technische universiteit eindhoven

Discovering Petri Nets

It's a kind of magic ...

Prof.dr.ir. Wil van der Aalst

Eindhoven University of Technology
Department of Information and Technology
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Outline

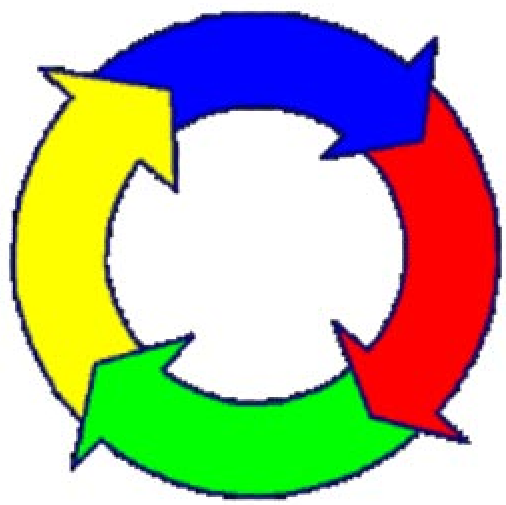
- Context: Closing the BPM loop
- Overview of process mining
- ProM toolset
- The alpha algorithm
- Alternative approaches
- Conclusion

Press arrow to start

The BPM life-cycle

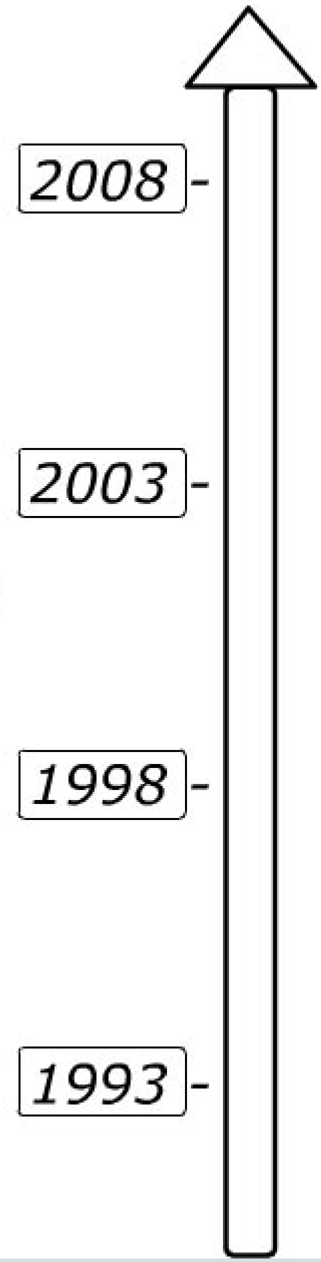
Diagnosis

Process enactment

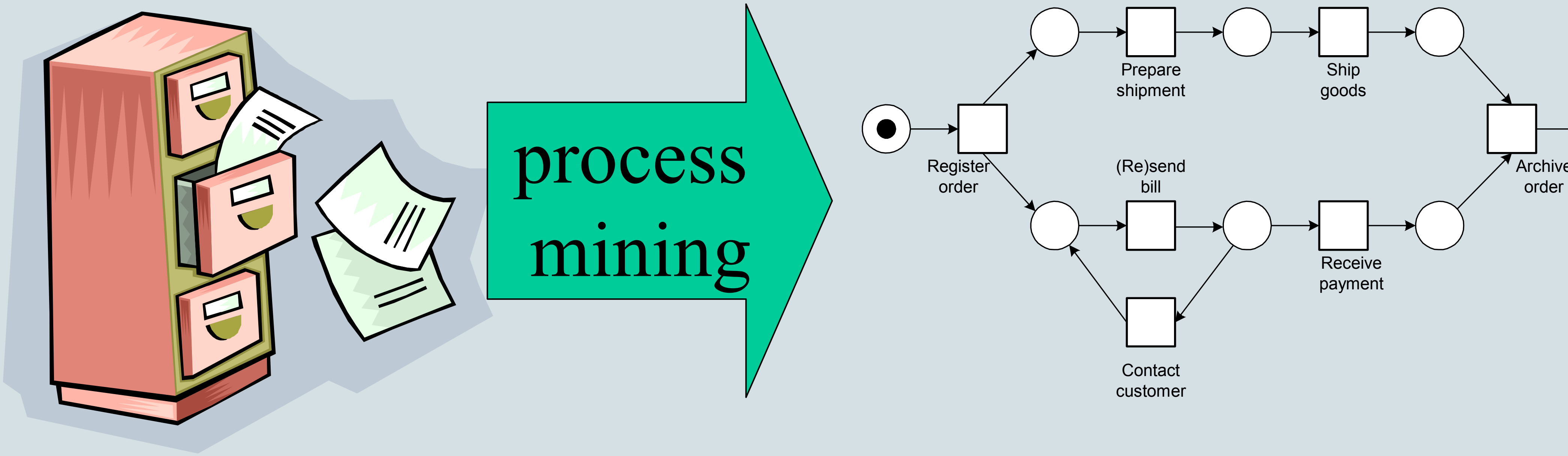


Process design

Implementation /configuraton

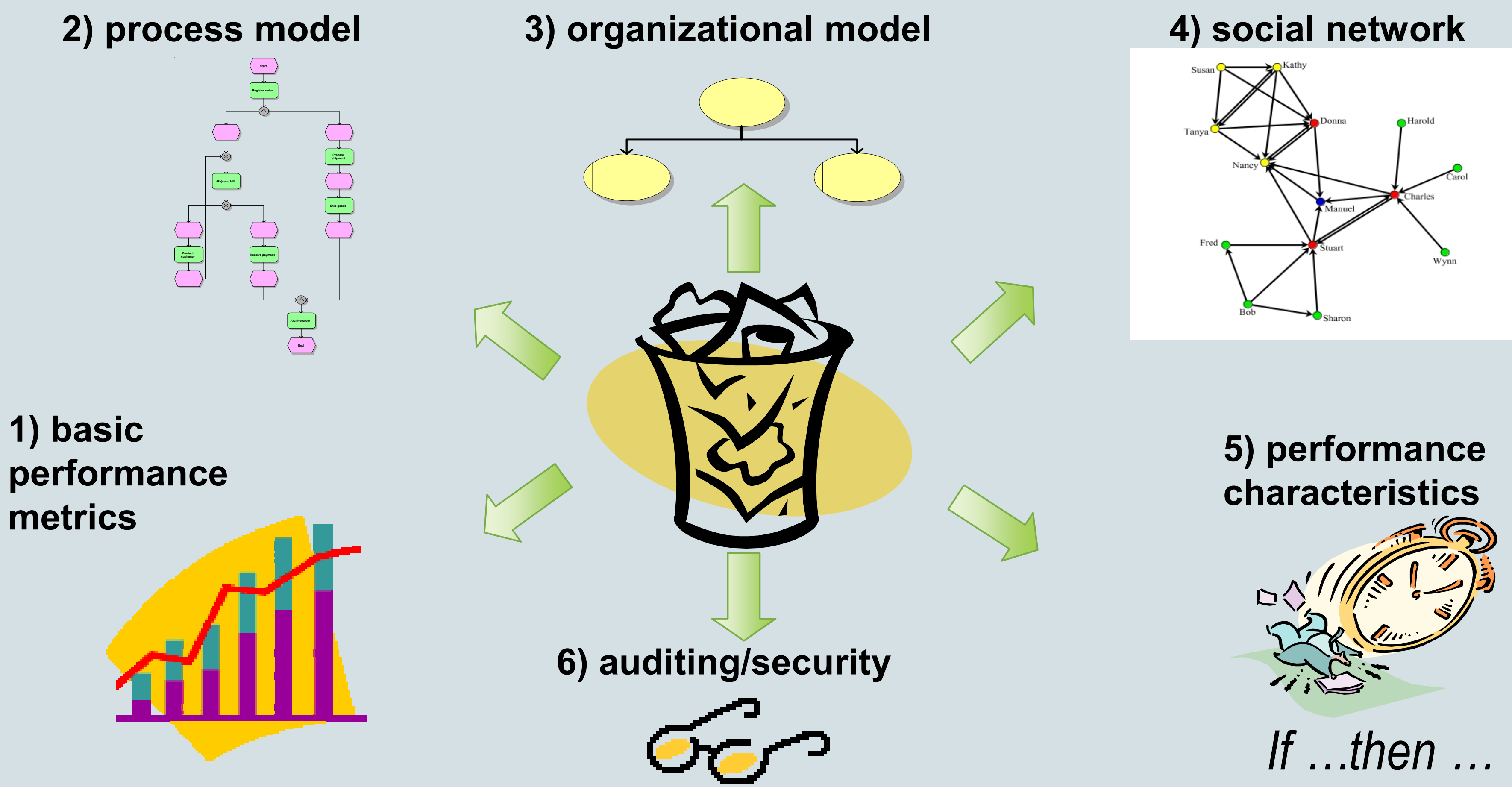


Process mining: Reversing the process



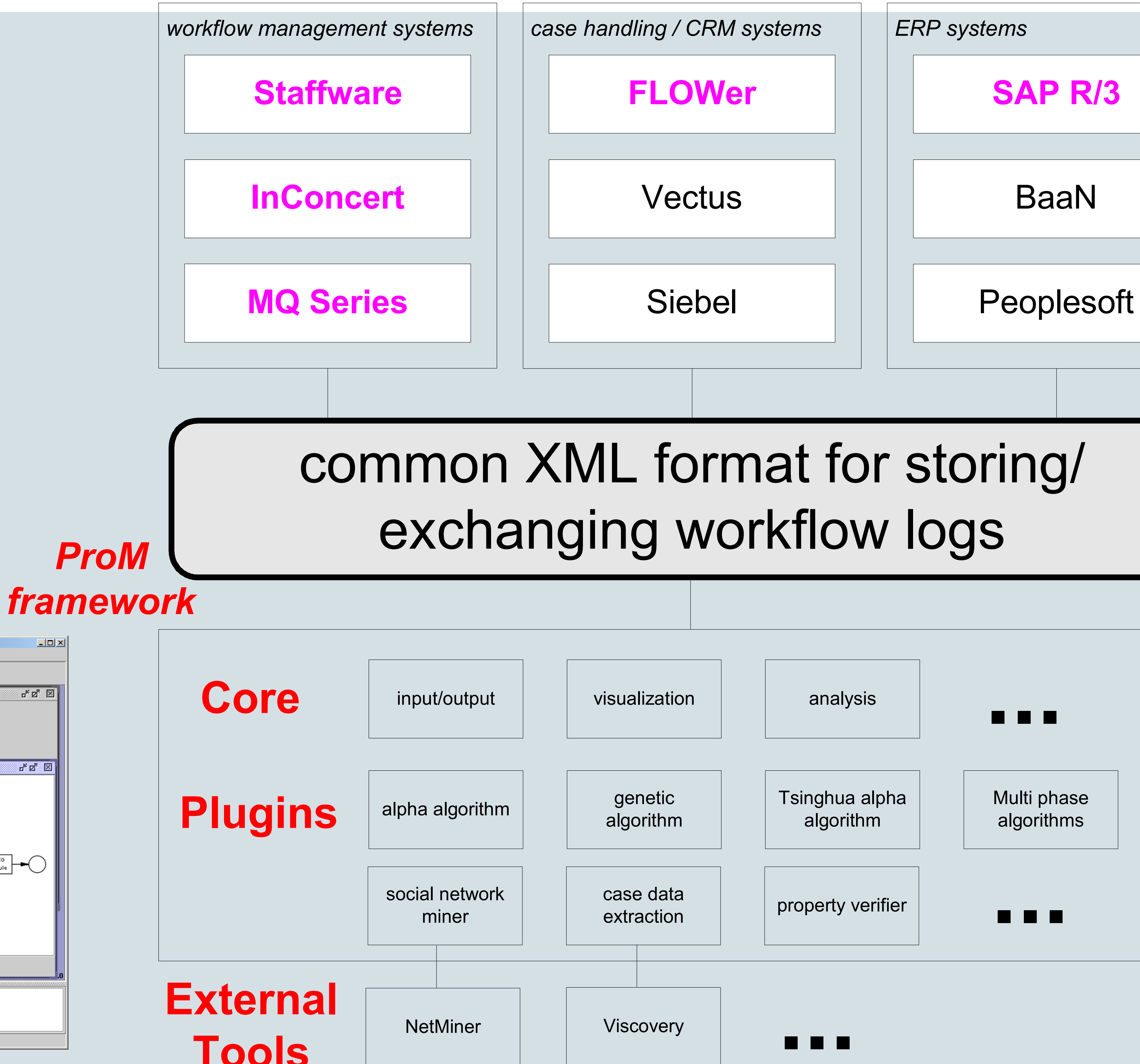
- Process mining can be used for:
  - Process discovery (What is the process?)
  - Delta analysis (Are we doing what was specified?)
  - Performance analysis (How can we improve?)

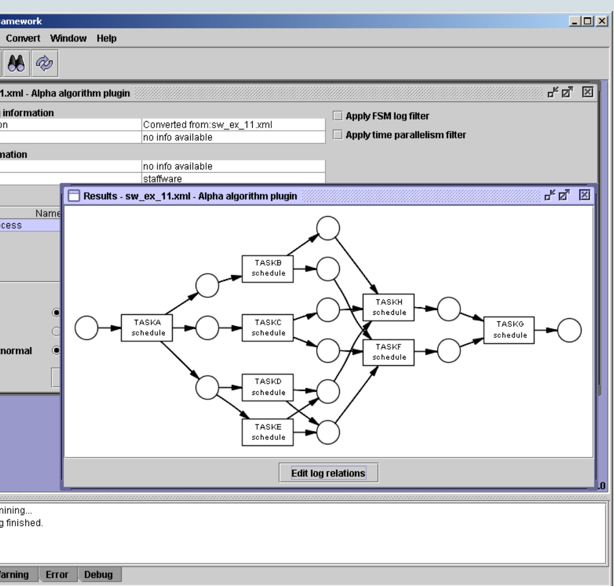
Process mining (overview)



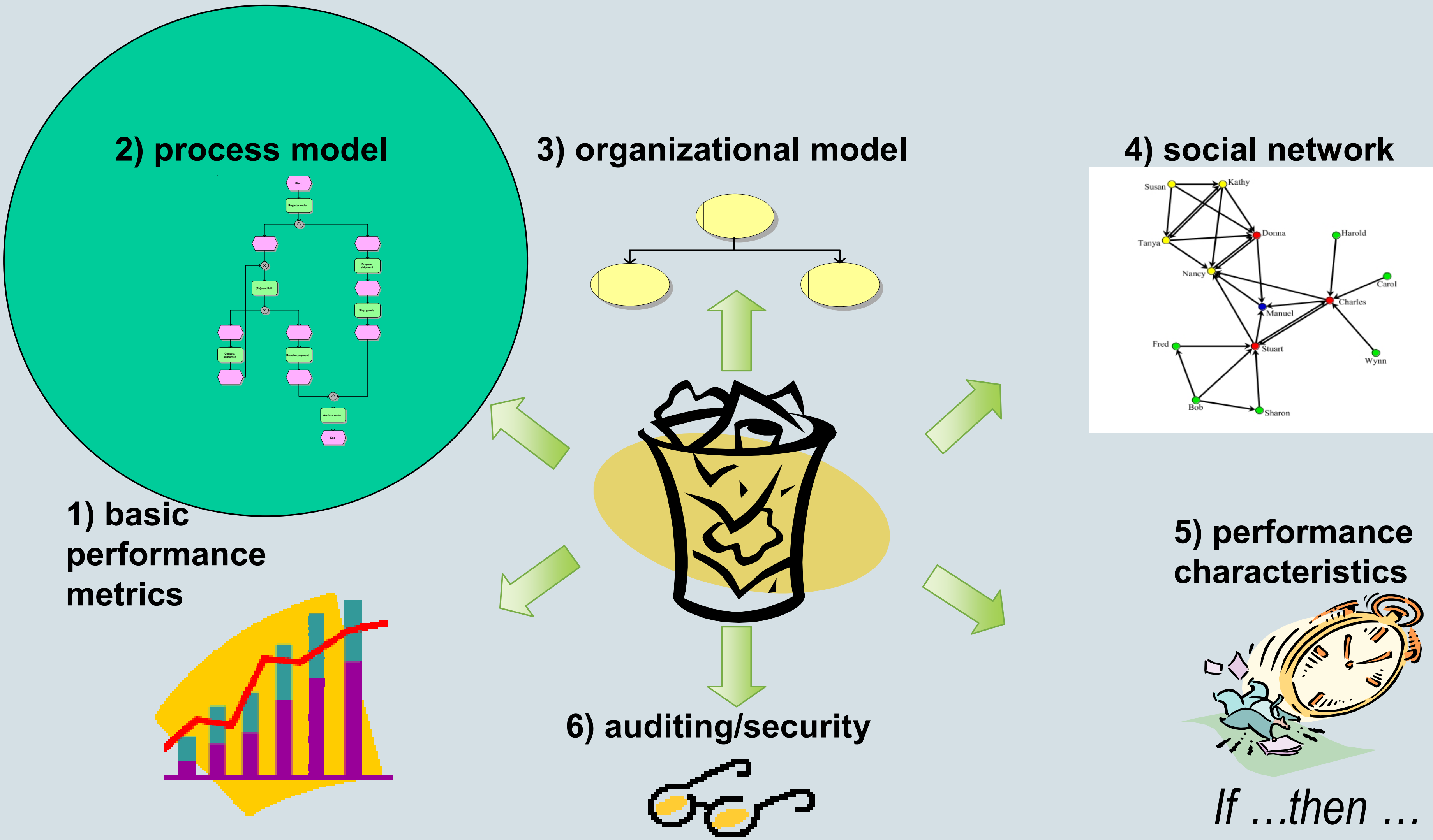
technische universiteit eindhoven

ProM

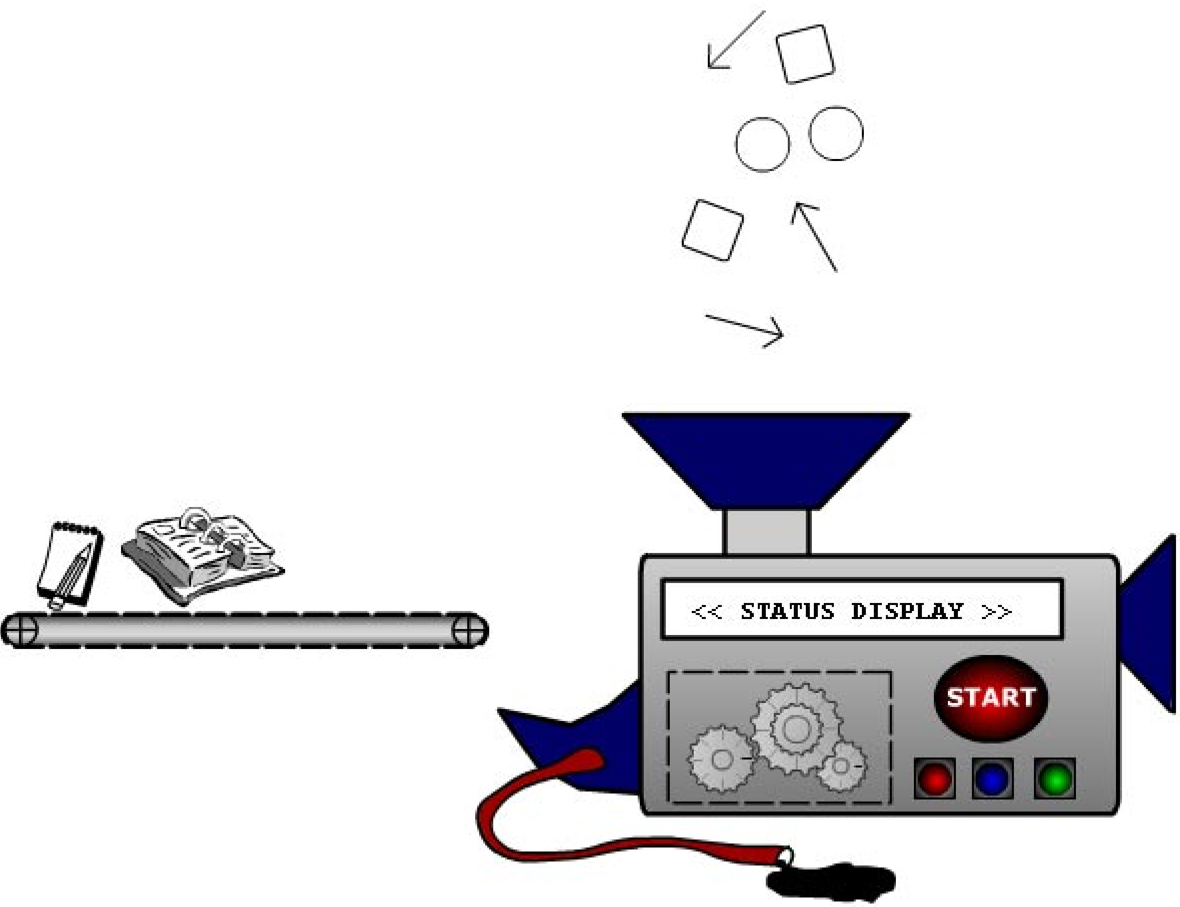




Let us focus on mining process models …



Process Mining



rocess log

- Minimal information in log: case id's and task id's.
- Additional information: event type, time, resources, and data.
- In this log there are three possible sequences:
  - ABCD
  - ACBD
  - EF

```
case 1 : task A
case 2 : task A
case 3 : task A
case 3 : task B
case 1 : task B
case 1 : task C
case 2 : task C
case 4 : task A
case 2 : task B
case 2 : task D
case 5 : task E
case 4 : task C
case 1 : task D
case 3 : task C
case 3 : task D
case 4 : task B
case 5 : task F
case 4 : task D
```

technische universiteit eindhoven

,→,||,# relations

Direct succession: x>y iff for some case x is directly followed by y.

Causality: x→y iff x>y and not y>x.

Parallel: x||y iff x>y and y>x

Choice: x#y iff not x>y and not y>x.

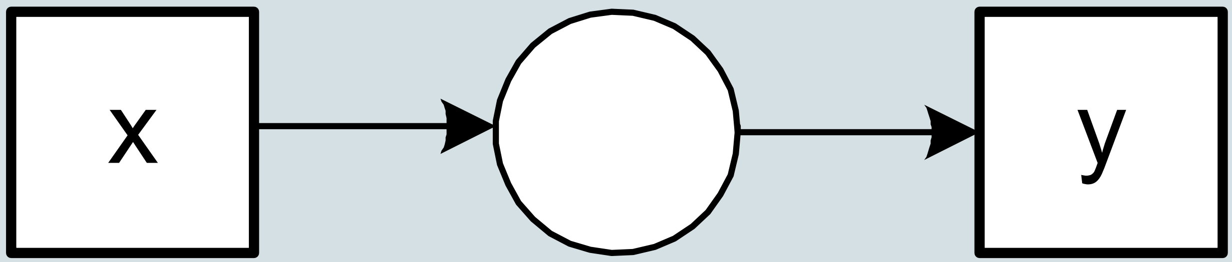
```
case 1 : task A
case 2 : task A
case 3 : task A
case 3 : task B
case 1 : task B
case 1 : task C
case 2 : task C
case 4 : task A
case 2 : task B
case 2 : task D
case 5 : task E
case 4 : task C
case 1 : task D
case 3 : task C
case 3 : task D
case 4 : task B
case 5 : task F
case 4 : task D
```

A>B
A>C
B>C
B>D
C>B
C>D
E>F

B||C
C||B

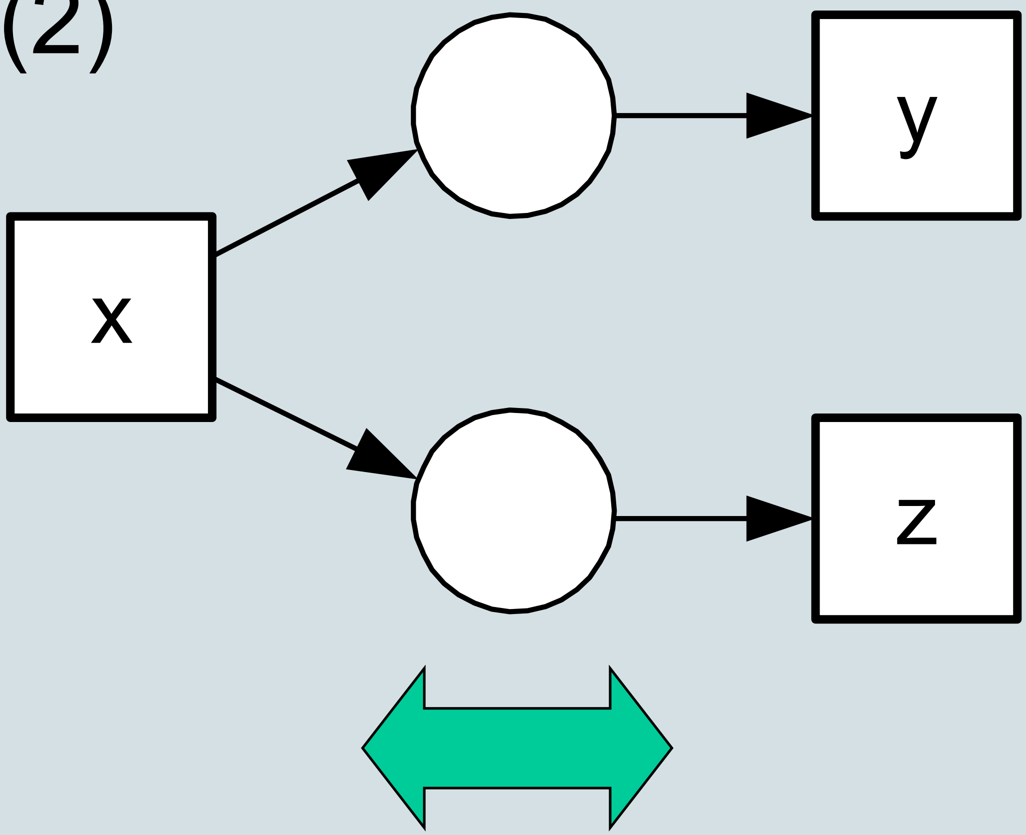
A→B
A→C
B→D
C→D
E→F

Basic idea (1)



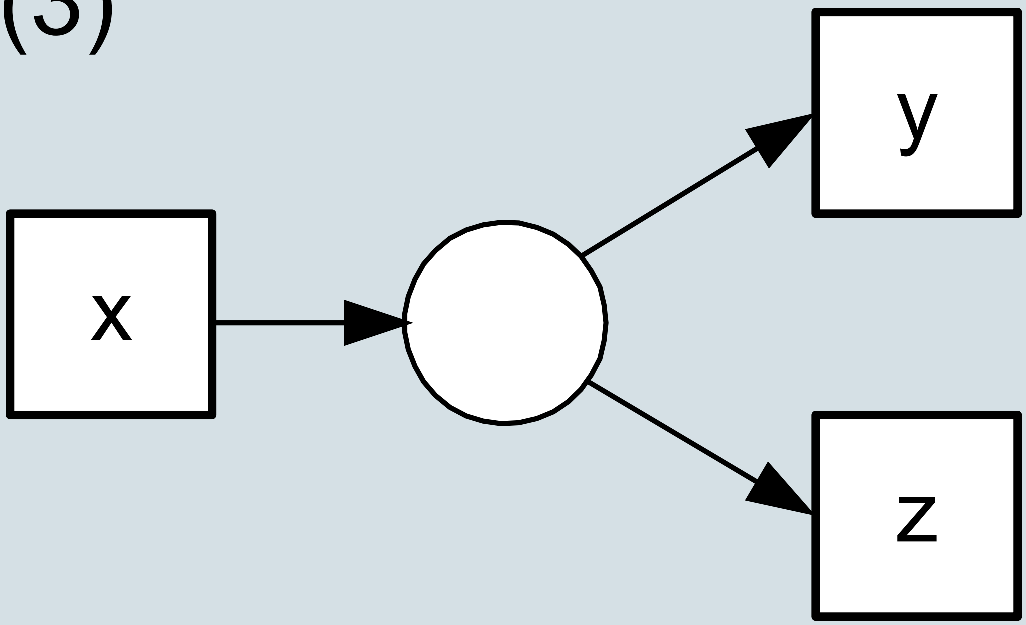
$$x \longrightarrow y$$

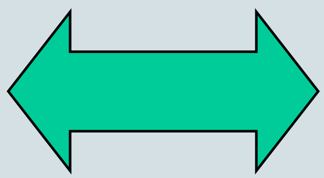
Basic idea (2)



x→y, x→z, and y||z

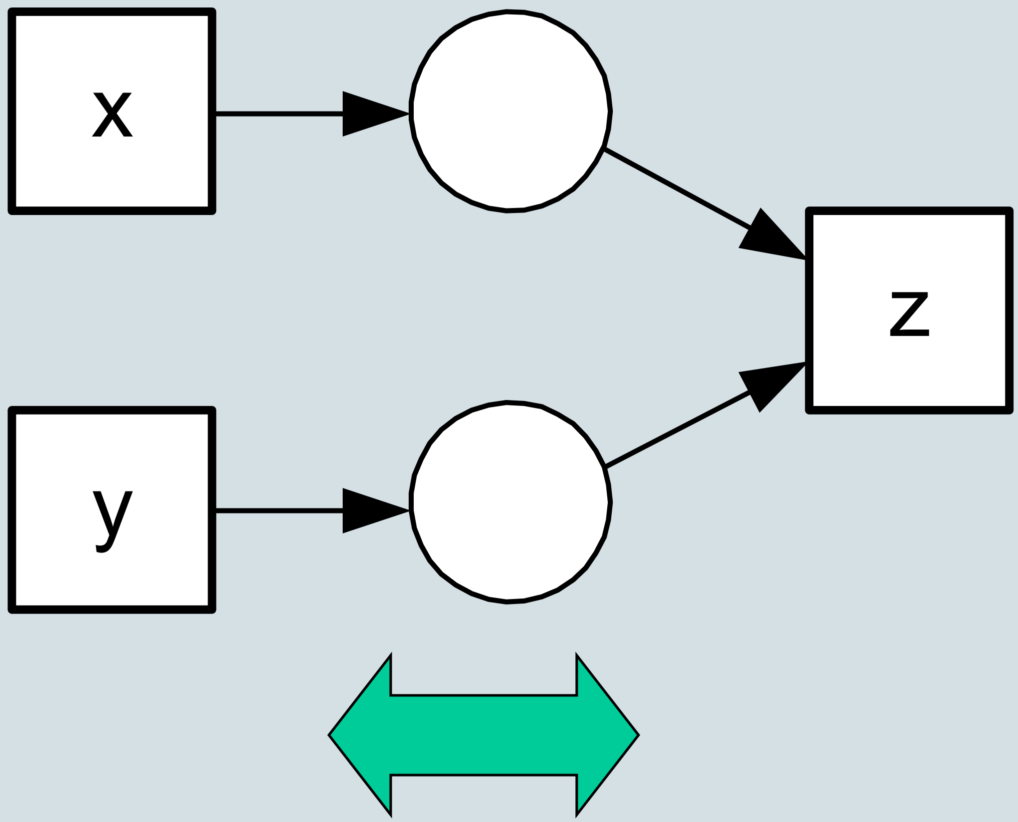
Basic idea (3)





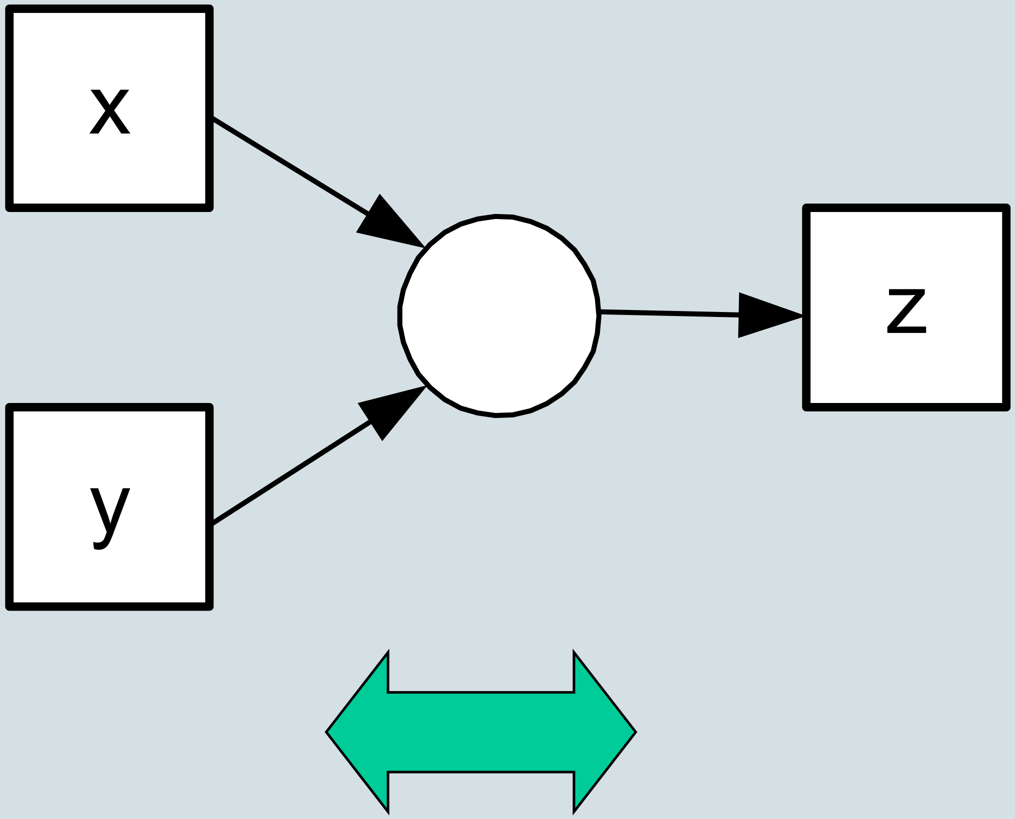
$x \rightarrow y$, $x \rightarrow z$, and $y \# z$

Basic idea (4)



x→z, y→z, and x||y

Basic idea (5)



x→z, y→z, and x#y

. is not that simple: Basic alpha algorithm

et W be a workflow log over T. $\alpha(W)$ is defined as follows.

$T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma\}$,

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

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

$X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1$
$a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,

$Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,

$P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W,o_W\}$,

$F_W = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup \{ (p_{(A,B)},b) \mid (A,B) \in Y_W \wedge b$
$B \} \cup \{ (i_W,t) \mid t \in T_I\} \cup \{ (t,o_W) \mid t \in T_O\}$, and

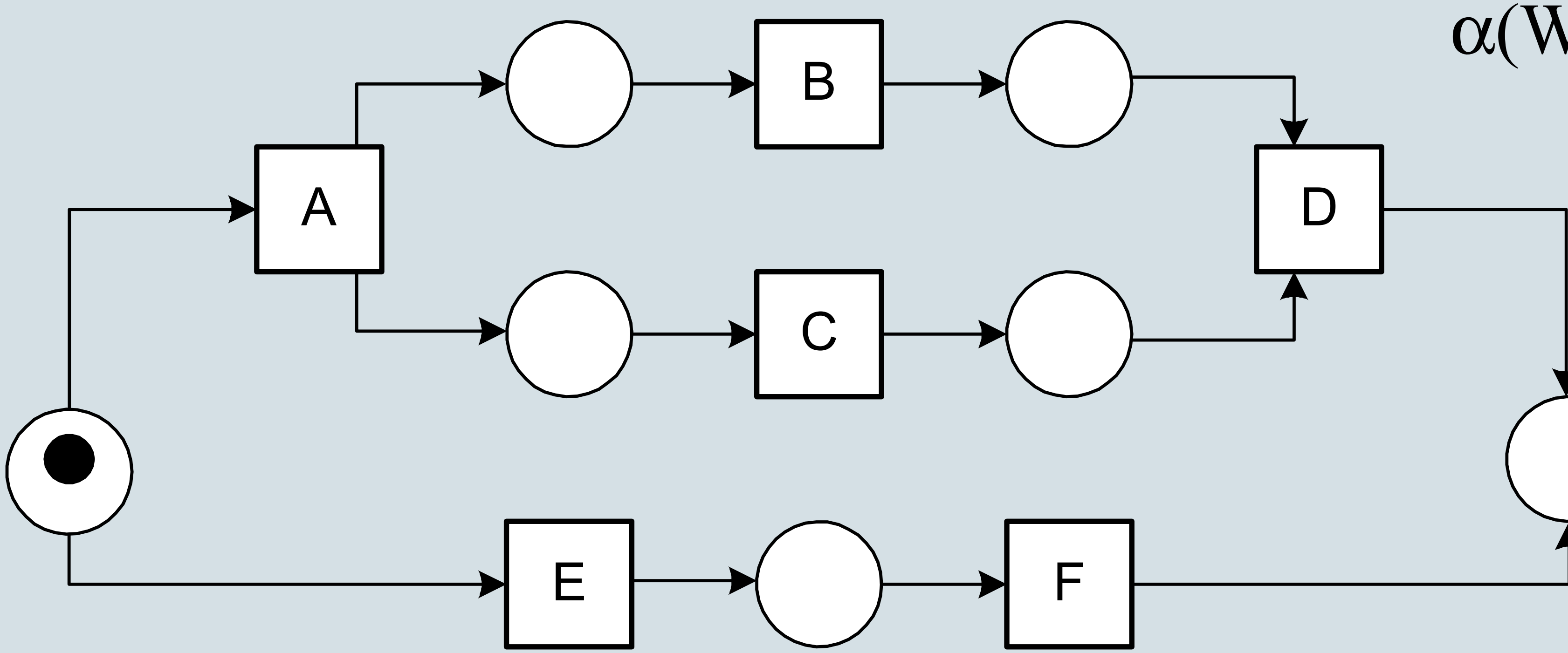
$\alpha(W) = (P_W,T_W,F_W)$.

e alpha algorithm has been proven to be correct for a large class of free-choice net

Example

se 1 : task A
se 2 : task A
se 3 : task A
se 3 : task B
se 1 : task B
se 1 : task C
se 2 : task C
se 4 : task A
se 2 : task B
se 2 : task D
se 5 : task E
se 4 : task C
se 1 : task D
se 3 : task C
se 3 : task D
se 4 : task B
se 5 : task F
se 4 : task D

α(W



Challenges

- Refining existing algorithm for (control-flow/process perspective)
  - *Hidden tasks*
  - *Duplicate tasks*
  - *Non-free-choice constructs*
  - *Loops*
  - *Detecting concurrency (implicit or explicit)*
  - *Mining and exploiting time*
  - *Dealing with noise*
  - *Dealing with incompleteness*
- Mining other perspectives (data, resources, roles, …)
- Gathering data from heterogeneous sources
- Visualization of results
- Delta analysis

Alternative approaches

- Something based on the Theory of Regions? (Note that we typically see only a fragment of all possible behaviors of the system, cf. Occam's Razor.)
- Current approaches:
  - The Tsinghua variant
  - Heuristics (cf. Thumb)
  - Multi-phase process mining
  - Genetic algorithms

Alternative: Multi-phase process mining (Boudewijn van Dongen)

Two phases:

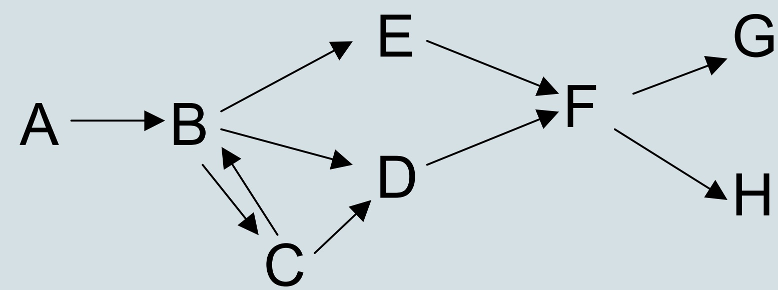
1) Create a visual description of each instance, without choices and loops (cf. runs or occurrence nets).
   - Comprehensive representation
   - Ideal for performance analysis (cf. ARIS PPM)
2) Aggregate multiple instances to one process model.
   - Only causal relations between tasks are required

Properties:

- More robust and multi-lingual (cf. EPCs).
- Possibility of inspect instances

Example

Causal relations:



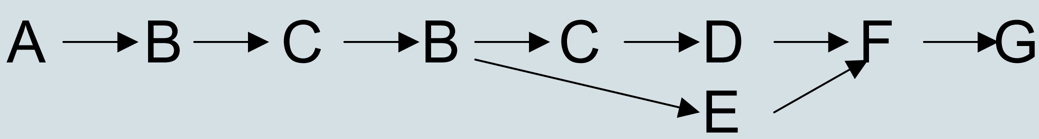
Log file:

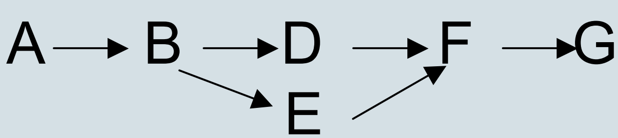
A, B, C, B, C, D, E, F, G

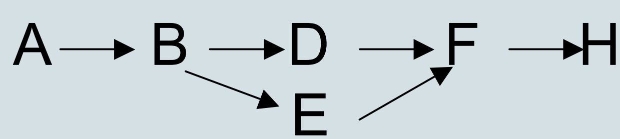
A, B, E, D, F, G

A, B, D, E, F, H

For each entry in every instance, find the closest causal predecessor and successor, and build instance graphs

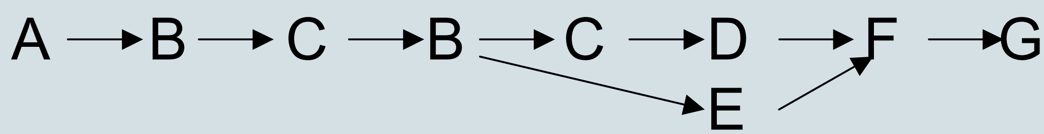




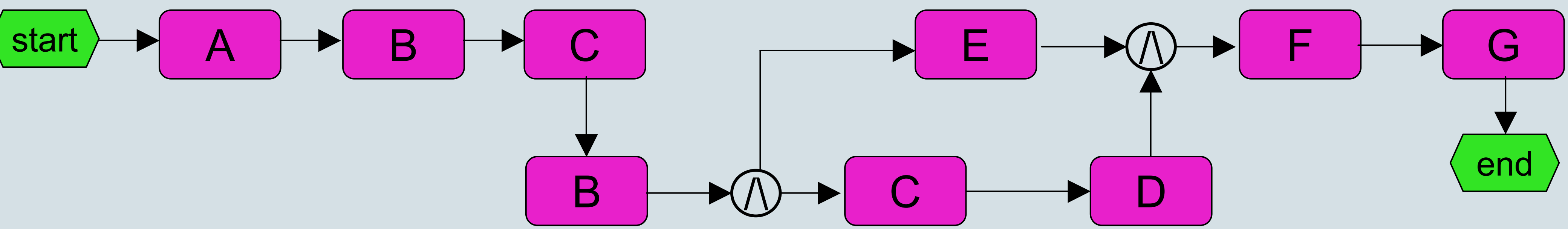


Translation

nstance graph:

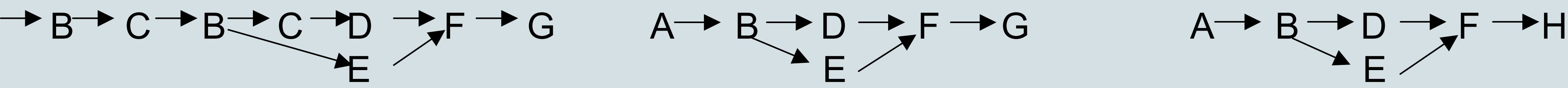


Transformation to EPC:

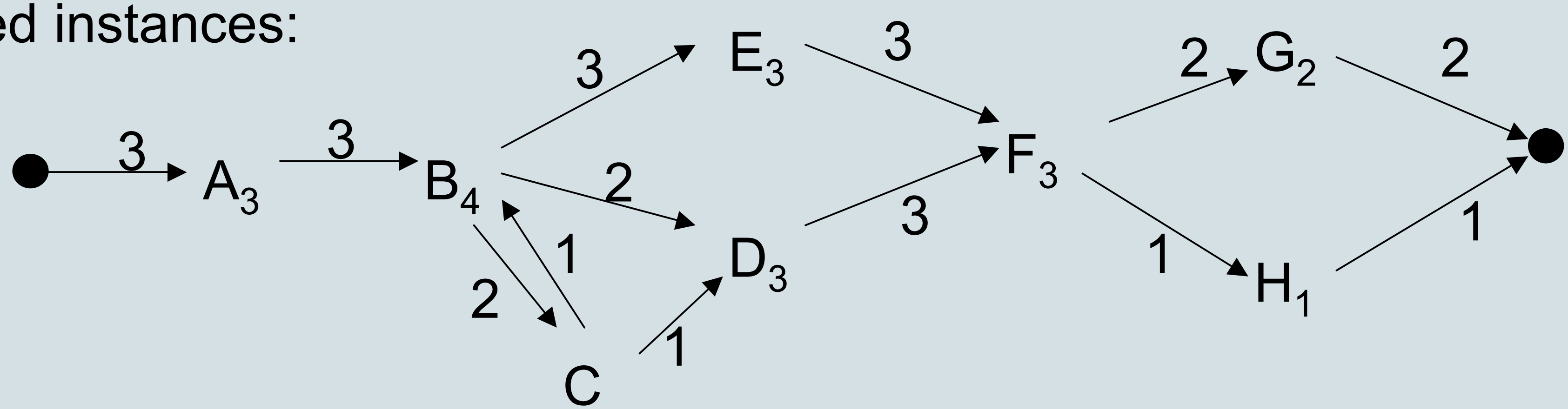


Aggregation

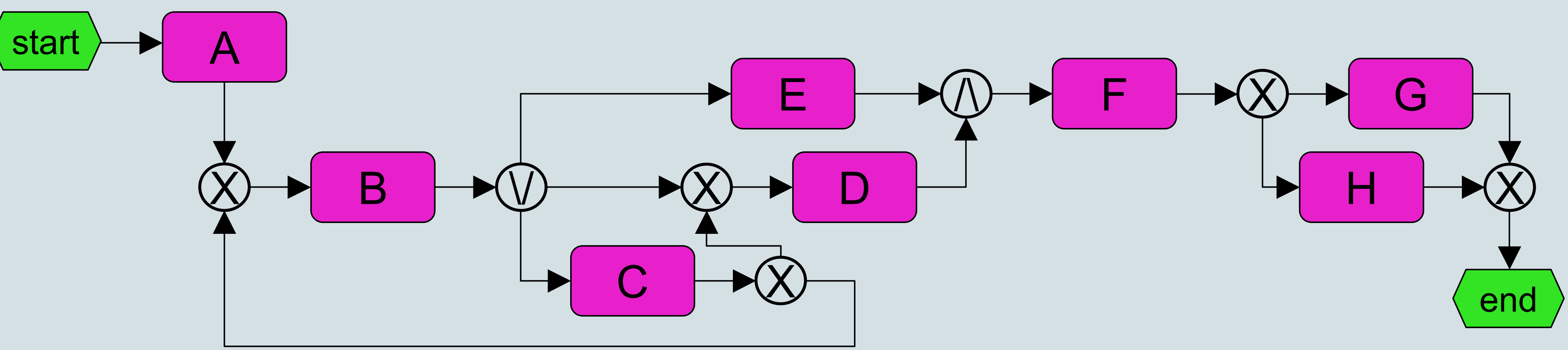
Three instance graphs:



Aggregated instances:



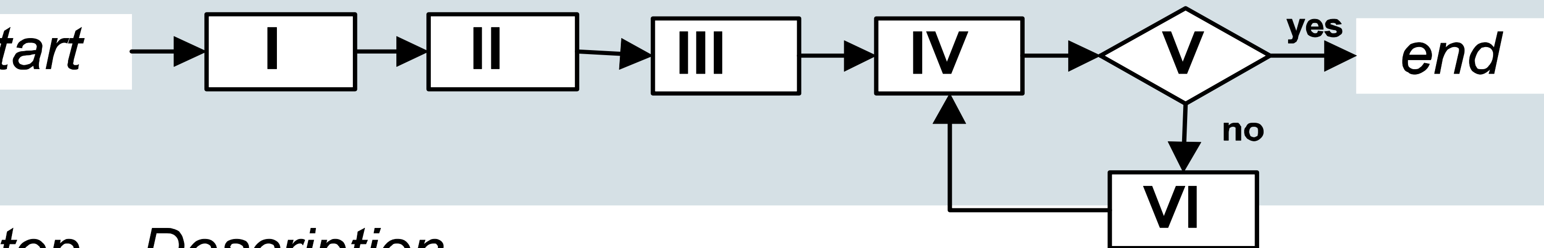
Transformation to EPC:



Alternative: Genetic Mining (Ana Karla Alves de Medeiros and Ton Weijters)

- Use genetic algorithms to:
  - mine process models from incomplete event logs
  - Tackle structural constructs (duplicate tasks, non-free choice etc.) that α-algorithm does not
- Issues:
  - internal representation of individuals
  - fitness measure
  - genetic operations (crossover and mutation)

Overview of approach



tep	*Description*
	Read event log
	Calculate dependency relations among workflow model elements
I	Build the initial population
/	Calculate individuals' fitness
	Stop and return the fittest individuals?
I	Create next population - use genetic operations

Example

Log:

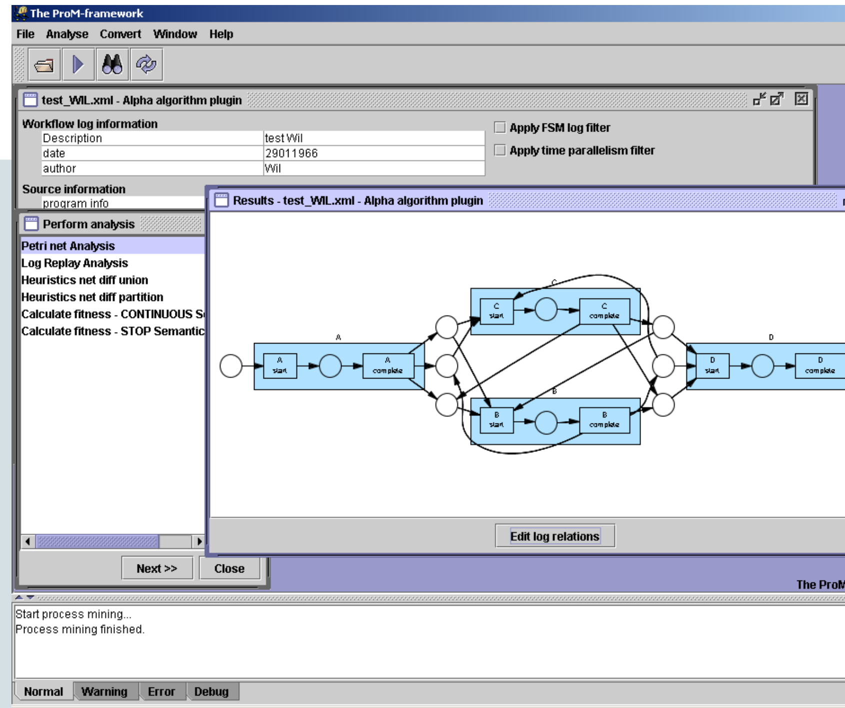
As,Ac,Bs,Bc,Cs,Cc,Ds,Dc

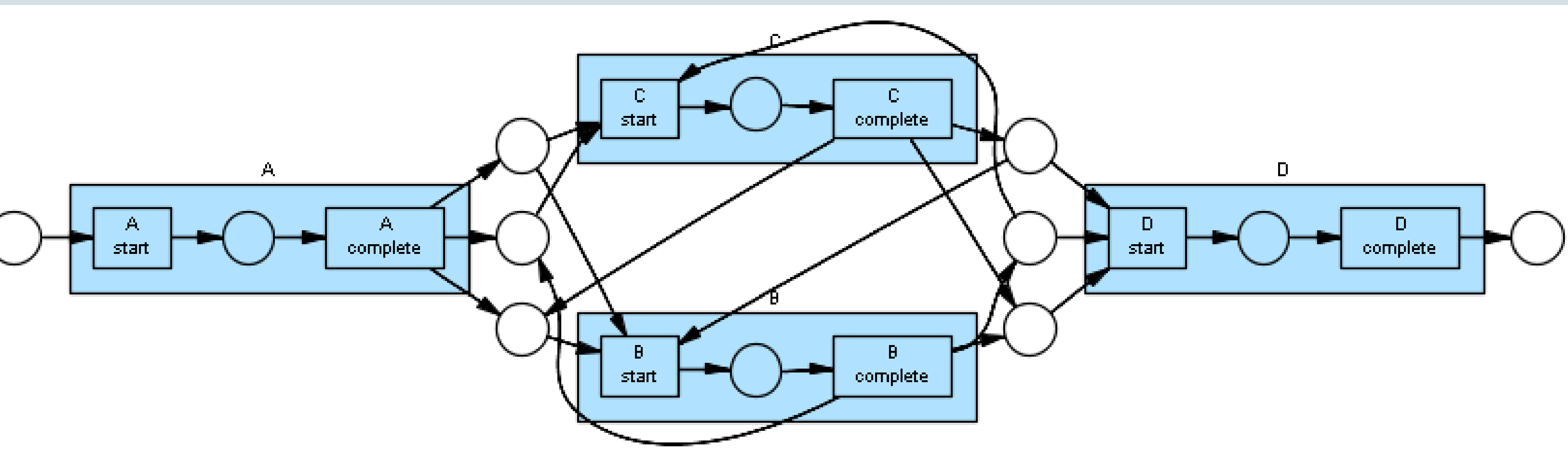
As,Ac,Bs,Bc,Cs,Cc,Ds,Dc

As,Ac,Cs,Cc,Bs,Bc,Ds,Dc

This log is incomplete if we assume that B and C are in parallel but never happen to overlap.

α-algorithm result

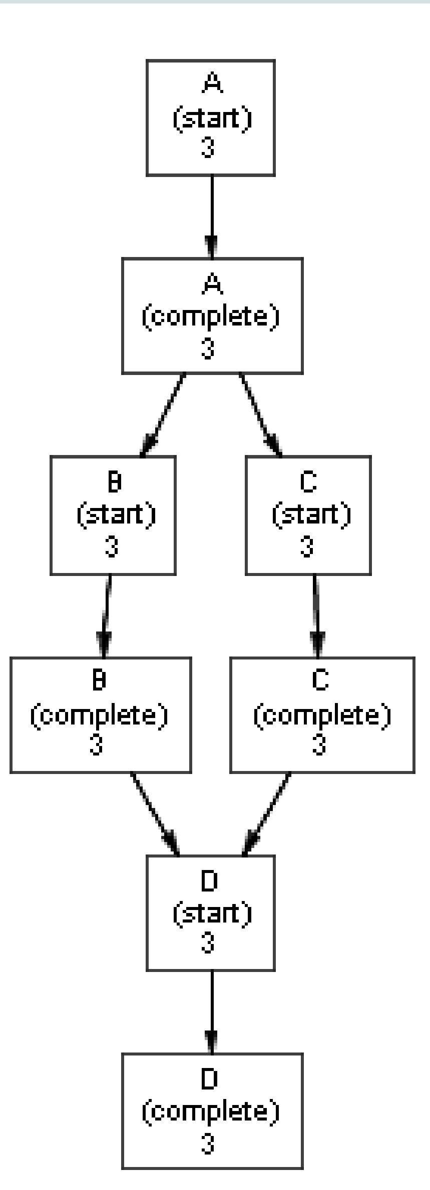




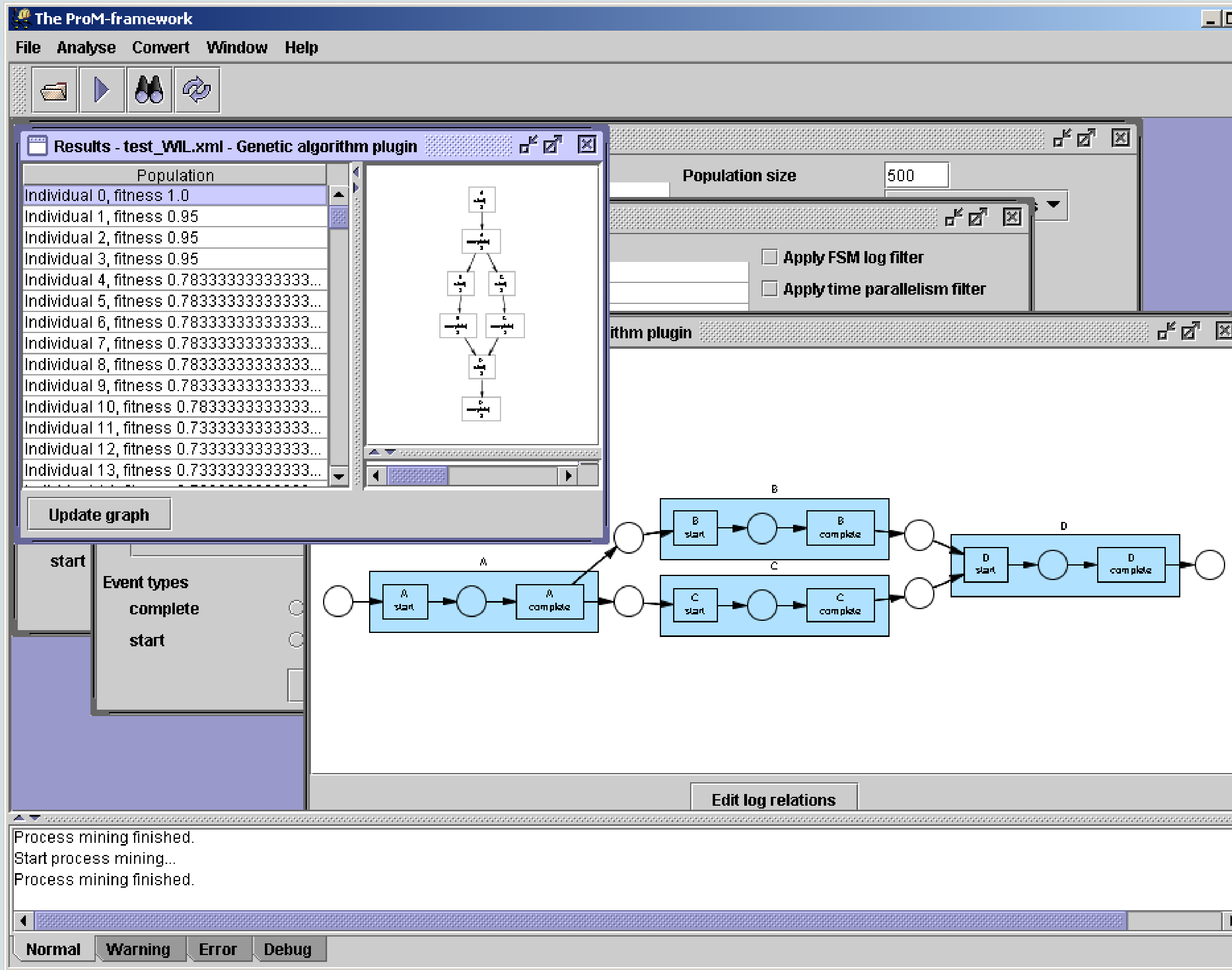
If we only consider the start or complete events, the result is OK. If not, more traces are needed of infer that B and C are in parallel.

technische universiteit eindhoven

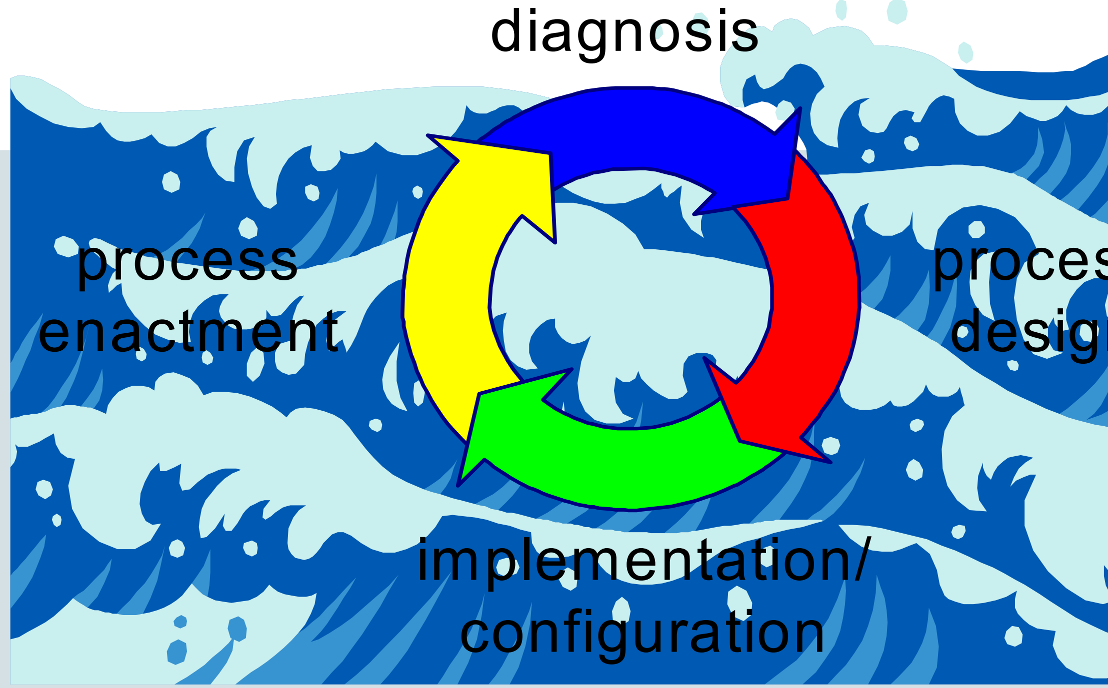
Genetic mining result



Fitness 100%



Conclusion



Process mining provides many interesting challenges for scientists, customers, users, managers, consultants, and tool developers.

The alpha algorithm is an example of an approach focusing on the process perspective.

It is related to but also quite different from the Theory of Regions.

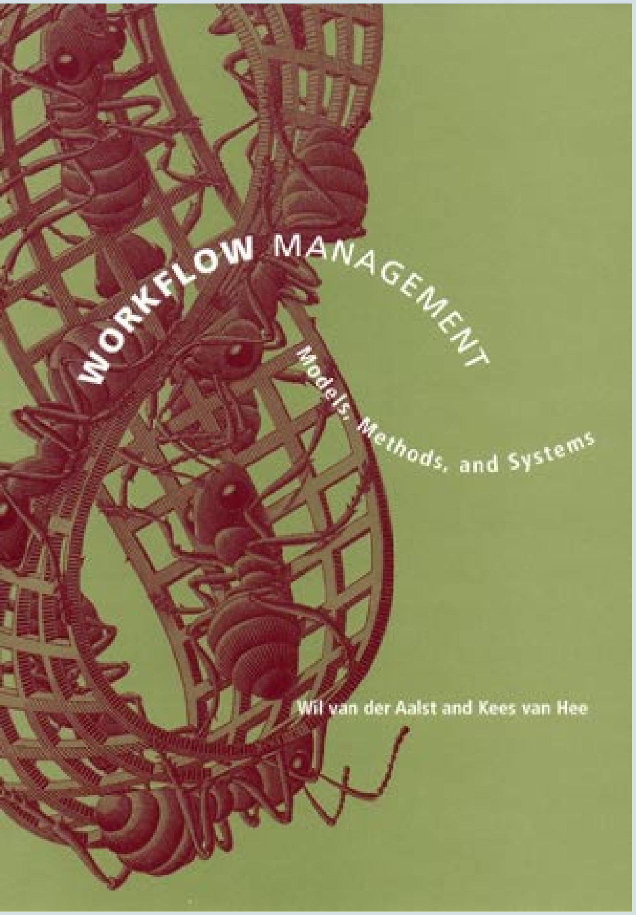
Interesting alternatives to the basic alpha algorithm are being developed.

More information

http://www.workflowcourse.com

http://www.workflowpatterns.com

http://www.processmining.org



W.M.P. van der Aalst and K.M. van Hee.
Workflow Management: Models, Methods, and Systems.
MIT press, Cambridge, MA, 2002/2004.