The Workflow Patterns Initiative and its Application to Open Source WFMSs

Petia Wohed

Dept. of Computer and Systems Sciences
Stockholm University &
The Royal Institute of Technology
Acknowledgement

• This presentation uses slides prepared by the following people:
  – Wil van der Aalst, TUE & QUT
  – Michael Adams, QUT
  – Lachlan Aldred, QUT
  – Arthur ter Hofstede, QUT
  – Nick Russell, QUT
  – Petia Wohed, SU/KTH
Who am I?

- Jun 2000 – PhD at SU
- Sep-Nov 2001 – QUT, Australia
- Jul-Dec 2002 – QUT, Australia
- Sep’04-Aug’05 – UHP, France
- Sep’05-Feb’06 – QUT, Australia
- May’07 – QUT, Australia

Detailed analyses of
- BML
- BPEL4WS (XLANG, WSFL)
- BPML
- UML 2.0 AD
- BPMN

A/Prof. Arthur ter Hofstede
Prof. Wil van der Aalst (TUE)
Outline

• Background – WFMS and PAIS
• Conceptual Foundation - the Workflow Patterns Initiative
  – Control-flow patterns
  – Data patterns
  – Resource patterns
  – Exception Handling patterns
• The YAWL language
• Evaluations of Open Source Systems
Terminology

• WF
  
  “automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for actions according to a set of procedural rules”

  WfMC

• PAIS
  
  “A software system that manages and executes operational processes involving people, applications, and/or resources on the bases of process models.”

  M. Dumas, W. van der Aalst, A. ter Hofstede
The PAIS life cycle

Setting the scene –
Workflow: What and Why?

• Support for coordination of humans and applications in performing business activities
• Explicit representation of control flow dependencies and resourcing strategies
• Benefits:
  – Improved efficiency (time, cost)
  – Compliance
  – Improved responsiveness
Motivation:
Problems in the field of Workflow/BPM

- Lack of commonly accepted conceptual foundations
- Lack of proper formal foundations (this despite the amount of buzz …)
- No lack of proposed standards …
- Tools are typically hard to use, expensive and not easily integrated
- Lack of support for processes that need to change on-the-fly
- Lack of proper support for exceptions
- Limited support for design time analysis (verification and validation)
- Resource perspective particularly underwhelming
- Insufficient support for inter-process communication
Lack of commonly accepted conceptual foundations

How do various workflow environments deal with this?

- Forbid
- Execute D once, ignore second triggering
- Execute D twice
- Execute D once or twice depending on execution …
Workflow Patterns Initiative

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

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

• Home Page: www.workflowpatterns.com

• Primary publication:

• Evaluations of commercial offerings, research prototypes, proposed standards for web service composition, etc
The Workflow Patterns Framework

These perspectives follow S. Jablonski and C. Bussler’s classification from:
The Workflow Patterns Framework

Control-flow P:s 20
- COSA
- FLOWer
- Eastman
- Meteor
- Mobile
- I-Flow
- Staffware
- InConcert
- Domino Workflow
- Visual Workflow
- Forte Conductor
- MQSeries/Workflow
- SAR R/3 Workflow
- Verve Workflow
- Changengine
- XPDL, BPEL4WS, BPML, WSFL, XLANG, WSCI, UML AD 1.4, UML AD 2.0, BPMN

Resource P:s - 43
- Staffware
- WebSphere MQ
- FLOWer
- COSA
- iPlanet
- BPEL4WS
- UML AD 2.0
- BPMN

Data P:s - 40
- Staffware
- MQSeries
- FLOWer
- COSA
- iPlanet

Language Development: YAWL/newYAWL

© YAWL Foundation
Impact of the Workflow Patterns

Systems inspired or directly influenced by the patterns
- FLOWer 3.0 of Pallas Athena
- Bizagi of Vision Software
- Staffware Process Suite
- Pectra Technology Inc.’s tool
- Life/A&H Claim System by InsuraPro
- Ivolutia Orchestration
- OpenWFE (an open source WFMS)
- Zebra (an open source WFMS)
- Alphaflow (an open source WFMS)
- jBPM (a free workflow engine)

Use of the workflow patterns in selecting a WFMS
- the Dutch Employee Insurance Administration Office
- the Dutch Justice Department

Other
- Pattern-based evaluations (e.g. ULTRAflow, OmniFlow, @enterprise, BPMN)
- Citations (160+ academic papers)
- Education (used in teaching at 10+ Universities)
- Web site: 250,000+ views
The New Control-flow Patterns

- **Basic Control-flow Patterns**
capture elementary aspects of control-flow (similar to the concepts provided by the WFMC).

- **Advanced Branching and Synchronization Patterns**
describe more complex branching and synchronization scenarios.

- **Iteration Patterns**
describe various ways in which iteration may be specified.

- **Termination Patterns**
address the issue of when the execution of a workflow is considered to be finished.

- **Multiple Instances (MI) Patterns**
delineate situations where there are multiple threads of execution in a workflow which relate to the same activity.

- **State-based Patterns**
reflect situations which are most easily modelled in workflow languages that support the notion of state.

- **Cancellation Patterns**
categorise the various cancellation scenarios that may be relevant for a workflow specification.

- **Trigger Patterns**
catalogue the different triggering mechanisms appearing in a process context.
Sequence

Description: An activity in a workflow process is enabled after the completion of a preceding activity in the same process.

Example: A receipt is printed after the train ticket is issued.

Animation:

Definition CPN:

![CPN Diagram]

*The original animation for this pattern was done by Wil van der Aalst and Vincent Almering in 2003.
Structured Synchronising Merge

**Description:** The convergence of two or more branches (which diverged earlier in the flow) into a single subsequent branch. The tread of control is passed to the subsequent branch when each active incoming branch has been enabled.

**Example:** Depending on the emergency, either or both of the *dispatch-police* and *dispatch-ambulance* activities are initiated. When all the emergency vehicles arrive at the accident, the *transfer-patient* activity commences.

**Context conditions:**

1. There must be a *single Multi-Choice* construct earlier in the process model with which the *Synchronizing Merge* is associated and it must merge all of the branches emanating from the *Multi-Choice*. These branches must either flow from the *Multi-Choice* to the *Synchronizing Merge* without any splits or joins or they must be *structured* in form (i.e. balanced splits and joins)

2. ...
**Animation**

**WCP7: Structured Synchronising Merge**

*The original animation for this pattern was done by Wil van der Aalst and Vincent Almering in 2003*
Animation

WCP36: Acyclic Synchronizing Merge

Multiple Choice

A

B

C

Deferred Choice

D

E

Sync. Merge
Animation

WCP37: General Synchronizing Merge

Multiple Choice → Merge → Deferred Choice → Sync. Merge

A → B → C → D → E
The Data Patterns

Data Visibility Patterns
- characterise the various ways in which data elements can be defined and utilised
- task data, block data, scope data, MI data, etc.

Data Transfer Patterns
- focus on the way in which data elements are actually transferred between one process element and another
- Data transfer
  - by value,
  - by reference, etc.

Data Interaction Patterns
- deal with the various ways in which data elements can be passed between components within a process instance and also with the operating environment
- data elements flowing
  - between task instances
  - to and from a block
  - to and from MI task, etc.

Data-based routing Patterns
- capture the various ways in which data elements can interact with other perspective and influence the overall execution of the process
- Task pre/postcondition
  - data existence
  - data value, etc.
Description: The ability to communicate data elements between one task instance and another within the same case.

- The communication of data elements is specified in a form that is independent of the task definitions themselves.

Example: The *Determine Fuel Consumption* task required the coordinates determined by the *Identify Shortest Route* task before it can proceed.
Sample Data Pattern: Data Interaction between Tasks

- **Integrated Control and Data Channels**
  - Task A uses $(X, Y)$
  - Task B uses $(X)$
  - Task C uses $(Y)$

- **Distinct Control and Data Channels**
  - Task A uses $(X, Y)$
  - Task B uses $(X)$
  - Task C uses $(Y)$

- **No Data Passing**
  - Global Shared Data
    - Define var $X$
    - Define var $Y$
  - Task A uses $(X, Y)$
  - Task B uses $(X)$
  - Task C uses $(Y)$
Workflow Resource Patterns

• Focus on the manner in which work is offered to, allocated to and managed by workflow participants
• Consider both the system and resource perspectives
• Assume the existence of a process model and related organisational model
• Take into account differing workflow paradigms:
  – richness of process model (esp. allocation directives)
  – autonomy of resources
  – alternate routing mechanisms
  – work management facilities
The Resource Patterns

- **Creation patterns**
  design-time work allocation directives

- **Push Patterns**
  workflow system proactively distributes work items.

- **Pull Patterns**
  resources proactively identify and commit to work items.

- **Detour Patterns**
  re-routing of work items.

- **Auto-start Patterns**
  automated commencement

- **Visibility Patterns**
  observability of workflow activities

- **Multiple Resource Patterns**
  work allocation involving multiple participants or resources.
Work Item Lifecycle

- **created**
- **offered to a single resource**
- **allocated to a single resource**
- **started**
- **suspended**
- **failed**
- **completed**

**Resource Patterns**

- S:create
- S:offer_s
- S:allocate
- S:allocate_m
- S:offer_m
- R:allocate_s
- R:allocate_m
- R:start
- R:start_s
- R:start_m
- R:suspend
- R:resume
- R:complete
- R:fail
Detour Patterns

Refer to situations where work allocations that have been made for resources are interrupted either by the WF system or at the instigation of the resource.

Delegation, Escalation, Deallocation, ...
Delegation

Description:

The ability for a resource to allocate a work item previously allocated to it to another resource.

Example:

Before going on leave, the Chief Accountant passed all of their outstanding work items onto the Assistant Accountant.
Animation:

WRP27: Delegation

Task:
- Review
- Audit

Allocation:
- Alan

Max
- Work List
  - Executing
  - Allocated
  - Offered

Nina
- Work List
  - Executing
  - Allocated
  - Offered

Alan
- Work List
  - Executing
  - Allocated
  - Offered

2005 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede
Exception Handling

- Absence of a common framework for describing exception handling strategies in workflow systems.
- Contribution:
  - First comprehensive survey of exception handling capabilities of workflow systems
  - First generic, graphical exception language for workflows.
- Exception is a deviation from normal execution arising during a business process
- Two types:
  - Expected ➔ transactional workflow ➔ exception handling
  - Unexpected ➔ adaptive/evolutionary workflow
Exception Types

- Work Item Failure
- Deadline Expiry
- Resource Unavailability
- External Trigger
- Constraint Violation
Exception handling strategies centre on
1. How the work item will be handled;
2. How the other work items in the case will be handled; and
3. What recovery action will be taken to resolve the effects of the exception.
1. Exception Handling at Work Item Level
2. Exception Handling at Case Level

- Continue workflow case - CWC
- Remove current case - RCC
- Remove all cases - RAC
3. Recovery Action

- No action - NIL
- Rollback - RBK
- Compensate - COM
Classifying Exception Handling Strategies

Exception patterns take the form of tuples comprising:

- How the task on which the exception is based should be handled;
- How the case and other related cases in the process model in which the exception is raised should be handled; and
- What recovery action (if any) is to be undertaken.

**Example:** SFF-CWC-COM

i.e., Force fail - Continue workflow case - Compensate
## Exception Handling Taxonomy

<table>
<thead>
<tr>
<th>Work Item Failure</th>
<th>Work Item Deadline</th>
<th>Resource Unavailable</th>
<th>External Trigger</th>
<th>Constraint Violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF-CWC-NIL</td>
<td>OCO-CWC-NIL</td>
<td>ORO-CWC-NIL</td>
<td>OCO-CWC-NIL</td>
<td>SCE-CWC-NIL</td>
</tr>
<tr>
<td>OFF-CWC-NIL</td>
<td>ORO-CWC-NIL</td>
<td>OFF-CWC-NIL</td>
<td>OFF-CWC-NIL</td>
<td>SRS-CWC-NIL</td>
</tr>
<tr>
<td>OFC-CWC-NIL</td>
<td>OFF-CWC-NIL</td>
<td>OFF-RCC-NIL</td>
<td>OFF-ROC-NIL</td>
<td>SRS-CWC-COM</td>
</tr>
<tr>
<td>OFC-CWC-NIL</td>
<td>OFF-RCC-NIL</td>
<td>OFC-CWC-NIL</td>
<td>OFC-CWC-NIL</td>
<td>SRS-CWC-RBK</td>
</tr>
<tr>
<td>AFF-CWC-NIL</td>
<td>ARO-CWC-NIL</td>
<td>ARO-CWC-NIL</td>
<td>ACA-CWC-NIL</td>
<td>SFF-CWC-NIL</td>
</tr>
<tr>
<td>AFF-CWC-NIL</td>
<td>ARA-CWC-NIL</td>
<td>AFF-CWC-NIL</td>
<td>AFF-CWC-NIL</td>
<td>SFF-CWC-COM</td>
</tr>
<tr>
<td>AFC-CWC-NIL</td>
<td>ARA-CWC-NIL</td>
<td>AFF-RCC-NIL</td>
<td>AFL-CWC-NIL</td>
<td>SFF-CWC-RBK</td>
</tr>
<tr>
<td>AFC-CWC-NIL</td>
<td>ARA-CWC-NIL</td>
<td>AFF-RCC-NIL</td>
<td>AFF-RCC-NIL</td>
<td>SFF-CWC-RBK</td>
</tr>
<tr>
<td>SRSC-WC-NIL</td>
<td>AFF-CWC-NIL</td>
<td>SRA-CWC-NIL</td>
<td>AFL-CWC-NIL</td>
<td>SFF-CWC-RBK</td>
</tr>
<tr>
<td>SRS-CWC-COM</td>
<td>AFF-RCC-NIL</td>
<td>SRA-CWC-NIL</td>
<td>AFL-CWC-NIL</td>
<td>SFF-CWC-RBK</td>
</tr>
<tr>
<td>SRS-CWC-RBK</td>
<td>AFC-CWC-NIL</td>
<td>SRA-CWC-COM</td>
<td>AFL-CWC-NIL</td>
<td>SFF-CWC-RBK</td>
</tr>
<tr>
<td>SFF-CWC-NIL</td>
<td>SRA-CWC-NIL</td>
<td>SRO-CWC-NIL</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-NIL</td>
</tr>
<tr>
<td>SFF-CWC-COM</td>
<td>SRA-CWC-NIL</td>
<td>SRO-CWC-RBK</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-COM</td>
</tr>
<tr>
<td>SFF-CWC-RBK</td>
<td>SRA-CWC-RBK</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-COM</td>
</tr>
<tr>
<td>SFF-CWC-NIL</td>
<td>SRO-CWC-COM</td>
<td>SFF-CWC-RBK</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-COM</td>
</tr>
<tr>
<td>SFF-CWC-COM</td>
<td>SRO-CWC-RBK</td>
<td>SFF-CWC-RBK</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-COM</td>
</tr>
<tr>
<td>SFF-CWC-RBK</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-COM</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-COM</td>
</tr>
<tr>
<td>SFF-CWC-RBK</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-RBK</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-COM</td>
</tr>
<tr>
<td>SFF-CWC-RBK</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-NIL</td>
<td>SFF-CWC-COM</td>
</tr>
</tbody>
</table>
Exception Handling Primitives

1. Remove current work item
2. Suspend current work item
3. Continue current work item or thread
4. Restart current work item
5. Remove selected/all work items in current case
6. Suspend selected/all work items in current case
7. Continue selected/all work items in current case
8. Force complete current work item
9. Remove selected/all work items in all cases
10. Suspend selected/all work items in all cases
11. Continue selected/all work items in all cases
12. Force fail current work item
13. Reallocate current work item
14. Reoffer current work item
15. Compensation task
16. Rollback task
A Generic Architecture for Exception Handling

work item failure

deadline expiry
<deadline>

constraint violation
<constraint>

exception handling definition

process definition

© YAWL Foundation
Some Observations

- Virtually **all** patterns have been observed in at least one system/language.
- Overall pattern support is generally **limited**, especially the resource perspective and exception handling.
- Provide detailed insight into relative strengths and weaknesses of various approaches.
- More research needed into relation between patterns and their suitability for specific types of applications as well as their realisation in terms of language constructs.
Outline

• Background – WFMS and PAIS
• Conceptual Foundation - the Workflow Patterns Initiative
  – Control-flow patterns
  – Data patterns
  – Resource patterns
  – Exception Handling patterns

• The YAWL language
• Evaluations of Open Source Systems
YAWL Overview

- Collaboration between TU/e and QUT
- Based on Workflow Patterns Initiative
- System development
  - Open source (currently LGPL)
  - Industry collaboration
    - M2 Investments – first:telecom
    - Intercontinental Hotels Group
- Main publication
- URLs:
  - www.yawlfoundation.org (research)
  - www.sourceforge.net/projects/yawl (system)
  - http://www.yawlgroup.com (consultancy)
YAWL vs Petri nets

• Petri nets have difficulties capturing:
  1. The General Synchronising Merge
  2. Patterns involving Multiple Instances
  3. Cancellation of a certain part of a process

• For the Control Flow Perspective, YAWL takes some concepts form Petri nets and adds constructs for:
  – OR-join to deal with General Synchronising Merge
  – Multiple Instance (MI) tasks
  – Cancellation regions
  – “Syntactic Sugar” (various splits/joins, direct connections between tasks)
YAWL notation

- Condition
- Start condition
- End condition
- XOR-split task
- OR-split task
- AND-split task
- XOR-join task
- OR-join task
- AND-join task

Composite task

Multiple Instance task

© YAWL Foundation
YAWL Example I

- Register
- Flight
- Hotel
- Car
- Pay
YAWL Example II

- register
- flight
- hotel
- car
- pay
The OR-join in YAWL

- The OR-join as originally proposed in YAWL has been generalised by Moe Wynn et al [WEAH05].
- The formalisation exploits algorithms for coverability analysis in reset nets.
- An OR-join is enabled iff one of its input places, say p, is marked, and it is not possible to reach a marking from the current marking that also marks p and a previously unmarked input place of the OR-join where:
  - Multiple instance tasks and composite tasks are treated as atomic tasks.
  - Other OR-joins (in the same decomposition) are treated as XOR-joins.
YAWL Example III

register

do_itinerary_segment

pay

register_itinerary_segment

flight

hotel

prepare_payment_information

car
Cancellation in YAWL

- The concept of cancellation region generalises 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.
General YAWL Example V

- register
- do_itinerary_segment
- pay
- booking_in_progress
- cancel
Service Oriented Architecture

YAWL

Web service

Worklist data interface
Exception interface
Workflow-app interface
Admin Interface
Process Interface

Organisational Structure
Workflow Logger
Workflow Console
Case Data
Process Repository

Adminstration Console
Process Model Editor

Worklist Service
SMS Messaging Service
Timeout Service
Web service Invoker Service

Worklist GUI Service

Custom Framework
Custom Service
Worklist Service
Outline

- Background – WFMS and PAIS
- Conceptual Foundation - the Workflow Patterns Initiative
  - Control-flow patterns
  - Data patterns
  - Resource patterns
  - Exception Handling patterns
- The YAWL language
- Evaluations of Open Source Systems
### Results – Open Source Tools: Control-flow perspective

<table>
<thead>
<tr>
<th>Advanced Synchronisation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Control-flow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Sequence</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>2 Parallel Split</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>3 Synchronisation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>4 Exclusive Choice</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>5 Simple Merge</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>6 Multiple Choice</td>
<td>-</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>7 Str. Synchronising Merge</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>8 Multiple Merge</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>9 Structured Discriminator</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>10 Arbitrary Cycles</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>11 Implicit Termination</td>
<td>+/-</td>
<td>+</td>
<td>+/-</td>
<td>-</td>
</tr>
<tr>
<td>12 MI without Synchronisation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>13 MI with a priori Design Time Knlg</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>14 MI with a priori Runtime Knlg</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>15 MI without a priori Runtime Knlg</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>27 Complete MI Activity</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>34 Static Partial Join for MI</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>35 Static Canc. Partial Join for MI</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>36 Dynamic Partial Join for MI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>28 Blocking Discriminator</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>29 Cancelling Discriminator</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>30 Structured Partial Join</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>31 Blocking Partial Join</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>32 Cancelling Partial Join</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>33 Generalised AND-Join</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>34 Static Partial Join for MI</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>35 Static Canc. Partial Join for MI</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>36 Dynamic Partial Join for MI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>37 Local Sync. AND-Join</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>38 General Sync. Merge</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>39 Critical Section</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>40 Interleaved Routing</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>41 Thread Merge</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>42 Thread Split</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Iteration</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 Arbitrary Cycles</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>21 Structured Loop</td>
<td>+/-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>11 Implicit Termination</td>
<td>+/-</td>
<td>+</td>
<td>+/-</td>
<td>-</td>
</tr>
<tr>
<td>12 MI without Synchronisation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>13 MI with a priori Design Time Knlg</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>14 MI with a priori Runtime Knlg</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>15 MI without a priori Runtime Knlg</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>27 Complete MI Activity</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>34 Static Partial Join for MI</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>35 Static Canc. Partial Join for MI</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>36 Dynamic Partial Join for MI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>37 Local Sync. AND-Join</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>38 General Sync. Merge</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>39 Critical Section</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>40 Interleaved Routing</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>41 Thread Merge</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>42 Thread Split</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

### State-based

- Deferred Choice
- Critical Section
- Interleaved Parallel Routing
- Interleaved Routing
- Milestone

### Cancellation

- Cancel Activity
- Cancel Case
- Cancel Region
- Cancel MI Activity

### Trigger

- Transient Trigger
- Persistent Trigger

---

© YAWL Foundation
## Results – Open Source Tools: Data perspective

<table>
<thead>
<tr>
<th>Data Visibility</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Data</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Block Data</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Scope Data</td>
<td>-</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>MI Data</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Case Data</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>MI Data</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Workflow Data</td>
<td>+/-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Environment Data</td>
<td>+/-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>between Tasks</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Task to Sub-workflow Decomp.</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Sub-workflow Decomp. to Task</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>to MI Task</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>from MI Task</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Case to Case</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

### Data Transfer

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Env. to Case - Push-Oriented</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Case to Env. - Pull-Oriented</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Workflow to Env. - Push-Oriented</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Env. to Workflow - Pull-Oriented</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Env. to Workflow - Push-Oriented</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Workflow to Env. - Pull-Oriented</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>by Value - Incoming</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>by Value - Outgoing</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Copy In/Copy Out</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>by Reference - Unlocked</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>by Reference - Locked</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Data Transformation - Input</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Data Transformation - Output</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Task Precondition Data Exist.</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Task Precondition Data Value</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Task Postcondition Data Exist.</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Task Postcondition Data Value</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Event-based Task Trigger</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Data-based Task Trigger</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Data-based Routing</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

The results for jBPM and OpenWFE are preliminary.
The evaluation of YAWL and new YAWL is done by Nick Russell.
## Results – Open Source Tools: Resource perspective

### Creation Patterns

<table>
<thead>
<tr>
<th>Creation Pattern</th>
<th>jBPM</th>
<th>Open WFE</th>
<th>YAWL</th>
<th>new YAWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Allocation</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Role-Based Allocation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Deferred Allocation</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Authorization</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Separation of Duties</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Case Handling</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Retain Familiar</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Capacity-based Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>History-based Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Organizational Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Automatic Execution</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

### Pull Patterns (cont.)

<table>
<thead>
<tr>
<th>Pull Pattern</th>
<th>jBPM</th>
<th>Open WFE</th>
<th>YAWL</th>
<th>new YAWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>System-Deterministic Work Queue Cont.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Resource-Deterministic Work Queue Cont.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Selection Autonomy</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Delegation</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Escalation</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Deallocation 1)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Stateful Reallocation</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Stateless Reallocation</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Suspension/Resumption</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Skip</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Commencement on Creation</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Commencement on Allocation</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Piled Execution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Chained Execution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Config. Unallocated WI Visibility</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Config. Allocated WI Visibility</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Simultaneous Execution</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Additional Resource</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

### Push Patterns

<table>
<thead>
<tr>
<th>Push Pattern</th>
<th>jBPM</th>
<th>Open WFE</th>
<th>YAWL</th>
<th>new YAWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution by Offer-Single Resource</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Distribution by Offer-Multiple Resources</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Distribution by Allocation-Single Resource</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Random Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Round Robin Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Shortest Queue</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Early Distribution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Distribution on Enablement</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Late Distribution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

### Auto-start Patterns

<table>
<thead>
<tr>
<th>Auto-start Pattern</th>
<th>jBPM</th>
<th>Open WFE</th>
<th>YAWL</th>
<th>new YAWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commencement on Creation</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Commencement on Allocation</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Piled Execution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Chained Execution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

### Pull Patterns (cont.)

<table>
<thead>
<tr>
<th>Pull Pattern</th>
<th>jBPM</th>
<th>Open WFE</th>
<th>YAWL</th>
<th>new YAWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config. Unallocated WI Visibility</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Config. Allocated WI Visibility</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Simultaneous Execution</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Additional Resource</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

### Multiple Resource Patterns

<table>
<thead>
<tr>
<th>Multiple Resource Pattern</th>
<th>jBPM</th>
<th>Open WFE</th>
<th>YAWL</th>
<th>new YAWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource-Init. Allocation</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Resource-Init. Exec. - Allocated WI</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Resource-Init. Exec. - Offered WI</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The results for jBPM and OpenWFE are preliminary.  
The evaluation of YAWL and new YAWL is done by Nick Russell.
Thanks!

Questions?