

DHd2018

Köln 2018-03-02

Funktionale und deklarative Programmierung-basierte Methode für nachhaltige, reproduzierbare und verifizierbare Datenkuration

Gioele Barabucci

Cologne Center for eHumanities

Universität zu Köln



Funktionale und deklarative Programmierung-basierte Methode für nachhaltige, reproduzierbare und verifizierbare Datenkuration

Gioele Barabucci
Cologne Center for eHumanities

<https://gioele.io/p/dhd2018>

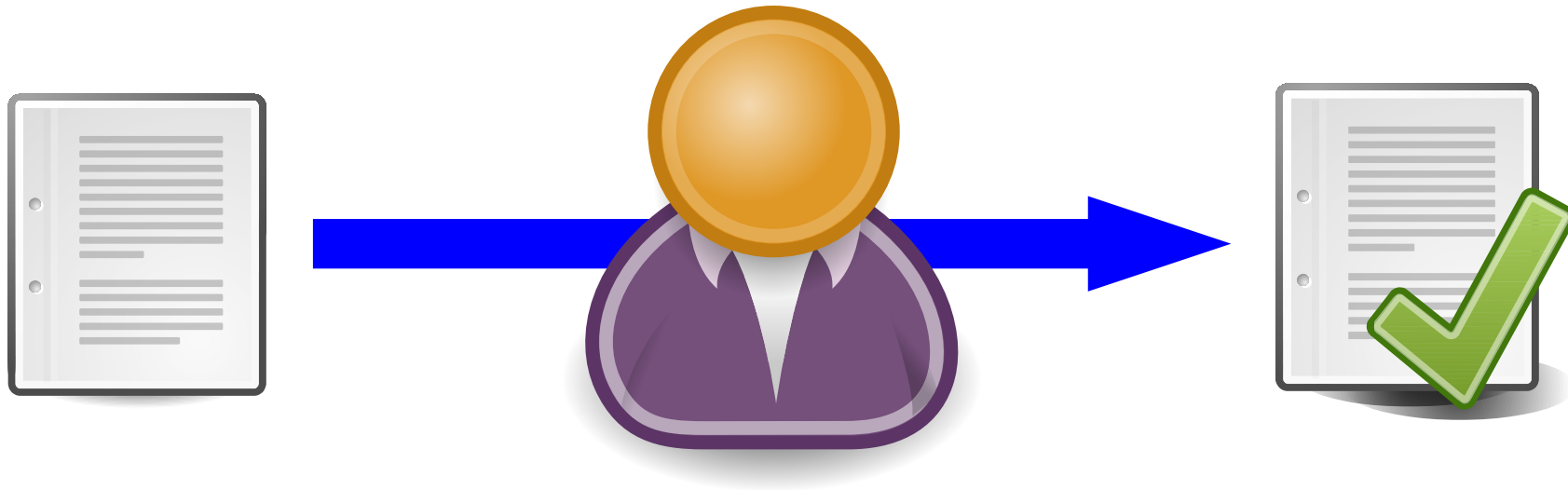
Teil 1

Das Problem

Kuration

»[one of the tasks of digital curators is to] intervene in the research process in order to translate or migrate data into new formats, to enhance it through additional layers of context or markup, to create connections between data sets, and to otherwise ensure that data is maintained in as highly-functional a form as possible.«

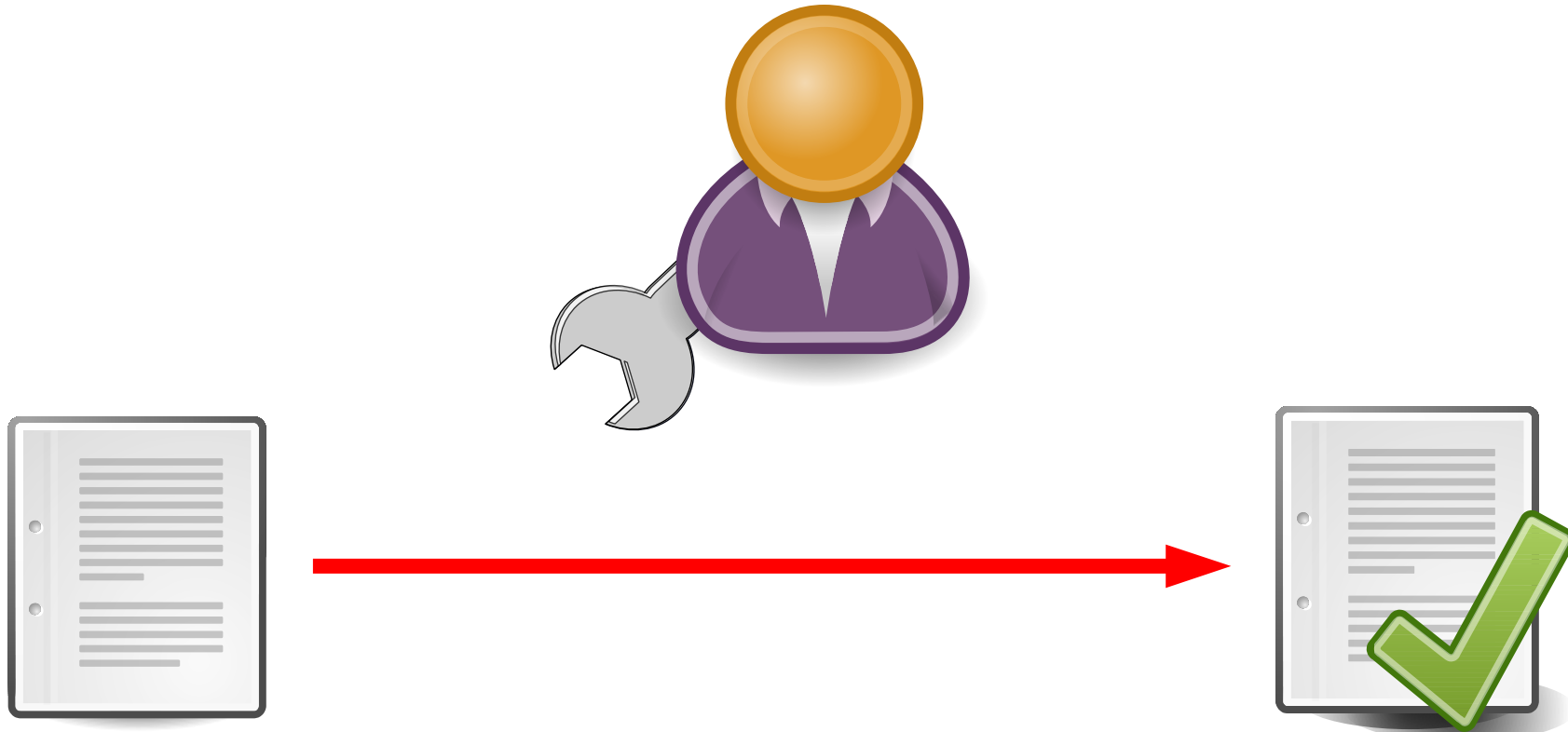
(Flanders & Muñoz 2017)



Gioele's opinion

Working code is the
best documentation of
the curation
process and activities

Die gute alte Kuration



... Zwei häufige Ereignisse...

- Neue Versionen der Dateien werden...
 - › gefunden
 - › hergestellt
 - › geliefert
- »Verflixt! Wir haben etwas falsch gemacht!«

Neue Versionen der Dateien

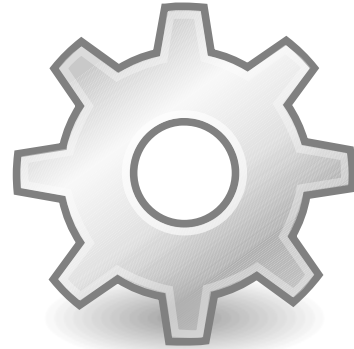
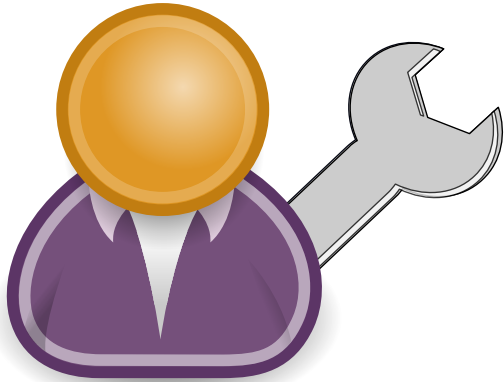
- Die ganze Arbeit muss von vorne begonnen werden.
- Haben wir alles dokumentiert? Alles?
- Sind die alte Anwendungen noch verfügbar?
 - › Auf welchem Rechner sind sie installiert?
 - › »Wie funktioniert eigentlich das Ding hier?«

Verflixt! wir haben etwas falsch gemacht!

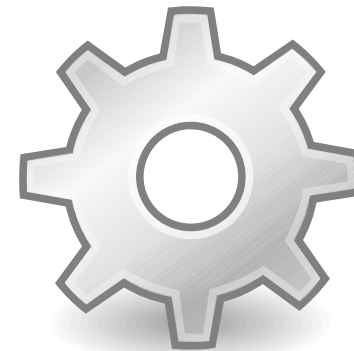
- Man muss nur Undo machen, oder?
- »Keine Angst, ist alles auf github!«
 - › Viel Spaß mit `revert`, Konflikte und verwirrte Commits.
 - › Zwei Tage später: »Es geht schneller wenn wir alles löschen und von vorne anfangen.«

Die neue Kuration

①



②



Teil 2

Die Lazarus-Methode

a.k.a.

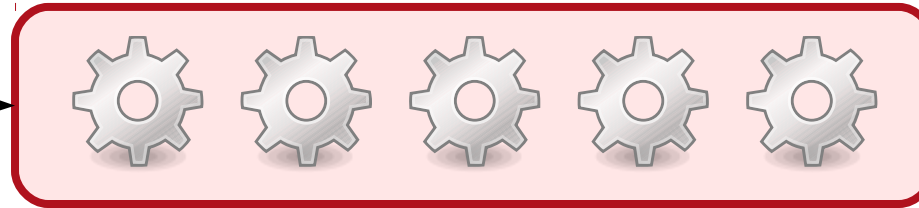
**Die „wiederholbare
Pipeline“-Methode**

Herkunft und andere Anwendungen

- Cologne Sanskrit Lexicon (LAZARUS-Projekt, 2015, DFG)
 - › ~400.000 Lemmata
 - › Martina Gödel (CCeH/textloop), Peter Dängeli (CCeH), Felix Rau (Institut für Linguistik), Gioele Barabucci (CCeH)
- Haller Online
 - › ~80.000 Sätze
- Papyrologie Wörter
 - › ~35.000 Sätze

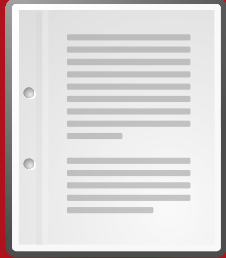


Lazarus-Methode

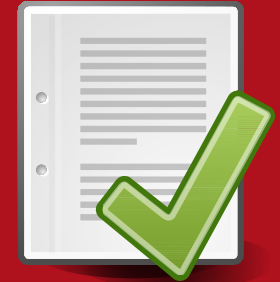
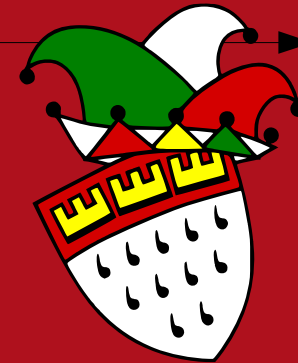
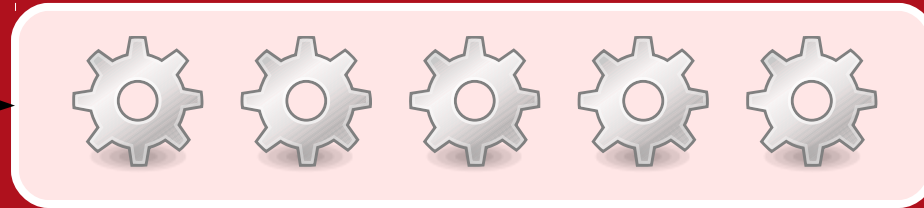


Hauptgrundsatz

Keine Datei wird „manuell“ modifiziert,
jede Operation muss von einem
Miniprogramm ausgeführt werden



Kölsche Methode

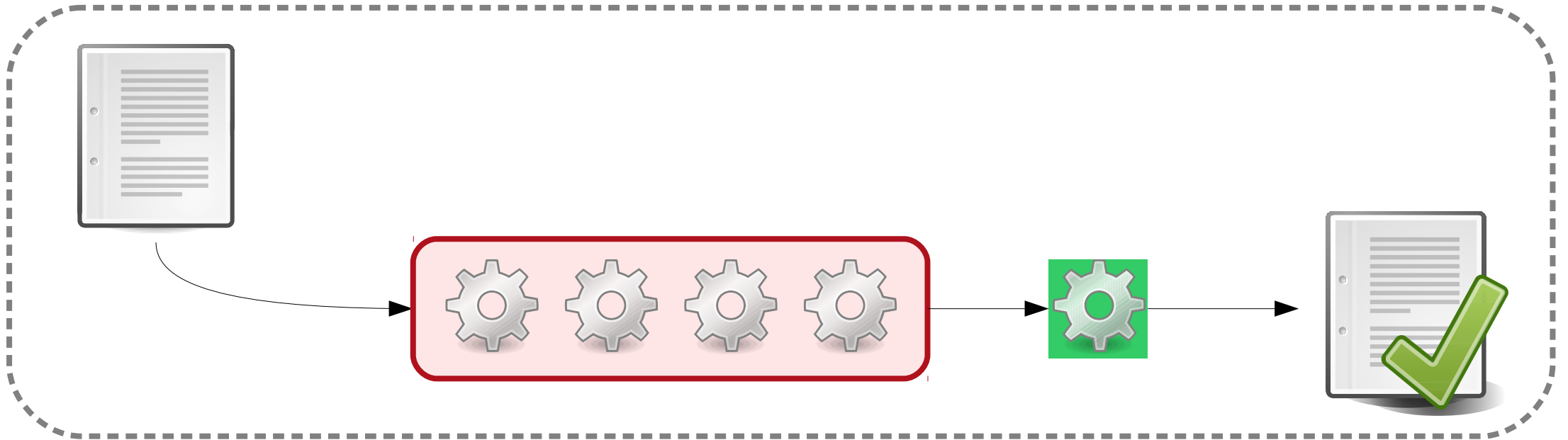


Grundgesetz

Kinn Datei weed metd'r Hand jeändert,
mallich Schritt muß vunn'em kleen
Projämmche ausjeführt wääde

Kölle Alaaf!

Lazarus-Methode



1. Die Miniprogramme 

2. Die Kurationspipeline 

3. Die Tests 

4. Der Makefile 

Miniprogramme

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="2.0">
  <xsl:import href="../common/identity.xsl"/>
  <xsl:import href="../common/chars.xsl"/>

  <!--
    This transformation reworks the placement of `;` and
    whitespace in cases where there is more than one `<ls>`.

    The result is more similar to the facsimile.
  -->

  <!--
    Turns <ls>Kalt2h. ; </ls><ls>Anukr.</ls>
    into <ls>Kalt2h.</ls>; <ls>Anukr.</ls>
  -->

  <xsl:template match="ls[ends-with(.,' ; ')]">
    <xsl:variable name="first-part"
                  select="concat(' ', substring-before(., ' ; '))"/>

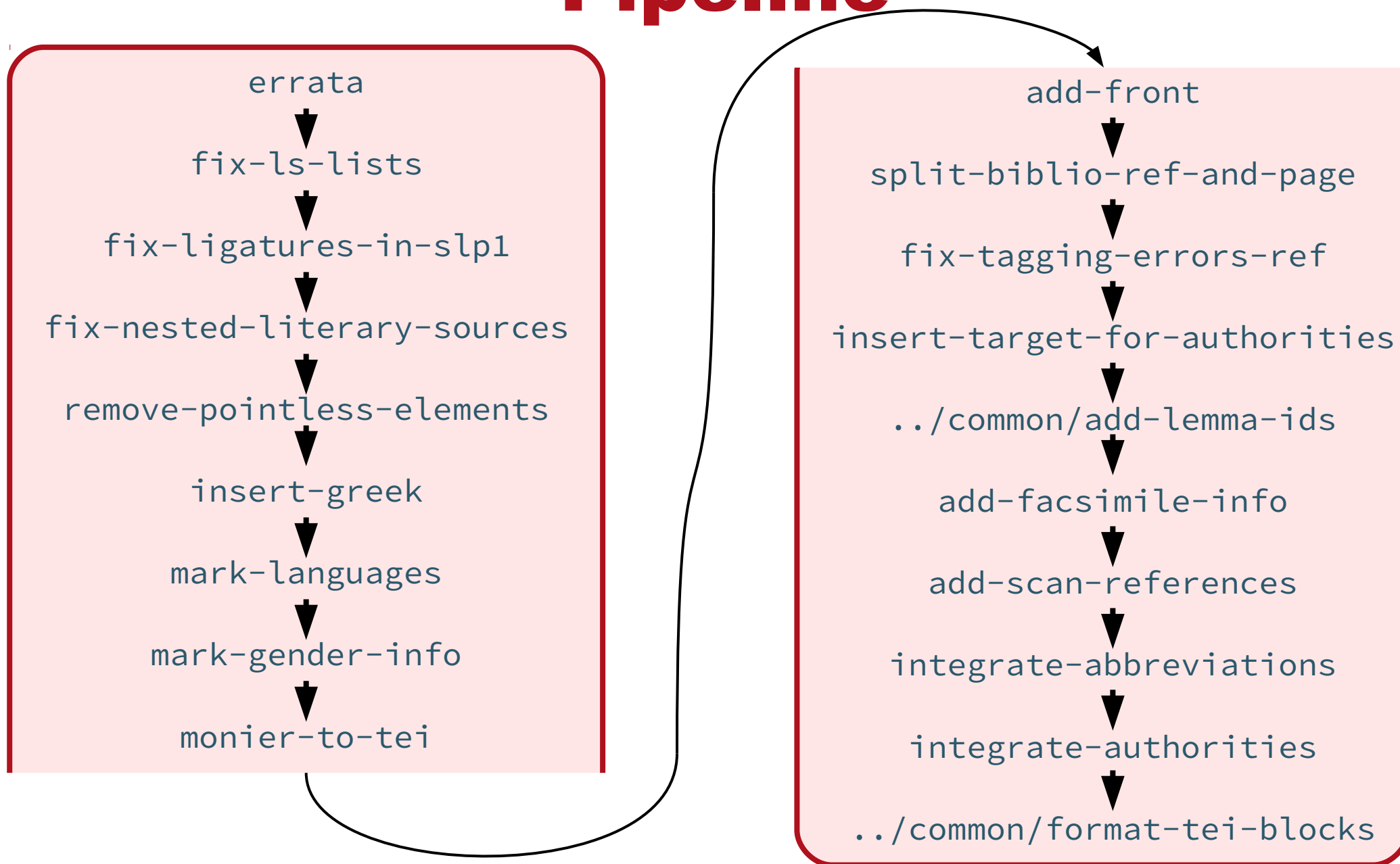
    <ls>
      <xsl:value-of select="$first-part"/>
    </ls>
    <xsl:text>;</xsl:text>
    <xsl:value-of select="$char-space"/>
  </xsl:template>
```



monier/fix-ls-lists.xsl

- 1 Miniprogram = 1 Kurationsschritt
- Sehr klein: 5 bis 15 Instruktionen
- *Idempotente* Funktionen
- Unabhängig von vielen technischen Details
- Wiederverwendbar

Pipeline



Tests

```
<pattern id="joins-split-words">
  <rule context="//tei:entry[.//tei:orth = 'abandon']">
    <assert test="contains(., 'connections')">The word
      `connections` has been joined correctly.</assert>
    <assert test="//tei:w = 'vi-utsfj'">The word `vi-utsfj`
      has been joined correctly inside a `s` element.</assert>
  </rule>
</pattern>
```



Beachten, dass alte Fehler nicht wiedereingeführt werden.

Guidelines
nachprüfen 

```
<pattern id="latin-words-are-properly-marked">
  <rule context="//tei:w[text() = 'axis']">
    <assert test="@xml:lang">Latin words should have
      an <span class='code'>xml:lang</span>
      attribute.</assert>

    <assert test="@xml:lang = 'lat'">The attribute
      <span class='code'>xml:lang</span> should be
      set to <span class='code'>lat</span> for Latin
      words, not <value-of select="@xml:lang"/>.</assert>
  </rule>
</pattern>
```

Makefile

```
monier.tei: $(wildcard $(XPROC_ROOT)/*.xml)
monier.tei: $(XPROC_EXECUTABLE)
monier.tei: abbreviations.tei
monier.tei: authorities.tei
monier.tei: greek.tei
monier.tei: monier.xml
    $(XPROC) -isource=$< result-url=$@ $(XPROC_ROOT)/monier/conversion.xml
```

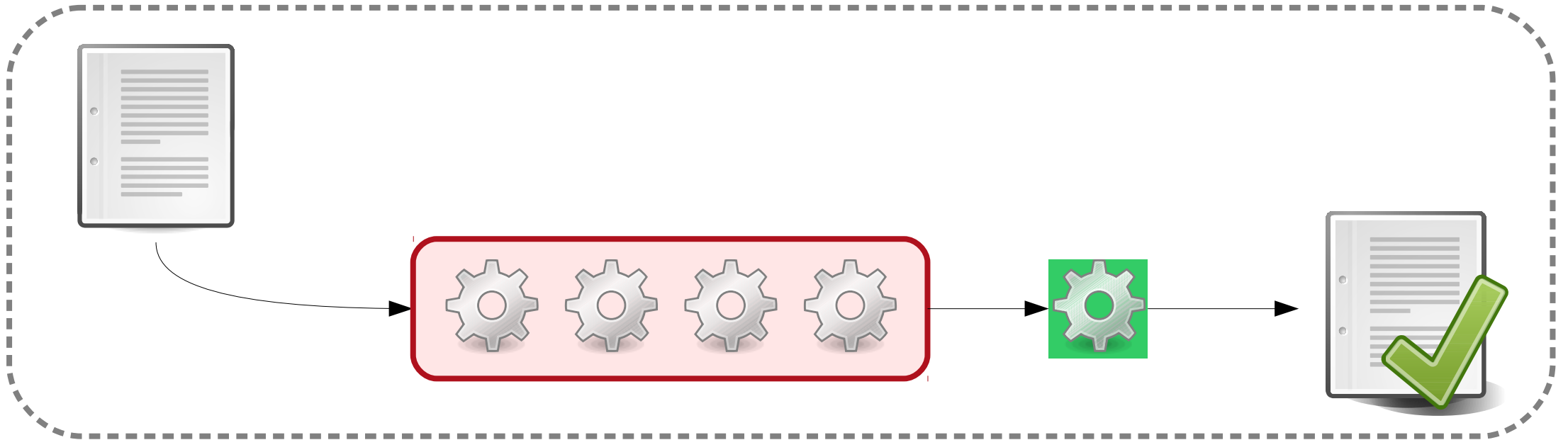
Dateien:

- welche Dateien brauchen wir?
- wo sind die Dateien?
- wie rufen wir sie ab?
- wo sollen die Ergebnisse gespeichert werden?

Tools:

- welche Anwendungen braucht man?
- woher kann man sie herunterladen?
- welche Parameter soll man sie angeben?

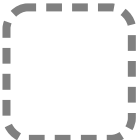
Lazarus-Methode



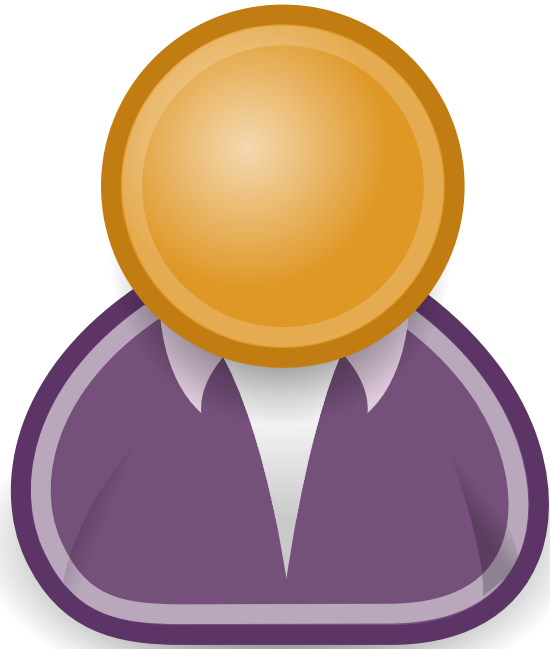
1. Die Miniprogramme 

2. Die Kurationspipeline 

3. Die Tests 

4. Der Makefile 

Rolle des Kurators



- gleiche Rolle und Verantwortung
- grundlegend veränderte tägliche Arbeit
- nicht mehr Daten manuell kuratieren
- dokumentieren, was kuratiert sein soll, in Form von Code

Teil 3

Pro und Contra

Gemütsruhe

- Um welche Dateien geht es?
 - › Wo sind sie?
 - › Wie rufe ich sie ab?
- Sind die Daten vollständig? Korrupt? Richtige Version?
- Welche Tools brauche ich? Version?
 - › Wo finde ich sie?
 - › Wie benutzt man sie?
- Wie kuratiert man die Daten eigentlich? Was muss ich tun?

Nachteile (oder: die Kosten)

- Man muss viel Code schreiben...
- ...und sogar in verschiedenen Programmiersprachen.
- Experimentieren wird etwas schwieriger.
- Viele Sachen, die der Kurator im Kopf hat muss er explizit dokumentieren.
- ...aber gute Dokumentation muss man sowieso aufschreiben.
(und »Working code is the best documentation of the curation process and activities« [Barabucci, seit Jahren])

Vorteile

- Robustheit gegen neue Daten und Fehler
- Gute/bessere Dokumentation/Forschungsdatenmanagement „umsonst“
- Nachvollziehbarkeit und Transparenz
- Testbarkeit
- Kohärenz über Jahre und in großen Arbeitsgruppen
- Nachvollziehbare Evolution der Kurationsprozess

Funktionale und deklarative Programmierung-basierte Methode für nachhaltige, reproduzierbare und verifizierbare Datenkuration

Vielen Dank

Gioele Barabucci

gioele.barabucci@uni-koeln.de

Cologne Center for eHumanities



Literatur / Verwandte Arbeiten

- G. Barabucci, F. Fischer (2017). „The formalization of textual criticism: bridging the gap between automated collation and edited critical texts“
- R. Crowley, R. Jonathan, J. Schäuble (2017). open-editions/corpus-joyce-ulysses-tei: Zenodo release (Version v0.1.1). Zenodo. 10.5281/zenodo.583139
- E. Dolstra, A. Löh (2008). „NixOS: A purely functional Linux distribution“
- P. Duvall, S. Matyas, A. Glover (2007). „Continuous integration: improving software quality and reducing risk“
- J. Flanders, T. Muñoz (2017). „An Introduction to Humanities Data Curation“
- R. Peng (2009). „Reproducible research and biostatistics“
- J. Reeve (2016). „Git-Lit: an Application of Distributed Version Control Technology toward the Creation of 50,000 Digital Scholarly Editions“
- S. Schoen, M. Perry (2014). „Why and how of reproducible builds: Distrusting our own infrastructure for safer software releases“
- N. Walsh (2007). „XProc: An XML pipeline language“
- J. van Zundert (2016). „Close Reading and Slow Programming — Computer Code as Digital Scholarly Edition.“