

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:

The screenshot shows the 'Log Summary' interface. On the left, a sidebar has a red arrow pointing to the 'Summary' icon. The main area is divided into two panels. The left panel, labeled 'All events', shows a table of event classes with 14 total classes. The right panel, labeled 'Start events', shows a table of start events with 6 total classes. Both tables are highlighted with red boxes and numbered 3 and 4 respectively. A third red box and number 5 highlight the 'End events' table on the right.

| Class | Occurrences (absolute) |
|-----------------------|------------------------|
| Take in charge ticket | 5060 |
| Resolve ticket | 4983 |
| Assign seriousness | 4938 |
| Closed | 4574 |
| Wait | 1463 |
| Require upgrade | 119 |
| Insert ticket | 118 |
| Create SW anomaly | 67 |
| Resolve SW anomaly | 13 |
| Schedule intervention | 5 |
| VERIFIED | 3 |
| INVALID | 2 |
| RESOLVED | 2 |
| DUPLICATE | 1 |

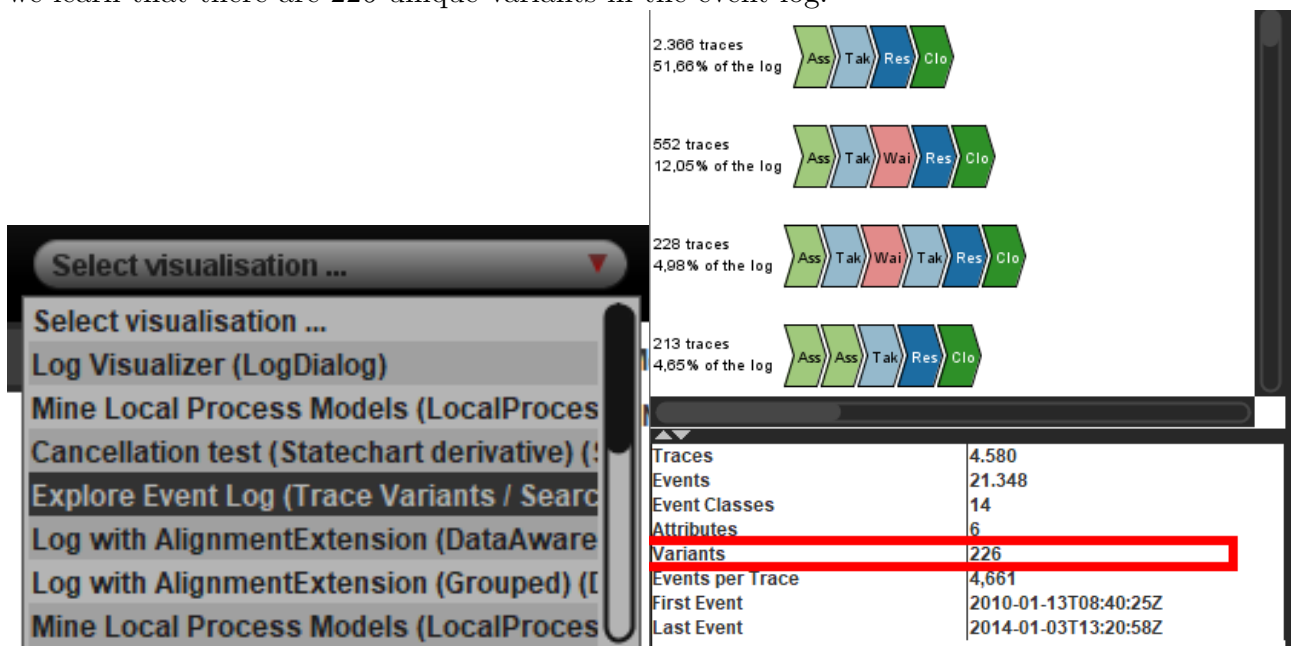
| Class | Occurrences (absolute) |
|-----------------------|------------------------|
| Assign seriousness | 4384 |
| Insert ticket | 118 |
| Take in charge ticket | 74 |
| Resolve ticket | 2 |
| Create SW anomaly | 1 |
| Wait | 1 |

| Class | Occurrences (absolute) |
|-----------------------|------------------------|
| Closed | 4557 |
| Resolve ticket | 10 |
| Wait | 8 |
| Require upgrade | 3 |
| Take in charge ticket | 1 |
| VERIFIED | 1 |

From (3) we get a table of occurrence frequencies for each activity. From (4) we get a table of occurrence frequencies for each start activity. From (5) we get a table of occurrence 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 ($\frac{1}{20}$ ratio of cases to variants, although distributed very unevenly).

From (4) we also observe that a quite high number of cases start with "illegal" activities (so not 'Assign seriousness' or 'Insert ticket'). It strikes out, that people managed in 74 cases to start their tickets with the "illegal" activity 'Take in charge ticket', while in comparison only 118 cases begin with the activity 'Insert ticket'.

(b)

From the introduction we learned that every trace has to start with 'Insert ticket' or 'Assign seriousness' 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'.

Start events
Only instances starting with a green event will be used.

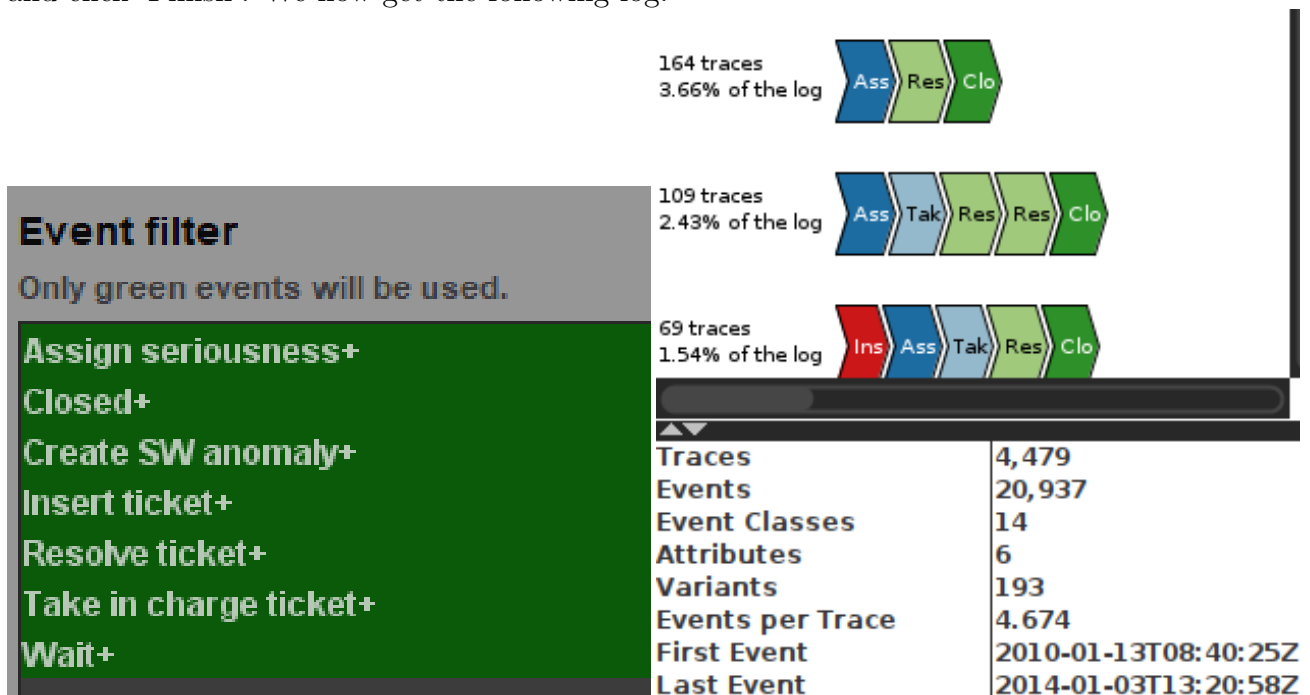
- Assign seriousness+
- Create SW anomaly+
- Insert ticket+**
- Resolve ticket+
- Take in charge ticket+
- Wait+

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

End events
Only instances ending with a green event will be used.

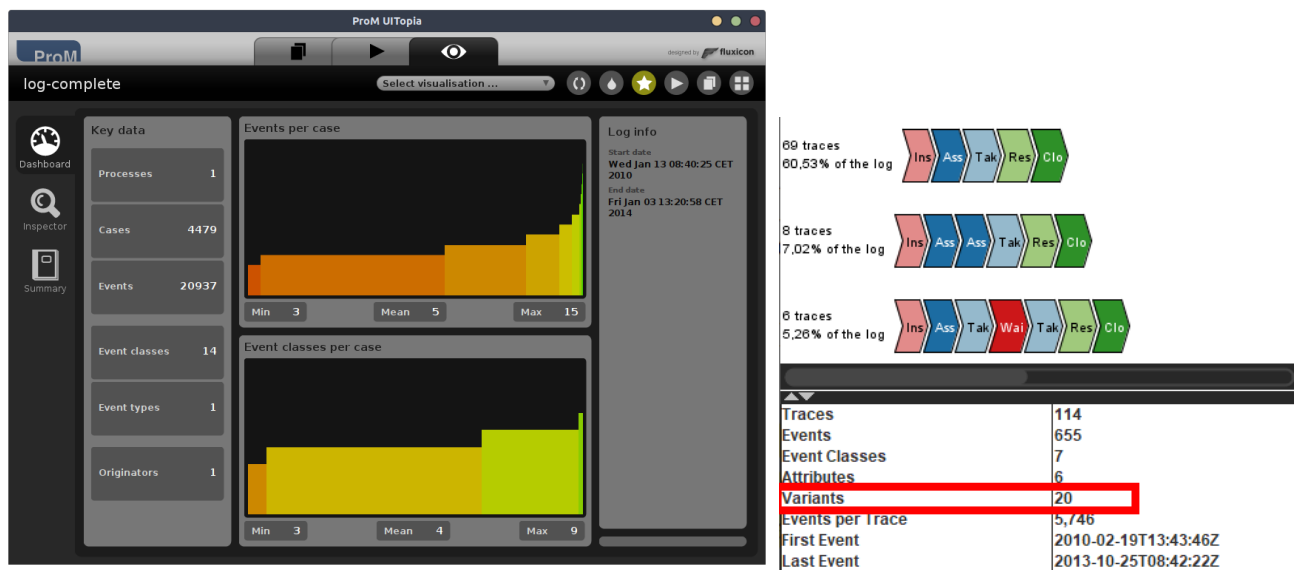
- Closed+**
- Wait+

Since we do not want to filter out any other events we select 100% of events in Event Filter and click 'Finish'. We now get the following log:

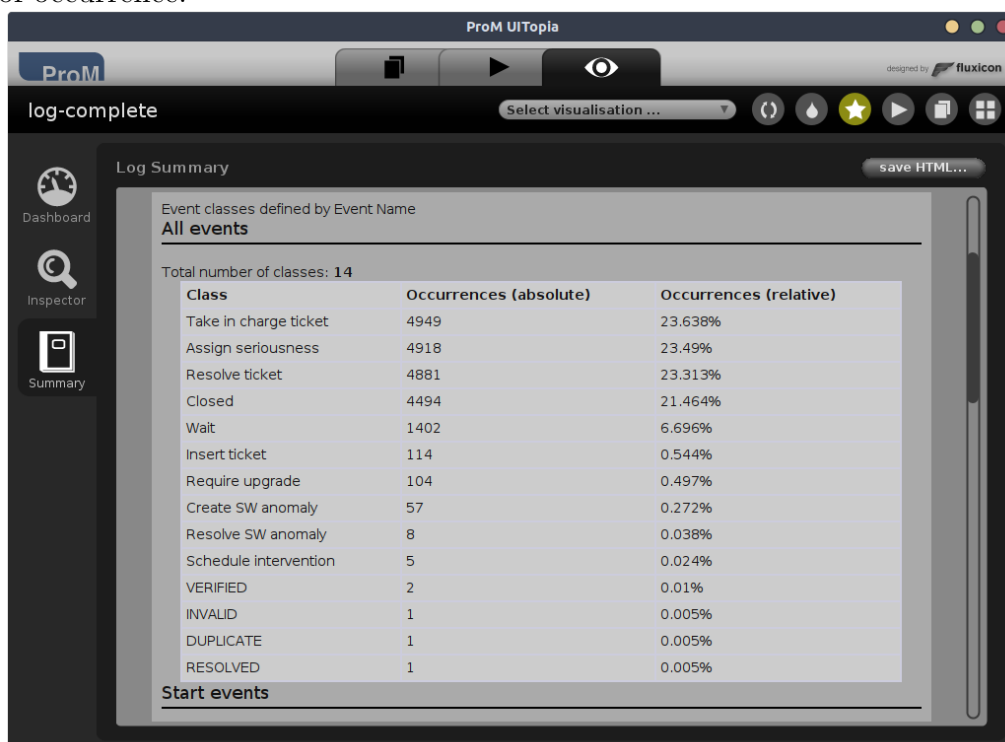


(c)

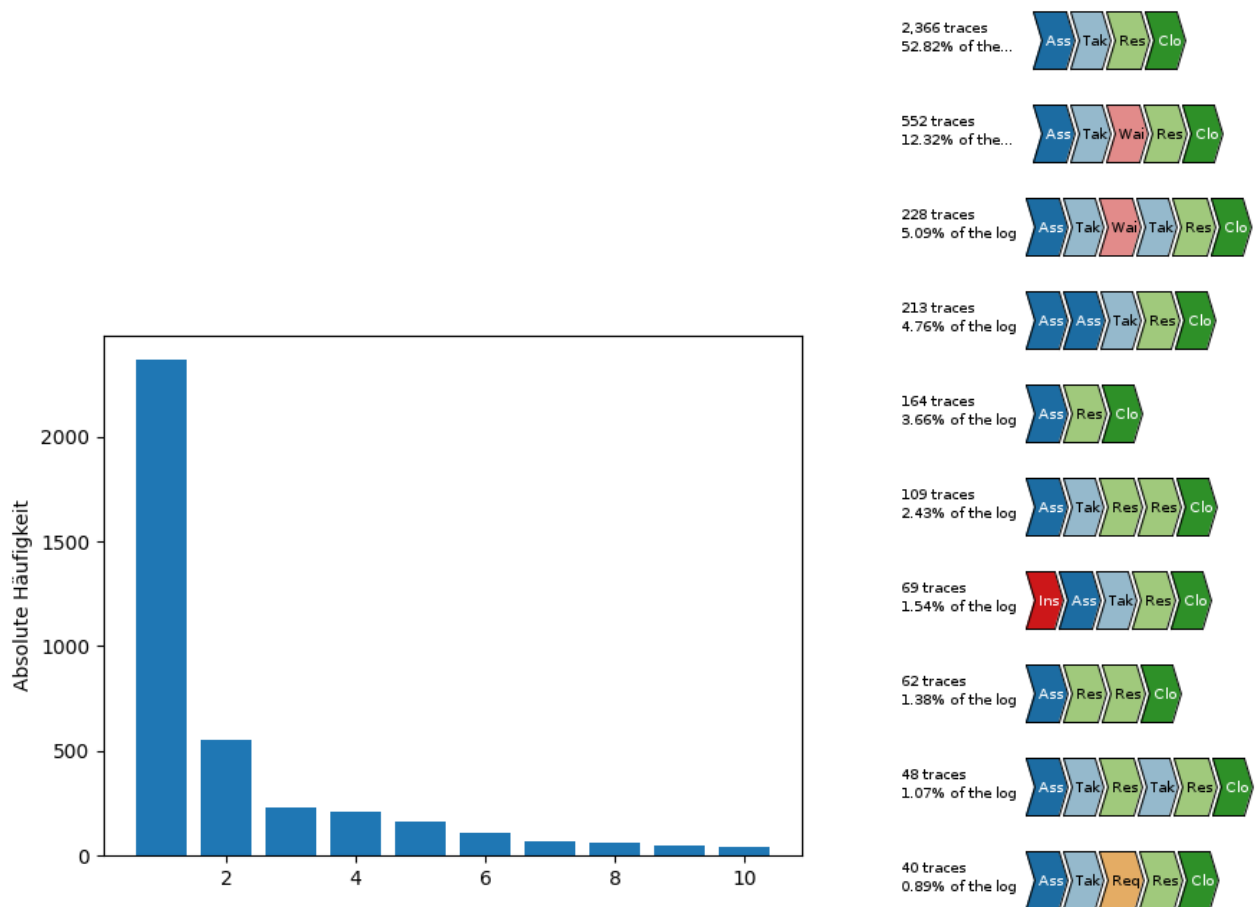
1. As in (a) we inspect the overview of *log-complete* to find 114 cases, 655 events and 7 activities. Also just like in (a) we select the visualization 'Explore Event Log' to find out there are 20 unique trace variants.



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



3. We read the counts of the top 10 traces from the Event Log Explorer in RapidMiner, and plugged those values into a short python script to create the following bar chart.

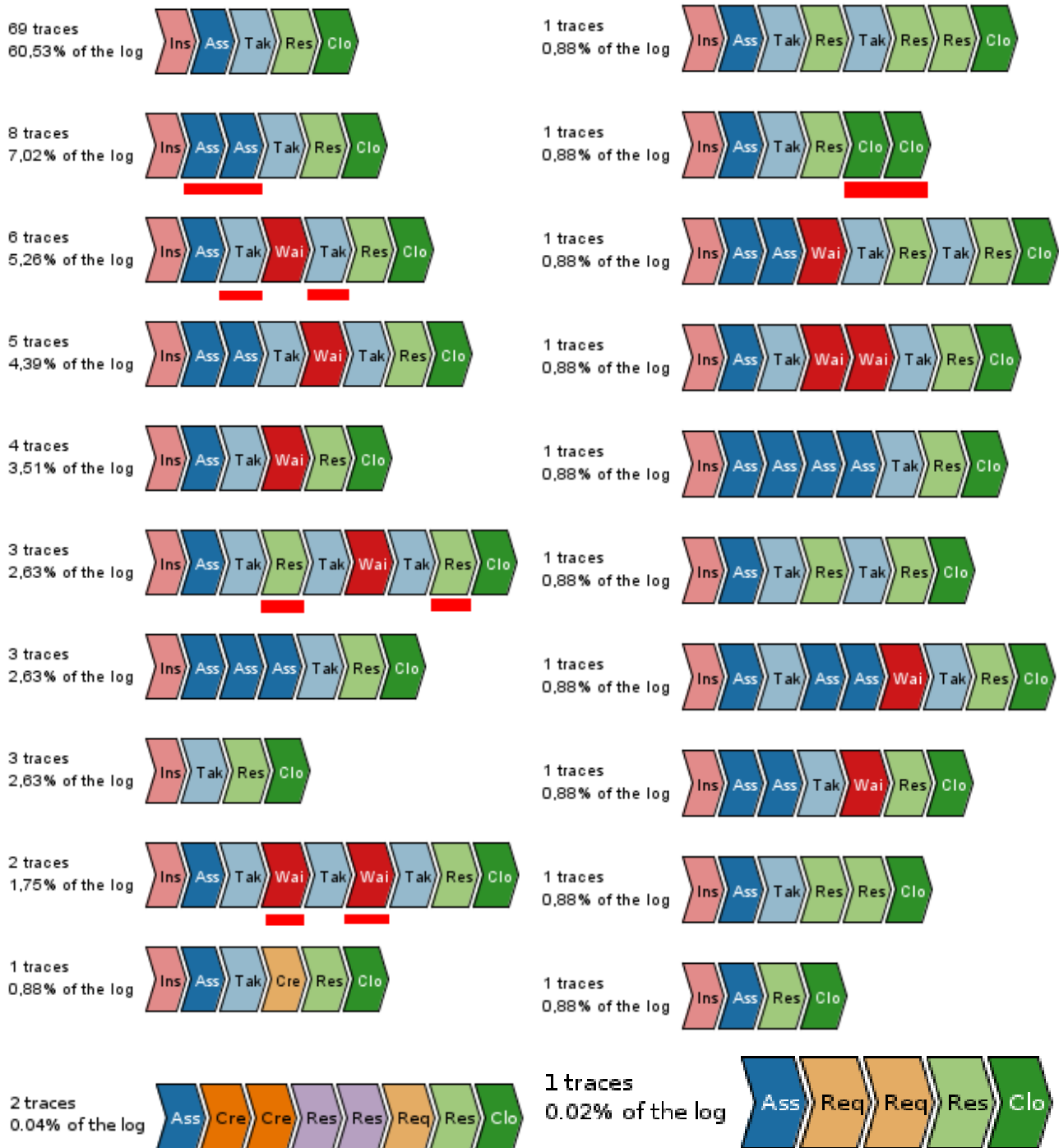


4. We apply the 'Add Throughput Time as Trace Attribute (In place)' plugin to *log-complete* and select 'DAYS' as the resolution to be used for the elapsed time. We then select the visualization 'Dotted Chart' on the result. To retrieve minimum, maximum and average trace durations we select 'T: throughputtime'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

(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")
```

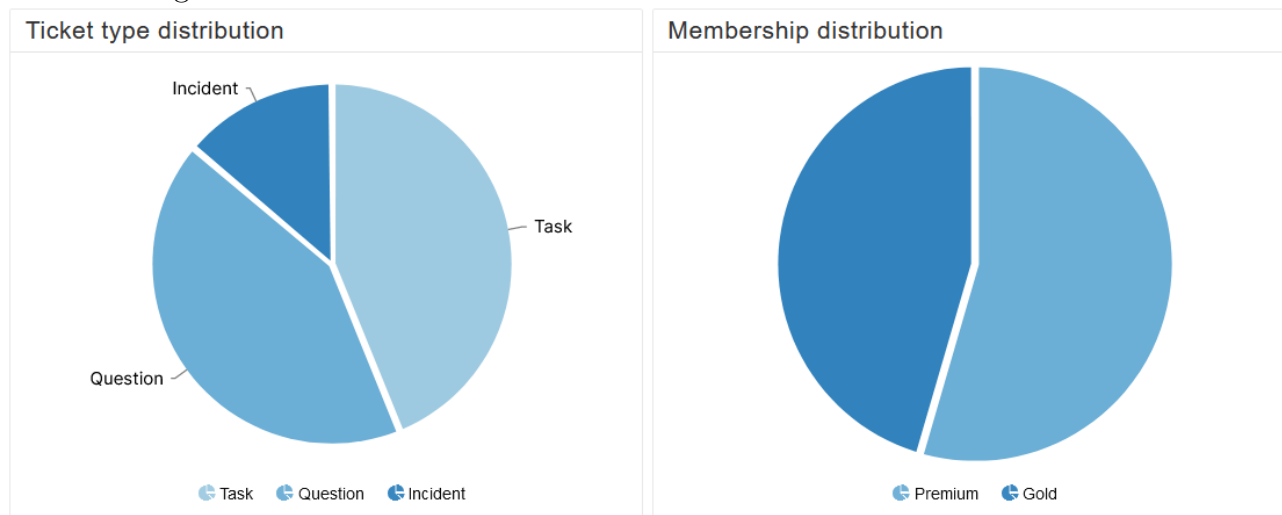
For *Membership* distribution we choose **MEMBERSHIP** as dimension:

```
"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 titled '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 handled by the given resource.

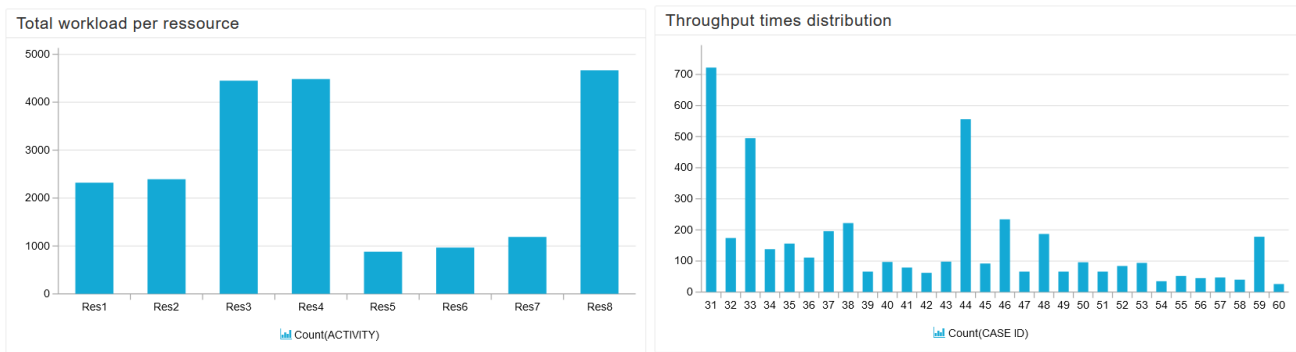
3. We created the column chart named 'Throughput times distribution' by selecting the total throughput time in days as our dimension:

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

and selecting

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

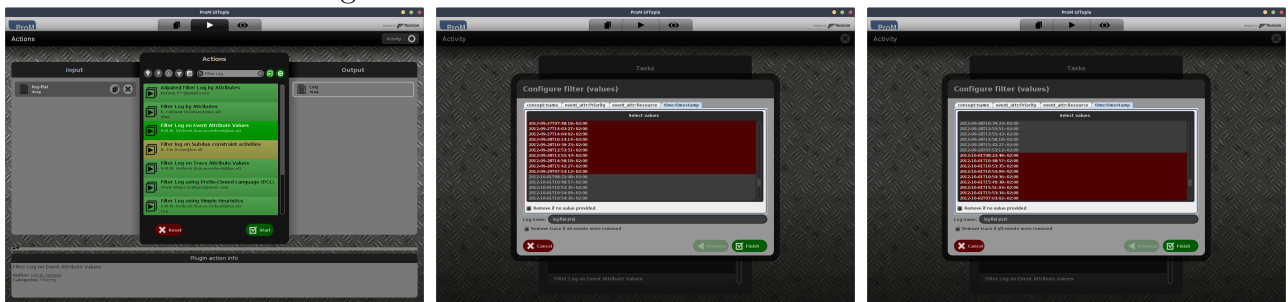
as our KPI. The x-Axis shows the throughput time in days and the y-Axis shows the number of cases with a given throughput time.



Question 2

(a)

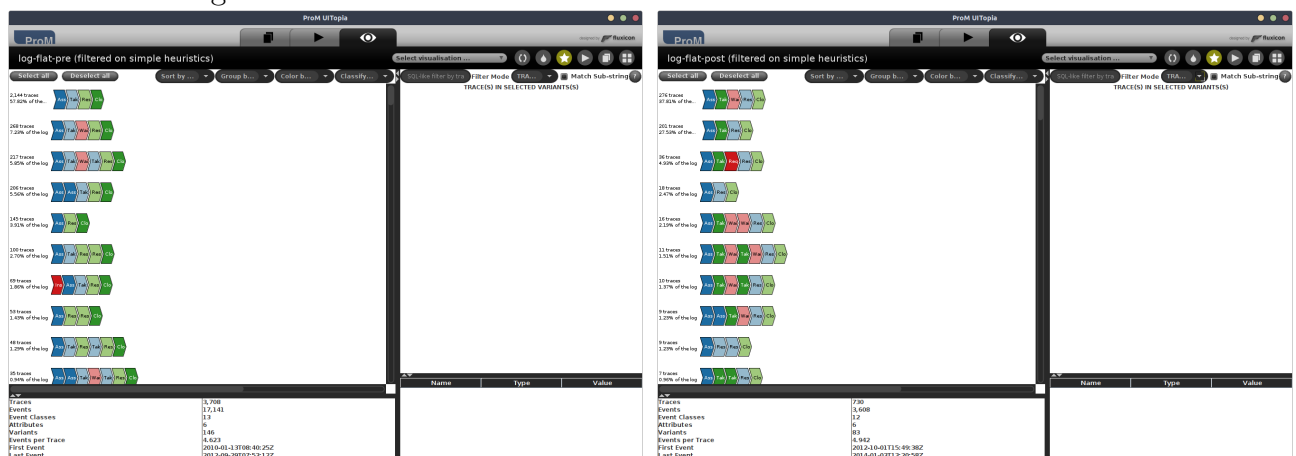
We can split the event log into two event logs using the plugin 'Filter Log on Event Attribute Values' with the following selections:



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) (<i>log-pre-complete</i>) | ii) (<i>log-pre-complete</i>) | iii) (<i>log-post-complete</i>) | iv) (<i>log-post-complete</i>) |
|------------|--------------------------------|---------------------------------|-----------------------------------|----------------------------------|
| 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

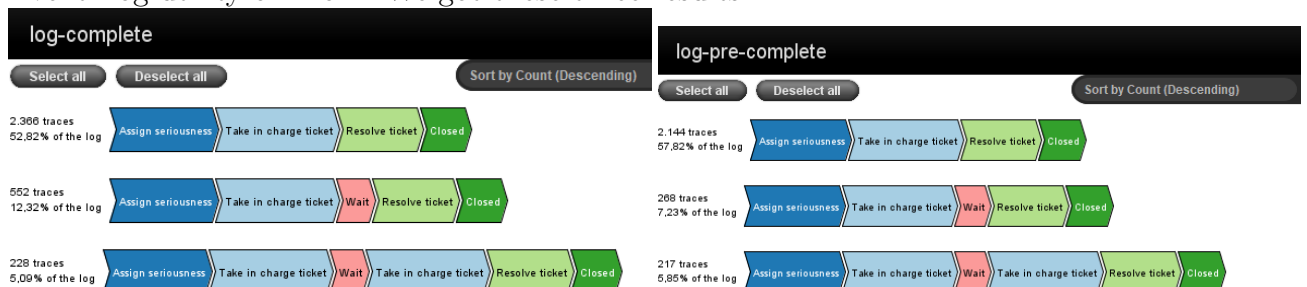
This results in the following Models:

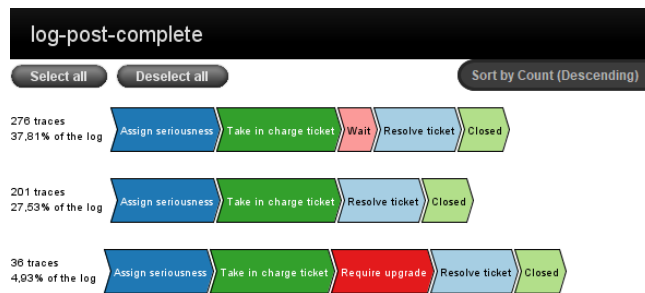


As we can see all these models exceed our fitness threshold of 90% and contain all activities occurring in their respective logs (*log-pre-complete* doesn't contain the activity DUPLICATE and *log-post-complete* doesn't contain the activities INVALID and RESOLVED).

(c)

1. We visualized the three most frequent trace variant and their frequencies with the Explore Event Log utility of ProM. We got these three results:





The first similarity observed between the three most frequent trace variants of all three event logs is, that they all start with the Assign seriousness and Take in charge ticket activities and finish with Resolve ticket and Closed. The second observation is the equality of the three most frequent trace variants of the *log-complete* and *log-pre-complete* event logs. This makes sense since most of the events of *log-complete* are in *log-pre-complete* and not in *log-post-complete*. This also explains the difference in absolute frequency of the most common trace variant between *log-complete* and *log-pre-complete* and the most frequent one of *log-post-complete* which differ by one order of magnitude. Another important difference is that in *log-complete* and *log-pre-complete* the most common trace variant has a relative frequency of over 50% whilst in *log-post-complete* the most common trace variants are more equally divided between the two most frequent traces with 38% and 28%. Finally the activity Require upgrade is only present in the three most common trace variants in the *log-post-complete* event log.

2. All trace variants are replayable on both models:

| | σ_{compl} | σ_{pre} | σ_{post} |
|------------|--|--|--|
| M_{pre} | $\langle \text{Assign seriousness, Take in charge ticket, Resolve ticket, Closed} \rangle$ | $\langle \text{Assign seriousness, Take in charge ticket, Resolve ticket, Closed} \rangle$ | $\langle \text{Assign seriousness, Take in charge ticket, Wait, Resolve ticket, Closed} \rangle$ |
| M_{post} | $\langle \text{Assign seriousness, Take in charge ticket, Resolve ticket, Closed} \rangle$ | $\langle \text{Assign seriousness, Take in charge ticket, Resolve ticket, Closed} \rangle$ | $\langle \text{Assign seriousness, Take in charge ticket, Wait, Resolve ticket, Closed} \rangle$ |

(d)

Question 3

(a)

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

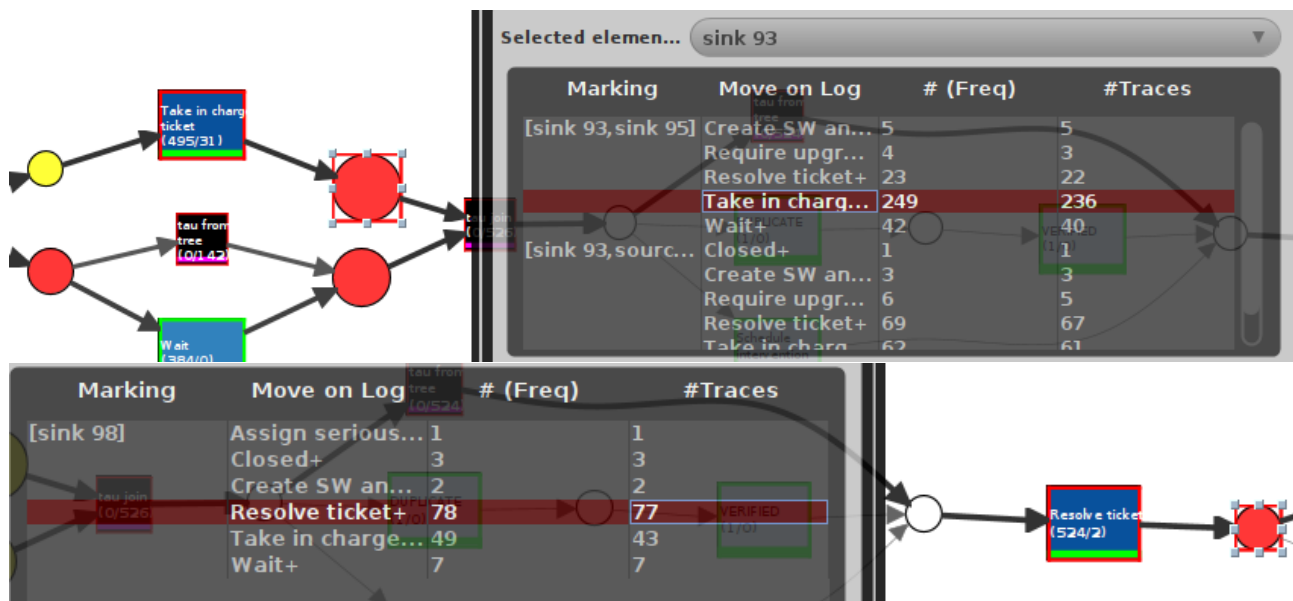
| | |
|------------------|----------------------|
| Traces | 2.018 |
| Events | 10.283 |
| Event Classes | 14 |
| Attributes | 6 |
| Variants | 201 |
| Events per Trace | 5.096 |
| First Event | 2010-01-13T12:30:37Z |
| Last Event | 2014-01-03T13:20:58Z |

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

see that not every case executes Assign seriousness. Perhaps this transition should be skipable by a tau-transition. However, the assignment introduction states that either 'Insert ticket' or 'Assign seriousness' should be executed first. Therefore, the model could also have an option between both in the beginning, instead of putting one behind the other.

That 'Take in charge ticket' was moved on the model 91 times should also be noted. That the model set this action in parallel with 'Wait' seems problematic. It should be possible to wait any time in the ticketing process in order to execute some activity later on.

Move on Log



We can analyze Moves on Logs by clicking on the big places in the visualization. As one can see, the transitions 'Take in charge ticket' and 'Resolve ticket' experience the most moves on log in the model. We used the Movement Container Filter to only allow traces with moves on logs in these transitions.

Question 4

(a)

We created the following OLAP Table:

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

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

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

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

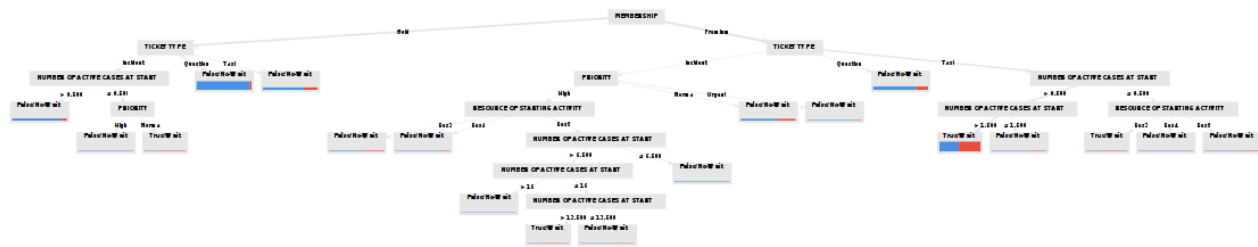
```
RUNNING_SUM (
  CASE WHEN MATCH_PROCESS_REGEX("event_table_csv"."ACTIVITY", 'Closed'$) = 1 THEN 0
  ELSE 1
  END
)
```

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

(b)

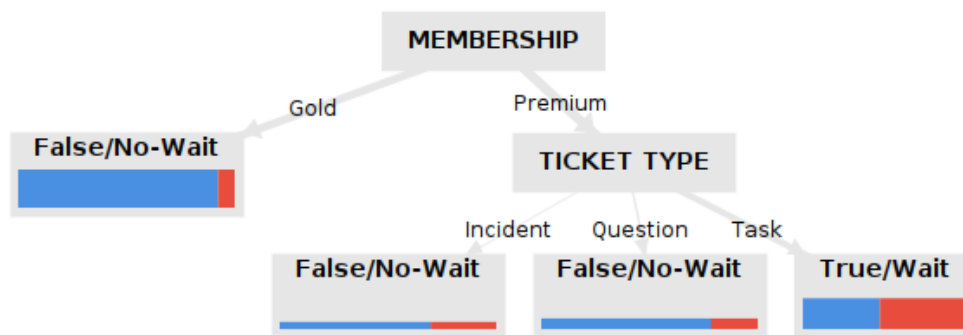
First, we export the OLAP Table and import it into RapidMiner as described in Instruction 2. The resulting decision tree is evidently too large to fit into a PDF, thus we have added the description in the Appendix:

July 3, 2022



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', we get a much more simplified, but comprehensible Decision Tree:



Here we see that about half the Premium Tasks will have to wait, while for all other tickets waiting is less likely.

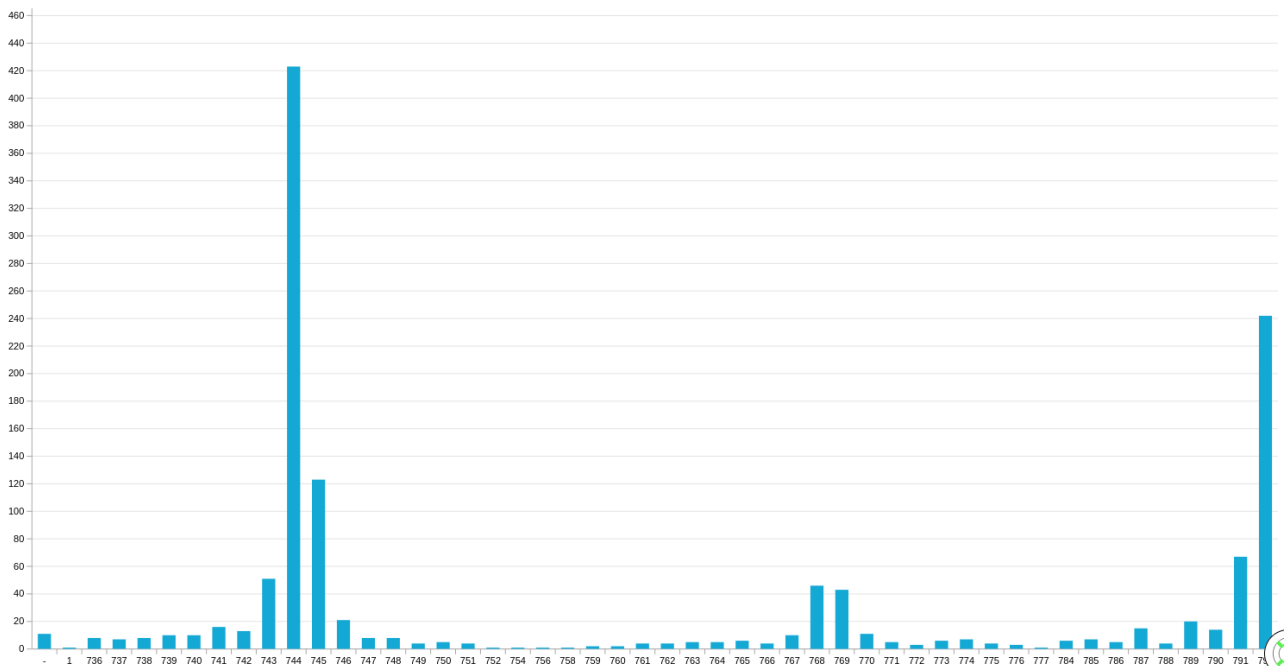
Question 5

(a)

Similarly to 1d), we created a new column chart. Here, we used the following Dimension:

```
CASE WHEN MATCH_PROCESS_REGEX("event_table_csv"."ACTIVITY", 'Closed'$) = 1
THEN CALC_THROUGHPUT(
CASE_START TO CASE_END,
REMAP_TIMESTAMPS("event_table_csv"."TIMESTAMP", HOURS)
)
ELSE CALC_THROUGHPUT(
CASE_START TO LAST_OCCURRENCE['Resolve ticket'],
REMAP_TIMESTAMPS("event_table_csv"."TIMESTAMP", HOURS)
)
END
```

Using the Case Count as a Dimension, we got the following chart. Note that this is just a tiny section of the entire chart. There are cases with throughput times up to 1440h:



(b)

We applied the `Quantile` function on the real throughput times (see above) for 0.3 and 0.7, respectively (here for 0.3):

```

QUANTILE(CASE WHEN MATCH_PROCESS_REGEX("event_table_csv"."ACTIVITY", 'Closed'$) = 1
THEN CALC_THROUGHPUT(
CASE_START TO CASE_END,
REMAP_TIMESTAMPS("event_table_csv"."TIMESTAMP", HOURS)
)
ELSE CALC_THROUGHPUT(
CASE_START TO LAST_OCCURRENCE['Resolve ticket'],
REMAP_TIMESTAMPS("event_table_csv"."TIMESTAMP", HOURS)
)
END, 0.3)

```

For the 0.3-quantile we got 795h, for the 0.7-quantile 1079h.

(c)

Approach equivalent to Question 4a). We used the following PQL queries:

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

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

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

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

```
CASE
```

```

WHEN (CASE WHEN MATCH_PROCESS_REGEX("event_table_csv"."ACTIVITY", 'Closed'$) = 1
THEN CALC_THROUGHPUT(
CASE_START TO CASE_END,

```



```

REMAP_TIMESTAMPS("event_table_csv"."TIMESTAMP",HOURS)
)
ELSE CALC_THROUGHPUT(
CASE_START TO LAST_OCCURRENCE['Resolve ticket'],
REMAP_TIMESTAMPS("event_table_csv"."TIMESTAMP",HOURS)
)
END) < 795 THEN 'Short'

WHEN (CASE WHEN MATCH_PROCESS_REGEX("event_table_csv"."ACTIVITY", 'Closed'$) = 1
THEN CALC_THROUGHPUT(
CASE_START TO CASE_END,
REMAP_TIMESTAMPS("event_table_csv"."TIMESTAMP",HOURS)
)
ELSE CALC_THROUGHPUT(
CASE_START TO LAST_OCCURRENCE['Resolve ticket'],
REMAP_TIMESTAMPS("event_table_csv"."TIMESTAMP",HOURS)
)
END) < 1079 THEN 'Medium'

ELSE 'Long'

END

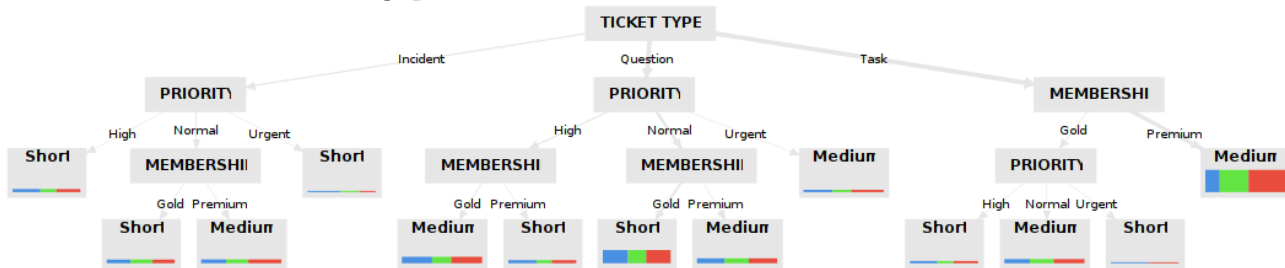
```

| CASE ID | TICKET TYPE | MEMBERSHIP | PRIORITY | Performance class |
|-----------|-------------|------------|----------|-------------------|
| Case 1 | Question | Gold | Normal | Short |
| Case 10 | Task | Premium | Normal | Long |
| Case 100 | Task | Premium | Normal | Long |
| Case 1000 | Incident | Gold | Urgent | Medium |
| Case 1001 | Task | Premium | Normal | Medium |
| Case 1002 | Task | Gold | High | Short |
| Case 1003 | Task | Premium | Normal | Medium |
| Case 1004 | Incident | Premium | Normal | Short |
| Case 1005 | Task | Premium | Normal | Long |
| Case 1006 | Question | Gold | High | Long |
| Case 1007 | Task | Premium | Normal | Long |
| Case 1008 | Task | Gold | High | Medium |
| Case 1009 | Task | Premium | High | Medium |
| Case 101 | Question | Premium | Normal | Long |
| Case 1010 | Task | Premium | Normal | Medium |
| Case 1011 | Question | Gold | Normal | Long |
| Case 1012 | Incident | Premium | Normal | Short |
| Case 1013 | Question | Premium | Normal | Medium |
| Case 1014 | Task | Premium | Normal | Short |
| Case 1015 | Incident | Gold | Normal | Short |
| Case 1016 | Question | Gold | High | Long |
| Case 1017 | Task | Premium | Normal | Medium |
| Case 1018 | Task | Premium | Normal | Long |
| Case 1019 | Task | Premium | Normal | Short |
| Case 102 | Question | Gold | Normal | Long |
| Case 1020 | Task | Premium | Normal | Long |
| Case 1021 | Question | Premium | Normal | Medium |
| Case 1022 | Question | Premium | Normal | Short |
| Case 1023 | Task | Premium | Normal | Short |
| Case 1024 | Task | Gold | High | Medium |
| Case 1025 | Task | Premium | Normal | Long |
| Case 1026 | Task | Premium | Normal | Medium |
| Case 1027 | Task | Premium | Normal | Medium |
| Case 1028 | Task | Gold | High | Short |
| Case 1029 | Question | Gold | Normal | Short |
| Case 103 | Task | Premium | Normal | Medium |

(d)

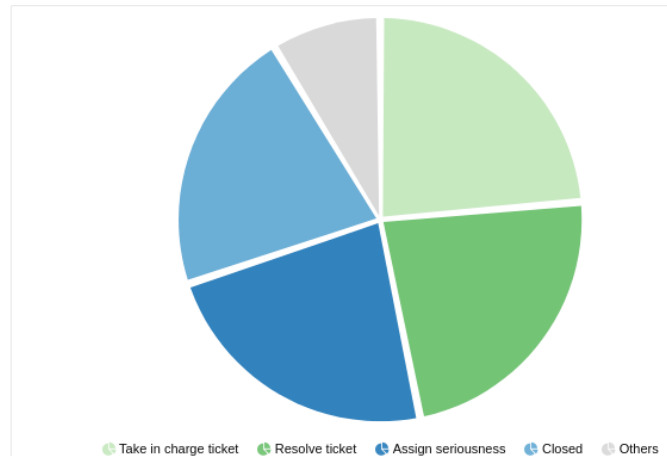
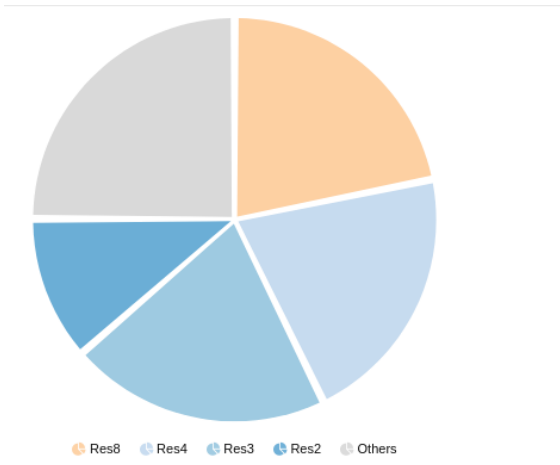
First, we export the OLAP Table and import it into RapidMiner as described in Instruction 2. Even after setting the parameters of the Decision Tree algorithm to extreme values (minimal gain at 1.0E-6), we could not find any clear variable that predicts the outcome of the performance class in any way. Thus, we conclude that the performance class (and thus the real

throughput time) has no strong correlation with neither Priority nor Ticket Type. The only thing noticeable was, that close to half the Tasks stemming from Premium Memberships were Medium rated in real throughput time.



Question 6

(a)



For the left Pie Chart we used "event_table_csv"."RESOURCE" as the Dimension and COUNT("event_table_csv"."RESOURCE") as the KPI.

For the right Pie Chart we used "event_table_csv"."ACTIVITY" as the Dimension and COUNT("event_table_csv"."ACTIVITY") as the KPI.

(b)

| RESOURCE | Assign seri... | Wait | Insert ticket | Resolve tic... | Create SW... | Take in ch... | Resolve S... | Closed | Require U... | RESOLVED | VERIFIED | Schedule i... | INVALID | DUPLICATE |
|----------|----------------|---------------|---------------|----------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|-------------|
| Res1 | 0 | 0 | 0 | 0.00043668... | 0 | 0 | 0 | 0.00196506... | 0.01310043... | 0.00021834... | 0 | 0.00043668... | 0.00021834... | 0 |
| Res2 | 0 | 0 | 0 | 0.00109170... | 0 | 0 | 0 | 0.00174672... | 0.01222707... | 0.00021834... | 0.00043668... | 0.00065502... | 0.00021834... | 0.000218... |
| Res3 | 0.34388646... | 0.04825327... | 0.01026200... | 0.35262008... | 0.00174672... | 0.21310043... | 0.00109170... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Res4 | 0.35807860... | 0.05087336... | 0.00829694... | 0.34126637... | 0.00065502... | 0.21834061... | 0.00043668... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Res5 | 0.00371179... | 0.05262008... | 0 | 0.00349344... | 0.00240174... | 0.12969432... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Res6 | 0.00371179... | 0.05611353... | 0 | 0.00305676... | 0.00436681... | 0.14323144... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Res7 | 0.00240174... | 0.05873362... | 0 | 0.00545851... | 0.00327510... | 0.18930131... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Res8 | 0.36637554... | 0.05109170... | 0.00720524... | 0.37838427... | 0.00218340... | 0.21091703... | 0.00131004... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SOURCE("event_table_csv"."RESOURCE")

```
1 SUM(CASE WHEN SOURCE ( "event_table_csv"."ACTIVITY" ) = '[ACTIVITY]' THEN
  1 ELSE 0 END) / 4580
```

Replace [ACTIVITY] with each of the 14 Activities for the corresponding Dimensions. 4580 is the number of cases obtained by COUNT_TABLE("case_table_csv").

(c)

(d)

| RESOURCE_FROM | RESOURCE_TO | Handover |
|---------------|-------------|-----------------------|
| Res1 | Res1 | 0.0032751091703056767 |
| Res1 | Res2 | 0.002183406113537118 |
| Res1 | Res3 | 0.00240174672489083 |
| Res1 | Res4 | 0.0034934497816593887 |
| Res1 | Res5 | 0.0002183406113537118 |
| Res1 | Res6 | 0.0006550218340611354 |
| Res1 | Res7 | 0.0006550218340611354 |
| Res1 | Res8 | 0.0034934497816593887 |
| Res2 | Res1 | 0.0017467248908296944 |
| Res2 | Res2 | 0.004366812227074236 |
| Res2 | Res3 | 0.0028384279475982535 |
| Res2 | Res4 | 0.0028384279475982535 |
| Res2 | Res5 | 0.0002183406113537118 |
| Res2 | Res7 | 0.0008733624454148472 |
| Res2 | Res8 | 0.003930131004366812 |
| Res3 | Res1 | 0.15807860262008733 |
| Res3 | Res2 | 0.16179039301310044 |
| Res3 | Res3 | 0.1665938864628821 |
| Res3 | Res4 | 0.1646288209606987 |
| Res3 | Res5 | 0.06506550218340611 |
| Res3 | Res6 | 0.07052401746724891 |
| Res3 | Res7 | 0.026200873362445413 |
| Res3 | Res8 | 0.15807860262008733 |
| Res4 | Res1 | 0.15960698689956332 |
| Res4 | Res2 | 0.15655021834061136 |
| Res4 | Res3 | 0.15676855895196506 |
| Res4 | Res4 | 0.1707423580786026 |
| Res4 | Res5 | 0.07074235807860262 |
| Res4 | Res6 | 0.07510917030567686 |
| Res4 | Res7 | 0.023799126637554586 |
| Res4 | Res8 | 0.1646288209606987 |
| Res5 | Res1 | 0.003056768558951965 |
| Res5 | Res2 | 0.00414847161572052 |
| Res5 | Res3 | 0.0665938864628821 |
| Res5 | Res4 | 0.05982532751091703 |
| Res5 | Res5 | 0.015065502183406113 |

SOURCE("event_table_csv"."RESOURCE")

TARGET("event_table_csv"."RESOURCE")

```
1 COUNT(SOURCE("event_table_csv"."ACTIVITY")) / 4580
```

4580 is the number of cases obtained by COUNT_TABLE("case_table_csv").

(e)

(f)

(g)

Appendix

Question 4b) Decision Tree description:

```

1 MEMBERSHIP = Gold
2 |   TICKET TYPE = Incident
3 | |   NUMBER OF ACTIVE CASES AT START > 0.500: False/No–Wait {False/No–
   | |   Wait=266, True/Wait=27}
4 | |   NUMBER OF ACTIVE CASES AT START <= 0.500
5 | | |   PRIORITY = High: False/No–Wait {False/No–Wait=2, True/Wait=0}
6 | | |   PRIORITY = Normal: True/Wait {False/No–Wait=1, True/Wait=2}
7 |   TICKET TYPE = Question: False/No–Wait {False/No–Wait=1391, True/Wait=41}
8 |   TICKET TYPE = Task: False/No–Wait {False/No–Wait=262, True/Wait=89}
9 MEMBERSHIP = Premium
10 |   TICKET TYPE = Incident

```

```

11 | |   PRIORITY = High
12 | |   RESOURCE OF STARTING ACTIVITY = Res3: False/No—Wait {False/No—
    |   Wait=19, True/Wait=11}
13 | |   RESOURCE OF STARTING ACTIVITY = Res4: False/No—Wait {False/No—
    |   Wait=22, True/Wait=4}
14 | |   RESOURCE OF STARTING ACTIVITY = Res8
15 | |   |   NUMBER OF ACTIVE CASES AT START > 6.500
16 | |   |   |   NUMBER OF ACTIVE CASES AT START > 16: False/No—Wait {
    |   False/No—Wait=11, True/Wait=1}
17 | |   |   |   NUMBER OF ACTIVE CASES AT START <= 16
18 | |   |   |   |   NUMBER OF ACTIVE CASES AT START > 13.500: True/Wait {
    |   False/No—Wait=1, True/Wait=2}
19 | |   |   |   |   NUMBER OF ACTIVE CASES AT START <= 13.500: False/No—
    |   Wait {False/No—Wait=8, True/Wait=3}
20 | |   |   |   NUMBER OF ACTIVE CASES AT START <= 6.500: False/No—Wait {
    |   False/No—Wait=3, True/Wait=0}
21 | |   PRIORITY = Normal: False/No—Wait {False/No—Wait=147, True/Wait=75}
22 | |   PRIORITY = Urgent: False/No—Wait {False/No—Wait=13, True/Wait=1}
23 | |   TICKET TYPE = Question: False/No—Wait {False/No—Wait=401, True/Wait=110}
24 | |   TICKET TYPE = Task
25 | |   NUMBER OF ACTIVE CASES AT START > 0.500
26 | |   |   NUMBER OF ACTIVE CASES AT START > 1.500: True/Wait {False/No—
    |   Wait=777, True/Wait=861}
27 | |   |   NUMBER OF ACTIVE CASES AT START <= 1.500: False/No—Wait {False/
    |   No—Wait=4, True/Wait=2}
28 | |   NUMBER OF ACTIVE CASES AT START <= 0.500
29 | |   |   RESOURCE OF STARTING ACTIVITY = Res3: True/Wait {False/No—Wait
    |   =2, True/Wait=4}
30 | |   |   RESOURCE OF STARTING ACTIVITY = Res4: False/No—Wait {False/No—
    |   Wait=9, True/Wait=1}
31 | |   |   RESOURCE OF STARTING ACTIVITY = Res8: False/No—Wait {False/No—
    |   Wait=5, True/Wait=2}

```