



# Metadatenworkflows mit Metafacture erstellen und verwalten

Fabian Steeg & Tobias Bülte

Offene Infrastruktur, Hochschulbibliothekszenrum NRW (hbz)

WWW,  
Mai 2022

<https://slides.lobid.org/2022-05-metafacture-workshop/> (PDF)



# Agenda

1. Einführung & Kontext
2. Workflows
3. Transformieren
4. Analysieren
5. Zusammenführen
6. Ausblick, Fragen & Diskussion

# 1. Kontext: Datentransformation

# Anwendungsszenarien

# Anwendungsszenarien

*Datenanalyse, z.B. Feldabdeckung im Katalog*

# Anwendungsszenarien

*Datenanalyse, z.B. Feldabdeckung im Katalog*

*Datenaufbereitung, z.B. zur Visualisierung mit Kibana*

# Anwendungsszenarien

*Datenanalyse*, z.B. Feldabdeckung im Katalog

*Datenaufbereitung*, z.B. zur Visualisierung mit Kibana

*Datenanreicherung*, z.B. Ergänzung von Daten aus Wikidata

# Anwendungsszenarien

*Datenanalyse*, z.B. Feldabdeckung im Katalog

*Datenaufbereitung*, z.B. zur Visualisierung mit Kibana

*Datenanreicherung*, z.B. Ergänzung von Daten aus Wikidata

*Datenaggregation* aus unterschiedlichen Quellen, z.B. **OERSI**



# Anwendungsszenarien

*Datenanalyse*, z.B. Feldabdeckung im Katalog

*Datenaufbereitung*, z.B. zur Visualisierung mit Kibana

*Datenanreicherung*, z.B. Ergänzung von Daten aus Wikidata

*Datenaggregation* aus unterschiedlichen Quellen, z.B. **OERSI**

*Systemmigration*, z.B. nach Alma oder Folio

# Wo wir Metafactory nutzen

# Wo wir Metafacture nutzen

Transformation der Daten des Verbundkatalogs für die  
Indexierung ([lobid.org](https://lobid.org))

# Wo wir Metafacture nutzen

Transformation der Daten des Verbundkatalogs für die  
Indexierung ([lobid.org](https://lobid.org))

Metadaten aus verschiedenen Quellen im OER Suchindex  
aggregieren ([OERSI](https://oersi.org))

# Wo wir Metafacture nutzen

Transformation der Daten des Verbundkatalogs für die  
Indexierung ([lobid.org](https://lobid.org))

Metadaten aus verschiedenen Quellen im OER Suchindex  
aggregieren ([OERSI](https://oersi.org))

Transformation der Daten der Rheinland-Pfälzischen  
Bibliographie

# Nur ein kleiner Teil

# Nur ein kleiner Teil

Sehr vielseitiges Tool mit vielen Anwendungsmöglichkeiten,  
Formaten, Modulen

# Nur ein kleiner Teil

Sehr vielseitiges Tool mit vielen Anwendungsmöglichkeiten,  
Formaten, Modulen

Wir nutzen davon nur einen kleinen Teil, eben für unsere  
Formate und Anwendungsfälle



# Nur ein kleiner Teil

Sehr vielseitiges Tool mit vielen Anwendungsmöglichkeiten,  
Formaten, Modulen

Wir nutzen davon nur einen kleinen Teil, eben für unsere  
Formate und Anwendungsfälle

Wir versuchen euch hier einen möglichst breiten Überblick  
zu geben über Konzepte und Möglichkeiten

# Annahmen

# Annahmen

Transformationen von Metadaten gehören zum täglichen Geschäft wissenschaftlicher Bibliotheken.

# Annahmen

Transformationen von Metadaten gehören zum täglichen Geschäft wissenschaftlicher Bibliotheken.

Es gibt viele unterschiedliche Methoden, die meist Programmierkenntnisse voraussetzen.

# Annahmen

Transformationen von Metadaten gehören zum täglichen Geschäft wissenschaftlicher Bibliotheken.

Es gibt viele unterschiedliche Methoden, die meist Programmierkenntnisse voraussetzen.

Datentransformationen werden meist im Zusammenspiel von Fachabteilungen & IT umgesetzt, verbunden mit größerem Kommunikationsaufwand.

# Annahmen

Transformationen von Metadaten gehören zum täglichen Geschäft wissenschaftlicher Bibliotheken.

Es gibt viele unterschiedliche Methoden, die meist Programmierkenntnisse voraussetzen.

Datentransformationen werden meist im Zusammenspiel von Fachabteilungen & IT umgesetzt, verbunden mit größerem Kommunikationsaufwand.

Bereits existierende, von anderen entwickelte Transformationsprozesse können nur bedingt entdeckt und nachgenutzt werden.

Das heißt:

# Das heißt:

Es gibt großes Potential, eine immer wiederkehrende Arbeit zugänglicher, kollaborativer und effizienter zu gestalten.



# Übergeordnete Ziele

# Übergeordnete Ziele

Ermächtigung der Fachebene zur Konfiguration von  
Datentransformationen

# Übergeordnete Ziele

Ermächtigung der Fachebene zur Konfiguration von  
Datentransformationen

Förderung von Praktiken zum Teilen und Auffinden von  
Transformationsprozessen

# Was ist Metafacture?

# Was ist Metafacture?

Ein vielseitiges Werkzeug zur Verarbeitung von semi-strukturierten Daten mit dem Fokus auf Bibliotheksdaten

# Was ist Metafacture?

Ein vielseitiges Werkzeug zur Verarbeitung von semi-strukturierten Daten mit dem Fokus auf Bibliotheksdaten  
nutzbar als Kommandozeilentool, als Java/JVM library,

# Was ist Metafacture?

Ein vielseitiges Werkzeug zur Verarbeitung von semi-strukturierten Daten mit dem Fokus auf Bibliotheksdaten  
nutzbar als Kommandozeilentool, als Java/JVM library,  
für Batch-Verarbeitung oder on-the-fly

# Was ist Metafacture?

Ein vielseitiges Werkzeug zur Verarbeitung von semi-strukturierten Daten mit dem Fokus auf Bibliotheksdaten

nutzbar als Kommandozeilentool, als Java/JVM library,

für Batch-Verarbeitung oder on-the-fly

offenes Framework: Weiterentwicklung, Wiederverwendung und Austausch (von einzelnen Modulen und ganzen Workflows)



# Metafactory-Historie

# Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph; damals schon Austausch mit dem hbz

# Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph; damals schon Austausch mit dem hbz

2013: Umzug auf GitHub, Open-Source-Projekt geworden

# Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph; damals schon Austausch mit dem hbz

2013: Umzug auf GitHub, Open-Source-Projekt geworden

2019: Mit der Zeit immer weniger DNB-Ressourcen für Metafactory, hbz wird Maintainer

# Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph; damals schon Austausch mit dem hbz

2013: Umzug auf GitHub, Open-Source-Projekt geworden

2019: Mit der Zeit immer weniger DNB-Ressourcen für Metafactory, hbz wird Maintainer

2019: Start von Metafactory Fix

# Metafactory-Historie

2011: Start der Entwicklung durch DNB im Rahmen von Culturegraph; damals schon Austausch mit dem hbz

2013: Umzug auf GitHub, Open-Source-Projekt geworden

2019: Mit der Zeit immer weniger DNB-Ressourcen für Metafactory, hbz wird Maintainer

2019: Start von Metafactory Fix

2021: Start von Metafactory Playground

# Wie Metafactory funktioniert

# Wie Metafactory funktioniert

Grundidee: Daten fließen durch mehrere Module:

→ read → decode → transform → encode → write →



# Wie Metafactory funktioniert

Grundidee: Daten fließen durch mehrere Module:

→ read → decode → transform → encode → write →

Jedes Modul erwartet Input eines bestimmten Typs und erzeugt Output eines bestimmten Typs

# Wie Metafacture funktioniert

Grundidee: Daten fließen durch mehrere Module:

→ read → decode → transform → encode → write →

Jedes Modul erwartet Input eines bestimmten Typs und erzeugt Output eines bestimmten Typs

Verschiedene Formate werden unterstützt (z.B. PICA, MARC), erweiterbares Framework für eigene Formate

# Wie Metafacture funktioniert

Grundidee: Daten fließen durch mehrere Module:

→ read → decode → transform → encode → write →

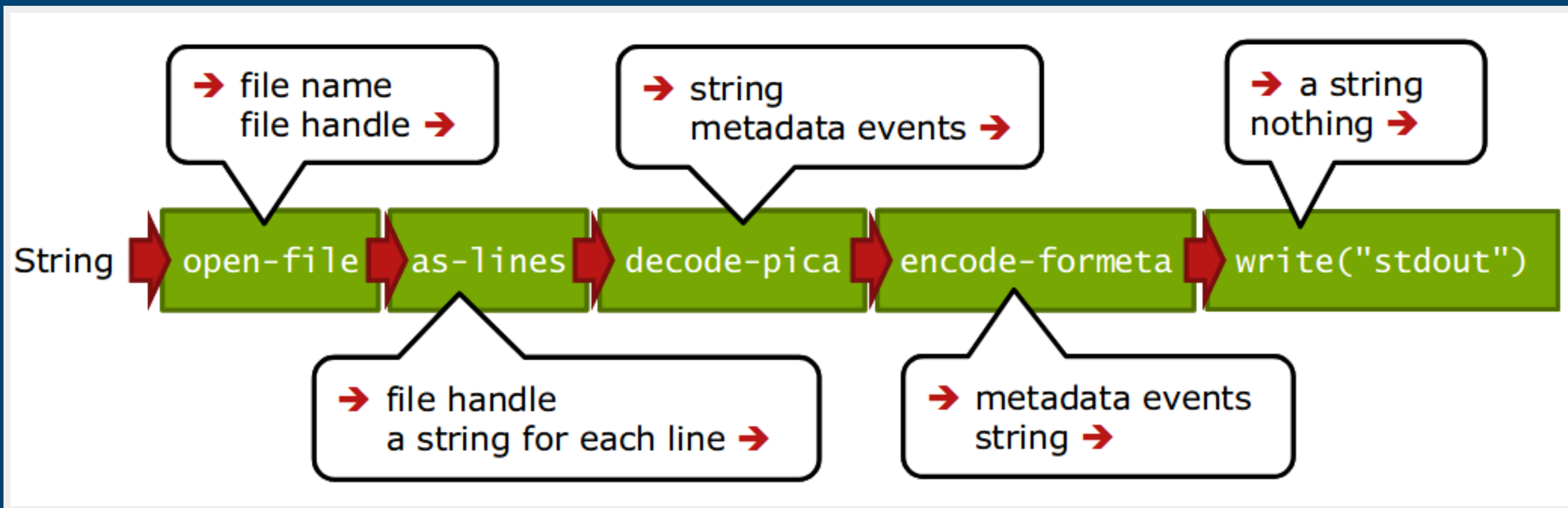
Jedes Modul erwartet Input eines bestimmten Typs und erzeugt Output eines bestimmten Typs

Verschiedene Formate werden unterstützt (z.B. PICA, MARC), erweiterbares Framework für eigene Formate

Durch Kombination einzelner Module, durch die unsere Daten fließen, bauen wir den Workflow

# 2. Workflows

# Ein Workflow



Aus: Christoph Böhme, [http://swib.org/swib13/slides/boehme\\_swib13\\_131.pdf](http://swib.org/swib13/slides/boehme_swib13_131.pdf)

# Workflows konfigurieren und ausführen

# Workflows konfigurieren und ausführen

Workflows können in Flux (einer speziellen Konfigurationssprache) oder mit Java (typischer über Java Generics) bearbeitet werden

# Workflows konfigurieren und ausführen

Workflows können in Flux (einer speziellen Konfigurationssprache) oder mit Java (typischer über Java Generics) bearbeitet werden

Flux-Workflows können in einem Texteditor editiert und auf der Kommandozeile ausgeführt werden, Java-Workflows funktionieren wie andere Java-Komponenten



# Workflows konfigurieren und ausführen

Workflows können in Flux (einer speziellen Konfigurationssprache) oder mit Java (typischer über Java Generics) bearbeitet werden

Flux-Workflows können in einem Texteditor editiert und auf der Kommandozeile ausgeführt werden, Java-Workflows funktionieren wie andere Java-Komponenten

Der Workshop führt in die Nutzung der Flux-Workflows ein, zum Ausführen verwenden wir den Metafactory Playground

# Metafactory Playground

# Metafactory Playground

Webbasierte Oberfläche zum Ausprobieren und  
Austauschen von Workflows

# Metafactory Playground

Webbasierte Oberfläche zum Ausprobieren und  
Austauschen von Workflows

Ziel: Einstiegshürde für Metafactory senken, unserer  
Erfahrung nach ein zentrales Problem bei der Metafactory-  
Nutzung

# Metafactory Playground

Webbasierte Oberfläche zum Ausprobieren und  
Austauschen von Workflows

Ziel: Einstiegshürde für Metafactory senken, unserer  
Erfahrung nach ein zentrales Problem bei der Metafactory-  
Nutzung

Für Entwicklung, Dokumentation, Tutorials, Workshops

# Metafactory Playground

Webbasierte Oberfläche zum Ausprobieren und  
Austauschen von Workflows

Ziel: Einstiegshürde für Metafactory senken, unserer  
Erfahrung nach ein zentrales Problem bei der Metafactory-  
Nutzung

Für Entwicklung, Dokumentation, Tutorials, Workshops

<https://metafactory.org/playground>

Wie funktioniert das in der Praxis? Lasst uns das  
gemeinsam ausprobieren.

(die folgenden Screenshots verlinken die Beispiele zum Ausprobieren im Playground)

▼ Load Examples Clear **▶ Process** ↻ Share ⬆ Import Workflow ⬇ Export Workflow

Data

```
1 001@ a501-2001A 01100:15-10-94001B 09999:12-06-06t16:10:17.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aag003@ 0482147
2 001@ 01a5001A 01140:08-12-99001B 09999:05-01-08t22:57:29.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aa003@ 0958090564
3 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U 0utf8001X 00002@ 0Aa1003@ 036180954
4 001@ a501-3001A 01240:01-08-95001B 09999:24-09-10t17:42:20.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Af003@ 09451840
5 001@ 01-2a5001A 01239:18-08-11001B 09999:05-09-11t23:31:44.000001D 01240:30-08-11001U 0utf8001X 00002@ 0Af003@ 01014417
```

Flux

```
1 PG_DATA | as-lines | decode-pica | encode-json | print ;
```

Fix Morph

Result

```
{"001@":{"a":"5","0":"1-2"},"001A":{"0":"1100:15-10-94"},"001B":{"0":"9999:12-06-06","t":"16:10:17.000"},"001D":{"0":"9999:99-99-99"},"001U":{"0":"utf8"},"001X":{"0":"0"},"002@":{"0":"Aag"},"003@":{"0":"482147350"},"006U":{"0":"94,P05"},"007E":{"0":"U 70.16407"},"007I":{"S":"o","0":"74057548"},"011@":{"a":"1970"},"017A":{"a":"rh"},"021A":{"a":"Die @Berufsfreiheit der Arbeitnehmer und ihre Ausgestaltung in völkerrechtlichen Verträgen","d":"Eine Grundrechtsbetrachtg"},"028A":{"9":"106884905","7":"Tn3","A":"gnd","0":"106884905","a":"Projahn","d":"Horst D."},"033A":{"p":"Würzburg"},"034D":{"a":"XXXVIII, 165 S."},"034I":{"a":"8"},"037C":{"a":"Würzburg, Jur. F., Diss. v. 7. Aug. 1970"}}
```



# Übung: Formatierung & Optionen

Playground-Beispiel anpassen:

```
PG_DATA  
| as-lines  
| decode-pica  
| encode-json(prettyPrinting = "true")  
| print  
;
```

▼ Load Examples Clear **▶ Process** ↻ Share ⬆ Import Workflow ⬇ Export Workflow

Data

```
1 001@ a501-2001A 01100:15-10-94001B 09999:12-06-06t16:10:17.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aag003@ 0482147
2 001@ 01a5001A 01140:08-12-99001B 09999:05-01-08t22:57:29.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aa003@ 095809056
3 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U Outf8001X 00002@ 0Aal003@ 036180957
4 001@ a501-3001A 01240:01-08-95001B 09999:24-09-10t17:42:20.000001D 09999:99-99-99001U Outf8001X 00002@ 0Af003@ 09451840
5 001@ 01-2a5001A 01239:18-08-11001B 09999:05-09-11t23:31:44.000001D 01240:30-08-11001U Outf8001X 00002@ 0Af003@ 01014417
```

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | encode-json(prettyPrinting = "true")
5 | print
6 ;
```

Fix Morph

Result

```
{
  "001@":{
    "a":"5",
    "O":"1-2"
  },
  "001A":{
    "O":"1100:15-10-94"
  },
}
```

# Dokumentation

# Dokumentation

Was für Module gibt es? Was machen die?

# Dokumentation

Was für Module gibt es? Was machen die?

<https://github.com/metafactory/metafactory-documentation/blob/master/flux-commands.md>

# Dokumentation

Was für Module gibt es? Was machen die?

<https://github.com/metafactory/metafactory-documentation/blob/master/flux-commands.md>

z.B. statt JSON als "Formeta" ausgeben, formatiert

# encode- formmeta(style='multiline')

```
'482147350' {  
  '001@' {  
    'a': '5',  
    '0': '1-2'  
  },  
  '001A' {  
    '0': '1100:15-10-94'  
  },  
  '001B' {  
    '0': '9999:12-06-06',  
    't': '16:10:17.000'  
  },  
  '001D' {  
    '0': '9999:99-99-99'  
  },  
  '001H' {
```

<https://github.com/hbz/metafacture-flux-examples/tree/master/sample1>

# 3. Transformieren



# Transformieren

# Transformieren

Manipulation von Feldnamen und -werten; filtern, kombinieren, trennen, normalisieren etc.

# Transformieren

Manipulation von Feldnamen und -werten; filtern, kombinieren, trennen, normalisieren etc.

Änderung der Struktur und Hierarchie eines Records etc.

# Transformieren

Manipulation von Feldnamen und -werten; filtern, kombinieren, trennen, normalisieren etc.

Änderung der Struktur und Hierarchie eines Records etc.

Feldwerte aus Lookup-Tabellen in externen Dateien (z.B. Freitextfelder -> kontrollierte Vokabulare)

# Transformationsmodul

# Transformationsmodul

Morph: XML-basiert, Feld- / Metadaten-Event-Ebene

# Transformationsmodul

Morph: XML-basiert, Feld- / Metadaten-Event-Ebene

Fix: eigene, Catmandu-Fix-artige Sprache, Record-basiert

# Metafactory Fix: Ziele



# Metafactory Fix: Ziele

Erleichterung der Transformationskonfiguration

# Metafactory Fix: Ziele

Erleichterung der Transformationskonfiguration

Anknüpfung an existierende Konfigurationssprache aus  
**Catmandu** (mittelfristiges Ziel: Standardisierung, s.  
<https://github.com/elag/FIG>)

# Metafactory Fix: Ziele

Erleichterung der Transformationskonfiguration

Anknüpfung an existierende Konfigurationssprache aus  
**Catmandu** (mittelfristiges Ziel: Standardisierung, s.  
<https://github.com/elag/FIG>)

Vergrößerung der Zielgruppe um Bibliothekar:innen und  
andere Metadatenfachleute (bei uns z.B. in OERSI, erster  
Anwendungsfall und Entwicklungsbegleitung zu Fix)

Load Examples Clear **Process** Share Import Workflow Export Workflow

Data

```
1 001@ a501-2001A 01100:15-10-94001B 09999:12-06-06t16:10:17.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aag003@ 0482147
2 001@ 01a5001A 01140:08-12-99001B 09999:05-01-08t22:57:29.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aa003@ 095809056
3 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U Outf8001X 00002@ 0Aal003@ 036180957
4 001@ a501-3001A 01240:01-08-95001B 09999:24-09-10t17:42:20.000001D 09999:99-99-99001U Outf8001X 00002@ 0Af003@ 09451840
5 001@ 01-2a5001A 01239:18-08-11001B 09999:05-09-11t23:31:44.000001D 01240:30-08-11001U Outf8001X 00002@ 0Af003@ 01014417
```

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pipe
4 | fix("retain('021A')")
5 | encode-json(prettyPrinting = "true")
6 | print;
```

Fix Morph

```
1
```

Result

```
{
  "021A":{
    "a": "Die @Berufsfreiheit der Arbeitnehmer und ihre Ausgestaltung in völkerrechtlichen Verträgen",
    "d": "Eine Grundrechtsbetrachtg"
  }
}
```

Load Examples Clear **Process** Share Import Workflow Export Workflow

Data

```
1 001@ a501-2001A 01100:15-10-94001B 09999:12-06-06t16:10:17.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aag003@ 0482147
2 001@ 01a5001A 01140:08-12-99001B 09999:05-01-08t22:57:29.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aa003@ 0958090564
3 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U 0utf8001X 00002@ 0Aal003@ 036180957
4 001@ a501-3001A 01240:01-08-95001B 09999:24-09-10t17:42:20.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Af003@ 09451840
5 001@ 01-2a5001A 01239:18-08-11001B 09999:05-09-11t23:31:44.000001D 01240:30-08-11001U 0utf8001X 00002@ 0Af003@ 01014417
```

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | fix
5 | encode-json(prettyPrinting = "true")
6 | print;
```

Fix

Morph

```
1 retain(['021A'])
```

Result

```
{
  "021A": {
    "a": "Die @Berufsfreiheit der Arbeitnehmer und ihre Ausgestaltung in völkerrechtlichen Verträgen",
    "d": "Eine Grundrechtsbetrachtg"
  }
}
```

# Metafacture Fix: Pfade

# Metafactory Fix: Pfade

Um die verschiedenen Elemente und Felder für die Transformation anzuwählen, muss man ihre Pfade angeben

# Metafactory Fix: Pfade

Um die verschiedenen Elemente und Felder für die Transformation anzuwählen, muss man ihre Pfade angeben

Einfache Elemente der obersten Ebene / Felder: `id`



# Metafacture Fix: Pfade

Um die verschiedenen Elemente und Felder für die Transformation anzuwählen, muss man ihre Pfade angeben

Einfache Elemente der obersten Ebene / Felder: ``id``

Elemente auf einer unteren Ebene / Unterfelder:  
``title.subtitle``

# Metafactory Fix: Pfade

Um die verschiedenen Elemente und Felder für die Transformation anzuwählen, muss man ihre Pfade angeben

Einfache Elemente der obersten Ebene / Felder: ``id``

Elemente auf einer unteren Ebene / Unterfelder:  
``title.subtitle``

Wiederholte Felder werden als Listen mit Index-Nummer angegeben: ``creator.1.name.firstName``

# Übung

# Übung

z.B. Titel, Verlag, Erscheinungsort und -jahr aus den PICA-Feldern 021A.a, 033A.n, 033A.p, 011@.n verwenden

# Übung

z.B. Titel, Verlag, Erscheinungsort und -jahr aus den PICA-Feldern 021A.a, 033A.n, 033A.p, 011@.n verwenden

Verlag und Erscheinungsort sollen in einem neuen Feld kombiniert werden

# Übung

z.B. Titel, Verlag, Erscheinungsort und -jahr aus den PICA-Feldern 021A.a, 033A.n, 033A.p, 011@.n verwenden

Verlag und Erscheinungsort sollen in einem neuen Feld kombiniert werden

Als Feldnamen wollen wir sprechende, nicht-numerische Bezeichnungen haben

# Übung

z.B. Titel, Verlag, Erscheinungsort und -jahr aus den PICA-Feldern 021A.a, 033A.n, 033A.p, 011@.n verwenden

Verlag und Erscheinungsort sollen in einem neuen Feld kombiniert werden

Als Feldnamen wollen wir sprechende, nicht-numerische Bezeichnungen haben

z.B. mit `move_field`, `paste`, `retain`, s. Dokumentation

move\_field, paste, retain



# move\_field, paste, retain

```
1{a: Faust, b {n: Goethe, v: JW}, c: Weimar}  
2{a: Räuber, b {n: Schiller, v: F}, c: Weimar}
```

# move\_field, paste, retain

```
1{a: Faust, b {n: Goethe, v: JW}, c: Weimar}  
2{a: Räuber, b {n: Schiller, v: F}, c: Weimar}
```

```
move_field(a, title)  
paste(author, b.v, b.n, '~aus', c)  
retain(title, author)
```

# move\_field, paste, retain

```
1{a: Faust, b {n: Goethe, v: JW}, c: Weimar}  
2{a: Räuber, b {n: Schiller, v: F}, c: Weimar}
```

```
move_field(a, title)  
paste(author, b.v, b.n, '~aus', c)  
retain(title, author)
```

```
{"title": "Faust", "author": "JW Goethe aus Weimar"}  
{"title": "Räuber", "author": "F Schiller aus Weimar"}
```

Beispiel im Playground

Load Examples Clear **Process** Share Import Workflow Export Workflow

Data

```
1 001@ 01a5001A 01140:08-12-99001B 09999:05-01-08t22:57:29.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aa003@ 095809056Z
2 001@ a501001A 01140:19-02-03001B 09999:19-06-11t01:20:13.000001D 09999:26-04-03001U 0utf8001X 00002@ 0Aa1003@ 03618095Z
3 001@ 02a5001A 01200:24-11-77001B 09999:16-01-08t01:01:54.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aa003@ 07805101Z
4 001@ a501-2001A 01140:28-06-99001B 09999:17-04-11t14:40:50.000001D 09999:99-99-99001U 0utf8001X 00002@ 0Aa003@ 0956745Z
5 001@ 01a5001A 09999:08-03-02001B 09999:12-03-04t11:47:27.000001D 09999:08-03-02001U 0utf8001X 00002@ 0Aa1003@ 05761214Z
```

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | fix
5 | encode-json(prettyPrinting = "true")
6 | print
7 ;
```

Fix Morph

```
1 move_field('021A.a', 'Title')
2 move_field('011@a', Year)
3 paste('Publisher', '033A.n', '~: ', '033A.p')
4 retain(Title, Year, Publisher)
```

Result

```
{
  "Title": "Zukunft Bildung",
  "Year": "1999",
  "Publisher": "Polit. Akad.: Wien"
}
```

# Lookup

# Lookup

"Feldwerte aus Lookup-Tabellen in externen Dateien (z.B. Freitextfelder -> kontrollierte Vokabulare)"

# Übung

# Übung

Feld 002@.0 → dcterms:format



# Übung

Feld 002@.0 → dcterms:format

In 002@.0, Position 1, A: print, B: audiovisual, O: online

# Übung

Feld 002@.0 → dcterms:format

In 002@.0, Position 1, A: print, B: audiovisual, O: online

z.B. mit `copy_field`, `substring`, `lookup`, `retain`, `s`.

Dokumentation

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | fix
5 | encode-json(prettyPrinting = "true")
6 | print;
```

Fix

Morph

```
1 copy_field('002@.0', 'dcterms:format')
2 substring('dcterms:format', '0', '1')
3 lookup('dcterms:format', A: print, B: audiovisual, O: online)
4 retain('002@', 'dcterms:format')
```

Result

```
{
  "002@":{
    "0": "Aa"
  },
  "dcterms:format": "print"
}
{
  "002@":{
    "0": "AaI"
  },
  "dcterms:format": "print"
}
```

# Export & Import

# Export & Import

Workflow kann zur lokalen Ausführung aus dem Playground exportiert werden (Dateien-Download)

# Export & Import

Workflow kann zur lokalen Ausführung aus dem Playground exportiert werden (Dateien-Download)

Flux-, Fix- und Daten-Datei können lokal bearbeitet und per Kommandozeile ausgeführt werden

# Export & Import

Workflow kann zur lokalen Ausführung aus dem Playground exportiert werden (Dateien-Download)

Flux-, Fix- und Daten-Datei können lokal bearbeitet und per Kommandozeile ausgeführt werden

Dateien können dann auch wieder in den Playground importiert werden (Dateien-Upload)

# Export & Import

Workflow kann zur lokalen Ausführung aus dem Playground exportiert werden (Dateien-Download)

Flux-, Fix- und Daten-Datei können lokal bearbeitet und per Kommandozeile ausgeführt werden

Dateien können dann auch wieder in den Playground importiert werden (Dateien-Upload)

<https://github.com/metafactory/metafactory-fix/releases>



▼ sample3x Lookup fields

 Clear

 Process

 Share

 Import Workflow

 Export Workflow

Data

▼

Flux

1 PG\_DATA

2 | as-lines

3 | decode-pica

4 | fix

5 | encode-json(prettyPrinting="true")

6 | print;

Fix Morph

1 copy\_field('002@.0', 'dterms:format')

2 substring('dterms:format', '0', '1')

3 lookup('dterms:format', A: print, B: audiovisual, 0: online)

4 retain('002@', 'dterms:format')

# Rückschau

# Rückschau

Workflows mit Flux

# Rückschau

Workflows mit Flux

Transformationen mit Fix

# Rückschau

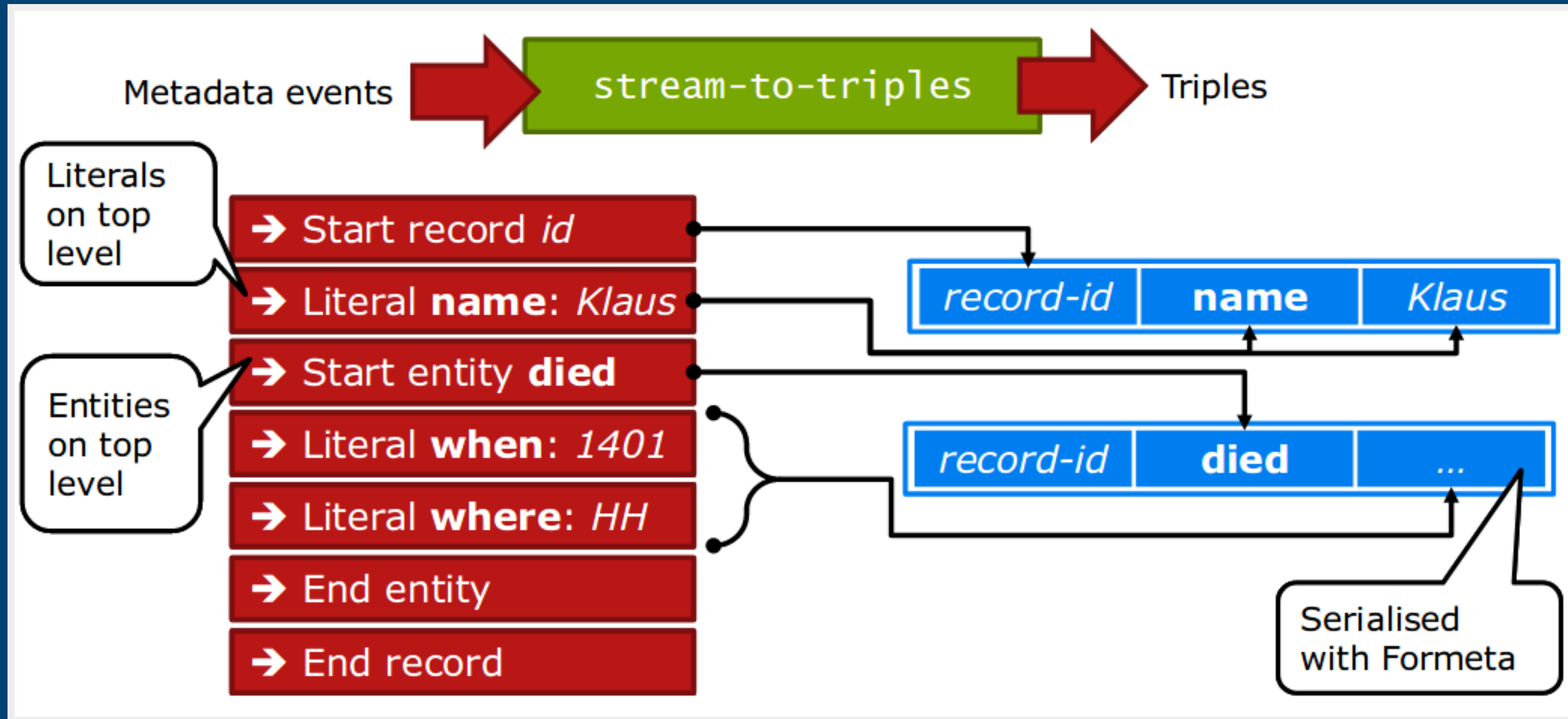
Workflows mit Flux

Transformationen mit Fix

Playground, Export, Import

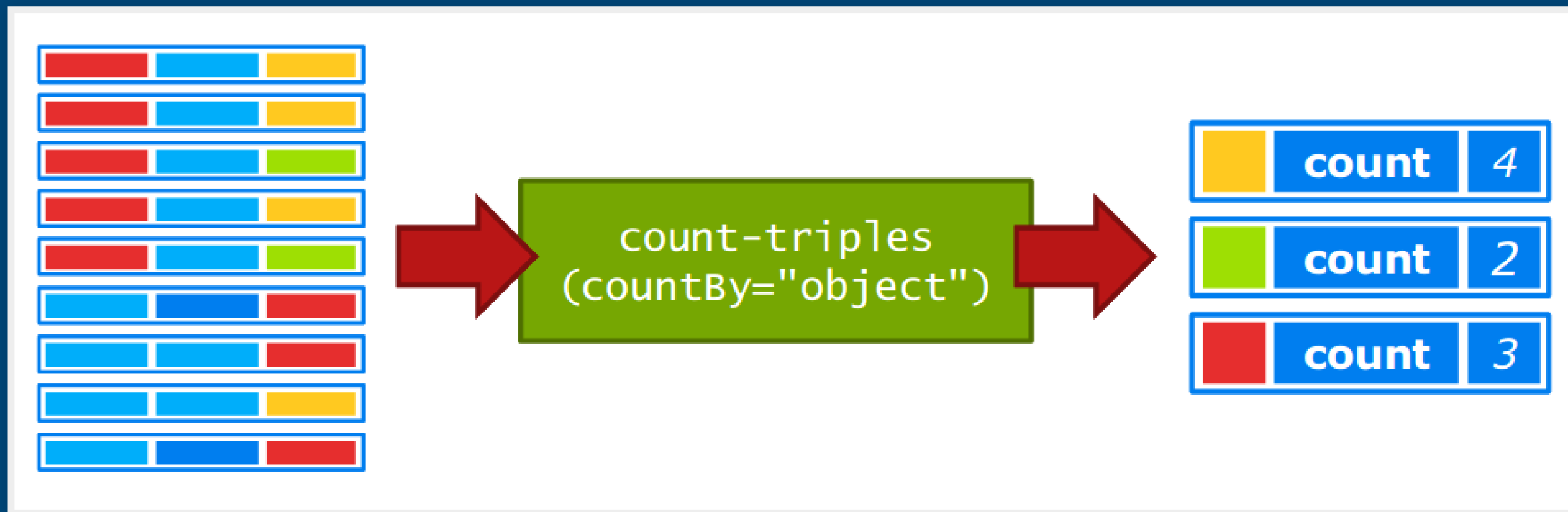
# 4. Analysieren

# stream-to-triples



Source: Christoph Böhme, [http://swib.org/swib13/slides/boehme\\_swib13\\_131.pdf](http://swib.org/swib13/slides/boehme_swib13_131.pdf)

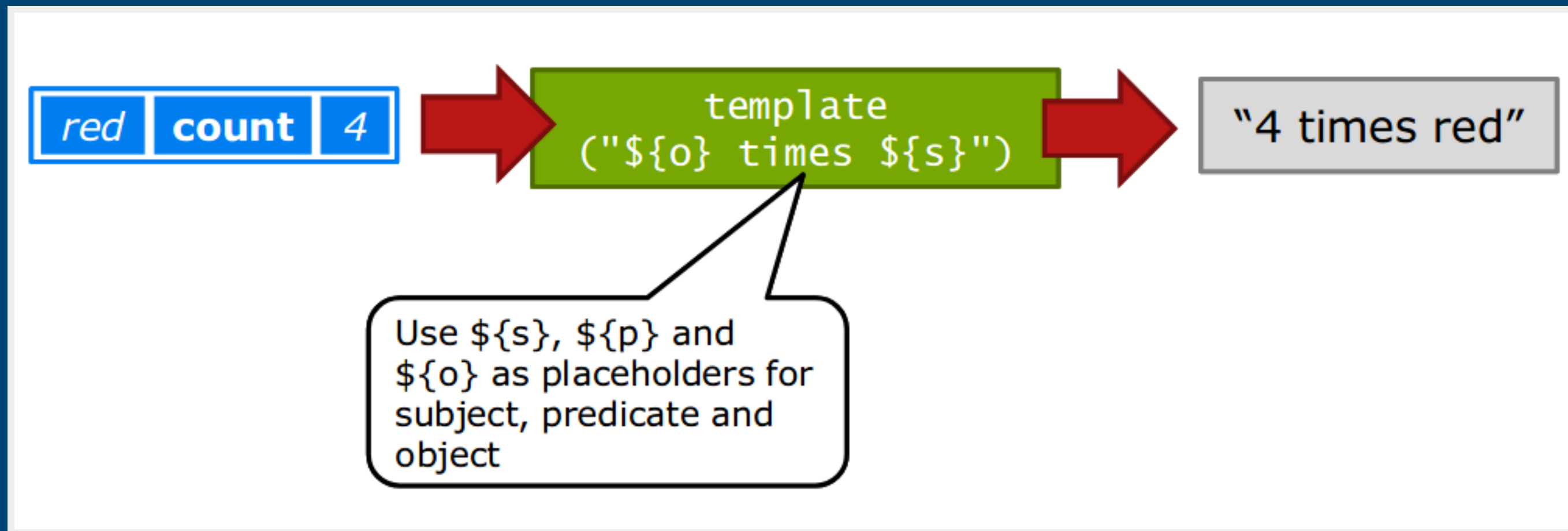
# count-triples



Source: Christoph Böhme, [http://swib.org/swib13/slides/boehme\\_swib13\\_131.pdf](http://swib.org/swib13/slides/boehme_swib13_131.pdf)



# template



Source: Christoph Böhme, [http://swib.org/swib13/slides/boehme\\_swib13\\_131.pdf](http://swib.org/swib13/slides/boehme_swib13_131.pdf)

# Beispiel: Feldwerte zählen

# Beispiel: Feldwerte zählen

Anzahl unterschiedlicher Werte im Edition-Feld (032@.a)

# Beispiel: Feldwerte zählen

Anzahl unterschiedlicher Werte im Edition-Feld (032@.a)

```
move_field('032@.a', 'Edition')  
retain('Edition')
```

# Beispiel: Feldwerte zählen

Anzahl unterschiedlicher Werte im Edition-Feld (032@.a)

```
move_field('032@.a', 'Edition')
retain('Edition')
```

```
PG_DATA
| as-lines
| decode-pica
| fix
| stream-to-triples
| count-triples(countBy = "object")
| template("${o} | ${s}")
| print
;
```

▼

Load Examples

✎

Clear

▶

Process

🔗

Share

📁

Import Workflow

📄

Export Workflow

Data

1

001@ 01a5001A 09999:07-04-01001B 09999:10-03-04t13:31:00.000001D 09999:07-04-01001U Outf8001X 00002@ 0Aal003@ 057034698

2

001@ 01-2a5001A 01200:28-09-87001B 09999:29-12-07t19:41:36.000001D 09999:99-99-99001U Outf8001X 00002@ 0Aan003@ 0871159

3

001@ 01a5001A 09999:31-10-01001B 00101:07-08-09t17:28:39.000001D 09999:31-10-01001U Outf8001X 00002@ 0Aal003@ 057273695

4

001@ a5001A 01145:29-03-04001B 01145:27-04-05t18:02:05.000001D 09999:10-05-08001U Outf8001X 00002@ 0Acl003@ 05605684876

5

001@ 01-2a5001A 01130:27-10-05001B 09999:23-11-05t10:44:17.000001D 01140:03-11-05001U Outf8001X 00002@ 0Af003@ 09768582

Flux

1

PG\_DATA

2

| as-lines

3

| decode-pica

4

| fix

5

| stream-to-triples

6

| count-triples(countBy = "object")

7

| template("\${o} | \${s}")

8

| print

9

;

Fix Morph

1

move\_field('032@a', 'Edition')

2

retain('Edition')

Result

1|(2. Aufl.)

1|3. Aufl.

1|[6. Aufl.]

# Daten per URL lesen

# Daten per URL lesen

Statt aus Datei lesen (oder PG\_DATA) auch per URL möglich



# Daten per URL lesen

Statt aus Datei lesen (oder PG\_DATA) auch per URL möglich

Statt ``"some.file" | open-file ...`` → ``"http://..." | open-http``

# Daten per URL lesen

Statt aus Datei lesen (oder PG\_DATA) auch per URL möglich

Statt ``"some.file" | open-file ...`` → ``"http://..." | open-http``

Aufgabe: voriges Beispiel, aber von URL lesen, was fällt auf?

# Daten per URL lesen

Statt aus Datei lesen (oder PG\_DATA) auch per URL möglich

Statt `"some.file" | open-file ...`` → `"http://..." | open-http``

Aufgabe: voriges Beispiel, aber von URL lesen, was fällt auf?

```
https://github.com/hbz/metafactory-flux-  
examples/blob/master/sample4/bib-data-1k.pica?  
raw=true
```

Flux

```
1 "https://github.com/hbz/metafactory-flux-examples/blob/master/sample4/bib-data-1k.pica?raw=true"
2 | open-http
3 | as-lines
4 | decode-pica
5 | fix
6 | stream-to-triples
7 | count-triples(countBy = "object")
8 | template("${o} | ${s}")
9 | print;
```

Fix

Morph

```
1 move_field('032@a', 'Edition')
2 retain('Edition')
```

Result

```
1|(16.-20. Taus.)
1|(Ausg. für die Volkswirtschaft), 3. Ausg., Stand der Unterlagen: 1981
1|(Versión rev.), 4. ed.
52|1. Aufl.
1|1. Aufl. d. Neuübers.
```

# Beispiel: Feldwerte zählen

```
1 | (1. Aufl.)
1 | (16.-20. Taus.)
2 | (2. Aufl.)
1 | (2. Aufl.) - 1:10 000
1 | (2. ed.)
1 | (3. rev. ed.)
1 | (Ausg. 1971/72)
1 | (Ausg. für die Volkswirtschaft), 1. Ausg., Stand der
Unterlagen: 1977
1 | (Ausg. für die Volkswirtschaft), 1. Ausg., Stand der
Unterlagen: 1986
1 | (Ausg. für die Volkswirtschaft), 1. Ausg., Stand der
Unterlagen: 1987
1 | (Ausg. für die Volkswirtschaft), 3. Ausg., Stand der
Unterlagen: 1980
2 | (Ausg. für die Volkswirtschaft), 2. Ausg., Stand der
```

<https://github.com/hbz/metafacture-flux-examples/tree/master/sample4>

# Übung: Muster zählen

# Übung: Muster zählen

Ausgabe der Anzahl von Geburtsjahren (060R.a) der Muster  
'yyyy' oder 'dd.mm.yyyy'

# Übung: Muster zählen

Ausgabe der Anzahl von Geburtsjahren (060R.a) der Muster  
'yyyy' oder 'dd.mm.yyyy'

Ausgabe der Anzahl der Werte in 060R.a, die beiden  
Mustern nicht entsprechen (Fehler)



# Übung: Muster zählen

Ausgabe der Anzahl von Geburtsjahren (060R.a) der Muster  
'yyyy' oder 'dd.mm.yyyy'

Ausgabe der Anzahl der Werte in 060R.a, die beiden  
Mustern nicht entsprechen (Fehler)

z.B. mit `if`, `elsif`, `any_match`, `move_field`, s. Dokumentation

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | fix
5 | stream-to-triples
6 | count-triples(countBy="predicate")
7 | template("${s}:\t ${o}")
8 | print;
```

Fix

Morph

```
1 if any_match('060R.a', '^\\d{2}\\.\\.\\d{2}\\.\\d{4}')
2 | move_field('060R.a', 'birth (full)')
3 elseif any_match('060R.a', '^\\d{2,4}')
4 | move_field('060R.a', 'birth (year)')
5 else
6 | move_field('060R.a', 'invalid birth')
7 end
8 retain('birth (full)', 'birth (year)', 'invalid birth')
9
```

Result

```
birth (year): 4
invalid birth: 1
```

Übung: was /  
warum invalid?

# Übung: was / warum invalid?

1. nur 'invalid birth' retainen

# Übung: was / warum invalid?

1. nur 'invalid birth' retainen
2. nicht zählen, Tripel nur ausgeben

# Übung: volles Datenset per URL

# Übung: volles Dataset per URL

Aufgabe: voriges Beispiel, aber von URL lesen:

# Übung: volles Dataset per URL

Aufgabe: voriges Beispiel, aber von URL lesen:

```
https://github.com/hbz/metafactory-flux-  
examples/blob/master/sample5/authority-data-  
1k.pica?raw=true
```



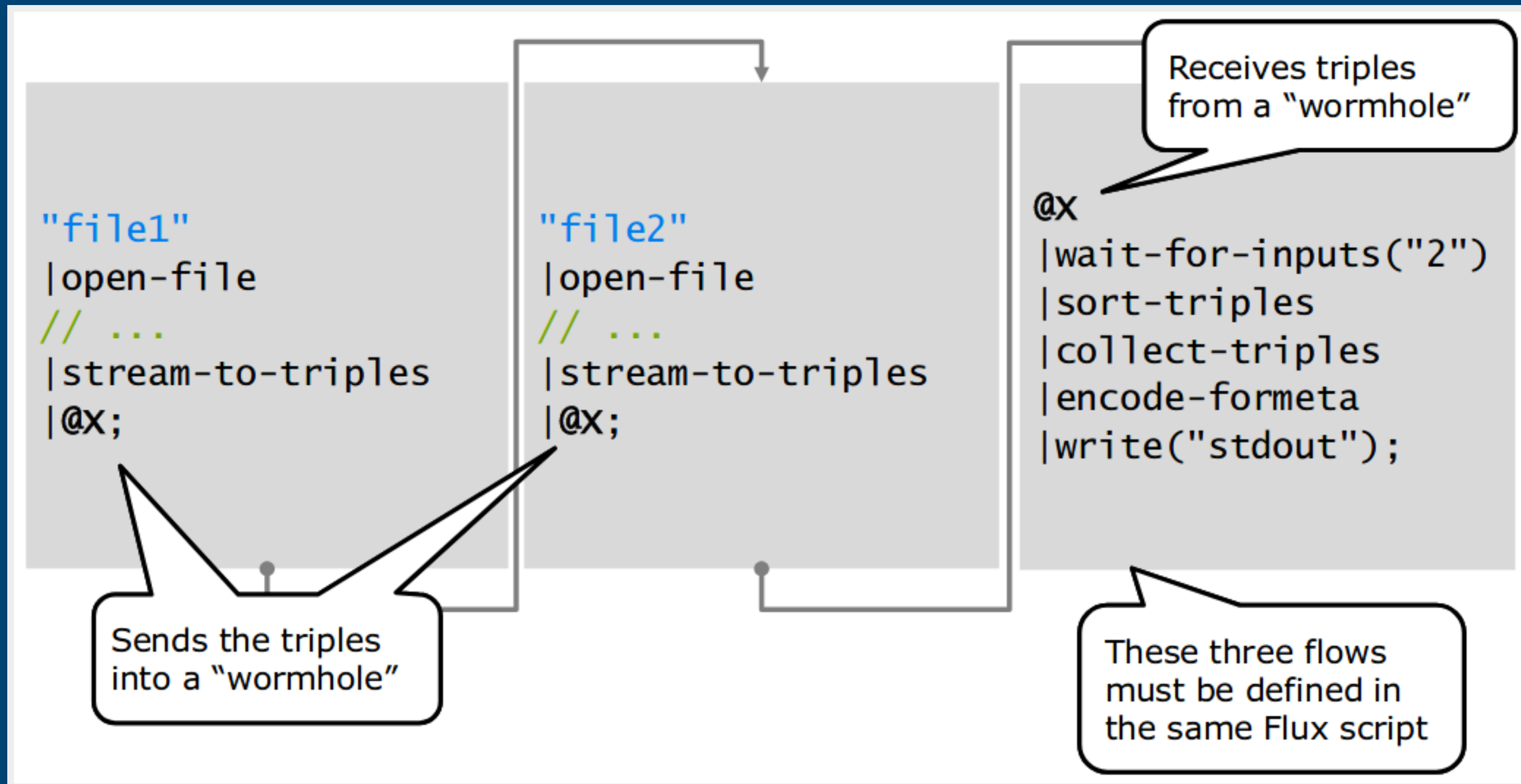
# Muster zählen, volle Daten, und Sterbejahre (060R.b)

```
birth (full):      873
birth (year):      4874
death (full):      460
death (year):      1587
invalid birth:      9
invalid death:      5
```

<https://github.com/hbz/metafacture-flux-examples/tree/master/sample5>

# 5. Zusammenführen

# Wurmloch



Aus: Christoph Böhme, [http://swib.org/swib13/slides/boehme\\_swib13\\_131.pdf](http://swib.org/swib13/slides/boehme_swib13_131.pdf)

# Unterschiedliche IDs

# Unterschiedliche IDs

Typischerweise haben Entitäten in unterschiedlichen  
Datenquellen unterschiedliche IDs

# Unterschiedliche IDs

Typischerweise haben Entitäten in unterschiedlichen  
Datenquellen unterschiedliche IDs

z.B. GND mit GND-ID, Wikipedia mit eigener ID, aber z.B.  
GND-ID in Feld 'gnd'

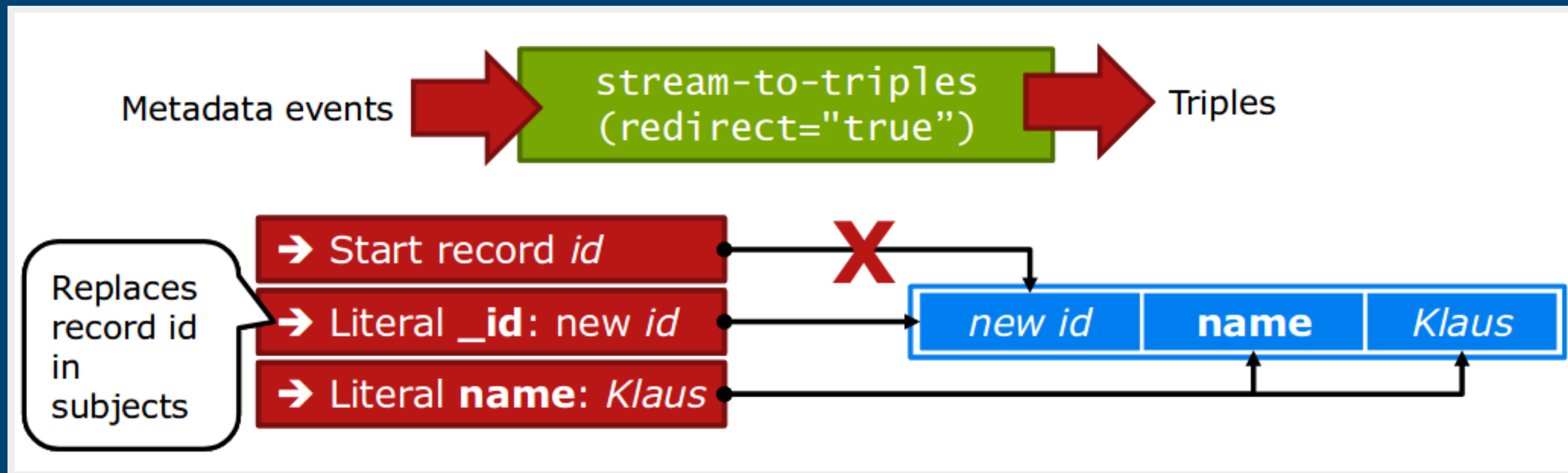
# Unterschiedliche IDs

Typischerweise haben Entitäten in unterschiedlichen Datenquellen unterschiedliche IDs

z.B. GND mit GND-ID, Wikipedia mit eigener ID, aber z.B. GND-ID in Feld 'gnd'

d.h. zum Zusammenführen brauchen wir einen Weg, die IDs aus einer der Datenquellen zu setzen

# redirect="true"



Source: Christoph Böhme, [http://swib.org/swib13/slides/boehme\\_swib13\\_131.pdf](http://swib.org/swib13/slides/boehme_swib13_131.pdf)



# Übung: Zusammenführen / Anreichern

Wir kombinieren GND-PICA-Daten (in PG\_DATA) mit Daten in  
wiki-persons.foma (enthalten GND-IDs im Feld 'gnd')

<https://github.com/hbz/metafacture-flux-examples/tree/master/sample6>

Flux

```
1 PG_DATA
2 | as-lines
3 | decode-pica
4 | stream-to-triples
5 | @X;
6
7 "https://raw.githubusercontent.com/hbz/metafacture-flux-examples/master/sample6/wiki-persons.foma"
8 | open-http
9 | as-lines
10 | decode-formeta
11 | fix
12 | stream-to-triples(redirect="true")
13 | @X;
14
15 @X
16 | wait-for-inputs("2")
17 | sort-triples(by="subject")
18 | collect-triples
19 | encode-json(prettyPrinting="true")
20 | print;
21
```

Fix

Morph

```
1 copy_field('gnd', '_id')
```

# Daten zusammenführen

```
'118514768' {  
  '001A' {  
    '0': '1250:01-07-88'  
  },  
  '001B' {  
    '0': '1140:26-07-13',  
    't': '08:58:08.000'  
  },  
  '001D' {  
    '0': '1220:16-06-08'  
  },  
  '001U' {  
    '0': 'utf8'  
  },  
  '001X' {  
    '0': '01'
```

<https://github.com/hbz/metafacture-flux-examples/tree/master/sample6>

# Aktueller Stand Fix

# Aktueller Stand Fix

Beispiele: produktiv in **OERSI** (diverse Web-Quellen → JSON),  
im Aufbau für **hbz-Verbundkatalog** (MARC → JSON) und  
**Rheinland-Pfälzische Bibliographie** (Allegro → JSON)

# Aktueller Stand Fix

Beispiele: produktiv in **OERSI** (diverse Web-Quellen → JSON),  
im Aufbau für **hbz-Verbundkatalog** (MARC → JSON) und  
**Rheinland-Pfälzische Bibliographie** (Allegro → JSON)

Tests: **unit** (Simulation von Input-Output-Events und Fix auf  
Code-Ebene; aktuell pass 422 / 447), **integration** (Input, Flux,  
Fix, Output als Dateien wie bei Real-World-Setup; aktuell  
pass 170 / 228, d.h. aktuell 75% "conformance")

# Aktueller Stand Fix

Beispiele: produktiv in **OERSI** (diverse Web-Quellen → JSON),  
im Aufbau für **hbz-Verbundkatalog** (MARC → JSON) und  
**Rheinland-Pfälzische Bibliographie** (Allegro → JSON)

Tests: **unit** (Simulation von Input-Output-Events und Fix auf  
Code-Ebene; aktuell pass 422 / 447), **integration** (Input, Flux,  
Fix, Output als Dateien wie bei Real-World-Setup; aktuell  
pass 170 / 228, d.h. aktuell 75% "conformance")

d.h. vieles geht schon, aber es gibt auch noch einiges zu tun

# Aktueller Stand Fix

Beispiele: produktiv in **OERSI** (diverse Web-Quellen → JSON),  
im Aufbau für **hbz-Verbundkatalog** (MARC → JSON) und  
**Rheinland-Pfälzische Bibliographie** (Allegro → JSON)

Tests: **unit** (Simulation von Input-Output-Events und Fix auf  
Code-Ebene; aktuell pass 422 / 447), **integration** (Input, Flux,  
Fix, Output als Dateien wie bei Real-World-Setup; aktuell  
pass 170 / 228, d.h. aktuell 75% "conformance")

d.h. vieles geht schon, aber es gibt auch noch einiges zu tun

**Aktuelle Dokumentation, GitHub-Repo, Playground**



# 7. Ausblick, Fragen & Diskussion

# Ausblick

# Standards nutzen und aufbauen

# Standards nutzen und aufbauen

SKOS Lookups (zum Andocken an [SkoHub Vocab](#)s)

# Standards nutzen und aufbauen

SKOS Lookups (zum Andocken an [SkoHub Vocab](#)s)

Entity Reconciliation mit [OpenRefine](#)-kompatiblen Diensten

# Standards nutzen und aufbauen

SKOS Lookups (zum Andocken an [SkoHub Vocab](#)s)

Entity Reconciliation mit [OpenRefine](#)-kompatiblen Diensten

Fix-Standardisierung, s. <https://github.com/elag/FIG>

# ETL Hub

# ETL Hub

ETL: Extract, Transform, Load



# ETL Hub

ETL: Extract, Transform, Load

mehr Kollaboration, Teilen & Auffinden von Workflows  
ermöglichen (nicht nur für Metafactory)

# ETL Hub

ETL: Extract, Transform, Load

mehr Kollaboration, Teilen & Auffinden von Workflows  
ermöglichen (nicht nur für Metafactory)

Entwicklung von Best Practices zur Paketierung und  
Beschreibung von ETL-Konfigurationen

# ETL Hub

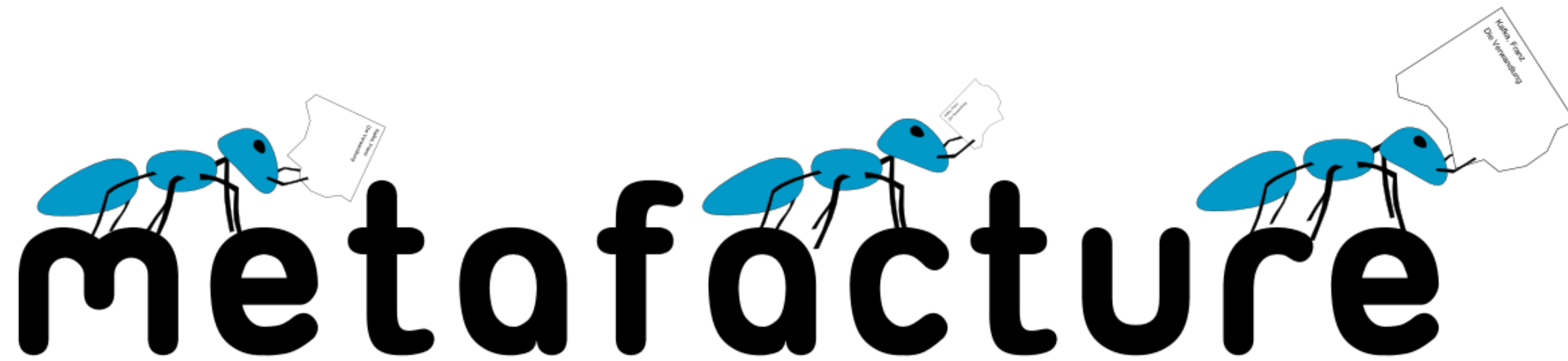
ETL: Extract, Transform, Load

mehr Kollaboration, Teilen & Auffinden von Workflows  
ermöglichen (nicht nur für Metafactory)

Entwicklung von Best Practices zur Paketierung und  
Beschreibung von ETL-Konfigurationen

Aufbau eines ETL Hubs zum Entdecken existierender ETL-  
Prozesse für die einfache Nachnutzung und Anpassung

# Fragen und Diskussion



<https://metafacture.org>