GameSS

EDUCATIONAL MATERIAL IN CYBER SECURITY

WHO IS THE MATERIAL FOR?

This module targets students, professionals, and researchers interested in better understanding how behavior can be modeled, discovered, and monitored in the context of security.

While no specific background is required, some general knowledge of business processes and some basics of data manipulation can be helpful to fully appreciate the module.

WHO MADE THIS MATERIAL?

- Andrea Burattin

  Associate Professor, DTU Compute

- Gemma Di Federico

  Postdoc, DTU Compute

WHAT ARE THE MAIN TAKEAWAYS FOR THIS CONTENT?

- Participants will learn how to model a process and how to extract it from event data using automated techniques. Once such a process model is available, another goal is to verify the presence of deviations.

INTRODUCTION

- Security aspects are not exclusively limited to automated systems. It is constantly necessary to take into consideration the human aspects and the ability of many systems to offer flexibility.
- However, flexible systems pose challenges in terms of security. For this reason, often, expected behaviors are modeled using processes that describe the expected flow of operations and the extent to which deviations can be acceptable. At the same time, systems implementing these processes leave traces of the executions, thus enabling forensics investigations.
- During the course, we will investigate how process mining techniques can help study the execution of processes as these are recorded. Textual descriptions of security processes will be presented together with some event logs referring to corresponding executions.

STRUCTURE OF THE LECTURE

- Process modelling
- Basics of process mining
  - Introduction
  - Discovery
  - Conformance

BASICS OF PROCESS MODELLING

The common denominator

- Common denominator of all aspects is an (implicit) notion of *process*

The common denominator

- Common denominator of all aspects is an (implicit) notion of *process*

- What is a process?

The common denominator

- Common denominator of all aspects is an (implicit) notion of *process*

- What is a process?
- *“A business process consists of a set of activities that are performed in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations.”*

M. Weske. Business Process Management: Concepts, Languages, Architectures. Springer, 2007.

The common denominator

- Common denominator of all aspects is an (implicit) notion of *process*

- What is a process?
- *"A business process consists of a set of activities that are performed in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations."*

M. Weske. Business Process Management: Concepts, Languages, Architectures. Springer, 2007.

- *"We define a business process as a collection of inter-related events, activities and decision points that involve a number of actors and objects, and that collectively lead to an outcome that is of value to at least one customer."*

M. Dumas, M. La Rosa, J. Mendling, H. Reijers. Fundamentals of Business Process Management. Springer, 2013.

The prerequisites

- In the data we want to analyse there is the notion of process, either implicit or explicit
  - In our context, a process is the structuring of activities underpinning how certain goals are achieved
  - This will drive the types of analyses we want / can perform

- Our processes are performed with the support of information system such that all our executions are traceable

The prerequisites

- In the data we want to analyse there is the notion of process, either implicit or explicit
  - In our context, a process is the structuring of activities underpinning how certain goals are achieved
  - This will drive the types of analyses we want / can perform

- Our processes are performed with the support of information system such that all our executions are traceable

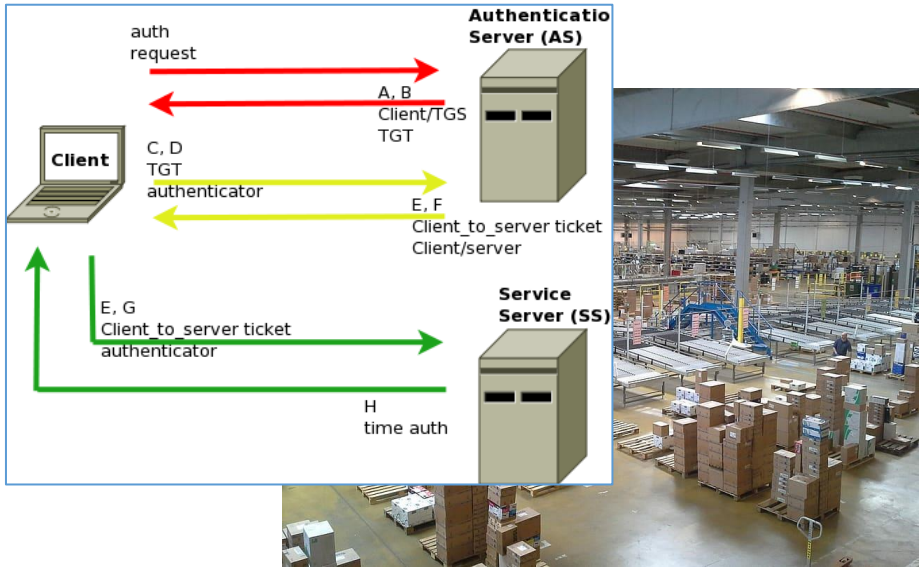
More structured

More knowledge-intensive

The prerequisites

- In the data we want to analyse there is the notion of process, either implicit or explicit
  - In our context, a process is the structuring of activities underpinning how certain goals are achieved
  - This will drive the types of analyses we want / can perform

- Our processes are performed with the support of information system such that all our executions are traceable



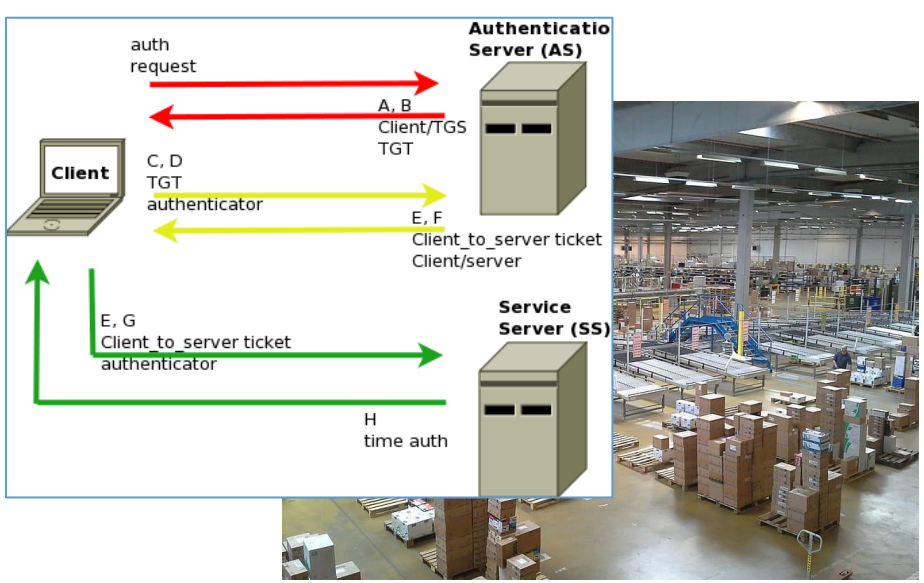
More structured

More knowledge-intensive

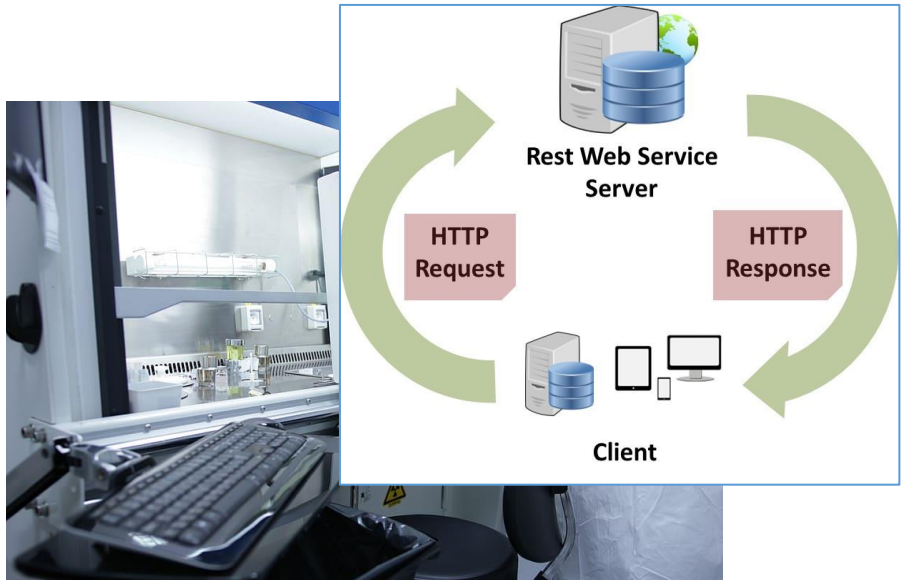
The prerequisites

- In the data we want to analyse there is the notion of process, either implicit or explicit
  - In our context, a process is the structuring of activities underpinning how certain goals are achieved
  - This will drive the types of analyses we want / can perform

- Our processes are performed with the support of information system such that all our executions are traceable



More structured



More knowledge-intensive

What is a model?

What is a model?

- *A **model** is an **abstraction** of reality according to a certain conceptualization. Once represented as a concrete artifact, a model can support communication, learning and analysis about relevant aspects of the underlying domain.* [...]

What is a model?

- *A **model** is an **abstraction** of reality according to a certain conceptualization. Once represented as a concrete artifact, a model can support communication, learning and analysis about relevant aspects of the underlying domain.* [...]
- *A **represented model** (a dusty diagram) created by an unknown predecessor is a **medium** to preserve and communicate a certain view of the world, and can serve as a vehicle for reasoning and problem solving, and for acquiring new knowledge (maybe having striking new ideas!) about this view of the world.*

What is a model?

- *A **model** is an **abstraction** of reality according to a certain conceptualization. Once represented as a concrete artifact, a model can support communication, learning and analysis about relevant aspects of the underlying domain.* [...]
- *A **represented model** (a dusty diagram) created by an unknown predecessor is a **medium** to preserve and communicate a certain view of the world, and can serve as a vehicle for reasoning and problem solving, and for acquiring new knowledge (maybe having striking new ideas!) about this view of the world.*

-- G. Guizzardi. *Ontological foundations for structural conceptual models*.
CTIT, Centre for Telematics and Information Technology, 2005.

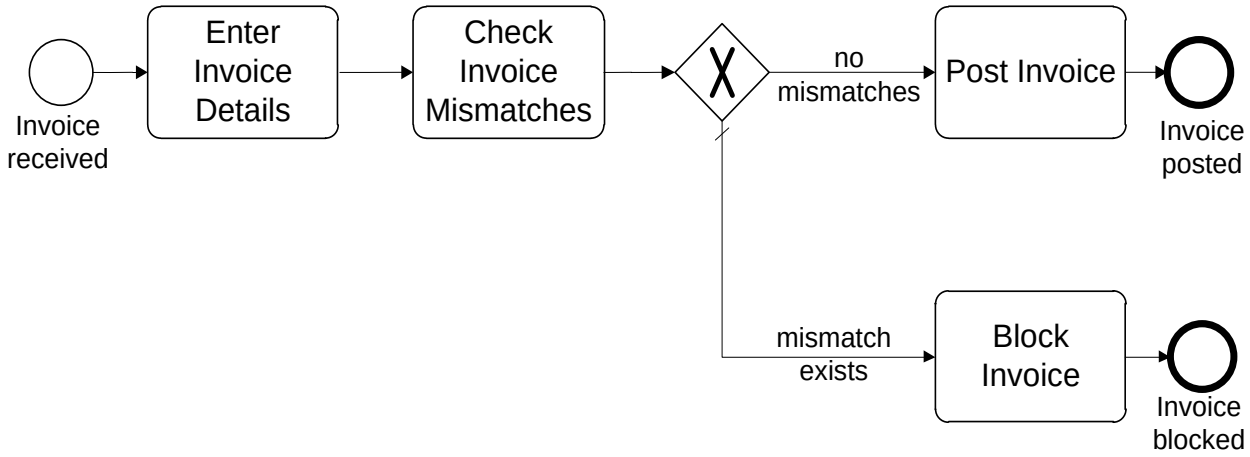
Different perspectives

Different perspectives

- What needs be done and when? – *Control flow*

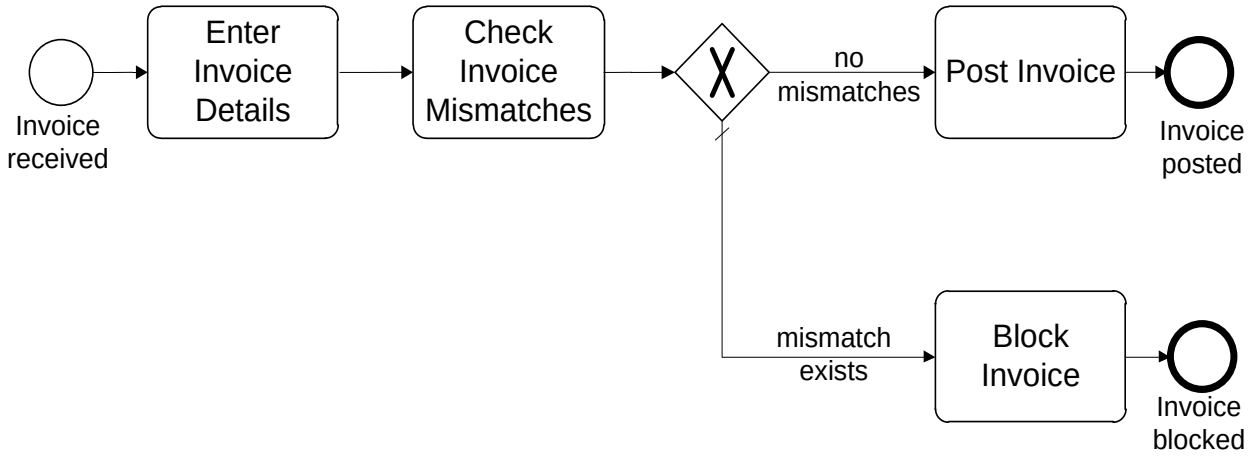
Different perspectives

- What needs be done and when? – *Control flow*



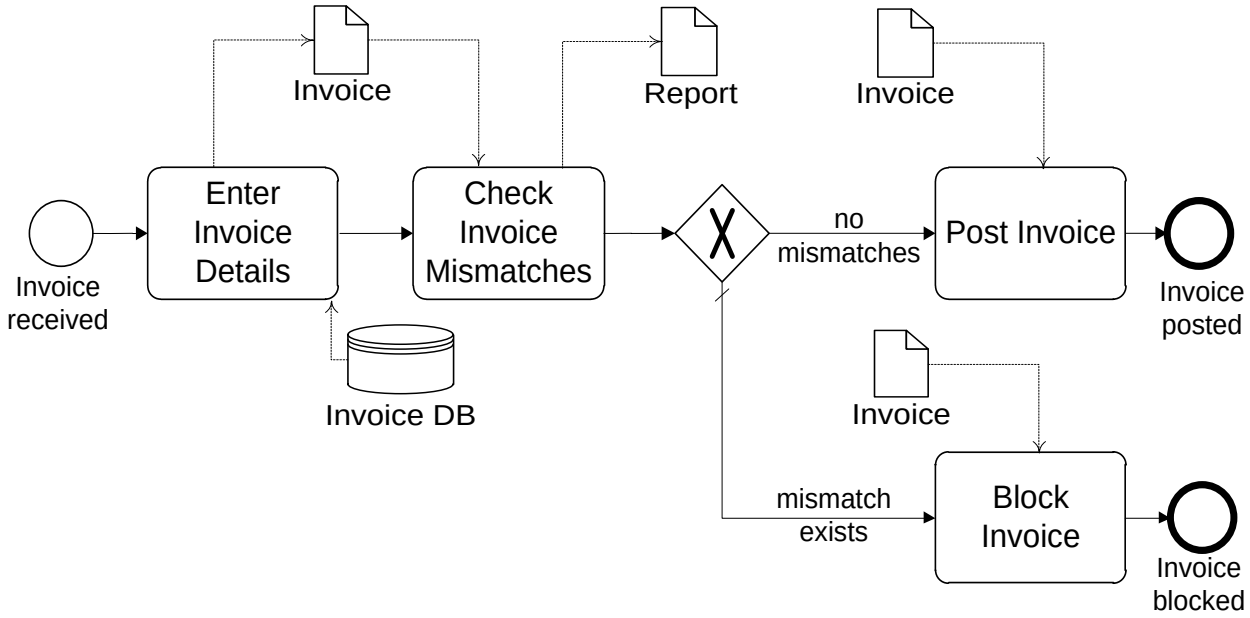
Different perspectives

- What needs be done and when? – *Control flow*
- What do we need to work on? – *Data*



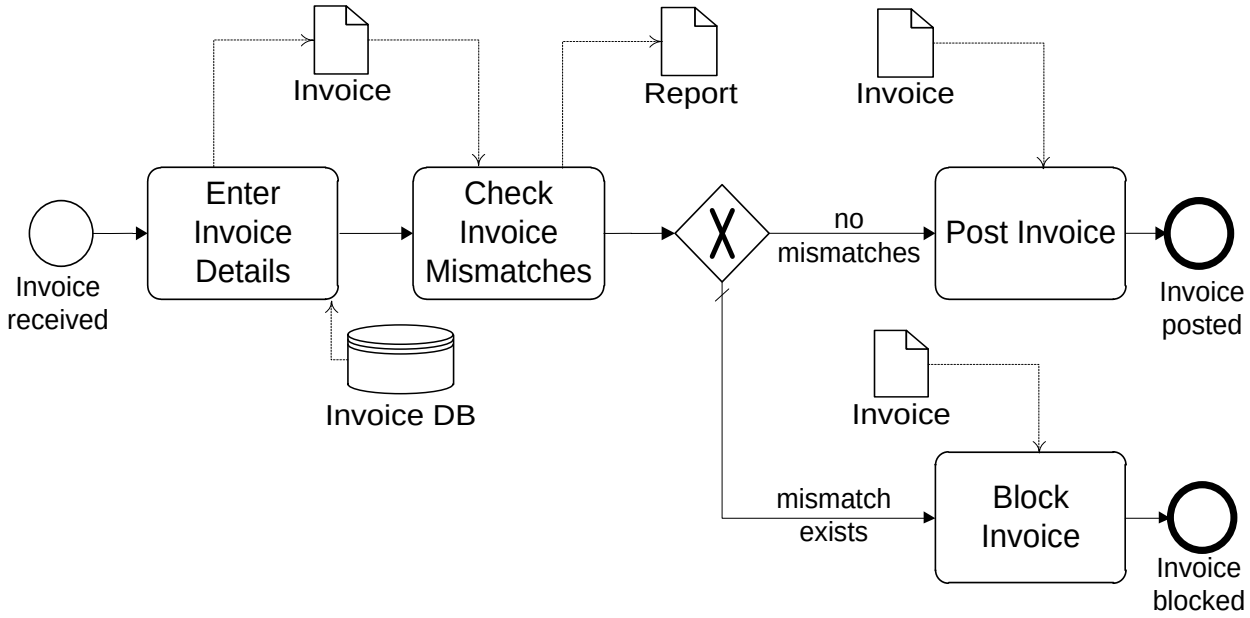
Different perspectives

- What needs be done and when? – *Control flow*
- What do we need to work on? – *Data*



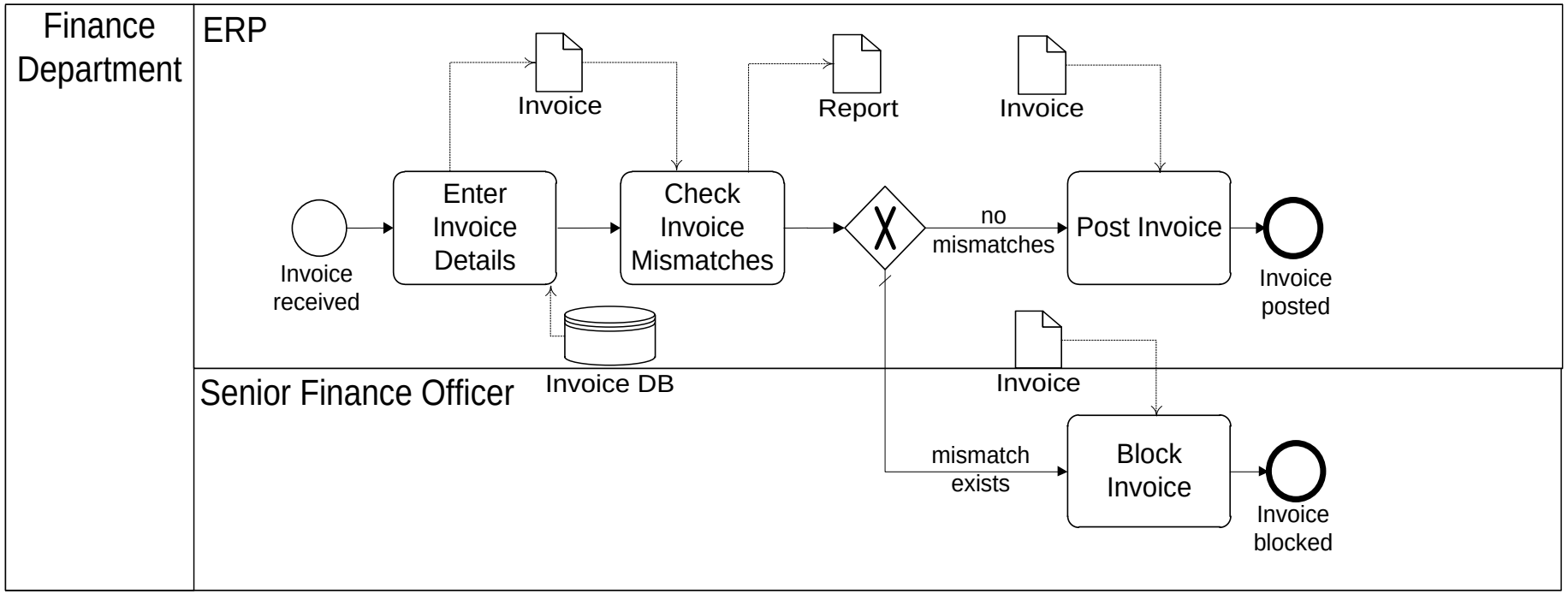
Different perspectives

- What needs be done and when? – *Control flow*
- What do we need to work on? – *Data*
- Who's doing the work? – *Resources (human & systems)*



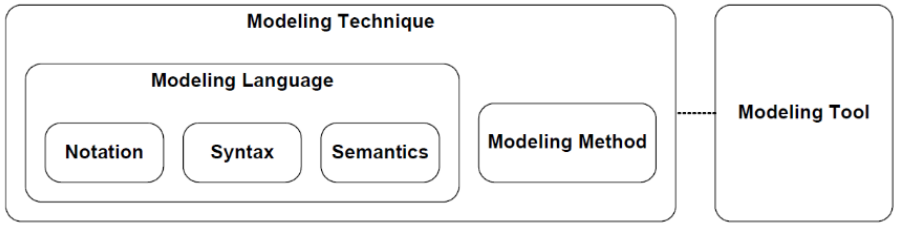
Different perspectives

- What needs be done and when? – *Control flow*
- What do we need to work on? – *Data*
- Who's doing the work? – *Resources (human & systems)*

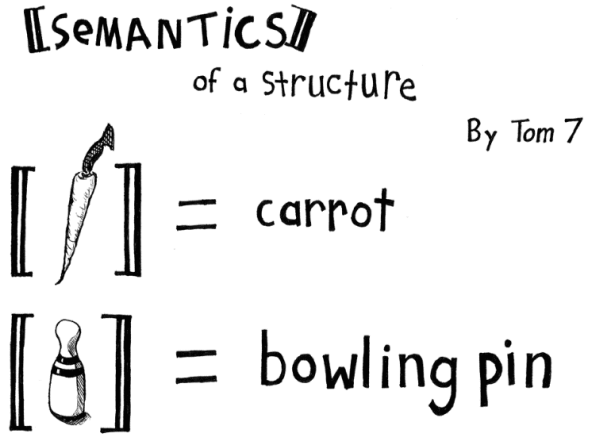


Behavioural Formalisms

- There is a zoo of process modeling languages
  - Different expressiveness
  - Different syntax, semantics, notation



- Precise definition of syntax and semantics is a prerequisite
  - Structure and meaning of model needs to be unambiguous



BPMN – Business Process Modeling Notation

Modeling language for processes, based on popular graphical flowcharts:

- Core set of notation elements
- Each core element has various subtypes

BPMN – Business Process Modeling Notation

Modeling language for processes, based on popular graphical flowcharts:

- Core set of notation elements
- Each core element has various subtypes

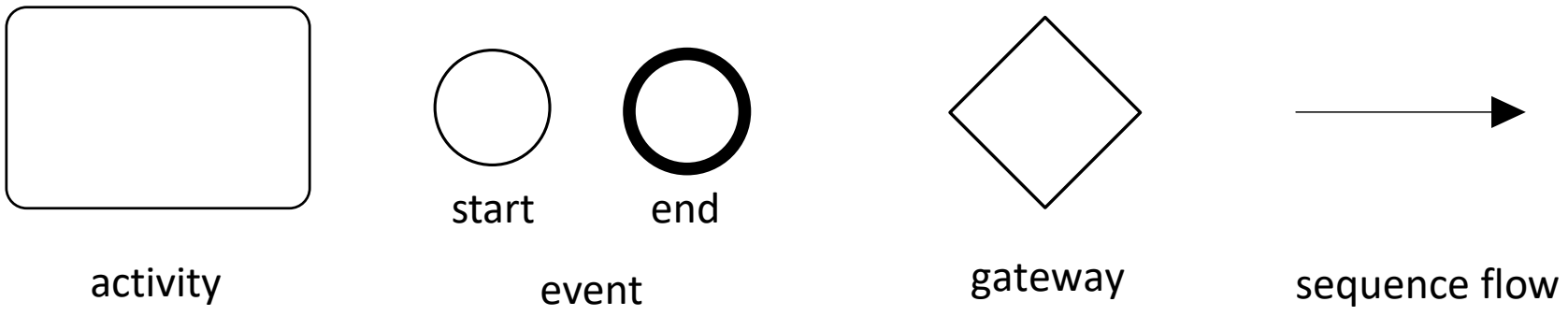
- A BPMN process model is a graph consisting of four types of **core elements**:

BPMN – Business Process Modeling Notation

Modeling language for processes, based on popular graphical flowcharts:

- Core set of notation elements
- Each core element has various subtypes

- A BPMN process model is a graph consisting of four types of **core elements**:



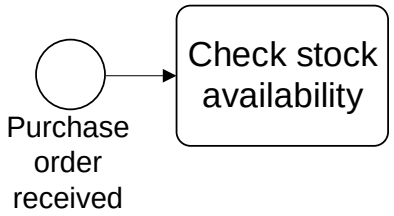
Let's consider an order to cash process

- A typical order-to-cash process is triggered by the receipt of a purchase order from a customer.
- The purchase order has to be checked against the stock regarding the availability of the item(s) requested. Depending on stock availability, the purchase order may be confirmed or rejected. If the purchase order is confirmed, an invoice is emitted, and the requested goods are shipped.
- The process is completed by archiving the order or if the order is rejected.

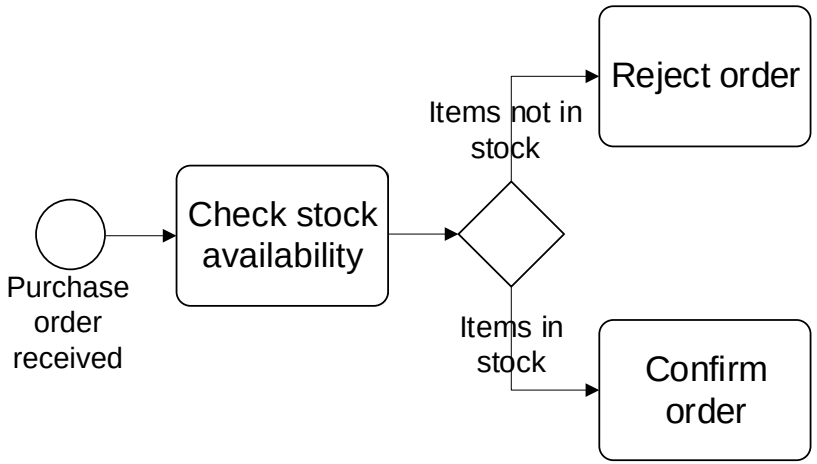
Solution in BPMN – Order to cash

Solution in BPMN – Order to cash

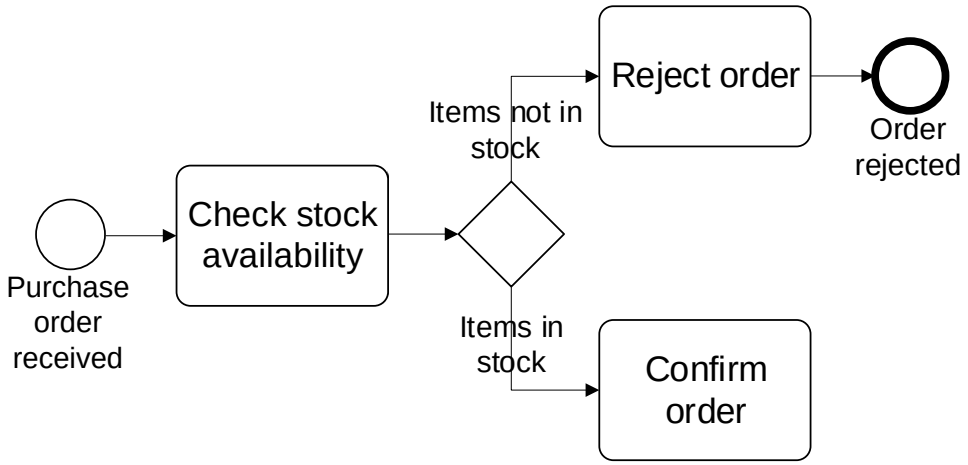
Solution in BPMN – Order to cash



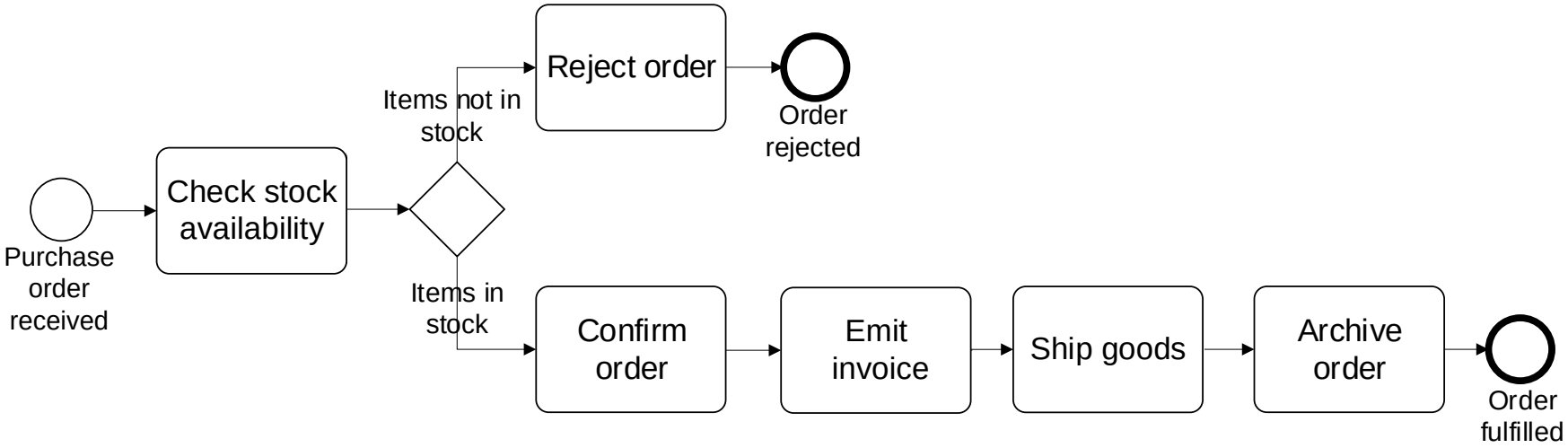
Solution in BPMN – Order to cash



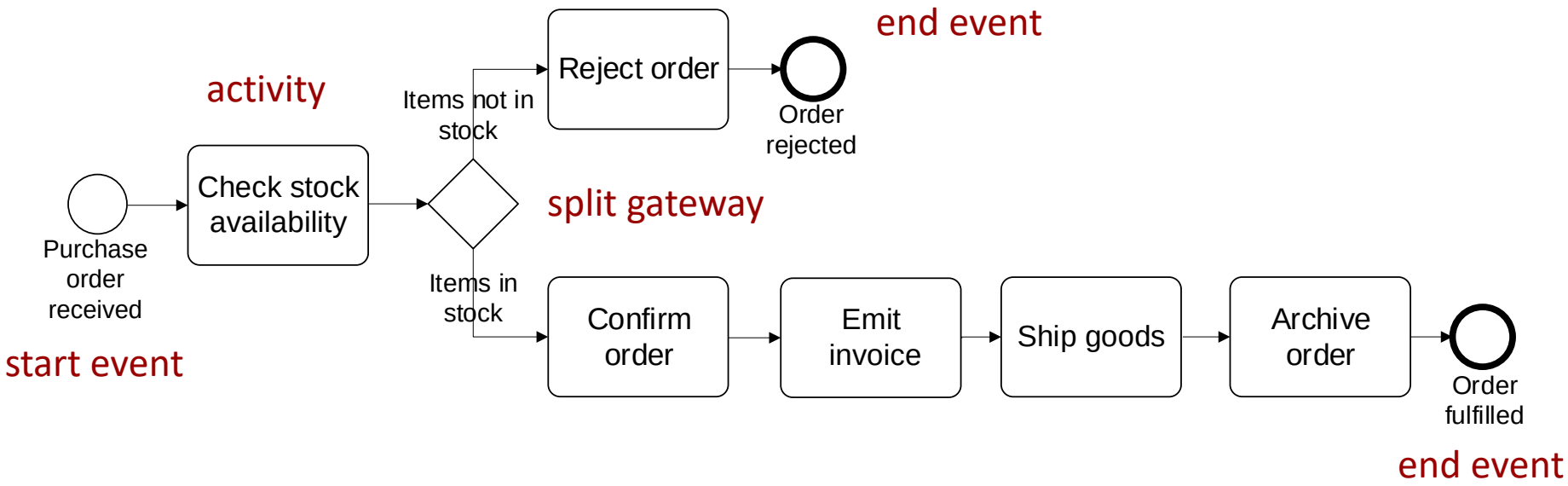
Solution in BPMN – Order to cash



Solution in BPMN – Order to cash



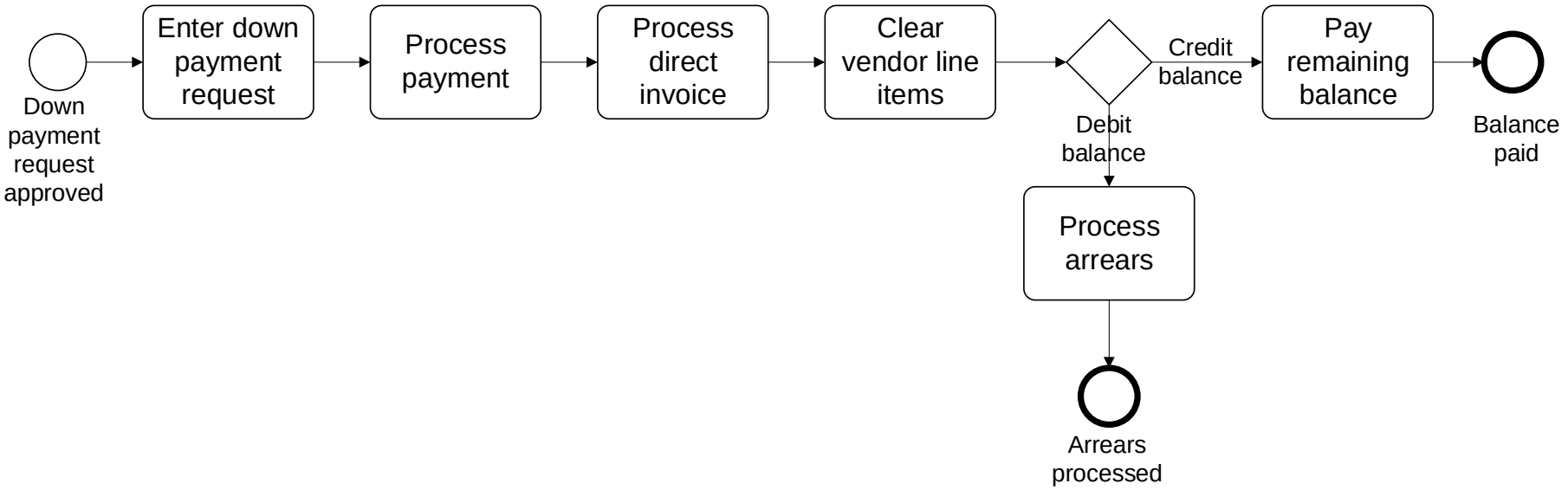
Solution in BPMN – Order to cash



One more example - Handle down payments

- This process starts when a request for down payment has been approved.
- It involves the entry and posting of a down payment in the form of a down payment request being entered into the system, the automatic subsequent payment, acquittal of the down payment through the processing of the direct invoice and the clearance of the vendor line items.
- The clearance of the vendor line items can result in a debit or credit balance. In case of debit balance, the arrears are processed, otherwise the remaining balance is paid.

Solution in BPMN - Handle down payments



BPMN core elements

- Activities
  - Represent work that has to be done in a process
  - There can be different types of activities

activity

BPMN core elements

- Activities
  - Represent work that has to be done in a process
  - There can be different types of activities

activity

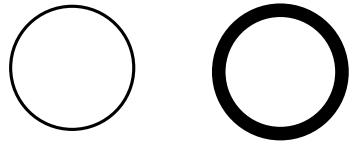
- Events

BPMN core elements

- Activities
  - Represent work that has to be done in a process
  - There can be different types of activities

activity

- Events
  - Represents conditions or outcomes of the process
  - There can be different types of events
    - A start event triggers a new process instance
    - An end event indicates that a process instance has completed with a certain outcome

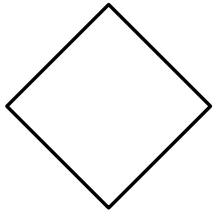


start event

end event

BPMN core elements

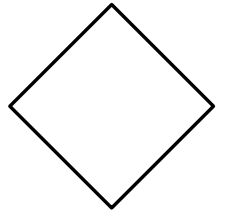
- Gateways
  - Describe how the control-flow forks and joins
  - There can be different types of gateways



gateway

BPMN core elements

- Gateways
  - Describe how the control-flow forks and joins
  - There can be different types of gateways

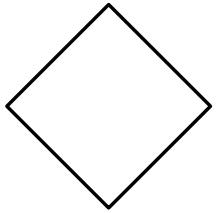


gateway

- Sequence flows

BPMN core elements

- Gateways
  - Describe how the control-flow forks and joins
  - There can be different types of gateways

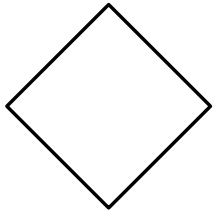


gateway

- Sequence flows
  - Model the order in which activities and events are performed

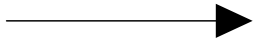
BPMN core elements

- Gateways
  - Describe how the control-flow forks and joins
  - There can be different types of gateways



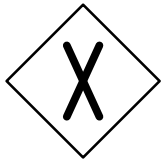
gateway

- Sequence flows
  - Model the order in which activities and events are performed
  - Sequence flows can have a condition to distinguish alternative branches

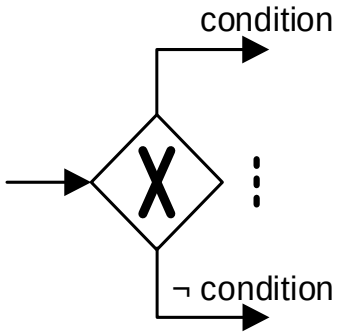


sequence flow

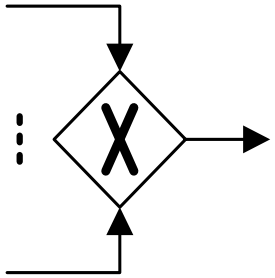
A little more on gateways: XOR Gateway



An *XOR Gateway* is used to make decisions and merged behaviour

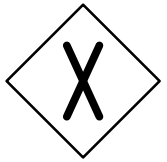


XOR-split: **one** outgoing branch will continue

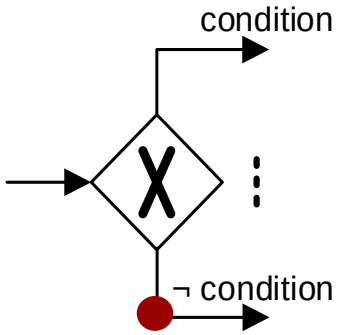


XOR-join: waits for **one** incoming branch

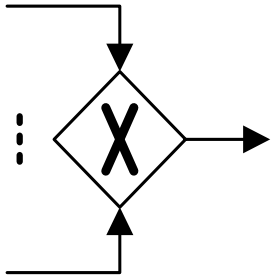
A little more on gateways: XOR Gateway



An *XOR Gateway* is used to make decisions and merged behaviour

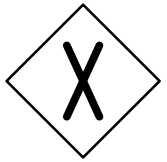


XOR-split: **one** outgoing branch will continue

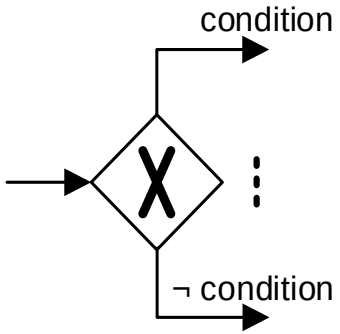


XOR-join: waits for **one** incoming branch

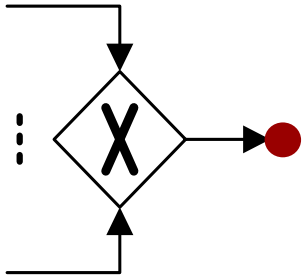
A little more on gateways: XOR Gateway



An *XOR Gateway* is used to make decisions and merged behaviour

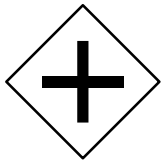


XOR-split: **one** outgoing branch will continue

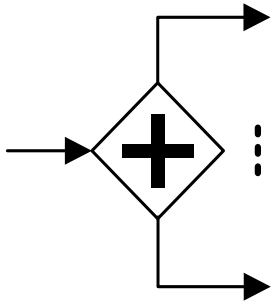


XOR-join: waits for **one** incoming branch

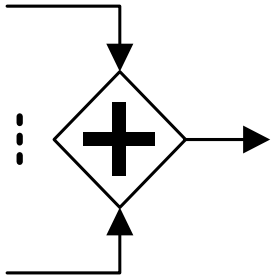
A little more on gateways: AND Gateway



An *AND Gateway* is used to handle “parallel” flows

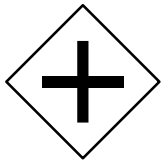


AND-split: **all** outgoing branches will continue

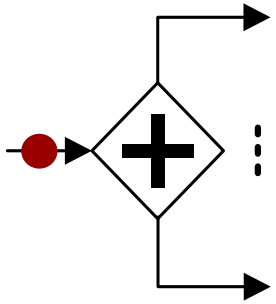


AND-join: waits for **all** incoming branches

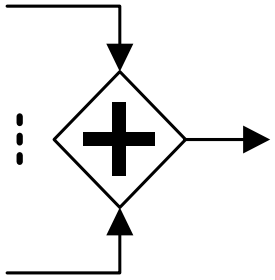
A little more on gateways: AND Gateway



An *AND Gateway* is used to handle “parallel” flows

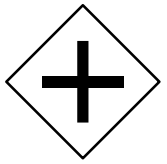


AND-split: **all** outgoing branches will continue

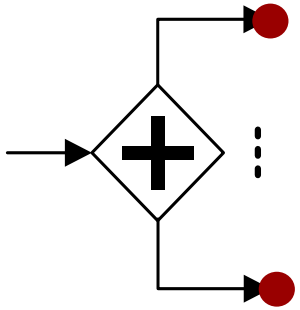


AND-join: waits for **all** incoming branches

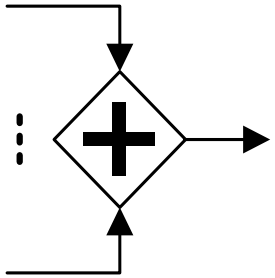
A little more on gateways: AND Gateway



An *AND Gateway* is used to handle “parallel” flows

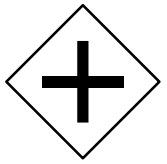


AND-split: **all** outgoing branches will continue

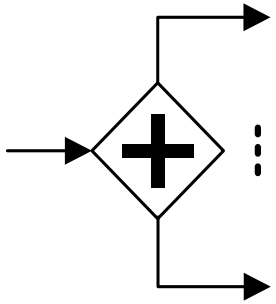


AND-join: waits for **all** incoming branches

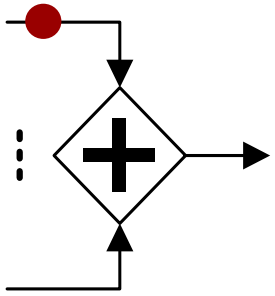
A little more on gateways: AND Gateway



An *AND Gateway* is used to handle “parallel” flows

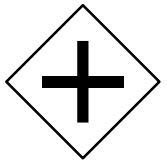


AND-split: **all** outgoing branches will continue

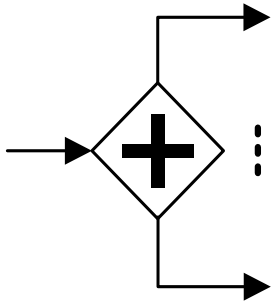


AND-join: waits for **all** incoming branches

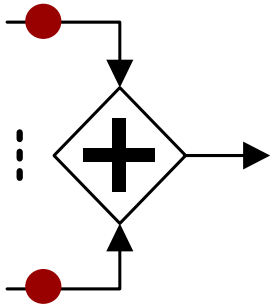
A little more on gateways: AND Gateway



An *AND Gateway* is used to handle “parallel” flows

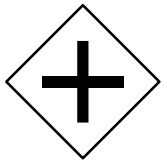


AND-split: **all** outgoing branches will continue

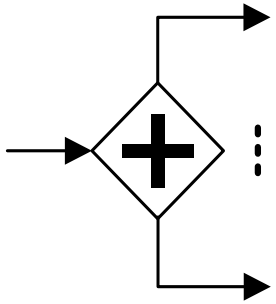


AND-join: waits for **all** incoming branches

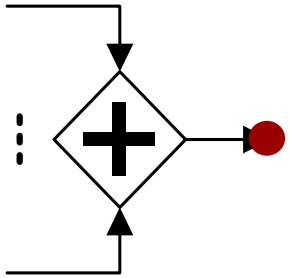
A little more on gateways: AND Gateway



An *AND Gateway* is used to handle “parallel” flows



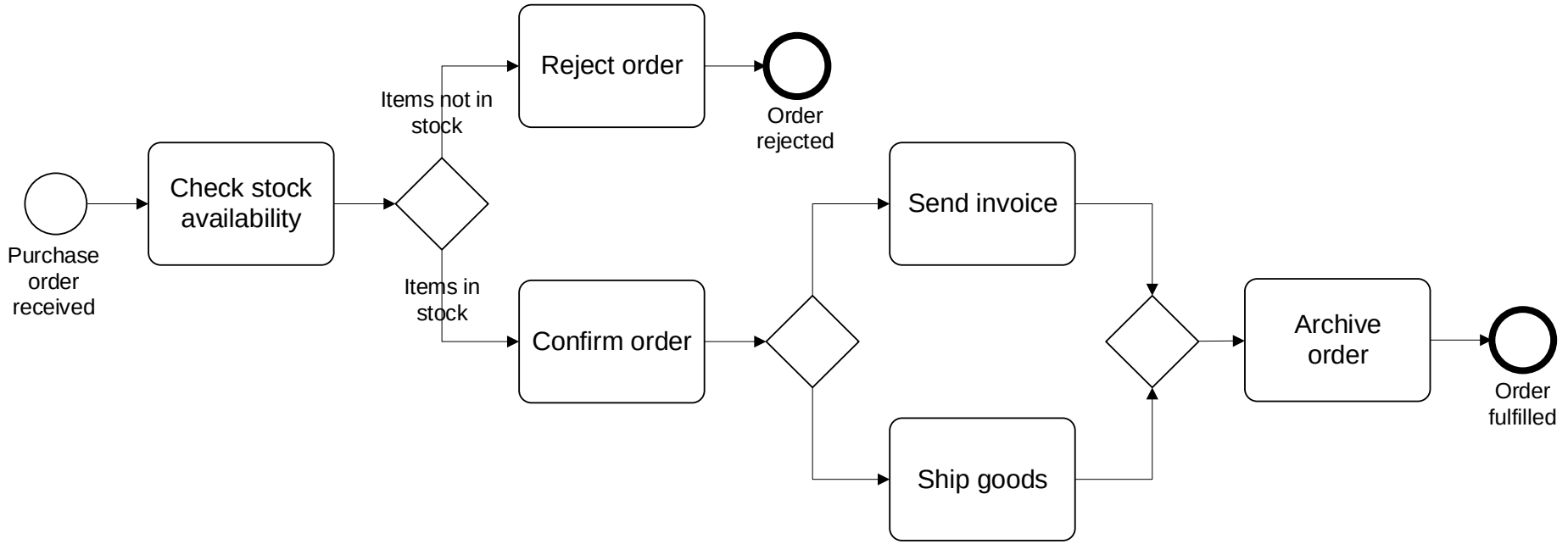
AND-split: **all** outgoing branches will continue



AND-join: waits for **all** incoming branches

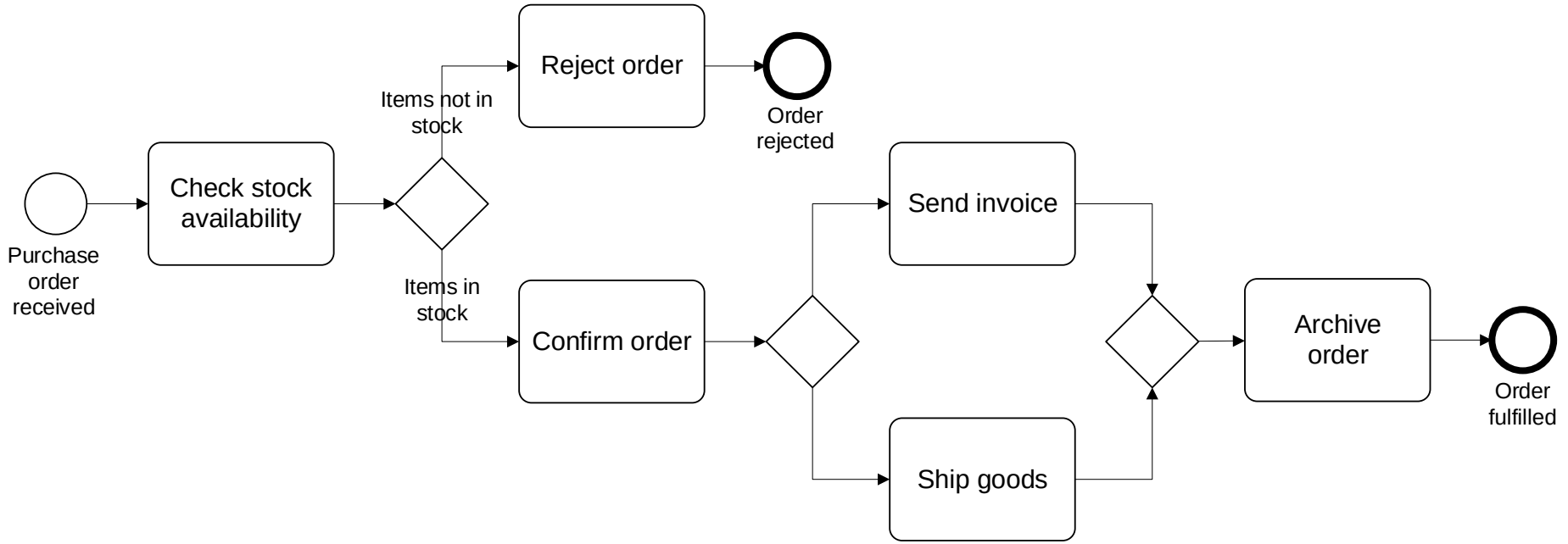
Let's reconsider our order-to-cash example

[…] If the purchase order is confirmed, an invoice is emitted and the goods requested are shipped. The process completes by archiving the order. […]



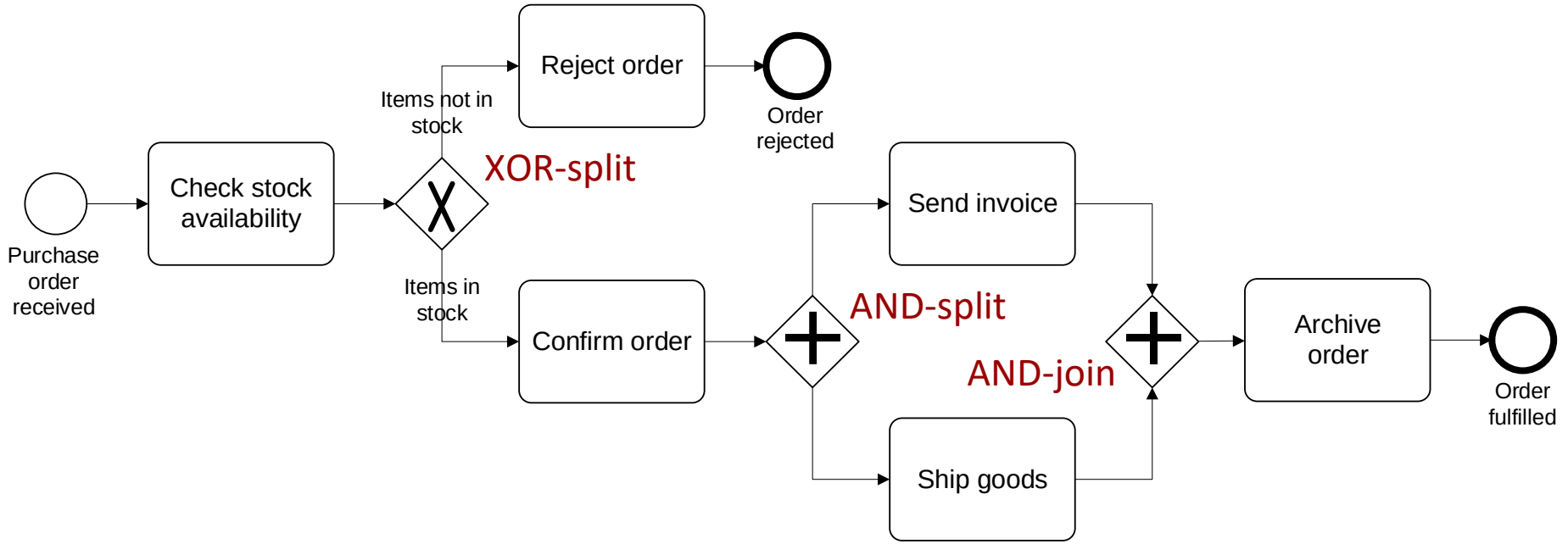
Let's reconsider our order-to-cash example

[…] If the purchase order is confirmed, an invoice is emitted and the goods requested are shipped. The process completes by archiving the order. […]



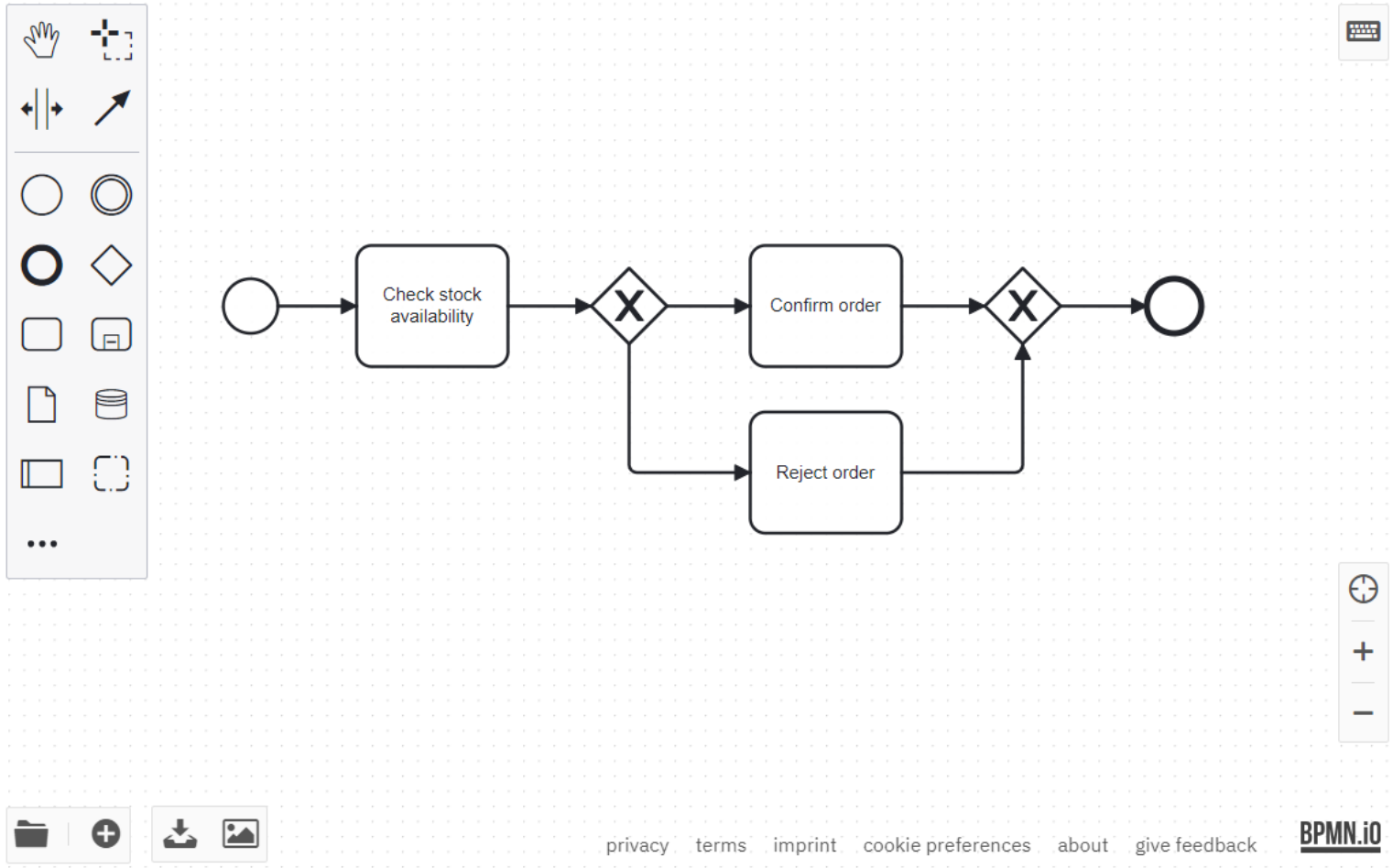
Let's reconsider our order-to-cash example

[…] If the purchase order is confirmed, an invoice is emitted and the goods requested are shipped. The process completes by archiving the order. […]



A platform to build BPMN diagrams

- Camunda/BPMN.io is an open source platform available at https://demo.bpmn.io/



BASICS OF PROCESS MINING

INTRODUCTION

What is process mining?

On the one hand, data science is an established discipline focusing on data processing and analysis.

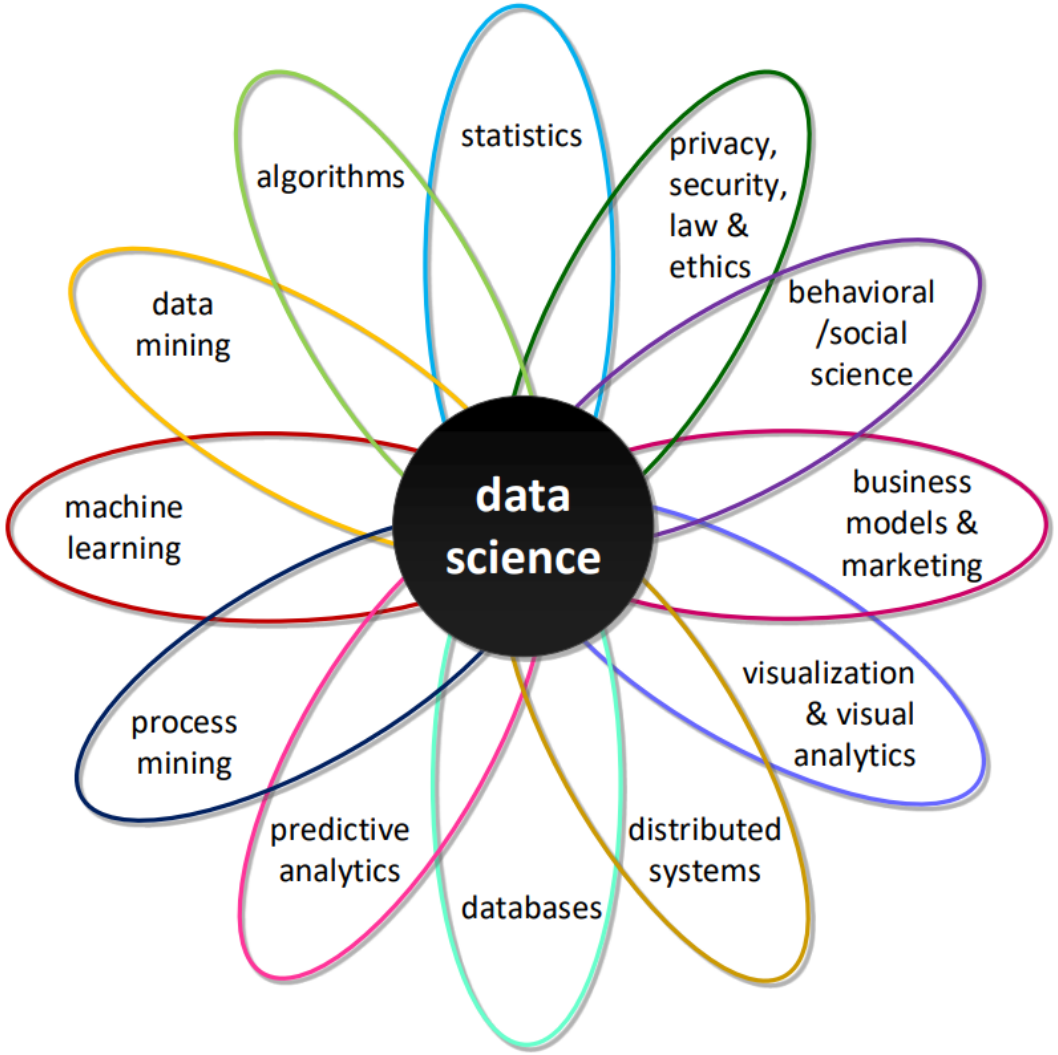


Image source and ©: Wil van der Aalst.

What is process mining?

On the other hand, process science focuses on the formalization, optimization, and study of processes.

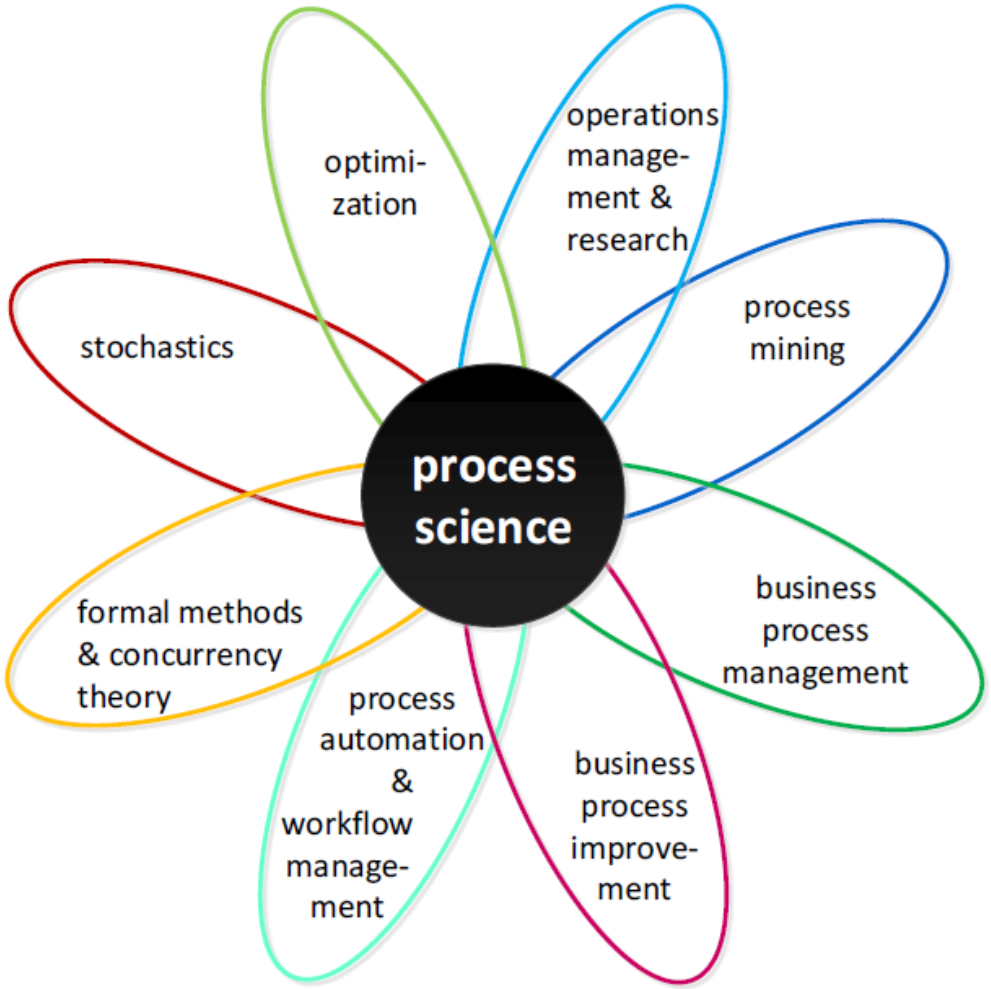


Image source and ©: Wil van der Aalst.

Process mining as the missing link

Process mining allows to leverage data science techniques to improve aspects related to the process dimensions.

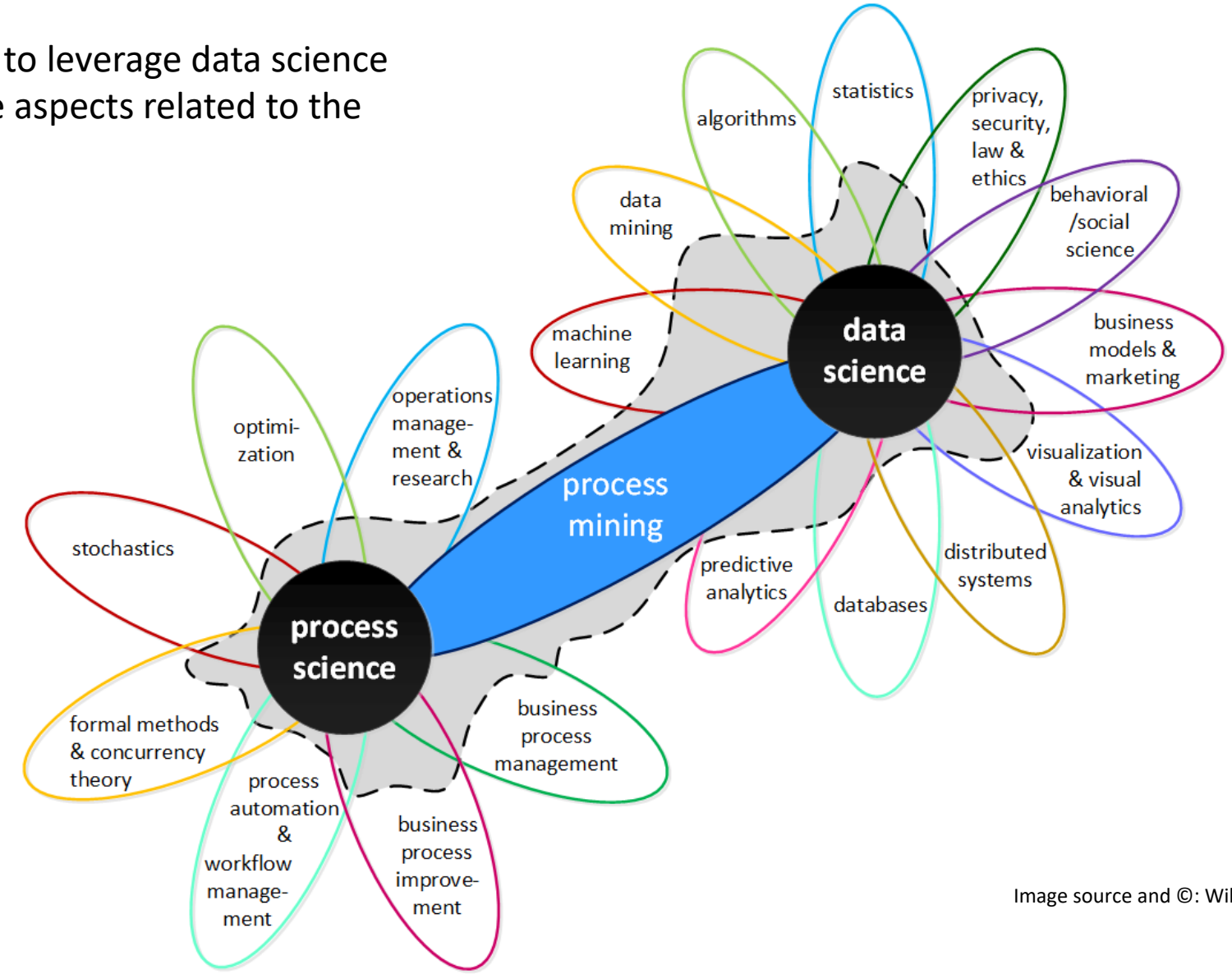


Image source and ©: Wil van der Aalst.

Events in Process Mining

- Event – *happening of interest*
  - Has timestamp: occurrence time, arrival time, …
  - Carries data

- Event type – type for events of similar structure and meaning
  - An *event type* is the set of attributes of the events
  - Events are *instances* of *event types*

Event Logs

- Process context
  - Activity – what has been executed?
  - Time – when has it been executed?
  - Case – for which process instance has it been executed?

Case ID	Activity	Start Time	End Time	Resource
8287	Enter customer data	08:34:15	08:37:44	User jsmith
8287	Check credit	08:37:52	08:38:05	Equifax service call
1399	Enter customer data	08:37:59	08:44:40	User sjones
8287	Enter order	08:38:09	08:38:39	ERP system call
1399	Check credit	08:44:58	08:45:06	Equifax service call
4283	Enter order	08:45:01	08:45:35	ERP system call
1399	Enter order	08:45:18	08:45:38	ERP system call

Example of event log

- A log consists of **cases** or **process instances**

Example of event log

- A log consists of **cases** or **process instances**
- A case consists of **events** and one event relates to one case

Example of event log

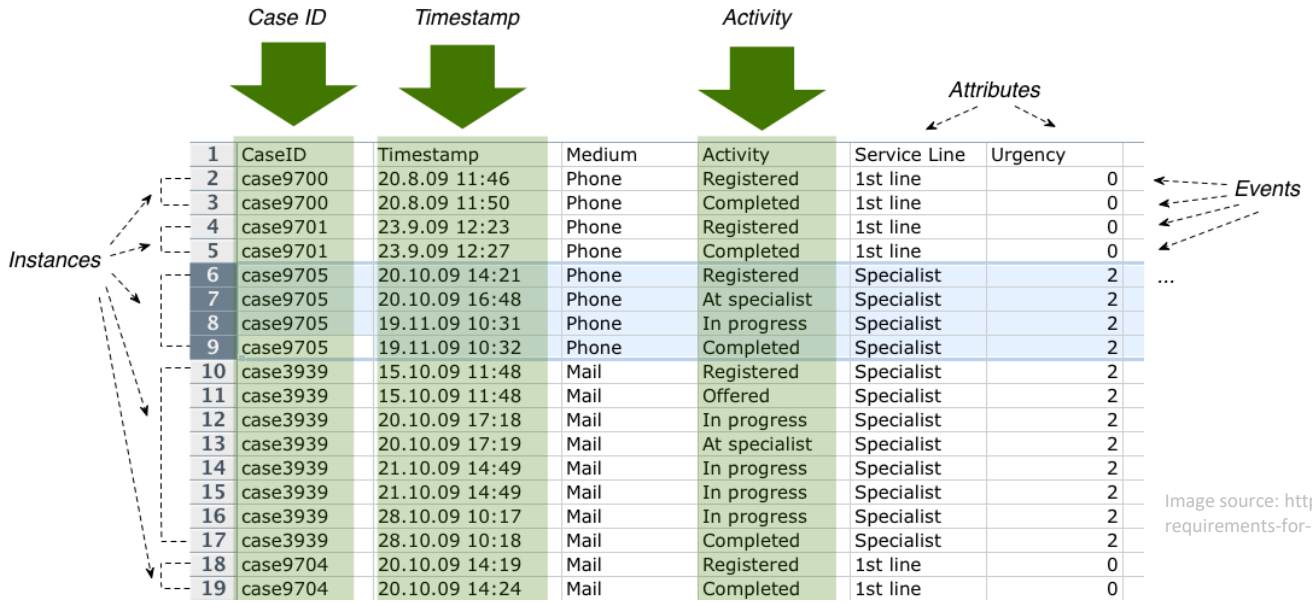
- A log consists of **cases** or **process instances**
- A case consists of **events** and one event relates to one case
- Events within a case are **ordered**

Example of event log

- A log consists of **cases** or **process instances**
- A case consists of **events** and one event relates to one case
- Events within a case are **ordered**
- Events can have **attributes**
  - Examples of typical attribute names are **activity**, **time**, costs, and resource

Example of event log

- A log consists of **cases** or **process instances**
- A case consists of **events** and one event relates to one case
- Events within a case are **ordered**
- Events can have **attributes**
  - Examples of typical attribute names are **activity**, **time**, costs, and resource



	CaseID	Timestamp	Medium	Activity	Service Line	Urgency
1						
2	case9700	20.8.09 11:46	Phone	Registered	1st line	0
3	case9700	20.8.09 11:50	Phone	Completed	1st line	0
4	case9701	23.9.09 12:23	Phone	Registered	1st line	0
5	case9701	23.9.09 12:27	Phone	Completed	1st line	0
6	case9705	20.10.09 14:21	Phone	Registered	Specialist	2
7	case9705	20.10.09 16:48	Phone	At specialist	Specialist	2
8	case9705	19.11.09 10:31	Phone	In progress	Specialist	2
9	case9705	19.11.09 10:32	Phone	Completed	Specialist	2
10	case3939	15.10.09 11:48	Mail	Registered	Specialist	2
11	case3939	15.10.09 11:48	Mail	Offered	Specialist	2
12	case3939	20.10.09 17:18	Mail	In progress	Specialist	2
13	case3939	20.10.09 17:19	Mail	At specialist	Specialist	2
14	case3939	21.10.09 14:49	Mail	In progress	Specialist	2
15	case3939	21.10.09 14:49	Mail	In progress	Specialist	2
16	case3939	28.10.09 10:17	Mail	In progress	Specialist	2
17	case3939	28.10.09 10:18	Mail	Completed	Specialist	2
18	case9704	20.10.09 14:19	Mail	Registered	1st line	0
19	case9704	20.10.09 14:24	Mail	Completed	1st line	0

Image source: https://fluxicon.com/blog/2012/02/data-requirements-for-process-mining/

Event data might come from any source and format

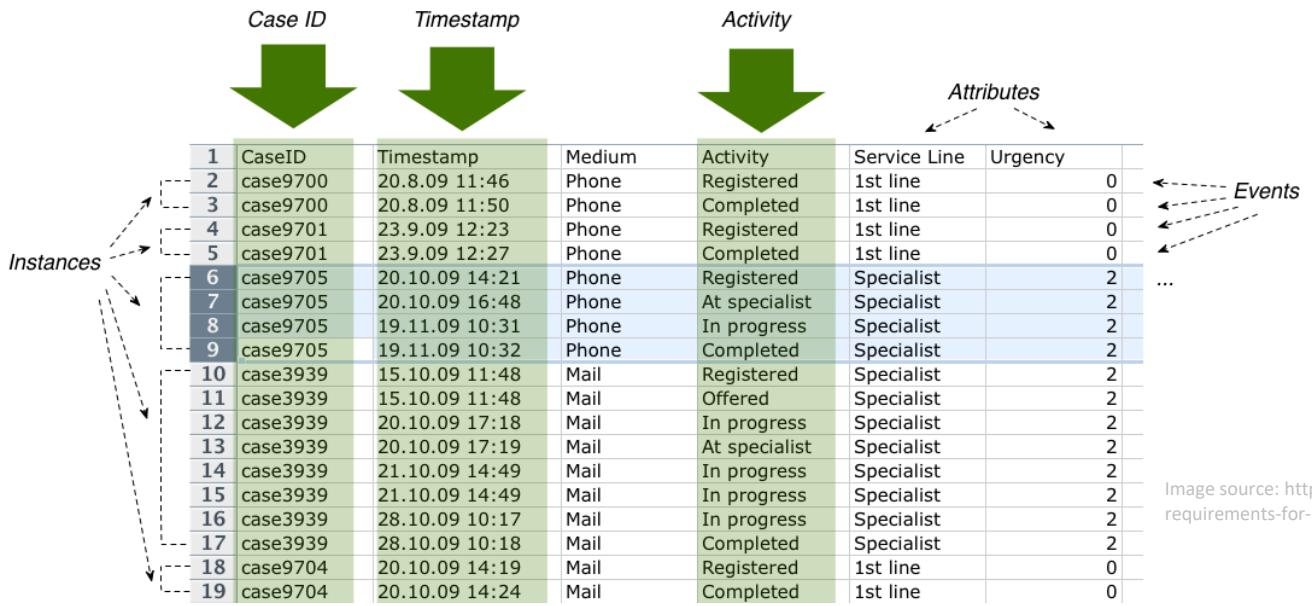
- A database system (e.g., patients in a hospital)
- Transaction logs (e.g., a trading system)
- ERP systems (e.g., Oracle, SAP)
- API to social media/websites (e.g., Twitter or Facebook)

- CSV files
- Spreadsheet
- …

Event data might come from any source and format

- A database system (e.g., patients in a hospital)
- Transaction logs (e.g., a trading system)
- ERP systems (e.g., Oracle, SAP)
- API to social media/websites (e.g., Twitter or Facebook)

- CSV files
- Spreadsheet
- …



	CaseID	Timestamp	Medium	Activity	Service Line	Urgency
1						
2	case9700	20.8.09 11:46	Phone	Registered	1st line	0
3	case9700	20.8.09 11:50	Phone	Completed	1st line	0
4	case9701	23.9.09 12:23	Phone	Registered	1st line	0
5	case9701	23.9.09 12:27	Phone	Completed	1st line	0
6	case9705	20.10.09 14:21	Phone	Registered	Specialist	2
7	case9705	20.10.09 16:48	Phone	At specialist	Specialist	2
8	case9705	19.11.09 10:31	Phone	In progress	Specialist	2
9	case9705	19.11.09 10:32	Phone	Completed	Specialist	2
10	case3939	15.10.09 11:48	Mail	Registered	Specialist	2
11	case3939	15.10.09 11:48	Mail	Offered	Specialist	2
12	case3939	20.10.09 17:18	Mail	In progress	Specialist	2
13	case3939	20.10.09 17:19	Mail	At specialist	Specialist	2
14	case3939	21.10.09 14:49	Mail	In progress	Specialist	2
15	case3939	21.10.09 14:49	Mail	In progress	Specialist	2
16	case3939	28.10.09 10:17	Mail	In progress	Specialist	2
17	case3939	28.10.09 10:18	Mail	Completed	Specialist	2
18	case9704	20.10.09 14:19	Mail	Registered	1st line	0
19	case9704	20.10.09 14:24	Mail	Completed	1st line	0

Image source: https://fluxicon.com/blog/2012/02/data-requirements-for-process-mining/

The Context

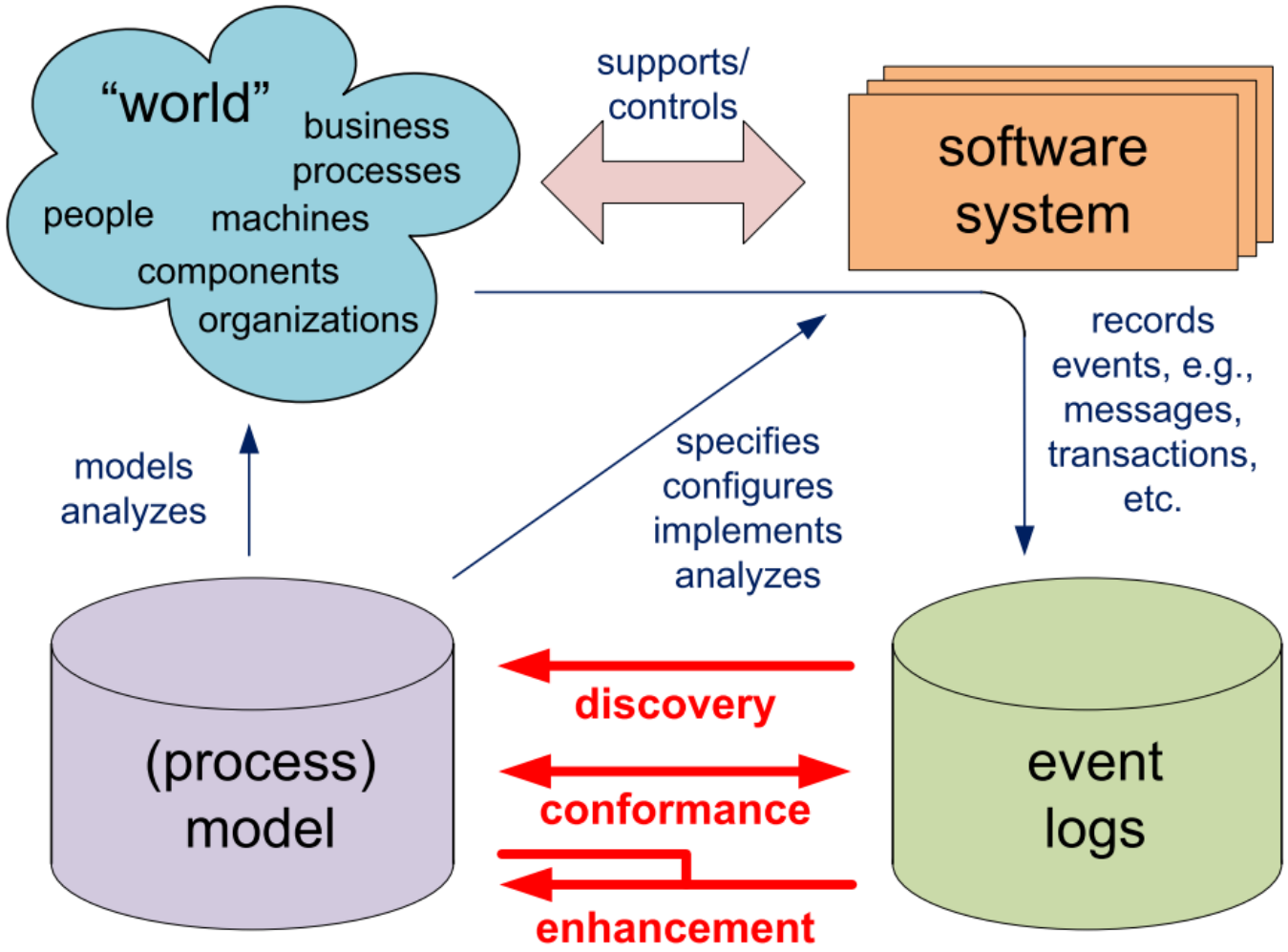


Image source: W. van der Aalst, "Process Mining", 2nd ed, Springer 2016.

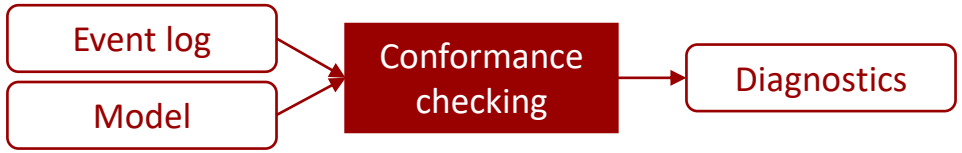
Process Mining Techniques

- Discovery
  - Evidence-based synthesis of process models
  - Use events observed

Process Mining Techniques

- Discovery
  - Evidence-based synthesis of process models
  - Use events observed

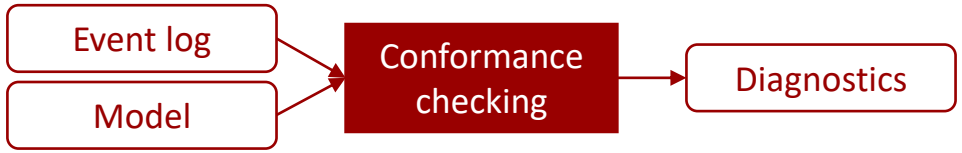
- Conformance checking
  - Detect discrepancies between a process model and events actually occurred



Process Mining Techniques

- Discovery
  - Evidence-based synthesis of process models
  - Use events observed

- Conformance checking
  - Detect discrepancies between a process model and events actually occurred



- Enhancement
  - Extend a process model with additional information
  - Example: a model is annotated with performance data

PM4Py

- PM4py is a Python library that implements state-of-the-art process mining algorithms
  - Open source (licensed under GPL)
  - Designed to be used in both academia and industry projects
- https://pm4py.fit.fraunhofer.de/

- Collection of PM4Py video-tutorials
  - https://www.youtube.com/playlist?list=PLkWuoFn9UEb5l41T4CMKPYHyRcL5ojI9Z

PM4Py – Event logs

- Follow the notebooks with the instructions on how to import and filter an event log
  - https://github.com/pm4py/pm4py-core/blob/release/notebooks/1_event_data.ipynb
  - https://github.com/pm4py/pm4py-core/blob/release/notebooks/2_event_data_filtering.ipynb

- Watch tutorials number #2, #3, #5 from https://www.youtube.com/playlist?list=PLkWuoFn9UEb5l41T4CMKPYHyRcL5ojI9Z

Importing CSV Files

Let's get started! We have prepared a small sample event log, containing behavior similar equal to the process model in Figure
find the sample event log here.

We are going to load the event data, and, we are going to count how many cases are present in the event log, as well as the nu
events. Note that, for all this, we are effectively using a third-party library called pandas. We do so because pandas is the de-fa
of loading/manipulating csv-based data. Hence, any process mining algorithm implemented in pm4py, using an event log as a
work directly with a pandas file!

In [1]:
```
import pandas as pd

df = pd.read_csv('data/running_example.csv', sep=';')
df
```

Out[1]:

	case_id	activity	timestamp	costs	org:resource
0	3	register request	2010-12-30 14:32:00+01:00	50	Pete
1	3	examine casually	2010-12-30 15:06:00+01:00	400	Mike
2	3	check ticket	2010-12-30 16:34:00+01:00	100	Ellen
3	3	decide	2011-01-06 09:18:00+01:00	200	Sara
4	3	reinitiate request	2011-01-06 12:18:00+01:00	200	Sara
5	3	examine thoroughly	2011-01-06 13:06:00+01:00	400	Sean
6	3	check ticket	2011-01-08 11:43:00+01:00	100	Pete
7	3	decide	2011-01-09 09:55:00+01:00	200	Sara
8	3	pay compensation	2011-01-15 10:45:00+01:00	200	Ellen
9	2	register request	2010-12-30 11:32:00+01:00	50	Mike
10	2	check ticket	2010-12-30 12:12:00+01:00	100	Mike
11	2	examine casually	2010-12-30 14:16:00+01:00	400	Sean
12	2	decide	2011-01-05 11:22:00+01:00	200	Sara
13	2	pay compensation	2011-01-08 12:05:00+01:00	200	Ellen
14	1	register request	2010-12-30 11:02:00+01:00	50	Pete
15	1	examine thoroughly	2010-12-31 10:06:00+01:00	400	Sue

BASICS OF PROCESS MINING

(CONTROL-FLOW) DISCOVERY

What is (control-flow) discovery?

- Control-flow discovery is the problem of discovering the control-flow of a process starting from the executions contained in an event log
- Input
  - An event log
- Output
  - A process model capable of replicating the executions in the event log

- Many algorithms are available, with different abstraction capabilities and tailored to specific scenarios (e.g., structured vs flexible processes; noise vs no-noise on the data)

Basics of control-flow discovery algorithms

- The majority of discovery algorithms work by identifying certain patterns, either on the event log or on some intermediate structure
- Examples of the most used algorithms are
  - Heuristics Miner
    - It computes the strength of certain patterns (sequence and parallelism) instantiated on all possible activities, and, when the strength is larger than a threshold, the pattern is added
    - Focus on identifying the most frequent behavior (therefore can cope with noisy data)
    - Extremely popular
  - Inductive Miner
    - From the log extracts an intermediate structure (called DFG) and, on this, identifies certain patterns (called operators) associated with specific control-flow patterns (e.g., sequence, parallelism, choice, loop)
    - Focus on the ability to explain the entire log
    - More recent than Heuristics Miner, also extremely popular
  - *…however, in here we focus on the general concepts.*

Processing the event log

file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Processing the event log

- When considering control-flow discovery, a first simplification consists of considering only the significant cases happening

file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Processing the event log

- When considering control-flow discovery, a first simplification consists of considering only the significant cases happening

- Identification of traces:

file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Processing the event log

- When considering control-flow discovery, a first simplification consists of considering only the significant cases happening

- Identification of traces:
  - Case 1: $\langle ABCD \rangle$

file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Processing the event log

- When considering control-flow discovery, a first simplification consists of considering only the significant cases happening

- Identification of traces:
  - Case 1: $\langle ABCD \rangle$
  - Case 2: $\langle ACBD \rangle$

file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Processing the event log

- When considering control-flow discovery, a first simplification consists of considering only the significant cases happening

- Identification of traces:
  - Case 1: $\langle ABCD \rangle$
  - Case 2: $\langle ACBD \rangle$
  - Case 3: $\langle ABCD \rangle$

file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Processing the event log

- When considering control-flow discovery, a first simplification consists of considering only the significant cases happening

- Identification of traces:
  - Case 1: $\langle ABCD \rangle$
  - Case 2: $\langle ACBD \rangle$
  - Case 3: $\langle ABCD \rangle$
  - Case 4: $\langle ACBD \rangle$

file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Processing the event log

- When considering control-flow discovery, a first simplification consists of considering only the significant cases happening

- Identification of traces:
  - Case 1: $\langle ABCD \rangle$
  - Case 2: $\langle ACBD \rangle$
  - Case 3: $\langle ABCD \rangle$
  - Case 4: $\langle ACBD \rangle$
  - Case 5: $\langle ACBEFEFBCD \rangle$

file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Processing the event log

- When considering control-flow discovery, a first simplification consists of considering only the significant cases happening

- Identification of traces:
  - Case 1: $\langle ABCD \rangle$
  - Case 2: $\langle ACBD \rangle$
  - Case 3: $\langle ABCD \rangle$
  - Case 4: $\langle ACBD \rangle$
  - Case 5: $\langle ACBEFEFBCD \rangle$

- Frequency of traces

file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Processing the event log

- When considering control-flow discovery, a first simplification consists of considering only the significant cases happening

- Identification of traces:
  - Case 1: $\langle ABCD \rangle$
  - Case 2: $\langle ACBD \rangle$
  - Case 3: $\langle ABCD \rangle$
  - Case 4: $\langle ACBD \rangle$
  - Case 5: $\langle ACBEFEFBCD \rangle$

- Frequency of traces
  - $W = \{\langle ABCD \rangle^2, \langle ACBD \rangle^2, \langle ACBEFEFBCD \rangle\}$

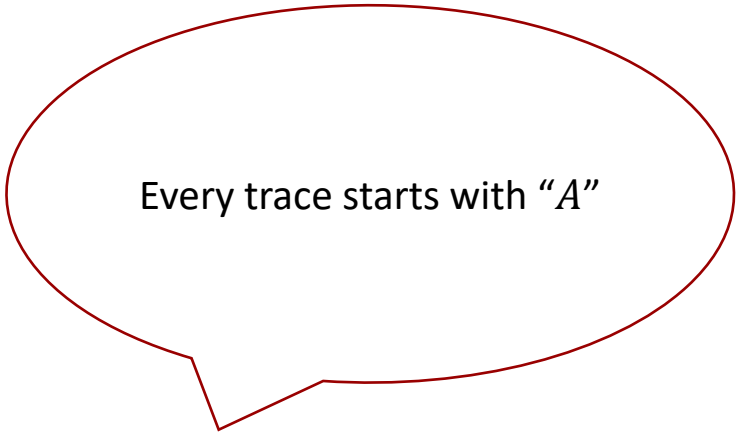
file: eventlog.csv

```
case 1; task A; 2024-01-01
case 2; task A; 2024-01-01
case 3; task A; 2024-01-01
case 3; task B; 2024-01-02
case 1; task B; 2024-01-03
case 1; task C; 2024-01-04
case 2; task C; 2024-01-04
case 4; task A; 2024-01-04
case 2; task B; 2024-01-05
case 2; task D; 2024-01-06
case 5; task A; 2024-01-06
case 4; task C; 2024-01-06
case 1; task D; 2024-01-06
case 3; task C; 2024-01-06
case 3; task D; 2024-01-07
case 4; task B; 2024-01-07
case 5; task C; 2024-01-07
case 4; task D; 2024-01-08
…
```

Discovery example

$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example

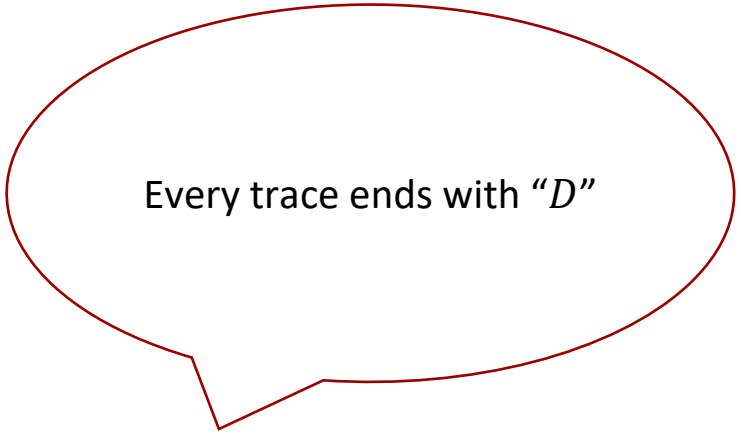


$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example

$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example

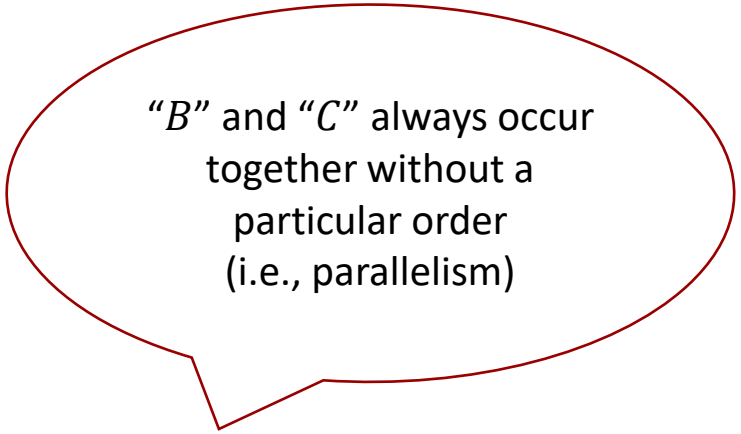


$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example

$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example

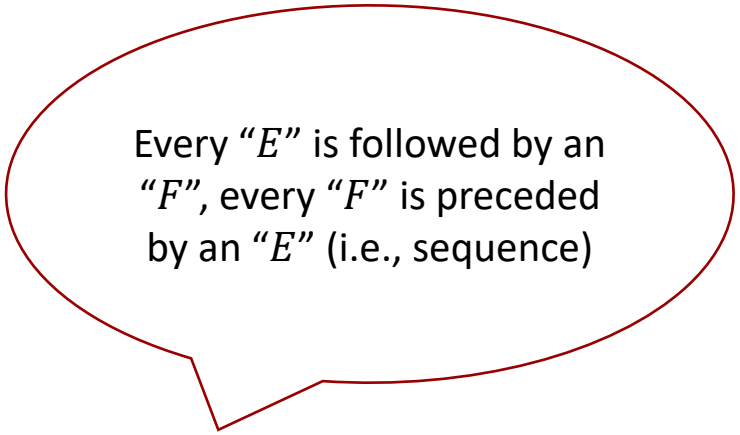


$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example

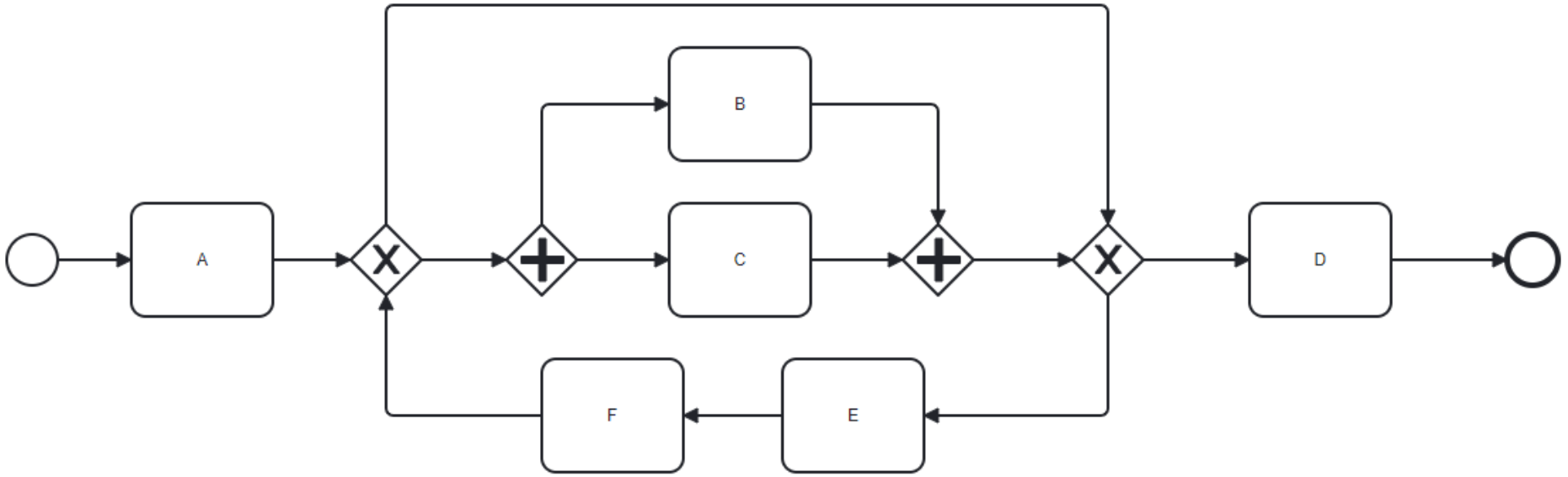
$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example



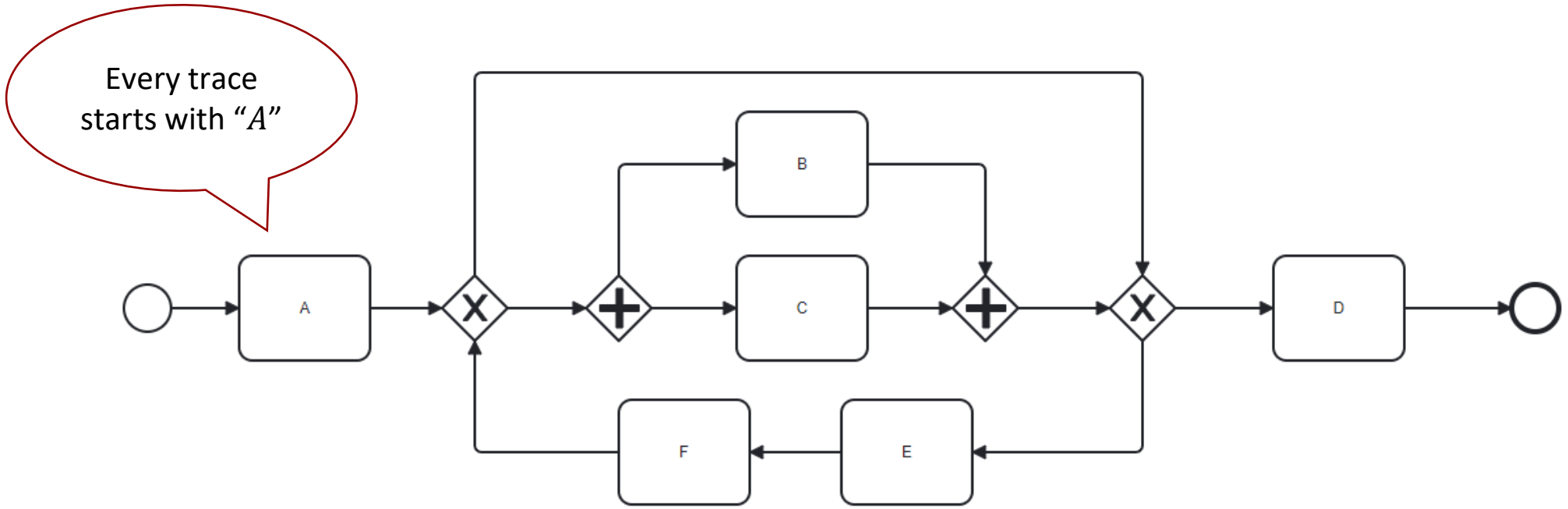
$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example



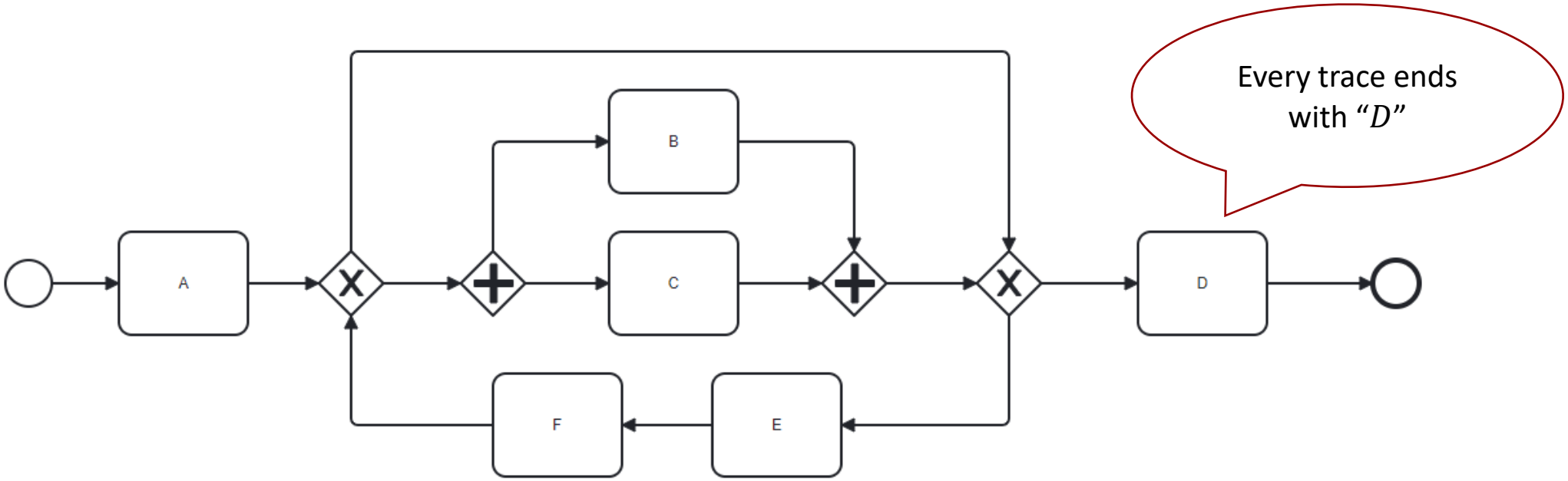
$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example



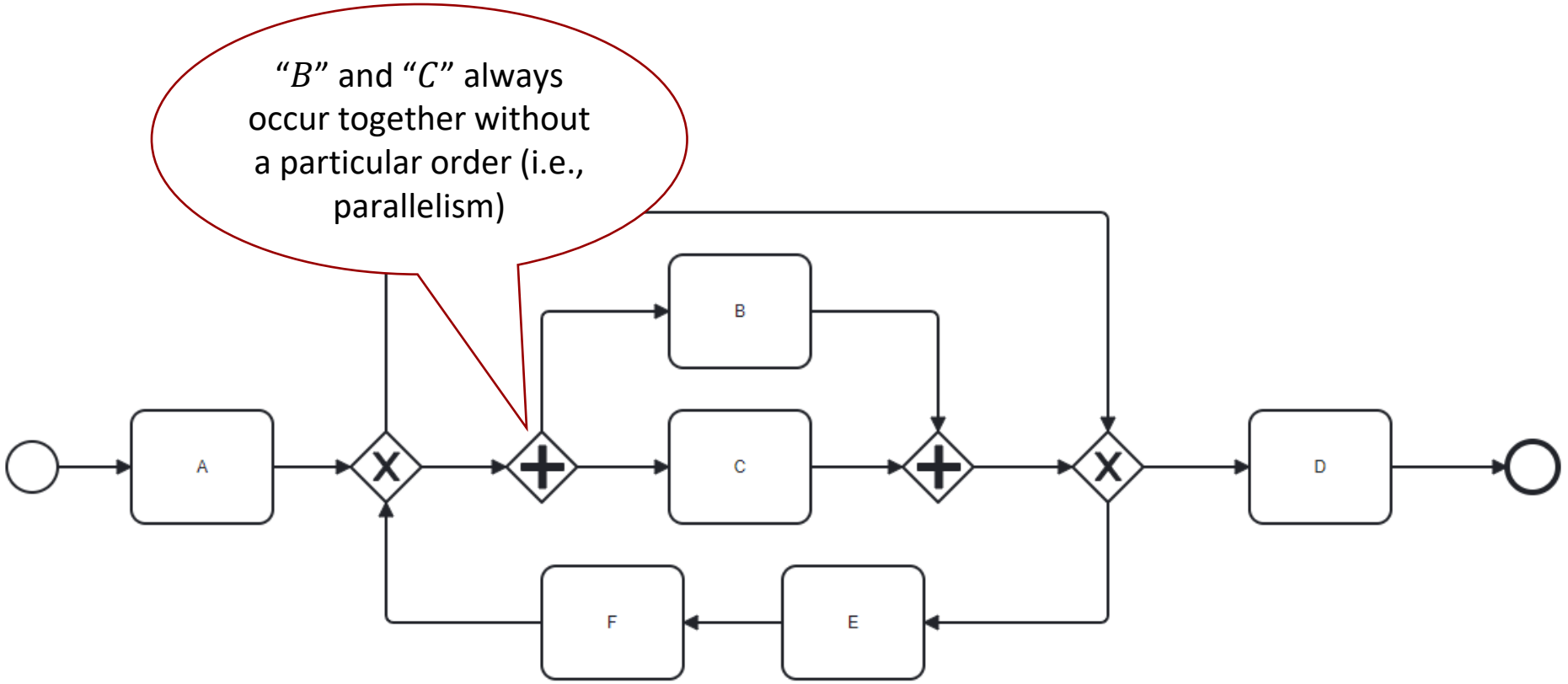
$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example



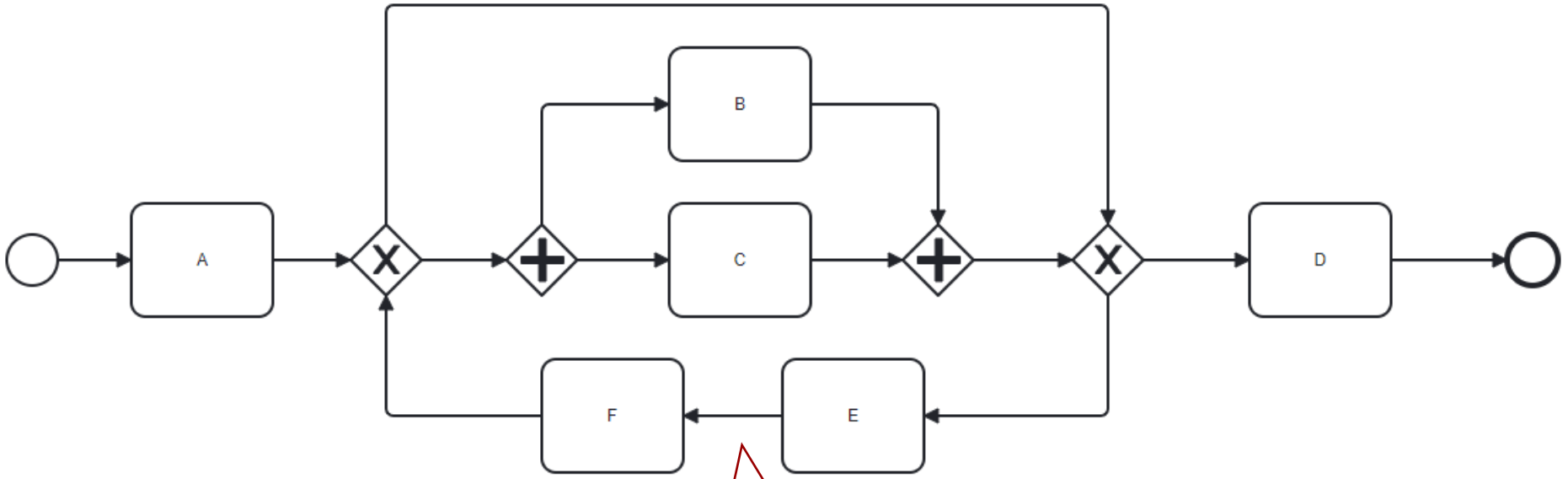
$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

Discovery example



$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

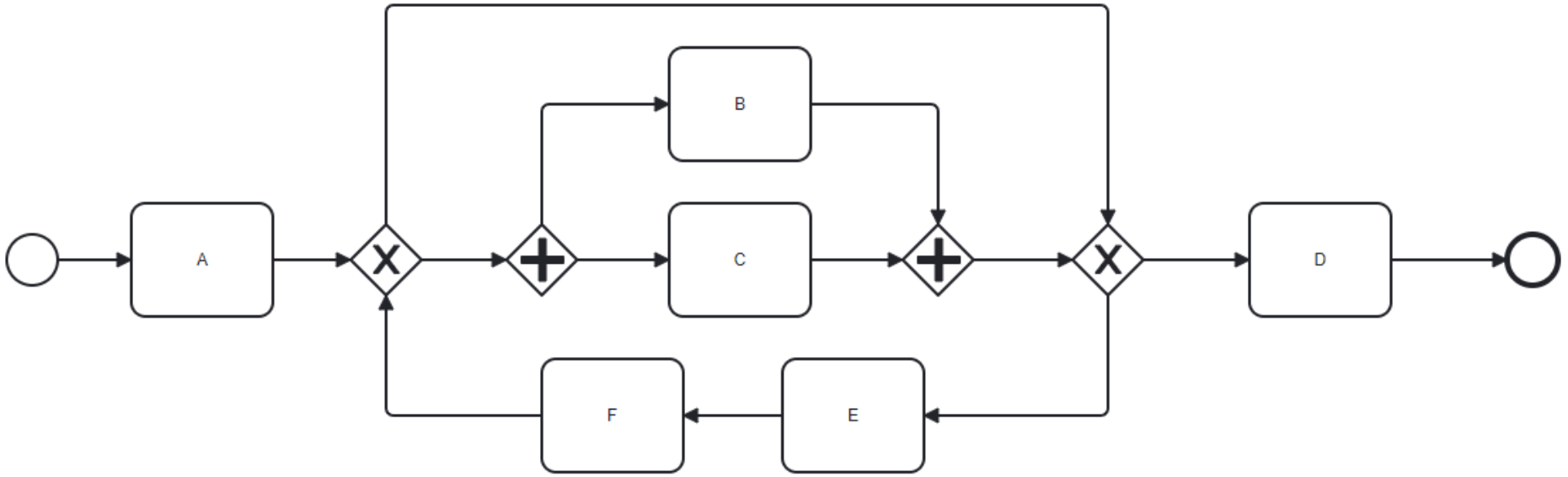
Discovery example



$W = \{\langle$ $BCD\rangle\}$

Every "E" is followed by an "F", every "F" is preceded by an "E" (i.e., sequence)

Discovery example



$$W = \{\langle ABCD\rangle^2, \langle ACBD\rangle^2, \langle ACBEFEFBCD\rangle\}$$

PM4Py – Control-flow discovery

- Follow the notebook with the instructions on how to perform control-flow discovery
  - https://github.com/pm4py/pm4py-core/blob/release/notebooks/3_process_discovery.ipynb

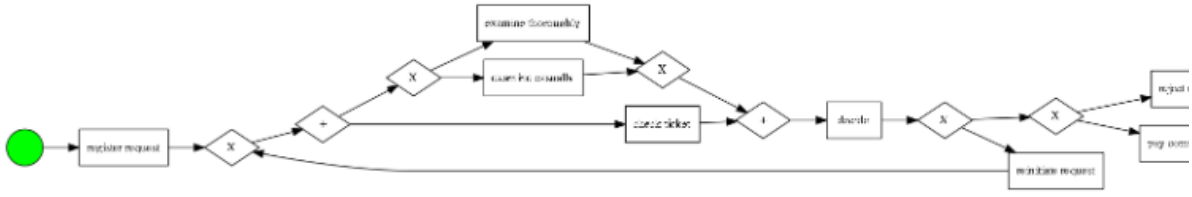
- Watch the tutorial https://www.youtube.com/watch?v=BJMp763Ye_o&list=PLkWuoFn9UEb5l41T4CMKPYHyRcL5ojI9Z&index=7

Obtaining a Process Model

There are three different process modeling notations that are currently supported in PM4Py. These notations are: BPMN as the ones shown earlier in this tutorial, *Process Trees* and *Petri nets*. A Petri net is a more mathematical modeling repre compared to BPMN. Often, the behavior of a Petri net is more difficult to comprehend compared to BPMN models. Hou mathematical nature, Petri nets are typically less ambiguous (i.e., confusion about their described behavior is not possib represent a strict subset of Petri nets and describe process behavior in a hierarchical manner. In this tutorial, we will foc BPMN models and process trees. For more information about Petri nets and their application to (business) process mod 'workflow' perspective), we refer to this article.

Interestingly, none of the algorithms implemented in PM4Py directly discovers a BPMN model. However, any process tr translated to a BPMN model. Since we have already discussed the basic operators of BPMN models, we will start with th process tree, which we convert to a BPMN model. Later, we will study the 'underlying' process tree. The algorithm that use is the 'Inductive Miner'; More details about the (inner workings of the) algorithm can be found in this presentation Consider the following code snippet showing how to obtain a BPMN model from an event log.

In [45]:
```
import pandas as pd
import pm4py
df = pm4py.format_dataframe(pd.read_csv('data/running_example.csv', sep=';'), case_id='case_id',activi
                            timestamp_key='timestamp')
bpmn_model = pm4py.discover_bpmn_inductive(df)
pm4py.view_bpmn(bpmn_model)
```



Observe that the process model that we discovered, describes the same behavior as the model that we have shown abo

As indicated, the algorithm used in this example actually discovers a *Process Tree*. Such a process tree is, mathematically *tree*, annotated with 'control-flow' information. We'll first use the following code snippet to discover a process tree base example, and, afterwards shortly analyze the model.

BASICS OF PROCESS MINING

CONFORMANCE CHECKING

What is conformance checking?

- Given an execution trace and a prescriptive model (i.e., a model that is supposed to be followed by all process instances), identify if the model allows the trace
- Input
  - A trace (part of an event log)
  - A prescriptive model
- Output
  - A measure indicating if – and to what extent – the model can replicate the trace

- Two main families of techniques are available
  - Token-based conformance checking: fast but can have problems if used with specific process models
  - Alignments: precise but computationally expensive

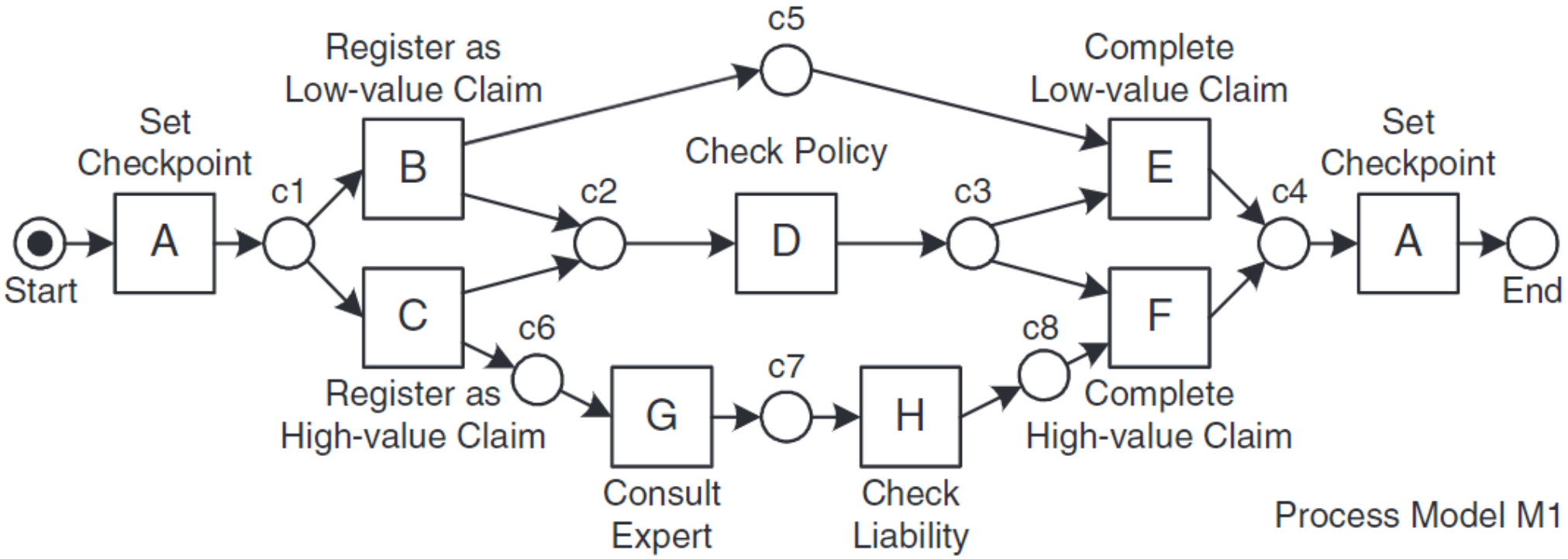
Conformance with token replay

- Idea
  - Try to “replay” each event in the trace
  - Try to execute activities of the process according to the trace
  - Analyse the relation between log and model semantics
  - Not ok:
    - Activity must be executed even though not all “tokens” were there
    - Some tokens remain in the process upon completion

- Notice: in the next couple of slides, Petri nets are used as a notation. If the reader is not familiar with this notation, this video should be watched: https://youtu.be/ISYIyVf1U6I
  - A more in-depth tutorial can be watched at https://youtu.be/yHfEmYsqgMk

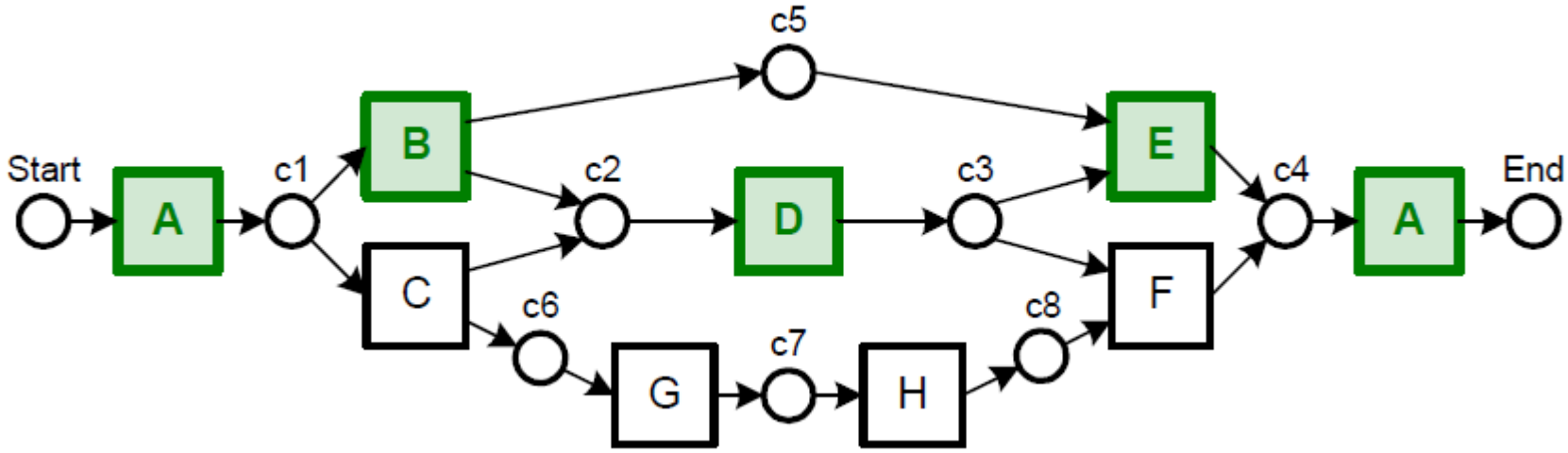
Conformance with token replay (cont.)

- Let's consider the following process model expressed as Petri net



Picture from: Rozinat and van der Aalst, "Conformance Checking of Processes Based on Monitoring Real Behavior", Information Systems, Volume 33, Issue 1, 2008

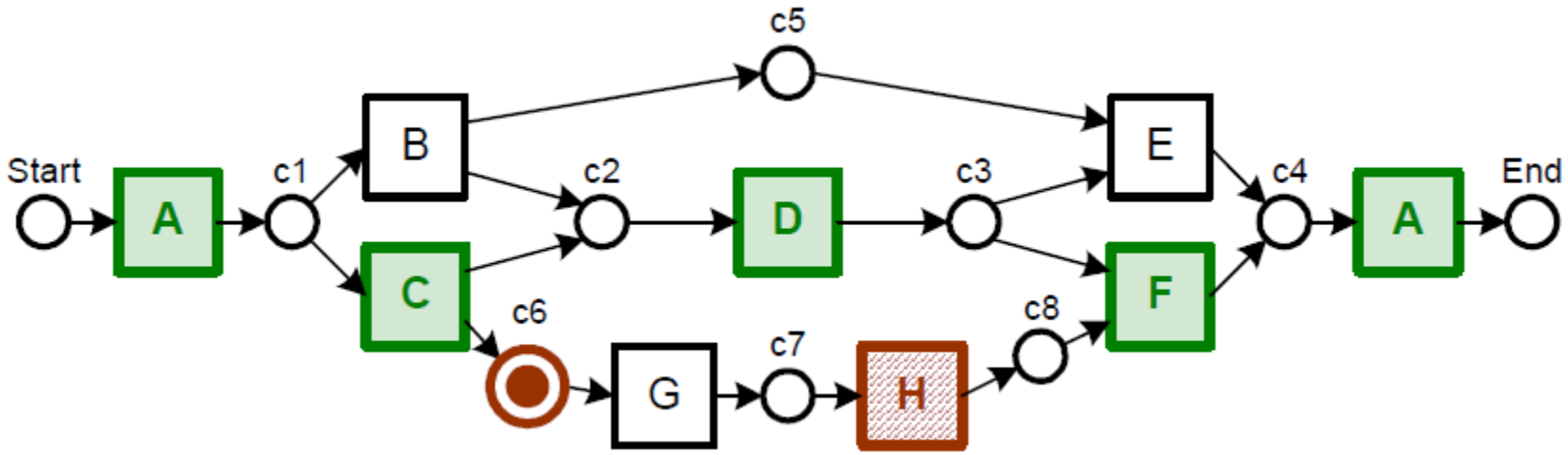
Conformance with token replay(cont.)



Frequency	Traces
1207	$\langle A, B, D, E, A \rangle$
145	$\langle A, C, D, G, H, F, A \rangle$
56	$\langle A, C, G, D, H, F, A \rangle$
23	$\langle A, C, H, D, F, A \rangle$
28	$\langle A, C, D, H, F, A \rangle$

Picture from: Rozinat and van der Aalst, "Conformance Checking of Processes Based on Monitoring Real Behavior", Information Systems, Volume 33, Issue 1, 2008

Conformance with token replay(cont.)



Frequency	Traces
1207	$\langle A, B, D, E, A \rangle$
145	$\langle A, C, D, G, H, F, A \rangle$
56	$\langle A, C, G, D, H, F, A \rangle$
23	$\langle A, C, H, D, F, A \rangle$
28	$\langle A, C, D, H, F, A \rangle$

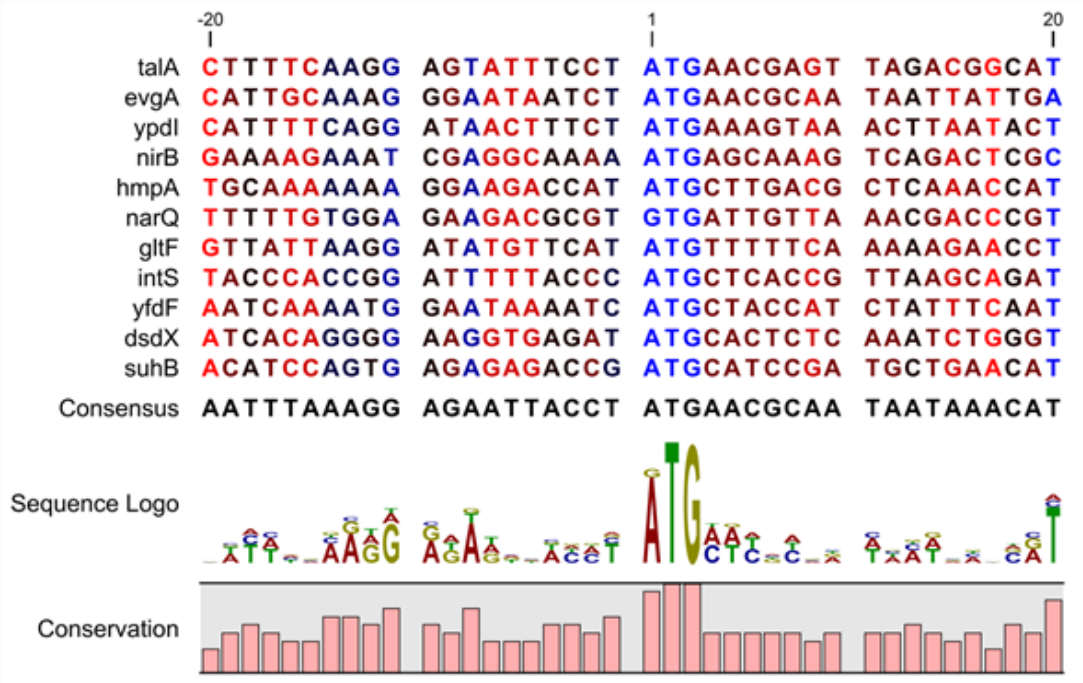
Picture from: Rozinat and van der Aalst, "Conformance Checking of Processes Based on Monitoring Real Behavior", Information Systems, Volume 33, Issue 1, 2008

Alignment-based techniques

- Assessing the conformance of an event log with a model based on the alignment of the process model's activities and events in a trace
  - Consider the set of activities of the model as a set of symbols
  - Then, each execution sequence of the model is a sequence of symbols
  - Each trace of the event log is also a sequence of symbols

Alignment-based techniques

- Assessing the conformance of an event log with a model based on the alignment of the process model's activities and events in a trace
  - Consider the set of activities of the model as a set of symbols
  - Then, each execution sequence of the model is a sequence of symbols
  - Each trace of the event log is also a sequence of symbols



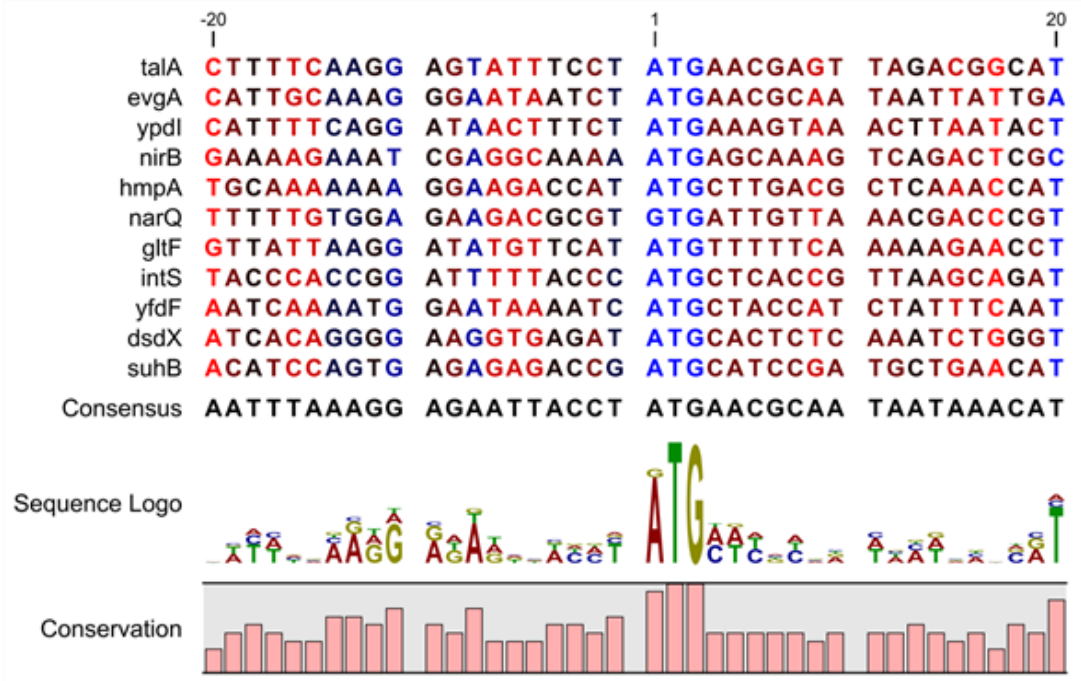
Picture from https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/900/index.php?manual=BE_Sequence_logo.html

Alignment-based techniques

- Assessing the conformance of an event log with a model based on the alignment of the process model's activities and events in a trace
  - Consider the set of activities of the model as a set of symbols
  - Then, each execution sequence of the model is a sequence of symbols
  - Each trace of the event log is also a sequence of symbols

- An alignment between two sequences comes from
  - Linking pairs of symbols in each sequence
  - Such that the order between aligned symbols is preserved

- Alignments allow quantification of conformance and insights on non-conformance



Picture from https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/900/index.php?manual=BE_Sequence_logo.html

PM4Py – Conformance checking

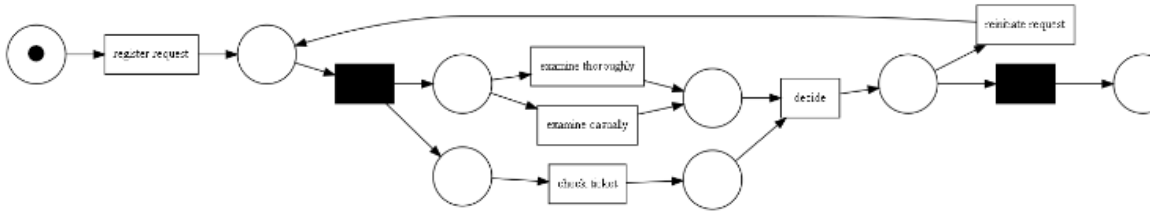
- Follow the notebook with the instructions on how to perform conformance checking
  - https://github.com/pm4py/pm4py-core/blob/release/notebooks/4_conformance_checking.ipynb

- Watch the tutorial https://www.youtube.com/watch?v=0YNvijqX3FY&list=PLkWuoFn9UEb5l41T4CMKPYHyRcL5ojI9Z&index=8

Token-Based-Replay

In order to understand token-based-replay, we first need to cover a bit of Petri net theory. Let's use the Petri net bas
example event log, as an example.

In [1]:
```
import pandas as pd
import pm4py
df = pm4py.format_dataframe(pd.read_csv('data/running_example.csv', sep=';'), case_id='case_id',act
                            timestamp_key='timestamp')
pn, im, fm = pm4py.discover_petri_net_inductive(df)
pm4py.view_petri_net(pn, im, fm)
```



Places and Transitions

Observe that the Petri net consists of two different type of nodes, i.e., cirlces and rectangles. We refer to the circles a
the rectangles as *transitions*. Furthermore, notice that, a place can only be connected (by means of an arc) to a trans
transition can only be connected (by means of an arc) to a place. Hence, *places never connect directly to places* and *t
directly to transitions*.

Tokens, Enabledness and Transition Firing

There is one place in the model containing a black 'dot'. This dot is referred to as a *token*. For convienence, let's call
token 'source'. A transition can consume and produce tokens, referred to as *firing a transition*. A transition is allowe
'incoming places' contain at least one token. Any transition for which this property holds is referred to as an *enablec
example net, only the 'source' place contains a token. Consequently, the only transition that has a token in all of its '
transition *register request*, i.e., it is enabled. If we diced to fire the an enabled transition, it consumes one token from
places' and it produces a token in each of its 'outgoing places'. For example, if we *fire* the *register request* transition,

FIND OUT MORE

https://gamess.dk/

GameSS

WHO IS BEHIND GameSS

Partners behind the project

SDU

DTU

IT UNIVERSITY OF CPH

AALBORG UNIVERSITET

Happy 42

Collaborators

ERHVERVSAKADEMI AARHUS

kea
KØBENHAVNS ERHVERVSAKADEMI

Dania
ERHVERVSAKADEMI

UCN

Erhvervsakademi og Professionshøjskole

Uddannelses- og Forskningsministeriet
Ministry of Higher Education and Science Denmark