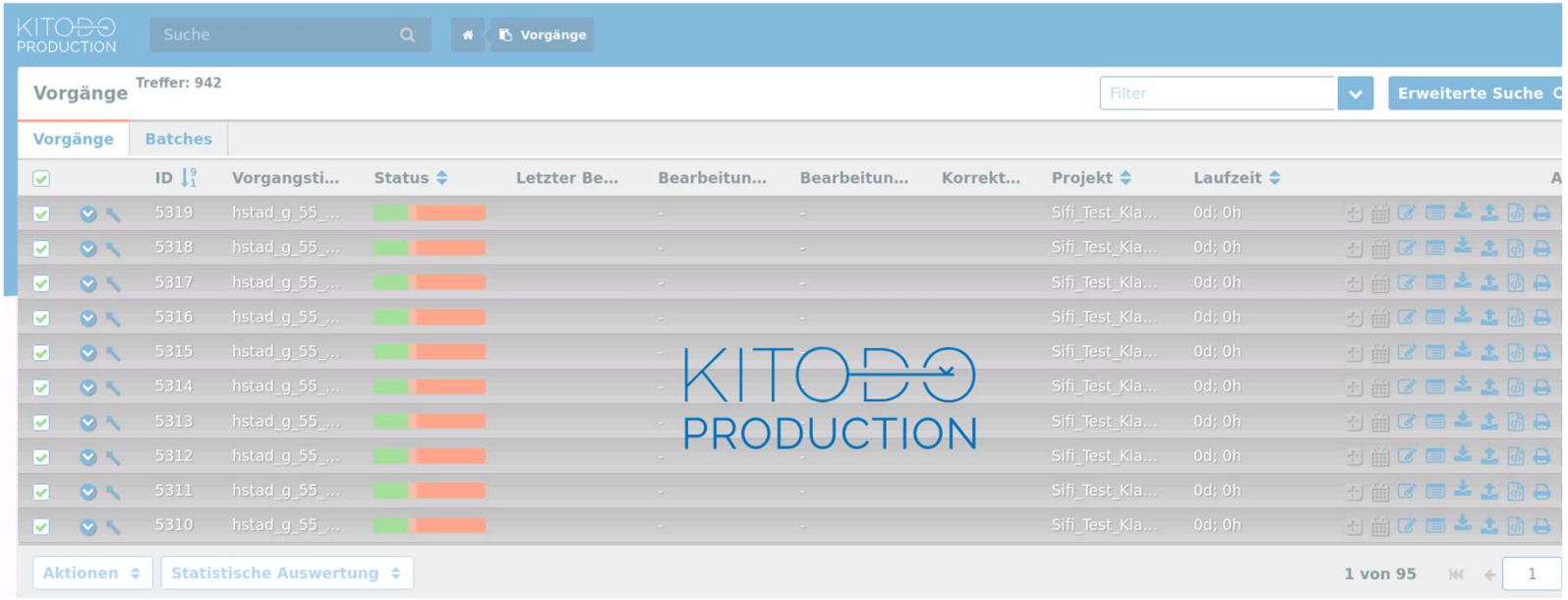
Entscheidung bedürfen, sollen automatisiert werden

HESSEN

Automatische massenhafte Datenverarbeitung

Performance:

- Import (CSV): circa 15min für ungefähr 1500 Vorgänge
- Schutzstatus prüfen und anschließendes Verschieben der Dateien vom Ingest in finale Struktur: ca. 1,5Std für ungefähr 1500 Vorgänge



HESSEN

Automatische massenhafte Datenverarbeitung

Herausforderungen:

- Zu viel Logging führt zu Deadlock und fehlerhaften Prozessen
- Erster Workflowschritt kann nicht automatisch sein (siehe https://github.com/kitodo/kitodo-production/issues/3672)
- Bearbeitung von Vorgängen ist möglich, noch während automatische Verarbeitung läuft
- Verändern des Bearbeitungsstatus eines Vorgangs wird direkt übernommen, klicken auf „Speichern“ nicht erforderlich ↦ mögliche Fehlerquelle
- Massenhaftes Löschen von Vorgängen bisher nicht möglich (siehe https://github.com/kitodo/kitodo-production/issues/5342)

VIELEN DANK FÜR DIE AUFMERKSAMKEIT!