Tutorial on Business Process Management

prof.dr.ir. Wil van der Aalst
vdaalst.com
Overview

"world" business processes
people services components organizations
models analyzes
specifies configures implements analyzes
supports/controls
records events, e.g., messages, transactions, etc.

(process) model
(software) system
e.g., systems like WebSphere, Oracle, TIBCO/Staffware, SAP, FLOWer, etc.

ed., dedicated formats such as IBM's Common Event Infrastructure (CEI) and MXML or proprietary formats stored in flat files or database tables.

design-time analysis
verification
performance analysis
run-time analysis

e.g. process models represented in BPMN, BPEL, EPCs, Petri nets, UML AD, etc. or other types of models such as social networks, organizational networks, decision trees, etc.
BPM Systems
WfMC Reference model (early nineties)
Basic Idea

Workflow (2)

Click on a work-item to select a piece of work for a specific case.

D:\www\wvdaalst\workflowcourse\examples.htm
www.workflowcourse.com
Examples of workflow/BPM systems

- BPM|one
- YAWL
- COSA
- FileNet
- Staffware
- Oracle BPEL
- SAP Workflow
- …
Focus on "classical" workflow management systems, but ... 

1. Information systems with hard-coded workflows (process & organization specific).

2. Custom-made information systems with generic workflow support (organization specific).

3. Generic software with embedded workflow functionality (e.g., the workflow components of ERP, CRM, PDM, etc. systems).

4. Generic software focusing on workflow functionality (e.g., Staffware, MQSeries Workflow, FLOWer, COSA, Oracle BPEL, Filenet, etc.).
The power of expressiveness

Modern Business Process Automation
YAWL and its Support Environment

Arthur H. M. ter Hofstede
Wil M. P. van der Aalst
Michael Adams
Nick Russell (Tak)

[Images of authors]
Workflow Patterns Initiative

• Started in 1999, joint work TU/e and QUT

• Objectives:
  • Identification of workflow modeling scenarios and solutions
  • Benchmarking
    − Workflow products (MQ/Series Workflow, Staffware, etc)
    − Proposed standards for web service composition (BPML, BPEL)
    − Process modelling languages (UML, BPMN)
  • Foundation for selecting workflow solutions

• Home Page: www.workflowpatterns.com

• Primary publication (most cited workflow paper ever):

• Evaluations of commercial offerings, research prototypes, proposed standards for web service composition, etc
YAWL vs Petri nets/simple notations

• Petri nets have difficulties capturing:
  • The General Synchronizing Merge (OR-join)
  • Patterns involving Multiple Instances (order containing order lines)
  • Cancellation of a certain part of a process (reset arcs in PN terms)

• For the Control Flow Perspective, YAWL takes some concepts from Petri nets and adds constructs for:
  • OR-join to deal with General Synchronizing Merge
  • Multiple Instance (MI) tasks
  • Cancellation regions
  • “Syntactic Sugar” (various splits/joins, direct connections between tasks)
  • Some of this Syntactic Sugar was already introduced.
YAWL notation

- **Composite task**
- **Multiple Instance task**

- Condition
- Start condition
- End condition
- XOR-split task
- OR-split task
- AND-split task
- XOR-join task
- OR-join task
- AND-join task
- Remove tokens
YAWL Editor
YAWL Control Centre

YAWL 2.0 Worklist: Carmen Brotsin - Windows Internet Explorer

www.yawlfoundation.org

Work Queues   Edit Profile   Team Queues   Admin Queues   Cases   Users   Org Data   Services   Logout

Offered (0)   Allocated (1)   Started (0)   Suspended (0)

Work Items

37 enter data_3

Specification:
- add yawl

Task:
- enter data

Case:
- 37

Status:
- Enabled

Created:
- Mar.14, 2009 15:57:03

Age:
- 00:01.154

Start

Deallocate

Delegate

Skip

Pile
Example step-by-step

- Processing of insurance claims involving registration, two checks, and a payment of rejection
- Five tasks:
  - register (register insurance claim)
  - checkA (check insurance policy)
  - checkB (check damage reported)
  - pay (pay for the damage)
  - reject (inform customer about rejection)
- Registration is followed by two checks which can be handled in parallel.
- Each of the checks results in “OK” or “not OK”.
- If both are OK, pay otherwise reject.
- No waste of work, i.e., stop after first "not OK".
- Three roles: register (for task register), checks (for both checks), and pay/reject (for final tasks).
Starting YAWL

- Start Engine (Select "Start Engine" from start menu).
- Start Editor (Select "YAWL-Editor" from start menu).
- Create new specification (Select Create Specification in Editor's menu)
Connecting and logging in

• In Editor connect to engine and resource service (under "Tools").
• The user name is always "admin" and the password is "YAWL".
• Also use this to access the run-time of YAWL (to administrate of execute work)
Control and data-flow in YAWL
single start

places/conditions may be dropped

single end

cancels red parts
Net Decomposition Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>string</td>
<td>Local</td>
</tr>
<tr>
<td>InsuranceOK</td>
<td>boolean</td>
<td>Local</td>
</tr>
<tr>
<td>DamageOK</td>
<td>boolean</td>
<td>Local</td>
</tr>
</tbody>
</table>
provide name

determine split/join type

set cancelation region

select/create lower level

mapping data down and up

roles, etc.
mapping from net level to task level and from task level to net level

only one mapping from task level to net level

task variable
mapping from net level to task level and from task level to net level
Task without task decomposition but with a cancellation region
Analysis in YAWL
save

basic check
(syntax/completeness)

soundness check
No problems were discovered in the analysis of this specification.
ResetNet Analysis Warning: The net Main Insurance Process does not have proper completion. A marking 2OutputCondition_2 larger than ...
ResetNet Analysis Warning: The net Main Insurance Process does not satisfy the soundness property.

ResetNet Analysis Warning: The net Main Insurance Process does not have proper completion. A marking 2OutputCondition_2 larger than the final marking is reachable. The net Main Insurance Process can deadlock at marking(s): 1c(check_damage_5_pay_6)+1OutputCondition_2
1c(check_insurance_4_pay_6)+1OutputCondition_2

Drag the visible window to another area of this net.
ResetNet Analysis Warning: The net Main_Insurance_Process does not have proper completion. A marking 2OutputCondition_2 larger than...
Main Insurance Process

1. Start
2. Register
3. Check insurance
4. Check damage
5. Decide
6. Pay
7. Cancel
8. End

Flow detail for Atomic Task "decide"

<table>
<thead>
<tr>
<th>Target Task</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>pay</td>
<td>(/Main_Insurance_Process/InsuranceK/text())='true'</td>
</tr>
<tr>
<td>cancel</td>
<td>(/Main_Insurance_Process/InsuranceK/text())='true'</td>
</tr>
</tbody>
</table>

The bottom-most flow will be used as the default.

Notes (decide)

No problems were discovered in the analysis of this specification.

Drag the visible window to another area of this net.
Resources in YAWL

The power of expressiveness

Where innovation starts
Resource information resides at two places:
Editor and YAWL Control Center

(1) Editor: specify roles, etc. per task

(2) Control Center: Users and Org Data

http://localhost:8080/resourceService/faces/Logi
users

add new user

privileges

roles

roles of this user

add/remove

positions

capabilities

add new user
Some hints

• Make sure to complete the organizational model before referring to it in the Editor.
• The Editor needs to connect to the Engine and Resource service (default usercode: admin and password: YAWL).
• Remember the login data for added users!
• The runtime user interface (YAWL Control Center) depends on whether one is administrator or not!
Lifecycle of work-items in YAWL
Step 1: Choose Behaviour At Interaction Points

There are three key decision points for managing the resourcing of work items spawned from a task. At each of these interaction points, you may choose to have the system dynamic make a decision on resourcing at each point, or alternately, allow a user to manually make each decision. Each interaction point is briefly described below:

- **Offer**: The point at which it is decided that a number of participants *could* undertake the work item.
- **Allocation**: The point at which one of the participants offered the work item is *nominated to do* that work item.
- **Start**: The point at which the participant allocated a work item *begins* working on it.

Offering a work item for this task to a number of participants is to be done by:  
- [ ] User  
- [ ] System  

Allocating a work item for this task to one of the offered participants is to be done by:  
- [ ] User  
- [ ] System  

Starting an allocated work item of this task is to be done by:  
- [ ] User  
- [ ] System  

Default setting is (System, User, User)
Step 2: Specify System Behaviour when Offering a Work Item

The offer process involves choosing which participants should be informed of the existence of the work item, one of whom will eventually do this work. As you have specified the system manage the offer process, you must now choose who the work item should be offered to. Begin by specifying a set of participants and/or to distribute offers of work to. You may also specify a net parameter which at runtime will contain a participant’s userid or the name of a role.

Participants:
- Wil van der Aalst (wvdalaalst)
- JC Bose (jbose)
- Carmen Bratosin (cbbratosin)
- Kees van Hee (kvheee)
- Eric Verbeek (everbeek)
- Jaap van der Woude (jvdwoude)

Roles:
- check role
- final tasks role
- register role

Net Parameters:

Unselect All
Step 3: Specify Distribution Set Filter(s)

Select the filters to be applied to the distribution set. Set parameter values for the selected filter as required.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter by Capability</td>
<td>OrgGroup</td>
<td>AIS</td>
</tr>
<tr>
<td>Filter by Organisational Data</td>
<td>Position</td>
<td>professor</td>
</tr>
</tbody>
</table>

Runtime Constraints

- [ ] Allow this task to be parsed to a single participant.
- [ ] Choose participant(s) who completed previous task:
- [ ] Do not choose participant(s) who completed previous task:
Step 4: Specify System Behaviour when Allocating a Work Item

The allocation process involves choosing a single participant, from those who are offered a work item, to actually undertake that work. As you have specified that the system dynamically do this, you must now select the strategy for doing so. Choose from one of the strategies below.

Choose the runtime allocation strategy:
- Random Choice
- Shortest Queue
- Round Robin (by frequency)
- Random Choice
- Round Robin (by time)
Step 5: Establish Default User Runtime Privileges for this Task

Can a participant suspend a started work item of this task?  
- No  - Yes

Can a participant reallocate a work item of this task to another participant, resetting state?  
- No  - Yes

Can a participant reallocate a work item of this task to another participant, retaining state?  
- No  - Yes

Can a participant deallocate themselves from a work item of this task?  
- No  - Yes

Can a participant delegate a work item of this task to another participant?  
- No  - Yes

Can a participant skip a work item of this task?  
- No  - Yes
Alternative scenarios
Step 1: Interaction Points

Each task passes through three decisions, or interaction points before a participant begins working on it. For each of the interaction points below, please specify whether the task is to be handled by the System (dynamically, based on the settings chosen later in this wizard), or by the User (manually, by a participant or an administrator), when the task is executed.

Offer: The task is made available to a number of participants:
- User
- System

Allocate: The task is assigned to a single participant:
- User
- System

Start: Work begins on the task:
- User
- System

Step 5: Establish Default User Runtime Privileges for this Task

- Can a participant suspend a started work item of this task? Yes/No
- Can a participant reallocate a work item of this task to another participant, resetting state? Yes/No
- Can a participant reallocate a work item of this task to another participant, retaining state? Yes/No
- Can a participant deallocate themselves from a work item of this task? Yes/No
- Can a participant delegate a work item of this task to another participant? Yes/No
- Can a participant skip a work item of this task? Yes/No
Step 1: Interaction Points

Each task passes through three decisions, or interaction points before a participant.

Step 4: Specify System Behaviour when Allocating a Work Item

The allocation process involves options for how participants can interact with the work. As you have specified that there are no interaction points, you can choose small strategy options.

Step 5: Establish Default User Runtime Privileges for this Task

- Can a participant suspend a started work item of this task? [No] [Yes]
- Can a participant reallocate a work item of this task to another participant, resetting state? [No] [Yes]
- Can a participant reallocate a work item of this task to another participant, retaining state? [No] [Yes]
- Can a participant deallocate themselves from a work item of this task? [No] [Yes]
- Can a participant delegate a work item of this task to another participant? [No] [Yes]
- Can a participant skip a work item of this task? [No] [Yes]
Executing work in YAWL
Loading processes and creating cases
Specification ID: insurance.yawl, Net ID: Main_Insurance_Process
Edit Work Item: 32.1

register
name: Case Jansen

[Buttons: Cancel, Save, Complete]
Specification ID: insurance.yawl, Net ID: Main_Insurance_Process

start → register → check insurance → check damage → c → dummy → pay → cancel → end
<table>
<thead>
<tr>
<th>Work Items</th>
<th>Specification</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.check_insurance_4</td>
<td>insurance.yawl</td>
<td>check insurance</td>
</tr>
</tbody>
</table>

**Specifications:**
- **Case:** 32
- **Status:** Enabled
- **Created:** Mar.14, 2009 15:25:08
- **Age:** 0:00:02:50

**Buttons:**
- Start
- Deallocate
- Delegate
- Skip
- Pile
Edit Work Item: 32.2

check insurance

insuranceOK:  
name:  Case Jansen

Cancel  Save  Complete
Specification ID: insurance.yawl, Net ID: Main_Insurance_Process

Start -> Register

Register -> Check Insurance

Check Insurance -> C

C -> Dummy

Dummy -> Cancel

End
Edit Work Item: 32.4

cancel
name: Case Jansen

[Buttons: Cancel, Save, Complete]
Completed!
Alternative paths
Design-time analysis of processes

linear algebraic analysis techniques
Markov chain analysis techniques
state-space analysis techniques
simulation
Questions

• Can the workflow deadlock?
• Is it always possible to terminate?
• Is a livelock possible?
• Are there "dead" activities?
• Is A always followed by B?
• Is it possible to terminate without doing X?
• Can Y be executed more than 4 times?
• Can C and D be executed in parallel?
• ...

Context independent notion: **Soundness**

Hot topic! Many approaches/results!
Case study
Approach

2003 non-trivial process models

collect characteristics

model analysis

compare

Thanks to Jan Mendling!
The SAP reference model contains more than 600 non-trivial process models expressed in terms of Event-driven Process Chains (EPCs).

<table>
<thead>
<tr>
<th>Hierarchy Level</th>
<th>Models</th>
<th>eEPC</th>
<th>Function Allocation Diagram</th>
<th>Process Selection Diagram</th>
<th>Role Activity Diagram</th>
<th>EPC</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>58</td>
<td>29</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>175</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>102</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>1226</td>
<td>724</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>502</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>8384</td>
<td>3035</td>
<td>3035</td>
<td>0</td>
<td>2014</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>All Levels</td>
<td>9844</td>
<td>3862</td>
<td>3035</td>
<td>29</td>
<td>2014</td>
<td>604</td>
<td>34</td>
</tr>
</tbody>
</table>
Simplistic approach: YAWL + Petri net invariants
Simplistic approach: YAWL + invariants

Analysis using transition invariants, i.e., only lower bound! ProM allows for more precise analysis
<table>
<thead>
<tr>
<th>Branch</th>
<th>Model</th>
<th>%</th>
<th>EPC</th>
<th>% EAV</th>
<th>EAV</th>
<th>CAV</th>
<th>AAV</th>
<th>Cycle</th>
<th>Error</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset Accounting</td>
<td>461</td>
<td>4.7%</td>
<td>43</td>
<td>7.1%</td>
<td>13.9</td>
<td>4.0</td>
<td>5.2</td>
<td>23.3</td>
<td>0</td>
<td>7 16.3%</td>
</tr>
<tr>
<td>Benefits Administration</td>
<td>50</td>
<td>0.5%</td>
<td>6</td>
<td>1.0%</td>
<td>9.5</td>
<td>3.3</td>
<td>5.8</td>
<td>19.7</td>
<td>3</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Compensation Management</td>
<td>122</td>
<td>1.2%</td>
<td>18</td>
<td>3.0%</td>
<td>7.6</td>
<td>3.4</td>
<td>3.3</td>
<td>13.7</td>
<td>3</td>
<td>1 5.6%</td>
</tr>
<tr>
<td>Customer Service</td>
<td>402</td>
<td>4.1%</td>
<td>41</td>
<td>6.8%</td>
<td>16.5</td>
<td>3.6</td>
<td>9.0</td>
<td>29.5</td>
<td>3</td>
<td>1 2.4%</td>
</tr>
<tr>
<td>Enterprise Core</td>
<td>0.1</td>
<td>6.1</td>
<td>32.1</td>
<td>0.0%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3 13.6%</td>
</tr>
<tr>
<td>Environment,</td>
<td>2.7</td>
<td>1.2</td>
<td>7.0</td>
<td>0.0%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Financial Acc</td>
<td>184</td>
<td>1.9%</td>
<td>3</td>
<td>0.5%</td>
<td>15.0</td>
<td>7.0</td>
<td>6.0</td>
<td>28.0</td>
<td>2</td>
<td>0 5.6%</td>
</tr>
<tr>
<td>Position Mana</td>
<td>37</td>
<td>0.4%</td>
<td>5</td>
<td>0.8%</td>
<td>12.0</td>
<td>3.0</td>
<td>6.6</td>
<td>24.0</td>
<td>3</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Inventory Management</td>
<td>541</td>
<td>5.5%</td>
<td>7</td>
<td>1.2%</td>
<td>5.7</td>
<td>3.1</td>
<td>2.1</td>
<td>11.4</td>
<td>0</td>
<td>1 14.3%</td>
</tr>
<tr>
<td>Organizational Management</td>
<td>15</td>
<td>0.2%</td>
<td>4</td>
<td>0.7%</td>
<td>7.3</td>
<td>1.5</td>
<td>4.0</td>
<td>12.3</td>
<td>0</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Personnel Administration</td>
<td>60</td>
<td>0.6%</td>
<td>10</td>
<td>1.7%</td>
<td>6.7</td>
<td>4</td>
<td>4</td>
<td>15.6</td>
<td>3</td>
<td>1 10.0%</td>
</tr>
<tr>
<td>Personnel Development</td>
<td>87</td>
<td>0.9%</td>
<td>1</td>
<td>0.2%</td>
<td>3</td>
<td>1.9</td>
<td>5.7</td>
<td>19.5</td>
<td>1</td>
<td>2 16.7%</td>
</tr>
<tr>
<td>Personnel Time Management</td>
<td>309</td>
<td>4.1%</td>
<td>3</td>
<td>0.8%</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>37.8</td>
<td>9</td>
<td>1 2.9%</td>
</tr>
<tr>
<td>Plant Maintenance</td>
<td>444</td>
<td>4.5%</td>
<td>7</td>
<td>1.2%</td>
<td>12.4</td>
<td>3.8</td>
<td>2.2</td>
<td>14.0</td>
<td>0</td>
<td>0 5.4%</td>
</tr>
<tr>
<td>Procurement</td>
<td>366</td>
<td>3.7%</td>
<td>20</td>
<td>3.4%</td>
<td>1.8</td>
<td>0.8</td>
<td>2.2</td>
<td>13.7</td>
<td>0</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Product Data Management</td>
<td>296</td>
<td>3.0%</td>
<td>17</td>
<td>2.8%</td>
<td>8.8</td>
<td>3.0</td>
<td>2.9</td>
<td>13.7</td>
<td>0</td>
<td>1 5.9%</td>
</tr>
<tr>
<td>Production</td>
<td>194</td>
<td>2.0%</td>
<td>17</td>
<td>2.8%</td>
<td>5.7</td>
<td>2.9</td>
<td>3.0</td>
<td>11.5</td>
<td>0</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Production Planning</td>
<td>347</td>
<td>3.5%</td>
<td>36</td>
<td>6.0%</td>
<td>8.5</td>
<td>3.8</td>
<td>2.2</td>
<td>14.0</td>
<td>0</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Project Management</td>
<td>209</td>
<td>2.1%</td>
<td>20</td>
<td>3.3%</td>
<td>20.5</td>
<td>3.8</td>
<td>11.7</td>
<td>37.8</td>
<td>1</td>
<td>1 5.0%</td>
</tr>
<tr>
<td>Quality Management</td>
<td>169</td>
<td>1.7%</td>
<td>6</td>
<td>1.0%</td>
<td>12.7</td>
<td>6.5</td>
<td>7.3</td>
<td>27.0</td>
<td>1</td>
<td>1 16.7%</td>
</tr>
<tr>
<td>Real Estate Management</td>
<td>56</td>
<td>0.6%</td>
<td>9</td>
<td>1.5%</td>
<td>7.4</td>
<td>2.6</td>
<td>4.1</td>
<td>13.8</td>
<td>3</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Recruitment</td>
<td>842</td>
<td>8.6%</td>
<td>1</td>
<td>0.2%</td>
<td>7.0</td>
<td>5.0</td>
<td>2.0</td>
<td>11.0</td>
<td>0</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Retail</td>
<td>568</td>
<td>5.8%</td>
<td>19</td>
<td>3.1%</td>
<td>16.5</td>
<td>10.2</td>
<td>7.9</td>
<td>36.0</td>
<td>1</td>
<td>1 5.3%</td>
</tr>
<tr>
<td>Revenue &amp; Cost Controlling</td>
<td>703</td>
<td>7.1%</td>
<td>76</td>
<td>12.6%</td>
<td>10.6</td>
<td>3.1</td>
<td>4.3</td>
<td>16.6</td>
<td>0</td>
<td>1 1.3%</td>
</tr>
<tr>
<td>Sales &amp; Distribution</td>
<td>95</td>
<td>1.0%</td>
<td>12</td>
<td>2.0%</td>
<td>13.0</td>
<td>2.7</td>
<td>6.2</td>
<td>22.2</td>
<td>0</td>
<td>1 8.3%</td>
</tr>
<tr>
<td>Training &amp; Event Management</td>
<td>116</td>
<td>1.2%</td>
<td>1</td>
<td>0.2%</td>
<td>24.0</td>
<td>7.0</td>
<td>16.0</td>
<td>48.0</td>
<td>0</td>
<td>0 0.0%</td>
</tr>
<tr>
<td>Travel Management</td>
<td>1761</td>
<td>17.9%</td>
<td>48</td>
<td>7.9%</td>
<td>10.5</td>
<td>3.5</td>
<td>4.5</td>
<td>18.1</td>
<td>0</td>
<td>6 12.5%</td>
</tr>
<tr>
<td>All 29 Branches</td>
<td>9844</td>
<td>100%</td>
<td>604</td>
<td>100%</td>
<td>11.5</td>
<td>4.0</td>
<td>5.2</td>
<td>20.8</td>
<td>33</td>
<td>34 5.6%</td>
</tr>
</tbody>
</table>
Example EPC running example (taken from SAP's reference model)
A crash course in EPCs ...
EPC Behavior

example path

starting point: 2003 EPCs

1. **SAP Reference Model**: The first collection of EPCs is the SAP Reference Model. The development of the SAP reference model started in 1992 and first models were presented at CEBIT’93 [10, p.VII]. We use the SAP reference model in its version from 2000 that includes 604 non-trivial EPCs.

2. **Service Model**: The second collection of EPCs stems from a German process reengineering project in the service sector. The project was carried out in the late 1990s. The models of this project include 381 non-trivial EPCs.

3. **Finance Model**: The third model collection contains EPCs of a process documentation project in the Austrian financial industry. It includes 935 EPCs.

4. **Consulting Model**: The fourth collection covers in total 83 EPCs from three different consulting companies.
2003 non-trivial process models

model analysis

collect characteristics

compare

EPC Soundness

Start and End Events
Reachability Graph
Calculation with ProM

analyzing soundness

Reduction Rules with xoEPC
Cost Reposting
Internal procurement to be triggered
Accounts payable/down payments to be triggered
Goods Issue Processing
[Stock Material]
Manual Entry of Statistical Key Figures
[Investment Project]
Open item is based on order
Funds are reserved
Manual Funds Reduction
Reserved funds were reduced
Open item was detailed
Goods issue is planned
Material is reserved
Goods issue is posted
Internal Activity Should Be Allocated
Internally
Posting error from primary system occurred
Statistical figures are to be entered
Internal Activity Was Allocated
Budget is assigned
Project Release
Wrongly-assigned object entered
Statistical key figure was entered for reference
Funds for anticipated costs are to be set aside
Reservation Processing
Object to be assigned was debited
Project is released
Manual Funds Reservation
Reserved funds are to be reduced
Actual costs incurred are known
Direct Activity Allocation
Stock material is to be reserved
Posting was rejected due to budget overrun
10.7% incorrect
<table>
<thead>
<tr>
<th>Parameter Parameter</th>
<th>Complete Sample</th>
<th>SAP Ref. Model</th>
<th>Services Model</th>
<th>Finance Model</th>
<th>Consulting Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>xoEPC errors</td>
<td>154</td>
<td>90</td>
<td>28</td>
<td>26</td>
<td>10</td>
</tr>
<tr>
<td>Unreduced EPCs</td>
<td>156</td>
<td>103</td>
<td>18</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>ProM error EPCs</td>
<td>115</td>
<td>75</td>
<td>16</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>EPCs with errors</td>
<td>215</td>
<td>126</td>
<td>37</td>
<td>31</td>
<td>21</td>
</tr>
<tr>
<td>EPCs in total</td>
<td>2003</td>
<td>604</td>
<td>381</td>
<td>935</td>
<td>83</td>
</tr>
<tr>
<td>Error ratio</td>
<td><strong>10.7%</strong></td>
<td><strong>20.9%</strong></td>
<td><strong>9.7%</strong></td>
<td><strong>3.3%</strong></td>
<td><strong>25.3%</strong></td>
</tr>
</tbody>
</table>
2003 non-trivial process models

Collect characteristics

Model analysis

Compare

10.7% incorrect
Model Metrics:
Coefficient of Network Connectivity (CNC)

$$CNC(G) = \frac{|A|}{|N|} = 1.043$$

"arcs divided by nodes"
more arcs implies more complexity
Model Metrics: Structuredness (\(\Phi\))

\[
\Phi_N = 1 - \frac{S_N(G')}{S_N(G)} = 0.652
\]

"compares reduced and unreduced graph" more structure implies less complexity
Model Metrics: Connector Heterogeneity (CH)

\[ CH(G) = - \sum_{l \in \{\text{and}, \text{xor}, \text{or}\}} p(l) \cdot \log_3(p(l)) = 0.873 \]

"entropy over the different connector types" more heterogeneity implies more complexity
2003 non-trivial process models

collect characteristics

compare

model analysis

10.7% incorrect

28 metrics
Correlation of Metrics and Errors

Sample of 2003 EPCs from practice

\[ r_{\text{Phi,hasError}} = -0.36 \]

\[ r_{\text{CH,hasError}} = 0.46 \]
Logistic Regression Model

\[ p(EPC) = \frac{e^{\text{logit}(EPC)}}{1 + e^{\text{logit}(EPC)}} \]

\[ \text{logit}(EPC) = +4.008 \, CNC + 0.094 \, MM + 3.409 \, CYC - 2.338 \, \Pi - 9.957 \, \Phi + 3.003 \, CH + 0.064 \, diam \]

*Nagelkerke R^2 has an excellent value of 0.901*
Logistic Regression Results

\[ p_i = \frac{e^{\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}}}{1 + e^{\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}}} \]

\[ \ln\left( \frac{p_i}{1-p_i} \right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i} \]
Error Prediction

\[ P(EPC) = \frac{e^{\logit(EPC)}}{1 + e^{\logit(EPC)}} = 0.811 > 0.5 \]

<table>
<thead>
<tr>
<th>Observed</th>
<th>hasErrors</th>
<th>0</th>
<th>1</th>
<th>Percentage Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 9</td>
<td>hasErrors</td>
<td>1724</td>
<td>37</td>
<td>97.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58</td>
<td>155</td>
<td>72.8</td>
</tr>
<tr>
<td>Overall Percentage</td>
<td></td>
<td></td>
<td></td>
<td>95.2</td>
</tr>
</tbody>
</table>
Holdout Sample Results

• 113 EPCs from books by
  • Scheer,
  • Becker & Schütte,
  • Staud, and
  • Seidlmeier

• Accuracy interval for prediction function
  – 81.15% to 96.77%
  – with 95% confidence
Conclusion: case study

- Verification is mature!
- Models are not!
- Errors can be predicted!
- How to improve?
  - Take models seriously, otherwise do not bother.
  - Improve teaching.
  - Build smart editors, e.g., based on patterns and anti-patterns.
Process Mining
Design-time analysis vs run-time analysis

Design-time analysis

- models
- analyses
- specifies
- configures
- implements
- analyzes
- supports/
  controls
- discovery
- conformance
- extension
- run-time
  analysis

Run-time analysis

- models
- analyzes
- extends
- conforms
- monitors
- supports/
  controls
- event
  logs
- records
  events, e.g.,
  messages,
  transactions,
  etc.

Software system

- “world”
  - business processes
  - people
  - services
  - components
  - organizations

Validation

Verification

Performance analysis

e.g. process models
represented in BPMN,
BPEL, EPCs, Petri nets,
UML AD, etc. or other
types of models such as
social networks,
organizational networks,
decision trees, etc.

E.g., systems like
WebSphere,
Oracle, TIBCO/
Staffware, SAP,
FLOWer, etc.

E.g., dedicated formats
such as IBM’s
Common Event
Infrastructure (CEI) and
MXML or proprietary
formats stored in flat
files or database
tables.
Starting point: event logs

event logs, audit trails, databases, message logs, etc.

unified event log (MXML)
Event log:
- processes
  - process instances
  - events

Per event:
- activity name
- (event type)
- (originator)
- (timestamp)
- (data)
<ProcessInstance id="51" description="">
  - <AuditTrailEntry>
    <WorkflowModelElement>invite reviewers</WorkflowModelElement>
    <EventType>start</EventType>
    <Timestamp>2006-08-28T00:00:00.000+01:00</Timestamp>
    <Originator>Mike</Originator>
  </AuditTrailEntry>
  - <AuditTrailEntry>
    <WorkflowModelElement>invite reviewers</WorkflowModelElement>
    <EventType>complete</EventType>
    <Timestamp>2006-08-31T00:00:00.000+01:00</Timestamp>
    <Originator>Mike</Originator>
  </AuditTrailEntry>
  - <AuditTrailEntry>
    <Data>
      <Attribute name="result">reject</Attribute>
    </Data>
    <WorkflowModelElement>get review 3</WorkflowModelElement>
    <EventType>complete</EventType>
    <Timestamp>2006-09-02T00:00:00.000+01:00</Timestamp>
    <Originator>Mary</Originator>
  </AuditTrailEntry>
  - <AuditTrailEntry>
    <WorkflowModelElement>time out 1</WorkflowModelElement>
    <EventType>complete</EventType>
    <Timestamp>2006-09-03T00:00:00.000+01:00</Timestamp>
    <Originator/>
  </AuditTrailEntry>
</ProcessInstance>
Process Mining: The alpha algorithm

\( \alpha \) algorithm
Alpha algorithm
Without transactional information (just completes)
Example log

- Minimal information in log: case id’s and task id’s.
- Additional information: event type, time, resources, and data.
- Sequences:
  - 1: ABCD
  - 2: ACBD
  - 3: ABCD
  - 4: ACBD
  - 5: EF
- So this log there are three possible sequences:
  - ABCD
  - ACBD
  - EF

<table>
<thead>
<tr>
<th>Case</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>E</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
</tr>
</tbody>
</table>
>\rightarrow,||,# relations

- **Direct succession:** \( x > y \) iff for some case \( x \) is directly followed by \( y \).
- **Causality:** \( x \rightarrow y \) 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 \( y > x \).
Basic idea (1)

\[ x \rightarrow y \]
Basic idea (2)

\[ x \rightarrow y, \ x \rightarrow z, \text{ and } y \parallel z \]
Basic idea (3)

$x \rightarrow y, \ x \rightarrow z, \ and \ y \# z$
Basic idea (4)

\[ x \rightarrow z, \ y \rightarrow z, \ \text{and} \ x \parallel y \]
Basic idea (5)

\[ x \rightarrow z, \quad y \rightarrow z, \quad \text{and} \quad x \# y \]
Let $W$ be a workflow log over $T$. $\alpha(W)$ is defined as follows.

1. $T_W = \{ t \in T \mid \exists \sigma \in W \ t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists \sigma \in W \ t = \text{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists \sigma \in W \ t = \text{last}(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \land A \neq \emptyset \land B \subseteq T_W \land B \neq \emptyset \land \forall_{a \in A} \forall_{b \in B} a \rightarrow_W b \land \forall_{a_1,a_2 \in A} a_1 \#_W a_2 \land \forall_{b_1,b_2 \in B} b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \land B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W,o_W\}$,
7. $F_W = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_W \land a \in A \} \cup \{ (p_{(A,B)},b) \mid (A,B) \in Y_W \land b \in B \} \cup \{ (i_W,t) \mid t \in T_I \} \cup \{ (t,o_W) \mid t \in T_O \}$, and
8. $\alpha(W) = (P_W, T_W, F_W)$. 
Example revisited

W:
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 2 : 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

\[ \alpha(W) \]
Exercise (1)

- What does the alpha algorithm produce for a log consisting only of the following traces?
  - ABCD
  - ACBD
  - AED

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

1. $T_W = \{ t \in T \mid \exists \sigma \in W \ t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists \sigma \in W \ t = first(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists \sigma \in W \ t = last(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \land A \neq \emptyset \land B \subseteq T_W \land B \neq \emptyset \land \forall a \in A \forall b \in B \ a \rightarrow_W b \land \forall a_1,a_2 \in A \ a_1 \#_W a_2 \land \forall b_1,b_2 \in B \ b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall (A',B') \in X A \subseteq A' \land B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{ i_W, o_W \}$,
7. $F_W = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_W \land a \in A \} \cup \{ (p_{(A,B)},b) \mid (A,B) \in Y_W \land b \in B \} \cup \{ (i_W,t) \mid t \in T_I \} \cup \{ (t,o_W) \mid t \in T_O \}$, and
8. $\alpha(W) = (P_W, T_W, F_W)$. 

- Direct succession: $x>y$ iff for some case $x$ is directly followed by $y$.
- Causality: $x \rightarrow y$ 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 $y>x$. 
Definition 3.1 (Workflow trace, Workflow log). Let $T$ be a set of tasks. $\sigma \in T^*$ is a workflow trace and $W \in \mathcal{P}(T^*)$ is a workflow log.

The workflow trace of case 1 in Table 1 is $ABCD$. The workflow log corresponding to Table 1 is

$$\{ABCD, ACBD, AED\}.$$
Definition 3.2 (Log-based ordering relations). Let $W$ be a workflow log over $T$, i.e., $W \in \mathcal{P}(T^*)$. Let $a, b \in T$:

- $a >_W b$ iff there is a trace $\sigma = t_1t_2t_3 \ldots t_{n-1}$ and $i \in \{1, \ldots, n-2\}$ such that $\sigma \in W$ and $t_i = a$ and $t_{i+1} = b$.
- $a \rightarrow_W b$ iff $a >_W b$ and $b \not>_W a$.
- $a \#_W b$ iff $a \not>_W b$ and $b \not>_W a$, and
- $a \parallel_W b$ iff $a >_W b$ and $b >_W a$.

Consider the workflow log $W = \{ABCD, ACBD, AED\}$ (i.e., the log shown in Table 1). Relation $>_W$ describes which tasks appeared in sequence (one directly following the other). Clearly, $A >_W B$, $A >_W C$, $A >_W E$, $B >_W C$, $B >_W D$, $C >_W B$, $C >_W D$, and $E >_W D$. Relation $\rightarrow_W$ can be computed from $>_W$ and is referred to as the (direct) causal relation derived from workflow log $W$. $A \rightarrow_W B$, $A \rightarrow_W C$, $A \rightarrow_W E$, $B \rightarrow_W D$, $C \rightarrow_W D$, and $E \rightarrow_W D$. Note that $B \not\rightarrow_W C$ because $C >_W B$. Relation $\parallel_W$ suggests potential parallelism. For log $W$, tasks $B$ and $C$ seem to be in parallel, i.e., $B \parallel_W C$ and $C \parallel_W B$. If two tasks can follow each other directly in any order, then all possible interleavings are present and,
Definition 4.3 (Mining algorithm $\alpha$). Let $W$ be a workflow log over $T$. $\alpha(W)$ is defined as follows:

1. $T_W = \{t \in T \mid \exists \sigma \in W t \in \sigma\}$,
2. $T_I = \{t \in T \mid \exists \sigma \in W t = \text{first}(\sigma)\}$,
3. $T_O = \{t \in T \mid \exists \sigma \in W t = \text{last}(\sigma)\}$,

4. $X_W = \{(A, B) \mid \forall a \in A \forall b \in B \forall \sigma \in W (b \leq a \Rightarrow b' \in \sigma)\}$,

5. $Y_W = \{(A, I) \mid B \subseteq B' \Rightarrow (A, B) = (A', B')\}$,

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

7. $F_W = \{(a, p_{(A,B)}) \mid (A, B) \in Y_W \land a \in A\}$
   \[\cup \{(p_{(A,B)}, b) \mid (A, B) \in Y_W \land b \in B\}\]
   \[\cup \{(i_W, t) \mid t \in T_I\} \cup \{(t, o_W) \mid t \in T_O\}\],

and

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

$W = \{ABCD, ACBD, AED\}$
Definition 4.3 (Mining algorithm $\alpha$). Let $W$ be a workflow log over $T$. $\alpha(W)$ is defined as follows:

1. $T_W = \{ t \in T \mid \exists \sigma \in W t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists \sigma \in W t = first(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists \sigma \in W t = last(\sigma) \}$,
4. $X_W = \{ (A, B) \mid A \subseteq T_W \land B \subseteq T_W \land \forall a \in A \forall b \in B a \rightarrow_W b \land \forall a_1, a_2 \in A a_1 \neq_W a_2 \land \forall b_1, b_2 \in B b_1 \neq_W b_2 \}$,

5. 

4. 

$$X_W = \{(\{A\}, \{B\}), (\{A\}, \{C\}), (\{A\}, \{E\}), (\{B\}, \{D\}), (\{C\}, \{D\}), (\{E\}, \{D\}), (\{A\}, \{B, E\}), (\{A\}, \{C, E\}), (\{B, E\}, \{D\}), (\{C, E\}, \{D\}) \}.$$
4. \[ X_W = \{ (\{A\}, \{B\}), (\{A\}, \{C\}), (\{A\}, \{E\}), (\{B\}, \{D\}), (\{C\}, \{D\}), (\{E\}, \{D\}), (\{A\}, \{B, E\}), (\{A\}, \{C, E\}), (\{B, E\}, \{D\}), (\{C, E\}, \{D\}) \} \}, \]

\[ W = \{ ABCD, ACBD, AED \} \]

5. \[ Y_W = \{ (A, B) \in X_W \mid \forall (A', B') \in X_W \ A \subseteq A' \land B \subseteq B' \rightarrow (A, B) = (A', B') \}, \]

\[ Y_W = \{ (\{A\}, \{B, E\}), (\{A\}, \{C, E\}), (\{B, E\}, \{D\}), (\{C, E\}, \{D\}) \} \}, \]

and

\[ \alpha(W) = (P_W, T_W, F_W). \]
5.

\[ Y_W = \{(\{A\}, \{B, E\}), (\{A\}, \{C, E\}), (\{B, E\}, \{D\}), (\{C, E\}, \{D\})\} \]

\[ \land \forall a \in A \forall b \in B a \rightarrow_W b \land \forall a_1, a_2 \in A a_1 \#_W a_2 \land \forall b_1, b_2 \in B b_1 \#_W b_2 \}

\[ W = \{ABCD, ACBD, AED\} \]

\[ P_W = \{i_W, o_W, p(\{A\}, \{B, E\}), p(\{A\}, \{C, E\}), p(\{B, E\}, \{D\}), p(\{C, E\}, \{D\})\} \]

\[ F_W = \{(i_W, A), (A, p(\{A\}, \{B, E\})), (p(\{A\}, \{B, E\}), B), \ldots, (D, o_W)\} \]

6. \[ P_W = \{p_{(A, B)} \mid (A, B) \in Y_W\} \cup \{i_W, o_W\} \]

7. \[ F_W = \{(a, p_{(A, B)}) \mid (A, B) \in Y_W \}\}

\[ \cup \{(p_{(A, B)}, b) \mid (A, B) \in Y_W \}\]

\[ \cup \{(i_W, t) \mid t \in T_I\} \cup \{t, o_W\} \]

8. \[ \alpha(W) = (P_W, T_W, F_W) \]
Exercise (2)

• What does the alpha algorithm produce for a log consisting only of the following traces?
  • ACD
  • BCE

Let $W$ be a workflow log over $T$. $\alpha(W)$ is defined as follows.
1. $T_W = \{ t \in T \mid \exists \sigma \in W \ t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists \sigma \in W \ t = \text{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists \sigma \in W \ t = \text{last}(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \land A \neq \emptyset \land B \subseteq T_W \land B \neq \emptyset \land \forall a \in A \forall b \in B \ a \rightarrow_W b \land \forall a_1,a_2 \in A \ a_1 \#_W a_2 \land \forall b_1,b_2 \in B \ b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall (A',B') \in X A \subseteq A' \land B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{ i_W, o_W \}$,
7. $F_W = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_W \land a \in A \} \cup \{ (p_{(A,B)},b) \mid (A,B) \in Y_W \land b \in B \} \cup \{ (i_W,t) \mid t \in T_I \} \cup \{ (t,o_W) \mid t \in T_O \}$, and
8. $\alpha(W) = (P_W, T_W, F_W)$.
Exercise (3)

• What does the alpha algorithm produce for a log consisting only of the following traces?
  • ACEG
  • AECG
  • BDFG
  • BFDG

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

1. $T_W = \{ t \in T \mid \exists \sigma \in W t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists \sigma \in W t = \text{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists \sigma \in W t = \text{last}(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \land A \neq \emptyset \land B \subseteq T_W \land B \neq \emptyset \land \forall a \in A \forall b \in B a \rightarrow_w b \land \forall a_1,a_2 \in A a_1 \#_w a_2 \land \forall b_1,b_2 \in B b_1 \#_w b_2 \}$,
5. $Y_W = \{ (A,B) \mid (A,B) \in X \land \forall (A',B') \in X A \subseteq A' \land B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{ i_W, o_W \}$,
7. $F_W = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_W \land a \in A \} \cup \{ (p_{(A,B)},b) \mid (A,B) \in Y_W \land b \in B \} \cup \{ (i_W,t) \mid t \in T_I \} \cup \{ (t,o_W) \mid t \in T_O \}$, and
8. $\alpha(W) = (P_W,T_W,F_W)$. 

• Direct succession: $x>y$ iff for some case $x$ is directly followed by $y$.
• Causality: $x\rightarrow y$ 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 $y>x$. 
Properties of the Alpha algorithm

• If log is complete with respect to relation >, it can be used to mine any SWF-net!
• *Structured Workflow Nets* (SWF-nets) have no implicit places and the following two constructs cannot be used:

(Short loops require some refinement but not a problem.)
Alpha algorithm

- Mainly of theoretical interest!
- Too simple to be applicable to real-life logs.
- Does not address issues such as noise, etc.
- Should NOT be taken as a benchmark.
- However, the algorithm reveals:
  - basic process mining ideas and concepts in 8 lines,
  - theoretical limits of process mining.
Basic test for any mining algorithm: Rediscovery

Can the mined process generate all the behavior in the log?

How close is the behavior of the mined process to the original one?
Controlled choices cannot be rediscovered (and in many cases this is good!)

Fig. 7. The nonfree-choice WF-net $N_6$ cannot be rediscovered by the $\alpha$ algorithm.
Log only contains information about behavior and not structure

Fig. 8. WF-net $N_8$ cannot be rediscovered by the $\alpha$ algorithm. Nevertheless, $\alpha$ returns a WF-net which is behavioral equivalent.
Completeness notion may be too crude in some cases

Fig. 9. Although both WF-nets are not behavioral equivalent they are identical with respect to $>$. 
Another example of behaviorally equivalent SWF-nets

Fig. 10. Both SWF-nets are behavioral equivalent and, therefore, any algorithm will be unable to distinguish $N_{12}$ from $N_{13}$ (assuming a notion of completeness based on $>$).
Silent steps (and duplicate steps) cannot be discovered
Simple process mining algorithms tend to:

- Have problems with complex control-flow constructs. For example, many process mining algorithms are unable to deal with non-free-choice constructs and complex nested loops.
- Not allow for duplicates. In the event log it is not possible to distinguish between activities that are logged in a similar way, i.e., there are multiple activities that have the same “footprint” in the log. As a result, most algorithms map these different activities onto a simple activity thus making the model incorrect or counter-intuitive.
- Silent steps. Things that are not recorded cannot be discovered.
- Underfit (i.e., overgeneralize) or overfit. Many algorithms have a tendency to overgeneralize, i.e., the discovered model allows for much more behavior than actually recorded in the log. In some circumstances this may be desirable. However, there seems to be a need to flexibly balance between “overfitting” and “underfitting”.
- Yield inconsistent models. For more complicated processes many algorithms have a tendency to produce models that may have deadlocks and/or livelocks. It seems vital that the generated models satisfy some soundness requirements (e.g., the soundness property).
Other Process Discovery Techniques
Overview of process discovery techniques

- Classical techniques (e.g., learning state machines and the theory of regions): cannot handle concurrency and/or do not generalize (i.e., if it did not happen, it cannot happen).

- Algorithmic techniques
  - Alpha miner
  - Alpha+, Alpha++, Alpha#
  - Heuristic miner
  - Multi phase miner
  - ...

- Genetic process mining
- Region-based process mining
  - State-based regions
  - Language based regions
Genetic Mining
(Ana Karla Alves de Medeiros et al.)

1. initial population
2. fitness test
3. select best parents
4. crossover
5. children
6. mutation
7. new population
Design choices

1. initial population
2. fitness test
3. select best parents
4. crossover
5. children
6. mutation
7. new population

representation

fitness

crossover
Properties of Genetic Mining

• Requires a lot of computing power.
• Can deal with noise, infrequent behavior, duplicate tasks, invisible tasks, etc.
• Allows for incremental improvement and combinations with other approaches (heuristics post-optimization, etc.).
Balancing Between Overfitting and Underfitting
Challenge: Balancing Between Underfitting and Overfitting
The essence

ABCD
ACBD
AED
ABCD
ABCD
AED
ACBD
...

A ➔ B ➔ C ➔ D
Any log containing activities A, B, C, D, and E.
Finding a balance

(a) ACD BCE ...
(b) ACD ACE BCE BCD ...

(c) A → C → D
   B → E

(d) A → C → D
   B → E

more behavior
Important observations

• Frequencies matter!
• Adding a place equals restricting behavior!
• "The model" does not exist!
2712 patients
29258 events
264 different activities
874 patients
10478 events
181 different activities
24 machines
154966 events
360 different activities
Problems

- Representational bias (i.e., generalization is driven by representation rather than log or preferences).
- Inability of dealing with or detecting noise.
- Wrong abstraction level.
- Limitations of current process modeling (visualization) techniques.
Example: Heusden

State horizon set to last activity.
More significant nodes are emphasized

Highlights more important paths
More to learn from maps...

**Aggregation**
Clustering of coherent, less significant structures

**Abstraction**
Removing isolated, less significant structures
Fuzzy miner

check if sufficient information is available
start 1.000

check if sufficient information is available
complete 0.519

register claim
complete 0.009

determine likelihood of claim
start 0.610

Cluster 19
5 elements
~0.348

Cluster 21
14 elements
~0.408

end complete 1.000
Showing reality
Relevant WWW sites

- http://www.processmining.org
- http://www.workflowpatterns.com
- http://www.workflowcourse.com
- http://www.vdaalst.com
Exercises Process Discovery
Exercise A1

• From some transactional system the following event log consisting of six traces is extracted:
  - a b e f
  - a b e c d b f
  - a b c e d b f
  - a b c d e b f
  - a b c d e b f
  - a e b c d b f

• Use the 8 steps of the alpha-algorithm (included on last page) to construct the corresponding Petri-net and draw the Petri-net (delivering all of the intermediate results is not necessary, only the resulting Petri-net is required).

• Give a trace possible according to the discovered model but not (yet) observed in the log.
Exercise A2

• From some transactional system the following event log consisting of six traces is extracted:
  - a b c d f
  - a c b d f
  - a b d c f
  - a b d c f
  - a c d b f
  - a d e f
  - a e d f

• Use the 8 steps of the alpha-algorithm (included on last page) to construct the corresponding Petri-net and draw the Petri-net (delivering all of the intermediate results is not necessary, only the resulting Petri-net is required).

• Give a trace possible according to the discovered model but not (yet) observed in the log.
Exercise A3

• From some transactional system the following event log consisting of six traces is extracted:
  - $a\ d\ e\ f\ h$
  - $a\ e\ d\ f\ h$
  - $g\ h$
  - $a\ b\ c\ d\ f\ h$
  - $a\ b\ c\ d\ f\ h$
  - $a\ b\ d\ c\ f\ h$
  - $a\ c\ d\ b\ f\ h$

• Use the 8 steps of the alpha-algorithm (included on last page) to construct the corresponding Petri-net and draw the Petri-net (delivering all of the intermediate results is not necessary, only the resulting Petri-net is required).

• Give a trace possible according to the discovered model but not (yet) observed in the log.
Hands-on with Prom
Download/install ProM 5.2

The software can be downloaded from http://prom.win.tue.nl/tools/prom/.
(More information can be found at www.processmining.org and in case of problems one can try http://prom.win.tue.nl/tools/prom/nightly5/ instead.)
To make the exercises use the files in http://wwwis.win.tue.nl/~wvdaalst/ProM-examples.zip
Get files

- Load file ProM-examples.zip and unzip.
- Note the files txt and mxml files.
Redo exercises 1-3 (see slides)

• Inspect the files
  • exercise1.txt/exercise1.mxml,
  • exercise2.txt/exercise2.mxml, and
  • exercise3.txt/exercise3.mxml.

• Redo exercises using ProM 5.2.
Consider the following log:
- a b c d f
- a c b d f
- a b d c f
- a b d c f
- a c d b f
- a c d b f
- a d e f
- a e d f
- a e d f

Use the Alpha algorithm/ProM to discover the corresponding process model (Filename: exercise4.mxml)
Exercise (5)

• Consider the following log:
  • a b c d e f b d c e g
  • a b d c e g
  • a b c d e f b c d e f b d c e g

• Use the Alpha algorithm/ProM to discover the corresponding process model (Filename: exercise5.mxmi)
Exercise (6)

• Consider the following log:
  • a b e f
  • a b e c d b f
  • a b c e d b f
  • a b c e d b f
  • a b c d e b f
  • a e b c d b f

• Use the Alpha algorithm/ProM to discover the corresponding process model (Filename: exercise6.mxmxl)
Exercise (7)

• Load exercise7.mxml
• Inspect the log (e.g. using the internet explorer)
• Use the Alpha algorithm to discover the corresponding process model.
Explore ProM using exercise7.mxml

- **[Heuristics miner]** discover process model and play with settings
- **[Alpha algorithm plugin]** discover process model and play with settings
- **[Fuzzy Miner]** discover process model and explore the various views
- **[Fuzzy Miner]** animate the model
- **[Social Network Miner]** discover the social network
Explore ProM using exercise7.mxml (2)

- [Woflan Analysis] verify correctness of model
- [Conformance Checker] check the conformance of the mined model
- [Conformance Checker] modify the log (delete and insert events) and the check the conformance
- [Performance Analysis with Petri net] analyze the performance (where are the bottlenecks)
- [Basic Performance Analysis] analyze the performance (explore all options)
- [(Advanced) Dotted Chart Analysis] construct dotted charts and explore all options
Explore ProM using exercise7.mxml (3)

- Convert discovered Petri net into EPC model
- Convert discovered Petri net into YAWL model
- Convert discovered Petri net into heuristic net
If time is left, ....

- Repeat the process using exercise8.mxml
- Repeat the process using repairexample.zip and repairexamplesample2.zip
- Use the above two files to follow the steps described in the ProM Framework Tutorial