UNIVERSITÀ DEGLI STUDI DI MILANO

BPI Challenge 2019
Project of Business information system

Professor Paolo Ceravolo

Author Mohammad Derakhshan

989201

Department of Computer science

Academic year 2022-2023

Content

Overview

In the Section 1, we have an introduction about the log file. We explore the four existed category and try to understand how an item is assigned to a category.
Then in section 2, we perform preprocessing and filtering. In subsection 2.1 We go deep into log file by analyzing frequent activities, frequent variants, we do start/end activity analysis, and try to figure out the most frequent variant pattern. Finally with the help of knowledge acquired, we filter out noise, uncompleted or even not frequent data.
Then in section 3, we apply different process mining algorithm trying to model the system. Section 3.1 dedicated to “Alpha miner” algorithm. Section 3.2 will focus on “Inductive miner” algorithm and section 3.3 is dedicated to “Heuristic miner” algorithm. In section 3.4 we draw process tree and finally in section 3.5 we compare performance of each algorithm by presenting a heatmap chart.
In section 4, we divide the log file into four categories and perform our operations on each category. Section 4.1 discus how we can segment log file, section 4.2 focuses on applying filters on each segment and section 4.2 compares the performance of each process discovery algorithm on each segment.

1. Introduction

The analytics are done on a log file containing the data related to a large multinational Dutch company working in the area of coatings and paints.

In the data, each purchase contains at least one line item. For each line item, almost we have four categories:

1. 3-way matching, invoice after goods receipt: For these items, the value of the goods receipt message should be matched against the value of an invoice receipt message and the value put during creation of the item (indicated by both the GR-based flag and the Goods Receipt flags set to true).
2. 3-way matching, invoice before goods receipt: Purchase Items that do require a goods receipt message, while they do not require GR-based invoicing (indicated by the GR-based IV flag set to false and the Goods Receipt flags set to true). For such purchase items, invoices can be entered before the goods are receipt, but they are blocked until goods are received. This unblocking can be done by a user, or by a batch process at regular intervals. Invoices should only be cleared if goods are received and the value matches with the invoice and the value at creation of the item.
3. 2-way matching (no goods receipt needed): For these items, the value of the invoice should match the value at creation (in full or partially until PO value is consumed), but there is no separate goods receipt message required (indicated by both the GR-based flag and the Goods Receipt flags set to false).
4. Consignment: For these items, there are no invoices on PO level as this is handled fully in a separate process. Here we see GR indicator is set to true, but the GR IV flag is set to false and also, we know by item type (consignment) that we do not expect an invoice against this item.

However, the complexity of data goes beyond these four divisions.

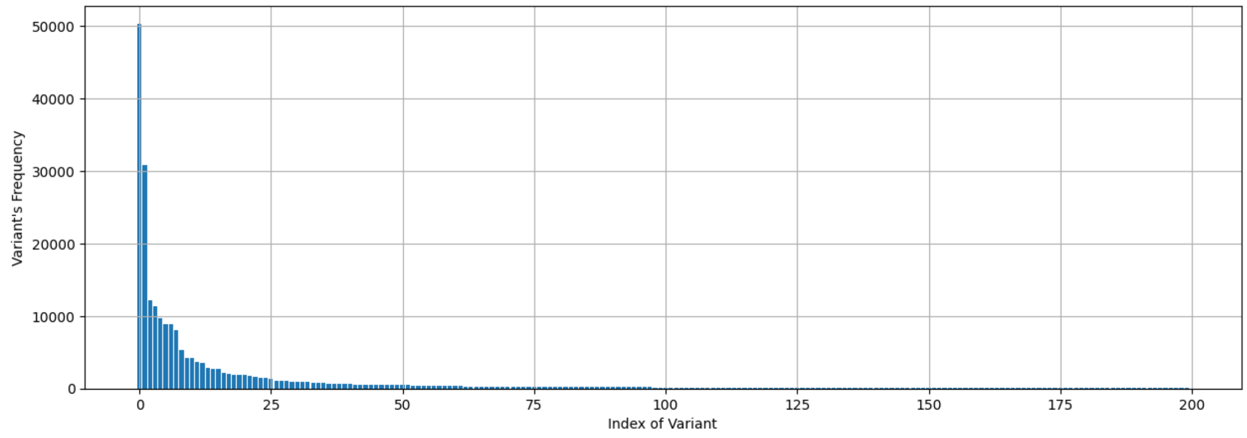
2. Preprocessing

In this section, we explore the log file and find a brief understanding of it. Also, we achieve a more robust log file by applying some filters and reducing the noise.

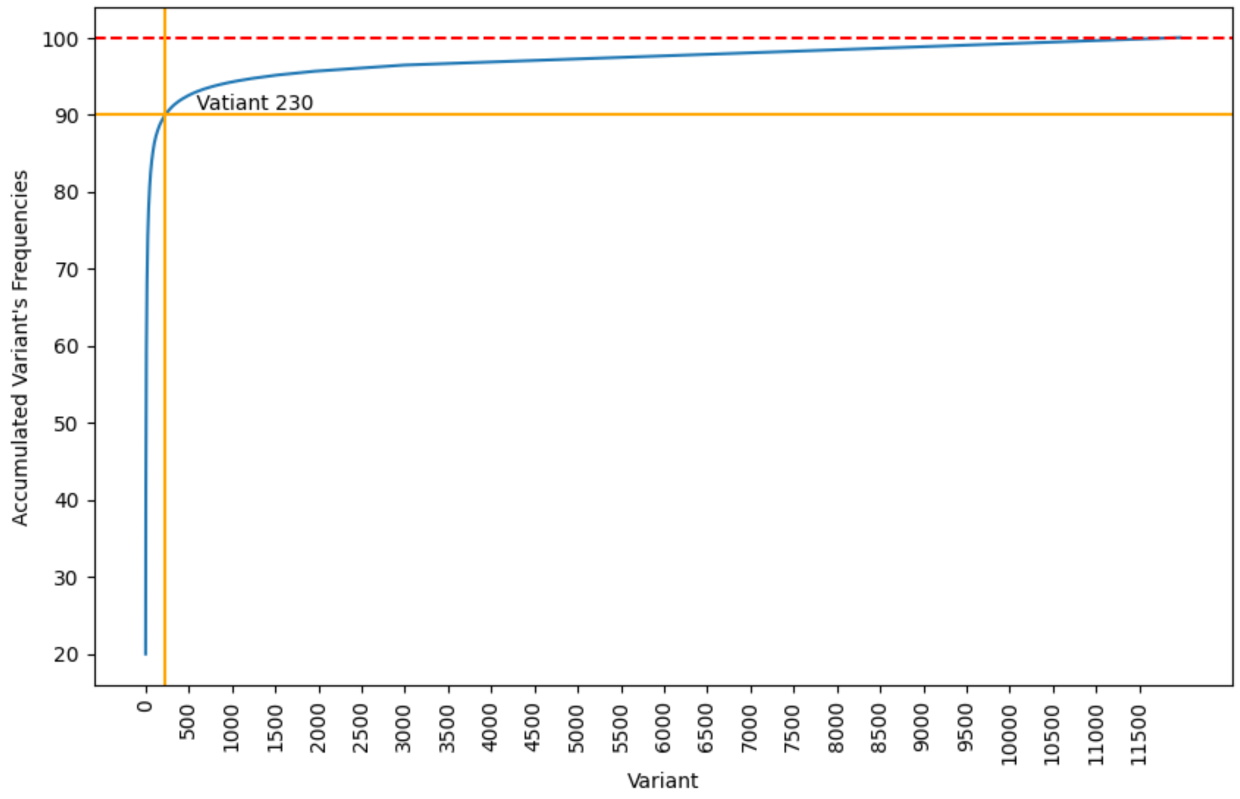
2.1. Exploring Log File

- **Analyzing variant frequency**

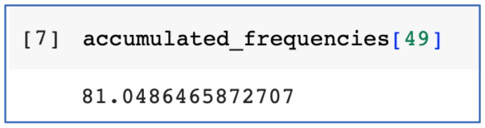
The first analyze we perform is variant frequency checking. We extract the variants from the log file using "`pm4py.get_variants(log)`". Then we sort them according to their frequency. Below is the frequency of top 200 variants.



As the chart illustrates, lots of information can be obtain by just considering the first top 50 variants. To understand it better, we try to compute accumulative frequency percentage. The bar chart regarding that is as below:



As the chart suggest, to have 90% of variants, we can consider top 231 items (230 + 1 = 231 due to zero index starting). It is notable that if we consider top 50 items, we will cover 81% of variants.



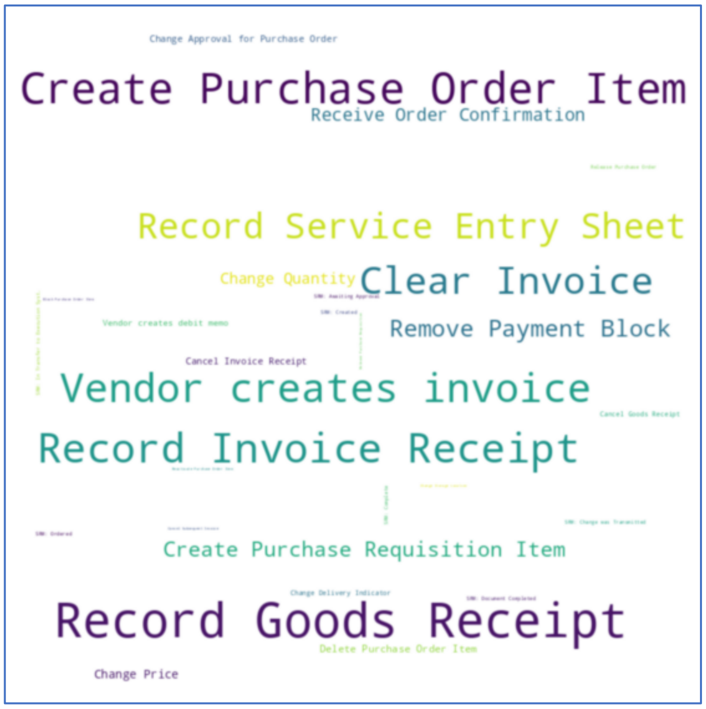
```
[7]  accumulated_frequencies[49]

     81.0486465872707
```

- **Analyzing activity frequency**

Another aspect to consider is activity frequency. We have 42 activities and only 10 of them create almost 96% of log file. The frequency of top ten activities is as follow:

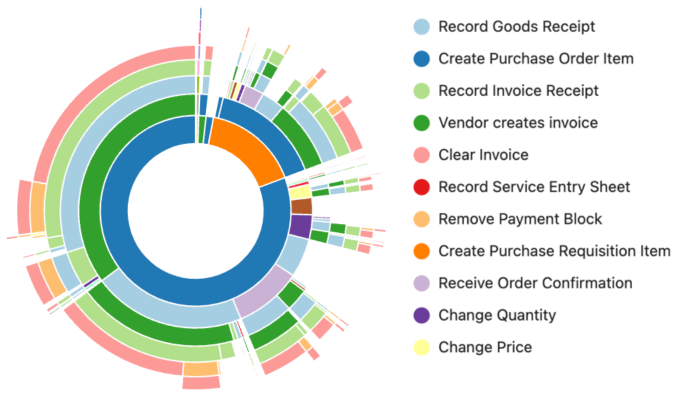
#	Activity Name	Frequency	Relative frequency
1	Record Goods Receipt	314,097	19.68 %
2	Create Purchase Order Item	251,734	15.77 %
3	Record Invoice Receipt	228,760	14.33 %
4	Vendor creates invoice	219,919	13.78 %
5	Clear Invoice	194,393	12.18 %
6	Record Service Entry Sheet	164,975	10.34 %
7	Remove Payment Block	57136	3.58%
8	Create Purchase Requisition Item	46592	2.92%
9	Receive Order Confirmation	32065	2.01%
10	Change Quantity	21449	1.34%

To understand distribution of activity frequency we consider following word cloud chart:



- **Analyzing variant sequence frequency**

Another noteworthy part is the sequence of variants. With the help of the Sunburst chart provided by PmTK, we realize that 81% of activities start with “Create purchase order item” and 16 % with “Create purchase requestion item.”



- **Analyzing start and end**

It is also valuable to check the list of start and end activities. By doing that, we can identify some anomalies that can be useful during the filtering phase.

Start Activity		
#	Name of activity	Number of occurrences
1	Create Purchase Order Item	199867
2	Create Purchase Requisition Item	46526
3	Vendor creates invoice	3457
4	SRM: Created	1360
5	Change Approval for Purchase Order	378
6	Vendor creates debit memo	122
7	Release Purchase Order	22
8	Change Currency	2

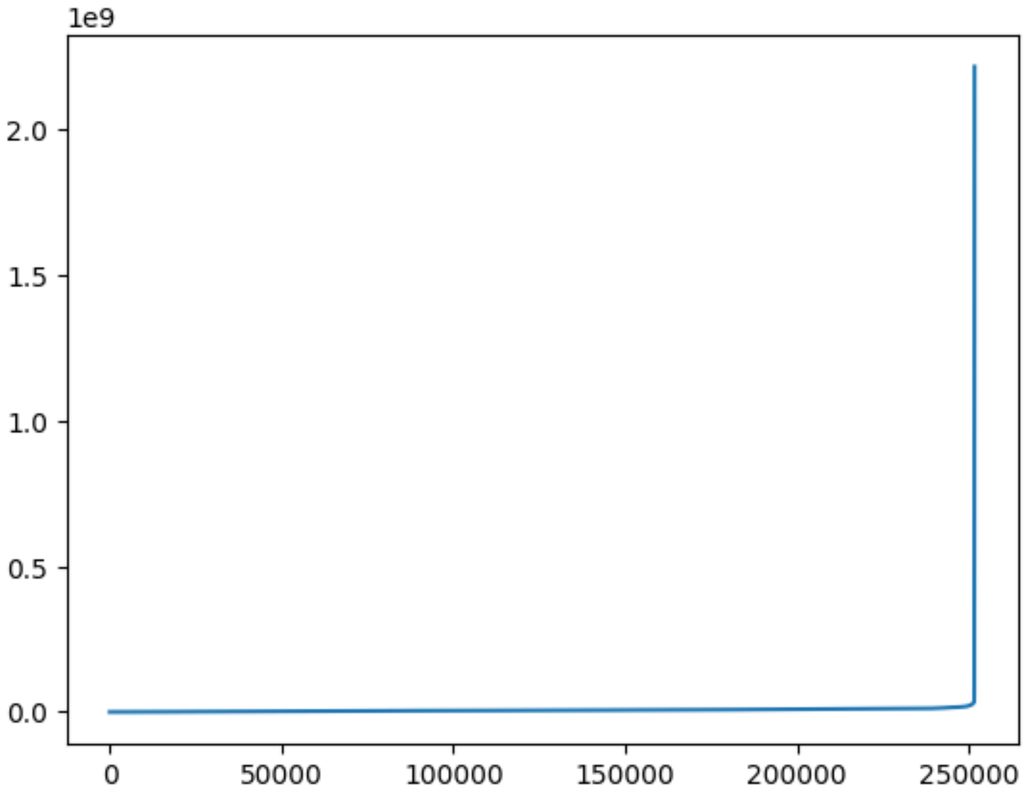
End Activity		
#	Name of activity	Number of occurrences
1	Clear Invoice	181328
2	Record Invoice Receipt	23091
3	Record Goods Receipt	22776
4	Delete Purchase Order Item	8123
5	Remove Payment Block	5397
6	Create Purchase Order Item	4447
7	Cancel Invoice Receipt	1348
8	Change Approval for Purchase Order	1346
9	Record Service Entry Sheet	921
10	Change Delivery Indicator	718

In total, we have 32 end activities. This table illustrates top ten.

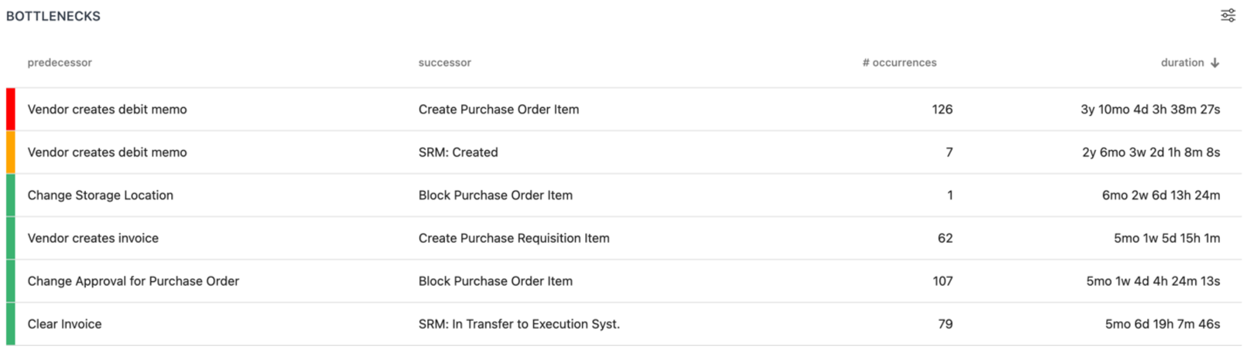
- **Analyzing case duration**

Another criterion we consider is case duration. One of the best ways to filter noises and anomalies is based on their life cycle.

By running the "`pm4py.get_all_case_durations(log)`" we obtain case duration in seconds. Then we try to illustrate duration distribution by drawing a line chart.



As the chart depicts, there are a few cases in which their duration is significantly higher than the rest. We may consider them as anomaly. We use pmTk throughput analytic tool to find bottlenecks. Here are top five time-consuming activities:



BOTTLENECKS

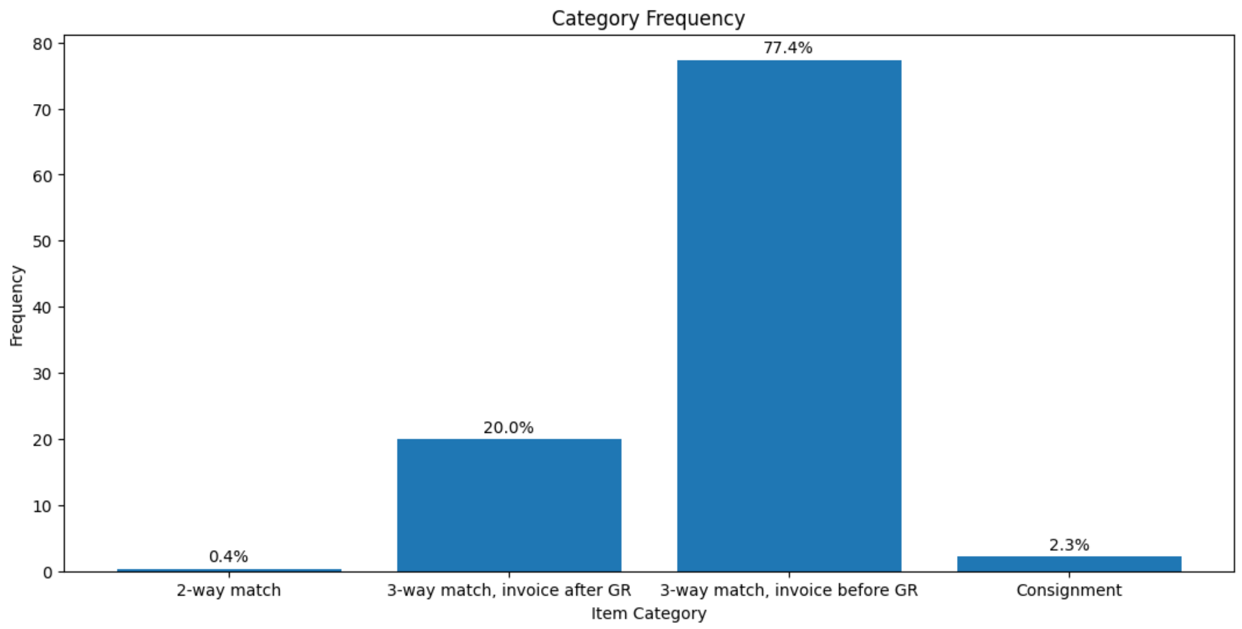
	predecessor	successor	# occurrences	duration ↓
🟥	Vendor creates debit memo	Create Purchase Order Item	126	3y 10mo 4d 3h 38m 27s
🟧	Vendor creates debit memo	SRM: Created	7	2y 6mo 3w 2d 1h 8m 8s
🟩	Change Storage Location	Block Purchase Order Item	1	6mo 2w 6d 13h 24m
🟩	Vendor creates invoice	Create Purchase Requisition Item	62	5mo 1w 5d 15h 1m
🟩	Change Approval for Purchase Order	Block Purchase Order Item	107	5mo 1w 4d 4h 24m 13s
🟩	Clear Invoice	SRM: In Transfer to Execution Syst.	79	5mo 6d 19h 7m 46s

- **Analyzing category frequency**

In addition, we check the frequency of each category. According to challenge description, we have four categories:

1. 3-way matching, invoice after goods receipt
2. 3-way matching, invoice before goods receipt
3. 2-way matching (no goods receipt needed)
4. Consignment

The frequency of each category is depicted in the following chart:

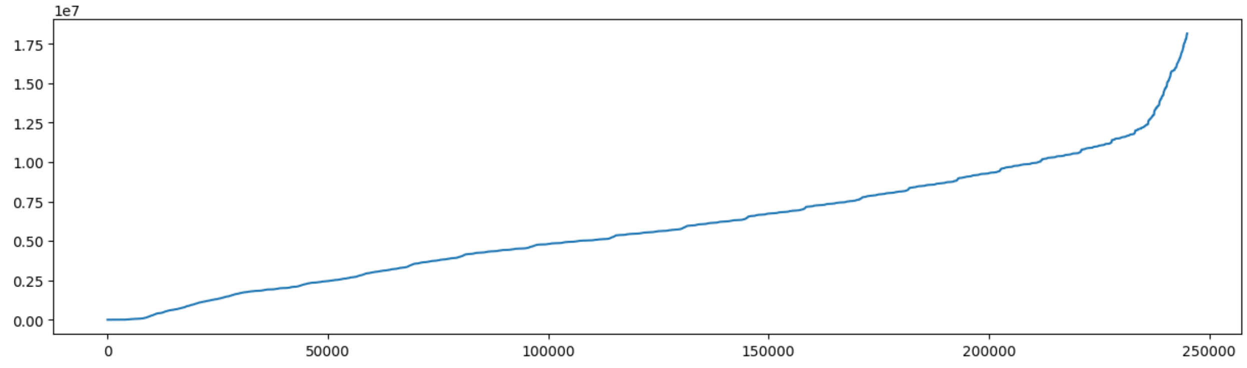


As the chart suggests, the most frequent category is “`3-way matching, invoice before goods receipt`” and the least frequent is “`2-way match`”.

2.2. Filtering and Processing Log file

- **Case Performance**

The first criterion we consider is completion duration. The minimum duration is 0 milliseconds, and the maximum duration is 70 years and 145 days. No case should take that long. Hence it indicates a misrecording in event logs. We filter such cases and only consider cases whose duration is between 1 minute and 210 days. After applying this filter, we try to draw a case duration chart. The result is as follows:



Considering the raw data case duration chart will show significant improvement in duration balance. This filter removed `251734 - 245042 = 6692` case from log file.

- **Top Variant Filter**

The second filter we apply is "top-k variant" we already found out that top 231 variants have 90% of frequency. By applying this filter, `245042 - 222355 = 22687` case are eliminated.

- **Start and End Activity**

Third filter is regarding start and end activities. According to our understanding about frequency of each activity, we only keep the records that start with:

```
"Create Purchase Order Item",
"Create Purchase Requisition Item",
"Vendor creates invoice",
"SRM: Created",
"Vendor creates debit memo"
```

And end with:

```
"Clear Invoice",
"Record Invoice Receipt",
"Record Goods Receipt",
"Delete Purchase Order Item",
"Remove Payment Block",
"Cancel Invoice Receipt",
"Change Approval for Purchase Order",
"Change Delivery Indicator",
"Cancel Goods Receipt",
"Receive Order Confirmation",
"Block Purchase Order Item",
"Set Payment Block",
"SRM: Change was Transmitted",
"SRM: Transfer Failed (E.Sys.)",
"SRM: In Transfer to Execution Syst.",
"Cancel Subsequent Invoice",
"SRM: Deleted",
"SRM: Transaction Completed",
"Update Order Confirmation"
```

After performing this filter, we remove `222355 - 219997 = 2358` cases.

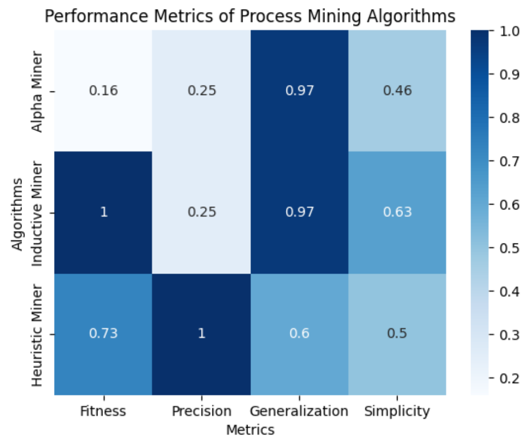
- **Activity Rework**

Another aspect to consider is repetition of activities. We filter out the cases in which has more than 2 repetitions regarding following activities:

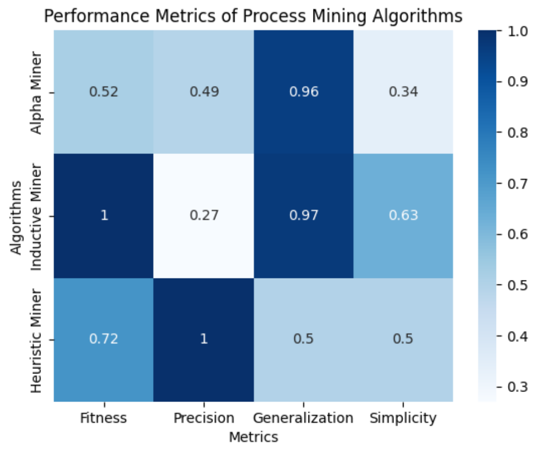
```
Clear Invoice
Record Goods Receipt
Record Invoice Receipt
```

By doing that, we remove `4100` cases. The remaining is `215897` cases.

Even though we will explore process discovery in the following sections, it is worth to mention the chart on the left shows the performance of different discovery algorithms before using this filter. On the right, we see the output after using this filter:



Performance before applying filter.



Performance after applying filter.

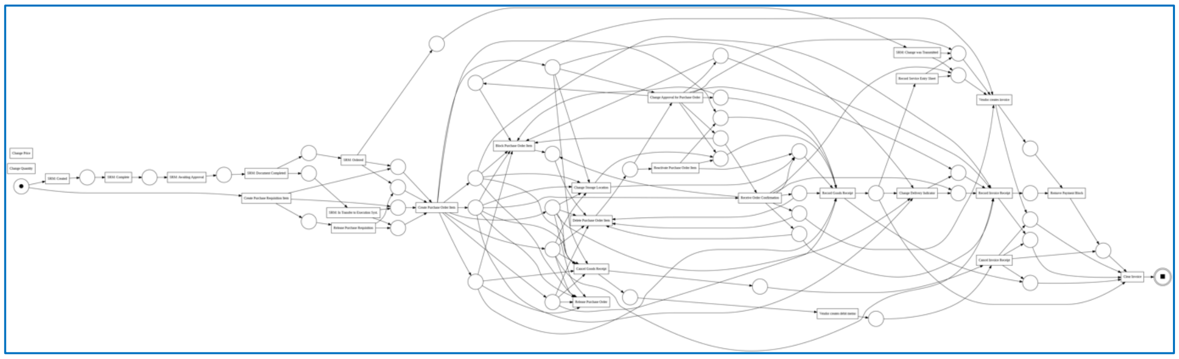
3. Process Discovery

In this section we try to apply different process discovery algorithm and do a comparison among them for four criteria of generalization, fitness, precision, and simplicity.

3.1. Alpha Miner

The first algorithm we use is Alpha Miner. The Alpha Miner is a process mining algorithm used to discover process models from event logs. It focuses on identifying causal dependencies between activities in a process. It analyzes the order in which events occur and infers the relationships between them, such as the precedence or concurrency of activities. The Alpha Miner algorithm is based on the observation of the so-called "directly follows" relation and uses this information to construct a process model, typically in the form of a Petri-net or a process flowchart.

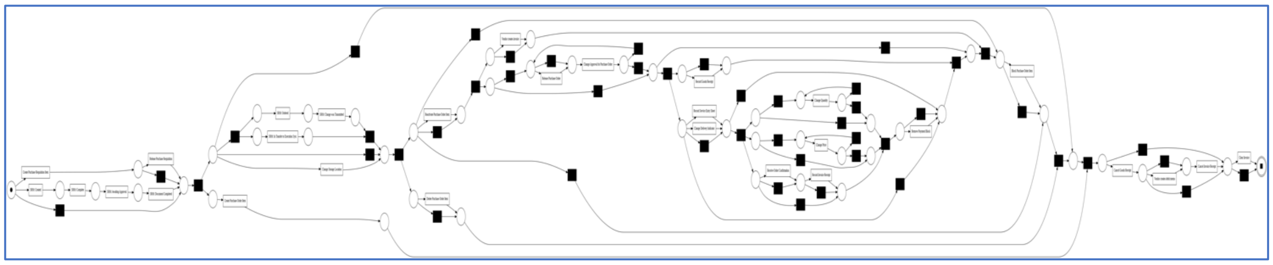
By applying this algorithm on log file and generating Petri-net from the result we will have following graph:


Change Price
Change Quantity
SRM: Created
SRM: Complete
SRM: Awaiting Approval
SRM: Document Completed
SRM: Ordered
Create Purchase Requisition Item
SRM: In Transfer to Execution Syst.
Release Purchase Requisition
Create Purchase Order Item
Block Purchase Order Item
Change Approval for Purchase Order
Reactivate Purchase Order Item
Change Storage Location
Delete Purchase Order Item
Cancel Goods Receipt
Release Purchase Order
Receive Order Confirmation
Record Goods Receipt
Change Delivery Indicator
SRM: Change was Transmitted
Record Service Entry Sheet
Vendor creates invoice
Record Invoice Receipt
Remove Payment Block
Cancel Invoice Receipt
Clear Invoice
Vendor creates debit memo

It seems that the algorithm was not successful to model the process well. There is no clear path from start to end.

3.2. Inductive Miner

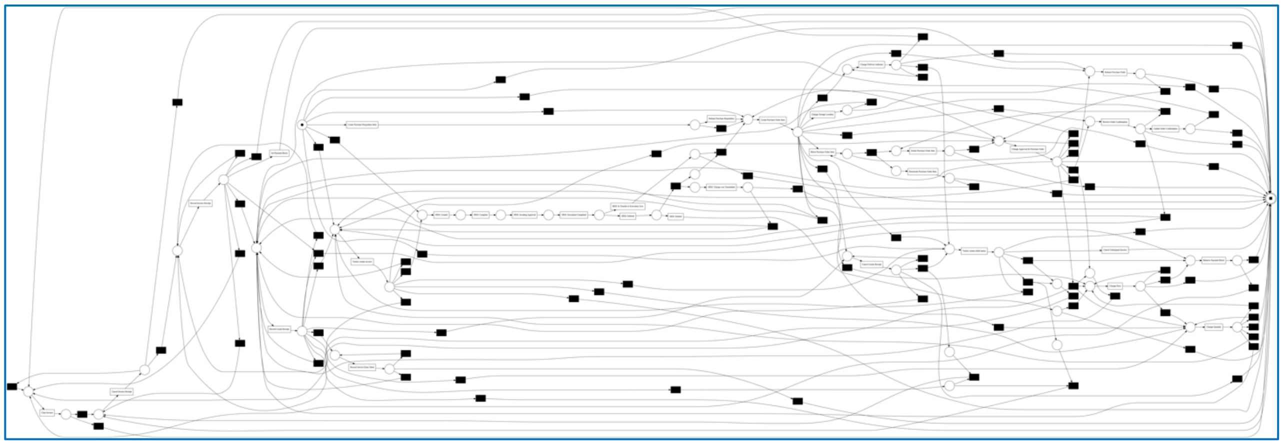
Next algorithm is Inductive Minder. The Inductive Miner is another process mining algorithm that aims to discover process models from event logs. It focuses on inferring the structure of a process model, including the possible choices and loops. Unlike the Alpha Miner, the Inductive Miner algorithm uses a bottom-up approach, starting from individual activities and gradually building a model by merging similar behavior. It handles more complex process behaviors, such as exclusive choices and parallelism, making it suitable for a wider range of process models. After applying this algorithm on filtered log file and producing Petri-net, we have the below output:



The output seems to be more precise than alpha-miner. The paths are clear and all activities has a start and end point.

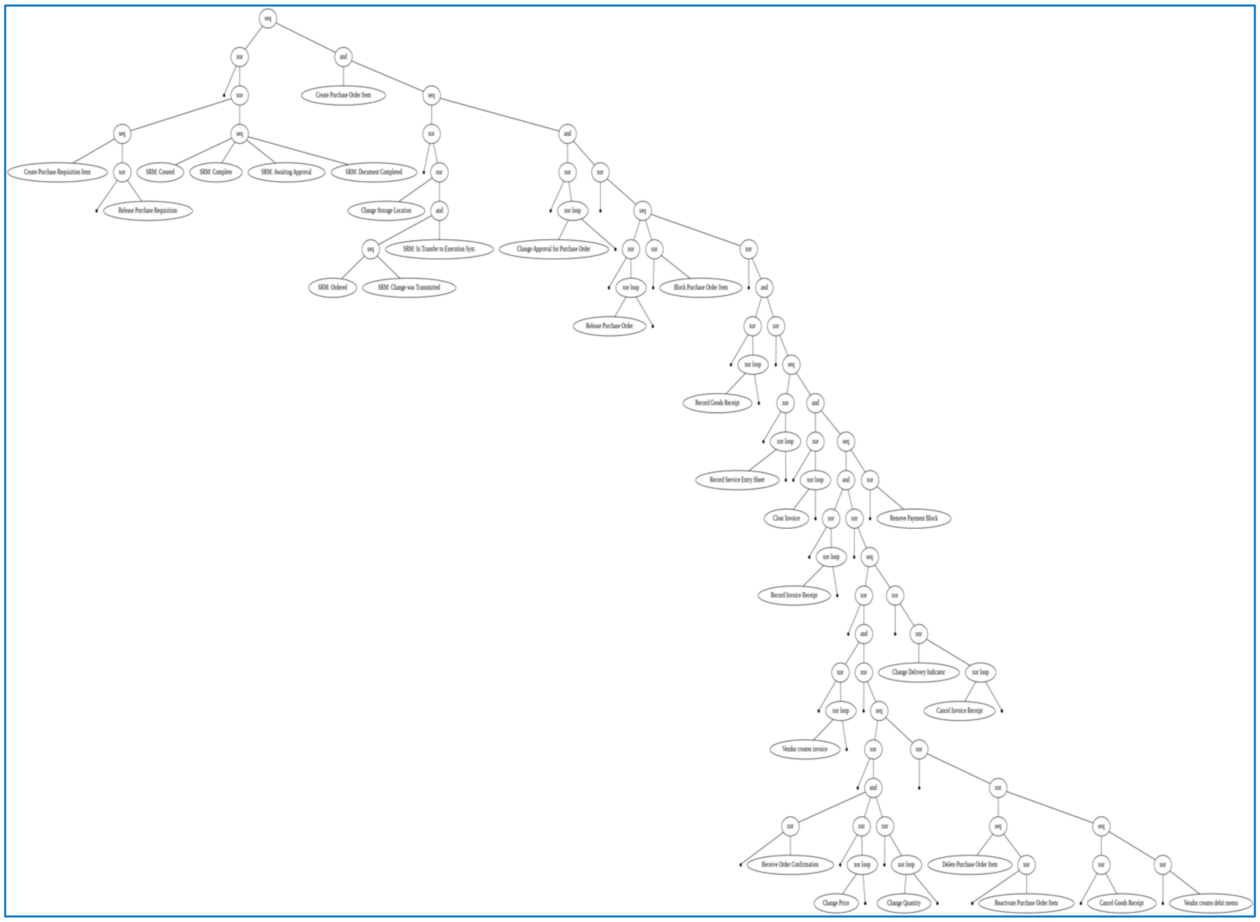
3.3. Heuristic Miner

The Heuristic Miner is a process mining algorithm that infers process models from event logs by applying heuristics to identify patterns and dependencies. It considers factors such as frequency, exclusive choices, and concurrency. In comparison to the Alpha Miner, the Heuristic Miner can handle noisy event logs and capture more complex process behaviors. Unlike the Inductive Miner, it follows a different approach by focusing on heuristics rather than a bottom-up or structural inference method. If we apply this algorithm on the log file and then try to produce the Petri-net out of it, we achieve following result:



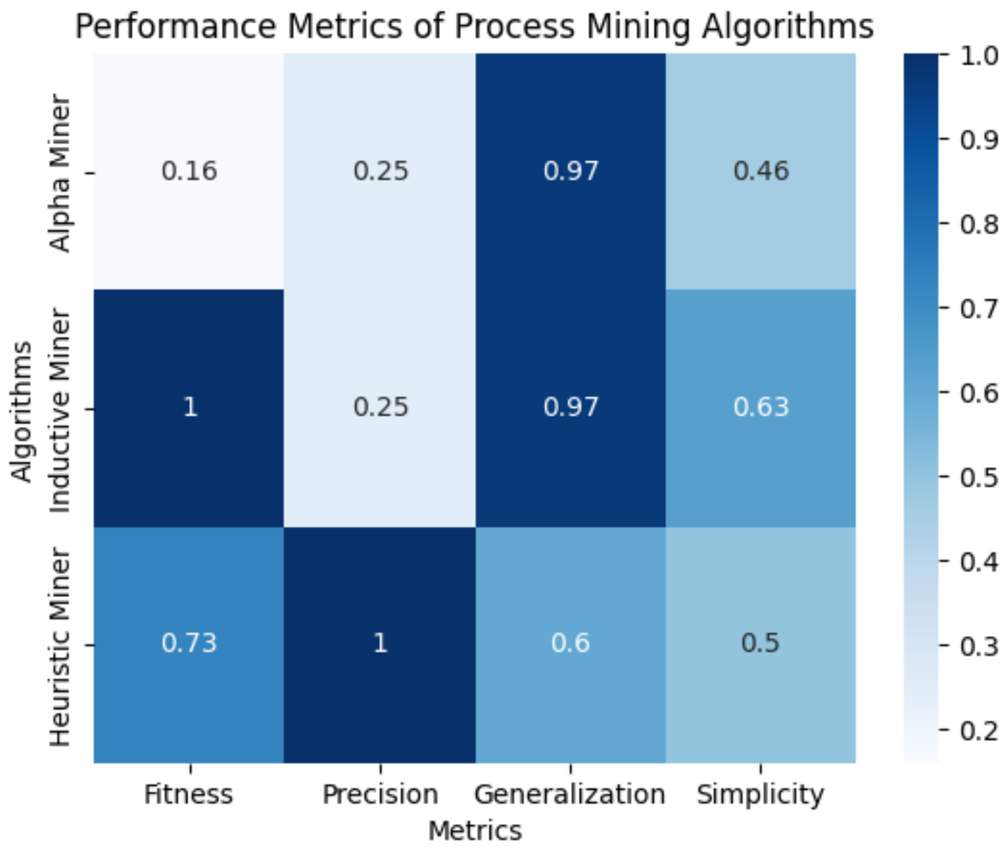
3.4. Process Tree

Process tree is a visual representation of a process model that captures the control flow and behavior of a process. It is a hierarchical structure that depicts the relationships between activities, decisions, and loops in a process. Process trees provide an intuitive and human-readable representation of a process, showing the order of activities, the possible choices, and the looping behavior. Process trees are commonly used to analyze and understand process behavior, identify bottlenecks, and optimize processes. After running process tree generator using pm4py, we obtain following graph:


Create Purchase Order Item
Create Purchase Requisition Item
SRM: Created
SRM: Complete
SRM: Awaiting Approval
SRM: Document Completed
Release Purchase Requisition
Change Storage Location
SRM: In Transfer to Execution Syst.
SRM: Ordered
SRM: Change was Transmitted
Change Approval for Purchase Order
Block Purchase Order Item
Release Purchase Order
Record Goods Receipt
Record Service Entry Sheet
Clear Invoice
Remove Payment Block
Record Invoice Receipt
Change Delivery Indicator
Cancel Invoice Receipt
Vendor creates invoice
Receive Order Confirmation
Delete Purchase Order Item
Change Price
Change Quantity
Reactivate Purchase Order Item
Cancel Goods Receipt
Vendor creates debit memo

3.5. Performance comparison

To understand how good each model is, we consider four aspects: recall, precision, generalization, and simplicity.



According to this data, it seems that Heuristic miner has better performance in general terms.

4. Data Segmentation

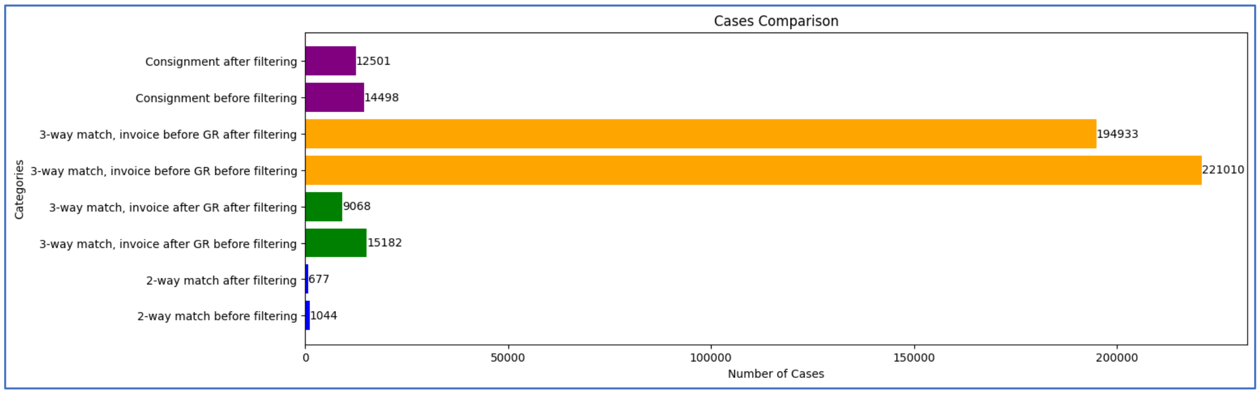
In this section, we try to categorize data according to four categories:

```
"2-way match"
"3-way match
"Invoice after GR"
"3-way match, invoice before GR"
"Consignment"
```

Then we repeat filtering, process discovery, conformance checking, etc. on each category.

4.1. Filtering

We apply the filters mentioned above to each category. After doing that, the number of remaining cases is as follow:

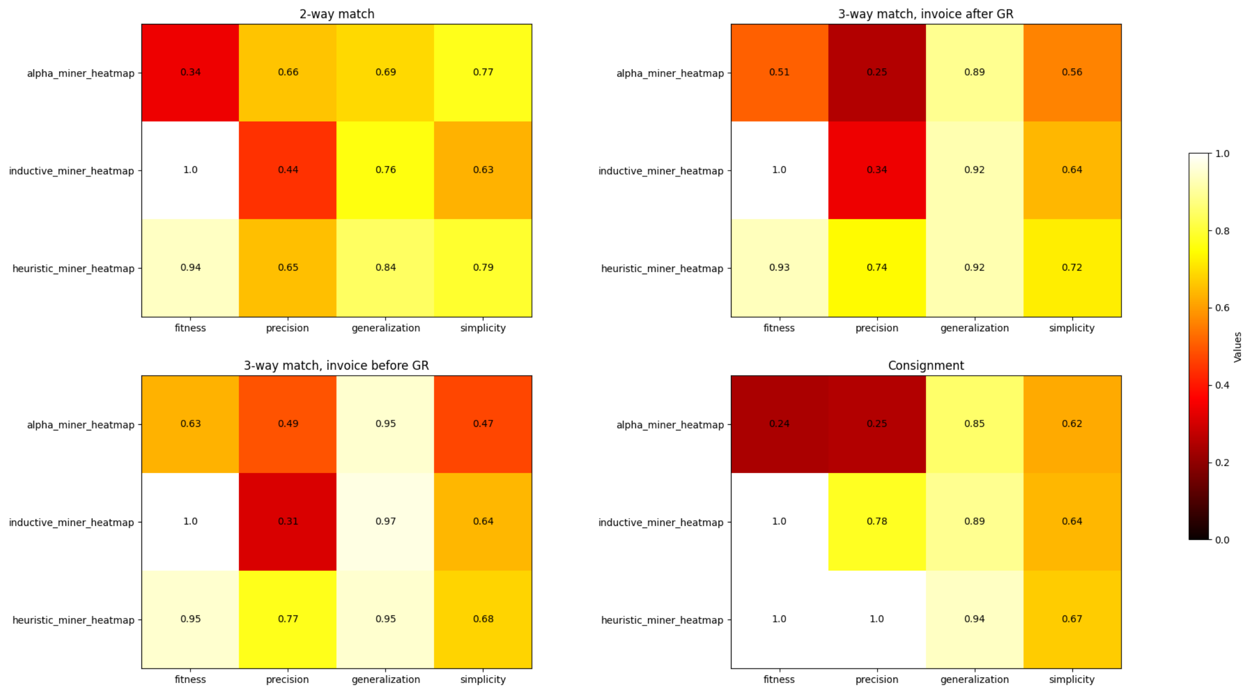


Another filter which we are capable to do is existence of specific activities. According to log description, for the category `3-way match, invoice after GR,` it should contain `"Record Goods Receipt"` and `"Record Invoice Receipt".` After applying this filter, we're going to have *8015* cases left in this category.

For `3-way match, invoice before GR,` we should consider presence of `"Record Goods Receipt".` After applying this filter, we're going to have *185608* cases left in this category.

4.2. Process Discovery

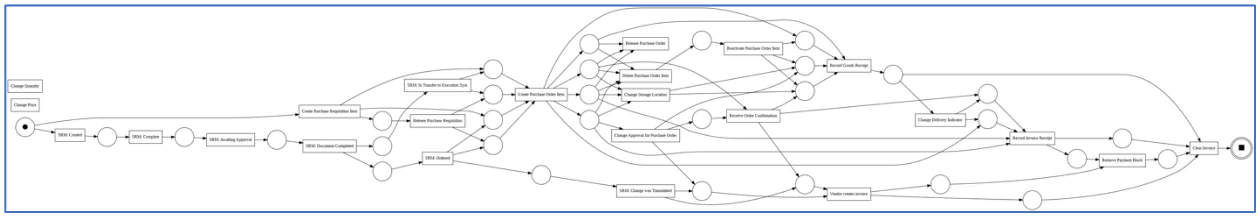
Now we apply three process discovery algorithms on each one of them and compute the performance of each generated model.



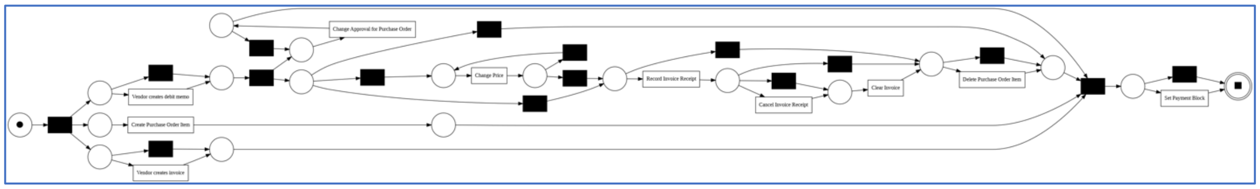
Even in this case, heuristic miner performs better compared to other approaches.

If we need to decide which segment each algorithm fits best,

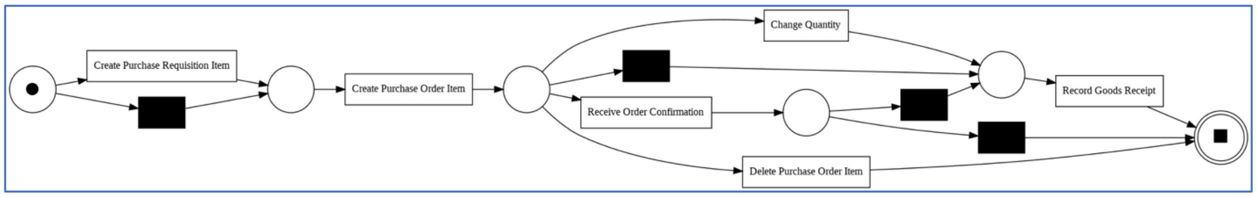
- Heuristic miner for Consignment
- Inductive miner for 2-way match and
- alpha miner, 3-way match, invoice before GR


Change Quantity
Change Price
SRM: Created
SRM: Complete
SRM: Awaiting Approval
Create Purchase Requisition Item
SRM: Document Completed
SRM: In Transfer to Execution Syst.
Release Purchase Requisition
SRM: Ordered
Create Purchase Order Item
Release Purchase Order
Delete Purchase Order Item
Change Storage Location
Change Approval for Purchase Order
SRM: Change was Transmitted
Reactivate Purchase Order Item
Receive Order Confirmation
Record Goods Receipt
Vendor creates invoice
Change Delivery Indicator
Record Invoice Receipt
Remove Payment Block
Clear Invoice

Alpha miner for three-way match, invoice before good receive.


Change Approval for Purchase Order
Vendor creates debit memo
Create Purchase Order Item
Vendor creates invoice
Change Price
Record Invoice Receipt
Cancel Invoice Receipt
Clear Invoice
Delete Purchase Order Item
Set Payment Block

Inductive miner two-way match


Create Purchase Requisition Item
Create Purchase Order Item
Change Quantity
Receive Order Confirmation
Delete Purchase Order Item
Record Goods Receipt

Heuristic miner, Consignment