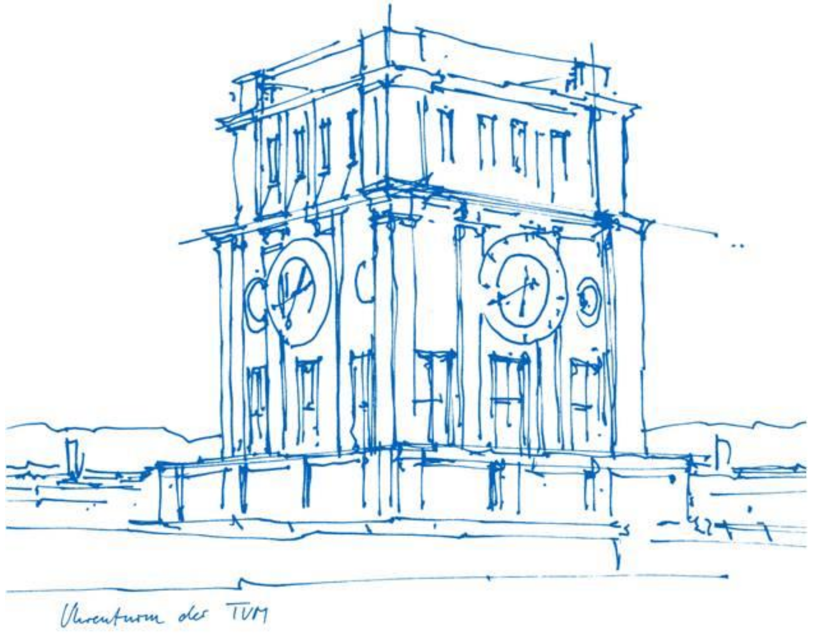
Final Presentation Process Mining

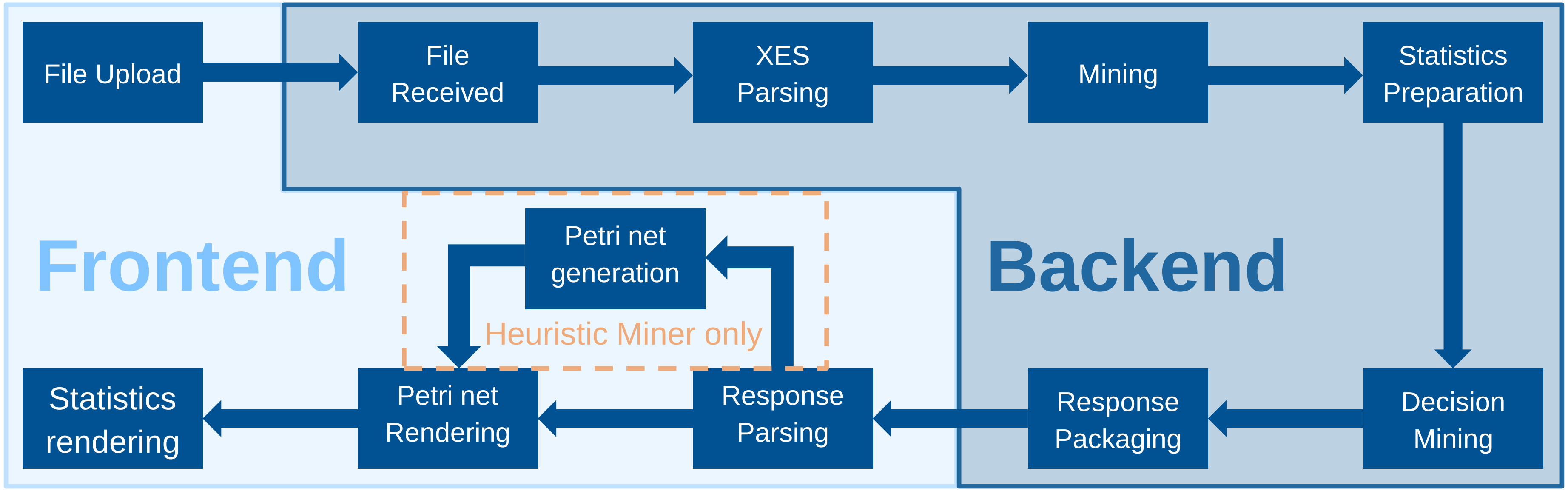
Jakob Steimle

Munich, 18. Juli 2022



Agenda

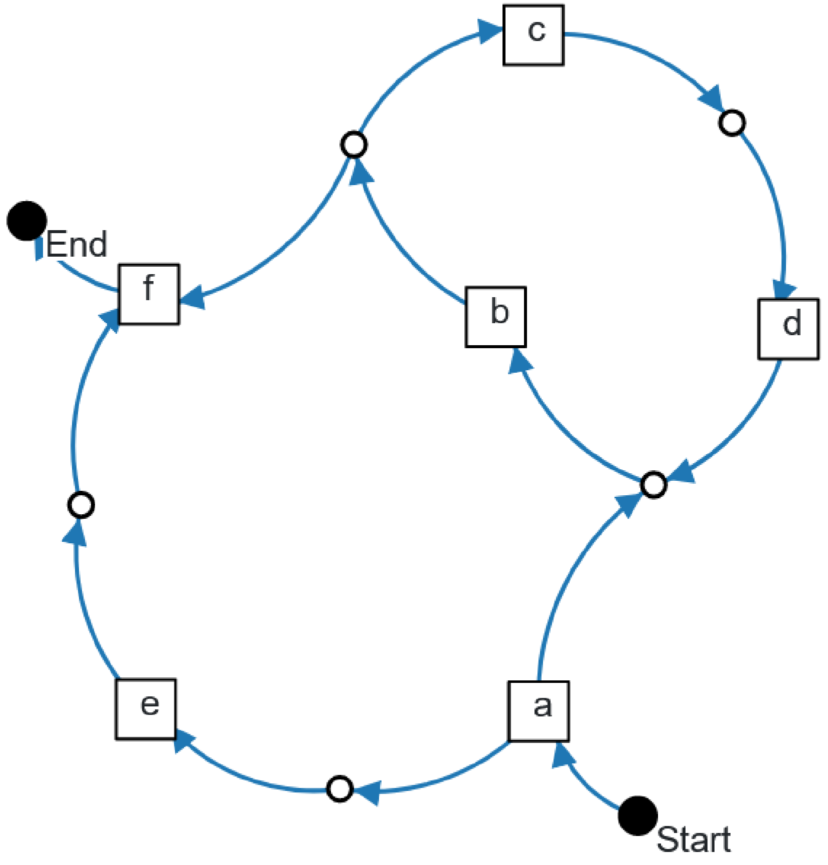
Process flow



Agenda

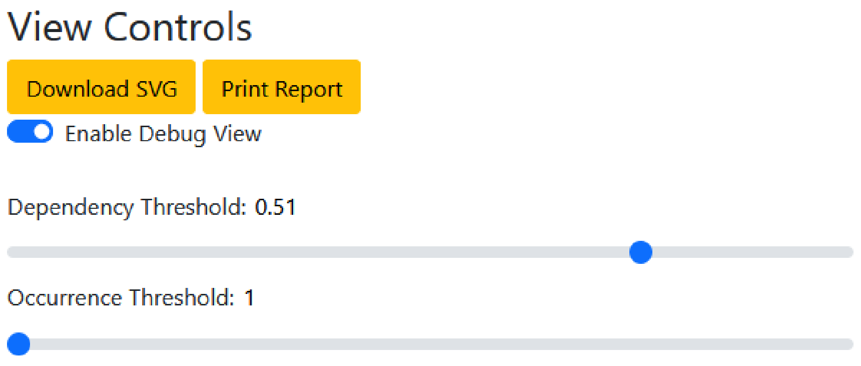
petrinet.js

- Custom Petri net rendering
- Includes further information on hover
- Can be redrawn rapidly when parameters are changed by the user
- Force field for layout and dynamic rearranging of nodes and transitions
- Based on the d3.js force layout, extended with petri net specific features as well as hover information



heuristic.js

- Generates a petri net from the dependency and succession matrix generated in the backend
- Calculates the input and output bindings for each transitions and attempts to convert the heuristic net to a petri net
- Identifies splits and joins based on the input and output bindings
- Creates a transition and place csv that is the parsed by petrinet.js
- Provides functionality for interactivity

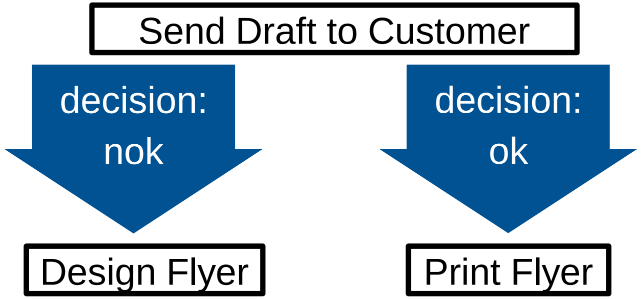


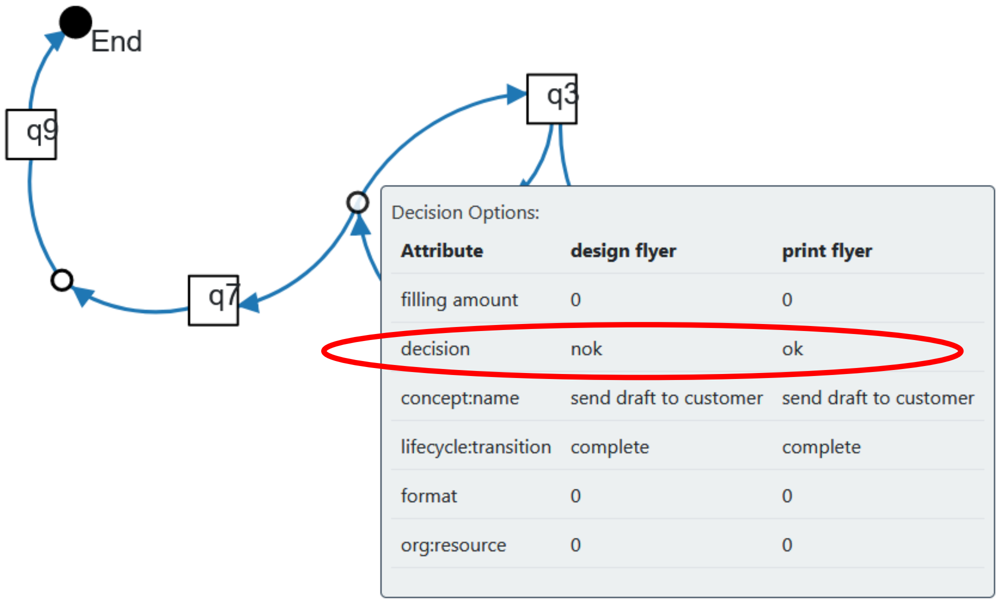
Dependency Threshold: 0.51

Occurrence Threshold: 1

decisions.js

- Shows rudimentary decision information
- Identifies decision nodes and associates them with the possible options
- Example:



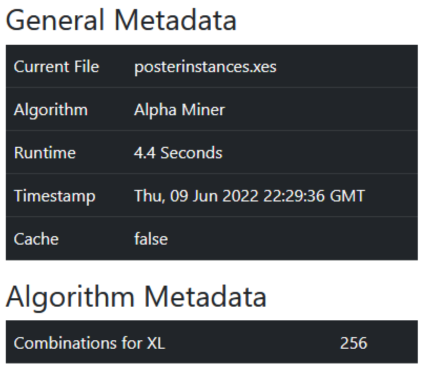


Decision Options:

Attribute	design flyer	print flyer
filling amount	0	0
decision	nok	ok
concept:name	send draft to customer	send draft to customer
lifecycle:transition	complete	complete
format	0	0
org:resource	0	0

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



General Metadata

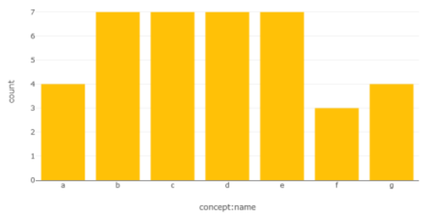
Current File	posterinstances.xes
Algorithm	Alpha Miner
Runtime	4.4 Seconds
Timestamp	Thu, 09 Jun 2022 22:29:36 GMT
Cache	false

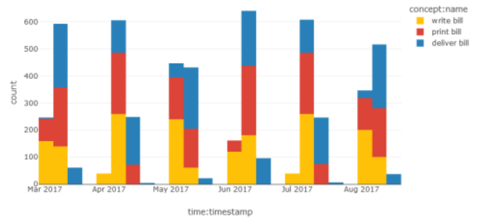
Algorithm Metadata

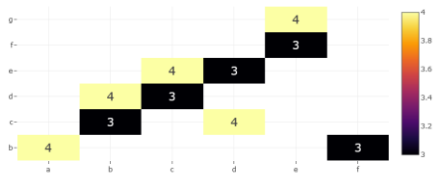
Combinations for XL	256

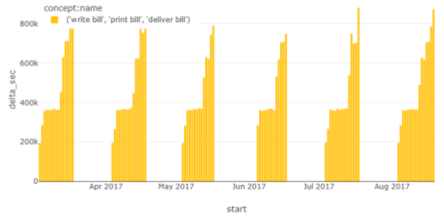
General Statistics

- 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)







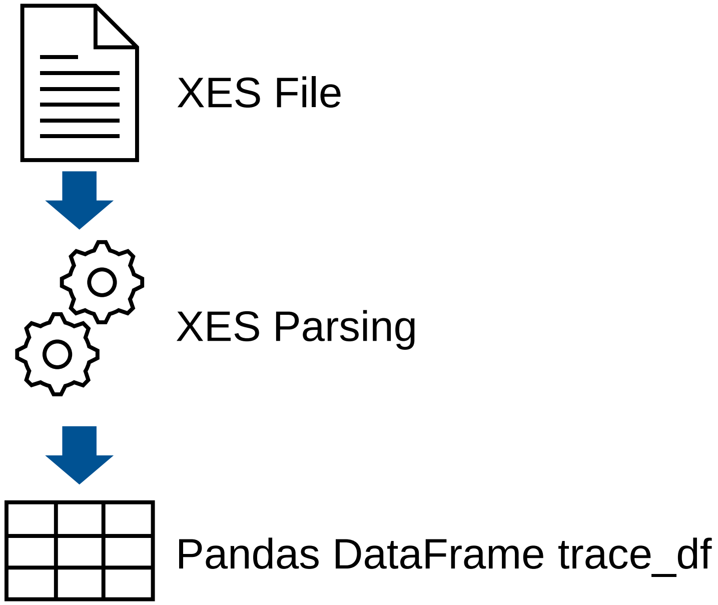


Jakob Steimle

Agenda

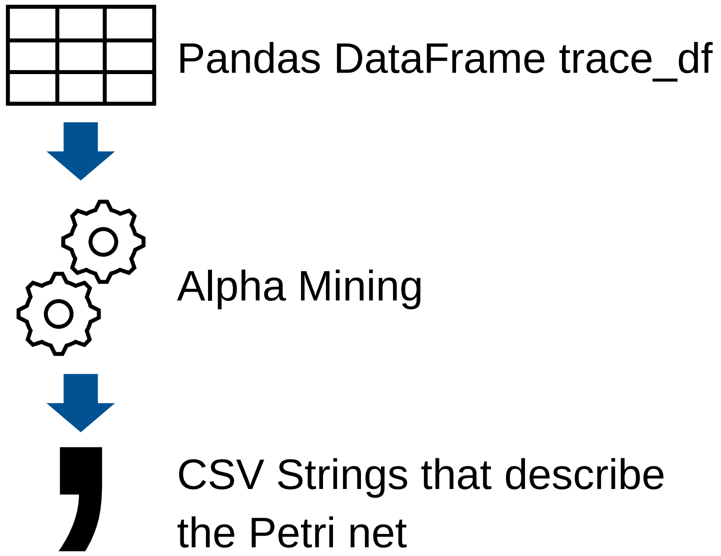
XES Parser

- Based on ElementTree in Python
- Supports (non-standard) XES Files that use a different Namespace
- Produces a Pandas DataFrame of the parsed logs
- Can read any included tags per Event



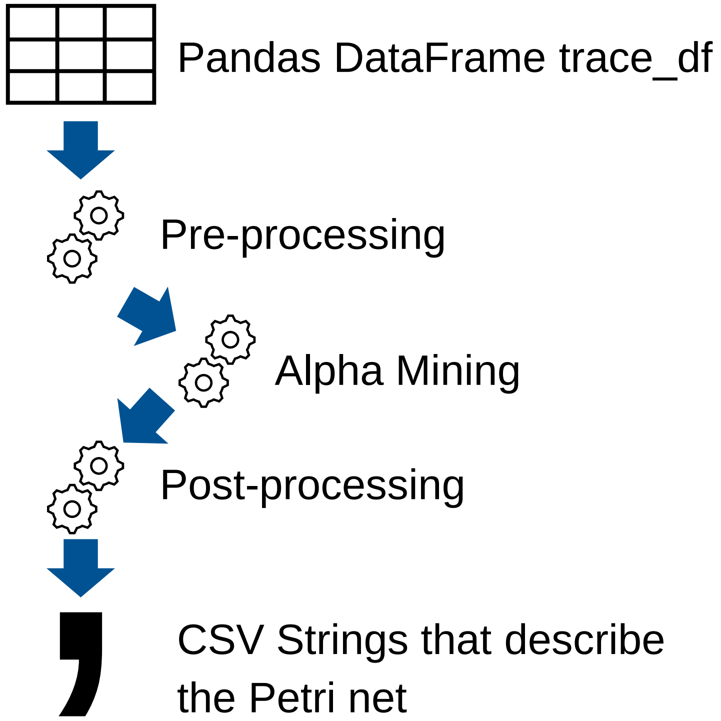
Alpha Miner

- Basic implementation using Pandas DataFrames and List operations
- Also generates transition information for association with the decision miner
- Since the petri net does not change based on user parameters all computation is done in the backend


Pandas DataFrame trace_df
Alpha Mining
CSV Strings that describe the Petri net

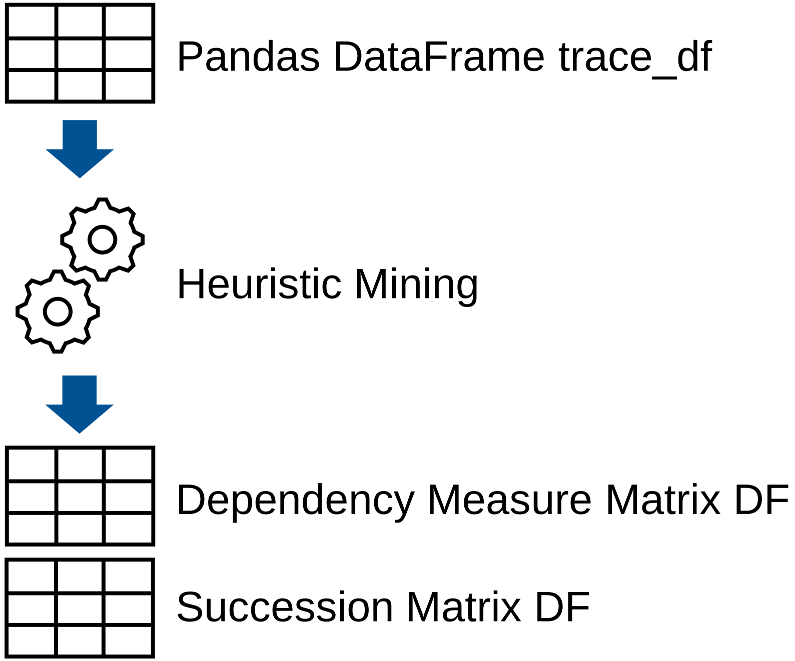
Alpha Plus Miner

- Very similar to the Alpha Miner
- Includes some further post- and pre-processing for mining loops of length 1


Pandas DataFrame trace_df
Pre-processing
Alpha Mining
Post-processing
CSV Strings that describe the Petri net

Heuristic Miner

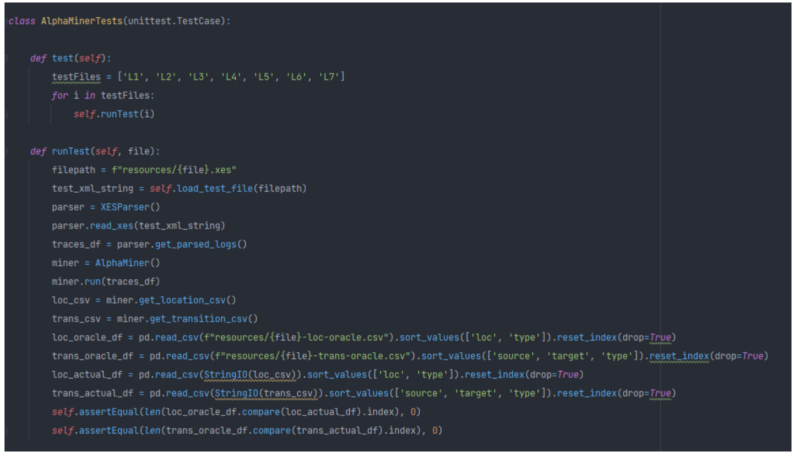
- Generates the dependency measure matrix as well as the succession matrix
- Since the output petri net depends on user parameters further calculations are done in the front end



Agenda

Testing

- Testing is done for XES parsing as well as the alpha miner, alpha plus miner and the heuristic miner
- Tests are static test run against a ‘oracle’ based on the example files provided
- The python unit test library 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