CHAPTER – 6

PROCESS MINING TOOLS

Processes are an integral part of today's world, driving services and internal functionalities in businesses, governmental bodies, and organizations around the globe. While there are plenty of systems available for supporting the execution of such processes, the current practices for monitoring and analyzing this execution in the organizational reality still leaves a lot to be desired. Process Mining is able to fill that gap, providing revolutionary means for the analysis and monitoring of real-life processes. The goal of process mining is to extract information about processes (i.e. a functional model) from event logs. To be able to perform process mining on event logs, some requirements need to be put on the event logs that serve as input.

The following requirements are mandatory and are assume to be met by all event logs that serve as input to process mining

![Diagram of event log fields](image)

Fig 6.1 - Mandatory fields of an event log.
General Event log structure

i. Each event refers to an activity or task, i.e. a well-defined step in the process (for instance ‘contact customer’).

ii. Each event refers to a case, i.e. a process instance (e.g. an insurance claim).

iii. Events are totally ordered.

iv. Each event refers to an originator (a person which executes the task).

v. Each event refers to a timestamp (i.e. the time at which it was recorded).

Such event logs are used as starting point for process mining.

Three different perspectives can be distinguished

6.1 ProM Framework

The goal of the first version of this framework was to provide a common basis for all kinds of process mining techniques, e.g., supporting the loading and filtering of event logs and the visualization of results. In 2004, the first fully functional version of ProM framework (ProM 1.1) was released. This version contained 29 plug-ins: 6 mining plug-ins (the classic $\alpha$ miner, the Tshinghua $\alpha$ miner, the genetic miner, the multi-phase miner, the social network miner, and the case data extraction miner), 7 analysis plug-ins (e.g., the LTL checker), 4 import plug-ins (e.g., plug-ins to load Petri nets and EPCs), 9 export plug-ins, and 3 conversion plug-ins (e.g., a plug-in to convert EPCs into Petri nets). Over time more plug-ins were added. For instance, ProM 4.0 (released in 2006) contained already 142 plug-ins. The ProM framework is a framework that was developed to support the various forms of process mining. ProM is an extensible framework that supports a wide variety of process mining techniques in the form of plug-ins. It is platform independent as it is implemented in Java (Wil M.P. van der Aalst, 2010).

ProM 5.2 was released in 2009. This version contained 286 plug-ins: 47 mining plug-ins, 96 analysis plug-ins, 22 import plug-ins, 45 export plug-ins, 44 conversion plug-ins, and 32 filter plug-ins. The spectacular growth of the number of plug-ins in the period from 2004 to 2009 illustrates that ProM realized its initial goal to provide a platform for the development of new process mining techniques. ProM has become the de facto standard for process mining. ProM 6 (released in November 2010) is based on XES rather than MXML. XES is the new process mining standard adopted by the IEEE Task Force on Process Mining (cf. Sect. 4.3).
Although ProM 5.2 was already able to load enormous event logs, scalability and efficiency were further improved by using OpenXES [48, 49]. ProM 6 can distribute the execution of plug-ins over multiple computers. This can be used to improve performance (e.g., using grid computing) and to offer ProM as a service. The user interface has been re-implemented to be able to deal with many plug-ins, logs, and models at the same time. Plug-ins are now distributed over so-called packages and can be chained into composite plug-ins. Packages contain related sets of plug-ins. ProM 6 provides a so-called package manager to add, remove, and update packages.
6.2 ProM Architecture

The basis for all process mining techniques is a process log. Such a log is a file generated by some information system, with information about the execution of a process. Since each information system has its own format for storing log files, we have developed a generic XML format for the ProM framework to store a log in. This format was based on a thorough comparison of the input needs of various existing (ad-hoc) process mining tools and the information typically contained in an audit trail or transaction log of some complex information system (e.g., an ERP or a WFM system).

![ProM Architecture Diagram](image)

**Fig 6.4 - ProM Architecture**

Another important feature of the ProM framework is that it allows for interaction between a large numbers of so-called plug-ins. A plug-in is basically the implementation of an algorithm that is of some use in the process mining area, where the implementation agrees
with the framework. Such plug-ins can be added to the entire framework with relative ease: Once the plug-in is ready it can be added to the framework. The ProM framework can read files in the XML format through the Log filter component. This component is able to deal with large data sets and sorts the events within a case on their timestamps before the actual mining starts. The framework allows plug-ins to operate on each other’s results in a standardized way. Typically, the mining results contain some kind of visualization, e.g., displaying a Petri net, an EPC or a Social network, or further analysis or conversion.

6.3 ProM Log Structure

The log files that can be accessed through the ProM framework need to be stored in the standard Mining eXtensible Markup Language (MXML) / XES log format.

6.3.1 MXML

MXML format is a generic XML format that has been developed for the ProM framework. The MXML log format was created in 2005 by Van Dongen and Van der Aalst. It was based on a thorough comparison of the required input of various existing (ad-hoc) process mining tools (e.g. EMiT, MinSoN, Little Thumb) and the information typically contained in the event logs of complex information systems (e.g. ERP or WFM systems). The goal of the format is to standardize the way of storing event log information. The main purpose of the format is to allow the application of analysis techniques on event logs.

```
<ProcessInstance id="Order 1" description="instance with Order 1">
  <Data>
    <Attribute name="totalValue"> 2142.38</Attribute>
  </Data>
  <AuditTrialEntry>
    <WorkflowModelElement>Create</WorkflowModelElement>
    <EventType>complete</EventType>
    <originator>Wil</originator>
    <timestamp>2009-01-03T15:30:00.000+01:00</timestamp>
    <Data>
      <Attribute name="current Value"> 2142.38</Attribute>
      <Attribute name="requestedBy"> Eric</Attribute>
      <Attribute name="supplier"> Fluxi Inc.</Attribute>
      <Attribute name="expectedDelivery"> 2009-01-12T12:00:00.000+01:00</Attribute>
    </Data>
  </AuditTrialEntry>
</ProcessInstance>
```

Fig 6.5 - Part of an MXML XML file with one trace and one event
The root element in this format is the Workflow Log element, which indicates that the log is a workflow log. Each Workflow log element contains an optional Data element, which can be used to store arbitrary textual data, an optional Source element, which can be used to store information about the source of this log (the information system that created it), and a number of Process elements (at least one), in which information about a specific process is stored. A Process element contains a number of Process Instance elements and an optional Data element, which again can be used to store arbitrary textual data. One Process Instance contains one case, i.e. one execution of the Process. In a Process Instance a number of events are described in Audit Trail Entry elements. An Audit Trail Entry consists of a Workflow Model Element (an activity or task ID), an Event Type, a (optional) Timestamp and an (optional) Originator (a person performing the event). Furthermore, additional information about the event may be stored in Data elements.
There exist various possible Event Types in an event log. An activity starts by being scheduled or by being skipped automatically (auto skip). When an activity has been scheduled, the control over that activity is put into the information system, which can then assign the activity to someone or to a group of persons. It is possible to reassign an assigned activity, which can be done by the information system, or by a user. A user can then start working on an activity that was assigned to him, or can decide to withdraw the activity or skip it (manual skip). This can be done before the activity was assigned even. A person that is working on an activity can suspend and resume it several times, but in the end he either has to complete or abort (pi abort) it. An activity can get aborted (pi abort) during its entire life cycle. Some information systems do not keep track of the type of event that occurred though. In such cases, each event is said to be of type normal.

6.3.2 XES

XES (eXtensible event Stream) is the successor of MXML. Based on many practical experiences with MXML, the XES format has been made less restrictive and truly extendible. In September 2010, the format was adopted by the IEEE Task Force on Process Mining. The format is supported by tools such as ProM (as of version 6), Nitro, XESame, and OpenXES. Each trace describes a sequential list of events corresponding to a particular case. The log, its traces, and its events may have any number of attributes. Attributes may be nested. There are five core types: String, Date, Int, Float, and Boolean. These correspond to the standard XML types: xs:string, xs:dateTime, xs:long, xs:double, and xs:boolean.

```
<trace>
  <string key="concept:name" value="Order 1"/>
  <float key="order:totalValue" value="2142.38"/>
  <event>
    <string key="concept:name" value="Create"/>
    <string key="lifecycle:transition" value="Complete"/>
    <string key="org.resource" value="Wil"/>
    <date key="time:timestamp" value="2009-01-03T15:30:00.000+01:00"/>
    <float key="order:currentValue" value="2142.38"/>
    <string key="details" value="Order creation details"/>
    <string key="requestedBy" value="Eric"/>
    <string key="supplier" value="Fluxi Inc."/>
    <date key="expectedDelivery" value="2009-01-12T12:00:00.000+01:00"/>
  </event>
</trace>
```

Fig 6.7 - Part of an XES XML file with one trace and one event
6.3.3 XES Meta model

XES does not prescribe a fixed set of mandatory attributes for each element (log, trace, and event); an event can have any number of attributes. However, to provide semantics for such attributes, the log refers to so-called extensions. An extension gives semantics to particular attributes. Users can define their own extensions. Each extension may define attributes that are considered to be standard when the extension is used. XES may declare particular attributes to be mandatory. A log holds two lists of global attributes: one for the traces and one for the events. XES supports the classifier concept. An XES log defines an arbitrary number of classifiers. Each classifier is specified by a list of attributes. Any two events that have the identical values with respect to these attributes are considered to be equal for that classifier. These attributes should be mandatory event attributes.
Three classifiers are defined in the XES. Classifier Activity classifies events based on the concept:name attribute. Classifier Resource classifies events based on the org:resource attribute. Classifier Both classifies events based on two attributes.

To bridge the gap between the log formats of the various information systems and the standard MXML/XES log format, another tool: the ProM-import framework was developed. This tool makes it possible to convert the logs of the various types of information systems to the standard MXML log format.

(a) ProM Import

The ProM Import Framework has been designed for converting event log data to MXML. It is designed as a developer-friendly framework.

![ProM Import Framework](image)

ProM Import Framework allows to extract process enactment event logs from a set of information systems. These can be exported in the MXML format. ProMimport is a tool supporting the conversion of different data sources to MXML, e.g., MS Access, Aris PPM, CSV, Apache, Adept, PeopleSoft, Subversion, SAP R/3, Protos, CPN Tools, Cognos, and Staffware.
(b) Nitro

Fig 6.10 – Nitro Event Log Statistical Analysis

(c) Disco

Disco has been designed to make the data import easy for automatic timestamp detection, considering the configuration setting and load the data sets.

Fig 6.11 – Disco Performance Map of an Event Log
A CSV or an Excel file can be configured with the columns relevant with the case ID, timestamp, activity name with other attributes to be included in the analysis. Disco is compatible with the academic toolsets ProM5 and ProM6. Disco can be used for importing and exporting the standard event log format MXML or XES. Disco also features a shortcut import and data exchange for previously imported datasets with up to 200x speed-up for very large datasets through the native FXL disco log file format.

6.4 Plug-Ins

As described in the ProM framework, there are 5 types of plug-ins

a. **Mining plug-ins** which implements some mining algorithm, e.g., mining algorithms that construct a Petri net based on some event log.

b. **Export plug-ins** which implements some “save as” functionality for some objects (such as graphs). For example, there are plug-ins to save EPCs, Petri nets (e.g., in PNML format [7]), spreadsheets, etc.

c. **Import plug-ins** which implements an “open” functionality for exported objects, e.g., load instance-EPCs from ARIS PPM.

d. **Analysis plug-ins** which typically implements some property analysis on some mining result. For example, for Petri nets there is a plug-in which constructs place invariants, transition invariants, and a coverability graph.

- **Verification** of process models (e.g., Woflan analysis)
- Verification of **Linear Temporal Logic (LTL)** formulas on a log
- Checking the **conformance** between a given process model and a log
- **Performance analysis** (Basic statistical analysis, and Performance Analysis with a given process model)

e. **Conversion plug-ins** which implements conversions between different data formats, e.g., from EPCs to Petri nets.

Finally, ProM sports a large array of **log filters**, which are a valuable tool for cleaning logs from undesired, or unimportant, artifacts.
Table 6.1 - Process Mining Plug-ins in ProM 6

<table>
<thead>
<tr>
<th>Plug In</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha Miner</td>
<td>Discovers a Petri Net using the alpha-algorithm.</td>
</tr>
<tr>
<td>Heuristic miner</td>
<td>Discovers a C-net using the heuristic algorithm.</td>
</tr>
<tr>
<td>Genetic miner</td>
<td>Discovers a C-net using genetic mining.</td>
</tr>
<tr>
<td>Fuzzy miner</td>
<td>Discovers a fuzzy model using fuzzy mining.</td>
</tr>
<tr>
<td>Transition system miner</td>
<td>Discovers a transition system based on a state representation function and a log.</td>
</tr>
<tr>
<td>Declare miner</td>
<td>Discovers a declare model.</td>
</tr>
<tr>
<td>ILP miner</td>
<td>Discovers a petri net using language based regions.</td>
</tr>
<tr>
<td>Simple log filter</td>
<td>Filtering a log by answering simple questions.</td>
</tr>
<tr>
<td>Dotted chart analysis</td>
<td>Creates a dotted chart but now events are aligned based on their context rather than time.</td>
</tr>
<tr>
<td>Trace alignment</td>
<td>Similar to dotted chart, but now events are aligned based on their context rather than time.</td>
</tr>
<tr>
<td>Guide tree miner</td>
<td>Clusters cases in a tree based on similarities.</td>
</tr>
<tr>
<td>Social network miner</td>
<td>Creates a social network based on a selected criterion.</td>
</tr>
<tr>
<td>LTL checker</td>
<td>Checks a property expressed in terms of LTL.</td>
</tr>
<tr>
<td>Fitness</td>
<td>Computes fitness of petri net based on event log.</td>
</tr>
<tr>
<td>ETConformance</td>
<td>Checks conformance by counting “escaping edges” from the state space of the log to the state space of the model.</td>
</tr>
<tr>
<td>Replay log on flexible model</td>
<td>Conformance checker based on A* algorithm ant it can also be applied to petri nets, C nets and YAWL models.</td>
</tr>
<tr>
<td>PomPom</td>
<td>Automatically abstracts from infrequently visited parts of a Petri net.</td>
</tr>
<tr>
<td>Transition system analyser</td>
<td>Creates a model to predict the remaining flow time.</td>
</tr>
</tbody>
</table>

6.5 SUMMARY

This chapter discussed about the open source process mining tool ProM and its architecture. It also discussed XES Meta Model and ProM import framework. The plug-ins of ProM6.2 was listed the proper description.