An Introduction to Business Process Modeling

Alejandro Vaisman
Department of Computer & Decision Engineering (CoDE)
Université Libre de Bruxelles
avaisman@ulb.ac.be

Outline (part 1)

- Motivation
- Introduction
- Petri nets and Workflow Nets
- Using Workflows to model Business Processes
- Workflow Patterns and YAWL
- BPMN and BPEL
- Design
- A Database Vision
- Summary

Outline

- **Motivation**
- Introduction
- Petri nets and Workflow Nets
- Using Workflows to model Business Processes
- Workflow Patterns and YAWL
- BPMN and BPEL
- Design
- A Database Vision
- Summary

Motivation

- Definition: A **Business Process** is a collection of related, structured activities that produce a specific service or product (serve a particular goal) for a particular customer or customers.
- **Business process management** (BPM) refers to methods, techniques, and tools that support the design, management, and analysis of business processes. It can be considered as an extension of classical **workflow management systems. (**even early, **office information systems**).
- **Workflow:** the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

Motivation

- A **Process Model** is a formalized view of a business process represented as a coordinate set of parallel and/or sequential set of process activities that are connected to achieve a common goal.
- Data modeling used to be the starting point of a system. Opposite to this, BPs are becoming the starting point => BPM follows the process-centric development paradigm.

Motivation

- Why should a database/BI practitioner be interested in BPM?
  - Most BI tools apply to relational data. Also the case in data mining (in general).
  - Not only relational data can be mined. Useful knowledge can be discovered from process data => IEEE Task Force on Process Mining (www.win.tue.nl/ieeetfpm/).
  - ETL is a key part in any BI project. ETL can be seen as a process that takes data from the sources to the DW.
  - Research in ETL as a workflow is being carried out (El Akkaoui et al., DOLAP 2011, DaWaK 2012; Vassiliadis et al., ER 2005 and others.)

Motivation

- Why should a database/BI practitioner be interested in BPM?

  - Since 2007, database researchers are pushing to apply many of the achievements in RDB (elegant model, declarative query languages, query optimization, efficient implementation) to process data. (Deutch & Milo, PODS 2011).

Motivation

Goals of the tutorial:

First part

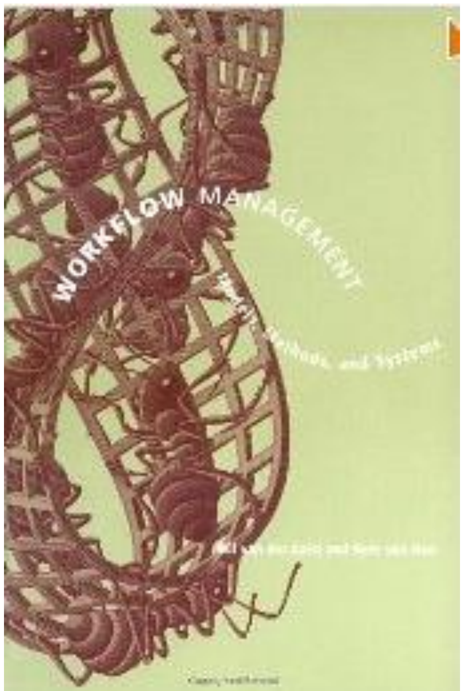
- Basics of BP modeling.
- Overview of methods, standards, and tools for BPM.

Second part

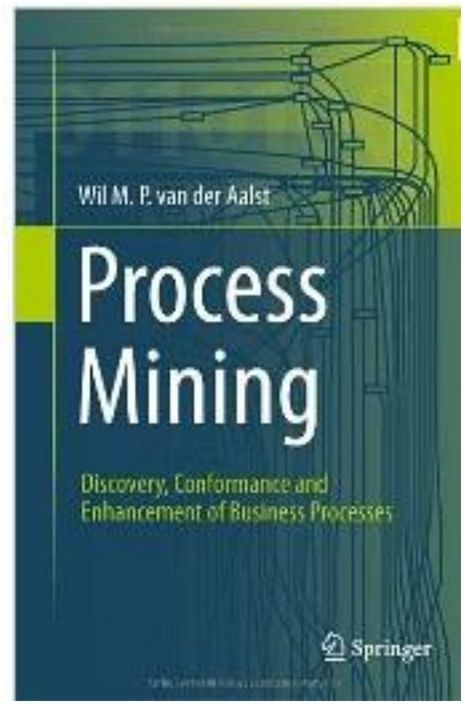
- Process Mining.
  - Discovering, conformance checking, Online PM.
  - Basic algorithms

Credits

Many slides in this tutorial adapted from:



Workflow Management: Models, Methods, and Systems. van der Aalst, van Hee, MIT Press, 2002.



Process Mining: Discovery, Conformance and Enhancement of Business Processes. Van der Aalst, Springer, 2011.

Outline

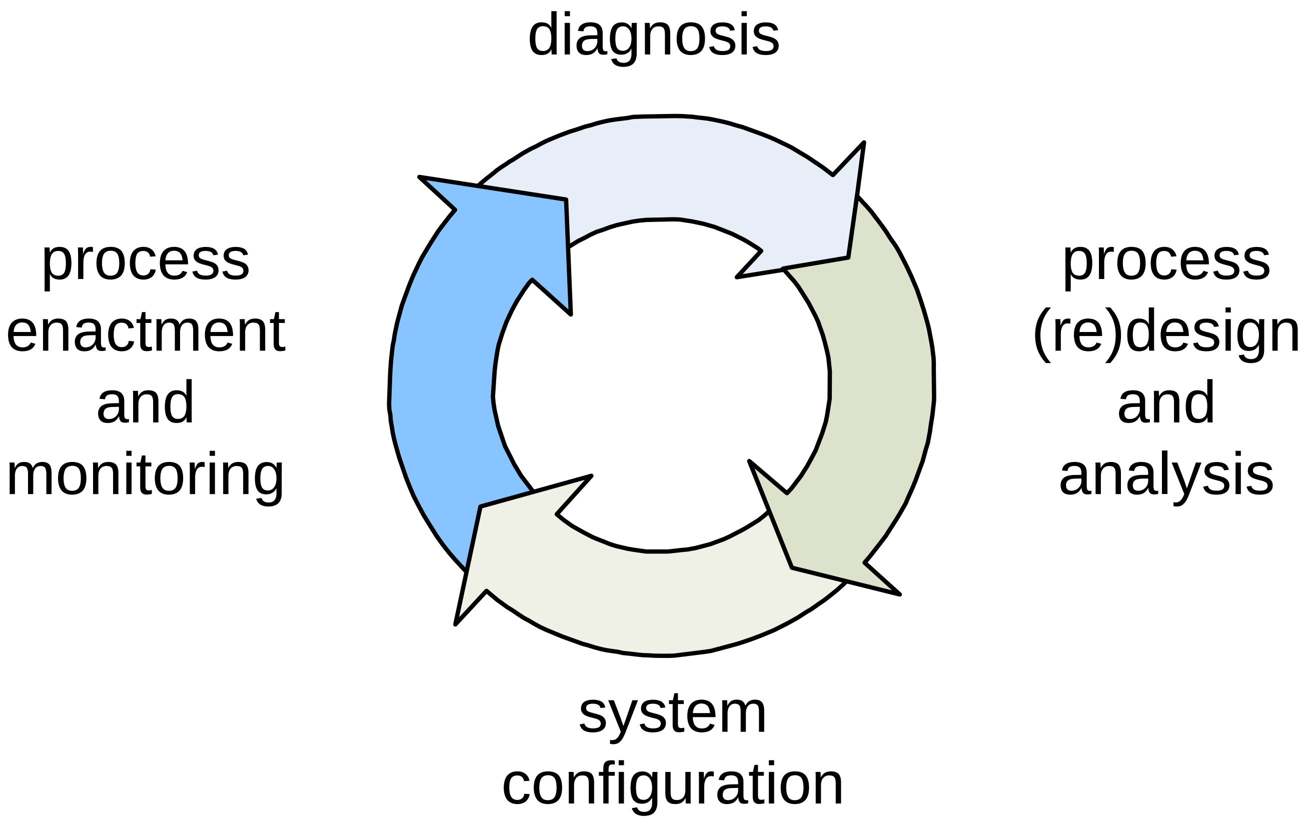
- Motivation
- **Introduction**
- Petri nets, and Workflow nets
- Using Workflows to model Business Processes
- Workflow Patterns and YAWL
- BPMN and BPEL
- Design
- A Database Vision
- Summary

Business Process lifecycle

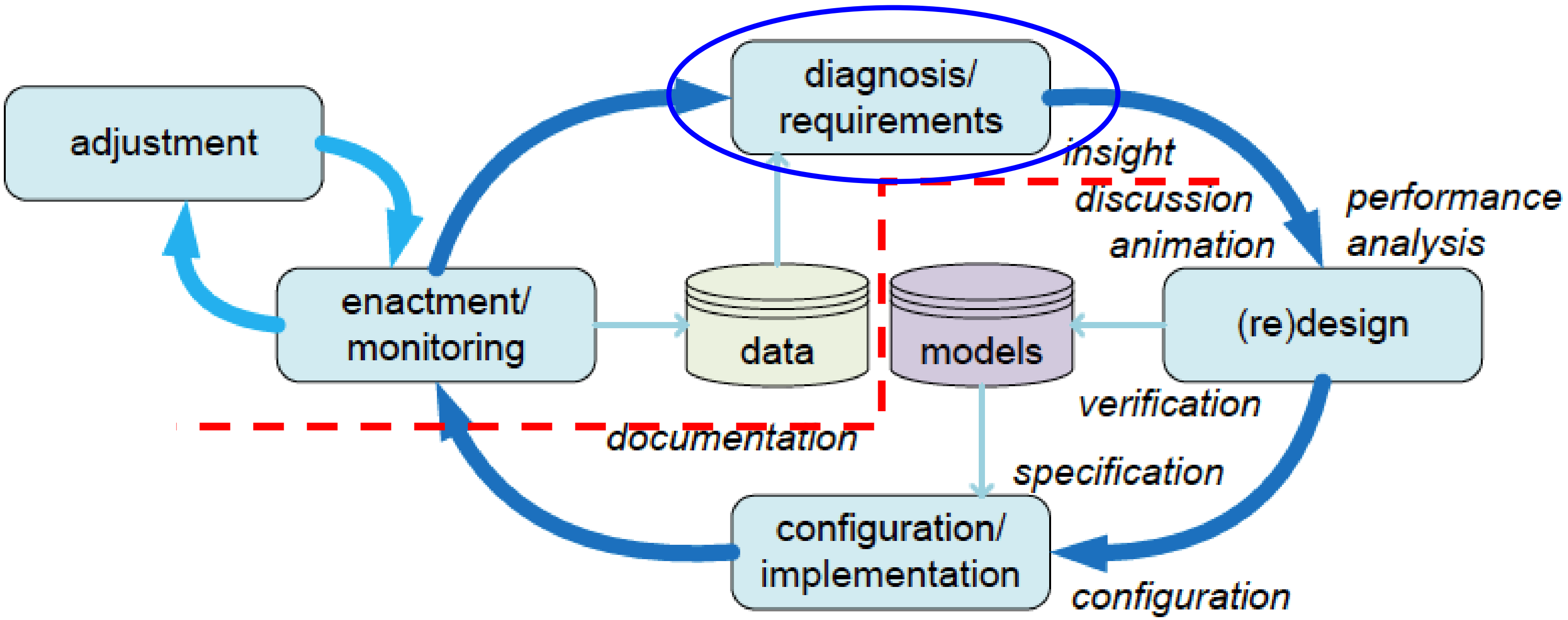
- **Design** phase: the process is designed (some guidelines later)
- **Configuration** phase: model coded into conventional software.
- **Enactment (execution) / monitoring** phase: process running and monitored by management, to see if changes are needed.
- **Adjustment** phase: changes made according to the previous phase.
- **Diagnosis/requirements** phase: evaluates the process and monitors new requirements (new policies, laws, etc.).

Poor performance or new requirements may require a new iteration of all the lifecycle.

Business Process lifecycle



Business Process lifecycle (detailed)



Outline

- Motivation
- Introduction
- **Petri nets and Workflow nets**
- Using Workflows to model Business Processes
- Workflow Patterns and YAWL
- BPMN and BPEL
- Design
- A Database Vision
- Summary

Workflow basics

- Early **BP** specifications lack of precise semantics, and were difficult to analyze.
- A formalism was needed.
- Petri nets adopted to design and analyze workflows.
  - Petri nets are formal.
  - They have associated analysis techniques.
- Drawback: they cannot specify certain control flow dependencies. Must be adapted to represent BPs.

Workflow basics

- A workflow system deals with **cases**. For example, in a process that handles insurance claims, a case is a particular claim; or issuing an air ticket is a case (i.e., an instance) of the process of issuing air tickets.
  - Cases are classified in *types* (cases handled in a similar way). A case has an *identity*, i.e., a case that can be univocally identified.
- The central component of a workflow is the **task or activity**. A task is a logical, indivisible unit of work. If anything goes wrong when performing a task, it must be rolled-back. (similar to atomicity in DBMS)

Workflow basics

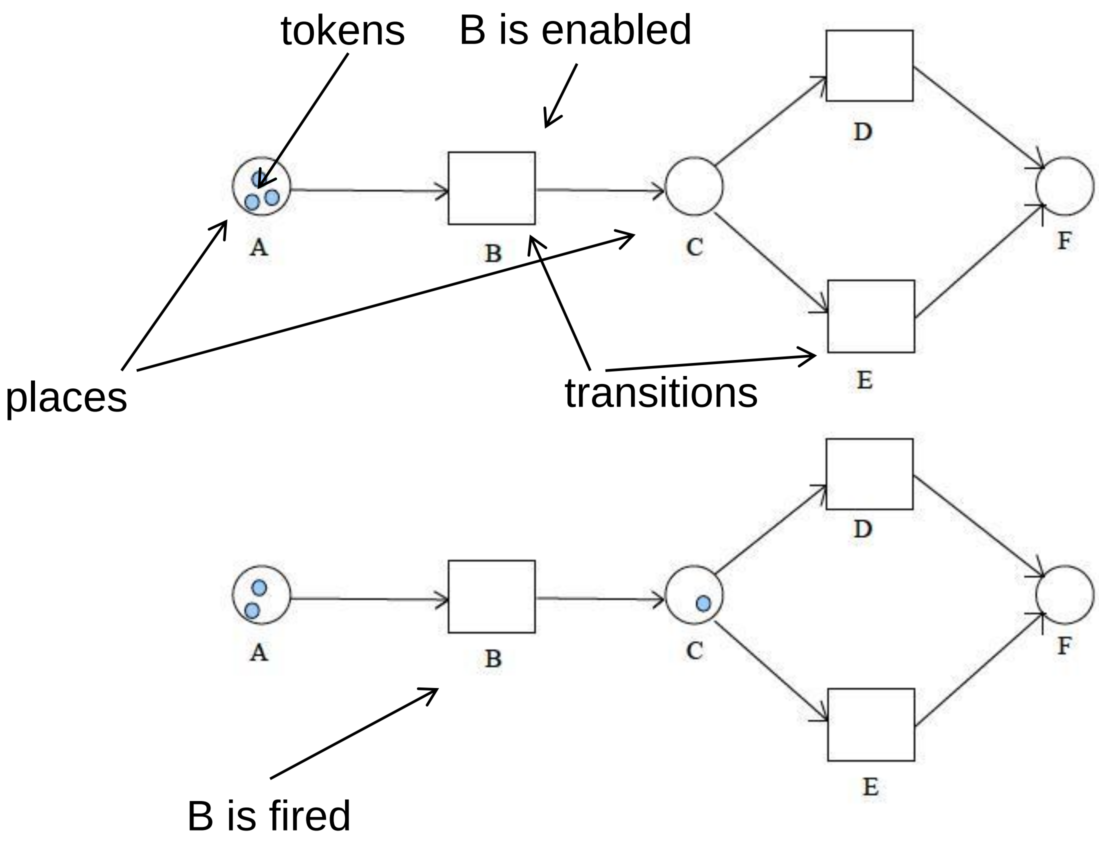
- **Process**: a procedure followed to handle a particular case type. Processes can be part of other ones, in which case we denote them **sub-processes.**
- **Routing:** refers to the way in which a process is carried out, in the sense that it defines the order of the tasks that compose a given process. Routing can be *sequential, parallel, selective, or iterative.*
- **Enactment**: triggering a task. Different ways: by a *resource initiative*, by an *external event* or action (like a message), or by *time signals*.

Petri nets

- Created in 1962 by Carl Adam Petri, to model and analyze processes.
- They have been used to model complex processes (particularly in the operating systems field)
- Main strength: they are fully formalized.
- Basic Petri nets have been extended in several ways, in particular to model BPs.
- A classic Petri net is a *directed bipartite graph* defined as follows.

A **Petri net** is a triple (P,T,F) where P is a finite set of *places*, T is a finite set of *Transitions*, such that $T \cap P = \Phi$, and
F is a set of arcs in $(P \times T) \cup (T \times P)$.
A **Marked Petri net** is a pair (N,M), where N= (P,T,F) is a Petri net, and M is a multiset over P denoted the ***marking*** of the net.

Petri nets: a simple example

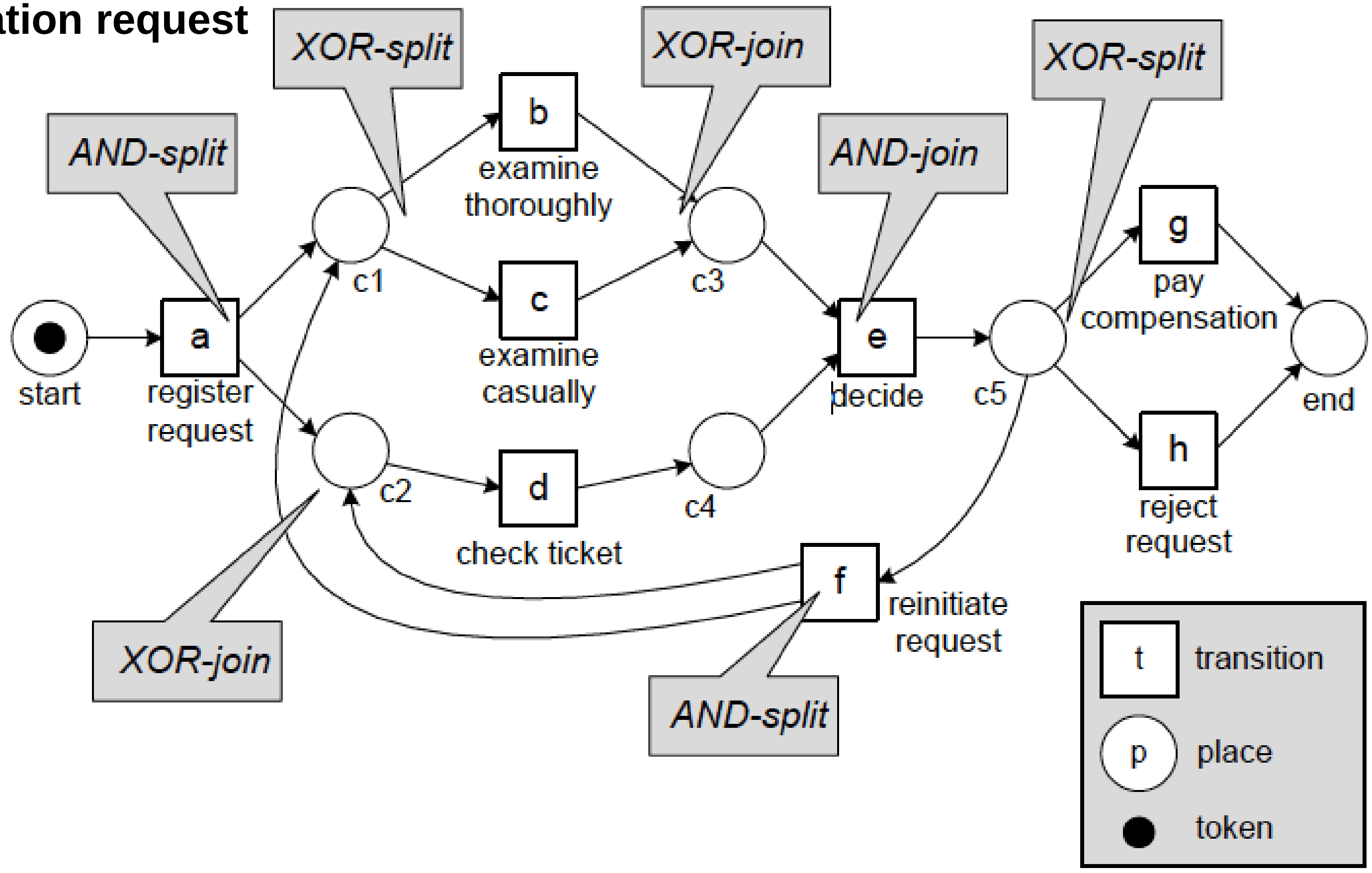


Petri nets: how do they work?

- A place **p** is called an **input place** of a transition **t** iff there is an arc from **p** to **t**.
- Conversely, it is called an **output place** iff there is an arc **t** to **p**.
- Places are represented as circles, and transitions as squares. Directed arcs link both kinds of figures.
- Places may contain **tokens** (probably more than one), indicated as black dots. When a transition takes a token from an input place and puts it in an output place, we say that the transition is **fired**.
- The state of a Petri net is defined by the position of the tokens in it. A transition may only be fired if it is **enabled**, meaning that **there is a token in all of its input places.**

Petri nets detailed

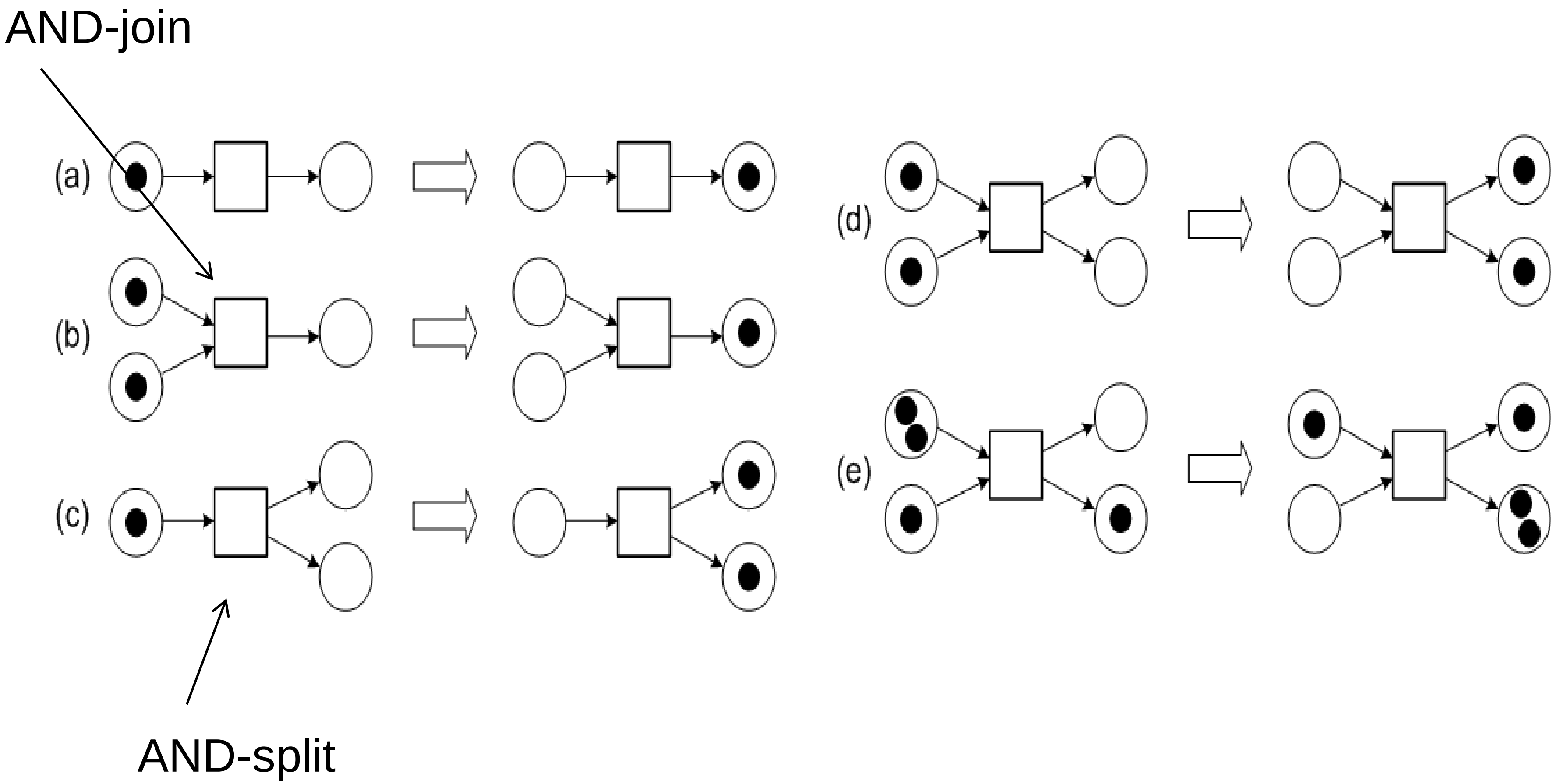
Example: ticket compensation request



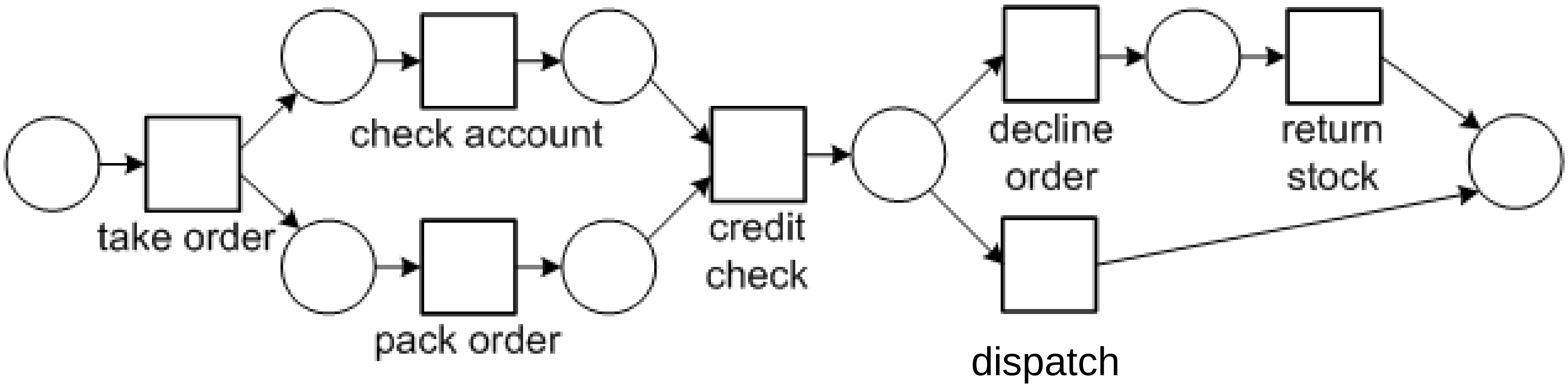
Petri nets: how do they work?

- The marking of the net in the previous slide is [*start*]. This marking enables transition **a**.
- Firing **a** results in the marking [c1,c2]. This enables transitions **b, c**, and **d.**
- Firing **b** results in the marking [c2,c3], **b** and **c** are not enabled now, and **d** is enabled. **e** is not enabled, because it needs the marking [c3,c4].
- If the marking is [*start*^{5}], there are 5 tokens initially. Firing **a** results in the marking [*start*^{4}*,* c2,c3].
- In general, (N,M)[t> denotes that t is enabled at marking M.
- (N,M)[t>(N,M’) denotes that firing t results in the marking M’.
- A labeled Petri net is a tuple N=(P,T,F,A,l) s.t. (P,T,F) is a Petri net, A is a set of activity labels, and l: T->A is a labeling function.
- •c1={a,f} means that c1 is an output place for **a** and **f**. c1•={b,c} means that c1 is an input place for **b** and **c**.

Petri nets: firing transitions



Petri nets: order fulfillment example



Extending Petri nets

- Basic Petri nets cannot capture complex situations, like the ones present in BPs.
- Three main extensions:

  - Color extension
  - Time extension
  - Hierarchical extension

  - These are called **High Level Petri Nets.**

Color extension

- **Uses colors to identify kinds of tokens.**

- A basic Petri net formalism does not allow to distinguish between a car policy claim and a fire policy claim, represented just as tokens.
- Solution : assign two different color for tokens representing each kind of claim.

- Analogous to assign a value to a token.

- When tokens have values, we can fire transitions based on *conditions over these token values.* For example “The token to be consumed must correspond to a car insurance code.”

Time and hierarchy extensions

- Classic Petri nets do not allow the modeling of time.
- In the **time extension**, each token is associated with a timestamp.
- E.g. a token with timestamp `10' can only be consumed from the time instant `10'*. Before that, the transition is not enabled*, even though there is a token in each input place.
- To adequately model hierarchical processes, the **hierarchical extension** allows to define sub-processes, represented as a double square indicating that this transition corresponds indeed to a sub-net (i.e., another Petri net).

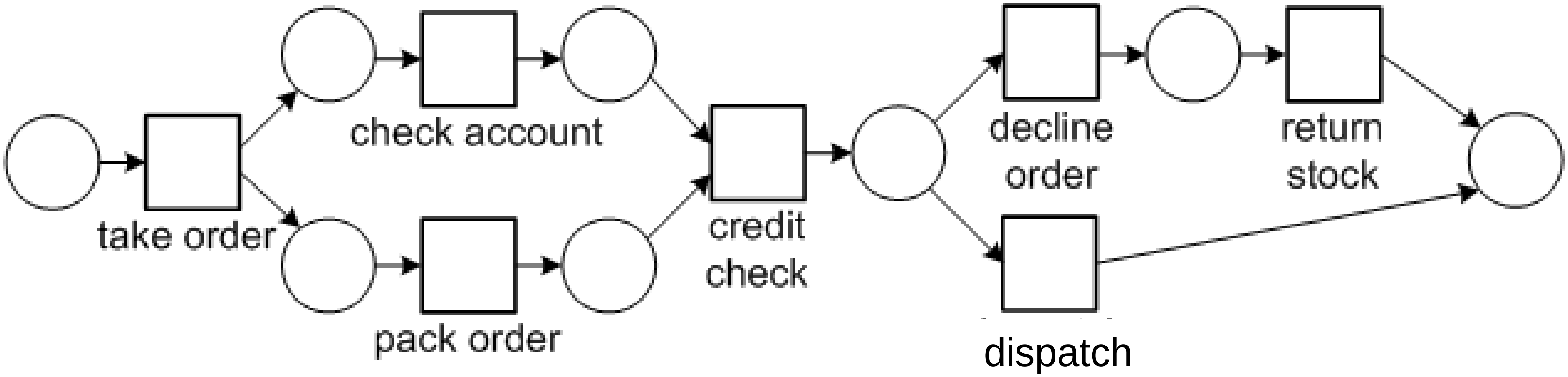
Workflow nets (WF-nets)

- Defined by Wil Van der Aalst in the mid-90's.
- A subset of Petri nets for modeling BP in terms of Petri nets.
- *A WF-net is a Petri net with a dedicated source place where the process starts, and a dedicated sink place where the process ends. In addition, all nodes are on a path from source to sink.*

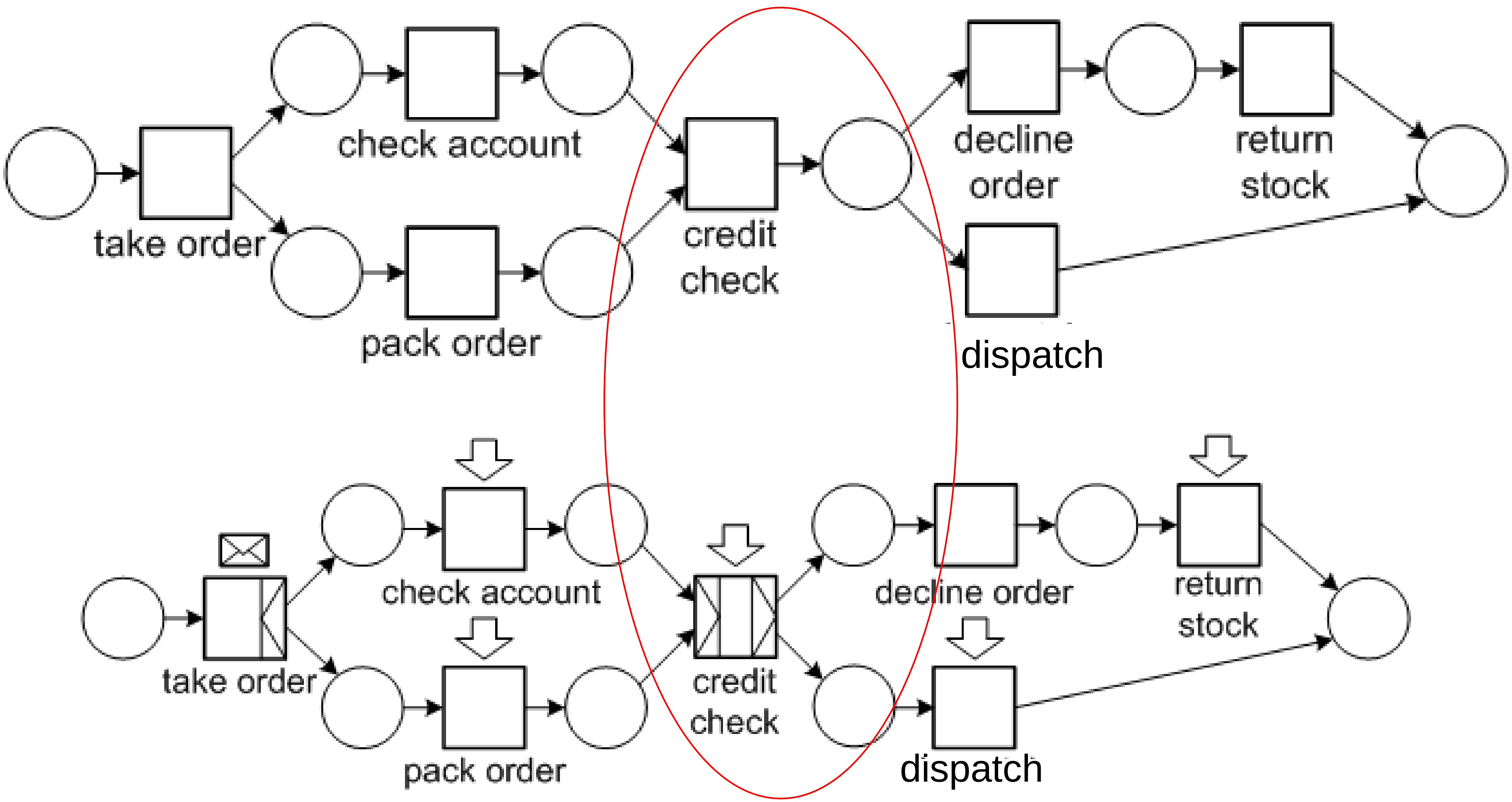
FORMAL DEFINITION:

- Let N= (P,T,F,A,l) be a labeled Petri net. **N is a Workflow net** iff (a) P contains an input place i such that $\bullet i = \phi$; (b) P contains an output place o s.t. $o\bullet = \phi$; (c) there is a directed path between any pair of nodes in the net, i.e., all transitions belong to a path from start to end.
- Note that BPs respond to the format of a WF-net.

Workflow nets notation

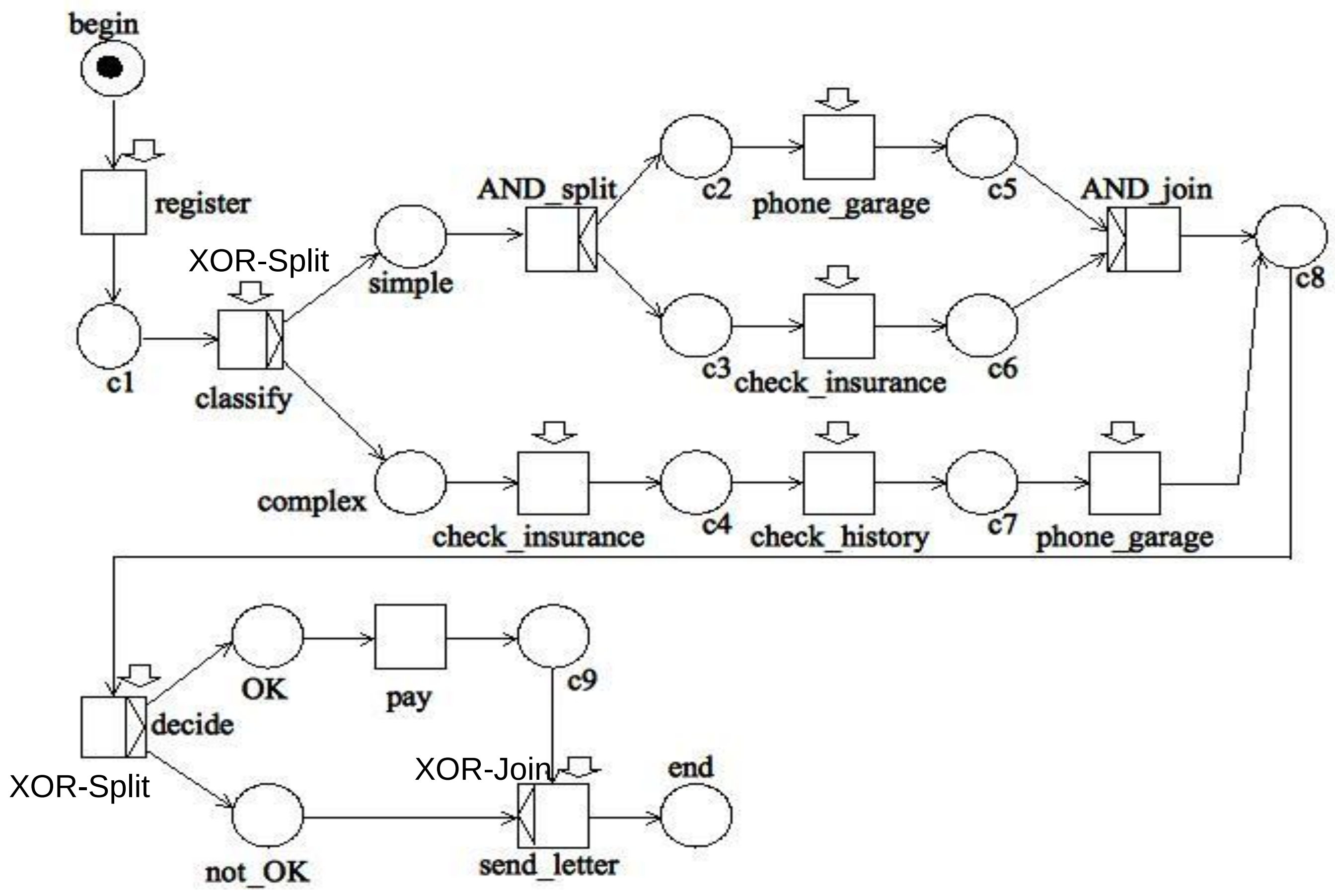


Workflow nets



Workflow nets – another example

Handling insurance claims

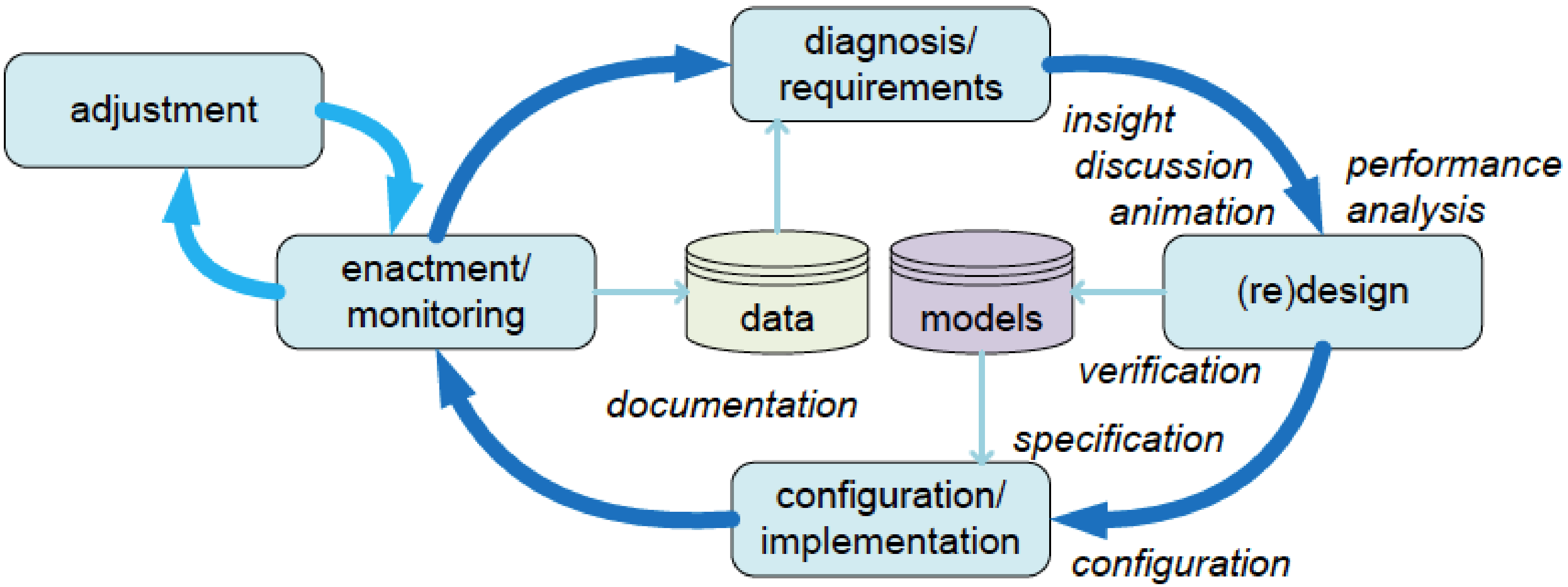


Workflow nets - soundness

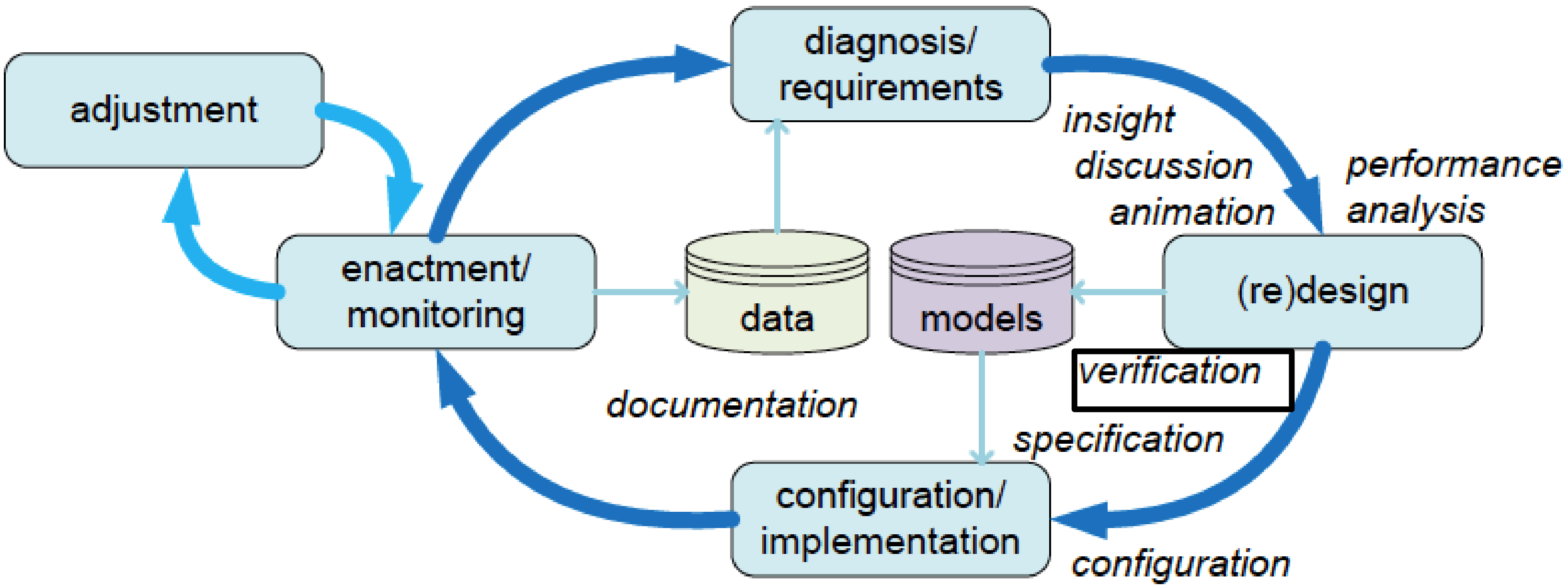
- Not all WF-nets represent a correct process.

- **[Safeness]** Places do not contain more than one token at the same time.
- **[Option to Complete]** Given an initial marking *i*, from every marking M reachable from *i*, $i \rightarrow^* M$, a marking *M'* can be reached that covers o, i.e. $M \rightarrow^* M'$ and M ≥ o.
  - *The net is free of deadlock and infinite loops.*
- **[Proper Completion]** Any marking M reachable from i, $M \rightarrow^* i$, that marks output place *o*, M ≥ *o*, marks no other place and only has one token in *o*, i.e. M = *o*.
  - *When the workflow terminates no other tasks are still running and termination is signalled only once.*
- **[No Dead Tasks]** For every transition t a marking M reachable from i, $i \rightarrow^* M$ can be found that enables t.
  - *The workflow does not contain any superfluous parts that can never be activated.*

Workflow nets - soundness



Workflow nets - soundness

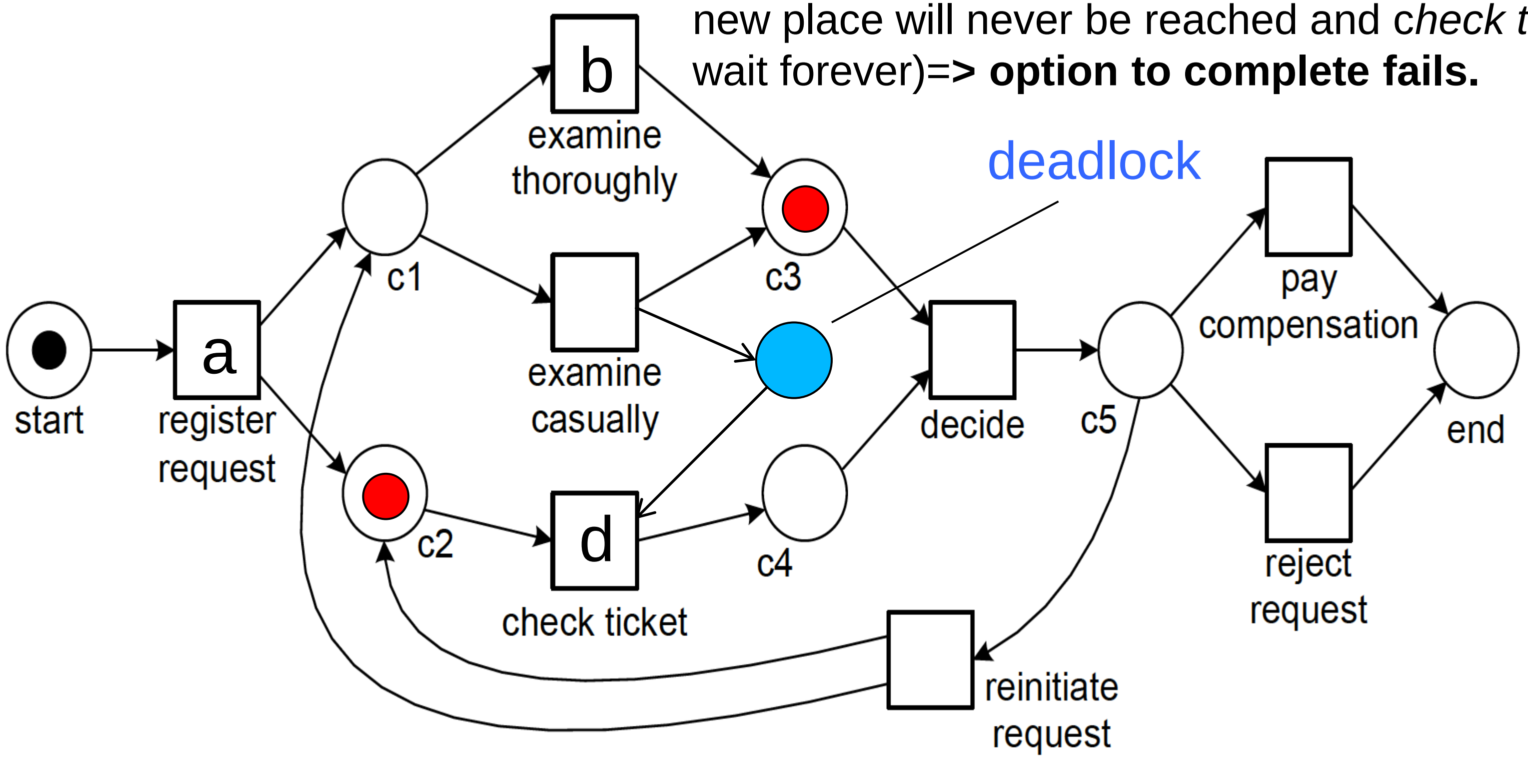


Business Process Lifecycle revisited

Workflow nets - soundness

check ticket must wait for *examine casually. But firing <a,b> generates the marking [c2,c3].* From this marking, the [*end*] marking cannot be reached => it is a *dead* marking: *d* will never be enabled because the new place will never be reached and *check ticket* will wait forever)=**> option to complete fails.**

Verification:



Outline

- Motivation
- Introduction
- Petri nets, and Workflow nets
- **Using Workflows to model Business Processes**
- Workflow Patterns and YAWL
- BPMN and BPEL
- Design
- A Database Vision
- Summary

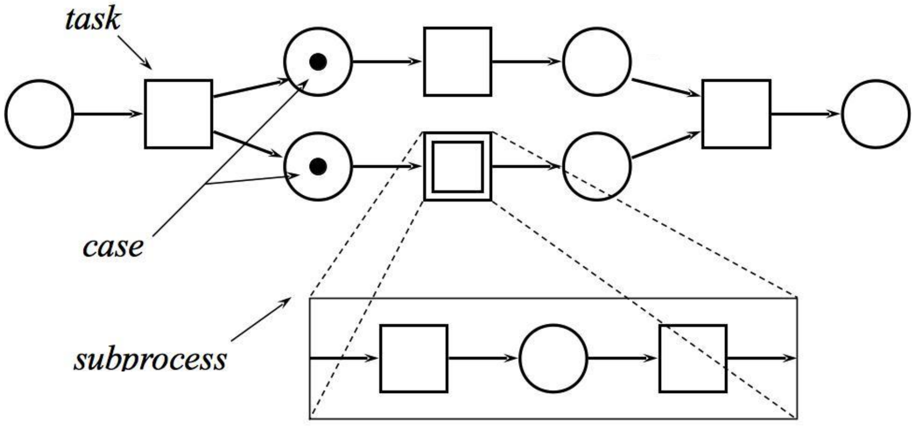
Representing Business Processes

- Three main components:

  - Processes
  - Routing
  - Enactment

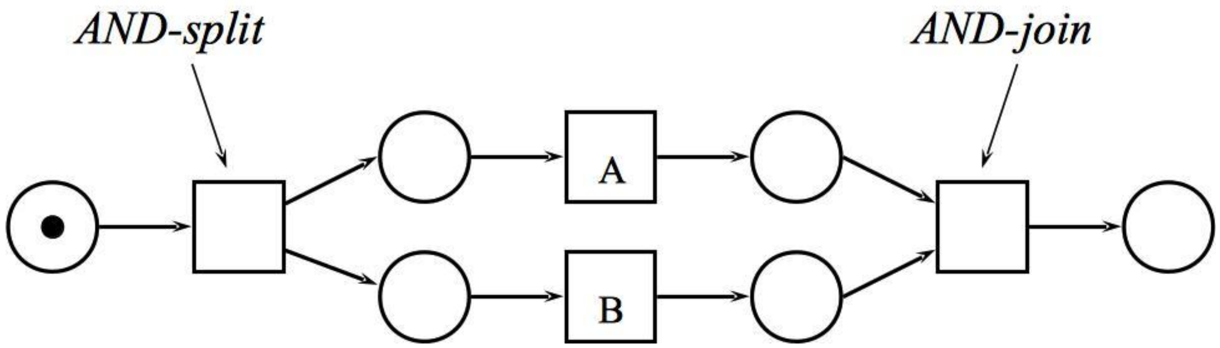
Representing processes

- A process indicates the tasks that must be carried out to perform a case.
- Processes may involve conditions, have sub-processes, and consist in the execution of many tasks.
- We have seen that they can be represented using Petri nets.
- Also, tasks in a workflow can be combined in a single process. In this way, a process can be composed of **sub-processes** which can also be represented using the **hierarchy extension** of Petri nets



Representing routing

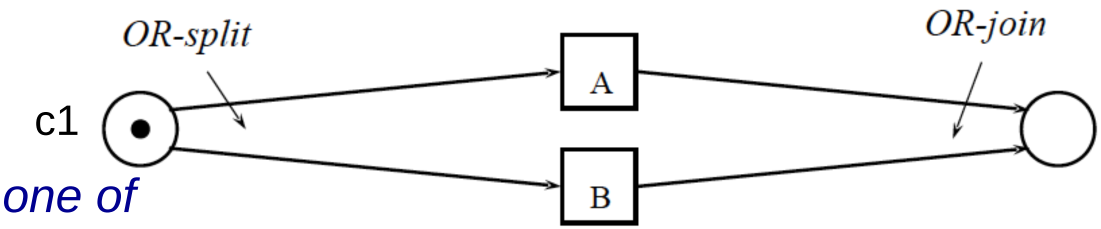
- There are four cases of routing:
  - Sequential: tasks are carried out one after the other, and modeled as two transitions linked by a place.
  - Parallel: two or more tasks can be carried out in parallel, in any order. (AND-split takes one token and creates two. AND-join takes two tokens and produces one).



Representing routing

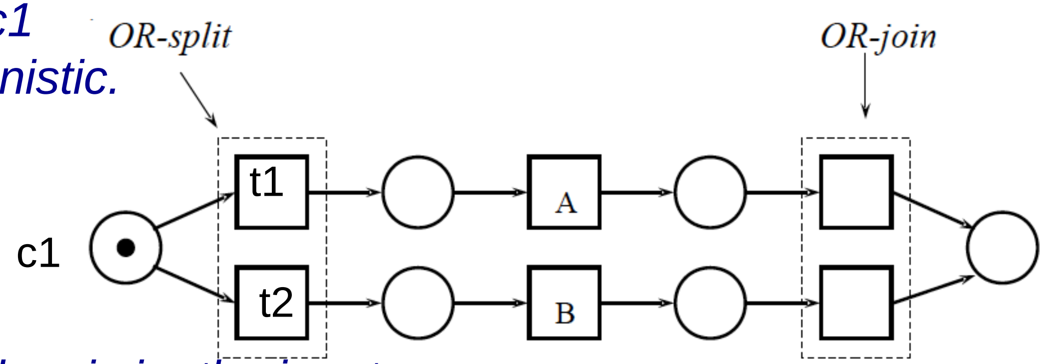
- Selective: Only one route is followed, depending on the outcome of a condition.

Implicit OR-Split



Choice made when one of the tasks (A or B) has to be performed (condition c1 fulfilled). Non-deterministic.

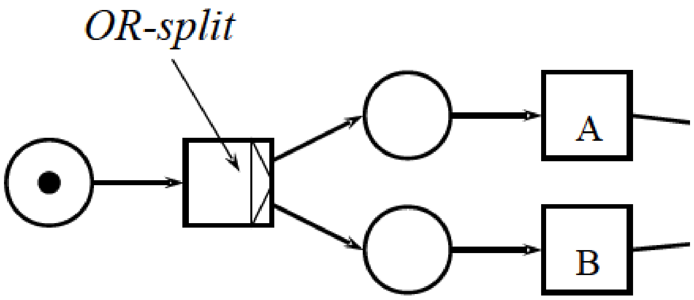
Explicit OR-Split



Choice made when token is in the input place. When condition c1 is fulfilled, t1 or t2 fires. Non-deterministic.

Representing routing

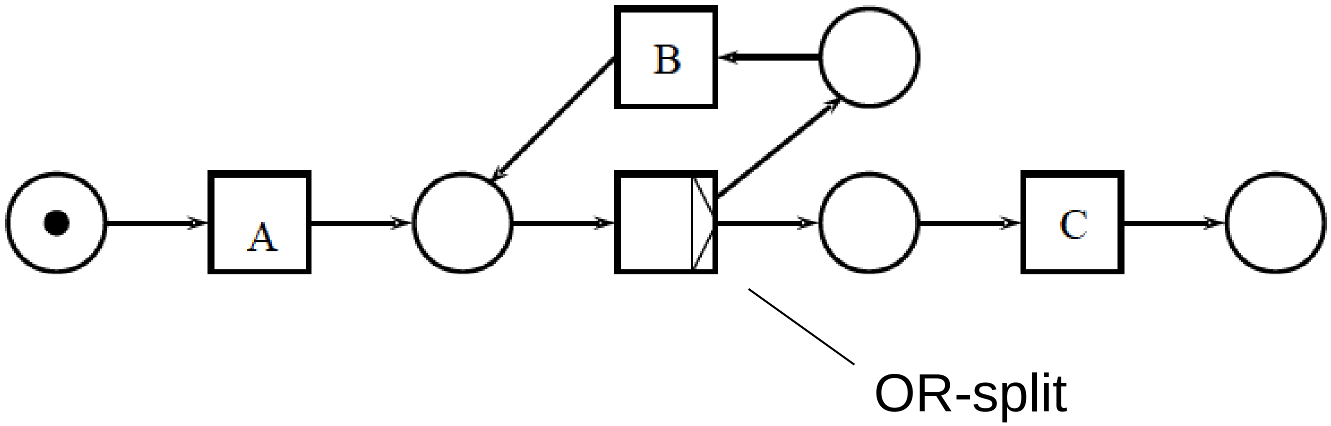
- *Deterministic* OR-Split: based on a condition.



Representing routing

- Iterative: A task or group of tasks, is executed repeatedly. The typical example is the repetition of a test until the desired outcome is achieved.

- A While..do example.



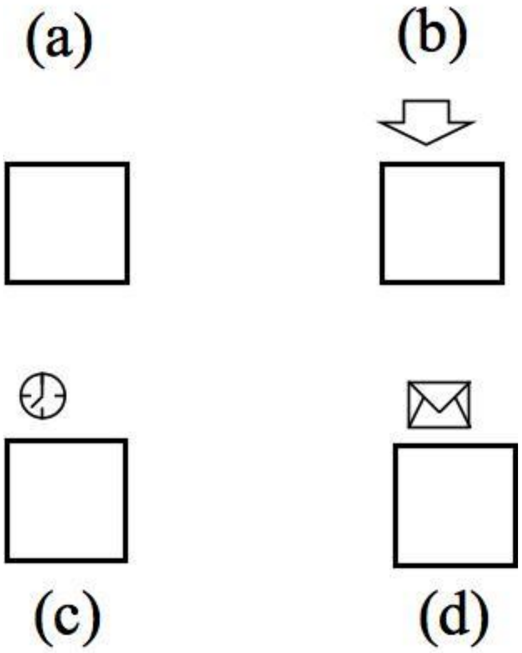
Representing enactment

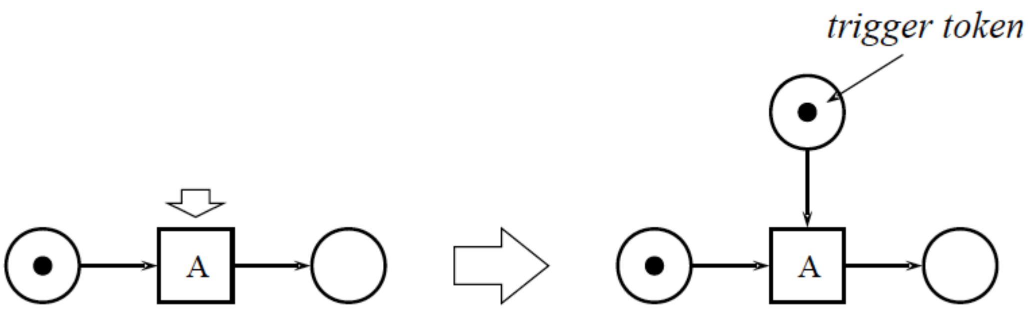
- Transitions are triggered as soon as the conditions in the places immediately behind them are satisfied (represented as single squares). Or:

    - (a) through a *resource* initiative, like an employee deciding to start the task.
    - (b) through an *external event*, like the arrival of a document.
    - (c) through a *time signal*, like the definition of an execution time for a task.

Representing enactment

- This symbology is a shorthand of different high-level Petri nets construct. For example, for case (b), the figure in the RHS shows the classic Petri net equivalent, which requires a “trigger” token.





Outline

- Motivation
- Introduction
- Petri nets, and Workflow nets
- Using Workflows to model Business Processes
- **Workflow Patterns and YAWL**
- BPMN and BPEL
- Design
- A Database Vision
- Summary

Workflow patterns

- Started in 1996, joint work TU/e and QUT (1999).
- Different types of patterns:
  - Control-flow patterns
  - Data patterns
  - Resource patterns
- Some of the people involved:
  Arthur ter Hofstede (QUT), Marlon Dumas (QUT), Nick Russel (QUT), Petia Wohed (DSV), Bartek Kiepuszewski (QUT), Alistair Barros (SAP), Oscar Ommert (EUT), Ton Pijpers (ATOS), Nataliya Muylar (EUT), Maja Pesic (EUT), Alexander Norta (EUT), Eric Verbeek, et al.

www.workflowpatterns.com

Workflow patterns

Control-flow perspective

- Focuses on the representation and execution of processes in terms of tasks and arcs indicating how the thread of control is passed between them.
- Abstracts from the actual implementation of individual tasks.

Data perspective

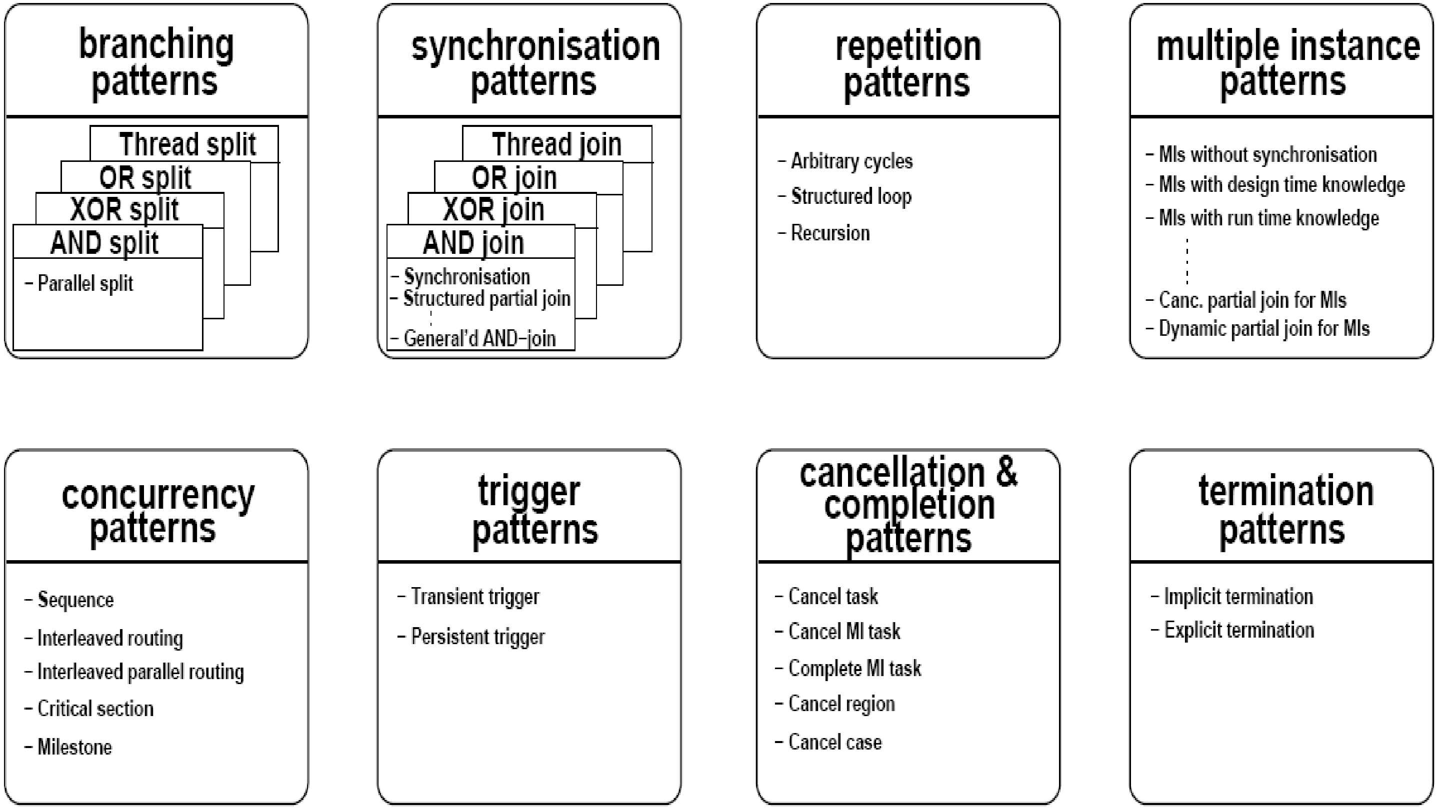
- Focuses on the representation and utilization of data in a process context.
- Considers both internal and external data resources and the interactions between them.

Resource perspective

- Focuses on the manner in which work is offered to, allocated to and managed by process participants.
- Considers both the system and resource perspectives.
- Assumes the existence of a process model and related organizational and work distribution models.

- **We only address control flow in the talk.**

Control flow patterns



Control flow patterns

- Branching Patterns
  capture branching scenarios in processes.
- Synchronisation Patterns
  describe synchronization scenarios arising in processes.
- Repetition Patterns
  describe various ways in which repetition may be specified.
- Multiple Instances (MI) Patterns
  delineate situations with multiple threads of execution in a workflow which relate to the same activity.
- Concurrency Patterns
  reflect situations where restrictions are imposed on the extent of concurrent control-flow in a process instance
- Trigger Patterns
  catalogue the different triggering mechanisms appearing in a process context.
- Cancellation and Completion Patterns
  categorize the various cancellation scenarios that may be relevant for a workflow specification.
- Termination Patterns
  address the issue of when the execution of a workflow is considered to be finished.

Branching patterns

AND Split

- Parallel split, initiation of parallel threads

OR Split

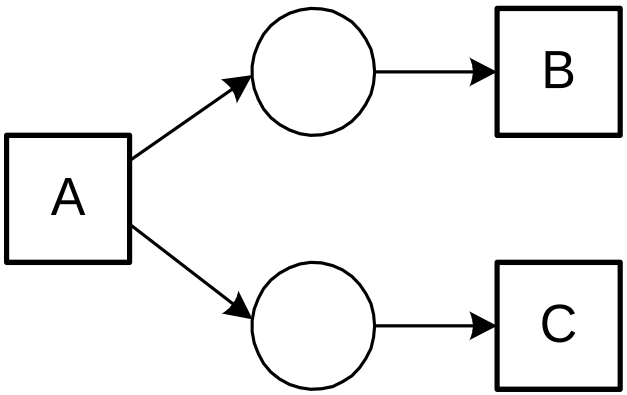
- Multi-choice, thread of control is passed to one or more outgoing branches

XOR Split

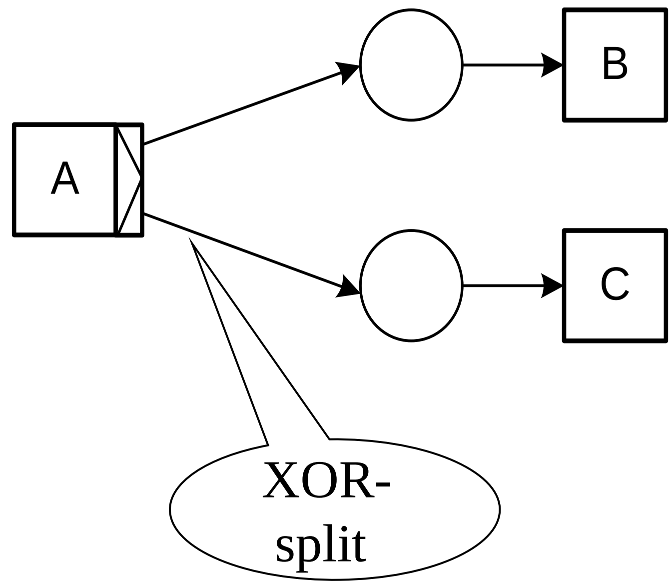
- Exclusive choice, thread of control is passed to exactly one of the outgoing branches
- Deferred choice, thread of control is passed to exactly one of the outgoing branches. Selection decision is deferred to the user and/or operating environment.

Branching patterns

Parallel split



XOR-split (exclusive choice)



Deferred choice

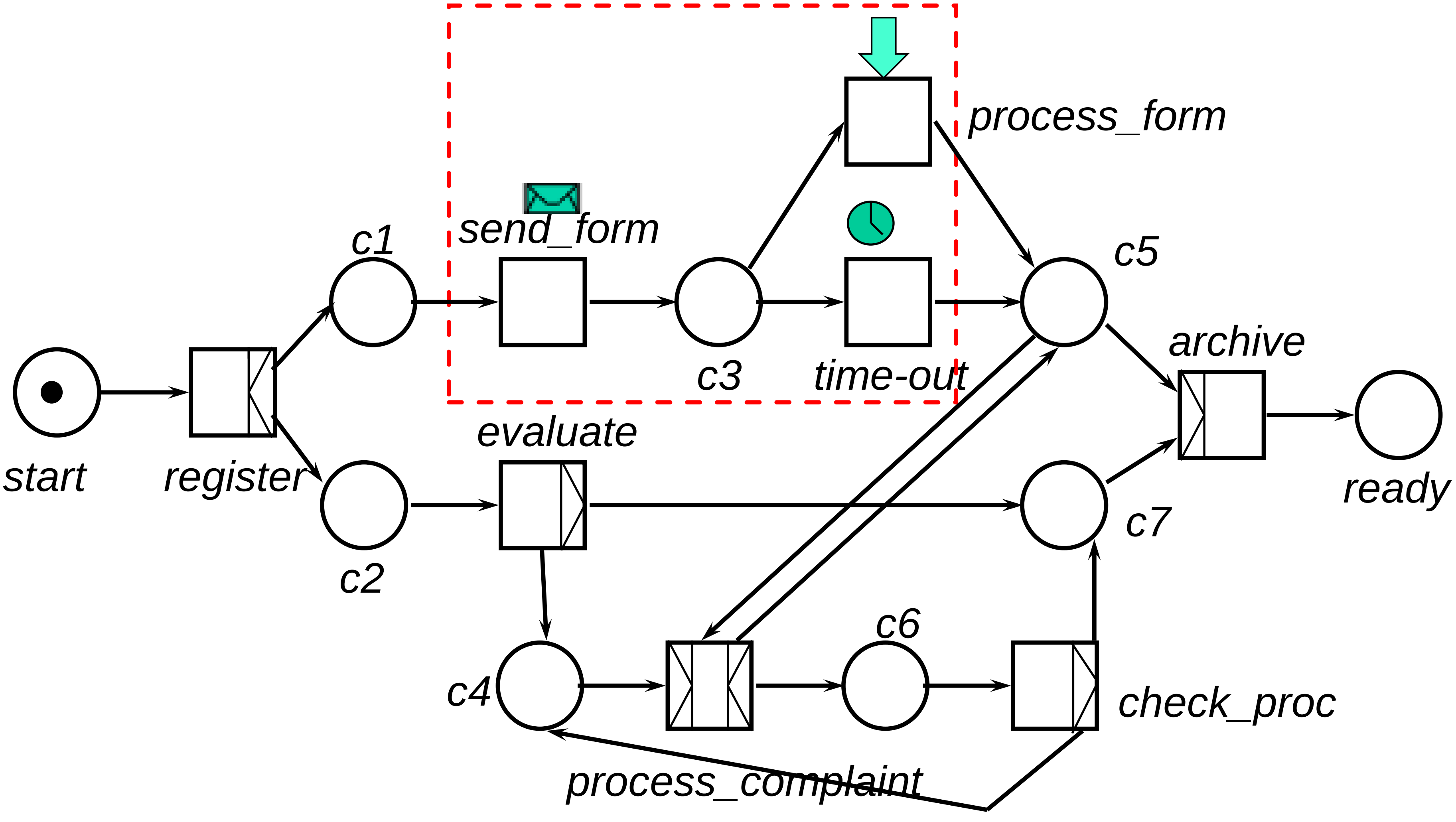
- Choice made by the environment not the system.

- Essential in workflow context.

- Naturally supported by notations that offer direct support for the notion of state, e.g. statecharts or Petri nets.

- Exclusive choice: choice made by the system, based on data

Deferred choice



Synchronisation patterns: some AND-join variants

- Generalized AND-Join, waits for all incoming threads
- Structured partial join, waits for some (but not all) incoming threads.
- Cancelling partial join, waits for some incoming threads, cancel the rest.

Cancellation and completion patterns

Categorize the various cancellation scenarios that may be relevant for a workflow specification

- Cancel task, withdraws a specified task instance.
- Cancel case, withdraws all task instances in a case.
- Cancel region, withdraws task instances in a specified region of a process. This region is emptied of tokens upon completion of that task.

YAWL (Yet Another Workflow Language)

- Defined by Wil van der Aalst and Arthur ter Hofstede in 2002
- Intention: to provide comprehensive support for the workflow patterns.
- Inspired by Workflow nets, but with direct support for
  - Cancelation.
  - Multiple executions of the same task in the same process instance.
  - Synchronization of active paths only (OR-join).
- YAWL has a support environment (Development started in 2003)
  - Editor.
  - Analysis.
  - Verification.

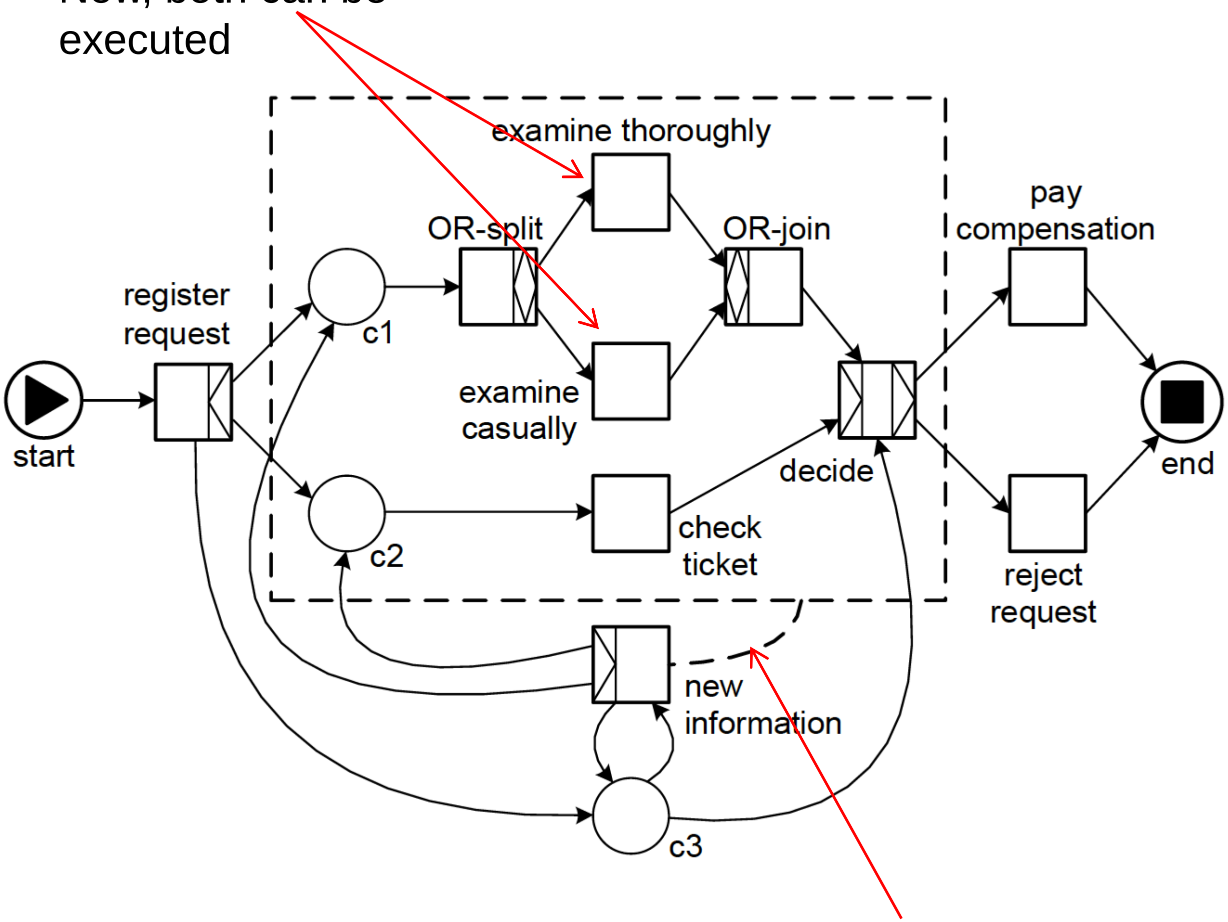
YAWL – (cont.)

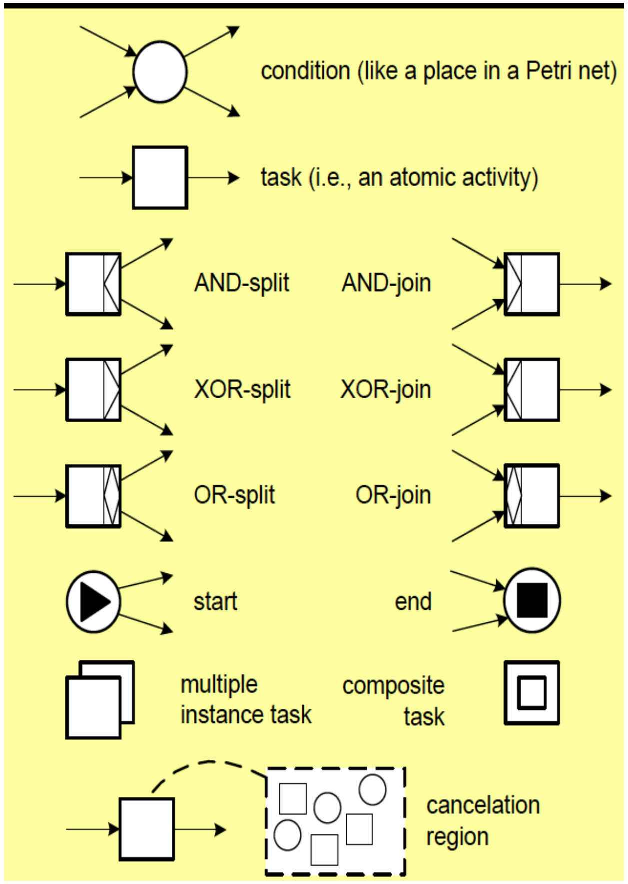
- Comprehensive approach for the Workflow Patterns
  - Original control-flow patterns, resource patterns, and exception handling patterns.
- Formal semantics
  - Original definition of YAWL: state-transition system.
  - Later: CPN (Coloured Petri nets) interpreter.
  - This removes ambiguity and allows verification.
- Flexibility support, eg., through handling exceptions.

- See **www.yawlfoundation.org**

YAWL – notation

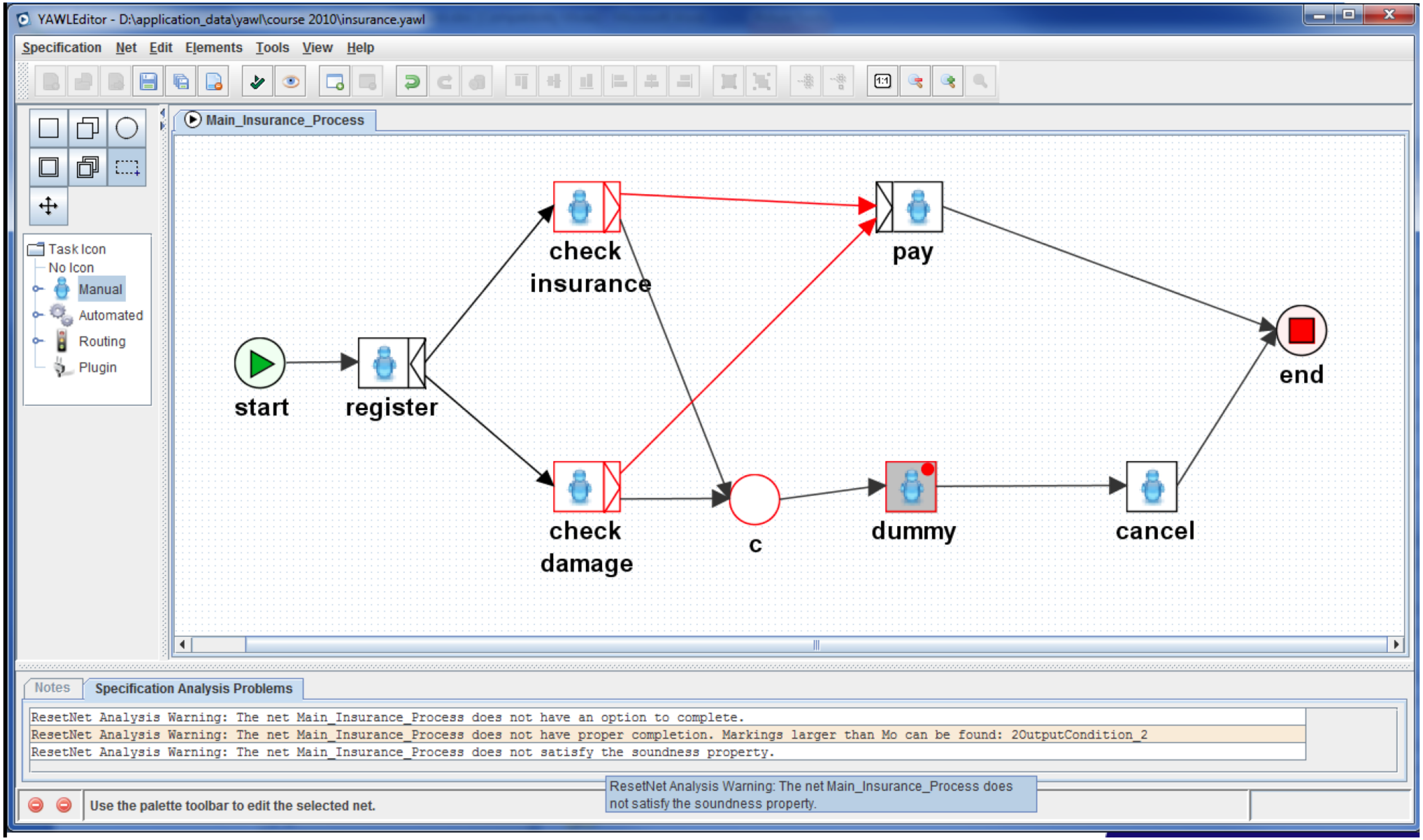
Now, both can be executed





When there is a token in c3, and *new-information* is executed, all tokens are removed from the cancellation region.

YAWL - software



Outline

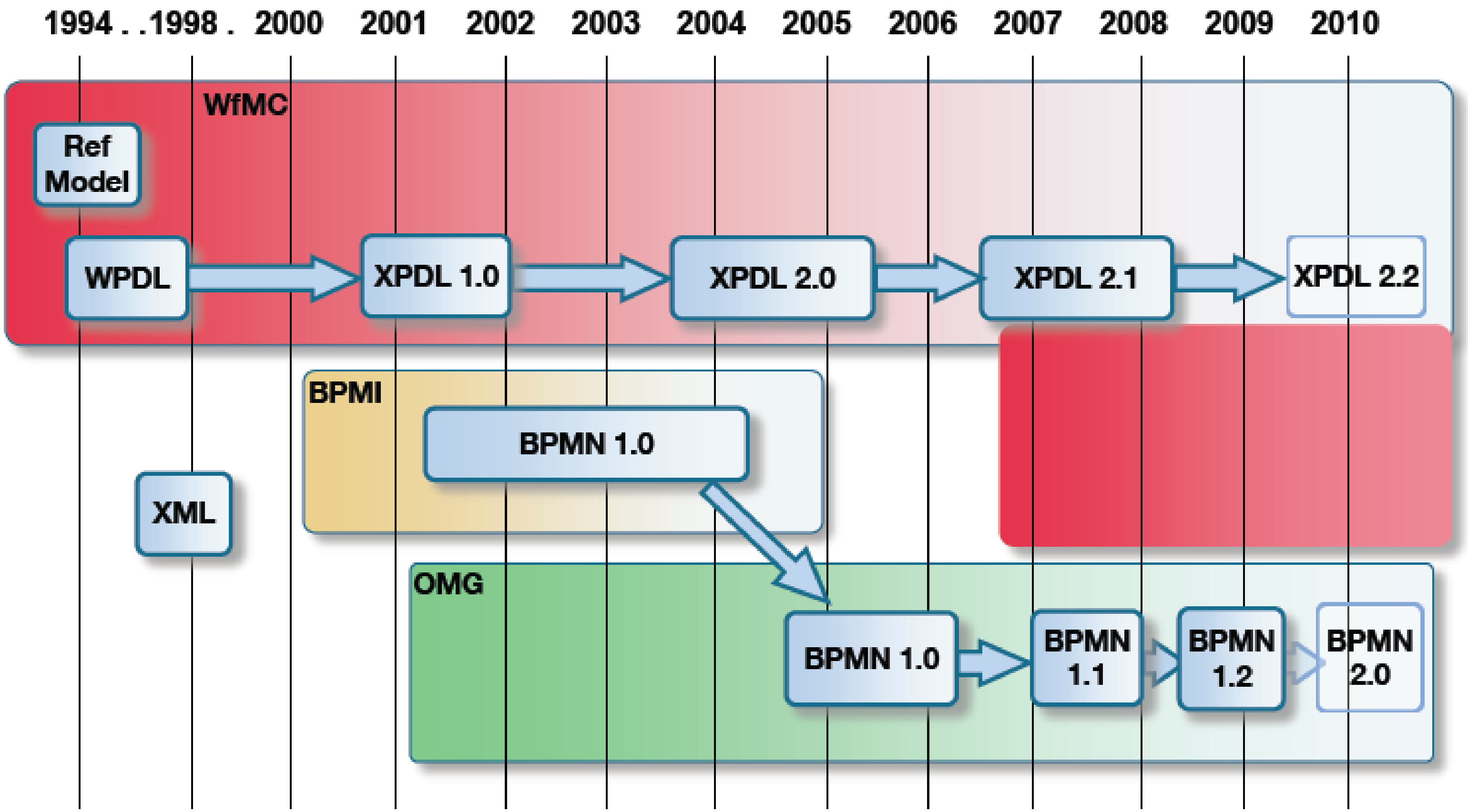
- Motivation
- Introduction
- Petri nets, and Workflow nets
- Using Workflows to model Business Processes
- Workflow Patterns and YAWL
- **BPMN and BPEL**
- Design
- A Database Vision
- Summary

BPMN

- BPMN (Business Process Modeling Notation) one of the most widely used to model BPs.
- Supported by most vendors.
- Standardized by OMG.
- BPMN aimed at:
    - (a) being acceptable and usable by the business community;
    - (b) being constrained to support only the concepts of modeling that are applicable to BPs;
    - (c) describing clearly a complex executable process.

- Differences with YAWL
    - routing logic associated with gateways rather than tasks.
    - events (unlike places), cannot have multiple incoming arcs.

BPMN - history



BPMN 2.0

- The BPMN 1.0 specification did not formally define the semantics of the Business Process Diagram.
- BPMN 2.0 partially solves this, and also contains significant changes, namely:
    - New event types: parallel multiple events.
    - Parallel event-based gateway.
    - Event sub-processes only carried out when an event occurs.
    - Updates on collaboration modeling.
    - Two new diagram types: (a) Choreography diagram, modeling data exchange between partners, where each data exchange is modeled as an activity. (b) Conversation diagram, an overview of several partners and their links.

BPMN 2.0

- **Five basic categories of elements:**

- Flow Objects.
  - Events
  - Activities
  - Gateways

- Data Objects.
  - Data objects
  - Data inputs
  - Data outputs
  - Data stores

BPMN 2.0

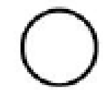
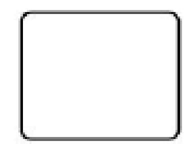
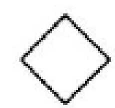
- **Five basic categories of elements:**

- Connecting Objects.
    - Sequence Flows;
    - Message Flows;
    - Associations;
    - Data Associations.

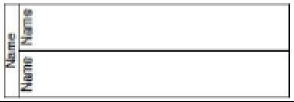
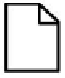
- Swimlanes.
  Used to group the primary modeling elements. Can be of two forms: Pools and Lanes.

- Artifacts. Used to provide additional information about the process. Include Group, and Text Annotation.

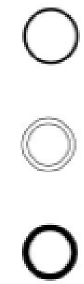
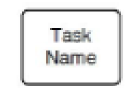
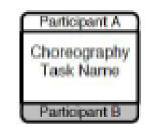
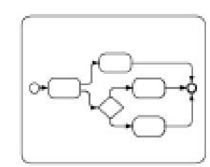
BPMN 2.0. Basic elements (from the OMG standard)

#	Element	Description	Notation
1	Event	An event is something that happens during the course of a process or a choreography. Events affect the flow of the model and usually have a cause (trigger) or an impact (result). There are three types of events, based on when they affect the flow: start, intermediate, and end	
2	Activity	An activity is a generic term for work that company performs in a process. An activity can be atomic or non-atomic (compound). The types of activities that are part of a process model are: *sub-process* and *task*. Activities are used in both standard processes and in choreographies	
3	Gateway	A gateway is used to control the divergence and convergence of sequence flows in a process and in a choreography. Thus, it will determine branching, forking, merging, and joining of paths. Internal markers will indicate the type of behavior control.	

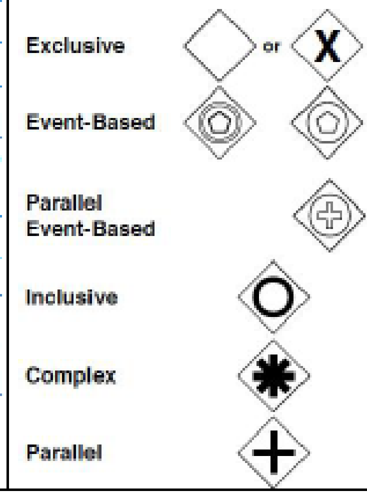
BPMN 2.0. Basic elements

4	Sequence Flow	A sequence flow is used to show the order that activities will be performed in a process and in a choreography	
5	Message Flow	A message flow is used to show the flow of messages between two participants that are prepared to send and receive them	
6	Association	An association is used to link information and artifacts with BPMN graphical elements.	
7	Pool	Graphical representation of a participant in a collaboration. A pool MAY have internal details, in the form of the process that will be executed. Or a Pool MAY have no internal details, i.e., it can be a "black box."	
8	Lane	A sub-partition within a process, sometimes within a pool. Extends the entire length of the process, either vertically or horizontally.	
9	Data Object	Provides information about what activities require to be performed and/or what they produce. Data objects can represent a singular object or a collection of objects. Data input and data output provide the same information for processes.	
10	Message	Is used to depict the contents of a communication between two participants.	
11	Text Annotations	A mechanism for a modeler to provide additional text information for the reader of a BPMN Diagram.	

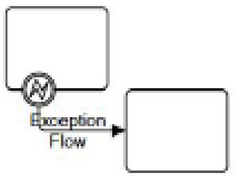
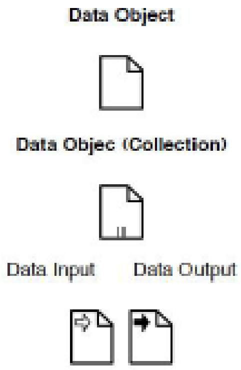
BPMN 2.0. Extended elements

#	Element	Description	Notation
1	Events	A start event indicates where a particular process or choreography will start. Intermediate events occur between a start event and an end event. They will not start or (directly) terminate the process. The end event indicates where a process or choreography will end.	
2	Task (atomic)	A task is an atomic activity that is included within a process. A task is used when the work in the process is not broken down to a finer level of process detail.	
3	Coreography	A choreography task is an atomic activity in a choreography. It represents a set of one (1) or more message exchanges. Each choreography task involves two (2) participants. The name of the choreography task and each of the participants are all displayed in the different bands that make up the shapes graphical notation.	
4	Collapsed Sub-process	The details of the sub-process are not visible in the diagram. A 'plus' sign in the lower-center of the shape indicates that the activity is a sub-process and has a lower level of detail.	
5	Expanded Sub-process	The boundary of the sub-process is expanded and the details (a process) are visible within its boundary. Note that sequence flows cannot cross the boundary of a sub-process.	

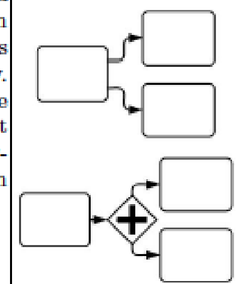
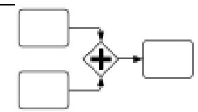
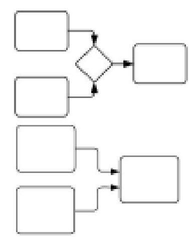
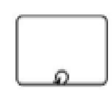
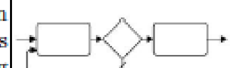
BPMN 2.0. Extended elements

8	Gateway types	Icons within the diamond shape of the gateway will indicate the type of flow control behavior. The types of control include: (a) exclusive decision and merging. Both exclusive and event-based perform exclusive decisions and merging Exclusive can be shown with or without the 'X' marker. (b) event-based and parallel event-based gateways can start a new instance of the process. (c) inclusive gateway decision and merging. (d) complex gateway – complex conditions and situations (e.g., 3 out of 5). (e) parallel gateway forking and joining. Each type of control affects both the incoming and outgoing flow.	

BPMN 2.0. Extended elements

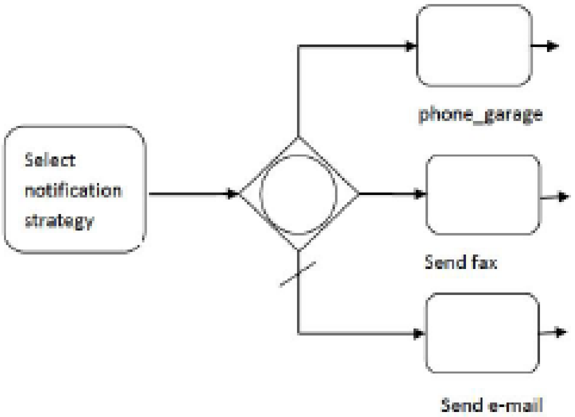
#	Element	Description	Notation
1	Conditional Flow	A sequence flow can have a condition expression that is evaluated at runtime to determine whether or not the sequence flow will be used (i.e., will a token travel down the sequence flow). If the conditional flow is outgoing from an activity, then the sequence flow will have a minidiamond at the beginning of the connector. If the conditional flow is outgoing from a gateway, then the line will not have a mini-diamond).	
2	Exception Flow	Occurs outside the normal flow of the process and is based upon an Intermediate event attached to the boundary of an activity that occurs during the performance of the process.	
3	Data Objects	Data Objects provide information about what activities require to be performed and/or what they produce. Data objects can represent a singular object or a collection of objects. Data input and data output provide the same information for processes.	

BPMN 2.0. Extended elements

4	Fork	BPMN uses the term 'fork' to refer to the dividing of a path into two or more parallel paths (also known as an AND-Split). It is a place in the process where activities can be performed concurrently, rather than sequentially. There are two options: (a) multiple outgoing sequence Flows can be used. This represents 'uncontrolled' flow. It is the preferred method for most situations. (b) A parallel gateway can be used. This is used rarely, usually in combination with other gateways.	
5	Join	BPMN uses the term 'join' to refer to the combining of two or more parallel paths into one path (also known as an AND-Join or synchronization). A parallel gateway is used to show the joining of multiple sequence flows.	
6	Merging	BPMN uses the term 'merge' to refer to the exclusive combining of two or more paths into one path (also known as an OR-Join). A merging exclusive gateway is used to show the merging of multiple sequence flows. If all the incoming flow is alternative, then a gateway is not needed. That is, uncontrolled flow provides the same behavior.	
7	Activity Loop	The attributes of tasks and sub-processes will determine if they are repeated or performed once. There are two types of loops: standard and multi- instance. A small looping indicator will be displayed at the bottom-center of the activity.	
8	Sequence Flow Loop	Loops can be created by connecting a sequence flow to an upstream object. An object is considered to be upstream if it has an outgoing sequence flow that leads to a series of other sequence flows, the last of which is an incoming sequence flow for the original object.	

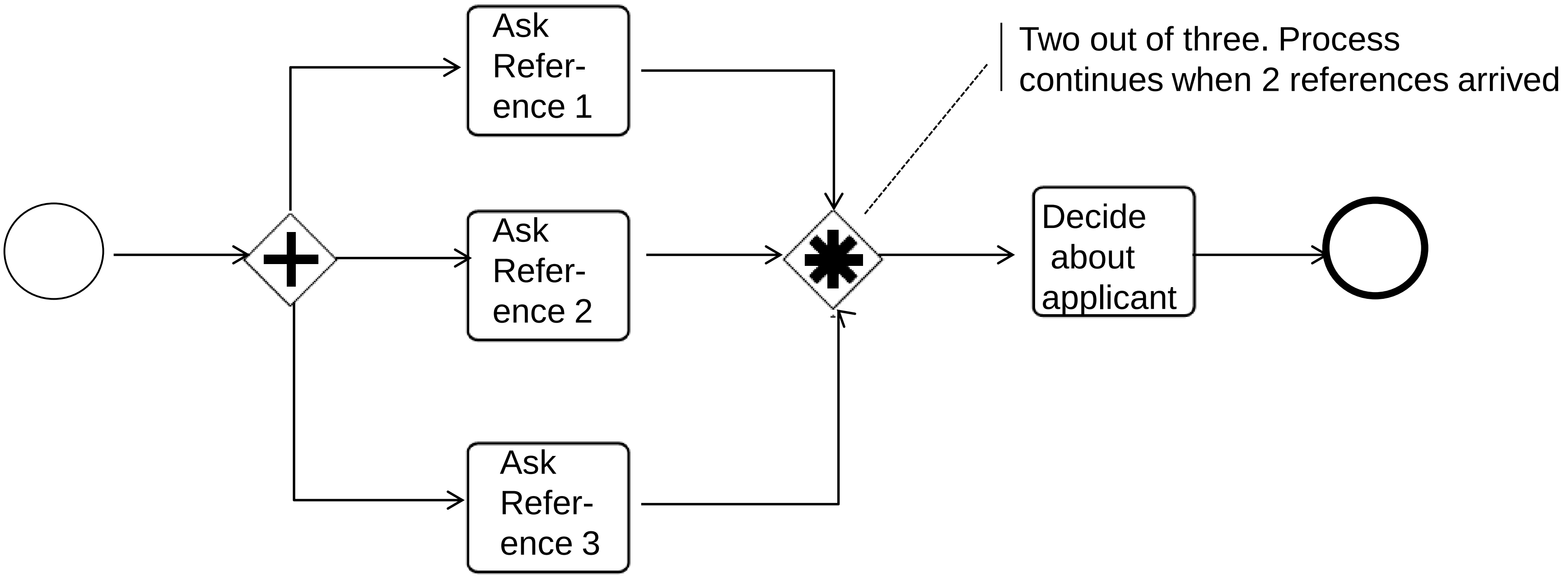
BPMN 2.0. Gateways

- Parallel gateways could be used to represent an AND-split (with the same semantics: a token is created for each output path).
- Exclusive gateways model an XOR-split.
- Inclusive gateways allow selecting or merging one or more paths (an OR-split). In the example below, email cannot be combined with any of the other two communication means. But phone garage and fax can be used together.

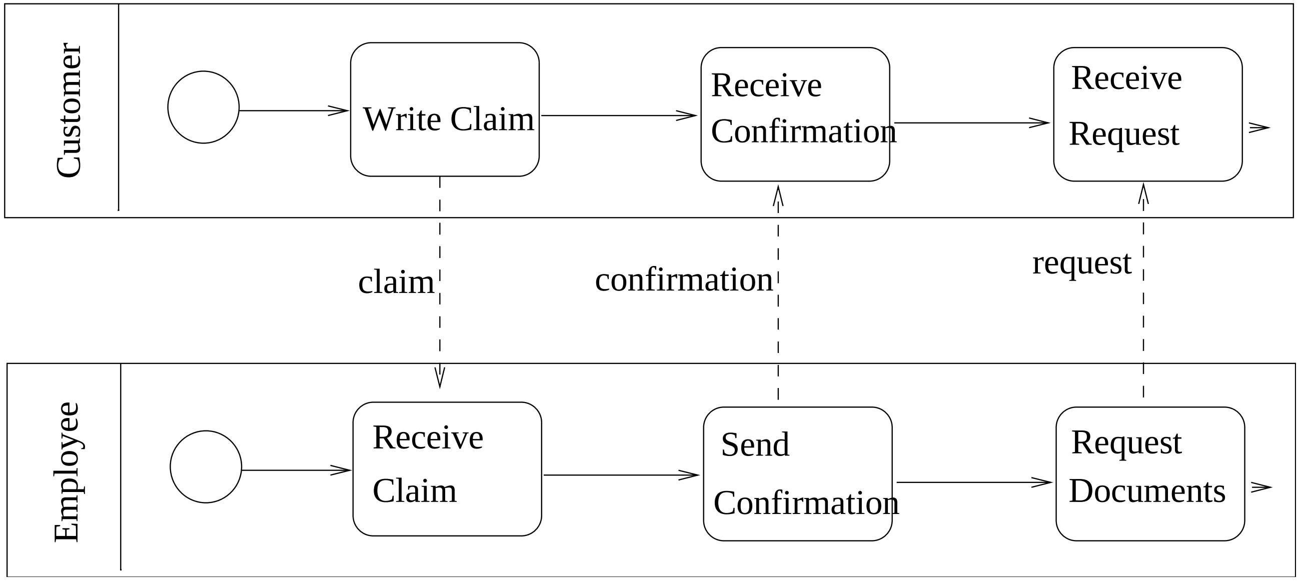


BPMN 2.0. Gateways

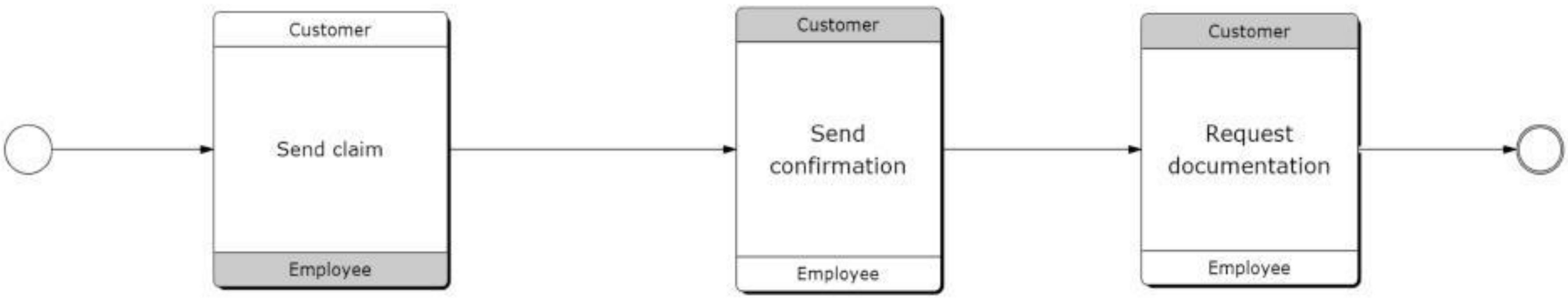
Complex gateways- Use arbitrary rules



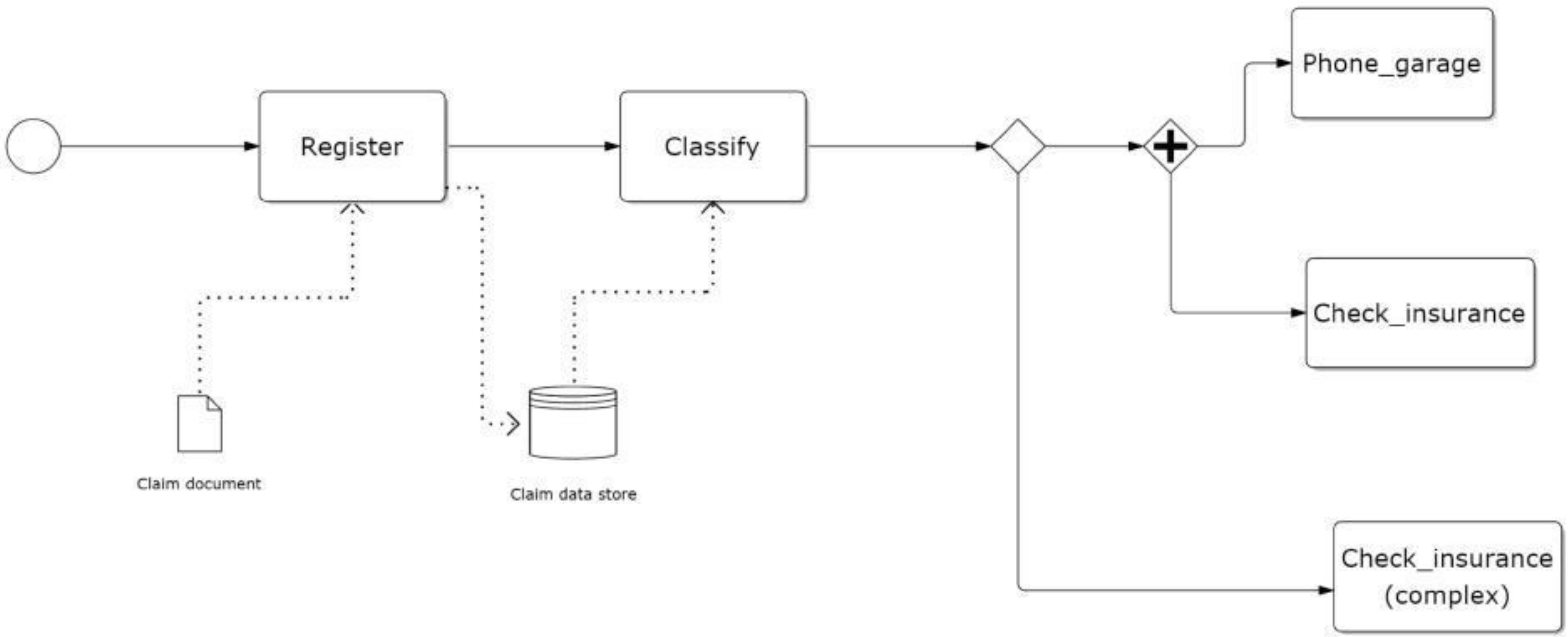
BPMN 2.0. Collaboration



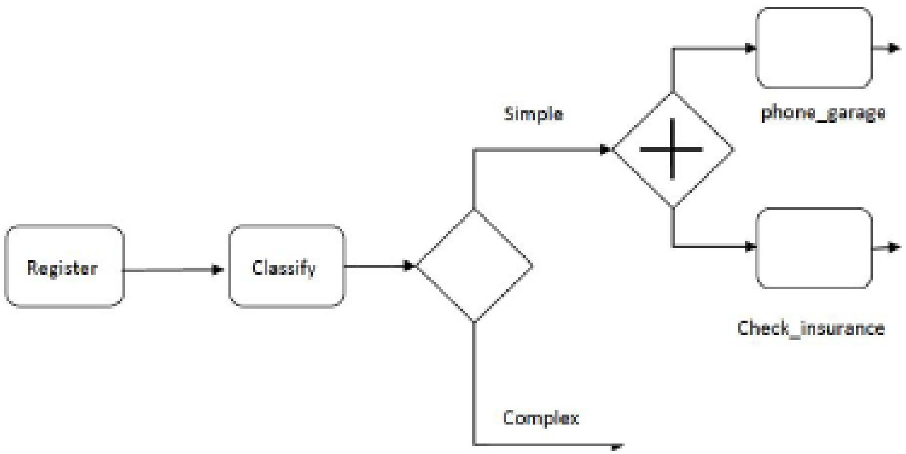
BPMN 2.0. Choreography

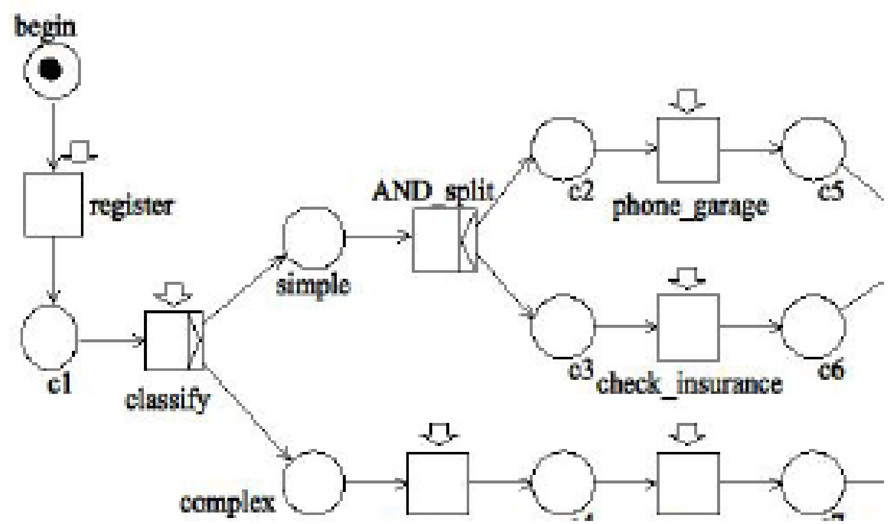


BPMN 2.0. Data objects

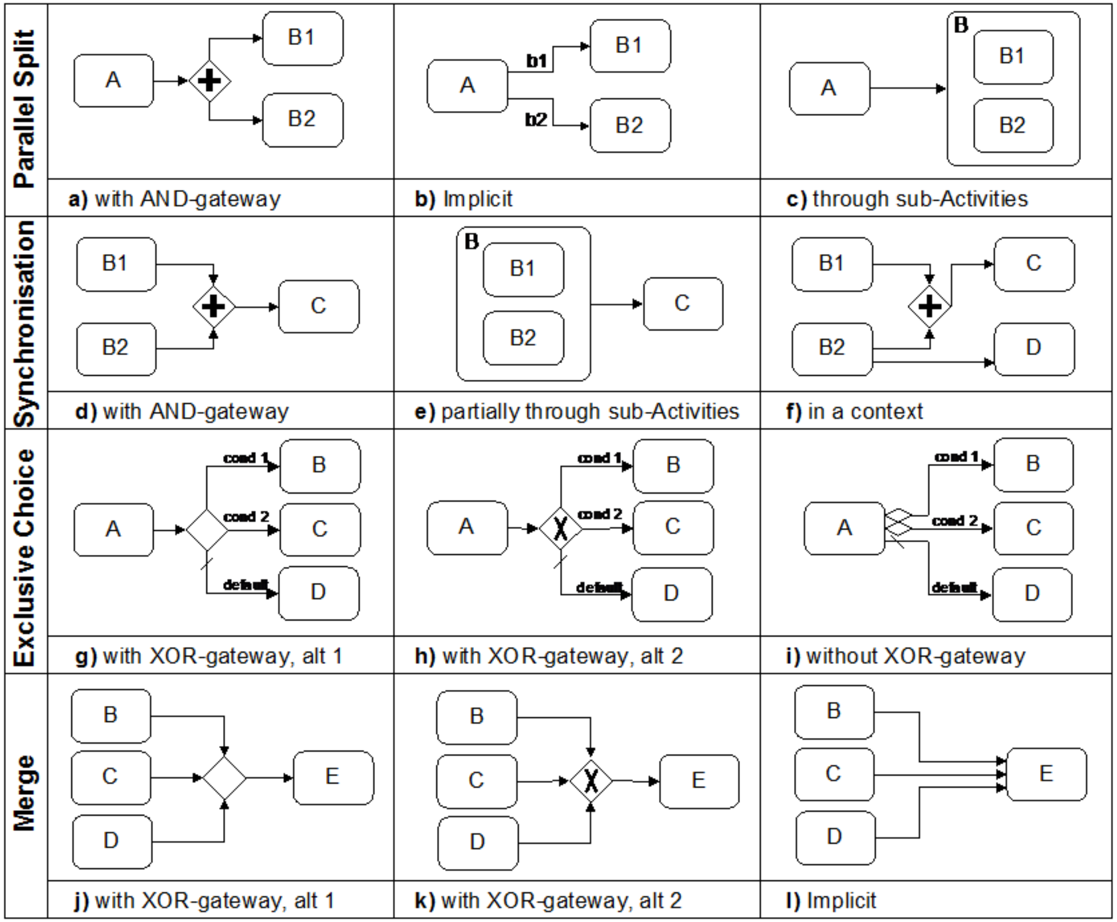


BPMN 2.0. Examples

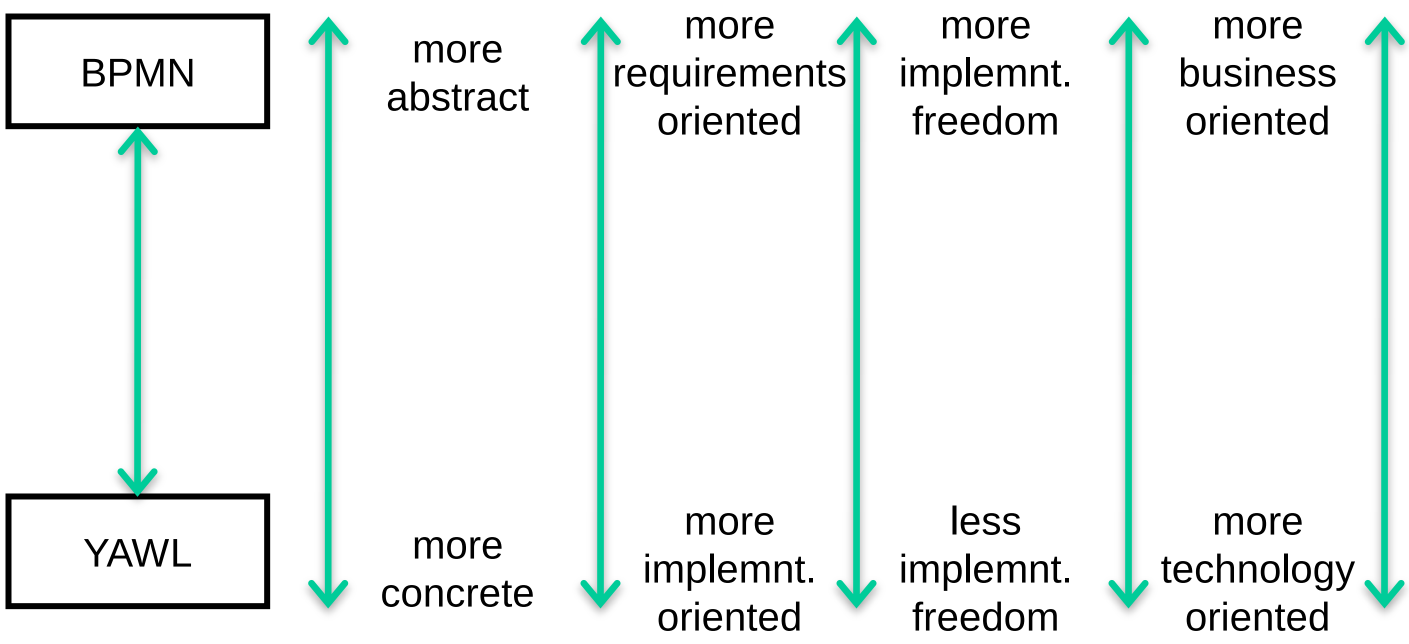




BPMN Solutions for basic patterns



BPMN vs YAWL


BPMN
YAWL
more abstract
more concrete
more requirements oriented
more implemnt. oriented
more implemnt. freedom
less implemnt. freedom
more business oriented
more technology oriented

Executing BPMN: BPEL

- BPEL is an OASIS (Organization for the Advancement of Structured Information Standards) standard executable language for specifying actions within BPs with web services.
- BPEL exchanges information by using web service interfaces exclusively.
- Provides a language for the specification of executable and abstract Business Processes.
- IBM and Microsoft defined their own, fairly similar languages: WSFL and XLANG, respectively. Then, combined these languages into BPEL4WS.
- BPEL4WS provides a language for the formal specification of business processes and business interaction protocols, extending the web services interaction model enabling it to support business transactions.
- BPEL adopts web services as its external communication mechanism. Thus, its messaging facilities depend on the Web Services Description Language (WSDL) 1.1 to describe outgoing and incoming messages.

Executing BPMN: BPEL

- No standard graphical notation for BPEL => vendors have created their own notations, enabling a direct visual representation of BPEL process descriptions (e.g., ORACLE BPEL).
- Other ones have proposed to use BPMN as a graphical front-end to capture BPEL process descriptions. Mapping of BPMN to BPEL has been implemented in a number of tools (e.g. BPMN2BPEL).
- Difficult to generate BPEL code from BPMN models.
- BPEL not a modeling language, although it has been used as such in database research.
- Moreover, there are tools that can execute BP from BPMN 2.0 specifications: Activiti, jBPM5, BizAgi, Roubroo. (See article: “BPEL: who needs it anyway?” http://www.bpm.com/bpel-who-needs-it.html)

BPEL process definition

- BPEL defines an executable process by specifying
  - *Activities* and their execution order
  - *Partners* interacting with the process
  - *Data* necessary for and resulting from the execution
  - *Messages* exchanged between the partners
  - *Fault handing* in case of errors and exceptions
- Example: a simplified structure of a BPEL process

```
<process name="..."
   targetNamespace="http://www..." >
   ...
   <partnerLinks> ...
   <messageExchanges> ...
   <variables> …
   …
   <faultHandlers> ...
   <eventHandlers> …
   ...
   activity
</process>
```

BPEL process definition

- Basic activities
  - *invoke*: invoking operations offered by partner Web services
  - *receive*: waiting for messages from partner Web services
  - *reply*: for capturing interactions
  - *wait*: delaying the process execution
  - *assign*: updating variables
  - *throw*: signaling faults
  - *rethrow*: propogating the faults that are not solved
  - *empty*: doing nothing
  - *exit*: ending a process immediately

```
<invoke name = "..."
    partnerLink = "..."
    operation = " ..."
    inputVariable = "..."
    outputVariable = "..." />
```

```
<receive name = "..."
    partnerLink = "..."
    operation = " ..."
    variable = "..."
    createInstance = "yes/no"
    messageExchange = "..." />
```

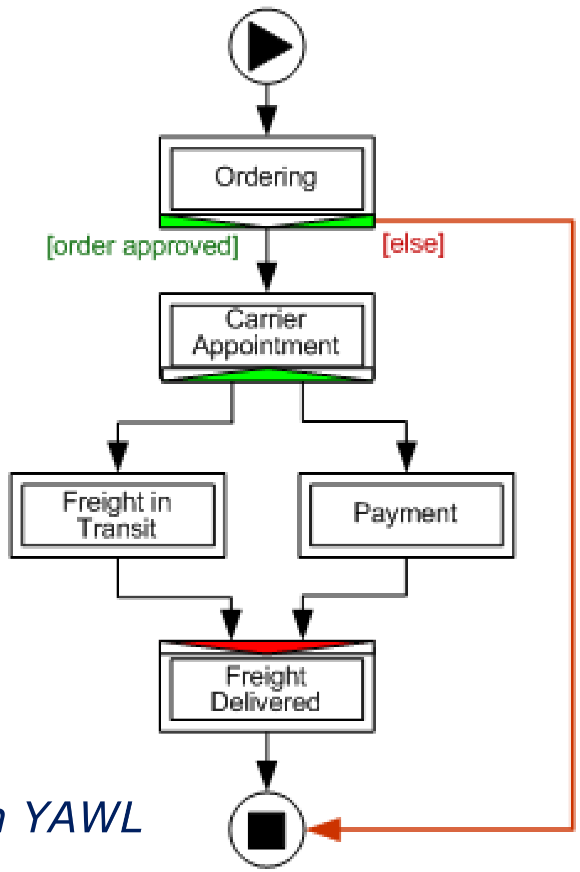
BPEL process definition

- Structured activities
  - *sequence*: activities being executed sequentially
  - *flow*: activities being executed in parallel
  - *if*: capturing conditional routing
  - *while*: structured looping
    - Condition is evaluated at the beginning of each iteration
  - *repeatUntil*: structured looping
    - Condition is evaluated at the end of each iteration
  - *forEach*: executing multiple instances

    of an activity with synchronisation
  - *scope*: grouping activities into blocks

```
<scope name="..." >
   <variables> ...
   <partnerLinks> ...
   <messageExchanges>...
   ....
   <eventHanlders>...
   <faultHandlers>...
   <compensationHanlders>...
   ...
   activity
</scope>
```

BPEL process definition



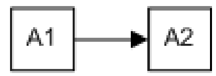
In YAWL

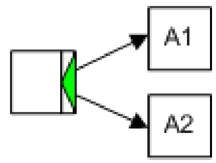
```
<sequence>
    <scope name = "Ordering"> ......... </scope>
    <if>
        <condition>
            $POApprovalResult = "approved"
        </condition>
        <sequence name ="continue">
            <scope name = "CarrierAppointment"> ......... </scope>
                <flow>
                    <scope name = "FreightInTransit"> ......... </scope>
                    <scope name = "Payment"> ......... </scope>
                </flow>
            <scope name = "FreightDelivered"> ......... </scope>
        </sequence>
        <else>
            <exit />
        </else>
    </if>
</sequence>
```

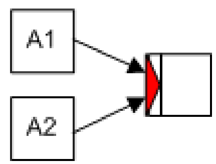
In BPEL

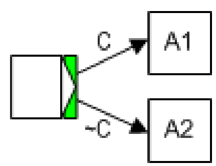
BPEL control patterns support (example)

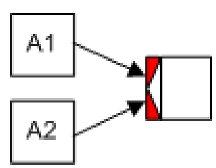
No.	*Pattern*	*YAWL*	*BPEL*

1	Sequence		<sequence>
			  activity A1
			  activity A2
			</sequence>

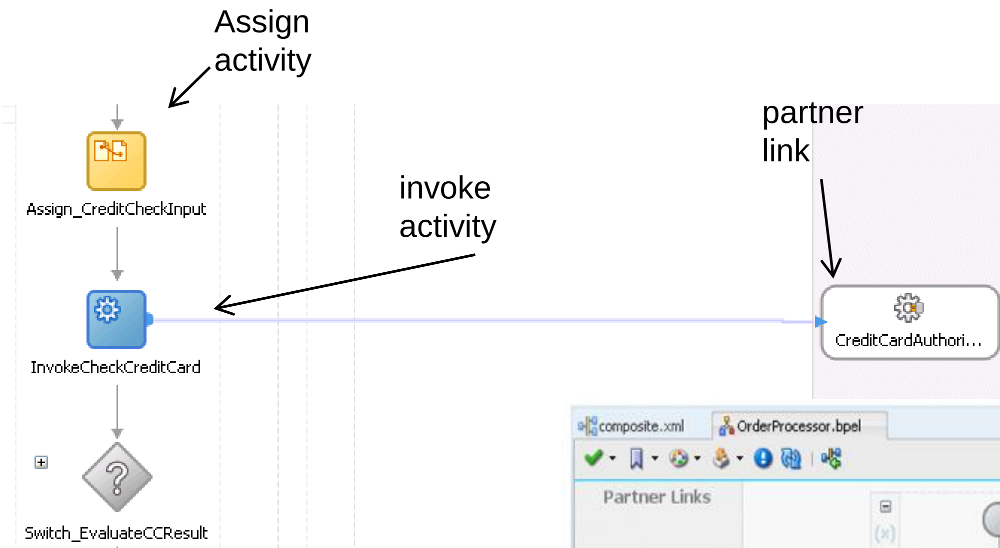
2	Parallel Split		<flow>
			  activity A1
			  activity A2
			</flow>

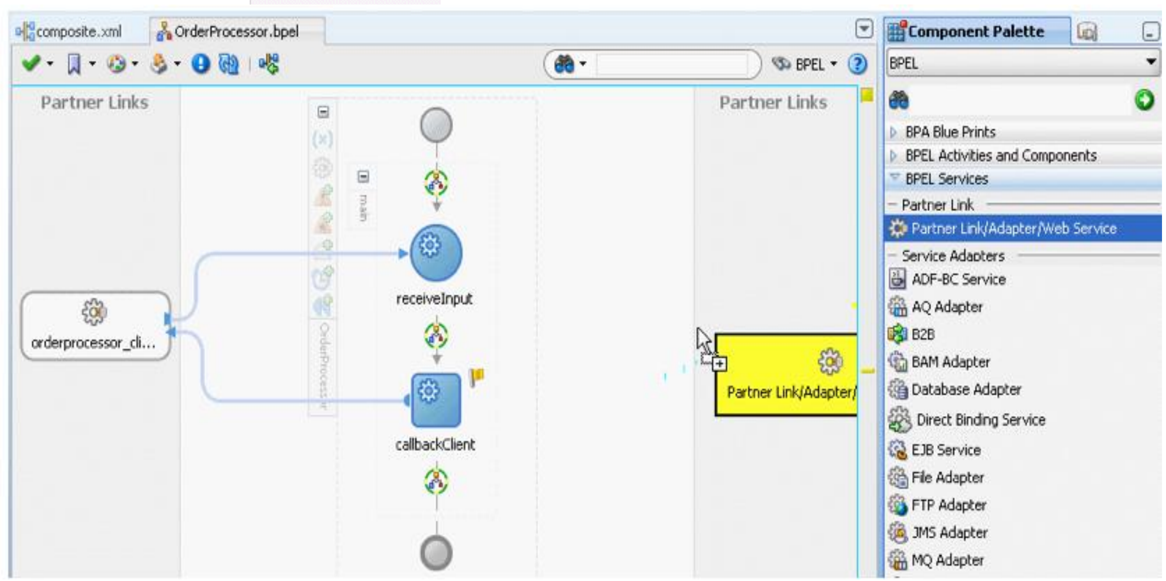
3	Synchronization		

4	Exclusive Choice		<if>
			  <condition>C</condition>
			  activity A1
			  <else>
			    activity A2
			  </else>
			</if>

5	Simple Merge		

BPEL tools



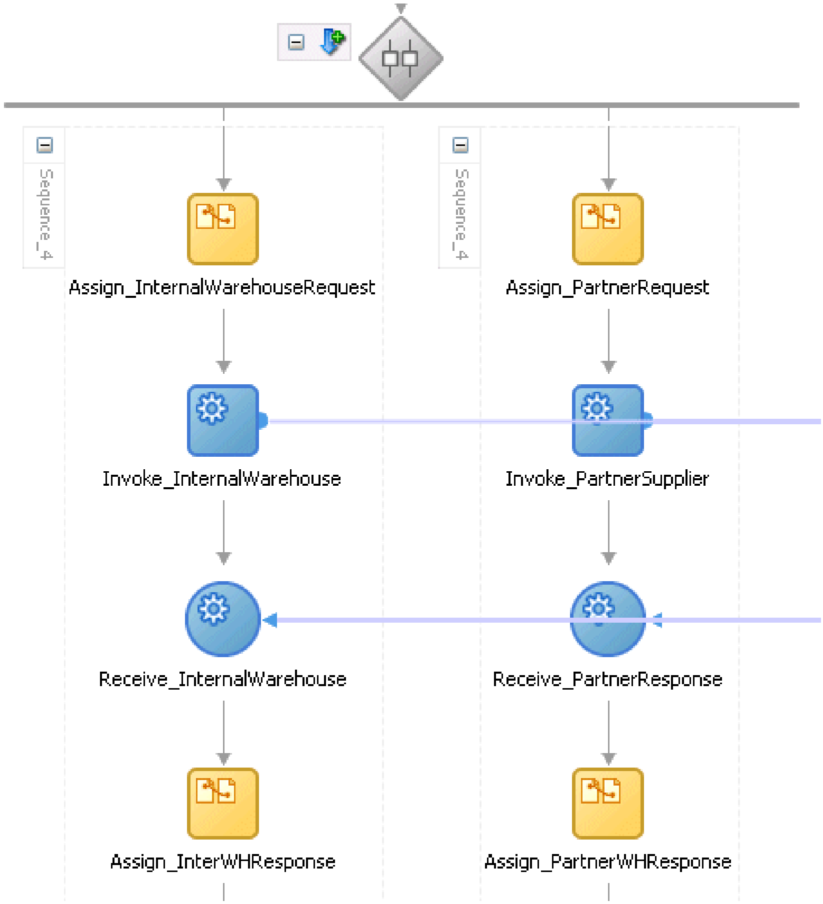


Jdeveloper BPEL designer

http://www.oracle.com/technology/bpel/

BPEL tools

Parallel flows



Pattern support evaluation

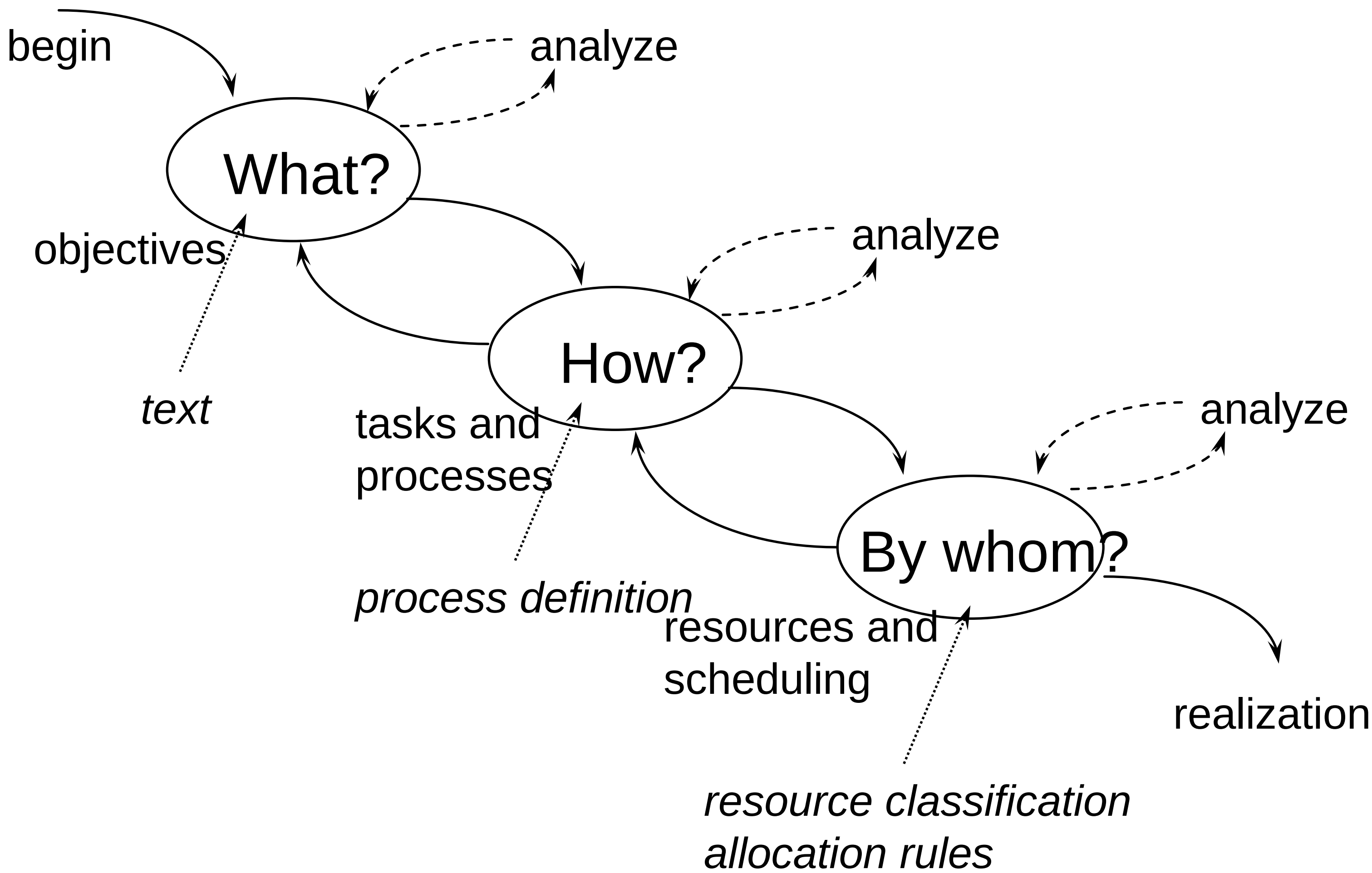
1 – BPMN **2 – UML AD** **3 – BPEL**

		1	2	3			1	2	3
Branching					**Multiple Instances**				
1	Sequence	+	+	+	12	MI without Synchronization	+	+	+
2	Parallel Split	+	+	+	13	MI with a priori Design Time Knlg	+	+	+
6	Multiple Choice	+	+	+	14	MI with a priori Runtime Knlg	+	+	-
4	Exclusive Choice	+	+	+	15	MI without a priori Runtime Knlg	-	-	-
16	Deferred Choice	+	+	+	34	Static Partial Join for MI	+/-	-	-
42	Thread Split	+	+	+/-	35	Cancelling Partial Join for MI	+/-	-	-
Synchronisation					36	Dynamic Partial Join for MI	-	-	-
3	Synchronization	+	+	+	**Concurrency**				
33	Generalised AND-Join	+	-	-	40	Interleaved Routing	+/-	-	+
30	Structured Partial Join	+/-	+/-	-	17	Interleaved Parallel Routing	+/-	-	+/-
31	Blocking Partial Join	+/-	+/-	-	39	Critical Section	-	-	+
32	Cancelling Partial Join	+/-	+	-	18	Milestone	-	-	-
9	Structured Discriminator	+/-	+	-	**Trigger**				
28	Blocking Discriminator	+/-	+/-	-	23	Transient Trigger	-	+	-
29	Cancelling Discriminator	+	+	-	24	Persistent Trigger	+	+	+
7	Str. Synchronizing Merge	+/-	-	+	**Cancellation & Completion**				
37	Local Synchronizing Merge	-	+/-	+	19	Cancel Activity	+	+	+
38	General Synchronizing Merge	-	-	-	20	Cancel Case	+	+	+
5	Simple Merge	+	+	+	25	Cancel Region	+/-	+	-
8	Multiple Merge	+	+	-	26	Cancel MI Activity	+	+	-
41	Thread Merge	+	+	+/-	27	Complete MI Activity	-	-	-
Repetition					**Termination**				
10	Arbitrary Cycles	+	+	-	11	Implicit Termination	+	+	+
21	Structured Loop	+	+	+	43	Explicit Termination	+	+	-
22	Recursion	-	-	-					

Outline

- Motivation
- Introduction
- Petri nets, and Workflow nets
- Using Workflows to model Business Processes
- Workflow Patterns and YAWL
- BPMN and BPEL
- **Design**
- A Database Vision
- Summary

Designing workflows



Designing workflows

Guidelines

- Start with the identification of a case.
  - A case is often initiated by a customer (internal or external).
  - A case has a life-cycle with begin and end.
  - A case cannot be divided, but the work can.
- Determine the scope of the process.
- Determine the goal of a process.
- Ignore the existence of resources during the design of a process.

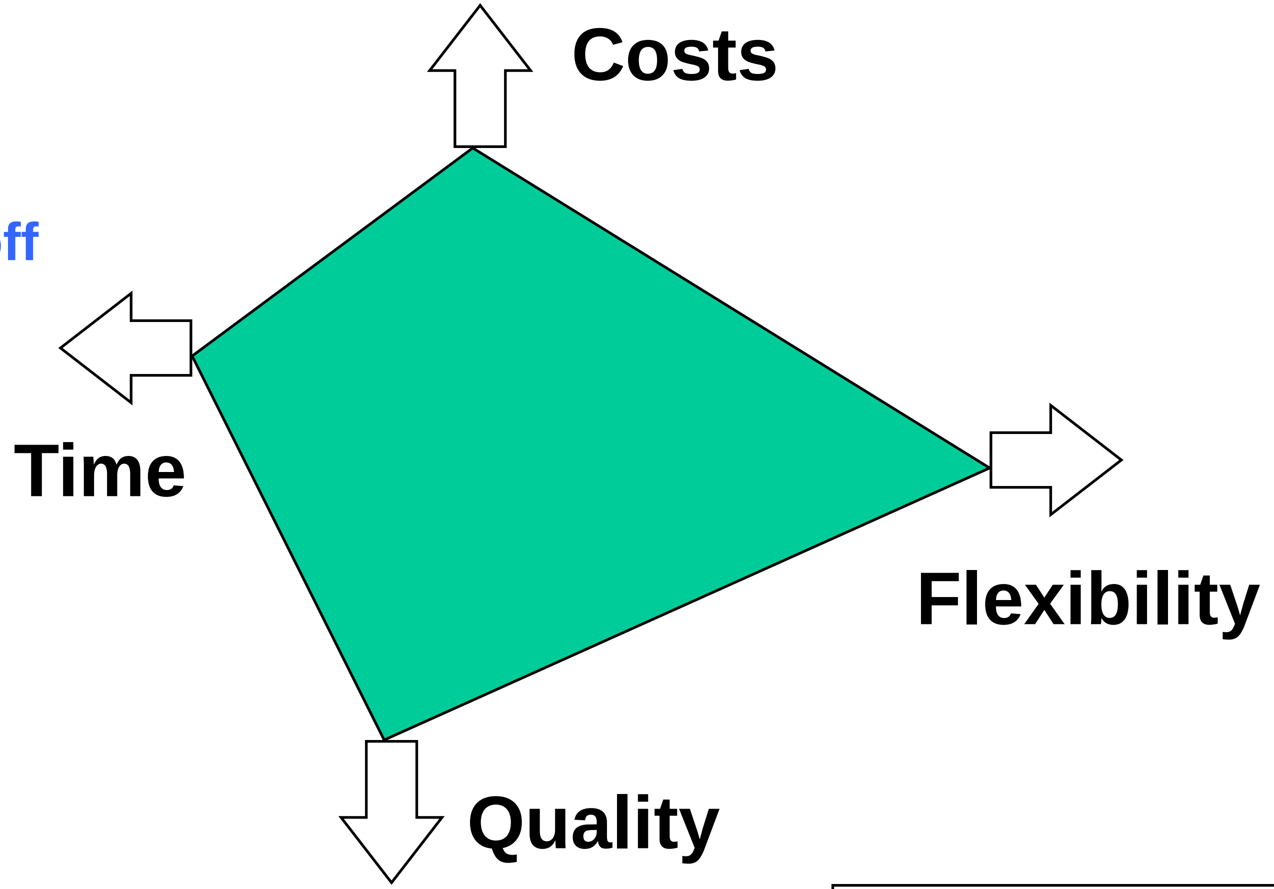
Designing workflows

Guidelines

- Workflow modeling is an iterative process.
  - tasks are split and joined during the process.
  - use hierarchy: divide and conquer.
- During the process a task should become a Logical Unit of Work (LUW).
  - atomic: commit or rollback.
  - a task is executed by the same person, at the same time, at the same place.
  - avoid setup times.
  - avoid large tasks (commit work should be limited).

Design criteria

Trade-off



(T+/-,Q+/-,C+/-,F+/-)

Design criteria

Design criterion 1: Time

- Throughput time is composed of:
  - service time (including set-up)
  - transport time (can often be reduced to 0)
  - waiting time
    - sharing of resources (limited capacity)
    - external communication
- There are several ways to evaluate throughput/waiting time:
  - average
  - variance
  - service level
  - ability to meet due dates

Design criteria

Design criterion 2: Quality

- External: satisfaction of the customer
  - Product: product meets specification/expectation.
  - Process: the way the product is delivered (service level)
- Internal: conditions of work
  - challenging
  - varying
  - controlling

There is often a positive correlation between external and internal quality.

Design criteria

Design criterion 3: Costs

- Type of costs
  - fixed or variable,
  - human, system (hardware/software), or external,
  - processing, management, or support.

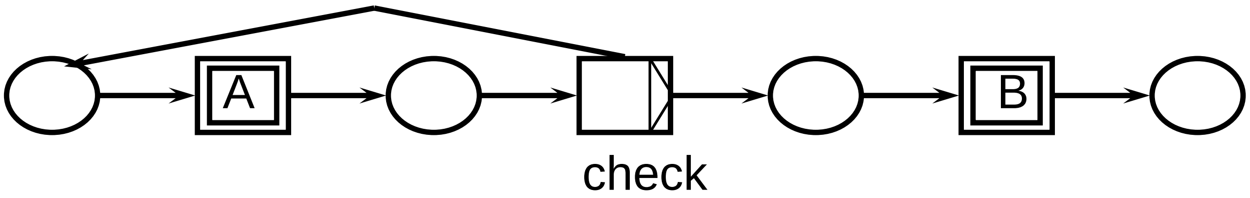
Design criteria

Design criterion 4: Flexibility

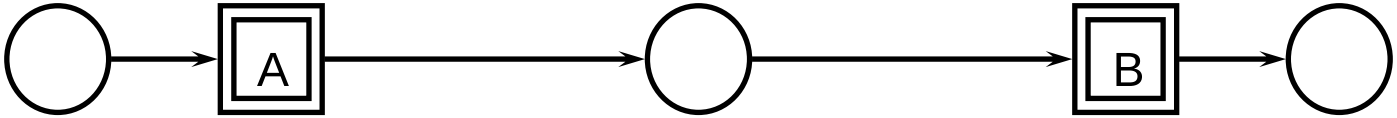
- The ability to react to changes.
- Flexibility of
  - resources (ability to execute many tasks/new tasks)
  - process (ability to handle various cases and changing workloads)
  - management (ability to change rules/allocation)
  - organization (ability to change the structure and responsiveness to wishes of the market and business partners)

Design criteria

(1) Check the necessity of each task



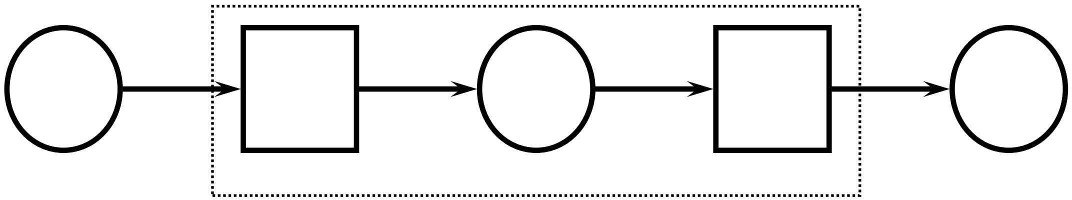
- Trade-off between the costs of the check and the costs of not doing the check.



(T+,Q-,C+/-)

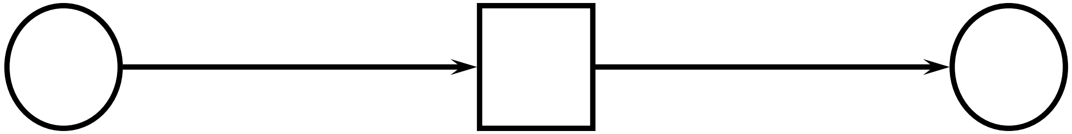
Design criteria

(2) (Re)consider the size of each task



Pros: less work to commit, allows for specialization.
Cons: setup time, fragmentation.

Pros: setup reduction, no fragmentation, more commitment.
Cons: more work to commit, one person needs to be qualified for both parts.



Also a trade-off between the complexity of the process and the complexity of a task.

(T+)

Design criteria

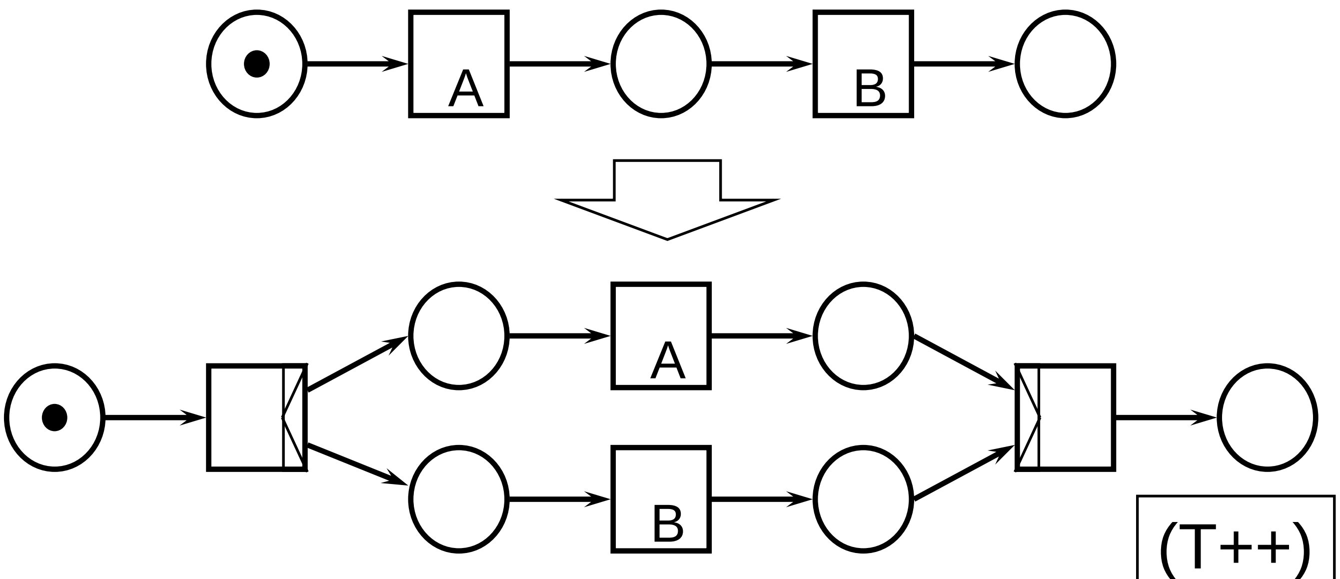
(3) Trade-off: one generic task or multiple specialized tasks

- Specialization may lead to:
  - the possibility to improve the allocation of resources
  - more support when executing the task
  - less flexibility
  - a more complex process
  - monotonicity

Design criteria

(4) Introduce as much parallelism as possible

- More parallelism leads to improved performance: reduction of waiting times and better use of capacity.
- IT infrastructures which allow for the sharing of data and work enable parallelism.



Outline

- Motivation
- Introduction
- Petri nets, and Workflow nets
- Using Workflows to model Business Processes
- Workflow Patterns and YAWL
- BPMN and BPEL
- Design
- **A Database Vision**
- Summary

BP: a database vision

- Success of DBMS due to:
  - Elegant relational model.
  - Declarative query languages.
  - Optimization techniques.
  - Efficient implementations.

- The same is still needed in BPs.
  - Models focus on flow OR data, but not on both.
  - From a DB point of view, research challenge is marrying ideas from database management with ideas of workflow and process flow management.
  - This has not yet been achieved.

BP: a database vision (cont.)

- Data in databases are of course, of interest to a company (e.g., the inventory).
- Suppose a company sales goods online.
    - An online order triggers a process.
    - The process queries the database.
    - If item not available, process does something:
        - issue a production order.
        - recommend similar products.
        - handle user's response to recommendation.
        - etc…

    - Not only inventory data are important:
        - process specification and possible execution flows (e.g., to optimize processes).
        - execution traces of finished processes (event log) (e.g., to analyze users' responses).

BP: a database vision (cont.)

- Kids of data:

    - Databases.
    - Process specification.
    - Event logs.

- Two worlds working separately so far:

    - BP world. (focus on flow).
    - Database world. (focus on data).

- However, BPs include data *and* logic (flow).
- A solution for explicit model and analysis of flow and data is still missing.

BP: a database vision (cont.)

- Needs for:
  - Modeling processes.
  - Generating instances for a given BP.
  - Analyzing process specifications and executions
    - future executions
    - past executions (logs)

BP: a database vision

- Modeling

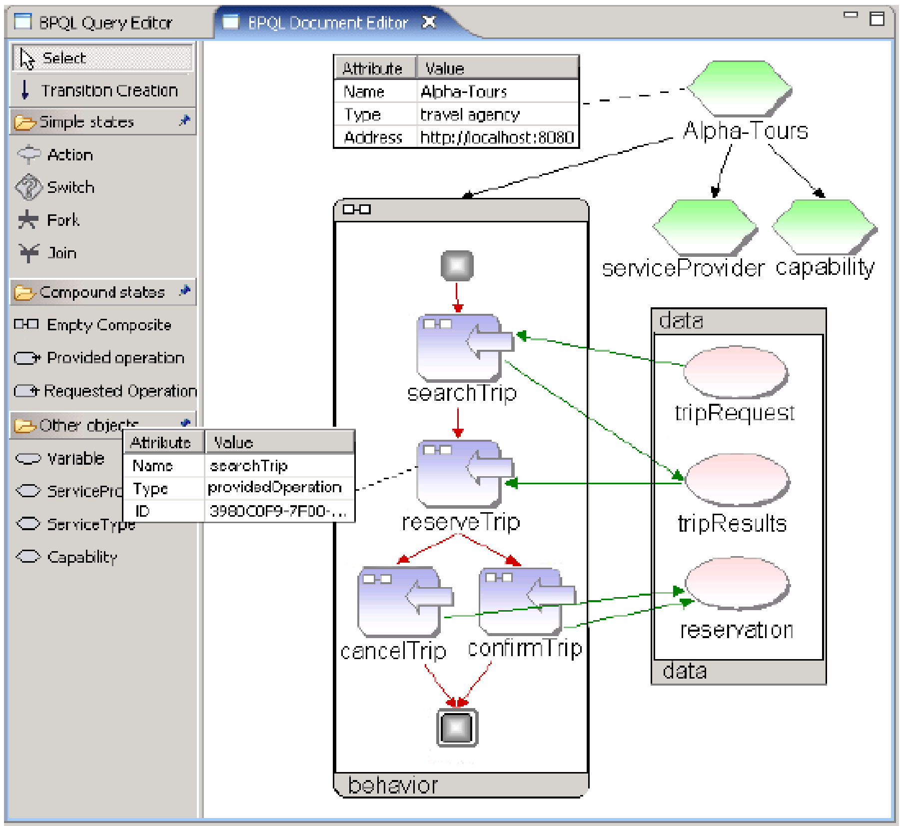
  - Models studied so far in this tutorial lack of explicit and clear treatment of data. Only consider data as something that goes together with the flow.

  - From the DB side, Beeri et al. (VLDB'06, VLDB'07) proposed a process model that is a DAG, whose nodes are business activities and edges reflect their ordering.
    - This model only partially considers data, e.g., do not support SQL queries on the underlying database.

- Open challenge: an integrated, expressive and intuitive model for flow and underlying data.

BP: a database vision

- Modeling (Beeri et al., VLDB ('06,'07)



- The model is an abstraction of BPEL
- For any activity, there are many possible implementations, chosen at runtime, represented as *guarding formulas*.
- However, these formulas do not support SQL queries, only data variables.

BP: a database vision

- Modeling (data)
  - Other line of models focus on data.
  - Some are even older than the term Business Process.
  - Ex. : Relational Transducers (Abiteboul et al., PODS'98)

$$
\begin{aligned}
Ord(user, prod) &\quad +:- \quad InOrd(user, prod) \\
PayedOrd(user, prod) &\quad +:- \quad InPayment(user, prod) \\
UnpaidOrd(user, prod) &\quad +:- \quad Ord(user, prod) \texttt{ AND} \\
&\quad\quad\quad\quad \texttt{NOT } PayedOrd(user, prod) \\
OutReceipt(user, prod) &\quad +:- \quad PayedOrd(user, prod) \texttt{ AND} \\
&\quad\quad\quad\quad InStock(prod)
\end{aligned}
$$

Relation names starting with "In": input relations; with "Out": output relations; other: state relations.

  - No distinction between flow and underlying data. The database stores it all. A datalog-like program is used to query and update the state, input and output relations.
  - Semantics: inflationary datalog. Satisfying RHS leads to adding tuples in the relation of the LHS.
  - **Underlying flow not easily detected from process specification.**

BP: a database vision

- Generation of a model instance.

- Once the model is chosen, for a given BP an instance of the model is created.
- Two ways of doing this:
    - Manually (as seen before).
    - Automatically: mining event logs. Aimed at obtaining a process instance from an event log. (to be discussed later).

BP: a database vision

- Querying and analyzing BP data.
    - Two kinds of analysis:
        - Possible future execution of BPs.
        - Querying logs of past executions.
            - To detect problems occurred at runtime.
            - Trends of process usage….
        - Therefore, as database people….

BP: a database vision

- Querying and analyzing BP data.

  - …we need a query language
    - Declarative.
    - Intuitive
    - Graphical (?)

  - Must allow
    - analysis of process and data (past and future). Only care for a single language instead of one language for past and another for future executions.
    - querying flow and data.
    - querying at different levels of granularity.
    - specifying boolean verification queries.
    - efficient query evaluation and optimization.

BP: a database vision

- Querying and analyzing BP data.
  - **Querying the future**
    - We want to test properties of the flow. Ex.:
      - “a user cannot place an order without giving CC details”.
      - “a user must have a positive balance in the account to place an order”.
      - “what is the probability that the user choses to order a recommended item if the one she ordered originally is not available?”
    - Research on this, based on temporal logic.
      - Use temporal quantifiers: A (after), B (before).
        - *Login* B *Order* expresses: “a user must login before placing an order”.
      - Problems:
        - Not easy to read.
        - Flow not explicitly stated.
        - Cannot retrieve paths of interest.

BP: a database vision

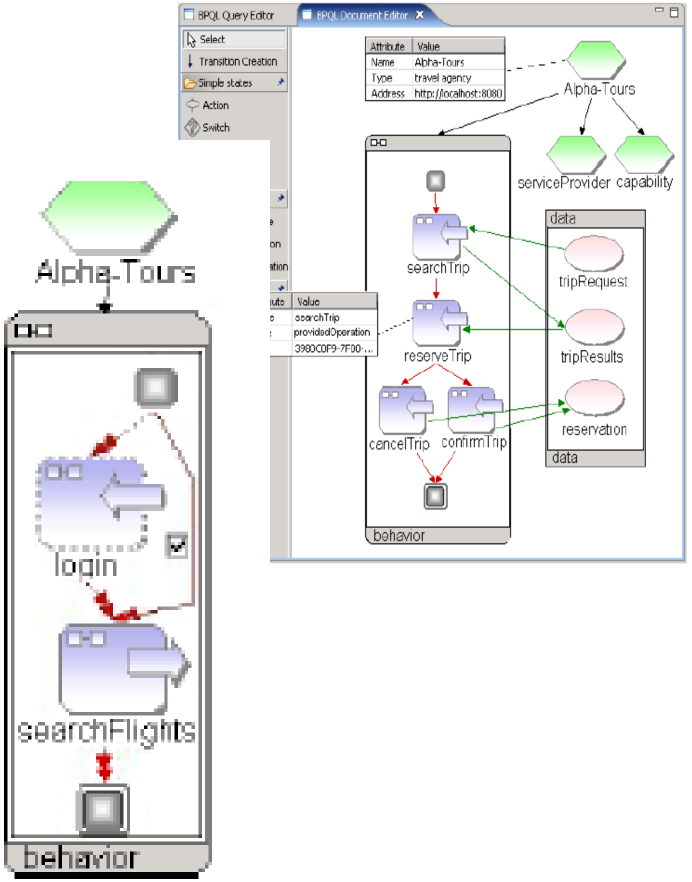
- Querying and analyzing BP data.
  - **Querying the future**
    - Deutch et al. (SIGMOD ‘05) proposed LTL-FO (linear temporal logic-first order)
      - Temporal logic used to query the flow, FO constructs used inside the predicates to query flow and database state at a point of the execution.
      - Ex: “when a user orders any product, account balance must be >0”.
        - ∇*user,product.A* (*Order*(*user,product*) => Ξ *sum* >0.*balance* (*user,sum*)).

BP: a database vision

- Querying and analyzing BP data.
  - **Querying the future**
    - Beeri et al. (VLDB ’06; 07): BP-QL.
    - Graphical language, different levels of granularity.
    - Uses BP patterns for querying (in some sense, similar idea that YAWL uses for flow modeling).

—

Query: “give me paths starting in a login Activity followed (directly or after some other activities), by a searchFlight activity”.



BP: a database vision

- Querying and analyzing BP data.
  - **Querying past executions**
    - Patterns of user behavior can be obtained from execution logs.
    - Challenge: normally, the event log stores the activities. We could also be interested in the *data that was manipulated.*
    - Logs can be viewed as graphs of the execution flow => lots of work on graph query languages, e.g., G (Cruz et al., SIGMOD'87), Graphlog (Consens & Mendelzon, SIGMOD'90). (BP-QL based on these).
    - None of them combines flow and data appropriately.
    - Sometimes, queries are not clear….then, data mining is a choice…….

Outline

- Motivation
- Introduction
- Petri nets and Workflow Nets
- Using Workflows to model Business Processes
- Workflow Patterns and YAWL
- BPMN and BPEL
- Design
- A Database Vision
- **Summary**

Summary

- Overview of BP modeling based on flow.
- Workflow nets, YAWL, BPMN 2.0 standard.
- Execution of BPs, e.g. BPEL.
- Data only very partially considered.

- In the second part, we will consider data, and integrate BPM and BI.

END OF PART 1

An Introduction to Business Process Modeling – Part 2

Alejandro Vaisman
Department of Computer & Decision Engineering (CoDE)
Université Libre de Bruxelles
{avaisman,ezimanyi}@ulb.ac.be

Outline

- Motivation. Process mining basics.
- Getting event data.
- Process discovery.
- Conformance checking.
- Online Process Mining.
- Tools.
- Conclusion.

Outline

- **Introduction: Process mining basics.**
- Getting event data.
- Process discovery.
- Conformance checking.
- Online Process Mining.
- Tools.
- Conclusion.

Process mining task force

- Process mining: one of the most important innovations in the field of BPM.
- PM joins ideas of process modeling and analysis, with data mining and machine learning.
- IEEE has established a **Task Force on Process Mining**.
- The goal of this Task Force: promote research, development, education and understanding of PM, by means of:
  - making end-users, developers, consultants, and researchers aware of the state-of-the-art in process mining,
  - promoting the use of PM techniques and tools.
  - standardizing efforts for logging event data.
  - organizing tutorials, special sessions, workshops, panels.
  - organizing conferences/workshop with IEEE CIS Technical Co-Sponsorship.

PM manifesto

- Defines PM characteristics, guidelines, and challenges.
- Process Mining characteristics.
  - *1. Not limited to control-flow discovery*. Control-flow discovery is often seen as the most relevant part of PM, but it is just one of the three basic forms of process mining (discovery, conformance, and enhancement).
  - *2. Not just a specific type of data mining*. Sort of "missing link" between data mining and traditional model-driven BPM. Most data mining techniques are not process-centric at all. New types of representations and algorithms are needed.
  - *3. Not limited to offline analysis.* Process mining techniques extract knowledge from historical event data. Although "post mortem" (historic) data are used, the results can be applied to currently running cases. For example, the completion time of a partially handled customer order can be predicted using a discovered process model.

PM manifesto: guidelines

Guiding Principles:

GP1: Event Data Should be Treated as First-Class Citizens

GP2: Log Extraction Should Be Driven by Questions

GP3: Concurrency, Choice and Other Basic Control-Flow Constructs Should be Supported

GP4: Events Should Be Related to Model Elements

GP5: Models Should Be Treated as Purposeful Abstractions of Reality

GP6: Process Mining Should Be a Continuous Process

PM manifesto: challenges

Challenges:

C1: Finding, Merging, and Cleaning Event Data

C2: Dealing with Complex Event Logs Having Diverse Characteristics

C3: Creating Representative Benchmarks

C4: Dealing with Concept Drift

C5: Improving the Representational Bias Used for Process Discovery

C6: Balancing Between Quality Criteria such as Fitness, Simplicity, Precision, and Generalization

C7: Cross-Organizational Mining

C8: Providing Operational Support

C9: Combining Process Mining With Other Types of Analysis

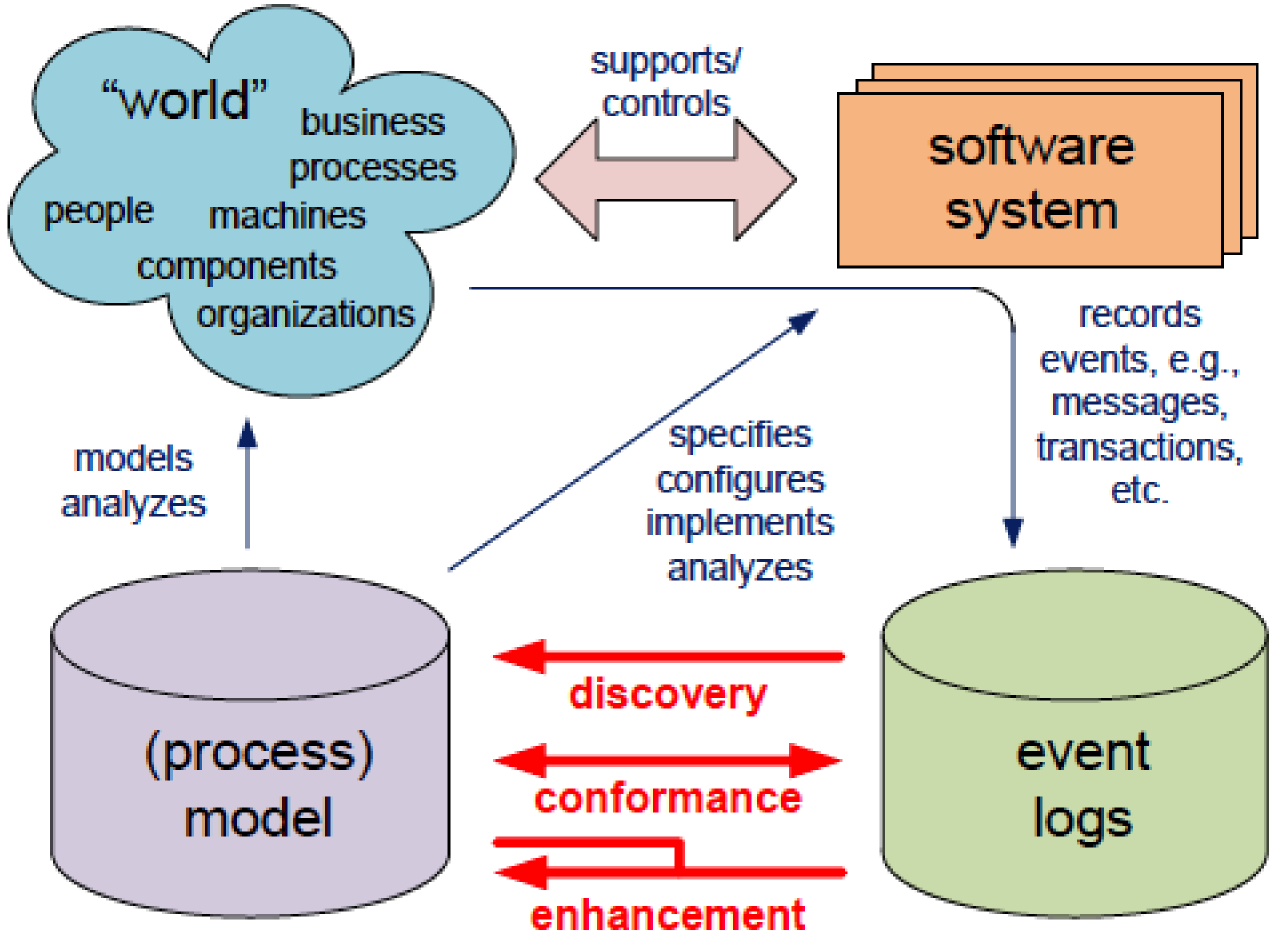
C10: Improving Usability for Non-Experts

C11: Improving Understandability for Non-Experts

Goals of process mining

- What really happened in the past?
- Why did it happen?
- What is likely to happen in the future?
- When and why do organizations and people deviate?
- How to control a process better?
- How to redesign a process to improve its performance?

Process mining



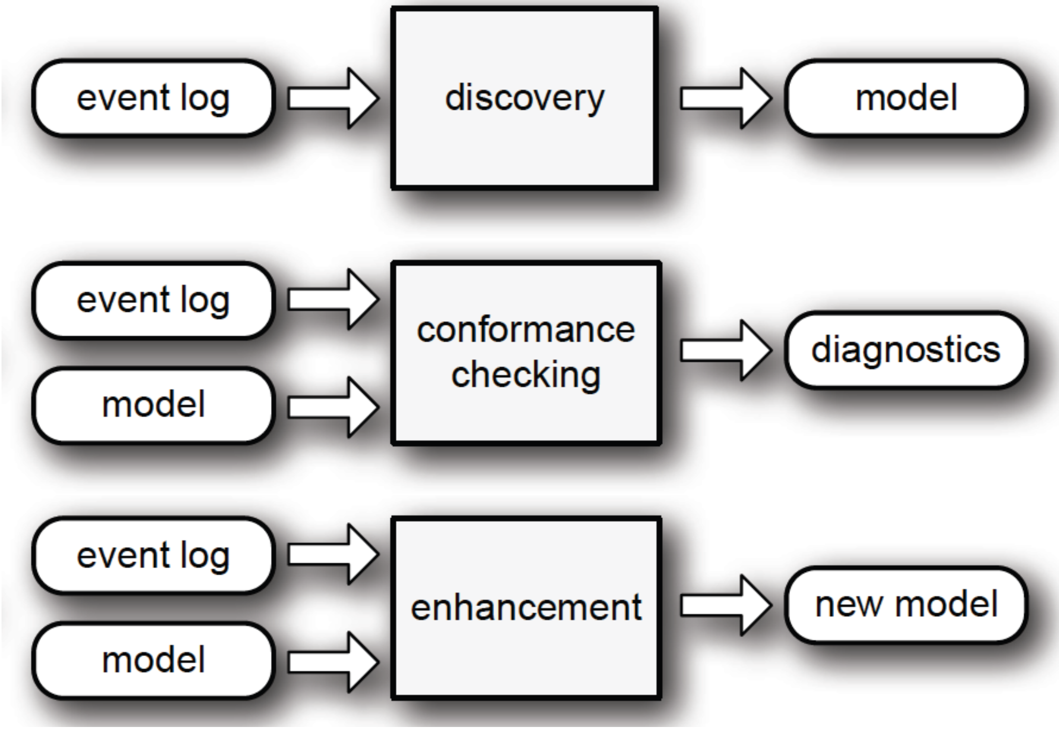
Process mining: three types

- Discovery: takes an event log and produces a model without any apriori information. Example: the α algorithm takes a log and produces a Petri net explaining the behavior indicated by the log.

- Conformance: an existing model is compared with an event log of the same process. Conformance checking verifies that reality as recorded in the log, conforms to the model (and viceversa).

- Enhancement: aimed at extending or improving an existing process model using information about the actual process, recorded in the log.

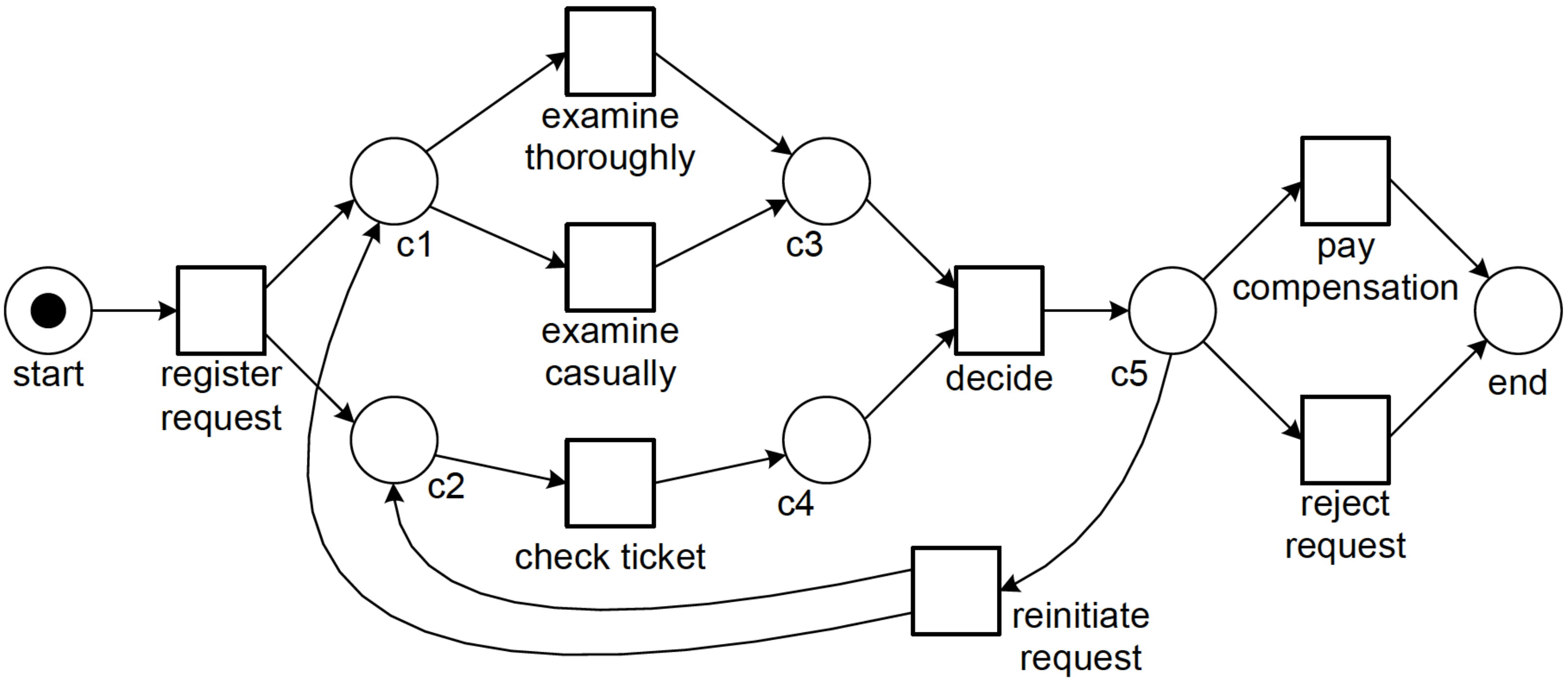
Process mining: three types



Process mining: perspectives

- **Orthogonal to the PM types:**
  - The control-flow perspective focuses on the control-flow, i.e., the ordering of activities. Mining this perspective tries to find a characterization of all possible paths.
  - The organizational perspective focuses on information about resources hidden in the log, i.e., which actors (e.g., people, systems, departments) are involved and how are they related. Goal: structuring the organization e.g., by classifying people.
  - The case perspective focuses on properties of cases, e.g., cases can also be characterized by the values of the corresponding data elements. For example, if a case represents a replenishment order.
  - The time perspective is concerned with the timing and frequency of events (e.g., to predict remaining process time).

Example. Compensation claim



Example. An event log

case id	event id	properties				
		timestamp	activity	resource	cost	...
1	35654423	30-12-2010:11.02	register request	Pete	50	...
	35654424	31-12-2010:10.06	examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	check ticket	Mike	100	...
	35654426	06-01-2011:11.18	decide	Sara	200	...
	35654427	07-01-2011:14.24	reject request	Pete	200	...
2	35654483	30-12-2010:11.32	register request	Mike	50	...
	35654485	30-12-2010:12.12	check ticket			
	35654487	30-12-2010:14.16	examine casually			
	35654488	05-01-2011:11.22	decide			
	35654489	08-01-2011:12.05	pay compensation			
3	35654521	30-12-2010:14.32	register request			
	35654522	30-12-2010:15.06	examine casually			
	35654524	30-12-2010:16.34	check ticket			
	35654525	06-01-2011:09.18	decide			
	35654526	06-01-2011:12.18	reinitiate request			
	35654527	06-01-2011:13.06	examine thoroughly			
	35654530	08-01-2011:11.43	check ticket			
	35654531	09-01-2011:09.55	decide			
	35654533	15-01-2011:10.45	pay compensation			
4	35654641	06-01-2011:15.02	register request			
	35654643	07-01-2011:12.06	check ticket			
	35654644	08-01-2011:14.43	examine thoroughly			
	35654645	09-01-2011:12.02	decide			
	35654647	12-01-2011:15.44	reject request			
5	35654711	06-01-2011:09.02	register request			
	35654712	07-01-2011:10.16	examine casually			
	35654714	08-01-2011:11.22	check ticket			
	35654715	10-01-2011:13.28	decide			
	35654716	11-01-2011:16.18	reinitiate request			
	35654718	14-01-2011:14.33	check ticket			
	35654719	16-01-2011:15.50	examine casually			
	35654720	19-01-2011:11.18	decide	Sara	200	...
	35654721	20-01-2011:12.48	reinitiate request	Sara	200	...

case id	event id	properties				
		timestamp	activity	resource	cost	...
1	35654423	30-12-2010:11.02	register request	Pete	50	...
	35654424	31-12-2010:10.06	examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	check ticket	Mike	100	...
	35654426	06-01-2011:11.18	decide	Sara	200	...
	35654427	07-01-2011:14.24	reject request	Pete	200	...
2	35654483	30-12-2010:11.32	register request	Mike	50	...
	35654485	30-12-2010:12.12	check ticket	Mike	100	...
	35654487	30-12-2010:14.16	examine casually	Pete	400	...
	35654488	05-01-2011:11.22	decide	Sara	200	...
	35654489	08-01-2011:12.05	pay compensation	Ellen	200	...

Requirements for a log

- Minimal requirements for an event log:
  - Event can be related to both, a case and an activity. Ex: case Id and Activity in the example log. Cost and resource are attributes.
  - Events within a case are ordered.
  - Sequence of activities: *trace.*
  - A compact representation can be obtained. (See next slide).

A simplified event log

case id	event id	properties				
		timestamp	activity	resource		
1	35654423	30-12-2010:11.02	register request	Pete		
	35654424	31-12-2010:10.06	examine thoroughly	Sue		
	35654425	05-01-2011:15.12	check ticket	Mike		
	35654426	06-01-2011:11.18	decide	Sara		
	35654427	07-01-2011:14.24	reject request	Pete		
2	35654483	30-12-2010:11.32	register request	Mike		
	35654485	30-12-2010:12.12	check ticket	Mike		
	35654487	30-12-2010:14.16	examine casually	Pete		
	35654488	05-01-2011:11.22	decide	Sara		
	35654489	08-01-2011:12.05	pay compensation	Ellen		
3	35654521	30-12-2010:14.32	register request	Pete		
	35654522	30-12-2010:15.06	examine casually	Mike		
	35654524	30-12-2010:16.34	check ticket	Ellen		
	35654525	06-01-2011:09.18	decide	Sara		
	35654526	06-01-2011:12.18	reinitiate request	Sara		
	35654527	06-01-2011:13.06	examine thoroughly	Sean		
	35654530	08-01-2011:11.43	check ticket	Pete		
	35654531	09-01-2011:09.55	decide	Sara		
	35654533	15-01-2011:10.45	pay compensation	Ellen		
4	35654641	06-01-2011:15.02	register request	Pete	50	...
	35654643	07-01-2011:12.06	check ticket	Mike	100	...
	35654644	08-01-2011:14.43	examine thoroughly	Sean	400	...
	35654645	09-01-2011:12.02	decide	Sara	200	...
	35654647	12-01-2011:15.44	reject request	Ellen	200	...
5	35654711	06-01-2011:09.02	register request	Ellen	50	...
	35654712	07-01-2011:10.16	examine casually	Mike	400	...
	35654714	08-01-2011:11.22	check ticket	Pete	100	...
	35654715	10-01-2011:13.28	decide	Sara	200	...
	35654716	11-01-2011:16.18	reinitiate request	Sara	200	...
	35654718	14-01-2011:14.33	check ticket	Ellen	100	...
	35654719	16-01-2011:15.50	examine casually	Mike	400	...
	35654720	19-01-2011:11.18	decide	Sara	200	...
	35654721	20-01-2011:12.48	reinitiate request	Sara	200	...
	35654722	21-01-2011:09.06	examine casually	Sue	400	...
	35654724	21-01-2011:11.34	check ticket	Pete	100	...
	35654725	23-01-2011:13.12	decide	Sara	200	...
	35654726	24-01-2011:14.56	reject request	Mike	200	...
6	35654871	06-01-2011:15.02	register request	Mike	50	...
	35654873	06-01-2011:16.06	examine casually	Ellen	400	...
	35654874	07-01-2011:16.22	check ticket	Mike	100	...
	35654875	07-01-2011:16.52	decide	Sara	200	...
	35654877	16-01-2011:11.47	pay compensation	Mike	200	

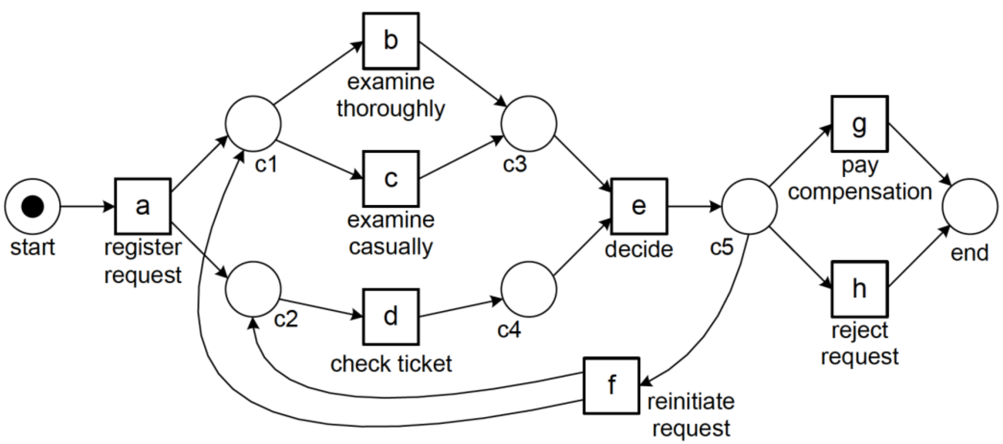
case id	trace
1	$\langle a,b,d,e,h \rangle$
2	$\langle a,d,c,e,g \rangle$
3	$\langle a,c,d,e,f,b,d,e,g \rangle$
4	$\langle a,d,b,e,h \rangle$
5	$\langle a,c,d,e,f,d,c,e,f,c,d,e,h \rangle$
6	$\langle a,c,d,e,g \rangle$
...	...

a = register request,
b = examine thoroughly,
c = examine casually,
d = check ticket,
e = decide,
f = reinitiate request,
g = pay compensation,
and h = reject request

A simplified event log

case id	event id	timestamp	activity	resource		
	35654423	30-12-2010:11.02	register request	Pete		
1	35654424	31-12-2010:10.06	examine thoroughly	Sue		
	35654425	05-01-2011:15.12	check ticket	Mike		
	35654426	06-01-2011:11.18	decide	Sara		
	35654427	07-01-2011:14.24	reject request	Pete		
	35654483	30-12-2010:11.32	register request	Mike		
2	35654485	30-12-2010:12.12	check ticket	Mike		
	35654487	30-12-2010:14.16	examine casually	Pete		
	35654488	05-01-2011:11.22	decide	Sara		
	35654489	08-01-2011:12.05	pay compensation	Ellen		
	35654521	30-12-2010:14.32	register request	Pete		
3	35654522	30-12-2010:15.06	examine casually	Mike		
	35654524	30-12-2010:16.34	check ticket	Ellen		
	35654525	06-01-2011:09.18	decide	Sara		
	35654526	06-01-2011:12.18	reinitiate request	Sara		
	35654527	06-01-2011:13.06	examine thoroughly	Sean		
	35654530	08-01-2011:11.43	check ticket	Pete		
	35654531	09-01-2011:09.55	decide	Sara		
	35654533	15-01-2011:10.45	pay compensation	Ellen		
	35654641	06-01-2011:15.02	register request	Pete	50	...
4	35654643	07-01-2011:12.06	check ticket	Mike	100	...

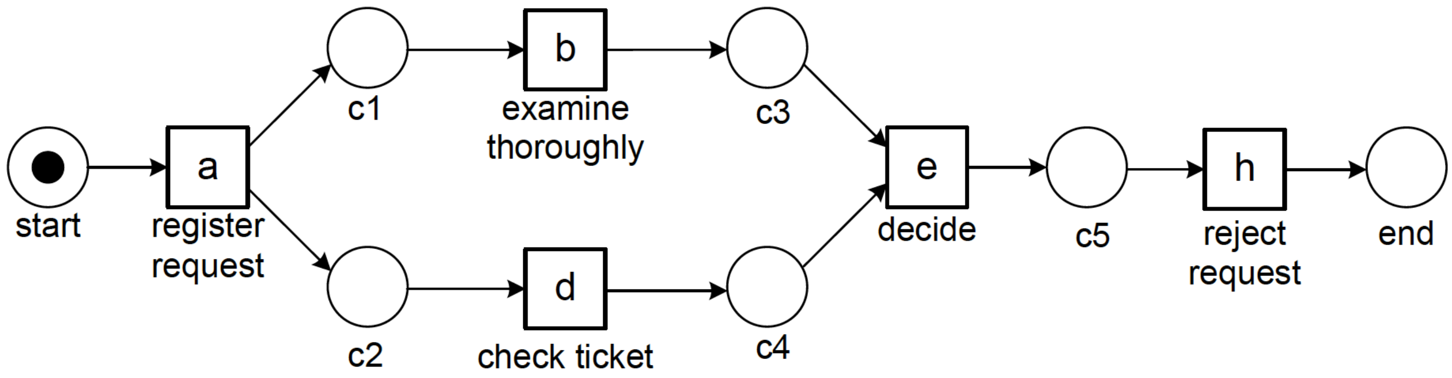
case id	trace
1	$\langle a,b,d,e,h \rangle$
2	$\langle a,d,c,e,g \rangle$
3	$\langle a,c,d,e,f,b,d,e,g \rangle$
4	$\langle a,d,b,e,h \rangle$
5	$\langle a,c,d,e,f,d,c,e,f,c,d,e,h \rangle$
6	$\langle a,c,d,e,g \rangle$
...	...



a = register request,
b = examine thoroughly,
c = examine casually,
d = check ticket,
e = decide,
f = reinitiate request,
g = pay compensation,
and h = reject request

Process mining example

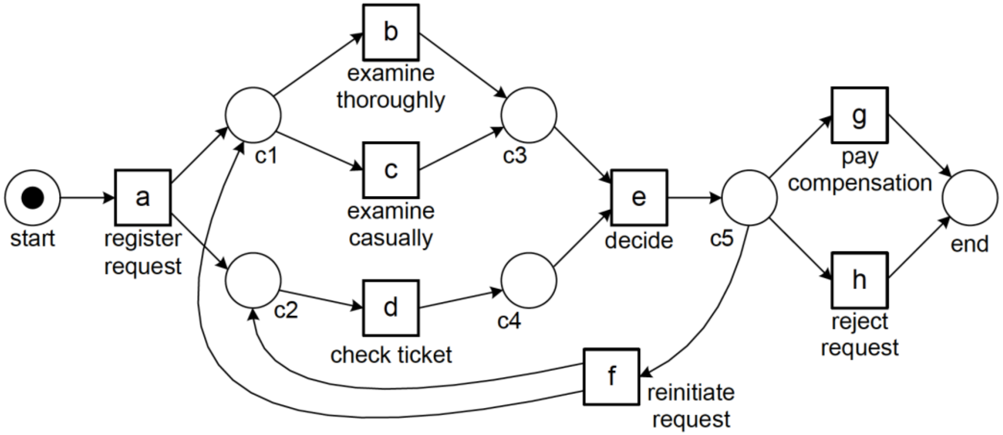
- The α algorithm based on cases 1 to 6 returns the graph in the previous slide. Risk of *overfitting* (model too specific), or conversely, *underfitting* (model too general).
- If we only give as input cases 1 and 4 (<*a,b,d,e,h*>,<*a,d,b,e,h*>) we obtain (using the α algorithm):



Checking conformance

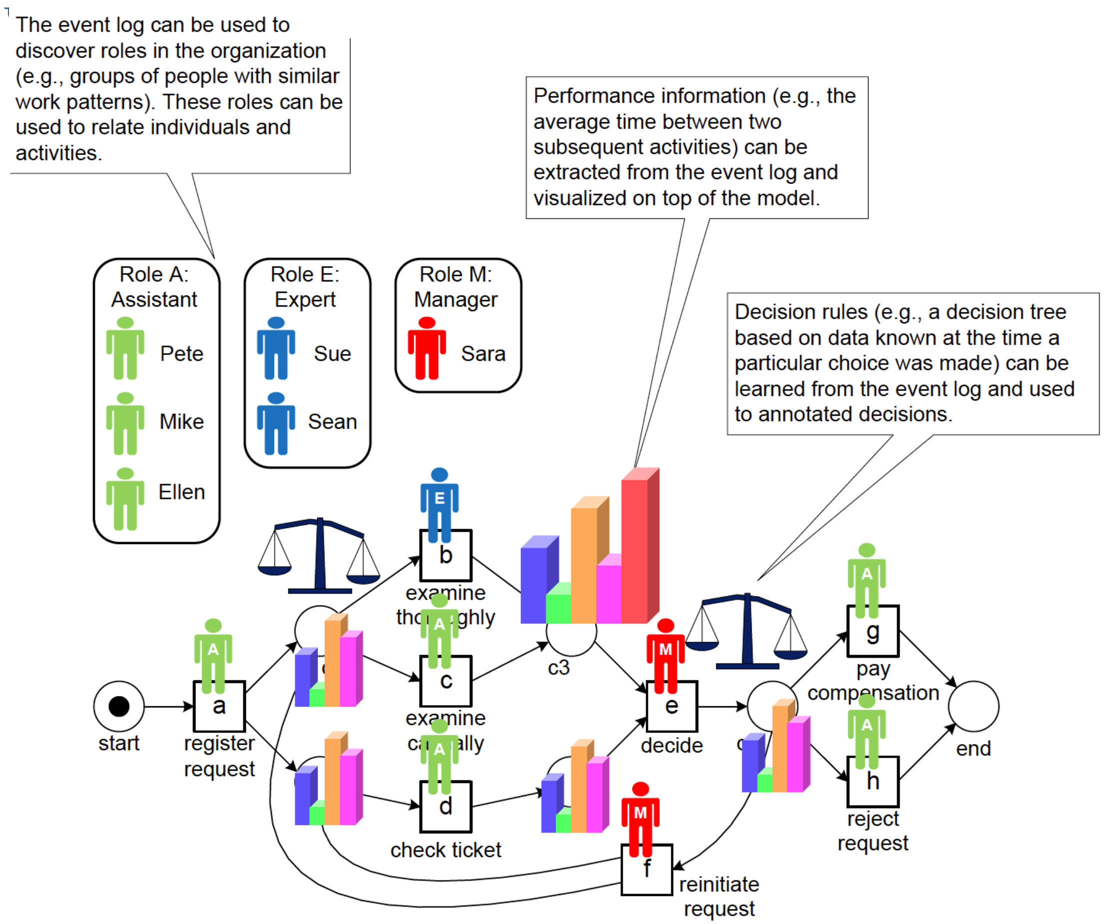
Cases 7,8, and 10 are not possible in the model below. Case 7 needs a *d,* case 8 does *not end in h or g.* In case 10, compensation was paid without a decision (*e*).

case id	trace
1	$\langle a,b,d,e,h \rangle$
2	$\langle a,d,c,e,g \rangle$
3	$\langle a,c,d,e,f,b,d,e,g \rangle$
4	$\langle a,d,b,e,h \rangle$
5	$\langle a,c,d,e,f,d,c,e,f,c,d,e,h \rangle$
6	$\langle a,c,d,e,g \rangle$
7	$\langle \mathbf{a,b,e,g} \rangle$
8	$\langle \mathbf{a,b,d,e} \rangle$
9	$\langle a,d,c,e,f,d,c,e,f,b,d,e,h \rangle$
10	$\langle \mathbf{a,c,d,e,f,b,d,g} \rangle$



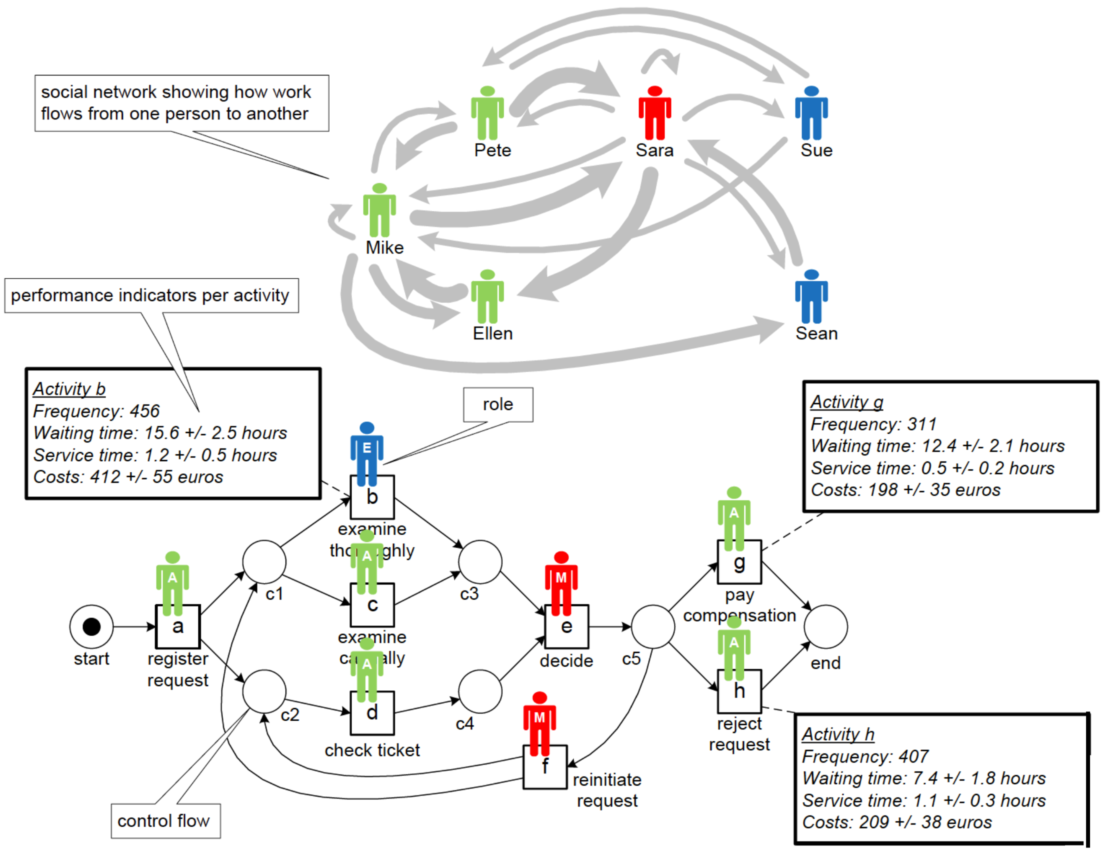
Other perspectives

- Attributes allow to extract other information.



Other perspectives

- Attributes allow to extract other information.



Outline

- Motivation. Process mining basics.
- **Getting event data.**
- Process discovery.
- Conformance checking.
- Online Process Mining.
- Tools.
- Conclusion.

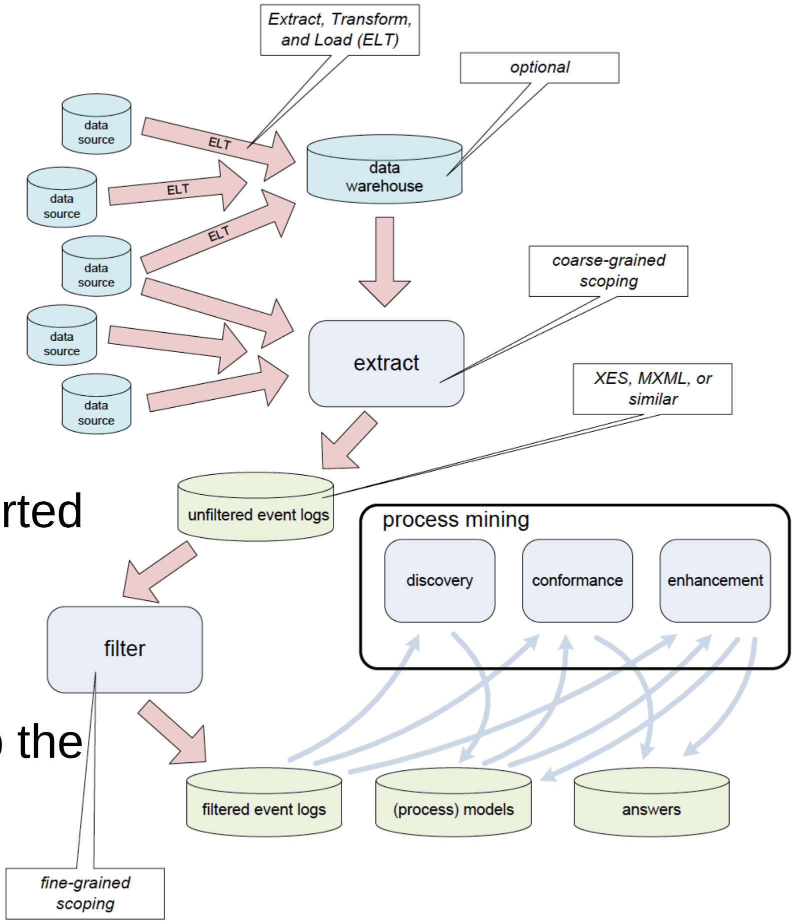
Getting the data

Overall Process Mining Workflow.

Even if a DW exists, it is most likely not process-oriented, but OLAP-oriented.

Data is extracted and converted an event log (formats: XES, MXML).

Data is filtered, e.g., to keep the most relevant activities (association rules can be used)

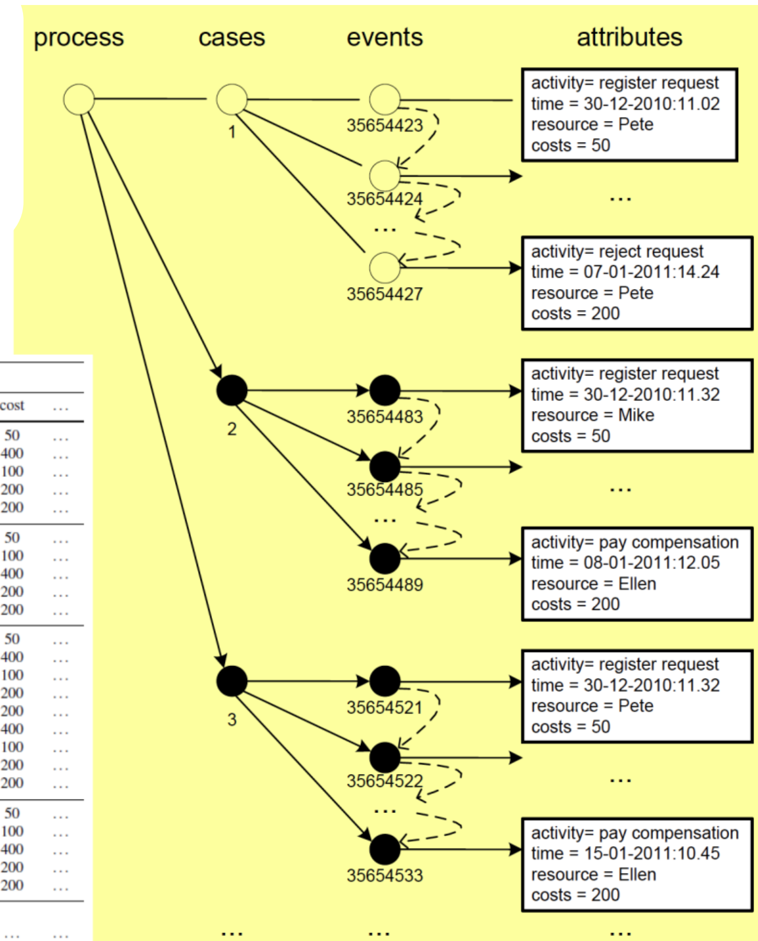


Getting the data

- A process:
  - consists of cases.
  - A case consists of events
  - Each event relates to precisely one case
  - Events within a case are ordered.
  - Events can have attributes.
  - Typical attribute names: activity, time, costs, and resource.

An event log

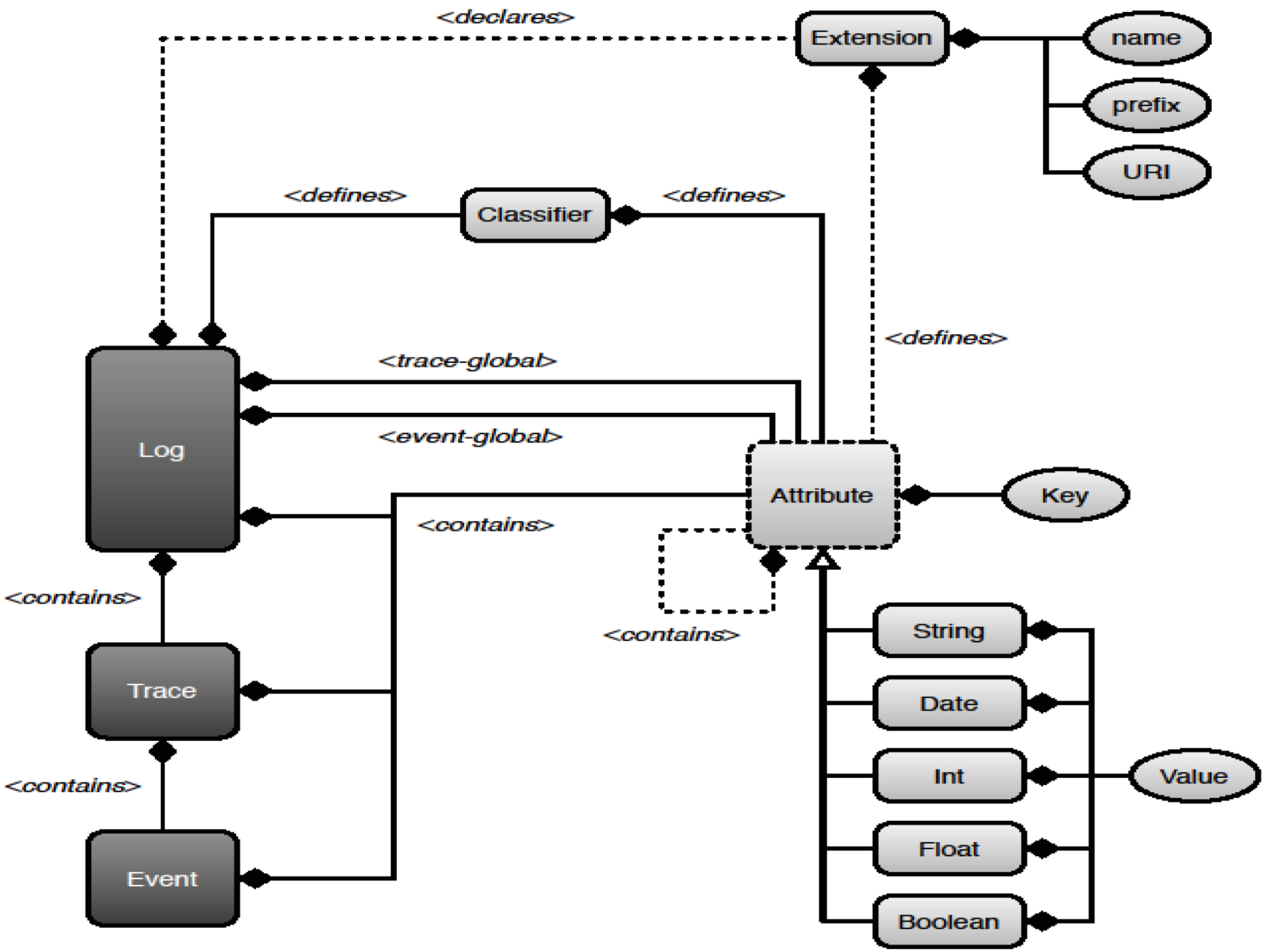
case id	event id	properties				
		timestamp	activity	resource	cost	...
1	35654423	30-12-2010:11.02	register request	Pete	50	...
	35654424	31-12-2010:10.06	examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	check ticket	Mike	100	...
	35654426	06-01-2011:11.18	decide	Sara	200	...
	35654427	07-01-2011:14.24	reject request	Pete	200	...
2	35654483	30-12-2010:11.32	register request	Mike	50	...
	35654485	30-12-2010:12.12	check ticket	Mike	100	...
	35654487	30-12-2010:14.16	examine casually	Pete	400	...
	35654488	05-01-2011:11.22	decide	Sara	200	...
	35654489	08-01-2011:12.05	pay compensation	Ellen	200	...
3	35654521	30-12-2010:14.32	register request	Pete	50	...
	35654522	30-12-2010:15.06	examine casually	Mike	400	...
	35654524	30-12-2010:16.34	check ticket	Ellen	100	...
	35654525	06-01-2011:09.18	decide	Sara	200	...
	35654526	06-01-2011:12.18	reinitiate request	Sara	200	...
	35654527	06-01-2011:13.06	examine thoroughly	Sean	400	...
	35654530	08-01-2011:11.43	check ticket	Pete	100	...
	35654531	09-01-2011:09.55	decide	Sara	200	...
	35654533	15-01-2011:10.45	pay compensation	Ellen	200	...
4	35654641	06-01-2011:15.02	register request	Pete	50	...
	35654643	07-01-2011:12.06	check ticket	Mike	100	...
	35654644	08-01-2011:14.43	examine thoroughly	Sean	400	...
	35654645	09-01-2011:12.02	decide	Sara	200	...
	35654647	12-01-2011:15.44	reject request	Ellen	200	...
...	...	...	...	...	...	...



Standards for event logs: XES

- See www.xes-standard.org.
- Adopted by the IEEE Task Force on Process Mining.
- Predecessor: MXML and SA-MXML.
- The format is supported by tools such as ProM (as of version 6), Nitro, XESame, and OpenXES.
- ProM import supports MXML.

XES metamodel

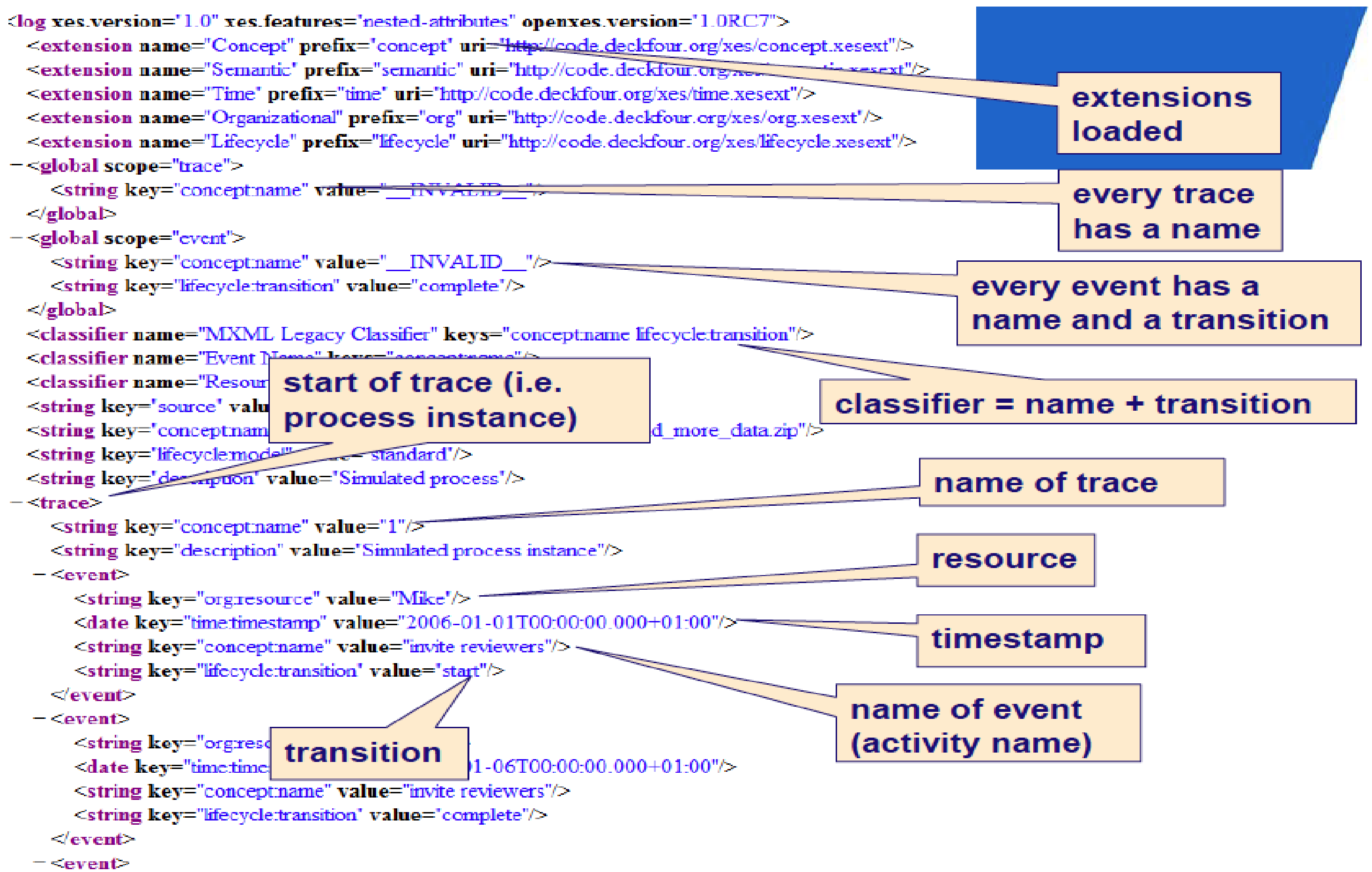


Standards for event logs: XES

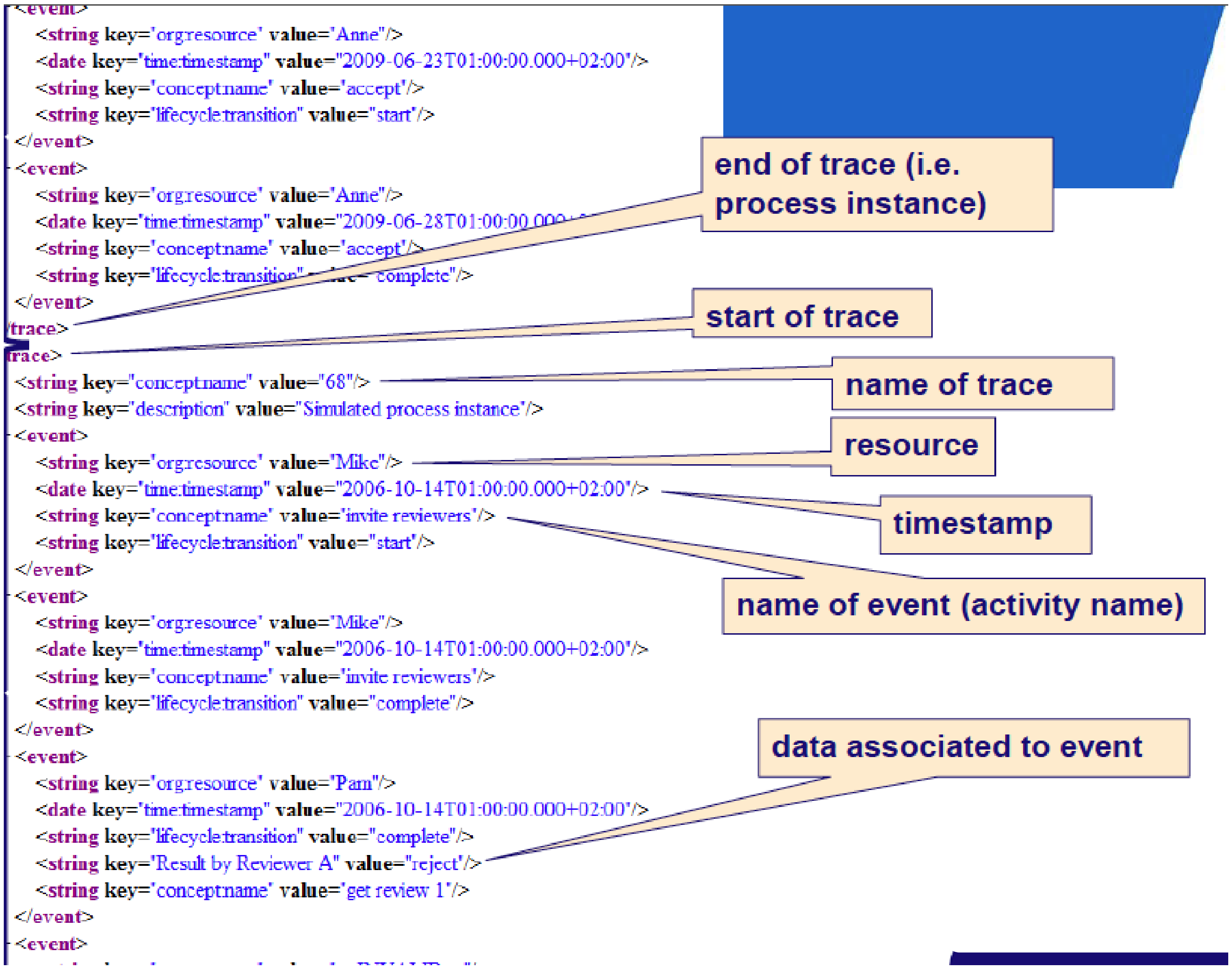
- Event log consists of
  - traces (process instances)
  - events

- Standard extensions
  - concept (for naming)
  - lifecycle (for transactional properties)
  - org (for the organizational perspective)
  - time (for timestamps)
  - semantic (for ontology references)

XES log example

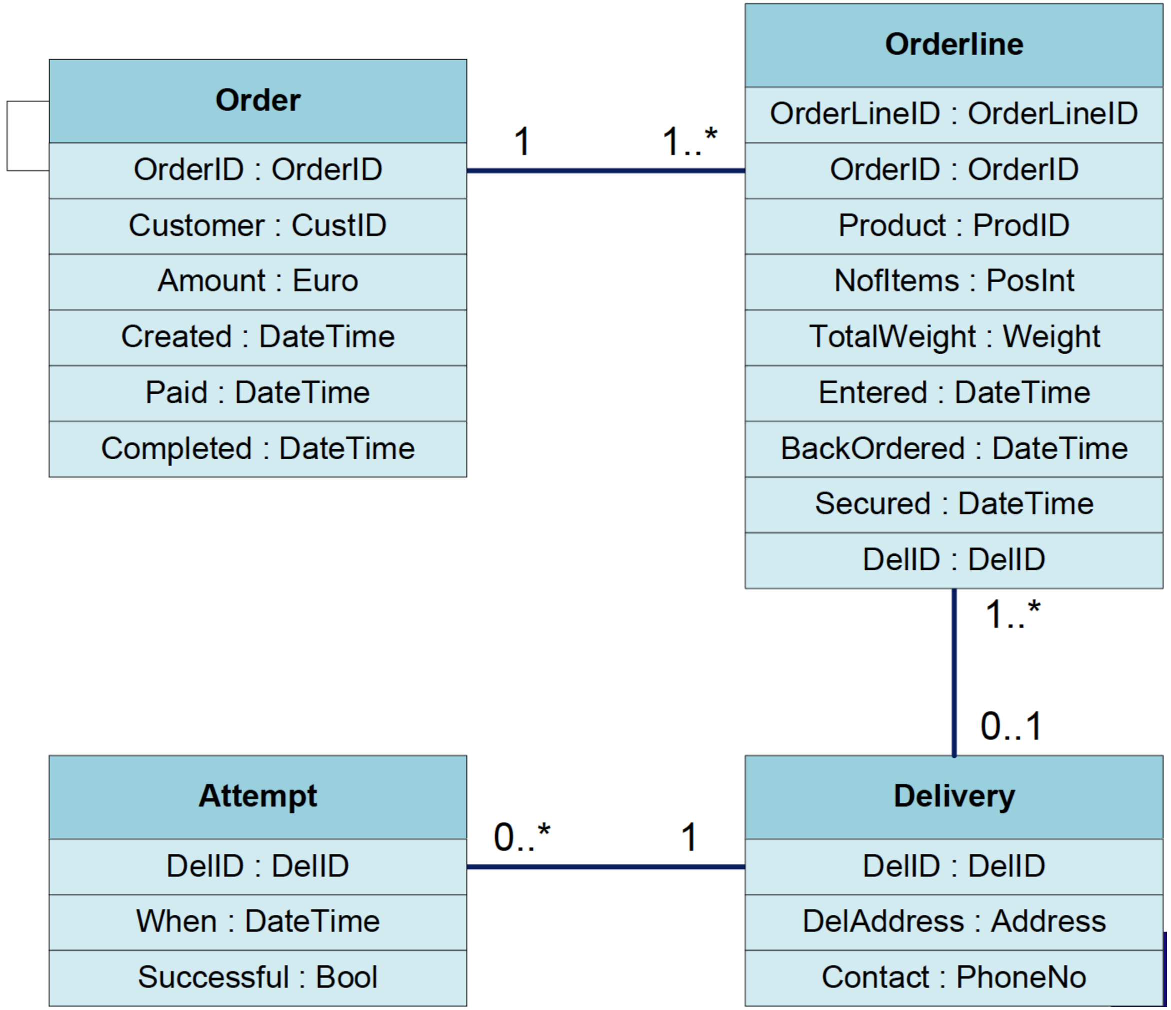


XES log example

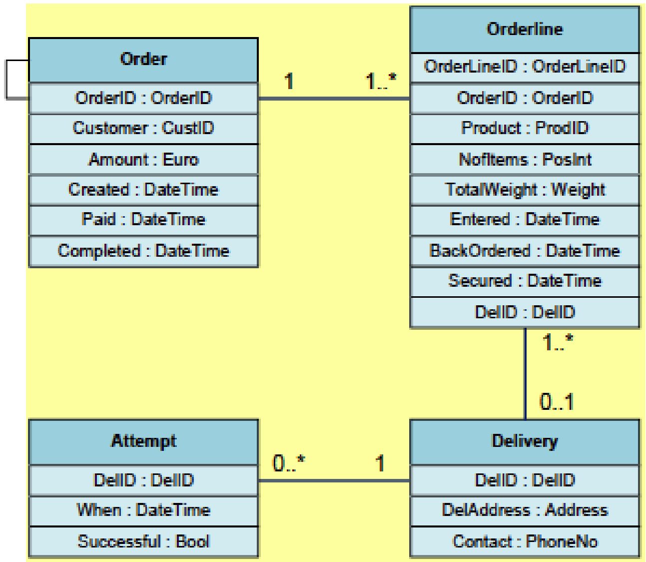



```
<event>
  <string key='org:resource' value='Anne'/>
  <date key='time:timestamp' value="2009-06-23T01:00:00.000+02:00"/>
  <string key='concept:name' value='accept'/>
  <string key='lifecycle:transition' value="start"/>
</event>
<event>
  <string key='org:resource' value='Anne'/>
  <date key='time:timestamp' value="2009-06-28T01:00:00.000...
  <string key='concept:name' value='accept'/>
  <string key='lifecycle:transition' value="complete"/>
</event>
</trace>
<trace>
<string key="concept:name" value="68"/>
<string key="description" value="Simulated process instance"/>
<event>
  <string key='org:resource' value='Mike'/>
  <date key='time:timestamp' value="2006-10-14T01:00:00.000+02:00"/>
  <string key='concept:name' value='invite reviewers'/>
  <string key='lifecycle:transition' value="start"/>
</event>
<event>
  <string key='org:resource' value='Mike'/>
  <date key='time:timestamp' value="2006-10-14T01:00:00.000+02:00"/>
  <string key='concept:name' value='invite reviewers'/>
  <string key='lifecycle:transition' value="complete"/>
</event>
<event>
  <string key='org:resource' value='Pam'/>
  <date key='time:timestamp' value="2006-10-14T01:00:00.000+02:00"/>
  <string key='lifecycle:transition' value="complete"/>
  <string key='Result by Reviewer A' value='reject'/>
  <string key='concept:name' value='get review 1'/>
</event>
<event>
```

Building an event log



Building an event log



Order

OrderID	Customer	Amount	Created	Paid	Completed
91245	John	100	28-11-2011:08.12	02-12-2011:13.45	05-12-2011:11.33
91561	Mike	530	28-11-2011:12.22	03-12-2011:14.34	05-12-2011:09.32
91812	Mary	234	29-11-2011:09.45	02-12-2011:09.44	04-12-2011:13.33
92233	Sue	110	29-11-2011:10.12	null	null
92345	Kirsten	195	29-11-2011:14.45	02-12-2011:13.45	null
92355	Pete	320	29-11-2011:16.32	null	null
...	...	...	...	...	...

Delivery

DelID	DelAddress	Contact
882345	5513VJ-22a	0497-2553660
882346	5513XG-45	040-2298761
...	...	...

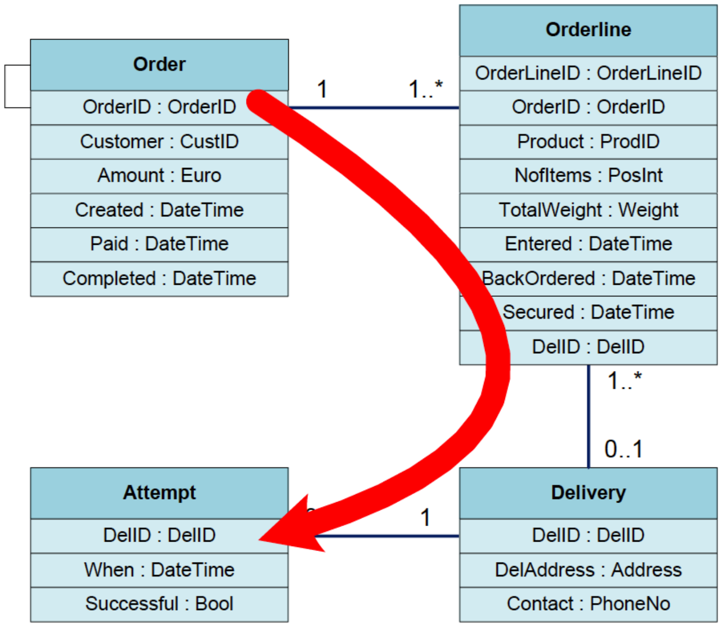
Attempt

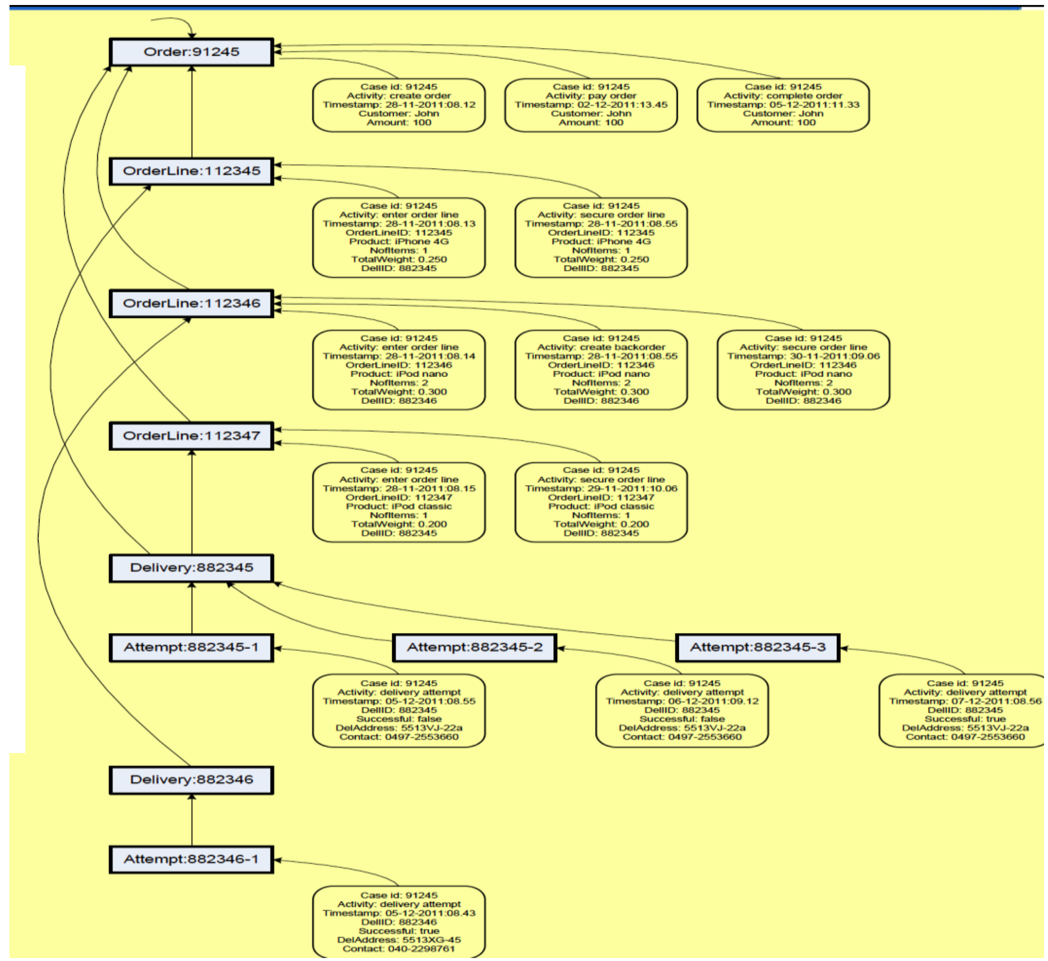
DelID	When	Successful
882345	05-12-2011:08.55	false
882345	06-12-2011:09.12	false
882345	07-12-2011:08.56	true
882346	05-12-2011:08.43	true
...	...	...

Orderline

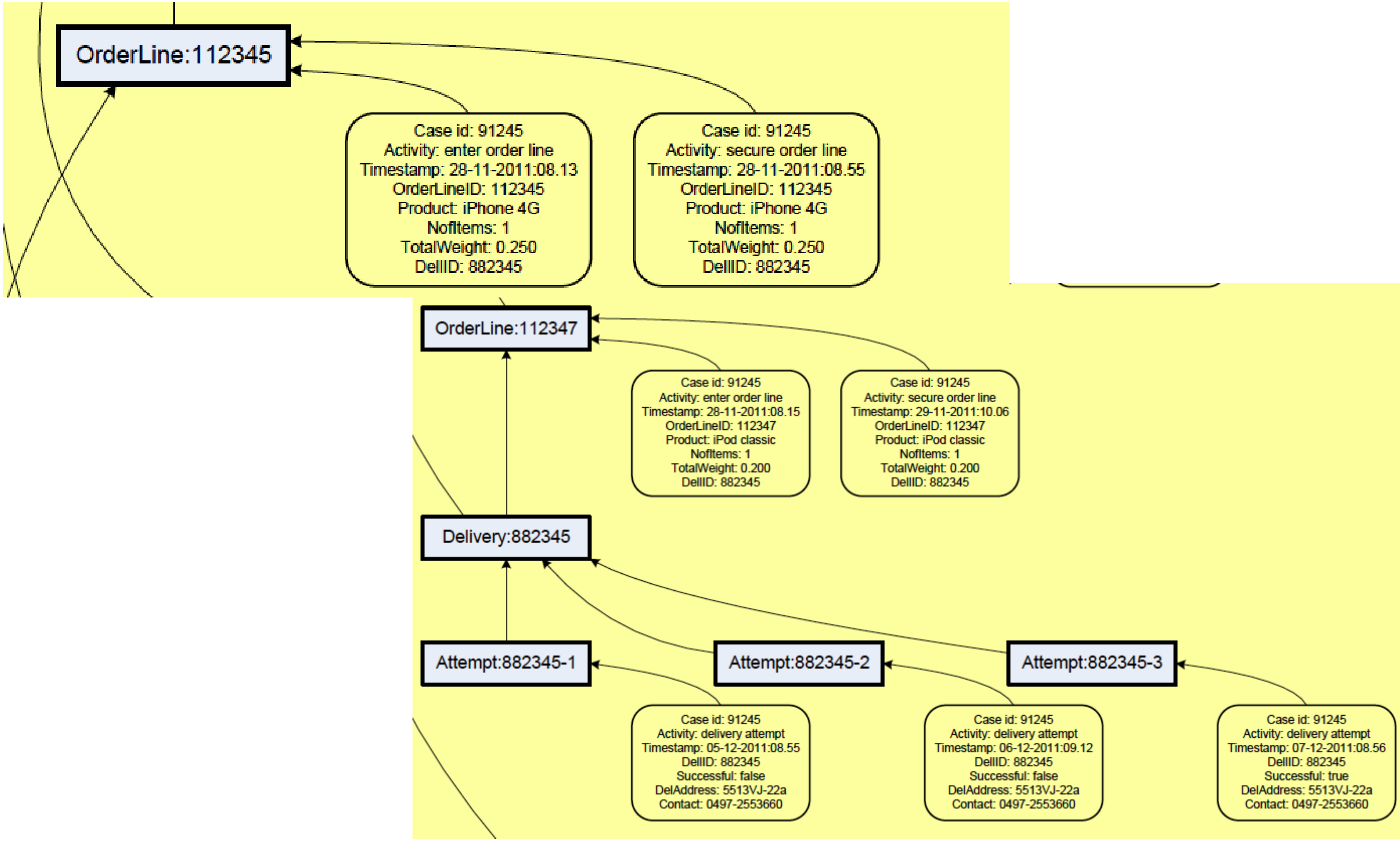
OrderLineID	OrderID	Product	NofItems	TotalWeight	Entered	BackOrdered	Secured	DelID
112345	91245	iPhone 4G	1	0.250	28-11-2011:08.13	null	28-11-2011:08.55	882345
112346	91245	iPod nano	2	0.300	28-11-2011:08.14	28-11-2011:08.55	30-11-2011:09.06	882346
112347	91245	iPod classic	1	0.200	28-11-2011:08.15	null	29-11-2011:10.06	882345
112448	91561	iPhone 4G	1	0.250	28-11-2011:12.23	null	28-11-2011:12.59	882345
112449	91561	iPod classic	1	0.200	28-11-2011:12.24	28-11-2011:16.22	null	null

Building an event log



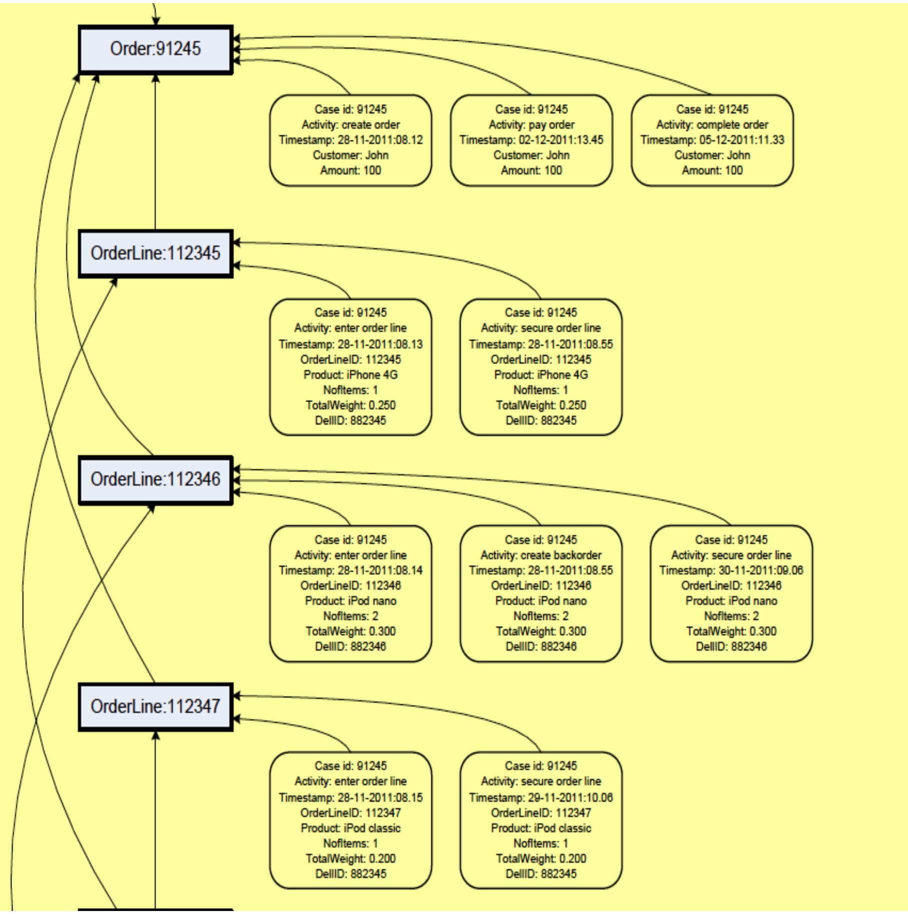


Building an event log



Building an event log

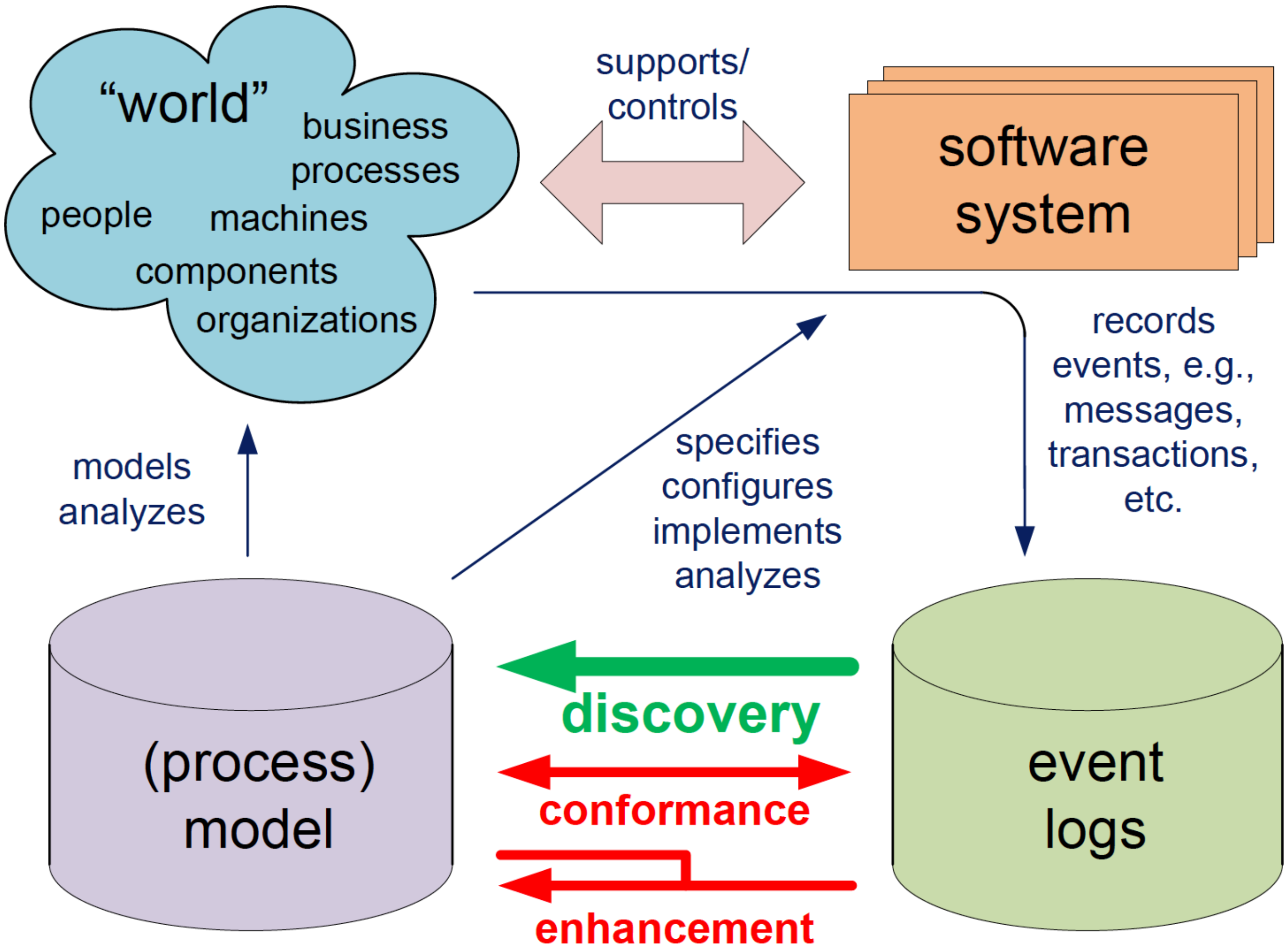
case id	activity	timestamp	other attributes
91245	create order	28-11-2011:08.12	Customer: John, Amount: 100
91245	enter order line	28-11-2011:08.13	OrderLineID: 112345, Product: iPhone 4G, NofItems: 1, TotalWeight: 0.250, DelIID: 882345
91245	enter order line	28-11-2011:08.14	OrderLineID: 112346, Product: iPod nano, NofItems: 2, TotalWeight: 0.300, DelIID: 882346
91245	enter order line	28-11-2011:08.15	OrderLineID: 112347, Product: iPod classic, NofItems: 1, TotalWeight: 0.200, DelIID: 882345
91245	secure order line	28-11-2011:08.55	OrderLineID: 112345, Product: iPhone 4G, NofItems: 1, TotalWeight: 0.250, DelIID: 882345
91245	create backorder	28-11-2011:08.55	OrderLineID: 112346, Product: iPod nano, NofItems: 2, TotalWeight: 0.300, DelIID: 882346
91245	secure order line	29-11-2011:10.06	OrderLineID: 112347, Product: iPod classic, NofItems: 1, TotalWeight: 0.200, DelIID: 882345
91245	secure order line	30-11-2011:09.06	OrderLineID: 112346, Product: iPod nano, NofItems: 2, TotalWeight: 0.300, DelIID: 882346
91245	pay order	02-12-2011:13.45	Customer: John, Amount: 100
91245	delivery attempt	05-12-2011:08.43	DelIID: 882346, Successful: true, DelAddress: 5513XG-45, Contact: 040-2298761
91245	delivery attempt	05-12-2011:08.55	DelIID: 882345, Successful: false, DelAddress: 5513VJ-22a, Contact: 0497-2553660
91245	complete order	05-12-2011:11.33	Customer: John, Amount: 100
91245	delivery attempt	06-12-2011:09.12	DelIID: 882345, Successful: false, DelAddress: 5513VJ-22a, Contact: 0497-2553660
91245	delivery attempt	07-12-2011:08.56	DelIID: 882345, Successful: true, DelAddress: 5513VJ-22a, Contact: 0497-2553660
91561	create order	28-11-2011:12.22	Customer: Mike, Amount: 530
91561	enter order line	28-11-2011:12.23	OrderLineID: 112448, Product: iPhone 4G, NofItems: 1, TotalWeight: 0.250, DelIID: 882345
...	...	...	...



Outline

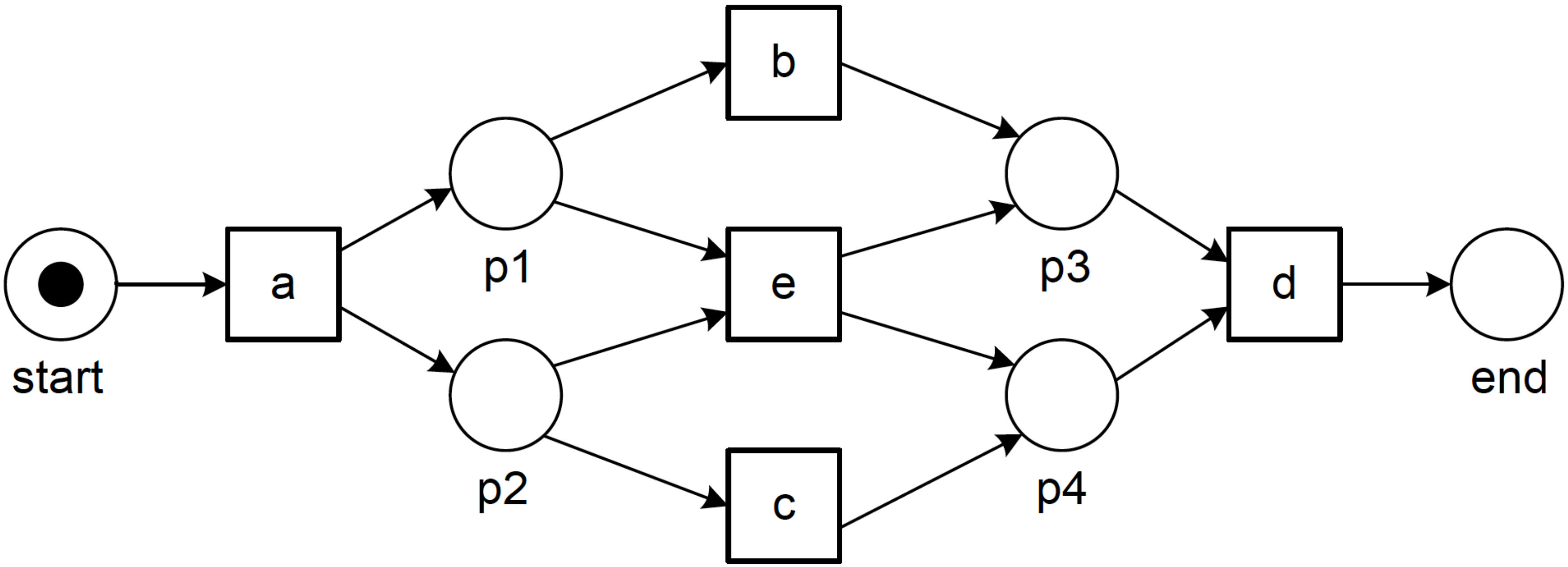
- Motivation. Process mining basics.
- Getting event data.
- Process discovery.
- Conformance checking.
- Online Process Mining.
- Tools.
- Conclusion.

Process discovery



Process discovery

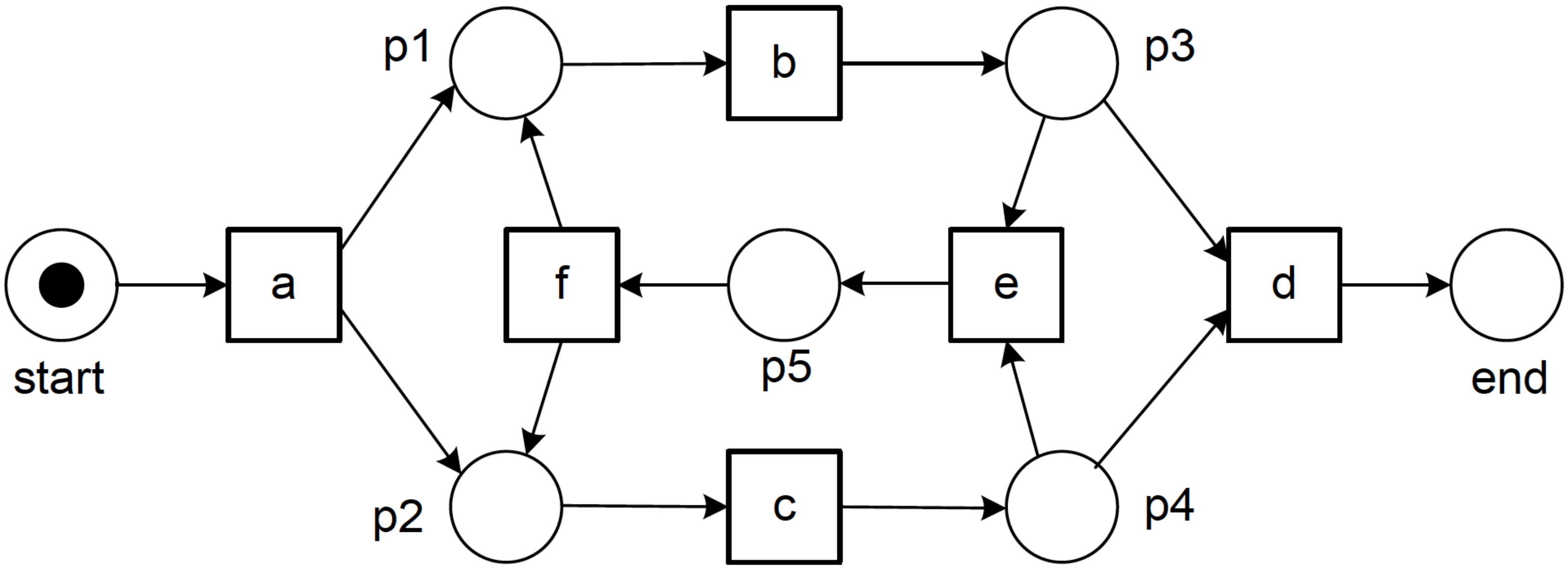
In this example, the event log contains all possible (correct) traces of the model.



$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

Process discovery

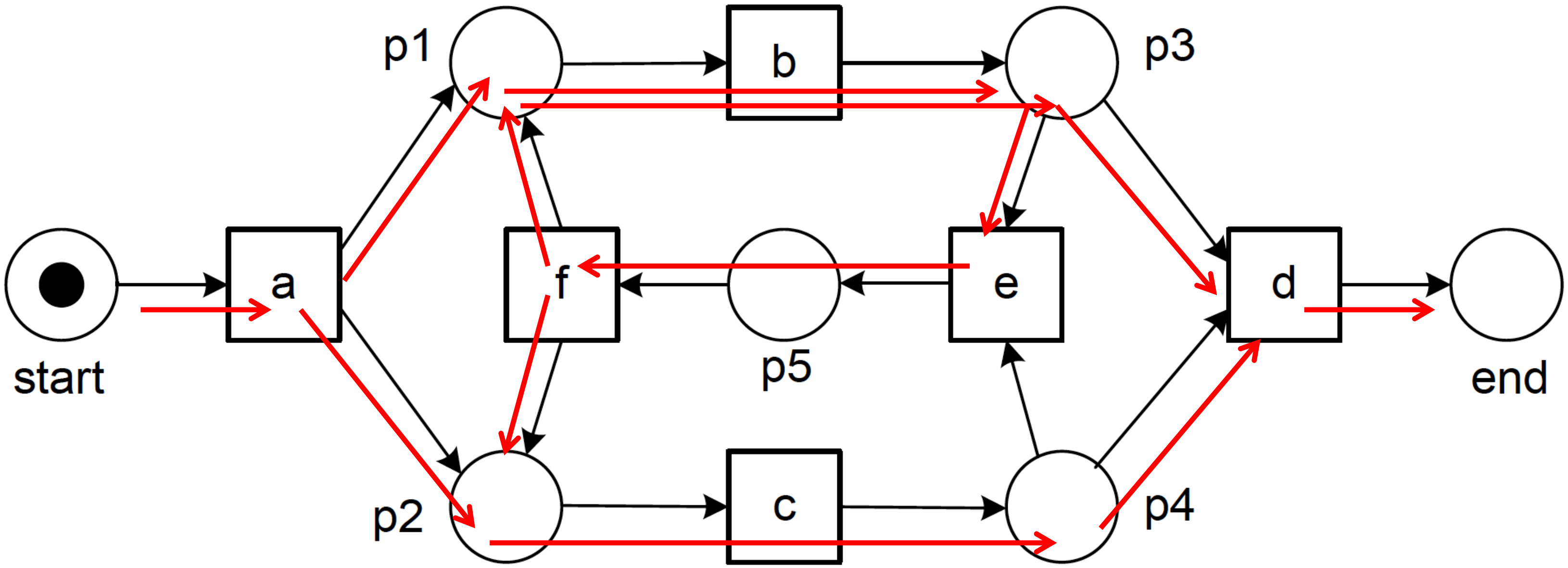
Generalization: event log contains only a subset of all possible traces of model.



$$L_2 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^4, \langle a,b,c,e,f,b,c,d\rangle^2, \langle a,b,c,e,f,c,b,d\rangle, \langle a,c,b,e,f,b,c,d\rangle^2, \langle a,c,b,e,f,b,c,e,f,c,b,d\rangle]$$

Process discovery

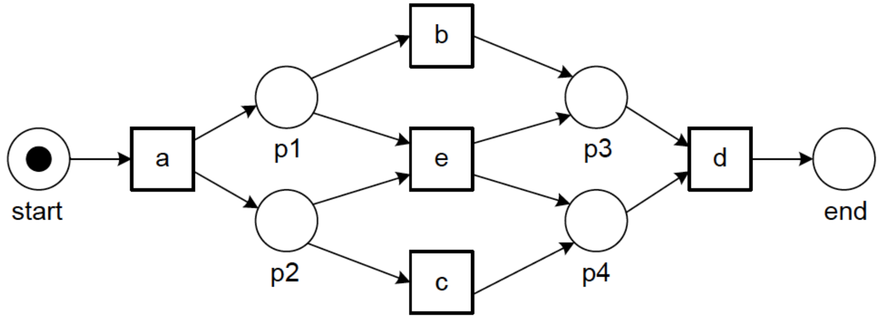
Generalization: event log contains only a subset of all possible traces of model.

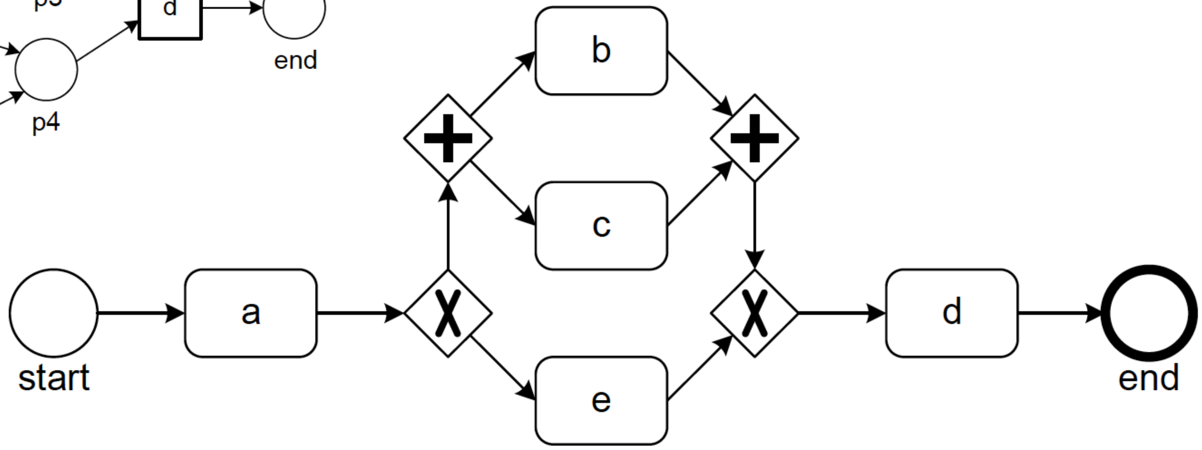


$$L_2 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^4, \langle a,b,c,e,f,b,c,d\rangle^2, \langle a,b,c,e,f,c,b,d\rangle, \langle a,c,b,e,f,b,c,d\rangle^2, \langle a,c,b,e,f,b,c,e,f,c,b,d\rangle]$$

Process discovery

Any notation could be used (Petri nets, YAWL, BPMN).

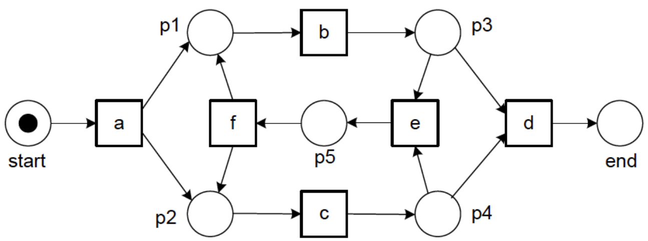


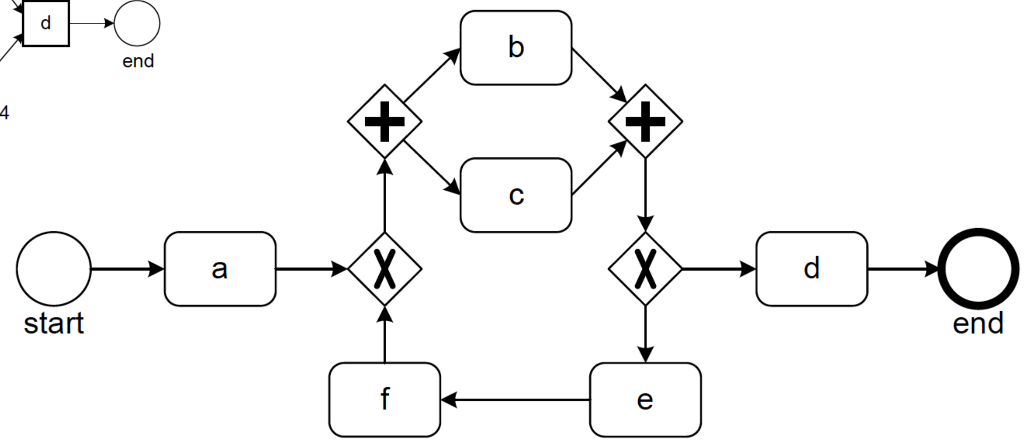


$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

Process discovery

Any notation could be used (Petri nets, YAWL, BPMN).





$$L_2 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^4, \langle a,b,c,e,f,b,c,d\rangle^2, \langle a,b,c,e,f,c,b,d\rangle,$$
$$\langle a,c,b,e,f,b,c,d\rangle^2, \langle a,c,b,e,f,b,c,e,f,c,b,d\rangle]$$

Process discovery

- **Trade-off between the following four quality criteria:**
  - Fitness: the discovered model should allow for the behavior seen in the event log.
  - Precision (avoid underfitting): the discovered model should not allow for behavior completely unrelated to what was seen in the event log.
  - Generalization (avoid overfitting): the discovered model should generalize the example behavior seen in the event log.
  - Simplicity: the discovered model should be as simple as possible.

Process discovery

- Let L be an event log. A process discovery algorithm is a function that maps L onto a process model such that the model is representative of the behavior seen in the event log.
- Challenge: find such algorithm.
- Definition above does not specify the process model.
- A simple event log is a multiset of traces over a set of activities A. In other words, L ε B(A*). Example:
  - $L=[\langle a,b,c,d\rangle^3,\langle a,c,b,d\rangle^2,\langle a,e,d\rangle]$
- Means that the path <a,b,c,d> is present three times in the log.

Process discovery

Recall:

- A *marked* Petri net is a pair (N,M), where N=(P,T,F) is a Petri net, and M is a multiset over P, denoted the *marking* of the net.
- We work with *sound* workflow (WF) nets, that means:
  - If the sink place is marked, all other places should be empty (proper completion).
  - It is always possible to mark the sink place.
  - The WF-net contains no dead transitions, i.e., all parts of the model are potentially reachable.

Process discovery: the α algorithm

- Given a simple event log, produces a WF-net that can replay the log.
- We explain it to give an idea, although in practice this algorithm has problems regarding noise, incomplete behavior, complex routing constructs.
- Simple, and many of its ideas were used in more sophisticated algorithms.
- Input: a simple event log L over A.
- Searches L for particular patterns.
- E.g.: if activity *a* is followed by *b, it is assumed that there is a causal dependency between a and b.*

Process discovery: the α algorithm

- There are four log-based ordering relations:
  - Direct succession**:** **$x >_L y$** iff for some case x is ***directly*** followed by y (i.e., y is an immediate successor or x).
  - Causality**:** **$x \rightarrow_L y$** iff x>y and not y>x.
  - Parallel**:** **$x \|_L y$** iff x>y and y>x.
  - Choice**:** **$x \#_L y$ iff** not x>y and not y>x.

Process discovery: the α algorithm

$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

abcd
acbd
aed

a>b
a>c
a>e
b>c
b>d
c>b
c>d
e>d

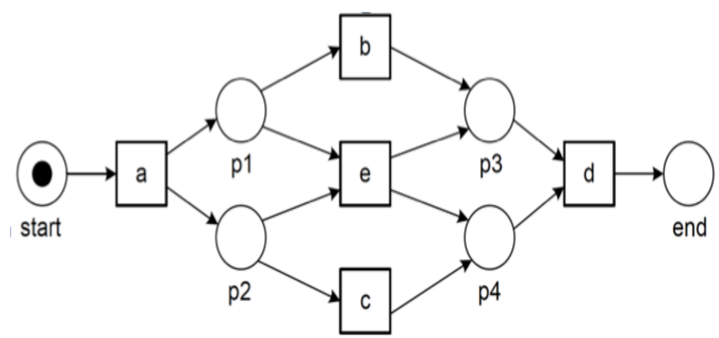
a→b
a→c
a→e
b→d
c→d
e→d

b||c
c||b

b>c ∧ c>b

b#e
e#b
c#e
a#d
. . .

not b>e
∧ not e>b



Process discovery: the α algorithm

- L_1=[<*a,b,c,d*>[3],<*a,c,b,d*>[2],<*a,e,d*>]

  $>_{L1}$= {(a,b), (a,c), (a,e), (b,c), (c,b), (b,d), (c,d), (e,d)}

  $\to_{L1}$= {(a,b), (a,c), (a,e), (b,d), (c,d), (e,d)} (contains all the pairs in a causality relation).

  $\#_{L1}$ = {(a,a), (a,d), (b,b), (b,e), (c,c), (c,e), (d,a), (d,d), (e,b), (e,c), (e,e)}

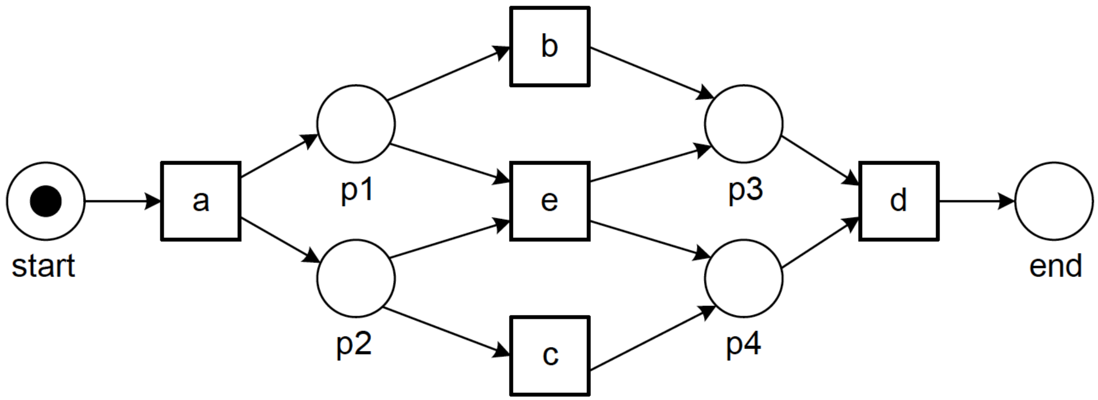
  $||_{L1}$ = {(b,c),(c,b)}

For any log, only one of x $\to_L$y, y$\to_L$x, x $\#_L$y, x $||_L$y holds. This can be captured in a matrix called *the footprint of the log*.

Process discovery: the α algorithm

Footprint matrix. (e.g., “there is a sequence pattern a -> b in L1”, and “a # d in L1” (neither a > d, nor d>a).

$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

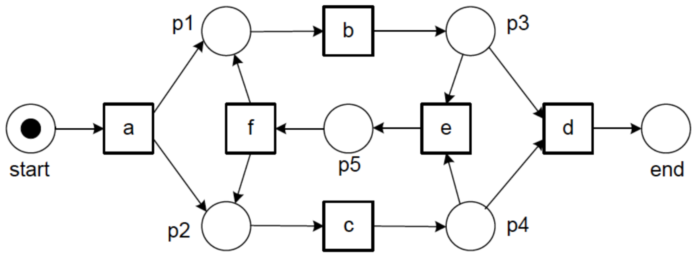


	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

Process discovery: the α algorithm

Another example

$$L_2 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^4, \langle a,b,c,e,f,b,c,d\rangle^2, \langle a,b,c,e,f,c,b,d\rangle,$$
$$\langle a,c,b,e,f,b,c,d\rangle^2, \langle a,c,b,e,f,b,c,e,f,c,b,d\rangle]$$

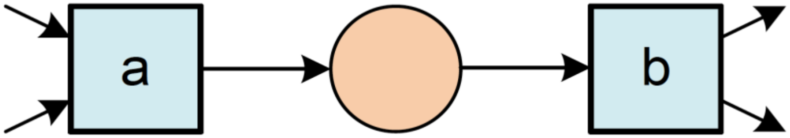


	a	b	c	d	e	f
a	#	$\rightarrow$	$\rightarrow$	#	#	#
b	$\leftarrow$	#	$\parallel$	$\rightarrow$	$\rightarrow$	$\leftarrow$
c	$\leftarrow$	$\parallel$	#	$\rightarrow$	$\rightarrow$	$\leftarrow$
d	#	$\leftarrow$	$\leftarrow$	#	#	#
e	#	$\leftarrow$	$\leftarrow$	#	#	$\rightarrow$
f	#	$\rightarrow$	$\rightarrow$	#	$\leftarrow$	#

Process discovery: the α algorithm

The log-based relationships can be used to discover patterns.

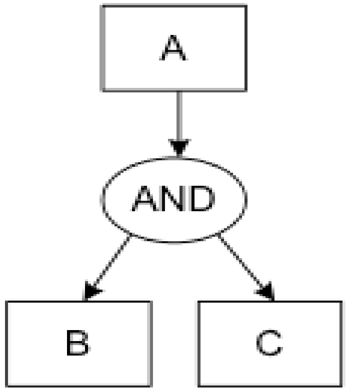
Sequence pattern a->b



Process discovery: the α algorithm

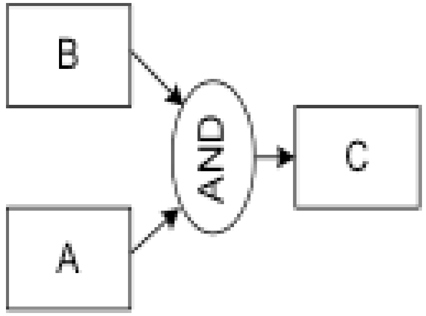
AND-split pattern

A -> B, A -> C,
B || C



AND-join pattern

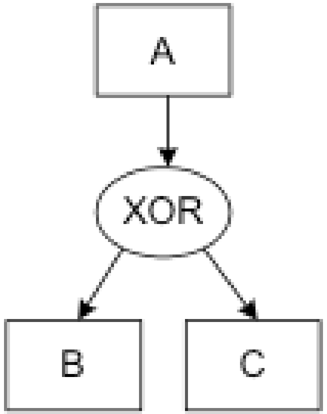
B -> C, A -> C,
B || A



Process discovery: the α algorithm

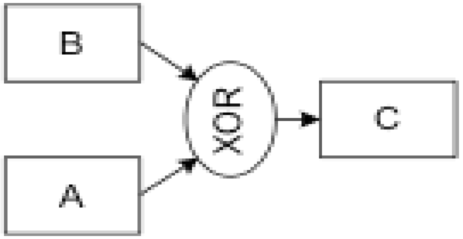
XOR-split pattern

A -> B, A -> C,
B # C.



XOR-join pattern

B -> C, A -> C,
B # A.



Process discovery: the α algorithm

Let L be an event log over T. α(L) is defined as follows.

1. $T_L = \{ t \in T \mid \exists_{\sigma \in L}\, t \in \sigma \}$,

2. $T_I = \{ t \in T \mid \exists_{\sigma \in L}\, t = \mathit{first}(\sigma) \}$,

3. $T_O = \{ t \in T \mid \exists_{\sigma \in L}\, t = \mathit{last}(\sigma) \}$,

- In step 1, T_L contains all the activities in the log. (σ is a trace in the log)
- In step 2, T_I contains all start activities (the ones appearing first in some trace).
- In step 3, T_0 contains all end activities activities (the ones appearing last in some trace).

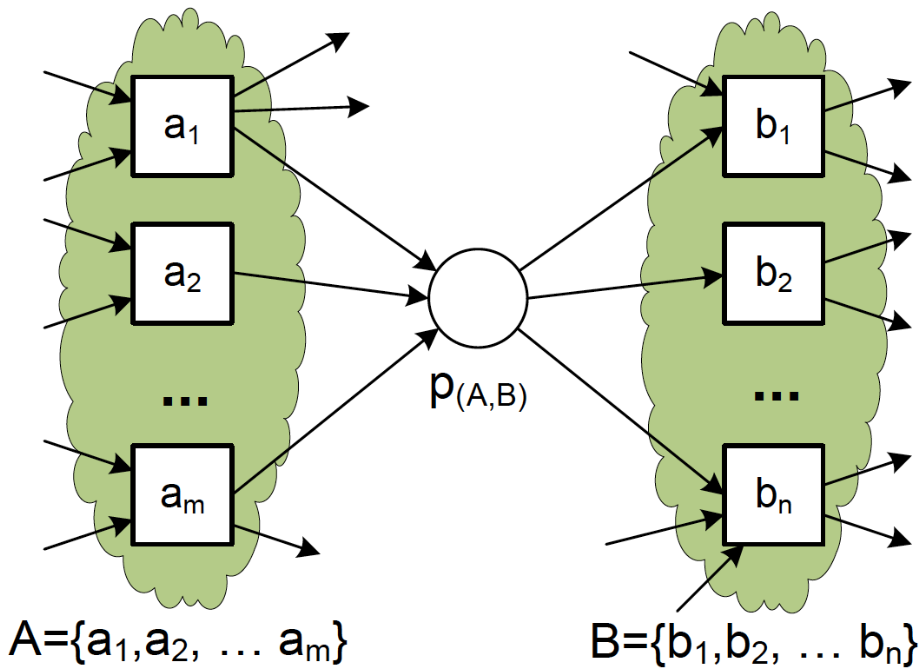
Process discovery: the α algorithm

Let L be an event log over T. α(L) is defined as follows.

1. $T_L = \{ t \in T \mid \exists_{\sigma \in L}\ t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in L}\ t = \mathit{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in L}\ t = \mathit{last}(\sigma) \}$,
4. $X_L = \{ (A,B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge \forall_{a \in A} \forall_{b \in B}\ a \rightarrow_L b \wedge \forall_{a1,a2 \in A}\ a_1 \#_L a_2 \wedge \forall_{b1,b2 \in B}\ b_1 \#_L b_2 \}$,
5. $Y_L = \{ (A,B) \in X_L \mid \forall_{(A',B') \in X_L}\ A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,

- Steps 4 and 5 are the core of the algorithm. The challenge is to determine the WF-net places and their connections.
- We want to build places p(A,B) s.t. A is the set of input transitions and B is the set of output transitions, and all elements in A have causal relationships with all elements in B, but there no causal relationships between elements within A and B. Next slide shows this.

Process discovery: the α algorithm

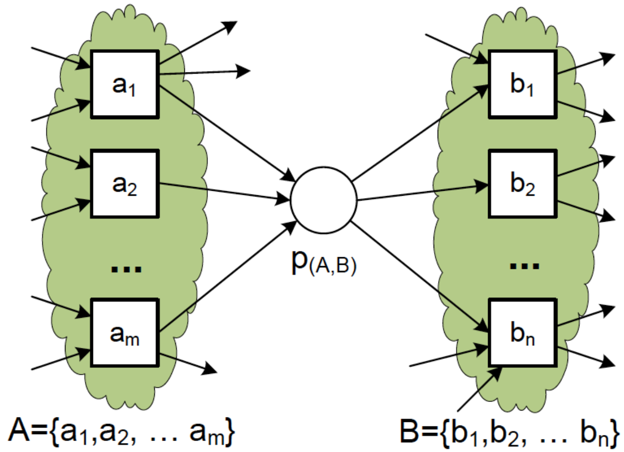


4. $X_L = \{ (A,B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_L b \wedge \forall_{a1,a2 \in A}\, a_1 \#_L a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_L b_2 \}$,

5. $Y_L = \{ (A,B) \in X_L \mid \forall_{(A',B') \in X_L}\, A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,

Process discovery: the α algorithm

The matrix has this form.

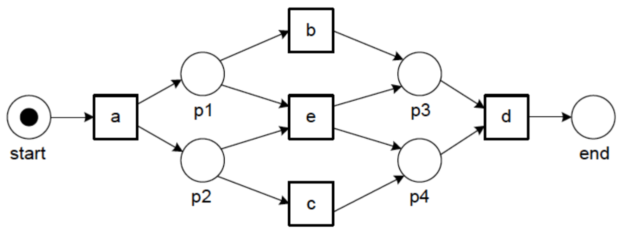


	a_1	a_2	…	a_m	b_1	b_2	…	b_n
a_1	#	#	…	#	$\rightarrow$	$\rightarrow$	…	$\rightarrow$
a_2	#	#	…	#	$\rightarrow$	$\rightarrow$	…	$\rightarrow$
…	…	…	…	…	…	…	…	…
a_m	#	#	…	#	$\rightarrow$	$\rightarrow$	…	$\rightarrow$
b_1	$\leftarrow$	$\leftarrow$	…	$\leftarrow$	#	#	…	#
b_2	$\leftarrow$	$\leftarrow$	…	$\leftarrow$	#	#	…	#
…	…	…	…	…	…	…	…	…
b_n	$\leftarrow$	$\leftarrow$	…	$\leftarrow$	#	#	…	#

Process discovery: the α algorithm

For the log $L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$.

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\Vert_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\Vert_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$



$$X_{L_1} = \{(\{a\},\{b\}),(\{a\},\{c\}),(\{a\},\{e\}),(\{a\},\{b,e\}),(\{a\},\{c,e\}),\\ (\{b\},\{d\}),(\{c\},\{d\}),(\{e\},\{d\}),(\{b,e\},\{d\}),(\{c,e\},\{d\})\}$$

Y_{L1} (step 5) keeps only the maximal pairs (A,B) in X_{L1}.

$$Y_{L_1} = \{(\{a\},\{b,e\}),(\{a\},\{c,e\}),(\{b,e\},\{d\}),(\{c,e\},\{d\})\}$$

Non-maximal pairs in X_{L1} have been removed.

Process discovery: the α algorithm

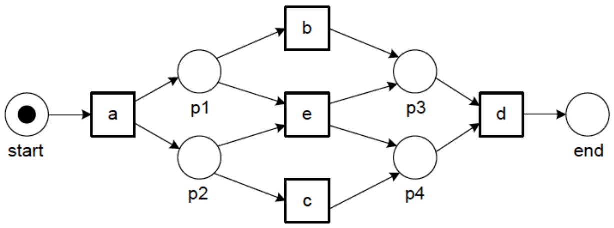
Let L be an event log over T. $\alpha(L)$ is defined as follows.

1. $T_L = \{ t \in T \mid \exists_{\sigma \in L}\, t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in L}\, t = \mathit{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in L}\, t = \mathit{last}(\sigma) \}$,
4. $X_L = \{ (A,B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_L b \wedge \forall_{a1,a2 \in A}\, a_1 \#_L a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_L b_2 \}$,
5. $Y_L = \{ (A,B) \in X_L \mid \forall_{(A',B') \in X_L}\, A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_L = \{ p_{(A,B)} \mid (A,B) \in Y_L \} \cup \{ i_L, o_L \}$,
7. $F_L = \{ (a, p_{(A,B)}) \mid (A,B) \in Y_L \wedge a \in A \} \cup \{ (p_{(A,B)}, b) \mid (A,B) \in Y_L \wedge b \in B \} \cup \{ (i_L, t) \mid t \in T_I \} \cup \{ (t, o_L) \mid t \in T_O \}$, and
8. $\alpha(L) = (P_L, T_L, F_L)$.

Process discovery: the α algorithm

For the log $L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$.

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\Vert_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\Vert_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$



Step 6 adds the initial and final places (only once initial and one final place).

Step 7 builds the edges.

Process discovery: the α algorithm

$$L_5 = [\langle a,b,e,f\rangle^2, \langle a,b,e,c,d,b,f\rangle^3, \langle a,b,c,e,d,b,f\rangle^2, \langle a,b,c,d,e,b,f\rangle^4, \langle a,e,b,c,d,b,f\rangle^3]$$

	a	b	c	d	e	f
a	$\#$	$\rightarrow$	$\#$	$\#$	$\rightarrow$	$\#$
b	$\leftarrow$	$\#$	$\rightarrow$	$\leftarrow$	$\parallel$	$\rightarrow$
c	$\#$	$\leftarrow$	$\#$	$\rightarrow$	$\parallel$	$\#$
d	$\#$	$\rightarrow$	$\leftarrow$	$\#$	$\parallel$	$\#$
e	$\leftarrow$	$\parallel$	$\parallel$	$\parallel$	$\#$	$\rightarrow$
f	$\#$	$\leftarrow$	$\#$	$\#$	$\leftarrow$	$\#$

Process discovery: the α algorithm

$$L_5 = [\langle a,b,e,f\rangle^2, \langle a,b,e,c,d,b,f\rangle^3, \langle a,b,c,e,d,b,f\rangle^2, \langle a,b,c,d,e,b,f\rangle^4, \langle a,e,b,c,d,b,f\rangle^3]$$

$$T_L = \{a,b,c,d,e,f\}$$

$$T_I = \{a\}$$

$$T_O = \{f\}$$

$$X_L = \{(\{a\},\{b\}),(\{a\},\{e\}),(\{b\},\{c\}),(\{b\},\{f\}),(\{c\},\{d\}), (\{d\},\{b\}),(\{e\},\{f\}),(\{a,d\},\{b\}),(\{b\},\{c,f\})\}$$

$$Y_L = \{(\{a\},\{e\}),(\{c\},\{d\}),(\{e\},\{f\}),(\{a,d\},\{b\}),(\{b\},\{c,f\})\}$$

$$P_L = \{p_{(\{a\},\{e\})}, p_{(\{c\},\{d\})}, p_{(\{e\},\{f\})}, p_{(\{a,d\},\{b\})}, p_{(\{b\},\{c,f\})}, i_L, o_L\}$$

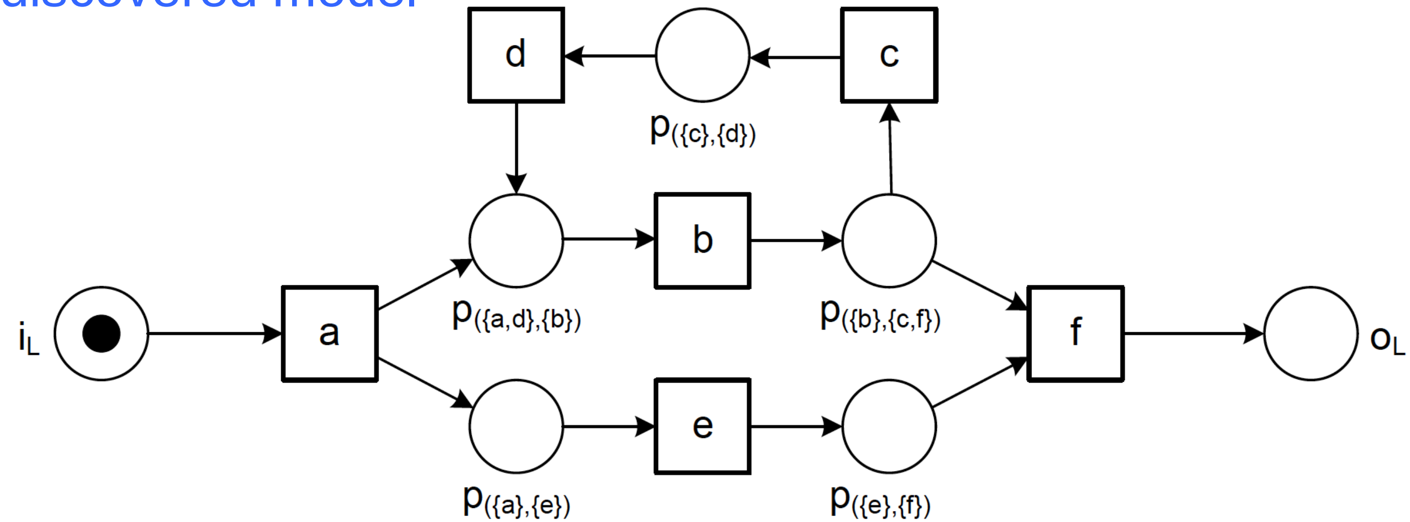
$$F_L = \{(a,p_{(\{a\},\{e\})}),(p_{(\{a\},\{e\})},e),(c,p_{(\{c\},\{d\})}),(p_{(\{c\},\{d\})},d), (e,p_{(\{e\},\{f\})}),(p_{(\{e\},\{f\})},f),(a,p_{(\{a,d\},\{b\})}),(d,p_{(\{a,d\},\{b\})}), (p_{(\{a,d\},\{b\})},b),(b,p_{(\{b\},\{c,f\})}),(p_{(\{b\},\{c,f\})},c),(p_{(\{b\},\{c,f\})},f), (i_L,a),(f,o_L)\}$$

$$\alpha(L) = (P_L, T_L, F_L)$$

Process discovery: the α algorithm

$$L_5 = [\langle a,b,e,f\rangle^2, \langle a,b,e,c,d,b,f\rangle^3, \langle a,b,c,e,d,b,f\rangle^2, \langle a,b,c,d,e,b,f\rangle^4, \langle a,e,b,c,d,b,f\rangle^3]$$

The discovered model is:



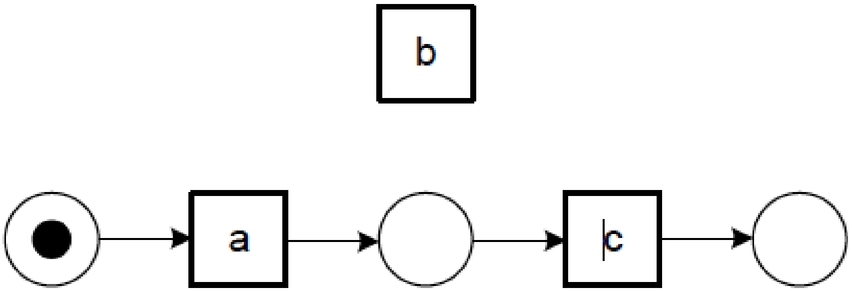
$$X_L = \{(\{a\},\{b\}),(\{a\},\{e\}),(\{b\},\{c\}),(\{b\},\{f\}),(\{c\},\{d\}), (\{d\},\{b\}),(\{e\},\{f\}),(\{a,d\},\{b\}),(\{b\},\{c,f\})\}$$

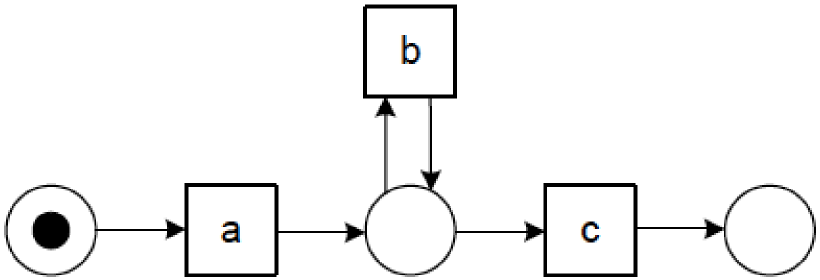
$$Y_L = \{(\{a\},\{e\}),(\{c\},\{d\}),(\{e\},\{f\}),(\{a,d\},\{b\}),(\{b\},\{c,f\})\}$$

Limitations of the α algorithm

Problems to identify
loops of length 1

$$L_7 = [\langle a,c\rangle^2, \langle a,b,c\rangle^3, \langle a,b,b,c\rangle^2, \langle a,b,b,b,b,c\rangle^1]$$

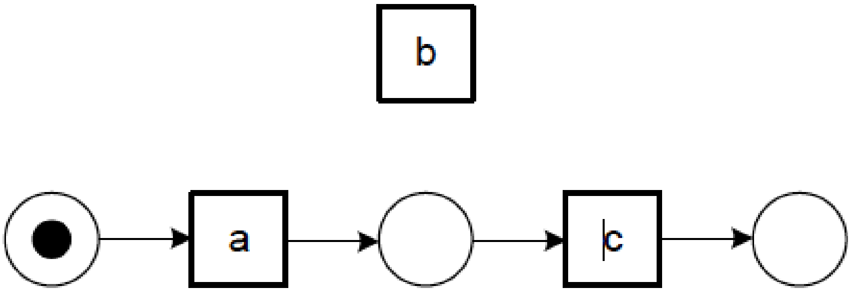




Limitations of the α algorithm

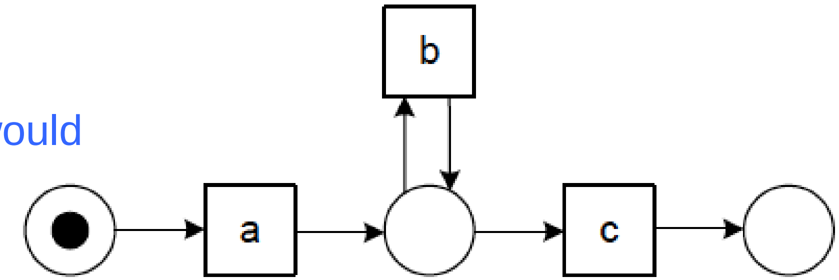
Problems to identify
loops of length 1

$$L_7 = [\langle a,c\rangle^2, \langle a,b,c\rangle^3, \langle a,b,b,c\rangle^2, \langle a,b,b,b,b,c\rangle^1]$$



All sets of the form ({*x*},{b}) and ({b},{*x*})are prevented, since b $>_{L7}$ b (causal relationship within a set). Then, the only place discovered is p({a}{c}).

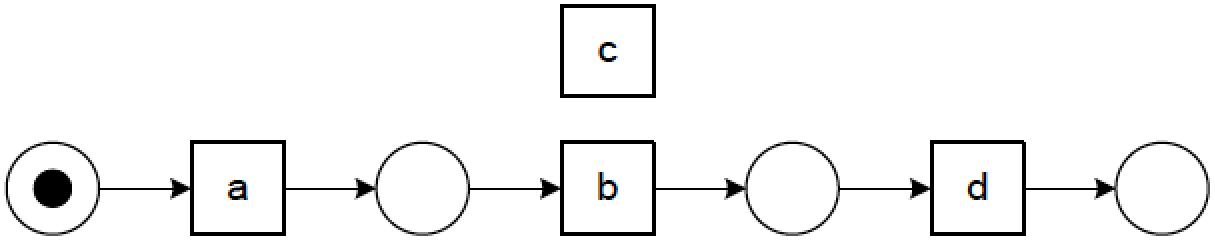
The model is incorrect (transition b is disconnected). An improved version would return =>

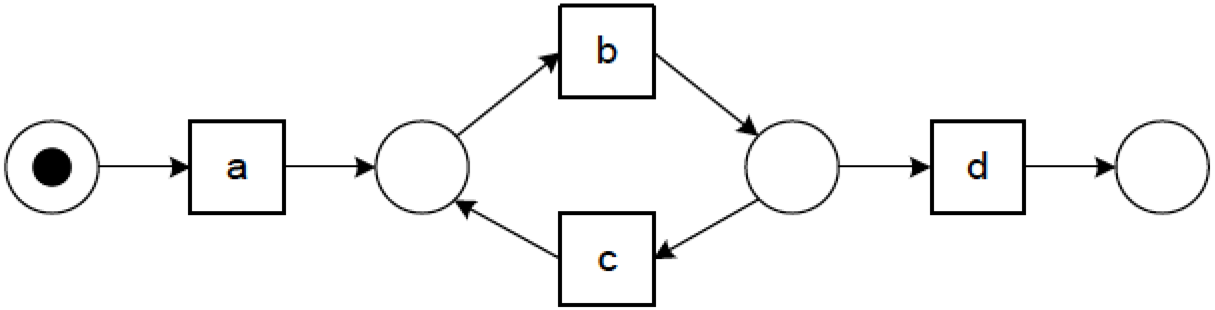


Limitations of the α algorithm

Problems to identify
loops of length 2

$$L_8 = [\langle a,b,d\rangle^3, \langle a,b,c,b,d\rangle^2, \langle a,b,c,b,c,b,d\rangle]$$

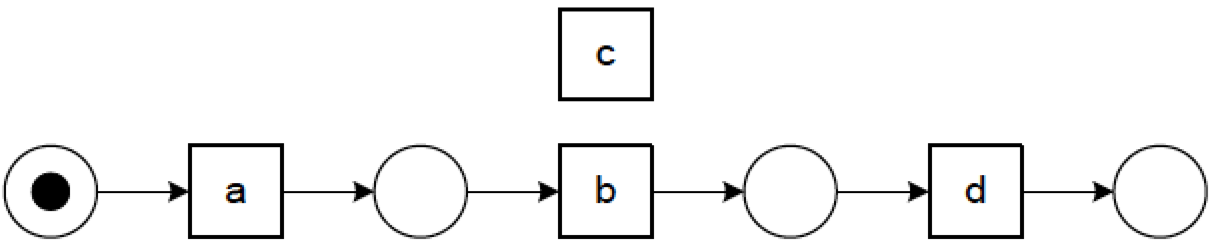




Limitations of the α algorithm

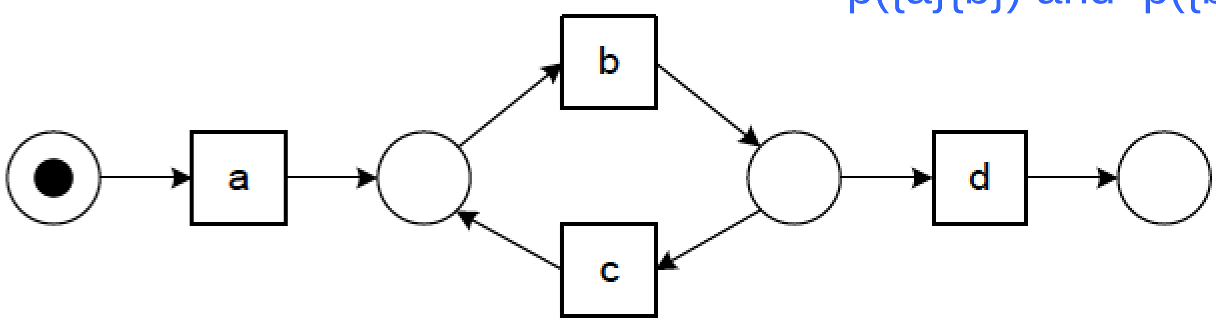
Problems to identify loops of length 2

$$L_8 = [\langle a,b,d\rangle^3, \langle a,b,c,b,d\rangle^2, \langle a,b,c,b,c,b,d\rangle]$$



All sets of the form ({b}{c}) and ({c}{b}) are prevented, since b $>_{L8}$ c and c $>_{L8}$ b (thus, b $\not\rightarrow$ c. Then, the only places discovered are p({a}{b}) and p({b}{d}).

The model is incorrect (transition c is disconnected). An improved version would return =>



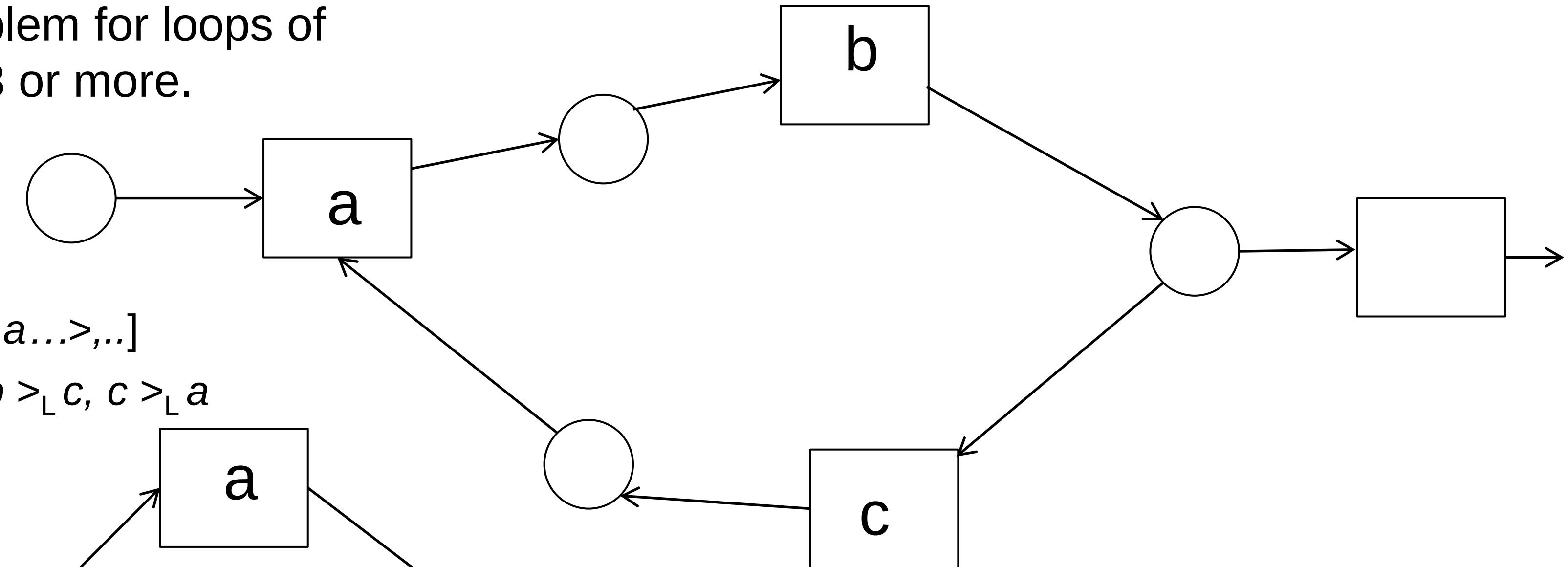
Limitations of the α algorithm

No problem for loops of
length 3 or more.

- Concurrency can be distinguished from loops, using the relation "$>_L$".
- In a loop, only $a >_L b$, $b >_L c$, $c >_L a$ are found.
- If there are three concurrent activities, we find $a >_L b$, $a >_L c$, $b >_L a$, $b >_L c$, $c >_L a$, $c >_L b$.
- In a case of a loop of length 2, we would have $a >_L b$ and $b >_L a$ in both cases.

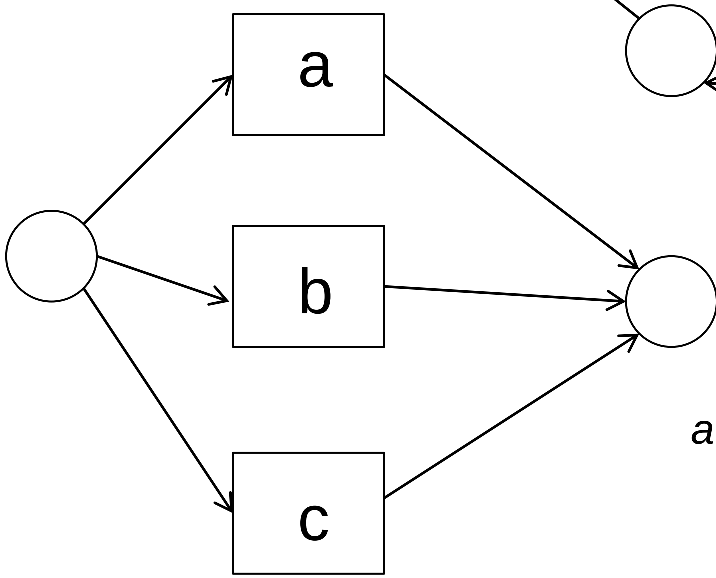
Limitations of the α algorithm

No problem for loops of
length 3 or more.



Ex.:
[< *a*,*b*,*c*,*a*…>,..]

$a >_L b$, $b >_L c$, $c >_L a$



Ex.:
[< *a*,*b*, *c*…>,..,< *c*,*b*,*a*…>]

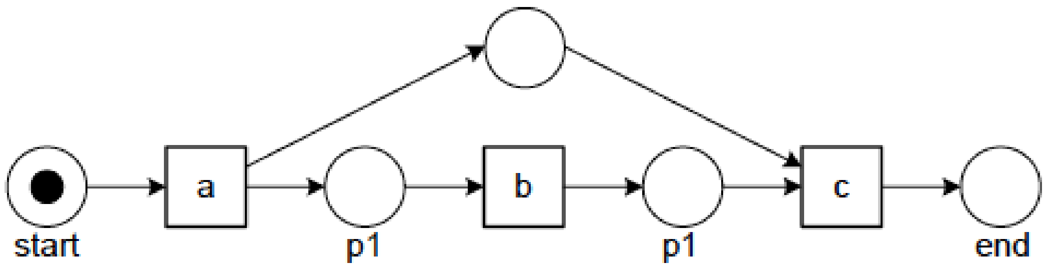
$a >_L b$, $a >_L c$, $b >_L a$, $b >_L c$, $c >_L a$, $c >_L b$.

Limitations of the α algorithm

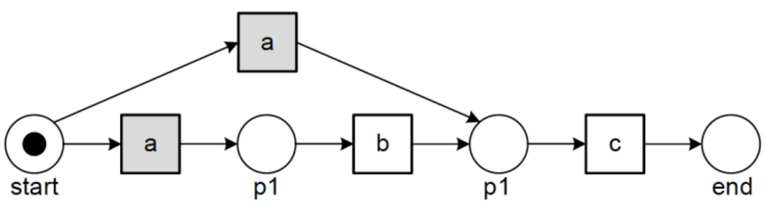
Representation

<*a,b,c*>^{20},<*a,c*>^{30}

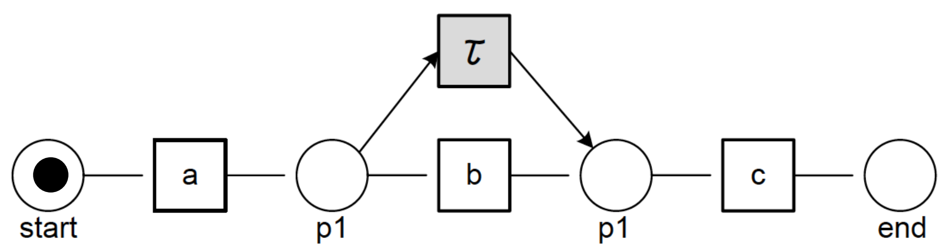
This model cannot reproduce the trace <*a,c*>



These models do, but are not valid:



(b)



Limitations of the α algorithm

Noise and Incompleteness.

- To discover a suitable process model we assumed that the event log contains a representative sample of behavior.

- Two related phenomena:
  - Noise: the event log contains rare and infrequent behavior not representative for the typical behavior of the process.

  - Incompleteness: the event log contains too few events to be able to discover some of the underlying control-flow structures.

PAGE 39

Limitations of the α algorithm

Noise

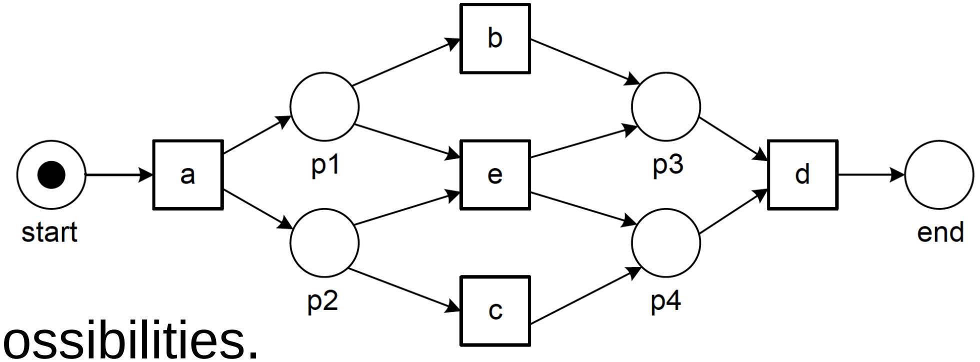
- Represents infrequent behavior, not errors.
- Must be filtered, if possible.
- *Support and confidence* can be used.
- We can define the support of $a >_L b$ based on the # of times that there is a trace <*…a,b,…*> in the event log. This threshold can be used for filtering noise.
- Example. <*…a,b,…*> appears 1,000 times, and |L|= 5000; support= 20%.
  If *a* appears 2000 times, Confidence = 1000/2000 = 0.5.

⇒ Define minimum support and/or confidence for appearing in the log.

Limitations of the α algorithm

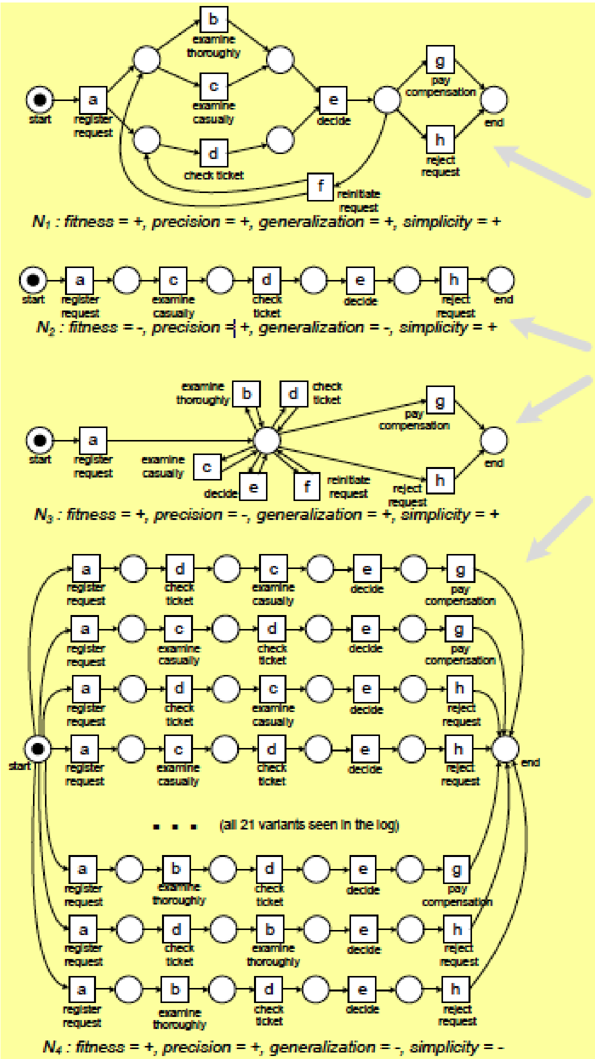
Incompleteness:

- Noise refers to the problem of having “too much data”.
- Incompleteness refers to having “too little data”.
- In this example, the traces in the log (“training set”) is the same as the set of possible traces in the model.
- Normally, this is not the case.
- <*a,b,e,c,d*> can occur, but this has not yet happenned.
- If the # of parallel activities is 10, then there are 10! = 3.628.800 interleaving possibilities.



$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

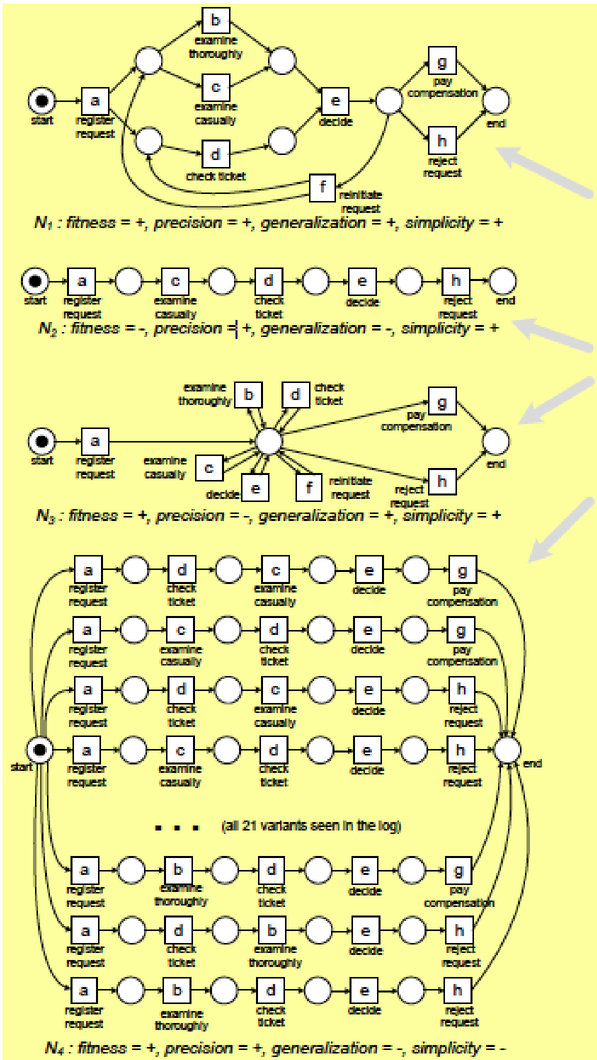
Choosing a model



#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

N1 is the model obtained with the α algorithm. Has a good balance between The FOUR characteristics We are looking for: fitness, simplicity, precision, and generalization.

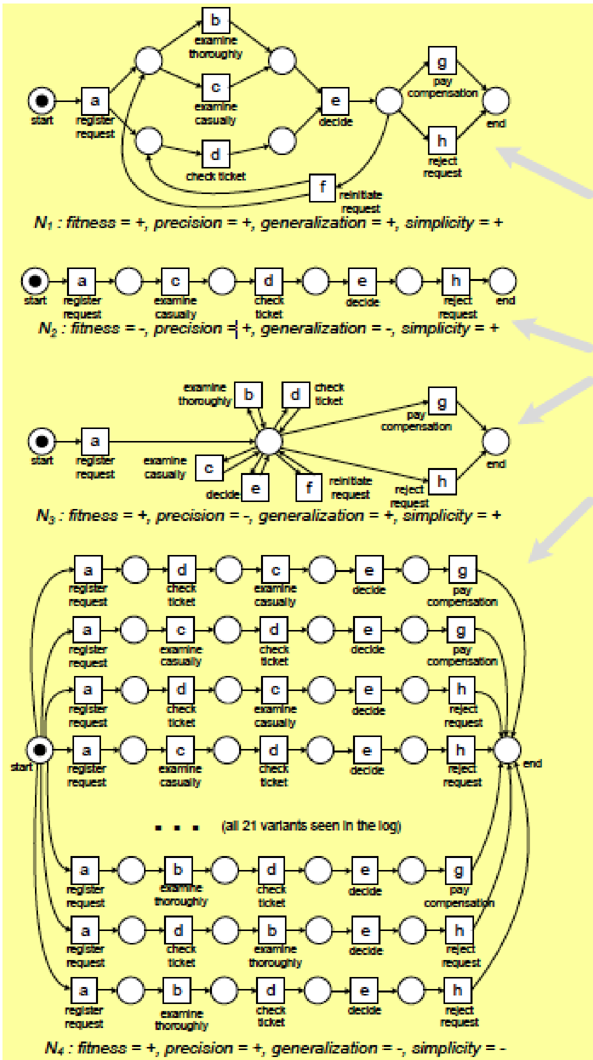
Choosing a model



#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

N2 only represents the most popular trace <*a,c,d,e,h*>. Leaves out 936 traces ⇒ less fitness and generalization.

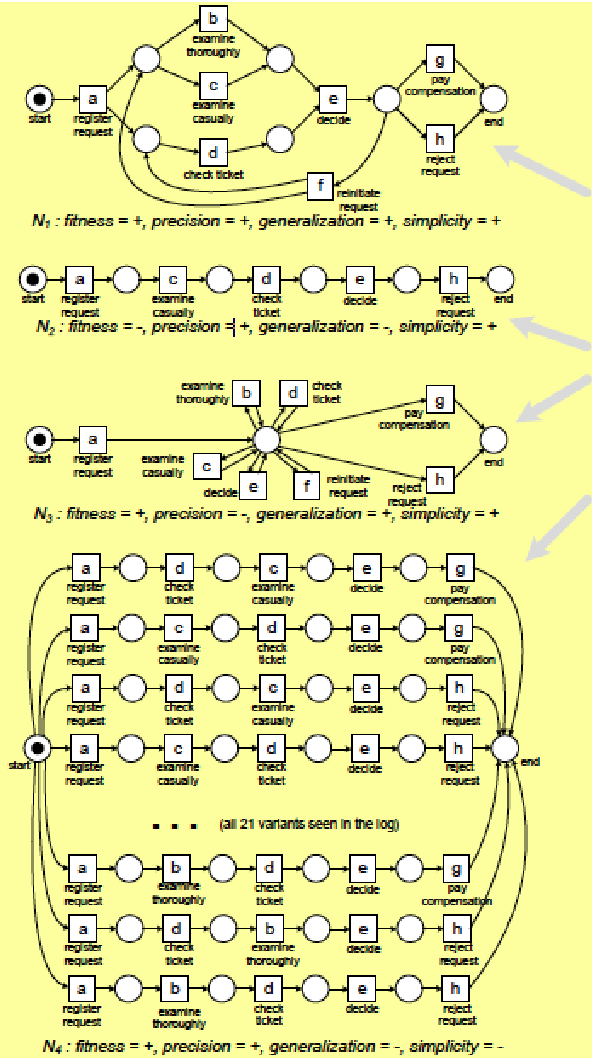
Choosing a model



#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

N3 is underfitting, i,e, lacks precision. But all traces in the log can be replayed.
The trace
<a,b,b,b,b,b,b,f,f,f,f,f,g> is possible, which is not very likely.

Choosing a model



#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

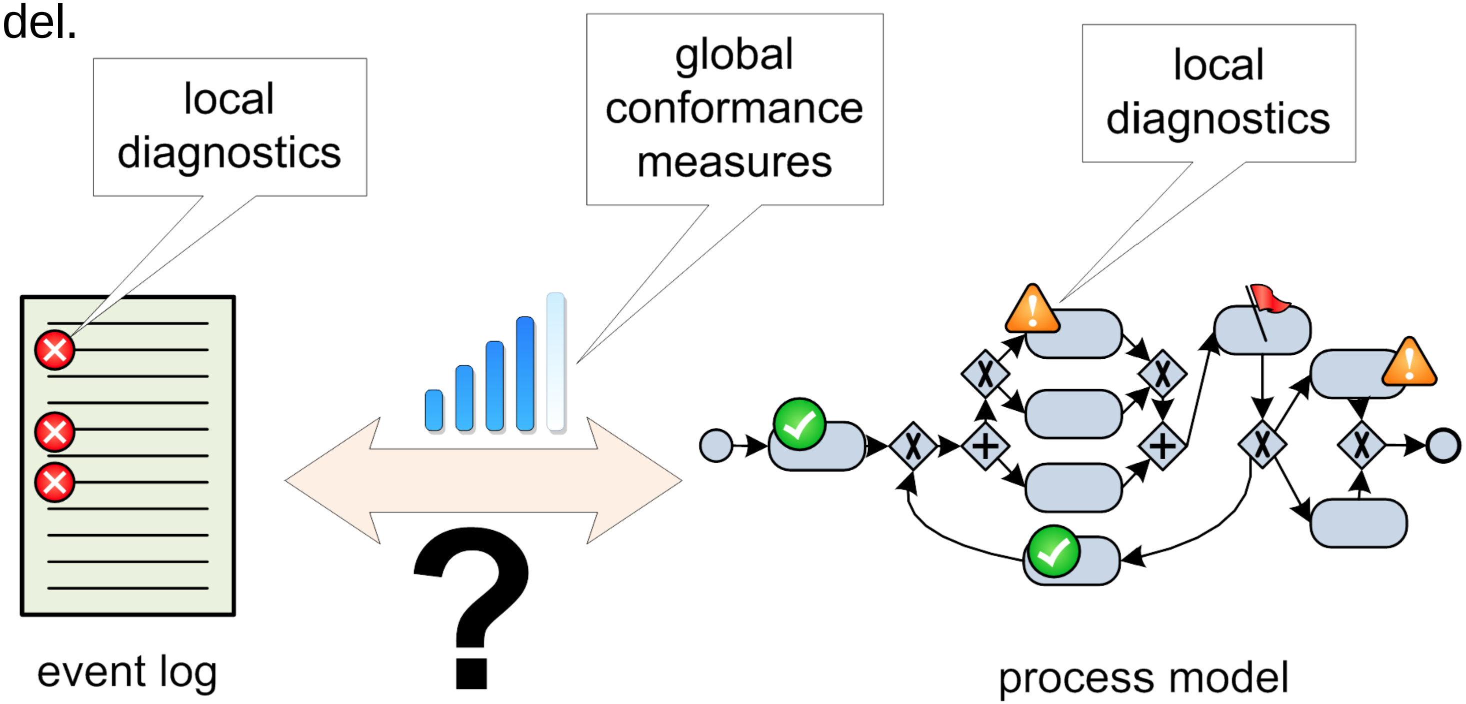
N4 enumerates all 21 traces. Is precise and has good fitness, but has overfitting (i.e., low generalization) and complex.

Outline

- Motivation. Process mining basics.
- Getting and storing event data: event logs
- Process discovery.
- **Conformance checking.**
- Online Process Mining.
- Tools.
- Conclusion.

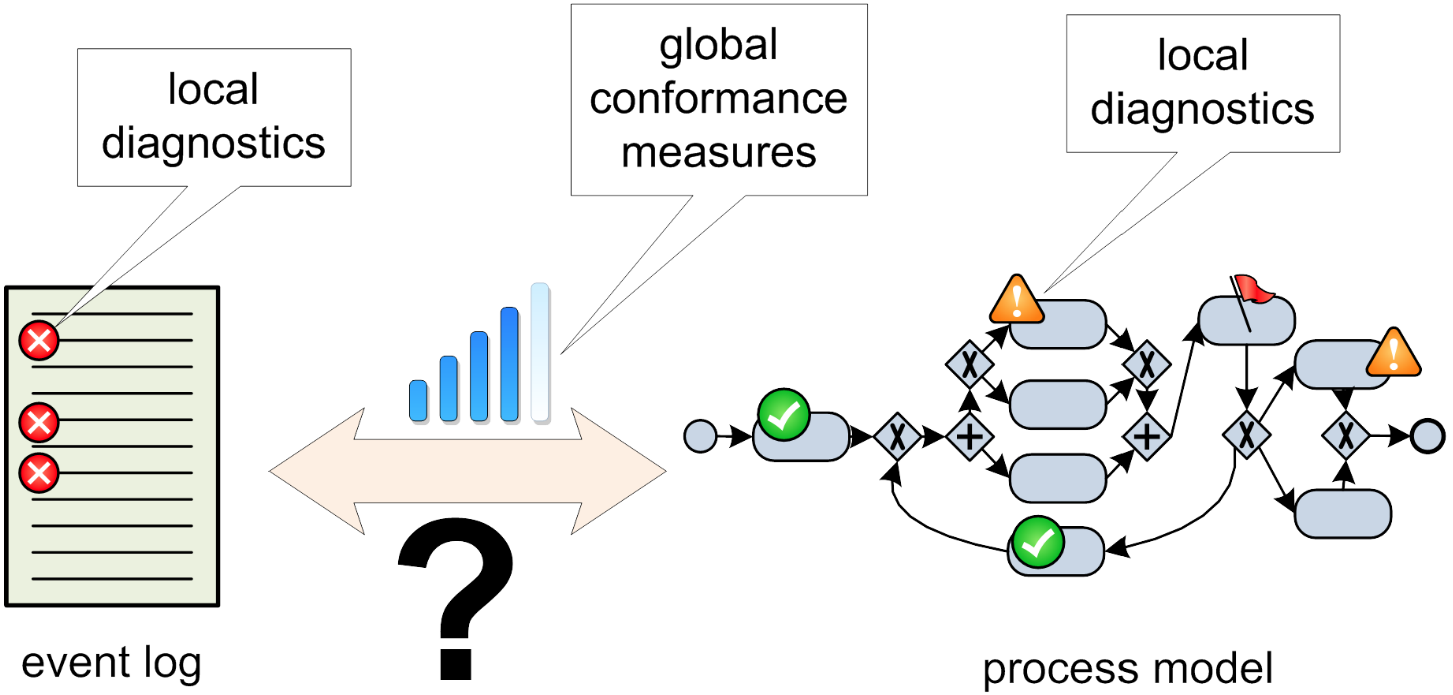
Conformance checking

Process model and log are compared to find commonalities and discrepancies.
Global conformance measures (e.g., 85% of the cases in the event log can be replayed by the model).
Local diagnostic: activity x was executed 15 times although not allowed by the model.



Conformance checking

Is the model wrong and does not reflect reality? => improve the model
Cases deviate from the model => corrective actions needed (e.g., improve control to enforce better conformance).
Sometimes deviations reflect independent behavior of stakeholders, that are good in practice.



Conformance checking

- **Recall the following four quality criteria:**
  - Fitness: the discovered model should allow for the behavior seen in the event log.
  - Precision (avoid underfitting): the discovered model should not allow for behavior completely unrelated to what was seen in the event log.
  - Generalization (avoid overfitting): the discovered model should generalize the example behavior seen in the event log.
  - Simplicity: the discovered model should be as simple as possible.

Conformance checking

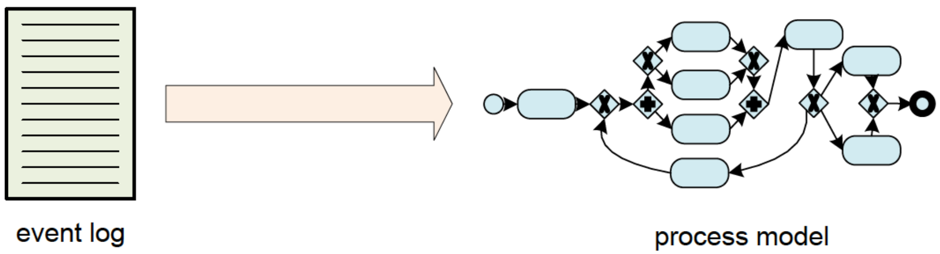
- **Recall the following four quality criteria:**
  - **Fitness: the discovered model should allow for the behavior seen in the event log.**
  - Precision (avoid underfitting): the discovered model should not allow for behavior completely unrelated to what was seen in the event log.
  - Generalization (avoid overfitting): the discovered model should generalize the example behavior seen in the event log.
  - Simplicity: the discovered model should be as simple as possible.

Conformance checking

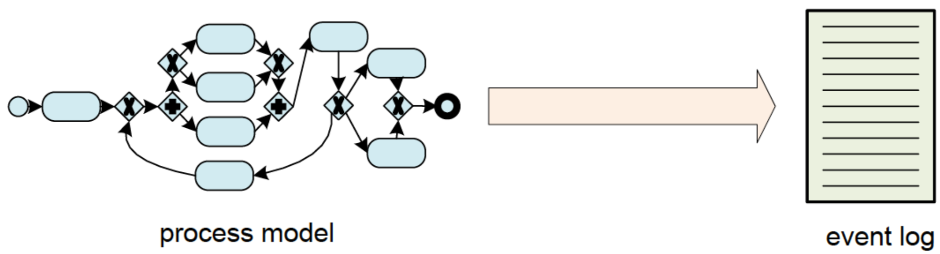
- **Recall the following four quality criteria:**
  - **Fitness: the discovered model should allow for the behavior seen in the event log.**
  - **This is the criteria related to conformance checking.**

Conformance checking

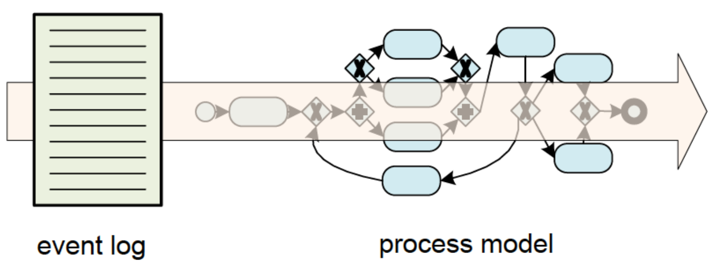
Play-In



Play-Out



Replay



Uses:

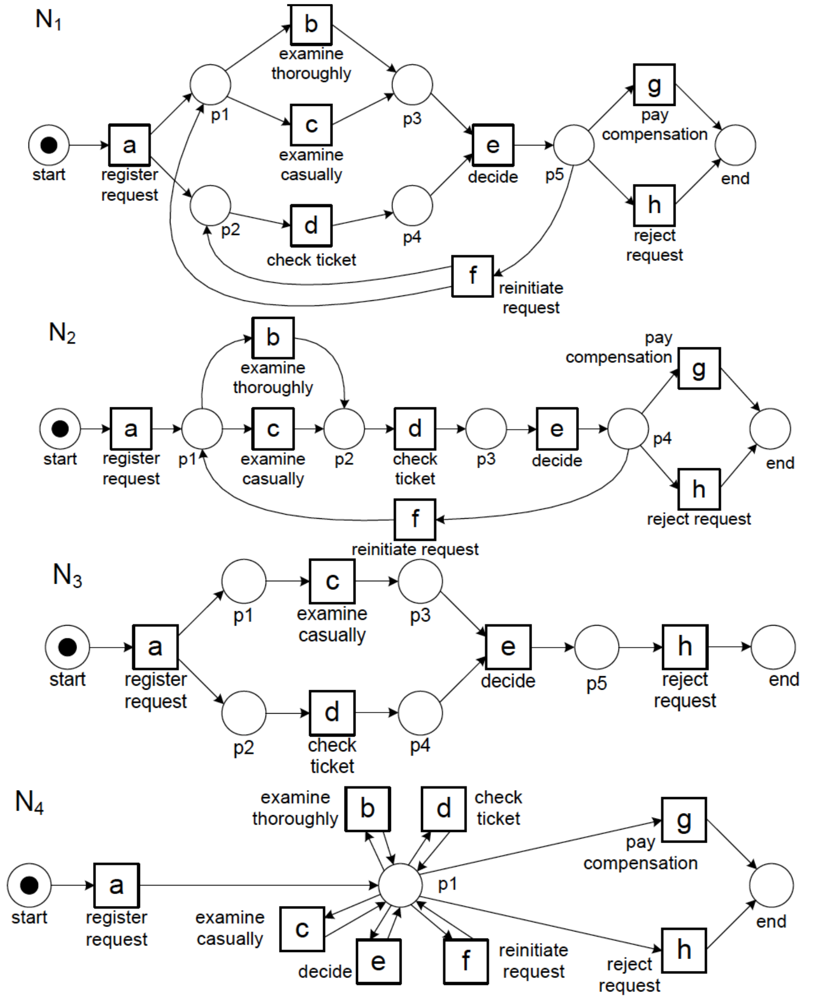
- extended model showing times, frequencies, etc.
- diagnostics
- predictions
- recommendations

Play-in: given a log, discovers a model.

Play out: given a model, generates (e.g., simulates) a log. i.e., repeatedly runs the model.

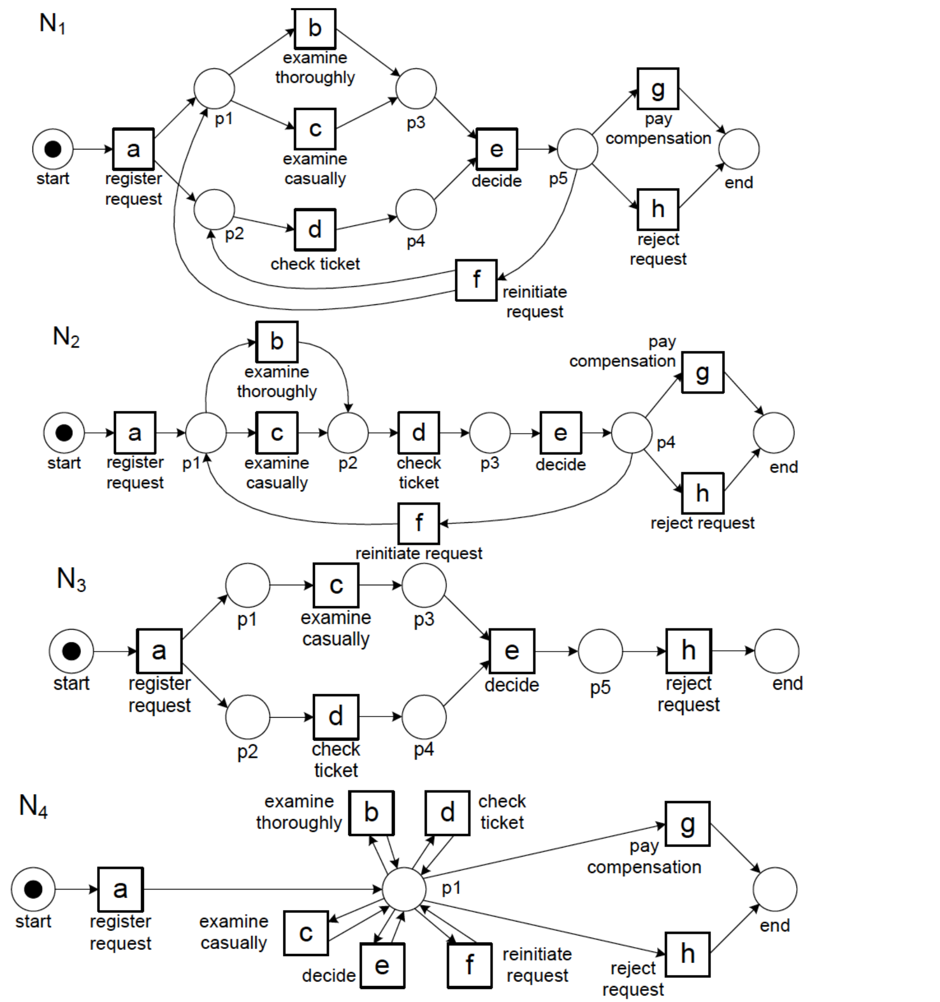
Replay: given a log AND a process model, the event log is replayed over the model. Used in conformance checking

Conformance checking



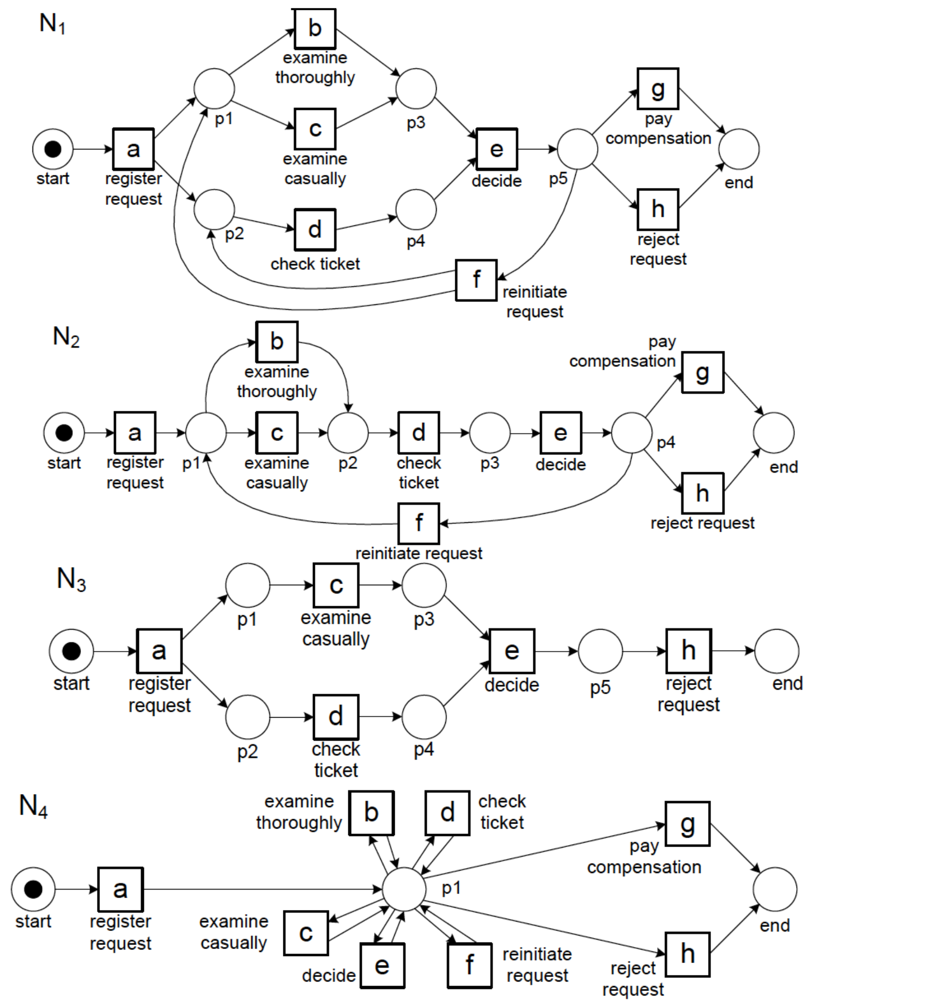
frequency	reference	trace
455	σ_1	$\langle a,c,d,e,h\rangle$
191	σ_2	$\langle a,b,d,e,g\rangle$
177	σ_3	$\langle a,d,c,e,h\rangle$
144	σ_4	$\langle a,b,d,e,h\rangle$
111	σ_5	$\langle a,c,d,e,g\rangle$
82	σ_6	$\langle a,d,c,e,g\rangle$
56	σ_7	$\langle a,d,b,e,h\rangle$
47	σ_8	$\langle a,c,d,e,f,d,b,e,h\rangle$
38	σ_9	$\langle a,d,b,e,g\rangle$
33	σ_{10}	$\langle a,c,d,e,f,b,d,e,h\rangle$
14	σ_{11}	$\langle a,c,d,e,f,b,d,e,g\rangle$
11	σ_{12}	$\langle a,c,d,e,f,d,b,e,g\rangle$
9	σ_{13}	$\langle a,d,c,e,f,c,d,e,h\rangle$
8	σ_{14}	$\langle a,d,c,e,f,d,b,e,h\rangle$
5	σ_{15}	$\langle a,d,c,e,f,b,d,e,g\rangle$
3	σ_{16}	$\langle a,c,d,e,f,b,d,e,f,d,b,e,g\rangle$
2	σ_{17}	$\langle a,d,c,e,f,d,b,e,g\rangle$
2	σ_{18}	$\langle a,d,c,e,f,b,d,e,f,b,d,e,g\rangle$
1	σ_{19}	$\langle a,d,c,e,f,d,b,e,f,b,d,e,h\rangle$
1	σ_{20}	$\langle a,d,b,e,f,b,d,e,f,d,b,e,g\rangle$
1	σ_{21}	$\langle a,d,c,e,f,d,b,e,f,c,d,e,f,d,b,e,g\rangle$

Conformance checking



N1: results from applying the α algorithm to L. Can replay all the log.
N2: a sequential model. Requires *b* or *c* to complete before d can be triggered. <*a,d,c,e,h*> is not possible.
N3: has no choices, the request is always rejected (i.e., cannot replay successful cases).
Ex: <*a,b,d,e,g*> is not possible.
N4: “flower model”: can replay all traces and many more.

Conformance checking



Naïve approach: measure conformance checking counting the % of cases that can be replayed. Thus:

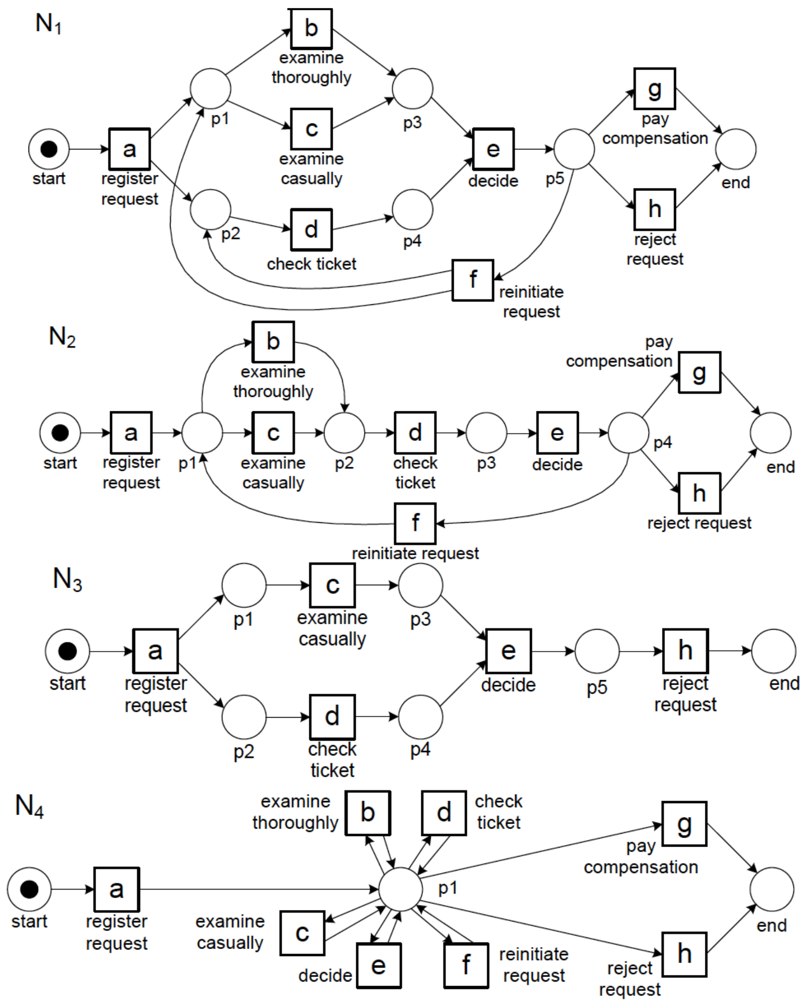
Fitness(N1) = 1

Fitness(N2) = 948/1391= 0.6815

Fitness(N3) = 632/1391= 0.4543

Fitness(N4) = 1

Conformance checking



This approach is not good. Does not consider traces that “almost” fit, e.g., that can replay 95% of the activities in a trace => Try using fitness measures at the level of events and NOT at the level of traces.

Idea: instead of stopping when finding a problem, and marking the trace as non-fitting, continue, forcing the transition to be enabled (e.g., adding a token). Then, count the number of missing and remaining tokens.

Conformance checking

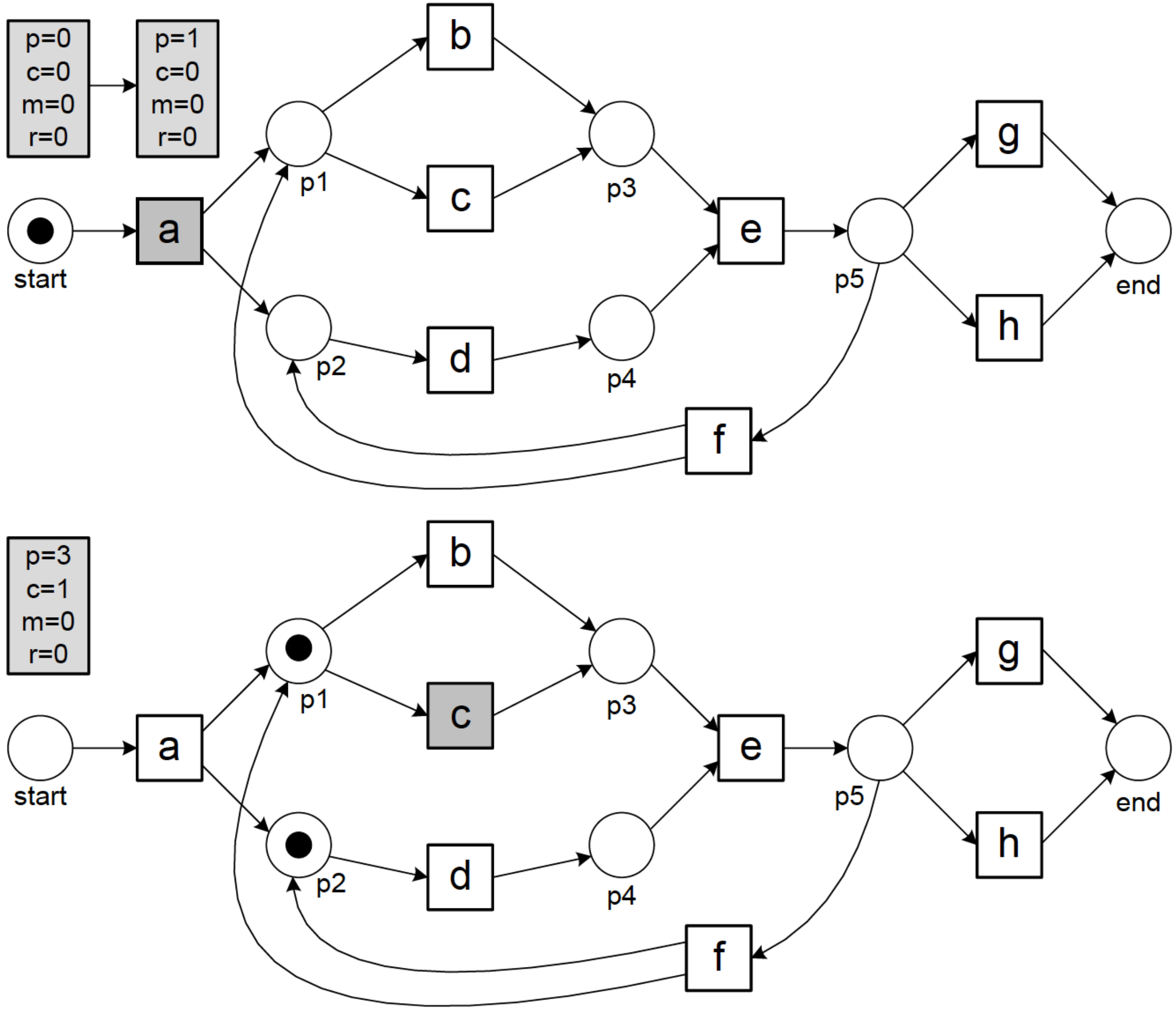
Let us replay
<a,c,d,e,h> on N1.

p = # of tokens produced.

c = # of tokens consumed.

m = # of missing tokens.

p = # of remaining tokens.



Conformance checking

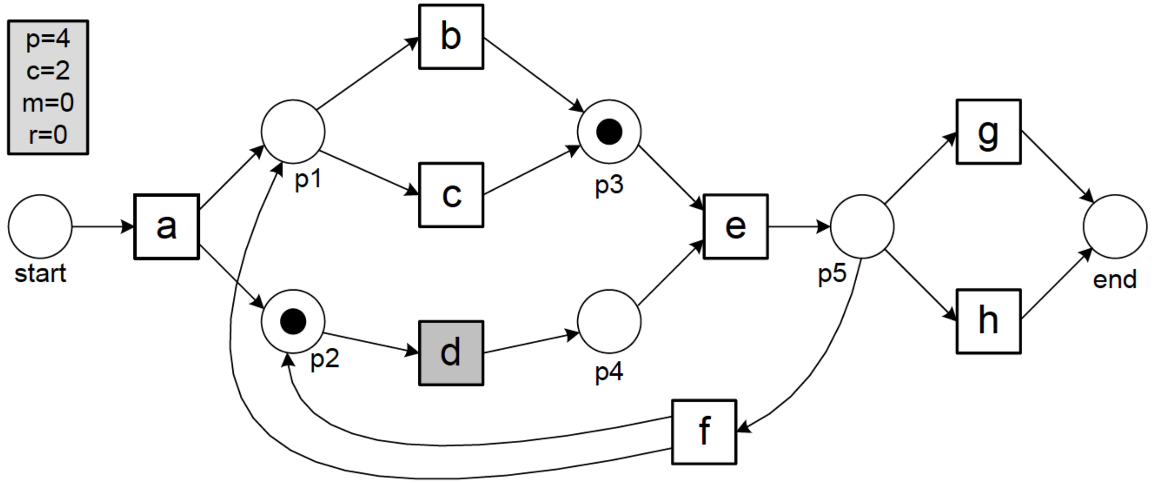
Let us replay <a,c,d,e,h> on N1.

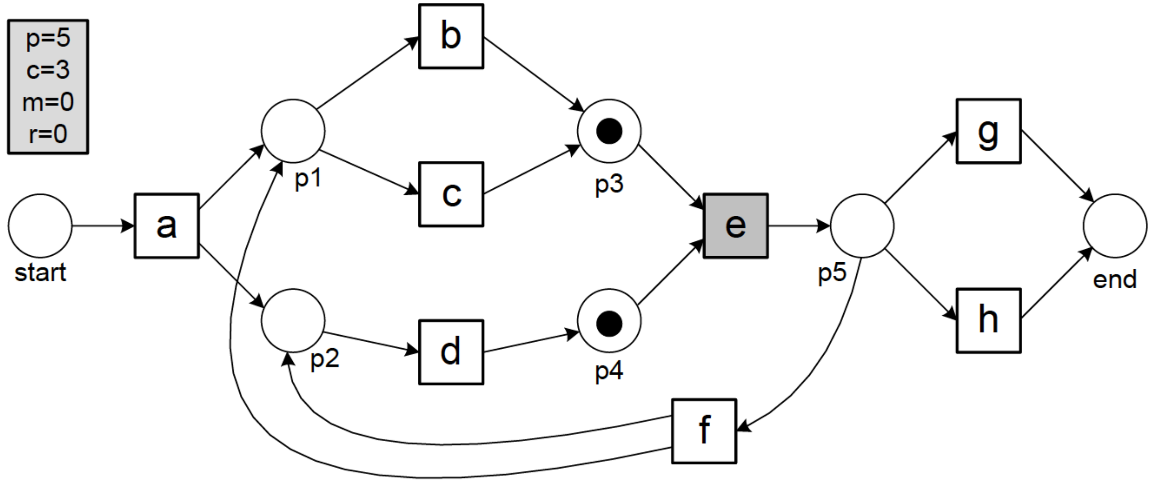
p = # of tokens produced.

c = # of tokens consumed.

m = # of missing tokens.

p = # of remaining tokens.

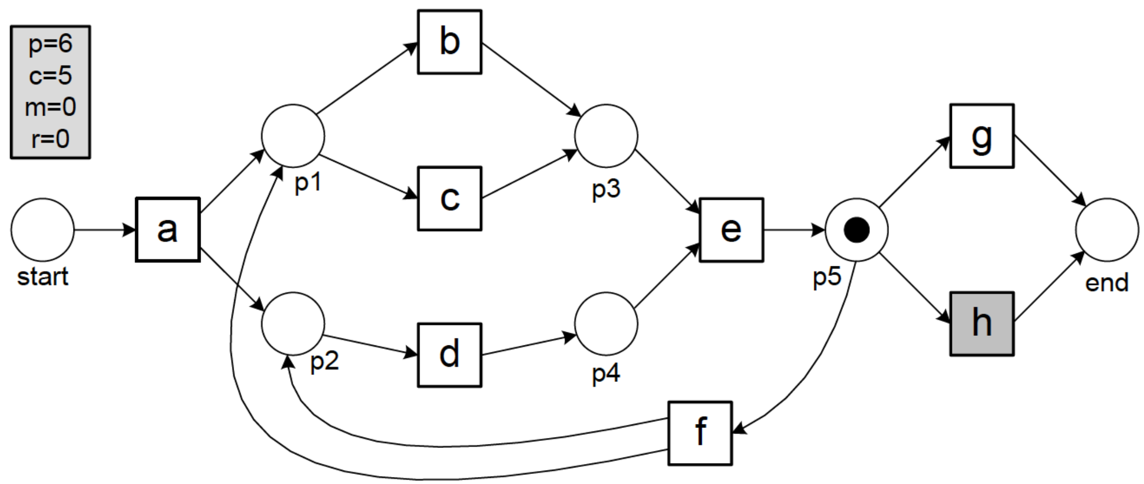


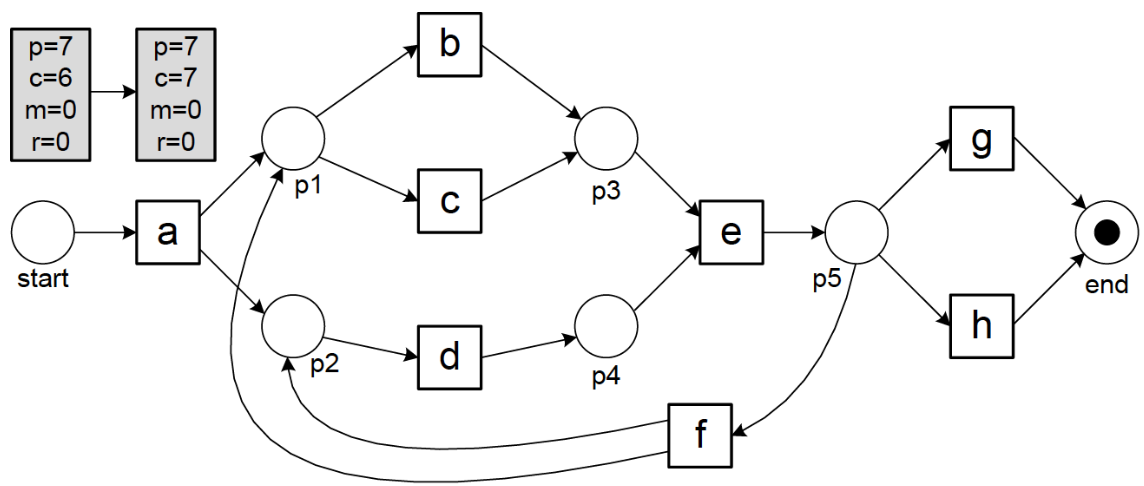


Conformance checking

Let us replay
<a,c,d,e,h> on N1.

The trace was
replayed correctly.
p = c = 7; m= r= 0





Conformance checking

Fitness is defined as:

$$\text{Fitness}(t, N) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

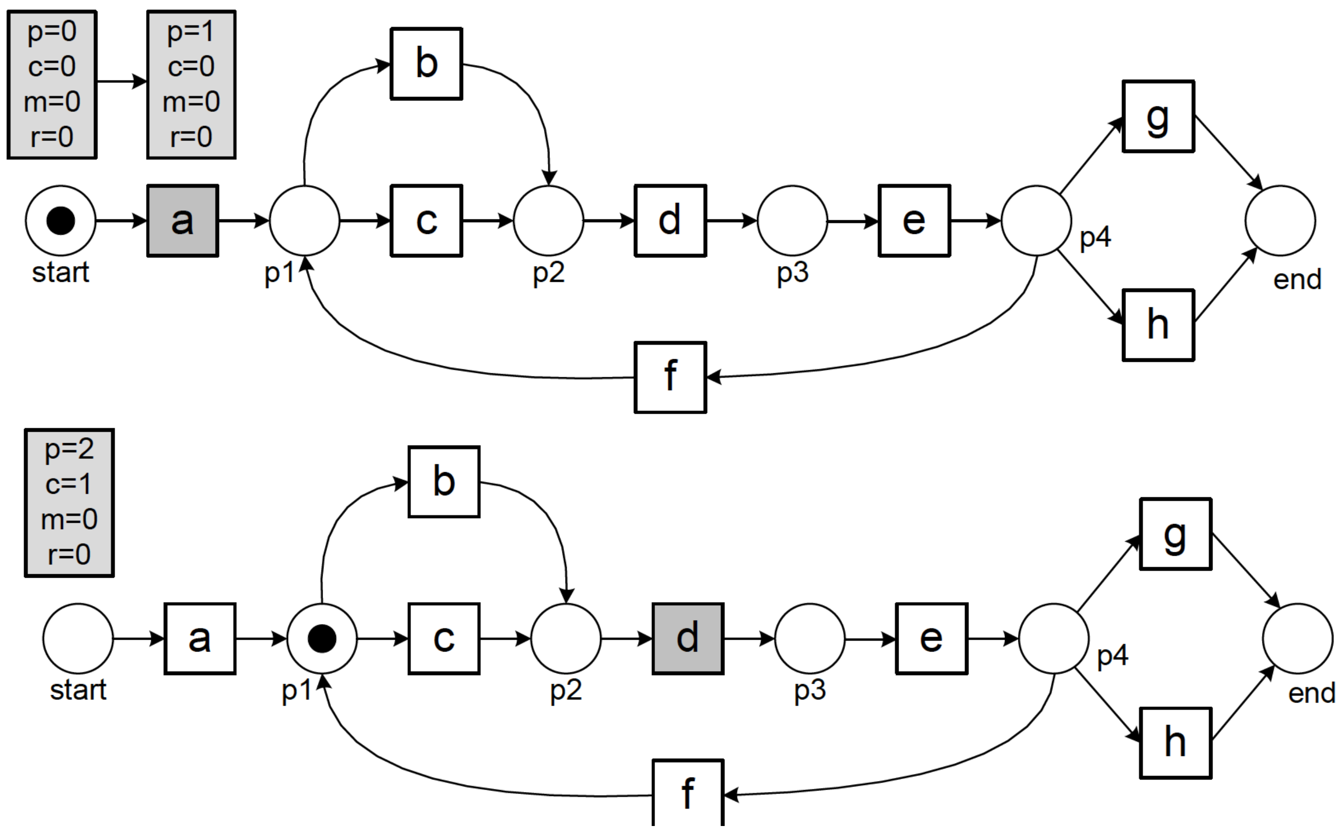
The first part computes the fraction of missing tokens relative to the number of consumed tokens.

The second part computes the fraction of remaining tokens relative to the number of produced tokens.

If there are neither missing nor remaining tokens, fitness = 1.

Conformance checking

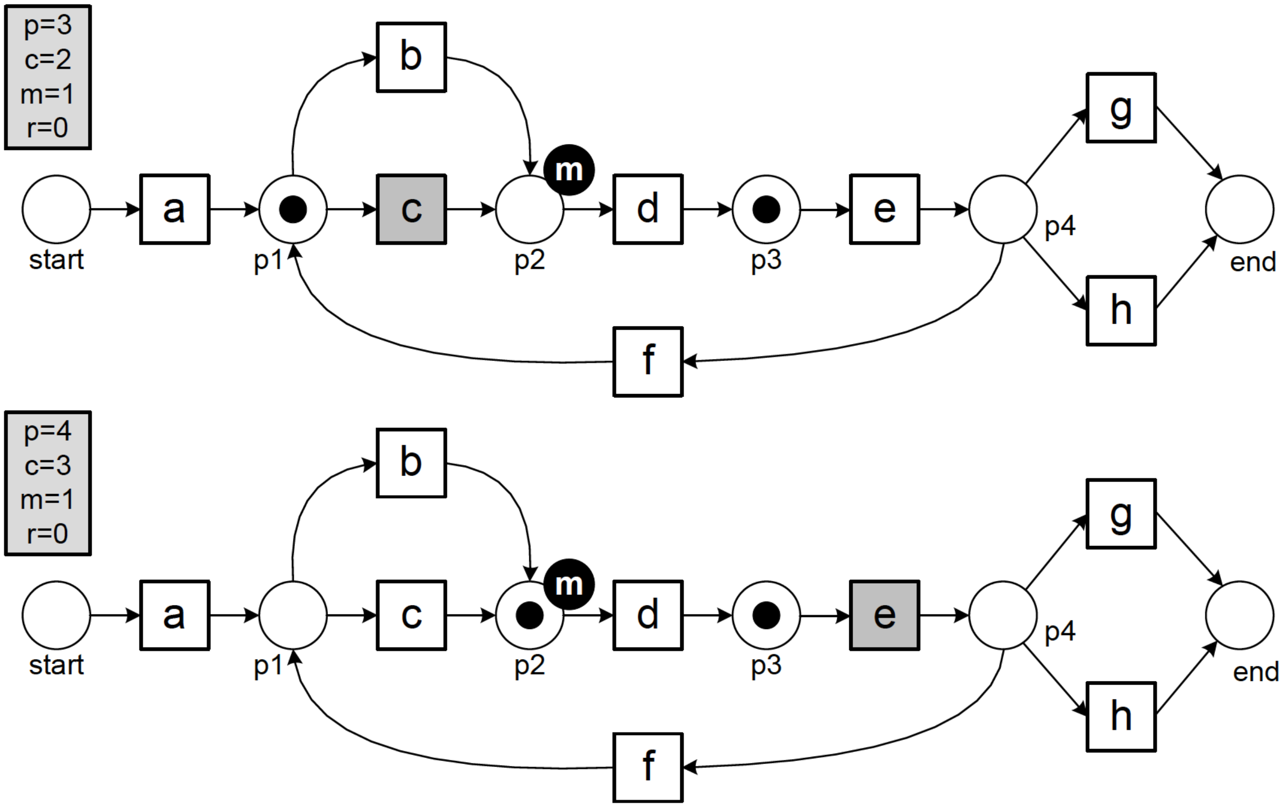
Let us replay <a,d,c,e,h> on N2.



Conformance checking

Let us replay <a,d,c,e,h> on N2.

d cannot be enabled, before *c* is. Then, we add a token at p2 and mark it as missing (m). Then, *d* can be fired.

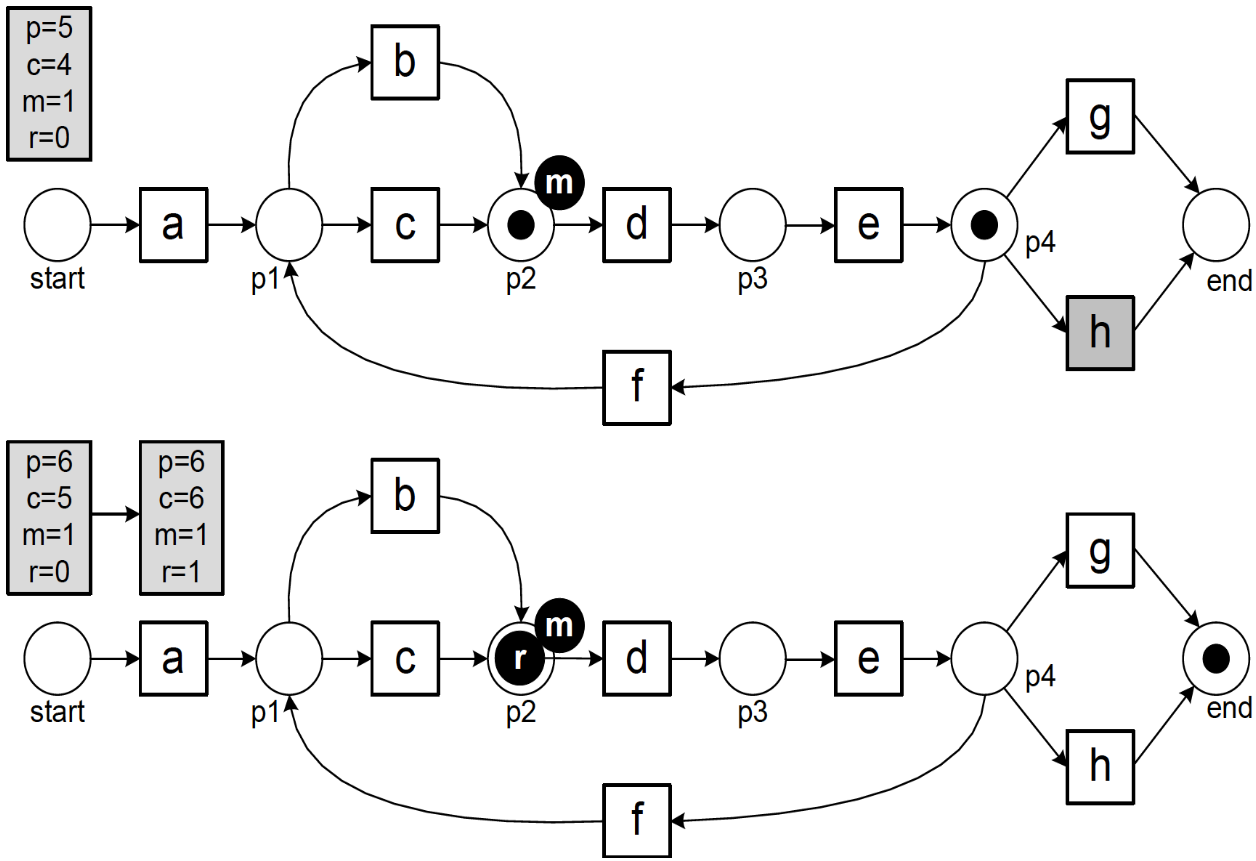


Conformance checking

Let us replay <a,d,c,e,h> on N_2.

d cannot be enabled, before *c* is. Then, we add a token at p2 and mark it as missing (m). Then, *d* can be fired.

Fitness (t, N_2) = ½ (1 – 1/6 + ½ (1 – 1/6) = 0.8333

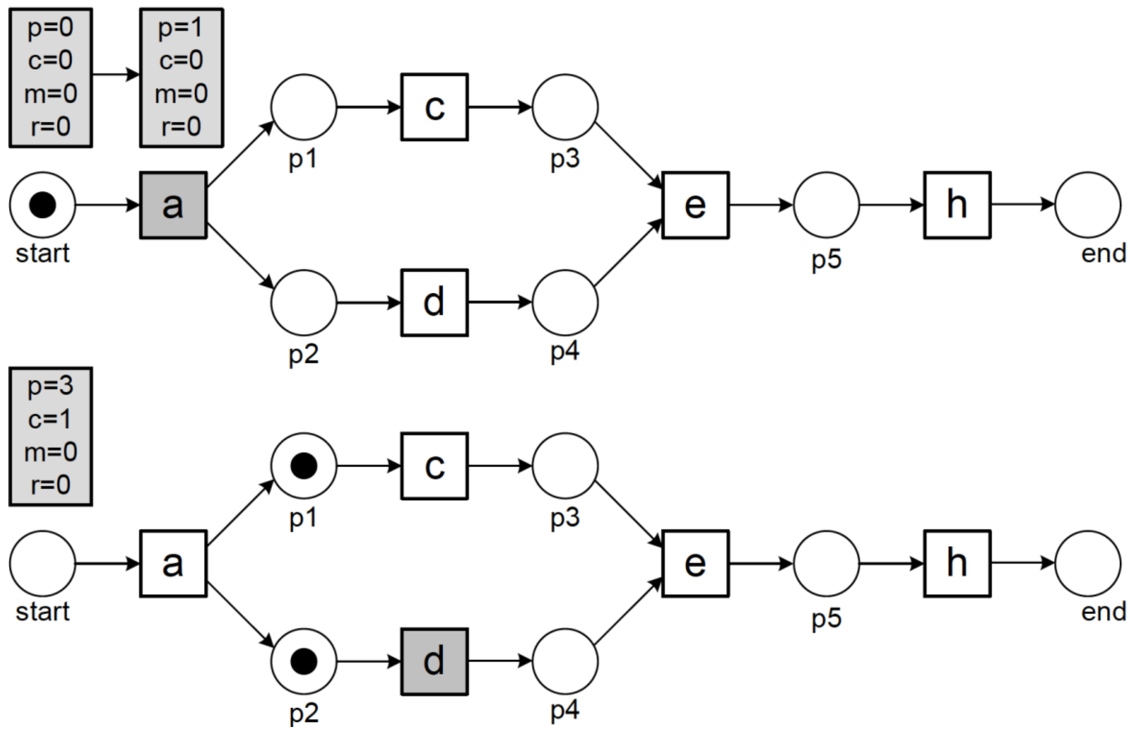


Conformance checking

Let us replay

<a,b,d,e,g> on N_3.

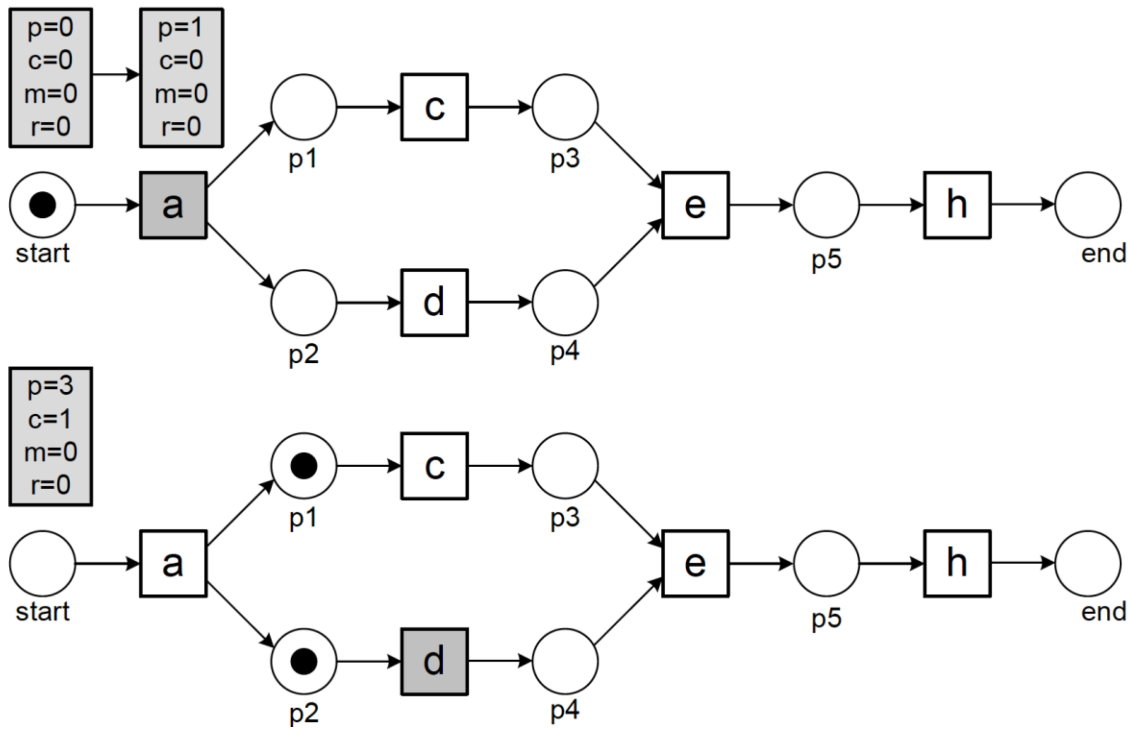
(the model does not contain all the activities in the log =>we only consider the trace <a,d,e>).



Conformance checking

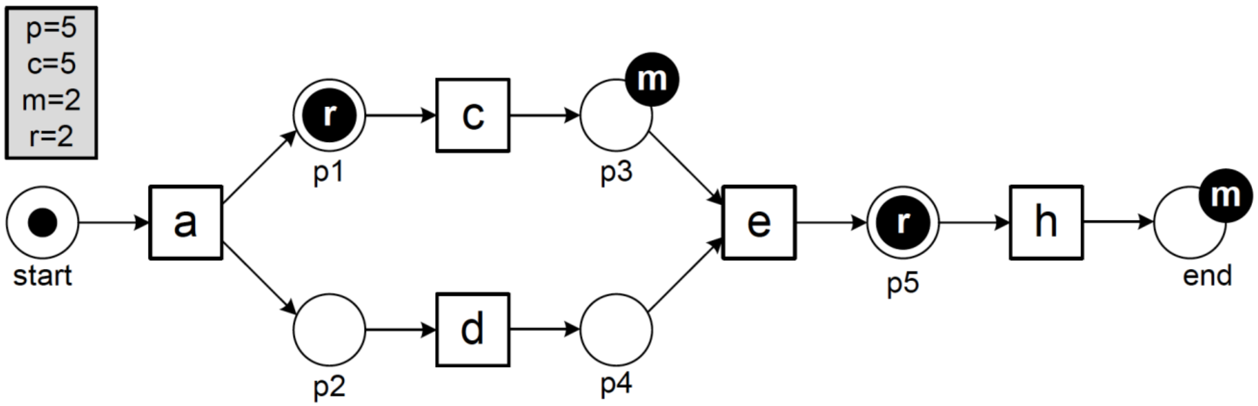
= > Let us replay
<a,d,e> on N_3.

Since *c* has not
been fired, *e*
cannot be fired
=> We add a token at
p3 and
mark it as missing.



Conformance checking

= > Let us replay
<a,d,e> on N_3.



Since *h* cannot be fired, we need to produce a token at the final State, and mark it as m.

Fitness (t, N_3) = ½ (1 – 2/5 + ½ (1 – 2/5) = 0.6.

Conformance checking

Computing fitness at log level yields:

$$fitness(L,N) = \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times m_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times c_{N,\sigma}}\right) + \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times r_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times p_{N,\sigma}}\right)$$

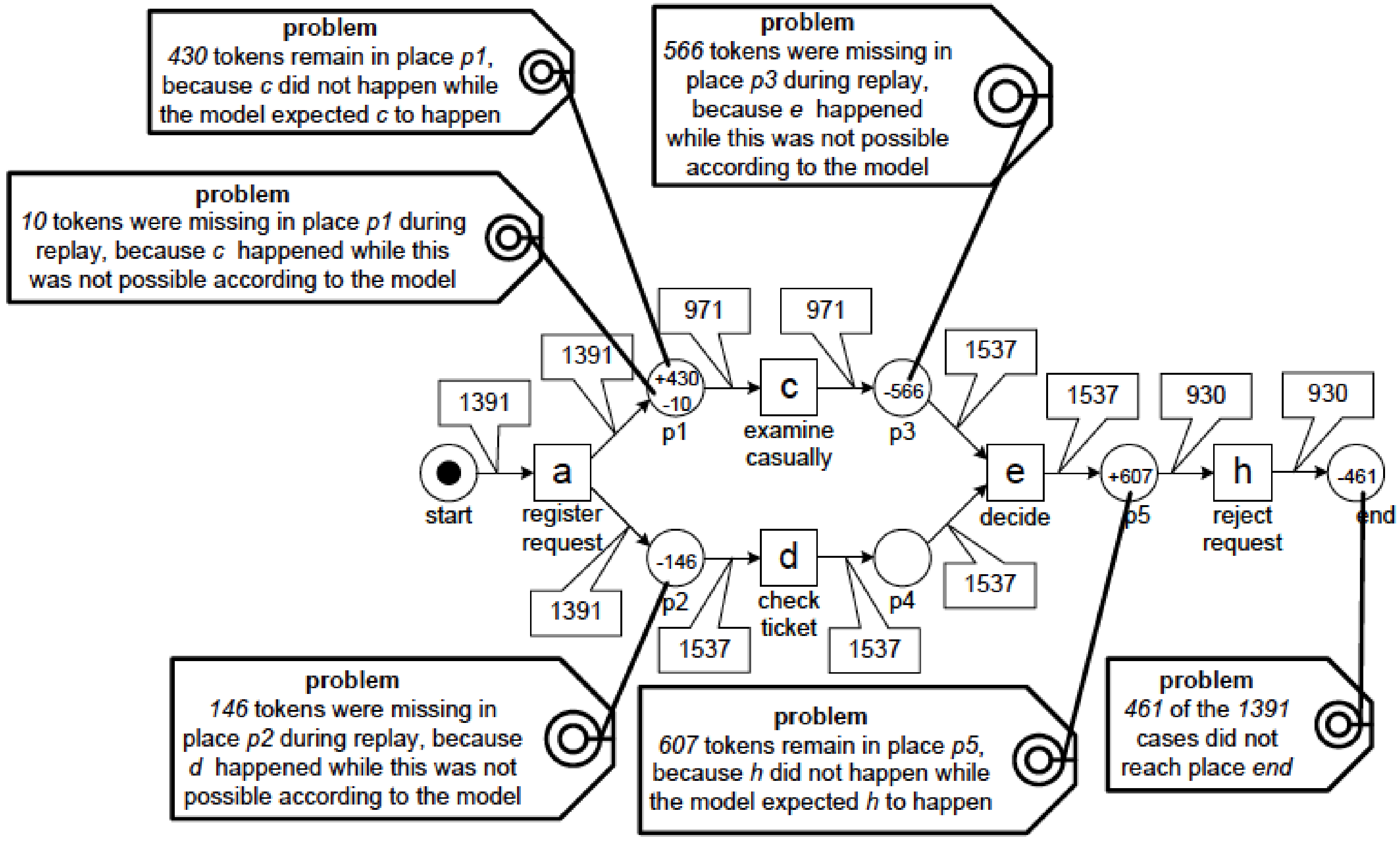
$m_{N,\sigma}$ =# of missing tokens when replaying σ on N.
$p_{N,\sigma}$ =# of produced tokens when replaying σ on N…etc.
L(σ) = frequency of trace σ.

In our example, if we play the entire *event log* we obtain:

Fitness (L, N_1) = 1
Fitness (L, N_2) = 0.9504
Fitness (L, N_3) = 0.8797
Fitness (L, N_4) = 1

Conformance checking: diagnosis example

Fitness (L, N_3) = 0.8797

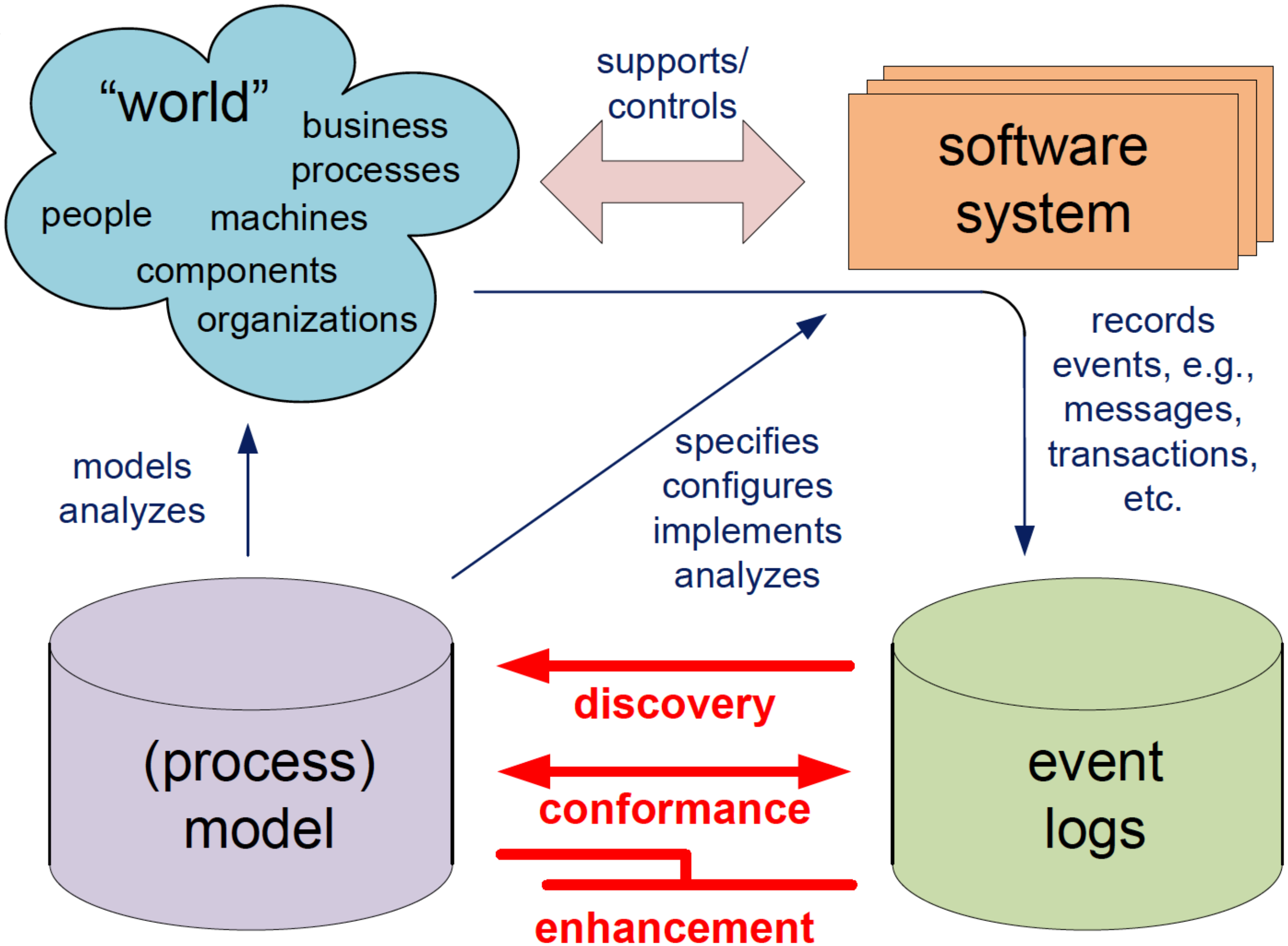


Outline

- Motivation. Process mining basics.
- Getting event data.
- Process discovery.
- Conformance checking.
- **Online Process Mining.**
- Tools.
- Conclusion.

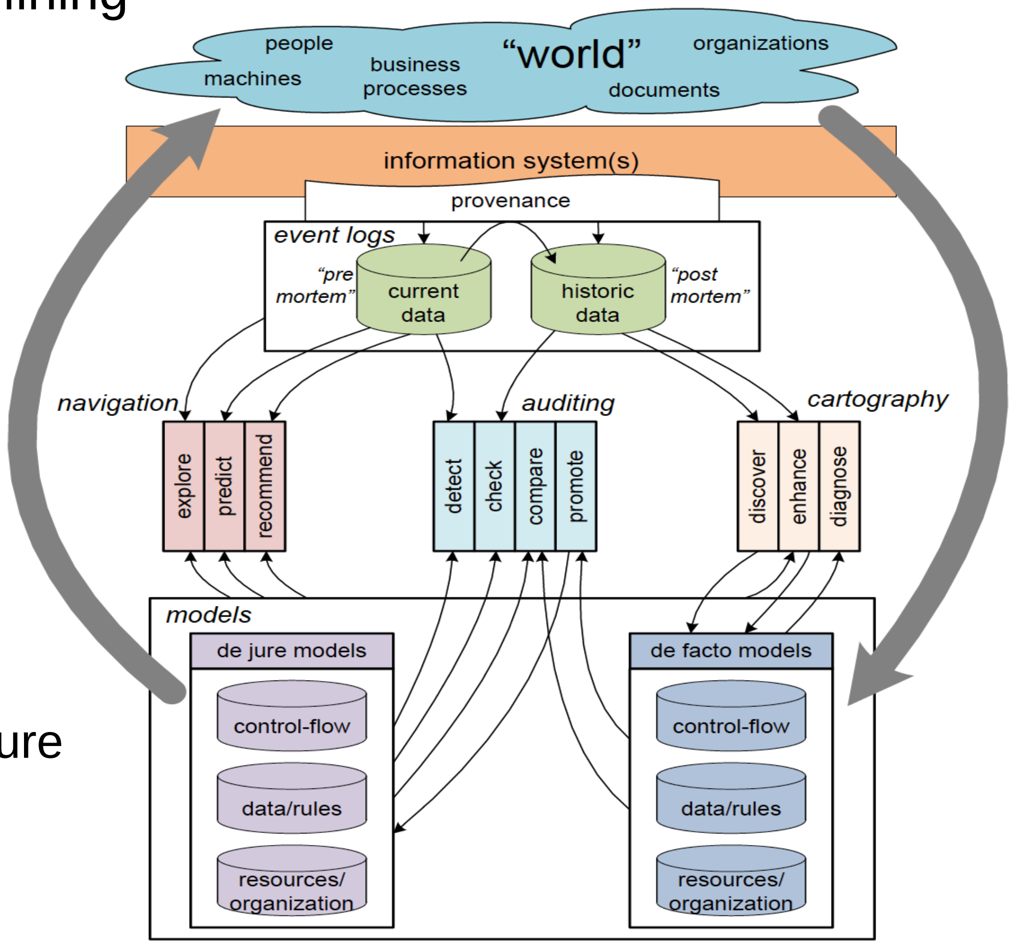
Online decision support

Basic Process Mining Framework



Online decision support

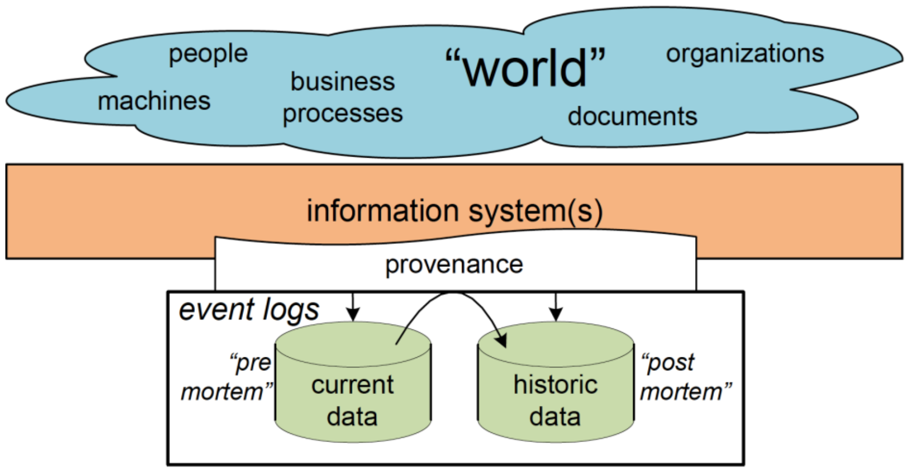
Refined Process Mining Framework



We next study this figure in detail.

Online decision support

Business Process Provenance: refers to a set of activities needed to ensure that history in event logs can serve as a reliable basis for process improvement and auditing.



"Post-mortem" event data refers to *completed cases* => data that can be used for auditing or improvement

"Pre-mortem" event data refers to cases that have *not yet been completed* => it may be possible that information in the event log about this case (i.e., current data) can be exploited to ensure the correct and/or efficient handling of this case.

Online decision support

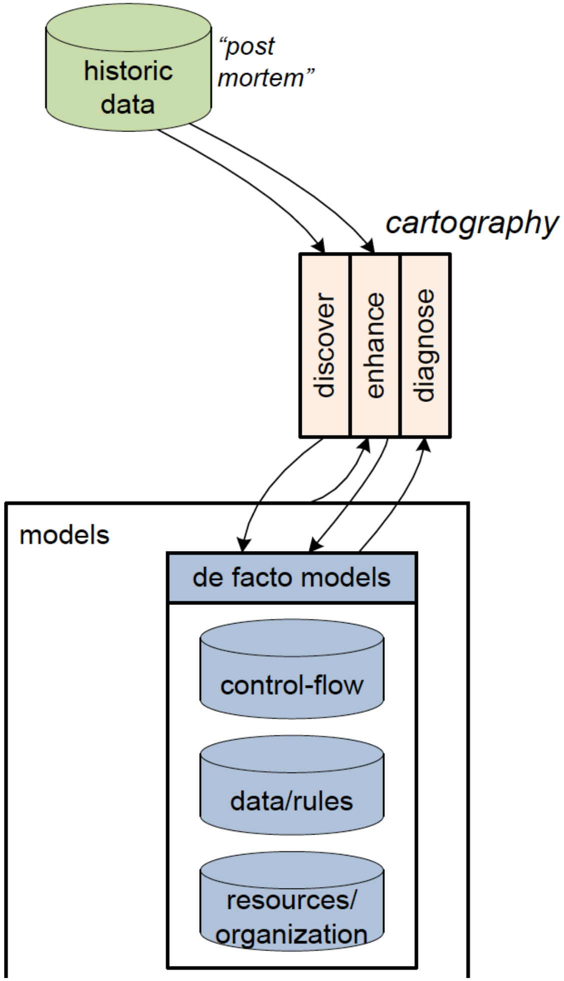
- **De jure models**: **normative**, i.e., it specifies how things should be done or handled. For example, a process model used to configure a BPM system is normative and forces people to work in a particular way.

- **De facto models**: **descriptive.** The goal is not to steer or control reality. Instead, de facto models aim at capturing reality.

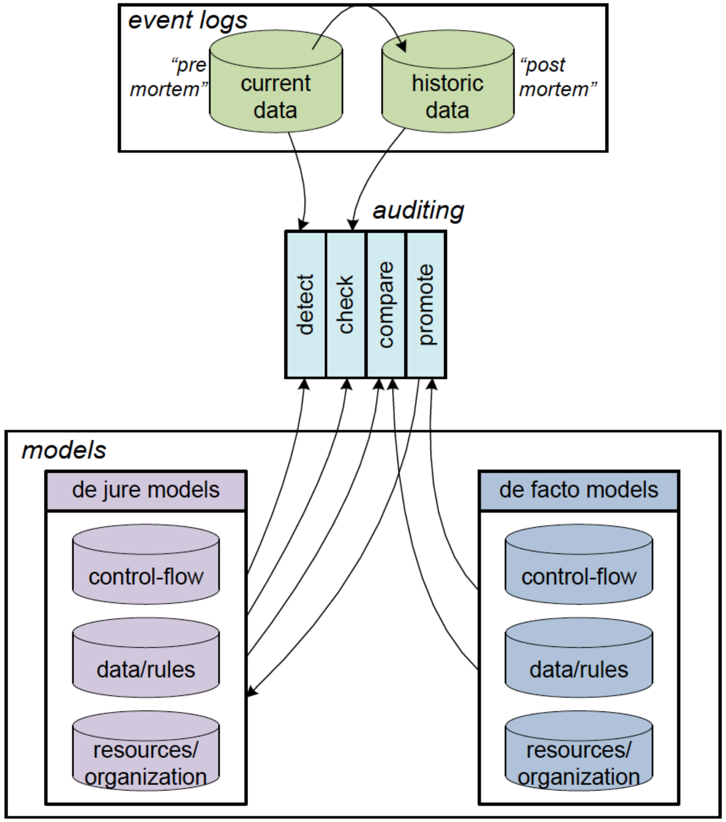
Online decision support

Cartography: three main activities over de facto models and post-mortem data. Like maps, aimed at describing reality.

- **Discover:** refers to the extraction of (process) models.
- **Enhance:** existing process models (either discovered or hand-made) can be related to event logs, it is possible to enhance these models.
- **Diagnose:** not direct use of event logs. Focuses on classical model-based analysis.



Online decision support



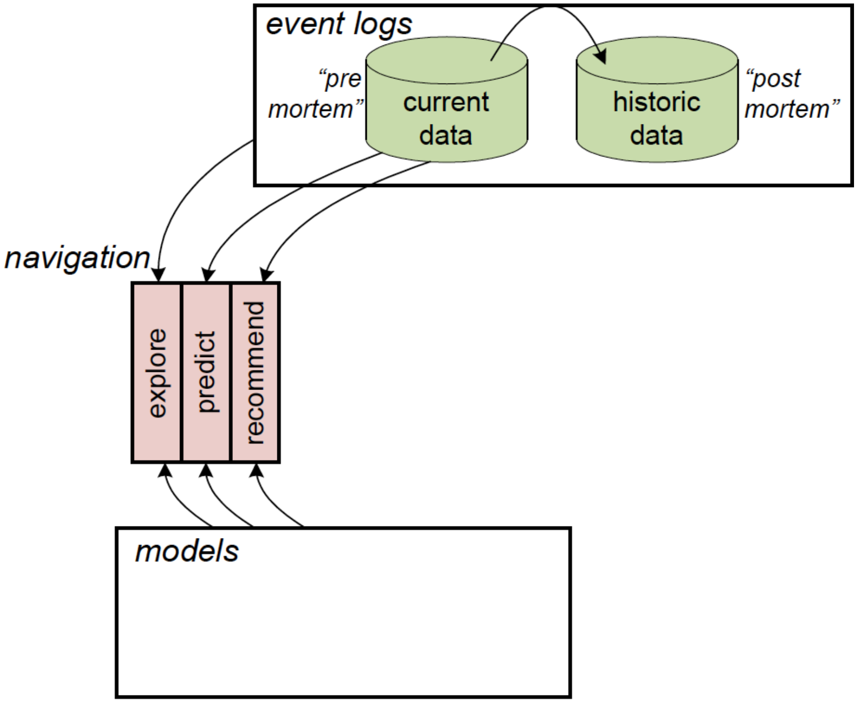
Auditing: set of activities that check if BPs are executed within certain boundaries.

- **Detect:** Compares de jure models with "pre mortem" data. When a predefined rule is violated, an alert is issued **online**.
- **Check**: finds deviations and quantifies the level of compliance **(offline)**.
- **Compare:** compares de facto models against de jure models to see in what way reality deviates from what was planned or expected.
- **Promote:** based on the comparison above, parts of the de facto model can be promoted to a new de jure model.

Online decision support

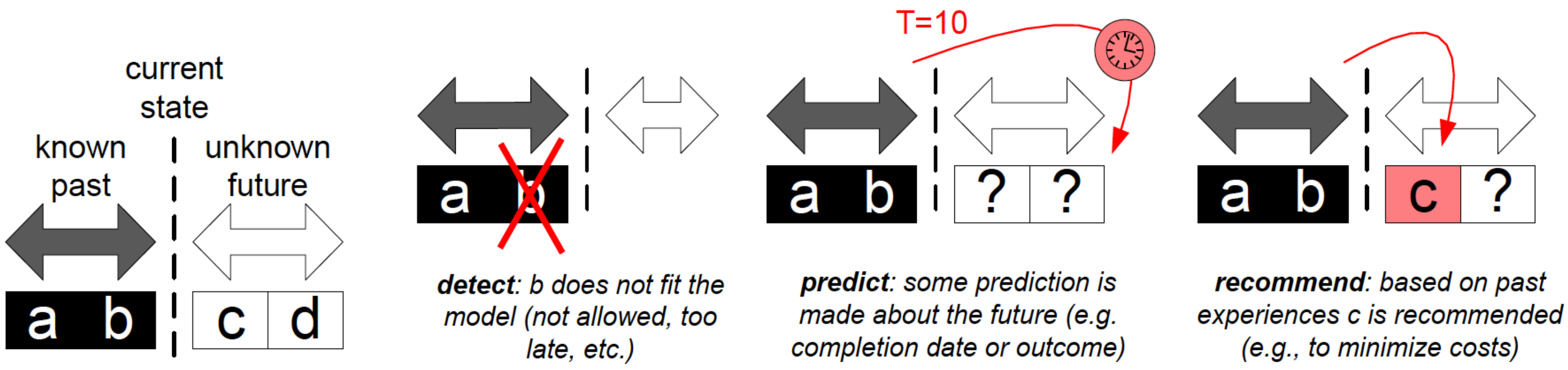
Navigation: set of activities that are forward-looking. Make predictions about the future.

- **Explore:** combination of event data and models to explore business processes at run-time. Compares running cases with similar ones handled earlier.
- **Predict:** combining information about running cases with models, predicts the future o a case, e.g., the remaining flow time and the probability of success.
- **Recommend:** information used for predicting the future can also be used to recommend suitable actions (e.g., to minimize costs or time).



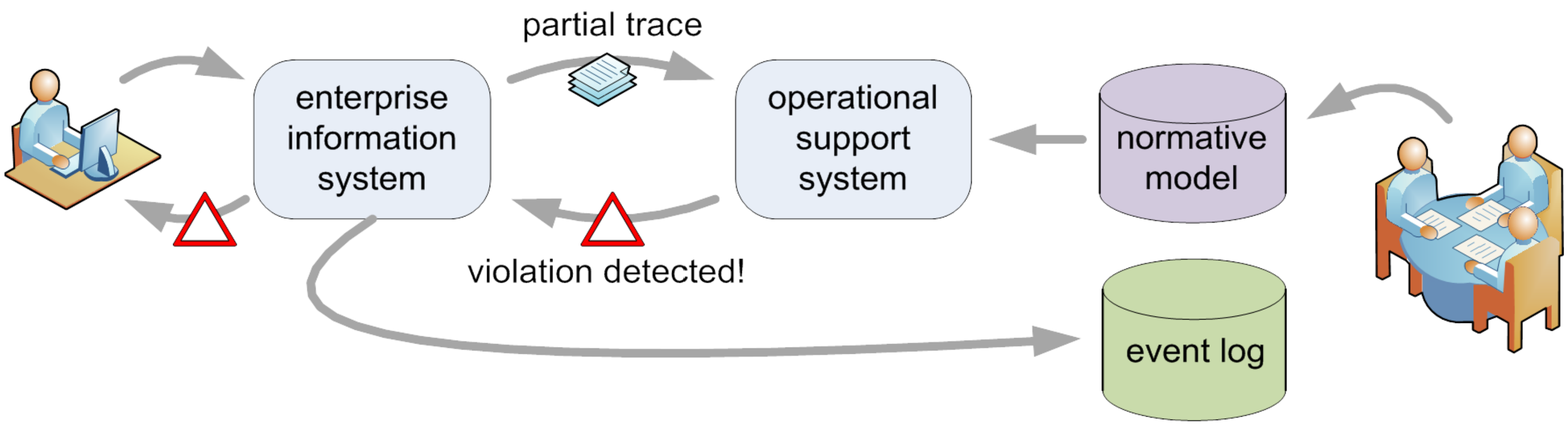
Online decision support

Traditionally**:** process mining was an off-line activity, using "post-mortem" data. Only completed cases were considered. Online process mining uses "pre-mortem" data. Running cases are analyzed, and OPM aimes at influencing running processes. A running case is associated to a ***partial trace***, and three activities: *detect, predict and recommend.*



Online decision support

Detect.



Online decision support

Detect.

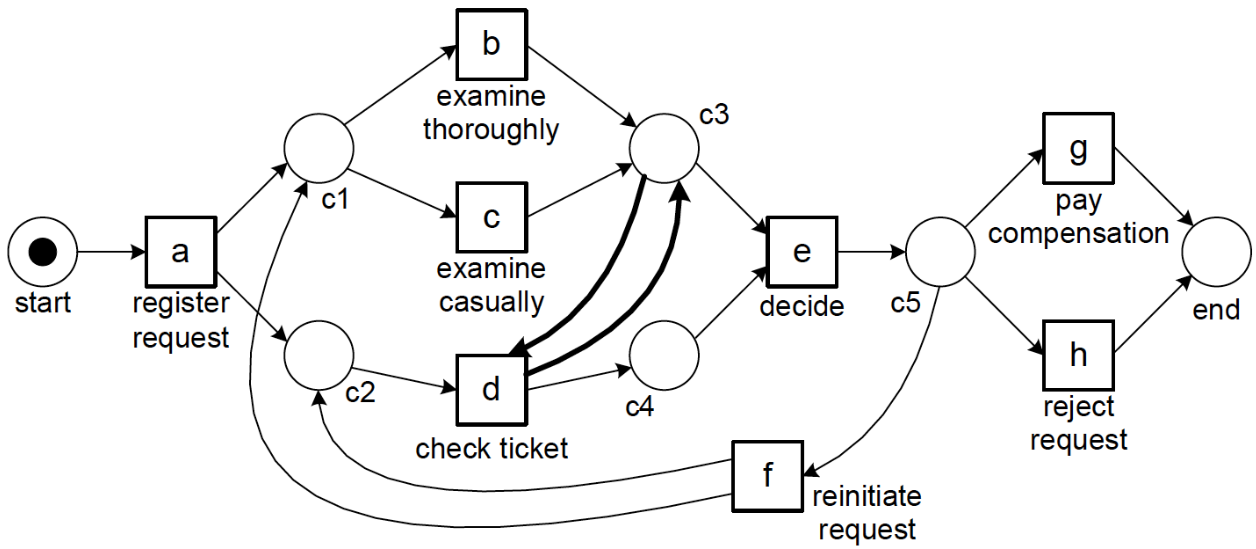
Detecting deviations at runtime. Two differences with conformance checking: (a) only considers the partial trace of a case, instead of the whole log; (b) if there is a deviation, we give immediate response.

Online decision support

Detect.

Detecting deviations at runtime. Two differences with conformance checking: (a) only considers the partial trace of a case, instead of the whole log; (b) if there is a deviation, we give immediate response.

Consider the model…

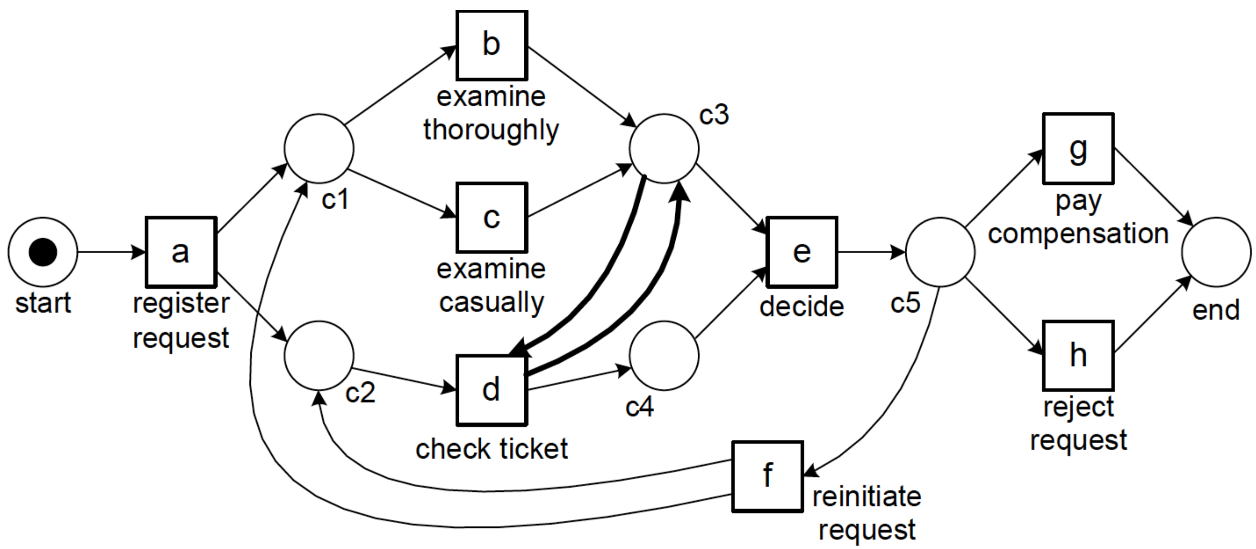


Online decision support

Detect.

…. the transactional view of the current log, and three cases.

case id	trace
1	$\langle a^{12}_{start}, a^{19}_{complete}, b^{25}_{start}, d^{26}_{start}, b^{32}_{complete}, d^{33}_{complete}, e^{35}_{start}, e^{40}_{complete}, h^{50}_{start}, h^{54}_{complete} \rangle$
2	$\langle a^{17}_{start}, a^{23}_{complete}, d^{28}_{start}, c^{30}_{start}, d^{32}_{complete}, c^{38}_{complete}, e^{50}_{start}, e^{59}_{complete}, g^{70}_{start}, g^{73}_{complete} \rangle$
3	$\langle a^{25}_{start}, a^{30}_{complete}, c^{32}_{start}, c^{35}_{complete}, d^{35}_{start}, d^{40}_{complete}, e^{45}_{start}, e^{50}_{complete}, f^{50}_{start}, f^{55}_{complete}, b^{60}_{start}, d^{62}_{start}, b^{65}_{complete}, d^{67}_{complete}, e^{80}_{start}, e^{87}_{complete}, g^{90}_{start}, g^{98}_{complete} \rangle$
...	...

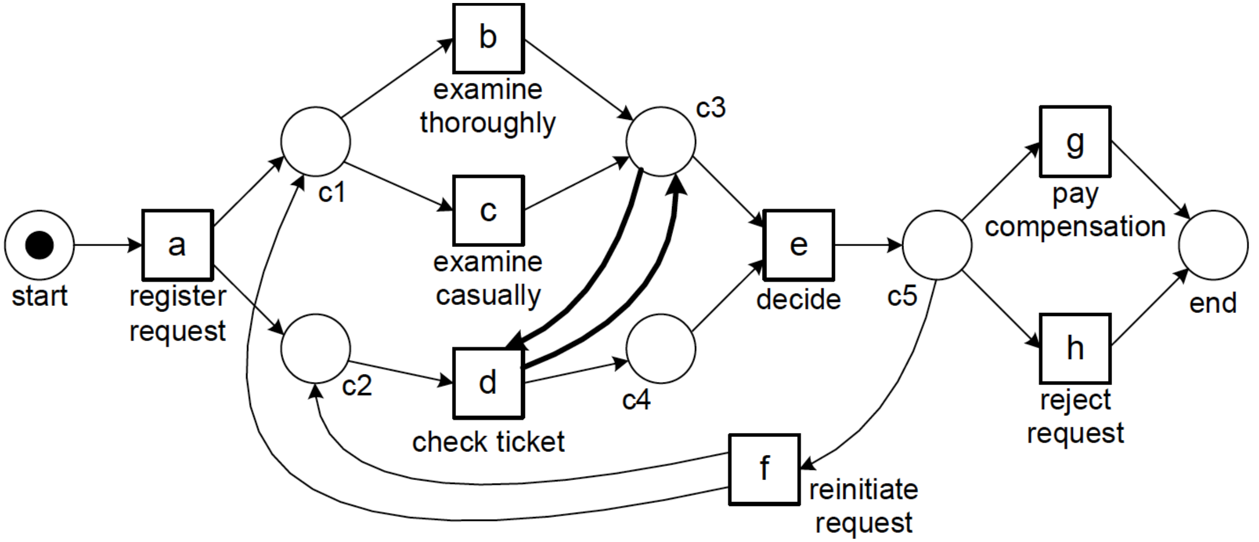


Online decision support

Detect.

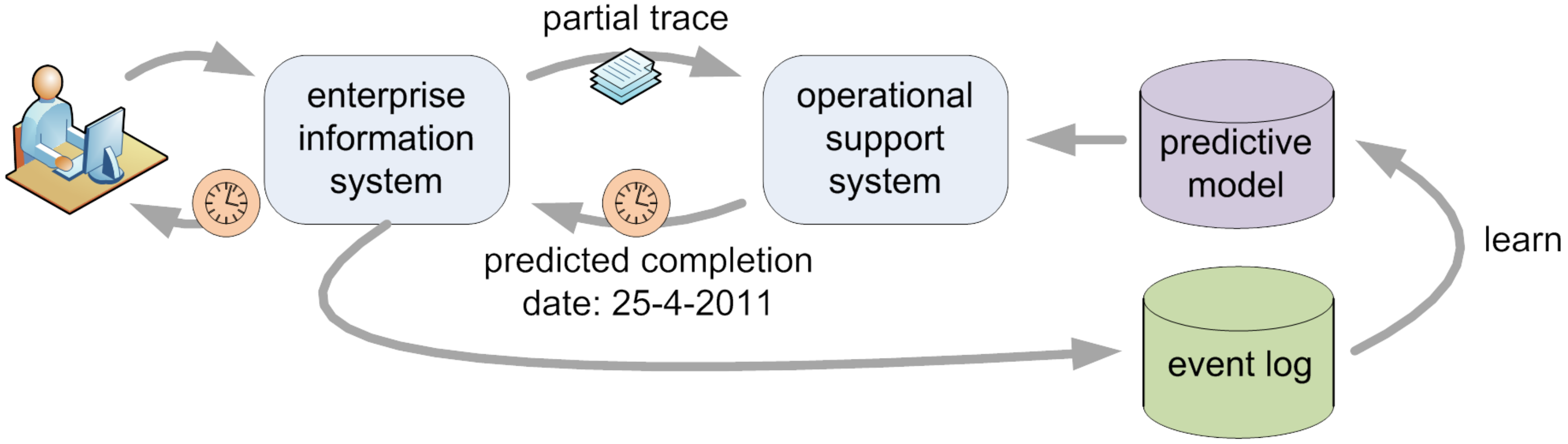
For the three cases, an **ALERT** fires when *d* starts: completion of d requires completion of *b* or *c.*

case id	trace
1	$\langle a^{12}_{start}, a^{19}_{complete}, b^{25}_{start}, d^{26}_{start}, b^{32}_{complete}, d^{33}_{complete}, e^{35}_{start}, e^{40}_{complete}, h^{50}_{start}, h^{54}_{complete} \rangle$
2	$\langle a^{17}_{start}, a^{23}_{complete}, d^{28}_{start}, c^{30}_{start}, d^{32}_{complete}, c^{38}_{complete}, e^{50}_{start}, e^{59}_{complete}, g^{70}_{start}, g^{73}_{complete} \rangle$
3	$\langle a^{25}_{start}, a^{30}_{complete}, c^{32}_{start}, c^{35}_{complete}, d^{35}_{start}, d^{40}_{complete}, e^{45}_{start}, e^{50}_{complete}, f^{50}_{start}, f^{55}_{complete}, b^{60}_{start}, d^{62}_{start}, b^{65}_{complete}, d^{67}_{complete}, e^{80}_{start}, e^{87}_{complete}, g^{90}_{start}, g^{98}_{complete} \rangle$
...	...



Online decision support

Predict.



Online decision support

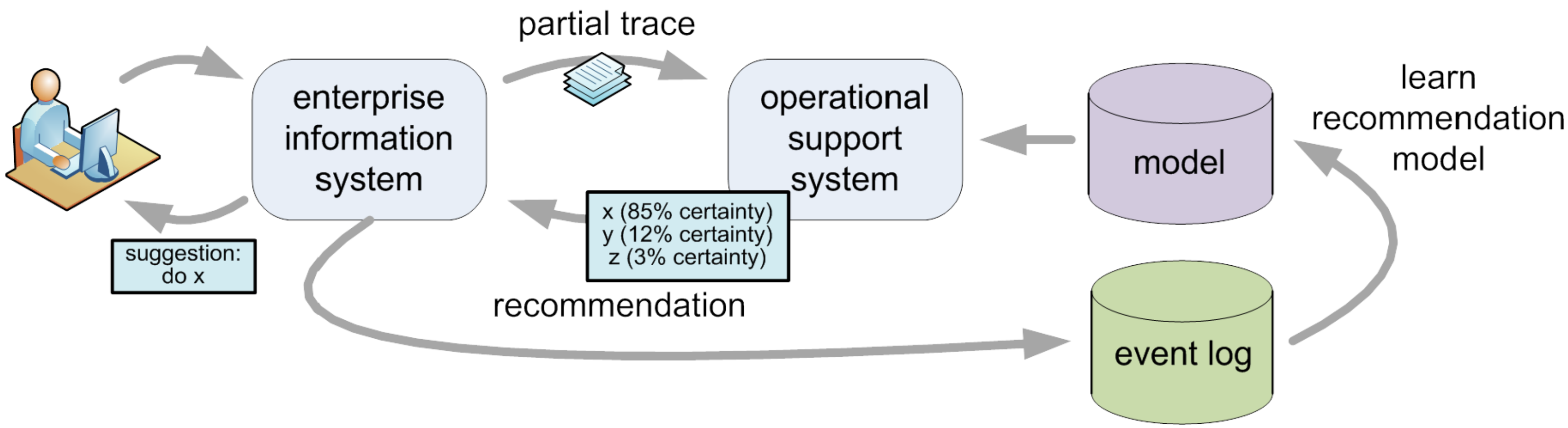
Predictions (examples).

- The predicted remaining flow time is 14 days;
- The predicted probability of meeting the legal deadline is 0.72;
- The total cost of this case will be 4500 euro;
- The predicted probability that activity *a* will occur is 0.34;
- The predicted probability that person r will work on this case is 0.57;
- The predicted probability that a case will be rejected is 0.67;
- The predicted total service time is 98 minutes.

To generate these predictions, many typical data mining tasks may be used.

Online decision support

Recommend.



Recommendation:

do action x with 85% of certainty of being the best one.

do action y with 12% of certainty.

do action z with 3% of certainty.

Online decision support

Recommend

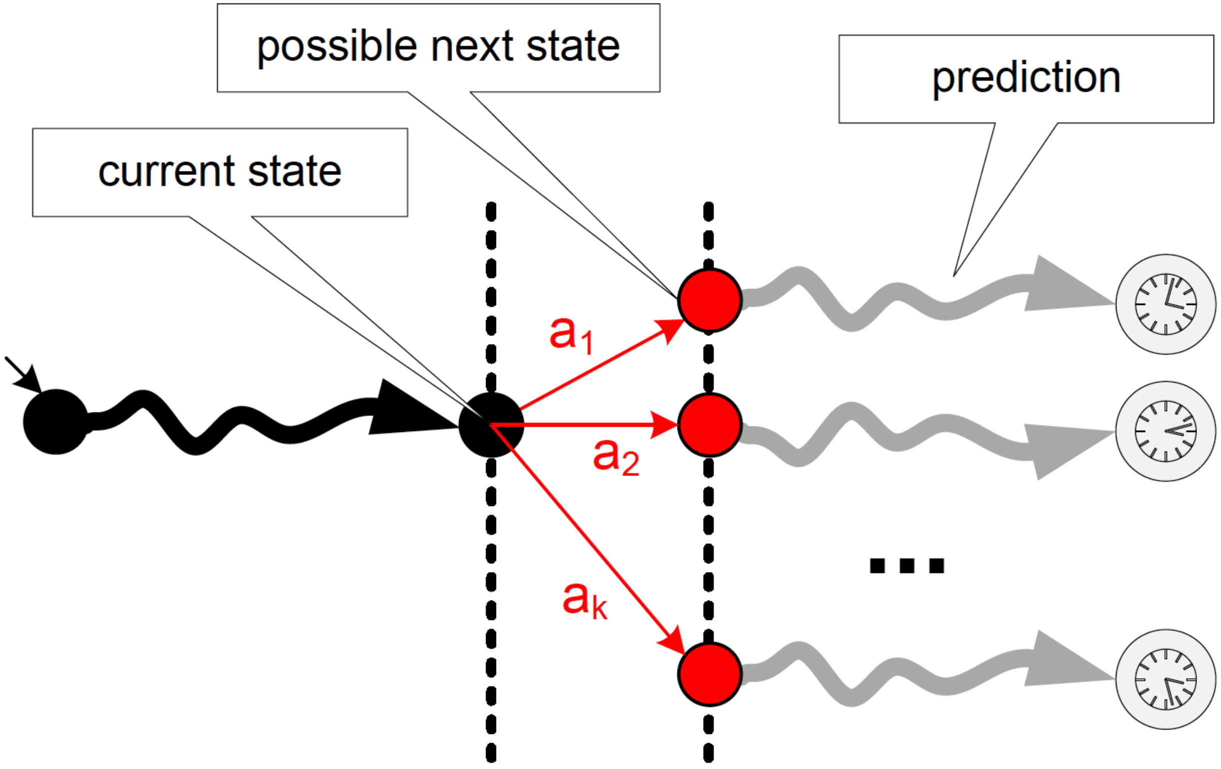
Examples:

- Next activity. Given three candidates *f,g,h*, which one is the best w.r.t. a given goal.
- Suitable resource.
- Routing decision.

- A recommendation is always given with respect to a specific goal. There is also a corresponding prediction of some indicator.
- Examples of goals:
  - minimize the remaining flow time; (indicator: flow time)
  - minimize the total costs;
  - maximize the fraction of cases handled within 4 weeks;
  - maximize the fraction of cases that is accepted; and
  - minimize resource usage.

Online decision support

Recommendation



Outline

- Motivation. Process mining basics.
- Getting event data.
- Process discovery.
- Conformance checking.
- Online Process Mining.
- **Tools.**
- Conclusion.

Tool support

Recall definition of BI (according to Forrester).

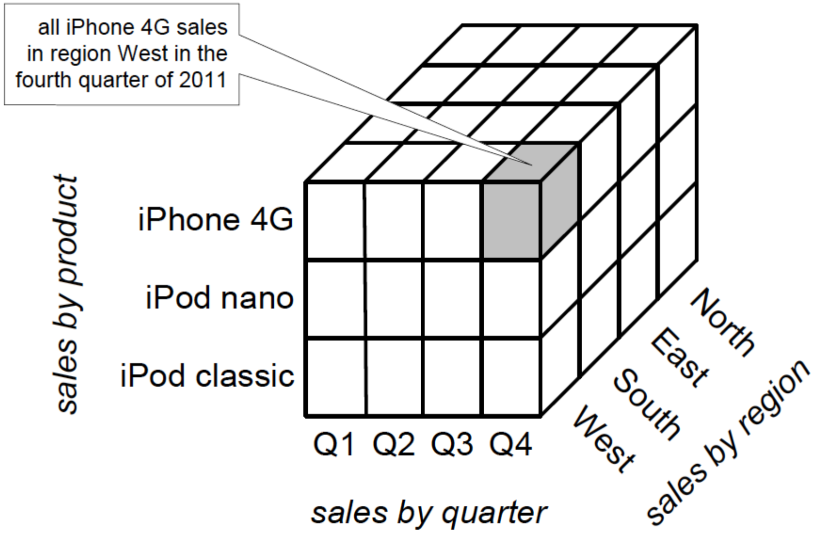
“BI is a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical, and operational insights and decision making”

Tool support

BI/OLAP/DW Tools:

Typical functions:

- ETL (Extract, Transform, and Load).
- Ad-hoc querying.
- Reporting.
- Interactive dashboards.
- Alert generation.
- Data Cube analysis (roll-up, drill-down, etc.).



Tool support

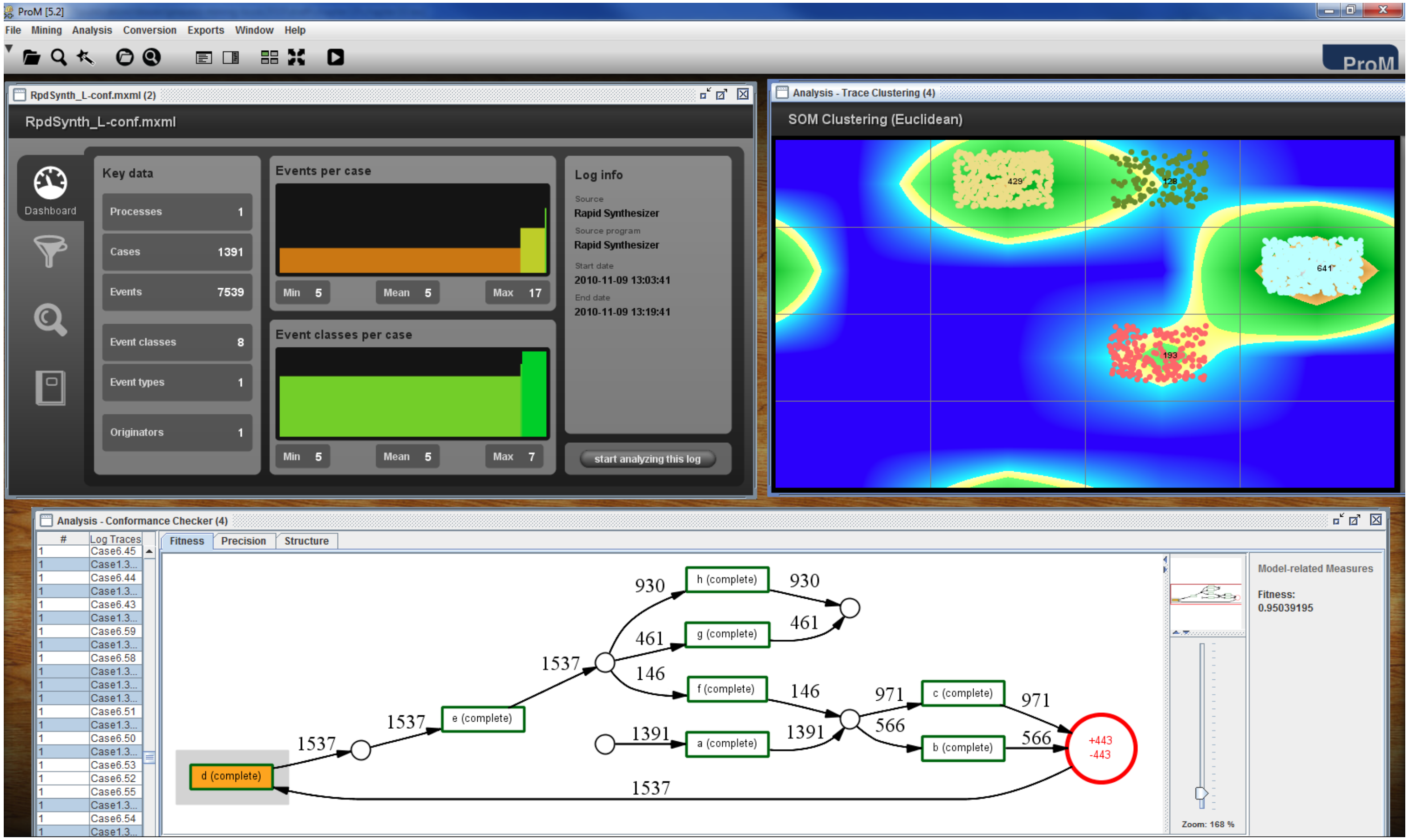
- Core of most BI tools is OLAP (Online Analitycal Processing)
- Input is relational data, NOT process/event data.
- Systems are data-centric rather than process-centric => mainstream BI products do not support process mining.
- The focus is on reporting and monitoring of KPIs

- Data mining tools are also data-centric, but provides “intelligence”.
- Even though these tools do not directly support event log data (like XES), the log can be preprocessed and analyzed with data mining tools.
- See for example WEKA (Waikato Environment for Knowledge Analysis, weka.wikispaces.com) and R (www.r-project.org).
- Typical frameworks not appropriate for process mining. Then, tools for PM started to appear.

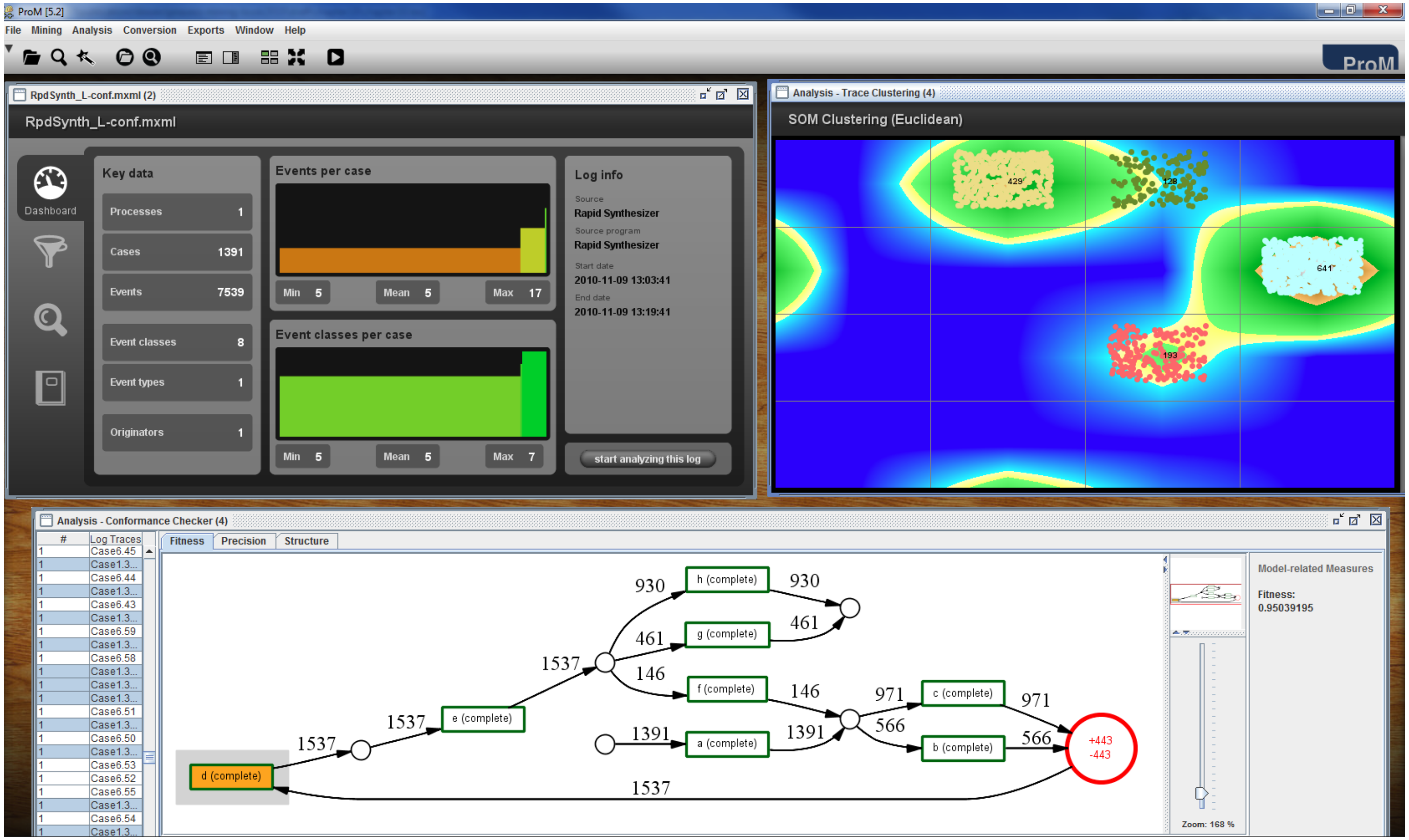
Tool support

- Available at www.processmining.org
- ProM supports all the techniques we have seen, and more complex ones.
- Pluggable architecture.
- Last version: ProM 6.

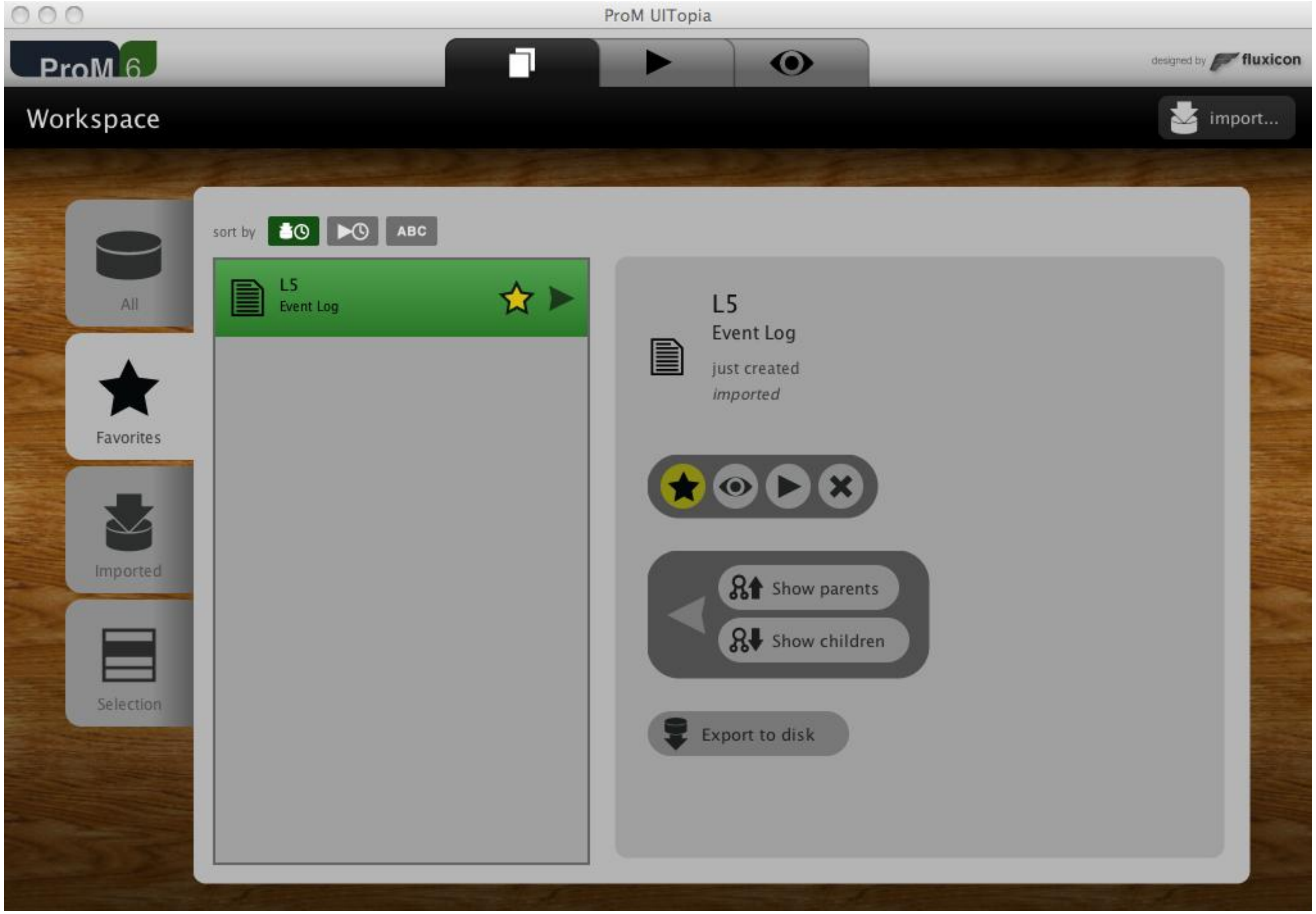
Tool support



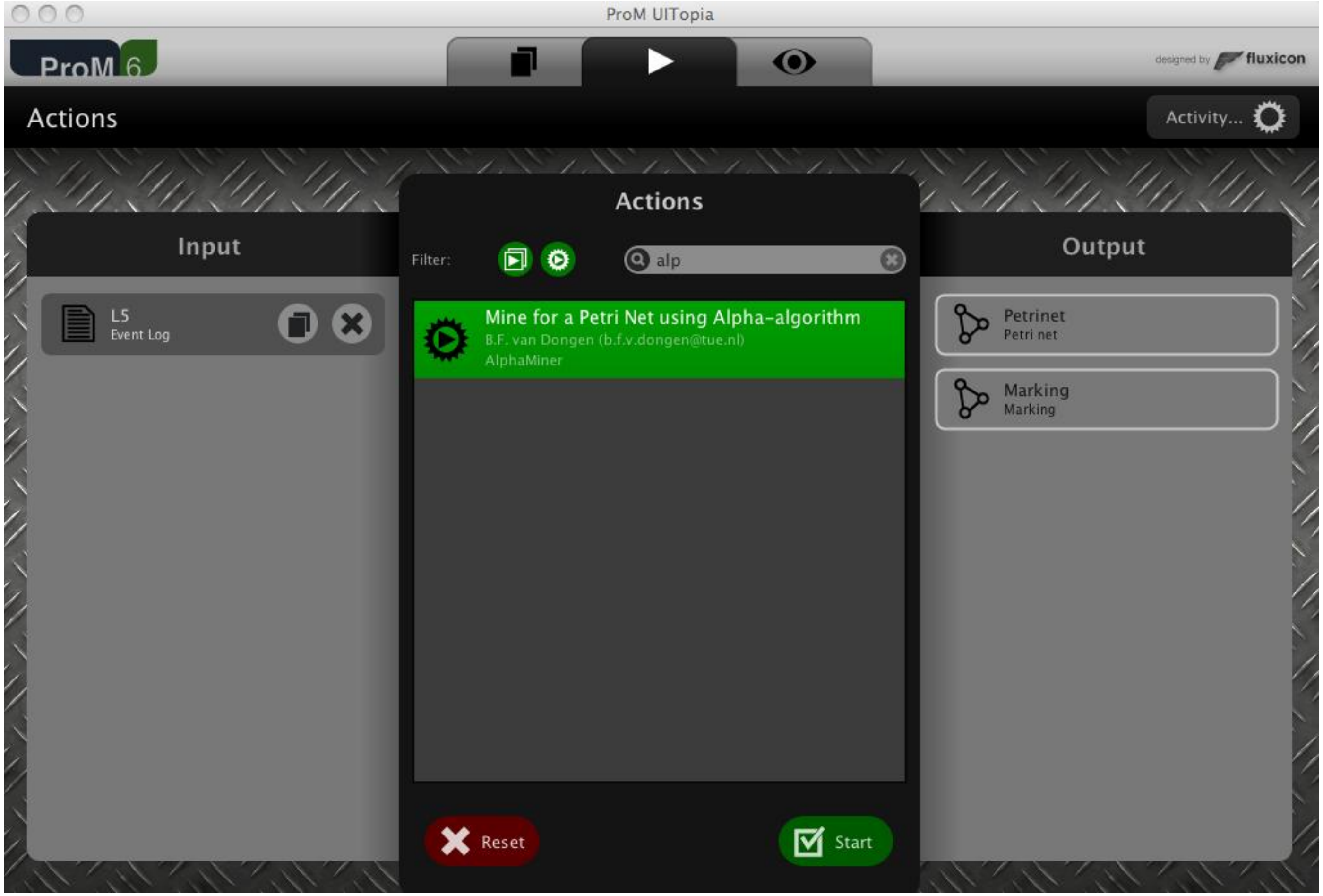
Tool support



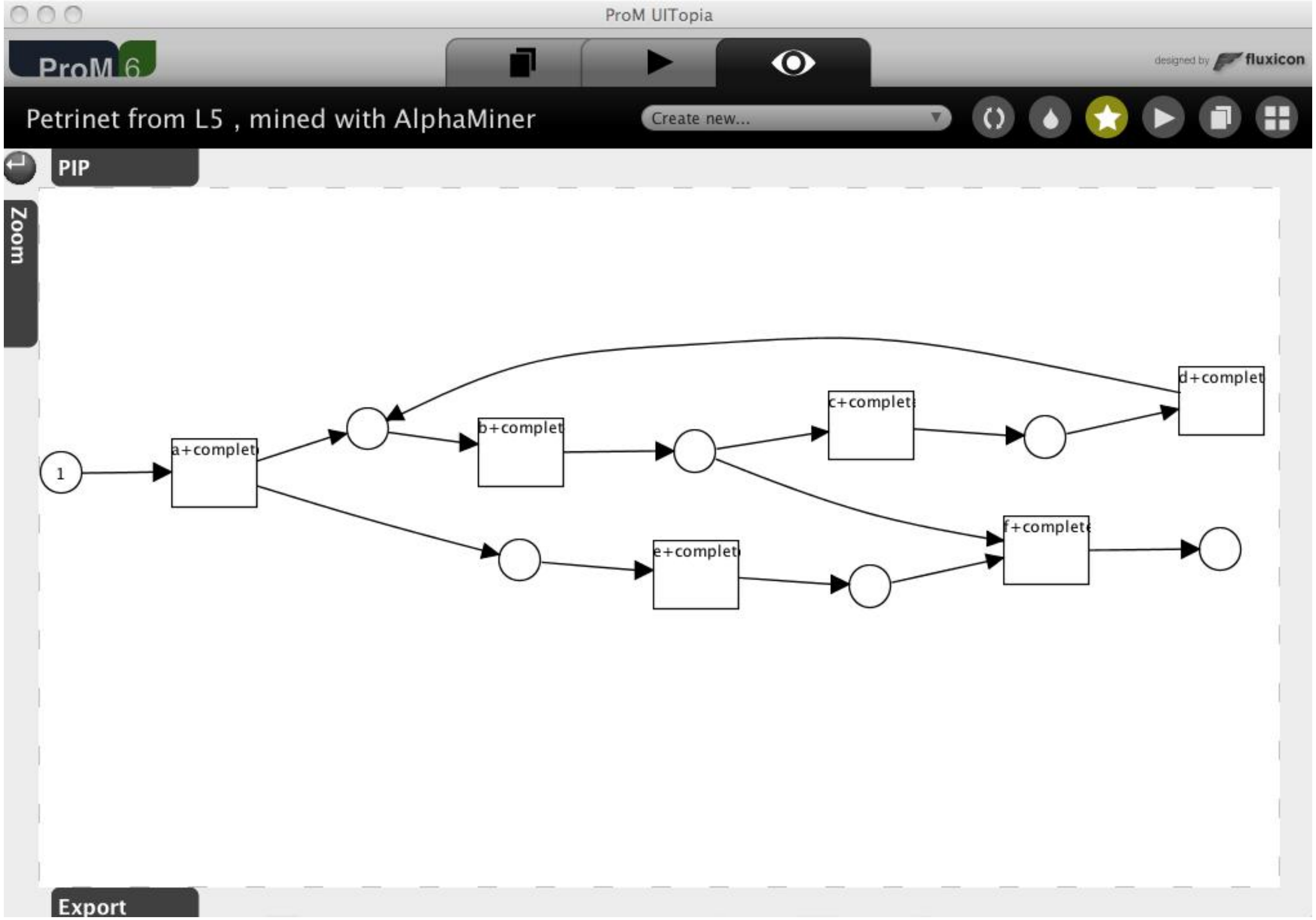
Tool support



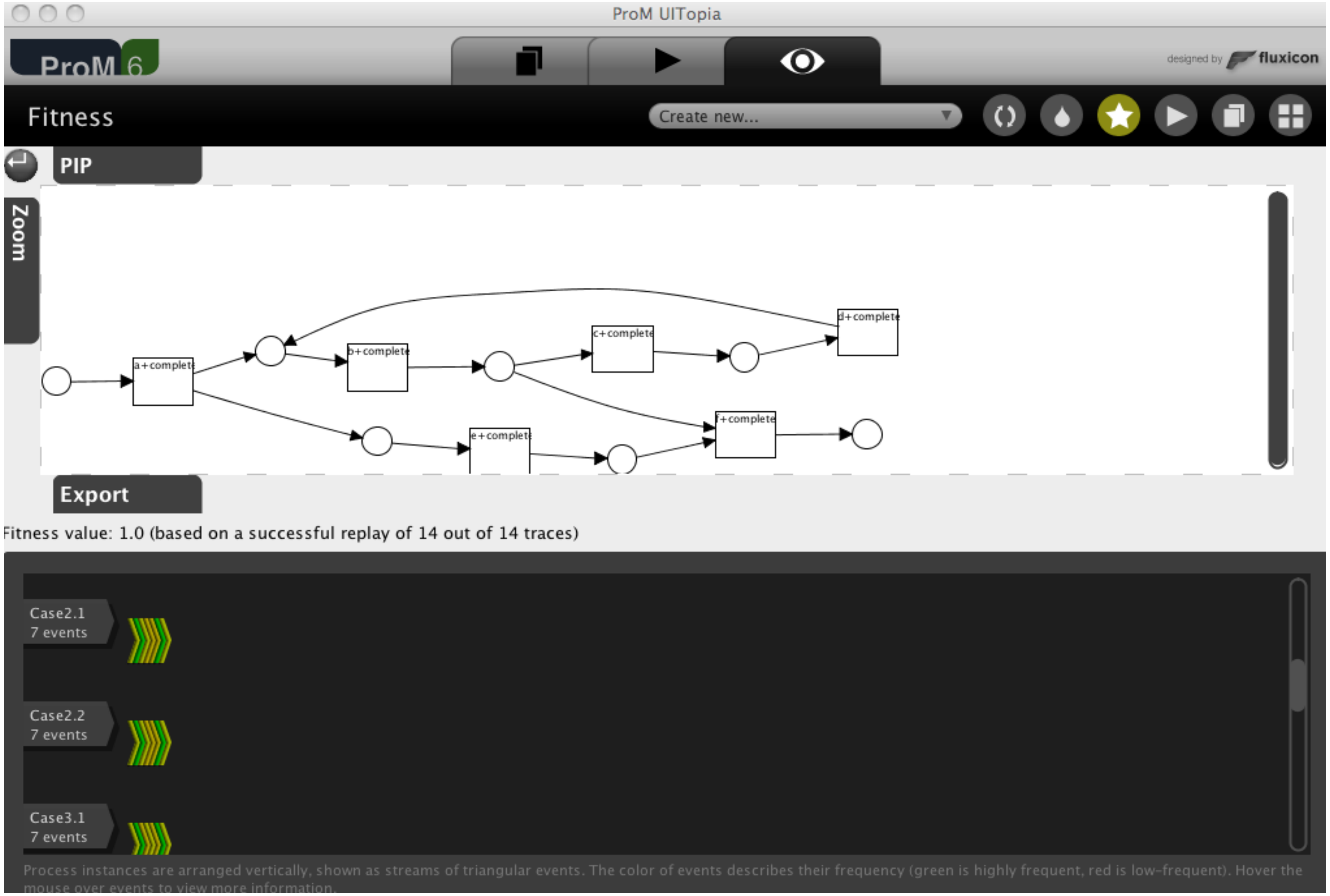
Tool support: ProM



Tool support: ProM



Tool support: ProM



Outline

- Motivation. Process mining basics.
- Getting event data.
- Process discovery.
- Conformance checking.
- Online Process Mining.
- Tools.
- **Conclusion.**

Conclusion

- Process mining: one of the most important innovations in the field of BPM.
- PM in some sense bridges the gap between BP and data mining/databases, and advances of the *integration between flow and data.*
- Still in its infancy => *Many open issues here*.
  - Data cleansing.
  - Benchmarking.
  - Algorithms.
  - Distributed PM (e.g., cross-organizational PM).
  - Privacy issues (e.g., keeping the log confidential). To what extent can we analyze the log without disclosing sensitive information? Which portion of the log can we analyze?
  - Could existing tools and methods from sequential patterns analysis be used in this domain?