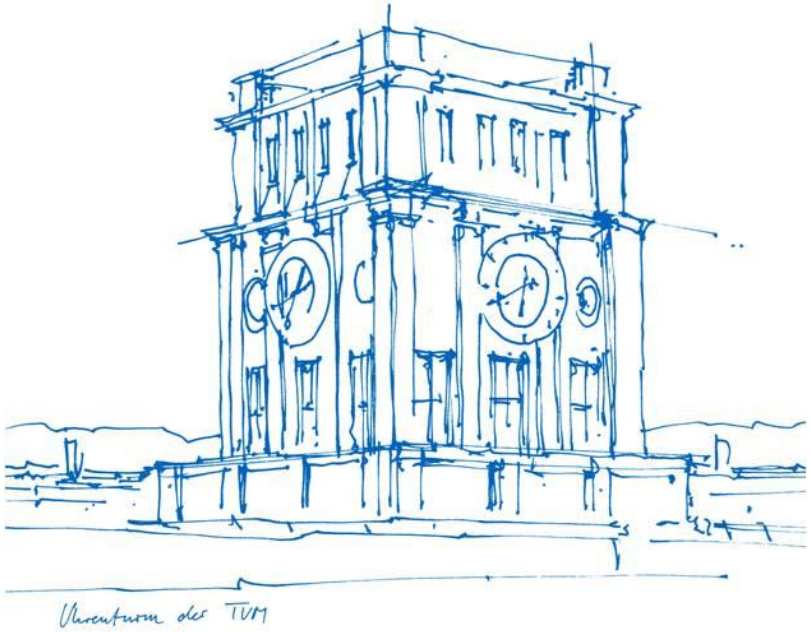
Intermediate Presentation Process Mining

Jakob Steimle

Munich, 23. Juni 2022



Status Report

Customer / Project:

Praktikum Process Mining

Status:

80 % Complete

Date of delivery:

2022 – 07 – 17

Completed Activities:

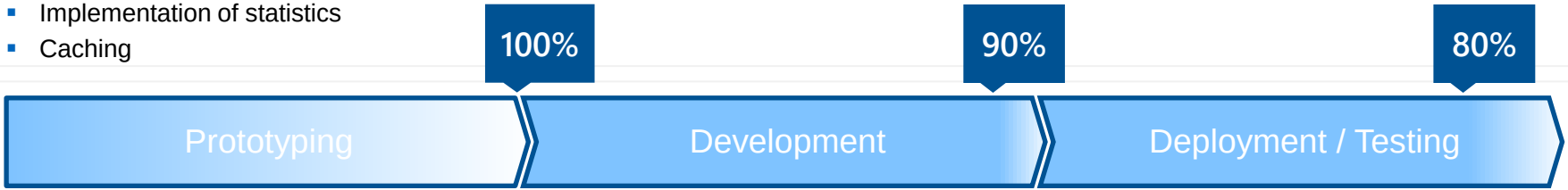
- Alpha miner
- Heuristic miner
- Upload mechanism
- Rudimentary statistics
- XES parser
- Custom visualization frontend
- Static tests for XES Parser and Alpha Miner

Next Activities:

- Tests for heuristic miner
- Further statistics
- Minor bug fixes
- Final deployment

Core challenges

- Implementation of statistics
- Caching



Jakob Steimle

Agenda

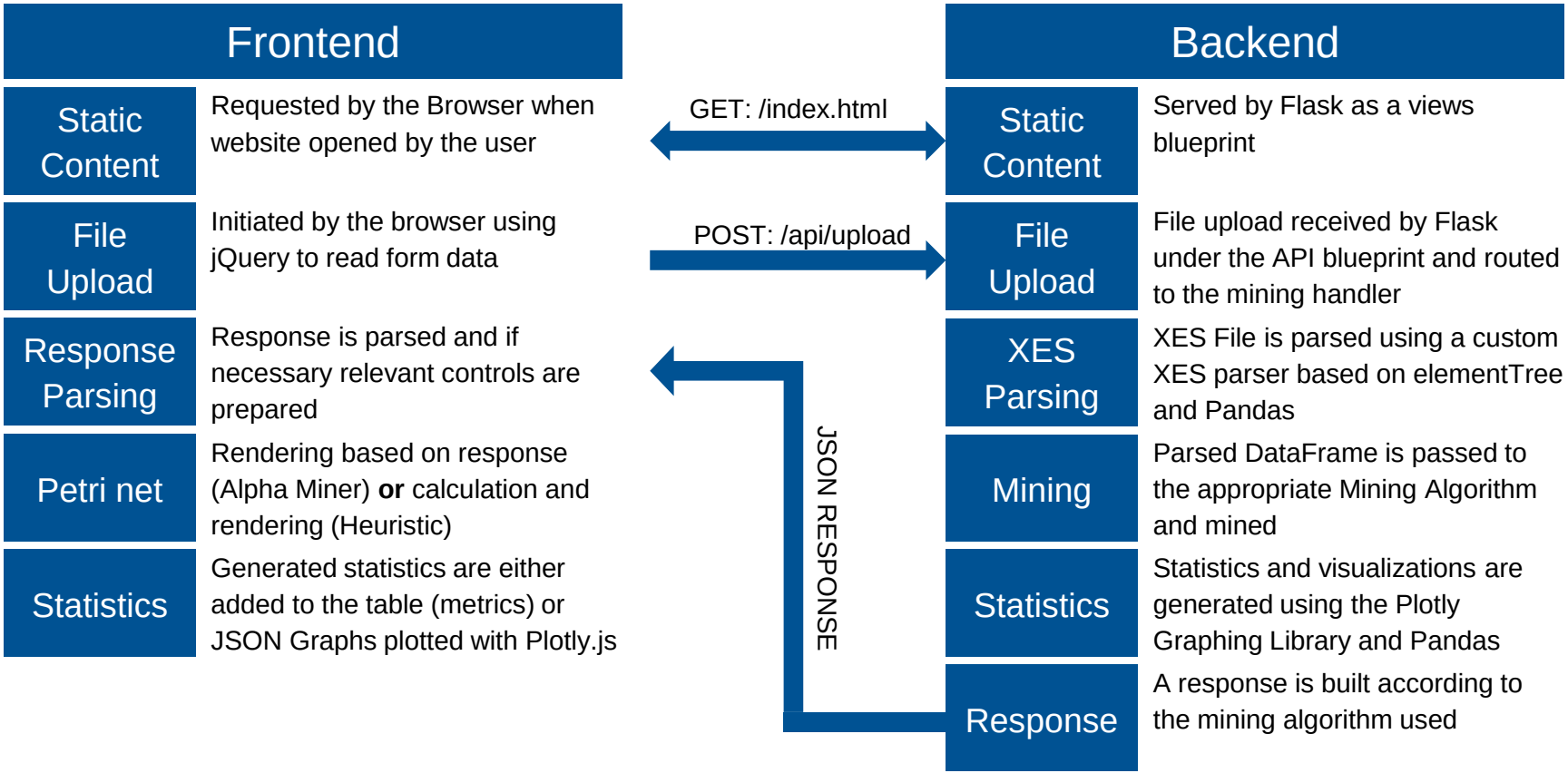
1 **Architektur**

2 UI

3 Miner

4 Testing

5 Demo


Frontend
Backend
Static Content
Requested by the Browser when website opened by the user
GET: /index.html
Static Content
Served by Flask as a views blueprint
File Upload
Initiated by the browser using jQuery to read form data
POST: /api/upload
File Upload
File upload received by Flask under the API blueprint and routed to the mining handler
Response Parsing
Response is parsed and if necessary relevant controls are prepared
XES Parsing
XES File is parsed using a custom XES parser based on elementTree and Pandas
Petri net
Rendering based on response (Alpha Miner) or calculation and rendering (Heuristic)
Mining
Parsed DataFrame is passed to the appropriate Mining Algorithm and mined
Statistics
Generated statistics are either added to the table (metrics) or JSON Graphs plotted with Plotly.js
JSON RESPONSE
Statistics
Statistics and visualizations are generated using the Plotly Graphing Library and Pandas
Response
A response is built according to the mining algorithm used

Agenda

1 Architektur

2 UI

3 Miner

4 Testing

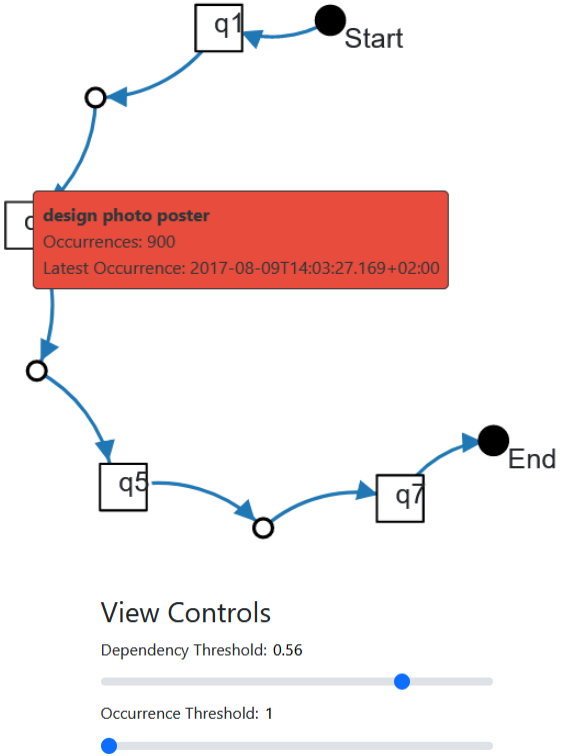
5 Demo

Petrinet.js

- Custom Petri net rendering based on D3.js
- Includes further information on hover
- Can be redrawn rapidly when parameters are changed by the user

Challenges

- Still some graphic issues with arrow heads disappearing behind the process steps
- Improving the display of the hover information
- Minor issue with naming of process steps if their length is too long

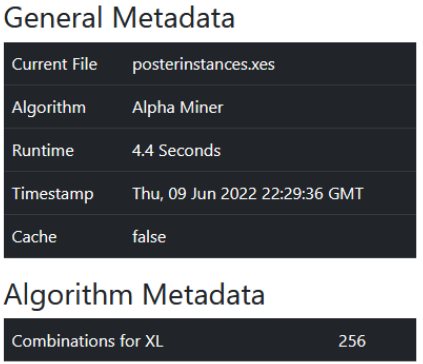


Statistics Table

- Contains general statistics about the run – e.g., timestamp, runtime, algorithm and filename
- Algorithm metadata renders relevant metadata related to the current algorithm e.g. number of combinations of X_L

Challenges

- Addition of further algorithm metadata and metrics such as “Parsing Measure” for the Heuristic Miner

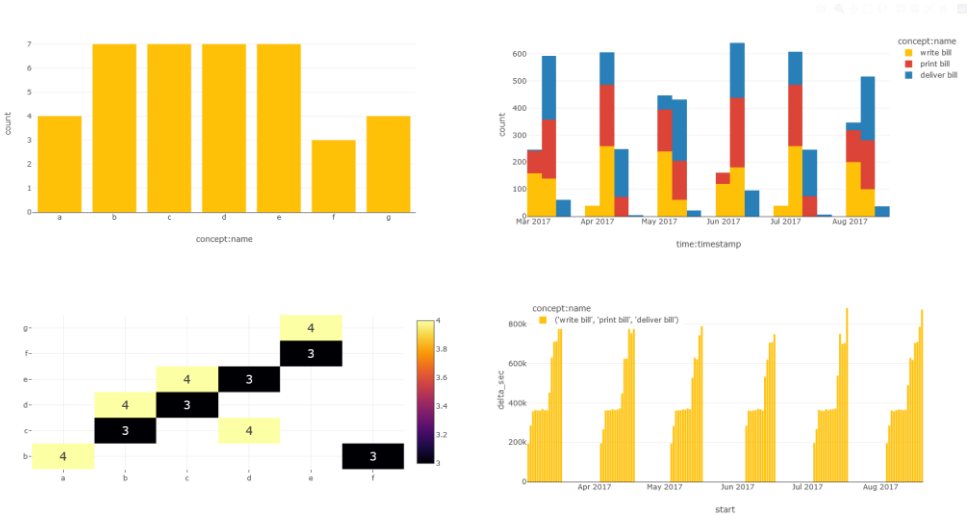


General Statistics

- Renders different statistics
- Currently implemented
  - Process step frequency (top left)
  - Process step frequency over time (top right)
  - Transition heat map (bottom left)
  - Median process chain execution time over time (bottom right)

Challenges

- Some general issues with rendering timestamps and legends
- Further statistics if desired



Agenda

1 Architektur

2 UI

3 Miner

4 Testing

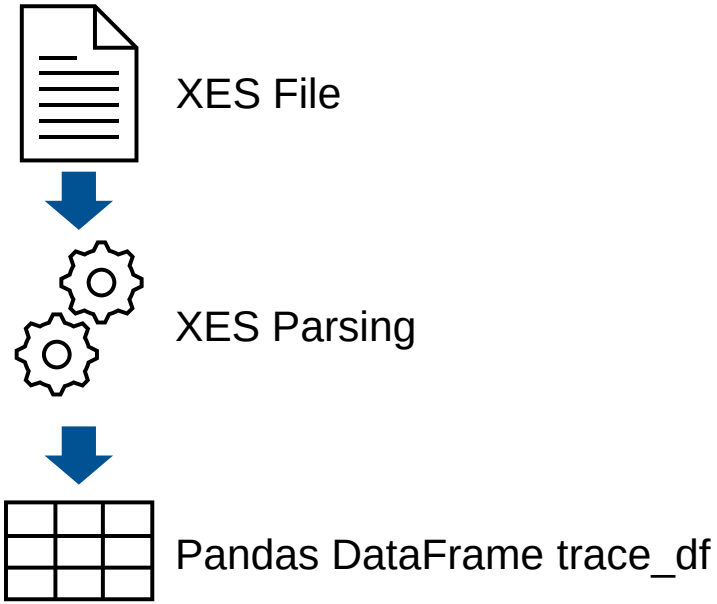
5 Demo

XES Parser

- Based on ElementTree in Python
- Produces a Pandas DataFrame of the parsed logs
- Can read any included tags per Event

Potential challenges

- Increasing the robustness when it comes to incomplete events, etc.

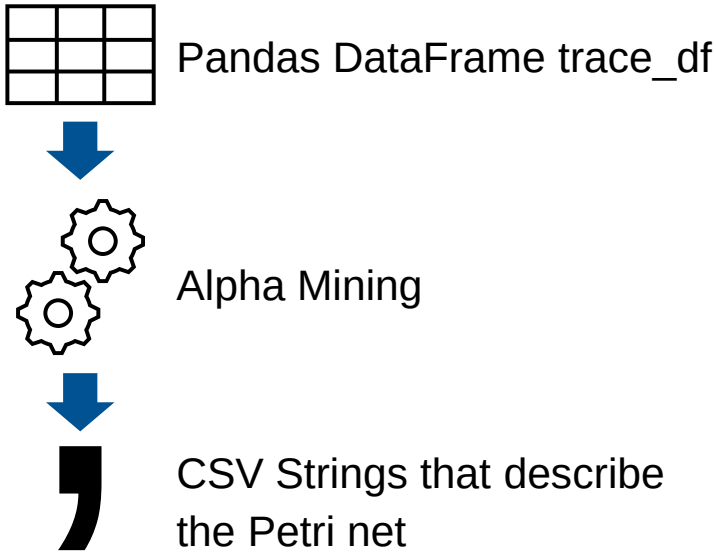


Alpha Miner

- Based on pandas DataFrames and some python list operations
- Reasonably inefficient
- Uses some power iterations from Itertools

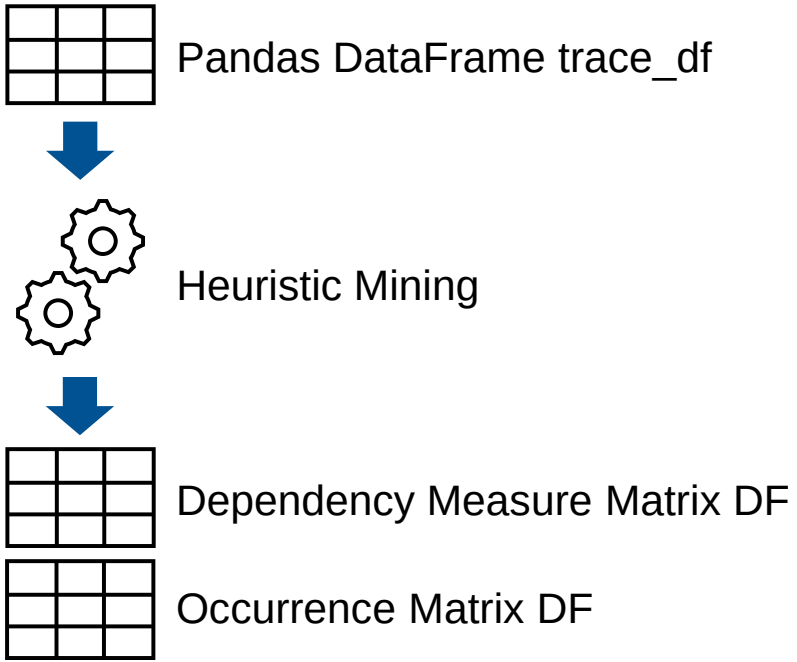
Challenges

- If it were desired: performance, as all permutations of process steps are considered



Heuristic Miner

- Based nearly entirely on pandas DataFrame Operations
- Reasonably efficient



Agenda

1 Architektur

2 UI

3 Miner

4 Testing

5 Demo

Testing

- Testing is done for XES parsing as well as the alpha miner (done) and the heuristic miner (WIP)
- Tests are static test run against a ‘oracle’ based on the example files provided
- The python unittest is used to create the test cases

```python
class AlphaMinerTests(unittest.TestCase):

    def test(self):
        testFiles = ['L1', 'L2', 'L3', 'L4', 'L5', 'L6', 'L7']
        for i in testFiles:
            self.runTest(i)

    def runTest(self, file):
        filepath = f"resources/{file}.xes"
        test_xml_string = self.load_test_file(filepath)
        parser = XESParser()
        parser.read_xes(test_xml_string)
        traces_df = parser.get_parsed_logs()
        miner = AlphaMiner()
        miner.run(traces_df)
        loc_csv = miner.get_location_csv()
        trans_csv = miner.get_transition_csv()
        loc_oracle_df = pd.read_csv(f"resources/{file}-loc-oracle.csv").sort_values(['loc', 'type']).reset_index(drop=True)
        trans_oracle_df = pd.read_csv(f"resources/{file}-trans-oracle.csv").sort_values(['source', 'target', 'type']).reset_index(drop=True)
        loc_actual_df = pd.read_csv(StringIO(loc_csv)).sort_values(['loc', 'type']).reset_index(drop=True)
        trans_actual_df = pd.read_csv(StringIO(trans_csv)).sort_values(['source', 'target', 'type']).reset_index(drop=True)
        self.assertEqual(len(loc_oracle_df.compare(loc_actual_df).index), 0)
        self.assertEqual(len(trans_oracle_df.compare(trans_actual_df).index), 0)
```

Agenda

1 Architektur

2 UI

3 Miner

4 Testing

5 Demo