

# Table of Contents

1. [Introduction to Process Mining](#)
  1. [Process and Data Science](#)
  2. [Starting Point: Event Logs](#)
  3. [Process Mining Types](#)
2. [Petri Nets: Foundations for Process Modeling](#)
  4. [Elements of a Petri Net](#)
  5. [Formal Definition of Petri Nets](#)
  6. [Syntactic Rules and Semantics](#)
  7. [Transition Enabling and Firing](#)
  8. [Typical Network Structures](#)
3. [Soundness Analysis of Process Models](#)
  1. [Reachability Graph](#)
  2. [Properties for Analysis](#)
  3. [Workflow Nets](#)
  4. [Soundness Definition](#)
  5. [Coverability Graphs](#)
4. [Process Discovery: Basic Approaches](#)
  6. [The Directly-Follow Relation](#)
  7. [The Alpha Algorithm](#)
  8. [Limitations of the Alpha Algorithm](#)
  9. [Evaluating Process Models](#)
5. [Advanced Process Discovery Algorithms](#)
  1. [Dependency Measures](#)
  2. [Heuristic Mining](#)
  3. [Causal Nets](#)
  4. [Region-Based Mining](#)
6. [Conformance Checking](#)
  1. [Token-Based Replay](#)
  2. [Footprint Comparison](#)
  3. [Alignments](#)
  4. [Precision Metrics](#)
7. [Mining Additional Perspectives](#)
  1. [Organizational Mining](#)
  2. [Decision Mining](#)
  3. [Time Perspective](#)
8. [BPMN and Process Simulation](#)

4. [BPMN Notation](#)
  5. [Elements of a Simulation Model](#)
  6. [Simulation Output Analysis](#)
  7. [Pitfalls of Simulation](#)
- 

# 1. Introduction to Process Mining

## 1.1 Process and Data Science

Process mining is positioned at the intersection of data science and process science. It aims to extract knowledge from event logs recorded by information systems to discover, monitor, and improve real business processes.

The field bridges the gap between:

- **Process science:** business process management, workflow management, process automation, and formal methods
- **Data science:** machine learning, data mining, statistics, and artificial intelligence

Process mining acknowledges that processes are everywhere in organizations, from administrative procedures to healthcare pathways, production workflows, and customer journeys.

## 1.2 Starting Point: Event Logs

The foundation of process mining is the event log, which contains records of activities executed in a process.

An event log typically contains:

- **Case ID:** Identifier for the process instance (e.g., order number)
- **Activity:** The action performed in the process
- **Timestamp:** When the activity was executed
- **Resource:** Who performed the activity
- **Other attributes:** Additional information related to the event

Example of a simplified event log:

Case ID	Activity	Timestamp	Resource	Other info
123456	Register Request	2023-01-01 10:00	John	Priority=High
123456	Examine File	2023-01-01 11:30	Mary	Complexity=Low
123457	Register Request	2023-01-01 10:15	John	Priority=Medium

Case ID	Activity	Timestamp	Resource	Other info
123456	Check Ticket	2023-01-01 14:00	Pete	Result=OK
123457	Examine File	2023-01-01 11:45	Mary	Complexity=High

## 1.3 Process Mining Types

Process mining can be categorized into three main types:

1. **Process Discovery:** Extracting process models from event logs to provide a visual representation of the process. This answers the question "what is actually happening?"
2. **Conformance Checking:** Comparing observed behavior (event logs) with normative/expected behavior (process models). This addresses the question "are we doing what we're supposed to do?"
3. **Enhancement:** Improving or extending existing process models using information extracted from event logs. This helps answer "how can we improve the process?"

Each type serves different purposes and can be applied in various scenarios such as business process improvement, compliance checking, or bottleneck identification.

## 2. Petri Nets: Foundations for Process Modeling

### 2.1 Elements of a Petri Net

Petri nets are a fundamental modeling formalism for representing processes. The key elements are:

- **Places:** Represented as circles, indicating conditions or states
- **Transitions:** Represented as rectangles or bars, indicating events or actions
- **Arcs:** Directed edges connecting places to transitions or transitions to places
- **Tokens:** Represented as black dots in places, indicating the current state of the system

Petri nets provide a balance between formal mathematical foundation and visual intuition, making them suitable for both analysis and communication.

### 2.2 Formal Definition of Petri Nets

A Petri net is formally defined as a triple  $(P, T, F)$  where:

- $P$  is a finite set of places
- $T$  is a finite set of transitions
- $F \subseteq (P \times T) \cup (T \times P)$  is a flow relation

For a given node  $x$  (place or transition), we define:

- $\bullet x = \{y \mid (y, x) \in F\}$  (the preset of  $x$ )
- $x\bullet = \{y \mid (x, y) \in F\}$  (the postset of  $x$ )

## 2.3 Syntactic Rules and Semantics

A Petri net follows these syntactic rules:

- It is a directed graph with two types of nodes: places and transitions
- Arcs can only connect places to transitions or transitions to places
- Places may hold zero or more tokens

The state of a Petri net is defined by its marking:

- A marking  $m$  is a function  $m: P \rightarrow \mathbb{N}$  that assigns to each place a non-negative number of tokens
- The initial marking  $m_0$  represents the initial state of the system

## 2.4 Transition Enabling and Firing

The dynamics of Petri nets are governed by transition enabling and firing rules:

1. **Enabling Rule:** A transition  $t$  is enabled at marking  $m$  if for all  $p \in \bullet t$ ,  $m(p) \geq 1$ 
  - In other words, each input place must contain at least one token
2. **Firing Rule:** When an enabled transition  $t$  fires:
  - It consumes one token from each input place
  - It produces one token in each output place

Formally, if transition  $t$  fires at marking  $m$ , the new marking  $m'$  is:  $m'(p) = m(p) - w((p,t)) + w((t,p))$  for all  $p \in P$

Where  $w$  is the weight function, which for simple Petri nets is 1 for existing arcs and 0 otherwise.

## 2.5 Typical Network Structures

Petri nets can represent common control-flow patterns:

1. **Sequence:** Two activities need to occur in a specific order

$\circ \rightarrow \square \rightarrow \circ \rightarrow \square \rightarrow \circ$

2. **Choice (XOR-split):** One of several paths is chosen

$\square \rightarrow \circ$

$\nearrow \quad \searrow$



### 3. Parallel Execution (AND-split and AND-join): Activities can execute concurrently



### 4. Iteration (Loop): Activities can be repeated



These basic structures can be combined to model complex business processes.

## 3. Soundness Analysis of Process Models

### 3.1 Reachability Graph

The reachability graph of a Petri net is a state-transition system that represents all possible markings reachable from the initial marking and the transitions between them.

Formally, a reachability graph is a directed graph where:

- Nodes represent markings reachable from the initial marking
- Edges represent transitions, labeled with the transition name

The algorithm to construct a reachability graph:

1. Label the initial marking  $m_0$  as the root and tag it "new"
2. While "new" markings exist:
  - a. Select a new marking  $m$
  - b. If no transitions are enabled at  $m$ , tag  $m$  "dead-end"
  - c. For each enabled transition  $t$  at  $m$ :
    - i. Obtain the marking  $m'$  that results from firing  $t$  at  $m$
    - ii. If  $m'$  does not appear in the graph, add  $m'$  and tag it "new"
    - iii. Draw an arc with label  $t$  from  $m$  to  $m'$

### 3.2 Properties for Analysis

Several important properties can be analyzed using the reachability graph:

1. **Boundedness:** A place  $p$  is  $k$ -bounded if it never contains more than  $k$  tokens. A Petri net is bounded if every place is bounded.
  - If  $k=1$ , the place (or net) is called safe
  - If there is no such  $k$ , the place (or net) is unbounded
2. **Deadlock Freedom:** A Petri net is deadlock-free if at least one transition is enabled at every reachable marking.
  - A deadlock is a marking with no enabled transitions
3. **Dead Transitions:** A transition  $t$  is dead if it is not enabled at any reachable marking.
4. **Liveness:** A transition  $t$  is live if from every reachable marking, there is a sequence of transitions that leads to a marking where  $t$  is enabled.
  - A Petri net is live if every transition is live
5. **Home-marking:** A marking  $m$  is a home-marking if from any reachable marking, there is a path to  $m$ .
  - A Petri net is reversible if the initial marking is a home-marking

### 3.3 Workflow Nets

A workflow net is a special type of Petri net tailored to model processes:

A Petri net  $N = (P, T, F)$  is a workflow net if:

1. It has a unique source place  $i$  with  $\bullet i = \emptyset$
2. It has a unique sink place  $o$  with  $o \bullet = \emptyset$
3. Every node in the net is on a path from  $i$  to  $o$

Workflow nets are particularly useful for business process modeling as they have a clear starting and ending point and all activities contribute to the completion of the process.


### 3.4 Soundness Definition

Soundness is a key correctness criterion for workflow nets:

A workflow net  $N = (P, T, F)$  with initial marking  $[i]$  is sound if:

1. **Option to complete:** For any marking reachable from  $[i]$ , it is possible to reach the marking  $[o]$
2. **Proper completion:** The marking  $[o]$  is the only reachable marking from  $[i]$  with at least one token in place  $o$
3. **No dead transitions:** There are no dead transitions in the net

In terms of the reachability graph:

1.  is a home marking
2. For each marking  $m$  in the reachability graph:
  - $m(o) < 2$

- if  $m(o) = 1$  then  $m(p) = 0$  for every  $p \in P \setminus \{o\}$

3. No transition is dead

An important theorem: A workflow net is sound if and only if its short-circuit net (adding a transition from  $o$  to  $i$ ) is live and bounded.

## 3.5 Coverability Graphs

For unbounded Petri nets, the reachability graph is infinite. To handle this, we use coverability graphs as finite abstractions.

The key idea is to represent unbounded growth of tokens using a special symbol  $\omega$ , which denotes "arbitrarily many" tokens.

The coverability graph construction algorithm is similar to the reachability graph algorithm, with an additional step:

- If, when adding a new marking  $m$ , there is a marking  $m'$  on the path from the initial marking to  $m$  such that  $m' < m$  ( $m$  is strictly greater than  $m'$  component-wise), then set  $m(p) = \omega$  for all places  $p$  where  $m(p) > m'(p)$

Properties of coverability graphs:

1. The coverability graph is always finite
2. A transition is dead if and only if it does not appear in the coverability graph
3. A place  $p$  is  $k$ -bounded if and only if  $p$  does not contain more than  $k$  tokens in any marking of the coverability graph

For workflow nets, the presence of  $\omega$  in the coverability graph indicates unboundedness and consequently, unsoundness.

## 4. Process Discovery: Basic Approaches

### 4.1 The Directly-Follow Relation

The most basic relationship that can be extracted from an event log is the directly-follows relation, denoted by  $>$ :

For activities  $a$  and  $b$ ,  $a > b$  if and only if there exists a trace in the log where  $a$  is directly followed by  $b$ .

This relation can be represented as a matrix, where the entry at row  $a$  and column  $b$  indicates how many times  $a$  is directly followed by  $b$  in the log.

Example for a log with traces  $[abcd, acbd, aed]$ :

>	a	b	c	d	e
a	0	2	1	0	1
b	0	0	1	2	0
c	0	1	0	1	0
d	0	0	0	0	0
e	0	0	0	1	0

The directly-follows relation can be used to construct a dependency graph, which is a simple process model showing which activities follow each other.

## 4.2 The Alpha Algorithm

The Alpha algorithm is a process discovery technique that extracts a Petri net from an event log. It extends the directly-follows relation with additional relations:

1. **Causality ( $\rightarrow$ ):**  $a \rightarrow b$  if  $a > b$  and not  $b > a$
2. **Parallel ( $\parallel$ ):**  $a \parallel b$  if  $a > b$  and  $b > a$
3. **Choice ( $\#$ ):**  $a \# b$  if not  $a > b$  and not  $b > a$

The algorithm steps are:

1. Identify the set of activities  $T\_L$  in the log  $L$
2. Identify the set of start activities  $T\_I$  (activities that appear at the beginning of at least one trace)
3. Identify the set of end activities  $T\_O$  (activities that appear at the end of at least one trace)
4. Create the footprint matrix showing the relations  $\rightarrow$ ,  $\parallel$ , and  $\#$  between all pairs of activities
5. Identify sets of activities  $(A, B)$  where:
  - $A$  and  $B$  are non-empty subsets of  $T\_L$
  - For all  $a$  in  $A$  and  $b$  in  $B$ ,  $a \rightarrow b$
  - For all  $a_1, a_2$  in  $A$ ,  $a_1 \# a_2$
  - For all  $b_1, b_2$  in  $B$ ,  $b_1 \# b_2$
6. Minimize these sets to get maximal pairs  $(A, B)$
7. For each maximal pair  $(A, B)$ , create a place  $p\_ (A,B)$  in the Petri net
8. Add a source place  $i$  and connect it to all start activities
9. Add a sink place  $o$  and connect all end activities to it
10. Connect each activity in  $A$  to place  $p(A,B)$ , and connect place  $p(A,B)$  to each activity in  $B$

The result is a Petri net that captures the behavior observed in the event log.



## 4.3 Limitations of the Alpha Algorithm

The Alpha algorithm has several limitations:

1. **Implicit Places:** It cannot discover implicit places (places that don't add constraints)
2. **Loops of Length 1 or 2:** It cannot handle short loops correctly
3. **Non-local Dependencies:** It cannot discover dependencies between activities that are not directly following each other
4. **Noise Handling:** It does not consider frequencies, so rare behaviors have the same impact as frequent ones
5. **Incompleteness:** It requires the log to contain a "complete" set of behaviors

More advanced algorithms have been developed to address these limitations.

## 4.4 Evaluating Process Models

Process discovery aims to balance four quality dimensions:

1. **Fitness:** The model's ability to replay the observed behavior in the log
  - A model with perfect fitness can reproduce all traces in the log
  - Low fitness means that the model cannot explain all observed behavior
2. **Precision:** The model's ability to avoid allowing behavior not observed in the log
  - A model with perfect precision only allows behavior seen in the log
  - Low precision means the model allows for more behavior than seen in the log (underfitting)
3. **Generalization:** The model's ability to generalize beyond the observed examples
  - Good generalization means the model can handle unseen but similar behavior
  - Poor generalization leads to overfitting to the specific examples in the log
4. **Simplicity:** The model's structural complexity
  - A simple model is easier to understand and analyze
  - Complex models might be more accurate but harder to work with

These dimensions often conflict with each other, requiring a balance based on the specific use case.

## 5. Advanced Process Discovery Algorithms

### 5.1 Dependency Measures

Advanced process discovery algorithms often use dependency measures to quantify the strength of causal relationships between activities.

The dependency measure between activities  $a$  and  $b$  is defined as:

$$a \Rightarrow b = (a > b - |b > a|) / (a > b + b > a + 1) \text{ if } a \neq b \quad a \Rightarrow a = a > a / (a > a + 1) \text{ if } a = b$$

Where:

- $a > b$  is the number of times  $a$  is directly followed by  $b$
- $|b > a|$  is the number of times  $b$  is directly followed by  $a$

Properties of the dependency measure:

- If  $a > b = b > a$  (parallel), then  $a \Rightarrow b = 0$
- If  $a > b \gg b > a$  (strong causality), then  $a \Rightarrow b \approx 1$
- $a \Rightarrow b = -(b \Rightarrow a)$

This measure allows for filtering out weak dependencies, which might be due to noise.

## 5.2 Heuristic Mining

Heuristic mining builds on the dependency measures to construct a process model. The main steps are:

1. Calculate the dependency measures between all pairs of activities
2. Apply thresholds to filter out weak dependencies
3. Learn splits and joins based on the dependency graph
4. Convert the resulting representation to a Petri net or other process model

Advantages of heuristic mining:

- Considers frequencies, so it's more robust to noise
- Can handle short loops correctly
- Produces models that focus on the main behavior

The approach uses windows around activities to learn how they are connected to other activities, capturing local patterns to infer the overall process structure.

## 5.3 Causal Nets

Causal nets (C-nets) provide a notation that can represent a wider range of behaviors than Petri nets.

A C-net consists of:

- Activities (nodes)
- Causal dependencies (arcs)
- Input and output bindings for each activity

Each activity has:

- Input bindings: Sets of activities that must all complete before the activity can execute

- Output bindings: Sets of activities that may be enabled after the activity completes

C-nets use the concept of obligations:

- An obligation (a,b) means activity a has occurred and activity b needs to occur later
- Activities fulfill obligations from their input bindings and create obligations for their output bindings

The advantage of C-nets is their ability to represent complex dependencies and choices not easily captured by Petri nets.

## 5.4 Region-Based Mining

Region-based mining approaches process discovery differently by first constructing a transition system from the event log and then converting it to a Petri net.

The main steps are:

1. Construct a transition system from the event log:
  - States represent process states (can be defined in different ways, e.g., as the set of activities executed so far)
  - Transitions represent moves from one state to another by executing an activity
2. Identify regions in the transition system:
  - A region is a set of states where all transitions with the same label either all enter, all exit, or all don't cross the region
  - Each region corresponds to a place in the Petri net
3. Construct a Petri net:
  - Each activity in the log becomes a transition
  - Each minimal non-trivial region becomes a place
  - Add arcs based on how transitions enter or exit regions
  - Place tokens in regions containing the initial state

Advantages of region-based mining:

- Can discover complex control flow patterns
- Provides a clear formal basis for process discovery
- Can be tailored by choosing different state representations

Limitations:

- Sensitive to the state representation chosen
- Can produce large or overly-complex models
- May require post-processing to get a workflow net

## 6. Conformance Checking

## 6.1 Token-Based Replay

Token-based replay is a technique for checking how well a process model fits an event log by replaying the traces on the model.

The approach uses four counters:

- p: produced tokens (including initial tokens)
- c: consumed tokens
- m: missing tokens (tokens added to enable transitions that should fire)
- r: remaining tokens (tokens left in the net after replay)

For each trace in the log:

1. Begin with the initial marking
2. Try to fire transitions corresponding to the activities in the trace
3. If a transition is not enabled but should fire, add missing tokens
4. After replay, count remaining tokens

Fitness at the trace level is calculated as:  $\text{fitness} = 1 - (m/c + r/p)/2$

Fitness at the log level is calculated by aggregating the counters across all traces:  $\text{fitness} = 1 - (\text{sum}(m)/\text{sum}(c) + \text{sum}(r)/\text{sum}(p))/2$

Perfect fitness (1.0) means all traces can be replayed without missing or remaining tokens.

## 6.2 Footprint Comparison

Another approach to conformance checking is comparing the footprint of the log with the footprint of the model.

The footprint matrix shows the relationships ( $\rightarrow$ ,  $\parallel$ ,  $\#$ ) between all pairs of activities.

Steps:

1. Create the footprint matrix for the log
2. Create the footprint matrix for the model
3. Count the number of cells where the matrices differ
4. Calculate conformance as:  $1 - (\text{number of differing cells}) / (\text{total number of cells})$

This approach is simple but has limitations:

- It doesn't pinpoint exactly where deviations occur
- It's not suitable for models with invisible transitions or duplicate activities
- It provides a global view but not detailed diagnostics

## 6.3 Alignments

Alignments provide a more sophisticated approach to conformance checking by finding the optimal match between traces in the log and possible execution paths in the model.

An alignment is a sequence of pairs (a,b) where:

- a is an activity in the log (or >> for "no match")
- b is an activity in the model (or >> for "no match")

The three types of moves in an alignment are:

1. **Synchronous move:** Both log and model execute the same activity (a,a)
2. **Move on log:** An activity occurs in the log but not in the model (a,>>)
3. **Move on model:** An activity occurs in the model but not in the log (>>,a)

The goal is to find an optimal alignment that minimizes the cost of non-synchronous moves. The cost function typically assigns:

- Cost 0 to synchronous moves
- Cost 1 to moves on log and moves on model
- Cost  $\infty$  to illegal moves (a,b) where  $a \neq b$  and  $a \neq >>$  and  $b \neq >>$

Fitness based on alignments is calculated as:  $\text{fitness} = 1 - (\text{cost of optimal alignment}) / (\text{cost of worst alignment})$

Where the worst alignment consists of:

- Moves on log for all events in the trace
- Moves on model for a shortest path from the initial to the final marking

Alignments provide detailed diagnostics about where and how traces deviate from the model.

## 6.4 Precision Metrics

Precision measures how much behavior the model allows that was not observed in the log.

The alignment-based approach to precision:

1. Replay the log on the model, creating a state space
2. At each state, count:
  - The number of activities actually observed to follow (used\_beh)
  - The number of activities allowed by the model to follow (allowed\_beh)
3. Calculate precision as the weighted average of used\_beh/allowed\_beh across all states

Formally:

$$\text{precision}(L,M) = \frac{\sum(s \in S) \text{freq}(s) \times (|\{s' \mid (s,s') \in E \text{ and } \text{freq}(s') > 0\}| / |\{s' \mid (s,s') \in E\}|)}{\sum(s \in S) \text{freq}(s)}$$

Where:

- S is the set of states
- E is the set of edges in the state space
- freq(s) is the frequency of state s in the log

A precision of 1.0 means the model only allows behavior observed in the log, while lower values indicate the model allows for more behavior than seen in the log.

## 7. Mining Additional Perspectives

### 7.1 Organizational Mining

Organizational mining extracts knowledge about resources (people, systems) involved in process execution. Key techniques include:

#### 1. **Resource-Activity Matrix:**

- Shows which resources perform which activities and how frequently
- Can be used to identify roles by clustering resources with similar profiles

#### 2. **Social Network Analysis:**

- Handover of work: Measures how frequently work is transferred from one resource to another
- Working together: Identifies resources that work on the same cases
- Similar task profiles: Finds resources that perform similar activities

#### 3. **Organizational Model Mining:**

- Discovers the organizational structure and roles
- Identifies groups and hierarchies

The handover of work relation is defined based on eventual dependencies between activities. For a trace  $\sigma$ , a handover between resources a and b exists if:

- a performs an activity that has an eventual dependency with an activity performed by b
- No other activity in between has an eventual dependency with the activity performed by b

This can be represented as a matrix showing the frequency of handovers between resources.

### 7.2 Decision Mining

Decision mining (also called decision point analysis) aims to discover the rules governing the choices made in a process.

The approach:

1. Identify decision points in the process model (e.g., places with multiple outgoing arcs in a Petri net)
2. For each decision point, collect data attributes available at the time the decision was made
3. Apply machine learning techniques (e.g., decision trees) to find rules that explain the choices

For example, at a decision point with branches to activities A and B, we collect cases where A was chosen and cases where B was chosen, along with their data attributes. A decision tree can then reveal rules like:

- If amount > 1000 then choose A
- If amount ≤ 1000 then choose B

Decision mining can:

- Make process models more precise by adding guards
- Provide insights into decision-making patterns
- Identify data attributes that influence process flow

## 7.3 Time Perspective

Time perspective mining extracts temporal patterns and performance information from event logs. Key aspects include:

1. **Processing Times (Service Times):** How long activities take to complete
  - Calculated as the difference between the completion and start timestamps of an activity
2. **Waiting Times:** How long cases wait between activities
  - Calculated as the time between the completion of one activity and the start of the next
3. **Cycle Times:** Total time from the start to the end of a case
  - Calculated as the difference between the timestamps of the first and last events
4. **Bottleneck Analysis:** Identifying activities or paths with long waiting times
5. **Resource Utilization:** How busy resources are over time

The approach typically involves:

1. Replaying the log on the model to align events to model elements
2. Collecting time statistics for activities, transitions, and places
3. Analyzing the distributions of these times
4. Visualizing the results using techniques like heatmaps

This information can be used to:

- Identify bottlenecks and inefficiencies
- Set realistic service level agreements
- Optimize resource allocation
- Improve process performance

## 8. BPMN and Process Simulation

### 8.1 BPMN Notation

Business Process Model and Notation (BPMN) is a standard graphical notation for business process models. The key elements are:

1. **Events:** Things that happen instantaneously
  - Start events: Trigger the process
  - End events: Indicate the end of the process
  - Intermediate events: Occur during the process
2. **Activities:** Work performed in the process
  - Tasks: Atomic activities
  - Sub-processes: Composite activities
3. **Gateways:** Control the flow of the process
  - Exclusive gateways (XOR): One path is chosen
  - Parallel gateways (AND): All paths are executed simultaneously
  - Inclusive gateways (OR): One or more paths are chosen
4. **Sequence Flows:** Connect activities to show the order of execution
5. **Pools and Lanes:** Represent organizations and roles/departments

BPMN uses token semantics similar to Petri nets:

- Tokens are created by start events
- They flow through the process
- They are removed by end events

BPMN can be mapped to Petri nets, though some BPMN constructs (like OR-joins and complex event handling) are more difficult to represent directly.

### 8.2 Elements of a Simulation Model

Process simulation extends process models with additional information to enable performance analysis:

1. **Processing Times of Activities:**
  - Fixed values or probability distributions (normal, exponential, etc.)



- The choice of distribution depends on the nature of the activity
2. **Branching Probabilities:**
    - Probability of taking each path at decision points
    - Can be derived from event logs
  3. **Arrival Rate of Process Instances:**
    - How frequently new cases arrive
    - Often modeled as an exponential distribution
    - May include a calendar (e.g., business hours only)
  4. **Resource Pools:**
    - Number of resources available
    - Working hours/calendar
    - Cost of resources
  5. **Assignment of Activities to Resources:**
    - Which resources can perform which activities
    - Priority rules for task assignment

These parameters can be estimated from event logs using process mining techniques.

## 8.3 Simulation Output Analysis

Process simulation generates various performance metrics:

1. **Cycle Time:** Total time to complete a process instance
  - Average, minimum, maximum, and distribution
2. **Activity Waiting Times:** How long activities wait before being executed
  - Identifies bottlenecks and resource constraints
3. **Resource Utilization:** Percentage of time resources are busy
  - Helps identify over/under-utilized resources
4. **Costs:** Total process costs based on resource usage and activity costs
5. **Throughput:** Number of cases completed per time unit

The results can be visualized using:

- Histograms: Show distributions of times
- Heatmaps: Highlight bottlenecks or high-cost activities
- Resource utilization charts: Show how busy resources are over time

Simulation supports "what-if" analysis by comparing different scenarios:

- What if we add more resources?
- What if we change the process structure?
- What if the arrival rate increases?

## 8.4 Pitfalls of Simulation

Process simulation has several challenges and limitations:

### 1. **Stochasticity:**

- Results vary between simulation runs
- Need multiple runs to get reliable results
- Should calculate confidence intervals
- Need to account for warm-up and cool-down periods

### 2. **Data Quality:**

- Simulation results are only as good as the input data
- Estimation of parameters may be inaccurate
- The "closed-world assumption" might not hold

### 3. **Model Simplification:**

- Models necessarily simplify reality
- Some dependencies and constraints might be missed
- Human behavior is complex and not always predictable

### 4. **Validation:**

- Difficult to validate simulation results against reality
- Should compare simulation of current situation with actual performance

Best practices include:

- Using multiple simulation runs
- Calculating confidence intervals
- Removing warm-up and cool-down periods
- Validating the model against historical data before making predictions
- Being cautious about the limitations and assumptions of the model

---

## Exam Format and Preparation

The course evaluation consists of:

### 1. **Written Exam (60% of the grade):**

- Covers theoretical concepts and exercise applications
- Includes questions on Petri nets, process discovery, conformance checking, and additional perspectives

### 2. **Project (40% of the grade):**

- Practical application of process mining techniques
- Requires a minimum grade of 18/30 to be admitted to the written exam

Sample exam questions typically cover:

- Process modeling using Workflow Nets
- Process discovery using the Alpha algorithm
- Region-based mining
- Calculating model precision
- Analyzing organizational perspectives

Preparation tips:

- Practice creating and analyzing Petri nets for soundness
- Understand the limitations of different process discovery algorithms
- Be able to apply conformance checking techniques manually
- Know how to interpret process mining results from different perspectives