Workflow Management Systems and YAWL

prof.dr.ir. Wil van der Aalst
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.).
Basic idea

• Separation of control and execution.
Data inside a WFS
Interfaces

- Process Definition Tools
- Administration & Monitoring Tools
- Workflow Enactment Service
- Workflow Engine(s)
- Invoked Applications
- Other Workflow Enactment Service(s)
- Workflow Client Applications

Interfaces:
- Interface 1: weak
- Interface 2: OK
- Interface 3: X
- Interface 4: weak
- Interface 5: initial

Transport Layer Components:
- Interface 1: weak
- Interface 2: OK
- Interface 3: X
- Interface 4: weak
- Interface 5: initial
Potential problem

server

workflow engine

<table>
<thead>
<tr>
<th>workflow engine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

in-basket (worklist)

<table>
<thead>
<tr>
<th>in-basket (worklist)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

DBMS

<table>
<thead>
<tr>
<th>DBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

applications

<table>
<thead>
<tr>
<th>applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

client

In-basket

<table>
<thead>
<tr>
<th>In-basket</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

application

<table>
<thead>
<tr>
<th>application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

database

<table>
<thead>
<tr>
<th>database</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

multi-vendor

multi-platform

multi-system

multi-location
The **ACID**-properties, known from transaction processing, should hold.

- **Atomicity**
  (atomic, "everything or nothing", rollback if necessary)
- **Consistency**
  (a completed task results in a proper state of the system)
- **Isolation**
  (tasks do not affected each other, even if they are executed in parallel)
- **Durability**
  (the result of a completed task may not get lost; commit tasks)
Users of a WFS

- manager
- administrator
- process analyst
- workflow management system
- designer
- end-user
- database designer/programmer
- application designer/programmer
Examples of systems

- BPM|one
- YAWL
- COSA
- FileNet
- Staffware
- Oracle BPEL
- SAP Workflow
- Pegasystems
- Metastorm
- Global 360
- …
More players (Ignore ranking: $)

Figure 1. Magic Quadrant for Business Process Management Suites

- IBM 3x
- Oracle
- Staffware (part of TIBCO)
- BPM|one (now part of Perceptive software)
- SAP

Source: Gartner (October 2010)

Figure 1. Magic Quadrant for Business Intelligence Platforms

- Microsoft
- Tableau
- SAP

Source: Gartner (January 2011)
Example BPM systems
Design tools: Modeling control-flow, data, resources, etc.
Process model in Staffware

![Workflow Definer - DC5](image)

**Find Error**

Condition Definition

Condition: `DC DAMAGE = 'Y'`

[OK] [Cancel] [Help]
Alternative model in Staffware to correct problem
Staffware: Defining a step

**Step Definition**
- Name: REGISTER
- Description: register

**Step Status**
- Step: REGISTER - register
- Form Commands:
  - Initial: [field]
  - Keep: [field]
  - Release: [field]

**Deadline Definition**
- Step: REGISTER - register
- Deadlines on this step are defined by:
  - Period
  - Expression
- Deadline Period:
  - Years: 0
  - Months: 0
  - Weeks: 0
  - Days: 0
  - Hours: 0
  - Minutes: 0
- Withdraw form from queue on expiry

A deadline is set under either of these conditions:

**Step Priority**
- You may enter constant values or Staffware Expressions in each of the following priority definitions. Do this if you wish to change the priority definition of this step from the default SW_CP_xxx field defined priority definitions.
- Use Priority Value (1-999): SW_CP_VALUE
- Automatic Priority Escalation:
  - Use the following expressions to automatically escalate this step's priority from its base priority value when it is sent to a work queue. Note, to RAISE a priority of a step you must SUBTRACT from its priority value.
  - Increment (negative to raise): SW_CP_INCREMENT
  - Number of increments [-1=unlimited]: SW_CP_NUMINCR
  - Increment Period:
    - SW_CP_INCPEIOD
  - Period Type ("m"ins, "h"ours, "d"ays): SW_CP_PERIODTYPE

Permissions:
- [ ] Forward
- [ ] Edit
Staffware: Defining a step (2)
BPM|one: Process Designer

Pallas Athena

perceptive software
a Lexmark company
BPM|one: Simulation
BPM|one: Case Type Designer
BPM|one: Adding data and forms
BPM|one: Linking tasks to resource
BPM|one: process mining

Quick discovery - appeal demo

BPM|one®

Process (re)design
Process analysis
Process adjustment
Process monitoring
Process execution

inner circle (operational)
outer circle (tactical)
YAWL Editor

Main Insurance Process

- start
- register
- check insurance
- check damage
- pay
- dummy
- cancel
- end

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

Update Parameters for Atomic Task "register"

- **Input Parameters**
  - XQuery: TaskVariable

- **Net Variables**
  - Name: InsuranceOK, Type: boolean, Usage: Local
  - Name: DamageOK, Type: boolean, Usage: Local

- **Task Variables**
  - Name: name, Type: string, Usage: Output Only

- **Output Parameters**
  - XQuery: 
    - {{/register/name/text}}
  - NetVariable: Name

Update Task Decomposition "register"

- **Task Decomposition Label**: register

- **Task Decomposition Variables**
  - Name: name, Type: string, Usage: Output Only

YAWL Registered Service Detail

- **YAWL Service**: Default Engine Worklist

External Interaction

- **Automated**: OFF

Done

Cancel
Resources in YAWL

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 set parameter which at runtime will contain a participant's user ID or the name of a role.

Participants:
- Wil van der Aalst (vdaalst)
- JC Bosse (bosse)
- Carmen Bratosin (cbratosin)
- Kees van Hee (kheeu)
- Eric Verbeek (everbeek)
- Jaap van der Woud (jvdwoud)

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

Net Parameters:

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 to use. Choose one of the strategies below.

Choose the runtime allocation strategy:

Random Choice
Shortest Queue
Round Robin (by frequency)
Round Random Choice
Round Robin (by time)
No problems were discovered in the analysis of this specification.
SAP Business Workflow/Webflow (SAP AG)

- **sequence**
- **choices**
- **concurrency**
Organizational units have positions that may or may not be occupied.

Positions may be associated to multiple jobs (kind of role).

You can assign a task to

- an organizational unit, if it is to apply to all subordinate positions
- a job, if it is to apply to all positions described by the job
- a position, if it is to apply to those persons (employees) or users who hold the position
- a person (employee), if it is to apply to this person
COSA: Control flow perspective

Based on Petri nets
COSA: Resource perspective
Multiple organizational models in COSA
Oracle: Control flow perspective

- Based on BPEL

```xml
<sequence name="main"> <!-- Receive input from requestor. Note: This maps to operation defined in CFP23_flow.wadl -->
  <receive name="receiveInput" partnerLink="client"
    portType="client:CFP23_flow" operation="process"
    variable="inputVariable" createInstance="yes"/> <!-- Generate reply to synchronous request -->
  <assign name="Assign_4"/>
  <copy>
    <from variable="inputVariable" part="payload"
      query="/client:CFP23_flowProcessRequest/client:input"/>
    <to variable="outputVariable" part="payload"
      query="/client:CFP23_flowProcessResponse/client:result"/>
  </copy>
</assign>
<flow name="Flow_1">
  <sequence name="Sequence_3">
    <wait name="Wait_1" for="PT1M"/>
    <assign name="Assign_3"/>
    <copy>
```
Oracle: Resource perspective
Worklist handler: offering work to resources

- Process Definition Tools
- Administration & Monitoring Tools
- Interface 1
- Workflow API and Interchange formats
- Workflow Enactment Service
- Workflow Engine(s)
- Invoked Applications
- Interface 2
- Interface 3
- Interface 4
- Other Workflow Enactment Service(s)
- Workflow Engine(s)
- Interface 5
- Workflow Client Applications
Staffware: the end-user’s view
Selecting and executing the first step

This is the first step of the Double Check Process

Enter name:  Case Jansen
Executing one of the two parallel steps
Check insurance

insuranceOK: [ ]
name: Case Jansen

[Cancel] [Save] [Complete]
Administration and monitoring tools

- Process Definition Tools
- Interface 1
- Workflow API and Interchange formats
- Workflow Enactment Service
- Interface 2
- Workflow Engine(s)
- Interface 3
- Invoked Applications
- Interface 4
- Other Workflow Enactment Service(s)
- Interface 5
- Administration & Monitoring Tools

Workflow Engine(s)

Workflow Client Applications
Staffware: The manager/administrator’s view

- Managing users/groups
- Monitoring and managing processes and cases
Managing users
Managing groups
Managing processes and cases

Staffware Case Manager

Available Procedures

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Live (Dead) Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPLAIN</td>
<td>complaints handling</td>
<td>3{0}</td>
</tr>
<tr>
<td>CORR01</td>
<td>Correspondentie</td>
<td>0{0}</td>
</tr>
<tr>
<td>DC</td>
<td>Case Double Check</td>
<td>0{0}</td>
</tr>
<tr>
<td>DC0</td>
<td>Double Check process</td>
<td>0{0}</td>
</tr>
<tr>
<td>DC2</td>
<td>Double Check process</td>
<td>0{5}</td>
</tr>
<tr>
<td>DC3</td>
<td>double check process</td>
<td>0{0}</td>
</tr>
<tr>
<td>DC4</td>
<td>Double Check process</td>
<td>0{0}</td>
</tr>
<tr>
<td>DC5</td>
<td>Double Check process</td>
<td>0{0}</td>
</tr>
<tr>
<td>DC6</td>
<td>Double Check process</td>
<td>2{5}</td>
</tr>
<tr>
<td>ERIC</td>
<td>eric</td>
<td>1{0}</td>
</tr>
<tr>
<td>GLDOVK01</td>
<td>Geldleenovereenkomsten</td>
<td>0{0}</td>
</tr>
</tbody>
</table>

80 Procedure(s), 99 Case(s)  Selected : 1 procedure

Procedure : DC6 on Node staffw_edlbp : Case Action Window

Available Cases

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Mail</th>
<th>Start Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DC6 both OK</td>
<td>0</td>
<td>18/04/2002 10:53</td>
</tr>
<tr>
<td>2</td>
<td>DC6 one not OK</td>
<td>0</td>
<td>18/04/2002 10:58</td>
</tr>
<tr>
<td>3</td>
<td>DC6 - Case Jansen</td>
<td>0</td>
<td>18/04/2002 11:09</td>
</tr>
<tr>
<td>4</td>
<td>DC6 both not OK seq 0</td>
<td>0</td>
<td>18/04/2002 11:00</td>
</tr>
<tr>
<td>5</td>
<td>DC6 - Case Van Balen1</td>
<td>1</td>
<td>18/04/2002 11:11</td>
</tr>
<tr>
<td>6</td>
<td>DC6 both not OK par 0</td>
<td>0</td>
<td>18/04/2002 11:00</td>
</tr>
<tr>
<td>7</td>
<td>DC6 - case Pietersen 1</td>
<td>1</td>
<td>18/04/2002 11:09</td>
</tr>
</tbody>
</table>

7 Case(s)  Selected 1 Case(s)
Monitoring individual cases

Procedure: DC6 on Node staffw_edlbp: Case Action Window

Available Cases

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Mail</th>
<th>Start Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DC6 both OK</td>
<td>0</td>
<td>18/04/2002 10:53</td>
</tr>
<tr>
<td>2</td>
<td>DC6 one not OK</td>
<td>0</td>
<td>18/04/2002 10:58</td>
</tr>
<tr>
<td>3</td>
<td>DC6 both not OK</td>
<td>0</td>
<td>18/04/2002 11:00</td>
</tr>
<tr>
<td>4</td>
<td>DC6 - Case Van Balen</td>
<td>0</td>
<td>18/04/2002 11:09</td>
</tr>
<tr>
<td>5</td>
<td>DC6 - Case Jansen</td>
<td>0</td>
<td>18/04/2002 11:09</td>
</tr>
<tr>
<td>7</td>
<td>DC6 - case Pietersen</td>
<td>0</td>
<td>18/04/2002 11:14</td>
</tr>
</tbody>
</table>

Case Audit Trail: DC6 : (5) - DC6 - Case Jansen

Case Events were logged in the following order:

- 18/04/2002 11:09 Case started by swadmin
- 18/04/2002 11:09 "register" processed to DC_register
- 18/04/2002 11:13 "register" released by swadmin
- 18/04/2002 11:13 "check insurance" processed to DC_checks
- 18/04/2002 11:13 "check damage" processed to DC_checks
- 18/04/2002 11:14 "check insurance" released by swadmin
- 18/04/2002 11:15 "check damage" released by swadmin
- 18/04/2002 11:15 "pay for damage" processed to DC_pay/reject
- 18/04/2002 11:17 "pay for damage" released by swadmin
- 18/04/2002 11:17 Case terminated normally
BPM|one: Managing users, roles, etc.
YAWL: Monitoring running cases

<table>
<thead>
<tr>
<th>Case</th>
<th>Spec Name</th>
<th>Version</th>
<th>Start Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1143</td>
<td>orderfulfilment</td>
<td>1.3</td>
<td>2010-06-22 15:05:16</td>
</tr>
<tr>
<td>1144</td>
<td>orderfulfilment</td>
<td>1.3</td>
<td>2010-06-22 15:05:21</td>
</tr>
<tr>
<td>1145</td>
<td>ProcessGiveMoney</td>
<td>0.38</td>
<td>2010-06-22 15:05:22</td>
</tr>
<tr>
<td>1146</td>
<td>PurchaseOrder</td>
<td>0.8</td>
<td>2010-06-22 15:05:25</td>
</tr>
<tr>
<td>1147</td>
<td>PurchaseOrder</td>
<td>0.8</td>
<td>2010-06-22 15:05:28</td>
</tr>
<tr>
<td>1148</td>
<td>FilmProcess_YAWL2_0_noResource</td>
<td>3.8</td>
<td>2010-06-22 15:05:34</td>
</tr>
<tr>
<td>1149</td>
<td>orderfulfilment</td>
<td>1.3</td>
<td>2010-06-22 15:05:40</td>
</tr>
<tr>
<td>1150</td>
<td>Casualty_Treatment</td>
<td>0.1</td>
<td>2010-06-22 15:06:24</td>
</tr>
</tbody>
</table>
YAWL: Workitems of selected case

---

<table>
<thead>
<tr>
<th>CaseID v</th>
<th>TaskID</th>
<th>Status</th>
<th>Service</th>
<th>Enabled</th>
<th>Started</th>
<th>Completed</th>
<th>Timer</th>
<th>Expires</th>
</tr>
</thead>
<tbody>
<tr>
<td>1151.3</td>
<td>3_Treat</td>
<td>Executing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Start Date: 2010-06-22 15:27:32

Starting Service: DefaultWorklist

Started By: Tom Hagen
YAWL: Managing roles

![YAWL interface screenshot](image-url)
YAWL: Managing users
Overview of systems mentioned

Staffware

BPM|one

COSA

Oracle BPEL

YAWL

SAP Workflow
Mapping to the four types of "workflow-like" systems

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
explicitly structured
implicitly structured
ad-hoc structured
unstructured

production workflow

case handling

ad-hoc workflow

groupware

data-driven

process-driven

Oracle BPEL (Oracle)
Staffware (TIBCO)
SAP Business Workflow/Webflow (SAP AG)
COSA (COSA GmbH)
BPM|one (Pallas Athena)
YAWL
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Modeling</th>
<th>Control-Flow</th>
<th>Resources</th>
<th>Workflow Management Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>Simulation</td>
<td>Verification</td>
<td>Process Discovery</td>
<td>Conformance Checking</td>
</tr>
<tr>
<td>Configurable Process Models</td>
<td>Service Orientation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**YAWL**

- The power of expressiveness
- Modern Business Process Automation

- Group assignment (3 points in groups of 3 persons)
- Written exam (7 points)
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
Some control-flow patterns (4 of 43)

Play with these animations on www.workflowpatterns.com
Two data patterns (2 of 40)

Play with these animations on www.workflowpatterns.com
Two resource patterns (2 of 43)

Play with these animations on www.workflowpatterns.com
YAWL Initiative

- Driven by the workflow patterns.
- Language design to show how to support many patterns without adding unnecessary complexity.
- Serves as an example for standards and commercial products.
- One of the most successful open source workflow management systems (more than 100,000 downloads)
YAWL vs Petri nets

- 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.
Notation used thus far is a subset of YAWL
YAWL notation

Composite task

Multiple Instance task
Multiple instances example

In between 1 and 10 witness statements are processed, witnesses cannot be added after registration of them has finished (Pattern 14).
Multiple instances example

An arbitrary number of witnesses can be processed, furthermore, more witnesses can be incorporated after processing has started but before archiving (Pattern 15).
Multiple instances example

In between 1 and 10 witness statements are to be processed; after three statements have been processed, or all that were started initially, archiving can start (Pattern 9 extended).
How many payments?
OR-split/join example

How many payments?
OR splits and OR joins

- OR split can be mapped onto an AND-split followed by XOR-splits.
- OR join has the "bus driver semantics" (this causes a paradox and other problems).
• "Other OR-joins" are assumed to be XOR joins (optimistic semantics).
• Hence, F fires once.
• If pessimistic semantics would have been assumed (ANDs), then F would fire once or twice (cf. c4+c3:2 and c4+c5+c3:1)
The concept of cancellation region generalizes the cancel activity and cancel case patterns.

Syntactically, a cancellation region consists of a number of tasks and places (possibly including implicit ones!) part of the same composite task and attached to a so-called cancellation task (also part of the same composite task).

Semantically, upon completion of the cancellation task all tokens in the cancellation region (or in decompositions of tasks in that region etc) are removed.
Cancellation example

- register
- do_itinerary_segment
- pay
- cancel

booking_in_progress
Cancellation = set of "reset arcs"
Installing YAWL

YAWL (Version YAWL 2.3) can be downloaded from www.yawlfoundation.org or http://sourceforge.net/projects/yawl (sheets show version YAWL4Study-2.0.1; minimal differences)
Architecture of YAWL
YAWL Control Centre
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
places/conditions may be dropped

single start

cancels red parts

single end
net variables
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
Update Task Decomposition "cancel"

Task Decomposition Label: cancel

Task 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>Input Only</td>
</tr>
</tbody>
</table>

YAWL Registered Service Detail

YAWL Service: Default Engine Worklist

External Interaction

Automated

Update Parameters for Atomic Task "cancel"

Input Parameters

<table>
<thead>
<tr>
<th>XQuery</th>
<th>Task Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>(/Main_Insurance_Process/Man...) name</td>
<td></td>
</tr>
</tbody>
</table>

Net 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>

Task Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Input Only</td>
</tr>
</tbody>
</table>

Output Parameters

<table>
<thead>
<tr>
<th>XQuery</th>
<th>Net Variable</th>
</tr>
</thead>
</table>

Done | Cancel
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:
markings: 1(c(check_damage_5_pay_6) + 1OutputCondition_2) 
1c(check_insurance_4_pay_6) + 1OutputCondition_2
Main Insurance Process

- Start
- Register
- Decide
- Check Insurance
- Check Damage
- Pay
- Cancel
- End

Notes (decide): Specification Analysis Problems

No problems were discovered in the analysis of this specification.
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/Login.jsp
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
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 (wvdaalst)
- JC Bose (jbose)
- Carmen Bratosin (cbbratosin)
- Kees van Hee (kvhee)
- Eric Verbeek (everbeek)
- Jaap van der Woude (jvdwoude)

Roles

- check role
- final tasks role
- register role

Net Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Refers to</th>
</tr>
</thead>
</table>

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 piled 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?  
- 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
Step 1: Interaction Points

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 item or be offered to. Begin by selecting the participants which at runtime will be able to perform the work item.

Participants
Will van der Aaist
Boudewijn van de Donk
Kees van Hee
Eric Verbook

Step 3: Specify Distribution Set Filter(s)

Select the filters to be used for distribution.

Filters
- [ ] Filter by Category
- [X] Filter by Organization

Runtime Constraints

Step 5: Establish Default User Runtime Privileges for this Task

Can a participant suspend a started work item of this task?  [ ] No  [X] Yes

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

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

Can a participant deallocate themselves from a work item of this task?  [ ] No  [X] Yes

Can a participant delegate a work item of this task to another participant?  [ ] No  [X] Yes

Can a participant skip a work item of this task?  [ ] No  [X] Yes
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 defining the strategies for work. As you have specified that you want to use the strategies below.

Choose:

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
<table>
<thead>
<tr>
<th>Work Items</th>
<th>Specification</th>
<th>Task</th>
<th>Case</th>
<th>Status</th>
<th>Created</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.1:register_3</td>
<td>insurance.yawl</td>
<td>register</td>
<td>32.1</td>
<td>Executing</td>
<td>Mar.14, 2009 15:10:55</td>
<td>0:00:11:54</td>
</tr>
</tbody>
</table>
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**
- **check insurance**
- **check damage**
- **pay**
- **dummy**
- **cancel**
- **end**

Graph showing the process flow with nodes and edges labeled with actions.
Edit Work Item: 32.4

cancel

name:  Case Jansen

[Cancel] [Save] [Complete]
Completed!

Specification ID: insurance.yawl, Net ID: Main_Insurance_Process

start → register → check insurance → check damage → c → dummy → cancel → end
Alternative paths
Advanced Topics: Data and Time

The power of expressiveness
Data in YAWL

- XML Schema is used to define data types.
- XQuery is used for expressions, e.g., mappings and conditions.
- Resources:
  - http://www.w3schools.com/Schema/default.asp
  - http://www.w3.org/XML/Schema
  - http://www.w3schools.com/xquery/default.asp
  - http://www.w3.org/TR/xquery/
Basic XML types

- Built-in Datatype Hierarchy
- anyType
- all complex types
- anySimpleType
- duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
boolean
base64Binary
hexBinary
float
double
anyURI
QName
NOTATION
string
decimal
normalizedString
integer
token
nonPositiveInteger
long
nonNegativeInteger
language
Name
NMTOKEN
negativeInteger
int
unsignedLong
positiveInteger
NCName
NMTOKENS
short
unsignedInt
ID
IDREF
ENTITY
IDREFS
ENTITIES
byte
unsignedShort
unsignedByte

- ur types
- derived by restriction
- built-in primitive types
- derived by list
- built-in derived types
- derived by extension or restriction
- complex types
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="PersonType">
    <xs:sequence>
      <xs:element name="Name" type="xs:string" />
      <xs:element name="age" type="xs:integer" />
      <xs:element name="Street" type="xs:string" />
      <xs:element name="Town" type="xs:string" />
      <xs:element name="CountyOptional" type="xs:string" minOccurs="0" />
      <xs:element name="PostCode" type="xs:string" />
      <xs:element name="Country">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="NZD" />
            <xs:enumeration value="DE" />
            <xs:enumeration value="ES" />
            <xs:enumeration value="UK" />
            <xs:enumeration value="US" />
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="PersonType">
    <xs:sequence>
      <xs:element name="Name" type="xs:string" />
      <xs:element name="age" type="xs:integer" />
      <xs:element name="Street" type="xs:string" />
      <xs:element name="Town" type="xs:string" />
      <xs:element name="CountyOptional" type="xs:string" minOccurs="0" />
      <xs:element name="PostCode" type="xs:string" />
      <xs:element name="Country">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="NED" />
            <xs:enumeration value="DE" />
            <xs:enumeration value="ES" />
            <xs:enumeration value="UK" />
            <xs:enumeration value="US" />
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
Adding two numbers
Update Task Parameter "s"

from element of net variable: sum

XQuery

```xml
<s>
  {Sum_Two_Numbers/var1/text()+Sum_Two_Numbers/var2/text()}
</s>
```

populates the task variable: s

Done  Cancel
Edit Work Item: 47.2

Perform Sum

s: 999

Cancel  Save  Complete
Making a choice

[Diagram showing a decision process with states such as 'Enter Two Booleans', 'Both True', 'At least one Not True', and 'No Icon', with options for editing nets and decomposition variables.]
## Flow detail for Atomic Task "Enter Two Booleans"

<table>
<thead>
<tr>
<th>Target Task</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both True</td>
<td>PathSelectionBooleans/bo...</td>
</tr>
<tr>
<td>At least one Not True</td>
<td>true()</td>
</tr>
</tbody>
</table>

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

---

**Update Flow Predicate**

Net variable: `bool2`

XPath Expression:

```
/PathSelectionBooleans/bool1/text()='true' and /PathSelectionBooleans/bool2/text()='true'
```
Similar example
Input Parameters

<table>
<thead>
<tr>
<th>XQuery</th>
<th>Task Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Create...  Update...  Remove...

Net Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>num2</td>
<td>long</td>
<td>Local</td>
</tr>
<tr>
<td>num1</td>
<td>long</td>
<td>Local</td>
</tr>
</tbody>
</table>

Task Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>long</td>
<td>Output Only</td>
</tr>
<tr>
<td>n2</td>
<td>long</td>
<td>Output Only</td>
</tr>
</tbody>
</table>

Output Parameters

<table>
<thead>
<tr>
<th>XQuery</th>
<th>NetVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>{/Enter_Two_Numbers/n2/text()}</td>
<td>num2</td>
</tr>
<tr>
<td>{/Enter_Two_Numbers/n1/text()}</td>
<td>num1</td>
</tr>
</tbody>
</table>

Create...  Update...  Remove...  Done
## Flow detail for Atomic Task "Enter Two Numbers"

<table>
<thead>
<tr>
<th>Target Task</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumbersEqual</td>
<td><code>number(/PathSelectionNumbers/num1/text())</code></td>
</tr>
<tr>
<td>Numbers Not Equal</td>
<td><code>true()</code></td>
</tr>
</tbody>
</table>

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

## Update Flow Predicate

**Net variable:** num2

XPath Expression:

```
number(/PathSelectionNumbers/num1/text()) = number(/PathSelectionNumbers/num2/text())
```
Example with loop
On the input side take the sum of the previous sum and the number of iterations left.
On the output side decrease the number of iterations left.
Exit if no iterations are left.

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

Exit if no iterations are left.
Edit Work Item: 52.1

Enter Number

n: 3

Cancel  Save  Complete
PerformSum

n1: 1
s: 6

Edit Work Item: 52.4

Cancel  Save  Complete
Loop over list
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="StringList">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="5" name="str" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
update Net Parameter "noStrings"

Input Parameters
- XQuery:

Net Variables
- Name: noStrings, Type: integer, Usage: Local
- Name: StringList, Type: StringList, Usage: Local

Output Parameters
- XQuery:
  - {count(/Enter_String_List/strList/*)} noStrings
  - {/Enter_String_List/strList/*} StringList

populates the net variable: noStrings

Done
### Input Parameters

<table>
<thead>
<tr>
<th>XQuery</th>
<th>Task Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{/LoopOverStringList/StringList/*}</code></td>
<td>sList</td>
</tr>
<tr>
<td><code>{/LoopOverStringList/noStrings/number()-1}</code></td>
<td>noItems</td>
</tr>
</tbody>
</table>

### Net Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>noStrings</td>
<td>integer</td>
<td>Local</td>
</tr>
<tr>
<td>StringList</td>
<td>StringList</td>
<td>Local</td>
</tr>
</tbody>
</table>

### Task Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>sList</td>
<td>StringList</td>
<td>Input &amp; Output</td>
</tr>
<tr>
<td>noItems</td>
<td>integer</td>
<td>Input &amp; Output</td>
</tr>
</tbody>
</table>

### Output Parameters

<table>
<thead>
<tr>
<th>XQuery</th>
<th>NetVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{/Peform_Operation/noItems/number()-1}</code></td>
<td>noStrings</td>
</tr>
<tr>
<td><code>{/Peform_Operation/sList/*}</code></td>
<td>StringList</td>
</tr>
</tbody>
</table>
Flow detail for Atomic Task "Perform Operation"

<table>
<thead>
<tr>
<th>Target Task</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>/LoopOverStringList/noStrings/number() &gt; 0</td>
<td>true()</td>
</tr>
</tbody>
</table>

Final Task

The bottom-most flow will be used as the default.
Edit Work Item: 53.1

Enter String List

strList

str: aap

str: noot

str: mies

[Buttons: Cancel, Save, Complete]
Timed tasks

Examples: P1Y4M3DT23H55M1.5S, P2M3D, PT10S

The time interval is specified in the following form "PnYnMnDTnHnMnS" where:

- P indicates the period (required)
- nY indicates the number of years
- nM indicates the number of months
- nD indicates the number of days
- T indicates the start of a time section (required if you are going to specify hours, minutes, or seconds)
- nH indicates the number of hours
- nM indicates the number of minutes
- nS indicates the number of seconds
Semantics

- **Activation on enablement**
  - Manual task: Task follows normal life-cycle, but is forced into state completed.
  - Automated: like a delay.
- **Activation on starting (only for manual tasks)**
Task B

PnYnMnDTnHnMnS where
- P indicates the period (required)
- nY indicates the number of years
- nM indicates the number of months
- nD indicates the number of days
- T indicates the start of a time section
- nH indicates the number of hours
- nM indicates the number of minutes
- nS indicates the number of seconds

- Task is required to timeout
- at the time of 27/02/2011 14:54:27
- after a duration of PT10S
- Timer begins: upon work item enablement
Task C

- **start**
- **task A**
- **task B**
- **task C** (highlighted)
- **task D**
- **end**

**Update Task Decomposition “Task C”**

- Task Decomposition Variables
  - Name: string
  - Type: Input, Output

**Set Timer Detail for Atomic Task “task C”**

- **Task is required to timeout**
- **Timeout:**
  - at the time of: 27/02/2011, 14:48:06
  - after a duration of: PT1M

- **Timer begins:** upon work item enablement

**YAWL Registered Service Detail**

- YAWL Service: Default Engine Worklist

**External Interaction**

- Automated

[Explanations]

- PnYnMnDTnHnMnS indicates the period (required)
- nY indicates the number of years
- nM indicates the number of months
- nD indicates the number of days
- T indicates the start of a time section
- nH indicates the number of hours
- nM indicates the number of minutes
- nS indicates the number of seconds
Scenario 1

Edit Work Item: 4.1

Task A
- Text: Hello World

Start task A, then task B, followed by task C, and finally task D before ending the process.
after 1 minute and 10 seconds ...
Scenario 2

after 10 seconds ...
complete C within 1 minute
Another example using OR-joins
Alternative scenario: only hotel is needed but booking fails.
See example posted

YAWL - Technical Manual

Version 2.1

© 2010 The YAWL Foundation
Exercises, see OASE
"BPMS-instruction-4-YAWL.ppt/pdf"