405801 Anna Garoufali
392326 Na Young Ahn
414382 Jahnavi Jaiswal

Assignment Part 2

Business Process Intelligence

Introduction

For assignment 2, we use a real life data set, *receipt phase event log* that contains records related to execution of the building permit application process in a municipality in the Netherlands. We want to retrieve meaningful intelligence of the real process via process mining. We first prepare a good sampled representation of the log by randomizing, handling incomplete traces. Then we examine the prepared event log by splitting into subsections of interest and gathering information on general characteristics. After that we mine for process models that correctly reflect the real process. We do so by trying out various mining techniques, optimizing for parameters and checking for conformance. After checking the quality, alignment and deviation we mine for decision points to have an explainable model for choosing a certain path. We then do a performance analysis, looking for bottlenecks and performance metrics followed by organizational mining to have deeper insight to improve organizational resource efficiency.

Question 0 - Randomizing the log

Using ProM6.10's plugin "Extract sample of traces (Random)" by F.Mannhardt, we created a randomized subset of the provided event log with 1700 cases using the student matriculation number 405801 as seed. (For screenshots, see "../BPI_assignment_part_2_405801_392326_414382/Q0")

Question 1 - Knowing the Event log

Part 1 - Incomplete Traces

Incomplete traces can hinder comprehensibility of the results, therefore must be removed before further analysis.
We assumed incomplete traces to be (and/or):

- traces with missing values,
- traces with missing start point,
- traces with missing end point.

The causes of incomplete traces could be:

- the timeline given by the events from our sampled data is not representative of the timeline of the real data. Thus, our sampled data may contain unfinished cases that is actually finished in the real data.
- some cases may never finish. For instance, the customer may never return with requested data.

Using ProM6.10, we removed the incomplete traces by:

1. removing empty values with "Remove all attributes with value "" (empty) (In Place)" by F. Mannhardt.
2. Then we used filtered with "Filter Log using Simple Heuristics" by H.M.W. Verbeek. For the start event, there is only one event option in our randomized_event_log.xes, thus we chose "Confirmation of receipt + complete". To choose the end events, we filtered top 80 percentage and retrieved three end events, namely: "T05 Print and send confirmation of receipt + complete", "T10 Determine necessity to stop indication + complete" and "T15 Print document X request unlicensed + complete".

3. After heuristically setting start and end events by selecting top 80 percentage, we filter for 100 percent of the events i.e take all event into consideration when setting traces.

Table 1: Number of cases and events after removing incomplete traces via simple heuristic filtering

	Before: randomized_event_log.xes	After "Filter log using Simple Heuristics"
cases	1700	1442
events	11477	10049

As a result of the aforementioned filtering for incomplete traces, we have deleted 258 incomplete traces out of 1700 cases in randomized_event_log.xes. We now store the remaining 1442 complete traces as randomized_event_log.xes. (For screenshots, see "../Q1/P1")

Part 2 - Splitting the event log

The attribute *channel* has 5 different values in the original log: *Internet*, *Desk*, *Post*, *e-mail* and *Intern*. Using Disco, all event logs were saved with the corresponding channel name. cf) file name: *randomized_filtered_ event_ log_ {internet, desk, post, e-mail, intern}.xes*

Part 3 - General characteristics

We used Disco for this section.

1. How many cases and events are in the event log?

	Internet	Desk	Post	e-mail	Intern	Total
cases	1276	100	47	18	1	1442
events	8943	698	286	116	6	10049

2. How many unique trace variants are in the event log?

	Internet	Desk	Post	e-mail	Intern
unique trace variants	80	19	9	6	1

3. What is the number of unique activities and unique resources?

	Internet	Desk	Post	e-mail	Intern
unique activities	25	17	8	7	6
unique resources	43	27	22	13	3

4. What are the set of start and the set of end activities in the cases?

<table>
<tr><th></th><th>Internet</th><th>Desk</th><th>Post</th><th>e-mail</th><th>Intern</th></tr>
<tr><td>set of start activities</td><td colspan="5">{Confirmation of Script}</td></tr>
<tr><td>set of end activities</td><td colspan="2">{T05 Print and send confirmation of receipt,
T10 Determine necessity to stop indication,
T15 Print document X request unlicensed}</td><td colspan="2">{T05 Print and send confirmation of receipt,
T10 Determine necessity to stop indication}</td><td>{T05 Print and send confirmation of receipt}</td></tr>
</table>

Part 4 - More specific characteristics

Following answers are based on *randomized_filtered_event_log_desk.xes*
(For screenshots, see "../Q1/P4")

1. What are the minimum and the maximum number of unique activities in a trace in the process?
   - min. number: 6
   - max. number: 18
2. How is the distribution of the department attribute?
   - **general**, *98.85%*
   - **experts**, *1.15%*
3. What is the most frequently executed activity?
   - T06 Determine necessity of stop advice, *17.48%*
4. Which activity is executed in one variant most often? How often?
   - As seen in the table below, for some variants there is no single most often executed activity i.e. all activities in respective variants were executed equally often. Hence, in these cases, the frequency mentioned indicates the frequency of all the activities executed by the respective variant.

Table 2: Most executed activity per variant

Variant	Most often executed activity	Frequency
Variant 1	All activities in the respective variant are executed equally	38
Variant 2	All activities in the respective variant are executed equally	13
Variant 3	All activities in the respective variant are executed equally	12
Variant 4	All activities in the respective variant are executed equally	8
Variant 5	All activities in the respective variant are executed equally	6
Variant 6	All activities in the respective variant are executed equally	6
Variant 7	All activities in the respective variant are executed equally	4
Variant 8	T02 Check confirmation of receipt	4
Variant 9	T06 Determine necessity of stop advice	2
Variant 10	T06 Determine necessity of stop advice	2
Variant 11	T06 Determine necessity of stop advice	3
Variant 12	T06 Determine necessity of stop advice	7
Variant 13	T06 Determine necessity of stop advice	2
Variant 14	T06 Determine necessity of stop advice	7
Variant 15	T06 Determine necessity of stop advice	4
Variant 16	All activities in the respective variant are executed equally	1
Variant 17	T06 Determine necessity of stop advice	3
Variant 18	All activities in the respective variant are executed equally	1
Variant 19	All activities in the respective variant are executed equally	1

Question 2 - Process Discovery

Part 1 - Splitting

The attribute *department* has 3 different values in the original log: **customer contact**, **experts**, and **general**. Using Disco, all event logs were saved with the corresponding department name. cf) file name: *randomized_filtered_event_log_internet_{customer_contact, experts, general}.xes*

Part 2 - Discover Petri nets

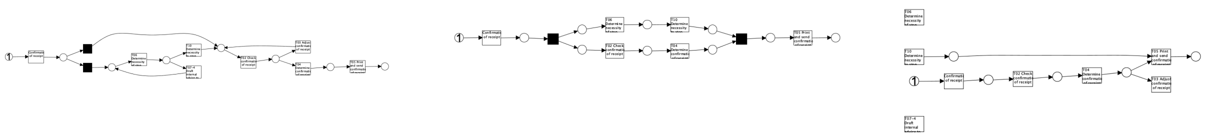
1. For all 3 event logs *randomized_filtered_event_log_internet_{customer_contact, experts, general}.xes* we need to discover a petri net that is i) sound, ii) covers approximately 80% of the traces, and has a precision above 0.5.
Using ProM6.10, we discovered petri nets with the following allowed and functioning plugins: *Mine Petri net with Inductive Miner*, *Interactive Data-aware Heuristic Miner*, and *Alpha Miner*.

(a) Plugin Inductive **(b)** Plugin Heuristic **(c)** Plugin Alpha

Figure 1: Plugins used to mine petri net

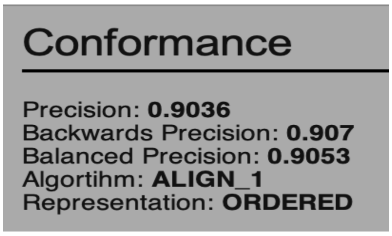
To ensure 80% of the traces, using Disco, we exported each event logs *randomized_filtered_event_log _internet_{customer_contact, experts, general}.xes* with path 20 and used as input for the plugins in ProM6.10. Of course, for the plugin *Mine Petri net with Inductive Miner* we have the option of using *Inductive Miner - Infrequent (IMf)* with noise = 0.2 to cover roughly 80% of the traces, but since other plugins do not support such functionality, for consistency, we applied a filtered input file for all plugins. For precision score we applied *Check Precision based on Align-ETConformance* by J.Munoz-Gama.

Note on parameters: We ended up applying noise = 0.2 for the Inductive Miner - infrequent, as we observed that, whether due to frequent error in Disco, there was no difference between filtered for path 20 and the original. Thus, for Inductive miner we applied noise 0.2. For Heuristic miner, we changed the dependency to 0.8 as there was no graph created with the default dependency 0.9. Lastly, for Alpha miner we manually entered source as initial marking and sink as final marking when applying the plugin for precision *Check Precision based on Align-ETConformance*. For rest of the parameter setting, we used the default setting. (For screenshots regarding plugins parameter settings, process and results, see "../Q2/P2")

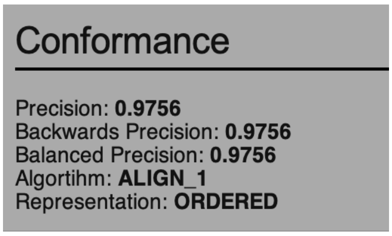


(a) Petri net Inductive **(b)** Petri net Heuristic **(c)** Petri net Alpha

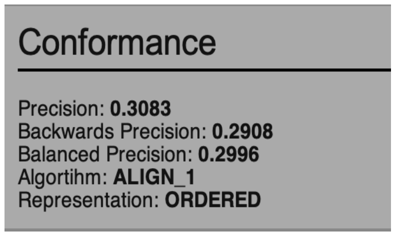
Figure 2: Petri nets for **customer contact**



(a) Precision Inductive

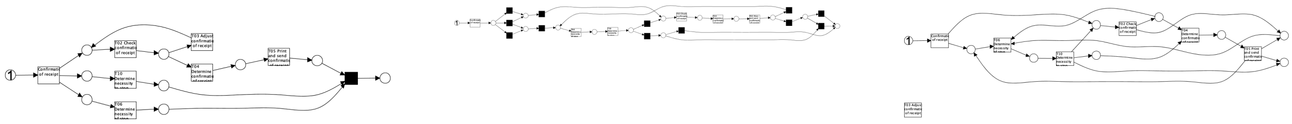


(b) Precision Heuristic



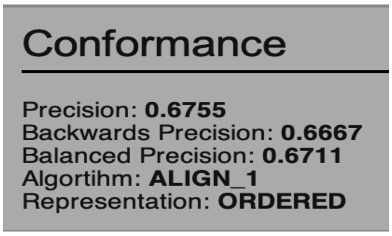
(c) Precision Alpha

Figure 3: Precision scores for **customer contact**

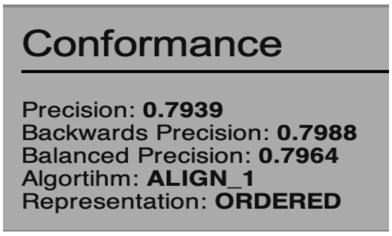


(a) Petri net Inductive **(b)** Petri net Heuristic **(c)** Petri net Alpha

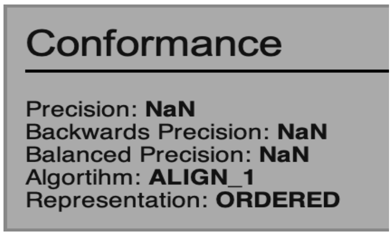
Figure 4: Petri nets for **experts** mined with inductive, heuristic, alpha miners



(a) Precision Inductive

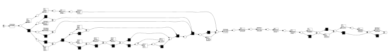


(b) Precision Heuristic

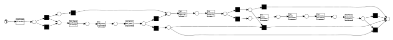


(c) Precision Alpha

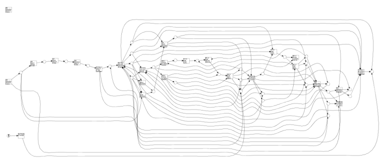
Figure 5: Precision scores for **experts**



(a) Petri net Inductive

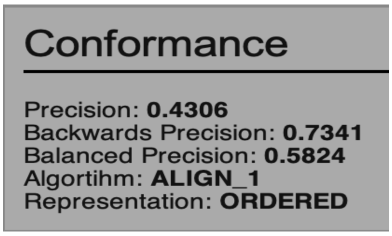


(b) Petri net Heuristic

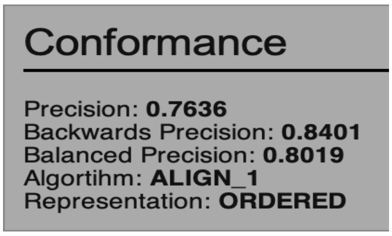


(c) Petri net Alpha

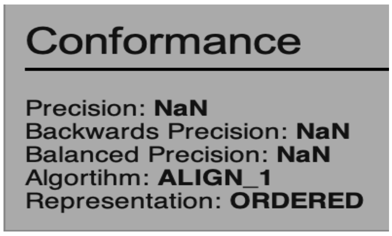
Figure 6: Petri nets for **general**



(a) Precision Inductive



(b) Precision Heuristic



(c) Precision Alpha

Figure 7: Precision scores for **general**

After creating at least 3 models for each event logs, we chose the following models with the highest precision for each event logs.

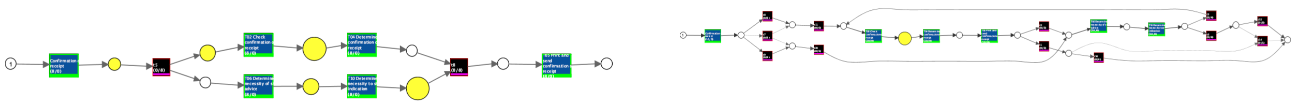
Table 3: Summary of best precision giving miner for each department log and precision scores

	customer contact	**experts**	**general**
Miner that gives highest precision	Heuristic	Heuristic	Heuristic
Precision score	0.9756	0.7939	0.7636

We learned that of various process mining methods, *heuristic mining*, *genetic mining* and *inductive mining* can account for incompleteness and noise. By applying various petri net mining techniques to our department logs, we confirm that heuristic and inductive mining deals better with incompleteness and noise, and provides better precision scores than, for example, alpha miner, and regional miners (state and language). Important to note is that although we had the highest precision with the heuristic miner, it may be that the inductive miner is the most practical and sound way to get the model as it gives a better balance of precision and generalization scores. This we have checked using the plugin *Measure precision/generalization* by Arya Adriensyah and getting precision and generalization score for most but the **general** inductive case, which our ProM6.10 crashed. (For screenshots, see "../Q2/P2") Also there is a chance that our heuristic miner was over-fitting due to the lower dependency setting of 0.8 we used instead of the default 0.9.

2. Comparing models for **experts** and **customer contact** there are 3 differences:

   - We observe more silent transitions in the petri net for the **experts**.
   - There is a parallel play in part, i.e. AND-split followed by AND-join, in **customer contact**, whereas the transitions in **experts** are played in sequentially.
   - There are loops in the petri net for **experts**, while there is no loop in the petri net for **customer contact**



(a) Replayed petri net for **customer contact** **(b)** Replayed petri net for **experts**

Figure 8: Replayed heuristic petri nets for **customer contact** and **experts**

3. Using Disco, the most frequent variants in each category are:

	general	**experts**	**customer contact**
Most frequent variant	Variant 1	Variant 1 and 2	Variant 1
Frequency	4375	21	18
Relative frequency	43.42%	27.63%	30.51%
Cases	625	3	3
Event per case	7	7	6
Activity	{Confirmation of receipt, T02 Check confirmation of receipt, T04 Determine confirmation of receipt, T05 Print and send confirmation of receipt, T06 Determine necessity of stop advice, T10 Determine necessity to stop indication, End}	{Confirmation of receipt, T06 Determine necessity of stop advice, T10 Determine necessity to stop indication, T02 Check confirmation of receipt, T04 Determine confirmation of receipt, T05 Print and send confirmation of receipt, End}	{Confirmation of receipt, T06 Determine necessity of stop advice, T10 Determine necessity to stop indication, T02 Check confirmation of receipt, T04 Determine confirmation of receipt, T05 Print and send confirmation of receipt}

Based on the table we observe the following similarities:

- All three department logs: **general**, **experts** and **customer contact** has around 6-7 events per case for the most frequent variant.
- Most frequent variant for all three department logs include start activity *Confirmation of receipt*
- Most frequent variant for all three department logs include the same activities, namely *T02*, *T04*, *T05*, *T06*, *T10*.

Some dissimilarities are:

- Most frequent variant for **customer contact** does not include the *End* activity.
- Each cases of most frequent variant account for different relative frequency.
- Most frequent variant for **general** has roughly 200 times more cases than that in other logs.

4. Generated C-net from event log
We used ProM6.10 and applied the plugin *Interactive Data-aware Heuristic Miner (iDHM)* to generate the following C-Net from the event log *randomized_filtered_ event_ log_ internet _general_ .xes*, containing roughly 15% of directly follows relations. We were given 4 available event classifiers, we chose the *Event Name*. The model has 92 directly follows relations. Hence 15% of those corresponds to 13/92, which results in *0.15 Frequency*. We chose *Dependency* of *0.8*. The other two, Bindings and Conditions are set by default.

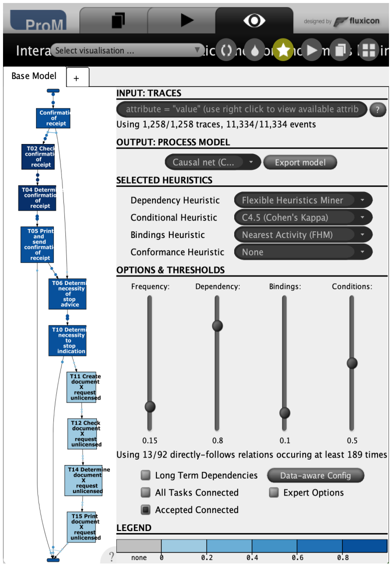


Figure 9: C-net

Two possible traces are:

1. L_1 = <Confirmation of receipt, T02 Check confirmation of receipt, T04 Determine confirmation of receipt, T05 Print and send confirmation of receipt, End>

2. L_2 = <Confirmation of receipt, T02 Check confirmation of receipt, T04 Determine confirmation of receipt, T05 Print and send confirmation of receipt, T06 Determine necessity of stop advice, T10 Determine necessity to stop indication, End>

Question 3 - Conformance Checking

Using Disco, we created the *variants_ event_ log.xes* from *randomized_ event_ log.xes*. When applying the filter to have roughly 85% of the customer's paths, we could only extract, from our sample, 14 most frequent variants instead of the 15. Hence, our results provided for this question are based on these 14 most frequent ones.

Based on *variants_ event_ log.xes*, we discover three models of varying degrees of noise threshold. We used ProM6.10 plugin *Mine Petri net with inductive miner* by S.J.J. Leemans with default settings(Variant = Inductive miner-infrequent (IMf)) and mined separately for noise thresholds $= \{0.0, 0.1, 0.2\}$. Please note the following figures for noise threshold 0.0 apply to noise setting 0.1 and 0.2 with respective changes in the noise threshold.

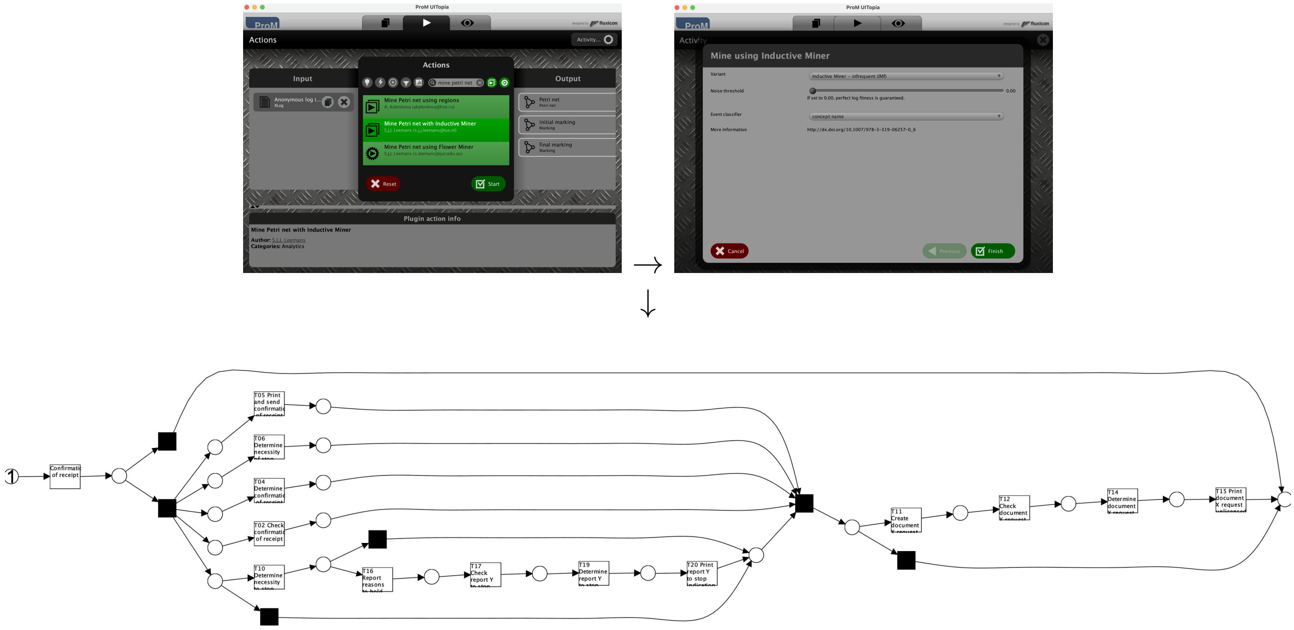


Figure 10: Inductive miner plugin, parameters and discovered petri net model for noise threshold 0.0

We then select the log and the respectively discovered petri net as input and use the plugin *Align log and model for repair(global costs)* by D. Fahland to get petri net replay result.

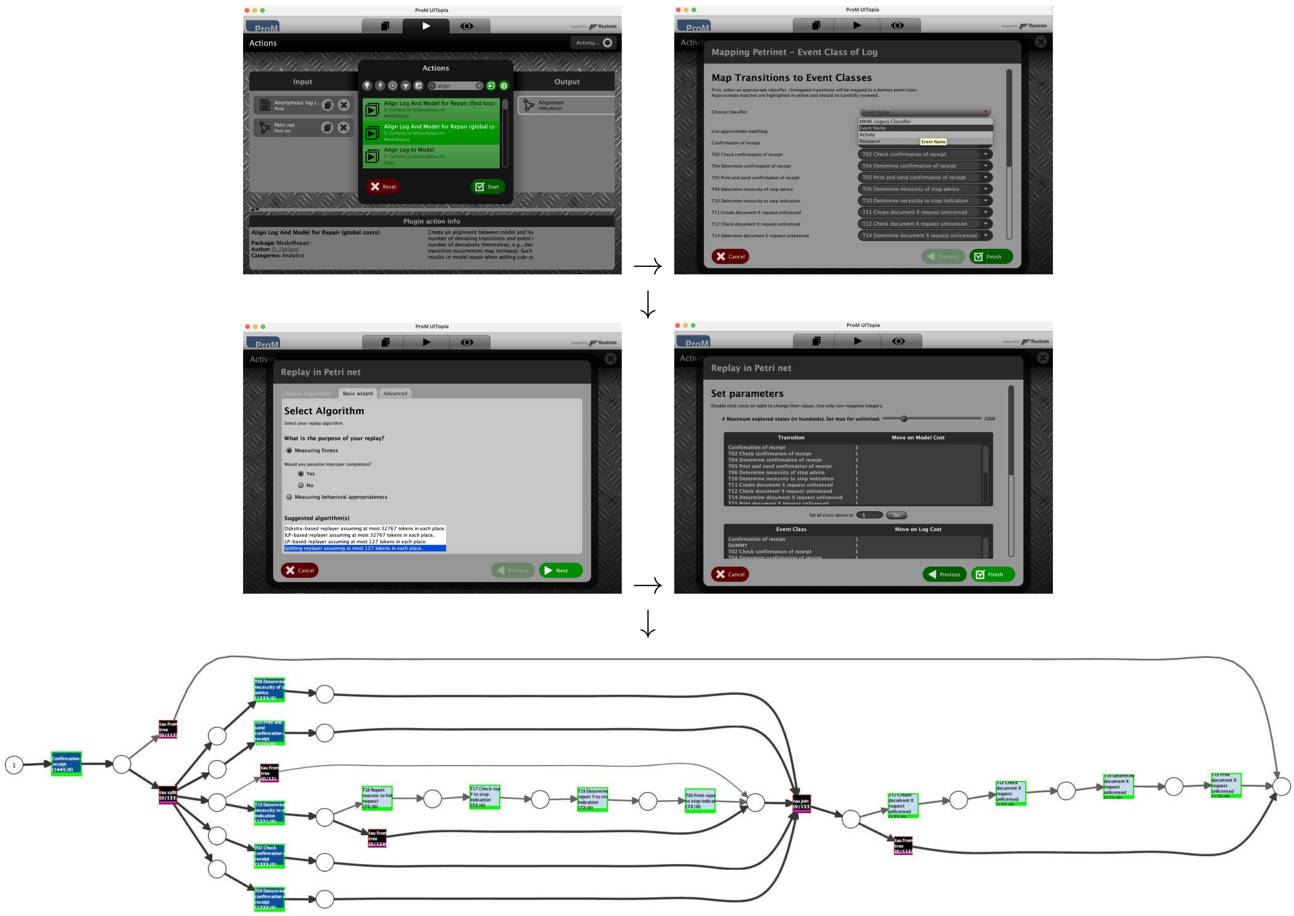


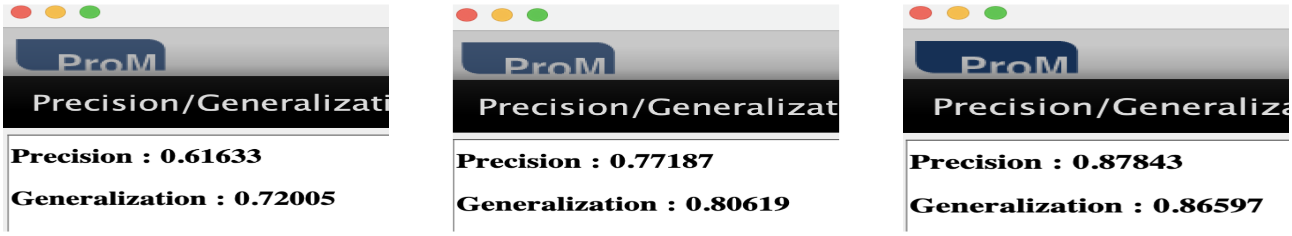
Figure 11: Replay plugin, parameters and replayed petri net for noise threhold 0.0

Now we apply the four quality criteria and deviations in behaviour for the three discovered

models.
The four quality measures are:

- Fitness (ability to explain observed behavior)
- Generalization (avoiding overfitting)
- Simplicity (Occam's razor)
- Precision (avoiding underfitting)

To discover the precision and generalization, we applied the plugin *Measure Precision/Generalization* by Arya Adriansyah under default settings and with the log, petri net & align as input.



(a) Noise Threshold 0.0 **(b)** Noise Threshold 0.1 **(c)** Noise Threshold 0.2

Figure 12: Precision and generalization for different noise threshold settings

- **Interpretation for model discovered from noise threshold 0.0:** After selecting the "all elements, aggregated" in element statistics, we see that the "number of move model only" is 5197 while the "number of move log only" is 0 which implies no problem and only moves on model for silent steps. As a result, it explains the observed behaviour perfectly with trace fitness, move-model fitness and move-log fitness as 1. Thus, it is a perfect fit.

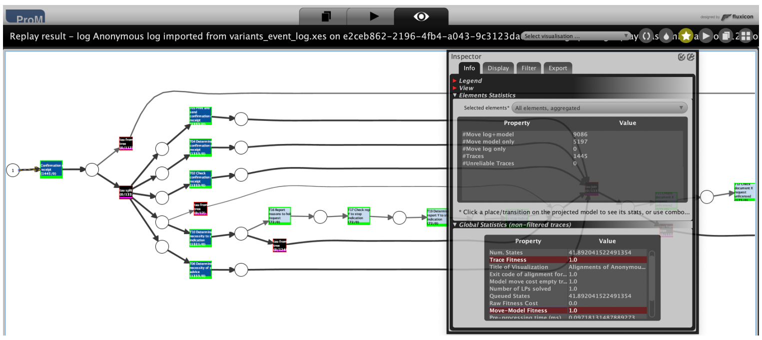


Figure 13: Quality measures for model from noise threshold 0.0

- **Interpretation for model discovered from noise threshold 0.1:** After selecting the "all elements, aggregated" in element statistics, we see that the "number of move model only" is 6177 while the "number of move log only" is 288 and the trace fitness is 0.9385.

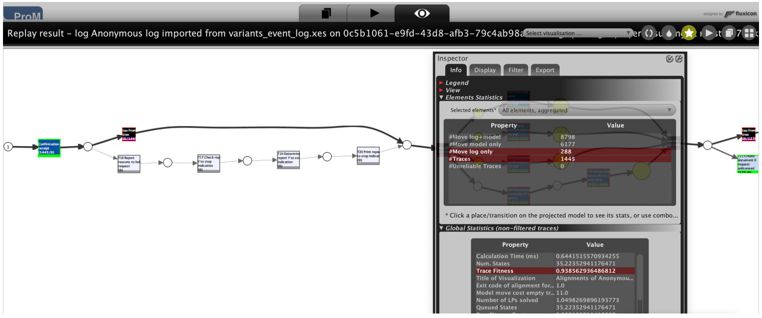


Figure 14: Quality measures for model from noise threshold 0.1

- **Interpretation for model discovered from noise threshold 0.2:** After selecting the "all elements, aggregated" in element statistics, we see that the "number of move model only" is 4472 while the "number of move log only" is 1257 and the trace fitness is 0.9185.

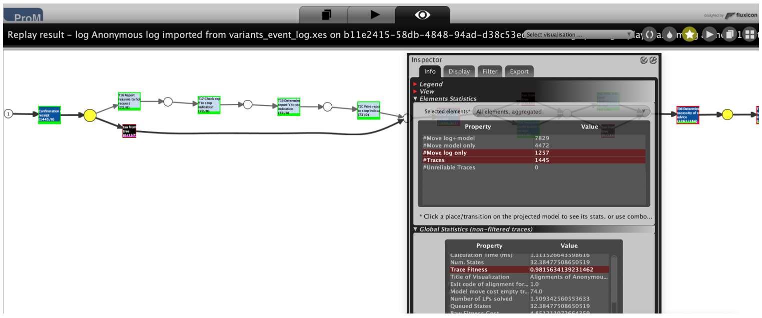


Figure 15: Quality measures for model from noise threshold 0.2

Question 4 - Decision Mining

1. Using the inductive miner with default settings, we discovered the petri net model of the *randomized_ event_ log* in ProM6.10 by importing the log and using the plugin *Mine Petri net with inductive miner* by S.J.J. Leemans with default settings (Variant = Inductive miner-infrequent (IMf). We discovered one decision point marked in figure 16.(For better quality screenshot, please refer to "../Q4")

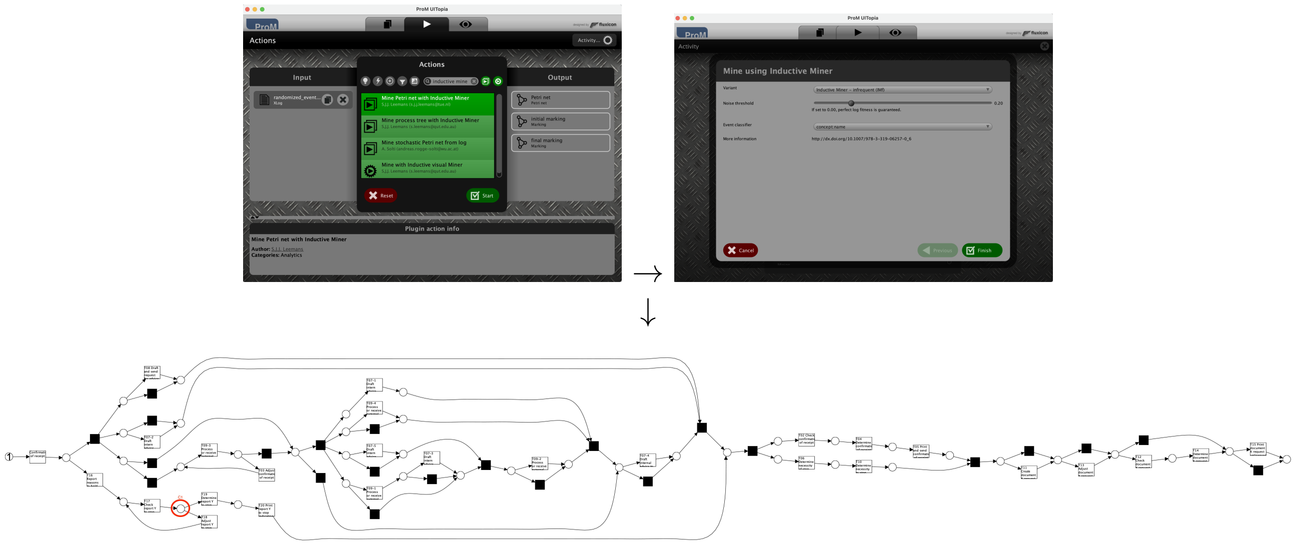


Figure 16: Plugin settings and discovered Petri net model

2. From the identified decision point in the petri net in (1), we chose a subset of relevant attributes: **cost**, and **org$3Agroup**.
   Using ProM6.10, we discovered the decision tree by:
   - Selecting the "petri net" and the *randomized_event_log* and using the plugin *Discovery of the Process data-flow (Decision Tree Miner)* by Massimilliano de Leoni, F.Mannhardt
   - In the configuration tab, selecting the two attributes **cost**, and **org$3Agroup** in the "Variables Considered" dropdown and keeping all the other options default

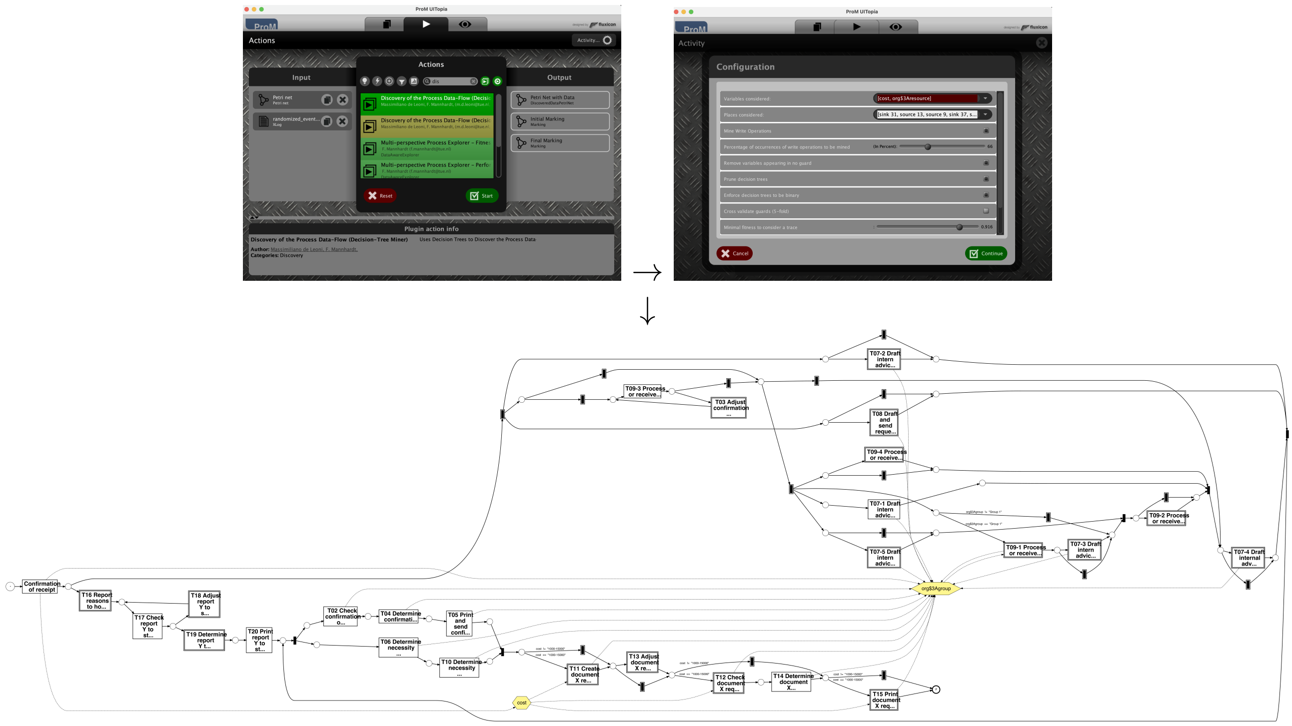


Figure 17: Plugin setting and discovered decision tree

The quality measures from the decision tree discovered from the same plugin are as follows:

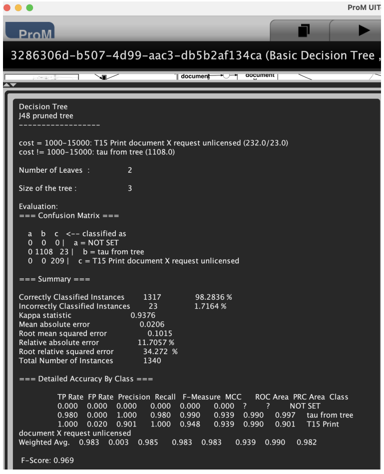


Figure 18: Quality measure from decision tree

Table 4: The weighted average of quality measures of decision tree.

Measures	Precision	Recall	F-Measure
weighted average	0.985	0.983	0.983

Since precision is the fraction of relevant instances among the retrieved instances while recall is fraction of relevant instances that are retrieved, we can deduce that higher value of precision indicate low false positive rate. Thus, a value of 0.985 for precision and a recall of 0.983 indicate that the decision tree is meaningful.

3. Using the decision tree plugin, we discover that the transition guards of "T14 Determine document X request unlicensed" are *cost !="1000-15000"* and *cost == "1000-15000"*, which implies activity "T14 Determine document X request unlicensed" follows "tau from tree" if *cost !="1000-15000"* while "T14 Determine document X request unlicensed" follows "T15 Print document X request unlicensed" if the *cost == "1000-15000"* that writes - **org$3Agroup**.

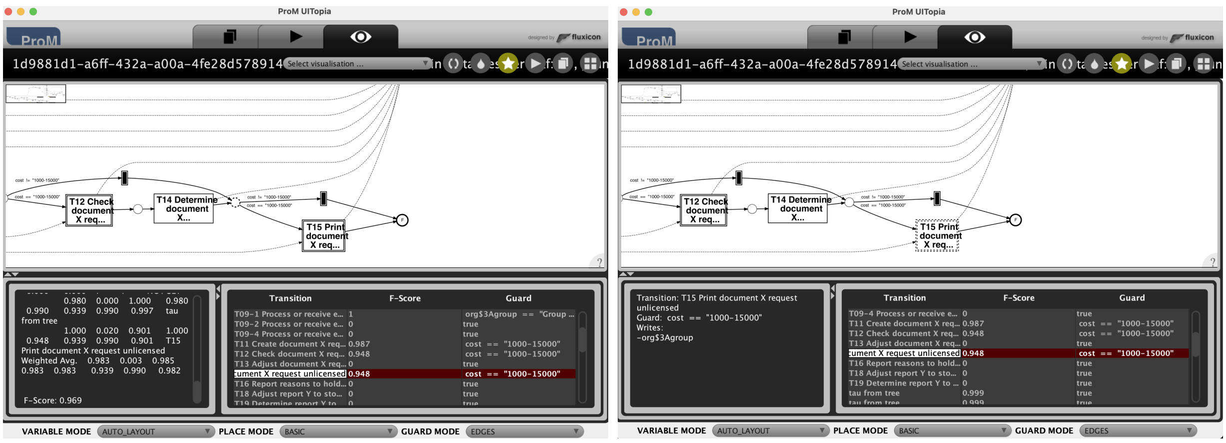


Figure 19: Transition guards from decision tree

Question 5 - Performance Analysis

1. We used ProM6.10 with the event log *randomized_event_log.xes* and the plugin *Replay a log on Petri Net for Performance/Conformance Analysis*. This plugin takes a petri net

as input, which we mined via the plugin *Mine Petri Net with Inductive Miner*(see Figure 8.) Following are screenshots that show the steps to analyze the performance of the event log as well to visualize of the petri net with the bottleneck marked as red.

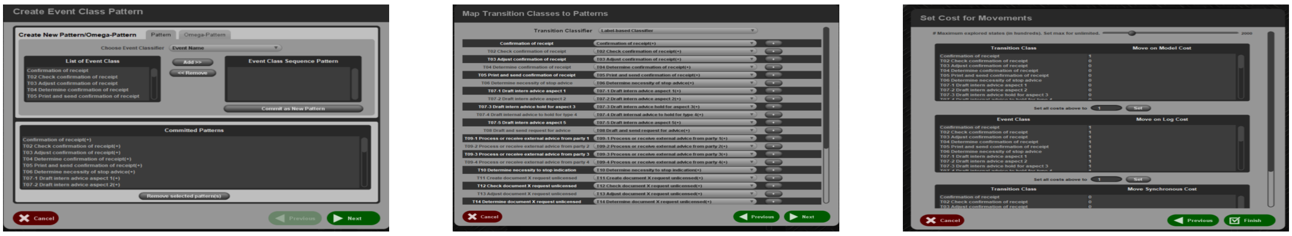


Figure 20: Settings of *Replay a log on Petri Net for Performance/Conformance Analysis*

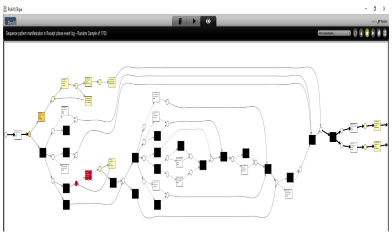
Visualizations like this one provide either conformance or performance information. Colors or sizes of the places or transitions can change as well as the thickness of the arcs. If we use time as a dimension the bottleneck occurs in the place that has been turned red, *T09 Process or receive external advice from party 3*.

The combobox provides all the information we need in order to justify why *T09 Process or receive external advice from party 3* is the bottleneck.

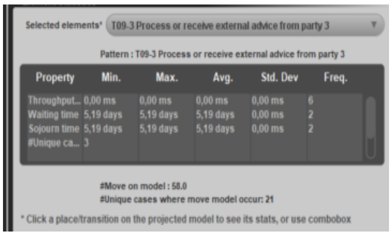
a) Case Throughput time (avg): 6.30 days
b) Case Throughput time (min): 0.00 mins
c) Case Throughput time (max): 9.19 months

Regarding the bottleneck we get the following sojourn time:

a) Sojourn time (avg): 5.19 days
b) Sojourn time (min): 5.19 days
c) Sojourn time (max): 5.19 days



(a) Petri net visualization



(b) Sojourn time of bottleneck

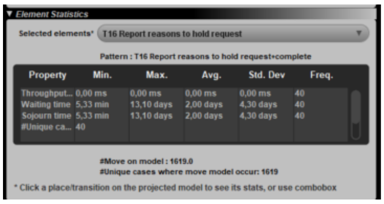
(c) Throughput time

Figure 21: Sojourn and throughput time

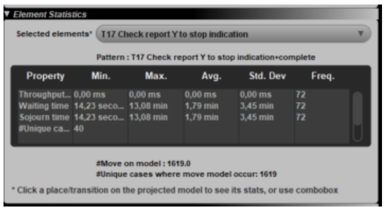
If we compare this activity to others, it is clear that there is a huge delay on this one. For instance, if we choose activity *T19 determine report Y to stop indication* we see that:

a) Sojourn time (avg): 30.39 secs
b) Sojourn time (min): 56.97 secs
c) Sojourn time (max): 13.35 secs

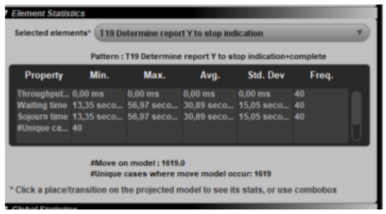
Similarly, for several other activities we provide the screenshots, that shows that the average time of each of them is less than the average time of the bottleneck.



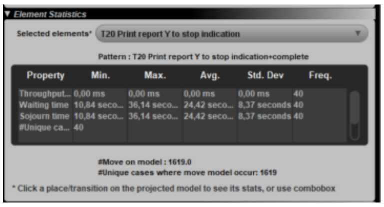
(a) T16 Report reasons to hold requests



(b) T17 Check report Y to stop indication



(c) T19 Determine report Y to stop indication



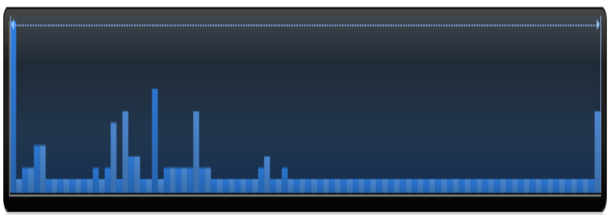
(d) T20 Print report Y to stop indication

Figure 22: Average sojourn time of several activities

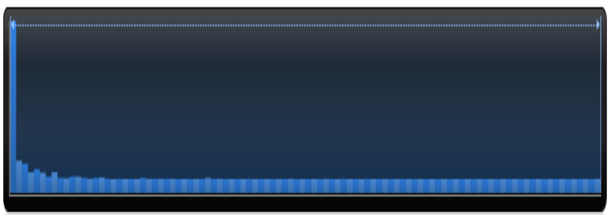
Since the bottleneck occurs for *T09 Process or receive external advice from party 3*, it could be suggested that the process or receive of external advice from party 3 could be replaced by an internal procedure. This way, the time of this activity would be significantly decreased as it won't be influenced by any external delays.

2. To answer this question both ProM6.10 and Disco have been used. Firstly, we inserted the *randomized_event_log.xes* into Disco. Then, by arranging the cases duration from the event log in decreasing order we saw that t_min = 0, t_max = 275 days and 20 hours and t_mid = 55 days. Afterwards, we filtered the event log by using *Timeframe*. The first event log starts on 02/10/10 and ends on 23/11/10. The second one starts on the 24/11/10 and ends on 23/01/12.

   In terms of performance analysis, we observe that unlike the process for traces with the throughput time in the first interval, in the second one, besides the first case, which has significantly high duration, the rest are almost close to zero.



(a) Case duration over number of cases in min_to_mid



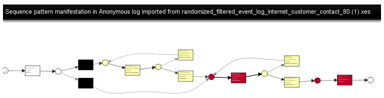
(b) Case duration over number of cases in mid_to_max

Figure 23

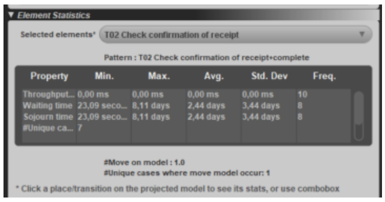
3. For questions 3 we followed the same procedure as in question 1, this time for the *randomized_filtered_event_log_internet_{customer_contact, experts, general}.xes* .

   We observe that the bottlenecks differ in each department. In detail,

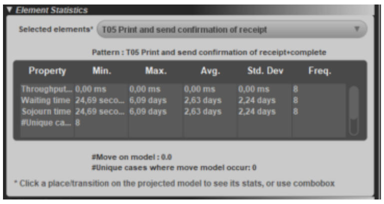
   a) In customer contact currently exist two bottlenecks, *T02 Check confirmation of receipt* and *T05 Print and send confirmation of receipt*. Both are related to the Confirmation of receipt phase.



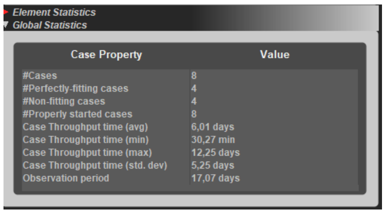
(a) Petri Net Visualization for the customer contact dept.



(b) T02 Check confirmation of receipt



(c) T05 Print and send confirmation of receipt

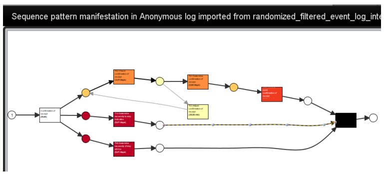


(d) Throughput time

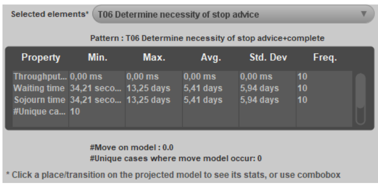
Figure 24: Average Sojourn and Throughput time of several activities within the customer contact dept.

b) In **experts** there exist two bottlenecks as well, *T06 Determine necessity of stop advice* and *T10 Determine necessity to stop indication.*

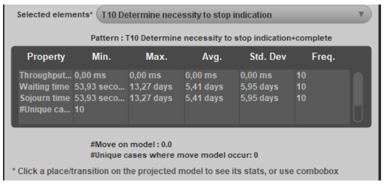
c) In general there is a single bottleneck, *T07-4 Draft internal advice to hold for type 4*, which is actually connected to some internal processes within the company.



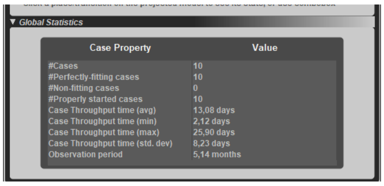
(a) Petri Net Visualization for the **experts** dept.



(b) T06 Determine necessity of stop advice

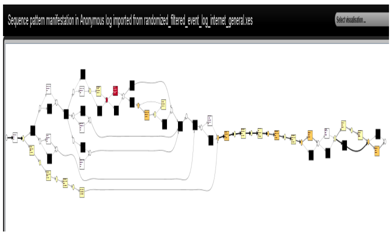


(c) T10 Determine necessity to stop indication

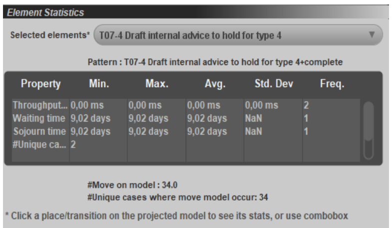


(d) Throughput time

Figure 25: Average Sojourn and Throughput time of several activities within the **experts** dept.



(a) Petri Net Visualization for the general Dept.



(b) T07-4 Draft internal advice to hold for type 4

(c) Throughput time

Figure 26: Average Sojourn and Throughput time of several activities within the general dept

Question 6 - Organizational Mining

Part 1

To create the following *Handover of work network* we, firstly, filtered the *randomized_ event_ log.xes* in Disco, by applying the variation filter, to obtain an event log, that contains only the most frequent variant. Next, we applied the plugin, *Mine for a Handover-of-Work Social Network* on default settings. Next, we provide an image of the Handover of work network as well as a table containing some of the exceptional nodes within it (Table 5).

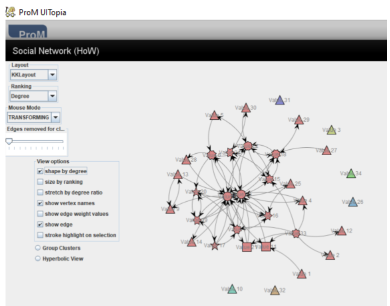


Figure 27: Handover of work network

Table 5: Exceptional Nodes

Nodes	Resources
Not connected	• Value 3 • Value 32 • Value 26 • Value 34 • Value 31 • Value 10
lots of incoming / outgoing arcs	• Value 23 • Value 22 • Value 24 • Value 6 • Value 18 • Value 17
Only in or outgoing arcs	• Value 27 • Value 29 • Value 1 • Value 12 • Value 2

Considering that each *value* represents a resource we might think that the handover represents a business hierarchy, meaning a pyramid-like structure used to organize employees into distinct levels. Hence we assume that the nodes with multiple incoming and arcs might correspond to employees higher up on the pyramid, who tend to have more responsibilities and higher salaries. Nodes with fewer arcs might represents the supervisors, who are responsible of training and evaluating the employees. Multiple nodes connected together might represent the employees, who tend to be the largest category and often have common roles within the company. Lastly, the non-connected nodes might correspond to independent contractors, who work for a company on a freelance basis.

Part 2

In order to find clusters for resources that work on similar tasks given the correlation coefficient, we used the *Mine for a similar-Task Network*. Out of the four options listed, we chose the Correlation coefficient, as it has been asked. The resulted clustering can be seen below.

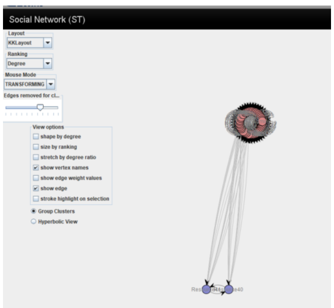
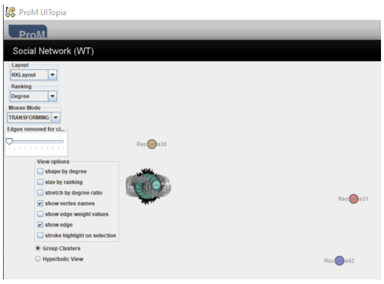
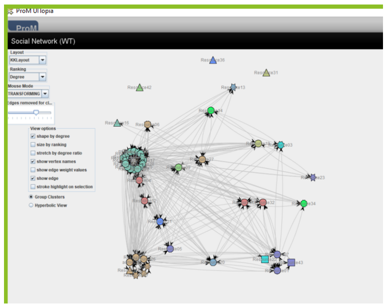


Figure 28: Resources working together splitted into 2 clusters

As seen in the picture above, *Resources 40 & 41* are grouped together into the smaller cluster.



(a) Resources working together splitted into 2 clusters



(b) Clustering that contains multiple cluster of resources

Part 3

For this task *(see image 23)* we used the *Mine for Working-Together Social Network*. By applying the lowest threshold, we observe that all resources are grouped together apart from *Resources 36, 42, 11*

For this task *(see image 24)* we used, again, the *Mine for Working-Together Social Network*. By applying the threshold accordingly we get the following clustering.
We note that there is one cluster, which contains several resources with different degrees, meaning that these nodes have different number of connections.Moreover, there two smaller clusters, which also contain nodes with a variant number of connections. There are also several small clusters, which consist of either one or two nodes. The only resource that doesn't belong to any cluster is *Resource 42*.

Conclusion

After a thorough analysis of the record and the event log, we learned the importance of having a well sampled event log with complete traces to have better comprehensibility. Generally examining the filtered event log we found most cases and events of building permits were handed over via channel: *internet*. We also observed that in channel: *desk*, activity *T06* was executed most frequently per variant. In process discovery, we found that inductive and heuristic miner can handle incompleteness relatively better than alpha and (state, language) regional miners, while inductive miner generally tends to give a better balanced precision and generalization score. Using heuristic miner, we were also able to deduct causal relation of transitions via c-net. Furthermore, we looked into the conformance with varying noise thresholds, which revealed a decreasing trace fitness score as noise threshold increases from 0 to 0.2. We also deducted two attributes that influence the decision e.g. **cost** and **org$3Agroup** and found interesting decision points with guards that writes the data attribute. Additionally, we conducted performance analysis of various petri nets, in terms of the time parameter rather than frequency or resources. We discovered that a bottleneck occurs on a place, when its average sojourn time is significantly higher than the rest of places within a trace. In the last part, organizational mining, we used different concepts, such as handover of work network or resources that work on similar clusters, retrieving information in how to perform a meaningful and effective clustering.