CHAPTER – 5

MODEL DISCOVERY

Model discovery is the first step of process mining. Business Models are the visuals of the business process and they are very powerful tool to express as well as easy to understand. Business models eliminate misinterpretations and avoid ambiguity. Giant business process cannot be either explained or analyzed without a proper model. A model conveys and communicates the message which is equivalent to hundreds of pages of text. Models and symbolic notations avoids the barrier of spoken languages. Process models are used to view the business process from various dimensions and various competent levels for further analysis and fine turning the business activity, which helps to accomplish the target and to cross the business milestones.

Business models may be classified into sequential models and concurrent models. Most of the models are sequential models which are linear in nature. Petri net model is the example for concurrent model and this model is based on the mathematical theory called concurrency theory. The selection of model and its notations are purely based on the need and suitability of the process activity. The fact of the present situation is that there is no single model notation that is fully suitable for all the process activity. For some activity WFN and for some other YAML notations are most suitable.

Since every business process is unique in its objectives, operations and style of functioning there is no single notation which is suitable for all the business situations. It is the tradeoff between the available notations and the demand of the process situation. One more fact to be acknowledged is that distinct model notations are suitable for various activities of the same process. Some situations demands personalized standalone modes which are most suitable and they are convenient to express as well as for analyze.

5.1 The art of modeling

The surplus of process modeling notations available today illustrate the relevance of process modeling. Some organizations may use only informal process models to structure
discussions and to document procedures. However, organizations that operates at a higher BPM maturity level use models that can be analyzed and used to enact operational processes. Today, most process models are made by hand and are not based on a rigorous analysis of existing process data. Since the industrial revolution, productivity has been increasing because of technical innovations, improvements in the organization of work, and the use of information technology. For the past few decades computers and digital communication infrastructures started to influence business processes. This resulted in dramatic changes in the organization of work and enabled new ways of doing business. Today, innovations in computing and communication are still the main drivers behind change in business processes. So, business processes have become more complex, heavily rely on information systems, and may span multiple organizations. Therefore, process modeling has become of the utmost importance. Process models assist in managing complexity by providing insight and documenting procedures. Information systems need to be configured and driven by precise instructions. Cross organizational processes can only function properly if there is a common agreement on the required interactions. As a result, process models are widely used in today’s organizations (Wil M.P. van der Aalst, 2010).

These are just some of the problems organizations face when making models by hand. Only experienced designers and analysts can make models that have a good predictive value and can be used as a starting point for a (re)implementation or redesign. An inadequate model can lead to wrong conclusions. Therefore, we advocate the use of event data. Process mining allows for the extraction of models based on facts. Moreover, process mining does not aim at creating a single model of the process. Instead, it provides various views on the same reality at different abstraction levels. For example, users can decide to look at the most frequent behavior to get a simple model (“80% model”). However, they can also inspect the full behavior by deriving the “100% model” covering all cases observed. Similarly, abstraction levels can be varied to create different views. Process mining can also reveal that people in organizations do not function as “machines”. On the one hand, it may be shown that all kinds of inefficiencies take place. On the other hand, process mining can also visualize the remarkable flexibility of some workers to deal with problems and varying workloads (Wil M.P. van der Aalst, 2010).
5.2 Process Models

It is not easy to make good process models. Yet, they are important. Fortunately, process mining can facilitate the construction of better models in less time. 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.

5.2.1 Transition Systems

The most basic process modeling notation is a transition system. A transition system consists of states and transitions. Figure 5.1 shows a transition system consisting of seven states.

![Transition System Diagram](image)

**Fig 5.1 - Transition System**

It models the handling of a request for compensation within an airline. The states are represented by black circles. There is one initial state labeled s1 and one final state labeled s7. Each state has a unique label. This label is merely an identifier and has no meaning. Transitions are represented by arcs. Each transition connects two states and is labeled with the name of an activity. Multiple arcs can bear the same label. For example, check ticket appears twice.

Any process model with executable semantics can be mapped onto a transition system. Therefore, many notions defined for transition systems can easily be translated to higher-level languages such as Petri nets, BPMN, and UML activity diagrams.
5.2.2 Petri Nets

Petri Nets are graphical and mathematical modeling notations. A Petri net is a four-tuple:

\[ PN = \langle P, T, I, O \rangle \]

- **P**: a finite set of places, \{p1, p2, ..., pn\}
- **T**: a finite set of transitions, \{t1, t2, ..., ts\}
- **I**: an input function, \((T \times P) \rightarrow \{0, 1\}\)
- **O**: an output function, \((T \times P) \rightarrow \{0, 1\}\)

Petri nets are traditionally used for describing and analyzing systems that are characterized as concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. Due to their graphical nature, Petri Nets can be used as a visualization technique like flow charts or block diagrams but with much more scope on concurrency aspects. As a strict mathematical notation, it is possible to apply formal concepts like linear algebraic equations or probability theory for investigating the behavior modeled system. In the last decade, Petri Nets have become a powerful concept in the area of business process modeling and work flow management (Wil M.P. van der Aalst, 2010).

Petri nets have been successfully used to model and analyze processes from many domains such as software and business processes especially workflow processes. Petri Net model consists of two parts:

1. The net structure that represents the static part of the system and
2. A marking that represents the overall state on the structure. The token distribution among the places of a Petri net is called its marking. When one or more tokens reside in a place, the place is said to be marked, otherwise it is unmarked.

Petri nets have been successfully used to model and analyze processes from many domains such as software and business processes especially workflow processes. The classical Petri net is a directed bipartite graph with two node types called places and transitions. The nodes are connected via directed arcs. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by rectangles.

**Petrinet Notations:**

A Petri Net consists of places, transitions and arcs that connect them.
**Places:** Places are represented by ellipses (often circles). Three annotations are associated with a place p:

- The place name;
- The name of the type (Type(p)) associated with the place; and
- The initial marking.

**Transitions:** Transitions are rectangles and arcs as arrows. Input arcs connect transitions with places.

**Arcs:** An arc is represented by an arrow: →

Places are passive components and are modeling the system state. They can contain TOKENS, depicted as black dots. The current state of the Petri net is given by the number of tokens on each place.

Transitions are active components modeling activities which can occur and cause a change of the state by a new assignment of token to places. Transitions are only allowed to occur if they are enable, which means that there is at least one token on each input place. By occurring, the transition removes a token from each input place and adds a token on each output place. The repeated occurrence of transitions and the resulting sequence of marking are called the token game. Petri Net model consist of two parts

- The net structure that represents the static part of the system and
- A marking that represents the overall state on the structure. The token distribution among the places of a Petri net is called its marking. When one or more tokens reside in a place, the place is said to be marked, otherwise it is unmarked.

Transitions symbolize actions; places symbolize states or conditions that need to be met before an action can be carried out.
BASIC CONSTRUCTS OF PETRI NETS

Fig 5.2 - Sequential action

Fig 5.3 - Dependency

Fig 5.4 - Conflict (decision, choice)

Fig 5.5 - Concurrency

Fig 5.6 - Cycles

Fig 5.7 - Synchronization

Fig 5.8 - Resource sharing
5.2.3 Workflow Nets

When modeling business processes in terms of Petri nets, we often consider a subclass of Petri nets known as Work Flow nets. A WF-net is a Petri net with a dedicated source place where the process starts and a dedicated sink place where the process ends. Moreover, all nodes are on a path from source to sink.

WF-nets are particularly relevant for business process modeling because they describe the life-cycle of cases of a given kind. Examples of cases are insurance claims, job applications, customer orders, replenishment orders, patients, and credit applications. The process model is instantiated once for each case. Each of these process instances has a well-defined start ("case creation") and end ("case completion"). In-between these points, activities are conducted according to a predefined procedure. One model may be instantiated many times. For example, the process of handling insurance claims may be executed for thousands or even millions of claims. These instances can be seen as copies of the same WF-net, i.e., tokens of different cases are not mixed. WF-nets are also a natural representation for process mining. There is an obvious relation between the firing sequences of a WF-net and the traces found in event logs. Note that one can only learn models based on examples. In the context of market basket analysis, i.e., finding patterns in what customers buy, one needs many examples of customers buying particular collections of products. Similarly, process discovery uses sequences of activities in which each sequence refers to a particular process instance. These can be seen as firing sequences of an unknown WF-net. Therefore, we will often focus on WF-nets.

5.2.4 YAWL

YAWL is both a workflow modeling language and an open-source workflow system. The acronym YAWL stands for “Yet Another Workflow Language”. 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.

![YAWL Notations](image)

Fig 5.9 - YAWL Notations

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.

Process model using YAWL notation.

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 that an OR-split may select 2 out of three 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 cancelation regions. A task may have a cancelation 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 cancelation region. YAWL’s cancelation 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 (Wil M.P. van der Aalst, 2010).

**Fig 5.10 - YAWL Process model**

**5.2.5 Business Process Modeling Notation**

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 by the OMG.
Atomic activities are 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 routing logic is not associated with tasks but with separate gateways.

![BPMN Notations](image)

**Fig 5.11 - BPMN Notations**

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. 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.
5.2.6 Event Driven Process Chains

Event-driven Process Chains (EPCs) provide a classical notation to model business processes. The notation is supported by products such as ARIS and SAP R/3. Basically, EPCs cover a limited subset of BPMN and YAWL while using a dedicated graphical notation.
Functions correspond to activities. A function has precisely one input arc and one output arc. Therefore, splitting and joining can only be modeled using connectors. These are comparable to the gateways in BPMN. Again splits and joins of type AND, XOR, and OR are supported. Like in BPMN, there are three types of events (start, intermediate, and end). Events and functions need to alternate along any path, i.e., it is not allowed to connect events to events or functions to functions.

Process model using EPC notation

![EPC Process Model](image)

**Fig 5.14 - EPC Process Model**

### 5.2.7 Casual Nets

Causal nets are a representation tailored toward process mining. A causal net is a graph where nodes represent activities and arcs represent causal dependencies. Each activity has a set of possible input bindings and a set of possible output bindings.
Activity “a” has only an empty input binding as this is the start activity. There are two possible output bindings: \{b, d\} and \{c, d\}. This means that a is followed by either b and d, or c and d. Activity e has two possible input bindings (\{b, d\} and \{c, d\}) and three possible output bindings (\{g\}, \{h\}, and \{f\}). Hence, e is preceded by either b and d, or c and d, and is succeeded by just g, h or f. Activity z is the end activity having two input bindings and one output binding (the empty binding). This activity has been added to create a unique end point. All executions commence with start activity a and finish with end activity z. Causal nets are particularly suitable for process mining given their declarative nature and expressiveness without introducing all kinds of additional model elements (places, conditions, events, gateways, etc.). Several process discovery and conformance checking approaches use a similar representation (Wil M.P. van der Aalst, 2010).

5.3 Model Discovery

The model generator automatically generates the customized visual model of the motor claim process by using unsupervised machine learning approach and inductive learning methodology. The software technology used in this model generator is .Net 4.0 framework.
and C# is used to build this application. Windows Presentation Frame work (WPF) is used for presentation and diagram. Quick Graph open source framework is also used. This application is targeted to run on Windows platform.

It is a .Net windows form based application using windows presentation framework. This application accepts a well defined trace table data. This trace table data can be entered manually by the user or can be loaded from a text file which is pre-generated.

![Diagram](image)

**Fig 5.16 - Load Trace from text file**

### 5.3.1 Generic Description

**Trace Manager:** Trace Manager is the primary module of this application and it is primarily responsible for process the traces and pass it to the Quick Graph to display as a visual process model. To build a process model successor and predecessor table is the important factors, which are generated by this trace manager module.
Successor and Predecessor Tables: Successor table is build by iterating through every trace. After constructing the successor table, based on the trace input and successor table, the predecessor for each entry is identified and updated.

Quick Graph: It is an open software graph framework which is used to create graphs and present in WPF diagram. After the construction of successor and predecessor tables, it is clear to disembark edges and vertices.

![Diagram](image)

Fig 5.17 - Generic Flow

5.4 Model Discovery from Insurance Motor Claim Process

Consider the processing of motor claims in an insurance company. First the customer registers the claim with the insurance company (task register the claim). The loss is recorded in the claim register (task examine) and a claim form is issued to the customer. The insurance company then verifies the policy records to see if the policy is in force and also confirms
64VB clause of the Insurance Act, 1938 of India (task Check 64 VB confirmation). The insurance company then decides (task decide) and either accepts the claim (task Honour the claim) or rejects the claim (task Repudiate the claim).

The insured is required to submit a detailed estimate of repairs from any repairer of his choice. Generally, these repairs are acceptable but at times the insurance company asks the customer to obtain repair estimate from another repairer, if they have reason to believe that the competence, moral hazard or business integrity of the first repairer is not satisfactory (task Panel assessment). Verification of all the claim documents by the approved loss assessor or the surveyor is the task Check documents.

Section 64 VB of the Insurance Act, 1938 of India stipulates that no risk can be assumed without prior payment of full premium except when

(i) The entire amount of the premium is guaranteed to be paid by a bank before the end of the first calendar month after the month in which the risk is assumed, or

(ii) An advance deposit is made with the insurer to the credit of the Insured sufficient to cover the payment of the entire amount of premium
Table 5.1 - A fragment of insurance event log: each row corresponds to an event

<table>
<thead>
<tr>
<th>CASE ID</th>
<th>EVENT ID</th>
<th>TASK</th>
<th>ORIGINATOR</th>
<th>TIMESTAMP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4587320</td>
<td>Register claim request</td>
<td>Raj</td>
<td>11/08/2011 15:05</td>
</tr>
<tr>
<td>1</td>
<td>4587321</td>
<td>Examine</td>
<td>Sam</td>
<td>11/08/2011 15:30</td>
</tr>
<tr>
<td></td>
<td>4587322</td>
<td>Check 64VB confirmation</td>
<td>Mano</td>
<td>11/08/2011 15:45</td>
</tr>
<tr>
<td></td>
<td>4587323</td>
<td>Decide</td>
<td>Ravi</td>
<td>11/08/2011 16:30</td>
</tr>
<tr>
<td></td>
<td>4587324</td>
<td>Repudiate the claim</td>
<td>Mani</td>
<td>11/08/2011 16:45</td>
</tr>
<tr>
<td></td>
<td>4587331</td>
<td>Register claim request</td>
<td>Sam</td>
<td>26/04/2011 11:30</td>
</tr>
<tr>
<td>2</td>
<td>4587332</td>
<td>Check 64VB confirmation</td>
<td>Mano</td>
<td>26/04/2011 11:45</td>
</tr>
<tr>
<td></td>
<td>4587333</td>
<td>Examine</td>
<td>Raj</td>
<td>26/04/2011 12:00</td>
</tr>
<tr>
<td></td>
<td>4587334</td>
<td>Decide</td>
<td>Ravi</td>
<td>26/04/2011 12:30</td>
</tr>
<tr>
<td></td>
<td>4587335</td>
<td>Honor the claim</td>
<td>Mani</td>
<td>26/04/2011 12:45</td>
</tr>
<tr>
<td></td>
<td>4587341</td>
<td>Register claim request</td>
<td>Raj</td>
<td>29/06/2011 18:20</td>
</tr>
<tr>
<td>3</td>
<td>4587342</td>
<td>Examine</td>
<td>Ravi</td>
<td>30/06/2011 09:45</td>
</tr>
<tr>
<td></td>
<td>4587343</td>
<td>Check 64VB confirmation</td>
<td>Mani</td>
<td>30/06/2011 10:15</td>
</tr>
<tr>
<td></td>
<td>4587344</td>
<td>Decide</td>
<td>Mano</td>
<td>30/06/2011 10:40</td>
</tr>
<tr>
<td></td>
<td>4587345</td>
<td>Panel assessment</td>
<td>Sam</td>
<td>30/06/2011 15:20</td>
</tr>
<tr>
<td></td>
<td>4587346</td>
<td>Examine</td>
<td>Mano</td>
<td>30/06/2011 16:00</td>
</tr>
<tr>
<td></td>
<td>4587347</td>
<td>Check documents</td>
<td>Sam</td>
<td>01/07/2011 10:00</td>
</tr>
<tr>
<td></td>
<td>4587348</td>
<td>Decide</td>
<td>Mano</td>
<td>01/07/2011 11:00</td>
</tr>
<tr>
<td></td>
<td>4587349</td>
<td>Honor the claim</td>
<td>Ravi</td>
<td>01/07/2011 13:15</td>
</tr>
<tr>
<td></td>
<td>4587351</td>
<td>Register claim request</td>
<td>Ravi</td>
<td>03/03/2011 10:00</td>
</tr>
<tr>
<td>4</td>
<td>4587352</td>
<td>Examine</td>
<td>Mani</td>
<td>03/03/2011 11:00</td>
</tr>
<tr>
<td></td>
<td>4587353</td>
<td>Check 64VB confirmation</td>
<td>Sam</td>
<td>03/03/2011 11:15</td>
</tr>
</tbody>
</table>
The above table represents the information in an event log. The bare minimum requirements for process mining are that an event can be related to both a case and an activity.

With the information from the above table we obtain the more compact information representation of the event log as shown below.

**Table 5.2: TRACE TABLE**

<table>
<thead>
<tr>
<th>CASE ID</th>
<th>TRACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(a,b,c,d,h)</td>
</tr>
<tr>
<td>2</td>
<td>(a,c,b,d,g)</td>
</tr>
<tr>
<td>3</td>
<td>(a,b,c,d,e,b,f,d,g)</td>
</tr>
<tr>
<td>4</td>
<td>(a,b,c,d,e,f,b,d,g)</td>
</tr>
<tr>
<td>5</td>
<td>(a,b,c,d,e,f,b,d,h)</td>
</tr>
</tbody>
</table>
In the above table, the case is represented by a sequence of activities referred to as trace and the activity names are represented by single letter labels. The various labels mentioned in the table denotes as

- a - Register claim request
- b - Examine
- c - Check 64VB confirmation
- d - Decide
- e - Panel assessment
- f - Check documents
- g - Honor the claim
- h - Repudiate the claim

The information given in the above table can be transformed into a process model.

5.5 **Model Generator of the Trace Table**

Model generator takes input from the trace table (Table – 5.2) and generates a model. Fig – 5.18 & Fig – 5.19 are the input screen shot of the model generator, Fig – 5.20 & Fig – 5.21 are the respective models for the inputs. Where a – h are the various activities and T1 – T14 are the transition states.

**Model Generator**

---

**Fig 5.18 Input Trace – Screen 1**

**Fig 5.19 Input Trace – Screen 2**
5.6 Summary

This chapter discussed about various standard model notations and the design of a personalized model generator which uses two tables namely successor table and predecessor tables. These two tables are collectively called as a Trace Manager. The input for the Trace Manager is a string of alphabets; each alphabet represents a particular activity of a case. The personalized model is most convenient and customized model for motor claim process.