Business 4720 - Class 9

Process Analytics

Joerg Evermann

Faculty of Business Administration
Memorial University of Newfoundland
jevermann@mun.ca

CC BY NC

Unless otherwise indicated, the copyright in this material is owned by Joerg Evermann. This material is licensed to you under the Creative Commons by-attribution non-commercial license (CC BY-NC 4.0)

MEMORIAL UNIVERSITY

This Class

What You Will Learn:

- Introduction to Process Data
- Introduction to Process Analytics
  - Process Discovery
  - Process Conformance Analysis
  - Process Performance Analysis

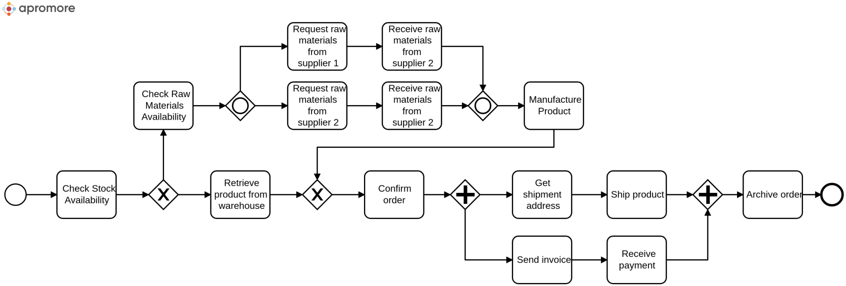
Business Processes

- Sequence of activities in a defined order
- Models describe processes
- Typically related to the processing of one type of business object
  - e.g. an order, a prescription, a complaint, etc.
- Standard notation: BPMN (”Business Process Modeling Notation”)

Basic BPMN

- Events (circles)
- Activities (boxes)
- Gateways (diamonds)
  - Exclusive (”X”) split and merge
  - Parallel (”+”) split and merge
  - Inclusive (”O”) split and merge

BPMN Example Process Model



Business Process Instances and Process Data

Case

- One instance of a business process
- Related to one particular business object (e.g. order 123, prescription R456, complaint C6789, etc.)

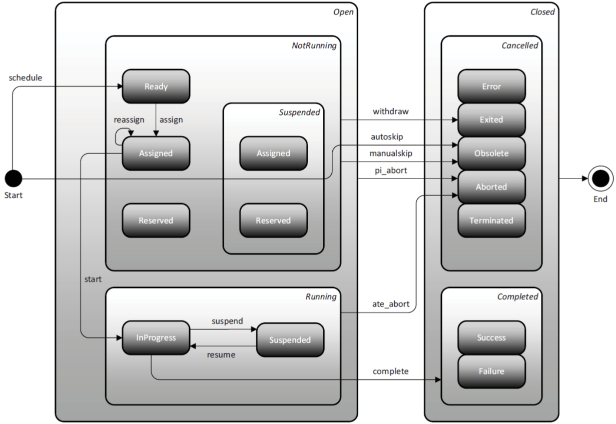
Trace

- Event data sequence for one case
- May includes attributes for the case and for each event
- May include resource information for each event
- May include timestamps for events

Event Log

- Set of traces for one process
- May include incomplete cases, may be sampled, etc.

Activity Lifecycle – Example



https://www.tf-pm.org/resources/xes-standard/about-xes/standard-extensions/lifecycle/standard

MEMORIAL UNIVERSITY

Event Log Data

Sources

- Process aware information systems
- Web server data
- ...

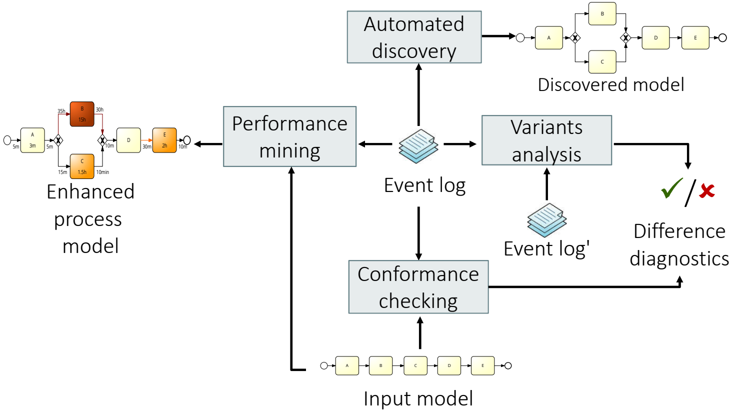
Formats

- CSV (one line per event)
- MXML (older XML format)
- XES ("eXtensible Event Stream") (current XML format)

Challenges

- Event correlation from multiple systems
- Noise, incompleteness
- Timestamping, batch processing
- Abstraction levels

Process Analytics



Source: Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A. Reijers (2018) "Fundamentals of Business Process Management", 2nd edition, Springer Verlag, Germany

Purpose

- Discover actual operations
- Check actual process against desired process
- Identify operational (performance) problems
- Improve operational processes
- External compliance analysis and reporting
- Identify implicit or de-facto organizational groups and relationships

Process Analytics in R

BupaR

- Hasselt University
- Open source R library
- Development since 2020
- Process visualization (DFG)
  - Frequencies metrics
  - Performance metrics
- Control flow analysis
- Rule-based conformance analysis
- Performance metrics
- Organizational analysis

`https://bupar.net/`

MEMORIAL UNIVERSITY

Process Analytics in Python

PM4PY

- Fraunhofer Institute for Applied Information Technology
- Open source python package, since 2018
- Process discovery
  - *Techniques*: Inductive miner, Heuristics miner, ...
- Conformance checking
  - *Techniques*: Token-based replay, Cost-based alignment, ...
- Log–Model Comparison
  - *Metrics*: Fitness, Precision, Generalization, Complexity, ...
- Decision mining
- Trace clustering
- LTL checking
- Social network discovery

`https://processintelligence.solutions/pm4py`

MEMORIAL UNIVERSITY

PM4PY Basics

Read a CSV dataset:

```
import pandas as pd
import pm4py

# Load the event log and parse date columns
log = pd.read_csv('https://evermann.ca/busi4720/\
   PurchasingExample.csv',
   parse_dates=['Start Timestamp', 'Complete Timestamp'],
   infer_datetime_format=True)

# Tell PM4PY about which columns represent case ID,
# activity name, and timestamp. Case ID and activity
# names must be string type
log['case:concept:name']=log['Case ID'].astype('string')
log['concept:name']=log['Activity'].astype('string')
log['time:timestamp']=log['Complete Timestamp']
log['org:resource']=log['Resource']
```

PM4PY Basics

Reading an XES file is easy:

```
log2 = pm4py.read_xes('BPI_Challenge_2012.xes.gz')
```

PM4PY Basic Statistics

```
num_cases = len(log['Case ID'].unique())
num_events = log.shape[0]

pm4py.get_start_activities(log)
pm4py.get_end_activities(log)

pm4py.get_all_case_durations(log)

# Useful only or XES-based event logs
pm4py.get_event_attributes(log)
pm4py.get_trace_attributes(log)
```

MEMORIAL UNIVERSITY

PM4PY Basic Statistics

- Variants are sets of traces with the same sequence of events

```
varients = pm4py.get_variants(log)
# Returns a dict with keys and values
list(variants.keys())
list(variants.values())

# Split the log into sub-logs
for variant, subdf in pm4py.split_by_process_variant(log):
    print(variant)
    print(subdf)
```

Process Discovery – DFG

Directly-Follows Graph (Dependency Graph) (Process Map)

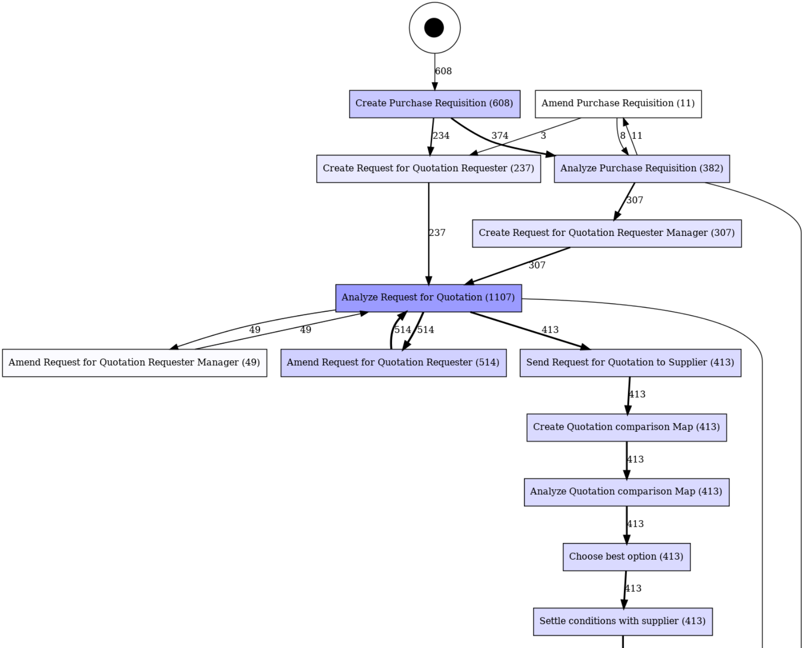
- Shows how often one activity directly follows another

```
dfg, start, end = pm4py.discover_dfg(log)

pm4py.view_dfg(dfg, start, end, rankdir='LR')

pm4py.save_vis_dfg(dfg=dfg,
    start_activities=start,
    end_activities=end,
    file_path='dfg.png', rankdir='TB')
```

Process Discovery



Hands-On Exercises

Use the event log from
`https://evermann.ca/busi4720/BPI_Challenge_2012.csv`

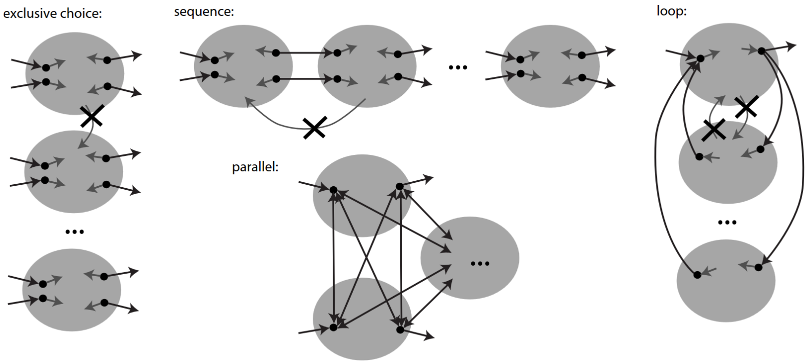
1. Read the data into a Pandas data frame using `read_csv()`. Parse the `Start Timestamp` and `Complete Timestamp` columns as dates.
2. Create the `case:concept:name`, `concept:name`, `time:timestamp`, `org:resource` columns as in the example above.
3. Create and visualize a directly-follows-graph (DFG)
4. How many variants are there? What is the most frequent variant?

Process Discovery – Inductive Miner

Principles

- Identifies subsets of activities by repeatedly ”cutting” the DFG
- Filter infrequent activities to deal with noise
- Mines a process tree that can be transformed to BPMN

Process Discovery – Inductive Miner



Source: Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P. (2013). Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In: Colom, JM., Desel, J. (eds) Application and Theory of Petri Nets and Concurrency. PETRI NETS 2013. Lecture Notes in Computer Science, vol 7927. Springer, Berlin, Heidelberg.

MEMORIAL UNIVERSITY

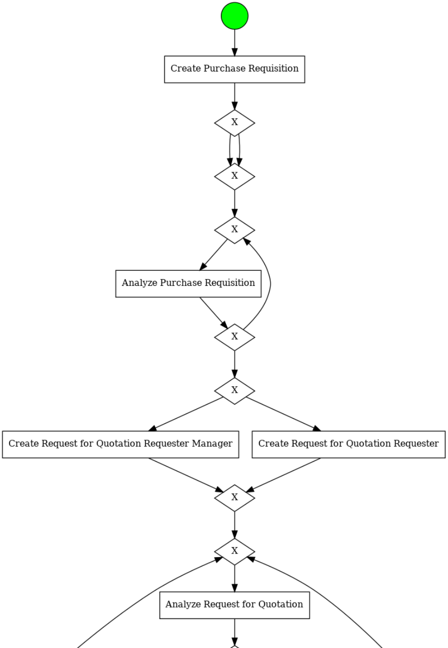
Process Discovery – Inductive Miner

```
bpmn_model = \
    pm4py.discover_bpmn_inductive(log, noise_threshold=0.5)

pm4py.view_bpmn(bpmn_model, rankdir='LR')

pm4py.save_vis_bpmn(bpmn_model,
   file_path='bpmn.png', rankdir='TB')
```

Process Discovery – Inductive Miner



Process Discovery – Heuristics Net

Principles

- Remove noise by frequency threshold on dependency graph
- Identifies loops of length 1 and length 2
- Identifies long-distance dependencies
- Removes non-observable activities

Frequencies in DFG

$$a \Rightarrow b = \left(\frac{|a > b| - |b > a|}{|a > b| + |b > a| + 1} \right)$$

MEMORIAL UNIVERSITY

Process Discovery – Heuristics Net

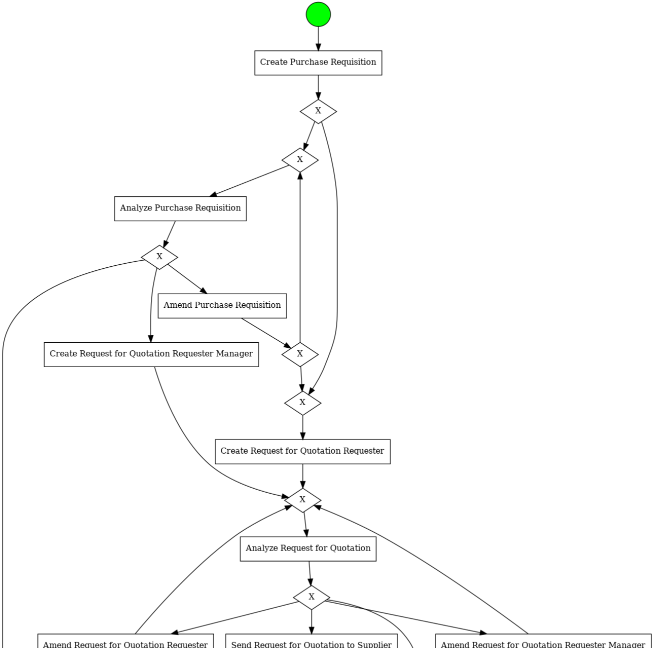
```
petri_net, initial_marking, final_marking = \
    pm4py.discover_petri_net_heuristics(log,
        dependency_threshold=0.6,
        and_threshold=0.65,
        loop_two_threshold=0.4)

pm4py.view_petri_net(petri_net)

bpmn_model2 = pm4py.convert_to_bpmn(
    petri_net, initial_marking, final_marking)

pm4py.view_bpmn(bpmn_model2)
pm4py.save_vis_bpmn(bpmn_model2, 'bpmn2.png', rankdir='TB')
```

Process Discovery – Heuristics net



Quality of Discovered Models

- **Fitness**: Can the model generate all traces in log?
- **Precision**: Does the model only generate traces in log?
- **Generalization**: Can the model generalize to ”sensible” traces not seen in log?
- **Complexity**: Is the model too complex to understand?

Fitness & Precision

Token-Based Replay

- Replays each trace of an event log on a process model
- Discovers missing and surplus tokens, i.e. model activities that cannot be executed, or model activities that are executed too often
- Percentage of traces that fit perfectly, average fitness

Alignments

- Aligns traces to process model
- "sync move": Activity in both trace and model
- "move on log": Activity in trace but not in model
- "mode on model": Activity in model but not in trace
- Percentage of traces that fit perfectly, average fitness

Fitness

```
petri_net, initial_marking, final_marking = \
    pm4py.discover_petri_net_inductive(log,
        noise_threshold=0.5)

fitness_alignments = pm4py.fitness_alignments(log,
    petri_net, initial_marking, final_marking)
print(fitness_alignments)

fitness_tbr = pm4py.fitness_token_based_replay(log,
    petri_net, initial_marking, final_marking)
print(fitness_tbr)
```

MEMORIAL UNIVERSITY

Precision

```
precision_alignments = pm4py.precision_alignments(log,
    petri_net, initial_marking, final_marking)
print(precision_alignments)

precision_tbr = pm4py.precision_token_based_replay(log,
    petri_net, initial_marking, final_marking)
print(precision_tbr)
```

Log Filtering

- Focus on subsets of logs
- Identify differences
- Simplify discovered models

Example Filters

`filter_activities_rework`	Keep cases where the specified activity occurs at least *n* times
`filter_case_size`	Keep cases having a length between *n* and *m* events
`filter_case_performance`	Keep cases having a duration between *n* and *m* seconds
`filter_directly_follows_relation`	Keep cases where *A* is followed immediately by *B*
`filter_end_activities`	Keep cases that end with the specified activity
`filter_event_attribute_values`	Keep cases or events in cases that satisfy the specified condition
`filter_eventually_follows_relation`	Keep cases where *A* is eventually followed by *B*
`filter_start_activities`	Keep cases that start with the specified activity
`filter_time_range`	Keep events occurring between two timestamps
`filter_trace_attribute_values`	Keep cases that satisfy the specified condition

MEMORIAL UNIVERSITY

Hands-On Exercises

Use the log from `https://evermann.ca/busi4720/PurchasingExample.csv'` and answer the following questions:

1. What are the types of activities in the log?
   - Use `unique()`
2. How often does each activity occur in the log?
   - Use `value_counts()`
3. Filter complete cases, that is, cases that end with activity "Pay invoice"
   - Use `pm4py.filtering.filter_end_activities`
4. Plot the case durations. What do you notice?
   - Use `pm4py.stats.get_all_case_durations`
   - Put case durations into a `pd.DataFrame`
   - 1 day = 86400 seconds
   - Use `px.histogram` or `pm4py.vis.view_case_duration_graph`

MEMORIAL UNIVERSITY

Hands-On Exercises [cont'd]

5. What is the mean case duration?
   - Use `mean()`
6. Split the log on the mean case duration
   - Use `pm4py.filtering.filter_case_performance`
7. Create BPMN models for each partial log and compare them. How do they differ?
8. Create a BPMN model for the total log. Compare the fitness and precision values compared to the partial logs.

Hands-On Exercises

9. What is the activity with the longest mean time?
   - Activities taking a long time may be bottle-neck in the process flow
   - Create a new column as the difference between the 'Complete Timestamp' and 'Start Timestamp' columns
   - Use `groupby()` and `mean()` on the data frame
10. What is the mean number of activities for each case?
    - Long cases may indicate problems
    - Calculate the number of activities for each case using `groupby()` and `count()` on the dataframe

MEMORIAL UNIVERSITY

Hands-On Exercises

11. Which activities are carried out more than once for some case
    - Repeated activities may indicate re-work or fixing of mistakes
    - Calculate the number of instances for each case for each activity using `groupby()` and `count()` on the dataframe
12. Are there cases that contain activity 'Pay invoice' but do not contain activity 'Send invoice'?
    - Non-compliant cases may represent a problem with controls and compliance
    - Use `pm4py.filtering.filter_eventually_follows_relatio`

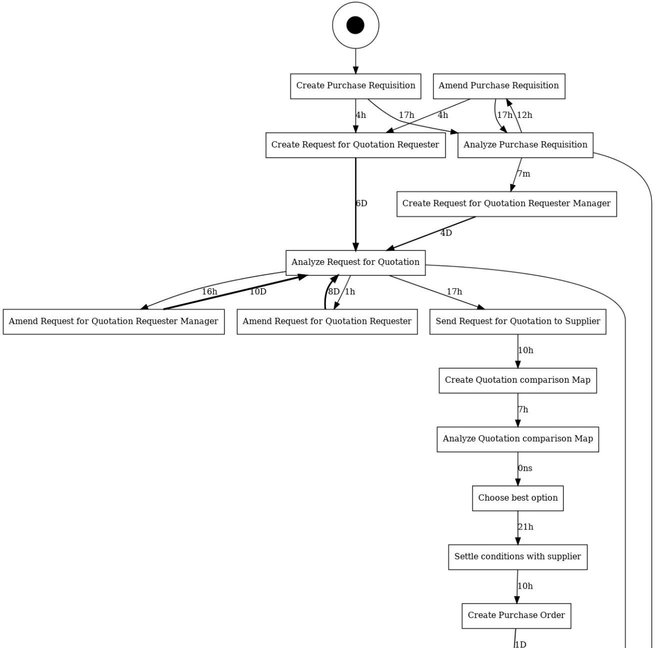
Performance Mining — DFG

- Identify median (mean, min, max, sum, stdev) waiting times

```
perf_dfg, start_activities, end_activities = \
    pm4py.discover_performance_dfg(log)

pm4py.view_performance_dfg(perf_dfg,
    start_activities, end_activities,
    aggregation_measure='median')
pm4py.save_vis_performance_dfg(perf_dfg,
    start_activities, end_activities,
    file_path='perfdfg.png', rankdir='TB')
```

Performance Mining — DFG

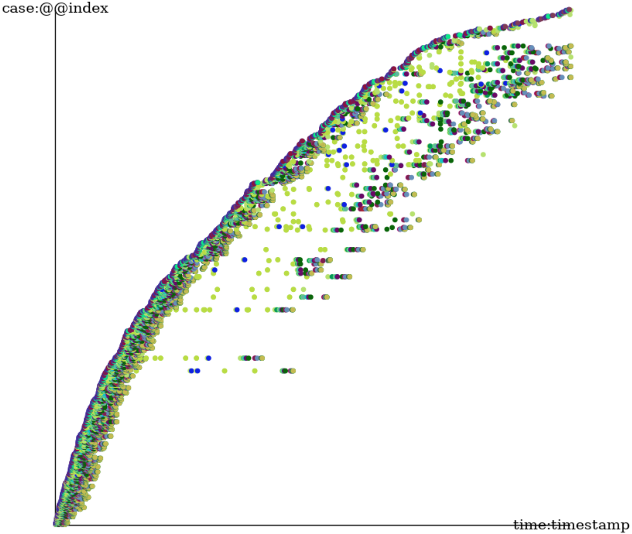


Performance Mining — Dotted Chart

- Identify batching of activities
- Identify different variants
- Case arrival and case finishing rates

```
pm4py.view_dotted_chart(log, show_legend=False)
pm4py.save_vis_dotted_chart(log,
    'dottedchart.png', show_legend=False)
```

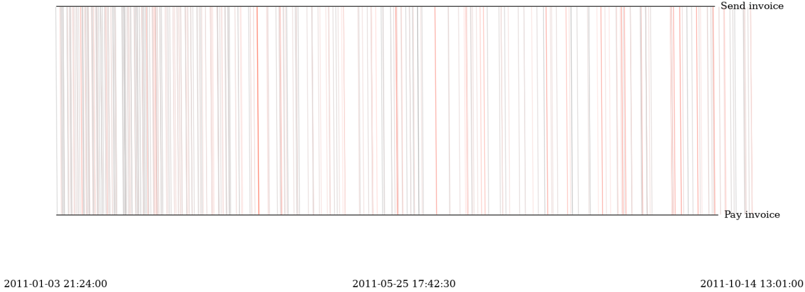
Performance Mining — Dotted Chart



- Identify variations in wait times

```
pm4py.view_performance_spectrum(log,
    ['Send invoice', 'Pay invoice'])
pm4py.save_vis_performance_spectrum(log,
    ['Send invoice', 'Pay invoice'],
    'perfspectrum.png')
```

Performance Mining — Performance Spectrum

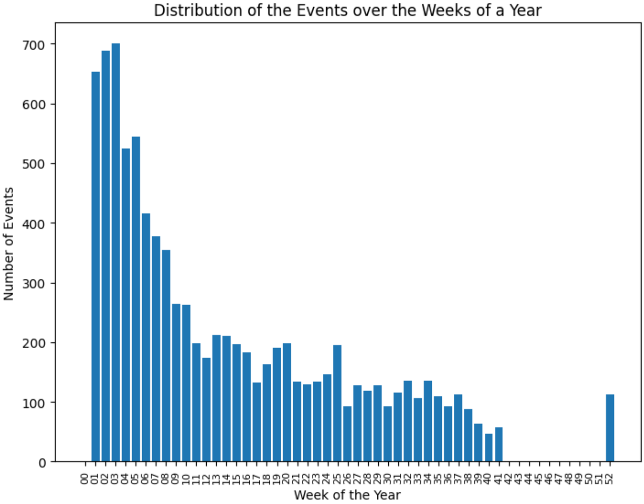


Performance Mining — Event Distribution

- Identify distribution of when events/activities occur

```
pm4py.view_events_distribution_graph(log, 'days_week')
pm4py.view_events_distribution_graph(log, 'days_month')
pm4py.view_events_distribution_graph(log, 'months')
pm4py.view_events_distribution_graph(log, 'weeks')
```

Performance Mining — Event Distribution

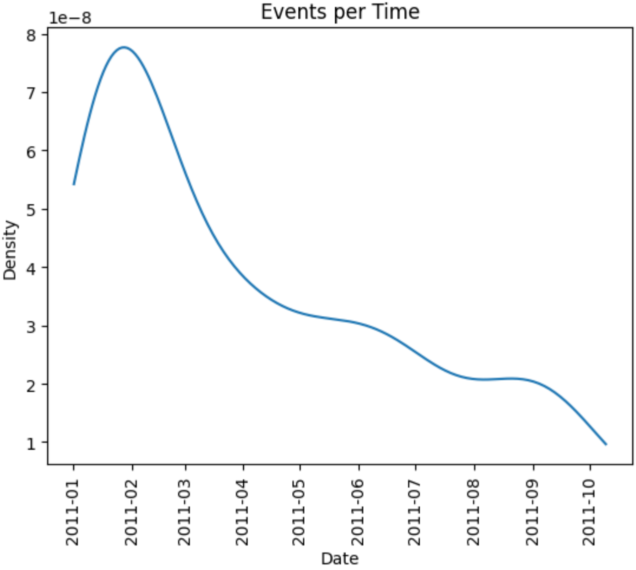


MEMORIAL UNIVERSITY

Performance Mining — Events per Time

```
pm4py.view_events_per_time_graph(log)
pm4py.save_vis_events_per_time_graph(log, 'eventspertime.png')
```

Performance Mining — Events per Time



Activity-Based Resource Similarity

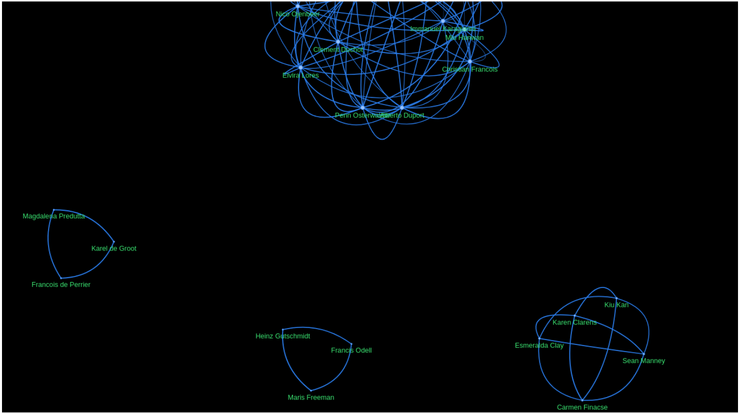
- Identify similar resources based on the set of activities they perform

```
sna_graph = \
 pm4py.discover_activity_based_resource_similarity(log)

pm4py.view_sna(sna_graph, variant_str='networkx')
pm4py.view_sna(sna_graph, variant_str='pyvis')

pm4py.save_vis_sna(sna_graph, 'ressimilarity.png',
    variant_str='networkx')
```

MEMORIAL UNIVERSITY


Nico Ojenbeer
Immanuel Karagianni
Miu Hanwan
Clement Duchot
Elvira Lores
Christian Francois
Alberto Duport
Magdalena Predutta
Karel de Groot
Francois de Perrier
Heinz Gutschmidt
Francis Odell
Maris Freeman
Kiu Kan
Karen Clarens
Esmeralda Clay
Sean Manney
Carmen Finacse

Handover of Work Network

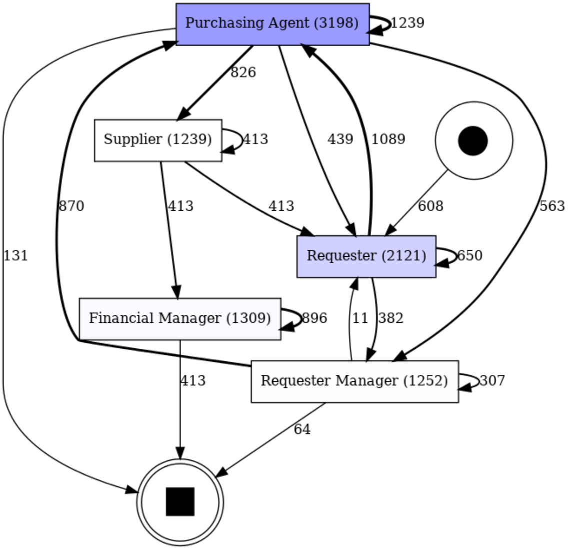
- Identify resources that pass work from one to another
- A DFG for resources instead of activities

```
dfg, start, end = pm4py.discover_dfg(log, activity_key='Role')
pm4py.view_dfg(dfg, start, end, rankdir='LR')

pm4py.save_vis_dfg(dfg=dfg,
    start_activities=start,
    end_activities=end,
    file_path='handover.png', rankdir='TB')
```

Organizational Mining

Handover of Work Network



Organizational Mining

Organizational Roles

- Identify similar resources based on the set of activities they perform

```
roles = pm4py.discover_organizational_roles(log)
print(roles)
```

Working-Together Network

- Resources work together, if they collaborate on some trace

```
sna_graph = pm4py.discover_working_together_network(log,
   resource_key='Role')
pm4py.view_sna(sna_graph, variant_str='pyvis')
pm4py.view_sna(sna_graph, variant_str='networkx')
```

Oganizational Mining

Working-Together Network