Business Process Modelling and Workflow Patterns

Arthur ter Hofstede

Queensland University of Technology
Brisbane, Australia

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
  - Marcello La Rosa, QUT
  - Nick Russell, QUT
  - Petia Wohed, SU/KTH
  - Moe Wynn, QUT

Outline

- Background – WfMS and PAIS
- Conceptual Foundation - the Workflow Patterns Initiative
  - Part I
    - Control-flow patterns
    - Data patterns
    - Resource patterns
- Next-Generation Business Process Management with YAWL
  - Part II
    - The YAWL language
    - The YAWL system

Terminology

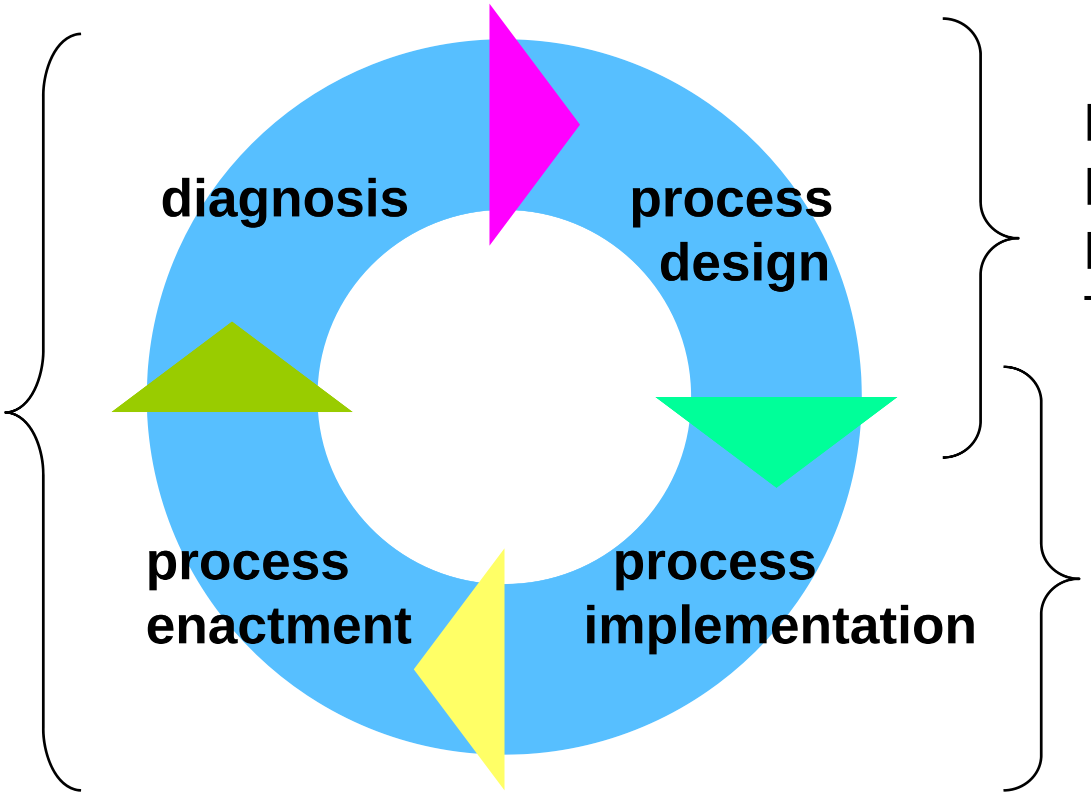
- WF
  - "The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action according to a set of procedural rules."

    WfMC, Terminology & Glossary, WFMC-TC-1011 3.0, February 1999

- 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, Process-Aware Information Systems: Bridging People and Software through Process Technology, John Wiley & Sons, 2005

The PAIS life cycle



diagnosis

process design

process implementation

process enactment

Project Management Tools

Business Process Modelling Tools

Workflow Management Systems

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

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

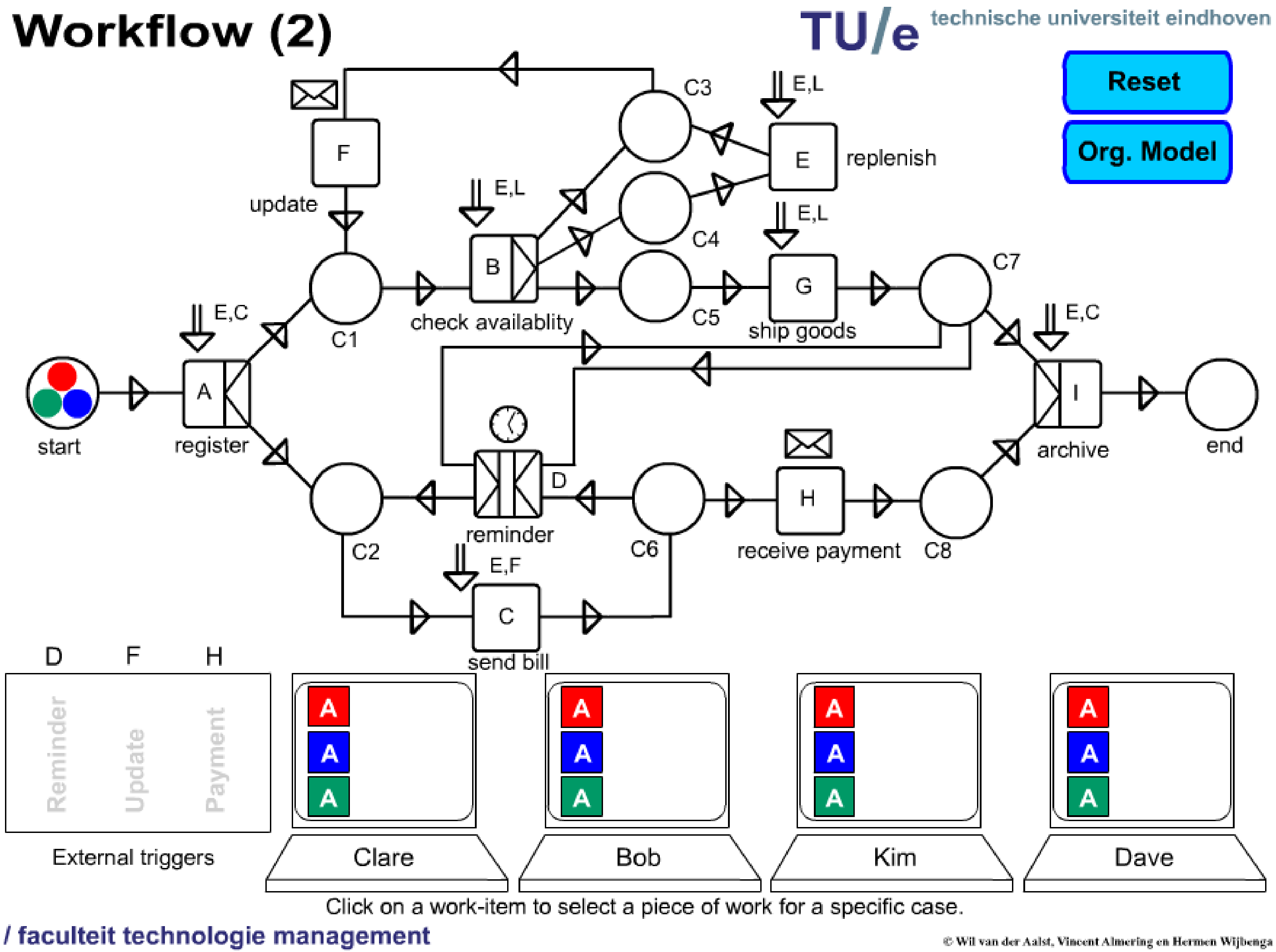
Perspectives

- Control-Flow
  - Which tasks need to be executed and in what order
- Data
  - What data elements exist, to whom are they visible, how are they passed on
- Resources
  - Who is authorised to execute certain tasks, are tasked assigned by the system or can participants volunteer for their execution, on what basis is work assigned

Sometimes these perspectives are explained in terms of *Who (Resource)*, *What (Data)* and *When (Control-flow)*

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

Workflow Animation



Workflow (2)

TU/e technische universiteit eindhoven

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

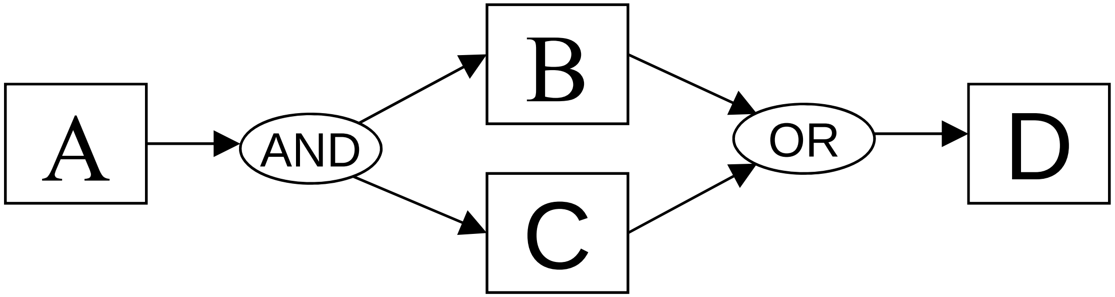
/ faculteit technologie management

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

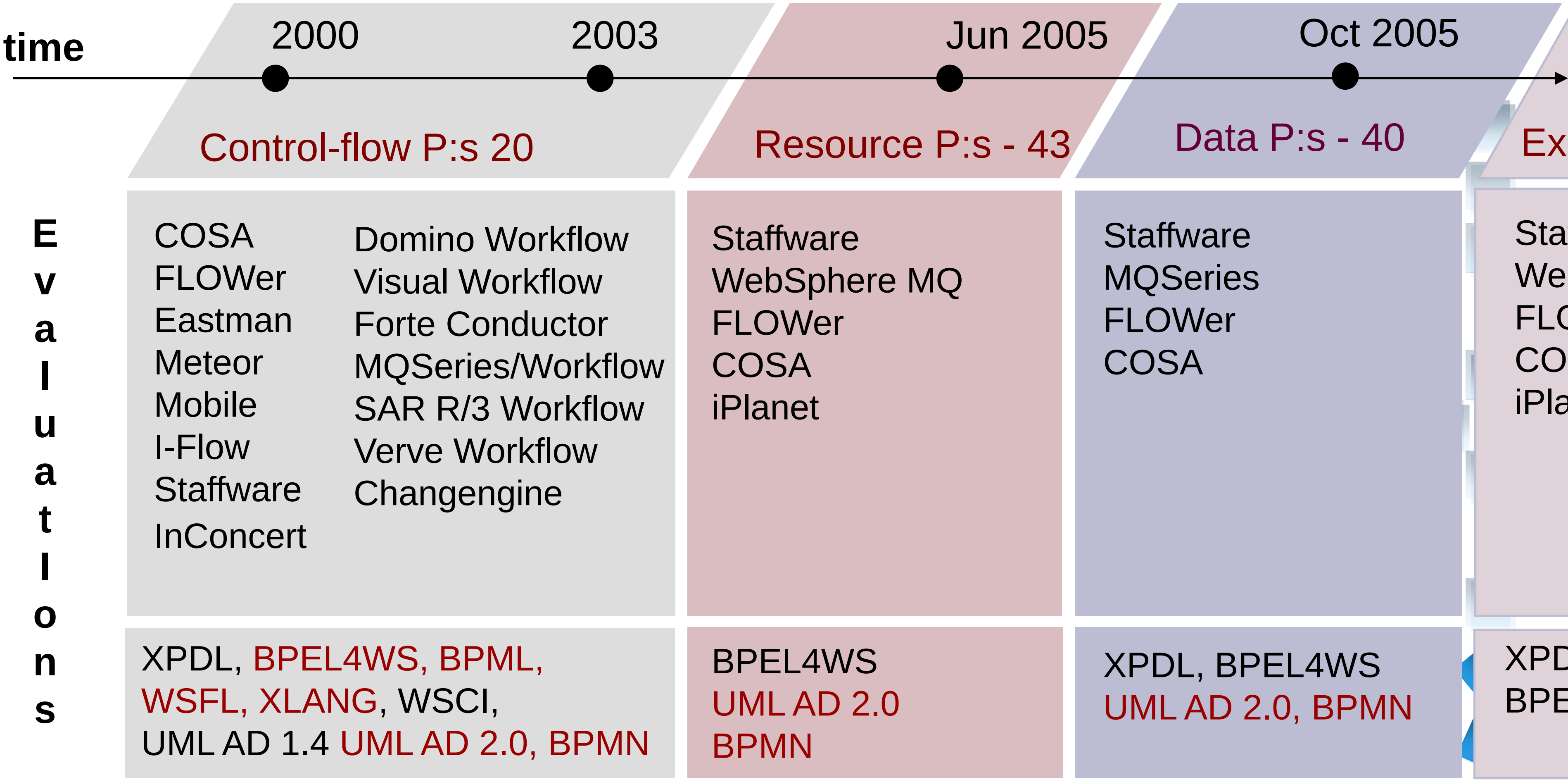
- **Started in 1999, joint work TU/e and QUT**
- **Objectives:**
  - Identification of workflow modelling scenarios and solutions
  - Benchmarking
    - Workflow products (MQ/Series Workflow, Staffware, etc)
    - Proposed standards for web service composition (BPML, BPEL)
    - Process modelling languages (UML, BPMN)
    - Open Source BPM offerings (jBPM, OpenWFE, Enhydra Shark)
  - 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**

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

The Workflow Patterns Framework



time

2000

2003

Jun 2005

Oct 2005

Control-flow P:s 20

Resource P:s - 43

Data P:s - 40

Ex

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

Sta
We
FLC
CO
iPla

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

XPD
BPE

Language Development: YAWL/newYAWL

The power of expressiveness

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
- Lynx Workflow 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 (1000+ according to Google Scholar)
- Education (used in teaching at 10+ Universities)

Why revisit control flow?

- Lack of precision
  - Solution: CPN formalisation + Context conditions
- Merging of various concepts
  - Solution: A number of patterns were split
- Certain patterns missing
  - Solution: Addition of those patterns

Approach

- All original patterns were reviewed
- All continue to serve a valid purpose
  - Some were retained in original form but with more explicit context conditions and detailed semantics
  - Others were subdivided along different dimensions (the most basic interpretation replaced the original pattern, the others were added)
- New patterns were identified and included
- Explicit evaluation criteria were added
- Evaluations were completely revised

The New Control-flow Patterns

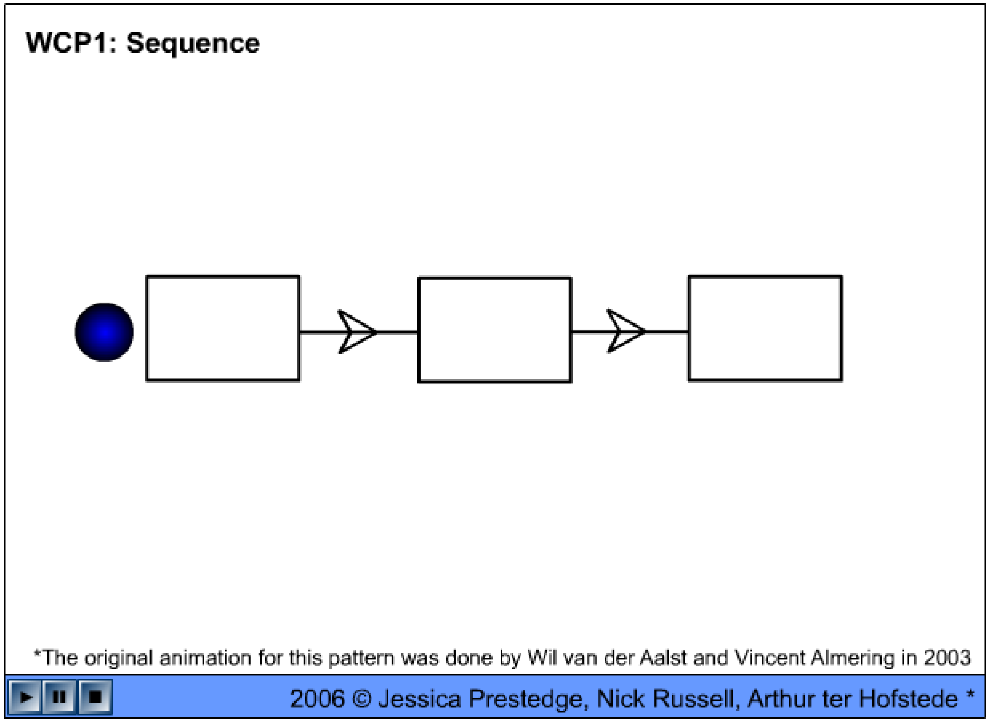
- Basic Control-flow Patterns
  capture elementary aspects of control-flow (similar to the concepts provided by the WFMC).
- Advanced Branching and Synchronization Patterns
  describe more complex branching and synchronization scenarios.
- Iteration Patterns
  describe various ways in which iteration 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 with multiple threads of execution in a workflow which relate to the same activity.
- State-based Patterns
  reflect situations which are most easily modelled in WF languages with an explicit 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.

Structure of a Control-flow pattern

- **Description**: *what is it?*
- **Synonym(s)**
- **Example(s)**
- **Motivation**: *why needed?*
- **Context**: *conditions + CPN formalisation*
- **Implementation**: *how typically realised?*
- **Issues**: *what problems can be encountered?*
- **Solutions**: *how and to what extent can these problems be overcome?*
- **Evaluation criterion**

Sequence

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



WCP1: Sequence

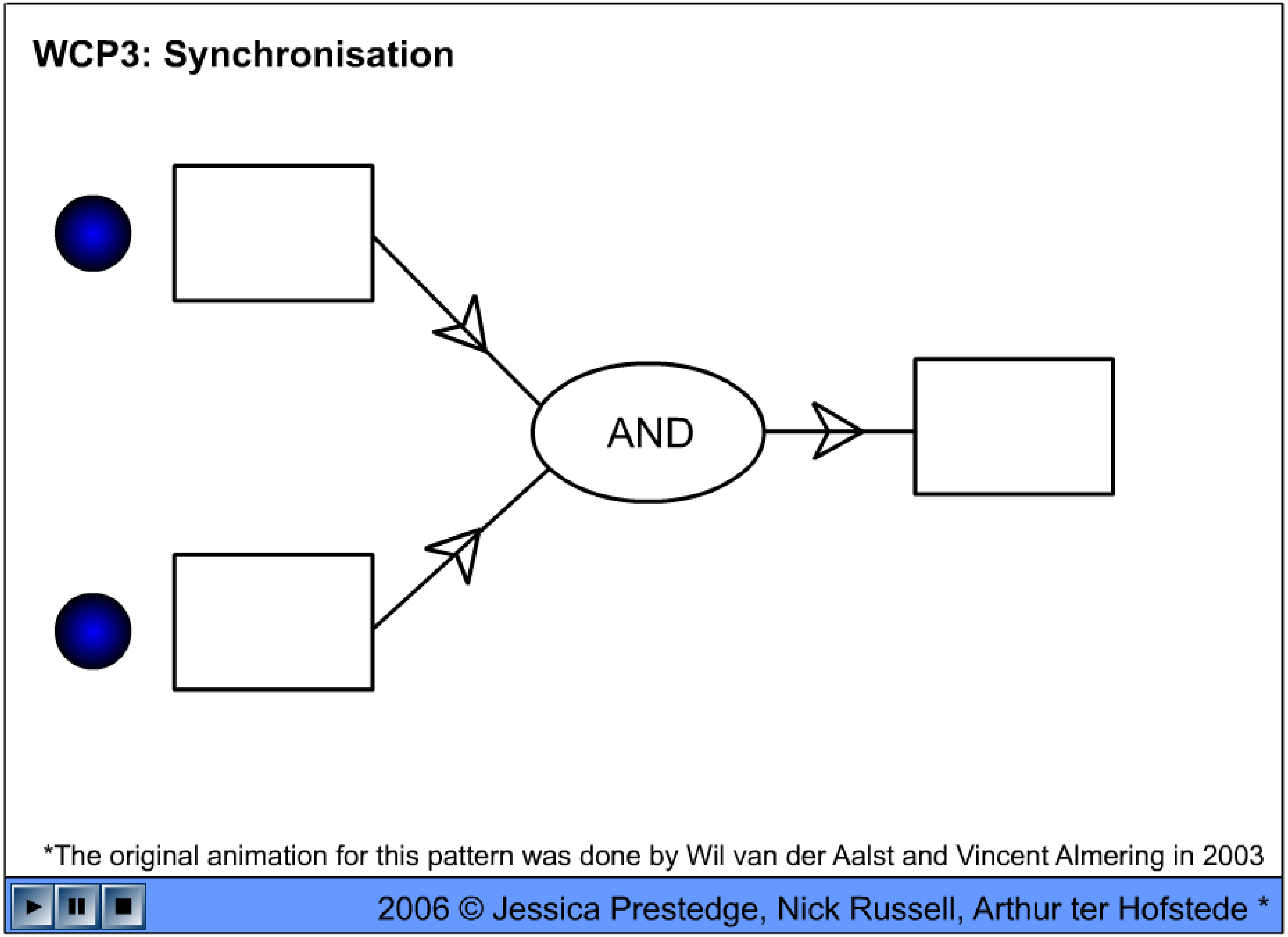
*The original animation for this pattern was done by Wil van der Aalst and Vincent Almering in 2003

2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede *

Basic Control-flow Patterns

- Pattern 2: Parallel split, initiation of parallel threads
- Pattern 3: Synchronisation, wait for all incoming threads
  - context assumption: each incoming branch will signal exactly once
- Pattern 4: Exclusive choice, thread of control is passed to exactly one of the outgoing branches
- Pattern 5: Simple merge, executes upon receiving signal of one of its incoming branches
  - context assumption: no preceding parallelism

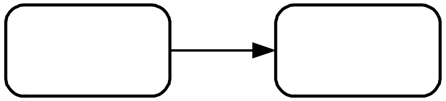
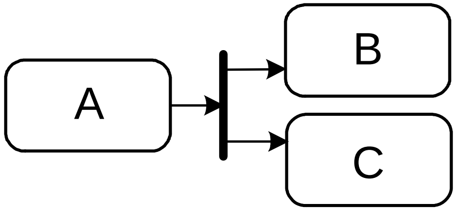
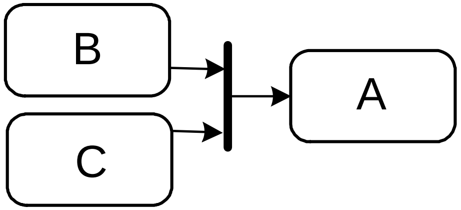
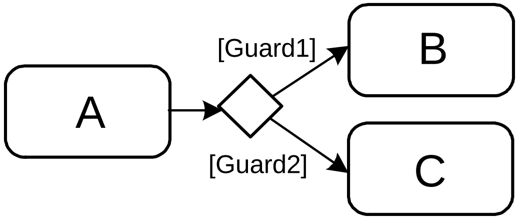
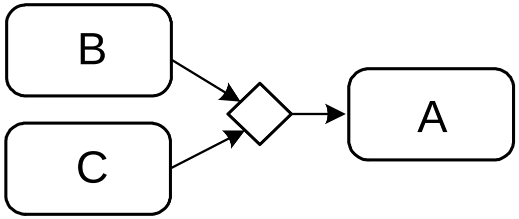
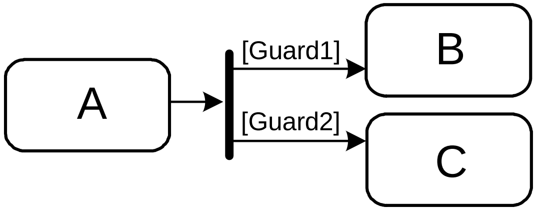
Synchronisation: Animation



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

2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede *

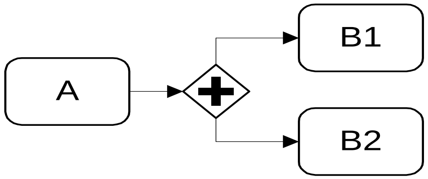
Basic Control-flow Patterns in UML2.0AD

Sequence	Parallel Split	Synchronisation
 **b)** Control flow in UML	 **d)** UML Explicit AND-split	 **f)** UML Explicit AND-join
Exclusive Choice	**Simple Merge/Multple Merge**	**Multiple Choice**
 **h)** UML Explicit XOR-split	 **j)** UML XOR-join	 **l)** UML OR-split

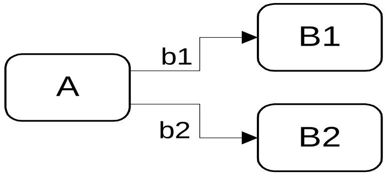
Basic Control-flow Patterns in BPMN

Several solutions for the most basic patterns

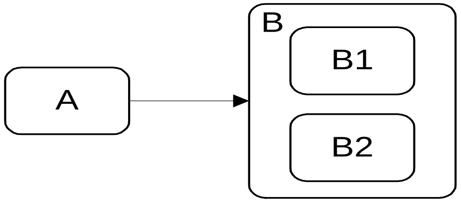
Parallel Split



a) with AND-gateway

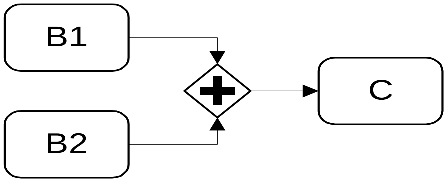


b) Implicit

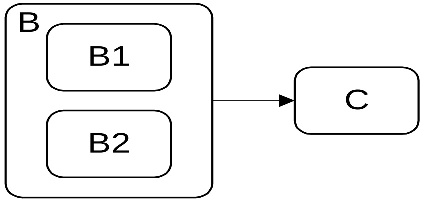


c) through sub-Activities

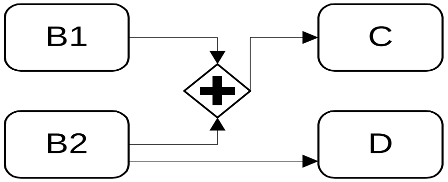
Synchronisation



d) with AND-gateway

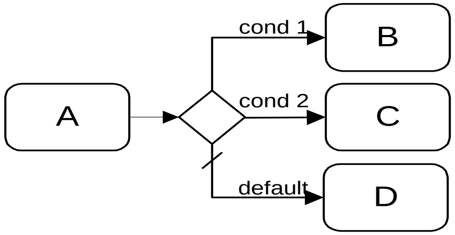


e) partially through sub-Activities

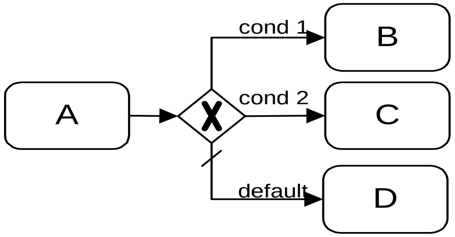


f) in a context

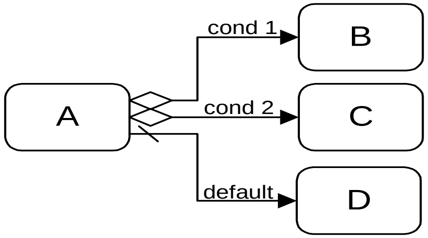
Exclusive Choice



g) with XOR-gateway, alt 1

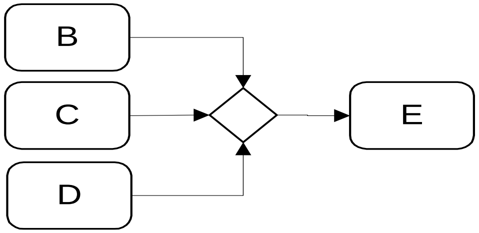


h) with XOR-gateway, alt 2

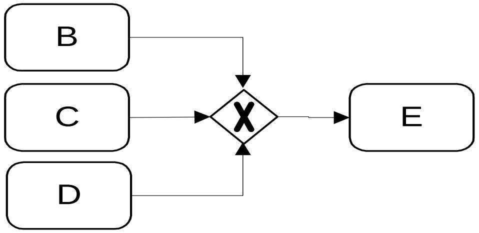


i) without XOR-gateway

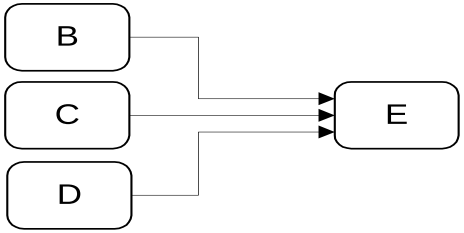
Merge



j) with XOR-gateway, alt 1



k) with XOR-gateway, alt 2



l) Implicit

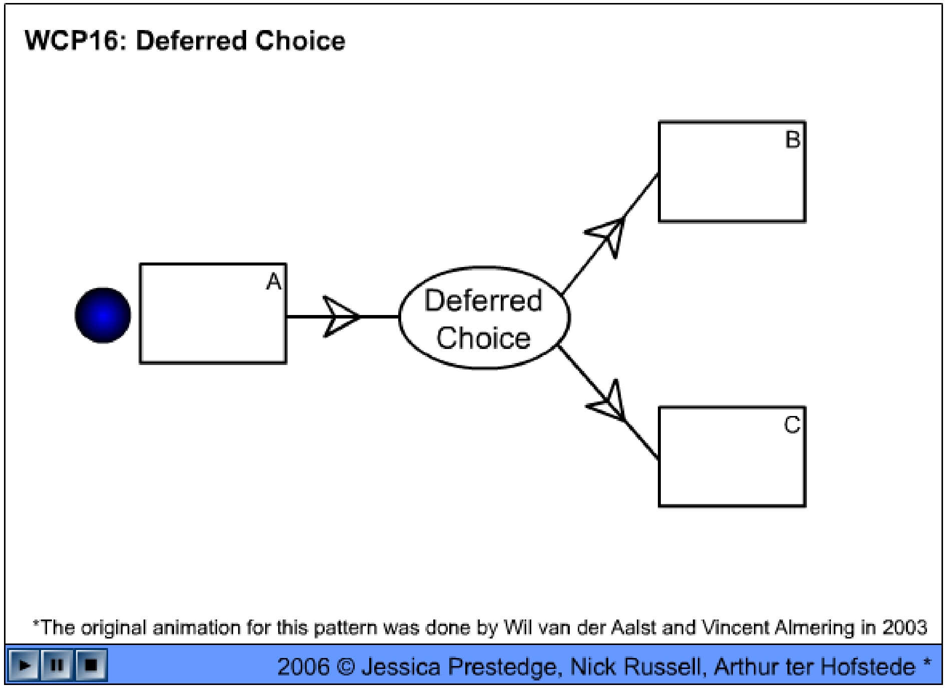
Deferred Choice

- Choice made by the environment not the system
- Essential in workflow context
- Not widely supported, though its importance seems to be increasingly recognised (e.g. BPEL)
- Naturally supported by notations that offer direct support for the notion of state, e.g. statecharts or Petri nets

WCP 4 Exclusive Choice

- Choice made by the system, based on data

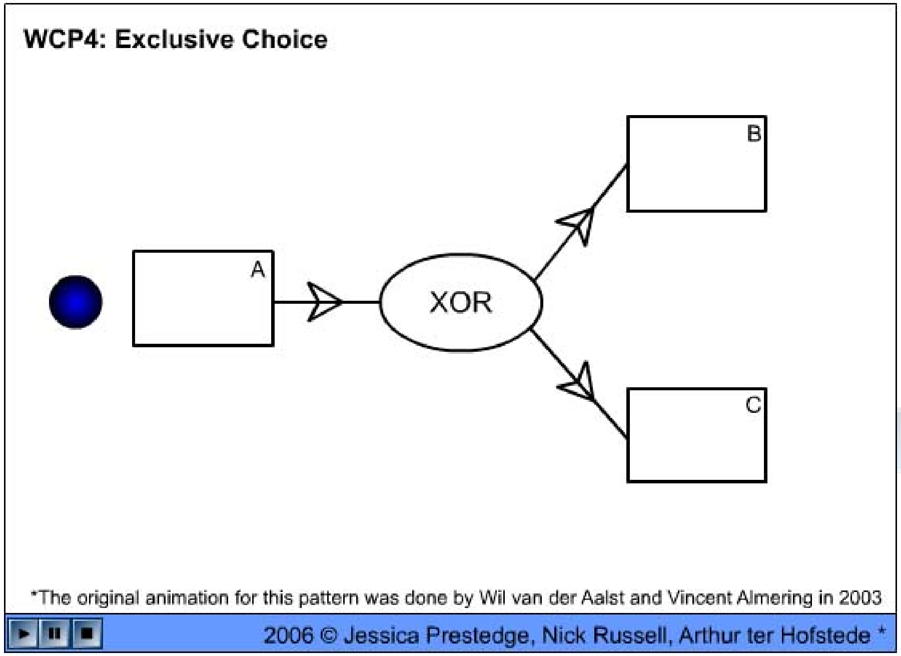
Deferred Choice vs Exclusive Choice: Animations



WCP16: Deferred Choice

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

2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede *



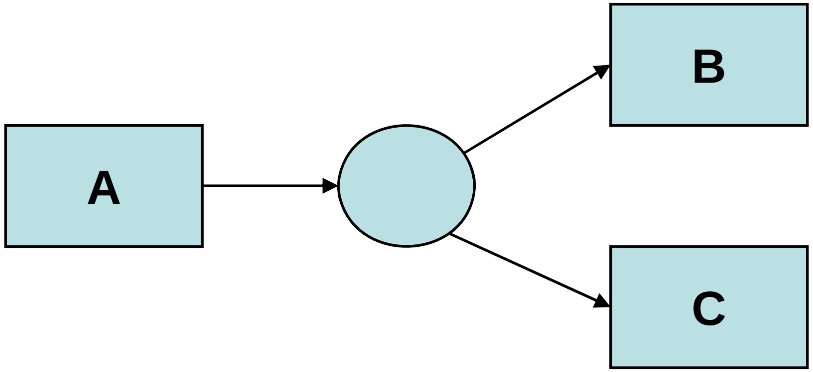
WCP4: Exclusive Choice

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

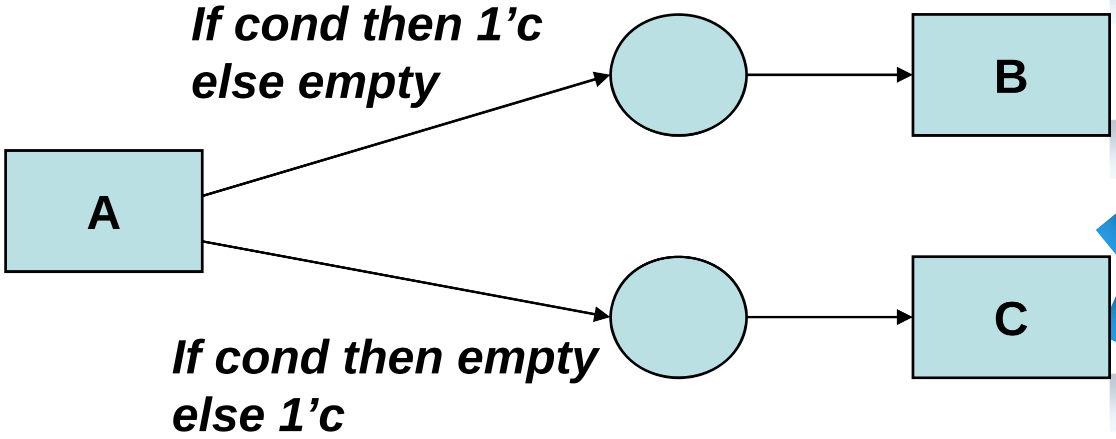
2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede *

Deferred Choice vs Exclusive Choice: Definitions

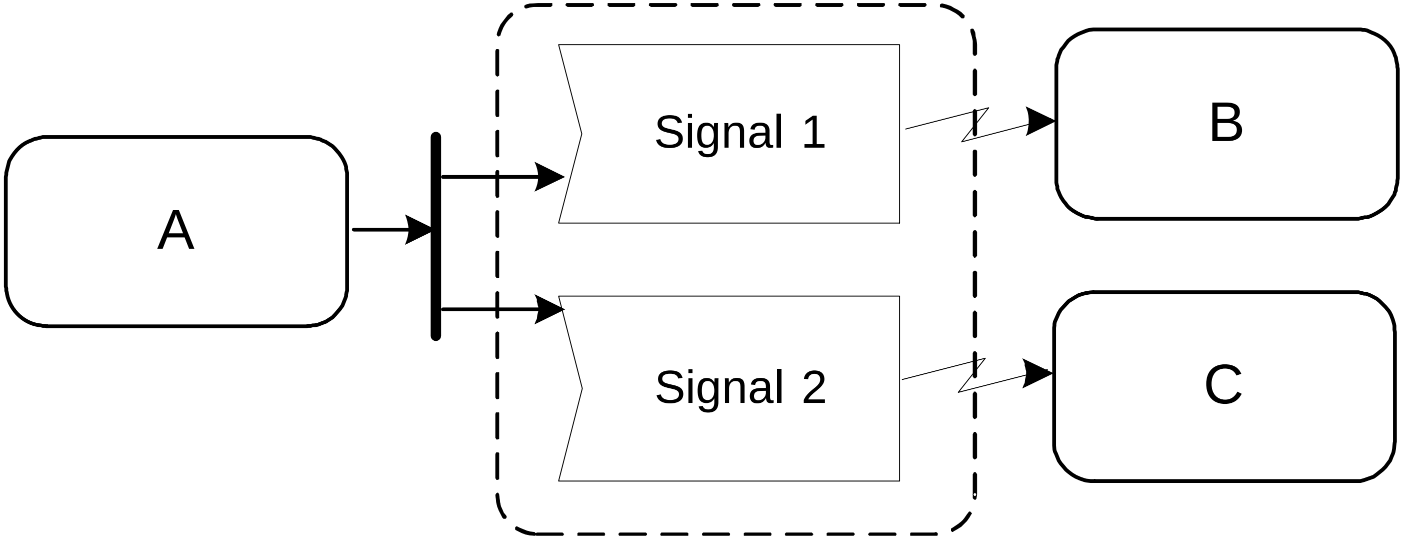
Deferred choice



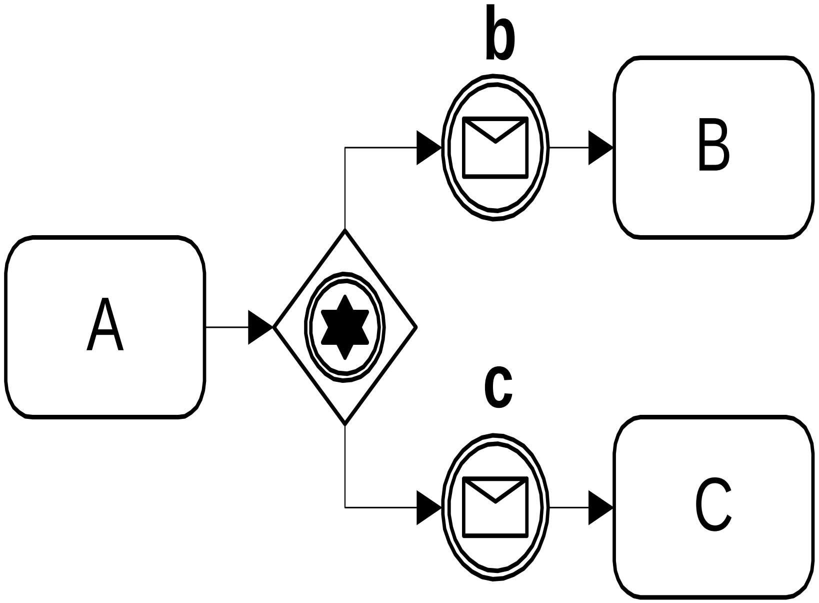
Exclusive choice



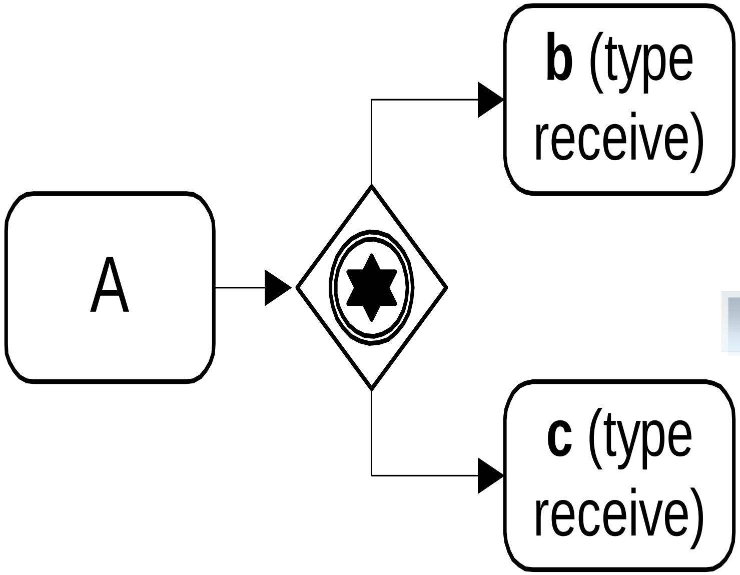
UML 2.0 AD - Deferred Choice



BPMN – Deferred Choice



with Event-Based Exclusive Gateway and Message Events



with Event-Based Exclusive Gateway and Receive Activities

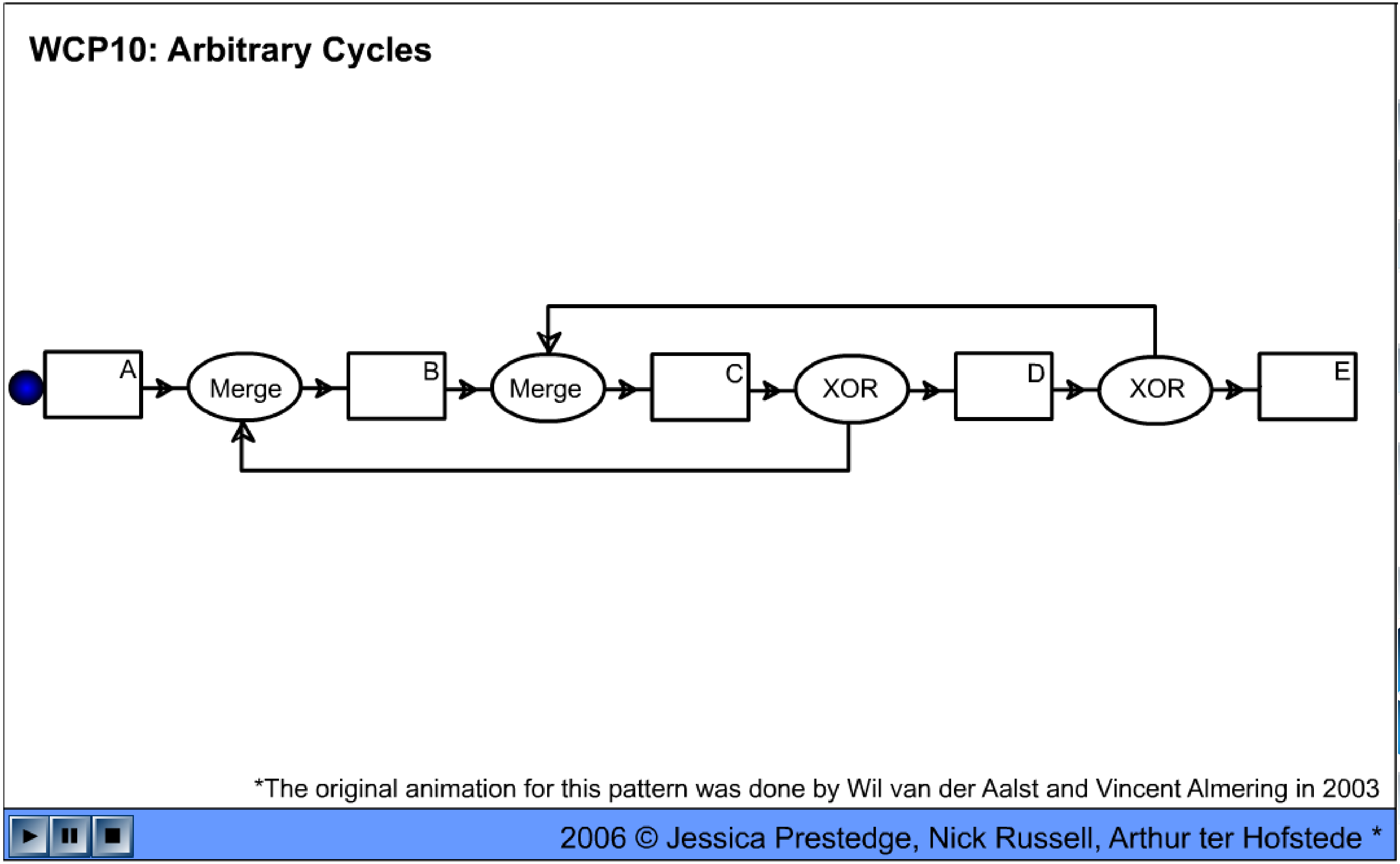
Arbitrary Cycles

- The ability to represent cycles in a process model that have more than one entry or exit point.

- Not all arbitrary cycles can be converted into structured ones (e.g. while or repeat loops; for further details see [KtHB00] and [Kie03]).

- Block structured offerings such as WebSphere MQ, FLOWer, SAP Workflow and BPEL are not able to represent arbitrary process structures.

Arbitrary Cycles: Animation



WCP10: Arbitrary Cycles

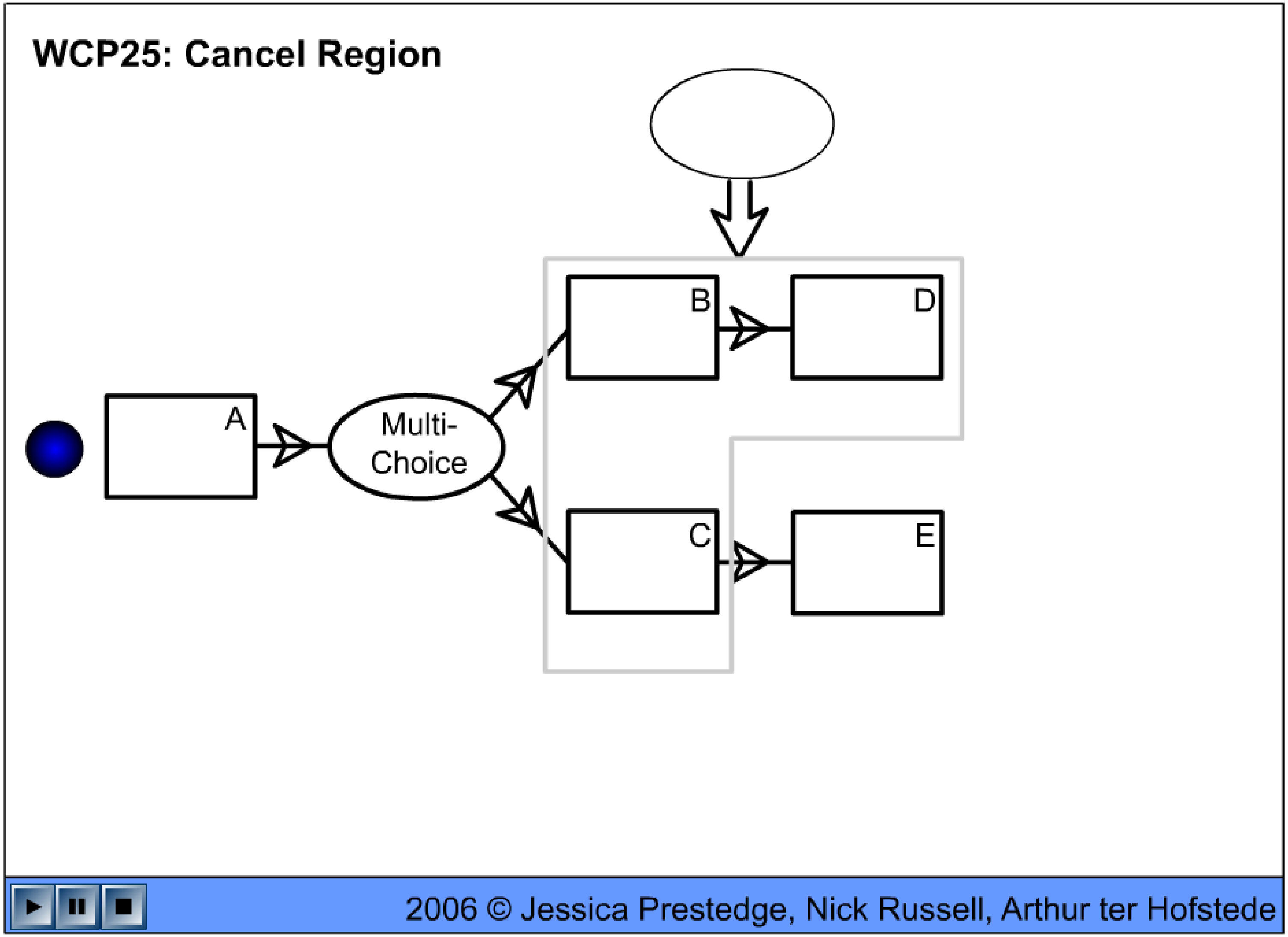
*The original animation for this pattern was done by Wil van der Aalst and Vincent Almering in 2003

2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede *

Cancellation Region

- Generalisation of Cancel Activity and Cancel Case
- Region associated with a task
- This region is emptied of tokens upon completion of that task.
- Rarely fully supported (only UML 2.0 Ads through InterruptibleActivityRegion construct and YAWL)

Cancellation Region: Animation



2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

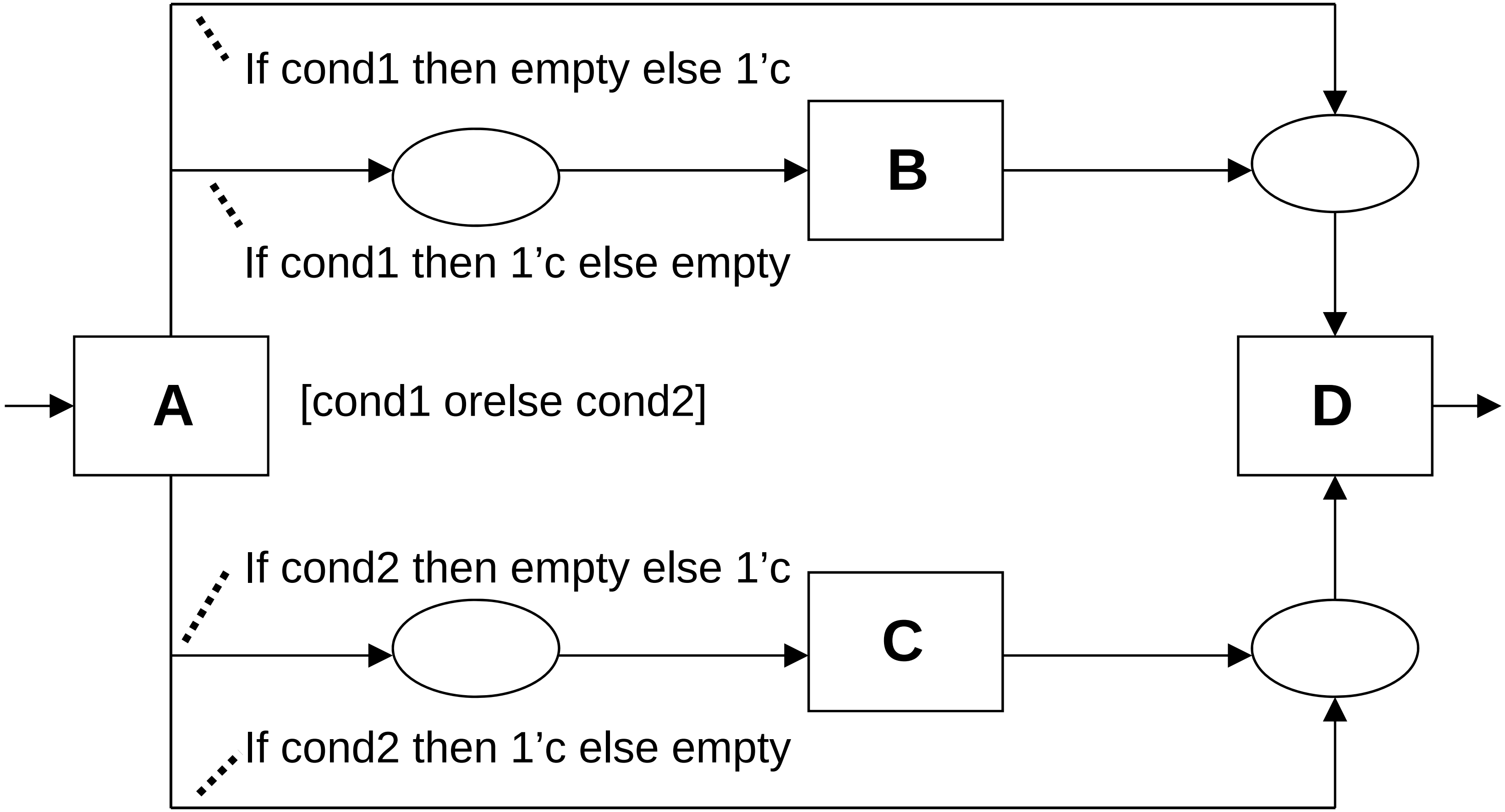
Structured Synchronising Merge

- 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.

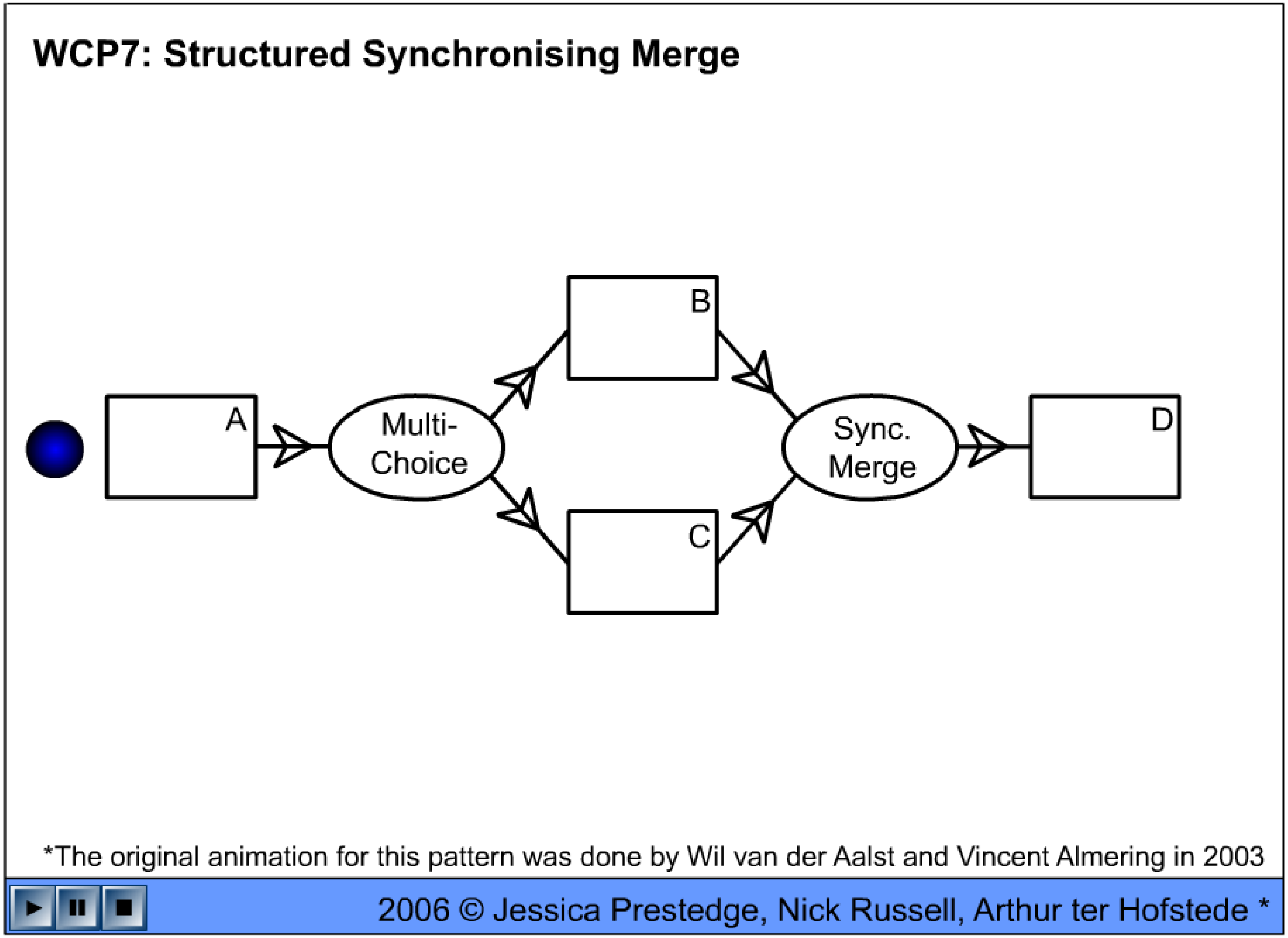
- Context conditions:
  - Single earlier corresponding multichoice
  - While merge has not fired, this multichoice cannot be re-enabled
  - No cancellation of selective branches after firing multichoice

These conditions are such that firing decision can be made based on local knowledge

Structured Synchronising Merge: Definition


If cond1 then empty else 1’c
B
If cond1 then 1’c else empty
A
[cond1 orelse cond2]
D
If cond2 then empty else 1’c
C
If cond2 then 1’c else empty

Structured Synchronising Merge: Animation



WCP7: Structured Synchronising Merge

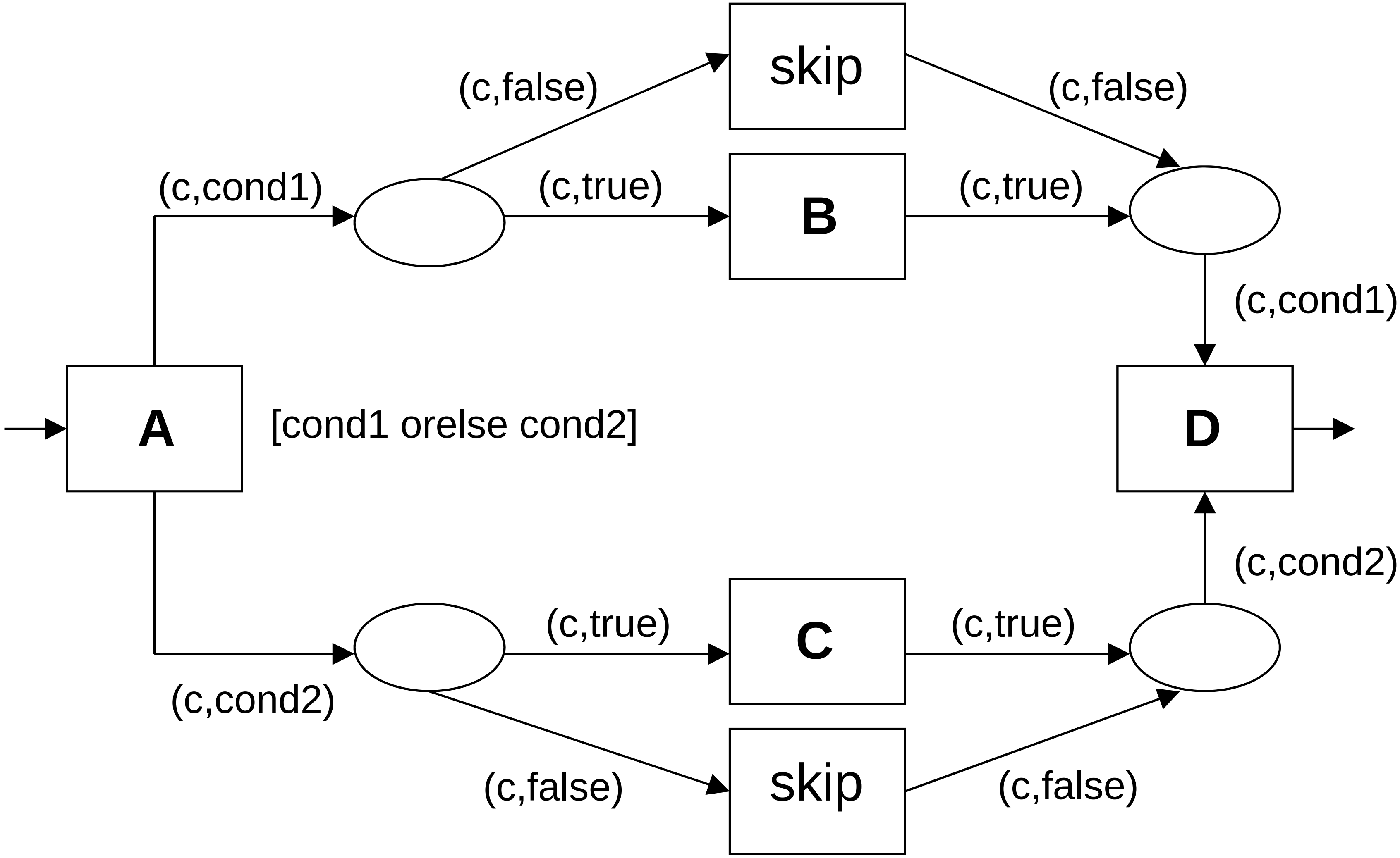
*The original animation for this pattern was done by Wil van der Aalst and Vincent Almering in 2003

2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede *

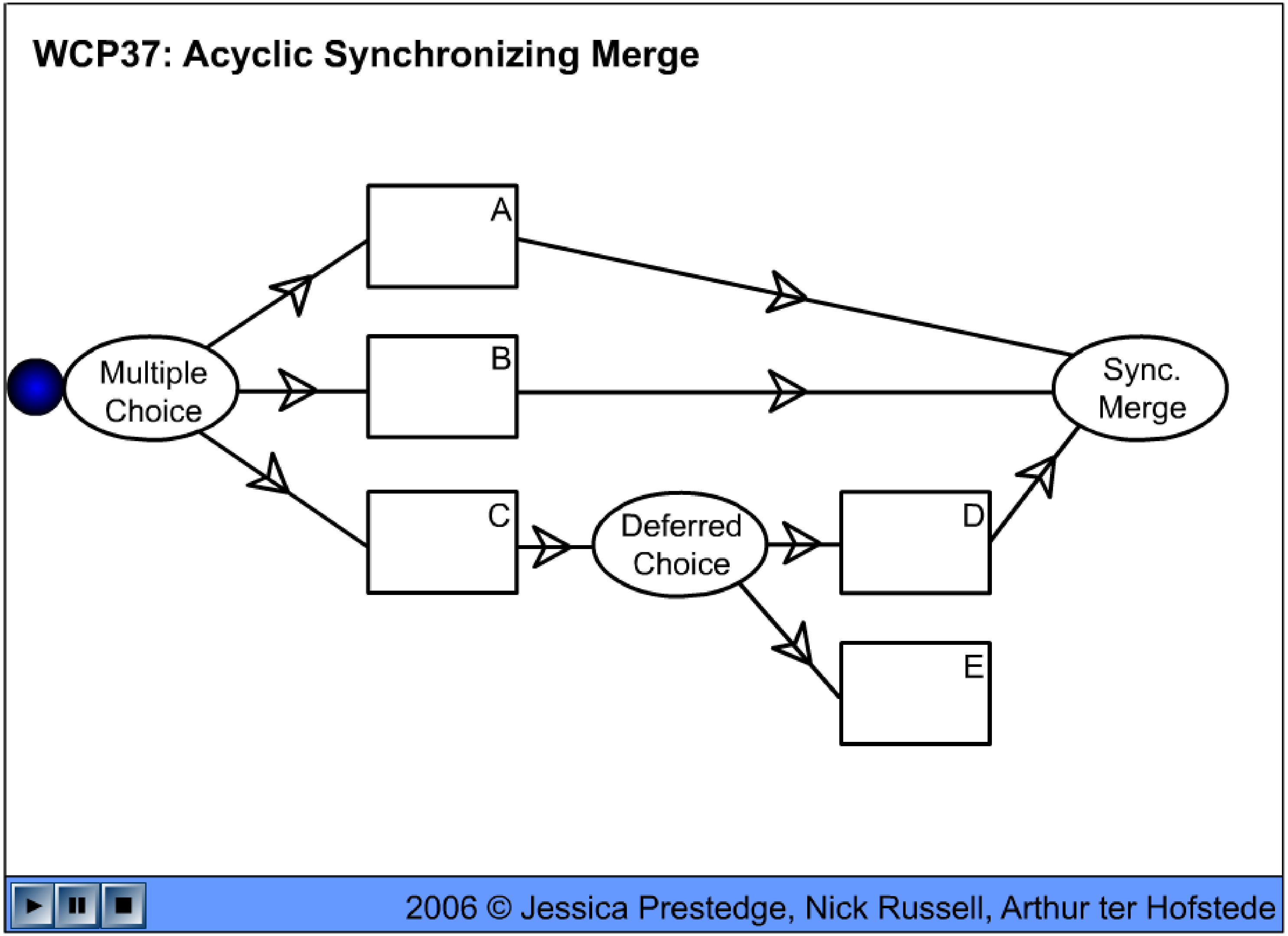
Acyclic Synchronising Merge

- This pattern does not require the process model to be structured, but its usage in the context of loops is restricted
- A possible realisation is through so-called “dead-path elimination” (see MQSeries – note the implication this has on the types of loops allowed)
- Evaluation of whether this merge is enabled can still be done locally: *Have you seen tokens on all incoming branches?*

Acyclic Synchronizing Merge: Definition



Acyclic Synchronizing Merge: Animation



WCP37: Acyclic Synchronizing Merge

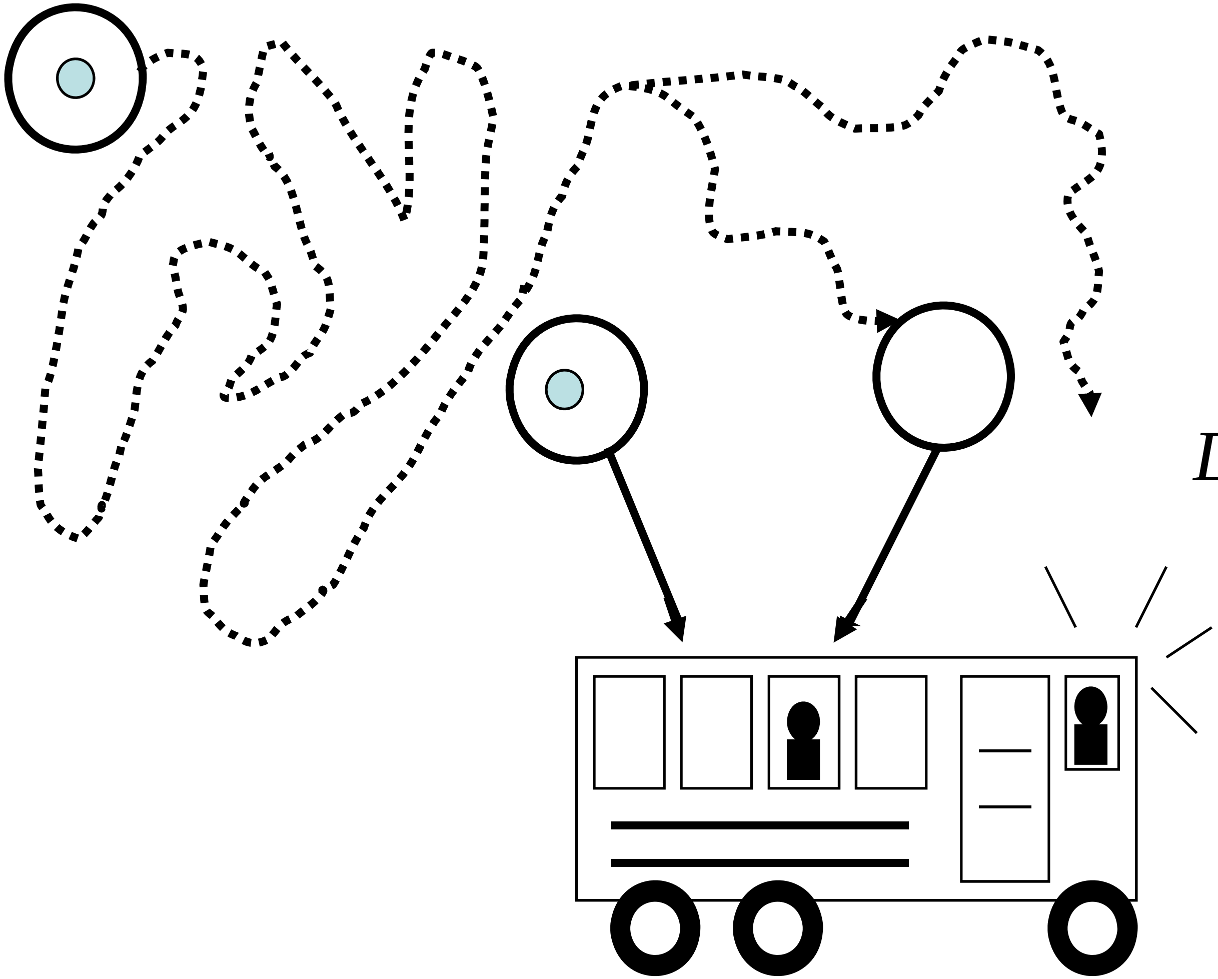
2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

A (Very) Complex CF Pattern: The General OR - Join

- Basically: *Wait only if you have to*
- Problems:
  - How should you treat other OR-joins?
  - How do you deal with cycles?
  - How do you treat decomposed tasks?
  - Is an analysis of the future possible?
    How complex will it be?

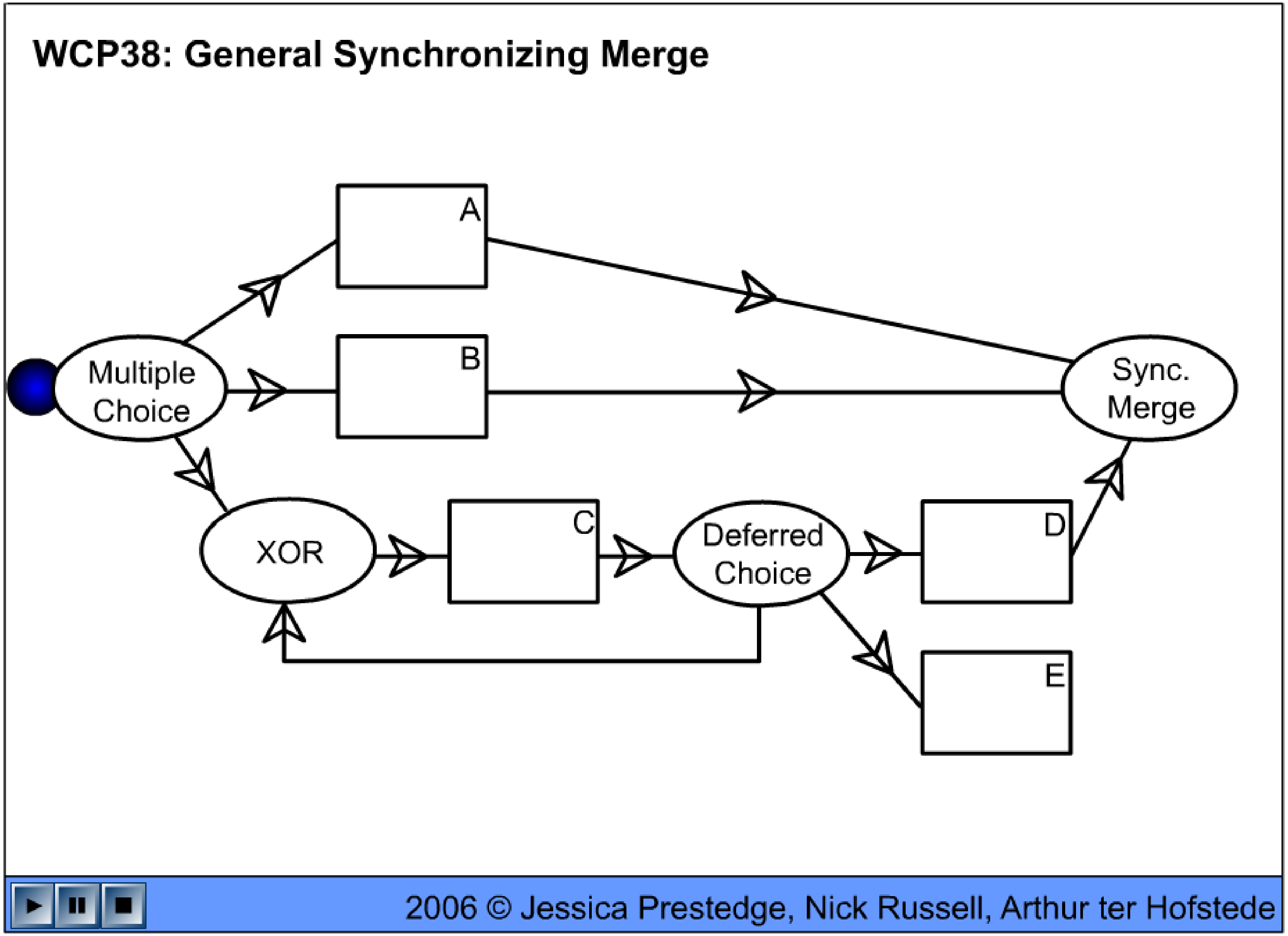
(for a detailed discussion see [WEAH05] and work by Kindler et al)

Non-local semantics of General OR - Join ("bus driver semantics")



Not only in EPCs but also in many WFM systems: Domino Workflow, Eastman, MQ Series, etc.

General Synchronizing Merge: Animation



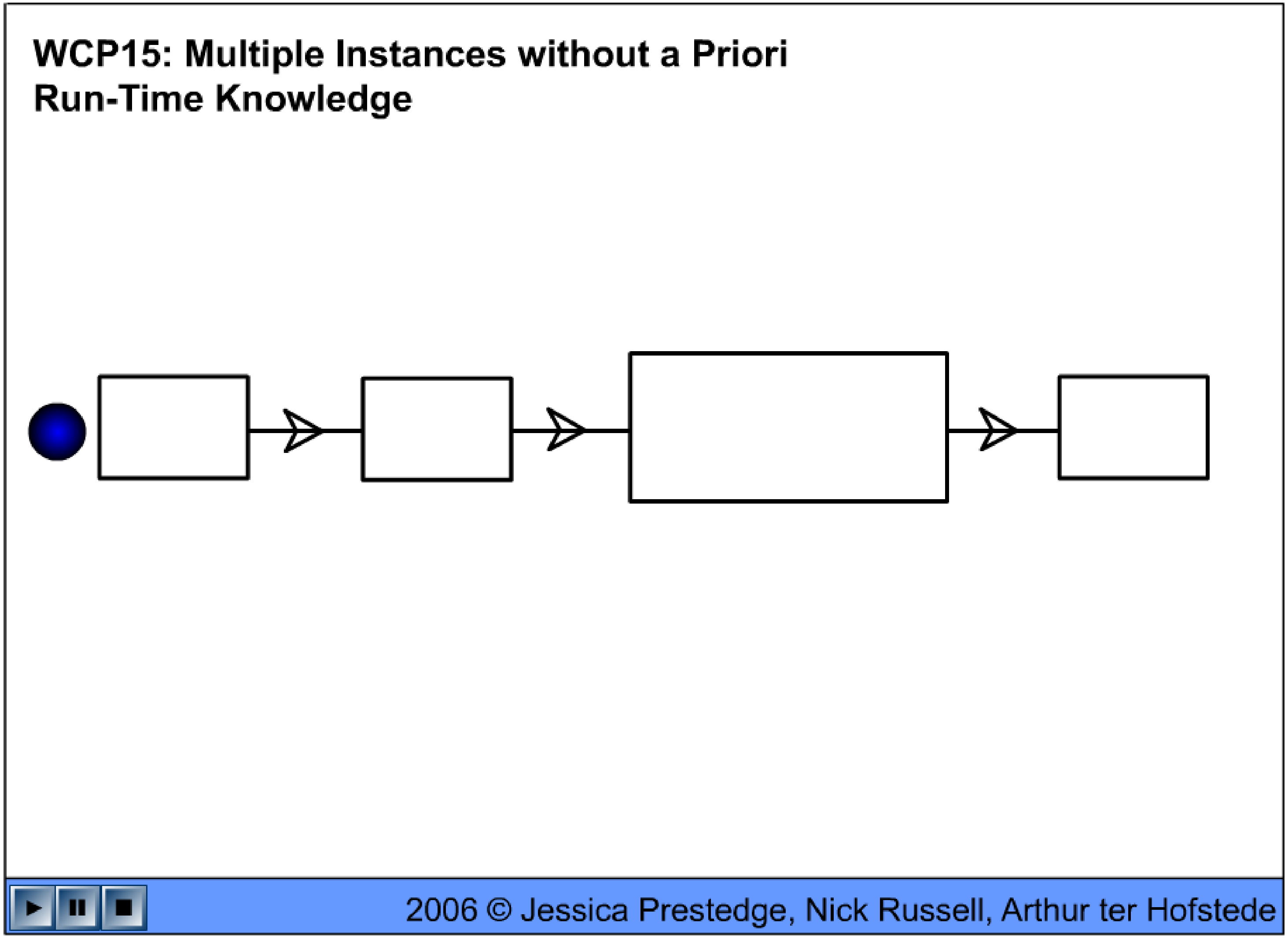
WCP38: General Synchronizing Merge

2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

Multiple Instance Tasks

- Sometimes multiple instances of the same task can be active within the same case
- Issues irt so-called MI-tasks are:
  - How to spawn them?
  - At what point do we know how many we would like to have?
    - *Design time, Runtime, Only when all have been created*
  - How do we synchronise these instances?
    - *Full vs Partial*
  - In case of partial synchronisation, what do we do with the remaining threads of execution?
- Limited support in UML and BPMN for the more advanced forms of synchronisation

MI without a Priori Run-Time Knowledge: Animation



WCP15: Multiple Instances without a Priori Run-Time Knowledge

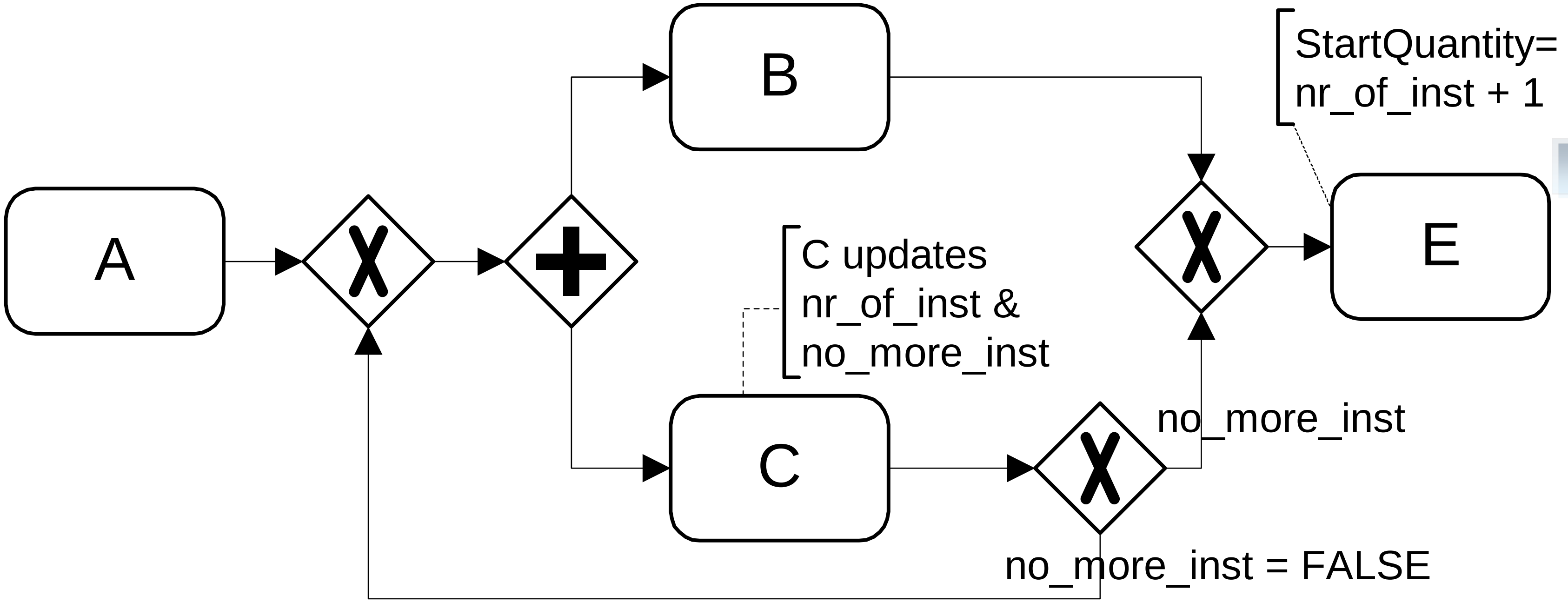
2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

MI without a Priori Run-Time Knowledge

Workaround in BPMN

In practice, a modeller has to implement this pattern.

The solution requires advanced data manipulation and special attribute settings.



Control-flow perspective: Evaluation

1 – BPMN 2 – UML AD 3 – BPEL

		1	2	3			1	2	3
Basic Control-flow					**Termination**				
1	Sequence	+	+	+	11	Implicit Termination	+	+	+
2	Parallel Split	+	+	+	43	Explicit Termination	+	+	-
3	Synchronisation	+	+	+	**Multiple Instances**				
4	Exclusive Choice	+	+	+	12	MI without Synchronisation	+	+	+
5	Simple Merge	+	+	+	13	MI with a priory Design Time Knlg	+	+	+
Advanced Synchronisation					14	MI with a priory Runtime Knlg	+	+	-
6	Multiple Choice	+	+	+	15	MI without a priory Runtime Knlg	-	-	-
7	Str. Synchronising Merge	+/-	-	+	27	Complete MI Activity	-	-	-
8	Multiple Merge	+	+	-	34	Static Partial Join for MI	+/-	-	-
9	Discriminator	+/-	+	-	35	Cancelling Partial Join for MI	+/-	-	-
28	Blocking Discriminator	+/-	+/-	-	36	Dynamic Partial Join for MI	-	-	-
29	Cancelling Discriminator	+	+	-	**State-based**				
30	Structured Partial Join	+/-	+/-	-	16	Deferred Choice	+	+	+
31	Blocking Partial Join	+/-	+/-	-	39	Critical Section	-	-	+
32	Cancelling Partial Join	+/-	+	-	17	Interleaved Parallel Routing	+/-	-	+/-
33	Generalised AND-Join	+	-	-	40	Interleaved Routing	+/-	-	+
37	Acyclic Synchronizing Merge	-	+/-	+	18	Milestone	-	-	-
38	General Synchronizing Merge	-	-	-	**Cancellation**				
41	Thread Merge	+	+	+/-	19	Cancel Activity	+	+	+
42	Thread Split	+	+	+/-	20	Cancel Case	+	+	+
Iteration					25	Cancel Region	+/-	+	-
10	Arbitrary Cycles	+	+	-	26	Cancel MI Activity	+	+	-
21	Structured Loop	+	+	+	**Trigger**				
22	Recursion	-	-	-	23	Transient Trigger	-	+	-
					24	Persistent Trigger	+	+	+

Data Pattern Categories

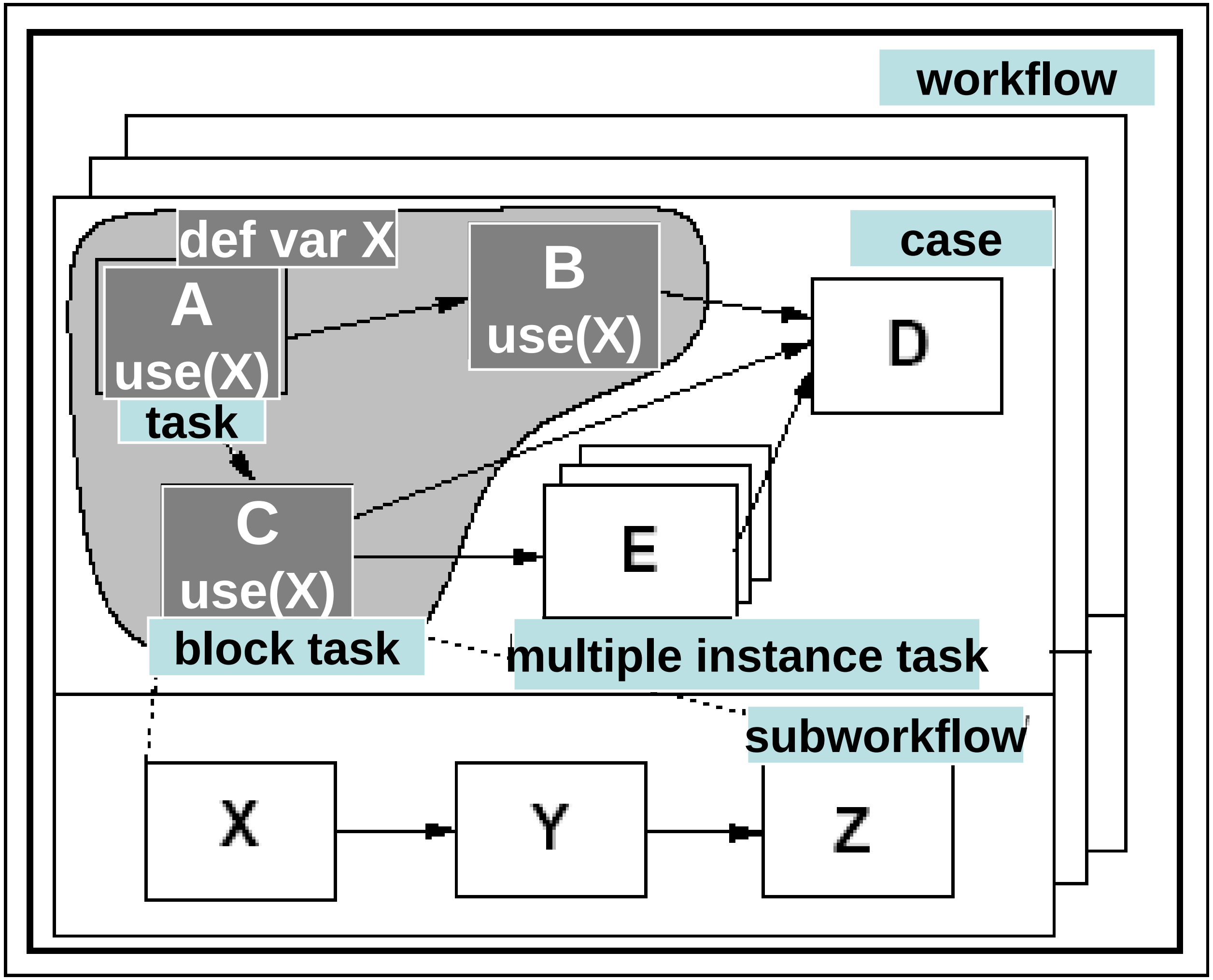
- **Data Visibility:** The extent and manner in which data elements can be viewed and utilised by workflow components.
- **Internal Data Interaction:** Data communication between active elements within a workflow.
- **External Data Interaction:** Data communication between active elements within a workflow and the external operating environment.
- **Data Transfer:** Data element transfer across the interface of a workflow component.
- **Data Routing:** The manner in which data elements can influence the operation of the workflow.

Data Visibility Patterns

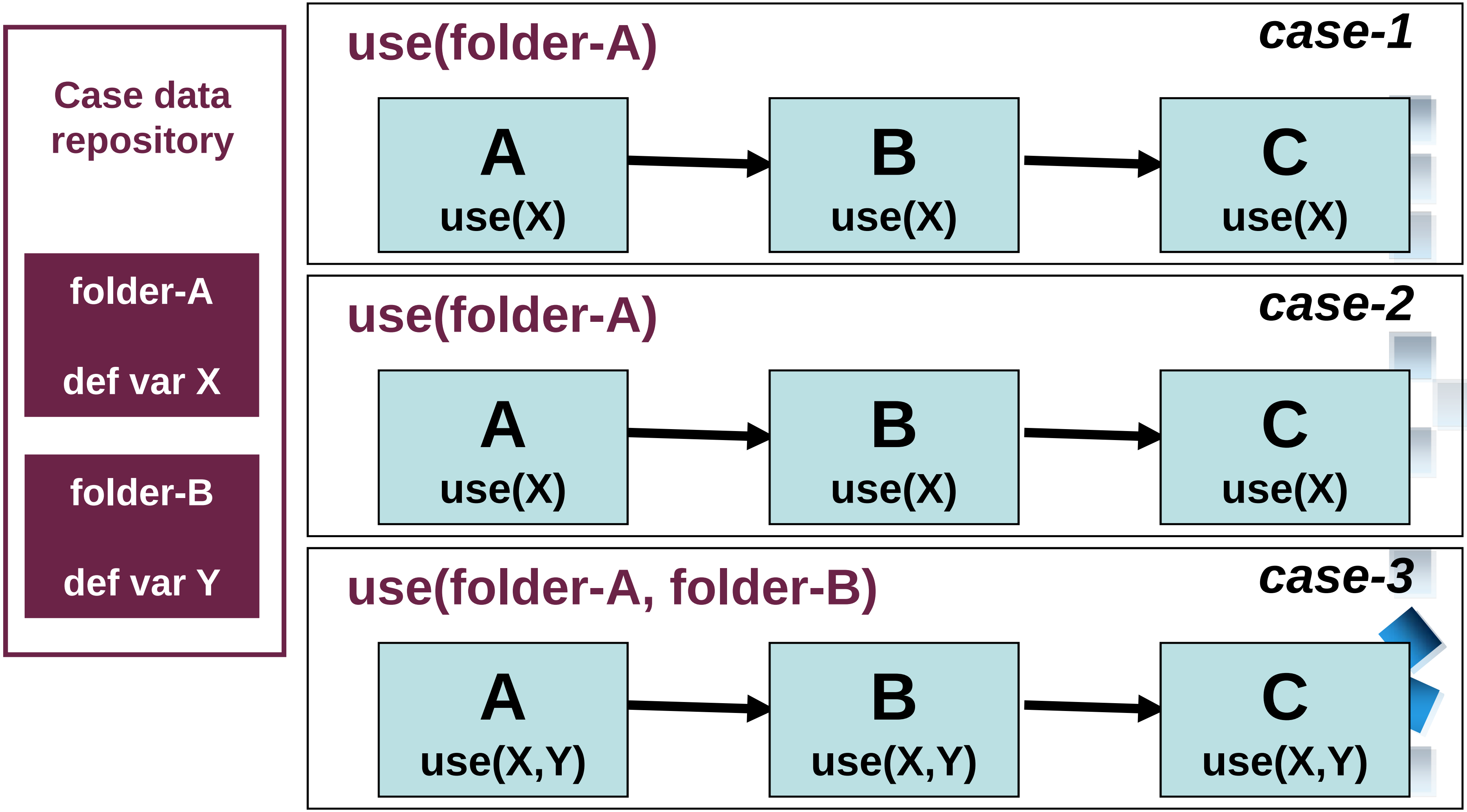
The extent and manner in which data elements can be viewed by workflow components

- Task data
- Block data
- Scope data
- Multiple Instance data
- Case data
- Folder data
- Workflow data
- Environment data

Scope Data


workflow
case
def var X
A
use(X)
task
B
use(X)
C
use(X)
block task
D
E
multiple instance task
subworkflow
X
Y
Z

Folder Data



Case data repository

folder-A

def var X

folder-B

def var Y

use(folder-A) *case-1*

A use(X) → B use(X) → C use(X)

use(folder-A) *case-2*

A use(X) → B use(X) → C use(X)

use(folder-A, folder-B) *case-3*

A use(X,Y) → B use(X,Y) → C use(X,Y)

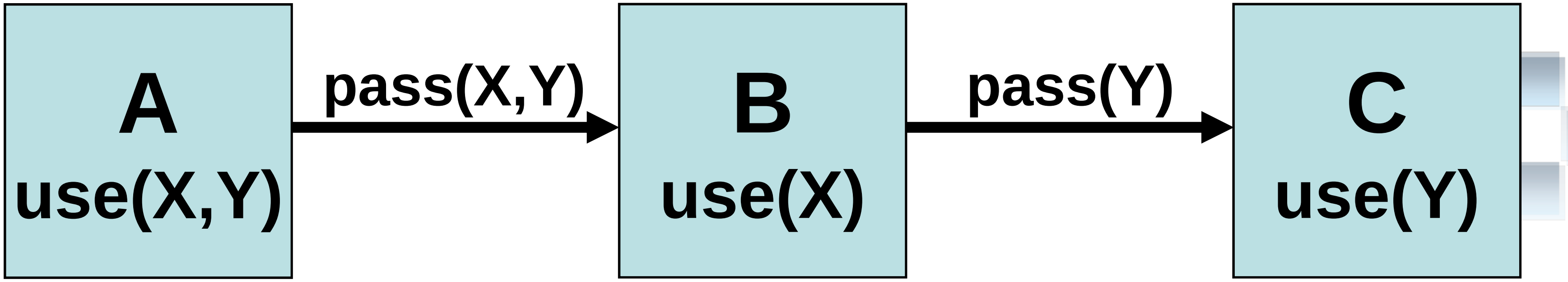
Data Interaction Patterns: Internal

Data communication between active elements within a workflow.

- Data interaction between tasks
- Block task to sub-workflow decomposition
- Sub-workflow decomposition to block task
- To multiple instance task
- From multiple instance task
- Case to case

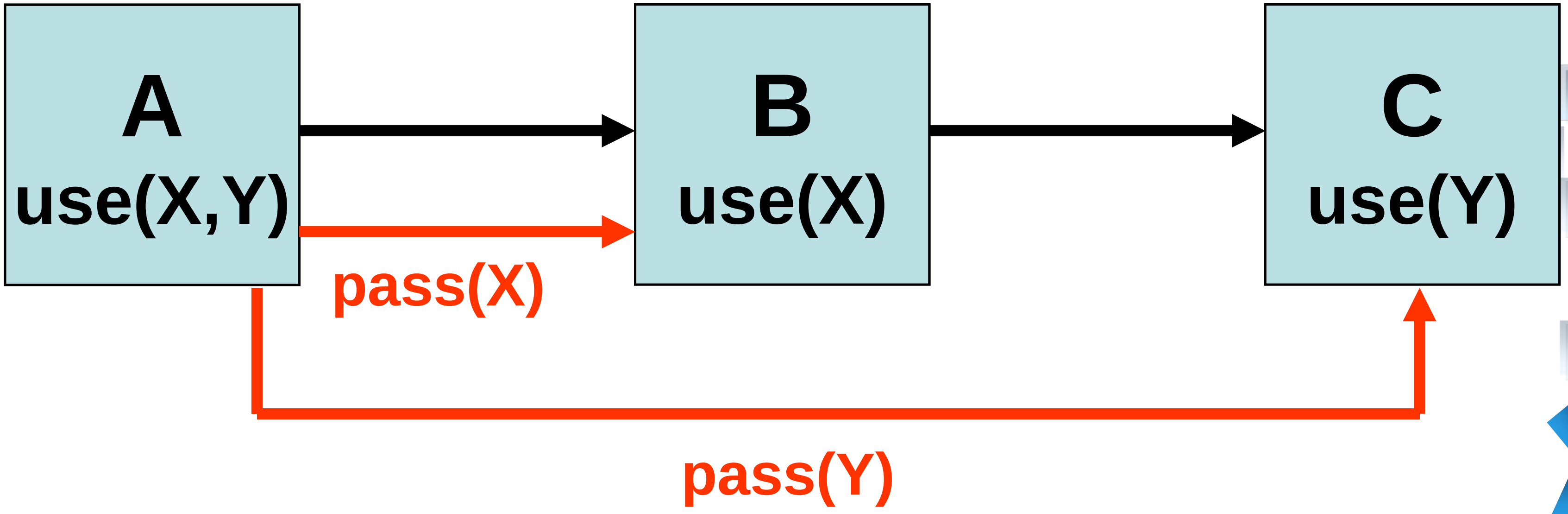
Data Interaction between Tasks

Integrated Control and Data Channels



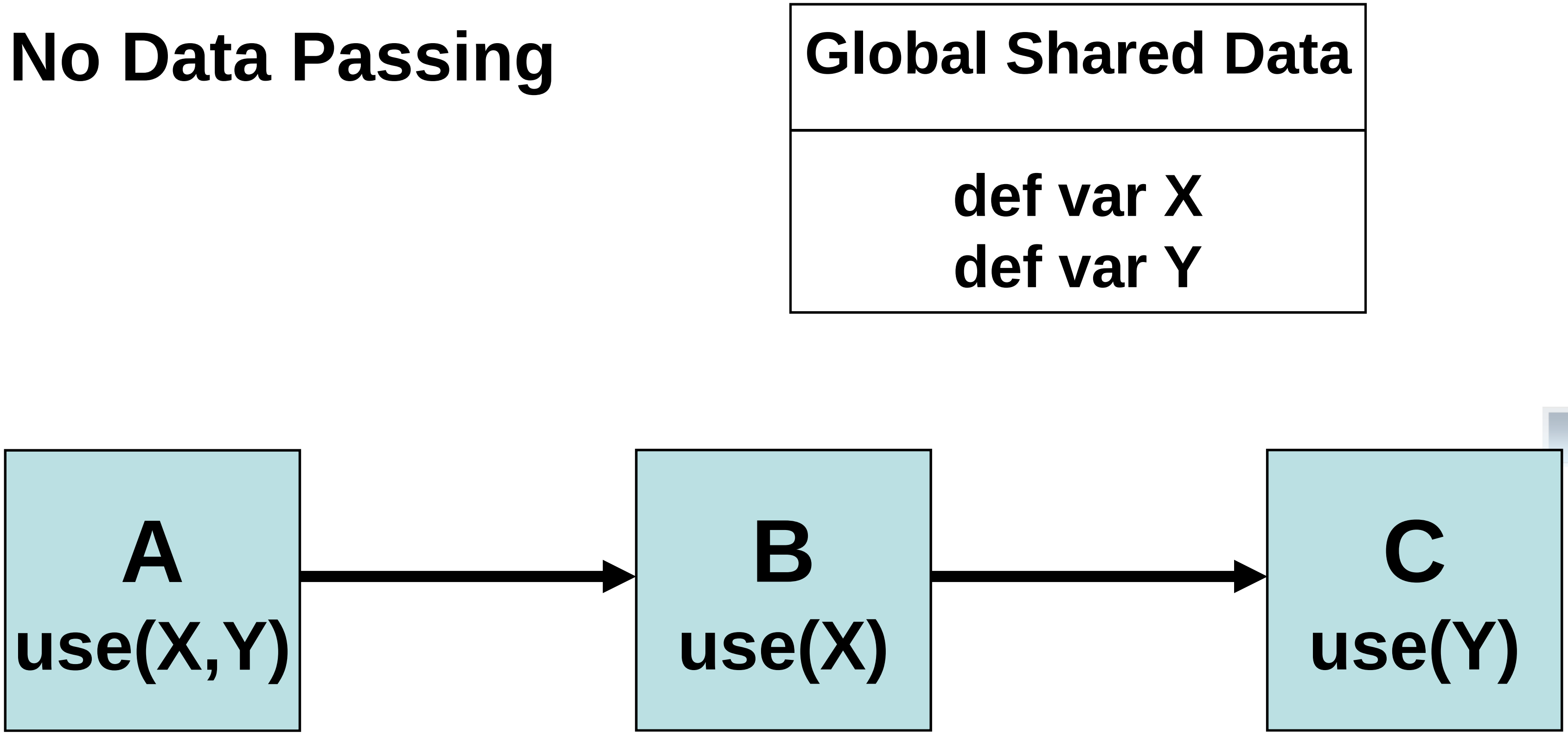
Data Interaction between Tasks

Distinct Control and Data Channels



Data Interaction between Tasks

No Data Passing

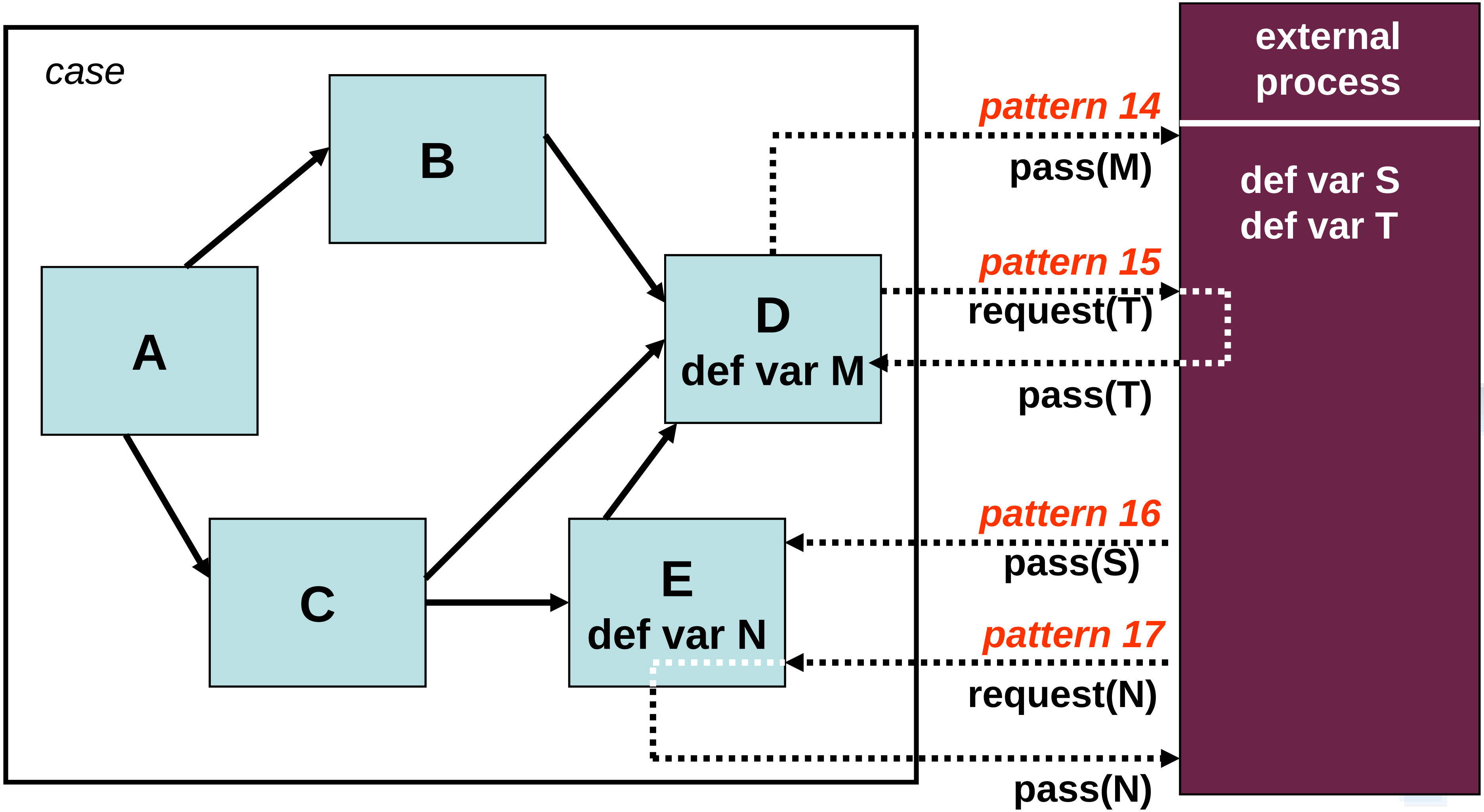


Data Interaction Patterns: External

Data communication between active elements within a workflow and the external operating environment.

- Task to Environment – Push-Oriented
- Environment to Task – Pull-Oriented
- Environment to Task – Push-Oriented
- Task to Environment – Pull-Oriented
- Case to Environment – Push-Oriented
- Environment to Case – Pull-Oriented
- Environment to Case – Push-Oriented
- Case to Environment – Pull-Oriented
- Workflow to Environment – Push-Oriented
- Environment to Workflow – Pull-Oriented
- Environment to Workflow – Push-Oriented
- Workflow to Environment – Pull-Oriented

Data Interaction - Task to/from Environment



Data Transfer Patterns

Focus on the means by which a data element is transferred across the interface of a workflow component.

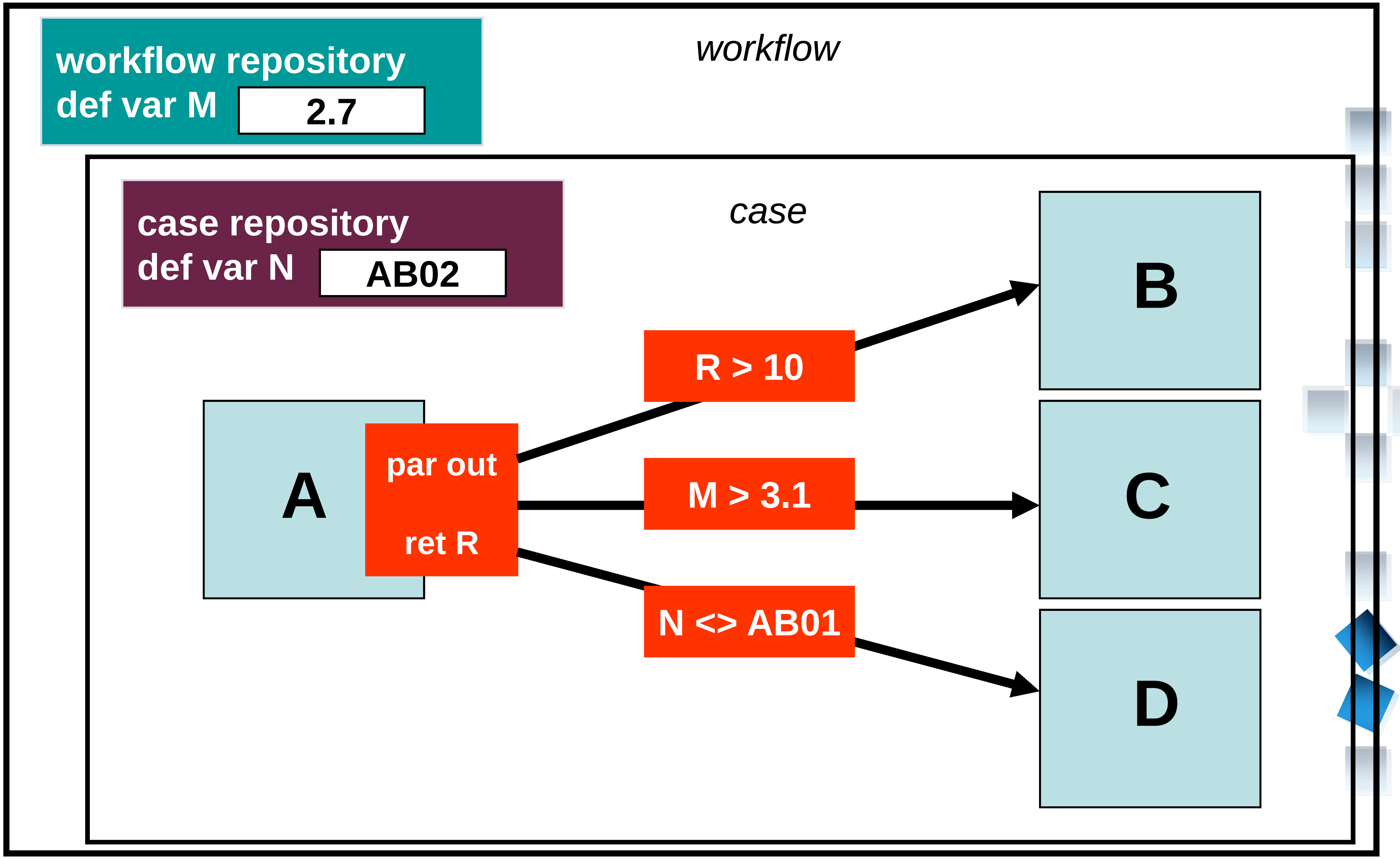
- Data Transfer by Value – Incoming
- Data Transfer by Value - Outgoing
- Data Transfer – Copy In/Copy Out
- Data Transfer by Reference – Unlocked
- Data Transfer by Reference – Locked
- Data Transformation – Input
- Data Transformation - Output

Data Routing Patterns

Focus on the manner in which data elements can influence the operation of the workflow.

- Task Precondition - Data Existence
- Task Precondition - Data Value
- Task Postcondition – Data Existence
- Task Postcondition - Data Value
- Event-based Task Trigger
- Data-based Task Trigger
- Data-based Routing

Data-based Routing



Data Perspective: Evaluation

1 – BPMN 2 – UML AD 3 – BPEL

		1	2	3			1	2	3
Data Visibility					**Data Interaction (External), cont.**				
1	Task Data	+	+/-	+/-	21	Env. to Case - Push-Oriented	-	-	-
2	Block Data	+	+	-	22	Case to Env. - Pull-Oriented	-	-	-
3	Scope Data	-	-	+	23	Workflow to Env. - Push-Oriented	-	-	-
4	MI Data	+/-	+	-	24	Env. to Workflow - Pull-Oriented	-	-	-
5	Case Data	+	-	+	25	Env. to Workflow - Push-Oriented	-	-	-
6	Folder Data	-	-	-	26	Workflow to Env. - Pull-Oriented	-	-	-
7	Workflow Data	-	+	-	**Data Transfer**				
8	Environment Data	-	-	+	27	by Value - Incoming	+	-	+
Data Interaction (Internal)					28	by Value - Outgoing	+	-	+
9	between Tasks	+	+	+	29	Copy In/Copy Out	+/-	-	-
10	Task to Sub-workflow Decomp.	+	+	-	30	by Reference - Unlocked	-	-	+
11	Sub-workflow Decomp. to Task	+	+	-	31	by Reference - Locked	+	+	+/-
12	to MI Task	-	+	-	32	Data Transformation - Input	+/-	+	-
13	from MI Task	-	+	-	33	Data Transformation - Output	+/-	+	-
14	Case to Case	-	-	+/-	**Data-based Routing**				
Data Interaction (External)					34	Task Precondition Data Exist.	+	+	+/-
15	Task to Env - Push-Oriented	+	-	+	35	Task Precondition Data Value	-	+	+
16	Env. to Task - Pull-Oriented	+	-	+	36	Task Postcondition Data Exist.	+	+	-
17	Env. to Task - Push-Oriented	+	-	+/-	37	Task Postcondition Data Value	-	+	-
18	Task to Env - Pull-Oriented	+	-	+/-	38	Event-based Task Trigger	+	+	+
19	Case to Env. - Push-Oriented	-	-	-	39	Data-based Task Trigger	+	-	+/-
20	Env. to Case - Pull-Oriented	-	-	-	40	Data-based Routing	+	+	+

General Findings

- Workflow data patterns generalise to all forms of PAIS
  - All patterns observed in surveyed tools;
  - Wide variations in support between tools.
- Data support for concurrent processes is limited:
  - Lack of concurrency control;
  - Minimal support for multiple instance tasks.
- Direct support for data patterns in current design tools is limited; and
- Current standards (e.g. XPDL, BPEL) do not provide useful guidance of data usage in workflow.

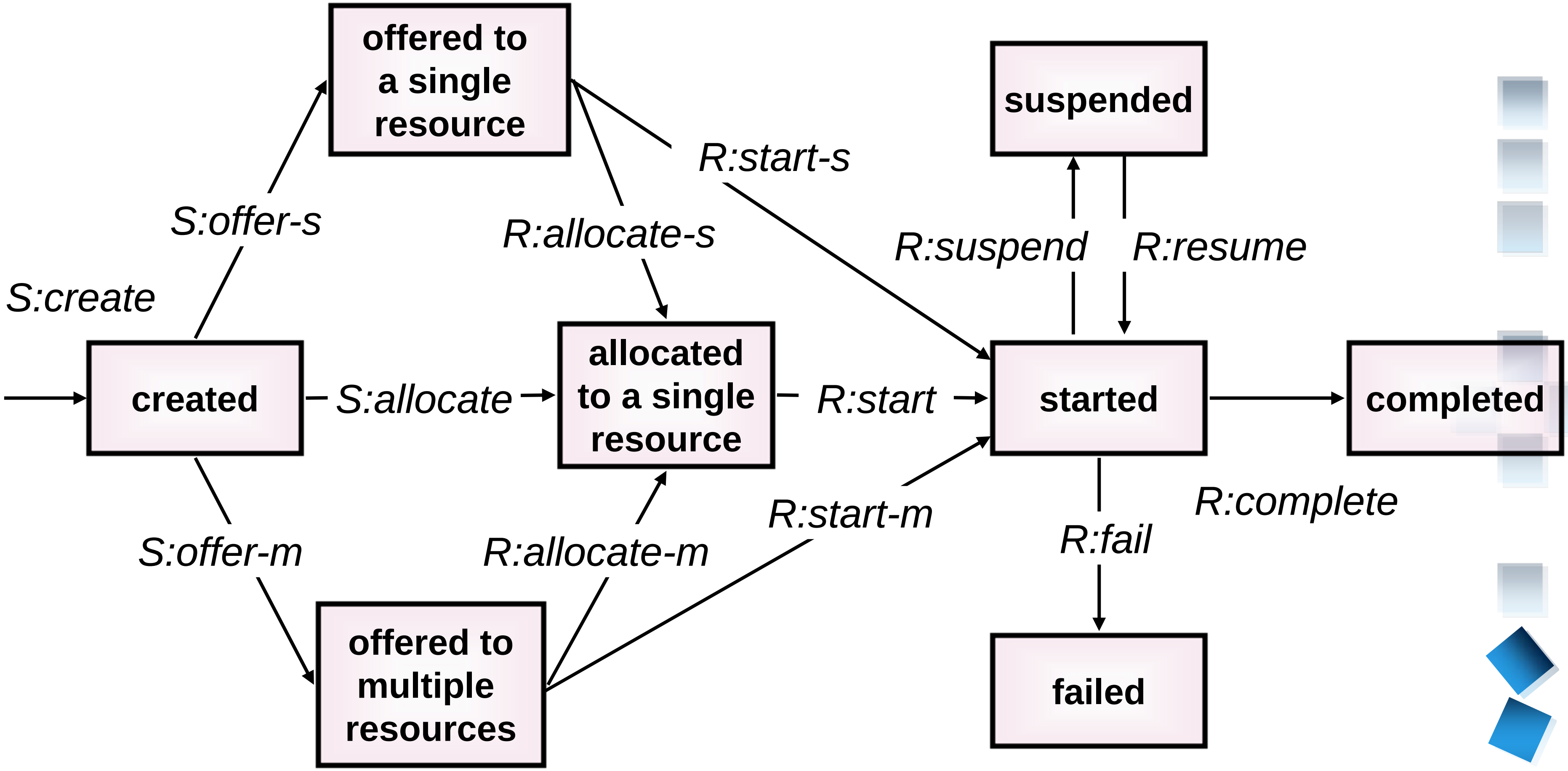
Workflow Resource Patterns

- 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

Resource Patterns Classes

- **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

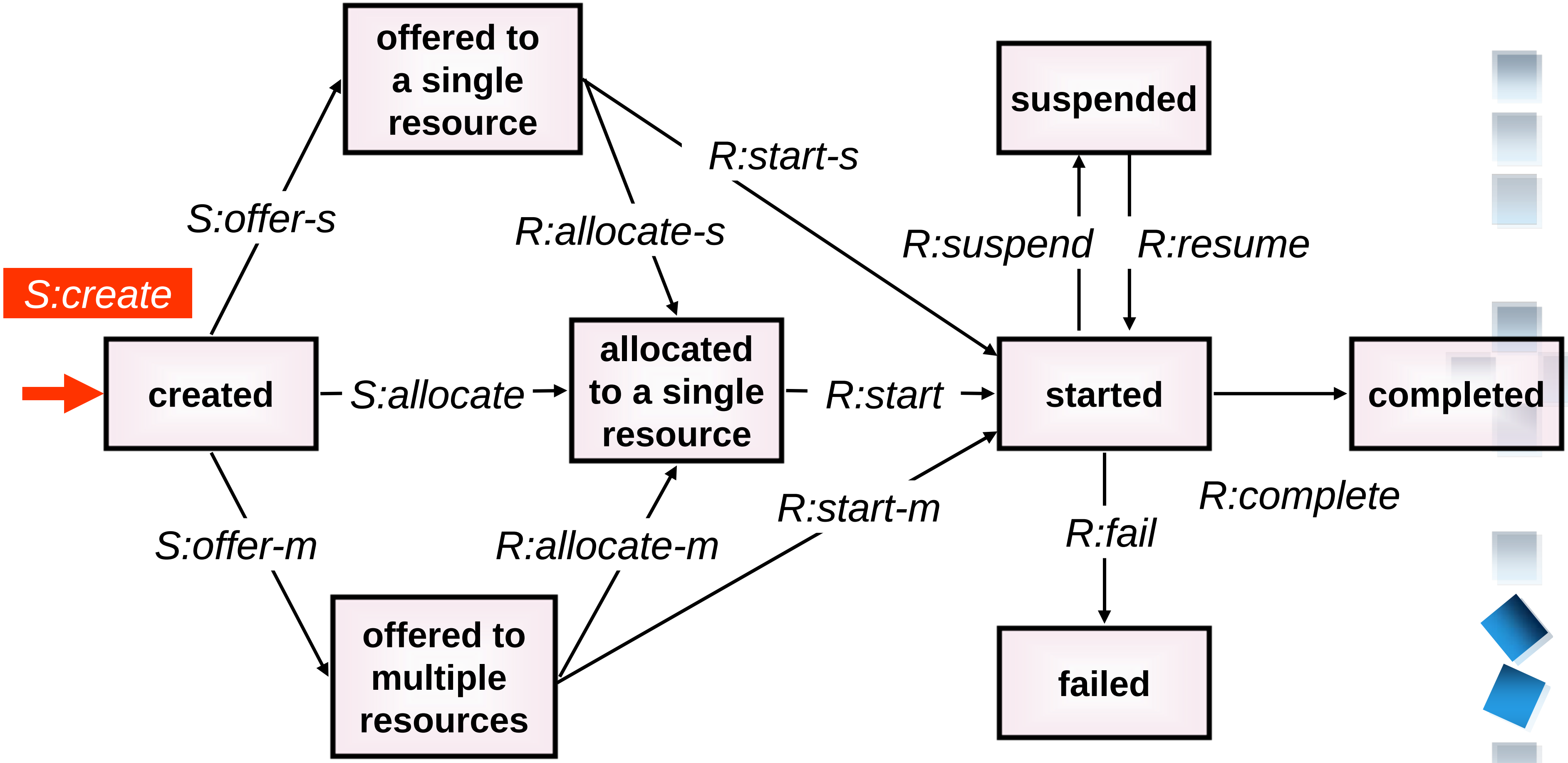


Creation Patterns

Design time considerations relating to which resources may execute a work item at runtime

- Direct Allocation
- Role-Based Allocation
- Deferred Allocation
- Authorisation
- Separation of Duties
- Case Handling
- Retain Familiar
- Capability-Based Allocation
- History-Based Allocation
- Organisational Allocation
- Automatic Execution

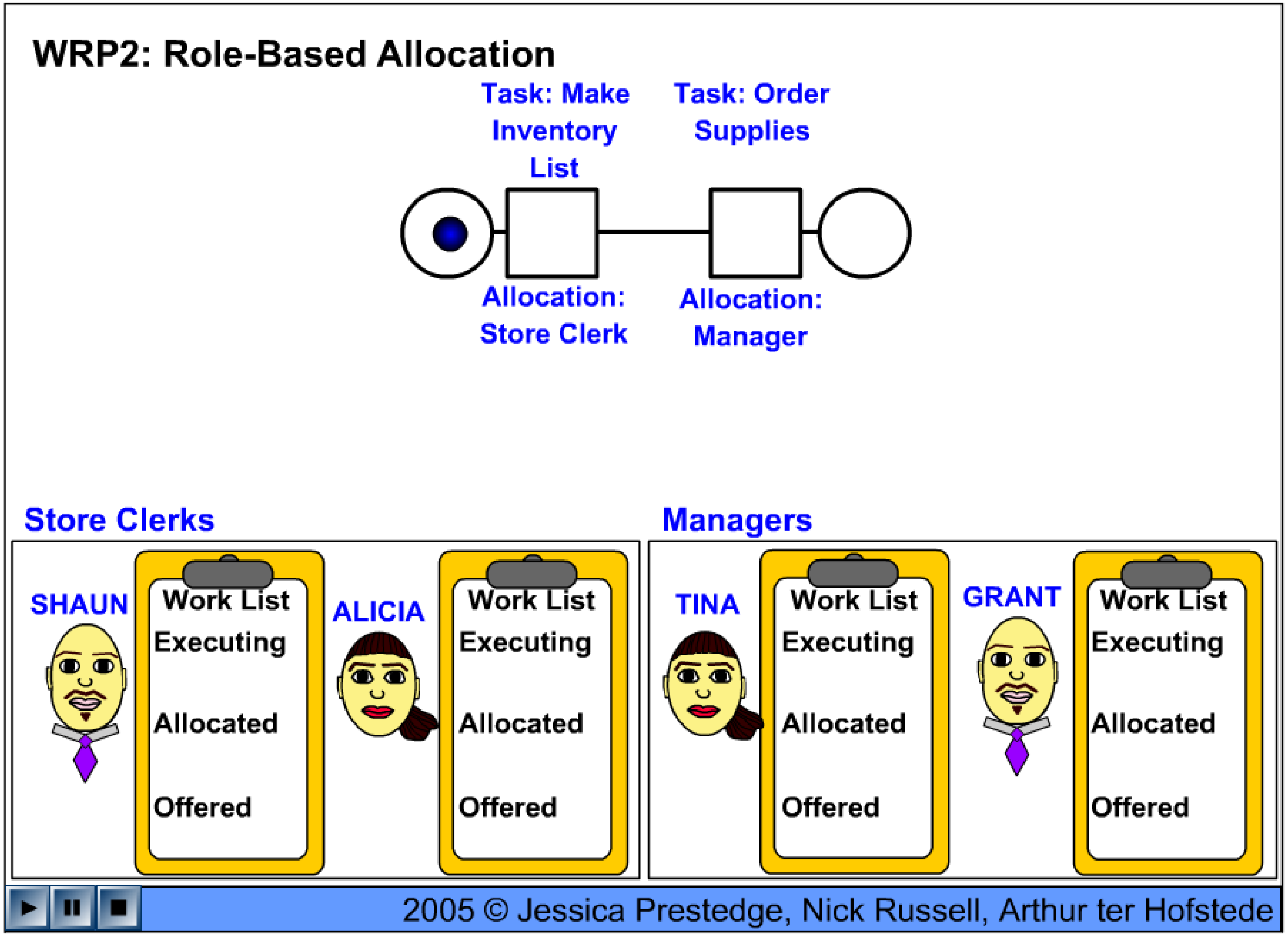
Creation Patterns



Role-based Allocation

- The ability to specify at design time that a task can only be executed by resources which correspond to a given role.
- Actual decision for distribution deferred till runtime, can be influenced without changing workflow specification (thus providing more flexibility)

Role-based Allocation: Animation



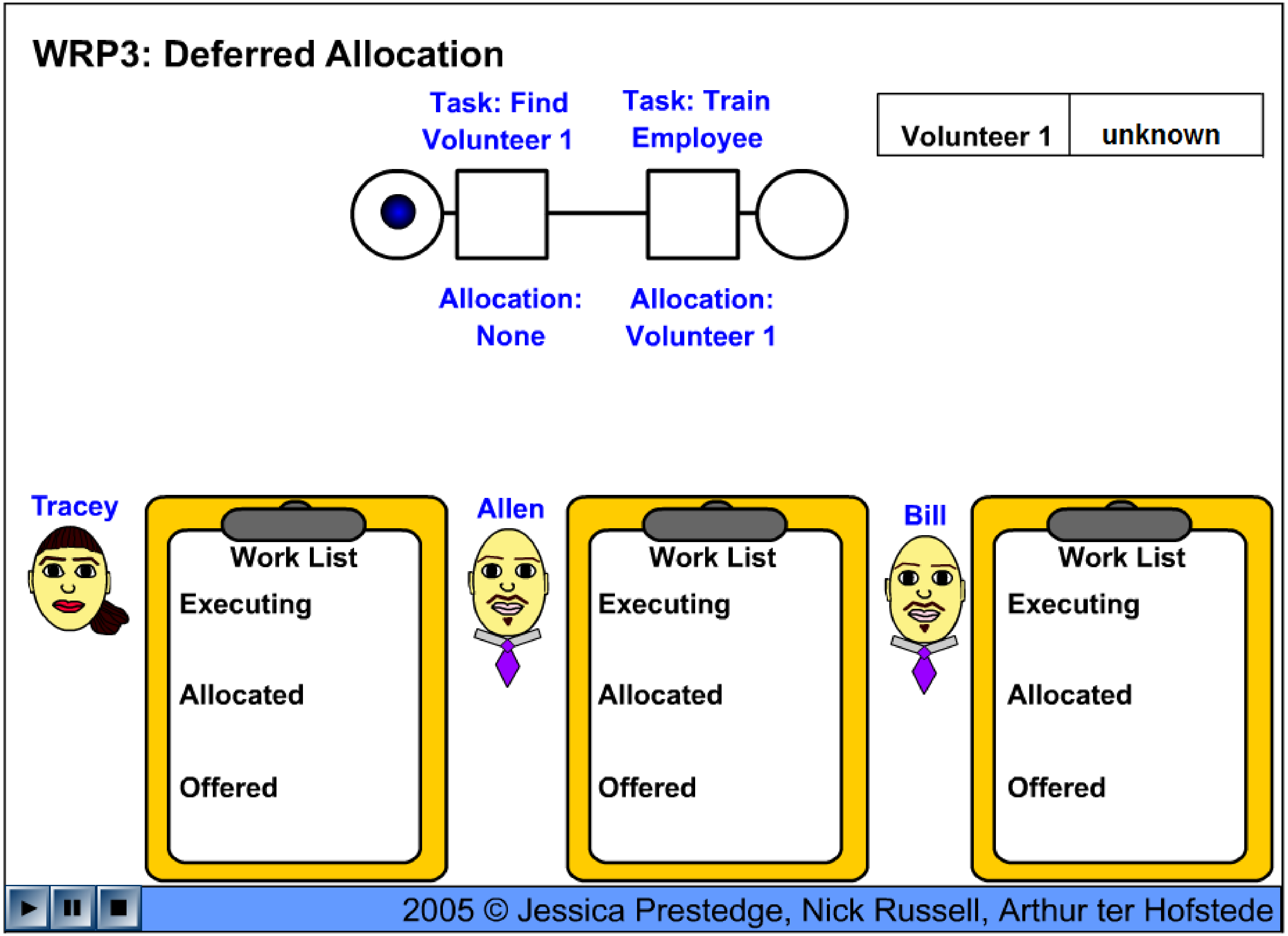
WRP2: Role-Based Allocation

2005 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

Deferred Allocation

- The ability to defer specifying the identity of the resource that will execute a task until runtime
- Takes deferral of resource allocation one step further
- Can be achieved through a variable that contains actual resource(s) to be used

Deferred Allocation: Animation



WRP3: Deferred Allocation

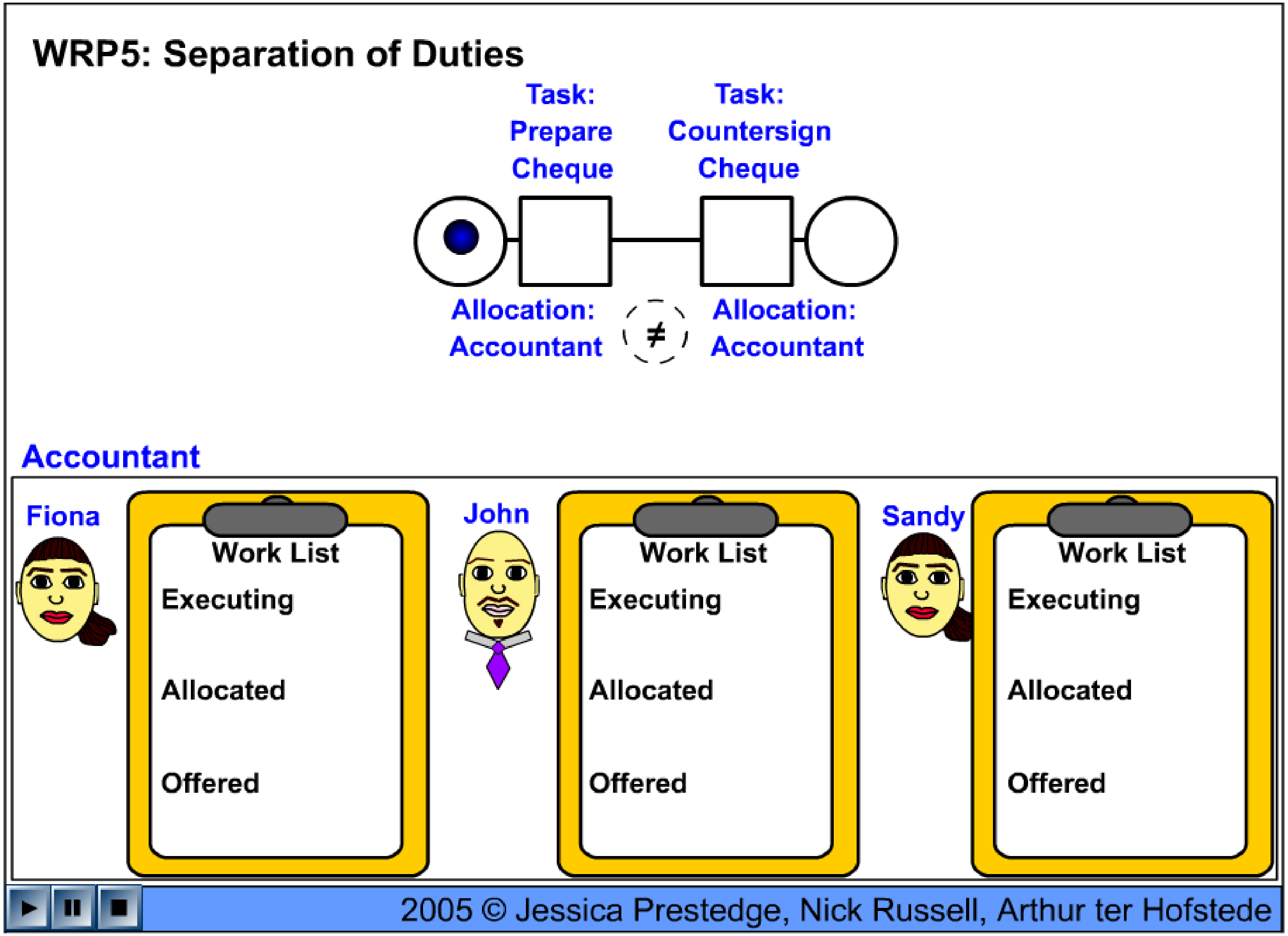
Volunteer 1	unknown

2005 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

Separation of Duties

- The ability to specify that two tasks must be allocated to different resources in a given workflow case.
- Also referred to as “4 eyes principle”

Separation of Duties: Animation



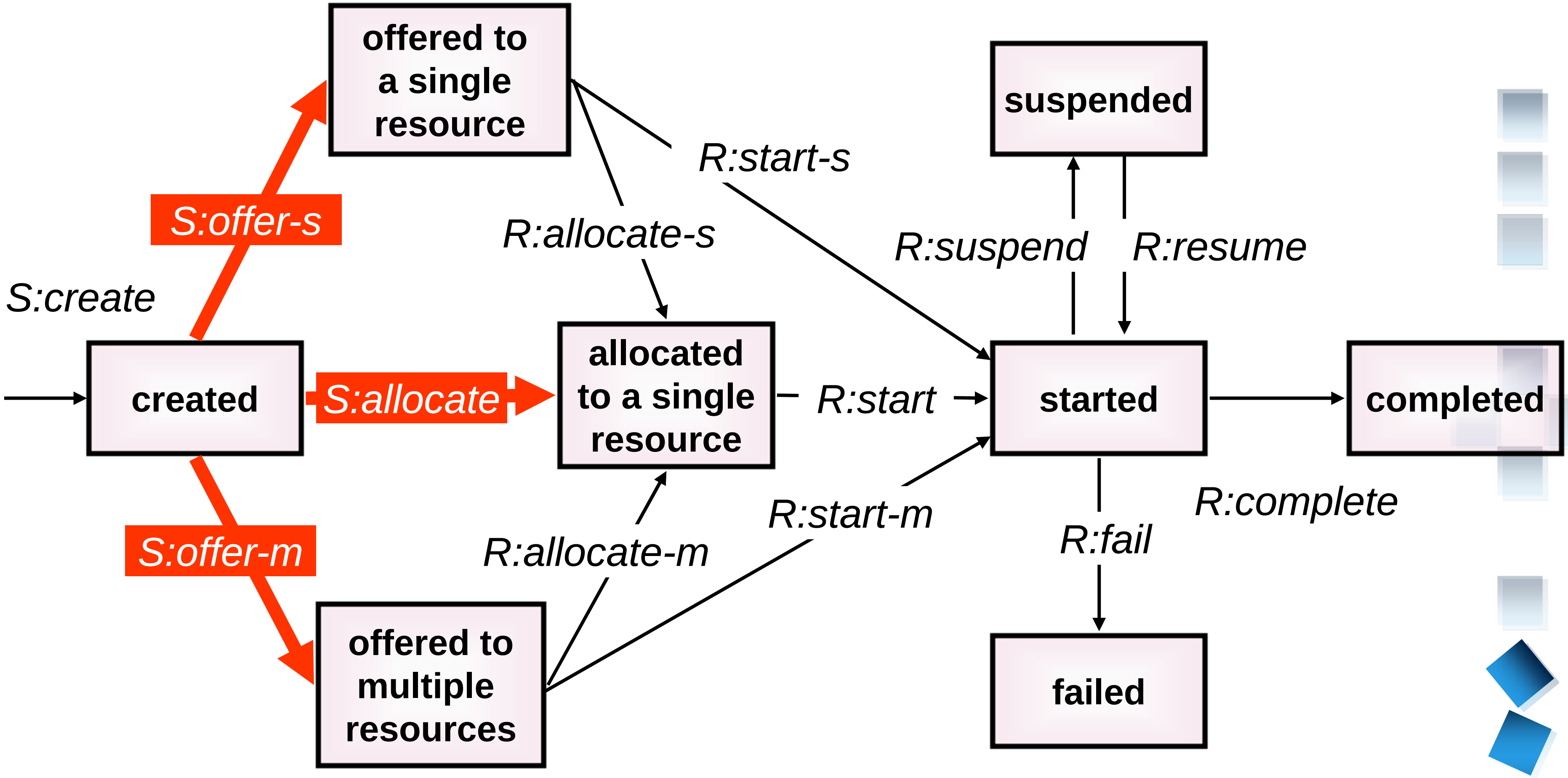
WRP5: Separation of Duties

2005 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

Push Patterns

- Correspond to situations where newly created work items are proactively routed to resources by the workflow system
- Key dimensions:
  - Offer or allocation
  - Single or multiple resources
  - Basis of allocation
  - Timing of distribution vs enablement

Push Patterns



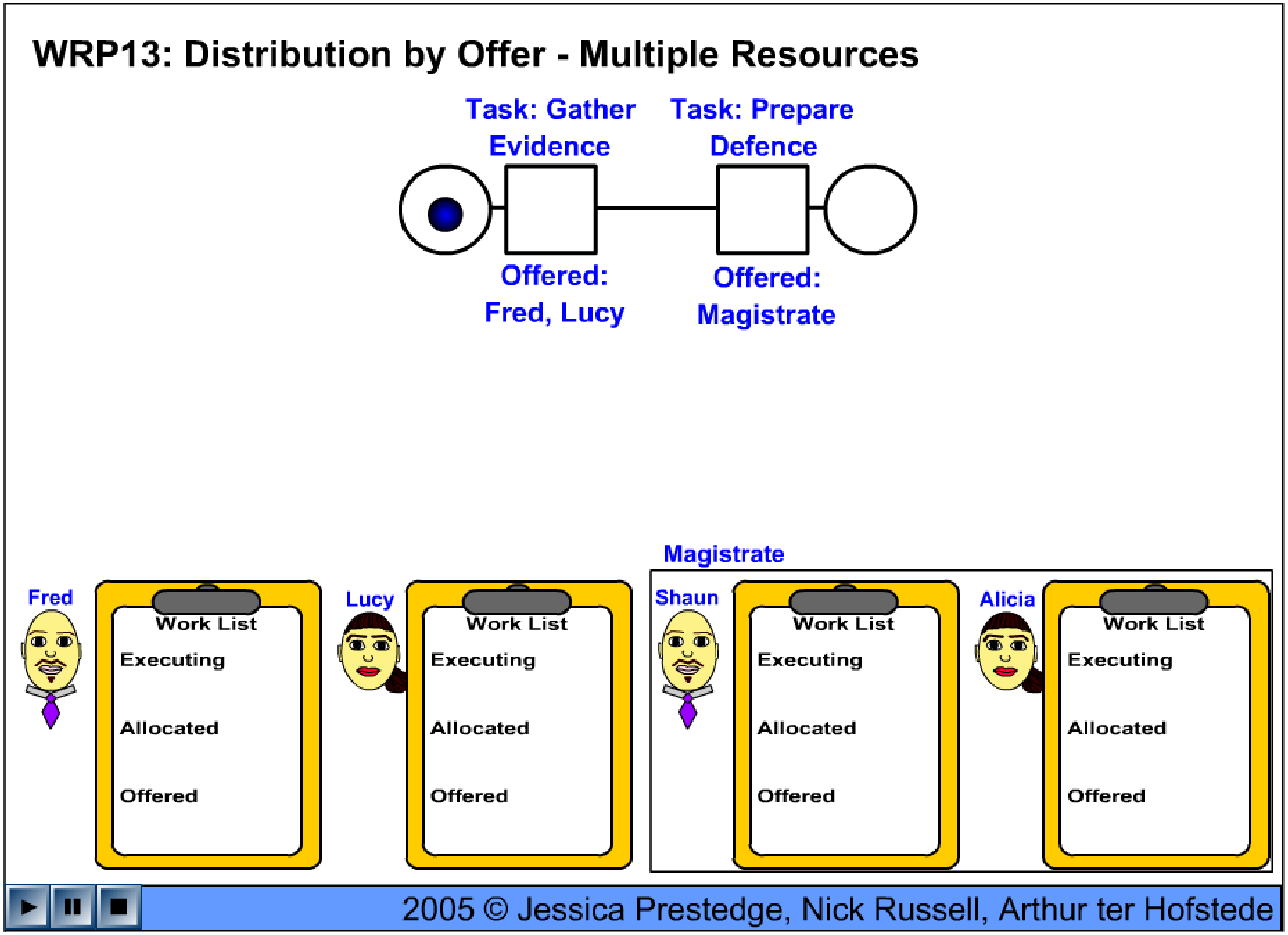
Push Patterns

- Distribution by Offer - Single Resource
- Distribution by Offer - Multiple Resources
- Distribution by Allocation - Single Resource
- Random Allocation
- Round Robin Allocation
- Shortest Queue
- Early Distribution
- Distribution on Enablement
- Late Distribution

Distribution by Offer – Multiple Resources

- The ability to offer a work item to a group of selected resources.
- Offering a work item to multiple resources is the workflow analogy to the act of "calling for a volunteer" in real life. It provides a means of advising a suitably qualified group of resources that a work item exists but leaves the onus with them as to who actually commits to undertaking the activity.
- Can be realised through so-called *work groups*

Distribution by Offer – Multiple Resources: Animation



WRP13: Distribution by Offer - Multiple Resources

Task: Gather Evidence

Task: Prepare Defence

Offered: Fred, Lucy

Offered: Magistrate

Magistrate

Fred

Work List
Executing
Allocated
Offered

Lucy

Work List
Executing
Allocated
Offered

Shaun

Work List
Executing
Allocated
Offered

Alicia

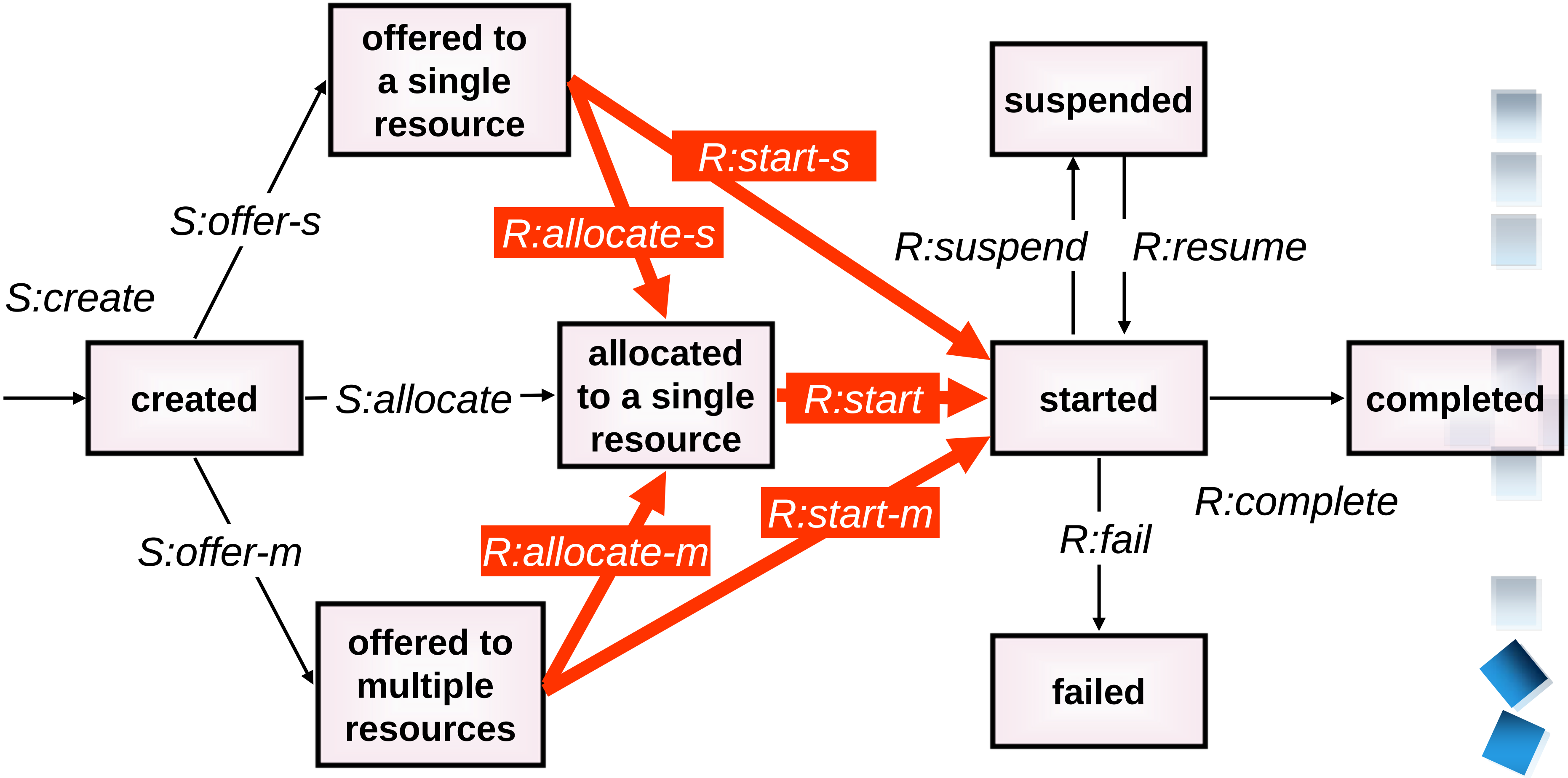
Work List
Executing
Allocated
Offered

2005 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

Pull Patterns

- Correspond to situations where a resource proactively seeks information on available work and commits to undertaking specific work items
- Key dimensions
  - Allocation vs execution
  - Configurability of work basket
  - Autonomy in selecting next work item

Pull Patterns



Pull Patterns

- Resource-Initiated Allocation
- Resource-Initiated Execution – Allocated Work Item
- Resource-Initiated Execution – Offered Work Item
- System-Determined Work List Management
- Resource-Determined Work List Management
- Selection Autonomy

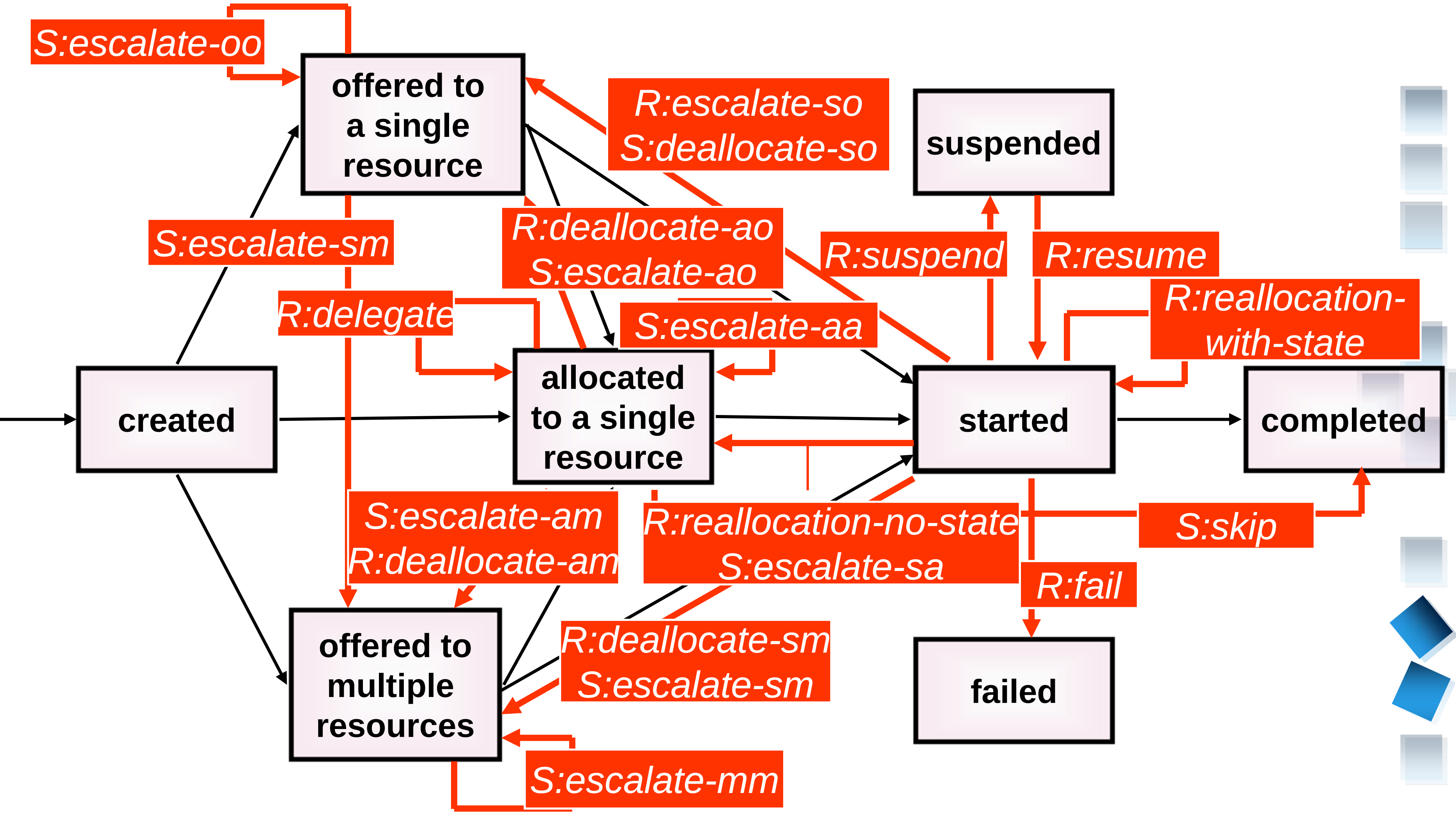
Resource-Initiated Allocation

- The ability for a resource to commit to undertake a work item without needing to commence working on it immediately.
- Provides a means for a resource to signal its intention to execute a given work item at some point although it may not commence working on it immediately.
- As a consequence the work item is considered to be allocated to the resource and it cannot be allocated to or executed by another resource.
- Two variants depending on whether the work item was offered to a single resource or to multiple

Detour Patterns

- Correspond to unplanned variations in work item routing
- Key dimensions
  - Initiator of re-routing action – system or resource
  - Execution state of work item
  - Recipient of re-routed work item

Detour Patterns



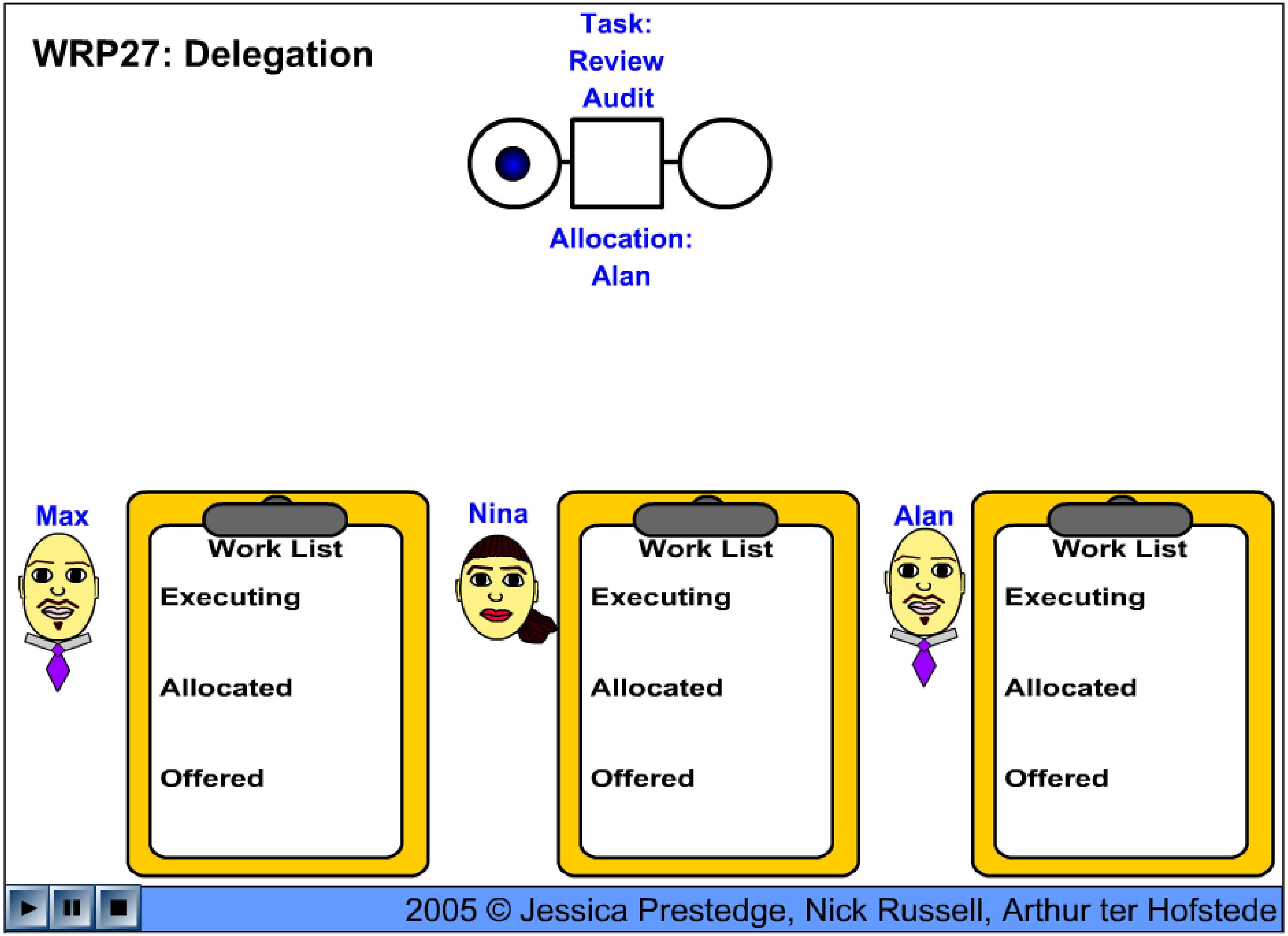
Detour Patterns

- Delegation
- Escalation
- Deallocation
- Stateful Reallocation
- Stateless Reallocation
- Suspension/Resumption
- Skip

Delegation

- The ability for a resource to allocate a work item previously allocated to it to another resource.
- Delegation provides a resource with a means of re-routing work items that it is unable to execute. This may be because the resource is unavailable (e.g. on vacation) or because they do not wish to take on any more work.
- What happens where a work item is delegated to a user who is not authorised to execute it?
  - This scenario is only a problem for workflow engines that support distinct task routing and authorisation mechanisms.
  - COSA’s solution is to allow the new user to see the work item but not to be able to delegate it, they can then delegate themselves or acquire necessay rights

Delegation: Animation



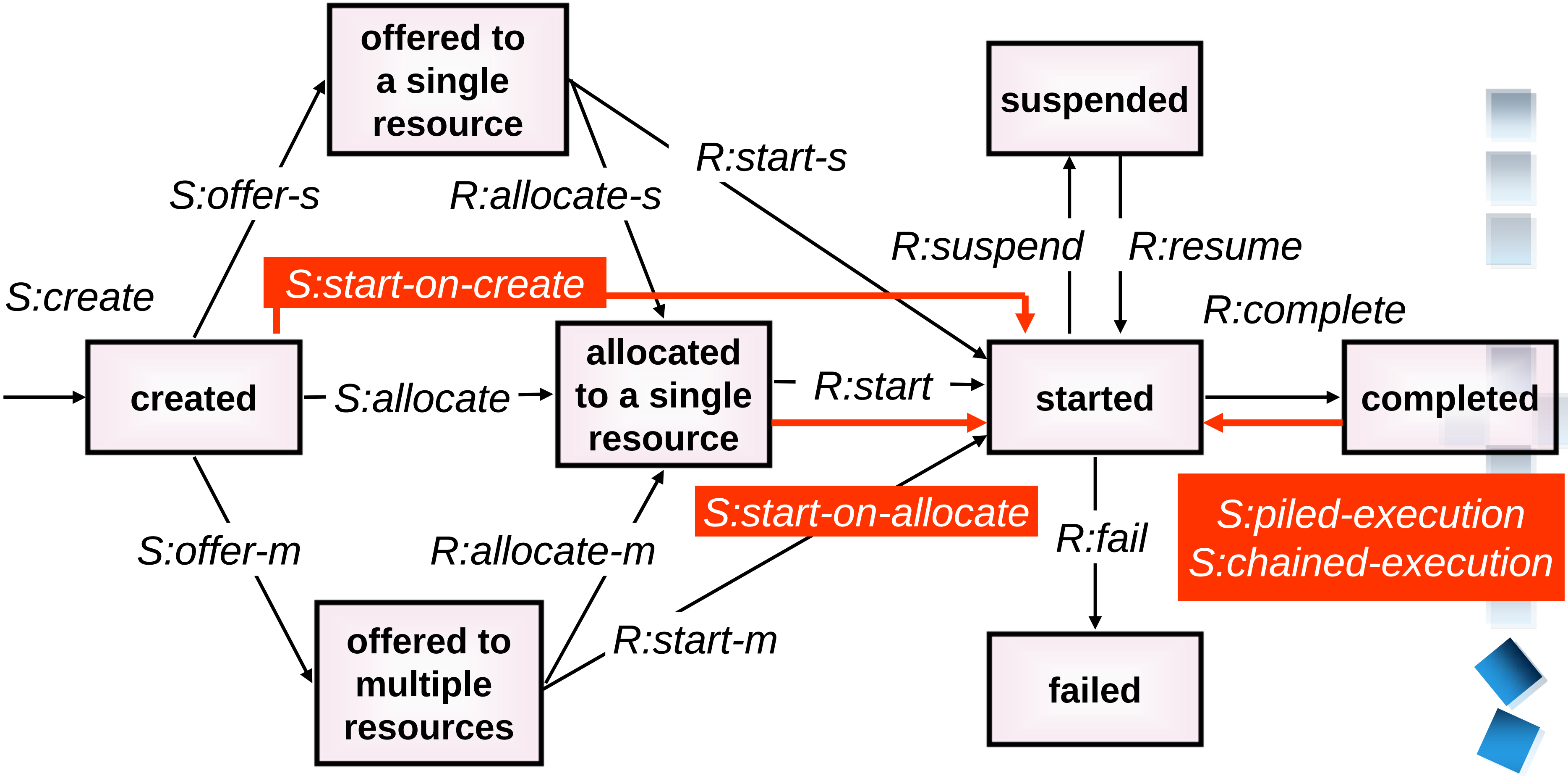
WRP27: Delegation

2005 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

Auto-start Patterns

- Relate to situations where execution of a work item is triggered by specific events in the lifecycle of a work item or related work items:
- e.g.
  - Creation
  - Allocation
  - Completion of preceding tasks
  - Completion of another instance of the same task

Auto-start Patterns



Auto-start Patterns

- Commence on Creation
- Commence on Allocation
- Piled execution
- Chained Execution

Resource perspective: Evaluation

1 – BPMN   2 – UML AD,   3 – Oracle BPEL PM (from [Mulyar 2005])

		1	2	3
Creation Patterns				
1	Direct Allocation	+	+	+
2	Role-Based Allocation	+	+	+
3	Deferredc 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	Distritubtion by Offer-Single Resource	-	-	+
13	Distritubtion by Offer-Multiple Resources	-	-	+
14	Distritubtion by Allocation-Single Resource	+	+	+
15	Random Allocation	-	-	+/-
16	Round Robin Allocation	-	-	+/-
17	Shortest Queue	-	-	+/-
18	Early Distribution	-	-	-
19	Distribution on Enablement	+	+	+
20	Lata Distribution	-	-	-
Pull Patterns				
21	Resource-Init. Allocation	-	-	-
22	Resource-Init. Exec. - Allocated WI	-	-	+
23	Resource-Init. Exec. - Offered WI	-	-	+

		1	2	3
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	-	-	+
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 Resources	-	-	+

Epilogue - I

- ***Patterns***
  - Provide an effective foundation for training workflow designers and developers;
  - Present a means of assessing and comparing tool capabilities and are particularly useful in tool evaluation and selection exercises (e.g. tender evaluations);
  - Offer the basis for vendors to identify functionality gaps and potential areas for enhancement.

Epilogue - II

- Patterns range from simple to (very) complex
- Patterns typically observed
- Comprehensive support lacking
- Problems so complex that informal approaches fall short

QUESTIONS?

References I

- **www.workflowpatterns.com**
- W.M.P. van der Aalst. Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language. QUT Technical report, FIT-TR-2003-06, Queensland University of Technology, Brisbane, 2003.
- W.M.P. van der Aalst. Don't go with the flow: Web services composition standards exposed. Web Services - Been there done that?, Trends & Controversies. IEEE Intelligent Systems 18(1):72-76.
- Wil M.P. van der Aalst and Kees M. van Hee. Workflow Management: Models, Methods, and Systems. The MIT Press, 2002.
- W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. Distributed and Parallel Databases, 14(3), pages 5-51, July 2003.
- W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Advanced Workflow Patterns. In O. Etzion and P. Scheuermann, editors, 7th International Conference on Cooperative Information Systems (CoopIS 2000), volume 1901 of Lecture Notes in Computer Science, pages 18-29. Springer-Verlag, Berlin, 2000.
- W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Web Service Composition Languages: Old Wine in New Bottles? In G. Chroust and C. Hofer, editors, Proceedings of the 29th EUROMICRO Conference: New Waves in System Architecture, pages 298-305. IEEE Computer Society, Los Alamitos, CA, 2003.
- W.M.P. van der Aalst, M. Dumas, A H.M. ter Hofstede, N. Russell, H. M.W Verbeek, and P. Wohed. Life After BPEL? In Proceedings of the 2nd International Workshop on Web Services and Formal Methods (WS-FM). Versailles, France, September 2005. Springer Verlag.

References II

- M. Dumas and A. ter Hofstede. UML Activity Diagrams as a Workflow Specification Language. In Proceedings of the International Conference on the Unified Modeling Language (UML), Toronto, Canada, October 2001. Springer Verlag.
- Marlon Dumas, Wil M.P. van der Aalst, Arthur H.M. ter Hofstede (Editors), Process-Aware Information Systems: Bridging People and Software Through Process Technology, John Wiley & Sons, September 2005.
- B. Kiepuszewski, A.H.M. ter Hofstede, C. Bussler. On Structured Workflow Modelling. Proceedings CAiSE’2000, Lecture Notes in Computer Science 1789, Stockholm, Sweden, June 2000.
- B. Kiepuszewski. Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2003.
- B. Kiepuszewski, A.H.M. ter Hofstede and W.M.P. van der Aalst. Fundamentals of Control Flow in Workflows. Acta Informatica 39(3):143-209, 2003.
- N.Mulyar. Pattern-based Evaluation of Oracle-BPEL. BPM Center Report BPM-05-24, BPMcenter.org, 2005.
- N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22 , BPMcenter.org, 2006.
- N. Russell, Wil M.P. van der Aalst , A.H.M. ter Hofstede , and Petia Wohed. On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling. In M. Stumptner, S. Hartmann, and Y. Kiyoki, editors, Proceedings of the Third Asia-Pacific Conference on Conceptual Modelling (APCCM2006), volume 53 of CRPIT, pages 95-104, Hobart, Australia, 2006. ACS.

References III

- N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns. QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, 2004.
- N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns: Identification, Representation and Tool Support. In L. Delcambre et al., editors, Proceedings of the 24th International Conference on Conceptual Modeling (ER 2005), volume 3716 of Lecture Notes in Computer Science, pages 353-368. Springer-Verlag, Berlin, 2005.
- N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Resource Patterns. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004.
- N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In O. Pastor and J. Falcao e Cunha, editors, Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05), volume 3520 of Lecture Notes in Computer Science, pages 216-232. Springer-Verlag, Berlin, 2005.
- N. Russell, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Exception Handling Patterns in Process-Aware Information Systems. BPM Center Report BPM-06-04 , BPMcenter.org, 2006.
- N. Russell, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Workflow Exception Patterns. In E. Dubois and K. Pohl, editors, Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE 06), volume 4001 of Lecture Notes in Computer Science, pages 288-302. Springer-Verlag, Berlin, 2006.
- P. Wohed, E. Perjons, M. Dumas, and A. ter Hofstede, Pattern-Based Analysis of EAI Languages: The Case of the Business Modeling Language. in Proc. of the 5th Int. Conf. on Enterprise Information Systems (ICEIS), Angers, France, April 2003.

References IV

- P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede, Analysis of Web Services Composition Languages: The Case of BPEL4WS. in I.Y. Song, et al, eds, 22nd Int. Conf. on Conceptual Modeling (ER 2003), vol. 2813 of LNCS, pp. 200-215, Springer-Verlag, 2003.

- P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell, Pattern-based Analysis of the Control-Flow Perspective of UML Activity Diagrams. in L. Delcambre et al., eds, *Proc. of the 24th Int. Conf. on Conceptual Modeling (ER 2005),* vol. 3716 of LNCS, pp. 63-78. Springer-Verlag, Berlin, 2005.

- P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell. Pattern-based Analysis of UML Activity Diagrams. BETA Working Paper Series, WP 129, Eindhoven University of Technology, Eindhoven, 2004.

- P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell, On the Suitability of BPMN for Business Process Modelling. in Proc. of the 4th Int. Conf. on Business Process Management, Vienna, September 2006.

- P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell. Pattern-based Analysis of BPMN - An extensive evaluation of the Control-flow, the Data and the Resource Perspectives (revised version) . BPM Center Report BPM-06-17 , BPMcenter.org, 2006.

- Petia Wohed, Nick Russell, Arthur HM ter Hofstede, Birger Andersson and Wil M.P. van der Aalst. Open Source Workflow: A Viable direction for BPM? In Proceedings of CAiSE 2008, Montpellier, France, June 2008 (short paper).

- P. Wohed, B. Andersson, A.H.M. ter Hofstede, N.C. Russell, W.M.P. van der Aalst. Patterns-based Evaluation of Open Source BPM Systems: The Cases of jBPM, OpenWFE, and Enhydra Shark. BPM Center Report BPM-07-12, BPMcenter.org, 2007.

- M. Wynn, D. Edmond, W.M.P. van der Aalst, A.H.M. ter Hofstede. Achieving a General, Formal and Decidable Approach to the OR-join in Workflow using Reset nets. Proceedings of ATPN’2005. Miami, Florida, 2005.

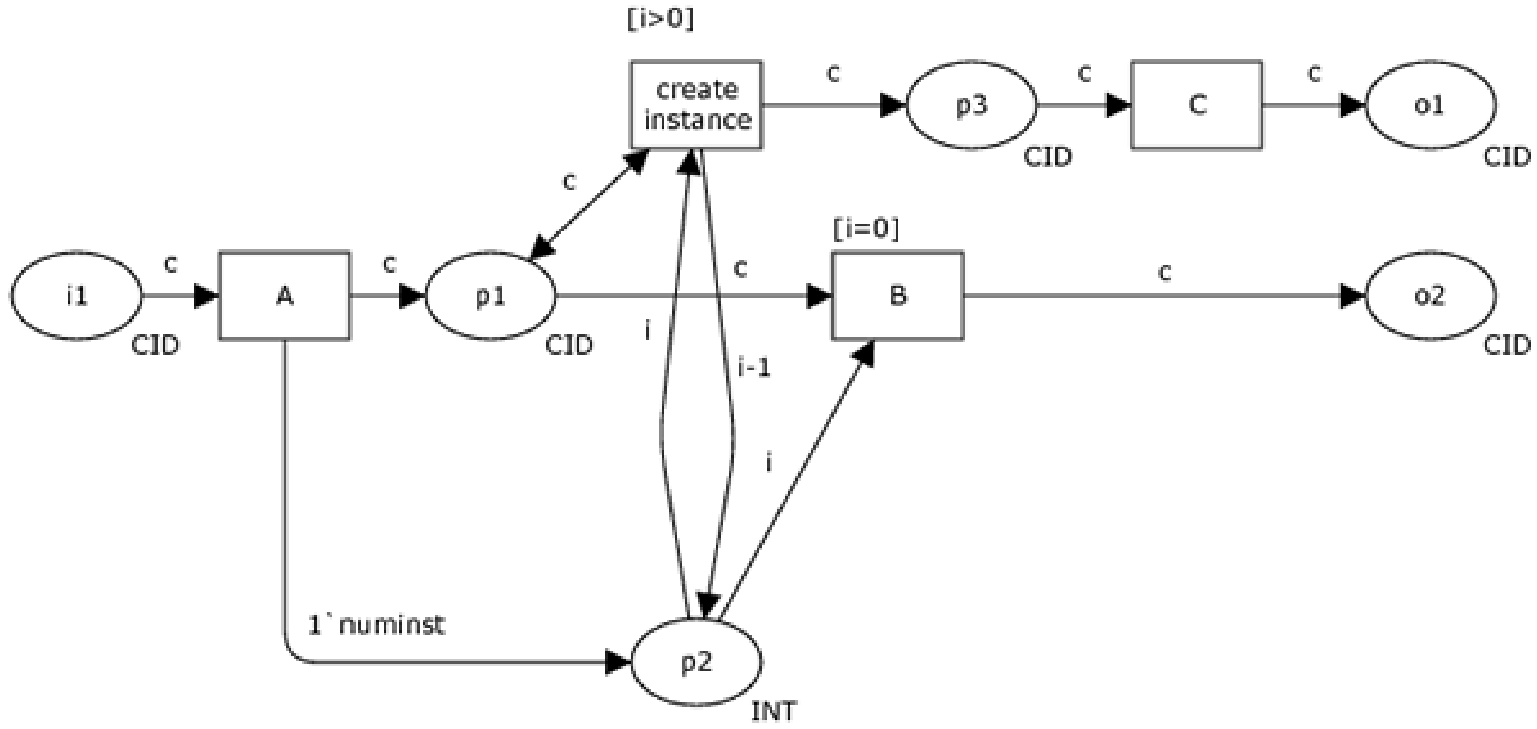
Disclaimer

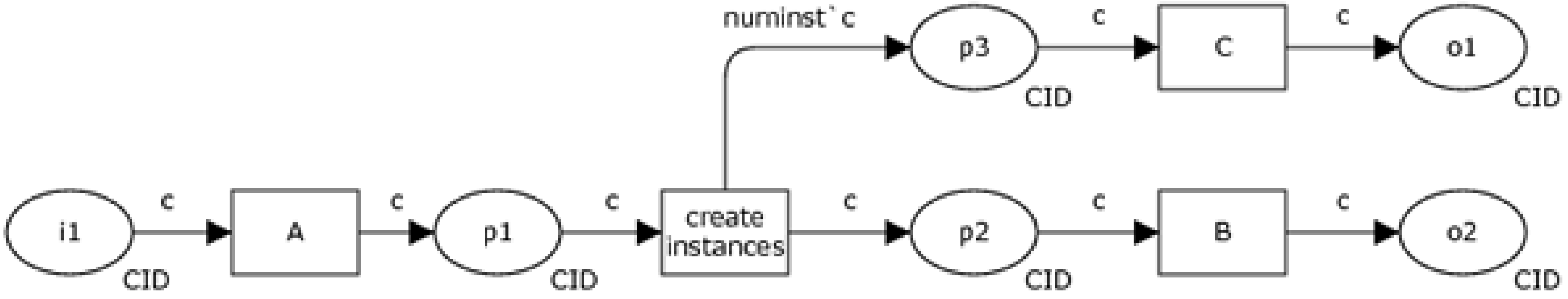
- No legal liability or responsibility is assumed for the accuracy and completeness of any product-specific information.

MI without Synchronization

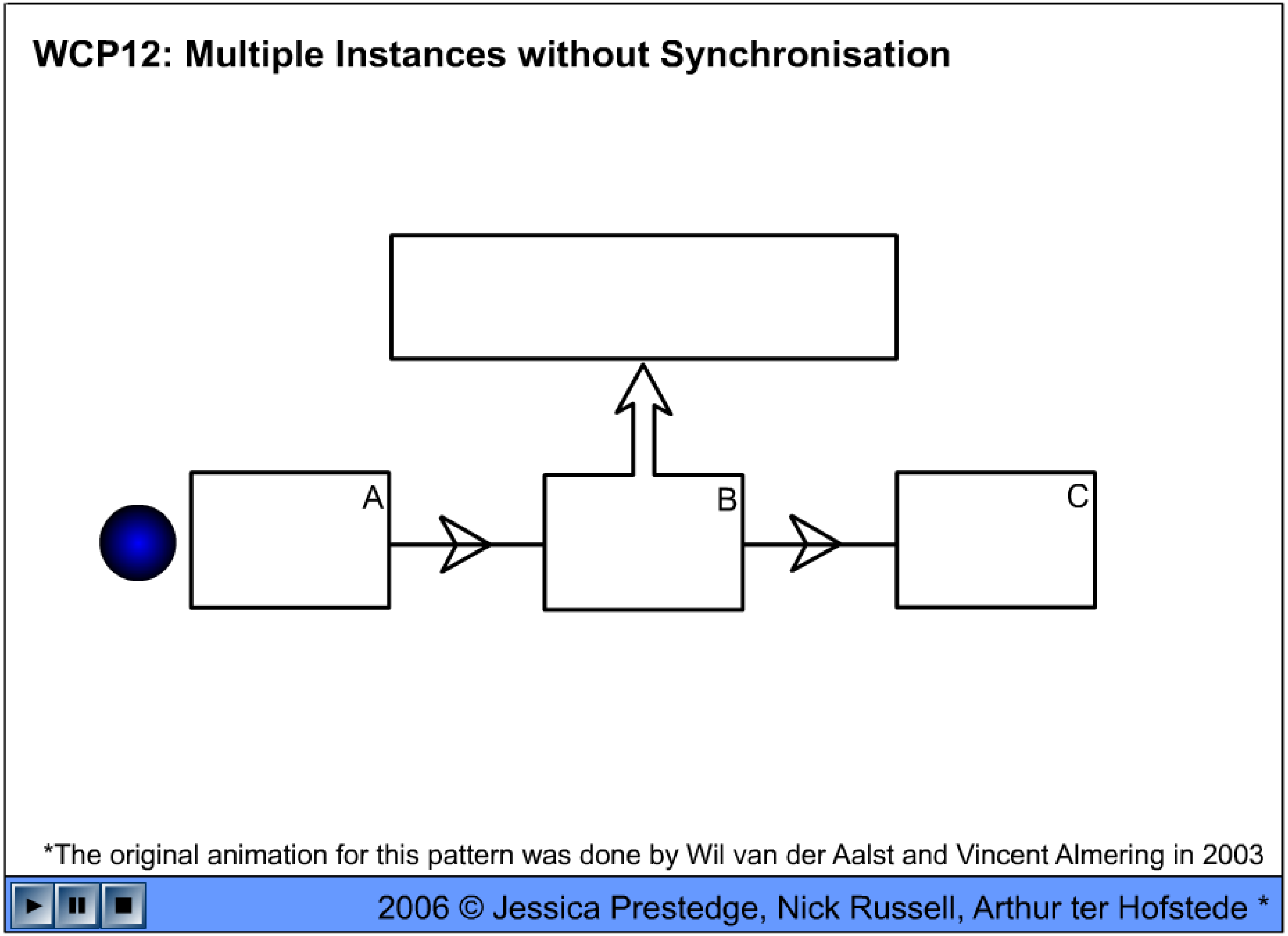
- Creates a number of concurrent instances of the same activity in a process instance, subsequent synchronisation of these instances is not required
  - Each of the instances created should run in the same context as the initiating process instance (and thus share the same case id and have access to the same data elements)
  - Instances created run independently from initiating activity

MI without Synchronization: Definition





MI without Synchronization: Animation



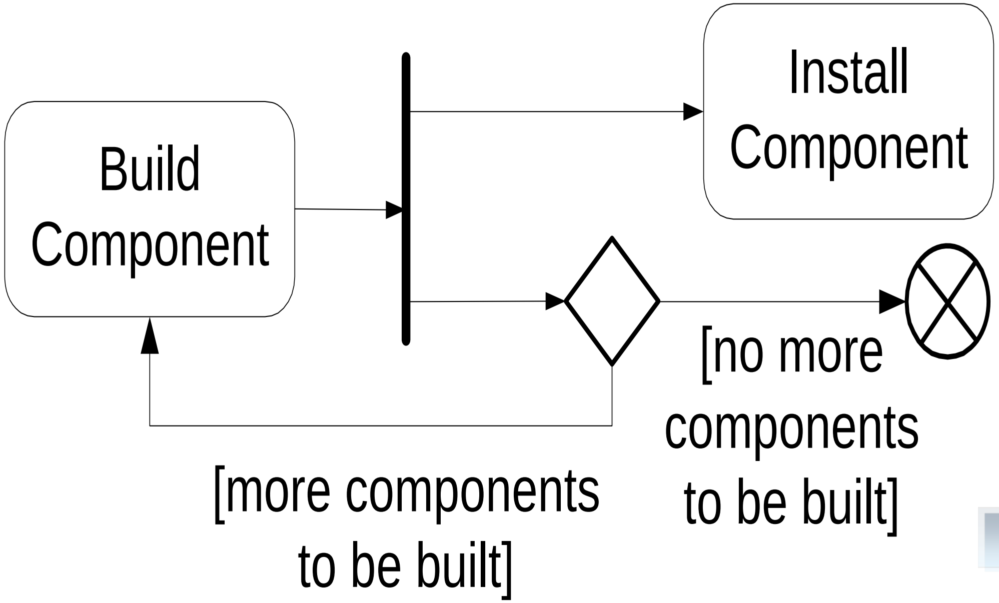
WCP12: Multiple Instances without Synchronisation

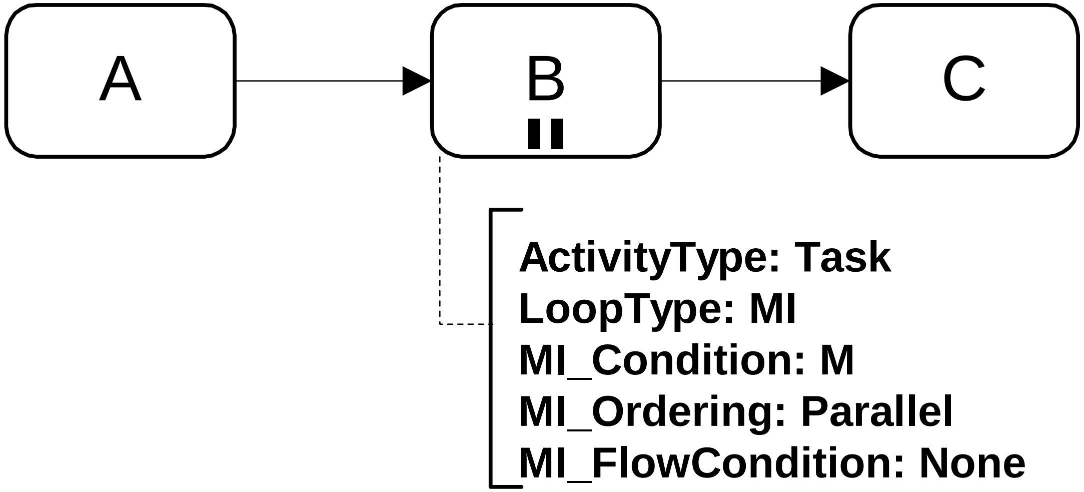
*The original animation for this pattern was done by Wil van der Aalst and Vincent Almering in 2003

2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede *

MI without Synchronization

Solution in UML 2.0 AD


Build Component
Install Component
[no more components to be built]
[more components to be built]



ActivityType: Task
LoopType: MI
MI_Condition: M
MI_Ordering: Parallel
MI_FlowCondition: None

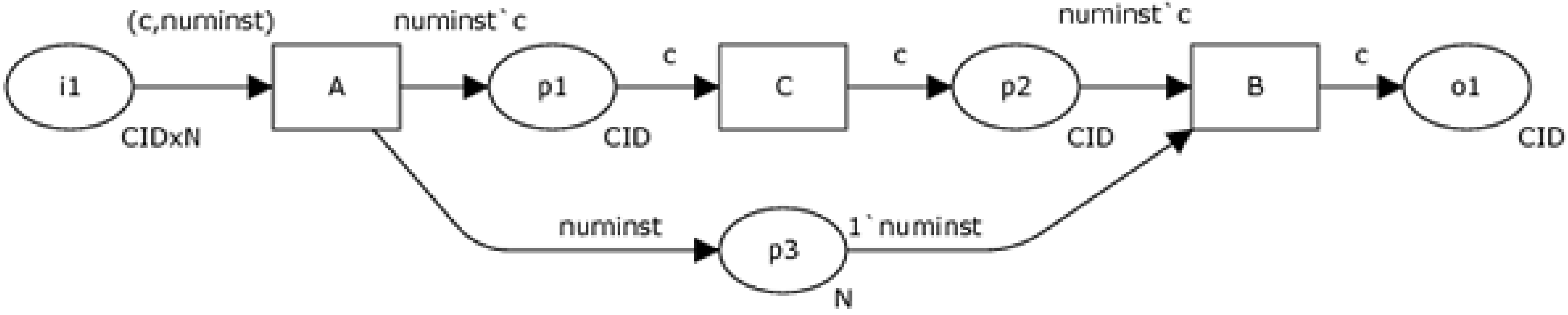
Solution in BPMN

MI with a Priori Run-Time Knowledge

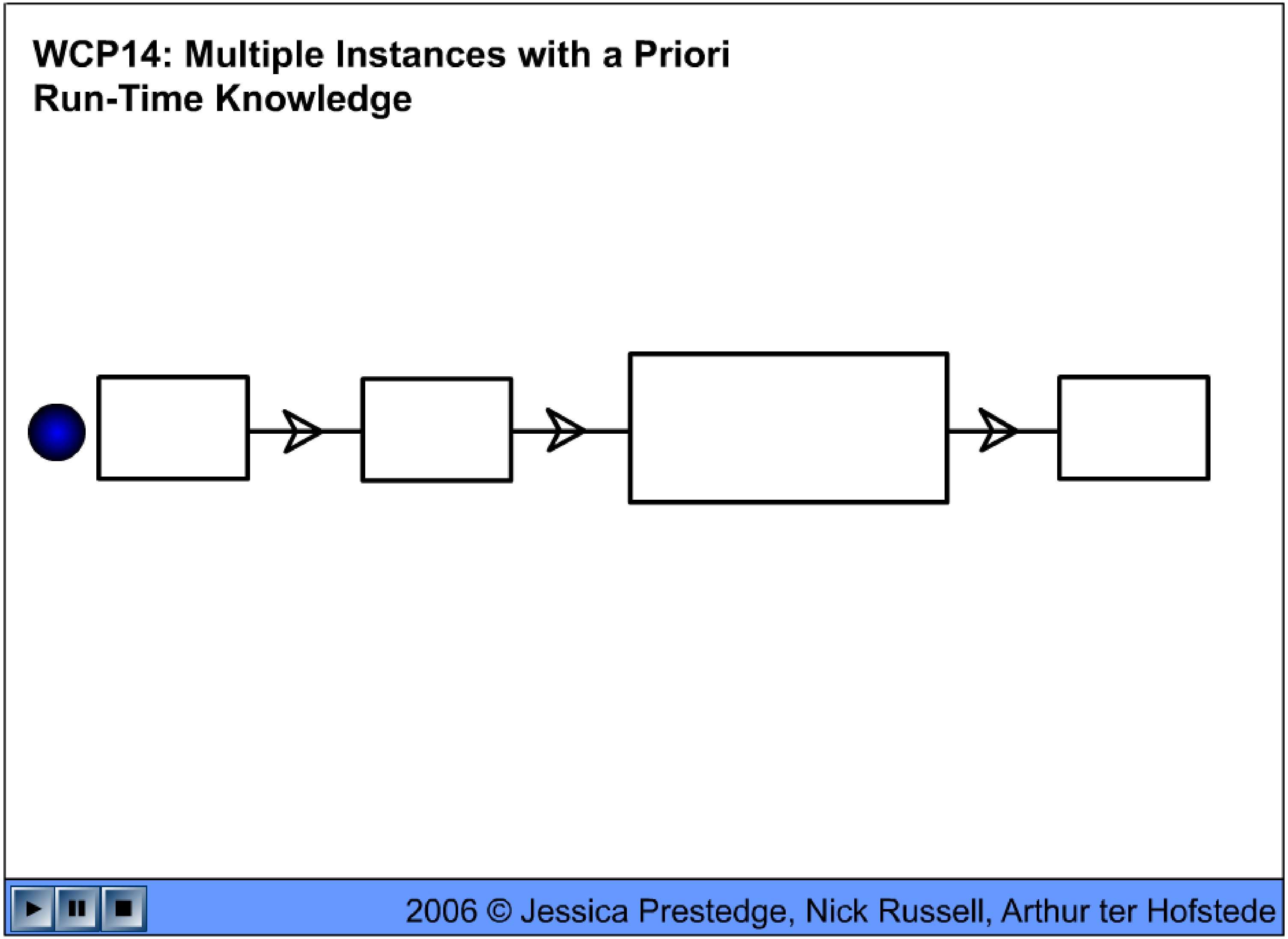
- Creation of a number of concurrently executing instances of an activity, this number is known at run-time. Synchronisation of all instances created is required. Instances created run independently from initiating activity

- Context condition:
  - The number of instances to be created must be known priori to the MI activity being invoked

MI with a Priori Run-Time Knowledge: Definition



MI with a Priori Run-Time Knowledge: Animation

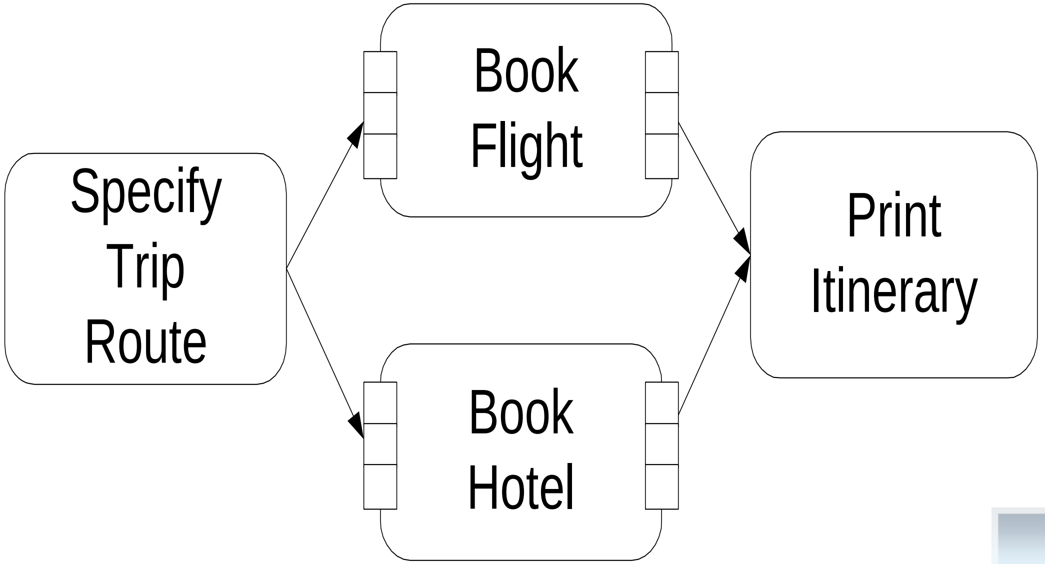


WCP14: Multiple Instances with a Priori Run-Time Knowledge

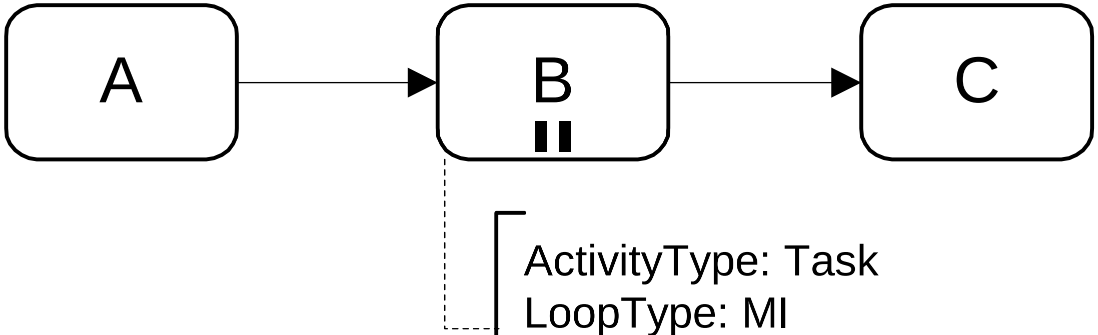
2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

MI with a Priori Run-Time Knowledge

Solution in UML 2.0 AD



Solution in BPMN



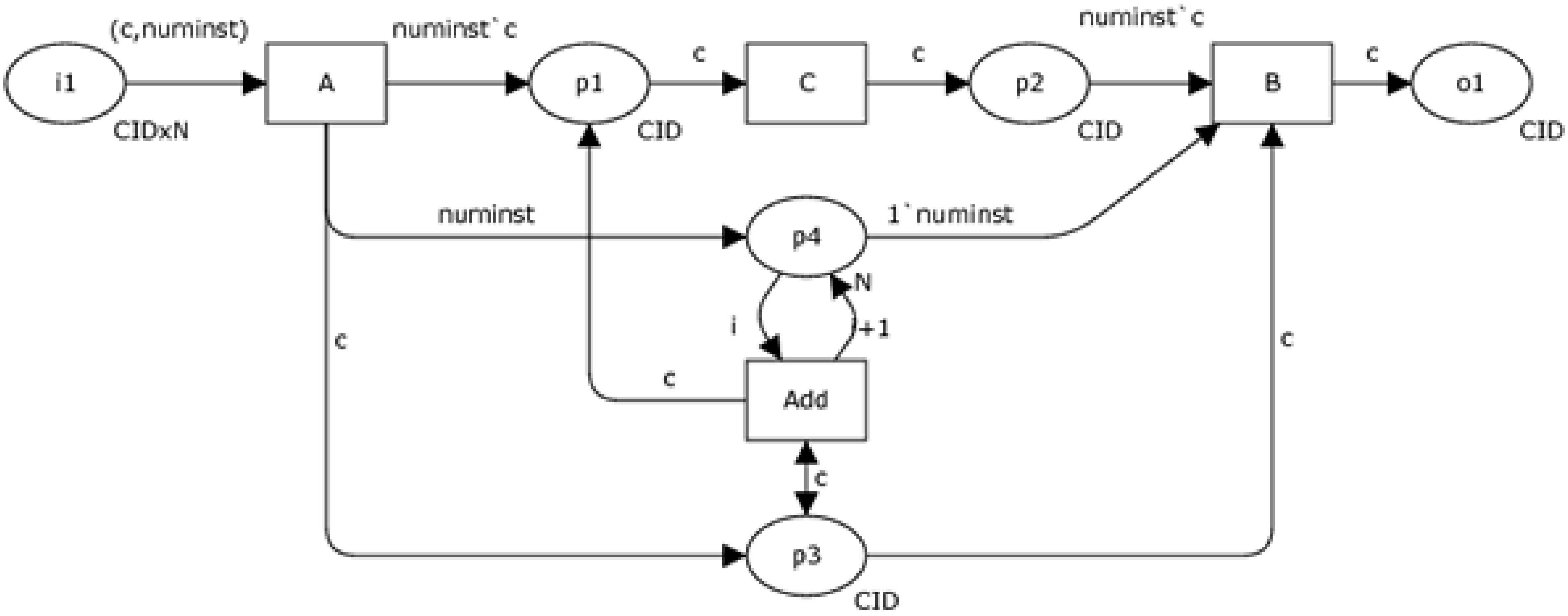
ActivityType: Task
LoopType: MI
MI_Condition: M
MI_Ordering: Parallel
MI_FlowCondition: **All**

MI without a Priori Run-Time Knowledge

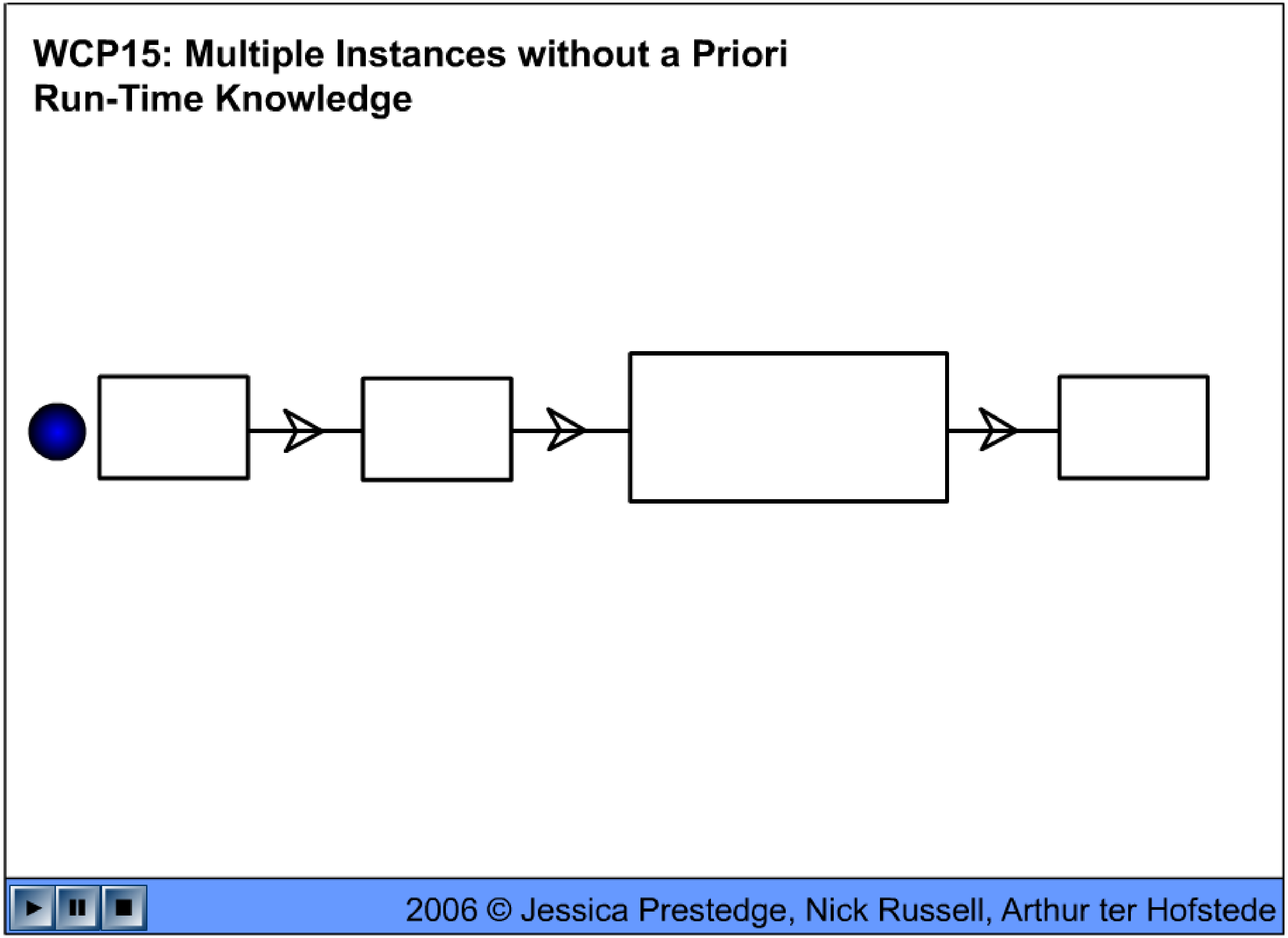
- Creation of a number of concurrently executing instances of an activity, this number is not known before invocation of the MI activity. Synchronisation of all instances created is required.

- More advanced MI patterns address:
  - Creation of new instances on-the-fly
  - Threshold for completion (partial join)
  - Cancellation of remaining threads in case of a partial join

MI without a Priori Run-Time Knowledge: Definition



MI without a Priori Run-Time Knowledge: Animation



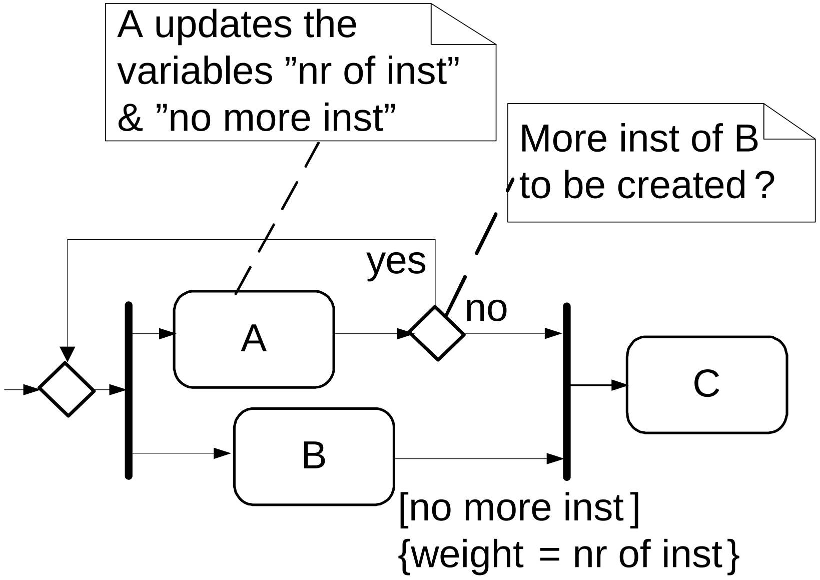
WCP15: Multiple Instances without a Priori Run-Time Knowledge

2006 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede

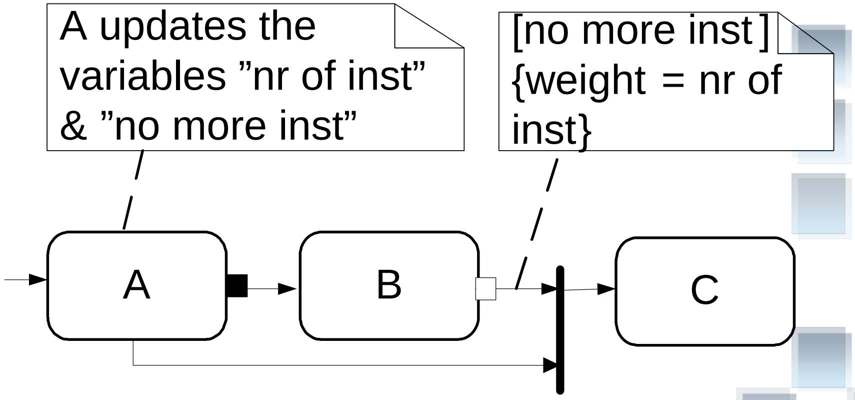
MI without a Priori Run-Time Knowledge

Workaround in UML 2.0 AD

Solution with variables


A updates the variables ”nr of inst” & ”no more inst”
More inst of B to be created ?
yes
no
A
B
C
[no more inst]
{weight = nr of inst}

Clearly, this is a work-around and requires a lot of effort on behalf of the modeller.



Solution with object streams and weights

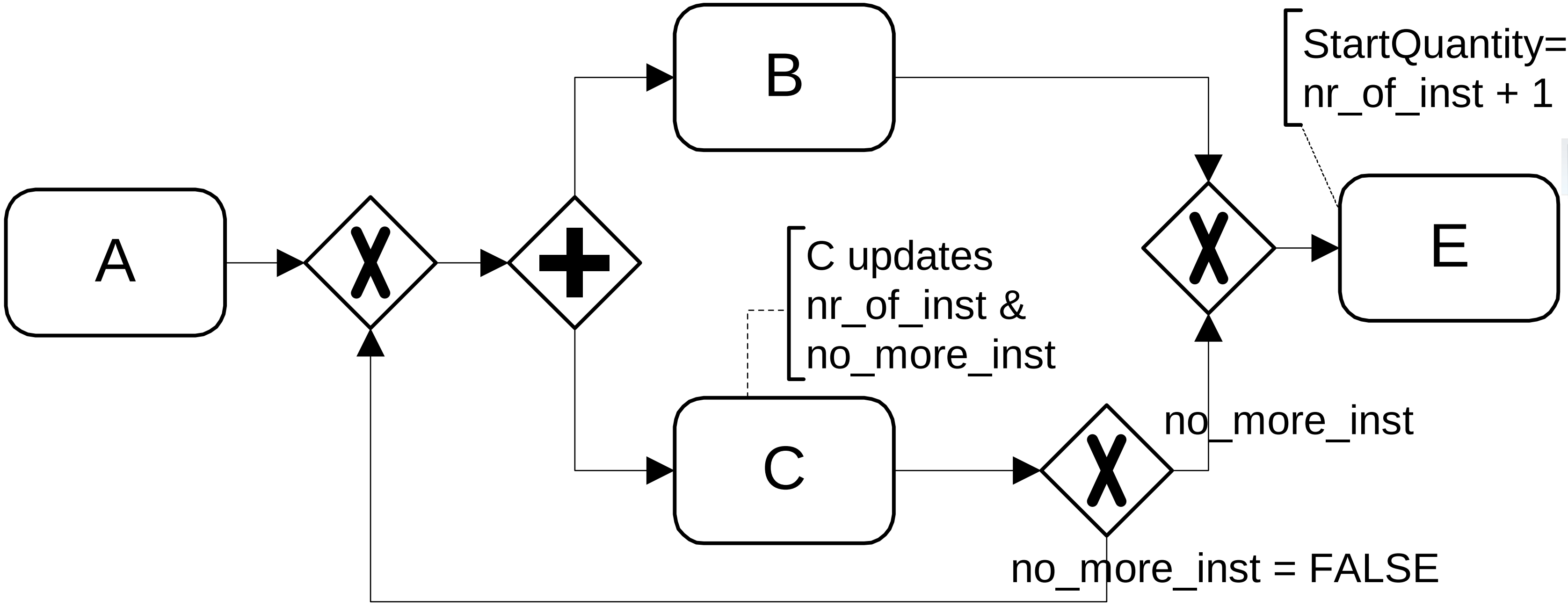
The solutions require advanced data manipulation and special attribute settings.

MI without a Priori Run-Time Knowledge

Workaround in BPMN

In practice, a modeller has to implement this pattern.

The solution requires advanced data manipulation and special attribute settings.


B
StartQuantity=
nr_of_inst + 1
A
E
C updates
nr_of_inst &
no_more_inst
C
no_more_inst
no_more_inst = FALSE