**Process mining of the Road Traffic Fine Management.**

**René in´t Veld** [Rene Intveld](https://www.futurelearn.com/courses/process-mining/4/steps/141404#user-options)

1. **Introduction**

In this peer assessment we are going to process mine the Road Traffic Fine Management as logged in the log file prepared for the FutureLearn course “[INTRODUCTION TO PROCESS MINING WITH PROM](https://www.futurelearn.com/courses/process-mining/4/todo/9080) -[EINDHOVEN UNIVERSITY OF TECHNOLOGY](https://www.futurelearn.com/partners/eindhoven-university-of-technology)”.

1. **Source**

We imported the file “Road\_Traffic\_Fine\_Management\_Process.xes” in ProM using the Naïve importer.

1. **Exploration of the log file.**

We use the *Log Visualiser* mode of ProM ( <https://github.com/reneintveld/process-mining/blob/master/fig1.jpg> ) to see that there are 150k cases, which consist of 560k events, on average 4 events per case. The minimum is 2 events, the maximum 20 events.

The events took place between 1-1-2000 and 18-6-2013.

In table 1 the kind of events are listed. We can see that there is a small subprocess of cases which involve the Prefecture. We could better filter this subprocess out to get a clear understanding.

*NB. The Inspector mode of the Log Visualiser is not working well on this big log and causes ProM to crash.*

The start event is always Create Fine. The class of the end event however can vary, see table 2. It seems that the process is not always registered to the completion of the process. Of course this could be caused by the limitation of the log (ending in June 2013), this is when we assume that the full process can take several years to complete. To investigate this we should compare the process starting in different years.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Class** | **Occurrences (absolute)** | **Occurrences (relative)** | | Create Fine | 150370 | 26,781% | | Send Fine | 103987 | 18,52% | | Add penalty | 79860 | 14,223% | | Insert Fine Notification | 79860 | 14,223% | | Payment | 77601 | 13,821% | | Send for Credit Collection | 59013 | 10,51% | | Insert Date Appeal to Prefecture | 4188 | 0,746% | | Send Appeal to Prefecture | 4141 | 0,738% | | Receive Result Appeal from Prefecture | 999 | 0,178% | | Notify Result Appeal to Offender | 896 | 0,16% | | Appeal to Judge | 555 | 0,099% | |

Table 1: Frequency of event classes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Class** | **Occurrences (absolute)** | **Occurrences (relative)** | | Payment | 67201 | 44,69% | | Send for Credit Collection | 58997 | 39,235% | | Send Fine | 20755 | 13,803% | | Send Appeal to Prefecture | 3144 | 2,091% | | Appeal to Judge | 134 | 0,089% | | Notify Result Appeal to Offender | 86 | 0,057% | | Receive Result Appeal from Prefecture | 53 | 0,035% | |

Table 2: Classes of end events.

Via the Explore Log Event mode of ProM ( <https://github.com/reneintveld/process-mining/blob/master/fig2.jpg> ) we can see that 45% of the cases only have 2 events, either Cre(ate fine) and Pay(ment) or Cre and Sen(d fine). The full description I have taken from the Event names.

It is remarkable that on one side we have events that have only create fine and payment. These fines might be paid on the spot? We have to investigate that later. On the other side we have fines for which a fine was sent, but for which no payment has been registered.

The most common process is Cre, Sen, Ins(ert fine notification), Add( penalty) and Sen(d for credit collection). Also in these cases no payment is registered. It might be that the credit collection uses a different system. To investigate this we need to see if ever a payment was made after Sending to credit collection. We can see that if we look at all the traces that Sending to credit collection is always the end state of the process, except a few cases where later an appeal was sent to the prefecture. This supports the assumption that credit collection is another process, which we cannot review with the current data.

From the Dotted Chart ( <https://github.com/reneintveld/process-mining/blob/master/fig3.jpg> ) we can see that the event Send for credit collection only takes place once a year, but is not registered in 2008! Furthermore it is obvious that the creation of fines is a stable process over time, in the sense that fines are created in a steady frequency.

1. **Filtering of the log file.**

The second step after ‘getting to know’ the data, is to apply a filtering. Usually real data is not ready to be mined. Some cases are not completed yet, different types of cases (with different processes) are combined, etc. Using the findings of the previous discussion, we recommend as a filtering to split the traces as follows:

* Traces which involve appeal to the prefecture (2%)
* Traces which end with credit collection (40%)
* Traces which end with payment (23%)
* Other (15%)

We will use only the plug-in “Filter Log using Simple Heuristics”, because this filter is able to produce the second and third subset.

When we look at the heuristic miner, one of the suggestions it makes is to filter out all traces ending on payment. We saved this filter as TrafficManagementPAY.

If we look only at events Create fine, Send fine and Payment, then we see in <https://github.com/reneintveld/process-mining/blob/master/fig4.jpg> that the creation of the fines is a quite regular process (maybe automated / cameras) but the sending of the fines is quite irregular. In some periods no fines are sent at all.

The payment follows sometime after the fines were sent, as can be seen in <https://github.com/reneintveld/process-mining/blob/master/fig5.jpg> .

Now we look at only traces ending on Sending to the credit collection. We saved this filter as

TrafficManagementCC. See <https://github.com/reneintveld/process-mining/blob/master/fig6.jpg> .

The sending of fines seems a quite regular process, but the inserting of a fine notification seems to be delayed during some periods. The sending to the credit collection is a very infrequent process, this happened in the beginning once a year (in January), but after 2007 it became quite irregular.

1. **Process discovery on real data**

In this peer assignment we discover a process model on the **Road\_Traffic\_Fine\_Management\_Process.xes.gz** event log. The task is to discover a process model and evaluate how well it describes the process (based on the event log). As input, we used 1 one of the 2 filtered versions TrafficManagementPAY in the previous step.

We used the heuristic miner because this gives the clearest picture of all the steps that could be taken and any short cuts in case of non-delayed payment. We used standard settings because the result was nice. See <https://github.com/reneintveld/process-mining/blob/master/fig7.jpg> .

.

Furthermore we checked the conformance of the discovered process.

First we converted the heuristic model into a PetriNet using the plug-in “Convert Heuristic Net into Petri Net”.

Then we took the petrinet and the event log to replay the process, using plug-in “Replay a log on Petri Net for Conformance Analysis”. However we were not able to perform this replay with ProM.

So, instead we mined a petrinet with the Inductive miner and replayed this model. Now we got a valid result, as can be seen in <https://github.com/reneintveld/process-mining/blob/master/fig8.jpg> .

The colours show how frequent steps were taken. The events Create fine and Payment were always in the trace, because of 1) Create fine is the start of the process and 2) the filter on processes ending with a payment.

There were no deviations from the model. This is because the model is quite general, allowing for several short cuts.