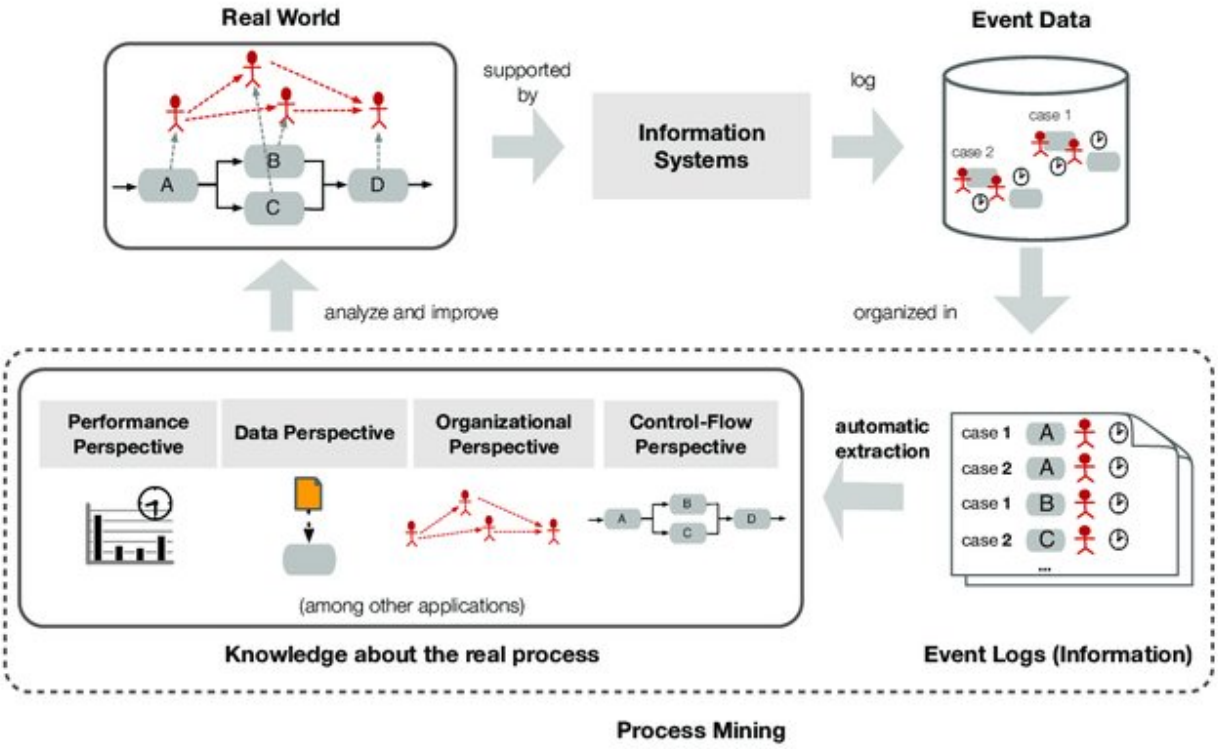
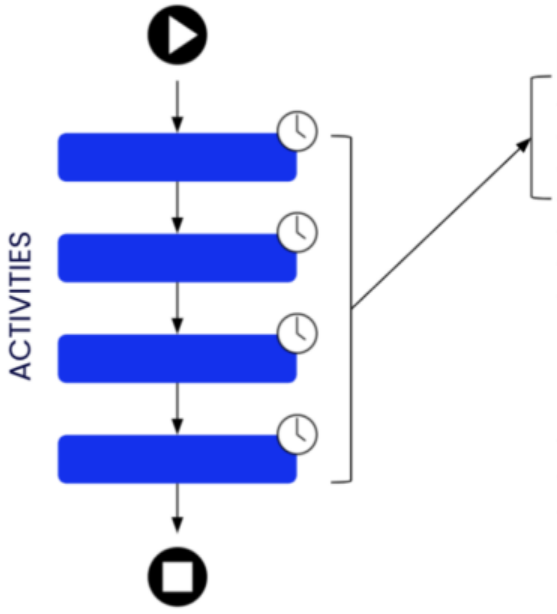
Process Mining with PM4Py





Process information is stored in **event logs**

CASE ID	ACTIVITY	TIMESTAMP
101	Create Order	04-29-2020 09:34:55
101	Approve Credit	04-29-2020 10:54:33
101	Create Delivery	04-30-2020 12:34:19
101	Ship Order	04-30-2020 18:54:38
201	Create Order	04-29-2020 09:34:55
202	Change Price	04-29-2020 09:34:55
203	Create Delivery	04-29-2020 09:34:55
204	Ship Order	04-29-2020 09:34:55
⋮		
N01	Create Order	03-15-2020 11:34:55
N02	Create Delivery	03-15-2020 13:22:57
N03	Change Quantity	03-15-2020 16:34:17
N04	Create Delivery	03-16-2020 02:15:09

Order Number	Activity	Employee	Date	Time
1337	Take Order	Lucy	April 1st 2020	1:37PM
1337	Note Address of Customer	Lucy	April 1st 2020	1:39PM
1337	Register Payment Method	Lucy	April 1st 2020	1:40PM
1337	Prepare Burger	Luigi	April 1st 2020	1:41PM
1337	Grab Soda	Lucy	April 1st 2020	1:42PM
1337	Put Burger in Box	Luigi	April 1st 2020	1:52PM
1337	Wrap Order	Lucy	April 1st 2020	1:53PM
1337	Deliver Order	Mike	April 1st 2020	1:55PM

Table 1: A simple example event log fragment, capturing the trace of process behavior for the first order described in the previous section.

1. ***Randy takes your order***
2. ***Randy notes down your preferred payment method***
3. ***Randy notes down your address***
4. ***Luigi prepares your burger***
5. ***Luigi puts your burger in a box***
6. ***Randy wraps your order***
7. ***John delivers your order***

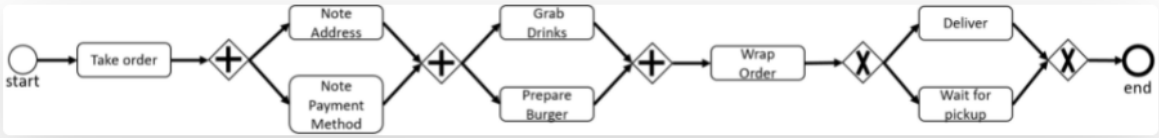


Figure 1: A simplified process model (using "Business Process Model and Notation", i.e., BPMN, notation) of the Burger Restaurant example process

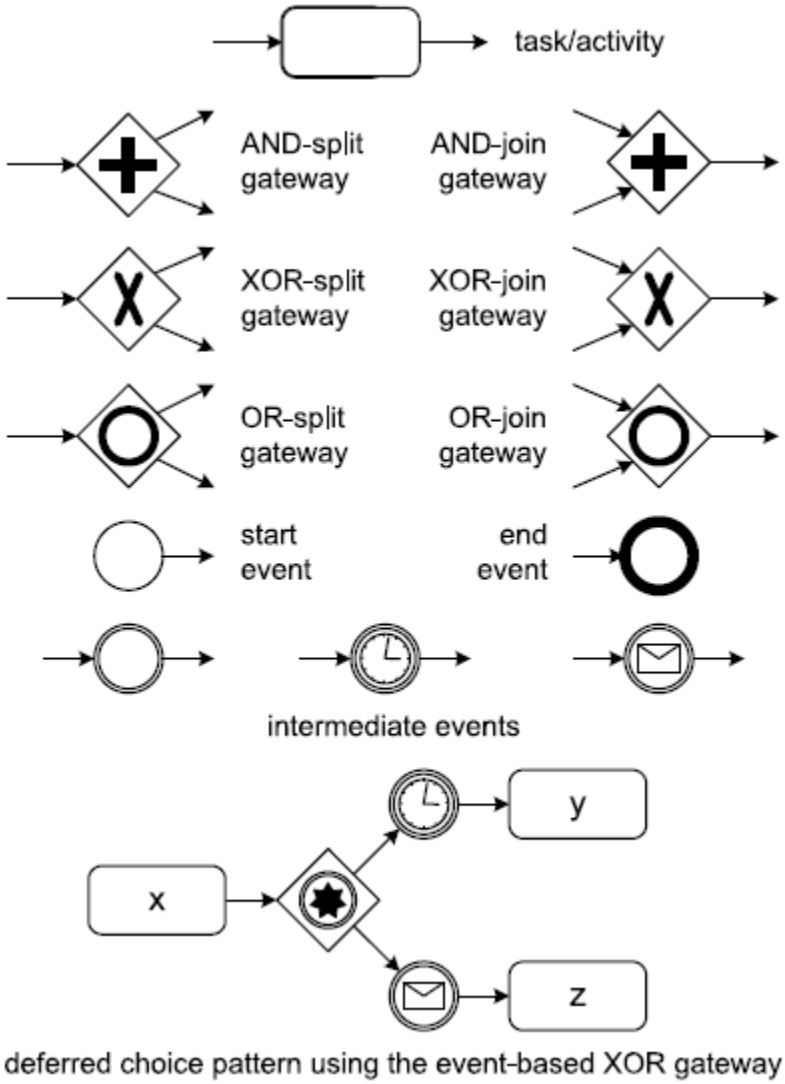


Figure 2: Fundamental elements of the "Business Process Model and Notation", i.e., BPMN, notation, taken from Process Mining: Data Science in Action; Wil M.P. van der Aalst (2016), page 69

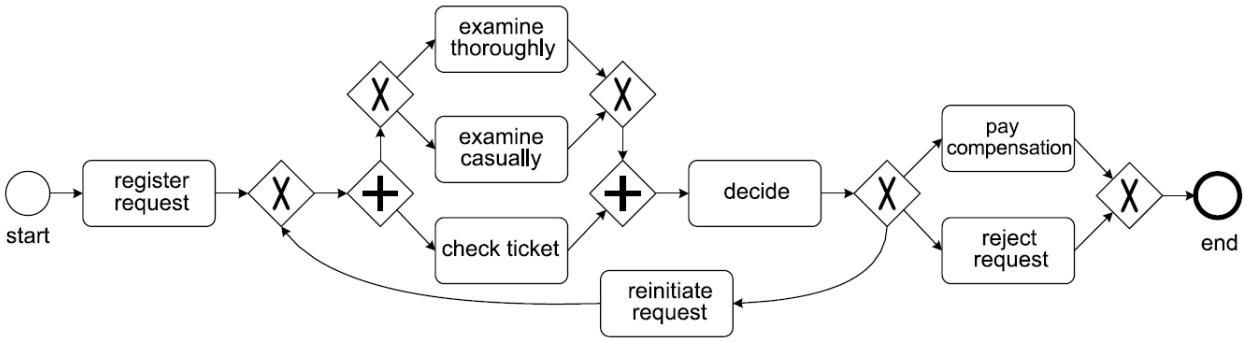
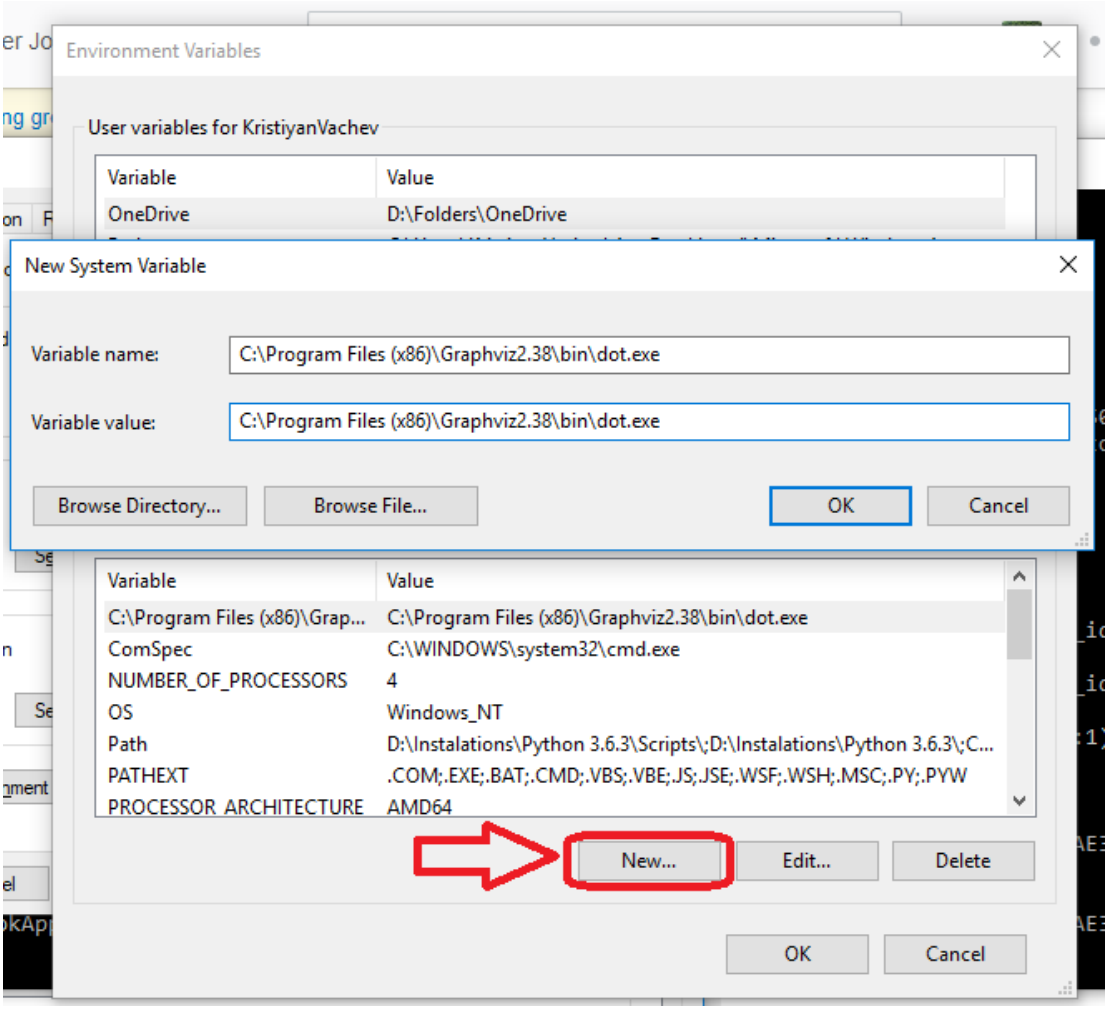


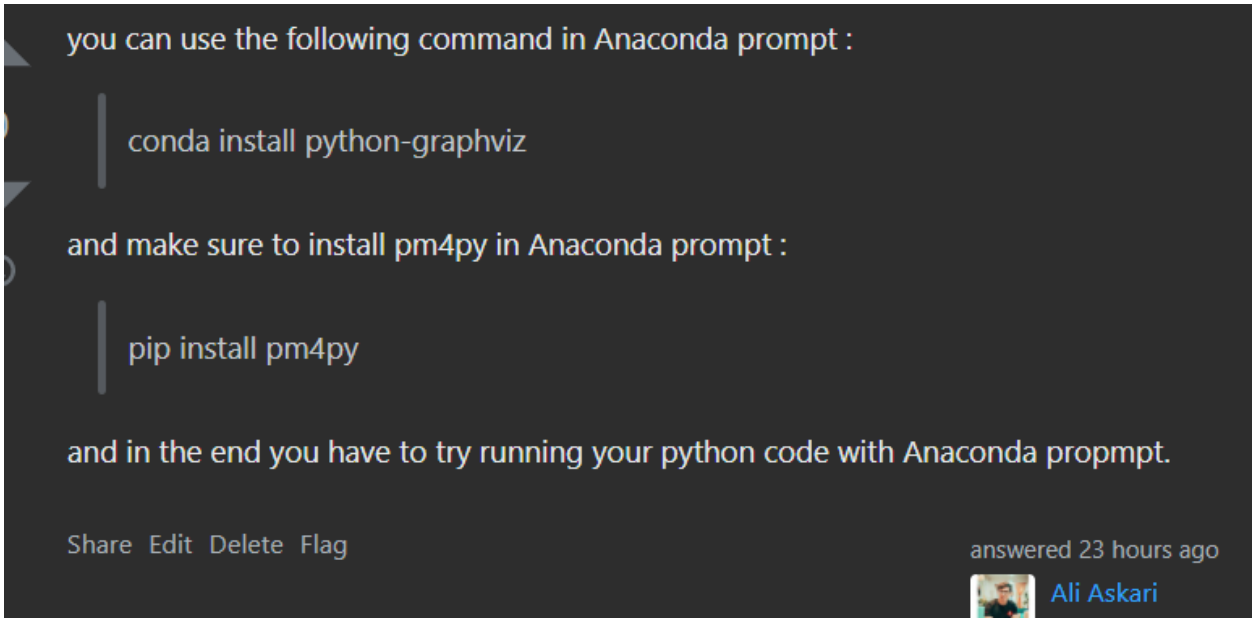
Figure 3: Running example BPMN-based process model describing the behavior of the simple process that we use in this tutorial.

Install PM4Py

"RuntimeError: Make sure the Graphviz executables are on your system's path" after installing Graphviz 2.38



Stack users advice for installation error



My advice for installation

Loading CSV Files

```
event_log =
pm4py.format_dataframe(pd.read_csv('C:/Users/ALI/Downloads/tamrins/running
-example.csv', sep=';'), case_id='case_id',
activity_key='activity', timestamp_key='timestamp')
```

Csv loading

Loading XES Files

```
log = pm4py.read_xes('C:/Users/ALI/Downloads/tamrins/running-example.xes')
```

XES loading

```xml
<?xml version='1.0' encoding='UTF-8'?>
<log xes.version="1849-2016">
  <string key="origin" value="csv"/>
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
  <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
  <extension name="Cost" prefix="cost" uri="http://www.xes-standard.org/cost.xesext"/>
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
  <trace>
    <string key="concept:name" value="1"/>
    <event>
      <string key="concept:name" value="register request"/>
      <date key="time:timestamp" value="2010-12-30T11:02:00.000+01:00"/>
      <int key="cost:total" value="50"/>
      <string key="org:resource" value="Pete"/>
      <int key="@@index" value="14"/>
    </event>
    <event>
      <string key="concept:name" value="examine thoroughly"/>
      <date key="time:timestamp" value="2010-12-31T10:06:00.000+01:00"/>
      <int key="cost:total" value="400"/>
      <string key="org:resource" value="Sue"/>
      <int key="@@index" value="15"/>
    </event>
    <event>
      <string key="concept:name" value="check ticket"/>
      <date key="time:timestamp" value="2011-01-05T15:12:00.000+01:00"/>
      <int key="cost:total" value="100"/>
      <string key="org:resource" value="Mike"/>
      <int key="@@index" value="16"/>
    </event>
    <event>
      <string key="concept:name" value="decide"/>
      <date key="time:timestamp" value="2011-01-06T11:18:00.000+01:00"/>
      <int key="cost:total" value="200"/>
      <string key="org:resource" value="Sara"/>
      <int key="@@index" value="17"/>
    </event>
```

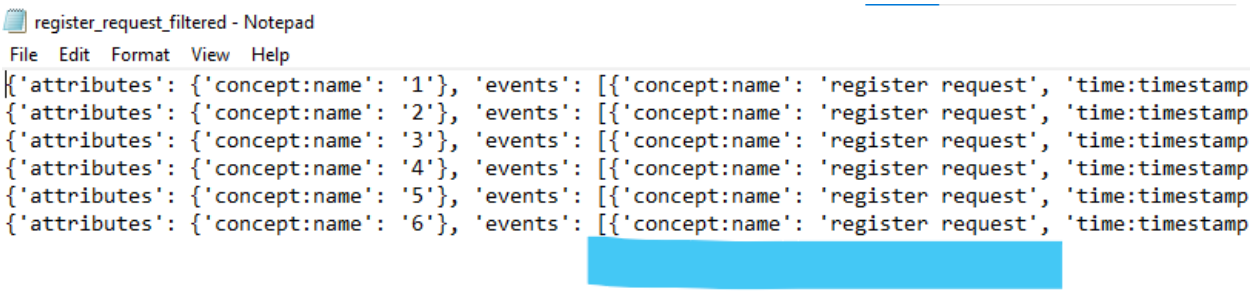
XES example dataset

Pre-Built Event Log Filters

```
#Pre-Built Event Log Filters
filtered = pm4py.filter_start_activities(log, {'register request'})
print(list[filtered])
with open('register_request_filtered.txt', 'w') as f:
    for i in filtered :
        f.write(str(i)+'\n')
```

Filter_start_activities

oE

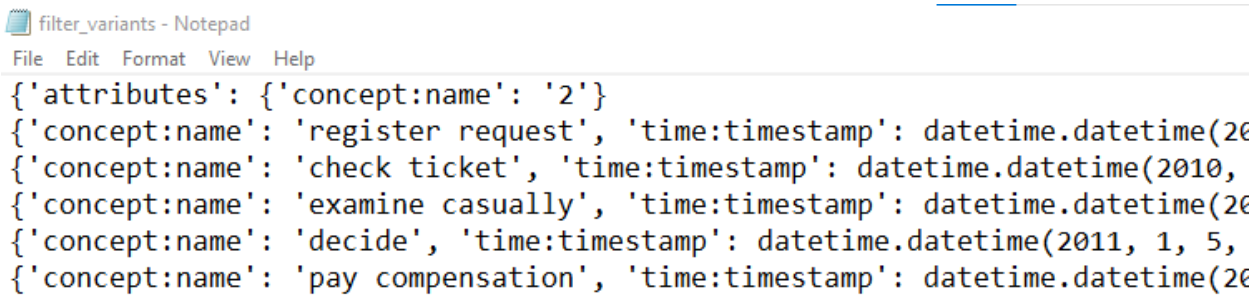


Output as a text

```python
filtered3 = pm4py.filter_variants(log, [
    ['register request', 'check ticket', 'examine casually', 'decide', 'pay compensation']])

print(str(filtered3))
with open('filter_variants.txt', 'w') as f:
    for i in filtered3 :
        for j in i :
            f.write(str(j)+'\n')
```

filter_variants



Output as a text

Obtaining a Process Model

```
import pandas as pd
event_log = pm4py.format_dataframe(pd.read_csv('C:/Users/ALI/Downloads/tamrins/running-example.csv', sep=';')
activity_key='activity', timestamp_key='timestamp')
#event_log.to_csv('C:/Users/ALI/Downloads/tamrins/running-example-exported.csv')
log = pm4py.read_xes('C:/Users/ALI/Downloads/tamrins/running-example.xes')


# Figure 6: BPMN model discovered based on the running example event data set, using the Inductive Miner impl
process_tree = pm4py.discover_tree_inductive(event_log)
bpmn_model = pm4py.convert_to_bpmn(process_tree)
pm4py.view_bpmn(bpmn_model)
```

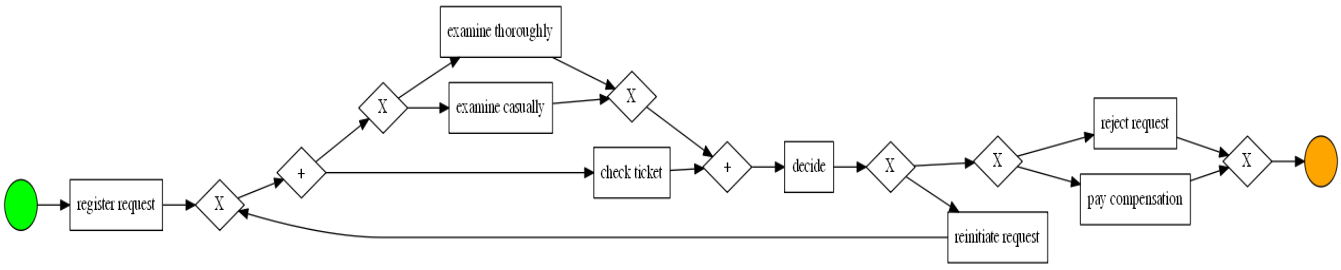


Figure 6: BPMN model discovered based on the running example event data set, using the Inductive Miner implementation of PM4Py.

```
#Figure 7: Process Tree model discovered based on the running example event data set, using
process_tree = pm4py.discover_tree_inductive(log)
pm4py.view_process_tree(process_tree)
```

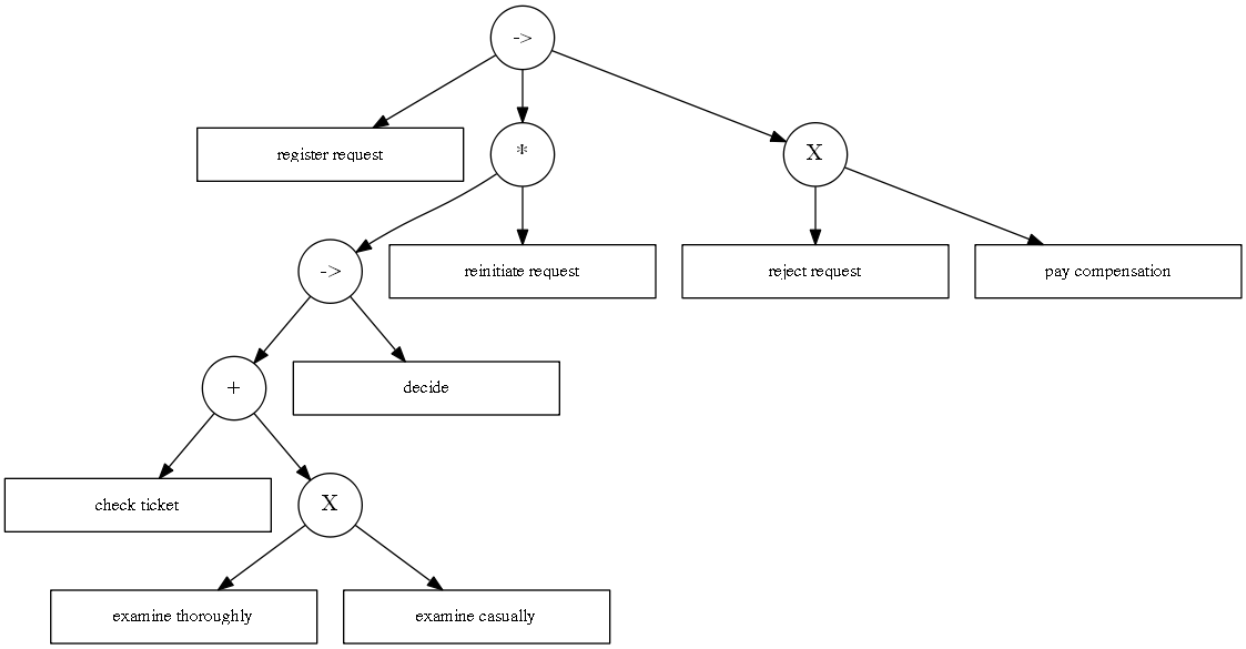


Figure 7: Process Tree model discovered based on the running example event data set, using the Inductive Miner implementation of PM4Py.

```
#Figure 8: Process Map (DFG-based) discovered based on the running example event data set.

dfg, start_activities, end_activities = pm4py.discover_dfg(log)
pm4py.view_dfg(dfg, start_activities, end_activities)
```

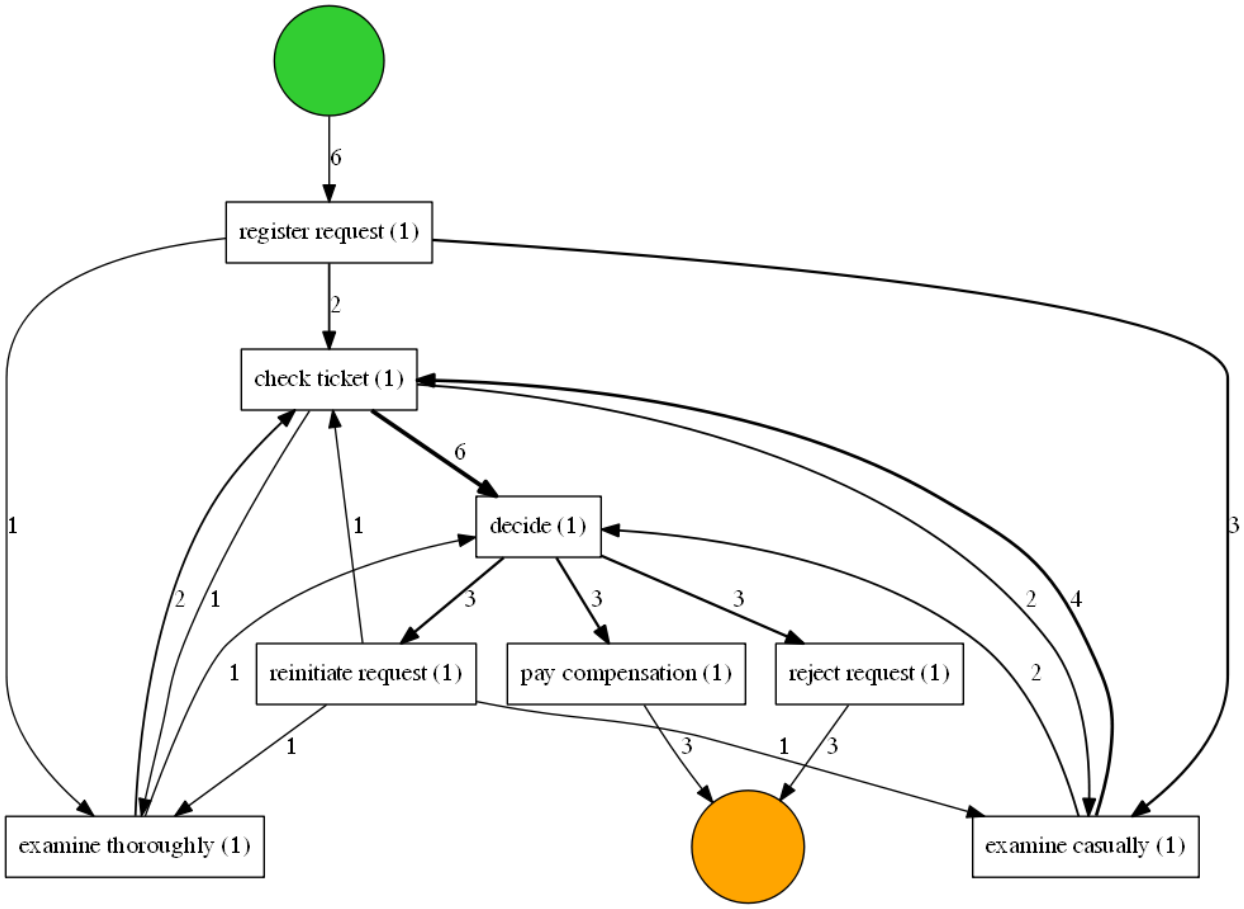
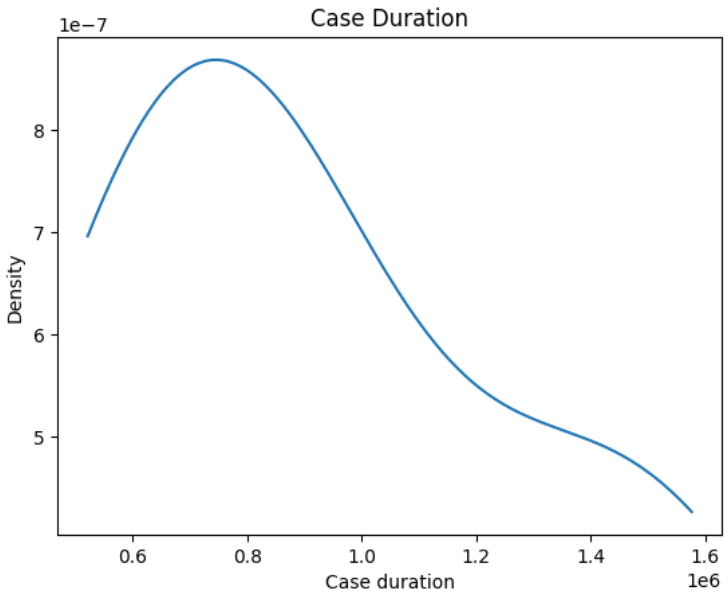
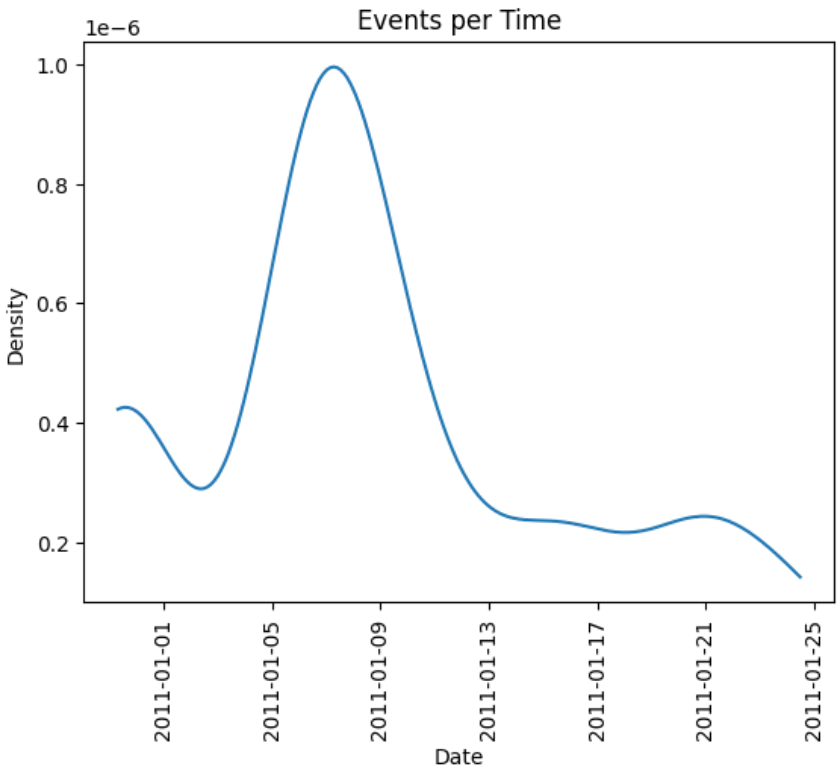


Figure 8: Process Map (DFG-based) discovered based on the running example event data set.

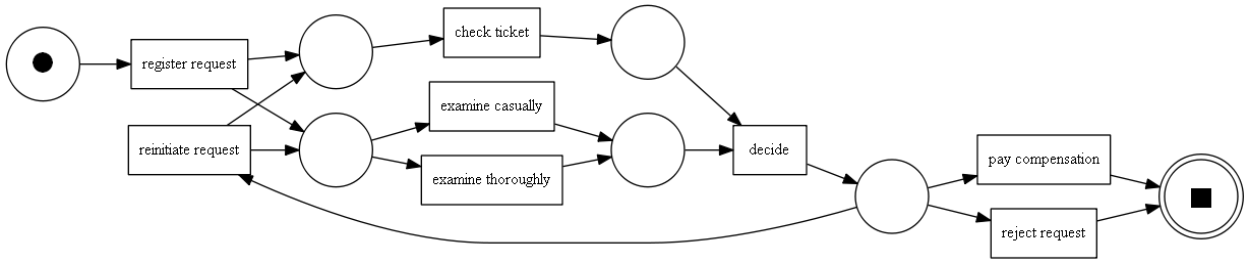


Distribution of case duration

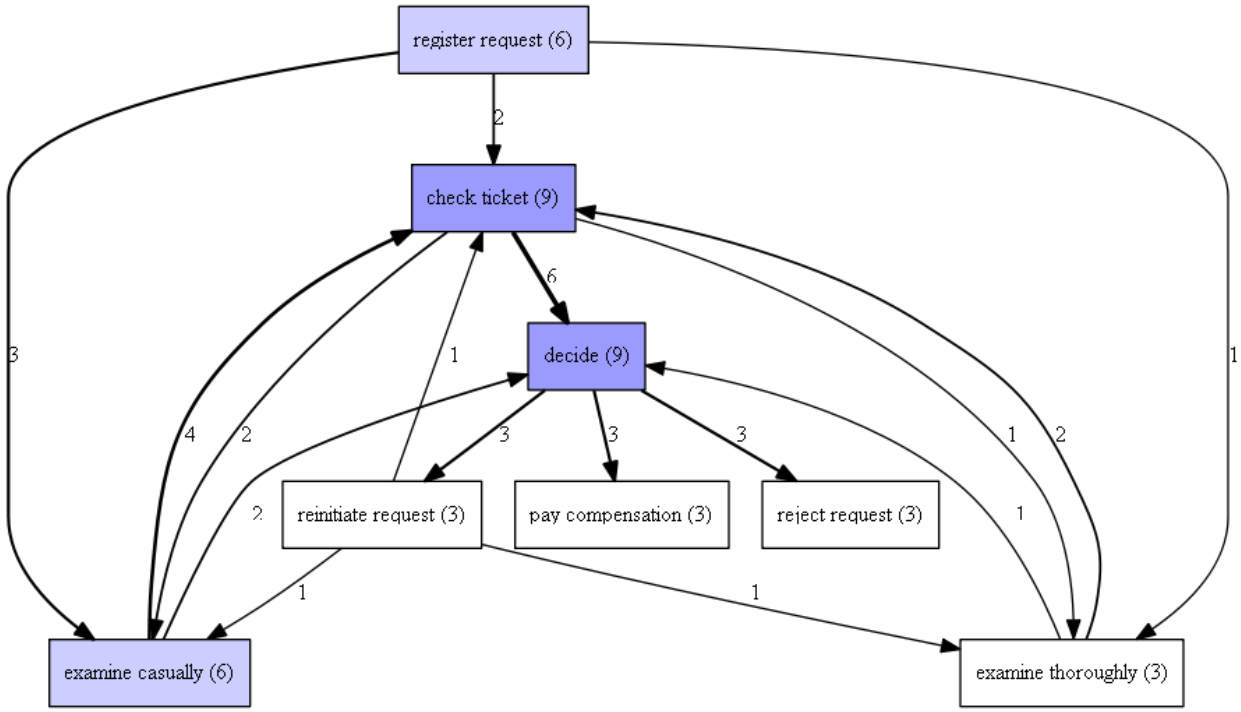


Distribution of events over time

Alpha Miner

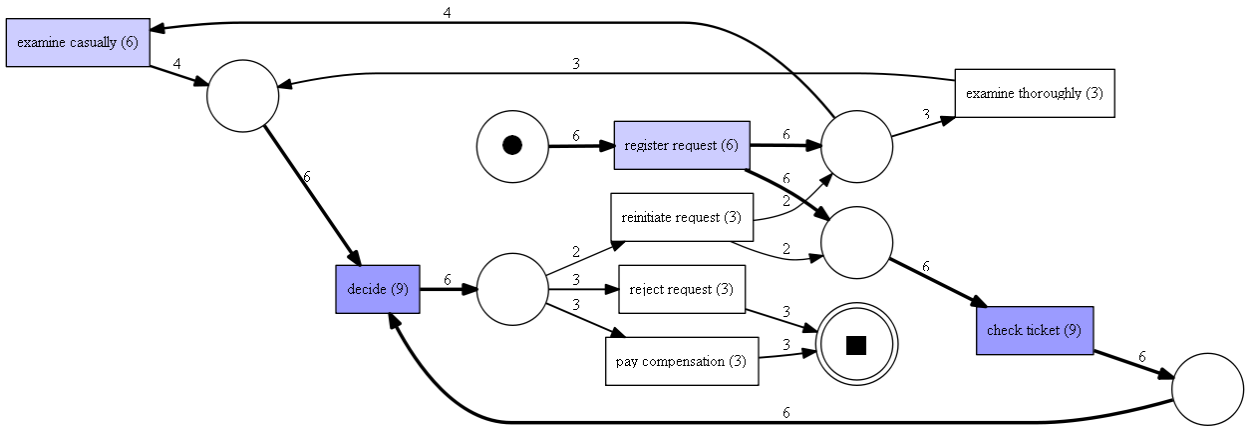


Directly-Follows Graph



Frequency/Performance

Similar to the Directly-Follows graph



PCA – Reducing the number of features

Some techniques (such as the clustering, prediction, anomaly detection) suffer if the dimensionality of the dataset is too high. Hence, a dimensionality reduction technique (as PCA) helps to cope with the complexity of the data.

Having a Pandas dataframe out of the features extracted from the log:

```
import pandas as pd

df = pd.DataFrame(data, columns=feature_names)
```

It is possible to reduce the number of features using a techniques like PCA.

Let's create the PCA with a number of components equal to 5, and apply the PCA to the dataframe.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=5)
df2 = pd.DataFrame(pca.fit_transform(df))
```

So, from more than 400 columns, we pass to 5 columns that contains most of the variance.

Rework (activities)

The rework statistic permits to identify the activities which have been repeated during the same process execution. This shows the underlying inefficiencies in the process.

{'check ticket': 2, 'decide': 2, 'examine casually': 1, 'reinitiate request': 1}

Rework (cases)

We define as rework at the case level the number of events of a case having an activity which has appeared previously in the case.

For example, if a case contains the following activities: A,B,A,B,C,D; the rework is 2 since the events in position 3 and 4 are referring to activities that have already been included previously.

{'1': {'number_activities': 5, 'rework': 0},
'2': {'number_activities': 5, 'rework': 0},
'3': {'number_activities': 9, 'rework': 2},
'4': {'number_activities': 5, 'rework': 0},
'5': {'number_activities': 13, 'rework': 7},
'6': {'number_activities': 5, 'rework': 0}}