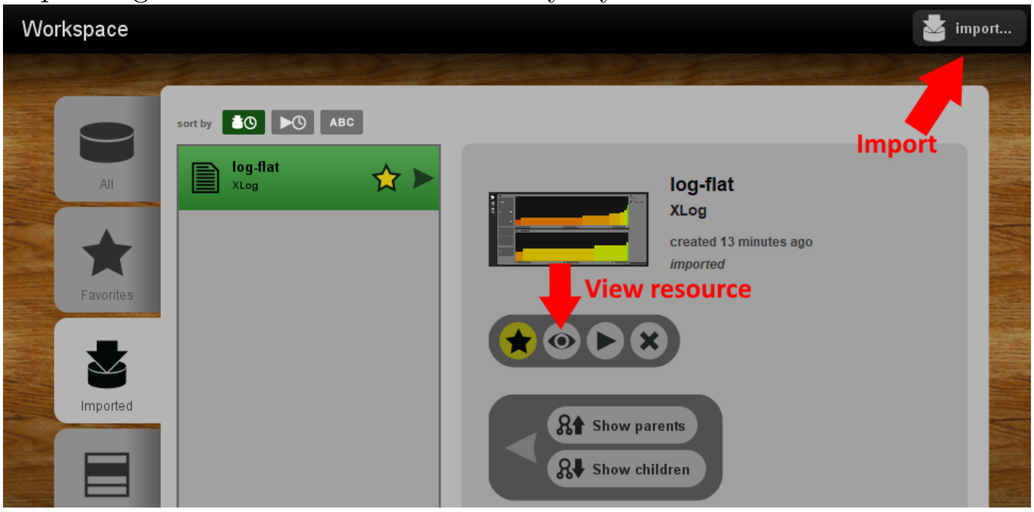
Question 1

(a)

Import log-flat.xes to ProM. Click the eye symbol to view resource.

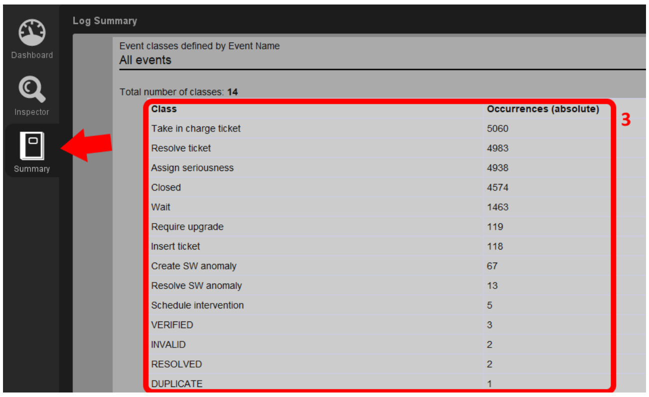


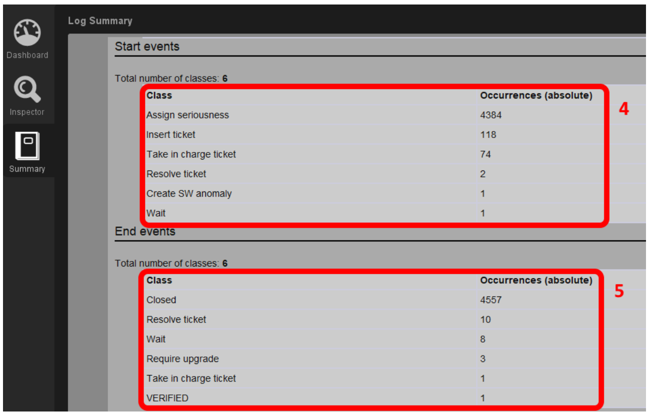
This results in the following view:

From it we can read

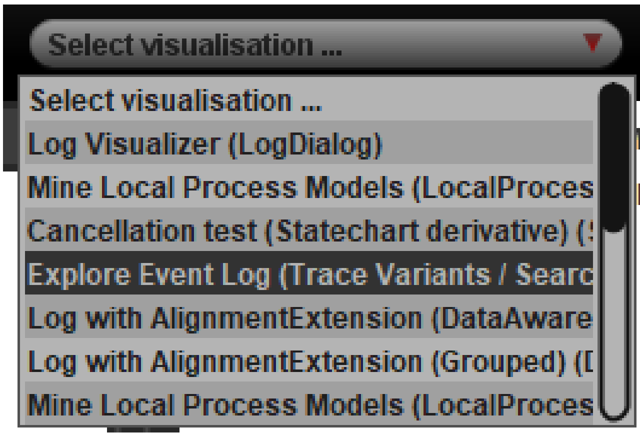
- the time period (1) covered by the event log, which is from 13.01.2010 to 03.01.2014
- the number of cases, events and activities of the log (2), being 4580, 21348 and 14 respectively (note that activities appear in ProM as event classes)

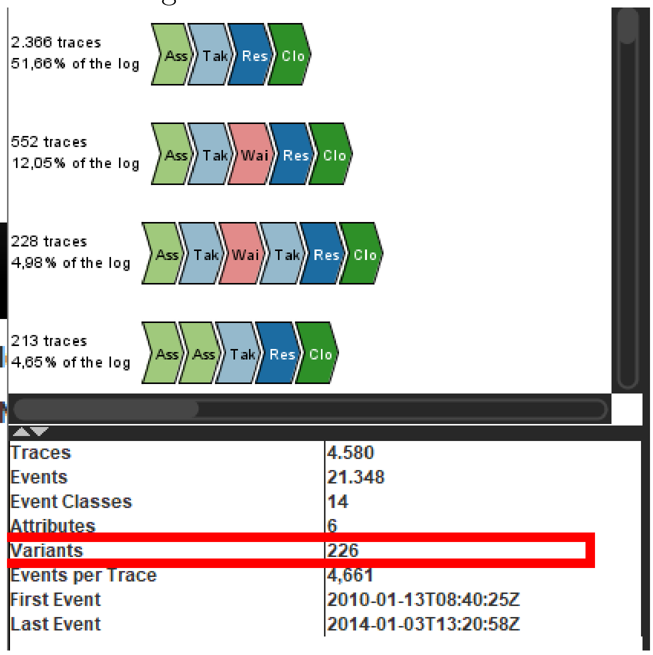
To gain more information on the activities we click the summary tab in the left which results in the following view:





From (3) we get a table of occurence frequencies for each activity. From (4) we get a table of occurence frequencies for each start activity. From (5) we get a table of occurence frequencies for each end activity.
To determine the number of unique trace variants we click on 'Select visualization' and select 'Explore Event Log'
Under this view all trace variants are listed and some further information is given. From this we learn that there are 226 unique variants in the event log.

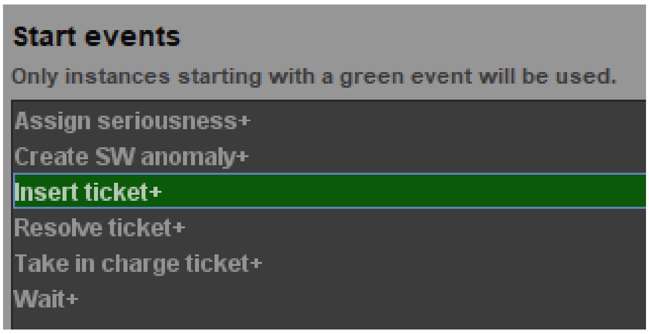




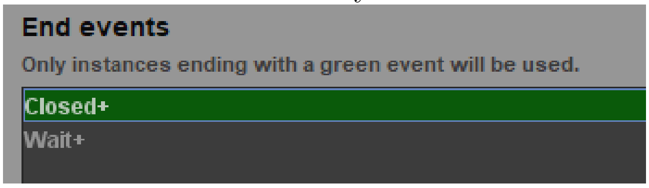
98% of tickets taken in charge are resolved ($\frac{4983}{5060}$) The variants seem to be quite diverse ($\tilde{1}$:20 ratio of cases to variants, although distributed very unevenly)

(b)

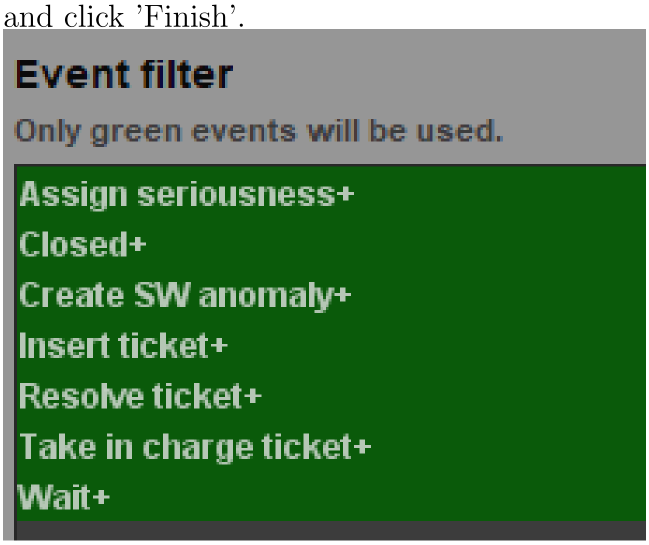
From the introduction we learned that every trace has to start with 'Insert ticket' and ends with 'Closed'. Therefore every trace that does not begin/end with these events must have started/ended outside of our observed time period, making it incomplete.
To filter out incomplete traces we go on the 'Actions' tab, select 'Filter Log using Simple Heuristics' and press 'Start'. In the first dialogue we just click 'Next'. In the next dialogue window we select 'Insert ticket' as our only start event and click 'Next'.



For the end events we only select 'Closed' and click 'Next'.



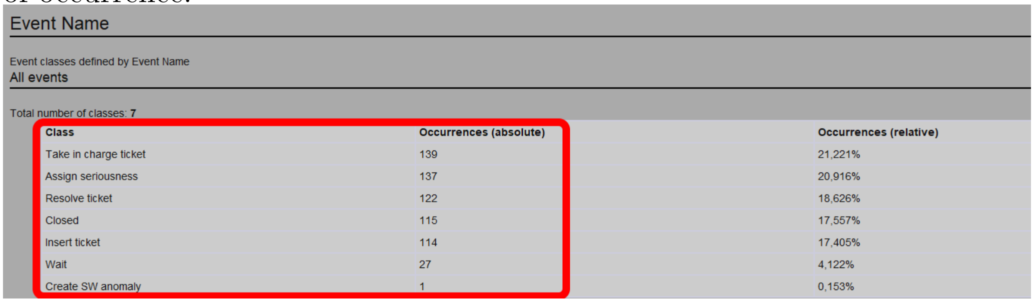
Since we do not want to filter out any other events we select 100% of events in Event Filter and click 'Finish'.



(c)

1. As in (a) we inspect the overview of *log-complete* to find 114 cases, 655 events and 7 activities. Also just like in (a) we select the visualization 'Explore Event Log' to find out there are 20 unique trace variants.

2. Just like in (a) we look at 'Summary' to find a table of activities along with their frequency of occurrence:


Event Name

Event classes defined by Event Name
All events

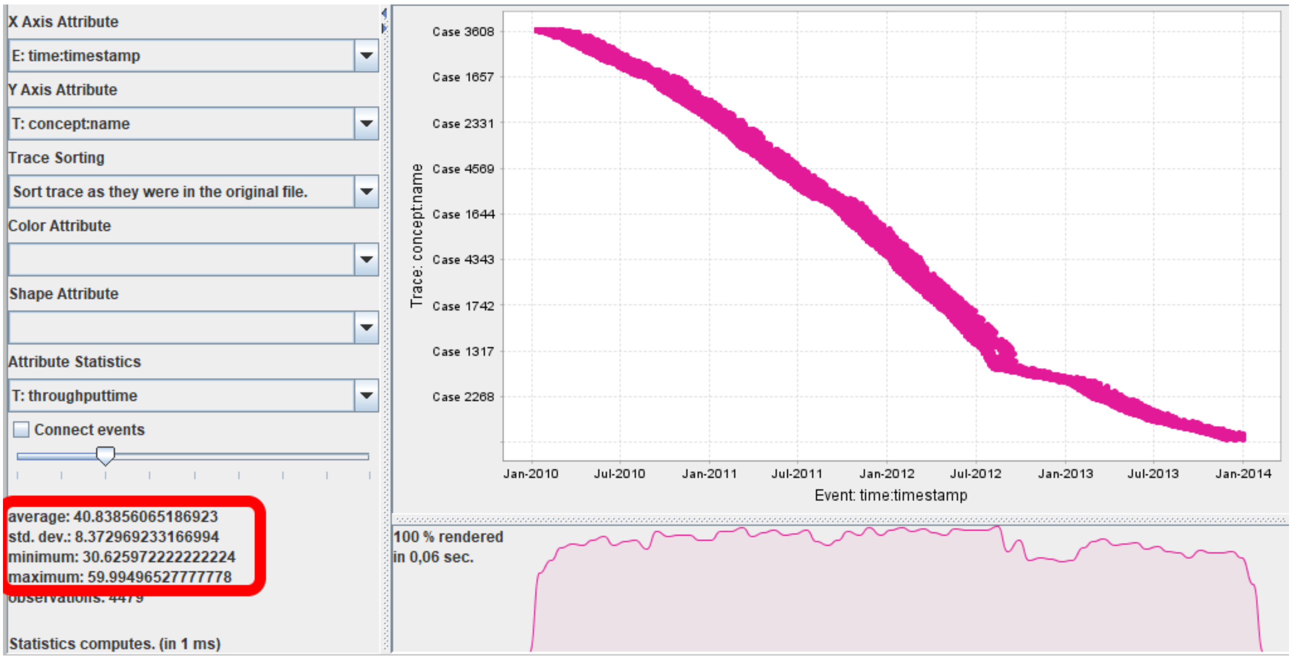
Total number of classes: 7

Class	Occurrences (absolute)	Occurrences (relative)
Take in charge ticket	139	21,221%
Assign seriousness	137	20,916%
Resolve ticket	122	18,626%
Closed	115	17,557%
Insert ticket	114	17,405%
Wait	27	4,122%
Create SW anomaly	1	0,153%

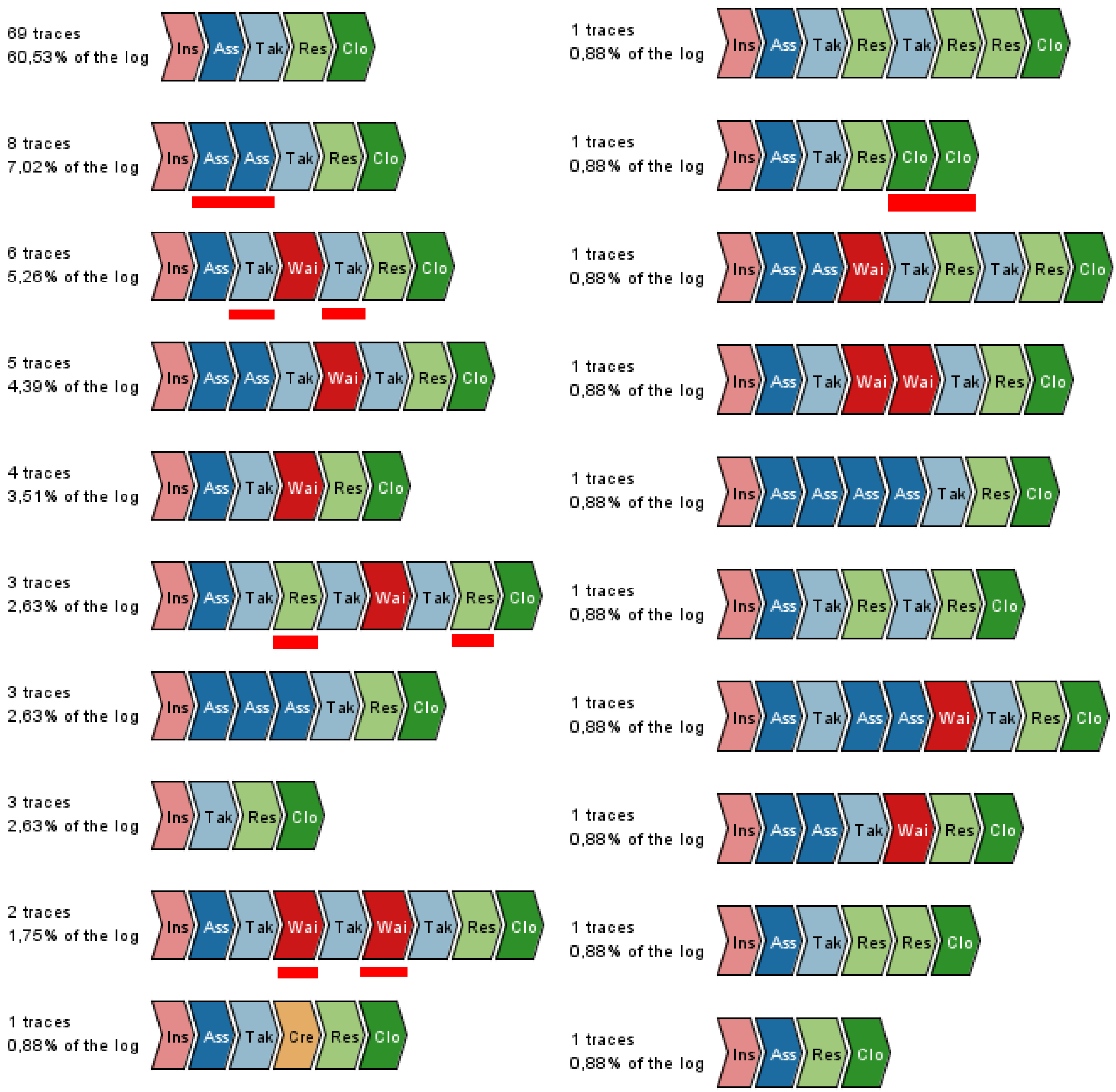
3.

4. We apply the 'Add Throughput Time as Trace Attribute (In place)' plugin to *log-complete* and select 'DAYS' as the resolution to be used for the elapsed time. We then select the visualization 'Dotted Chart' on the result. To retrieve minimum, maximum and average trace durations we select 'T: throughputtime' as Attribute Statistics and get the following rounded result:

- minimum trace duration: 31 days
- maximum trace duration: 60 days
- average trace duration: 41 days



5. To find out which activities appear more than once in at least one trace we again take a look at the 'Explore Event Log' visualization:



The tasks that appear more than once (as marked in the visualization) are

- Ass: Assign seriousness
- Tak: Take in charge ticket
- Res: Resolve ticket
- Wai: Wait
- Clo: Closed

(d)

1. We choose Pie Charts for these visualization. For *Ticket type* distribution we choose `TICKET TYPE` as dimension:

```
"case_table_csv"."TICKET TYPE"
```

and `COUNT(TICKET TYPE)` as KPI:

```
COUNT("case_table_csv"."TICKET TYPE")
```

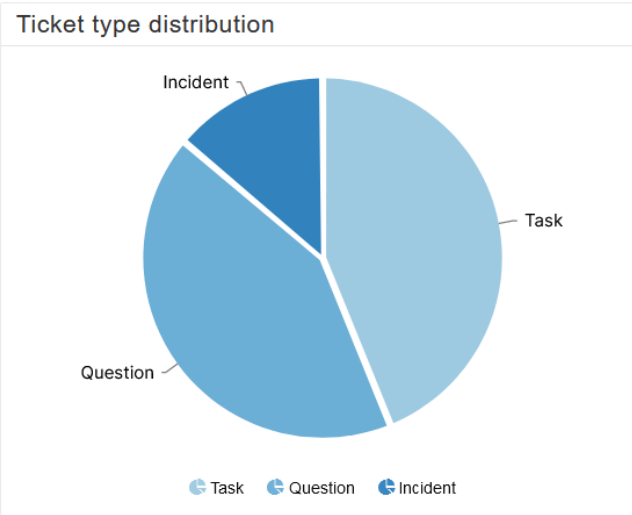
For *Membership* distribution we choose `MEMBERSHIP` as dimension:

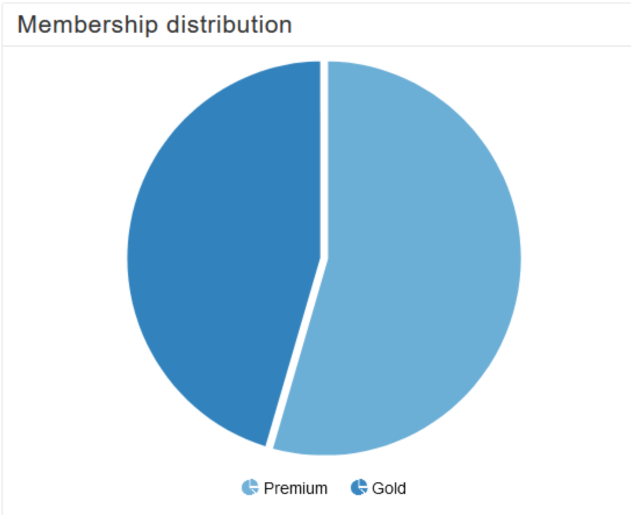
```
"case_table_csv"."MEMBERSHIP"
```

and `COUNT(MEMBERSHIP)` as KPI:

```
COUNT("case_table_csv"."MEMBERSHIP")
```

The resulting distribution visualization can be seen below.





2. We obtained the column chart titeled 'Total workload per ressource' by using

`"event_table_csv"."RESOURCE"`

as dimension and

`COUNT("event_table_csv"."ACTIVITY")`

as KPI. The x-Axis shows the resources 1-8 in ascending order, the y-Axis shows the summed number of activities handeled by the given resource.

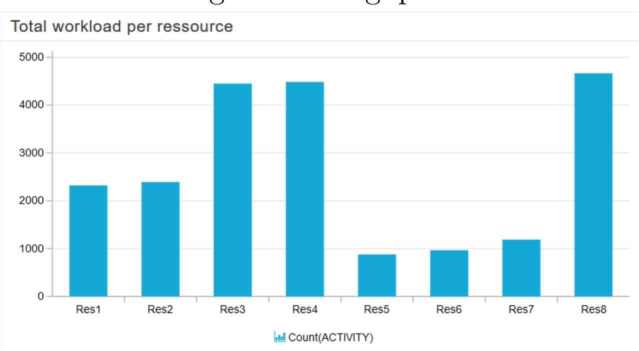
3. We created the column chart named 'Throughput times distribution' by selecting the total throughput time in days as our dimension:

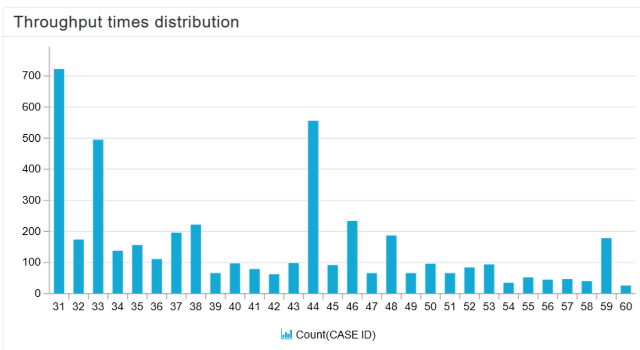
`CALC_THROUGHPUT(ALL_OCCURRENCE['Process Start'] TO ALL_OCCURRENCE['Process End'], REMAP_`

and selecting

`COUNT("case_table_csv"."CASE ID")`

as our KPI. The x-Axis shows the throughput time in days and the y-Axis shows the number of cases with a given throughput time.

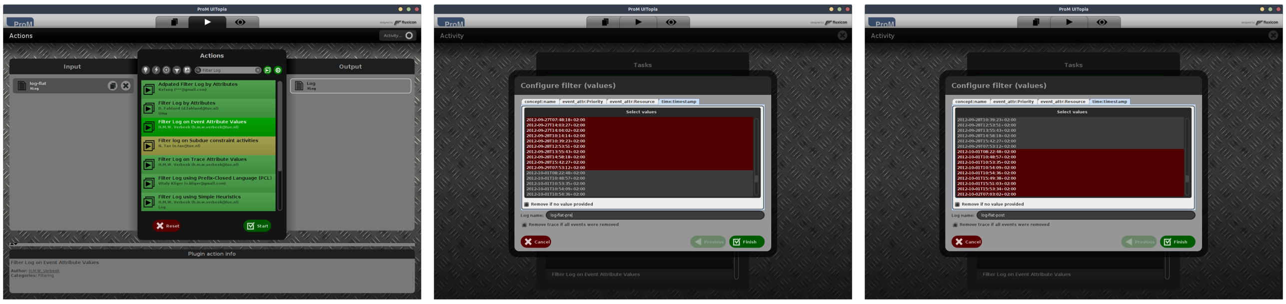




Question 2

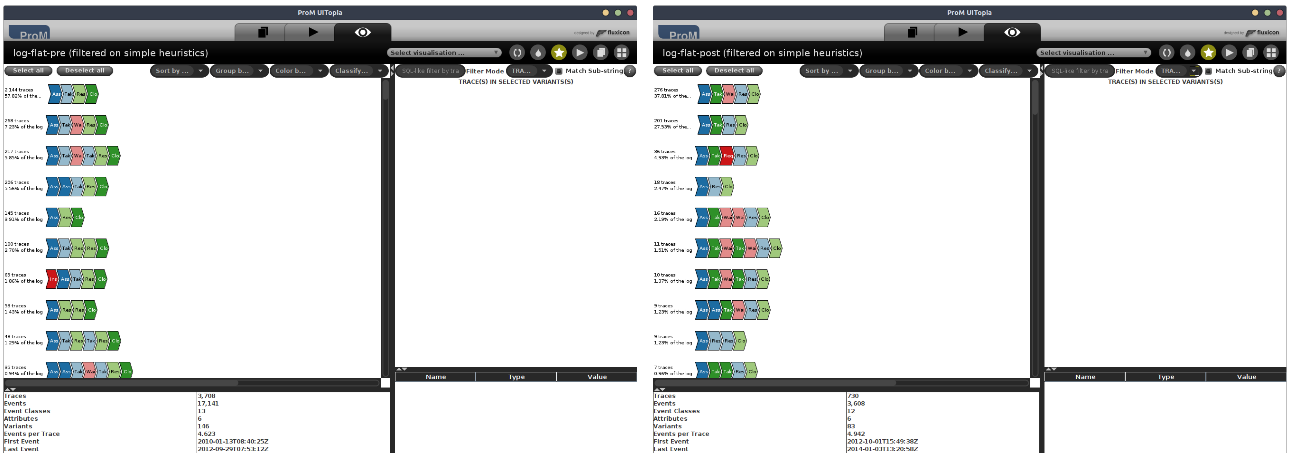
(a)

We can split the event log into two event logs using the plugin 'Filter Log on Event Attribute Values' with the following selections:



Afterwards, we can apply our filtering to only allow valid traces, as seen before.

For the event log pre 01.10.2012 we get 13 unique activities across 3708 cases, as a whole consisting of 6 variants.
For the event log post (including) 01.10.2012 we get 12 unique activities across 730 cases, as a whole consisting of 6 variants.



(b)

1. We apply the following workflow both for *log-pre-complete* and *log-post-complete*:

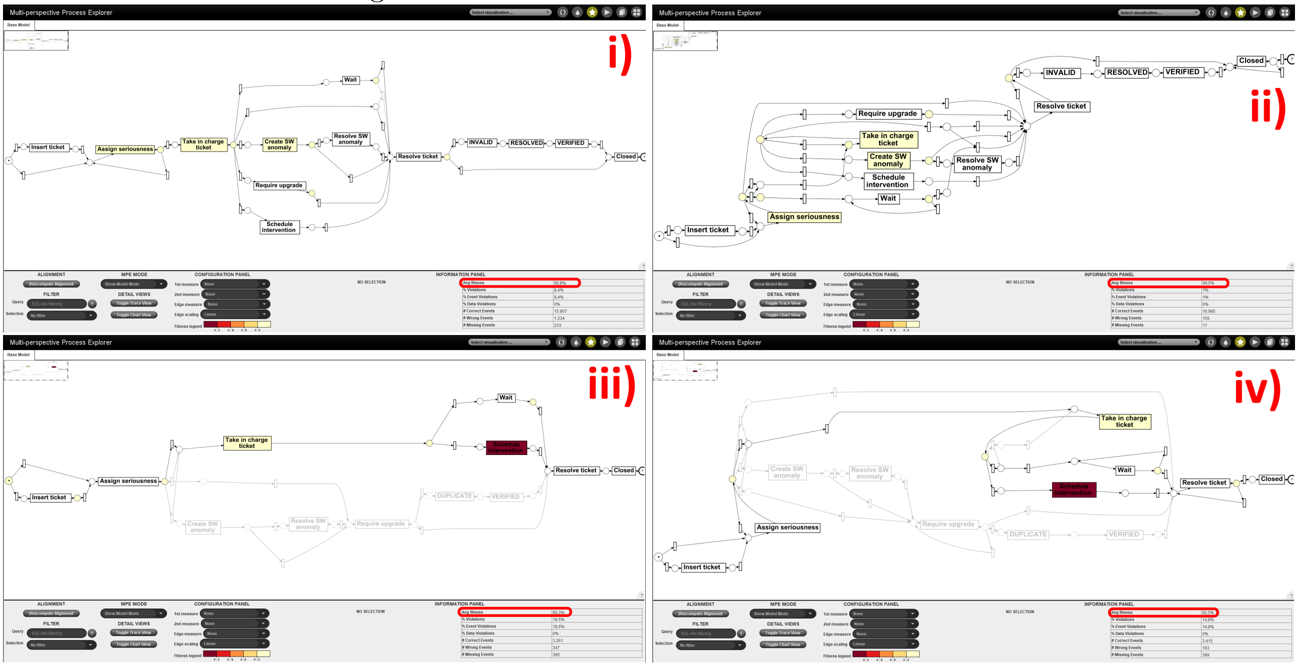
    1. Apply plugin 'Interactive Data-aware Heuristic Miner (iDHM)'

    2. Set 'Options & Thresholds' to
       All tasks connected: True

	i) (*log-pre-complete*)	ii) (*log-pre-complete*)	iii) (*log-post-complete*)	iv) (*log-post-comple*
Frequency	0.1	0	0.1	0
Dependency	0.9	0.9	0.9	0.9
Bindings	0.1	0.1	0.1	0.1
Conditions	0.5	0.5	0.5	0.5

3. Select 'Petri net' as 'Output: Process Model' and click 'Export model'
4. Use the log and the newly created Petri net as input for the 'Multi-perspective Process Miner' plugin

This results in the following Models:



As we can see all these models exceed our fitness threshold of 90% and contain all activities occuring in their respective logs (*log-pre-complete* doesn't contain the activity DUPLICATE and *log-post.complete* doesn't contain the activities INVALID and RESOLVED).

(c)

(d)

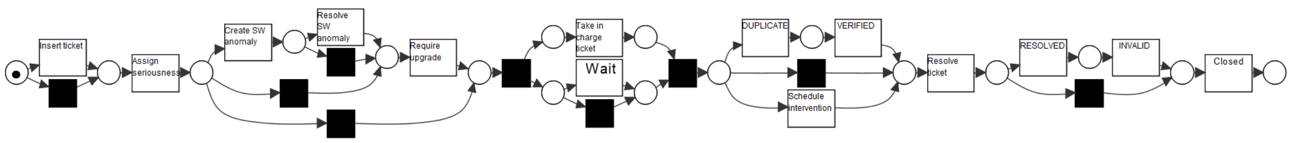
Question 3

(a)

We use the Plugin 'Use Trace Attribute Values' to only select Tickets of type 'Task'. As visualization we then choose 'Explore Event Log'. This gives us the table below:

Traces	2.018
Events	10.283
Event Classes	14
Attributes	6
Variants	201
Events per Trace	5,096
First Event	2010-01-13T12:30:37Z
Last Event	2014-01-03T13:20:58Z

From it we can take that there are 2018 traces, 201 trace variants and 10283 events. We then apply the plugin 'Mine Petri net with Inductive Miner' on this filtered log and make sure we choose 'Inductive Miner - Infrequent (IMf)' as our variant and set it to 20% by assigning the Noise threshold to 0.20. This results in the following Model:



(b)

We use the plugin 'Multi-perspective Process Explorer' on our Petri net from (a). This gives us the following information on fitness and precision:

Avg fitness	92,3%
% Violations	14,2%
% Event Violations	14,2%
% Data Violations	0%
# Correct Events	8.966
# Wrong Events	1.317
# Missing Events	166

Avg activity precision	82,9%
# Moves Observed	33.237
# Moves Possible	40.096
Avg fitness	92,3%
% Violations	14,2%
% Event Violations	14,2%
% Data Violations	0%
# Correct Events	8.966
# Wrong Events	1.317
# Missing Events	166

We obtain the percentage of fitting traces by calculating 100% minus the percentage of violations (100% - 14.2%), resulting in 85,8% fitting traces. Alignment-based fitness (92.3%) and precision (82.9%) can be read from the tables above.
By using the inductive miner again and adjusting the infrequency parameter to 10% we obtain a process model with better fitness, precision and more perfectly fitting traces than before. The stats of the new model can be seen below:

Avg activity precision	83,6%
# Moves Observed	35.324
# Moves Possible	42.242
Avg fitness	92,4%
% Violations	14%
% Event Violations	14%
% Data Violations	0%
# Correct Events	8.985
# Wrong Events	1.298
# Missing Events	169

(c)

Question 4

(a)

We created the following OLAP Table:

CASE ID	TICKET TYPE	PRIORITY	RESOURCE OF STARTING ACTI...	NUMBER OF ACTIVE CASES AT ...	Decision
Case 1	Question	Normal	Res4	18	False/No-Wait
Case 10	Task	Normal	Res4	18	False/No-Wait
Case 100	Task	Normal	Res3	18	False/No-Wait
Case 1000	Incident	Urgent	Res8	18	False/No-Wait
Case 1001	Task	Normal	Res4	18	False/No-Wait
Case 1002	Task	High	Res3	18	False/No-Wait
Case 1003	Task	Normal	Res4	18	False/No-Wait
Case 1004	Incident	Normal	Res4	18	False/No-Wait
Case 1005	Task	Normal	Res3	18	False/No-Wait
Case 1006	Question	High	Res4	18	False/No-Wait
Case 1007	Task	Normal	Res3	18	False/No-Wait
Case 1008	Task	High	Res8	18	False/No-Wait
Case 1009	Task	High	Res8	18	True/Wait
Case 101	Question	Normal	Res3	18	False/No-Wait
Case 1010	Task	Normal	Res4	18	False/No-Wait
Case 1011	Question	Normal	Res8	18	False/No-Wait
Case 1012	Incident	Normal	Res8	18	False/No-Wait
Case 1013	Question	Normal	Res4	18	False/No-Wait
Case 1014	Task	Normal	Res4	18	True/Wait
Case 1015	Incident	Normal	Res8	18	False/No-Wait
Case 1016	Question	High	Res8	18	False/No-Wait
Case 1017	Task	Normal	Res4	18	True/Wait
Case 1018	Task	Normal	Res4	18	False/No-Wait
Case 1019	Task	Normal	Res8	18	True/Wait
Case 102	Question	Normal	Res3	18	False/No-Wait
Case 1020	Task	Normal	Res4	18	True/Wait
Case 1021	Question	Normal	Res3	18	False/No-Wait
Case 1022	Question	Normal	Res8	18	False/No-Wait
Case 1023	Task	Normal	Res3	18	False/No-Wait
Case 1024	Task	High	Res4	18	False/No-Wait
Case 1025	Task	Normal	Res4	18	True/Wait
Case 1026	Task	Normal	Res8	18	False/No-Wait
Case 1027	Task	Normal	Res3	18	True/Wait
Case 1028	Task	High	Res4	18	False/No-Wait
Case 1029	Question	Normal	Res3	18	False/No-Wait
Case 103	Task	Normal	Res4	18	True/Wait

For this, we used the following PQL Queries in the order of columns in the image left to right:

```
"case_table_csv"."CASE ID"
```

```
"case_table_csv"."TICKET TYPE"
```

```
"event_table_csv"."PRIORITY"
```

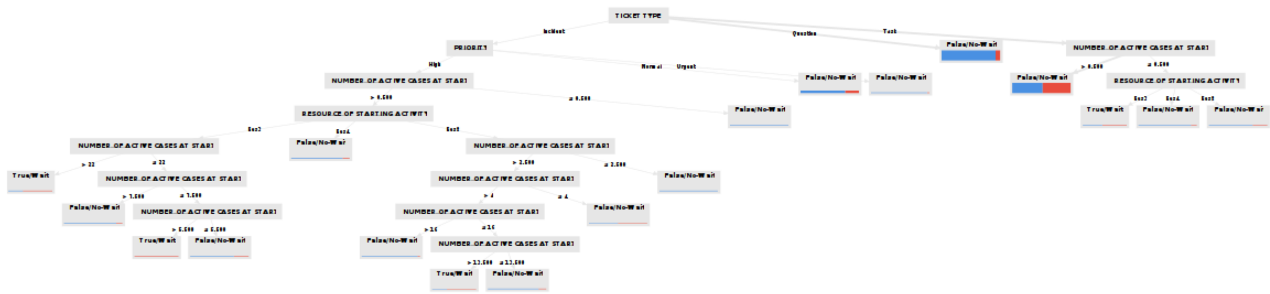
```
PU_FIRST ( "case_table_csv", "event_table_csv"."RESOURCE")
```

```
RUNNING_SUM (
  CASE WHEN MATCH_PROCESS_REGEX("event_table_csv"."ACTIVITY", 'Closed'$) = 1 THEN 0
  ELSE 1
  END
)
```

```
CASE WHEN
MATCH_PROCESS_REGEX ( "event_table_csv"."ACTIVITY", 'Wait' ) = 1 THEN 'True/Wait'
ELSE 'False/No-Wait'
END
```

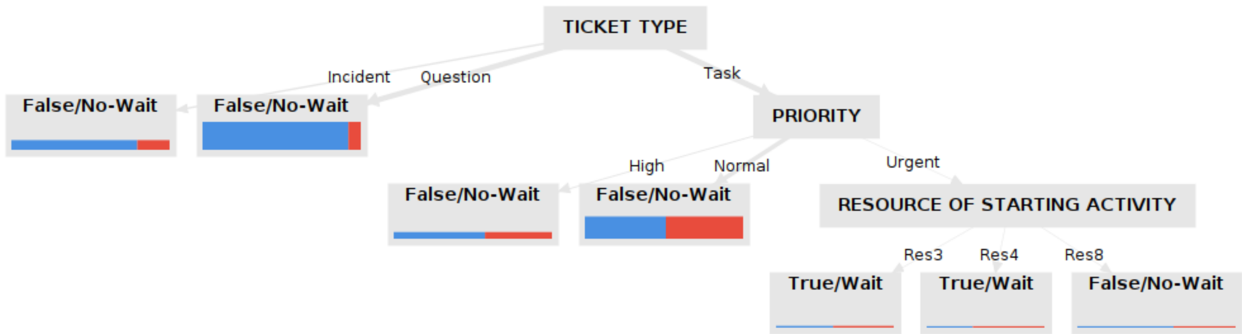
(b)

First, we export the OLAP Table and import it into RapidMiner as described in Instruction 2. After setting the attribute Decision as our label, we had to adjust the minimal gain ratio to 0.006 in order to see more than just one Wait-Leaf. The resulting decision tree is evidently too large to fit into a PDF, thus we have added the description in the Appendix:



We found that some tasks using Resource 3 would wait, even if they were the only task running at creation. From the tree we can also observe that some high priority incidents (using Resource 3 or 8) would have to wait depending on the number of active cases at start, although there does not seem to be any connection to the prior predictor variables.

After removing the attribute 'number of active cases at start' and further lowering the minimal gain ratio to 0.001, we get a much more simplified, but comprehensible Decision Tree:



Here we see that urgent tasks using the starting resource 3 or 4 are quite likely to be set into waiting mode.

Question 5

Question 6