Business Process Intelligence SS23

Report Assignment Part 1

Submitted To:

Prof. Dr. Wil van der Aalst, Bianka Bakkulari M. Sc., Harry Beyel M. Sc., Benedikt Knopp M. Sc., Nina Graves M. Sc., Christopher Schwanen M. Sc. Chair of Process and Data Science
RWTH Aachen University

Submitted By:

Lilly Puppala (441273)

Ramanuj Sharma (441358)

Ozan Sap (411851)

Table of contents

Question 1: PQL

a) The following PQL formulae are used to create the OLAP table (Refer to Figure 1) -

1. **CASE ID:** "case_table"."case:concept:name"

2. **DEFECT TYPE:** "case_table"."DEFECTTYPE"

3. **PHONE TYPE**"case_table"."PHONETYPE"

4. **DOCUMENT HAPPENED:** COUNT (CASE WHEN "activity_table"."CONCEPT:NAME"='Document' THEN 1 END)

5. **DEFECT FIXED:** CASE WHEN PU_LAST ("case_table", "activity_table"."DEFECTFIXED") = 1 THEN 'TRUE' ELSE 'FALSE' END

6. **THROUGHPUT TIME:** CALC_THROUGHPUT(ALL_OCCURRENCE['Process Start'] TO ALL_OCCURRENCE['Process End'], REMAP_TIMESTAMPS("activity_table"."TIME:TIMESTAMP", MINUTES))

7. **NUMBER OF REFURBISHED (SIMPLE):** COUNT (CASE WHEN "activity_table"."CONCEPT:NAME"='Refurbish (Simple)' THEN 1 END)

8. **NUMBER OF REFURBISHED (SIMPLE):** COUNT(CASE WHEN "activity_table"."CONCEPT:NAME"='Refurbish (Complex)' THEN 1 END

Case ID	Defect Type	Phone Type	Document Happ...	Defect Fixed	Throughput	Number of Refurbished(Sim...	Number of Refurbished(Com...
1	6	T2	1	TRUE	47	0	1
2	8	T2	1	TRUE	81	0	1
3	2	T2	1	TRUE	48	1	0
4	1	T1	1	TRUE	64	1	0
5	5	T1	1	TRUE	44	0	1
6	5	T1	1	TRUE	63	0	1
7	10	T2	1	TRUE	159	0	3
8	3	T3	1	TRUE	58	2	0
9	3	T3	1	TRUE	57	1	0
10	3	T1	1	TRUE	65	2	0
11	10	T3	1	TRUE	75	0	1
12	2	T3	1	TRUE	84	2	0
13	7	T1	1	TRUE	86	0	1
14	8	T3	1	TRUE	102	0	2
15	7	T3	1	TRUE	75	0	1
16	7	T2	1	TRUE	52	0	1
17	10	T3	1	TRUE	108	0	3
18	5	T3	1	TRUE	75	1	1
19	9	T3	1	TRUE	87	0	1
20	8	T1	1	TRUE	66	0	1
21	4	T1	1	TRUE	81	2	0
22	8	T2	1	TRUE	70	0	1
23	5	T3	1	TRUE	59	0	1
24	8	T2	1	TRUE	50	0	1
25	7	T3	1	TRUE	61	0	1
26	8	T2	1	TRUE	73	0	1
27	1	T3	1	TRUE	46	1	0
28	7	T1	1	TRUE	89	0	1
29	10	T2	1	TRUE	89	0	1
30	8	T2	1	TRUE	58	0	1

Figure 1: Resulting OLAP table

b) We can simply observe that when we raise the connection percentage to 84.1 (Figure 2b), there is a new arc in the process explorer with 307 traces from ‘Detect Required Improvement’ to ‘Refurbish (Simple)’. This arc depicts that the process can now skip ‘Notify Sales’. Refer to Figure 2a and Figure 2b.

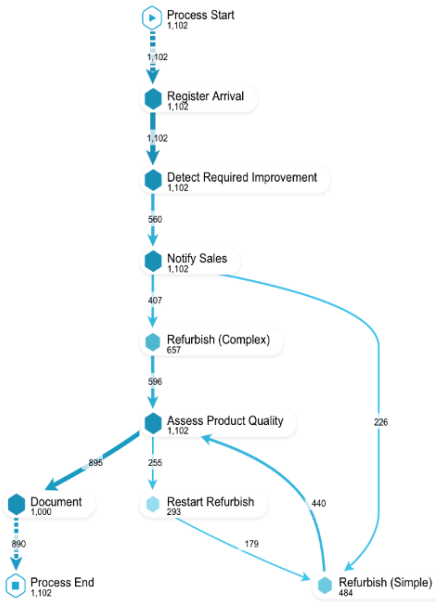


Figure 2a: Process Explorer 80.6% Connections

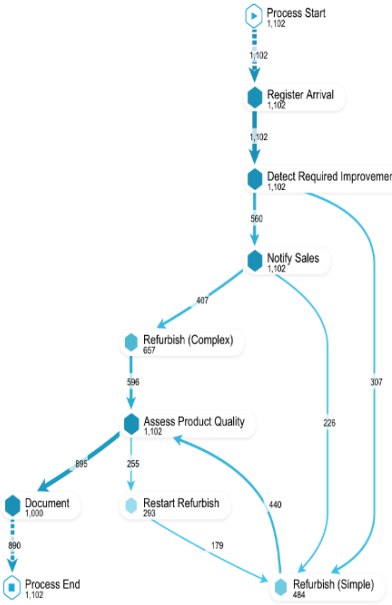


Figure 2b: Process Explorer 84.1% connections

Figure 2: Process Explorer

c) As we can see from the trace diagram of the most common variant (Figure 3a), there is nearly 30% (333 cases) of the traces through ‘Refurbish (Complex)’ whereas in Figure 3b, we can observe that when we consider 2 most common variants, there is a new path with 104 cases through ‘Refurbish (Simple)’ taking the total number of cases to 437.

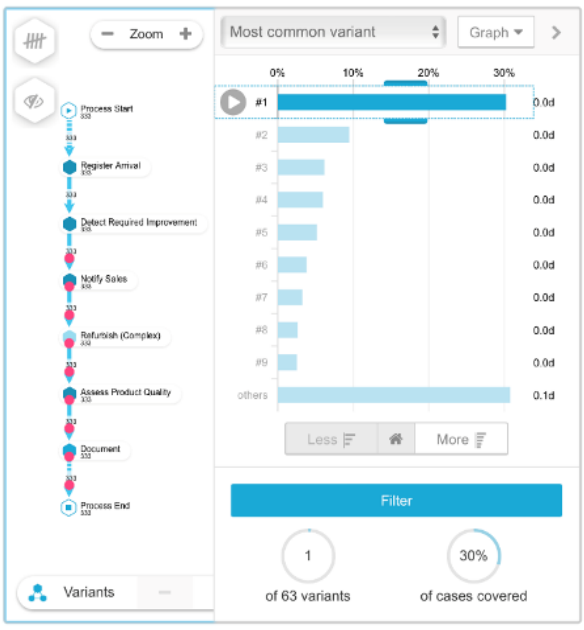


Figure 3a: Most Common Variant

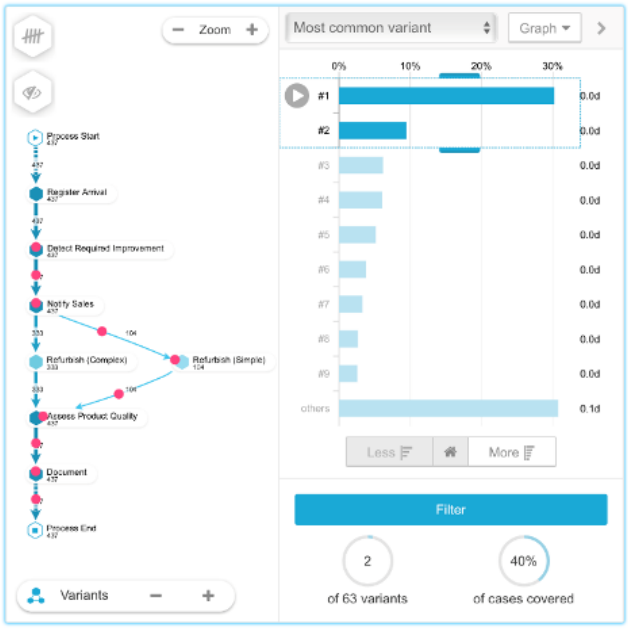


Figure 3b: Two most common variants

Figure 3: Variant Explore

Question 2: Decision Tree

a)

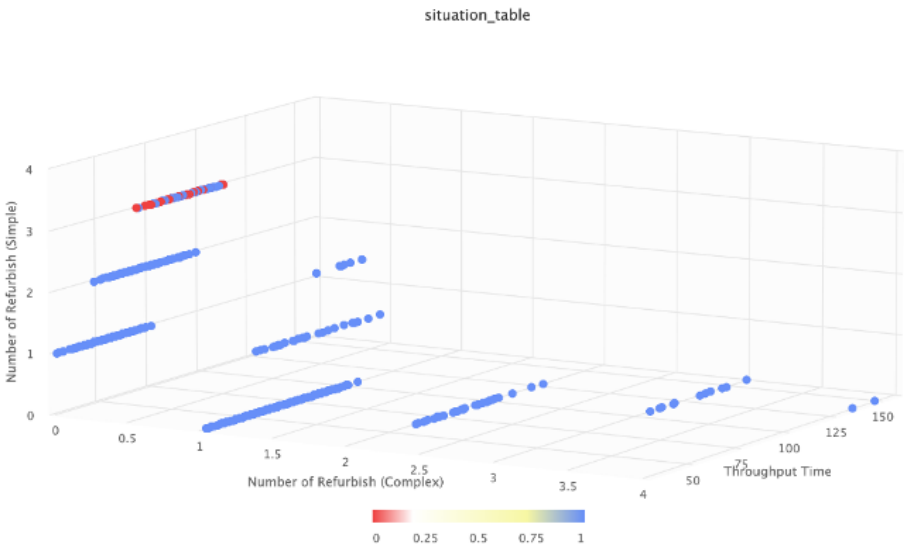


Figure 4: 3D Scatter Visualizations

From Fig. 4, we can say that all the instances that could not be fixed (red) do not have any value from 'number of refurbished (complex)' and have a value of 3 from 'number of refurbished (simple).' We can also observe that these instances lie between the throughput value 68 minutes to 111 minutes (about 2 hours). We can also say that the maximum number of refurbishment cycles first appears to be three. However, if we look at the right side of the plot, we can see 2 cases that required four refurbishment cycles before the defect was fixed (blue). As a result, this idea seems to be limiting. Even for the throughput time we can state that values don't vary much when defect fixed and not fixed as they coincide with each other in many cases.

b) Figure 5 shows the pipeline of the Decision Tree with parameters provided by the instructors.

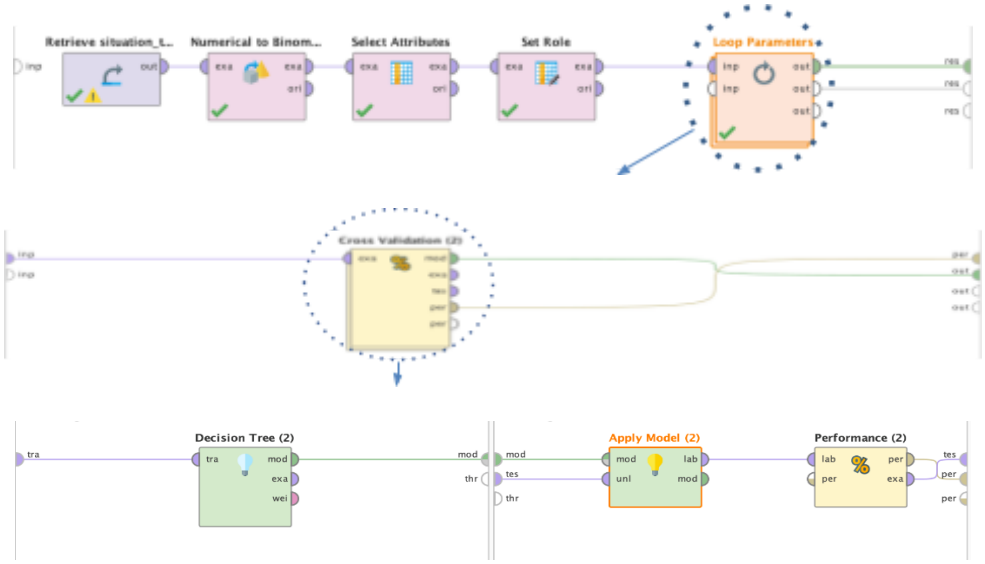


Figure 5: Pipeline of Decision Tree

iteration	Decisio...	Decisio...	Cross Validation (...	Cross ...	acc...
20	6	gini_index	true	5	0.970
3	3	gain_ratio	true	5	0.968
12	5	informa...	true	5	0.967
6	6	gain_ratio	true	5	0.967
5	5	gain_ratio	true	5	0.967
13	6	informa...	true	5	0.966
19	5	gini_index	true	5	0.966
7	7	gain_ratio	true	5	0.966
10	3	informa...	true	5	0.966
1	1	gain_ratio	true	5	0.966
16	2	gini_index	true	5	0.966
2	2	gain_ratio	true	5	0.966
15	1	gini_index	true	5	0.966
8	1	informa...	true	5	0.966
9	2	informa...	true	5	0.966
14	7	informa...	true	5	0.965
4	4	gain_ratio	true	5	0.964
21	7	gini_index	true	5	0.963
17	3	gini_index	true	5	0.962
11	4	informa...	true	5	0.960
18	4	gini_index	true	5	0.960

Figure 6: Accuracy Table

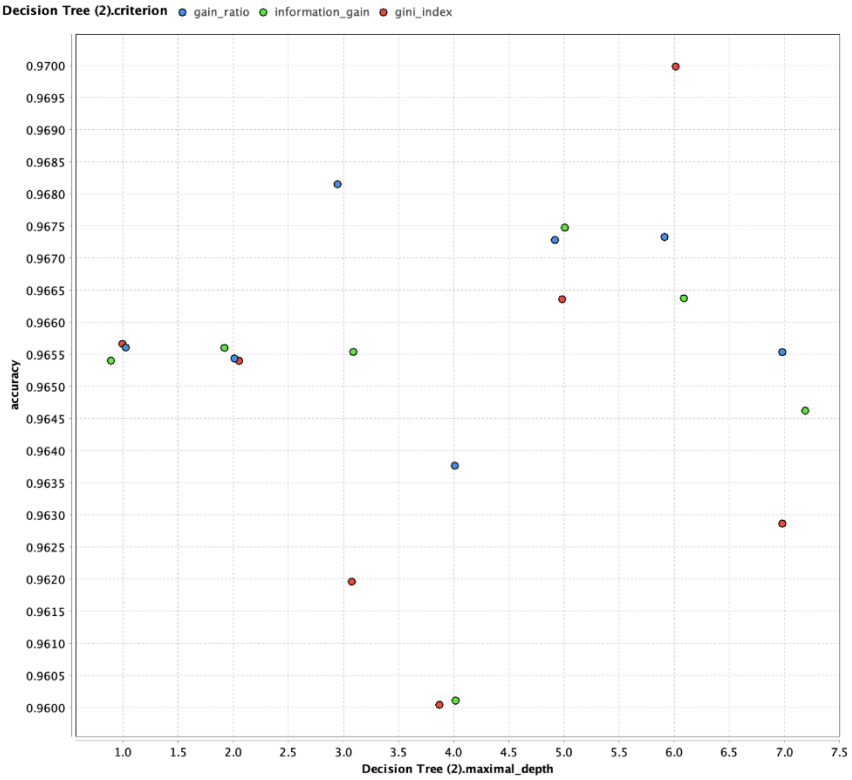


Figure 7: Accuracy vs Depth and Criteria Graph

Figure 7 shows the plot between accuracy vs depth which is obtained from the table shown above in figure 6. Jitter has been applied for a better understanding of criteria with accuracy which shows that the maximum accuracy is obtained for a tree with depth 6 and criteria is gain-ratio. The tree is shown in figure 8.

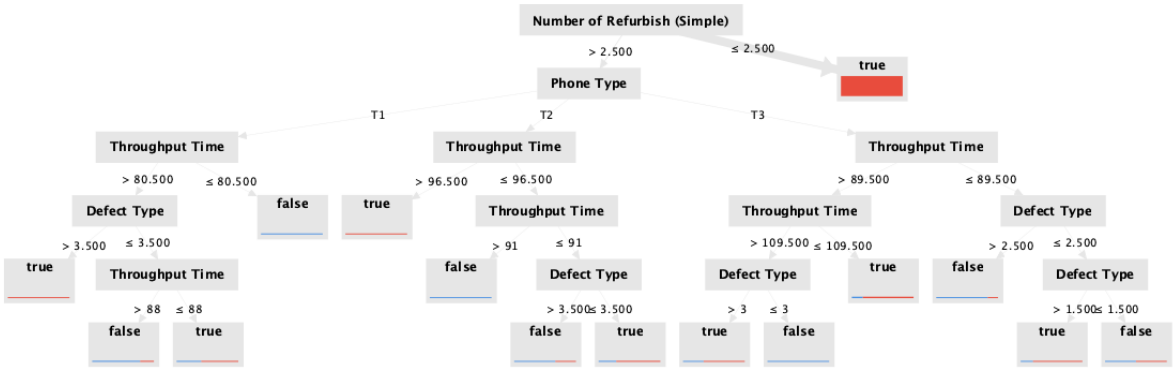


Figure 8: Decision Tree with the highest accuracy

c) The tree shown in figure 8 which answers about the leaf with the most accuracy has the number of refurbish (simple) less than or equal to 2.500 having a total of 1014 true and 0 false cases. The other one has a phone type T1, throughput time less than or equal to 80.500 having a total of 5 false and 0 true cases.

Question 3: Clustering

a) To use K-means Clustering (k=3), we have used Select Attributes to consider only the attributes that are mentioned in the question. We have used the Z-transformation technique for normalization and join to show the unnormalized data (Figure 9). We have attained the following clusters -

Cluster 0: 620 items	Cluster 1: 267 items	Cluster 2: 215 items

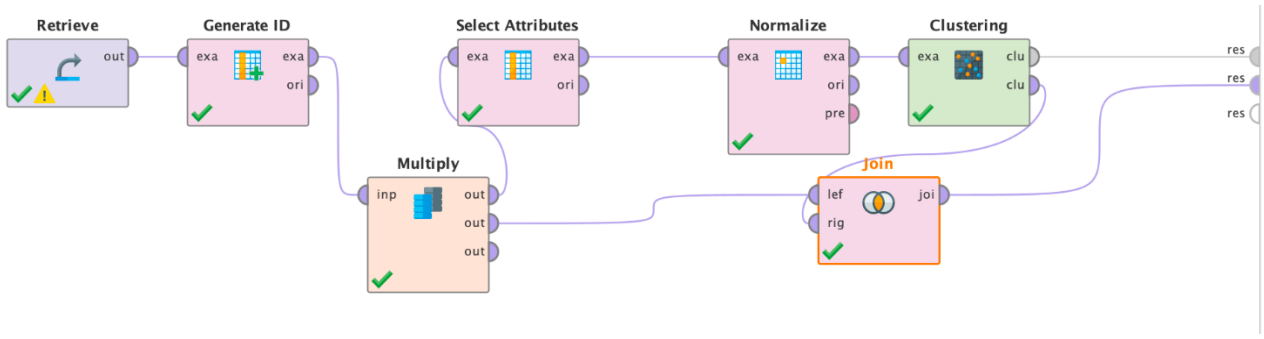


Figure 9: Pipeline for K-Means Clustering

From the 3D Scatter Visualization (Figure 10), we can observe the following attributes for each cluster:

- Cluster 0: Most of the cases in this cluster have either 0 or 1 number of refurbishments for simple as well as complex attributes. The throughput time is between 29-70 minutes.
- Cluster 1: The cases in this cluster have at least 1 complex refurbishment going up to 4 and most of the cases have 0 simple refurbishments with some cases having. The minimum throughput time is 67 minutes, and the maximum is up to 159 minutes.
- Cluster 2: This cluster has only 5 cases with complex refurbishments and the rest have 0. They have at least 1 simple refurbishment going to 3 as the maximum.

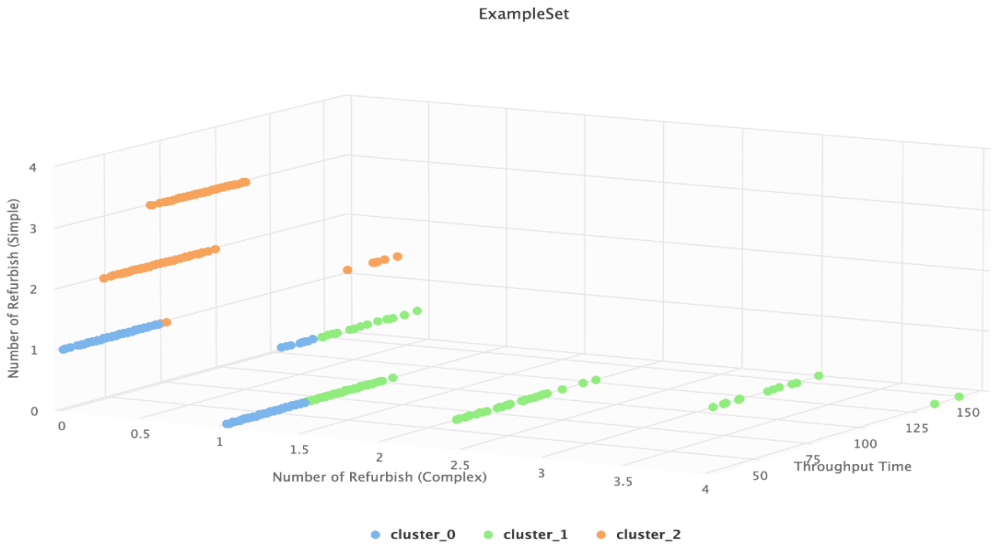


Figure 10: 3D Scatter Visualization for 3 Clusters

b) This pipeline (Figure 11) is similar to that K-means with the only difference being that of the De-Normalize operator which has been used to denormalize the data.

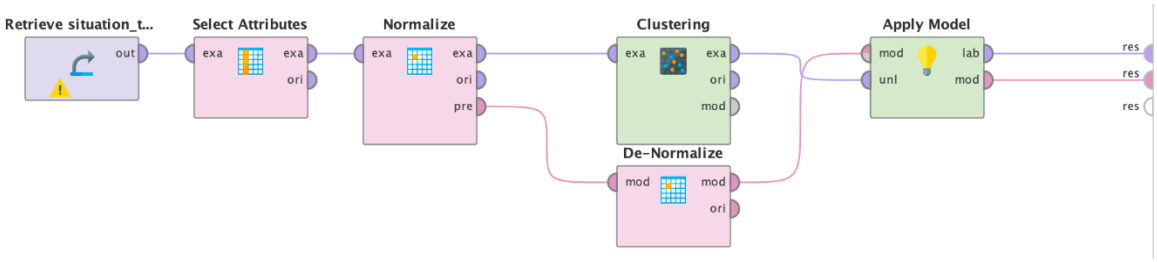


Figure 11: Pipeline for DBSCAN Clustering

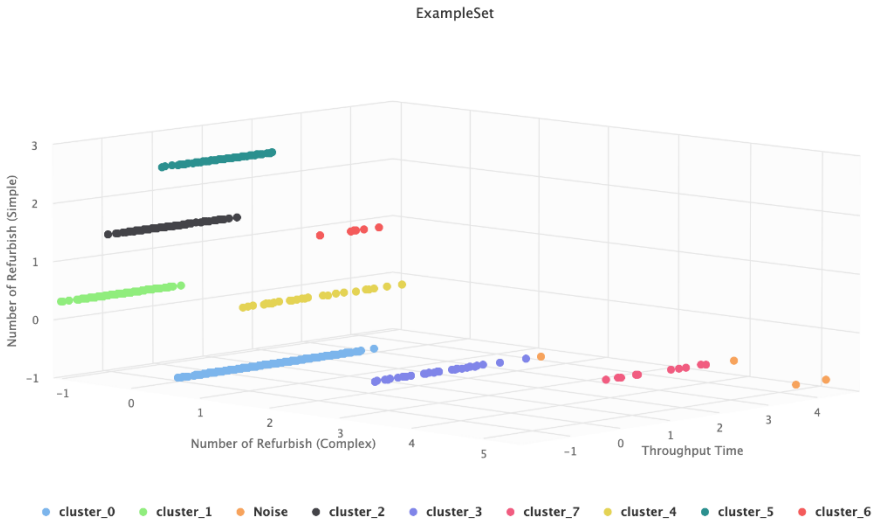


Figure 12: 3D Scatter Visualization for 8 Clusters

As shown in Figure 12, we can see that there are 8 clusters formed in total and a noise cluster has also been formed which are not a part of any cluster.

Cluster Number	**Number of Refurbishments (Simple)**	**Number of Refurbishments (Complex)**
0	0	1
1	1	0
2	2	0
3	0	2
4	1	1
5	3	0
6	2	1
7	0	3

Question 4: Association Rules

a) The Process is shown in Figure 13.

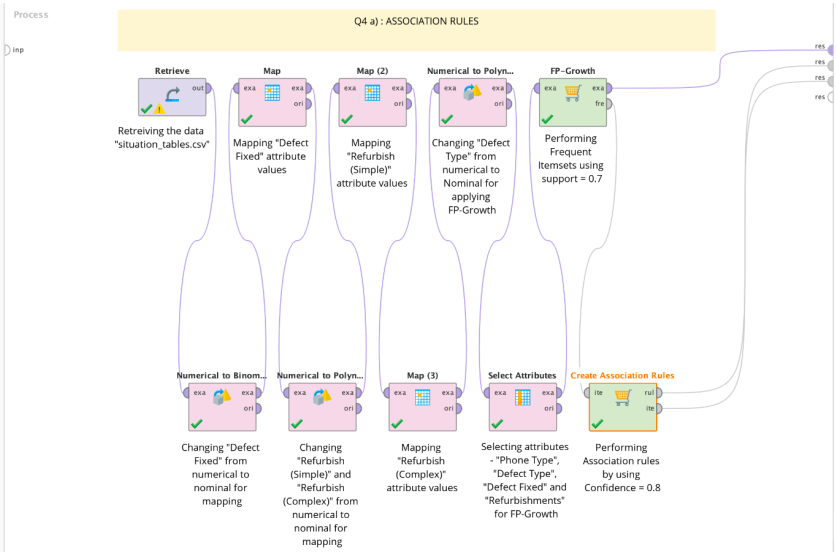


Figure 13: Process for Association Rules

No.	Premises	Conclusion	Support	Confidence	LaPlace	Gain	p-s	Lift	Convic...
22	one complex refurbishment, T2	no simple refurbishment	0.218	0.956	0.992	-0.238	0.090	1.705	10.022
23	one complex refurbishment, T2	Defect fixed, no simple refurbishment	0.218	0.956	0.992	-0.238	0.090	1.705	10.022
24	Defect fixed, one complex refurbishment, T2	no simple refurbishment	0.218	0.956	0.992	-0.238	0.090	1.705	10.022
12	no simple refurbishment, T2	one complex refurbishment	0.218	0.927	0.986	-0.252	0.089	1.685	6.135
13	no simple refurbishment, T2	Defect fixed, one complex refurbishment	0.218	0.927	0.986	-0.252	0.089	1.685	6.135
14	Defect fixed, no simple refurbishment, T2	one complex refurbishment	0.218	0.927	0.986	-0.252	0.089	1.685	6.135
9	no simple refurbishment	one complex refurbishment	0.515	0.917	0.970	-0.607	0.206	1.668	5.454
10	no simple refurbishment	Defect fixed, one complex refurbishment	0.515	0.917	0.970	-0.607	0.206	1.668	5.454
11	Defect fixed, no simple refurbishment	one complex refurbishment	0.515	0.917	0.970	-0.607	0.206	1.668	5.454
18	one complex refurbishment	no simple refurbishment	0.515	0.936	0.977	-0.585	0.206	1.668	6.825
19	one complex refurbishment	Defect fixed, no simple refurbishment	0.515	0.936	0.977	-0.585	0.206	1.668	6.825
20	Defect fixed, one complex refurbishment	no simple refurbishment	0.515	0.936	0.977	-0.585	0.206	1.668	6.825
15	one complex refurbishment, T1	no simple refurbishment	0.163	0.933	0.990	-0.187	0.065	1.663	6.520
16	one complex refurbishment, T1	Defect fixed, no simple refurbishment	0.163	0.933	0.990	-0.187	0.065	1.663	6.520
17	Defect fixed, one complex refurbishment, T1	no simple refurbishment	0.163	0.933	0.990	-0.187	0.065	1.663	6.520
4	no simple refurbishment, T1	one complex refurbishment	0.163	0.905	0.985	-0.198	0.064	1.645	4.714
5	no simple refurbishment, T1	Defect fixed, one complex refurbishment	0.163	0.905	0.985	-0.198	0.064	1.645	4.714
6	Defect fixed, no simple refurbishment, T1	one complex refurbishment	0.163	0.905	0.985	-0.198	0.064	1.645	4.714
27	no simple refurbishment	Defect fixed	0.561	1	1	-0.561	0.019	1.036	∞
28	one complex refurbishment	Defect fixed	0.550	1	1	-0.550	0.019	1.036	∞
29	one simple refurbishment	Defect fixed	0.246	1	1	-0.246	0.008	1.036	∞
30	no simple refurbishment, one complex refurbis...	Defect fixed	0.515	1	1	-0.515	0.018	1.036	∞
31	no simple refurbishment, T3	Defect fixed	0.145	1	1	-0.145	0.005	1.036	∞
32	no simple refurbishment, T1	Defect fixed	0.181	1	1	-0.181	0.006	1.036	∞
33	no simple refurbishment, T2	Defect fixed	0.235	1	1	-0.235	0.008	1.036	∞

Figure 14: Association Rules sorted by lift from highest to lowest

From Figure 14, **The first rule** is -

{one complex refurbishment , T2}⇒ no simple refurbishments

Support - 0.218	Confidence - 0.956	Lift - 1.705

This rule suggests that there is a high likelihood that if one complex refurbishment is performed for T2 phone type, then no simple refurbishments are performed. This says that T2 phone types usually do not go through any number of simple refurbishments after going through just a single complex refurbishment. As the confidence is 0.956, this suggests that 95.6% of the cases didn't need any further simple refurbishment after only 1 complex refurbishment activity for the phone type T2. The support of 0.218 suggests that 21.8% of the cases have one complex refurbishment only done for T2 phone type in their process. The lift is above 1, this says that there is a strong positive relationship between having one complex refurbishment activity for phone type T2 and no simple refurbishments.

From Figure 14, the **Other 2 rules** -
Both the rules have the following values as same -

Support - 0.218	Confidence - 0.956	Lift - 1.705

1. {one complex refurbishment , T2}⇒ no simple refurbishments, Defect Fixed

This rule also indicates that if there are one complex refurbishments performed for the phone type T2, the likelihood of the defect being fixed is high(95.6%) without requiring any simple refurbishments. Although the number of cases with 1 complex refurbishment for T2 is only 21.8% of all cases. This could help the company in cutting down resource costs by not performing unnecessary refurbishments on phone type T2. The lift value above 1 says that there is a strong positive relationship between having one complex refurbishment activity for phone type T2 and defect being fixed without any simple refurbishments.

2. {Defect fixed, one complex refurbishment, T2} ⇒ no simple refurbishments

This rule suggests that out of the 21.8% of all the cases consisting of T2 type of phones have gone through one complex refurbishment and got their defect fixed, 95.6% of them have not required any simple refurbishments. This premise and conclusion are supported by strong positive correlation by lift value greater than 1.

b) For this question, we use the process generated above and add the filter examples operator to filter the Defect not fixed values.

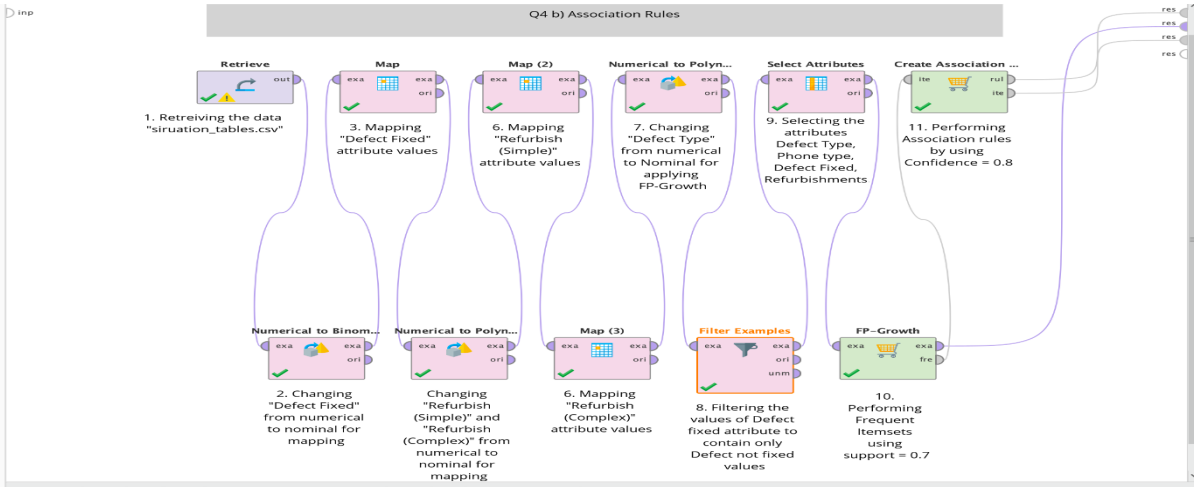


Figure 15: Process for Association Rules with Filter for only Defect not Fixed values.

No.	Premises	Conclusion	Support	Confidence	LaPlace	Gain	p-s	Lift	Convic...
1	Defect not fixed	many simple refurbishment	1	1	1	-1	0	1	?
2	many simple refurbishment	Defect not fixed	1	1	1	-1	0	1	?
3	Defect not fixed	no complex refurbishment	1	1	1	-1	0	1	?
4	no complex refurbishment	Defect not fixed	1	1	1	-1	0	1	?
5	T3	Defect not fixed	0.447	1	1	-0.447	0	1	?
6	T1	Defect not fixed	0.368	1	1	-0.368	0	1	?
7	2	Defect not fixed	0.316	1	1	-0.316	0	1	?
8	4	Defect not fixed	0.289	1	1	-0.289	0	1	?
9	3	Defect not fixed	0.237	1	1	-0.237	0	1	?
10	T2	Defect not fixed	0.184	1	1	-0.184	0	1	?
11	1	Defect not fixed	0.158	1	1	-0.158	0	1	?
12	many simple refurbishment	no complex refurbishment	1	1	1	-1	0	1	?
13	no complex refurbishment	many simple refurbishment	1	1	1	-1	0	1	?
14	T3	many simple refurbishment	0.447	1	1	-0.447	0	1	?
15	T1	many simple refurbishment	0.368	1	1	-0.368	0	1	?
16	2	many simple refurbishment	0.316	1	1	-0.316	0	1	?
17	4	many simple refurbishment	0.289	1	1	-0.289	0	1	?
18	3	many simple refurbishment	0.237	1	1	-0.237	0	1	?
19	T2	many simple refurbishment	0.184	1	1	-0.184	0	1	?
20	1	many simple refurbishment	0.158	1	1	-0.158	0	1	?
21	T3	no complex refurbishment	0.447	1	1	-0.447	0	1	?
22	T1	no complex refurbishment	0.368	1	1	-0.368	0	1	?
23	2	no complex refurbishment	0.316	1	1	-0.316	0	1	?
24	4	no complex refurbishment	0.289	1	1	-0.289	0	1	?
25	3	no complex refurbishment	0.237	1	1	-0.237	0	1	?

Figure 16a: Association Rules with filter

No.	Premises	Conclusion	Support	Confidence	LaPlace	Gain	p-s	Lift ↓	Convic...
2	many simple refurbishment	Defect not fixed	1	1	1	-1	0	1	?
4	no complex refurbishment	Defect not fixed	1	1	1	-1	0	1	?
5	T3	Defect not fixed	0.447	1	1	-0.447	0	1	?
6	T1	Defect not fixed	0.368	1	1	-0.368	0	1	?
7	2	Defect not fixed	0.316	1	1	-0.316	0	1	?
8	4	Defect not fixed	0.289	1	1	-0.289	0	1	?
9	3	Defect not fixed	0.237	1	1	-0.237	0	1	?
10	T2	Defect not fixed	0.184	1	1	-0.184	0	1	?
11	1	Defect not fixed	0.158	1	1	-0.158	0	1	?
29	many simple refurbishment	Defect not fixed, no complex refurbishment	1	1	1	-1	0	1	?
31	no complex refurbishment	Defect not fixed, many simple refurbishment	1	1	1	-1	0	1	?
33	many simple refurbishment, no complex refur...	Defect not fixed	1	1	1	-1	0	1	?
34	T3	Defect not fixed, many simple refurbishment	0.447	1	1	-0.447	0	1	?
36	many simple refurbishment, T3	Defect not fixed	0.447	1	1	-0.447	0	1	?
37	T1	Defect not fixed, many simple refurbishment	0.368	1	1	-0.368	0	1	?
39	many simple refurbishment, T1	Defect not fixed	0.368	1	1	-0.368	0	1	?
40	2	Defect not fixed, many simple refurbishment	0.316	1	1	-0.316	0	1	?
42	many simple refurbishment, 2	Defect not fixed	0.316	1	1	-0.316	0	1	?
43	4	Defect not fixed, many simple refurbishment	0.289	1	1	-0.289	0	1	?
45	many simple refurbishment, 4	Defect not fixed	0.289	1	1	-0.289	0	1	?
46	3	Defect not fixed, many simple refurbishment	0.237	1	1	-0.237	0	1	?
48	many simple refurbishment, 3	Defect not fixed	0.237	1	1	-0.237	0	1	?
49	T2	Defect not fixed, many simple refurbishment	0.184	1	1	-0.184	0	1	?
51	many simple refurbishment, T2	Defect not fixed	0.184	1	1	-0.184	0	1	?
52	1	Defect not fixed, many simple refurbishment	0.158	1	1	-0.158	0	1	?

Figure 16b: Association Rules with Defect not Fixed as conclusion

Here (Figure 13), we are using the filter examples operator to filter out the Defect Fixed values and then we are selecting the required attributes and plotting FP growth with 0.7 support and then Create Association rules with 0.8 confidence.

Observations:

Since, the support for the first 2 rules is 1, it says 100% of instances in the filtered example set are cases with “many simple refurbishments”,”Defect not Fixed” and “no complex refurbishment” as the values in their respective attributes.

The confidence is also 1, stating, for 100% of these cases, if defect is not fixed, then many simple refurbishments are performed on it and no complex refurbishment performed as per the second rule.

The filtered example set consists of 38 examples, out of which all of them have “many simple refurbishments”,”no complex refurbishment” and “Defect not fixed” as the values in their respective attributes. This tells us that If the defect is not fixed, then those cases have gone through many simple refurbishments and no complex refurbishments. The company can save resources on this process if they would not perform more than 2 simple refurbishments and instead complex refurbishment is performed. However, after viewing the lift value we realize that is not true.

The lift being 1 implies that there is no association or correlation between the premise and conclusion in this case because all our examples contain only Defect not Fixed as their value. The presence of “no complex refurbishment” or “many simple refurbishment” does not affect the defect not being fixed.

Since all our rules have lift value 1, we can disregard these rules although the confidence is high.

Question 5: Process Exploration

a) (i). The following activities (Table 2) are contained in the log. This solution is derived using "Directly-Follows activity graph" visualization on ProM" (Figure 17a).The nodes in the graph state the activities in the log.

1. Register Arrival	5. Restart Refurbish
2. Detect Required Improvement	6. Refurbish (Complex)
3. Refurbish (Simple)	7. Notify Sales
4. Assess Product Quality	8. Document

Table 2: List of Activities

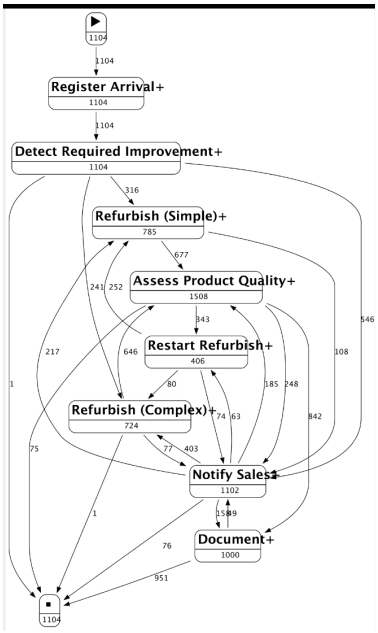


Figure 17a: Directly-Follows activity Graph Visualization

Figure 17b: Explore Event Log Visualization

(ii). The number of trace variants is 64. This solution is derived using "Explore Event Log" visualization in Prom (Figure 17b) under variants in the visualization after selecting all the traces.

(iii). The following attributes (Table 3) are featured with each event -

Concept:name	durationInMinutes	Org:resource	Time:timestamp	defectType	phoneType	defectFixed
(ALL) Register Arrival,Detect Required Improvement,Notify Sales,Refurbish (simple) simple ,Refurbish (complex), Assess Product Quality, Restart Refurbish,Document	**(ALL)** Register Arrival,Detect Required Improvement,Notify Sales,Refurbish (simple) simple ,Refurbish (complex), Assess Product Quality, Restart Refurbish,Document	**(ALL)** Register Arrival,Detect Required Improvement,Notify Sales,Refurbish (simple) simple ,Refurbish (complex), Assess Product Quality, Restart Refurbish,Document	**(ALL)** Register Arrival,Detect Required Improvement,Notify Sales,Refurbish (simple) simple ,Refurbish (complex), Assess Product Quality, Restart Refurbish,Document	Detect Required Improvement	Detect Required Improvement	Assess Product Quality, Document

Table 3: Attributes associated with Events

This solution is derived using "Explore Event Log" visualization in Prom (Figure 17b) by selecting each event in the trace and visualizing the attributes associated with each activity.

Description: Concept:name is the name of the activity, durationInMinutes, gives the time taken to finish the activity while Org: Resource states the employee or resource that performs the activity, and Time:timestamp is the timestamp when the activity happened. These attributes are associated with all the events because these are the basic attributes of any event table. However, defectType and phoneType which state the type of defect and phone are only associated with Detect Required Improvement because, identifying them is necessary only by the said event. Finally, defectFixed is associated only with Assess Product Quality and Document because the final update is only necessary by them to assess if it's fixed or not and to document the last update into the system before the process ends.

(iv). There are a total of 5 end activities -

1.. Document	2. Notify Sales	3. Detect Required Improvement	4. Refurbish (Complex)	5. Assess Product Quality

The most frequent end event is **Document which happens 951** times as an end event out of the 1104 traces.This solution is derived using "Directly follows activity graph" visualization on ProM (Figure 17a). The arrows leading to the final marking are traced back to activities to find the end activities.

v). The activity which is most frequently performed after Detect Required Improvement is **Notify Sales by 546 times.** This solution is derived using "Directly follows activity graph" visualization on ProM (Figure 17a). The arrow and frequency following the "Detect Required Improvement" activity is followed by "Notify Sales".

b) (i). The trace variant (Figure 18) < Register Arrival , Detect Required Improvement > could be noise. Because, there is only one case in which after "Detect Required Improvement" no activity is performed instead of "Refurbish (Complex)" or "Refurbish (Simple)" or "Notify Sales". Therefore, the process is incomplete in this variant.

Figure 18: The noise trace

The solution is derived using the "Directly follows activity graph" visualisation on ProM (Figure 17a). The arrow and frequency following the "Detect Required Improvement" activity is followed to end activity. Also, using "Explore Event Log" on ProM (Figure 17b) by sorting the traces according to frequency, we can see the traces with least frequency.

(ii). For this solution (Figure 19a and Figure 19b), we used the "Filter Event Log" plugin (Figure 20) and added 2 filters.

1. Filter by frequency - We set the lower bound of the threshold to 10 to filter out traces with frequency less than 10.
2. Filter by end event - We selected the end event as "Document" to filter out traces that end with Document as end event as its the most frequent end activity.

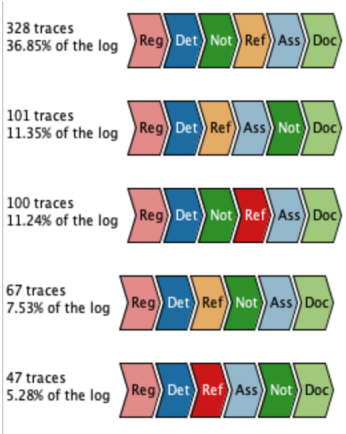


Figure 19a: The five most frequent trace variants

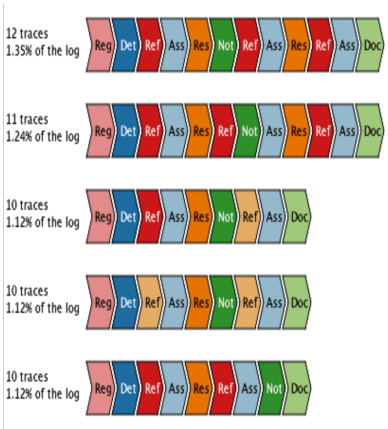


Figure 19b: The five least frequent trace variants

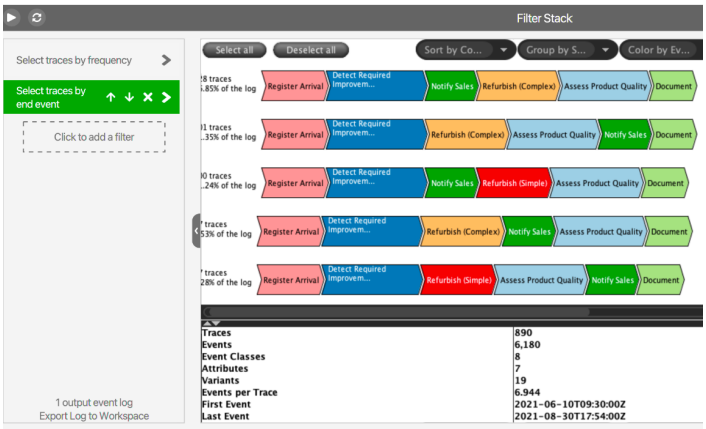


Figure 20: The Filter Event Log plugin

c) (i). From Monday to Friday, the company starts its first event at 9:30, however this is performed by the System. From 10:00, the employees start working till 20:00 with a break from 14:00 to 15:00.
Therefore the business hours are - **Monday to Friday from 10:00 to 20:00 with 14:00 to 15:00** being the break period. We use the following settings on the Dotted Chart (Figure 21a)to derive our answer.

X-Axis Attribute	Y-Axis Attribute	Color Attribute
E: time:timeSinceDayStart	E: time:dayOfWeek	E: time:dayOfWeek

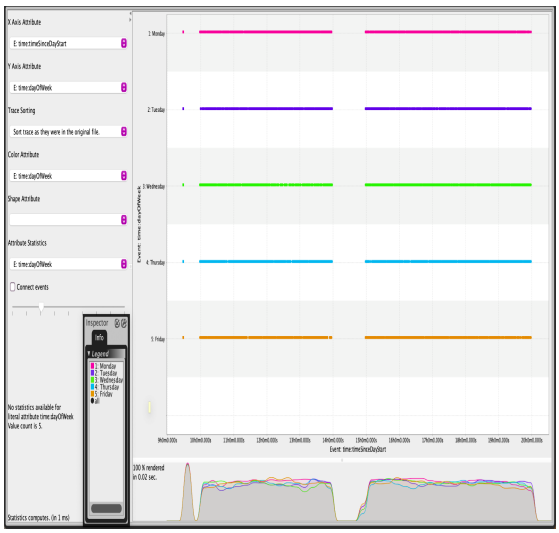


Figure 21a: Business Hours Visualization

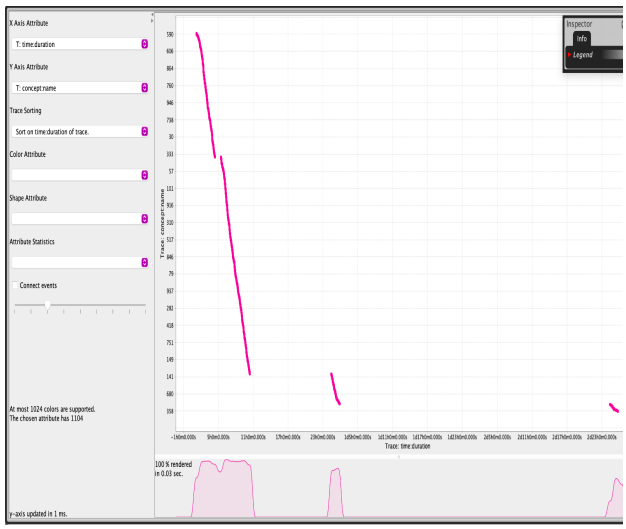


Figure 21b: Fraction of cases Visualization

(ii) 993 traces out of 1104 traces finished within the first 12 hours. Therefore, nearly **90%** of the traces finished within the 12 hours after the trace started. We use the following settings on the Dotted Chart (Figure 21b)to derive our answer. After sorting by duration, when we zoom in to view the trace number, we can see that trace number 994 onwards, the duration is more than 23 hours.

X-Axis Attribute	Y-Axis Attribute	Trace sorting
T: time:duration	T: concept:name	Sort on time: duration of trace

(iii) The activities each employee are responsible are as follows:

- Angela, Carlos, Jisele, Joaquin, Jorge and Julio - Asses Product Quality and Detect Required Improvement
- Daniela, Marcelo and Mario - Refurbish (Simple)
- Deigo, Miguel and Susana - Refurbish (Complex)
- System - Register Arrival, Restart Refurbish, Notify Sales and Document.

We use the following settings on the Dotted Chart (Figure 22a)to derive our answer.

X-Axis Attribute	Y-Axis Attribute	Color Attribute
E: org:resource	T: concept:name	E: concept:name

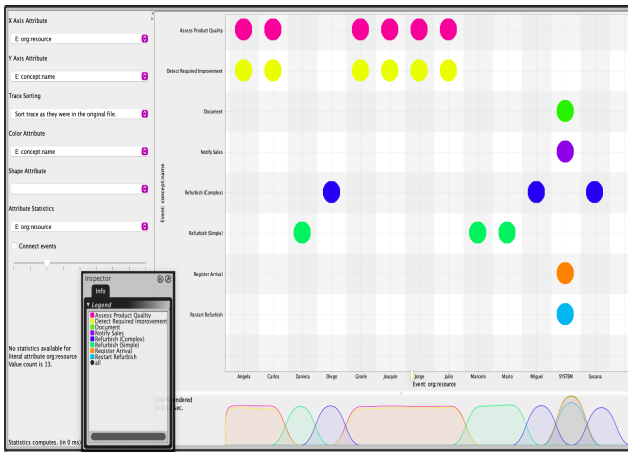


Figure 22a: Each employee activities Visualization

Figure 22b: Most time taking Activity Visualization

(iv). The activity “Refurbish (Complex)” seems to have a maximum duration of 50 minutes. It, on average takes 30 minutes while the other activities are a max of 20 minutes to finish. We use the following settings on the Dotted Chart (Figure 22b)to derive our answer.

X-Axis Attribute	Y-Axis Attribute	Color Attribute
E: durationInMinutes	E: concept:name	E: concept:name

(v). From 1st September till 4th September, the activity Document is not performed. We use the following settings on the Dotted Chart (Figure 23a)to derive our answer.

X-Axis Attribute	Y-Axis Attribute	Color Attribute
E: time:timestamp	E: concept:name	E: time:dayOfWeek

(vi). The employee that takes the least time to finish “Refurbish(Simple)” is Mario while the one that takes the most time is Daniela.We use the following settings on the Dotted Chart (Figure 23b)to derive our answer.

X-Axis Attribute	Y-Axis Attribute	Color Attribute
E: concept:name	E: durationInMinutes	E: org:resource

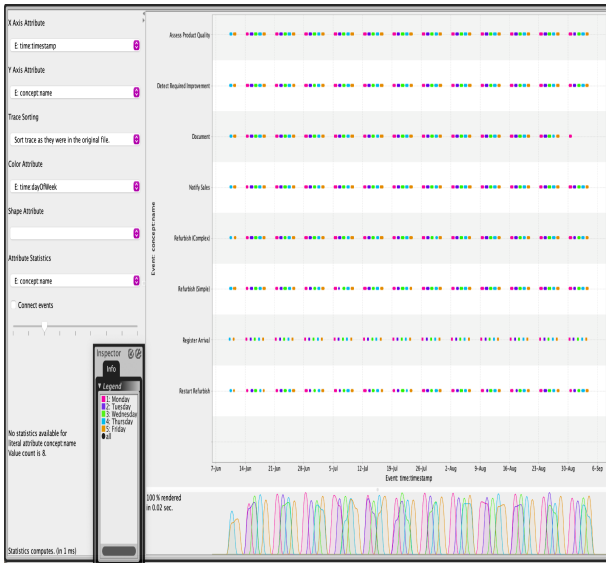


Figure 23a: Long term behavior Visualization

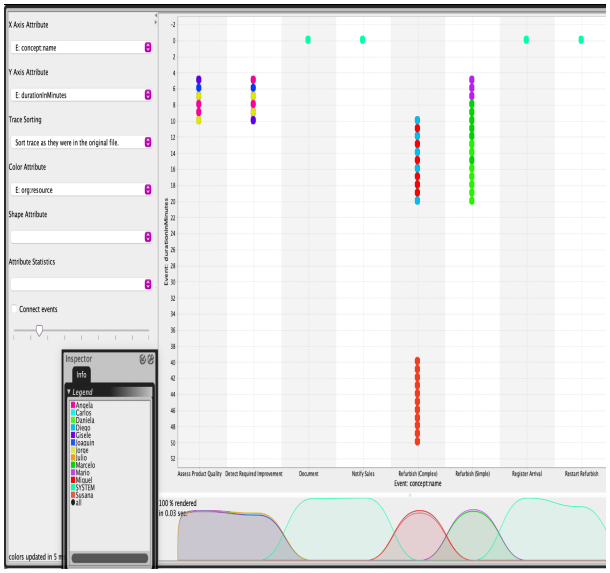


Figure 23b: Employee for Refurbish(Simple) Visualization

d)

(i) The filtered log after plotting the dotted chart for the duration of traces. First we use the "Filter log by simple heuristics" to filter out the activity "Restart Arrival" and then we use the following settings on the Dotted Chart (Figure 24)to derive our answer.

X-Axis Attribute	Y-Axis Attribute	Trace sorting	Color Attribute
T: concept:name	T: time:duration	Sort on time: duration of trace	E: concept:name

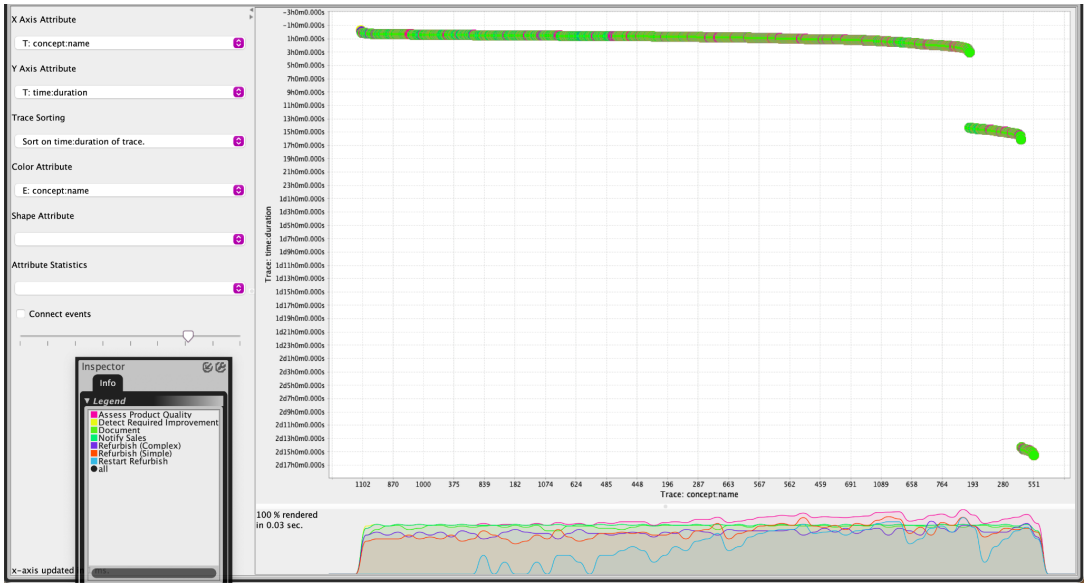
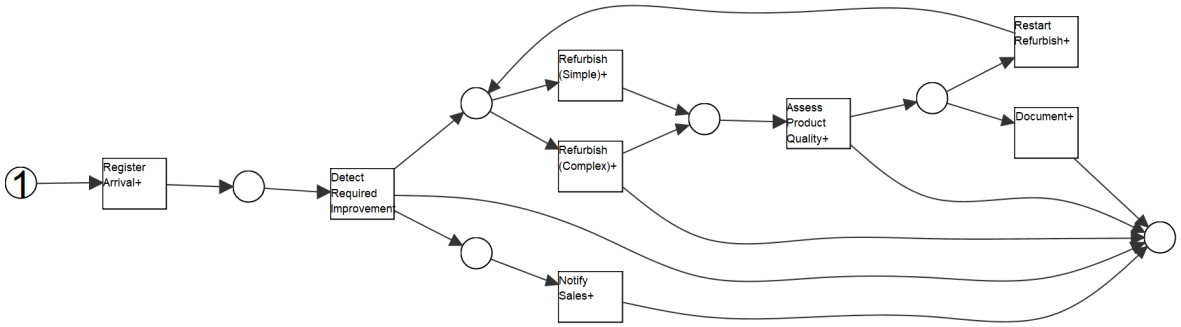


Figure 24: Filtered Dotted Chart for trace duration

(ii). The predominant time intervals(for 90% of the traces) for trace durations are between 30 minutes to 3 hours after a trace starts.These traces are finished in one shift, probably from 10:00 to 14:00 or 15:00 to 19:00 before the business hours end. Nearly 8% of the remaining traces take from 14 and half hours to 16 hours. These could be the traces that started in the last one hour of the business day and are on hold for the whole 14 hours of non-business hours till the next day 10:00 which leads to the minimum 14 hour duration. Next, nearly 2% of the traces take 2 days 14 hours to 2 days 16 hours as these traces might have been started at the last hour of a friday, therefore on hold the week end till the business hours open again on monday at 10:00. This is how we can relate it to the business hours in c(i).

Question 6: Alpha Miner



6.a.i. α(L) is a workflow net because it fulfills the requirements of a workflow net: it has a source place for which there is no input, it has a sink place from which there is no output and every node and place are reachable on a path from the source to the sink.

6.a.ii. The state space of α(L) is infinite. This is because there can be an infinite loop with Refurbish (Complex) → Assess Product Quality → Restart Refurbish → Refurbish (Complex). Not only is there a loop but also this loop adds 2 new tokens to the sink. Because there is a way to generate new tokens in a loop infinitely, the state space is infinite.

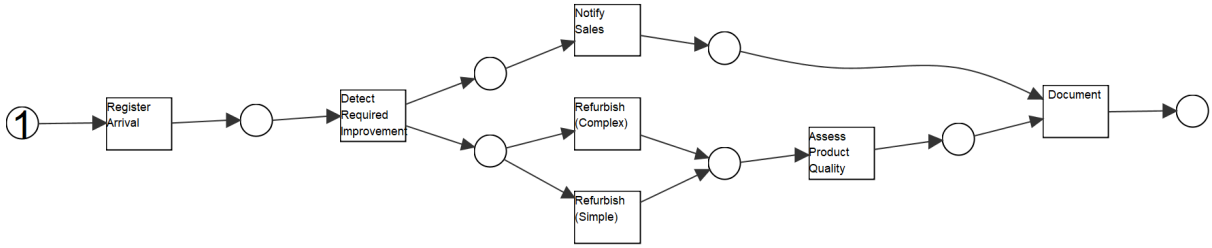


Figure 26: Resulting Petri-net on L'

6.b.i.

Trace variants of L':

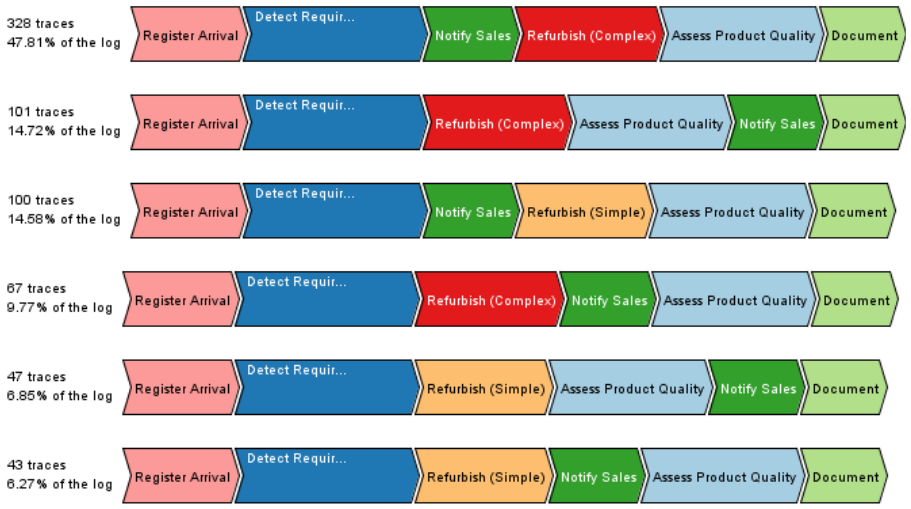


Figure 27: Filtering L to L'

Trace variants of α(L'):

R := Register Arrival
I := Detect Required Improvement
N := Notify Sales
C := Refurbish (Complex)
S := Refurbish (Simple)
A := Assess Product Quality
D := Document

[<R,I,N,C,A,D>, <R,I,N,S,A,D>, <R,I,C,N,A,D>, <R,I,S,N,A,D>, <R,I,C,A,N,D>, <R,I,S,A,N,D>]

6.b.ii. Fitness and precision are both 1. Model can only reproduce traces in the log (precision) and it can produce all of them (fitness). We can see from figure 26 and figure 27 that all of our trace variants can be replayed perfectly by the model. When we look at the formula for Precision, our 'FP' is also equal to 0 because our model accepts the 6 trace variants which are there in the filtered log.

6.c.i.

	A	C	D	I	N	R	S
A	#	←	→	#	→	#	←
C	→	#	#	←	←	#	#
D	←	#	#	#	←	#	#
I	#	→	#	#	→	←	#
N	←	→	→	←	#	#	→
R	#	#	#	→	#	#	#
S	→	#	#	#	←	#	#

Table 4: Footprint Matrix

6.c.ii.

A	{A}	{A}	{C}	{I}	{I}	{N}	{N}	{N}	{R}	{S}
B	{D}	{N}	{A}	{C}	{N}	{C}	{D}	{S}	{I}	{A}

XL3 = {({R}, {I}), ({I}, {N}), ({I}, {C}), ({N}, {C}), ({N}, {D}), ({N}, {S}), ({C}, {A}), ({A}, {N}), ({A}, {D}), ({S}, {A}), ({N}, {C, D}), ({N}, {C, S}), ({N}, {D, S}), ({N}, {C, D, S}), ({I, A}, {N}), ({C, S}, {A})}

6.c.iii.

A	{A}	~~{A}~~	~~{C}~~	{I}	~~{I}~~	~~{N}~~	~~{N}~~	~~{N}~~	{R}	~~{S}~~	{N}	{S,C}	{A,I}
B	{D}	~~{N}~~	~~{A}~~	{C}	~~{N}~~	~~{C}~~	~~{D}~~	~~{S}~~	{I}	~~{A}~~	{C,D,S}	{A}	{N}

YL3 = {({R}, {I}), ({I}, {C}), ({A}, {D}), ({N}, {C, D, S}), ({I, A}, {N}), ({C, S}, {A})}

6.c.iv.

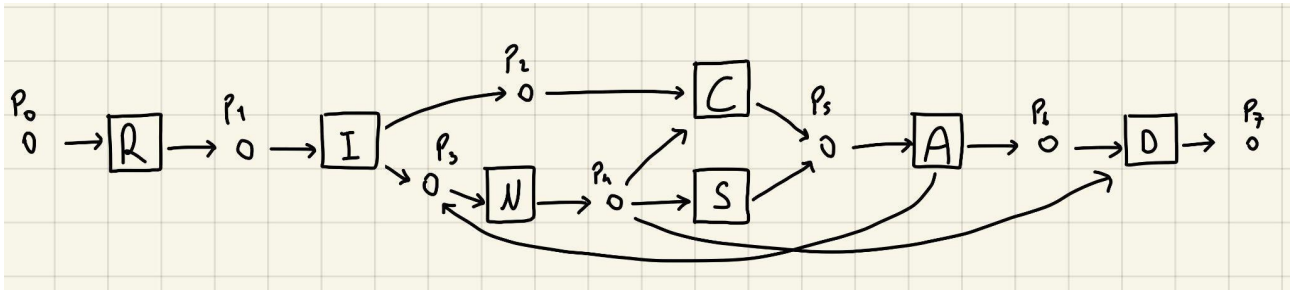
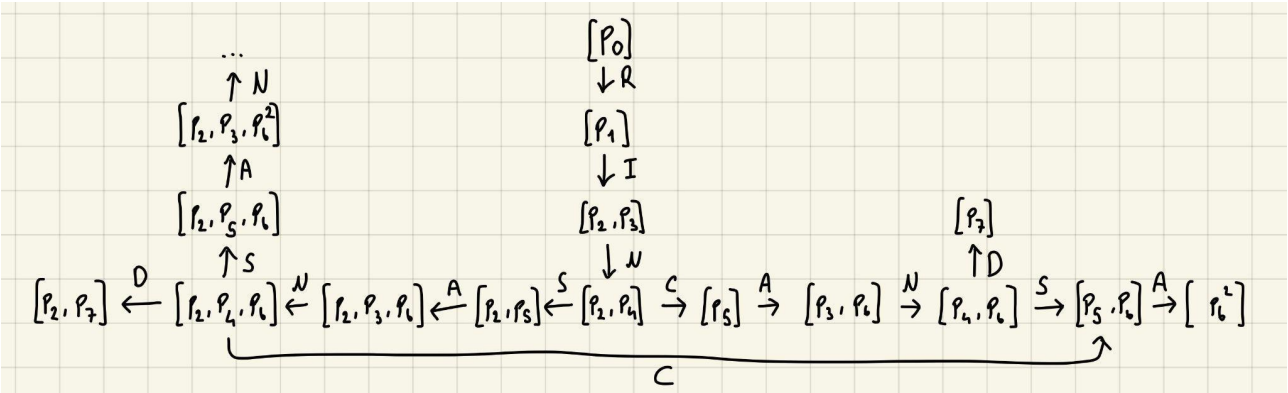


Figure 28: Alpha Algorithm Output

6.d.i.



Safeness: Violated. There are states where a place has more than one token.
Option to complete: Violated. There are states from which the sink cannot be reached.
Proper completion: Violated. There is a state where a place has a token when the sink also has one.
Freedom of dead transitions: Not violated. All transitions can be fired.
Therefore it is not a sound net.

6.d.ii. <R,I,N,C,A,N,D> is an accepted trace. None of the trace variants in L_3 are accepted by $\alpha(L_3)$.

6.e.i. Restart Refurbish → Refurbish (Complex) holds in $\alpha(L_1)$ but not in $\alpha(L_2)$.
e (ii) If L '2 be a log that has the same footprint matrix as L2 along with Restart Refurbish -> Refurbish (Complex), then YES, $\alpha(L'2) = \alpha(L1)$.
Explanation - $\alpha(L2)$ and $\alpha(L1)$ petri nets are almost identical with the traces that they replay except that Refurbish complex or Refurbish simple cannot happen after Restart Refurbish. The other difference they hold is in the Y pairs set.
In petri net α(L1):
Sets inside Y in this case = {({Detect Required Improvement, Restart Refurbish}, {Refurbish (Complex), Refurbish (Simple)})}
Trace sequence = Register Arrival -> Detect Required Improvement, -> Refurbish (Complex) XOR Refurbish (Simple) -> Access Product Quality -> Restart Refurbish> [Refurbish (Complex) XOR Refurbish (Simple)] any number of times
In petri net α(L2):
Sets inside Y in this case = {({Detect Required Improvement}, {Refurbish (Complex), Refurbish (Simple)}),({Detect Required Improvement, Restart Refurbish}, {Refurbish (Simple)})}
Trace sequence = Register Arrival -> Detect Required Improvement, -> Refurbish (Complex) XOR Refurbish (Simple) -> Access Product Quality -> Restart Refurbish> deadlock
If we add , Restart Refurbish -> Refurbish (Complex) , then deadlock will be cleared with tokens in the input places for Refurbish (Simple) and also Refurbish (Complex) can happen any number of times.
In petri net α(L'2):
Sets inside Y in this case ={({Detect Required Improvement, Restart Refurbish}, {Refurbish (Complex), Refurbish (Simple)})}
Trace sequence = Register Arrival -> Detect Required Improvement, -> Refurbish (Complex) XOR Refurbish (Simple) -> Access Product Quality -> Restart Refurbish> [Refurbish (Complex) XOR Refurbish (Simple)] any number of times

Therefore we get exactly the same pairs for Y and other sets are also not affected by this and same traces can be replayed by both $\alpha(L1)$ and $\alpha(L'2)$

Question 7: Heuristic Miner

a) The frequency slider determines the threshold for including dependencies in the generated dependency graph. When the frequency slider is set to 0.2, it means that only dependencies with a frequency greater than or equal to 20% of the total number of traces in the event log will be considered. In this case (Figure 29a and Figure 29b), the dependency between Detect Required Improvement and Notify Sales, along with Assess Product Quality with Notify Sales and Restart Refurbish are filtered out as they are less frequent dependencies when threshold is 0.5. Therefore, by adjusting the frequency slider, we can control the level of detail and the number of dependencies shown in the graph, allowing us to focus on the most relevant and significant relationships in our process.

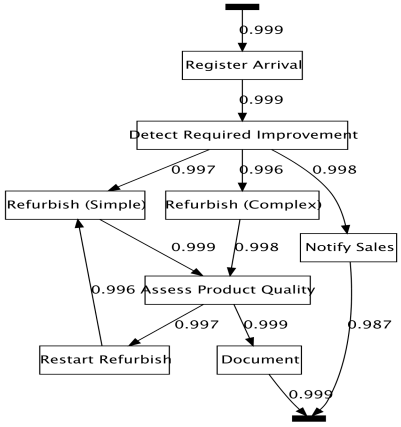


Figure 29a) Dependency graph with frequency = 0.2

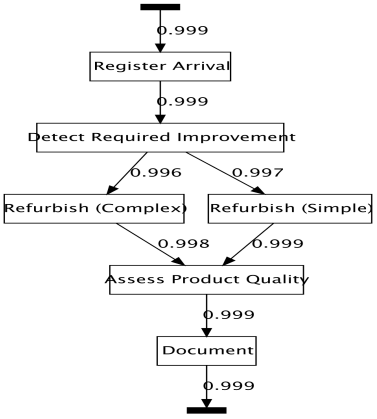


Figure 29b) Dependency graph with frequency = 0.5

b) The weights displayed on the arcs are the dependency measures between the two events. Dependency measures show how certain we are that a dependency between those two events exist. Let > be the directly follows relation between two events where A > B means that A is directly followed by B. |A > B| is the number of times this relation happens. Dependency measure is calculated with

$$A \Rightarrow B = (|A > B| - |B > A|)/(|A > B| + |B > A| + 1)$$

which means we believe there is a higher chance of B depending on A if A is often followed by B and B is rarely followed by A.
let I = "Detect Required Improvements", N = "Notify Sales", A = "Assess Product Quality"
When the value is close to 0, it indicates that the two activities in the connection are happening at the same time or concurrently. If one activity consistently occurs before another activity, but not the other way around, the dependency measure approaches 1. The arc from I to N has a weight of 0.998 because there are 546 traces where I is directly followed by N and 0 traces where N is directly followed by I. (546 - 0)/(546 + 0 + 1)≈ 0.998. Which implies that there is a high chance that N is dependent on I. While the connection between A and N has a value of 0.145. This suggests that the second connection is more concurrent over the first connection.

c) The total number of traces in which “Refurbish (Complex)” was needed are 618 out of 1104 traces. Thus accounting for nearly 58% of the traces. While, “Refurbish (Simple)” was needed in 445 traces, accounting for nearly 42%. When the dependency slider is set to 0, we can also see that, together, they were needed in 39 traces and Refurbish (Complex) alone in 1 trace.

Binding	Absolute Frequency	Relative Frequency
1. Notify Sales, Refurbish (Complex)	618	58%
2. Notify Sales, Refurbish (Simple)	445	42%

In the event log, the total number of times, “Refurbish (Complex)” happened is 723 times while “Refurbish (Simple)” happened is 783 times.

d) No, trace σ1 is not valid on this C-net. This is due to the pending obligation – {(Refurbish (Simple),Assess Product Quality)}(Table 4).

Input Binding	Activity	Output Binding	State / Pending Obl.
∅	*start*	{Register Arrival}	{(*start*, Register Arrival)}
{*start*}	Register Arrival	{Detect Required Improvement}	{(Register Arrival,Detect Required Improvement)}
{Register Arrival}	Detect Required Improvement	{Refurbish (Complex), Refurbish (Simple), Notify Sales}	{(Detect Required Improvement,Refurbish (Complex)), (Detect Required Improvement,Refurbish (Simple)), (Detect Required Improvement, Notify Sales)}
{Detect Required Improvement}	Refurbish (Complex)	{Assess Product Quality}	{(Refurbish (Complex),Assess Product Quality), (Detect Required Improvement,Refurbish (Simple)), (Detect Required Improvement, Notify Sales)}
{Detect Required Improvement}	Refurbish (Simple)	{Assess Product Quality}	{(Refurbish (Complex),Assess Product Quality), (Refurbish (Simple),Assess Product Quality), (Detect Required Improvement, Notify Sales)}
{Refurbish (Complex)}	Assess Product Quality	{Document}	{(Refurbish (Simple),Assess Product Quality), (Assess Product Quality, Document), (Detect Required Improvement, Notify Sales)}
{Detect Required Improvement}	Notify Sales	{end}	{(Refurbish (Simple),Assess Product Quality), (Assess Product Quality, Document), (Notify Sales, end)}
{Assess Product Quality}	Document	{end}	{(Refurbish (Simple),Assess Product Quality), (Document,end), (Notify Sales, end)}
{Notify Sales, Document}	end	∅	{(Refurbish (Simple),Assess Product Quality)}

Table 5: Pending Obligation for trace σ1

e)The binding slider should be set to 0.2. Setting the binding slider to 0.2 ensures that the result is documented and Notify Sales is performed before ending the process. Also, either Refurbish(Complex) or Refurbish(Simple) is performed, not both when bindings threshold is 0.2.

f) Refurbish(Complex) occurs 618 times along with Notify Sales while Refurbish(Simple) occurs 445 times. There, as visualized from the output bindings of the "Detect Required Improvements" , Therefore, when the sales team is notified about a product, it is more likely a complex refurbishment is required.

g) Infrequent behavior can be excluded in heuristic miner by selectively increasing the frequency(direct succession), dependency and binding threshold. The less frequent trace variants are filtered using the frequency slider, while the less frequent dependencies between activities are filtered by the dependency slider and less frequent bindings between activities are filtered using the bindings filter. While in Alpha miner infrequent behavior can only be excluded by filtering the event log of infrequent traces beforehand.

h) (i) According to the Petri netα(L1), either Refurbish (Simple) or Refurbish (Complex) can be performed after Restart Refurbish but not both and the loop can be performed infinitely.
In Petri net α(L2), after Restart Refurbish, neither Refurbish (Simple) or Refurbish (Complex) can happen because Refurbish(Simple) needs a token in 2 places, leading to a deadlock. Therefore, only Restart(Complex) can happen once in each trace before Restart Refurbish.
The petri net(Figure below) we plotted shows that Refurbish (Simple) can happen as a loop for infinite times if Restart Refurbish happens irrespective of whether Refurbish (Complex) already happened in the trace or not.

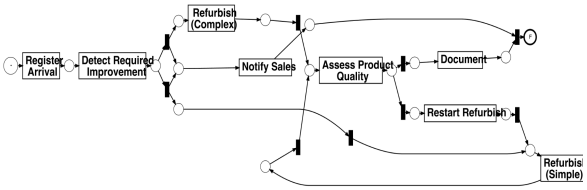


Figure 30: Petri net from Heuristic miner

ii) Heuristic Miner allows execution of activities any number of times given their dependencies are satisfied while alpha miner is sequential where order of activities is strictly followed. The silent transitions in Heuristic Miner are used to execute activities given that their dependencies and bindings are satisfied.