Process Mining for Artifact-Centric Processes

Dirk Fahland
based on joint work with
Wil v.d. Aalst, Boudewijn v. Dongen, Marlon Dumas, Massimiliano de Leoni, Luciano Garcia Banuelos, Viara Popova
A process that needs multiple instances
A process that needs multiple instances
A process that needs multiple instances
A process that needs multiple instances
A process that needs multiple instances

comperror relations between process instances
Behavior observed in this process

name: split
eventid: ...
time: ...
to: [delivery1, delivery2]

order1

delivery1

delivery2

order2

4 cases + interactions between cases

follow-up
Classical Process Discovery

\[
\begin{align*}
c1: & \quad A \; B \; C \; D \; E \\
c2: & \quad A \; C \; B \; D \; E \\
c3: & \quad A \; F \; D \; E \\
\ldots
\end{align*}
\]

assumptions

- case = sequence of events of this case
- cases are isolated: event A in c1 happens only in c1 (and not in c2)
- cases of the same process
  - one unique case id,
  - each event associated to exactly one case id
Classical Process Discovery

```plaintext
assumptions
- case = sequence of events of this case
- cases are isolated:
  - event A in c1 happens only in c1 (and not in c2)
- cases of the same process
  - one unique case id,
  - each event associated to exactly one case id
```

```
c1: A B C D E
   c2: A C B D E
   c3: A F D E
   ...
```
Process Mining (classical)

- Classical log
- Discovery
- Extension/Repair
- Conformance

Model:
- A → B

Verification:
- ✓ / ✗
Process Mining for Artifacts

1. Artifact log
2. Generalizes classical log
3. Discovery, extension/repair, conformance

DB

A → B

A, B → C

✓ / ✗
Artifact Log - Example

name: split
eventid: ...
time: ...
to: [delivery1, delivery2]

order1

order
delivery

delivery1
delivery2

order2

4 cases + interactions between cases

follow-up
Artifact Log - Example

capture artifact interaction by a case: \{send-event, receive-event\}

event can participate in several cases (cases overlap on joint events)

events are ordered within a case by their timestamps
Artifact Log - Example
Artifact logs and models (Asynchronous)

- cases ↔ life-cycles
- interaction cases ↔ channels
Different correlation of events

order1 / package1 / delivery1

order2 / package2 / delivery2

order

+ 1 + + *

package

follow-up

package3

Delivery 1

order1 + package1 + delivery1

order2 + package2 + delivery2
Artifact logs and models (Synchronous)

- cases ⇔ life-cycles
- artifacts synchronize on shared actions
Compositionality

one log per “component”

is a set of logs that overlap

artifacts

artifact log

component models

classical

A
B

L
L

A
B

A
B

A
B

A
B

A
B

A
B

components + sync.
Mining for Artifacts: reduction to classical
discovery
extension/repair
conformance

one log per “component”

compose by overlap in L

decompose

classical

component models

artifacts

artifact log

A
B
C

Process Mining for Artifacts

1. artifact log
2. model
3. conformance
discovery
extension/repair

DB 

classical log

generalizes
Typical Process in an ERP System

Build to Order

Material A
Material B

Material B
Material C

Material A
Material C

Alice

Bob

order product X

order product Y

order materials

ACME Inc.

Mega Corp.
n-to-m relations $\rightarrow$ database

**id attributes**

<table>
<thead>
<tr>
<th>polID</th>
<th>cust.</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>po1</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td>po2</td>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>

**time-stamp attributes**

<table>
<thead>
<tr>
<th>ProductOrder</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>created</td>
<td>processed</td>
<td>built</td>
<td>shipped</td>
</tr>
<tr>
<td>po1</td>
<td>30-08 9:22</td>
<td>30-08 13:12</td>
<td>01-09 15:12</td>
<td>03-09 10:15</td>
</tr>
<tr>
<td>po2</td>
<td>30-08 10:15</td>
<td>30-08 13:14</td>
<td>01-09 16:13</td>
<td>03-09 17:18</td>
</tr>
</tbody>
</table>

**Customer**

<table>
<thead>
<tr>
<th>cust.</th>
<th>address</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**data attributes**

<table>
<thead>
<tr>
<th>MaterialOrder</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>completed</td>
<td>sent</td>
<td>received</td>
<td></td>
</tr>
<tr>
<td>mo3</td>
<td>30-08 13:15</td>
<td>30-08 14:15</td>
<td>01-09 9:05</td>
<td></td>
</tr>
<tr>
<td>mo4</td>
<td>30-08 13:17</td>
<td>30-08 16:12</td>
<td>01-09 10:13</td>
<td></td>
</tr>
</tbody>
</table>

**OrderedMaterial**

<table>
<thead>
<tr>
<th>OrderedMaterial</th>
<th>polID</th>
<th>moID</th>
<th>type</th>
<th>added</th>
</tr>
</thead>
<tbody>
<tr>
<td>po1</td>
<td>mo3</td>
<td>B</td>
<td>30-08 13:13</td>
<td></td>
</tr>
<tr>
<td>po1</td>
<td>mo4</td>
<td>A</td>
<td>30-08 13:14</td>
<td></td>
</tr>
<tr>
<td>po2</td>
<td>mo3</td>
<td>B</td>
<td>30-08 13:15</td>
<td></td>
</tr>
<tr>
<td>po2</td>
<td>mo4</td>
<td>C</td>
<td>30-08 13:16</td>
<td></td>
</tr>
</tbody>
</table>

**MaterialOrder**

<table>
<thead>
<tr>
<th>MaterialOrder</th>
<th>molD</th>
<th>suppl.</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>mo3</td>
<td>ACME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mo4</td>
<td>MEGA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Extracting event logs from artifact systems

- Reality: data in a relational DB
  - events stored as time-stamped attributes in tables
  - multiple primary keys → multiple notions of case
  - tables are related → one event related to multiple cases
Approach to log extraction & discovery

log f. PO
log f. MO
model f. PO
model f. MO

decompose by primary keys
related by primary foreign-key relations

discovery
discovery
Find Artifact Schemas

decompose by primary keys

related by primary foreign-key relations
Step 0: discover database schema

- document schema vs. **actual** schema → identify
  - column types (esp. time-stamped columns)
  - primary keys
  - foreign keys
- various (non-trivial) techniques available
- key discovery is NP-complete in the size of the table(s)
- result:
Step 1: decompose schema into processes

= schema summarization

find:
1. sets of corresponding tables
2. links between those
Automatic Schema Summarization

= group similar tables through clustering

- define a distance between any 2 tables
  - by relations
  - by information content

- tables that are close to each other → same cluster

- # of clusters: user input
Grouping by Clustering

1. **structural distance**
   many references = close

2. **information distance**
   importance of each table = **entropy** (is maximal if all records are different)
   distance: 2 tables with high entropies → large distance

3. **weighted distance**
   by structure + information

4. **k-means clustering:**
   k clusters based on weighted distance

*most important table of cluster = table with least distance to all → key attribute of the cluster*
Artifact Schema ➔ Artifact Log

- Decompose by primary keys
- Related by primary foreign-key relations

**Model f. PO**
- PO: log f.
- MO: log f.

**Process Model**
- Customer
  - cust
  - custId
- MaterialOrder
  - moid
  - sent
  - received
- OrderedMaterial
  - podId
  - type
  - added
- ProductOrder
  - podId
  - custId
  - created
  - processed
  - built
  - shipped

**Discovery**
- PO
- MO
Log Extraction

cluster = set of related tables + primary key of most important table

case id

<table>
<thead>
<tr>
<th>poID</th>
<th>cust.</th>
<th>...</th>
<th>created</th>
<th>processed</th>
<th>built</th>
<th>shipped</th>
</tr>
</thead>
<tbody>
<tr>
<td>po1</td>
<td>Alice</td>
<td></td>
<td>30-08 9:22</td>
<td>30-08 13:12</td>
<td>01-09 15:12</td>
<td>03-09 10:15</td>
</tr>
<tr>
<td>po2</td>
<td>Bob</td>
<td></td>
<td>30-08 10:15</td>
<td>30-08 13:14</td>
<td>01-09 16:13</td>
<td>03-09 17:18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>poID</th>
<th>moID</th>
<th>type</th>
<th>added</th>
</tr>
</thead>
<tbody>
<tr>
<td>po1</td>
<td>mo3</td>
<td>B</td>
<td>30-08 13:13</td>
</tr>
<tr>
<td>po1</td>
<td>mo4</td>
<td>A</td>
<td>30-08 13:14</td>
</tr>
<tr>
<td>po2</td>
<td>mo3</td>
<td>B</td>
<td>30-08 13:15</td>
</tr>
<tr>
<td>po2</td>
<td>mo4</td>
<td>C</td>
<td>30-08 13:16</td>
</tr>
</tbody>
</table>
Log Extraction

cluster = set of related tables + primary key of most important table

time-stamped attribute \rightarrow event

po1:(created, poID=po1, time=30-08 9:22, ...)

<table>
<thead>
<tr>
<th>poID</th>
<th>cust.</th>
<th>created</th>
<th>processed</th>
<th>built</th>
<th>shipped</th>
</tr>
</thead>
<tbody>
<tr>
<td>po1</td>
<td>Alice</td>
<td>30-08 9:22</td>
<td>30-08 13:12</td>
<td>01-09 15:12</td>
<td>03-09 10:15</td>
</tr>
<tr>
<td>po2</td>
<td>Bob</td>
<td>30-08 10:15</td>
<td>30-08 13:14</td>
<td>01-09 16:13</td>
<td>03-09 17:18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>poID</th>
<th>moID</th>
<th>type</th>
<th>added</th>
</tr>
</thead>
<tbody>
<tr>
<td>po1</td>
<td>mo3</td>
<td>B</td>
<td>30-08 13:13</td>
</tr>
<tr>
<td>po1</td>
<td>mo4</td>
<td>A</td>
<td>30-08 13:14</td>
</tr>
<tr>
<td>po2</td>
<td>mo3</td>
<td>B</td>
<td>30-08 13:15</td>
</tr>
<tr>
<td>po2</td>
<td>mo4</td>
<td>C</td>
<td>30-08 13:16</td>
</tr>
</tbody>
</table>
Log Extraction

cluster = set of related tables + primary key of most important table

case id

time-stamped attribute → event

related attributes → event attributes

po1: (created, poID=po1, time=30-08 9:22, cust.=Alice, ...)
Log Extraction

**cluster** = set of related tables + primary key of most important table

case id

time-stamped attribute → event

related attributes → event attributes

<table>
<thead>
<tr>
<th>polID</th>
<th>cust.</th>
<th>...</th>
<th>created</th>
<th>processed</th>
<th>built</th>
<th>shipped</th>
</tr>
</thead>
<tbody>
<tr>
<td>po1</td>
<td>Alice</td>
<td></td>
<td>30-08 9:22</td>
<td>30-08 13:12</td>
<td>01-09 15:12</td>
<td>03-09 10:15</td>
</tr>
<tr>
<td>po2</td>
<td>Bob</td>
<td></td>
<td>30-08 10:15</td>
<td>30-08 13:14</td>
<td>01-09 16:13</td>
<td>03-09 17:18</td>
</tr>
</tbody>
</table>

po1:(created, polID=po1, time=30-08 9:22, cust.=Alice, …)

(processed, polID=po1, time=30-08 13:12, …)

<table>
<thead>
<tr>
<th>polID</th>
<th>molID</th>
<th>type</th>
<th>added</th>
</tr>
</thead>
<tbody>
<tr>
<td>po1</td>
<td>mo3</td>
<td>B</td>
<td>30-08 13:13</td>
</tr>
<tr>
<td>po1</td>
<td>mo4</td>
<td>A</td>
<td>30-08 13:14</td>
</tr>
<tr>
<td>po2</td>
<td>mo3</td>
<td>B</td>
<td>30-08 13:15</td>
</tr>
<tr>
<td>po2</td>
<td>mo4</td>
<td>C</td>
<td>30-08 13:16</td>
</tr>
</tbody>
</table>
Log Extraction

cluster = set of related tables + primary key of most important table

case id

time-stamped attribute → event

related attributes → event attributes

<table>
<thead>
<tr>
<th>poID</th>
<th>cust.</th>
<th>...</th>
<th>created</th>
<th>processed</th>
<th>built</th>
<th>shipped</th>
</tr>
</thead>
<tbody>
<tr>
<td>po1</td>
<td>Alice</td>
<td></td>
<td>30-08 9:22</td>
<td>30-08 13:12</td>
<td>01-09 15:12</td>
<td>03-09 10:15</td>
</tr>
<tr>
<td>po2</td>
<td>Bob</td>
<td></td>
<td>30-08 10:15</td>
<td>30-08 13:14</td>
<td>01-09 16:13</td>
<td>03-09 17:18</td>
</tr>
</tbody>
</table>

po1: (created, poID=po1, time=30-08 9:22, cust.=Alice, ...)
(processed, poID=po1, time=30-08 13:12, ...)
added, poID=po1, time=30-08 13:13, moID=mo3, ...

refers to artifact “MaterialOrder”
Outline

decompose by primary keys

compose by primary foreign-key relations

log f. quote

log f. order

discovery

discovery

process model

model f. order

model f. quote
**Resulting Model(s)**

**Product Order**
- create
- processed
- added
- built
- shipped

**Material Order**
- added
- completed
- sent
- received

(added, poID=po1, …, moID=mo3)
Open issues

- performance
  - key discovery: NP-complete in R (\# of records)
  - \(\rightarrow\) sampling of data, domain knowledge, semi-automatic

- requires good database structure
  - proper relations, proper keys
  - otherwise wrong clusters are formed
  - events don’t get right attributes
  - \(\rightarrow\) semi-automatic approach

- events shared by multiple cases
  - just a reference between two tables does not tell an event participated was shared by two cases \(\rightarrow\) working on it
Summary: Mining for Artifact Lifecycles

- ProM: discovery
- + data
- ProM: model repair
- + data
- ProM: conformance checker

- Decompose
- Compose
Summary: Mining for Artifacts

one log per artifact

raw log/event stream

ProM: Extract from Raw Log

ProM: Extract from DB

artifact log

Decompose

Compose

DB

artifact model