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, WFMCTC-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.”

The PAIS life cycle

- Diagnosis
- Process design
- Process enactment
- Process implementation

Project Management Tools

Business Process Modelling Tools

Workflow Management Systems

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 tasks 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 Animation

Workflow (2)

 TU/e technische universiteit eindhoven

© Wil van der Aalst, Vincent Almering and Herman Wijbenga
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:

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

<table>
<thead>
<tr>
<th>Time</th>
<th>Control-flow P:s</th>
<th>Resource P:s</th>
<th>Data P:s</th>
<th>Revised Control-flow P:s</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>20</td>
<td>43</td>
<td>40</td>
<td>43</td>
</tr>
<tr>
<td>2003</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jun 2005</td>
<td>20</td>
<td>43</td>
<td>40</td>
<td>43</td>
</tr>
<tr>
<td>Oct 2005</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sep 2006</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Control-flow P:s 20**
  - W. van der Aalst
  - A. ter Hofstede
  - B. Kiepuszewski
  - A. Barros

- **Resource P:s - 43**
  - N. Russell
  - W. van der Aalst
  - A. ter Hofstede
  - D. Edmond

- **Data P:s - 40**
  - N. Russell
  - A. ter Hofstede
  - D. Edmond
  - W. van der Aalst

- **Revised Control-flow P:s 43**
  - N. Russell
  - A. ter Hofstede
  - W. van der Aalst
  - N. Mulyar

**The ordering of activities in a process**
- CoopIS’2000
- DAPD’2003

**Resource definition & work distribution in a process**
- CAiSE’2005

**Data representation and handling in a process**
- ER’2005

- 23 new patterns
- Formalised in CPN notation

**TR**
## The Workflow Patterns Framework

### Evaluations

<table>
<thead>
<tr>
<th>Time</th>
<th>Control-flow P:s</th>
<th>Resource P:s</th>
<th>Data P:s</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>20</td>
<td>43</td>
<td>40</td>
</tr>
<tr>
<td>2003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jun 2005</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oct 2005</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


- COSA
- FLOWer
- Eastman
- Meteor
- Mobile
- I-Flow
- Staffware
- InConcert
- Domino Workflow
- Visual Workflow
- Forte Conductor
- MQSeries/Workflow
- SAR R/3 Workflow
- Verve Workflow
- Changengine

### Evaluations (Jun 2005)

- Staffware
- WebSphere MQ
- FLOWer
- COSA
- iPlanet

### Evaluations (Oct 2005)

- Staffware
- MQSeries
- FLOWer
- COSA

### Language Development

**YAWL/newYAWL**
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.

*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

WCP3: 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 *
Basic Control-flow Patterns in UML 2.0AD

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Parallel Split</th>
<th>Synchronisation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>b) Control flow in UML</td>
<td>d) UML Explicit AND-split</td>
<td>f) UML Explicit AND-join</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exclusive Choice</th>
<th>Simple Merge/Multiple Merge</th>
<th>Multiple Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4" alt="Diagram" /></td>
<td><img src="image5" alt="Diagram" /></td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
<tr>
<td>h) UML Explicit XOR-split</td>
<td>j) UML XOR-join</td>
<td>l) UML OR-split</td>
</tr>
</tbody>
</table>
## Basic Control-flow Patterns in BPMN

### Several solutions for the most basic patterns

<table>
<thead>
<tr>
<th>Parallel Split</th>
<th>a) with AND-gateway</th>
<th>b) Implicit</th>
<th>c) through sub-Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Parallel Split" /></td>
<td><img src="image" alt="a) with AND-gateway" /></td>
<td><img src="image" alt="b) Implicit" /></td>
<td><img src="image" alt="c) through sub-Activities" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synchronisation</th>
<th>d) with AND-gateway</th>
<th>e) partially through sub-Activities</th>
<th>f) in a context</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Synchronisation" /></td>
<td><img src="image" alt="d) with AND-gateway" /></td>
<td><img src="image" alt="e) partially through sub-Activities" /></td>
<td><img src="image" alt="f) in a context" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exclusive Choice</th>
<th>g) with XOR-gateway, alt 1</th>
<th>h) with XOR-gateway, alt 2</th>
<th>i) without XOR-gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Exclusive Choice" /></td>
<td><img src="image" alt="g) with XOR-gateway, alt 1" /></td>
<td><img src="image" alt="h) with XOR-gateway, alt 2" /></td>
<td><img src="image" alt="i) without XOR-gateway" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Merge</th>
<th>j) with XOR-gateway, alt 1</th>
<th>k) with XOR-gateway, alt 2</th>
<th>l) Implicit</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Merge" /></td>
<td><img src="image" alt="j) with XOR-gateway, alt 1" /></td>
<td><img src="image" alt="k) with XOR-gateway, alt 2" /></td>
<td><img src="image" alt="l) Implicit" /></td>
</tr>
</tbody>
</table>
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

WCP4: Exclusive Choice

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

2008 © Jessica Prestedge, Nick Russell, Arthur ter Hofsteede
Deferred Choice vs Exclusive Choice: Definitions

**Deferred choice**

If \( \text{cond} \) then 1’c else empty

**Exclusive choice**

If \( \text{cond} \) then 1’c else 1’c
UML 2.0 AD - Deferred Choice

A

Signal 1

B

Signal 2

C

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.
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)
WCP25: Cancel Region

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

If cond1 then 1’c else empty

If cond2 then empty else 1’c

If cond2 then 1’c else empty

[cond1 orelse cond2]
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

A → [cond1 or else cond2] → B → skip → (c, false) → (c, true) → D → (c, true) → (c, cond1) → (c, cond2) → C → (c, true) → (c, true) → (c, cond2) → (c, cond2) → (c, false) → skip → (c, false) → (c, false)
WCP37: Acyclic Synchronizing Merge

Multiple Choice

A

Sync. Merge

B

Deferred Choice

C

D

E

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.
WCP38: General Synchronizing Merge
Multiple Instance Tasks

- Sometimes multiple instances of the same task can be active within the same case

- Issues in 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
WCP15: Multiple Instances 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.

MI without a Priori Run-Time Knowledge
Control-flow perspective: Evaluation

### Basic Control-flow

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>11 Implicit Termination</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Parallel Split</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>43 Explicit Termination</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Synchronisation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>Multiple Instances</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exclusive Choice</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>12 MI without Synchronisation</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Simple Merge</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>13 MI with a priory Design Time Knlg</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

### Advanced Synchronisation

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple Choice</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>15 MI without a priory Runtime Knlg</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Str. Synchronising Merge</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
<td>27 Complete MI Activity</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Multiple Merge</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>34 Static Partial Join for MI</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Discriminator</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>35 Cancelling Partial Join for MI</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Blocking Discriminator</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>36 Dynamic Partial Join for MI</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Cancelling Discriminator</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>State-based</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structured Partial Join</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>16 Deferred Choice</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Blocking Partial Join</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>39 Critical Section</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Cancelling Partial Join</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>17 Interleaved Parallel Routing</td>
<td>+/-</td>
<td>-</td>
<td>+/-</td>
</tr>
<tr>
<td>Generalised AND-Join</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>40 Interleaved Routing</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Acyclic Synchronizing Merge</td>
<td>-</td>
<td>+/-</td>
<td>+</td>
<td>18 Milestone</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>General Synchronizing Merge</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Cancellation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thread Merge</td>
<td>+</td>
<td>+</td>
<td>+/-</td>
<td>19 Cancel Activity</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Thread Split</td>
<td>+</td>
<td>+</td>
<td>+/-</td>
<td>20 Cancel Case</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

### Iteration

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary Cycles</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>26 Cancel MI Activity</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Structured Loop</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>Trigger</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recursion</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>23 Transient Trigger</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24 Persistent Trigger</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
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
- def var X
- use(X)
- task
- B
- use(X)
- case
- C
- use(X)
- block task
- multiple instance task
- subworkflow
- X
- Y
- Z

Folder Data

Case data repository

- folder-A
  - def var X
- folder-B
  - def var Y

Case data repository

- use(folder-A)
- use(folder-A)
- use(folder-A, folder-B)

Case-1

A use(X) → B use(X) → C use(X)

Case-2

A use(X) → B use(X) → C use(X)

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

A
\text{use}(X,Y) \quad \text{pass}(X,Y) \quad \text{use}(X) \quad \text{pass}(Y) \quad \text{use}(Y)

Distinct Control and Data Channels

A
\text{use}(X,Y)

\text{pass}(X)

B
\text{use}(X)

\text{pass}(Y)

C
\text{use}(Y)
Data Interaction between Tasks

No Data Passing

Global Shared Data

- def var X
- def var Y

A
use(X,Y)

B
use(X)

C
use(Y)
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

case

A

B

def var M

C

def var N

def var S
def var T

D

E

pattern 14

pattern 15

pattern 16

pattern 17

pass(M)

request(T)

pass(T)

pass(S)

request(N)

pass(N)

external process

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

workflow repository
def var M 2.7

case repository
def var N AB02

workflow

case

A par out ret R

R > 10
M > 3.1
N <> AB01

B
C
D

Data Perspective: Evaluation

1 – BPMN    2 – UML AD    3 – BPEL

<table>
<thead>
<tr>
<th>Data Visibility</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Data Interaction (External), cont.</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Data</td>
<td>+</td>
<td>+/-</td>
<td>+/-</td>
<td>21 Env. to Case - Push-Oriented</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Block Data</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>22 Case to Env. - Pull-Oriented</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Scope Data</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>23 Workflow to Env. - Push-Oriented</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MI Data</td>
<td>+/-</td>
<td>+</td>
<td>-</td>
<td>24 Env. to Workflow - Pull-Oriented</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Case Data</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>25 Env. to Workflow - Push-Oriented</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Folder Data</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>26 Workflow to Env. - Pull-Oriented</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Workflow Data</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>Data Transfer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment Data</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>27 by Value - Incoming</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

Data Interaction (Internal)

| 9 between Tasks | + | + | +  | 28 by Value - Outgoing     | + | - | + |
| Task to Sub-workflow Decomp. | + | + | -  | 30 by Reference - Unlocked | - | - | + |
| Sub-workflow Decomp. to Task   | + | + | -  | 31 by Reference - Locked    | + | + | +/- |
| to MI Task                   | - | + | -  | 32 Data Transformation - Input | +/- | + | - |
| from MI Task                 | - | + | -  | 33 Data Transformation - Output | +/- | + | - |
| Case to Case                 | - | - | +/- | Data-based Routing           |    |    |    |

Data Interaction (External), cont.

| 15 Task to Env - Push-Oriented | + | - | +  | 34 Task Precondition Data Exist. | + | + | +/- |
| 16 Env. to Task - Pull-Oriented | + | - | +  | 35 Task Precondition Data Value | - | + | + |
| 17 Env. to Task - Push-Oriented | + | - | +/- | 36 Task Postcondition Data Exist. | + | + | - |
| 18 Task to Env - Pull-Oriented | + | - | +/- | 37 Task Postcondition Data Value | - | + | - |
| 19 Case to Env. - Push-Oriented | - | - | -  | 38 Event-based Task Trigger     | + | + | + |
| 20 Env. to Case - Pull-Oriented | - | - | -  | 39 Data-based Task Trigger      | + | - | +/- |
|                            |    |    |    | 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

- **Created**: S:create
- **Allocated to a single resource**: S:allocate, R:allocate-s
- **Offered to a single resource**: S:offer-s
- **Offered to multiple resources**: S:offer-m
- **Allocated to a single resource**: R:allocate-s
- **Started**: R:start
- **Suspended**: R:start-s, R:suspend, R:resume
- **Failed**: R:fail
- **Completed**: R:complete

- **Created**: S:create
- **Allocated to a single resource**: S:allocate, R:allocate-s
- **Offered to a single resource**: S:offer-s
- **Offered to multiple resources**: S:offer-m
- **Allocated to a single resource**: R:allocate-s
- **Started**: R:start
- **Suspended**: R:start-s, R:suspend, R:resume
- **Failed**: R:fail
- **Completed**: R:complete
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

- **created**
  - offered to a single resource
  - offered to multiple resources

- **allocated to a single resource**
  - allocated
  - S:allocate
  - R:allocate-s
  - R:allocate-m

- **started**
  - R:start
  - R:start-m
  - R:fail
  - R:resume
  - R:suspend

- **completed**
  - R:complete

- **suspended**

- **failed**

- **create**
  - S:create
  - S:offer-s
  - S:offer-m

- **allocate**
  - S:allocate
  - R:allocate-s
  - R:allocate-m
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)
WRP2: Role-Based Allocation

Task: Make Inventory List

Task: Order Supplies

Allocation: Store Clerk

Allocation: Manager

Store Clerks

SHAUN
Work List Executing
Allocated
Offered

ALICIA
Work List Executing
Allocated
Offered

Managers

TINA
Work List Executing
Allocated
Offered

GRANT
Work List Executing
Allocated
Offered

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

Task: Find Volunteer 1
Task: Train Employee

Allocation: None
Allocation: Volunteer 1

Volunteer 1 unknown

Tracey
Work List
Executing
Allocated
Offered

Allen
Work List
Executing
Allocated
Offered

Bill
Work List
Executing
Allocated
Offered

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”
WRP5: Separation of Duties

Task: Prepare Cheque
Task: Countersign Cheque

Allocation: Accountant

Accountant

Fiona
Work List
Executing
Allocated
Offered

John
Work List
Executing
Allocated
Offered

Sandy
Work List
Executing
Allocated
Offered

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

S:create

S:offer-s

S:allocate

S:offer-m

R:allocate-s

R:allocate-m

R:start

R:start-s

R:start-m

R:suspend

R:resume

R:fail

R:complete

completed

started

failed

suspended

offered to a single resource

allocated to a single resource

offered to multiple resources
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
• 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
WRP13: Distribution by Offer - Multiple Resources

Task: Gather Evidence  
Offered: Fred, Lucy

Task: Prepare Defence  
Offered: Magistrate


Distribution by Offer – Multiple Resources: Animation

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

- **Created**
  - **S:offer-s**
  - **S:allocate**
  - **S:offer-m**

- **Allocated to a Single Resource**
  - **R:allocate-s**
  - **R:start-s**

- **Allocated to Multiple Resources**
  - **R:allocate-m**
  - **R:start-m**

- **Failed**
  - **R:fail**

- **Started**
  - **R:start**
  - **R:suspend**
  - **R:resume**

- **Suspension**
  - **R:suspend**
  - **R:resume**

- **Completed**
  - **R:complete**
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

created → offered to a single resource → allocated to a single resource → started → completed

offered to multiple resources → R:delegate → suspended

S:escalate-sm → R:deallocate-sm → S:escalate-mm

R:escalate-so → S:deallocate-so → R:deallocate-ao → R:reallocation-no-state → S:escalate-ao

R:deallocate-sm → S:escalate-sa → R:deallocate-am → S:escalate-am

S:escalate-oo → R:reallocation-with-state

S:escalate-sm → R:fail → S:skip

R:escalate-aa → S:escalate-oo → R:deallocate-ao → S:escalate-oo

R:escalate-mm → S:escalate-sm → R:deallocate-sm → S:escalate-sm

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 necessary rights.
Delegation: Animation

WRP27: Delegation

Task:
Review
Audit

Allocation:
Alan

Max
Work List
Executing
Allocated
Offered

Nina
Work List
Executing
Allocated
Offered

Alan
Work List
Executing
Allocated
Offered

2005 © Jessica Prestedge, Nick Russell, Arthur ter Hofstede
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

- **created**
  - offered to a single resource
    - **S:offer-s**
    - **R:allocate-s**

- **allocated to a single resource**
  - **R:start-s**
  - **R:suspend**
  - **R:resume**
  - **R:complete**

- **suspended**

- **started**
  - **R:start**
  - **S:start-on-allocate**
  - **S:start-on-create**
  - **R:start-m**

- **failed**
  - **R:fail**
  - **S:piled-execution**
  - **S:chained-execution**

- **completed**

- **offered to multiple resources**
  - **S:offer-m**
  - **R:allocate-m**

- **started**
  - **S:allocate**
  - **S:create**
  - **S:offer**

- **auto-start Patterns**

Auto-start Patterns

- Commence on Creation
- Commence on Allocation
- Piled execution
- Chained Execution
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Creation Patterns</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>Pull Patterns (cont.)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Direct Allocation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>24 System-Determ. Work Queue Cont.</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2 Role-Based Allocation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>25 Resource-Determ. Work Queue Cont.</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>3 Deferred Allocation</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>26 Selection Autonomy</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>4 Authorization</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td><strong>Detour Patterns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 Separation of Duties</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>27 Delegation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>6 Case Handling</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>28 Escalation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>7 Retain Familiar</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>29 Delegation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>8 Capacity-based Allocation</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>30 Stateful Reallocation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>9 History-based Allocation</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
<td>31 Stateless Reallocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10 Organizational Allocation</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
<td>32 Suspension/Resumption</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>11 Automatic Execution</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>33 Skip</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Push Patterns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 Distribution by Offer-Single Resource</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>34 Redo</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13 Distribution by Offer-Multiple Resources</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td><strong>Auto-start Patterns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 Distribution by Allocation-Single Resource</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>35 Pre-do</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>15 Random Allocation</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
<td>36 Commencement on Creation</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>16 Round Robin Allocation</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
<td>37 Commencement on Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>17 Shortest Queue</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
<td>38 Piled Execution</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>18 Early Distribution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td><strong>Visibility Patterns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19 Distribution on Enablement</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>39 Chained Execution</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>20 Lata Distribution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>40 Config. Unallocated WI Visibility</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Pull Patterns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21 Resource-Init. Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>41 Config. Allocated WI Visibility</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22 Resource-Init. Exec. - Allocated WI</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>42 Simultaneous Execution</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>23 Resource-Init. Exec. - Offered WI</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>43 Additional Resources</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>
• **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.
• Patterns range from simple to (very) complex
• Patterns typically observed
• Comprehensive support lacking
• Problems so complex that informal approaches fall short
References I

- www.workflowpatterns.com


References II


References III


- 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


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
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

Solution in BPMN

ActivityType: Task
LoopType: MI
MI_Condition: M
MI_Ordering: Parallel
MI_FlowCondition: None
• 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
WCP14: Multiple Instances with a Priori Run-Time Knowledge
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
WCP15: Multiple Instances 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

A

no

[no more inst]

{weight = nr of inst}

B

C

Solution with object streams and weights

A updates the variables "nr of inst" & "no more inst"

[no more inst]

{weight = nr of inst}

A -> B -> C

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

The solutions require advanced data manipulation and special attribute settings.
Workaround in BPMN

In practice, a modeller has to implement this pattern. The solution requires advanced data manipulation and special attribute settings.

MI without a Priori Run-Time Knowledge