



DIPARTIMENTO
MATEMATICA

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

PROCESS MINING

A.Y. 2023/2024
Project Report

Q1. Design and Validation for a Process Model

The aim of the first question was to provide a Petri net of the process based on the provided description. Here I report the draw of the obtained Petri net.

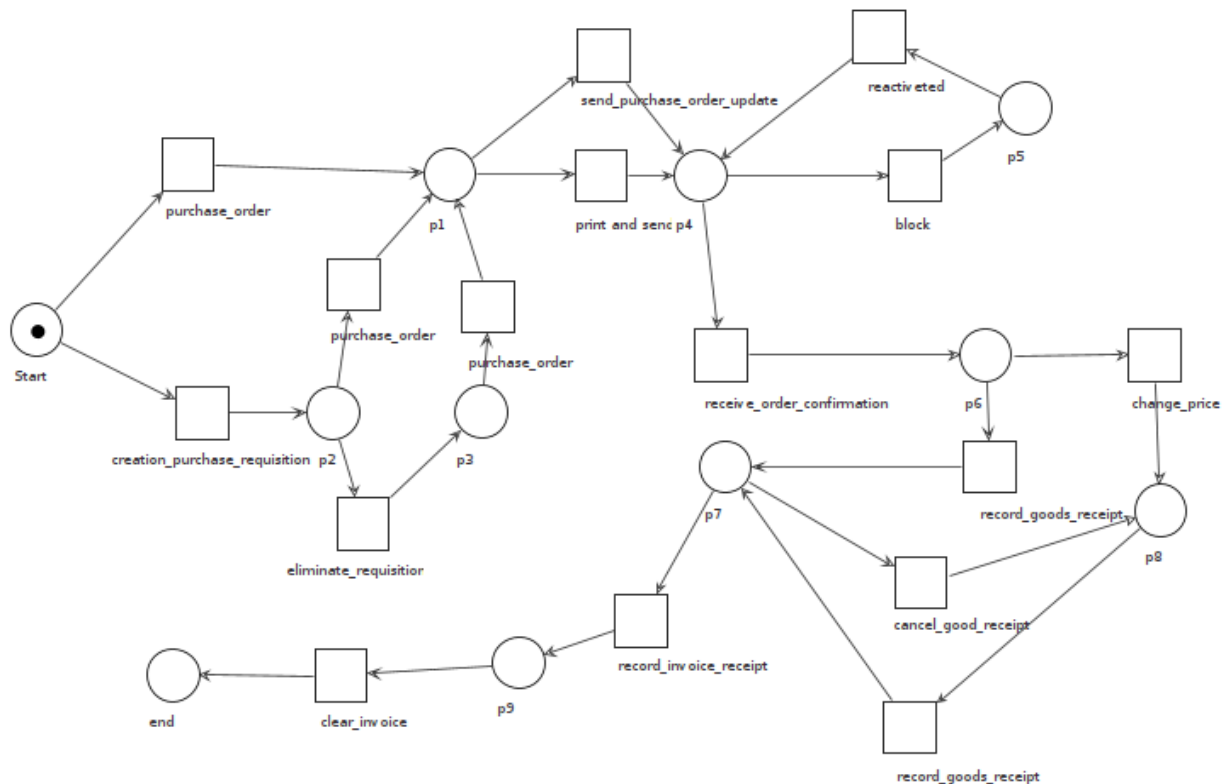


Fig.1 Petri net of the process obtained using the WoPeD software

The Petri net starts from the place *Start*. According to the description of the process, we can have two different transitions according to the type of process. In order to be able to respect this differentiation, right at the beginning there is a XOR-split: in this way small processes can directly go to the transition *purchase_order*, while the other processes can go to *creation_purchase_requisition*. Following this last path, we find another XOR-split. In this case we can choose between *purchase_order* transition or *eliminate_requisition* and then *purchase_order*. In this way we are able to satisfy the request to eliminate the purchase requisition if it is deemed unnecessary.

Once again here we have a XOR-split: we can modify the purchase (*send_purchase_order_update*) or print and send it (*print_and_send*).

At this point we encounter a loop. This happens because, according to the process description, the purchase order can be repeatedly blocked (*block*) and reactivated (*reactivated*). Once we exit the loop, we encounter the transition *receive_order_confirmation*. At this point, two different paths can be chosen. Let's analyze the one starting with *record_goods_receipt*.

This transition, if the goods contain any sort of problem, will be followed by *cancel_good_receipt*, followed again by *record_goods_receipt*. In order to generalize

the process, I decided to make these two transitions as a loop. In this way, the purchase can be eliminated and recreated an infinite number of times. If the goods don't contain any error or after they have been substituted, it encounters the transition *record_invoice_receipt*.

Going back to *receive_order_confirmation*, the other path that we can follow is given by *change_price*. At this point the path encounters the loop made of deleting and recording the receipt goods transitions. Once we exit the loop, we are again on the same path as the other branch. After firing *record_invoice_receipt* there will be *clear_invoice* transition which leads to the final place.

Before going on, I checked for the soundness of the net using the tool provided by the WoPeD software. Since the net is sound, as shown by the image, I proceed calculating the value of fitness and precision.

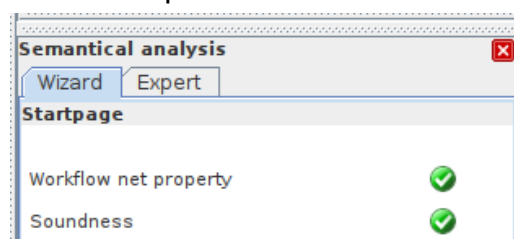


Fig.2 Petri net soundness

The drawn Petri net is used to check the conformance using alignments against the event log.

Before doing this I imported the file *purchase_order.xes* in ProM and tried to filter out the incomplete events. Since they were all complete I filtered out those events that were not starting with *creation_purchase_requisition* or *purchase_order* and ending with *clear_invoice*.

This is more clear if we take a look at the result produced by the Pm4py. As we can see there are multiple initial states and multiple final states. Since, in order to be able to process a purchase, at least we have to create it, I filtered out all those activities that were not starting with *creation_purchase_requisition* or *purchase_order*. I proceeded in the same way and filtered out all the events that were not ending with *clear_invoice*.

Start activities: {'Create Purchase Requisition': 7280, 'Create Purchase Order': 2663, 'Send Purchase Order Update': 24, 'Vendor Creates Invoice': 33}

End activities: {'Clear Invoice': 8704, 'Record Goods Receipt': 704, 'Vendor Creates Invoice': 37, 'Delete Purchase Requisition': 42, 'Change Price': 372, 'Send Purchase Order Update': 32, 'Receive Order Confirmation': 69, 'Block Purchase Order': 3, 'Change Quantity': 20, 'Reactivate Purchase Order': 3, 'Cancel Goods Receipt': 3, 'Delete Purchase Order': 6, 'Change Vendor': 1, 'Change PR Approval': 3, 'Adjustment Charge': 1}

Fig. 3: Dictionaries of starting and ending activities

So, as we can see from the image, at the beginning we had 10000 process instances, after the filtering we are left with 8658.

Log Summary	Log Summary
Total number of process instances: 10000	Total number of process instances: 8658
Total number of events: 75677	Total number of events: 63913

Fig.3 Number of process instances before and after the filtering

At this point we consider the filtered log file as an input file, along with the drawn Petri net, for the tool *“Replay a Log on Petri Net for Conformance Analysis”*. The tool asks to map the transitions of the provided Petri net to the Event Classes. In this case, the only mapping that we have to do, regards *purchase_order* which has to be mapped to *Create Purchase Order + complete*. At this point the visualization can be changed to *“Project alignment to log”*, which shows that, on average, the trace fitness is equal to 0.8 as asked by the problem.

ALIGNMENT STATISTICS			
Trace Fitness			
Average/case	0,80	Precision : 0,98327 Generalization : 0,99999	
Max.	1		
Min.	0,09		
Std. Deviation	0,12		
#Cases with value...	809		

Fig.4 Values of fitness, precision and generalization

To check the precision it has been used the tool *“Measure precision/generalization”* which takes as input the filter file, the designed Petri net and the result produced by the *“Replay a Log on Petri Net for Conformance Analysis”* tool. As we can see the value of precision is greater than 0.9 as requested by the problem.

Q2. Process Discovery

The aim of the second question is to mine the process using different algorithms.

The algorithms I decided to use are Inductive Visual Miner and iDHM.

All the images of the Petri net are provided in the .zip file.

2.1 Inductive Visual Miner

The first Petri net is obtained by using the “*Inductive Visual Miner*” plugin. As input I used the complete log file. Leaving all the default parameters I noticed that the model was too big and complicated. To be consistent, I tried to calculate the values of fitness and precision. I noticed that, to calculate the precision, it was taking too long, as proof that the model was too complicated.

In order to solve the problem, I started to reduce the number of activities leaving the number of paths as default (0.8) and at each step calculate the value of fitness and precision to have an idea how good the model was.

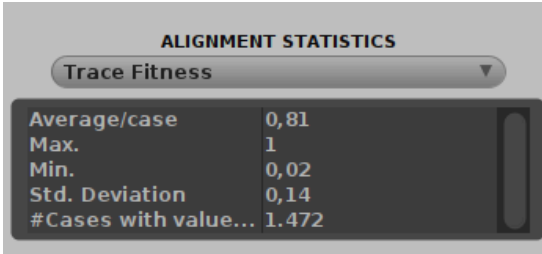
The first significant reduction of the petri net can be found when we allow 0.4 activities where we obtain 0.92 as fitness and 0.57 as precision. (File

Q2_1_Petrinet_Inductive.pnml)

At this point I tried to reduce the number of paths since there were a lot of move model only.

Using the parameters of 0.4 for the activities and 0.7 for the paths, I managed to obtain 0.81 for the fitness and 0.83 for the precision. (File

Q2_2_Petrinet_Inductive.pnml)



ALIGNMENT STATISTICS	
Trace Fitness	
Average/case	0,81
Max.	1
Min.	0,02
Std. Deviation	0,14
#Cases with value...	1.472

Precision : 0,82871

Generalization : 1,00000

Fig. 5 Values of fitness and precision of the model produced with the Inductive Visual Miner

2.3 Interactive Data-aware Heuristic Miner (iDHM)

The second plugin I used to obtain a Petri net is the “*Interactive Data-aware Heuristic Miner (iDHM)*”.

As for the other method, I provided as input the complete log. I then selected as an event classifier the *Event Name*. At this point, as a first try, I left the default parameters.

Once I computed the value of the fitness, I noticed that every trace was starting with an unobservable move, meaning that there was a silent transition in the model. In order to try to fix this problem I used the plugin “*Reduce Silent Transition*”. I, then, used the resulting Petri net again to calculate the value of the fitness. The value of

the fitness is the same as before, but this time every trace is starting with a synchronous move or with a model move only.

At this point I calculated the value of the precision using as input the log file, the reduced Petri net and the result produced by the replay plugin.

The value of the fitness was greater than 0.8, while it took time to calculate the precision.

I proceeded to modify the value of the parameters. At the end using 0.2 for both frequency and dependency I managed to produce a result with fitness and precision greater than 0.8. Here I report the results.



Fig. 7 Values of fitness and precision of the model produced with the iDHM

Looking at the results of fitness and precision produced by these two models, we can see that the best model is the one returned by the iDHM plugin.

If we compare this model and the one that I drew we can see that there are some differences.

The first thing that we can notice is that in the model produced by the iDHM plugin there isn't an initial XOR. In essence, the only action performed at the beginning is *Create Purchase Requisition*. Another difference is that *Receive Order Confirmation* and *Vendor Create Invoice* can be performed in parallel. We can also notice that in the model I drew, there are two different loops, while here there is no loop.

To conclude, the Petri net produced by the plugin is simpler than the one I drew, but still can reach a good value of precision and fitness.

Q3. Simulation Parameters and Simulation

In this last question the aim was to simulate the process and try to optimize it.

In order to do that, I used the BIMP simulator.

As first thing, we have to upload a *.BPMN* file which can be easily produced using the plugin *Convert Data Petri net to BPMN diagram* offered by Prom.

The Petri net I used as model is the one produced by the “*Inductive visual miner*” obtained using the complete log as input and default parameters. I decided to use this model because, since we are trying to optimize the process, I wanted to have the most general model possible.

After uploading the file, I started completing all the requested parameters.

As *Internal Arrival Time* I used Exponential with the value of 102 seconds (1m 42s) as produced by the python function. The *total number of process instances* is 10000 since we are considering the complete log file. Then, I decided to discard the 5% of the instances both from the head and the tail, in order to consider a warm up time.

As Initial date of the scenario I used the date provided by the Log file.



Fig. 8 Pm4py case arrival rate and start and ending value of the process

For the Resources section I used the result produced by the Pm4py function. Here I provide a table in which we can see which activities belong to which role and the number of resources for each role.

ROLE	ACTIVITIES	RESOURCES
Role1	Adjustment Charge, Cancel Goods Receipt, Cancel Invoice Receipt, Change PR Approval, Change Price, Change Quantity, Change Vendor, Create Purchase Order, Create Purchase Requisition, Delete Purchase Requisition, Print and Send Purchase Order (Paper), Reactivate Purchase Order, Receive Order Confirmation, Record	32

	Goods Receipt, Record Invoice Receipt, Send Purchase Order Update, Vendor Creates Invoice	
Role2	Block Purchase Order	24
Role3	Change Currency	19
Role4	Clear Invoice	1
Role5	Delete Purchase Order	17
Role6	Dun Order Confirmation	9
Role7	Send Overdue Notice	13

Table 1 Roles in the event log

As we can notice, the first role is the one with the greatest number of activities and resources, while the fourth role has only one resource. This will be something to take into account when optimizing the process.

For the *Timetables/Work schedules* part, I used the Pm4py function and calculated it for each resource.

Here I provide an example for the resource Quinna.

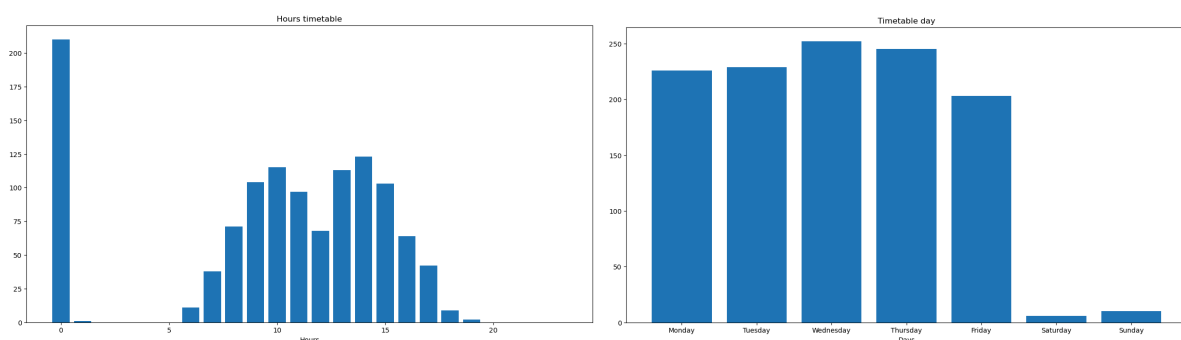


Fig. 9 Plot of the Timetable and Work schedules of the resource Quinna

As we can see, resources are working every day, but mostly from Monday to Friday. Looking at the timetable we have a peak at 0 while a normal behavior from 7 to 19. For these reasons, in the Default section I choose to consider the period of work 24/7.

For the *Task* section, I provide an example for the activity *Cancel Goods Receipt*. Since in all the graphs there was a peak around 0, I decided to interpret the distributions as Uniform, except for the *Print and Send Purchase Order (Paper)* which is Fixed since the standard deviation is equal to zero.

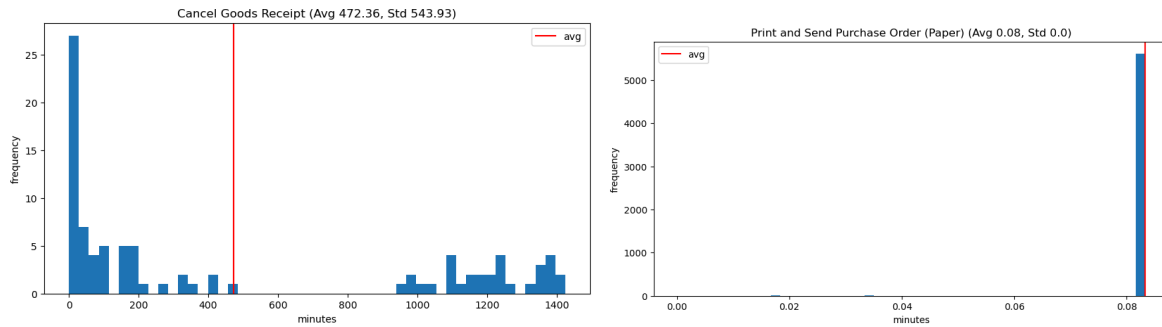


Fig. 10 Distribution of the activities Cancel Goods Receipt and Print and Send Purchase Order (Paper)

For the branch probabilities I converted the Petri net into a BPMN diagram and then I used the *Multi-perspective Process Explorer* plugin in order to be able to obtain all the percentages of each XOR branch.

At this point I started the simulation, but the application returned a message error due to the fact that the model is not converging. In order to have a better understanding of where the problem could be, I uploaded the log file into the software Disco and saw that there is a bottleneck on the activity *Clear Invoice*. This happens because all the other activities are executed very quickly, while *Clear Invoice* is executed by just one resource who is not able to execute this activity at this rate.

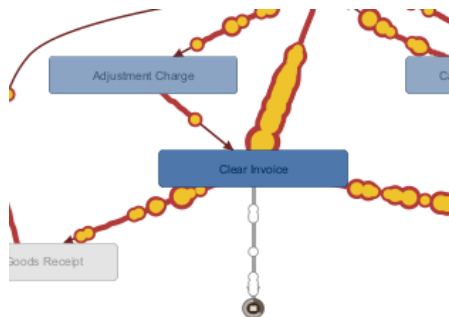


Fig. 11 Bottleneck on Clear Invoice

In order to be able to compute a first simulation, I had to reduce the *number of process instances* from 10000 to 2000. Here I report the results of the average time and the percentage of occupation of the resources.

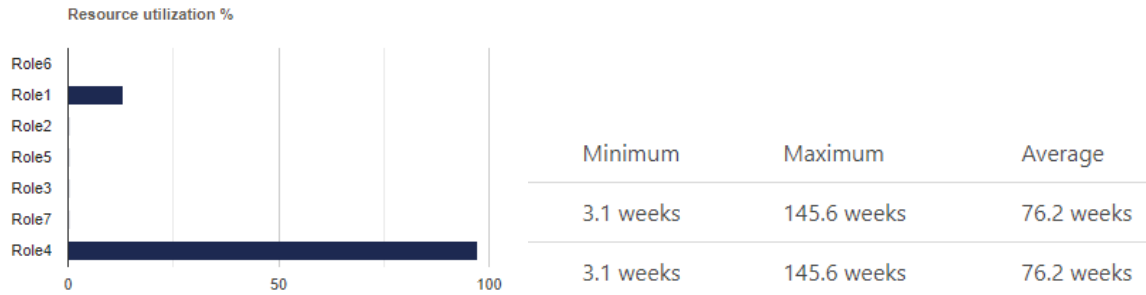


Fig. 12 Resource utilization and average time of execution of the process for 2000 instances

From these images, it is clear that we have to increase the number of resources in Role4.

The first attempt was to decrease the number of resources from Role1 and utilize those resources on Role4. I passed from 32 to 20 resources for Role1 and from 1 to 13 for Role4. Then I tried to run the simulation on 10000 instances without being able to do it. So I decided to also distribute the activity for all the other roles. Thanks to this change I was able to run 10000 instances of the process. Here I report the result of this first optimization.



Fig. 13 Resource utilization and average time of execution of the process after the first optimization

As we can see, thanks to this small change, the average time of execution has decreased from 76 weeks to 55. Also the resources are now occupied more uniformly.

In order to try to understand how these changes have affected the process, I uploaded the simulation on disco to see the result.

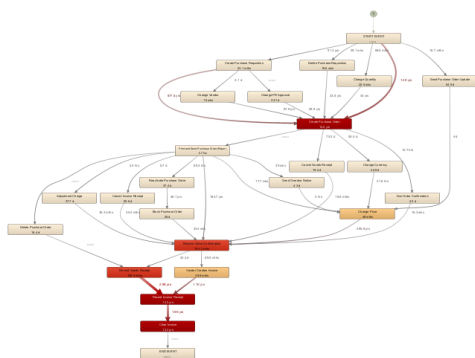


Fig. 14 Performance map of the process after the first optimization

As we can see now there are other bottlenecks.

In order to try to optimize the process I kept moving resources and activity from one role to another and at some point I also started adding resources.

After some attempts I obtained the following distribution

ROLE	ACTIVITIES	RESOURCES
Role1	Adjustment Charge, Cancel Goods Receipt, Change Currency, Change Vendor, Record Goods Receipt,	17
Role2	Block Purchase Order, Create Purchase Requisition, Record Invoice Receipt, Send Purchase Order Update	25
Role3	Create Purchase Order	23
Role4	Cancel Invoice Receipt, Clear Invoice, Delete Purchase Requisition,	21
Role5	Change PR Approval, Change Quantity, Delete Purchase Order, Send Overdue Notice, Vendor Creates Invoice	7
Role6	Change Price, Dun Order Confirmation, Reactivate Purchase Order,	7
Role7	Print and Send Purchase Order (Paper), Receive Order Confirmation,	15

Table 2 Roles in the event log of the final optimization

If we take a look at the performance on disco, we can see that there is still a bottleneck on *Create Purchase Order*, *Record Invoice Receipt* and *Clear Invoice*, but now, since activities and resources are more distributed, the process is quicker and smoother.

This can be seen also by the results produced by the simulation



Fig. 13 Resource utilization and average time of execution of the process after the final optimization

As we can see, thanks to these small changes, we managed to decrease the average time from 55 weeks to 21.

In conclusion, my suggestion for the company is to better distribute the resources and activities in the different roles without the necessity of adding new resources.