

CHAPTER TWO :

BASIC CONCEPTS OF PROCESS MINING

1. introduction :

As already mentioned earlier, the process mining is a collection of domains and, in this chapter, we going to highlight the minimum requirements for start to apply the process mining

2.Process Model :

The goal of a process model is to decide which activities need to be executed and in what order. Activities can be executed sequentially, activities can be optional or concurrent, and the repeated execution of the same activity may be possible.

3.Play-In, Play-Out, and Replay :

Process mining function on make relation between process model and event log using three terms Play-In, Play-Out, and Replay to reflect on this relation.

Play out : when we start from process model to get the event data .(fig 2.1)

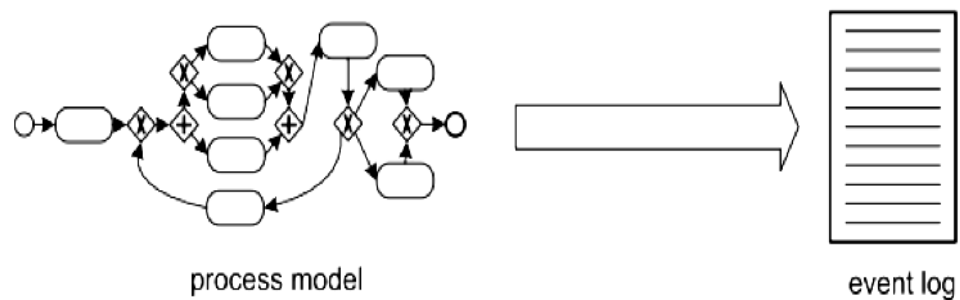


fig 2.1 the play out technique

play in : Play-In is the opposite of Play-Out , when we start from event data to get the process model .(fig 2.2)

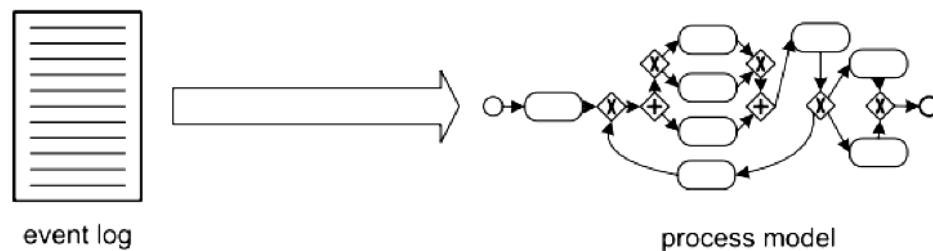
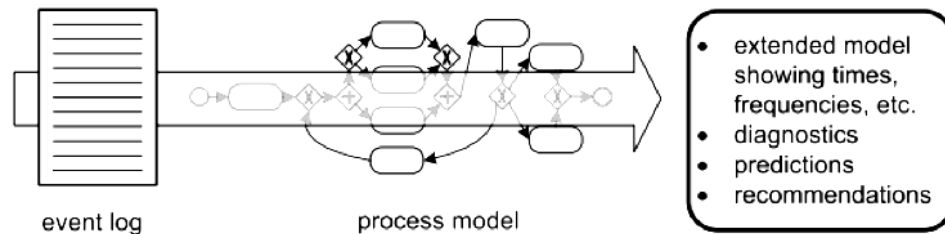


fig 2.2 the play in technique

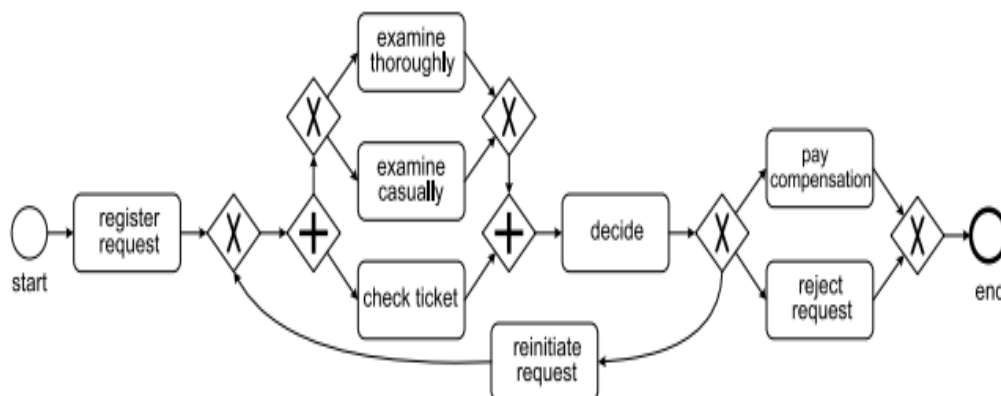
Replay : uses an event log and a process model as input. The event log is “replayed” on top of the process model ,mixture of play in and play on (fig 2,3)

**fig 2.3** the play in technique

Note : in process model we use (play in) in discovery step and (replay)for the the others steps

4.Business Process Modeling Notation (BPMN)

Recently, the Business Process Modeling Notation (BPMN) has become one of the most widely used languages to model business processes. BPMN is supported by many tool vendors and has been standardized . Figure 2.4 shows the BPMN model . Figure 2.5 shows a small subset of all notational elements. Atomic activities are[1] called tasks. Like in YAWL activities can be nested. Most of the constructs can be easily understood after the introduction to YAWL. A notable difference is that the

**fig 2.4** : Process model using the BPMN notation [1]

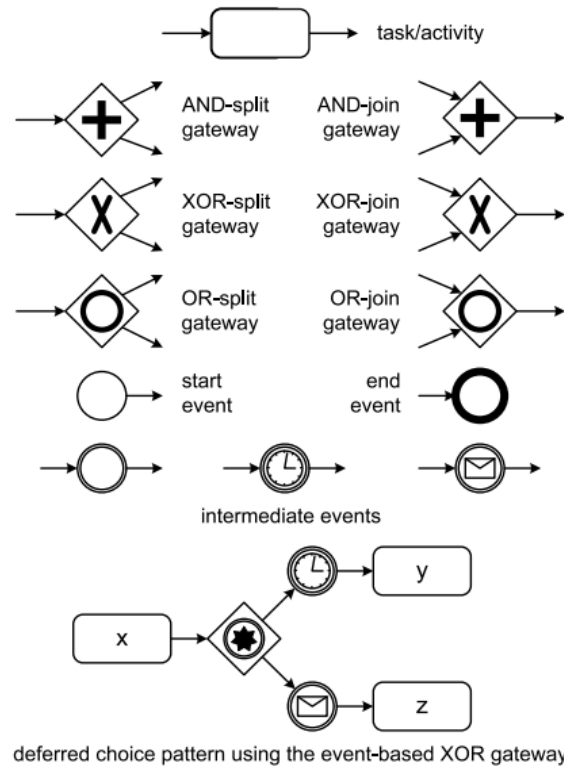


fig 2.5 : BPMN notation [2]

Routing logic is not associated with tasks but with separate gateways. Figure 2.5 shows that there are split and join gateways of different types: AND, XOR, OR. The splits are based on data conditions. An event is comparable to a place in a Petri net. However, the semantics of places in Petri nets and events in BPMN are quite different. There is no need to insert events in-between activities and events cannot have multiple input or output arcs. Start events have one outgoing arc, intermediate events have one incoming and one outgoing arc, and end events have one incoming arc. [1] Unlike in YAWL or a Petri net, one cannot have events with multiple incoming or outgoing arcs; splitting and joining needs to be done using gateways.

To model the so-called deferred choice workflow pattern one needs to use the event-based XOR gateway shown in Fig 2.5. [2].

5. Petri Nets

The goal of process discovery is to distil a process model from some event log. Here we use Petri nets to represent such models. In fact, we extract a subclass of Petri nets known as workflow nets (WF-nets).

Definition 2.1 : An Petri net is a tuple $(P; T; F)$ where:

1. P is a finite set of places,
2. T is a finite set of transitions such that $P \cap T = \emptyset$, and

3. $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, called the flow relation.

An example Petri net is shown in Figure 2.6. This Petri net has six places represented by circles and four transitions represented by squares. Places may contain tokens. For example, in Figure 2.6 both p_1 and p_6 contain one token, p_3 contains two tokens, and the other places are empty. The state, also called marking, is the distribution of tokens over places. A marked Petri net is a pair $(N; M)$, where $N = (P; T; F)$ is a Petri net and where $M \in IB(P)$ is a bag over P denoting the marking of the net. The initial marking of the Petri net shown in Figure 2.6 is $[p_1; p_3^2; p_6]$. The set of all marked Petri nets is denoted $N \cdot [1]$

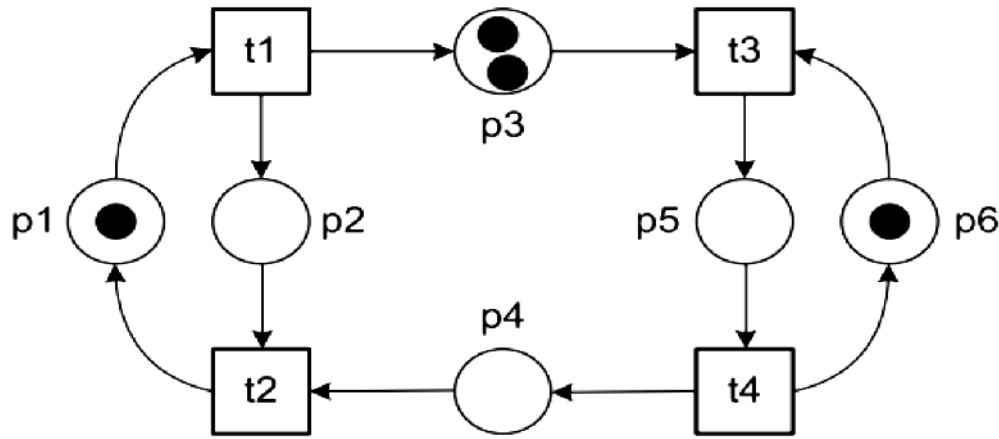


fig 2.6 A Petri net with six places (p_1, p_2, p_3, p_4, p_5 , and p_6) and four transitions (t_1, t_2, t_3 , and t_4).[4]

Let $N = (P; T; F)$ be a Petri net. Elements of $P \cup T$ are called nodes. A node x is an input node of another node y iff there is a directed arc from x to y (i.e., $(x; y) \in F$). Node x is an output node of y iff $(y; x) \in F$. For any $x \in P \cup T$, $\bullet x = \{y \mid (y; x) \in F\}$ and $x \bullet = \{y \mid (x; y) \in F\}$. In Figure 3, $\bullet t_3 = \{p_3, p_6\}$ and $t_3 \bullet = \{p_5\}$. The dynamic behavior of such a marked Petri net is defined by the so-called firing rule. A transition is enabled if each of its input places contains a token. An enabled transition can fire thereby consuming one token from each input place and producing one token for each output place.[3]

Definition 2.2 : (Firing rule). Let $(N; M)$ be a marked Petri net with $N = (P; T; F)$. Transition $t \in T$ is enabled, denoted $(N; M)[t]$, iff $\bullet t \leq M$. The firing rule $\rightarrow \subseteq N \times T \times N$ is the smallest relation satisfying for any $(N; M) \in N$ and any $t \in T$, $(N; M)[t] \rightarrow (N; (M - \bullet t) + t \bullet)$. [2]

In the marking shown in Figure 2.6, both t_1 and t_3 are enabled. The other two transitions are not enabled because at least one of the input places is empty. If t_1 fires, one token is consumed (from p_1) and two tokens are produced (one for p_2 and one for p_3). Formally, $(N; [p_1; p_3^2; p_6])[t_1] \rightarrow (N; [p_2; p_3^3; p_6])$. So the resulting marking is $[p_2; p_3^3; p_6]$. If t_3 fires in the

initial state, two tokens are consumed (one from p_3 and one from p_6) and one token is produced (for p_5). Formally, $(N; [p_1; p_3; p_6]) \xrightarrow{t_3} (N; [p_1; p_3; p_5])$. Let $(N; M_0)$ with $N = (P; T; F)$ be a marked P/T net. A sequence $\sigma \in T^*$ is called a firing sequence of $(N; M_0)$ iff, for some natural number $n \in \mathbb{N}$, there exist markings $M_1; \dots; M_n$ and transitions $t_1; \dots; t_n \in T$ such that $\sigma = t_1 \dots t_n$ and, for all i with $0 \leq i < n$, $(N; M_i) \xrightarrow{t_{i+1}}$ and $(N; M_i) \xrightarrow{t_{i+1}} (N; M_{i+1})$. Let $(N; M_0)$ be the marked Petri net shown in Figure 2.6, i.e., $M_0 = [p_1; p_3; p_6]$. The empty sequence $\sigma = \epsilon$ is enabled in $(N; M_0)$. The sequence $\sigma = t_1; t_3$ is also enabled and results in marking $[p_2; p_3; p_5]$. Another possible firing sequence is $\sigma = t_3; t_4; t_3; t_1; t_4; t_3; t_2; t_1$. A marking M is reachable from the initial marking M_0 iff there exists a sequence of enabled transitions whose firing leads from M_0 to M . The set of reachable markings of $(N; M_0)$ is denoted. $[N; M_0]$. [2]

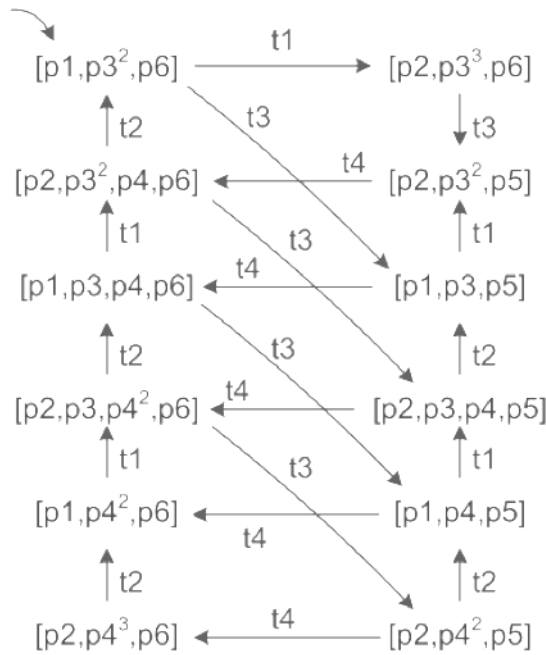


Fig. 2.7. The reachability graph of the marked Petri net shown in Figure 1[5]

For the marked Petri net shown in Figure 2.6 there are 12 reachable states. These states can be computed using the so-called reachability graph shown in Figure 2.7. All nodes correspond to reachable markings and each arc corresponds to the firing of a particular transition. Any path in the reachability graph corresponds to a possible firing sequence. For example, using Figure 2.7 it is easy to see that $t_3; t_4; t_3; t_1; t_4; t_3; t_2; t_1$ is indeed possible and results in $[p_2; p_3; p_4; p_5]$. [1] Marked net may be unbounded, have an infinite number of reachable states. In this

case, the reachability graph is infinitely large, but one can still construct the so-called coverability graph .

6.Workflow Nets

For process discovery, we look at processes that are instantiated multiple times, the same process is executed for multiple cases. For example, the process of handling insurance claims may be executed for thousands or even millions of claims. Such processes have a clear starting point and a clear ending point. Therefore, the following subclass of Petri nets (WFnets) is most relevant for process discovery.

Definition 2.3 : (Workflow nets). Let $N = (P; T; F)$ be a Petri net and t a fresh identifier not in $P \cup T$. N is a workflow net (WF-net) iff:

1. object creation: P contains an input place i (also called source place) such that $\bullet i = 0$;
2. object completion: P contains an output place o (also called sink place) such that $o \bullet = 0$;
3. connectedness: $N^- = (P; T \cup \{t\}; F \cup \{f(o; t)\}; (t; i)\}$ is strongly connected, i.e., there is a directed path between any pair of nodes in N^- .

Clearly, Figure 2.6 is not a WF-net because a source and sink place are missing.

Figure 2.8 shows an example of a WF-net: $\bullet \text{start} = 1$, $\text{end} \bullet = 1$, and every node is on a path from start to end.[3]

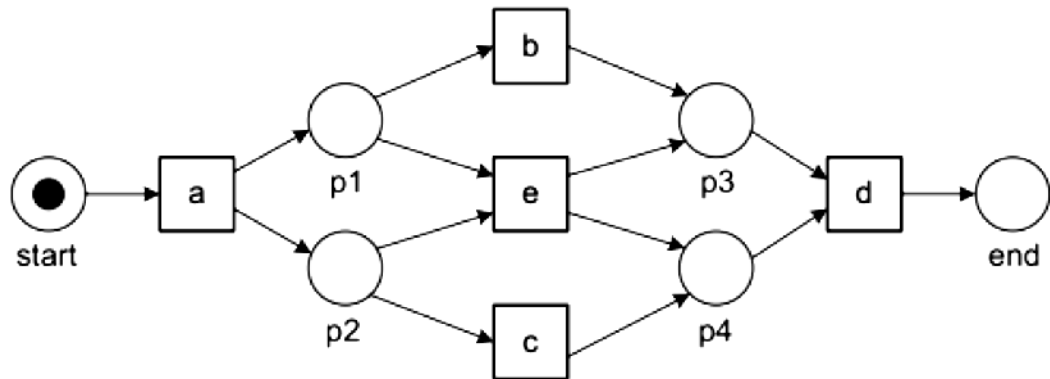


Fig.2.8. A workflow net with source place $i = \text{start}$ and sink place $o = \text{end}$ [3]

Not every WF-net represents a correct process. For example, a process represented by a WF-net may exhibit errors such as deadlocks, tasks which can never become active, livelocks, garbage being left in the process after termination, etc. Therefore, we define the following correctness criterion[1]

Definition2.4 : (Soundness).

Let $N = (P; T; F)$ be a WF-net with input place i and output place o . N is sound iff:

1. safeness: $(N; [i])$ is safe, i.e., places cannot hold multiple tokens at the same time,
2. proper completion: for any marking $M \in [N; [i]]$, $o \in M$ implies $M = [o]$,
3. option to complete: for any marking $M \in [N; [i]]$, $[o] \in [N; M]$, and
4. absence of dead tasks: $(N; [i])$ contains no dead transitions (i.e., for any $t \in T$, there is a firing sequence enabling t).[3]

The WF-net shown in Figure 2,6 is sound. Soundness can be verified using standard Petri-net-based analysis techniques. In fact soundness corresponds to liveness and safeness of the corresponding short-circuited net . This way efficient algorithms and tools can be applied. An example of a tool tailored towards the analysis of WF-nets is Woflan . This functionality is also embedded in our process mining tool ProM .[2]

7.YAWL :

YAWL is both a workflow modeling language and an open-source workflow system . The acronym YAWL stands for “Yet Another Workflow Language”. The development of the YAWL language was heavily influenced by the Workflow Patterns Initiative mentioned earlier. Based on a systematic analysis of the constructs used by existing process modeling notations and workflow languages, a large collection of patterns was identified. These patterns cover all workflow perspectives, i.e., there are control-flow patterns, data patterns, resource patterns, change patterns, exception patterns, etc. The aim of YAWL is to offer direct support for many patterns while keeping the language simple. It can be seen as a reference implementation of the most important workflow patterns. Over time, the YAWL language and the YAWL system have increasingly become synonymous and have garnered widespread interest from both practitioners and the academic community alike. YAWL is currently one of the most widely used open-source workflow systems.[1]

Here we restrict ourselves to the control-flow perspective. Figure 2.9 shows the main constructs. Each process has a dedicated start and end condition, like in WF- nets. Activities in YAWL are called tasks. Conditions in YAWL correspond to places in Petri nets. However, it is also possible to directly connect tasks without putting a condition in-between. Tasks have—depending on their type—a well-defined split and join semantics. An AND-join/AND-split task behaves like a transition, i.e., it needs to consume one token via each of the incoming arcs and produces a token along each of the outgoing arcs. An XOR-split selects precisely one of its outgoing arcs. The selection is based on evaluating data conditions. Only one token is produced and sent along the selected arc. An XOR-join is enabled once for every incoming

token and does not need to synchronize. An OR-split selects one or more of its outgoing arcs. This selection is again based on evaluating data conditions. Note

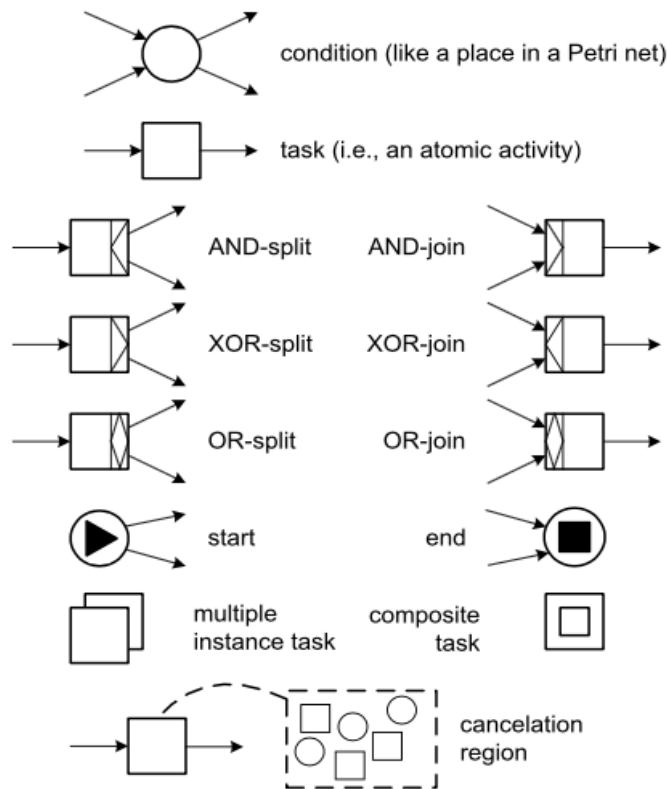


Figure 2.9 YAWL notation[1]

That an OR-split may select 2 out of three 3 outgoing arcs. The semantics of the *OR-join* are more involved. The OR-join requires at least one input token, but also synchronizes tokens that are “on their way” to the OR-join. As long as another token may arrive via one of the ingoing arcs, the OR-join waits. YAWL also supports *cancellation regions*. A task may have a cancellation region consisting of conditions, tasks, and arcs. Once the task completes all tokens are removed from this region. Note that tokens for the task’s output conditions are produced after emptying the cancellation region. YAWL’s cancellation regions provide a powerful mechanism to abort work in parallel branches and to reset parts of the workflow. Tasks in a YAWL model can be *atomic* or *composite*. A composite task refers to another YAWL model. This way models can be structured hierarchically. Atomic and composite tasks can be instantiated multiple times in parallel. For example, when handling a customer order, some tasks needs to be executed for every order line. These order lines can be processed in any

order. Therefore, a loop construct is less suitable. Figure 2.9 shows the icon for such a multiple instance task and all other constructs just mentioned. Figure 2.10 shows an example YAWL model for the handling of a request for compensation within an airline. To show some of the features of YAWL, we extended the process described in limitation of model with some more complex behaviors. In the new model it is possible that both examinations are executed. By using an OR-split and an OR-join *examine causally* and/or *examine thoroughly* are executed. The model has also been extended with a cancelation region (see dotted box in Fig 2.10).

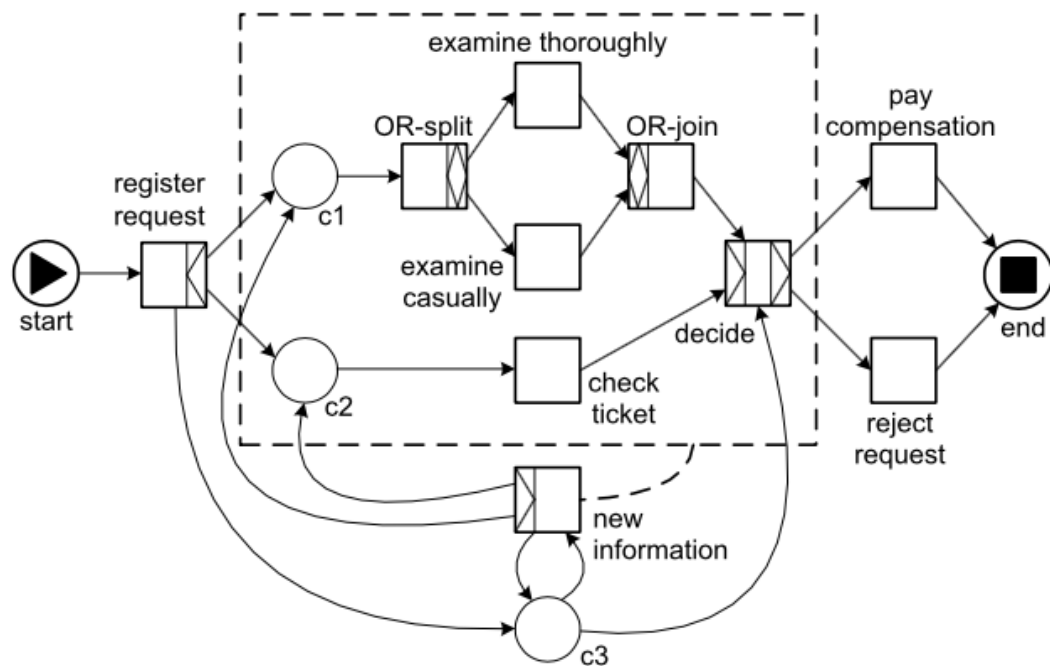


Fig 2.10 Process model using the YAWL notation [2]

8.Prom tools :

Process mining allows for the extraction information from event logs and is closely related to BAM (Business Activity Monitoring), BOM (Business Operations Management), BPI (Business Process Intelligence), and data/workflow mining. Unlike classical data mining techniques the focus is on processes and questions that transcend the simple performance-related queries supported by tools such as Business Objects, Congos BI, and Hyperion.

ProM is a generic open-source framework for implementing process mining algorithms in a standard environment .

ProM 5.2 features over 280 plug-ins for process mining, analysis, monitoring and conversion. And is available as binary distribution files for Windows, Mac OS X and Unix platforms, and as source code under the terms of the CPL license. Although ProM (up to version 5.2) has been a huge success, it did have a number of limitations. First of all, the plug-in concept lacked a clean separation between its actual process mining algorithm and its GUI. As a result, one could only use an algorithm in a context that allowed the user to provide necessary parameters through a GUI. When process mining applications become more mature/challenging, this limitation is not acceptable anymore.[5]

A second limitation was the fact that the framework was unaware of the inputs required by the plug-ins and the outputs they delivered (for example, plug-ins were not explicitly annotated with type information to allow chaining of plug-ins). As a result, it was not really possible to define ‘macro’ plug-ins. The framework simply did not know which plug-ins could be ‘chained’ into a single plug-in. A third shortcoming were problems related to licenses.

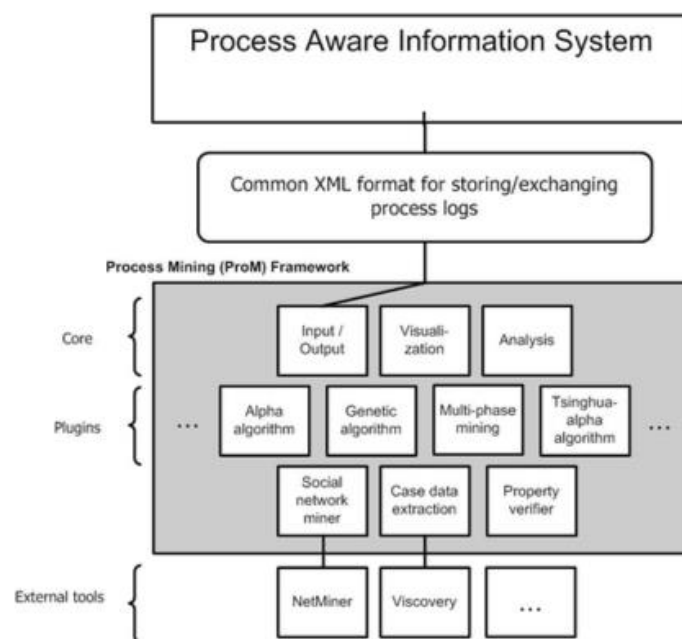


fig 2.11: gives an overview of the ProM framework. We discuss the core of the ProM framework components individually[4]

Input/output component This component handles the input and output of ProM framework files. As Figure 1 suggests, the input for the ProM framework are process logs stored in a common XML format. The structure of this XML file can be found in Appendix A. The output of the ProM framework is flexible. The ProM frame- work also supports different commercial

product formats such as NetMiner and Viscovery). These tools are analytical tools used in the industry to improve business processes through providing statistical data and since they have not been implemented as a plugin within the ProM framework, having an export function, for these industrial standard tools is a plus.

Visualization The framework provides a visualization plug-in that handles the actual drawing of rectangles, squares, circles or any graphical two dimensional object for the Process Mining framework. This component uses the DOT language, which is the language that is used to describe the graphical elements .

Analysis The analysis component comprises of plug-ins capable of Petri net analysis, log replay analysis, and checking EPCs for their correctness using their relative plug-ins. There are also plug-ins to compare a log and a model (i.e., conformance testing) or a log and an Linear Temporal Logic (LTL) formula. [6]

The plug-ins supported in the current ProM framework has evolved from smaller, deprecated, independent mining tools. Furthermore, the ProM framework supports different modelling languages (e.g., Petri nets, and EPCs) and other output formats such a Social Networks.

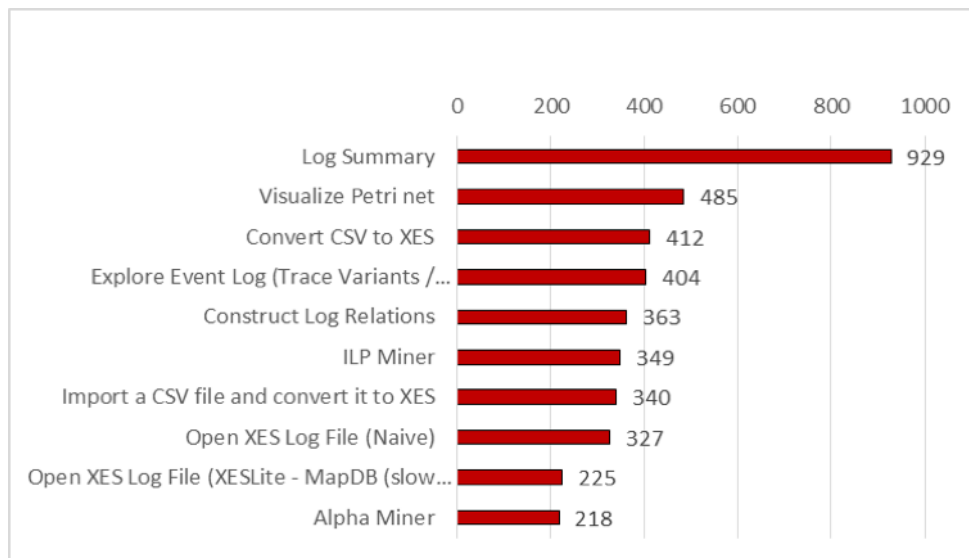


Fig 2.12Top-10 ProM plug-ins May 2018 by Eric Verbeek Scientific Engineer[8]

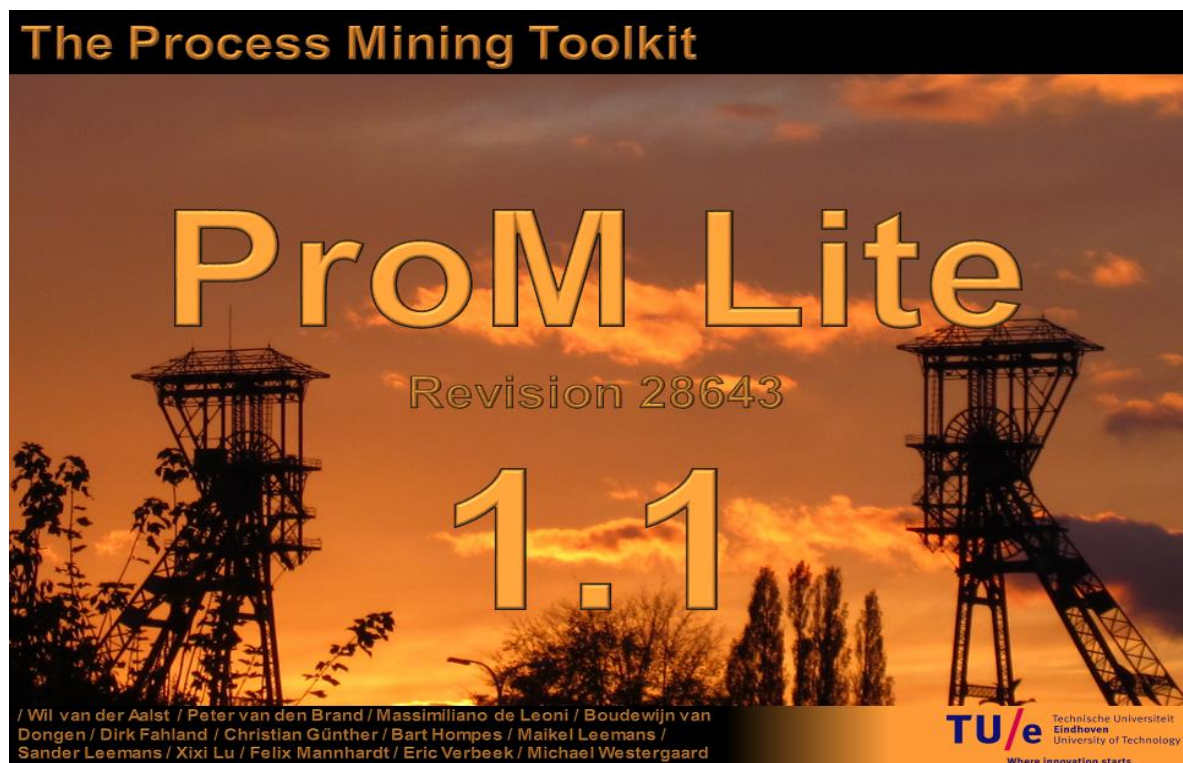


fig 2.13 prom lite version 1.1



figure 2.14 screenshot from Prom lite 1.1

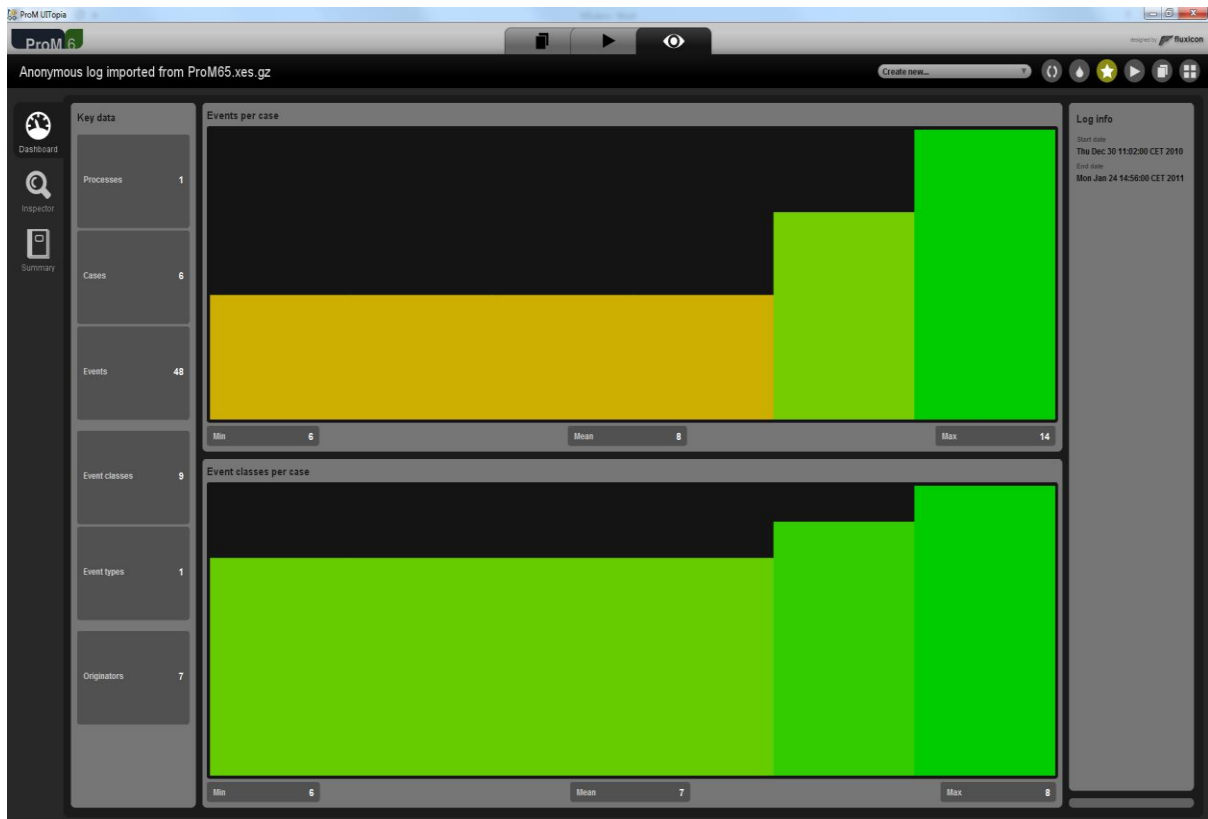


fig 2.15 dashboard of Prom lite 1.1

9.Event Log :

One of the big advantages of Process Mining is that it starts with the data that is already there, and usually it starts very simple. There is no need to first set up a data collection framework. Instead you can use data that accumulate as a byproduct of the increasing automation and digitization of your business processes. These data are collected right now by the various IT systems you already have in place to support your business.[7] If you are interested in Process Mining but are still new to this area, you probably have the following question:

«What kind of data do I need to do process mining»

Or, if you have heard about process mining through academia, you might ask:

« What exactly is an event log? »

The core idea of process mining is to analyze data from a process perspective. You want to answer questions such as “How does my As-is process currently look like?”, “Are there waste and unnecessary steps that could be eliminated?”, “Where are the bottlenecks?”, and “Are there deviations from the rules and prescribed processes?”.

To be able to do that, Process Mining approaches data with a mental model that maps the data to a process view.

9.1 Classification in data mining

To understand what this means, let us first take a look at another mental model: The mental model for classification in data mining. Assume that you have a widget factory and you want to understand which kinds of customers are buying your widgets. On the left side below, you see a very simple example of a data set. There are columns for the attributes Name, Salary, Sex, Age, and Buy widget. Each row forms one instance in the data set that can be used for learning the classification rules.

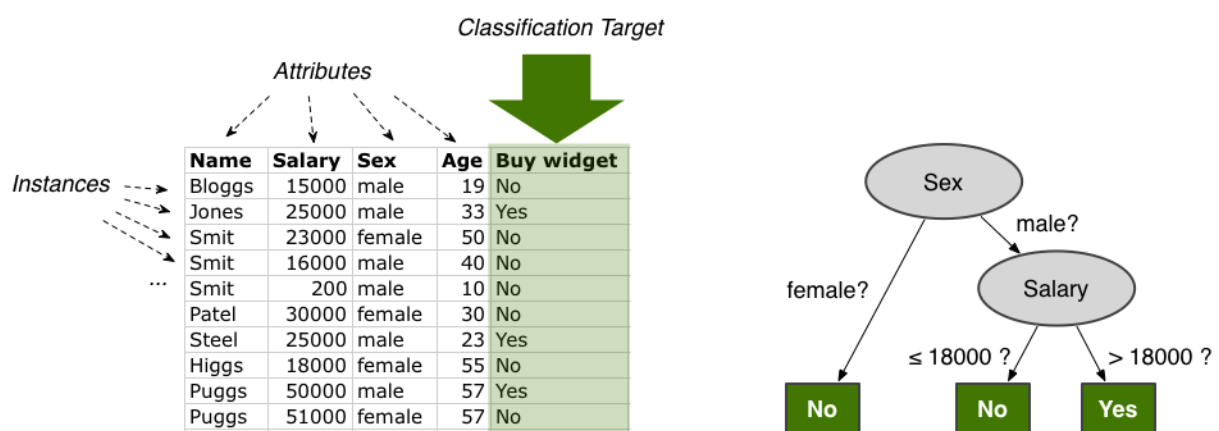


fig 2.16 : classification with data mining [9]

Before the classification algorithm can be started, one needs to determine which of the columns is the *target class*. Because we want to find out who is buying the widgets, we would make the Buy widget column the classification target. A data mining tool such as Weka would then be able to construct a decision tree like depicted on the right. The result shows that only males with a high salary are buying the widgets. If we would want to derive rules for another attribute, for example, predict how old the customers will typically be that buy our widgets, then the Age column would be the classification target.

9.2 The mental model for process mining

For process mining, we have a slightly different meta model in mind because we look at the data from a *process perspective*. Below, you see a simplified example data set from an internal call center case study. In contrast to the data mining example, an individual row does not represent a complete *process instance*, but just an *event*. That's where the term *event log* comes from.

- Each event corresponds to an **activity** that was executed in the process.
- Multiple events are linked together in a process instance or **case**.
- Logically, each case forms a sequence of events—ordered by their **timestamp**.

From the data sample below, you can see why even doing simple process-related analyses, such as measuring the frequency of process flow variants, or the time between activities, is impossible using standard tools such Excel. Process instances are scattered over multiple rows in a spreadsheet (not necessarily sorted!) and can only be linked by adopting a process-oriented meta model. [9]

9.3 The three requirements

The basis of process mining is to look at historical process data precisely with such a “process lens”. It’s actually quite simple. Regardless of where your data come from (database, log files, Excel sheet, data warehouse, etc.), the three minimal requirements are the following:

Case ID: A case identifier, also called process instance ID1, is necessary to distinguish different executions of the same process. What precisely the case ID is depends on the domain of the process.

For example, in a call center, the case ID would be a service request number. In a hospital, this would be the patient ID.

Activity: There should be names for different process steps or status changes that were performed in the process. If you have only one entry (one row) for each process instance, then your data is not detailed enough. Your data needs to be on the transactional level (you should have access to the history of each case) and should not be aggregated to the case level.

Timestamp: At least one timestamp is needed to bring the events in the right order. Of course you also need timestamps to identify delays between activities and identify bottlenecks in your process.[8] .If you have a start and complete timestamp for each activity in the process, then a distinction between active and idle times in the process becomes possible. Additional columns can be included for the analysis if available. For example, in the data sample there are further attributes that categorize the service request: A case was opened by phone, resolved by an external specialist, and the urgency was categorized as level 2. We might also include the resource or department that performed an activity. But the mandatory columns are just the three requirements above.

	A	B	C	D	E	F
1	student	action	stuff	time		
2	student0	inscription in	ibticem	2017-01-02 11:40:11		
3	student0	Registration	anwer	2017-01-02 12:47:33		
4	student0	identification	djaber	2017-01-02 12:53:50		
5	student0	rejected	houda	2017-01-05 13:23:09		
6	student0	Final Registr	zouhir	2017-01-09 08:29:28		
7	student1	inscription in	ibticem	2017-01-02 12:50:35		
8	student1	identification	djaber	2017-01-02 13:59:14		
9	student1	Registration	anwer	2017-01-02 14:20:19		
10	student1	pay the ticke	noura	2017-01-06 09:13:40		
11	student1	accepted	affaf	2017-01-06 11:47:36		
12	student1	Final Registr	ibticem	2017-01-06 16:49:21		
13	student2	inscription in	ibticem	2017-01-04 10:02:49		
14	student2	identification	djaber	2017-01-06 09:05:01		
15	student2	pay the ticke	oumaïma	2017-01-10 08:05:47		
16	student2	Registration	anwer	2017-01-10 09:12:09		
17	student2	accepted	affaf	2017-01-13 10:32:51		
18	student2	Final Registr	zouhir	2017-01-17 11:54:48		
19	student3	inscription in	ibticem	2017-01-05 16:10:58		
20	student3	Registration	anwer	2017-01-06 08:15:00		
21	student3	pay the ticke	oumaïma	2017-01-11 11:36:39		
22	student3	identification	iman	2017-01-11 12:38:58		
23	student3	accepted	affaf	2017-01-13 13:51:21		
24	student3	Final Registr	zouhir	2017-01-17 13:17:06		
25	student4	inscription in	rahim	2017-01-09 12:35:37		
26	student4	Registration	anwer	2017-01-09 13:44:25		
27	student4	identification	iman	2017-01-10 13:49:40		
28	student4	pay the ticke	noura	2017-01-13 13:39:04		
29	student4	accepted	affaf	2017-01-19 08:53:27		
30	student4	Final Registr	zouhir	2017-01-23 08:58:31		
31	student5	inscription in	ibticem	2017-01-11 09:58:23		
32	student5	identification	djaber	2017-01-11 16:17:39		
33	student5	pay the ticke	oumaïma	2017-01-11 16:25:11		
34	student5	Registration	anwer	2017-01-11 17:29:14		

fig 2.17 event log example

10.Conclusion :

In fact, there is a lot of concepts used in process mining As we have already mentioned, in this chapter Just mention the most important only. we see in the next chapter how we can collect all of this basic concept to start to practice the process mining . in chapter three we will present the main application for process mining .

