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

© YAWL Foundation

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

© YAWL Foundation

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

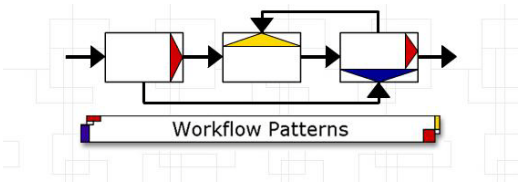
- BML
- BPEL4WS (XLANG, WSFL)
- BPML
- UML 2.0 AD
- BPMN

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

QUT TU/e

© YAWL Foundation

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 Foundation

Terminology

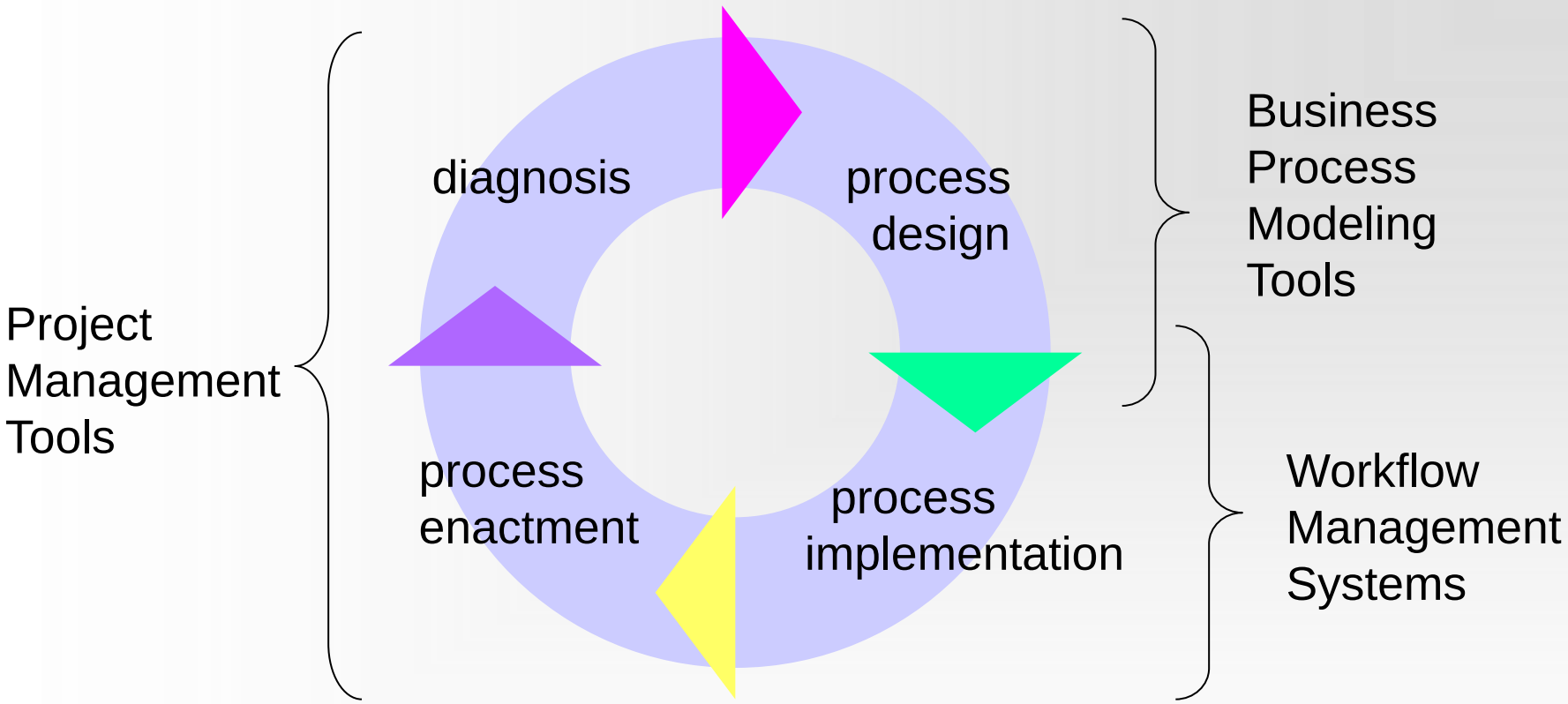
- WF
  - **"automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for actions according to a set of procedural rules"**

    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

© YAWL Foundation

The PAIS life cycle



M. Dumas, W. van der Aalst, A. ter Hofstede, *Process-Aware Information Systems: Bridging People and Software through Process Technology*, John Wiley & Sons, 2005

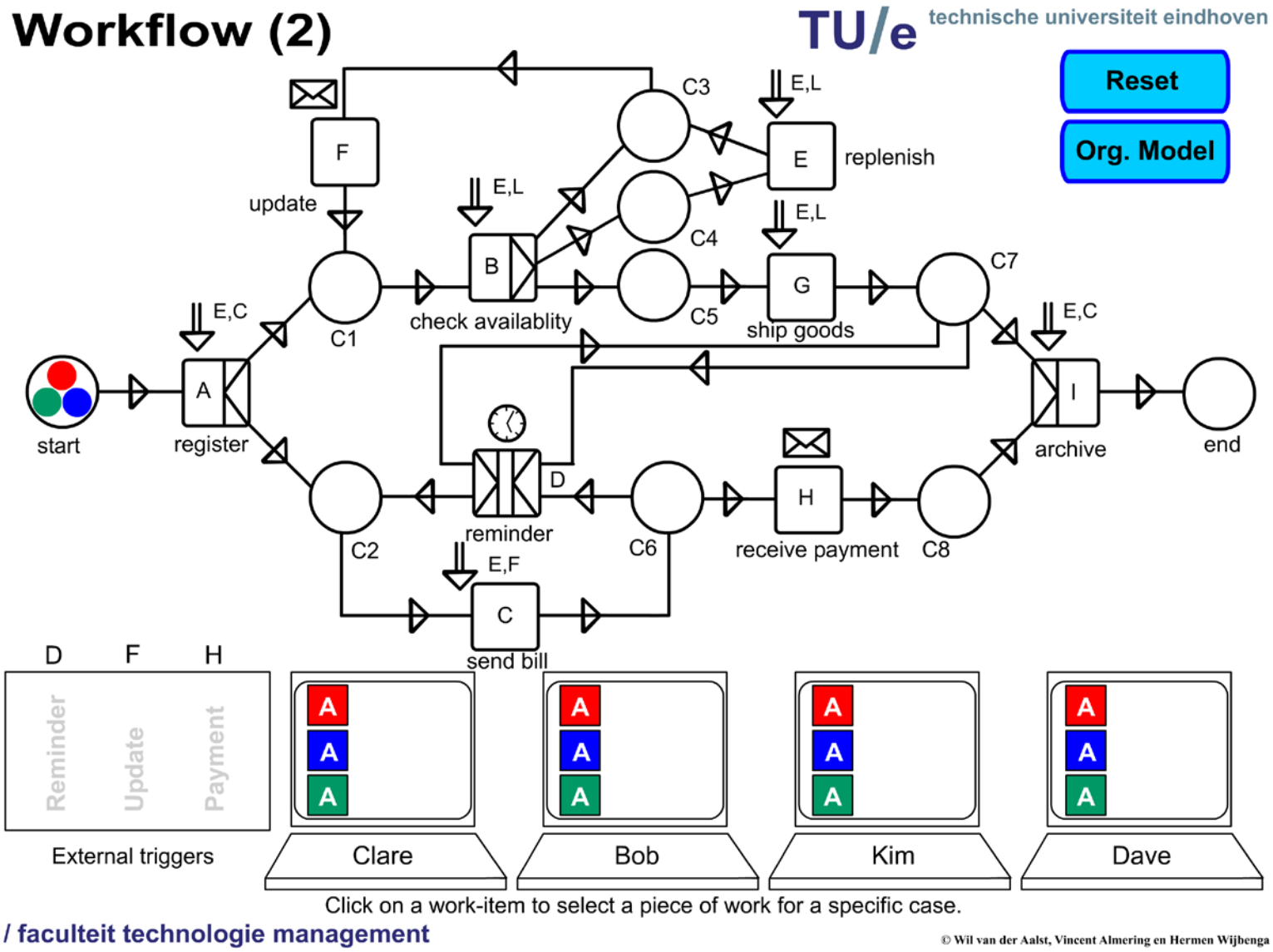
© YAWL Foundation

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

© YAWL Foundation

Workflow Animation



© Wil van der Aalst, Vincent Almering and Herman Wijbenga

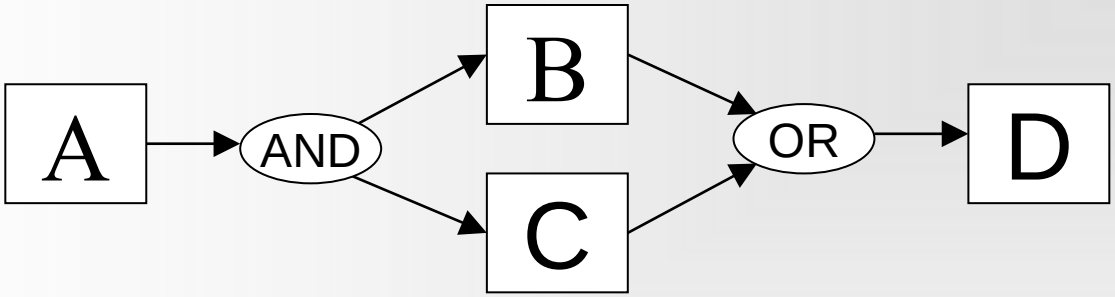
Motivation:
Problems in the field of Workflow/BPM

- 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

© YAWL Foundation

Lack of commonly accepted conceptual foundations

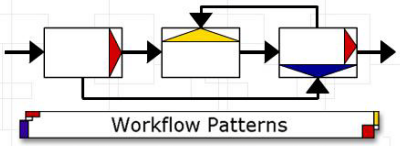
How do various workflow environments deal with this?



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

© YAWL Foundation

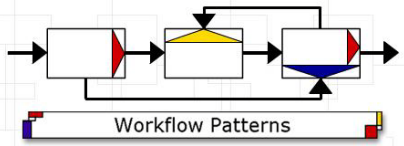
Workflow Patterns Initiative



- 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:
  - W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros, “Workflow Patterns”, *Distributed and Parallel Databases* 14(3):5-51, 2003.
- Evaluations of commercial offerings, research prototypes, proposed standards for web service composition, etc

© YAWL Foundation

The Workflow Patterns Framework



time

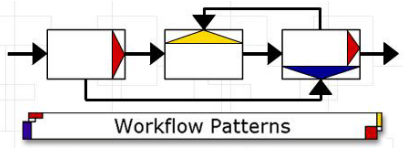
2000	2003	Jun 2005	Oct 2005	Sep 2006
Control-flow P:s 20		Resource P:s - 43	Data P:s - 40	revised Control-flow P:s 43
W. van der Aalst A. ter Hofstede B. Kiepuszewski A. Barros		N. Russell W. van der Aalst A. ter Hofstede D. Edmond	N. Russell A. ter Hofstede D. Edmond W. van der Aalst	N. Russell A. ter Hofstede W. van der Aalst N. Mulyar
The ordering of activities in a process		Resource definition & work distribution in a process	Data representation and handling in a process	- 23 new patterns - Formalised in CPN notation
CoopIS'2000 DAPD'2003		CAiSE'2005	ER'2005	TR

These perspectives follow S. Jablonski and C. Bussler's classification from:
Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press, 1996

www.workflowpatterns.com

© YAWL Foundation

The Workflow Patterns Framework



time

2000	2003	Jun 2005	Oct 2005
Control-flow P:s 20		Resource P:s - 43	Data P:s - 40

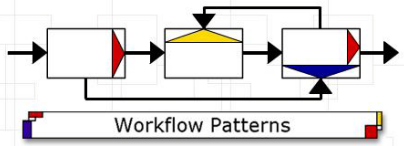
Evaluations

COSA FLOWer Eastman Meteor Mobile I-Flow Staffware InConcert	Domino Workflow Visual Workflow Forte Conductor MQSeries/Workflow SAR R/3 Workflow Verve Workflow Changengine	Staffware WebSphere MQ FLOWer COSA iPlanet	Staffware MQSeries FLOWer COSA
XPDL, BPEL4WS, BPML, WSFL, XLANG, WSCI, UML AD 1.4 UML AD 2.0, BPMN		BPEL4WS UML AD 2.0 BPMN	XPDL, BPEL4WS UML AD 2.0, BPMN

LanguageDevelopment: 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

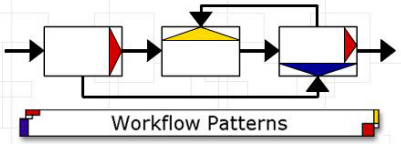
the Dutch Employee Insurance Administration Office
the Dutch Justice Department

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

© YAWL Foundation

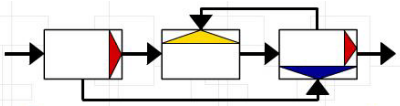
The New Control-flow Patterns



- **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.

© YAWL Foundation

Sequence

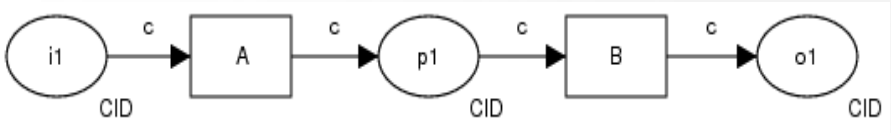


Basic Patterns: 1

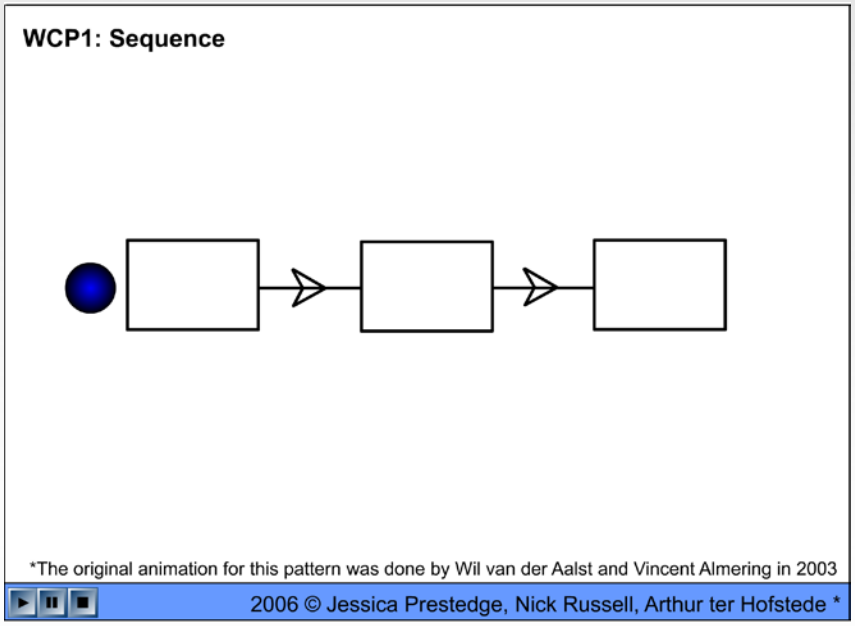
Description: An activity in a workflow process is enabled after the completion of a preceding activity in the same process.

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

Definition CPN:

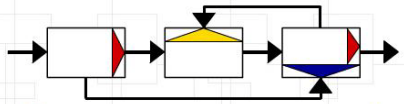


Animation:



© YAWL Foundation

Structured Synchronising Merge



Advanced: 7

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.

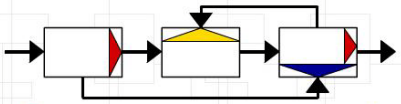
Example: Depending on the emergency, either or both of the *dispatch-police* and *dispatch-ambulance* activities are initiated. When all the emergency vehicles arrive at the accident, the *transfer-patient* activity commences.

Context conditions:

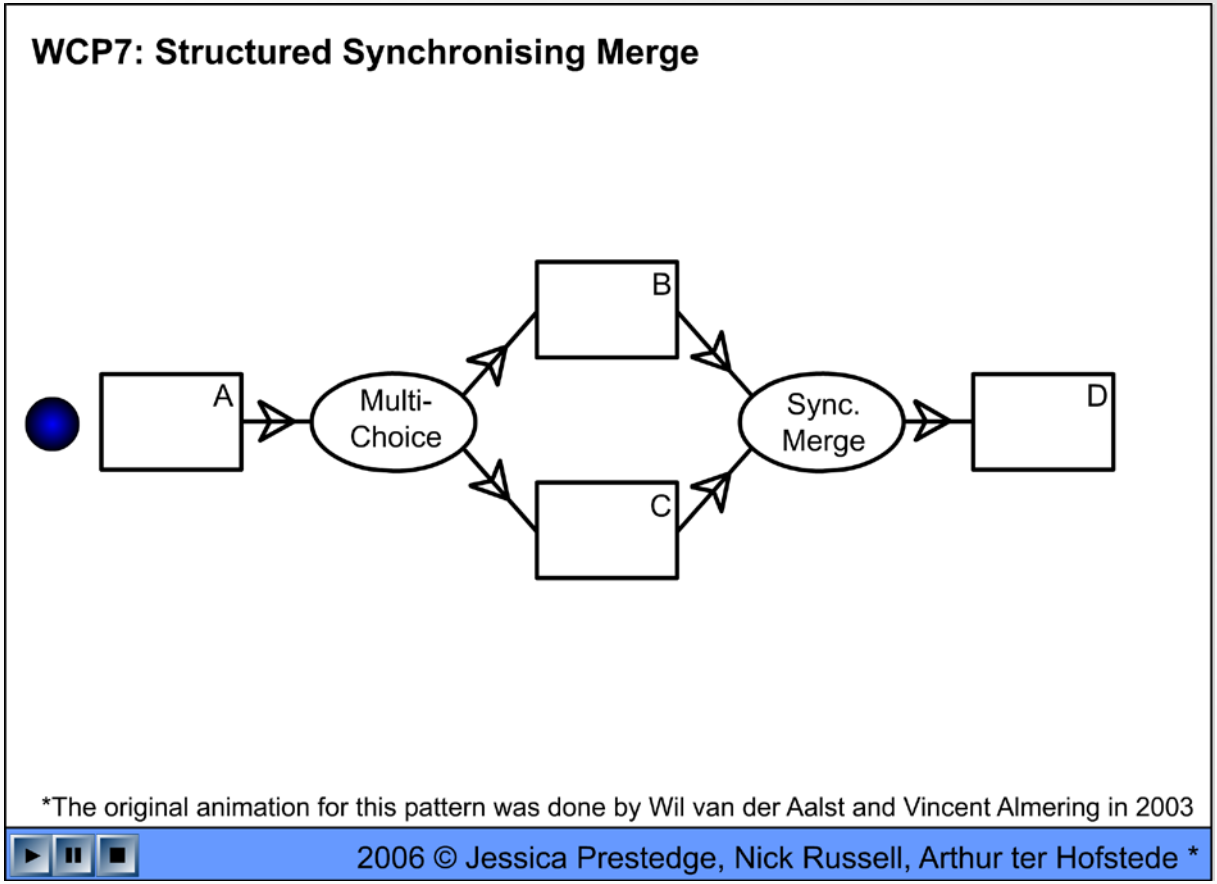
1. There must be a single *Multi-Choice* construct earlier in the process model with which the *Synchronizing Merge* is associated and it must merge all of the branches emanating from the *Multi-Choice*. These branches must either flow from the *Multi-Choice* to the *Synchronizing Merge* without any splits or joins or they must be structured in form (i.e. balanced splits and joins)
2. …

© YAWL Foundation

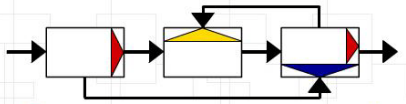
Animation

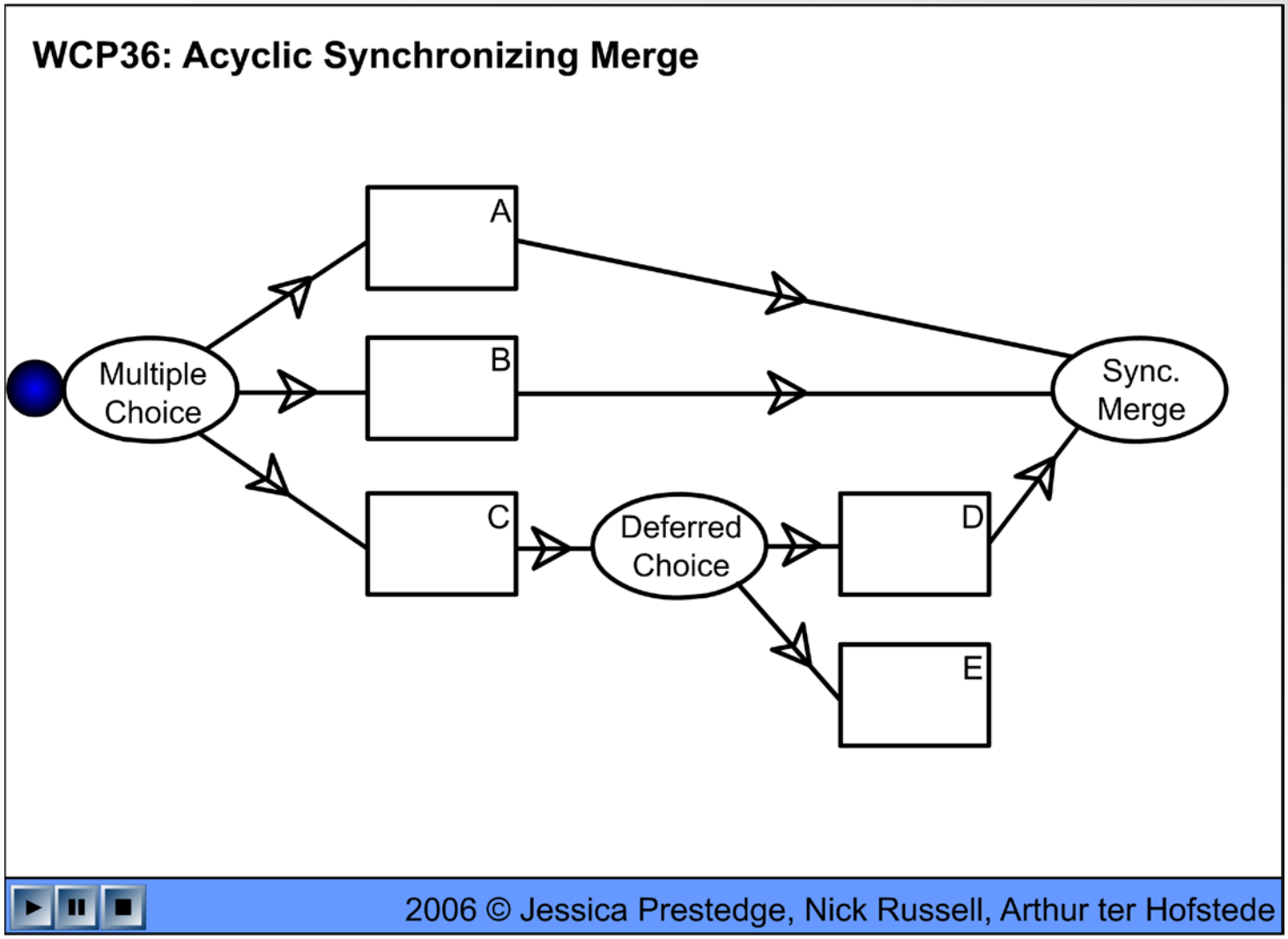


Advanced: 7

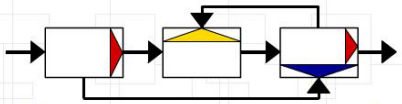


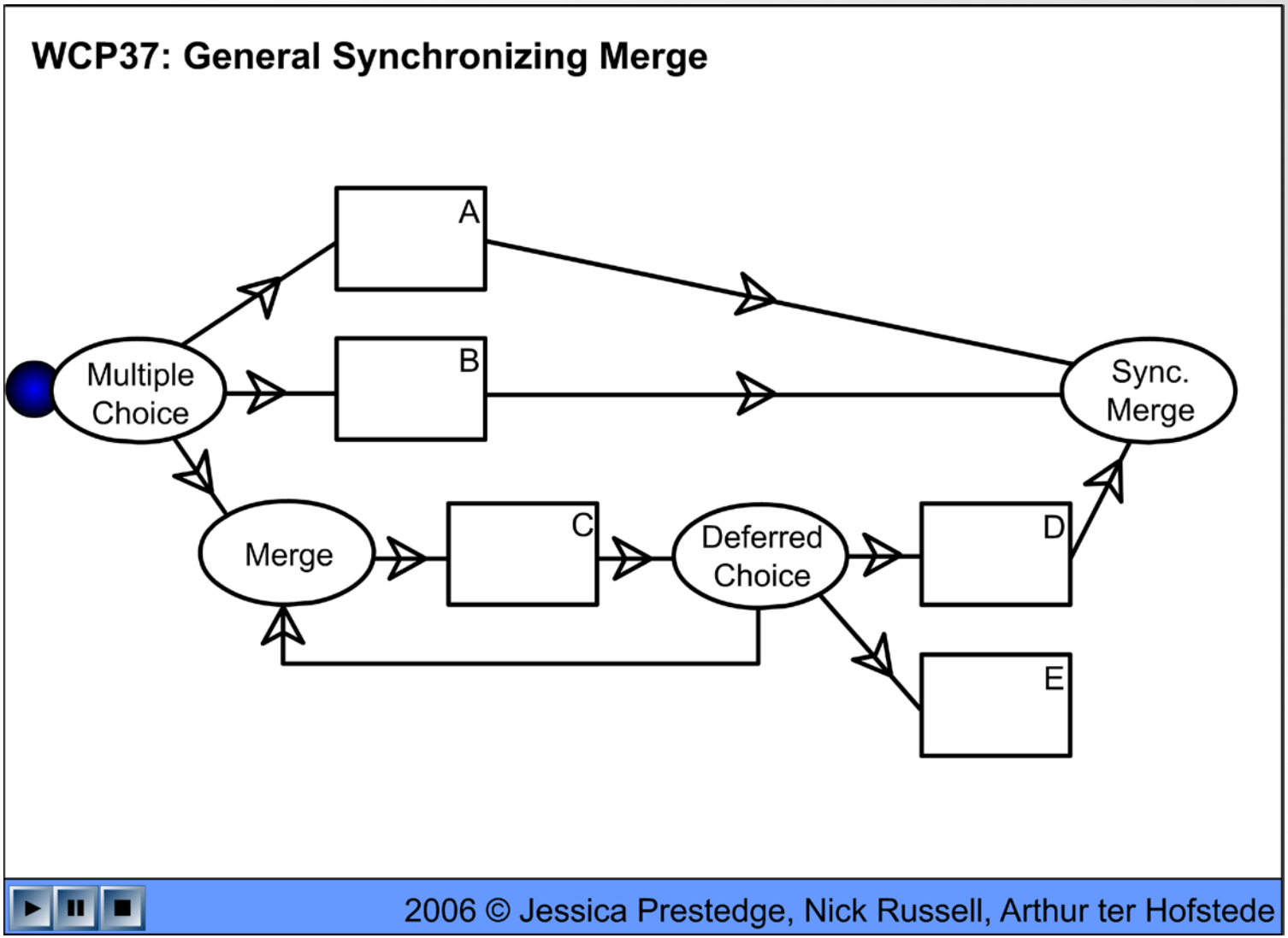
Animation



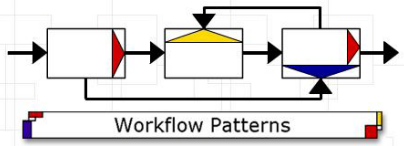

WCP36: Acyclic Synchronizing Merge
A
B
C
Multiple Choice
Sync. Merge
Deferred Choice
D
E
2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

Animation




WCP37: General Synchronizing Merge
A
B
Multiple Choice
Sync. Merge
Merge
C
Deferred Choice
D
E
2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

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 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 Transfer Patterns

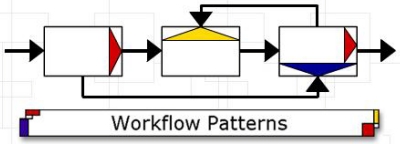
- focus on the way in which data elements are actually transferred between one process element and another
- Data transfer
  - by value,
  - by reference, etc.

Data-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/postcondtion
  - data existence
  - data value, etc.

© YAWL Foundation

Sample Data Pattern:
Data Interaction between Tasks



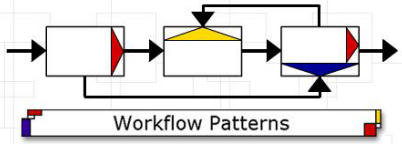
Description: The ability to communicate data elements between one task instance and another within the same case.

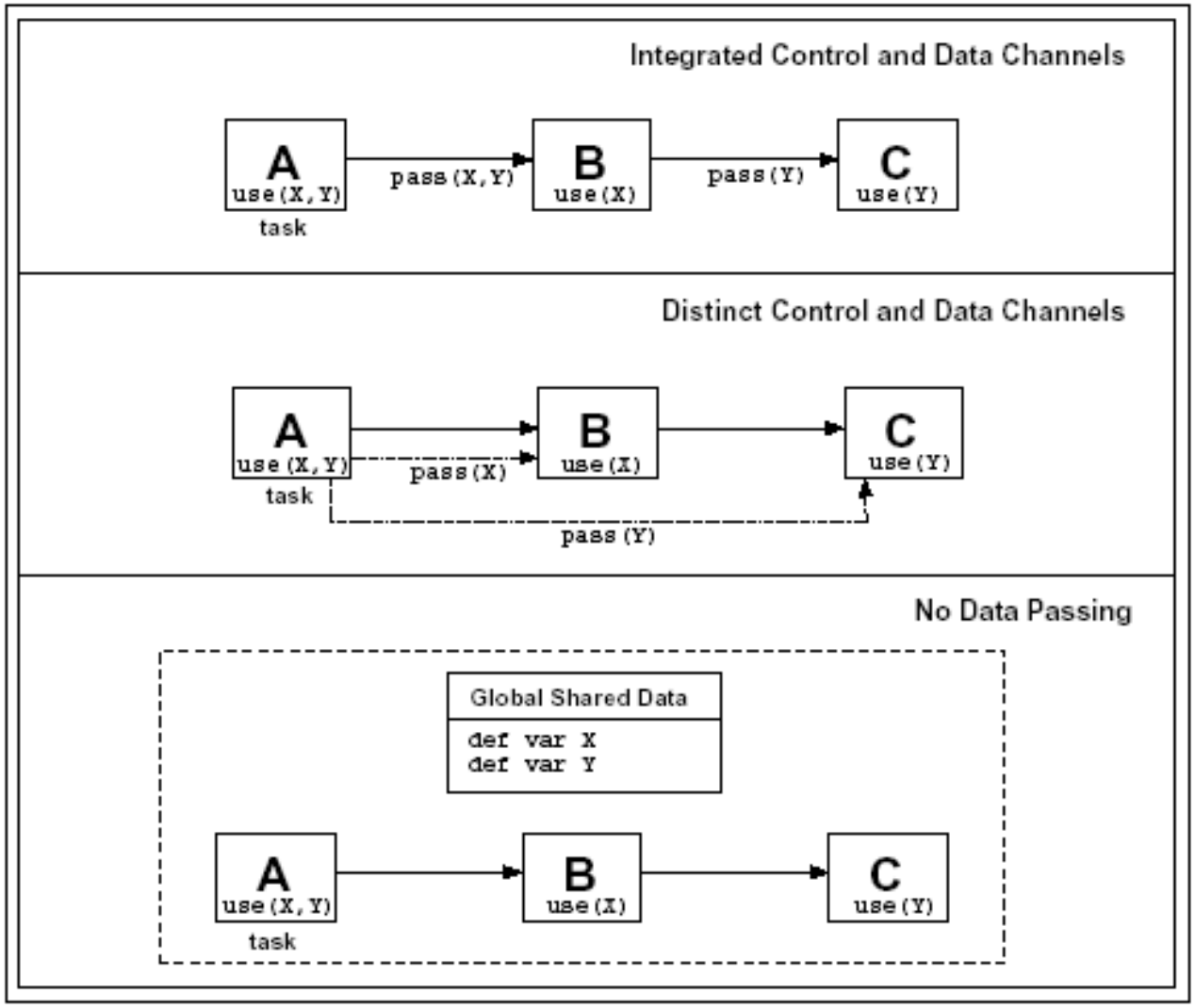
- The communication of data elements is specified in a form that is independent of the task definitions themselves.

Example: The *Determine Fuel Consumption* task required the coordinates determined by the *Identify Shortest Route* task before it can proceed.

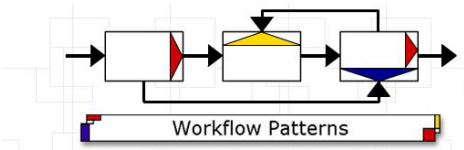
© YAWL Foundation

Sample Data Pattern:
Data Interaction between Tasks





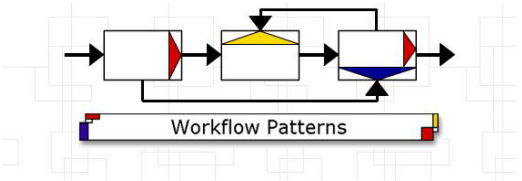
Workflow Resource Patterns



- 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

© YAWL Foundation

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

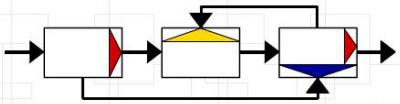
  .

- Multiple Resource Patterns

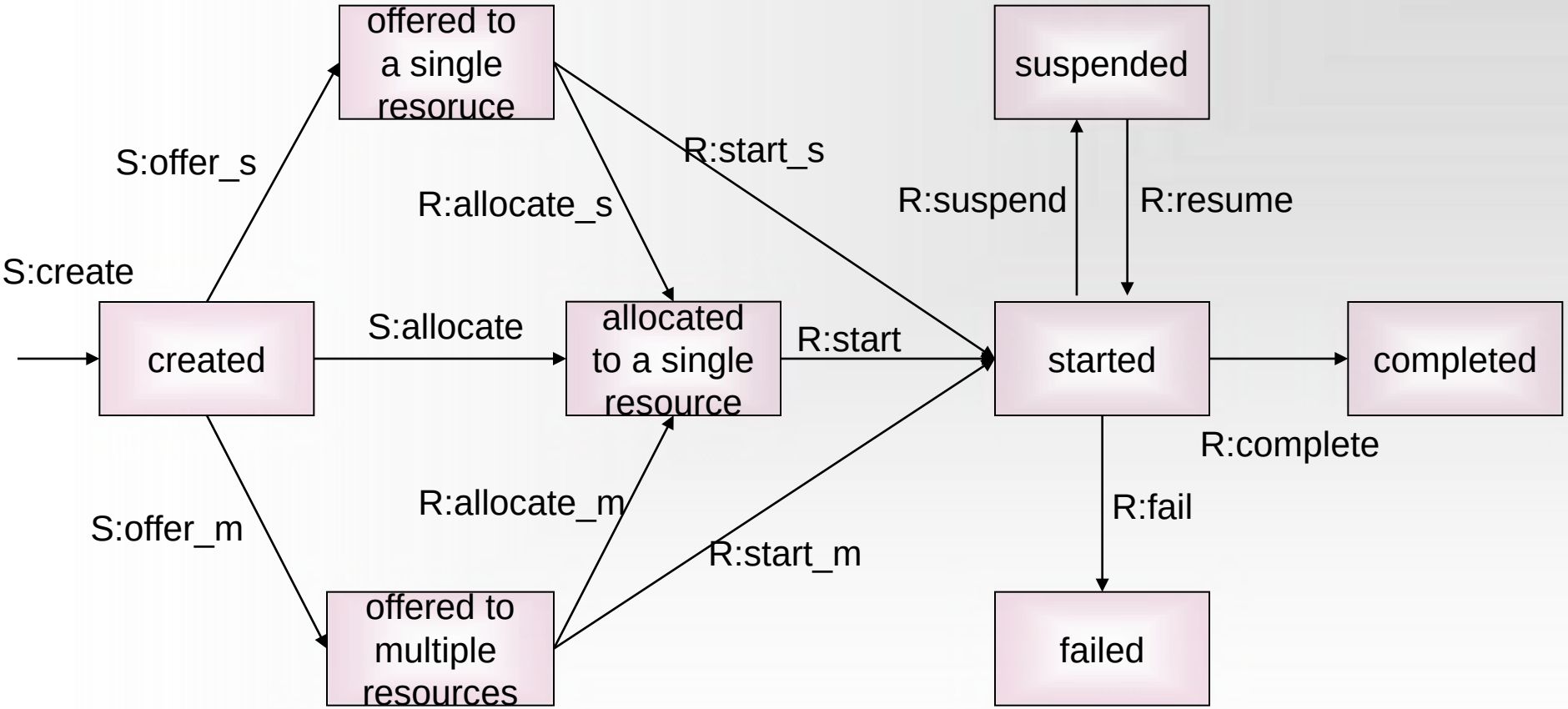
  work allocation involving multiple participants or resources.

© YAWL Foundation

Work Item Lifecycle



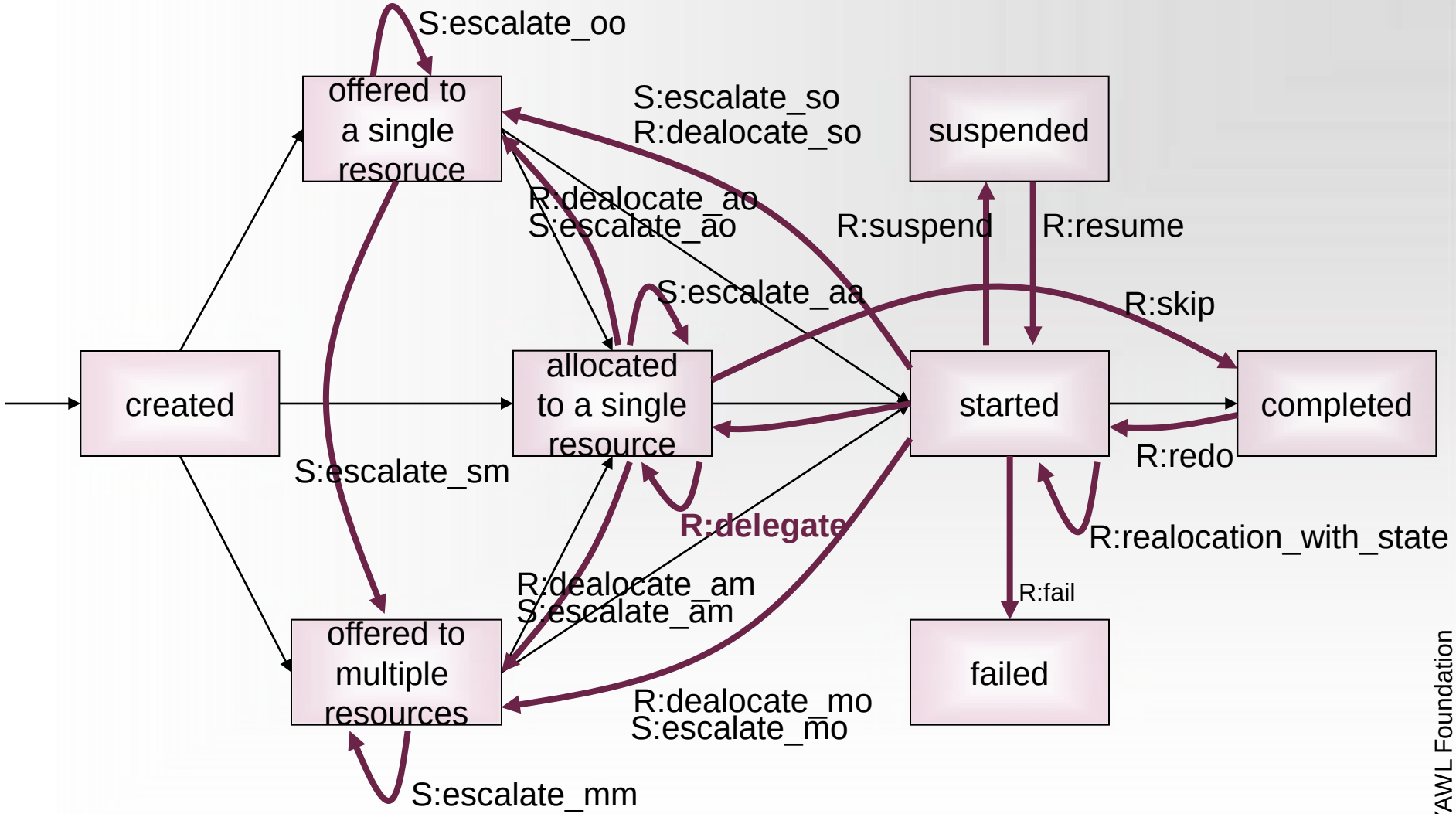
Resource Patterns



© YAWL Foundation

Detour Patterns

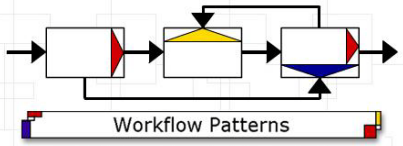
Refer to situations where work allocations that have been made for resources are interrupted either by the WF system or at the instigation of the resource.



Delegation, Escalation, Deallocation, ...

© YAWL Foundation

Delegation



Description:

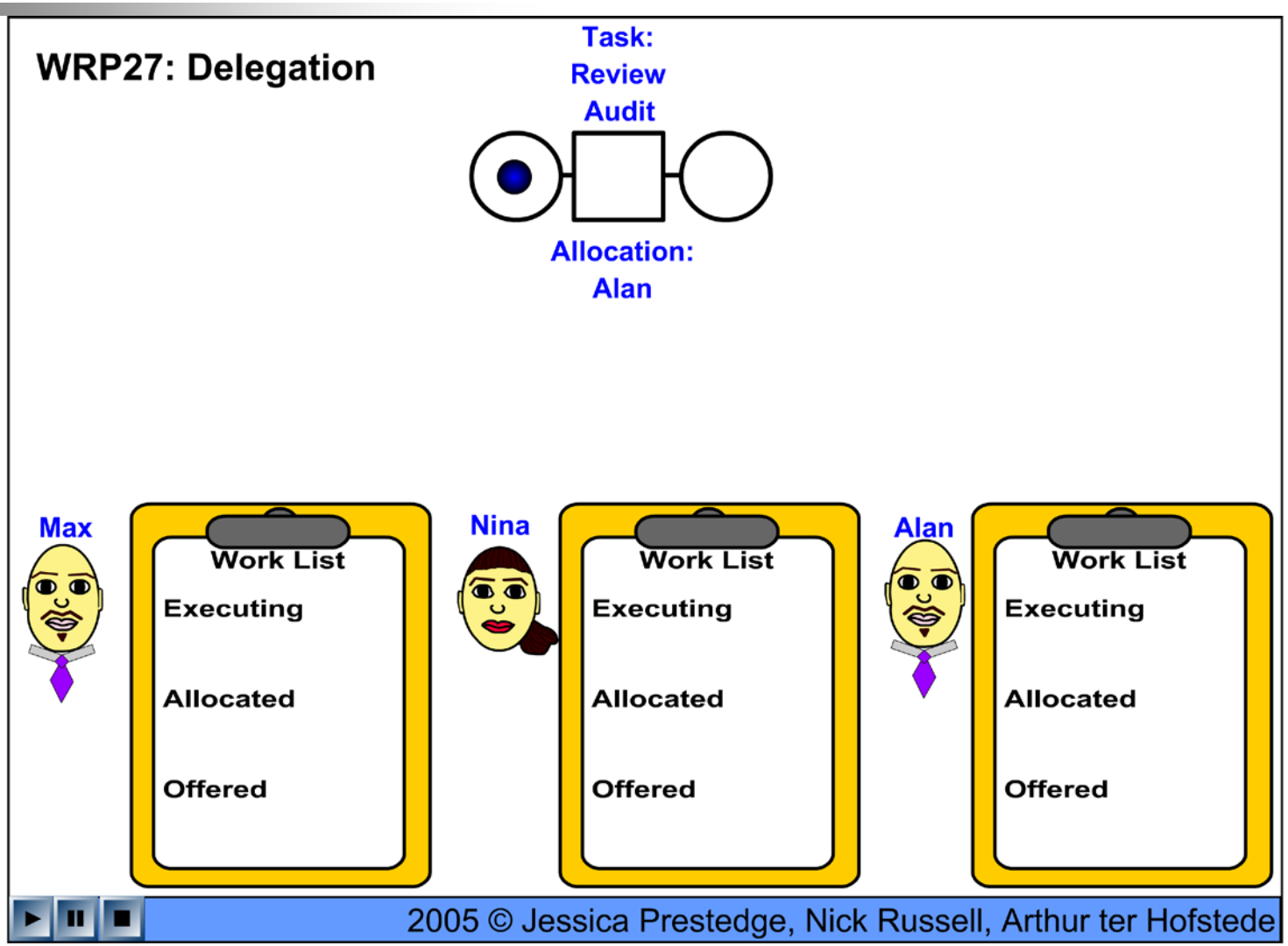
The ability for a resource to allocate a work item previously allocated to it to another resource.

Example:

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

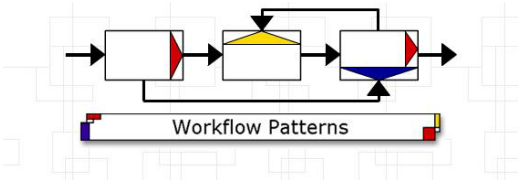
© YAWL Foundation

Animation:



© YAWL Foundation

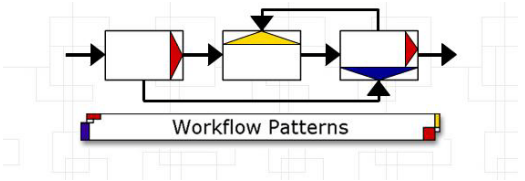
Exception Handling



- 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

© YAWL Foundation

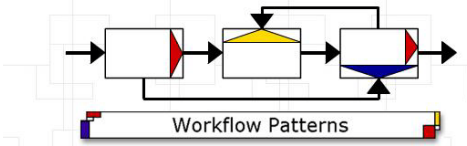
Exception Types



- Work Item Failure
- Deadline Expiry
- Resource Unavailability
- External Trigger
- Constraint Violation

© YAWL Foundation

Exception Handling Fundamentals

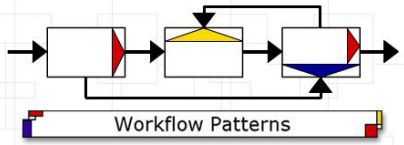


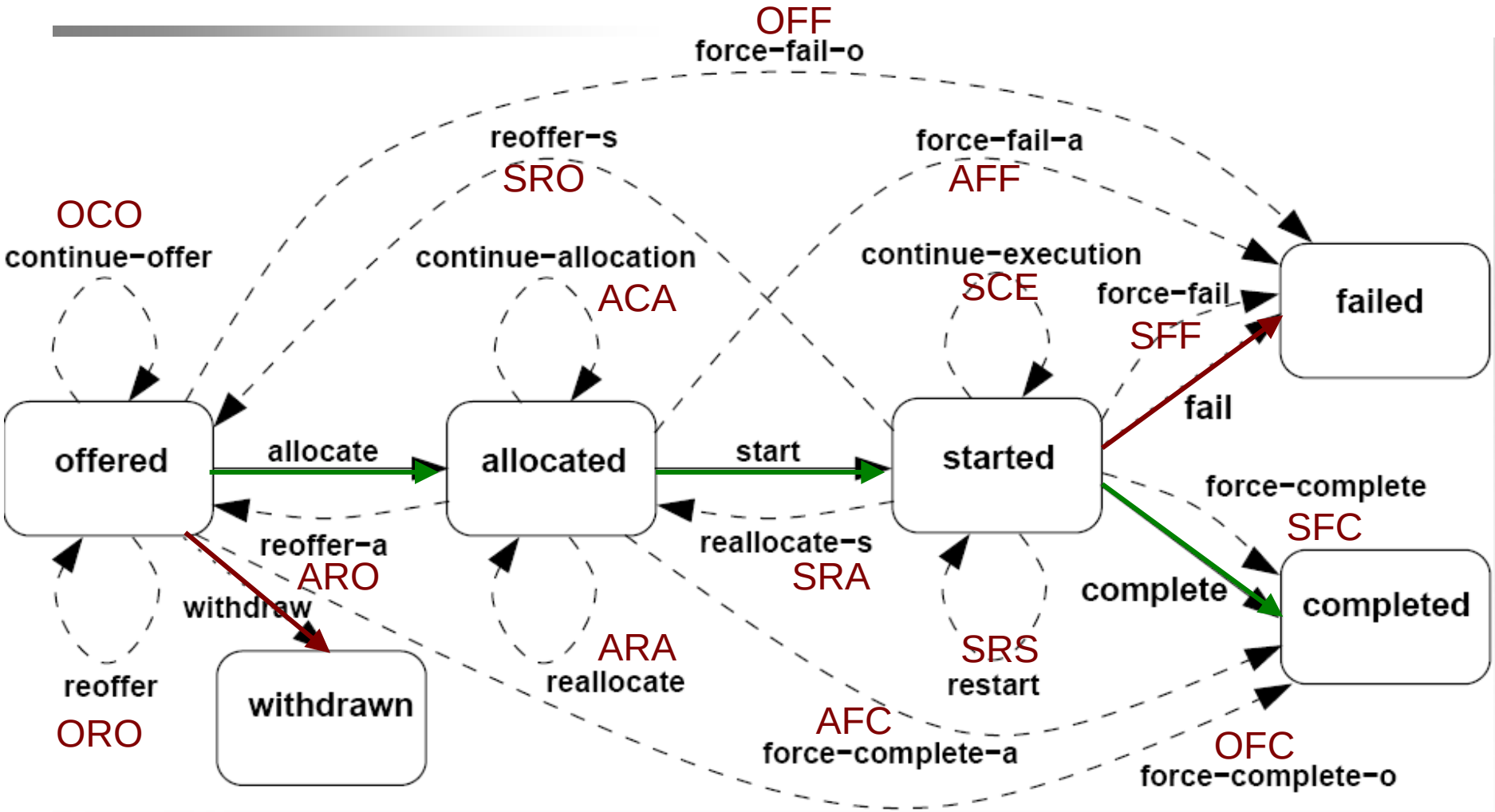
Exception handling strategies centre on

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.

© YAWL Foundation

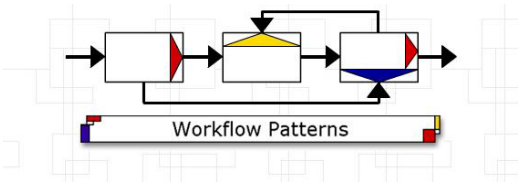
1. Exception Handling at Work Item Level





© YAWL Foundation

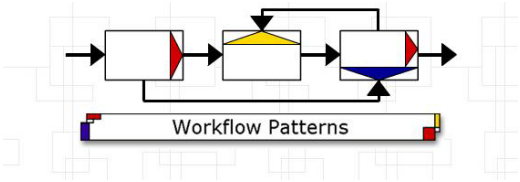
2. Exception Handling at Case Level



- Continue workflow case - CWC
- Remove current case - RCC
- Remove all cases - RAC

© YAWL Foundation

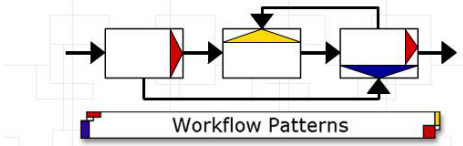
3. Recovery Action



- No action - NIL
- Rollback - RBK
- Compensate - COM

© YAWL Foundation

Classifying Exception Handling Strategies



Exception patterns take the form of tuples comprising:

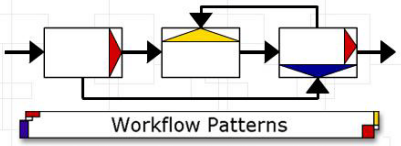
- How the task on which the exception is based should be handled;
- How the case and other related cases in the process model in which the exception is raised should be handled; and
- What recovery action (if any) is to be undertaken.

Example: SFF-CWC-COM

i.e., Force fail - Continue workflow case - Compensate

© YAWL Foundation

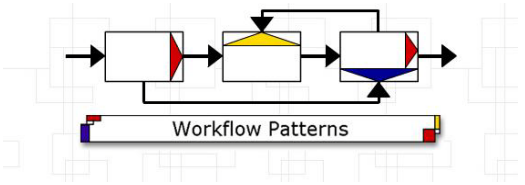
Exception Handling Taxonomy

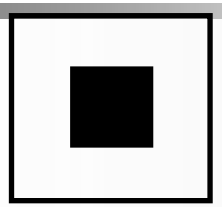


Work Item Failure	Work Item Deadline	Resource Unavailable	External Trigger	Constraint Violation
OFF-CWC-NIL	OCO-CWC-NIL	ORO-CWC-NIL	OCO-CWC-NIL	SCE-CWC-NIL
OFF-CWC-COM	ORO-CWC-NIL	OFF-CWC-NIL	OFF-CWC-NIL	SRS-CWC-NIL
OFC-CWC-NIL	OFF-CWC-NIL	OFF-RCC-NIL	OFF-RCC-NIL	SRS-CWC-COM
OFC-CWC-COM	OFF-RCC-NIL	OFC-CWC-NIL	OFC-CWC-NIL	SRS-CWC-RBK
AFF-CWC-NIL	OFC-CWC-NIL	ARO-CWC-NIL	ACA-CWC-NIL	SFF-CWC-NIL
AFF-CWC-COM	ACA-CWC-NIL	ARA-CWC-NIL	AFF-CWC-NIL	SFF-CWC-COM
AFC-CWC-NIL	ARA-CWC-NIL	AFF-CWC-NIL	AFF-RCC-NIL	SFF-CWC-RBK
AFC-CWC-COM	ARO-CWC-NIL	AFF-RCC-NIL	AFC-CWC-NIL	SFF-RCC-NIL
SRS-CWC-NIL	AFF-CWC-NIL	AFC-CWC-NIL	SCE-CWC-NIL	SFF-RCC-COM
SRS-CWC-COM	AFF-RCC-NIL	SRA-CWC-NIL	SRS-CWC-NIL	SFF-RCC-RBK
SRS-CWC-RBK	AFC-CWC-NIL	SRA-CWC-COM	SRS-CWC-COM	SFF-RAC-NIL
SFF-CWC-NIL	SCE-CWC-NIL	SRA-CWC-RBK	SRS-CWC-RBK	SFC-CWC-NIL
SFF-CWC-COM	SCE-CWC-COM	SRO-CWC-NIL	SFF-CWC-NIL	SFC-CWC-COM
SFF-CWC-RBK	SRS-CWC-NIL	SRO-CWC-COM	SFF-CWC-COM	
SFF-RCC-NIL	SRS-CWC-COM	SRO-CWC-RBK	SFF-CWC-RBK	
SFF-RCC-COM	SRS-CWC-RBK	SFF-CWC-NIL	SFF-RCC-NIL	
SFF-RCC-RBK	SRA-CWC-NIL	SFF-CWC-COM	SFF-RCC-COM	
SFC-CWC-NIL	SRA-CWC-COM	SFF-CWC-RBK	SFF-RCC-RBK	
SFC-CWC-COM	SRA-CWC-RBK	SFF-RCC-NIL	SFF-RAC-NIL	
SFC-CWC-RBK	SRO-CWC-NIL	SFF-RCC-COM	SFC-CWC-NIL	
	SRO-CWC-COM	SFF-RCC-RBK	SFC-CWC-COM	
	SRO-CWC-RBK	SFF-RAC-NIL		
	SFF-CWC-NIL	SFC-CWC-NIL		
	SFF-CWC-COM	SFC-CWC-COM		
	SFF-CWC-RBK			
	SFF-RCC-NIL			
	SFF-RCC-COM			
	SFF-RCC-RBK			
	SFC-CWC-NIL			
	SFC-CWC-COM			

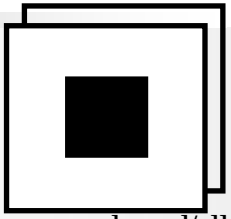
© YAWL Foundation

Exception Handling Primitives

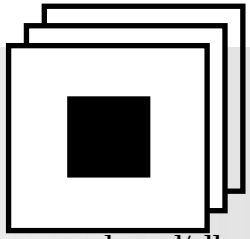




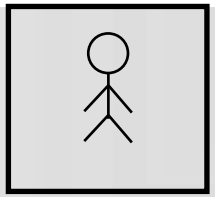
1. Remove current work item



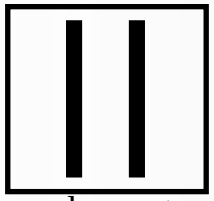
5. Remove selected/all work items in current case



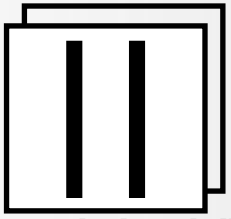
9. Remove selected/all work items in all cases



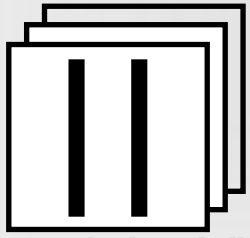
13. Reallocate current work item



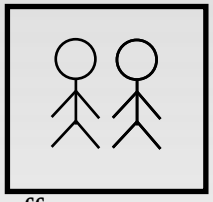
2. Suspend current work item



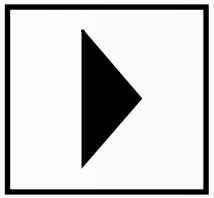
6. Suspend selected/all work items in current case



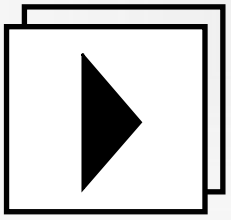
10. Suspend selected/all work items in all cases



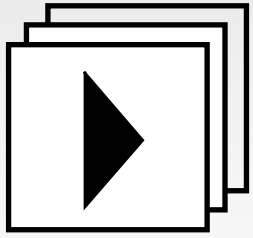
14. Reoffer current work item



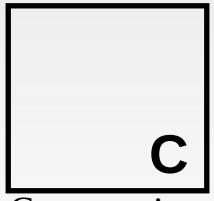
3. Continue current work item or thread



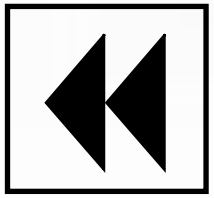
7. Continue selected/all work items in current case



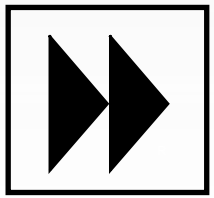
11. Continue selected/all work items in all cases



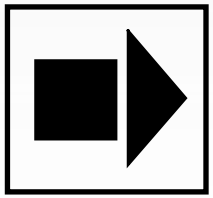
15. Compensation task



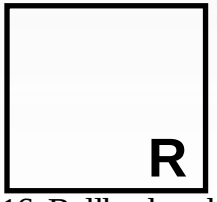
4. Restart current work item



8. Force complete current work item



12. Force fail current work item

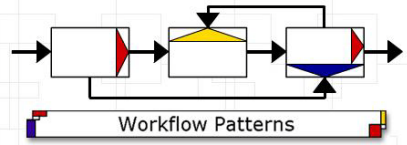


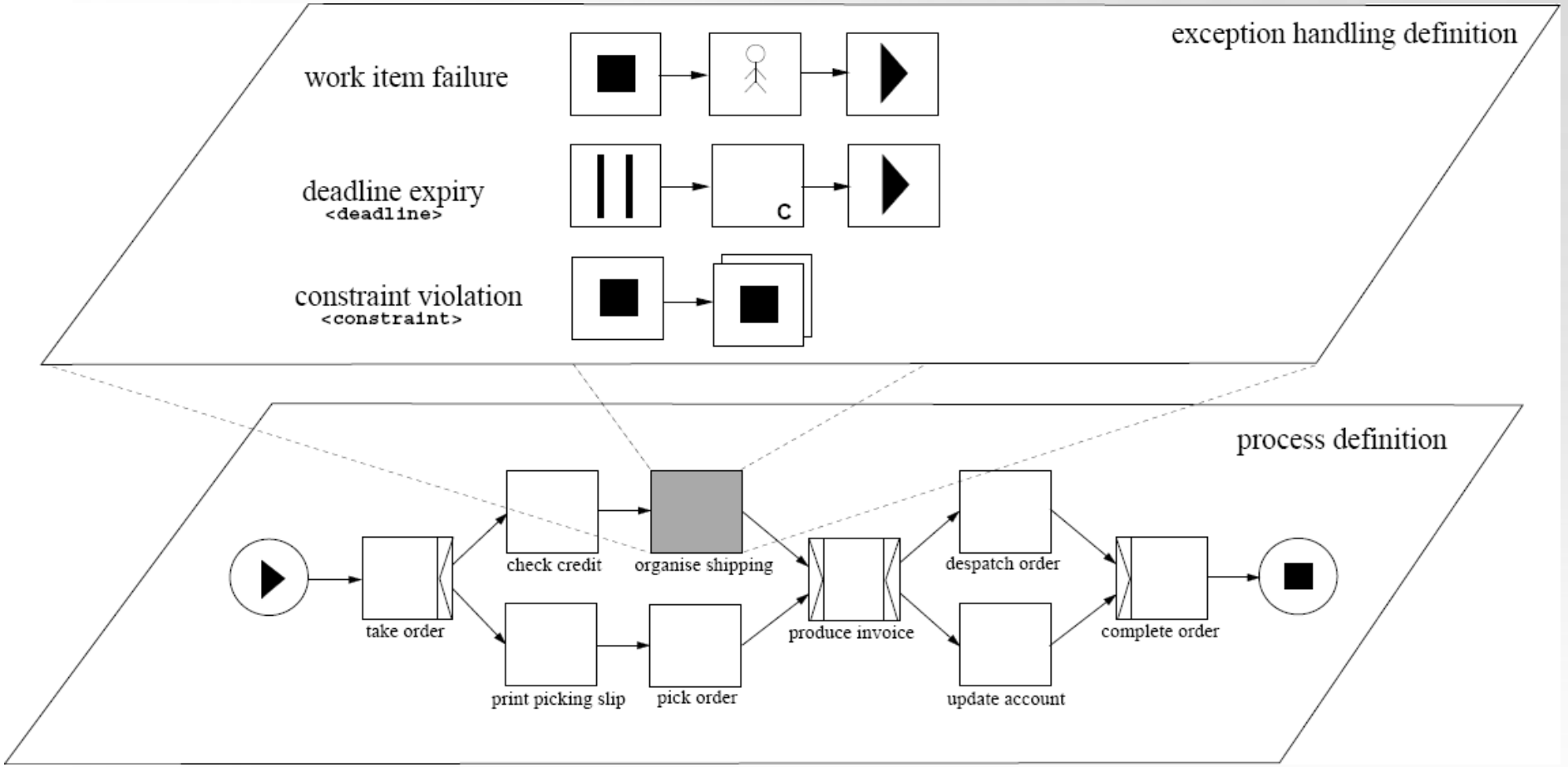
16. Rollback task

© YAWL Foundation

© YAWL Foundation

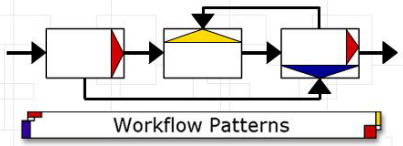
A Generic Architecture for Exception Handling





© 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

© YAWL Foundation

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 Foundation

YAWL Overview

- Collaboration between TU/e and QUT
- Based on Workflow Patterns Initiative
- YAWL: 2002 – newYAWL: 2007
- System development
  - Open source (currently LGPL)
  - Industry collaboration
    - M2 Investments – first:telecom
    - Intercontinental Hotels Group
- Main publication
  - [AH05] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language, Information Systems 30(4):245-275, 2005
- URLs:
  - www.yawlfoundation.org (research)
  - www.sourceforge.net/projects/yawl (system)
  - http://www.yawlgroup.com (consultancy)

© YAWL Foundation

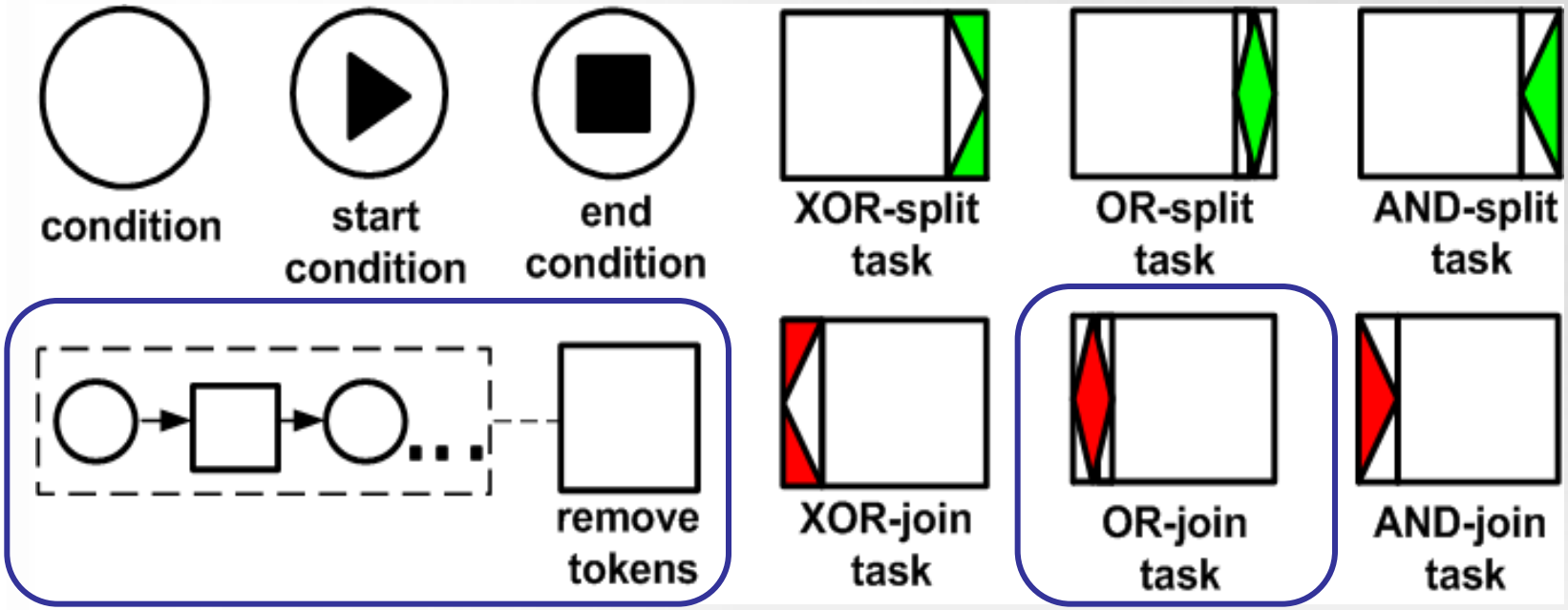
YAWL vs Petri nets

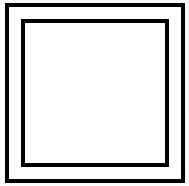
- Petri nets have difficulties capturing:
  1. The General Synchronising Merge
  2. Patterns involving Multiple Instances
  3. Cancellation of a certain part of a process

- 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)

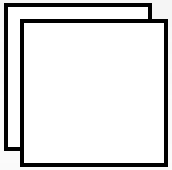
© YAWL Foundation

YAWL notation





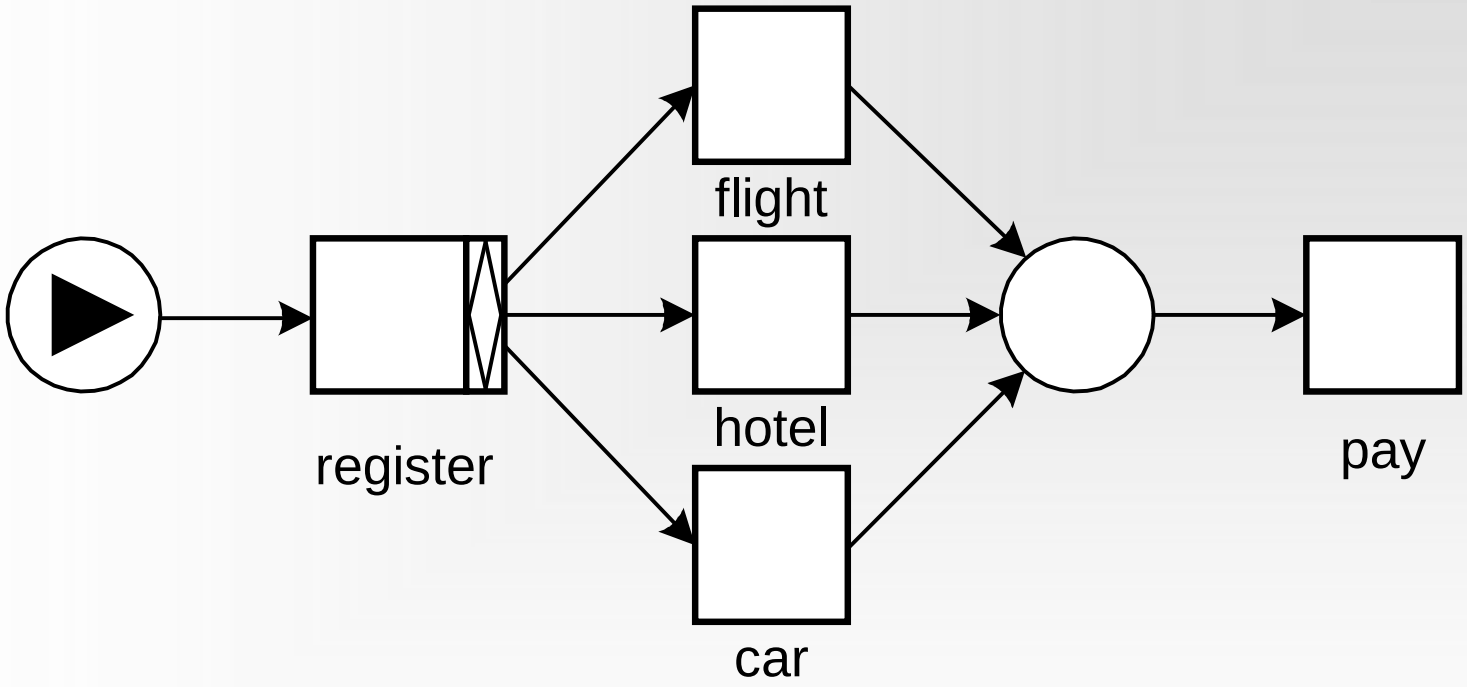
Composite task



Multiple Instance task

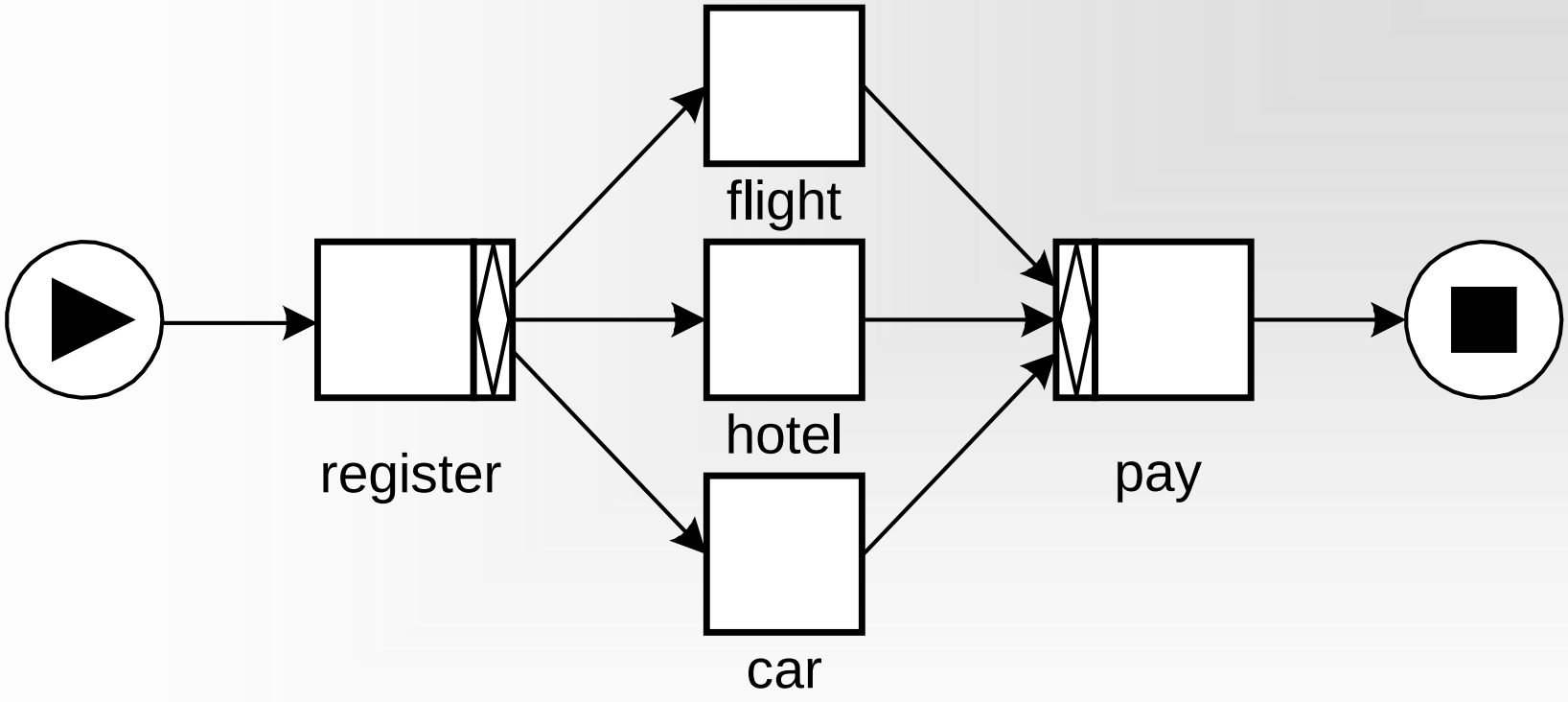
© YAWL Foundation

YAWL Example I



© YAWL Foundation

YAWL Example II

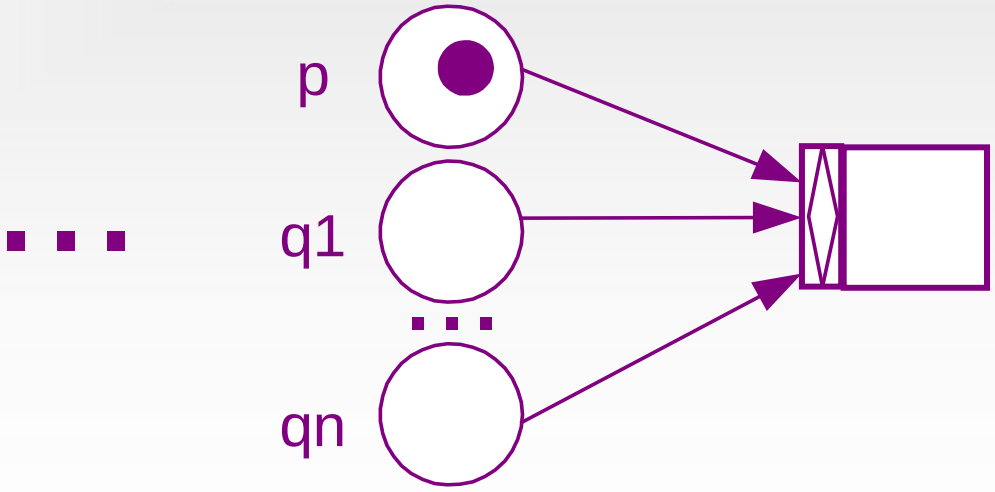


© YAWL Foundation

The OR-join in YAWL

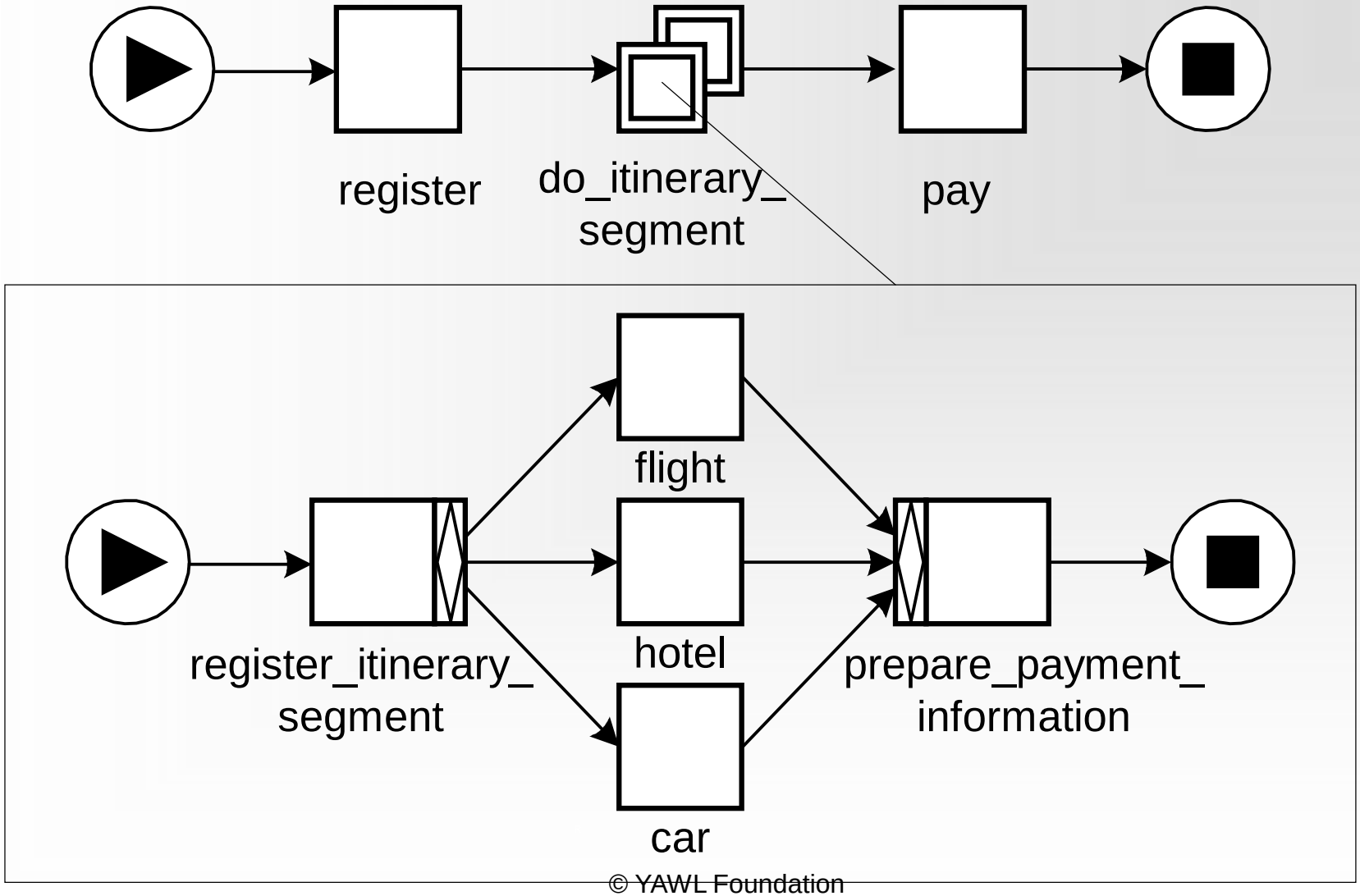
- The OR-join as originally proposed in YAWL has been generalised by Moe Wynn et al [WEAH05].
- The formalisation exploits algorithms for coverability analysis in reset nets
- An OR-join is enabled iff one of its input places, say p, is marked, and it is not possible to reach a marking from the current 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 Foundation

YAWL Example III



© YAWL Foundation

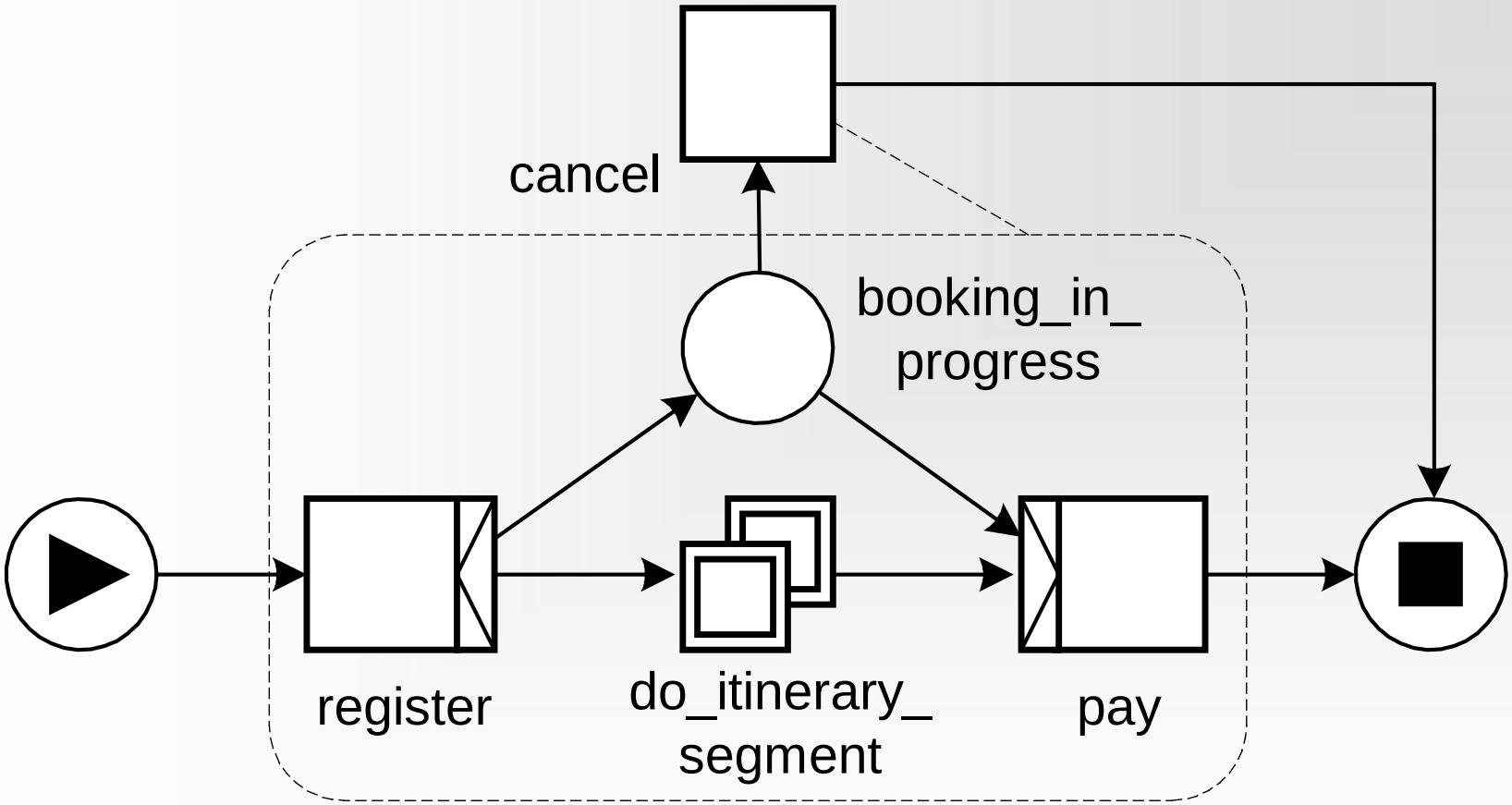
© YAWL Foundation

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.

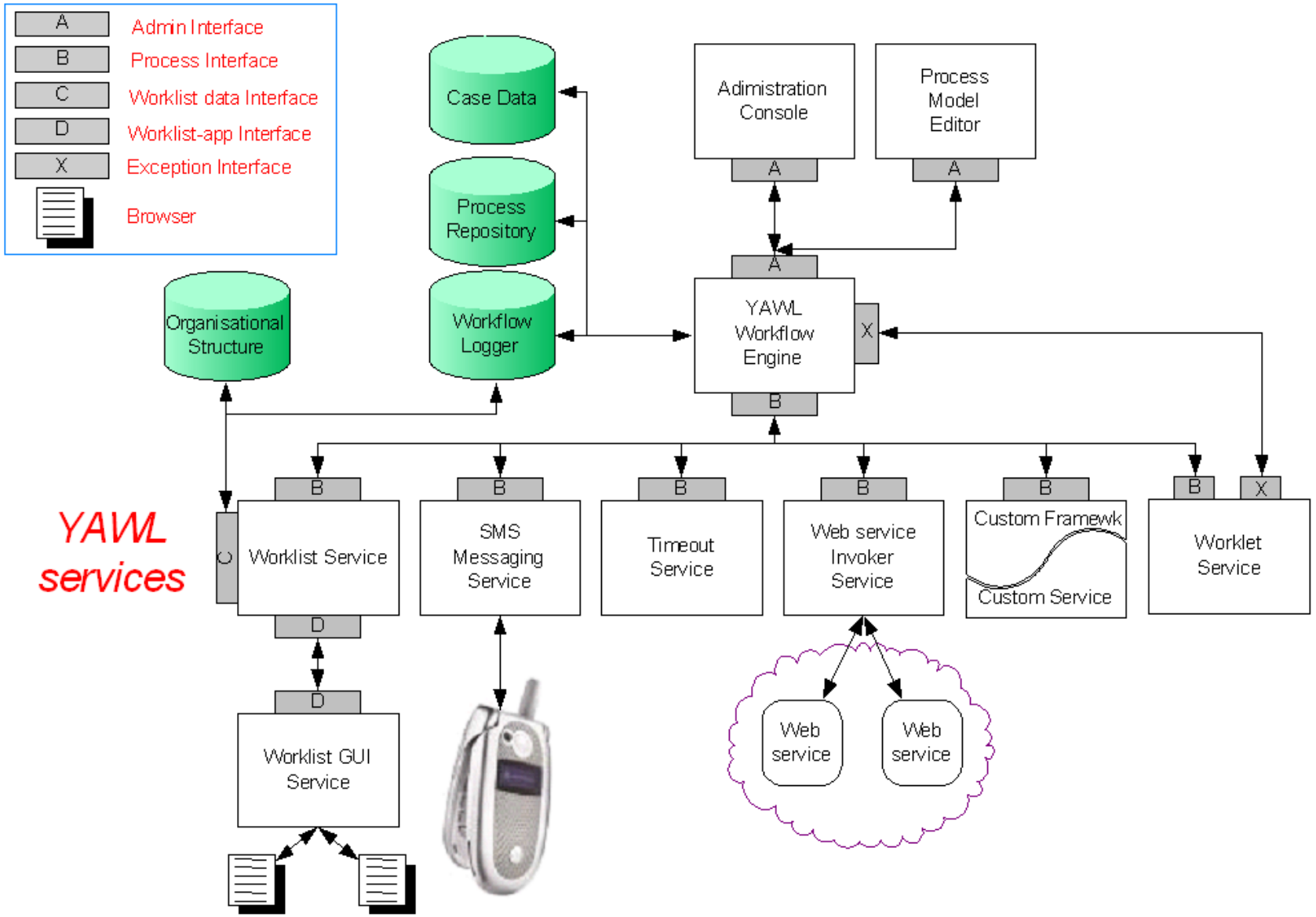
© YAWL Foundation

General YAWL Example V



© YAWL Foundation

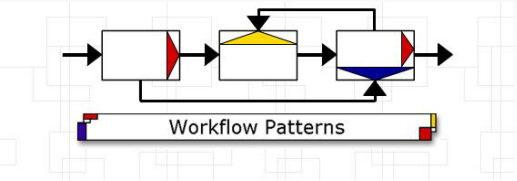
Service Oriented Architecture



© YAWL Foundation

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 Foundation

Results – Open Source Tools: Control-flow perspective

1 – jBPM
2 – Open WFE
3 – YAWL
4 – new YAWL

		1	2	3	4
Basic Control-flow					
1	Sequence	+	+	+	+
2	Parallel Split	+	+	+	+
3	Synchronisation	+	+	+	+
4	Exclusive Choice	+	+/-	+	+
5	Simple Merge	+	+	+	+
Advanced Synchronisation					
6	Multiple Choice	-	+/-	+	+
7	Str. Synchronising Merge	-	-	+	+
8	Multiple Merge	+	-	+	+
9	Structured Discriminator	-	+	+	+
28	Blocking Discriminator	-	-	-	+
29	Cancelling Discriminator	-	+	+	+
30	Structured Partial Join	-	+	-	+
31	Blocking Partial Join	-	-	-	+
32	Cancelling Partial Join	-	+	-	+
33	Generalised AND-Join	+	-	+	+
37	Local Sync. Merge	+	+/-	+	+
38	General Sync. Merge	-	-	+	+
41	Thread Merge	-	-	-	+
42	Thread Split	-	-	-	+
Iteration					
10	Arbitrary Cycles	+	+/-	+	+
21	Structured Loop	+/-	+	-	+
22	Recursion	-	+	-	+

		1	2	3	4
Termination					
11	Implicit Termination	+/-	+	+/-	-
43	Explicit Termination	-	-	-	+
Multiple Instances					
12	MI without Synchronisation	+	+	+	+
13	MI with a priori Design Time Knlg	-	+	+	+
14	MI with a priori Runtime Knlg	-	+	+	+
15	MI without a priori Runtime Knlg	-	+	+	+
27	Complete MI Activity	-	-	-	+
34	Static Partial Join for MI	-	+	-	+
35	Static Canc. Partial Join for MI	-	+	+	+
36	Dynamic Partial Join for MI	-	-	-	+
State-based					
16	Deferred Choice	+	-	+	+
39	Critical Section	-	-	+	+
17	Interleaved Parallel Routing	-	+/-	+	+
40	Interleaved Routing	-	+	+	+
18	Milestone	-	-	+	+
Cancellation					
19	Cancel Activity	-	-	+	+
20	Cancel Case	-	+	+	+
25	Cancel Region	-	-	+	+
26	Cancel MI Activity	-	-	+	+
Trigger					
23	Transient Trigger	-	-	-	+
24	Persistent Trigger	-	-	-	+

© YAWL Foundation

Results – Open Source Tools: Data perspective

1 – jBPM
2 – Open WFE
3 – YAWL
4 – new YAWL

		1	2	3	4
Data Visibility					
1	Task Data	+/-	-	-	+
2	Block Data	+/-	-	+	+
3	Scope Data	-	+/-	-	+
4	MI Data	+/-	-	+	+
5	Case Data	+	+	-	+
6	Folder Data	+/-	-	-	+
7	Workflow Data	+/-	+	-	+
8	Environment Data	+/-	+	-	+
Data Interaction (Internal)					
9	between Tasks	+	+	+	+
10	Task to Sub-workflow Decomp.	+	+	+	+
11	Sub-workflow Decomp. to Task	+	+	+	+
12	to MI Task	+/-	-	+	+
13	from MI Task	+/-	-	+	+
14	Case to Case	+/-	+/-	-	+
Data Interaction (External)					
15	Task to Env - Push-Oriented	+/-	-	-	+
16	Env. to Task - Pull-Oriented	+/-	-	-	+
17	Env. to Task - Push-Oriented	+/-	-	-	+
18	Task to Env - Pull-Oriented	+/-	-	-	+
19	Case to Env. - Push-Oriented	+/-	+	-	+
20	Env. to Case - Pull-Oriented	+/-	+	-	+

		1	2	3	4
Data Interaction (External), cont.					
21	Env. to Case - Push-Oriented	+/-	+	-	+
22	Case to Env. - Pull-Oriented	+/-	+	-	+
23	Workflow to Env. - Push-Oriented	+/-	+	-	+
24	Env. to Workflow - Pull-Oriented	+/-	+	-	+
25	Env. to Workflow - Push-Oriented	+/-	+	-	+
26	Workflow to Env. - Pull-Oriented	+/-	+	-	+
Data Transfer					
27	by Value - Incoming	-	-	+	+
28	by Value - Outgoing	-	-	+	+
29	Copy In/Copy Out	+	-	-	+
30	by Reference - Unlocked	+	+	-	-
31	by Reference - Locked	-	-	-	+/-
32	Data Transformation - Input	+	+	+	+
33	Data Transformation - Output	+	+	+	+
Data-based Routing					
34	Task Precondition Data Exist.	-	+	-	+
35	Task Precondition Data Value	-	+	-	+
36	Task Postcondition Data Exist.	-	-	-	+
37	Task Postcondition Data Value	-	-	-	+
38	Event-based Task Trigger	-	-	-	+
39	Data-based Task Trigger	-	-	-	+
40	Data-based Routing	+	+	+	+

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

1 – jBPM
2 – Open WFE
3 – YAWL
4 – new YAWL

		1	2	3	4
Creation Patterns					
1	Direct Allocation	+	-	+	+
2	Role-Based Allocation	+	+	+	+
3	Deferred Allocation	-	+	-	+
4	Authorization	-	+	-	+
5	Separation of Duties	-	-	-	+
6	Case Handling	-	-	-	-
7	Retain Familiar	-	-	-	+
8	Capacity-based Allocation	-	-	-	+
9	History-based Allocation	-	-	-	+
10	Organizational Allocation	-	-	-	+
11	Automatic Execution	+	+	+	+
Push Patterns					
12	Distribution by Offer-Single Resource	-	-	+	+
13	Distribution by Offer-Multiple Resources	-	+	+	+
14	Distribution by Allocation-Single Resource	+	-	-	+
15	Random Allocation	-	-	-	+
16	Round Robin Allocation	-	-	-	+
17	Shortest Queue	-	-	-	+
18	Early Distribution	-	-	-	-
19	Distribution on Enablement	+	+	+	+
20	Late Distribution	-	-	-	+
Pull Patterns					
21	Resource-Init. Allocation	-	-	+	+
22	Resource-Init. Exec. - Allocated WI	-	-	+	+
23	Resource-Init. Exec. - Offered WI	-	+	-	+

		1	2	3	4
Pull Patterns (cont.)					
24	System-Determ. Work Queue Cont.	-	-	-	+
25	Resource-Determ. Work Queue Cont.	-	-	-	+
26	Selection Autonomy	-	+	-	+
Detour Patterns					
27	Delegation	-	-	-	+
28	Escalation	-	+	-	+
29	Deallocation 1)	-	-	-	+
30	Stateful Reallocation	-	+	-	+
31	Stateless Reallocation	-	+	-	+
32	Suspension/Resumption	-	-	-	+
33	Skip	-	-	-	+
34	Redo	-	-	-	-
35	Pre-do	-	-	-	-
Auto-start Patterns					
36	Commencement on Creation	+	-	-	+
37	Commencement on Allocation	-	+	-	+
38	Piled Execution	-	-	-	+
39	Chained Execution	-	-	-	+
Visibility Patterns					
40	Config. Unallocated WI Visibility	-	+	-	+
41	Config. Allocated WI Visibility	-	+	-	+
Multiple Resource Patterns					
42	Simultaneous Execution	+	-	-	+
43	Additional Resource	-	-	-	-

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

Thanks!

Questions?

© YAWL Foundation