BUSI 2710

Class 20 - Process Mining

While workflow management systems are explicitly driven and configured by process models, many of the software systems that are used by organizations are not “process-aware”. In other words, they provide certain functions, but do not enforce a sequence or process in which those functions are to be used. Examples of these are accounting systems, logistics and inventory systems, manufacturing control systems, and many others.

However, what even most of those systems provide event logs: they record what activities their users have carried out what point in time. These event logs are the basic input to process mining. Process mining uses this information to

- Discover the process that has actually been followed
- Compare the discovered, actual process to a prescribed one in order to ensure compliance
- Evaluate the performance of the actual process
- Assess the compliance of the actual process with certain business rules
- Discover the distribution of work among human resources in the actual process

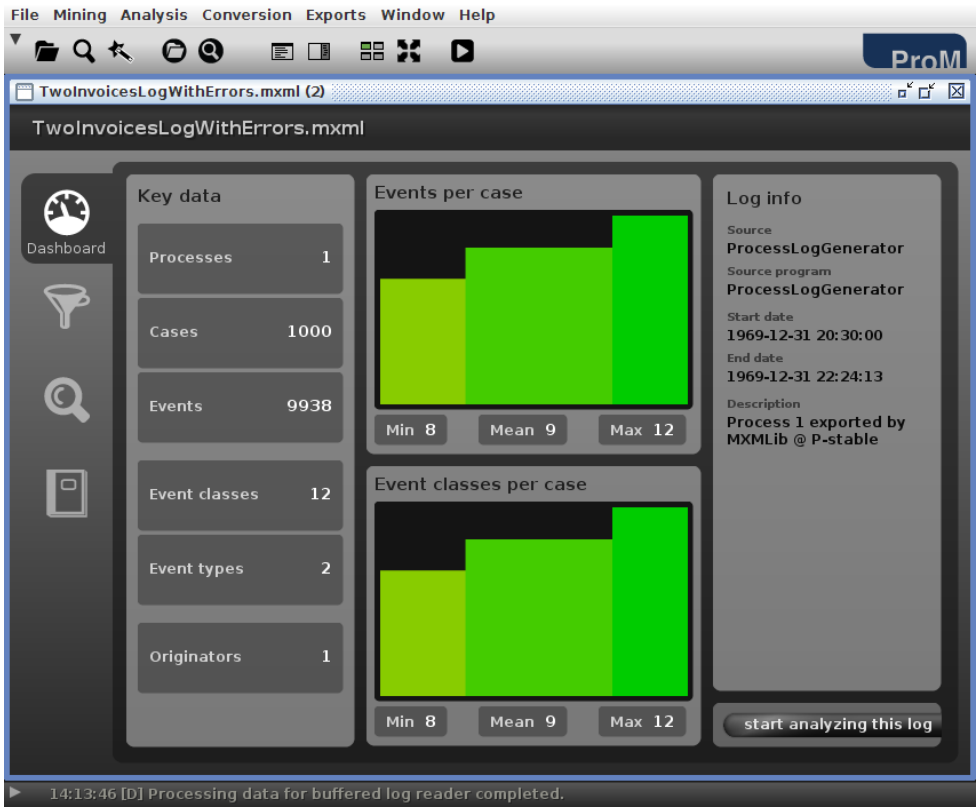
Of course, process mining can also be used with data from workflow management systems like YAWL. Once a process has been executed a large number of times, workflow mining can provide performance information or compliance information. Finally, process mining can be performed on simulated event logs to offer analysis methods that the simulation software may not provide.

In this tutorial, you will use a Process Mining software called ProM to analyze an example event log.

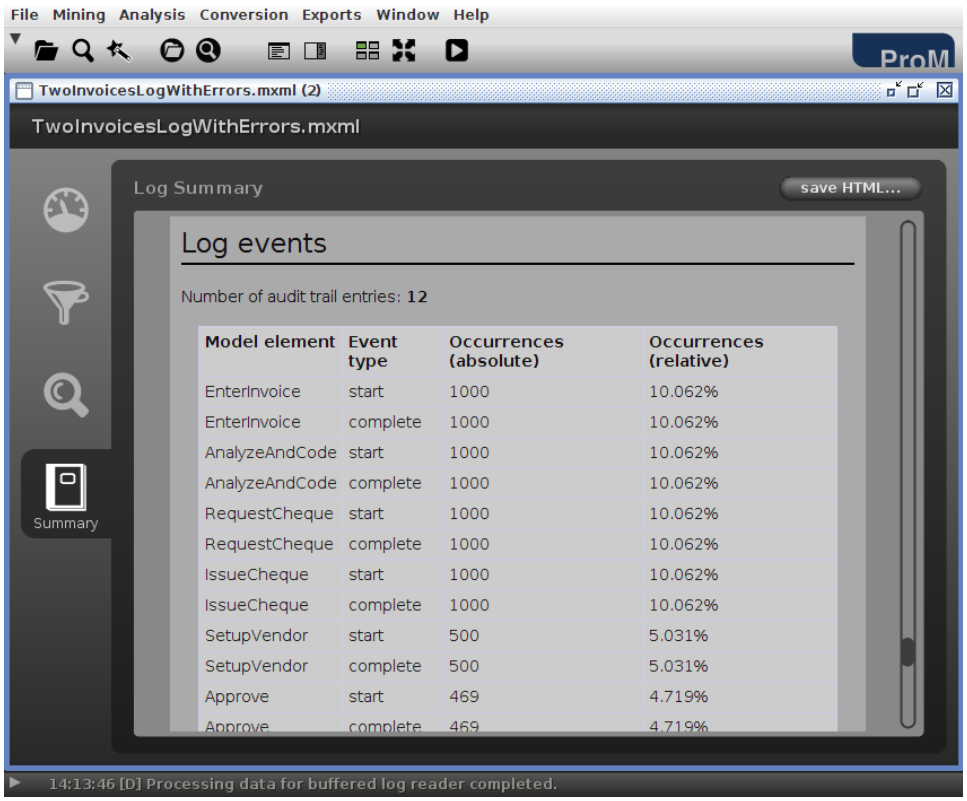
Exercise 0 (Preparation)

Start the ProM software using the icon on your desktop or the entry in the Start menu:

Download the file “EventLog.mxml” from the course web site to your computer and open the file with ProM: “File” ==> “Open supported file...”

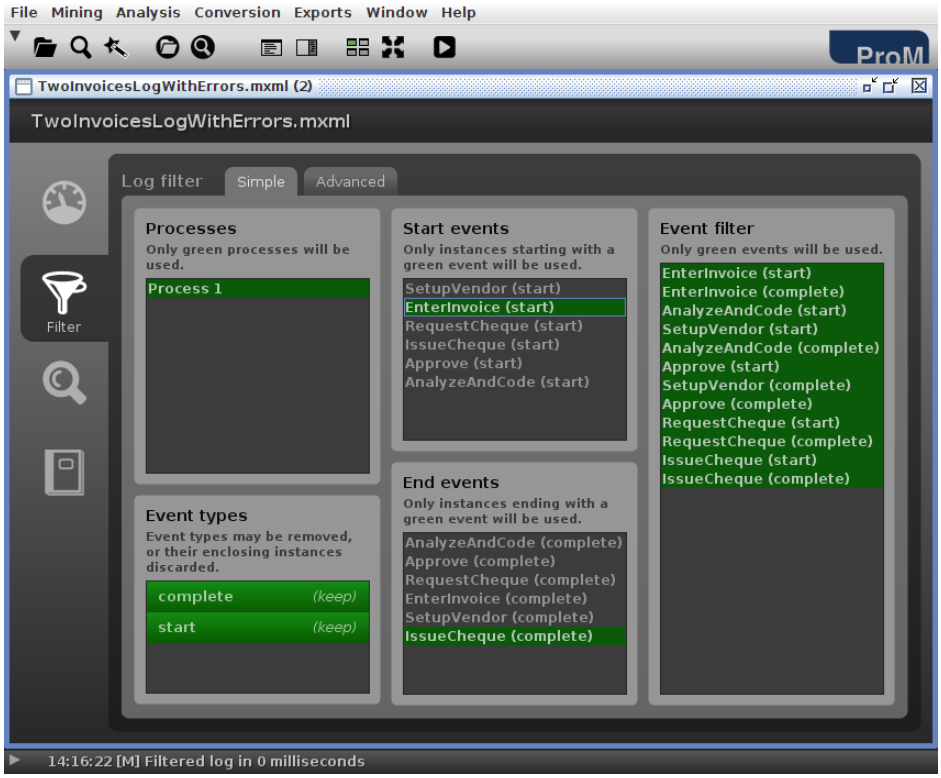


You can view basic information about the event log by selecting the “Summary” tab (icon looks like a notebook). In this case, you should be able to see that the event log contains information about 1000 process instances (“cases”) and the naming of the events should give you a clue as to what kind of business process we are looking at. Notice that some events do not occur for all processes.



Exercise 1 (Filtering)

Sometimes it is useful to filter out certain cases from the log, e.g. incomplete ones, or ones that do not appear to conform to recognized business practice. Selecting the “Filter” tab (icon looks like a funnel), you can see that events with which cases start and end. In this case, we know that the accounts payable process should start with entering an invoice, and end when the cheque issue is complete, so we can filter for those start and end events:



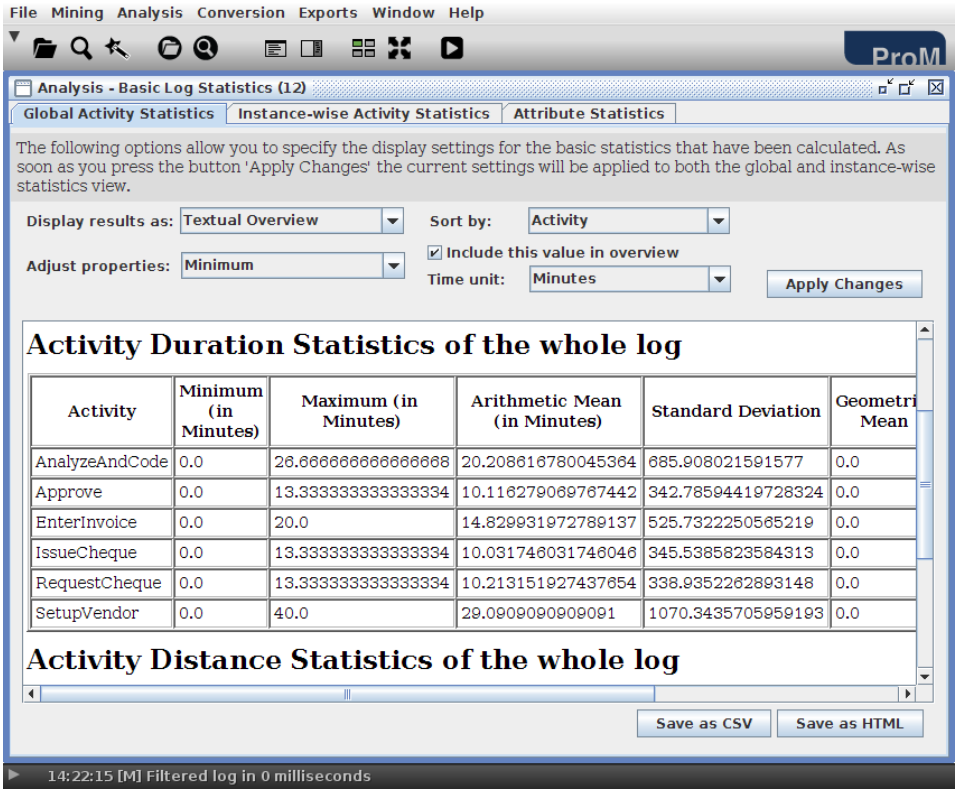
Exercise 2 (Basic Log Information)

One of the basic kinds of information that is business-relevant is how much time certain activities take, how variable the timing of the activities is, and what the shortest or longest time for activities is. Select “Analysis” ==> “Filtered EventLog ...” ==> “Basic Log Statistics”. The resulting table shows basic information from the log, which can also be shown in form of diagrams.

Questions:

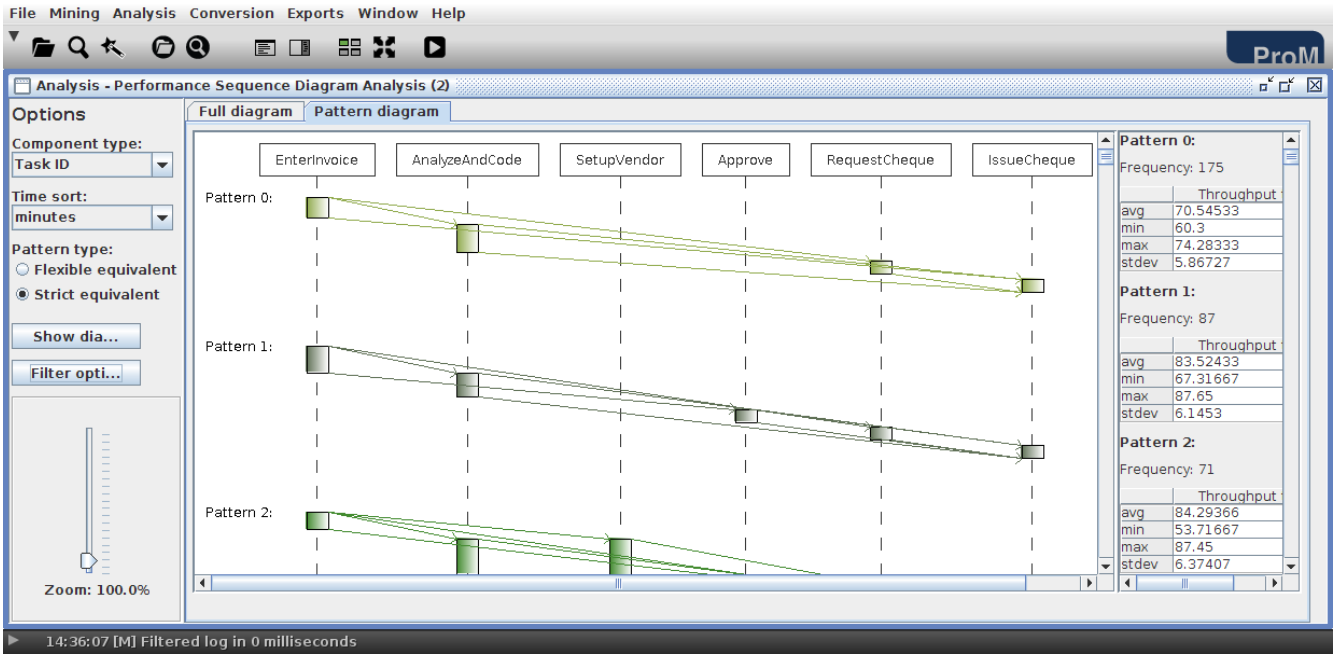
- Which activity takes longest? On average, how long does it take?
- The duration of which activity has the greatest variability?
  - Why is variability of activity durations bad? What could be the cause? What could a business do about it?

Close the log information window.



Exercise 3 (Identifying Patterns)

It is often useful to identify different sequences in which activities are carried out. To see this, select “Analysis” ==> “Filtered EventLog ...” ==> “Performance Sequence Diagram Analysis”. In the resulting window, chose the “Pattern Diagram” tab, select “minutes” for the time sort, select “Strict equivalent” for the pattern type, then click “Show Diagram”.



Questions:

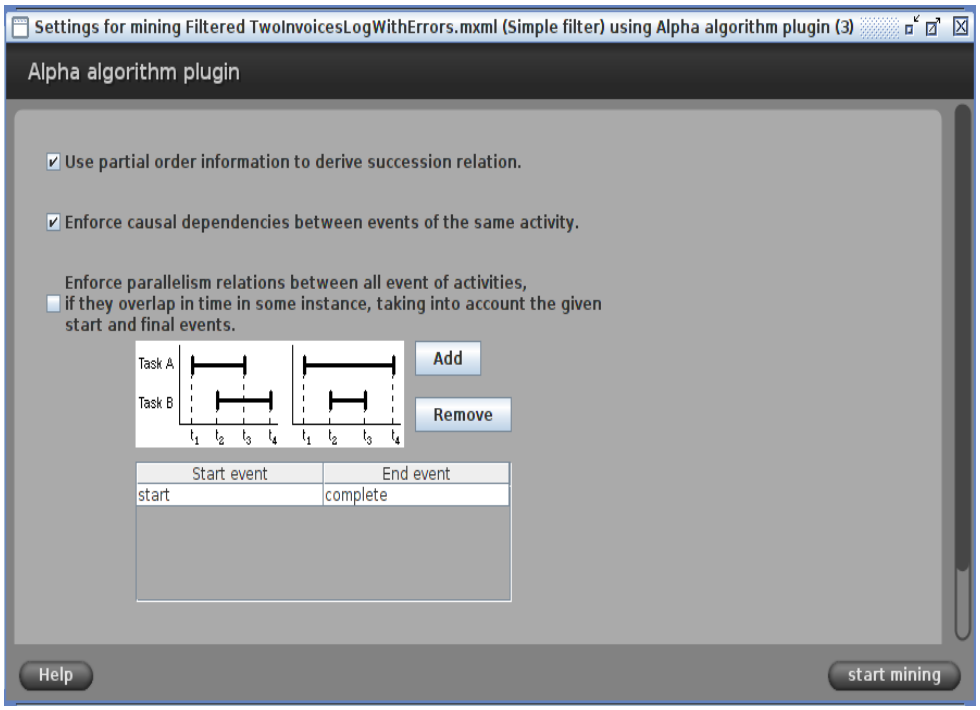
- How many different patterns of activity sequences are there?
  - Why do you think there are so many patterns?
- Which pattern is the most frequent one?

Close the analysis window.

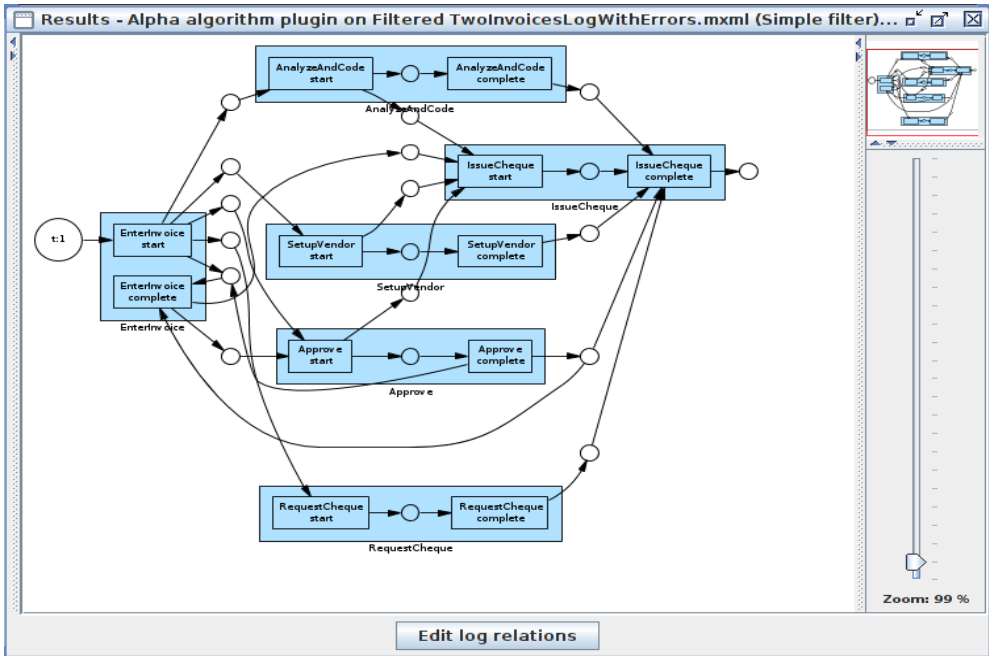
Exercise 4 (Process Mining)

Identifying different sequences of activities may sometimes be sufficient to identify the process that was followed. However, we can go a step further; we can ask what the general process model is that could have given rise to all of the observed activity sequence patterns. For this, we can use different notions of process miners. We begin with the basic "Alpha" process mining tool. Select "Mining" ==> "Filtered EventLog ..." ==> "Alpha algorithm plugin".

In the following dialog, unselect the final option, then click "start mining".



The result is a Petri Net model of a process with which all cases in the event log comply.

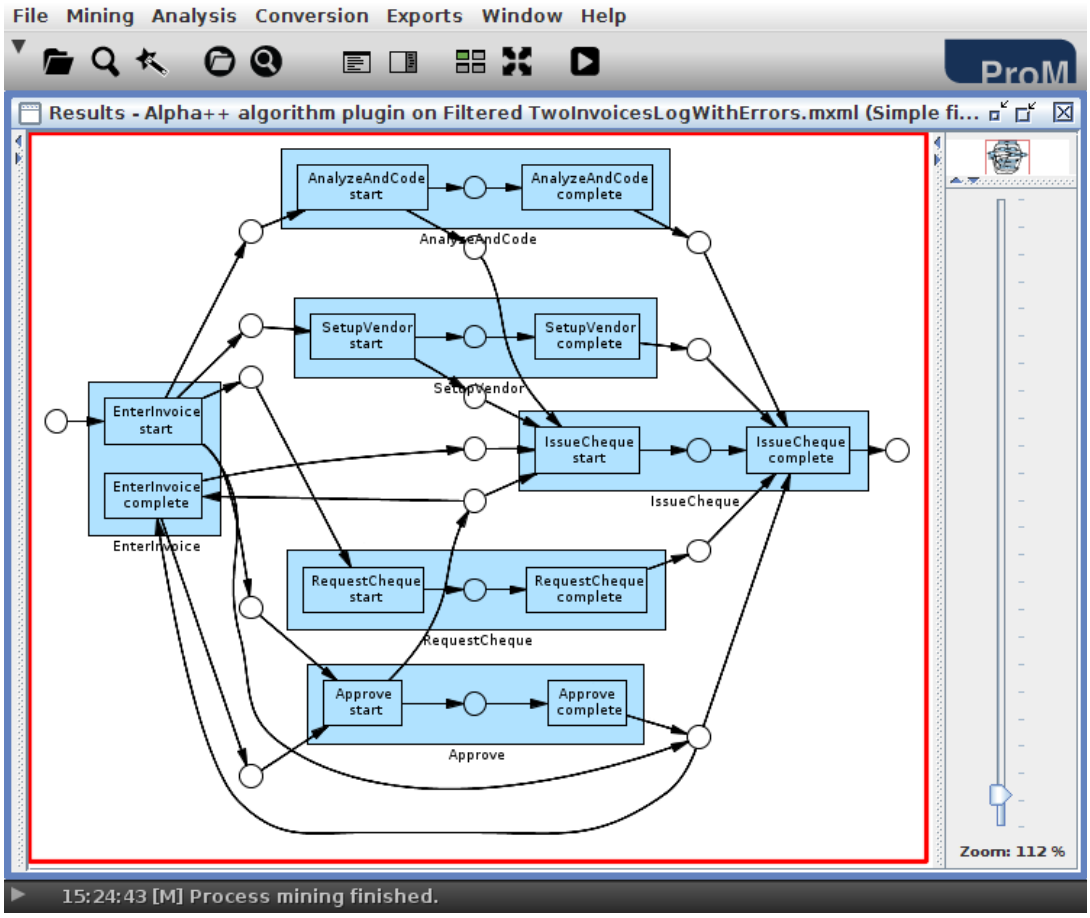


Question:

- Why do you think the Petri net model does not look like the one that you created earlier in the course?

Close the Petri net window and the Alpha miner window.

Another algorithm is the “Alpha++” algorithm, which improves on the original alpha algorithm by taking into account “noise” in the event log. In other words, it recognizes that some cases may be erroneous, for example have missing activities, etc.. Select “Mining” ==> “Filtered EventLog ...” ==> “Alpha++ algorithm plugin”. This algorithm also creates a Petri Net.

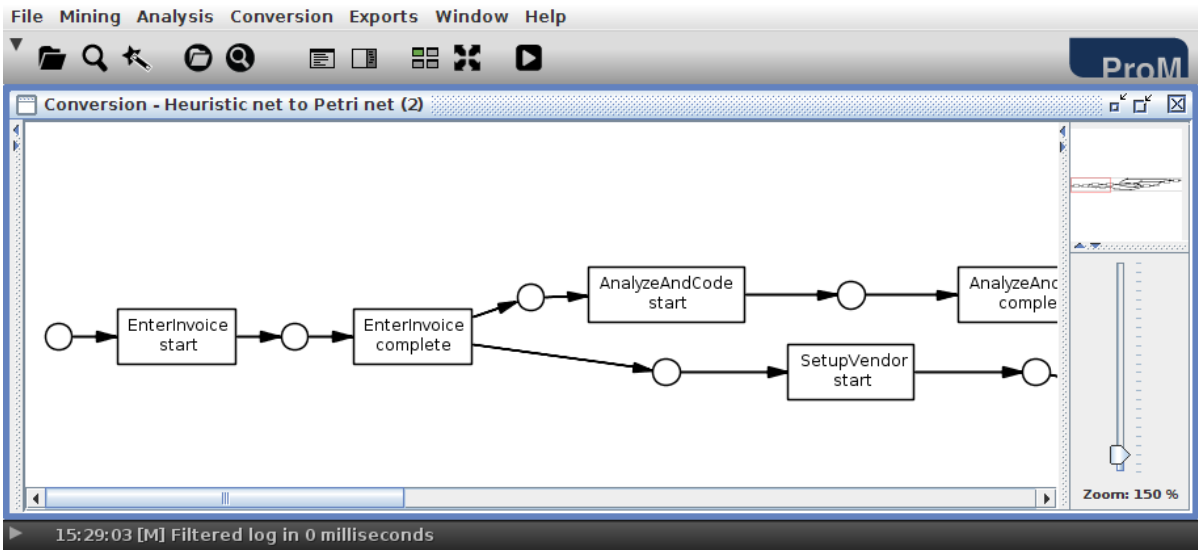


Question:

- Compare this Petri Net to the one you created earlier in the course, and to the one created by the "Alpha" algorithm above.
  - What differences do you notice?
  - How would you explain the differences

Close the Petri net window.

Finally, we use the "Heuristics Miner" to discover the process from the event log. Select "Mining" ==> "Filtered Event Log ... " ==> "Heuristics Miner". In the following window, click "start mining". The heuristics miner creates a "Heuristics net", which can be transformed to a Petri Net. Select "Conversion" ==> "Heuristics Net" ==> "Heuristics Net to Petri Net".



Question: Compare this Petri Net to the one you created earlier in the course, and to the one created by the "Alpha" algorithm above. What differences do you notice? How would you explain the differences?

Question: Given that you know the actual process, which of the three generated Petri Nets do think best reflects the process as you know it?

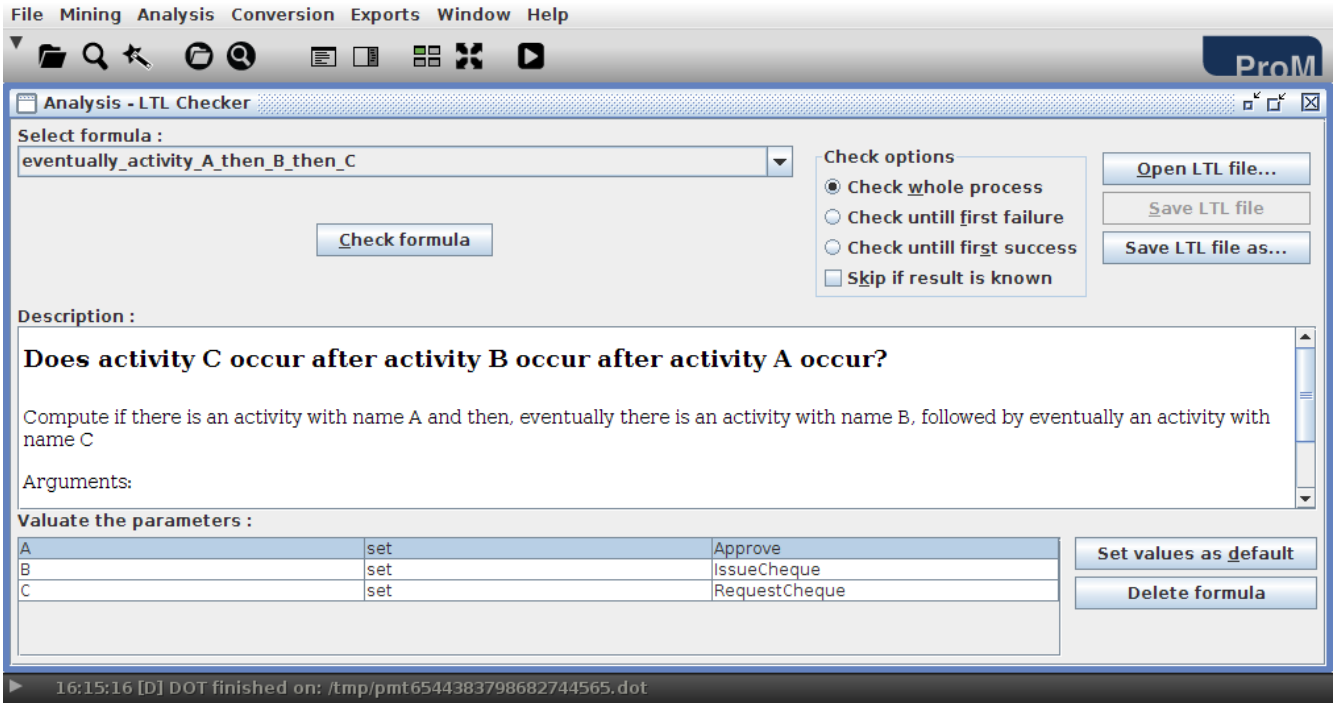
Close the Petri net window, the heuristics net window and the heuristics miner window.

Exercise 5 (Compliance Analysis)

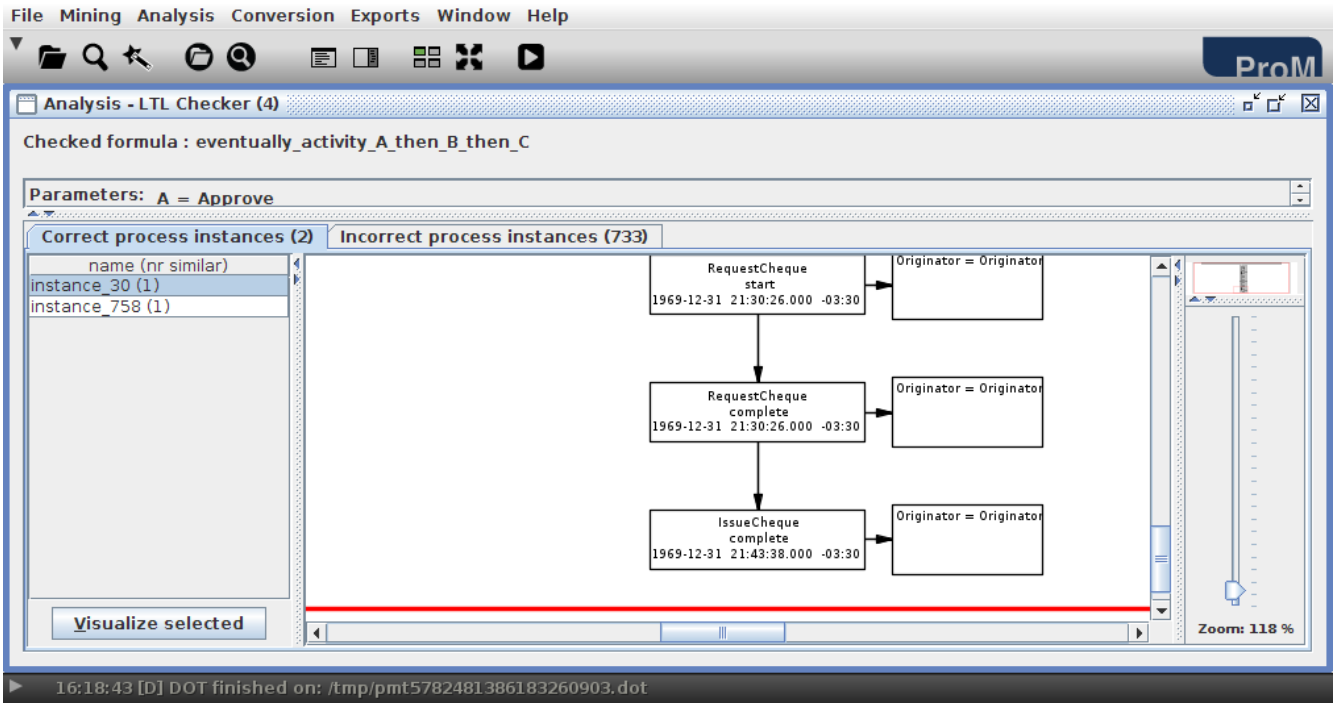
Oftentimes, it is necessary to show that a process actually behaves in a certain way and complies with some rules. Such rules may be certain accounting principles, quality control constraints or other business rules. In short, there are many cases where an organization has to demonstrate compliance. A useful tool for this is LTL ("Linear Temporal Logic").

To show how this works, we want to answer the question if there are any cases where an invoice has been approved, but the cheque has been issued before it was requested?

Select "Analysis" ==> "Filtered Event Log ... " ==> "LTL Checker". In the following window select the formula "eventually_activity_A_then_B_then_c". For the values of the parameters A, B, and C, fill in "Approve", "IssueCheque" and "RequestCheque" (correct spelling is important!!), as in the following picture, then click "Check formula".



The result shows two cases where this was the case. You can explore these further, by selecting each one and clicking “Visualize selected”.



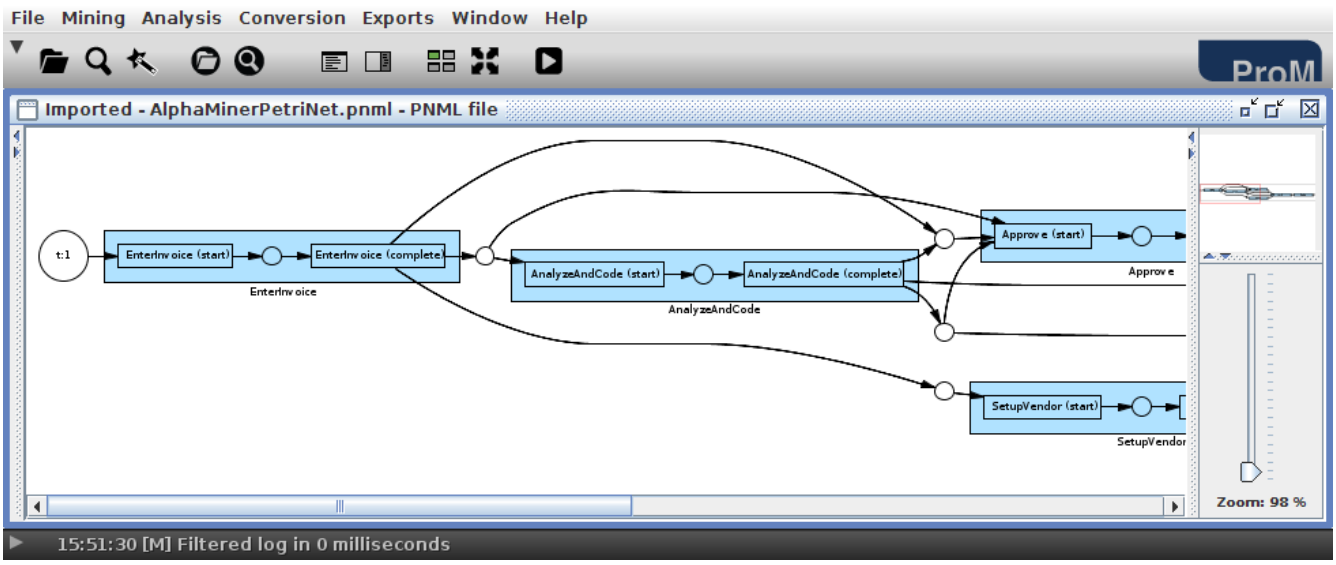
Close the visualization and the LTL window.

Exercise 6 (Performance Analysis)

For the performance analysis, we use an event log that does not contain any noise.

- In ProM, close the previous event log that you have worked with.
- Download the event log “EventLogNoNoise.mxml” from the course website and save it to your computer.
- Open the file with ProM: “File” ==> “Open supported file...”
- Select “Mining” ==> “Raw Event Log ... “ ==> “Alpha algorithm plugin”.
- In the following window, click “start mining”.

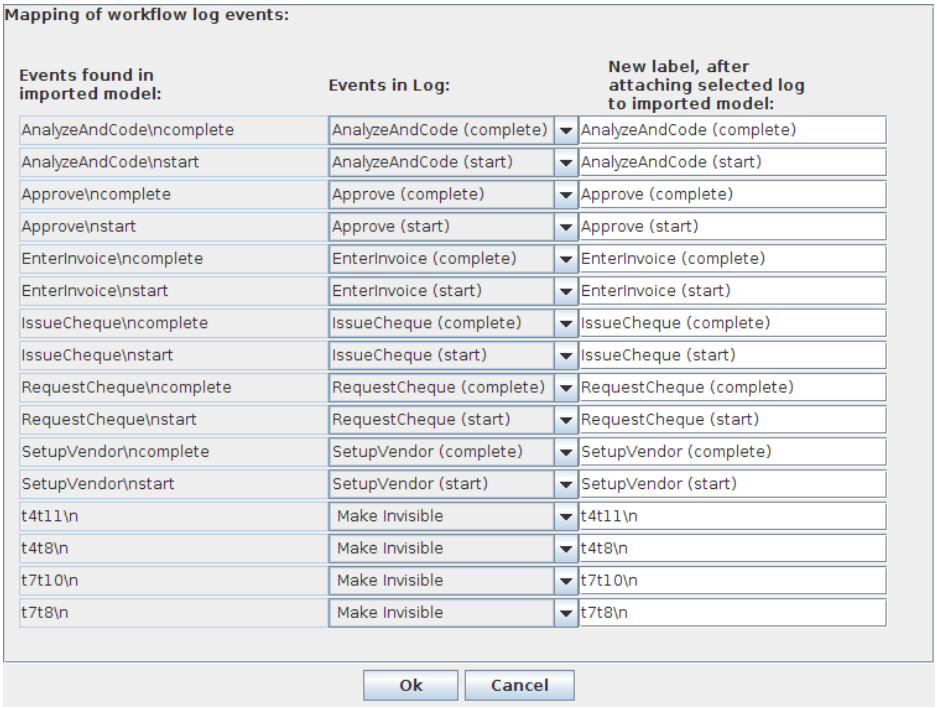
Without noise in the event log, the generated Petri Net looks a little bit more like the one generated by the heuristics miner, or the one that you created earlier in the course.



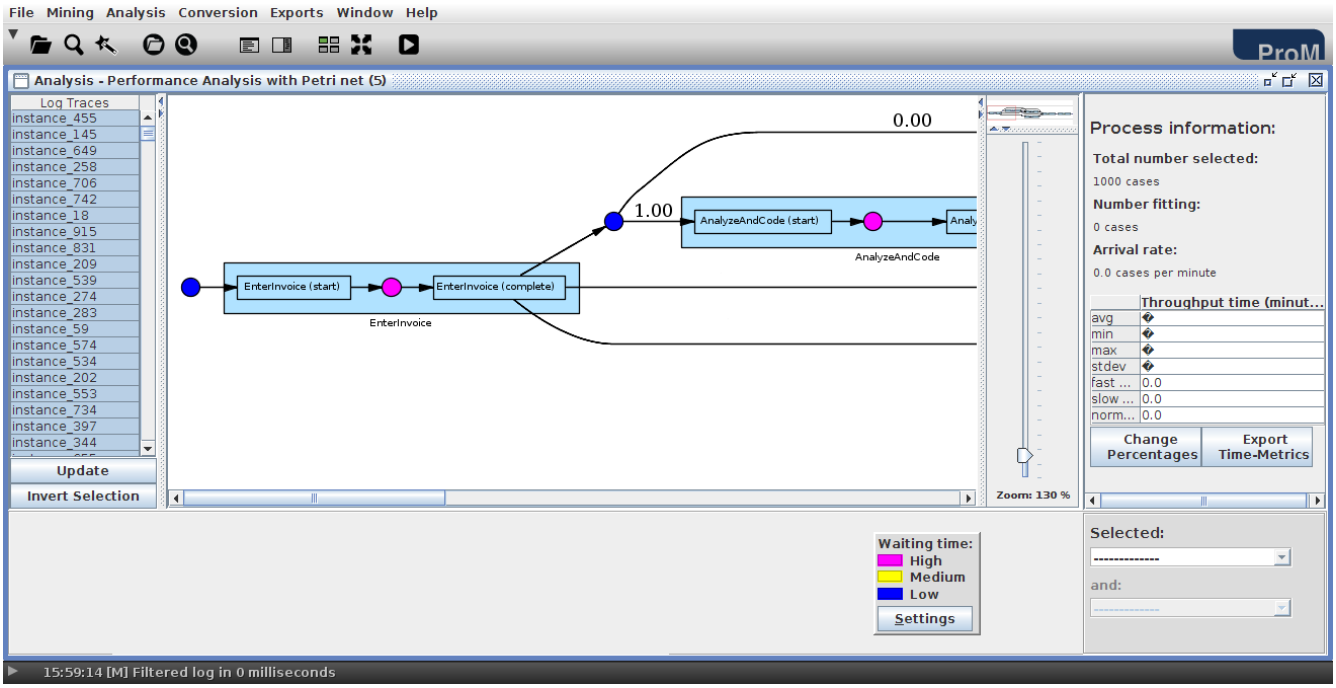
Save the Petri Net by selecting “Exports” ==> “Selected Petri Net” ==> “PNML 1.3.2 file”. Give the file a name and save it on your computer so you can find it again.

Close the Petri net window and the Alpha miner window.

Select the event log window and open the Petri Net file that you just saved, using “File” ==> “Open PNML File” ==> “With Raw EventLogNoNoise”. In the dialog box, select the Petri Net file that you just saved. The following dialog associates the transitions in the Petri Net with the events in the log, i.e. it matches names between the events in the log and the names of transitions in the petri net model. Simply click “OK”.



Select “Analysis” ==> “Selected Petri Net” ==> “Performance Analysis with Petri Net”. In the following dialog box, change the time measure to “minutes” and click “Start Analysis”. ProM now starts to play the cases in the event log through the Petri Net. This may take a few minutes to complete.



The places in the Petri Net are now color coded to indicate how long a case spends on that place before the following transition is fired. Thus, the places between tasks (e.g. between the “EnterInvoice (complete)” event and the “AnalyzeAndCode (start)”) represent the waiting times. The places within tasks (e.g. between the “EnterInvoice (start)” and “EnterInvoice (complete)” event) represent the time this task (e.g. “EnterInvoice”) takes.

The Petri Net is interactive and you can click on any place, any event, or select any two events to show frequency, arrival rate, and timing information. Explore the net and the data it contains.

Take Away Message:

This is not intended to be a complete coverage of process mining but to give you an impression of the kinds of tools that are available to analyze event logs. Looking at each of these, you should think about how they can be applied in a business context, when they are useful, and what business questions can be answered using process mining techniques. For more background, you should also read the textbook chapter on process mining (Chapter 17)